Vrije Universiteit Brussel

FACULTEIT WETENSCHAPPEN EN BIO-INGENIEURSWETENSCHAPPEN
Departement Computerwetenschappen
Web & Information Systems Engineering

The Next Generation of Input Devices: SpeeG version 2

Proefschrift ingediend met het oog op het behalen van de titel Master in de Ingenieurswetenschappen: Computerwetenschappen, door

Sven De Kock

Promotor: Prof. Dr. Beat Signer
Begeleider: Lode Hoste

Augustus 2012

Vrije Universiteit Brussel

FACULTY OF SCIENCE AND BIO-ENGINEERING SCIENCES
Department of Computer Science
Web & Information Systems Engineering

The Next Generation of Input Devices: SpeeG version 2

Dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science in Applied Sciences and Engineering: Computer Science, by

Sven De Kock

Promotor: Prof. Dr. Beat Signer
Advisor: Lode Hoste

August 2012

Samenvatting

Het invoeren van tekst op personal computers wordt al geruime tijd gedomineerd door toetsenborden omwille van hun efficiëntie. Een toetsenbord is, door de fysische beperkingen, echter niet praktisch voor mobiele en embedded systemen. Een gelijkaardig probleem bestaat ook voor de tekstinvoer op televisiesystemen. Een toetsenbord van volledige grootte past niet in de woonkamer en geeft een rommelige indruk. Om het even welk extra apparaat kan de woonkamer rommelig maken.

Spraak is een ideale kandidaat voor het invoeren van tekst. In een conversatie spreken mensen gemiddeld aan een snelheid van 196 woorden per minuut (WPM). De gemiddelde typsnelheid op een toetsenbord is 38 WPM. Spraak is ook een intuïtief middel omdat we het elke dag gebruiken om mee te communiceren. Spraakherkenning is niet perfect en de snelheid waarmee we tekst kunnen invoeren is bijgevolg sterk afhankelijk van hoe efficiënt we de fouten in de spraakherkenningresultaten kunnen verbeteren. Fouten kunnen zich echter opnieuw voordoen als spraak wordt gebruikt om deze fouten te verbeteren. Daarom moet er een modaliteit toegevoegd worden om de spraakherkenningfouten te verbeteren.

In dit werk stellen we SpeeG v2 voor, een tekstinvoersysteem dat gebruik maakt van spraakherkenning. Om de spraakherkenningfouten te verbeteren maken we gebruik van gebaren die opgenomen worden door een Microsoft Kinect sensor. Deze sensor zorgt voor een extra modaliteit zonder de omgeving van de gebruiker aan te passen. We introduceren vier prototypes voor tekstinvoer die spraakherkenning en gebaren combineren. Elk prototype verschilt in de interactiemethode. Dit werd gedaan om de bruikbaarheid van elk prototype te testen. De vier prototypes noemen Scroller, Scroller Auto, Typewriter and Typewriter Drag.

We hebben de prototypes geëvalueerd in een kwantitatieve en kwalitatieve studie. De beste gemiddelde prestatie kwam van het Typewriter prototype met een gemiddelde snelheid van 21,04 WPM. De Typewriter Drag had een gemiddelde snelheid van 15,31 WPM. Daarna volgden de Scroller (13,59 WPM) en Scroller Auto (9,69 WPM) prototypes. De gemiddelde word error

rate (WER) voor correctie was 20,29%. Na de correctie was de WER 0% voor alle prototypes. Onze kwantitatieve studie toonde verhoogde prestaties en een lagere WER in vergelijking met SpeeG v1. De kwalitatieve studie toonde aan dat de deelnemers een voorkeur hadden voor het Typewriter en Typewriter Drag prototype omdat ze sneller aan voelden.

Abstract

Text input for personal computer has been dominated by keyboards because of their efficiency. However for mobile or embedded systems a full sized keyboard is not practical because of physical limitations. A similar problem exists for text entry on television systems. A full sized keyboard would clutter the living room. In fact any extra device can clutter the living room.

Speech is a prime candidate for text input. In a conversation people speak at a rate of 196 words per minute (WPM). The average typing speed on a keyboard is only 38 WPM. It is also a intuitive device because we use it every day in communicating with each other. However speech recognition is not perfect and thus the speed at which text is entered greatly relies on how efficient speech recognition errors are corrected. Errors can re-occur if speech is used to correct these errors. Therefore an additional modality should be used to correct speech recognition errors.

In this dissertation we introduce SpeeG v2, a text entry system that uses speech recognition. To correct speech recognition errors we use gestures recorded from a Microsoft Kinect sensor. This device provides a non-intrusive additional modality for error correction. We introduce four text entry prototypes that combine speech recognition and gestures. Each prototype differs in interaction method from another. This was done to investigate the usability of each interaction method. The four prototypes are named Scroller, Scroller Auto, Typewriter and Typewriter Drag.

We evaluated the prototypes in a quantitative and qualitative study. The best mean performance was recorded for the Typewriter prototype with a mean text entry speed of 21,04 WPM. The Typewriter Drag had a mean entry speed of 15,31 WPM. After that came the Scroller (13,59 WPM) and Scroller Auto (9,69 WPM) was the slowest to enter text with. The mean word error rate (WER) before correction was 20,29%. After correction all participants had a WER of 0%. Our quantitative study showed increased performance and a lower WER after correction over SpeeG v1. The qualitative study showed participants prefered using the Typewriter and Typewriter Drag prototypes because they were faster.

Acknowledgments

In this short text I would like to thank all people that helped make this dissertation possible. First and foremost I would like to thank my promoter Prof. Dr. Beat Signer for promoting this dissertation as well as the opportunity to work on it.

I would like to thank my advisor Lode Hoste because he spent many many hours helping me on my work for this dissertation.

I would also like to thank all people who participated in the user study included in this work which are in alphabetic order: Bruno, Eline, Kevin, Lode, Max, Niels, Simon and Thierry.

My friends and family also deserve a thank you for encouraging me throughout my studies.

A final thanks to you for reading this dissertation.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Research context

The field of text entry systems has been dominated by keyboards. This is because text entry systems involve two significant tradeoffs: between potential efficiency and training time, and between device size and character-set size [Ward et al., 2000]. A QWERTY keyboard is undoubtedly the most popular text entry system. It has been for a long time. One of the reasons for its success is dated back to when typewriters were used. QWERTY typewriters were used all over the world. It being a widely used standard for typewrites has caused people to become familiar with it. Wide spread familiarity cause training time to go down and efficiency to go up. Making the QWERTY keyboard a very efficient text entry system.

For some systems a keyboard is not a practical solution for text entry. A television is one of those. People do not like yet another controller lying around in the living room. Most solutions for this scenario rework the keyboard to a smaller size or rearrange the keys. However these devices are not as efficient as a full sized keyboard or require training to become efficient.

Speech is the most natural way of communicating with each other. In a conversation people speak at a rate of 196 words per minute [Yuan et al., 2006]. The average typing speed on a keyboard is only 38 words per minute [Ostrach, 1997]. Therefore, speech looks to be a prime candidate for text entry.

It is also a intuitive device because we use it every day in communication.

1.2 Problem statement

While speaking is about five times faster than typing, it is also a much more complex medium. The domain of speech recognition studies this medium. A voice is like a fingerprint, every person has a unique vocal anatomy. This is one of many difficulties speech recognition faces. Speech recognition is not perfect because of the complexity of speaker variability, environmental variability and context variability. For instance, a speech recognizer is more likely to have a better result at slower speaking rates. A speed of 100 words per minute is generally more likely to be correctly processed by a speech recognizer instead of a conversation rate of 196 words per minute. This is about the talking speed for a person who is giving a presentation. At this speed sounds are articulated better, thus increasing the clarity of the signal. However speech recognition should be able to handle higher and lower speaking rates, different accents, articulations or intonations.

Since a speech recognizer does not always give the correct result it is subject to correction. The speed at which text is corrected is thus very important since it determines the entry speed. Most interfaces that use speech recognition work in two steps. In a first step the user decides when speech is recorded and decides when it is stopped by pressing one or more buttons. Then the audio is processed and the user gets little to no feedback. The second step is to correct the errors made by the speech recognizer. In this step speech recognition is disabled and another modality is used. Other modalities are used because using speech again can cause the same error to re-appear (cascading errors). **We believe combining the two steps into one leads to a more intuitive and natural notion of speaking and communication.** However there is a lack of interfaces that try to do this in current work. This leads to the problem statement of this dissertation.

Problem statement:
How can we combine speech recognition and error correction continuously, meanwhile minimizing errors and maximizing efficiency?

1.3 Research goals

In SpeeG version 1 (SpeeG v1) speech recognition suffered from poor accuracy and speed. Speech recognition accuracy depends on a lot of factors.

These factors should be investigated so that speech recognition accuracy and speed is improved in SpeeG version 2 (SpeeG v2).

Even with improved accuracy a speech recognizer is not perfect and errors can occur. The performance of a text entry system that uses speech recognition thus depends on how fast errors are corrected. In SpeeG v2 we research how to efficiently correct errors.

SpeeG v2 should be usable by as many people as possible. Most people want to enter text quickly and do not like a training phase. Thus SpeeG v2 should support speech recognition with a general profile to enter text without a learning phase. However a limitation of speech recognition is that it depends greatly on the acoustic and language models used. Acoustic models are trained on samples from a specific corpus and these samples are spoken by native speakers. Because of this speech recognition accuracy is better for native speakers. For non-native speakers who get terrible speech recognition accuracy the only option to improve it is to train the acoustic model. SpeeG v2 should support training the speech recognizer for improved accuracy if it is required.

1.4 Characteristics of SpeeG v2

Text entry is used in many scenarios. **The scenario we focus on is text entry on a television system.** In this scenario we focus on several aspects of text entry, like visualization, cascading errors and physical stain. These aspects characterize SpeeG v2 and are the basis for creating the SpeeG v2 text entry prototypes. The SpeeG v2 characteristics are described below.

Continuous nature of speech
Speech is a continuous phenomenon. Most text entry systems that utilize speech recognition do it in a two step process. First speech recognition is activated and a user starts speaking. When the user is done speaking the speech recognition is disabled. Then the second step is started, correcting the results from the speech recognition process. We propose to keep the *continuous nature of speech* and thus keep speech recognition active while correcting errors.

Considering non-native users
Speech recognition accuracy depends greatly on the used models and the user. There are two acoustic models for the English language: US and UK. Each of them is made from spoken samples from native speakers. However

a non-native user should be able to use the speech recognizer and get good results. If speech recognition results are greatly impacted by a user he should be able to train the speech recognizer. *We propose the use of profiles so that both a general profile can be used if accuracy is good and a trained profile can be used when speech recognition accuracy is terrible.*

Sentence-level recognition
In the evaluation of SpeeG v1 users spoke one word at a time. The option was present to speak an entire sentence, however it remained largely unused. We argue that it is unnatural to speak only one word at a time because people communicate in sentences. *We propose sentence-level speech recognition and force users to speak entire sentences.*

Avoid cascading errors
A side effect of using speech recognition is that errors frequently occur. A major challenge is to avoid errors reoccurring [Vertanen and Kristensson, 2009]. For instance if speech is corrected with speech, the exact same errors could be produced by the speech recognizer. This has a negative impact on efficiency. *We propose to avoid cascading errors by using a different modality to correct the speech recognition hypothesis.*

Visualize speech
When using speech recognition it is important to always inform the user what is hypothesized. Preferably feedback should be given from the first moment a user starts speaking to indicate speech is being processed. *We propose to always visualize speech from the first sound until a final hypothesis is produced.*

Imprecise input
Speech is a fast medium and thus a prime candidate for text input. However the speech recognition process is not perfect and another modality should be used to avoid cascading errors. *We propose to use a always-on camera to monitor the activity of the user. Gestures are recorded and used to correct speech recognition errors.* However gestures are not as precise as other input devices (like a mouse or stylus) and this has to be taken into account when designing a user interface.

Minimize physical strain
Gestures can cause users to experience physical strain. This was observed in SpeeG v1. *We propose to minimize physical strain because it negatively impacts usability.*

1.5 Overview of this work

First we introduce the basic concepts of speech recognition in Chapter 2. We also explain why speech recognition is difficult and not perfect. Then we discuss related work in Chapter 3. After this we propose four prototypes and discuss their features in Chapter 4. We evaluate the performance and usability of these prototypes and compare them to related work in Chapter 5. Finally we end this dissertation with a conclusion and propose future work.

2

Speech recognition

Everywhere around the world spoken language is used as primary form of communication [Gaikwad et al., 2010]. Whether it is a radio announcement or a conversation in the street, speech is a very important part of our lives. It has been the most used form of communicating for centuries and there are absolutely no indications that this will change. Therefore, speech is also the most natural form of communication.

Besides speech being the primary communication medium, it is also a very efficient medium. The average speaking rate in a English conversation is 196 words per minute (WPM from now on) [Yuan et al., 2006]. In conversations the speaking rate can vary from 111 up to to 291 WPM [Yuan et al., 2006]. If we compare this rate with the speed for the most popular input device, a keyboard, then speech is much faster. Some people are very good at typing and in a professional environment it can be very efficient. However, in a study on 4000 persons the average typing speed was 38 WPM [Ostrach, 1997]. In another study the average was 33 WPM (for transcription) [Karat et al., 1999].

According to the numbers above speech could be considered a prime candidate for input. The main challenge is that the high numbers above are based on speech in conversations. When humans communicate with machines this is not as efficient. The main reason for this is that language and speech are difficult concepts to master for machines, because of many elements. We will elaborate on these difficulties and how a machine is able to process speech

in this chapter.

2.1 Sound and Speech

Because of the importance of speech in our lives, a lot of research has gone into understanding how the human vocal anatomy works. Since the beginning of computers, people have been working on integrating language and speech into systems [Huang et al., 2001]. Although computers have evolved a lot over the years current systems still lack the ability to speak, listen and understand what is being said. The reason for this is that language and speech are complicated phenomenas. The following book [Huang et al., 2001] was used to investigate these phenomenas.

Human speech generates a number of waves of pressure that are contractions of air molecules. They exit the body through the mouth and nostrils of the speaker. In most of the world's languages, phonemes can be split up into two classes, consonants and vowels. The concept of a phoneme is explained later in this chapter. Consonants are articulated with a complete or partial closure of the vocal tract, while vowels are articulated with an open vocal tract. It is very important to know that there are many factors that determine the sound that a human produces. Besides the lungs and vocal cords for the production of air and sound, articulation is largely determined by the mouth, tongue, teeth and lips. For instance the lips can be rounded or spread to affect vowel quality and closed for consonants like "p", "b" or "m".

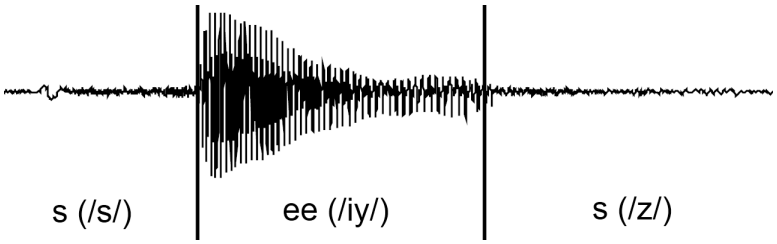


Figure 2.1: A sound wave for the word "sees"

Looking at the sound waves that a human produces, we can distinct voiced and voiceless sounds. The voiced sounds have a roughly regular pattern and voiceless sounds don't. Vowels are voiced sounds and are easier to distinct from one another. Because of the lack of pattern in voiceless sounds, it is harder to recognize the difference between them. In figure 2.1 the difference between voiced and voiceless sounds is shown in the word "sees".

Phonemes are accepted to be the smallest unit in speech. They are combined to form syllables and words to give meaning to the sound being produced. For instance the word head(/hh eh d/) can be formed from the phonemes /hh/, /eh/ and /d/. A syllable is thought to be between phones and word level. Phonemes are sometimes referred to as phones. There is no specific reason for this and both are accepted by the research community. For the sake of consistency the term phoneme will be used in this work.

In general a syllable is centered around a vowel. A word like "verbal" has two syllables (ver-bal). Syllables have structure and sounds within the syllable influence one another. The sounds that two people produce are however very different from each other given they are speaking the same sentence. It is apparent that both sounds can be understood by other humans, but machines have a harder time coping with the lack of similarity. What a machine needs to do is analyze the sounds that are produced from the speaking and try to match the sound to phonemes, syllables and words.

2.2 Statistical foundations of speech recognition

Because of the complexity of speech a speech recognition systems is also complex. The best approach is based on statistical modelization of the speech signal [Chou and Juang, 2002]. The techniques that are used in speech recognition are described below.

Analyzing speech signals deals with uncertainty. For instance if two persons speak the same sentence. The sound that they produce will be different from each other. This is because each person has a different vocal anatomy and this phenomenon is compared to a fingerprint [Huang et al., 2001]. If this was not the case and everyone created the same sound for a specific sentence, then speech recognition would be relatively easy. However everyone has a different vocal anatomy and to add to the complexity it is possible to speak a specific sentence in a slower or faster rate, with possibly different intonations. So one person is able to speak one sentence in a wide range of different ways. Yet a machine should always detect these variations as the same sentence. This is why analyzing speech deals with a large amount of uncertainty. The use of probability theory and statistics provides a mathematical model for handling this uncertainty.

2.2.1 Probability theory

Probability theory provides insight into the likelihood of the occurrence of a specific event. It can provide a degree of confidence for the outcome of a event. In probability theory S, sample space, is the term that refers to the collection of possible outcomes. An event, A, is a subset of this collection. The probability that event A will occur in this sample space is described by $P(A)$. This is computed by dividing the total number of occurances, N_S, by the number of occurances that have the outcome event A, N_A. The formula,

$$P(A) = \frac{N_A}{N_S}[Huanget\ al.,\ 2001] \tag{2.1}$$

can have a value between zero and one,

$$\forall A : 0 \leq P(A) \leq 1[Huanget\ al.,\ 2001] \tag{2.2}$$

The lower bound is zero. This means that event A will never occur and that the sample space S is empty. The other bound, one, is achieved if A always occurs. In this case $A = S$. This can be concluded because of the property that when

$$\bigcup_{i=1}^{n} A_i = S[Huanget\ al.,\ 2001] \tag{2.3}$$

, then

$$P(A_1 \cup A_2 \cup ... \cup A_n) = \sum_{i=1}^{n} P(A_i) = 1[Huanget\ al.,\ 2001] \tag{2.4}$$

is always true. So if $P(A_i) = 1$ then the only event in S is A_i and thus, $A_i = S$. This is the very basics of probability theory. Another important notation is $P(AB)$, which describes the joint probability of event A and B occurring. The calculation of $P(AB)$ follows from 2.1 and is:

$$P(AB) = \frac{N_{AB}}{N_S}[Huanget\ al.,\ 2001] \tag{2.5}$$

So now we can compute the probability that event A and event B occur at the same time, $P(AB)$. But another very important probability is the conditional probability, $P(A|B)$. This is the probability that event A will occur after event B has occurred. This probability can be computed by using the previous probabilities,

$$P(A|B) = \frac{P(AB)}{P(B)}[Huanget\ al.,\ 2001] \tag{2.6}$$

2.2.2 Bayes' theorem

With all what we defined we can come to the Bayes' theorem. This theorem is the basis of pattern recognition and is thus very important in speech recognition. Until now we were able to compute the conditional probability with the definition of the joint probability. To explain this theorem I introduce a partition $A_1, A_2, ..., A_n$ of S, where S has n events and B is any event in S. Because of the definition of a partition and any A_nB being disjoint from the others we can say that

$$P(B) = \sum_{k=1}^{n} P(A_kB) = \sum_{k=1}^{n} P(A_k)P(B|A_k)[Huanget\ al.,\ 2001] \qquad (2.7)$$

is true and come to the Bayes' theorem:

$$P(A_i|B) = \frac{P(A_iB)}{P(B)} = \frac{P(A_iB)P(A_i)}{\sum_{k=1}^{n} P(B|A_k)P(A_k)}[Huanget\ al.,\ 2001] \qquad (2.8)$$

And thus th Bayes' theorem helps us in computing conditional probabilities.

2.2.3 The law of large numbers

The law of large numbers is important in probability theory. This theorem states that if a you perform an experiment a large number of times then the average result that you get out of these experiments should be close to the expected value of the experiment. In speech recognition this is useful if you sample a large amount of data from a person or a number of persons that speak the same sentence. What you get out of this is that the average value that is spoken for this sentence is the best one to work with. However in speech recognition we have some added problems. For instance native speakers will speak a given word in one particular manner, where non-native speakers will not.[Huang et al., 2001] If we then sample Native and non-native speaker, the average value will not be close to the native or the non-native average and we have less recognition because both can be recognized less precise. In this case samples should be divided into subgroups for better recognition.

The law is also used for predicting next words. Usually a speech recognizer is trained with a large amount of sentences from a reliable source, for instance from the Wall Street Journal. From these sentences a dictionary is formed that includes all words from the source sentences. But a more important piece of information is the probabilities that are computed from the source. The probability that a word will occur in a sentence can be computed. But

also what position it can occur and what words it will likely follow or be followed by. This information can be very valuable in a speech recognition engine provided the speaker speaks sentences resembling sentences from the source. In theory it is good to increase the sample space because you will get a more accurate probability from it. For speech recognition however you should be careful when increasing the sample space.

2.2.4 Statistical inference

Statistical inference is the technique of drawing conclusions from data. In speech recognition this is important because if you want to match sound to a word, you want to know how well that word is matched and how well it matches with other words. To be able to draw conclusions from data significance testing is used. The significance level, or p-value is the value that you get from your tests. Popular tests are: Z-test, x^2 test, matched-pairs test, sign test and magnitude-difference test. The explanation of these tests is out of the scope of this thesis. More information can be found in [Huang et al., 2001].

2.2.5 Pattern recognition

People are able to understand what other people say. Because of some function in the brain this is evident to us. The problem we face is that machines are not able to process sound like people do. In order to give meaning to sound and convert it to words we need to use pattern recognition. With pattern recognition we can match sound waves to phonemes if we find similarities. Pattern recognition uses the statistical constructs that are previously described to find the probability of a sound being matched to a word. In speech recognition the pattern recognition algorithms use statistical inference to find the best word for a given sound. Some algorithms even provide the N-best words for a given sound. Important is that the confidence for the words are given.

The use of statistics in pattern recognition allows you to use probabilistic pattern recognition algorithms. This has many advantages. You are able to see how well a sound is matched to a word by a confidence score that is based on a statistical calculation. And if nothing is matched, or if the confidence score is too low then the algorithm is able to show that there are no meaningful results.

2.2.6 Hidden Markov Models

Almost all modern speech recognizers use a Hidden Markov Model for acoustic modeling. A Hidden Markov Model is a Markov Model that has unobservable states. The simplest Markov Model is a Markov chain. A Markov chain is a system that has a number of states. In a Markov chain the states are known. The probability of a sequence of state occurring is in this case as simple as following the states and calculating the probabilities from the state transitions. In a Hidden Markov Model however the states are not known. However the output and state transitions are known. So with a Hidden Markov Model you are looking for a combination of states that best matches the output.

There are three basic problems that need to be solved in order to use a Hidden Markov Model in Speech recognition.[Huang et al., 2001]

- 1. The evaluation problem: What is the probability of a model generating a specific sequence of states?
  Solving this problem allows measuring how well a Hidden Markov Model matches to a observation and allows to find the previous state by looking for the best probability for the previous state.

- 2. The decoding problem: What is the most likely sequence of states given a sequence of observations in a model.
  Solving this problem allows to find the best matching state sequence and uncover the hidden states to provide a solution.

- 3. The learning problem: How can you adjust the parameters to maximize joint probabilities?
  Solving this allows you to provide training data to the model to increase the probabilities and thus recognition in later uses of the model.

2.3 Design of a Speech recognizer

The techniques that a speech recognizer are built on statistical principles and have been explained in the previous section. In this section a a modern speech recognizer is decomposed and it's major architectural components are explained [Huang et al., 2001][Huang and Deng, 2010]. The basic stucture of a speech recognition system is displayed in Figure 2.2.

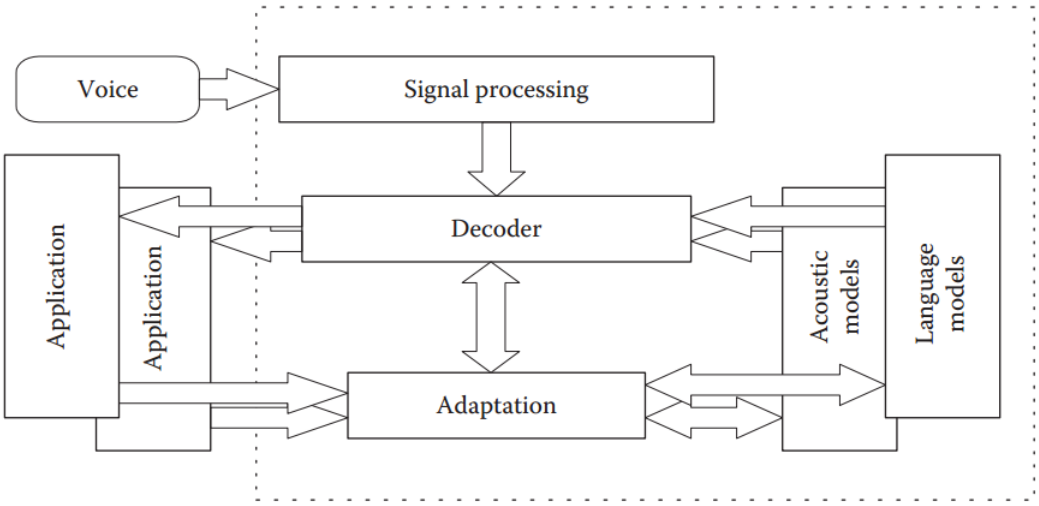


Figure 2.2: This is the basic structure of a modern speech recognizer [Huang and Deng, 2010].

2.3.1 Language modeling

The language model is the first important part of a modern speech recognizer. It is the model that contains specific information about the language that should be recognized. In essence this model contains the probabilities that a given word will occur in a sequence of words. A language model is created by analyzing a large amount of bodies of text.

A basic language model is a model that assumes the next word depends only on the prior history of words [Vertanen, 2009]. The most popular language model is the n-gram language model. In this language model the next word depends only on the prior (N-1) words [Vertanen, 2009]. A language model also defines a vocabulary. The vocabulary is a list of all the words that are used in the language model. If a word is not in the list, then that word is called out-of-vocabulary. Besides words, this list can also contain keywords. For instance the keyword "<s>" or "<sil>" is commonly used to refer to a silence. With this keyword you are able to model the beginning of sentence or a point where natural silences occur. The main advantage of a n-gram model boils down to a time vs space issue. A n-gram model is bigger and requires more space than a basic model. But it allows for a combination of words to be recognized which is more efficient than exhaustively searching for other possibilities. In Table 2.1 you can see an example of a part of an n-gram language model. The model that was used was a 5k non-verbalized punctuation 3-gram language model. In a 5k language model the vocabulary

contains approximately 5000 words.

probability	3-gram
-0.6582	a brussels based
-0.9145	a brussels court
-1.2917	a brussels hospital
-1.6048	a brussels hotel
-1.3290	a brussels house
-1.5283	a brussels suburb

Table 2.1: Example 3-grams from the 5k non-verbalized punctuation 3-gram language model from [Vertanen, 2007].

In some cases language models do not need to be complex. Consider a speech recognition system that only needs to recognize a number of commands, like "open menu" or "open file". In this case, instead of a language model, a deterministic grammar is used. This can only be used when the entire language combinations that should be recognized, in this case a number of commands is known. Figure 2.3 is an example of a deterministic grammar based on the JSGF grammar standard [Oracle, 1998]. In this example the keyword "<sil>" denotes a silence or a out-of-vocabulary word or sequence. In a model like this an occurrence of a word is not based on probabilities and thus the probability that a word will occur given it's predecessors is zero or a constant number.

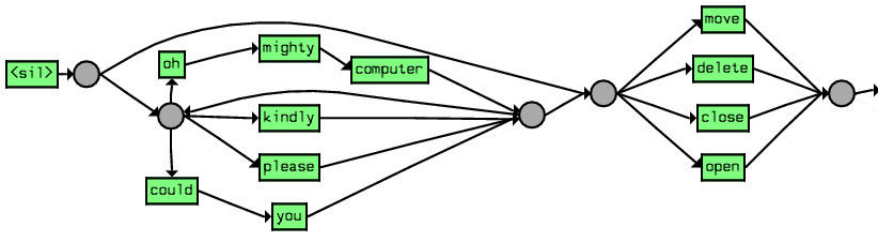


Figure 2.3: A deterministic grammar based on a JSGF grammar from [CMU Sphinx, 2011b].

2.3.2 Acoustic modeling

Acoustic models include the representation of knowledge about acoustics, phonetics, microphone and environment variability, gender and dialect differences among speakers, etc [Huang and Deng, 2010]. It models how words are said by decomposing them into phonemes. Words can have more than one phonetic decomposition. A word typically has two or less phonetic decompositions. A good implementation of an acoustic model deals with

speaker, context and environment variability. These are very important problems in speech recognition and will be discussed in a late section.

Phonemes are a good way to decompose words. But phonemes within a word can be dependent on the phoneme in front or after it. The reason for this is because of co-articulation. When a triphone model was first introduced in an open source speech recognizer it showed that the error rate decreased by 24-44% [Lee et al., 1989]. Now a lot of speech recognizers use a triphone model to represent words in their acoustic model. A triphone is commonly denoted as $x - y + z$. With x beign the monophone that precedes the monophone y and z the monophone that follows the monophone y. Herby describing the sounds around the phone y. An example of a acoustic model van be seen in Table 2.2. The same example as a triphone model can be seen in Table 2.3. In these examples the word below can be decomposed into two distinct phoneme sequences, and the chance that one word is pronounced in on of the both ways is equal. When training is done on this model one way of pronouncing below can have a higher likelyhood than the other. So in this case the model in the example is a general model that is not adapted to a specific speaker.

below	0.5	b ah l ow
below	0.5	b iy l ow

Table 2.2: Example from the 5k non-verbalized punctuation 3-gram dictionary from [Vertanen, 2007].

below	0.5	b+ah b-ah+l ah-l+ow l-ow
below	0.5	b+iy b-iy+l iy-l+ow l-ow

Table 2.3: Triphone conversion of the example from the 5k non-verbalized punctuation 3-gram dictionary from [Vertanen, 2007].

2.3.3 Decoder

The signal processing part of the speech recognizer transforms the acoustic signal input into a feature vector. The job of the decoder to find the best matching word sequence for that feature vector. It does this by searching the acoustic and language model. Therefore a decoder has to have a searching algorithm. Speech recognition search is usually done with the Viterbi decoder, or A* stack decoder [Huang and Deng, 2010]. The biggest problem facing the decoder is to efficiently search the different sizes of vocabularies. For instance a deterministic grammar will be easier to search than a n-gram language model. And a n-gram language model with a vocabulary of 5000

words will be easier to search than one witch a vocabulary of 60000 words. These differences will be noticeable when speech is processed and a user must understand that a bigger model requires more time. If time is a great issue to the user than smaller models should be considered.

2.4 Difficulties of speech recognition

2.4.1 Speaker variability and language complexity

As explained in the previous section, human speech is a series of sound waves. Every person has a *unique vocal anatomy*[Huang et al., 2001]. This is the source of the first and largest difficulty in speech recognition, speaker variability. This problem is addressed by using probability theory to make an educated guess. The recognition accuracy is aided by using a trained dictionary that is adapted to the speaker. For instance native speakers benefit from using a dictionary that is trained with samples from native speakers. Accuracy will be best if all samples in the dictionary are from the same subject and that subject is the speaker.

Not only do the voices differ from person to person. If a person were to speak a specific sentence two times, the sound waves would never be exactly the same [Forsberg, 2003][Huang et al., 2001]. The sound that men an women produce are also very different from each other. The tone of a woman's voice is notably higher and this is viewable in the sound waves. Besides anatomical differences, a person speaks in a specific way. For instance some speak in a specific dialect, while others speak formal. And when stressed a speaker will tend to speak at a faster speed than when relaxed. All these differences and more[Hirtle, 2004] make it harder for a speech recognizer to accurately compute a result. Seen has how there are a lot of parameters that make up the sound waves a person produces the best way to get good accuracy is to adapt the speech recognizer to a specific person. If a speech recognizer needs to be made to recognize a wide variety of people then the best way to handle this is to train the language and acoustic models to a large amount of people that have common properties with the intended users. For instance if French natives will use the recognizer in an English language than it is obvious the models should be trained to their dialects and specific tendencies.

Written language communicates to one person where spoken language is usually communication between two or more people. Written language is totally different from spoken language [Forsberg, 2003]. Some aspects of speech are not meant to be processed, because they are not valid. Spoken

language can contain hesitations, half spoken words, repetitions, slips of the tongue [Forsberg, 2003]. They can also contain coughs, or other sounds like background noise that are not meant to be recognized. When background noise is involved the quality and setup of the microphone should be checked. This problem can be solved by changing the location, quality or setup of the microphone. The problem of excess sound being produced is one that needs to be filtered out by the speech recognizer.

2.4.2 Environment and noise

For some sounds this is as simple as taking samples of the noise sound and identifying them in the recognition process. But for hesitations, slips of the tongue and half spoken words the speech recognizer to a more thorough analysis. These utterances are very hard to detect because they can be similar to utterances that are meant to be detected. Although the case described above might seem simple, it is only the case when background noise is removable, i.e. when you are alone in a room. But speech recognition should also be possible in a noisy environment. This might seem evident to a human, but it is very hard to distinguish two persons or two sources of noise from each other because they will be combined into one wave of sound. Environment variability is a big problem in speech recognition. A possible solution for this problem is to use a microphone array. This will not solve the problem completely, but can improve error rates [Seltzer, 2003]. The function of a microphone array is very simple. Because it has multiple inputs, you can analyze different acoustic signals an more accuratly determine the source of noise.

2.4.3 Ambiguity

In some cases the speech recognizer can not be held accountable for a wrong word sequence because of ambiguity in the language. Consider the existence of homophones. A homophone is a word that shares the same pronunciation with another word, but is different in meaning. Homophones can be spelled the same, but the problem becomes clear when we consider the case where the spelling is different. For instance the words ”knew” and ”new” or ”sea” and ”see” share the same pronunciation, and thus the same decomposition is phonemes. This problem can be helped by a good language model so that one word can be eliminated in a specific context.

A problem related to the homophone problem is the word boundary ambiguity problem [Forsberg, 2003]. The problem is best illustrated with an

example from [Forsberg, 2003], originally from [Gold and Morgan, 2000]:

- It's not easy to wreck a nice beach.
- It's not easy to recognize speech.
- It's not easy to wreck an ice beach.

These three sentences can be recognized from the same vocal input. This problem can be solved by leaving clear pauses between words or speaking in a slower rate. However this will make the speech recognizer fell less natural to work with. But keep in mind you are speaking to a machine, not a human.

2.5 Speech recognizers

In this section popular speech recognizers will be discussed. Each of them are used in different scenarios and target different uses.

2.5.1 Dragon NaturallySpeaking

Dragon NaturallySpeaking is a commercial speech recognizer developed by Nuance Communications. There are many different versions of Dragon NaturallySpeaking. Some of them are for professional environments like for legal or medical use. These versions have adapted models that fit these environments. The more general versions of Dragon NaturallySpeaking are "Home" and "Premium" and "Professional". Nuance claims to offer up to 99% accuracy out of the box with all of their products. This number is impressive, but seems very unlikely if we look at all the problems (in the previous section) that a speech recognizer faces. Dragon NaturallySpeaking is primarily used and meant to be used for dictation.

The software supports multiple user profiles. Every time you create a new user profile you will be prompted to train the speech recognizer for about 4 minutes as initial training. You can skip this training, however it is not recommended because training will improve accuracy. Additional training can be done to improve accuracy. Dragon offers dedicated software to guide someone trough a single training process.

A limitation of using Dragon is that developers can only access one result from the speech recognizer. Some speech recognizers offer a n-best result list, which can be useful because of homophones or word boundary ambiguity.

2.5.2 CMU Sphinx

CMU Sphinx is an open source toolkit for speech recognition. It offers a number of speech recognizers that were developed at Carnegie Mellon University. The most robust speech recognizer they offer is called Sphinx4. Sphinx4 is a completely adjustable and modifiable speech recognizer that is written in Java. The main advantage of using Sphinx is that it allows you modify any and every aspect of the speech recognizer, whether it is the decoder, language or acoustic model. The language and acoustic models are not included in Sphinx. They should be acquired from other sources.

The freedom you have when using Sphinx however has a downside. If you are making a speech recognizer for a specific purpose, you will need to invest a lot of time into learning, adapting and tweaking the speech recognizer for your use. CMU Sphinx offers a tool to train acoustic models called Sphinxtrain. This tool however is not likeunlike the software included in Dragon NaturallySpeaking or Microsoft Speech Recognition. It is a low level command line tool.

2.5.3 Microsoft Speech Recognition

Microsoft Speech Recognition is a feature of Microsoft Windows and comes bundled with it. The recognizer can be used as dictation software like Dragon NaturallySpeaking. It also includes a Speech Application Programming Interface or in short SAPI. This SAPI allows a programmer to create one or more instances of the Microsoft Speech Recognition engine in any ".Net" programming language. The advantage of using the SAPI is that you can get a n-best list of your recognition results. So if the recognizer mis-recognized a sentence for instance because of word boundary ambiguity then it is likely to be in the n-best list as an alternative to the recognized sentence. Another advantage is that you can create your own grammars. This is useful if you want to make an application act upon a series of commands.

Microsoft Speech Recognition offers a tool for configuring the speech recognizer and supports user profiles. Just like Dragon NaturallySpeaking it offers training right after creating a new profile. Additional training can be done to improve accuracy.

2.6 Conclusion

The accuracy of automatic speech recognition remains one of the most important research challenges after years of research and development [Huang and Deng, 2010]. Looking at the variability in speech discussed above there are a lot of factors that make speech recognition a difficult process. Speech recognition is still far from perfect. It is thus very important that speech recognition errors are corrected as efficiently as possible.

3
Related work

For personal computer systems, the keyboard is the most popular text-entry system. However for hand held devices, no single text-entry system is dominant [Ward et al., 2000]. This is why the field of mobile text-entry systems shows potential for research. Keyboards are dominant because many people are trained to use them. There were little or no alternatives to traditional keyboards, like there is now for hand held devices (different sizes, touch screen, etc.). The only alternative was a change of keyboard layout. In the competition between keyboard layouts the QWERTY keyboard made it, again because of familiarity. It was the layout most used on typewriters, dating back to 1878. Even with improved layouts like Dvorak that was made to increase performance and comfort [Martin, 1972], the QWERTY layout remained dominant [Ward et al., 2000].

Personal computer systems are slowly being replaced by new systems and different form factors, like mobile phones, tablets and smart television systems. These systems rarely use the traditional keyboard and mouse as input devices. Instead different modalities can be used. Multimodal text-entry is becoming an increasingly popular domain of research. One reason for this can be because users naturally tend to switch modality to correct errors produced using one modality[Suhm et al., 2001]. In this chapter we discuss developments in the area of (multimodal) text-entry systems that are important for this dissertation.

3.1 Dasher

Dasher is a data entry interface developed at Cambridge University and presented in the paper [Ward et al., 2000]. The basic concept of Dasher is that a user moves trough the interface space to the next letter he needs to spell a word. The interface space is a dynamic space that at the start is a list of the alphabet just like Figure 3.1 located at the far right of the interface. The height of the boxes around the letters are attributed to the statistical prediction of a letter occurring according to the language model that dasher uses. At the start all boxes are the same height, but once a letter is moved towards each screen refresh the box sizes are recalculated for a constant accurate visual representation according to the language model. *Dasher is made to be used on a personal computer system for people with limited mobility. The input method for moving through the interface space is a mouse, but eye tracking is proposed as future work.*

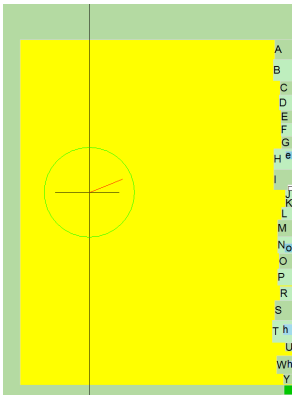


Figure 3.1: Boxes in Dasher

A letter is selected if it passes the middle of the screen highlighted by a small horizontal line. Each time a letter is selected it adds to the previous letter to potentially form a word. *For selecting a word, no clicking is involved.* Instead the interface is in constant movement and the center of the screen moves towards the mouse position and thus the letter closest to the mouse will come closer to the center of the screen. With that movement the probability box becomes bigger to cover almost the entire screen when near the center, like in Figure 3.2.

Continuous gestures

Dasher is driven by continuous gestures [Ward et al., 2000]. This has the advantage that inaccurate gestures can be compensated with later gestures.

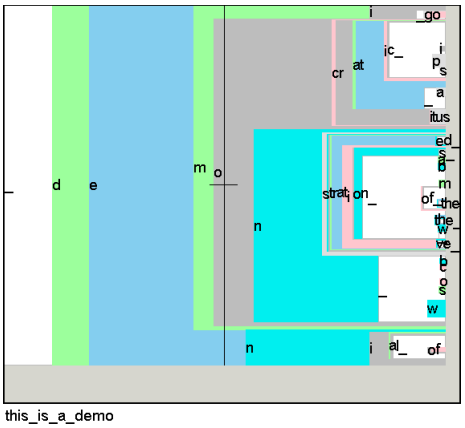


Figure 3.2: A demonstration of Dasher

But its continuous nature has the disadvantage that it requires sustained visual attention from the user [Ward et al., 2000]. This is in complete contrast with a keyboard, where little or no visual attention is required depending on the users skill level. A keyboard is also not continuous. It requires key strokes which can cause an action by the interface.

Language model

Dasher uses a language model and probability theory to adapt its interface to the English language. This is very much like speech recognition described in the previous chapter. The language model determines which letters are more likely to appear. This can aid the selection procedure. In Figure 3.3 we can see that the language model groups objection and object oriented as following the sequence object.

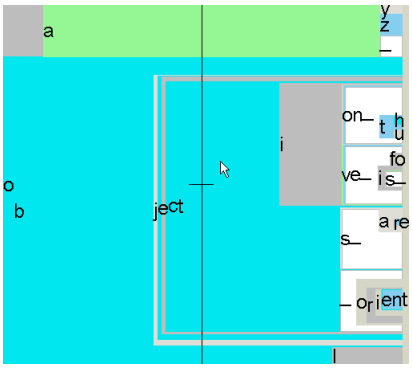


Figure 3.3: Selected "object" in Dasher

In theory this language model is a modeled to all words in the English language. However like in speech recognition this is impossible. First of all language models are trained to a large corpus. While this means that particular sequences will be very likely to occur. It also means that if you want to make a sequence combination that is not likely to occur, it will be more difficult to select. To help aid this problem the authors have created a minimum height for the boxes around the letters to be. Although it is still significantly harder to select letter sequences that are less likely to occur. According to the authors [Ward et al., 2000] a perfect language model should double Dasher's speed, which would make it about as fast as a keyboard.

Performance

Because Dasher requires uninterrupted visual attention, they did not let the participants enter text from hardcopy. Instead participants entered text dictated from the book Emma, from Jane Austen. This book contains 883 Kbytes of text. 18 Kbytes were selected for dictation and the remaining part was used to train the language model. Because of this training of the language model on a text that is very similarly written then the text that will be entered, the chances for success increase. The authors state that this experiment is modeled to a scenario for typical usage of Dasher.

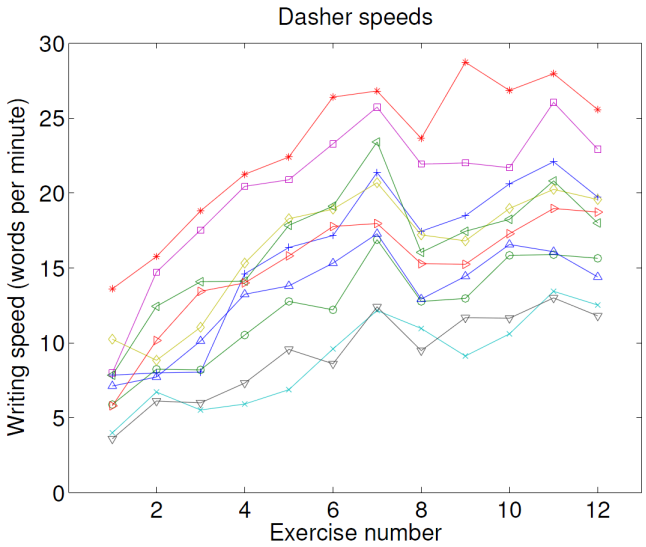


Figure 3.4: Writing speed (wpm) of dasher

Writing speeds using Dasher were initially low. In Figure 3.4 we see that in the first session the entry speed ranged from 3 to 14 words per minute (WPM). Slow initial speeds are attributed to the learning of the interface

for the participant. For all text-entry systems, potential speed depends on training. This is why we see an increase in writing speed using Dasher for all participants. However we do see a dip in efficiency for exercise 8. This is due to the fact that this exercise had a word which was not likely to be formed which caused participants to doubt its spelling. The maximum speed a person was able to get in Dasher was 34 WPM. This did not occur in the official experiment and was achieved by one of the authors.

3.2 Speech Dasher

Speech Dasher allows writing using a combination of speech and the Dasher user interface [Vertanen and MacKay, 2010]. *The interface also targets people with limited mobility and is used on a personal computer system. It can be controlled with a mouse or with Gaze. Gaze is a eye tracking system.* Speech Dasher extends the Dasher interface from the previous section by using the output of a speech recognizer to instead of using a letter per letter selection, selecting a complete word. So where in Dasher we selected letters, now select words, potentially speeding up the entire process. There can be different possible combinations and recognitions for one utterance. Only the top predictions are added into the interface directly. The other possibilities and the letter by letter selection option are reachable by selecting the special character "*". This was done because having many choices made the interface difficult to navigate [Vertanen and MacKay, 2010].

Multimodal

Speech recognition is not perfect. This is because there are many factors in human speech and the recognition process that make it hard to accurately recognize speech. Speech is a fast text-entry solution. Measurements have been made at 102 WPM [Vertanen and MacKay, 2010]. But the most challenging part is correcting the errors in the recognized speech. Studies have shown that correcting speech with more speech can cause returning errors and frustration [Cohen et al., 1998]. Furthermore, users tend to naturally want to switch modality to correct errors [Suhm et al., 2001].

Speech recognition

The Speech Dasher project used the PocketSphinx [CMU Sphinx, 2011a] speech recognizer. which is related to the Sphinx speech recognizer from the previous chapter. PocketSphinx is made to be used on devices with limited

computing power like mobile phones or tablets. A trigram language model was used in Speech Dasher that was trained with newswire text [Vertanen and MacKay, 2010]. A UK or US acoustic model was used depending on the best fit for a participant.

Performance

To evaluate Speech Dasher a user study was performed. Three participants were used in this study. One English, one American and one of German descent. This was done to test the significance of the speech recognition results. The participants were asked to write a number of newswire sentences containing from eight to twelve words. They were asked to enter this in Dasher and Speech Dasher to compare the difference in performance. The results can be seen in Figure 3.5.

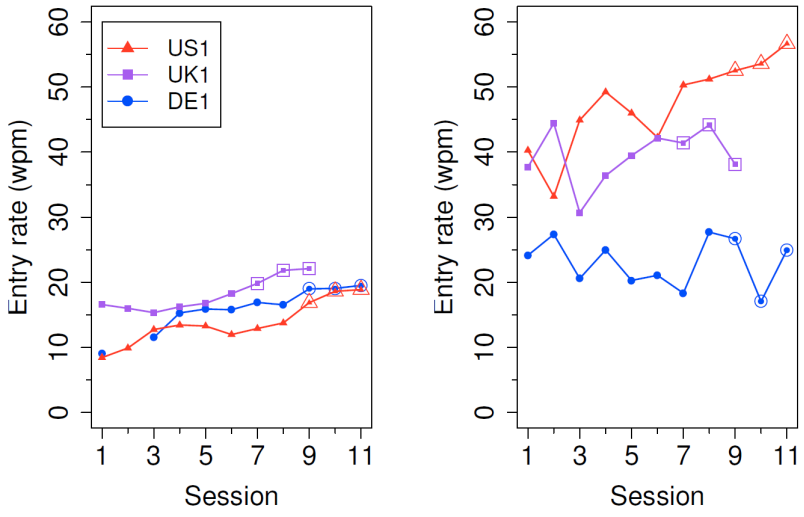


Figure 3.5: Performance of Dasher (left) and Speech Dasher (right)

In the results we can see that Speech Dasher is a significant improvement over Dasher for the English and American users. The performance for the German user is not as good because of the recognizer used a US acoustic model. The average text entry rate for Dasher was 20 WPM compared to 40 WPM for Speech Dasher. The word error rate for Dasher was 1,3% and for Speech Dasher 1,8%.

3.3 SpeeG v1

SpeeG v1 can be used for set-top boxes, game consoles and media centers in combination with a television screen. SpeeG v1 is a project that combines the Dasher user interface with skeletal tracking. Speech Dasher uses a mouse or Gaze as input device. In SpeeG v1 gestures are recorded from the skeletal tracking system and they are used as input device. Thus the SpeeG v1 prototype is a multimodal user interface that combines speech and gestures. The speech is used as primary input and gestures are used to correct errors from that speech input. To test efficiency a qualitative and quantitative study was performed.

Continious

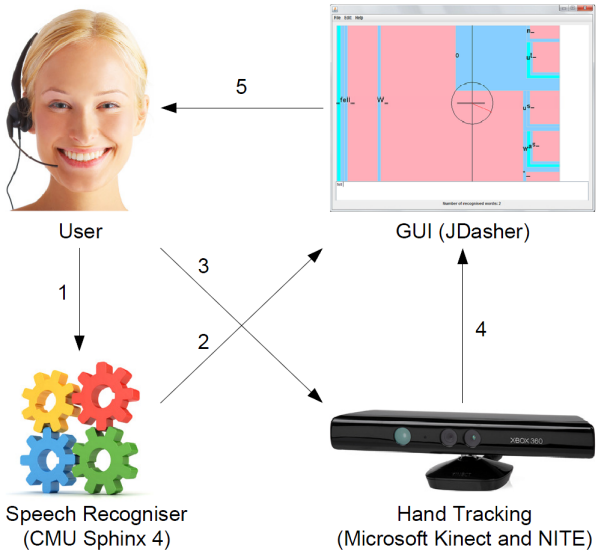


Figure 3.6: Data flow in SpeeG v1

The data flow of SpeeG v1 can be seen in Figure 3.6. Speech is processed by a speech recognizer and the result is sent to the JDasher[1] interface. Meanwhile the speech recognizer is still active and continues to recognize what the user says and adds it to the Dasher interface like a queue. This is conceptually different from the Speech Dasher project. In Speech Dasher a "MIC ON" button and "MIC OFF" button was used to define what utterances should be recognized. In Speech Dasher the continuous nature of the initial Dasher interface was lost. *In SpeeG v1 this continuous nature was reintroduced by supporting always active speech input.* SpeeG v1 used the Sphinx

[1]JDasher is a implementation of the Dasher interface in Java.

4 speech recognizer from CMU Sphinx and a Darpa Wall Street Journal Triram language model [Hoste et al., 2012]. The recognizer was modified to return a list of alternative sound alike words.

Microsoft Kinect

The Microsoft Kinect sensor is a game controller for the Xbox360[2] that was released in 2010. Libraries (Kinect SDK, OpenNI, NITE and FAAST) can be used to interact with the device. The Kinect sensor tracks the skeleton of persons in front of it. This allows a users body to become a input device. The initial intention was to use the human body as a game controller. However since the previously mentioned libraries can be used to interact with it people started to use it for different things. In SpeeG v1 it is used to track the gestures of a user.

Error correction

Correcting the errors that were made in the speech recognition process is vital to the performance of a text entry system using speech. The SpeeG v1 interface is differently implemented than the Speech Dasher interface. In Speech Dasher only the most accurate recognition results are shown. Other results and the original letter-per-letter Dasher approach are available via a special asterisk sign. In SpeeG v1 the original letter-by-letter Dasher interface was kept, but enriched with 2 to 20 alternative words. A special sign was inserted at the end to skip a word that was recognized but should not have been.

Performance

For SpeeG v1 a qualitative an quantitative user study was performed to check performance and usability of the SpeeG v1 prototype. In the study four text input systems are tested. In each system the same six sentences were entered. These sentences were proposed in the field of speech recognition from [MacKenzie and Soukoreff, 2003],[Garofolo et al., 1993] and were difficult for speech recognition. The result of the study can be seen in Figure 3.7 and Figure 3.8.

In the speech only test each participant used the speech recognizer from the SpeeG v1 prototype. Corrections were made only by re-speaking a failed word. After five failed attempts they were asked to move on. The Kinect-only test was a performance test done with a user interface that is included

[2]The Xbox360 is a game console produced by Microsoft. http://www.xbox.com

in the Xbox360. It is used for entering a search query. This interface uses only the Kinect sensor as input. In the SpeeG study each participant was asked to enter the sentences using the SpeeG v1 interface. A last study was conducted where the participants had to enter the sentences using a xbox360 controller and the visual on-screen keyboard of the xbox360 console. These four tests make up the quantitative study. In the qualitative study the participants were asked to fill in a form about their user experience in the quantitative study.

In Figure 3.7 text entry speeds are compared for each sentence and input interface. In Figure 3.8 the number of errors are displayed for each sentence and input interface.

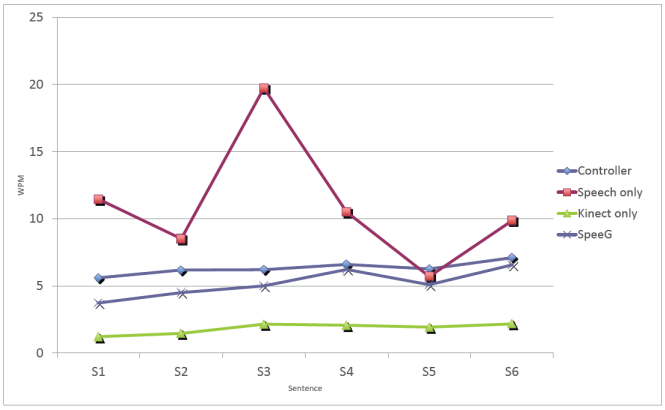


Figure 3.7: Mean WPM for each sentence.

Controller The controller interface is a bit faster than the SpeeG v1 interface with an average speed of 6,32 WPM. The number is similar to a related study with a virtual keyboard where the average was 5,79 WPM [Hoste et al., 2012].

Speech only Speech only is clearly the fastest entry method, with an average of 10,96 WPM. This is not a surprise because speech is a fast medium. However the number is low compared to 100 WPM which a speech recognizer is capable of handling. The time includes the time it took to correct the sentence by re-speaking words. The number of errors were high compared to the other tested text input systems.

Kinect only The Kinect only test is the slowest, with an average of 1,83 WPM. Participants made a large amount of errors. Two participants refused to finish the study because they found it too frustrating. Participants found

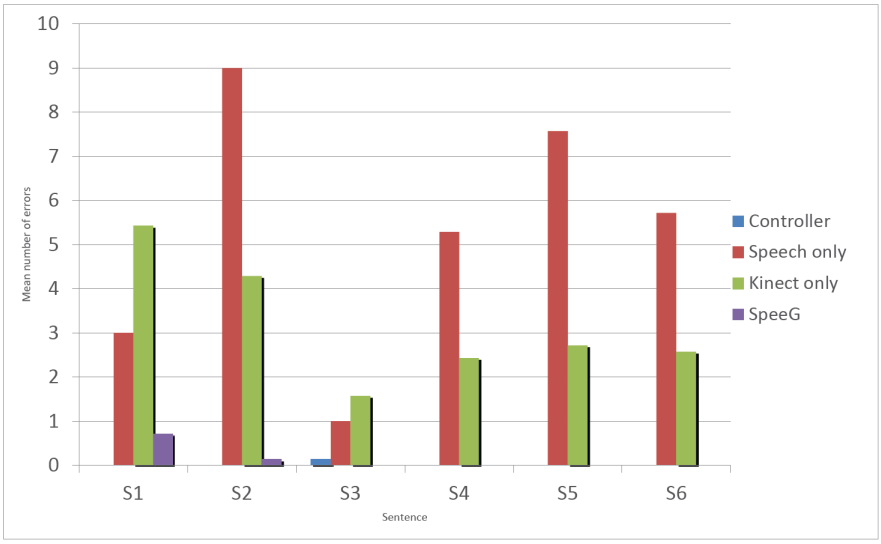


Figure 3.8: Mean number of errors for each sentence.

it frustrating because they made a lot of errors which can be seen in Figure 3.8. However there were more errors in the Speech only test.

SpeeG The SpeeG interface was slightly slower than the controller, with an average of 5,18 WPM. It also performed similarly in number of errors. The number of errors decrease as more sentences got produced. This can be due to the participants not being used to the interface. In the study of the Dasher interface this was also noticed. Speed increase and errors dropped as users got more experience with the interface.

Conclusions and future work

These are the conclusions and future work proposals from SpeeG v1:

Speech recognition The speech recognition was not performing well. There are a number of things that caused this. First of all, users spoke the sentences one word at a time and left clear pauses between words. The pauses cause the rate at which a complete sentence was recognized to be slow. It is also a less natural way of talking which may also have affected recognition. Because of the pauses only one word was recognized at a time. With word level recognition the language model becomes useless. A sound wave gets analyzed and decomposed into phonemes. This phoneme sequence will form a word. But because of homophones multiple words can be matched

to that sequence. There can also be multiple decompositions of the sound wave. Normally a language model is used to check for the likelihood of a particular phoneme sequence occurring given the previous words. However in word level recognition this is impossible since only one word will be uttered and the previous words are unknown.

Training All participants were non-native speakers with a wide variety of accents. This impacts speech recognition accuracy because the acoustic models that are available for the Sphinx speech recognizer are trained to native speakers. So a conclusion was to improve speech recognition by providing a minimal training phase for each user.

Automatic training is also suggested as an option to improve recognition accuracy. Since the recognized utterance is corrected by the user, it can be fed back to the recognizer so that it can learn from its mistakes.

Character-level A limitation of the SpeeG v1 prototype was that no out-of-vocabulary words could be entered. This can be useful when names should be entered. A character-level model is proposed as a solution.

Visual attention The SpeeG v1 prototype is based on the Dasher interface. It requires a users complete visual attention. A user is unable to perform another action while using this interface. This is not necessarily a bad thing but because of it a scenario where to tasks are combined (like sending a text message and driving a car) is excluded.

Physical strain During the evaluation of the SpeeG v1 prototype participants experienced discomfort. This was due to them having to keep their hand raised for the entire exercise. The combination of this and requiring complete visual attention is a fatiguing action. A proposition was made in [Hoste et al., 2012] to change the gestures to smaller ones. Also some special gestures can be added to invoke actions like skipping a word, confirming a word or entering character-level recognition.

Queuing speech The SpeeG v1 prototype combines speech and correction in a continuous way. Thus when you speak faster than you are able to correct the speech this is queued. In the interface this is not visualized. The reason for this is because the Dasher interface does not support it causing a user to doubt the last word he uttered and potentially making mistakes.

Game-like In the qualitative study a number of participants noted the interface to have a game-like element to it. "The navigation felt more like a game than an action" they said. More than half of the participants preferred the SpeeG v1 prototype over all others.

3.4 Parakeet

Parakeet is a text entry system for mobile touch screen devices. It was built to be used in a mobile scenario as an alternative to current text entry systems. Its design was guided by experiments and validated by a user study [Vertanen and Kristensson, 2009]. Parakeet just like SpeeG v1 is built to capture the efficiency of the human voice. It is similar in that both use a multimodal interface to correct their speech input. The most important task is to efficiently correct the mistakes from speech recognition. If speech recognition results are corrected by re-speaking that can cause the exact same error to reappear. This is why *avoid cascading errors* is the most important design principle in Parakeet and why speech is not used in the correction procedure.

Speech recognition

In a typical speech recognition interface, such as Dragon NaturallySpeaking, only the best recognition hypothesis is shown to the user [Vertanen and Kristensson, 2009]. A user needs to take explicit action to navigate to alternatives. However at that point the user is unaware of alternatives. Instead of hiding the *hypothesis space* behind a series of actions, Parakeet shows alternatives directly in the interface. Thus avoiding unnecessary actions from the user.

PocketSphinx was used as a speech recognizer. A US and UK acoustic model was used to cover different accents. The UK model was trained with 16 hours of data and the US model with 211 hours of data. To increase accuracy, gender specific models were created. A trigram language model was used with a vocabulary of 5000 words. After recognition the output from PocketSphinx was reprocessed by an algorithm the authors created. This algorithm prunes unlikely choices and checks for likely alternatives.

Continuous

Parakeet works in a two step process. First a user activates the microphone and speaks a sentence. When the sentence is spoken the microphone is disabled and the speech recognizer processes the voice input. The second

step is correcting the hypothesis from the speech recognizer. A system that uses this approach does not fully use the continuous nature of speech.

User interface

When Parakeet is started you have the option to record speech. This is controlled with a button that turns the microphone on and off. When the microphone is turned off the speech is processed. Then the user is greeted by a grid with on the top row the sequence of words that the speech recognizer finds most probable. A word sequence is selected by interacting with the touch screen. This interaction can be a single touch or a fluent motion. The grid and selection of "sales of imported cars and trucks are" can be seen in Figure 3.9.

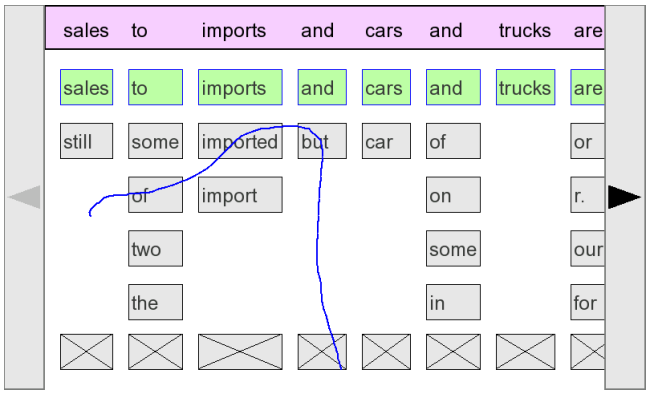


Figure 3.9: Navigating through Parakeet interface

Multimodal

The speech recognition results are prone to mistakes. This is especially the case in the mobile environment that Parakeet targets because environmental noise can be high. Since mobile devices are mostly equipped with touch screen this was considered to be the most practical interaction technique. Corrections can only be made by interacting with the touch screen. The touch screen is used to insert, edit, substitute or delete words. This interaction can be a single touch or a fluent motion. Editing a word is done with a predictive virtual keyboard seen in Figure 3.10. For writing a word to be inserted the predictive virtual keyboard is also used.

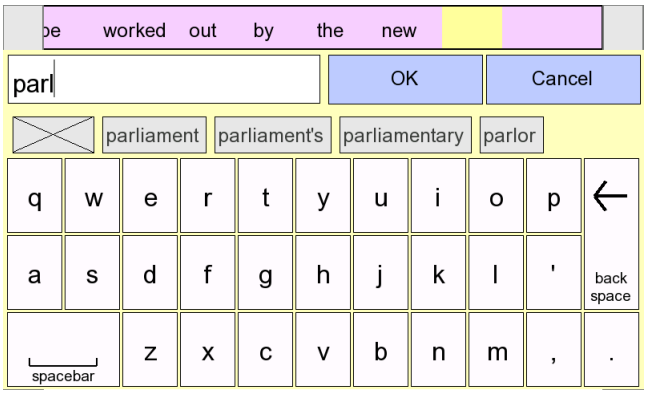


Figure 3.10: Predictive virtual keyboard in Parakeet

Performance

Parakeet was designed to be used in a mobile environment. So its performance was tested in a real world scenario by a expert user. This was done to show the potential of Parakeet. The results of the study can be seen in Figure 3.11.

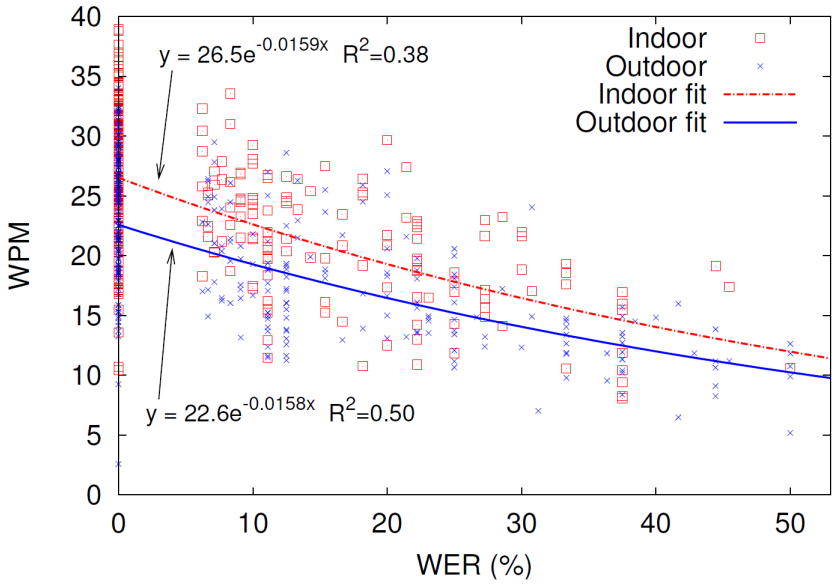


Figure 3.11: A plot of the entry and error rate for Parakeet

It is clear that the amount of errors are closely related to speed. This was the expected result and shows that correcting procedure is the most time consuming. In indoor conditions an average entry rate of 24,43 WPM was

recorded. Outdoor conditions caused the average entry rate to go down to 19,60 WPM. Given the conditions these results are good.

3.5 Overview

We have described a number of text entry systems. In this section we discuss how they compare to the requirements from Chapter 1.

Continuous nature of speech

SpeeG v1 is the only text entry systems that uses continuous speech input. Speech Dasher and Parakeet work in two steps where speech recognition is disabled in the second step for correcting the speech recognition hypothesis.

Consider non-native users

If we look at Speech Dasher we see that *speech has great potential to be a fast input device* because it outperforms Dasher. This can be seen in Figure 3.5. However we see that *performance is far worse for non-native English speaking users* (like DE1 in Figure 3.5). This is because for the English language there only exists a US and UK acoustic model. If Speech Dasher were to have offered training of the acoustic model, then the results for the non-native speaker (DE1) would have been better and thus a larger audience can be reached. None of the described text entry systems suggest training of the acoustic model to non-native users.

Sentence-level recognition

All text entry systems we reviewed that use speech recognition use it on a sentence level. However in the SpeeG v1 system none of the users used this feature. Instead each word was uttered separately. Because of this the language model was not used and the speech recognition is prone to more errors.

Avoid cascading errors

All text entry systems we reviewed that use speech recognition avoid cascading errors by providing a different modality for correcting the speech recognition hypothesis.

Visualize speech

In Speech Dasher a intermediate hypothesis is shown to the user from the first moment he speaks until the final hypothesis. Parakeet does not show partial results. The final hypothesis is shown when the speech recognizer has processed the voice input. In Parakeet partial result are harder to visualize because the speech recognizer doe not process the voice input in real time. It can not do this because of the limited computing abilities of a mobile device. SpeeG v1 also does not show a partial hypothesis from the speech recognizer. SpeeG v1 is not used in a scenario with limited computing abilities so partial results could have been shown.

Imprecise input

Dasher is controlled with a mouse. Speech Dasher can be controlled with a mouse or a eye-tracking system. SpeeG v1 uses skeletal tracking and Parakeet uses a touch screen as input device. A mouse is a precise input method. However eye-tracking and skeletal tracking are much more imprecise and prone to errors. The user interface has to take this into account. In the initial Dasher paper[Ward et al., 2000] eye tracking was proposed as future work and was already considered to be used. Imprecise input was considered to be used when Dasher was designed. Parakeet uses a touch screen as interaction method for the user. A touch screen is more precise than a tracking system, however larger areas should be considered because small targets can be hard to hit. So considerations should be made for touch screens as well.

Minimize physical strain

Physical strain is only an issue with SpeeG v1. The combination of gesture input combined with an interface that needs continuous attention from the user does not allow the user to rest his arms. Smaller gestures are proposed to minimize physical strain.

Overview

Given the requirements that were discussed here we have constructed this table as an overview of the section:

Requirement	Dasher	Speech Dasher	SpeeG v1	Parakeet
Continuous nature of speech		No	Yes	No
Consider non-native users		No	No	No
Sentence-level recognition		Yes	No*	Yes
Avoid cascading errors	Yes	Yes	Yes	Yes
Visualize speech		Yes	No*	No*
Imprecise input	No	Yes*	Yes	No*
Minimize physical strain	Yes	Yes	No	Yes

4

SpeeG v2

SpeeG v2 is the logical successor of research from the SpeeG v1 project. This project combines speech recognition and error correction in a continuous approach. To my knowledge there is no other project that researches this. Since it was a first investigation in this field there are points of improvement. These points were researched and a design principles were constructed from the research.

4.1 Design principles

Speech is the most natural way of communicating with each other. In a conversation people speak at a rate of 196 WPM [Yuan et al., 2006]. The average typing speed on a keyboard is only 38 WPM [Ostrach, 1997]. While speaking is about five times faster than typing, it is also a much more complex medium. The domain of speech recognition deals with these problems. A voice is like a fingerprint. Every person has a unique vocal anatomy. This is one of many difficulties in speech recognition. Speech recognition is never perfect because of the large amount of difficulties involved. For instance, a speech recognizer is more likely to have a better result at slower speaking rates. A speed of 100 WPM is generally acceptable instead of a conversation rate of 196 WPM. This is about the talking speed for a person who is giving a presentation.

A speech recognizer does not always give the correct result so the speed at

which text is corrected is very important. In most interfaces this is done in a two step procedure (like in: [Vertanen and Kristensson, 2009][Vertanen and MacKay, 2010]). In a first step the user decides when speech is recorded and decides when it is stopped by pressing one or more buttons. Then the audio is processed and the user get little or no feedback. The second step is to correct the errors made by the speech recognizer. In this step speech recognition is disabled and another modality is used. Our assumption is that combining the two steps into one will lead to a more intuitive and natural notion of speaking and communication. SpeeG v1 was the first attempt at combining these two steps. The project is described in detail in Chapter 3.

The design principle discussed below are directly linked to the Characteristics of SpeeG v2 in Chapter 1.

4.1.1 Speech Recognition

Sentence-level recognition

In SpeeG v1 speech recognition was done with the CMU Sphinx 4 speech recognizer. Word-level recognition was used by the users. A trigram language model was used, but the speech recognizer had no use for it since each utterance was one single word. This is not only an unnatural way of talking but also slow. The speech only study showed an average of 10,96 WPM. In SpeeG v2 one sentence is uttered at a time. This should increase accuracy because the speech recognizer can reason about the grammatical sequence of words. Thus making use of the language model. Speed should also be increased because one sentence will be entered in one utterance. In order to avoid people using word-level recognition SpeeG v2 prototypes do not accept word sequences containing less than 3 words. This also helps filter out noise like coughs or background noise.

Accuracy and training

A future work suggestion from SpeeG v1 was to increase recognition accuracy by training the speech recognizer and provide a profile for each user. Accuracy is very important. But the eventual goal of this work is to provide a text entry tool usable by a large amount of people. Although training can be efficient, like in the Parakeet project [Vertanen and Kristensson, 2009]. It requires a lot of training data. Asking people to go through a long training phase before even being able to enter text can irritate users. In SpeeG v2 a general speech recognizer is used to reach a large amount of people. There is also support for profiles and training the speech recognizer. Offering both a general and training possibility reaches a broad audience. Speech recogni-

tion accuracy is an issue with non-native speakers. People who want more accuracy can use the training feature. The training procedure takes about 7 minutes to complete. SpeeG v2 considers non-native users.

Character-level recognition

A comment on the SpeeG v1 project was to include character-level speech recognition for words that are not in the speech recognizers vocabulary, like names. SpeeG v2 intends to support character-level recognition with its spelling mode. Character-level recognition is more accurate than word-level because the vocabulary is smaller. In theory you only need twenty six letters in the vocabulary. A reasonably small vocabulary for words contains five thousand words. This large difference is noticeable in speech recognition accuracy. However, non-native speakers can have difficulty with spelling because some letters have different phonemes int their native language. To help increase accuracy in spelling mode, a natural language feature is included. This feature allows you to include a example words following your letter. An example is the utterance "a as in alpha". The speech recognizer can than evaluate the results from both the letter and the word to correctly recognize a letter. SpeeG v2 considers non-native users.

Explore hypothesis space

There are many difficulties correctly recognizing speech. Two examples are homophones and word boundary ambiguity. In these two cases, even if the recognizer correctly relates the speech to phonemes. There can still be multiple choices that are valid. A language model helps in finding a more probable choice. But the others could still have been correct. It is thus useful for a interface that uses speech recognition to explore the hypothesis space. A design choice that proved to be successful in the Parakeet project. SpeeG v2 explores the hypothesis space. This principle is also applied in Parakeet [Vertanen and Kristensson, 2009].

Speech recognizer

SpeeG v1 used CMU Sphinx 4. In SpeeG v2 different speech recognizers were researched. Popular choices are Dragon NaturallySpeaking, Microsoft Speech and CMU Sphinx 4. All of them support sentence-level recognition and training. In Dragon NaturallySpeaking there is no support to explore hypothesis space and in CMU Sphinx 4 if training is applied character-level recognition should be trained separately. SpeeG v2 uses Microsoft Speech as a speech recognizer because its feature set best matches the design principles. Microsoft Speech was more accurate than CMU Sphinx 4 in initial tests and

contains more features, like the natural language feature for spelling mode or the ability to train a speech recognition profile that also trains the spelling model.

4.1.2 User interface

Kinect

SpeeG v1 used the Kinect sensor as input device for navigating their user interface. Because of this there is no physical controller involved in the text entry stage. We also use the Kinect sensor because the same scenario as in SpeeG v1 is used. SpeeG v2 uses the Kinect sensor to correct speech recognition errors and avoid cascading errors. Skeletal tracking is imprecise input and prone to deviations in movement. Target areas are made sufficiently large to account for these deviations.

Visualize and queue speech

In the SpeeG v1 user interface there is no visualization of the results from the speech recognition if a previous word is not yet corrected. Because of this users can lose track of where they are in a sentence and errors might follow. In SpeeG v2 both partial and final speech recognition results are visualized. Speech that is not yet processed is always visualized so a user does not forget what he already said.

Physical strain

A comment on the SpeeG v1 project was that it put physical strain on its users. It is a phenomenon that is largely due to using the Kinect as a input device. Although the JDasher interface did not help. The JDasher interface requires the input device to be actively involved in navigation. Meaning that users had to keep their hand raised for the entire time they were using the interface. SpeeG v2 minimizes physical strain by allowing a resting position.

Explore hypothesis space

Since the JDasher interface has some properties that can hinder a good user experience other interfaces were researched. During the research it became clear that there is a lack of interfaces that combine speech and gestures from a depth sensor like the Kinect. However, a related field that use gestures are touch screens. A grid layout has been a successful layout for these kind of interfaces. Like in the Parakeet [Vertanen and Kristensson, 2009]

project. This is why the SpeeG v2 interface uses a grid layout to visualize the hypothesis space.

4.2 Prototypes

Since the design decision was taken to investigate a grid like structure we introduce a general grid user interface as the SpeeG v2 user interface. We introduce four prototypes. Each prototype shares the common grid user interface but is differently interacted with to correct speech recognition errors.

The Scroller prototype is based on the interaction of SpeeG v1. Horizontal movement of the right hand controls the speed and vertical movement controls which word is selected. The difference between both is that instead of the dynamic space used in SpeeG v1, the user interface contains a static grid with alternatives. Processing words is done on a step-by-step basis. The Scroller Auto prototype removes this step-by-step processing and introduces a dynamic grid that scrolls as words get processed.

The Typewriter prototype is based on the concept of a old-fashioned typewriter that needs to be pushed back to write on a new line. In the typewriter prototype both all movement of the right hand of the user controls the selection of words in a static grid. When the end of the grid is reached all words in the grid are considered processed and the next sequence of words is inserted in it. The Typewriter Drag prototype uses the same interaction with the sole exception that when the end of the grid is reached the user has to drag his hand across the user interface as if he was to push back a typewriter to move to the next line.

4.2.1 Interaction and architecture

The four prototypes share the same interaction with the speech recognizer and the skeletal tracking of the Kinect sensor. The difference between prototypes lies in the user interaction when correcting speech recognition errors (selection process). The common architecture is displayed in Figure 4.1.

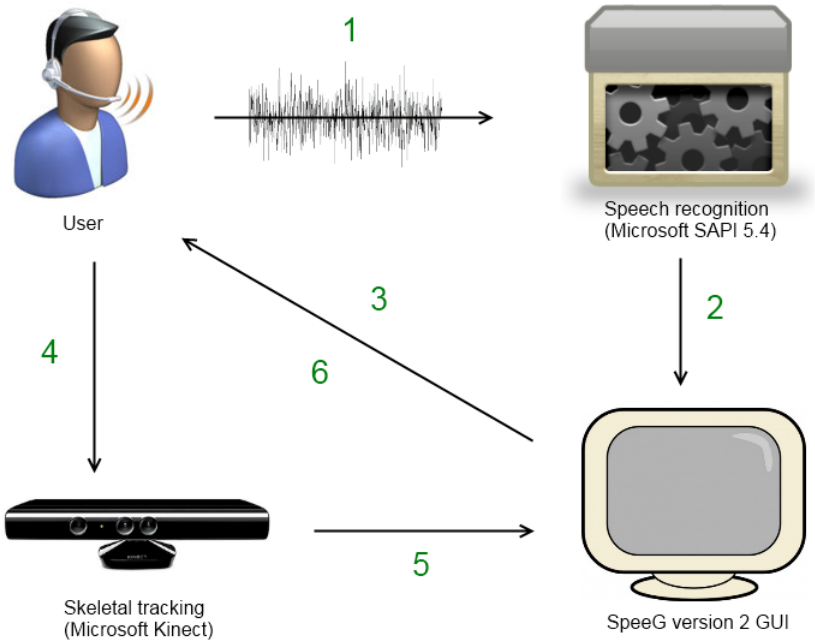


Figure 4.1: Interacting with SpeeG v2

First a user will utter a sentence (1) and the speech recognizer will translate this into a word sequence (2). At any point when a user speaks the SpeeG GUI will visualize what the speech recognizer thinks to be the correct word sequence (3). Even when the user has not finished his sentence partial results will be shown. When the user has spoken his sentence completely and the speech recognizer has sent its final hypothesis to the GUI, the user is informed (3). Then he can start correcting the recognition result using his right hand as input device (4) and skeletal tracking to convert this information to coordinates (5). The user is informed of the skeletal tracking information in the GUI (6). The selection process differs in each prototype and will be explained later.

Because of the continuous nature of both input devices the sequence 1, 2, 3 is independent of sequence 4, 5, 6 and can overlap or occur in parallel. Because of this the continuous nature of speech is kept and speech can be used at any point in the correction process. Communication between the Speech recognizer and GUI is done over a socket connection. This allows for abstraction and evolution of both applications independent of each other. In future work one might want to use a different speech recognizer or another GUI. This evolution is supported with this architecture.

4.2.2 General features

In the design principles of the User Interface a grid layout was proposed because it has been proven successful in previous work. This layout is related to that of a interface for touch screens. Motion from the Kinect gestural tracking is not as accurate as that of a hand on a touch screen. Because of this interacting areas were made sufficiently large so to correct for less accurate movement. The interface that was developed can be seen in Figure 4.2. All prototypes have this interface in common.

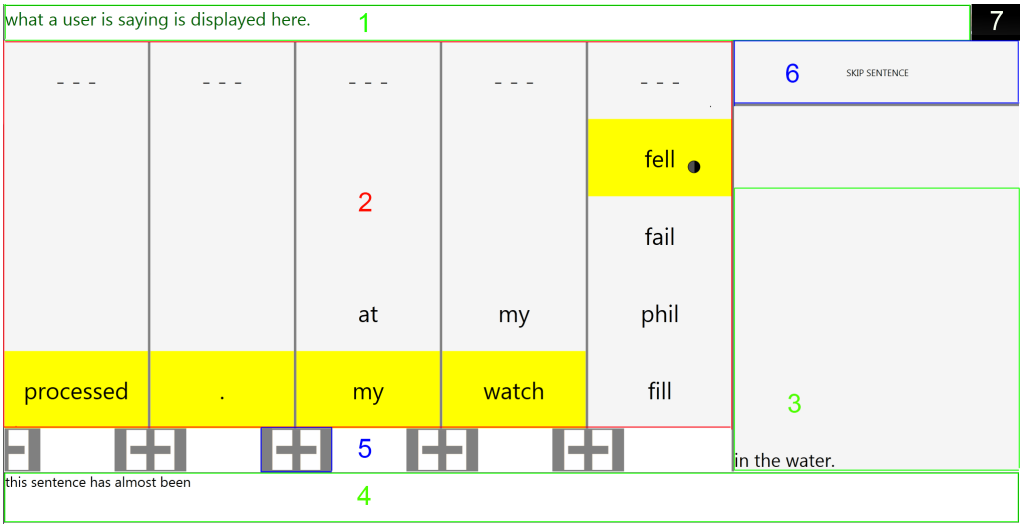


Figure 4.2: The SpeeG v2 common user interface witch each part highlighted

1. Visualizing what a user is saying:
   When a user first starts to speak feedback will be shown in this part. Intermediate results will be shown in black and once the speech recognizer has sent its final hypothesis the text will turn green. A valid sentence is a sequence of three or more words. If a word sequence of less words is recognized than that text will be colored red and not considered a sentence. This is done to filter out noise and short involuntary utterances. After a valid sentence is recognized and a short delay this text is sent to part 3 (What still needs to be processed).

2. Processing speech recognition results in a grid layout:
   This gird contains all the words that are being processed. In Figure 4.2 this is the word sequence "processed . my watch fell". There are two sentences being processed here. The first is "this sentence has almost been processed ." and the second one is "my watch fell in the water .". A dot is considered a word to provide a clear separation between sentences. In the grid the most likely words are on the bottom. In Figure

4.2 the speech recognizer found "fill" to be more likely than "phil", "fail" and "fell". In this case it is corrected and "fell" is selected, since this is what the user uttered. In the top row of every column there is a possibility to skip one particular word denoted by "- - -". A word that has a yellow background is the selected word. The way in which a user interacts with this grid (and selects alternatives) is different for every prototype and will be explained later. After a sequence of words is processed it is sent to part 4 (What has been processed).

3. What still needs to be processed:
   This part contains the word sequence that follows the word sequence from part 2. In Figure 4.2 "in the water ." is part of the sentence "my watch fell in the water ." which is being processed. The design principle to queue speech is the direct cause for creating this part. If a user was to speak a few sentences and only then start to correct them he would be able to know what was already spoken with this visualization.

4. What has been processed:
   All words that have been processed are visualized here. This is the part that contains the corrected text.

5. Insert word(s):
   The blue box which looks like plus sign is used to insert one or more words between the two words it is located under. Every prototype has this insert feature. If the insert feature is activated the following screen (Figure 4.3) will be shown to the user. In the middle a box will appear with the option to insert or cancel. The insert feature works with the concept or respeaking. If the user utters one or more words when this feature is active it will be shown in the box. If the recognition result is incorrect the user simply needs to utter it again. The user can repeat this as many times as he wants. When one or more words are selected to be inserted, they are put on the location of the selected plus sign. In Figure 4.3 this is between "my" and "watch".

6. Skip sentence:
   If a sentence is mostly incorrectly recognized or if errors were made, the user can use the "skip sentence" feature. When activated it will delete the sentence that has the most words in the grid. In the case of Figure 4.2 there are more words from the sentence "my watch fell in the water ." than from the previous one. So it will be deleted.

7. Kinect input:
   Part 7 is the visualization of what the Kinect sensor is seeing. The user can see himself in it. Depth images from the Kinect sensor are analyzed. By moving his right hand the user can control the black

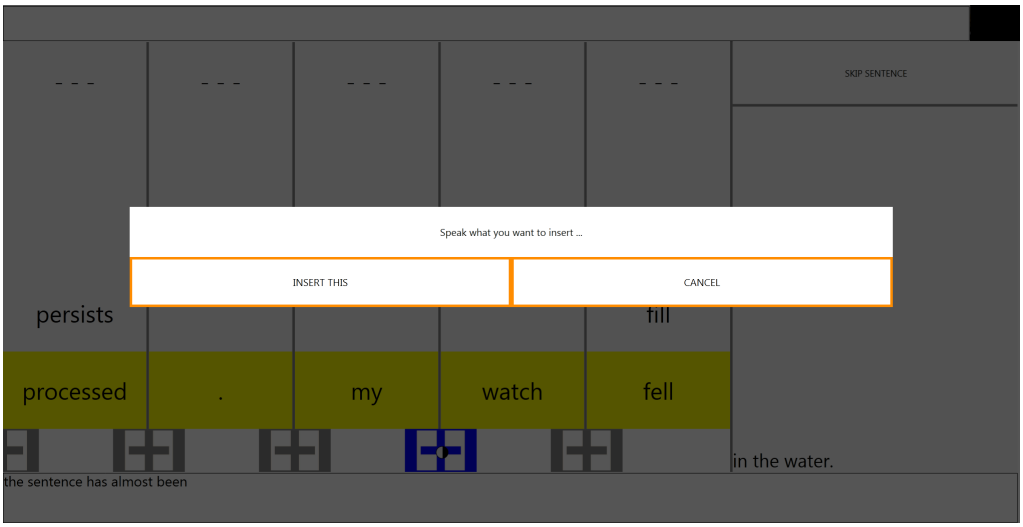


Figure 4.3: The insert screen that is shown on top of the general UI to insert one or more words

circle seen in Figure 4.4. When the insert or skip sentence features are selected the black circle will slowly turn orange. Once it is completely orange the feature is activated. This gives the use some time to correct himself when a feature is accidentally selected.

Figure 4.4: The insert screen that is shown on top of the general UI to insert one or more words

4.2.3 Correction methods

We introduced four ways to correct speech recognition results. For each error that is included in a speech recognition hypothesis there is a way to correct them. We give an overview of the correction methods that can be used and explain what errors should be fixed with that correction method.

Alternative list

The alternative list are the list of words in one column of the grid. It is a way to correct words that are *substituted* in the speech recognition hypothesis compared to the desired result.

Spelling

The spelling mode is activated and deactivated with a left hand up gesture. This correction method is provided to correct a word in detail. Just like the alternative list it is a way to correct words that are *substituted* in the speech recognition hypothesis compared to the desired result.

Insert

The insert feature is located on the bottom of the grid and iconized by a plus sign. It is a way to correct words that are *deleted* in the speech recognition hypothesis compared to the desired result.

Delete

The delete correction method is visualized by three vertical lines on top of each column in the grid. When it is selected that word is skipped. It is a way to correct words that are *inserted* in the speech recognition hypothesis compared to the desired result.

Skip sentence

The skip sentence feature is visualized by a button located by the top right corner of the grid. It is a way to delete a speech recognition hypothesis because it deviated greatly from the desired result. The reason for its use can be because of poor accuracy, background or accidental noise. Although it was designed to be used in the cases of excess noise it can be used when speech recognition accuracy is notices. We do not recommend its use for poor accuracy because it can lead to cascading errors.

4.2.4 Scroller Prototype

The Scroller prototype is loosely based on concepts from the Dasher interface. The interface is still controlled by navigating towards the next word. But in this case the central column is the active column where the user is selecting words. Moving the black circle in the grid both controls the speed of the Scroller and what word is selected in the active column. Moving it on the y-axis controls what word is selected, which is clarified in Figure 4.5 with a blue arrow. A green bar is put in the interface to show how much time remains before switching to the next word. Moving the circle on the x-axis controls the speed in which the bar turns grey. This is highlighted in Figure 4.5 with numbers. When the bar is fully grey a step occurs and the

next word is put into the active column. This prototype works on a step-like basis. If the circle is in the center column the speed will be 0, meaning the Scroller is paused. The speed at which the bar turns grey is increases as the numbers in 4.5 increase. Negative numbers means that once the bar is completely grey the next word is the one before the active one. Hence providing a way to go back and correct a possible error that was made.

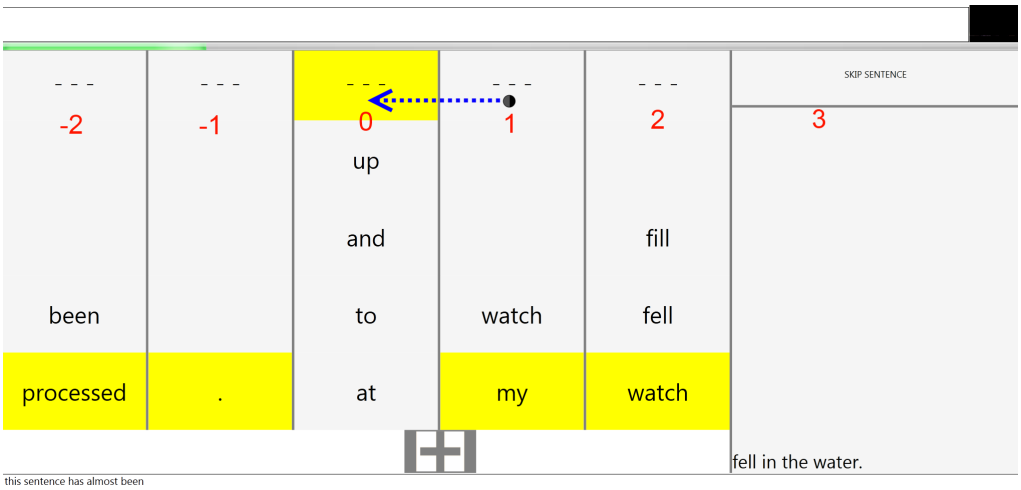


Figure 4.5: Scroller prototype interface where each speed is denoted with a number

4.2.5 Scroller Auto Prototype

This prototype is a variation of the Scroller prototype. The difference is that the green bar is removed and columns will move automatically. Moving the circle on the x and y-axis still controls the speed and what word is selected. With this prototype the navigation is not on a step based but continuous. Figure 4.5 shows the Scroller Auto prototype. The active column is the one that is most has the most presence in the central column. In the case of Figure 4.5 the column with the words ”might” and ”my” is active and ”might” is selected because the black circle is aligned with it.

4.2.6 Typewriter Prototype

The Typewriter prototype is based on the popularity of typewriters. Even though typewriter are seldom used nowadays, people still remember them and know how they were used. This is a knowledge that this prototype is aimed to exploit. It can decrease learning time and increase usability.

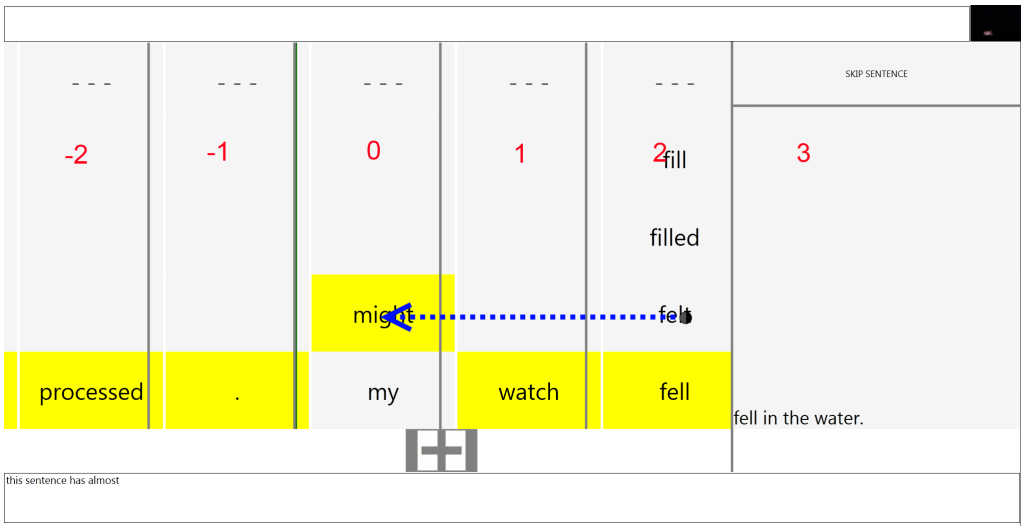


Figure 4.6: Scroller Auto prototype interface where each speed is denoted with a number

Navigation through the grid is different from previous prototypes. Now selecting a word is not dependent of an active column. Instead the word underneath the black circle is selected which is highlighted in Figure 4.7 with a blue line. One the red area in Figure 4.7 is reached the words will be pushed away just like a typewriter is pushed back at the end of a line and the next words that need to be processed appear. This prototype was built to navigate fast though hypothesis space. It has the downside that when a error is made and the red zone in Figure 4.7 is reached there is no going back.

4.2.7 Typewriter Drag Prototype

What differs this prototype from the Typewriter prototype is that once the red zone from Figure 4.7 is reached the user has to drag the stacks back just like you were to push the head of a typewriter back to start a new line. Also mistakes can be undone because there is a back button which allows dragging in the opposite direction. Once dragging is activated the columns that were processed change color and dragging is stopped by moving in the opposite direction. Then the user interface will be updated.

4.2.8 Spelling mode

A feature that is not visualized in Figure 4.2 or discussed in the prototypes is the spelling mode. It is a very important feature. If a word is not in

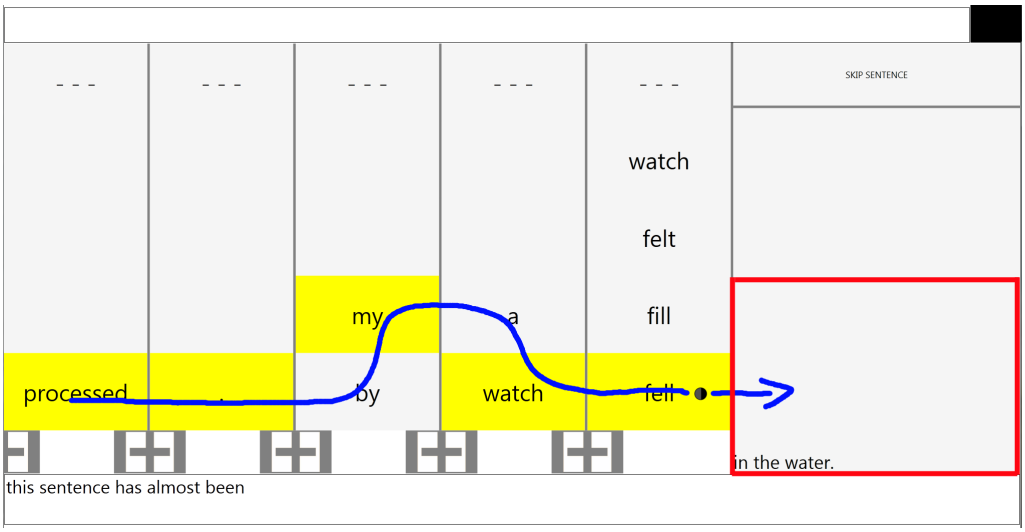


Figure 4.7: Navigating through the Typewriter interface

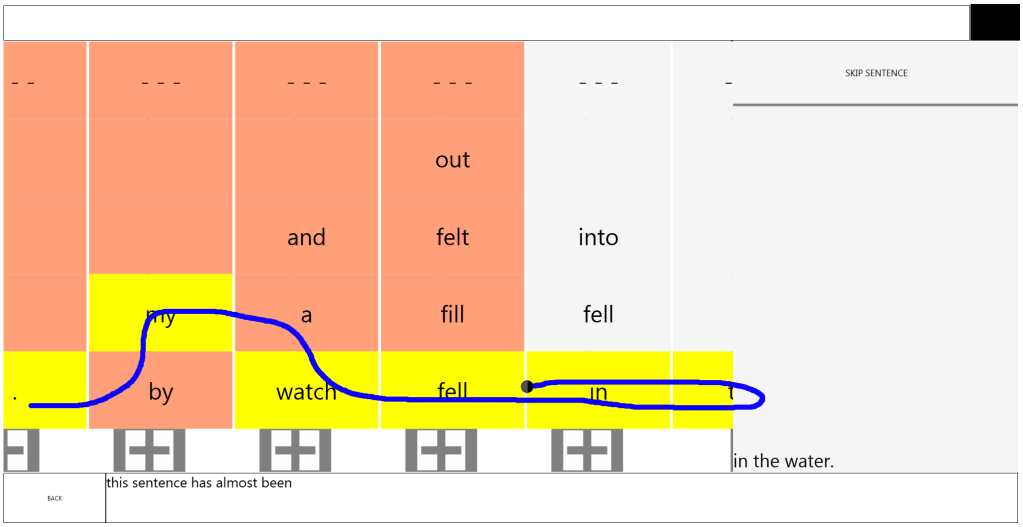


Figure 4.8: Dragging in the Typewriter Drag interface to move on

the you can choose to correct it with spelling. Spelling mode is activated when a user lifts his left hand over his head. It can only be activated again if his hand goes below the head in a resting position. You can see spelling mode in Figure 4.9. A user notices he is in spelling mode when the borders between columns are purple, each column contains only letters and a filler denoted with "*end*" is behind the last letter.

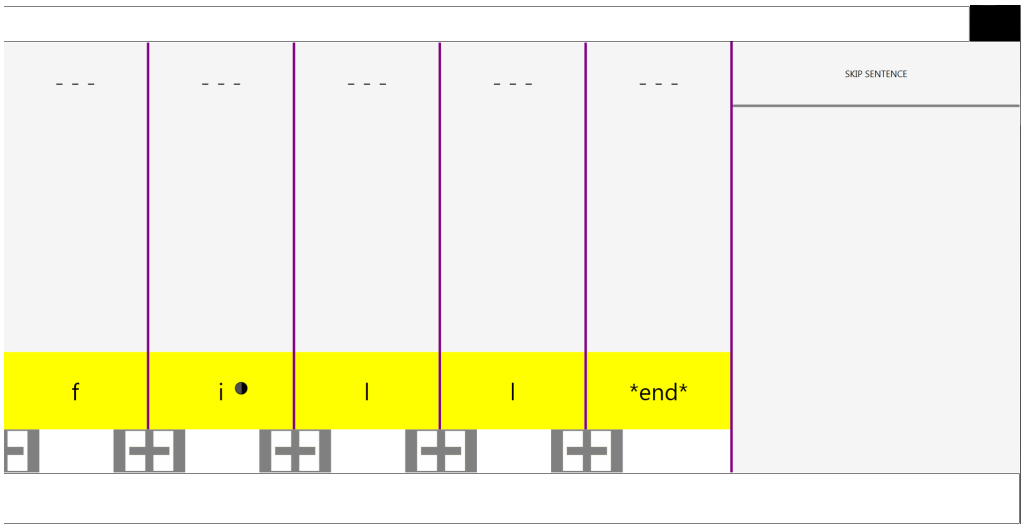


Figure 4.9: Correcting the word "fill" in spelling mode

The way in which the spelling grid is navigated is different for each prototype. It is the same way as the grid would be navigated when not in spelling mode. With the exception that in spelling mode each time a user utters a letter it will overwrite a existing letter in the word. The letter that is overwritten is located in the currently active column. In the case of the Scroller prototypes it will be the center column and for the others the active column is the one located under the black circle, which the users controls.

When spelling mode is activated the speech recognizer switches from a full English vocabulary to a more restricted one allowing only words of one character, which are usually letters, but can also be numbers. Microsoft Speech has a natural language feature that allows a user to elaborate on his spelling by following it with the keywords "as in". For example a user is able to say "b as is beta" to clarify that he means the letter "b". Because of errors in recognition can be corrected with more accurately.

5 Evaluation

5.1 Evaluation strategy

To evaluate the speech recognizer and proposed prototypes we have conducted a quantitative and qualitative user study. First we examined the accuracy of the speech recognizer on nine participants. Then the prototypes were evaluated to check their performance and usability using those same nine participants. This section describes how we the evaluation took place.

5.1.1 Participants and method

The study featured nine participant. All but two participant have a computer science background. Among the participants there were eight native Dutch speaking and one French. The participants have a variety of different accents coming from different parts in Belgium. For each user the English US acoustic model was used in the Microsoft Speech Recognizer 8.0. No training was applied to the speech recognizer for any user. Participants P1 to P7 conducted the study in the same room. The room had air conditioning and suffered from some background noise. To record audio a headset was used because it minimizes background noise and has good acoustic properties. Participants P1, P2, P3 and P4 used a Sennheiser PC 21-II headset in the study. The others used a Creative Fatal1ty headset. Participants P8 and P9 conducted the study in a zoom with less background noise.

5.1.2 Performance measure

Words per minute (WPM)

In each test the time it took the participant to process the sentence was recorded starting from the point the first sound of the sentence was uttered until the time the punctuation dot was processed. To compute the text entry speed the standard measure for a word was used. A word is considered a five character sequence, including spaces and punctuation. The text entry speed is computed in words per minute (WPM).

Example: The sentence "He will allow a rare lie." is exactly 5 words long. Entering this sentence in 15 seconds would result in a text entry rate of 20 WPM.

Word error rate (WER)

In all tests the word error rate was computed. This number is the percentage of words the speech recognizer wrongly hypothesized. The standard for computing the word error rate was used.

$$WER = \frac{S + D + I}{N} \tag{5.1}$$

S being the number of substitutions appearing in the result hypothesis, D the number of deletes, I the number of inserts and N the number of words in the sentence. Instead of keeping track of each, the number $(S + D + I)$ was noted as the number of errors from the speech recognizer in each test. In this computation there is a possibility to have more errors than words. In that case the WER exceeds 100%.
Note: here a word is considered an actual word, not a five character sequence.

Example: A user utters the sentence "He will allow a rare lie". The speech recognizer hypothesis is "He wheel allow rare lie in". The word error rate is 50%.

$$WER = \frac{S + D + I}{N} = \frac{\#(wheel) + \#(a) + \#(in)}{\#(He, will, allow, a, rare, lie)} = \frac{3}{6} = 50\% \tag{5.2}$$

Correction methods

For each user the correction method applied to correcting the speech recognition hypothesis was recorded. We compare correction methods to relate to the performance. The user has the choice of using the alternative list,

spelling, the insert feature or the delete option. When recognition accuracy was bad the option to skip the sentence and re-speak it completely could be used.

5.1.3 Speech only

Participants were asked to read these six sentences. These sentences originate from [Garofolo et al., 1993], [MacKenzie and Soukoreff, 2003] and are commonly used in the speech recognition community. To measure the amount of errors the speech recognizer made, each participant was asked to read each sentence. When the sentence was recognized correctly they were asked to move on. If one or more errors appeared in the recognition result this amount was noted. A participant was asked to repeat the sentence up to a maximum of five times when an error occurred. The number of tries each participant took was noted along with the average number of errors in a sentence.

These are the sentences:

- S1: This was easy for us.
- S2: He will allow a rare lie.
- S3: Did you eat yet.
- S4: My watch fell in the water.
- S5: The world is a stage.
- S6: Peek out the window.

5.1.4 Prototypes

To test the performance of the prototypes each participant was asked to test each prototype. To start a prototype was chosen uniformly distributed over all participants. When a Scroller prototype was chosen it was followed with its closely related Scroller Auto to minimize confusion and vice versa. The same was done for the Typewriter and Typewriter Drag prototypes when one of them was chosen. After a short introduction the participants had five minutes to become familiar with the prototype. Then participants were asked to produce the same six sentences from the speech only test. For each sentence the time and the number of errors the speech recognizer made was recorded. What feature the participant used to correct the sentence was also recorded. After this quantitative study a qualitative study was conducted

to investigate the usability of the prototypes. Each participant was asked to fill in a questionnaire on their experience with the prototypes and speech recognition. This questionnaire can be found in Chapter A.

5.2 Results

5.2.1 Speech only

Words per minute

The text entry speed per sentence for each user are displayed in Figure 5.1. The mean speeds per sentence are displayed in Table 5.1.

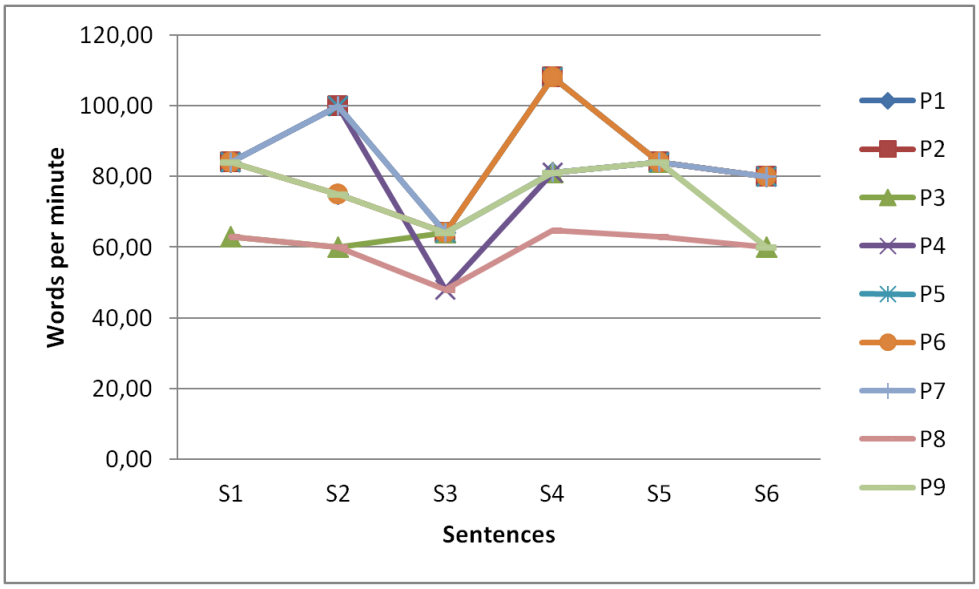


Figure 5.1: Speech only text entry speed per sentence for each user

S1	S2	S3	S4	S5	S6
79,33	82,78	60,44	88,20	81,67	73,33

Table 5.1: Mean WPM for each sentence in speech only

The sentences were all produced in between 48 and 108 WPM where the mean value for all participants was 77.63 WPM. Participant P2 spoke the fastest in each sentence. This participant has experience with speech recognition and felt very comfortable using it. In contrast the slowest speaker was participant P8. This participant never used speech recognition, and

does not have a computer science background. When this participant spoke, clear pauses were left between each word even when given the instruction to speak as natural as possible.

Words error rate

The WER per sentence for each user are displayed in Figure 5.1. The mean WER per sentence is displayed in Table 5.2.

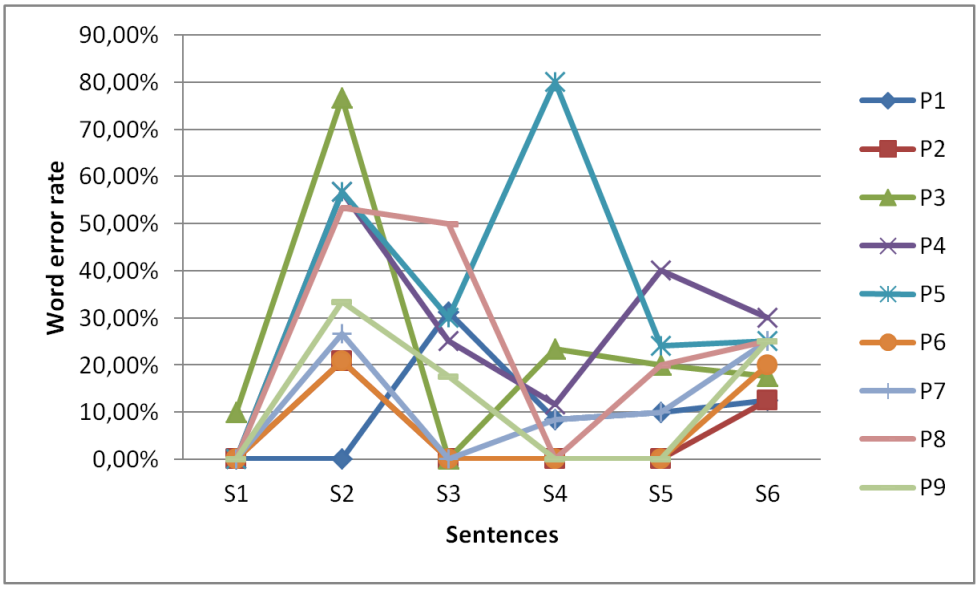


Figure 5.2: Speech only word error rate per sentence for each user

S1	S2	S3	S4	S5	S6
1,11%	38,33%	17,08%	14,63%	13,78%	21,39%

Table 5.2: Mean WER for each sentence in speech only

The amount of errors that occur are dependent on the complexity of the sentence and the person that pronounces it. We know that all sentences are of sufficient complexity since they are used in the field of speech recognition to test recognizers. Sentence S1 was best recognized for each participant, with a WER of 10% for participant P3 and no errors for all other participants. The speech recognizer had the most difficulty with sentence S2. Except for participant P1 everyone experienced a WER of 20% or more. In all other sentences we see a variaety of results. Some participants experience little or no errors while for others the recognition is far from perfect with a WER

going up to 80% maximum. However for most participants the WER stayed below 50%, with the exception of sentence S2.

5.2.2 Scroller

Words per minute

The text entry speed per sentence for each user are displayed in Figure 5.3. The mean speeds per sentence are displayed in Table 5.3.

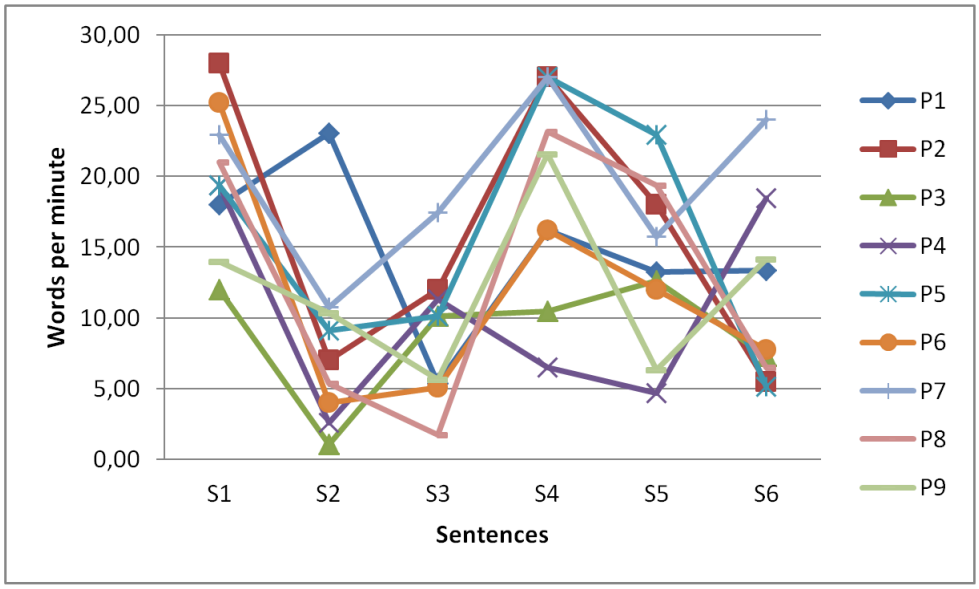


Figure 5.3: Scroller text entry speed per sentence for each user

S1	S2	S3	S4	S5	S6
19,99	8,13	8,74	19,45	13,87	11,33

Table 5.3: Mean WPM for each sentence using Scroller

All sentences were entered in between 1,05 and 28 WPM using the Scroller prototype. For each Participant the greatest speed was recorded for either sentence S1 or S4. With the exception of participant P1 who had his greatest speed processing sentence S2. This sentence was the most difficult to process for the other users and thus also has the lowest mean value of 8,13 WPM.

Words error rate

The WER per sentence for each user are displayed in Figure 5.4. The mean WER per sentence is displayed in Table 5.4.

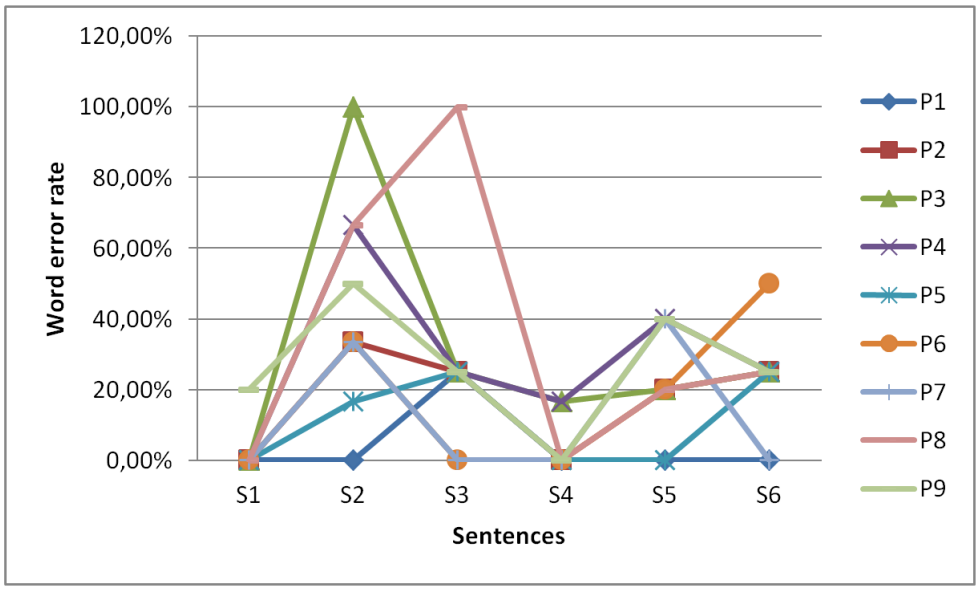


Figure 5.4: Scroller prototype word error rate per sentence for each user

S1	S2	S3	S4	S5	S6
2,22%	44,44%	27,78%	3,70%	22,22%	22,22%

Table 5.4: Mean WER for each sentence using Scroller

The lowest mean WER occurs for S1 and S4. In sentence S1 participant P9 had an error while the others did not. Sentence S2 contains the most errors. Only two participants managed to get a WER of less than 30%. From the figures we see that the WPM is dependent on the WER. Take for instance Participant P2. He reaches the highest entry speed out of all participants for sentence S1 and S4. However when the WER goes up his entry speed goes down to a low point of 6,98 for sentence S2 also reaching his highest WER of 33,33%. The same can be said for the other participants. Highest speeds are reached when no errors occur and lowest speeds confirm to a high WER.

Correction methods

The used correction methods for the Scroller prototype are displayed in Figure 5.5.

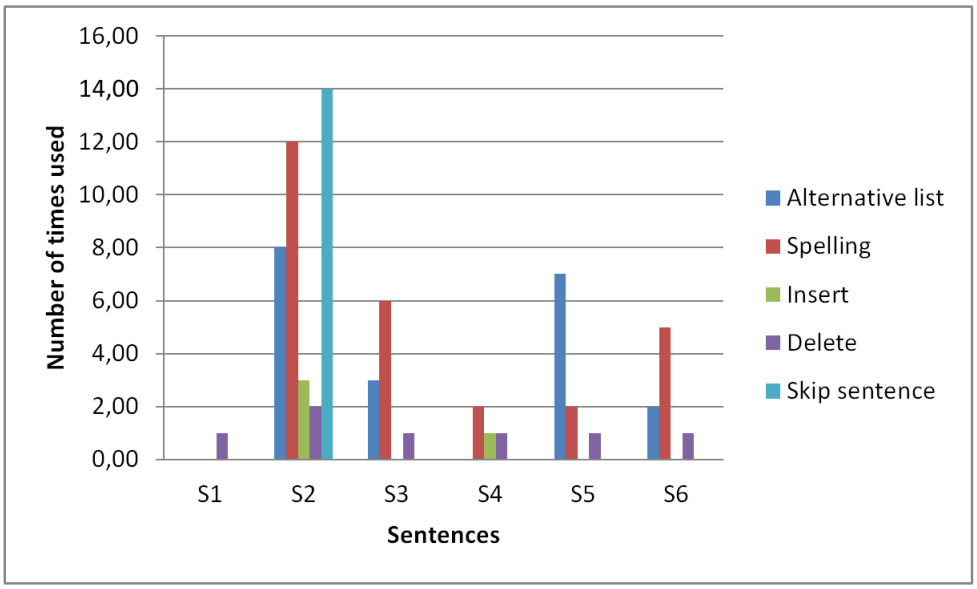


Figure 5.5: Scroller prototype correction methods used

The alternative list was used in sentences S2, S3, S5 and S6. It was unused in sentences S1 and S4 because little errors occurred in the speech recognition hypothesis. Spelling was used in all sentences but S1. Overall spelling was used more than the alternative list with the exception of sentences S1 and S5. The insert feature was only used in sentences S2 and S4. This tells us that the speech recognition hypothesis contained too little words only three times. In all sentences except S2 one delete occurred and in sentence S2 two deletes occurred. This tells us that one or two participants at most had a speech recognition hypotheses with an excess word. The skip sentence feature was only used in sentence S2. Sentence S2 was the sentence that contained the most amount of errors which can be seen in Table 5.4. Although this feature was used 14 times it was all used by participant P3. Participant P3 found the recognition hypothesis so bad for that sentence that he re-spoke the sentence 14 times. On the 14th attempt he was satisfied and corrected the hypothesis.

Overall we notice that the speech recognition hypothesis contained the right amount of words because most modifications were done using spelling or the alternative list.

Discussion

All participants used the prototype successfully. All participants were able to produce the asked sentences without remaining errors. We noticed that using the skip sentence and re-speaking the sentence did not change the resulting hypothesis much. The same speech recognition errors kept re-occurring. This goes against our principle of avoiding cascading errors.

5.2.3 Scroller Auto

Words per minute

The text entry speed per sentence for each user are displayed in Figure 5.6. The mean speeds per sentence are displayed in Table 5.5.

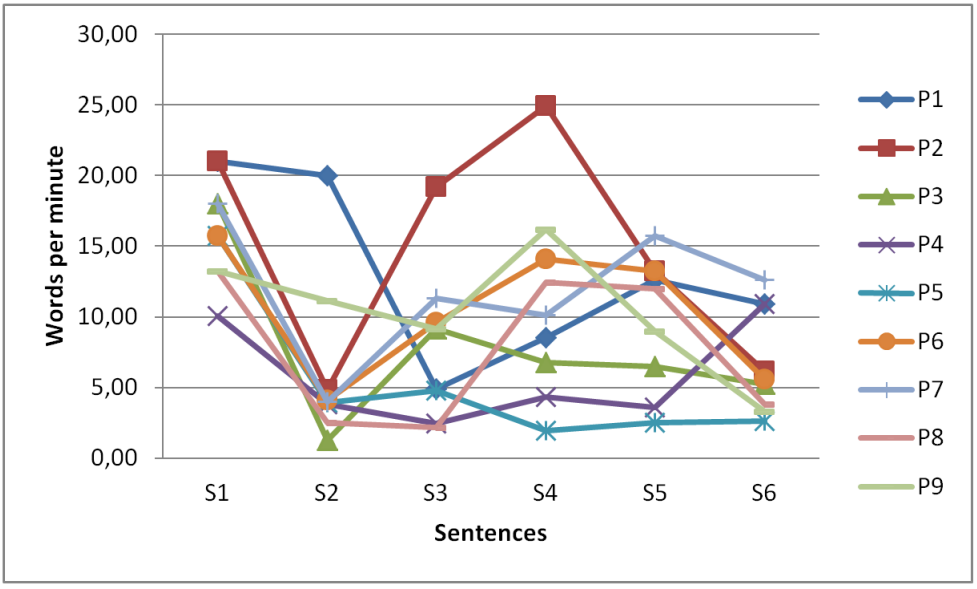


Figure 5.6: ScrollerAuto text entry speed per sentence for each user

S1	S2	S3	S4	S5	S6
16,23	6,17	8,08	11,04	9,83	6,80

Table 5.5: Mean WPM for each sentence using ScrollerAuto

All sentences were entered in between 1,24 and 24,92 WPM. We notice that sentence S2 has the slowest mean entry rate and sentence S1 the highest. Again participant P2 achieved a better speed for sentence S2 than the others. This is exactly what we noticed in the results from the previous prototype.

For each sentence the mean entry speed is lower than in the Scroller prototype.

Words error rate

The word error rate per sentence for each user are displayed in Figure 5.7. The mean WER per sentence is displayed in Table 5.6.

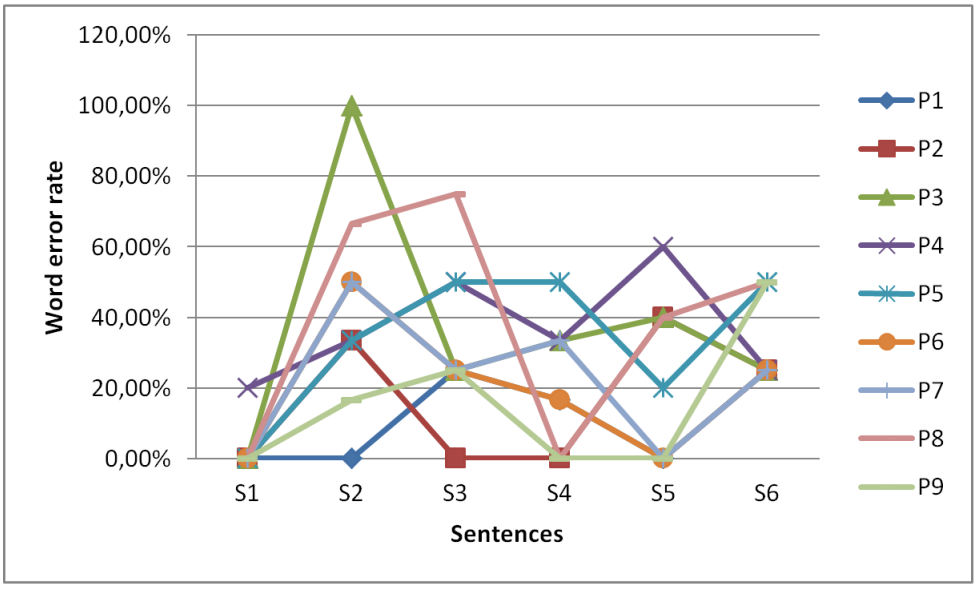


Figure 5.7: ScrollerAuto prototype word error rate per sentence for each user

S1	S2	S3	S4	S5	S6
2,22%	42,59%	33,33%	20,37%	22,22%	33,33%

Table 5.6: Mean WER for each sentence using ScrollerAuto

The lowest mean WER occurs for sentence S1. In this sentence participant P4 had an error while the others did not. It resulted in him having the slowest entry speed for that sentence. Sentence S2 contains the most errors, just like in all previous WER figures. Again a strong correlation is noticed between WPM and WER. When more errors occur extra time is needed to correct them, leading to a slower text entry speed.

Correction methods

The used correction methods for the Scroller Auto prototype are displayed in Figure 5.8.

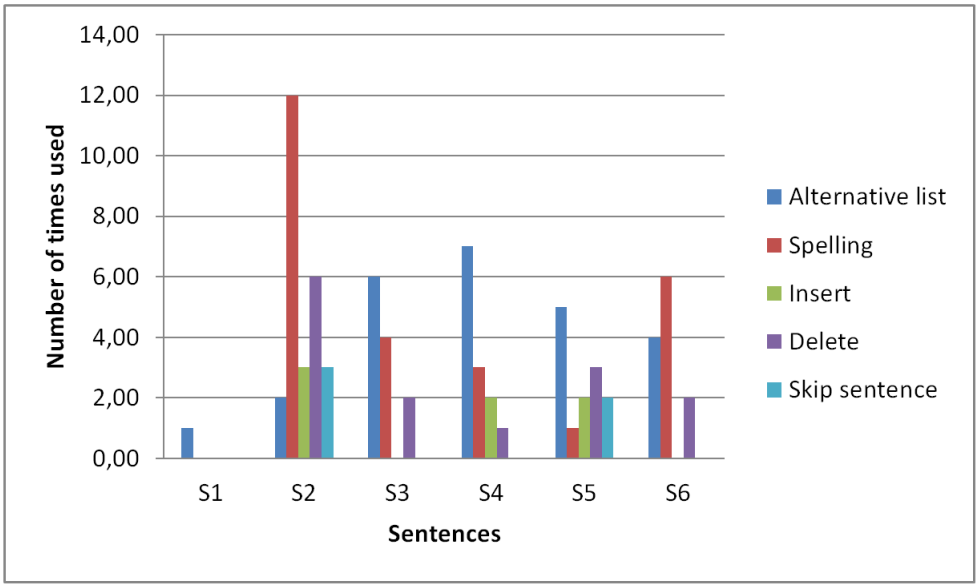


Figure 5.8: Scroller Auto prototype correction methods used

The alternative list was used in all sentences. In sentence S1 it was only used once but there was only one error. Spelling was used in all sentences but S1. Overall the alternative list was used more than spelling with the exception of sentences S2 and S6. The insert feature was used in sentences S2, S4 and S5. Deleting a word was done in all sentences except S1. The skip sentence feature was only used in sentence S2 and S5. The relation between these two sentences is that they both contain five words. This feature was used 5 times and 4 of these were done by participant P3. The remaining one was used by participant P4.

Overall we notice that the speech recognition hypothesis contained the right amount of words because most modification were done using spelling or the alternative list.

Discussion

All participants used the prototype successfully. A common mistake was accidental selection of a word. The participants did not know all the time which column was active in the grid. All participants were able to produce

the asked sentences without remaining errors. We noticed that using the skip sentence and re-speaking the sentence did not change the resulting hypothesis much.

5.2.4 Typewriter

Words per minute

The text entry speed per sentence for each user are displayed in Figure 5.9. The mean speeds per sentence are displayed in Table 5.7.

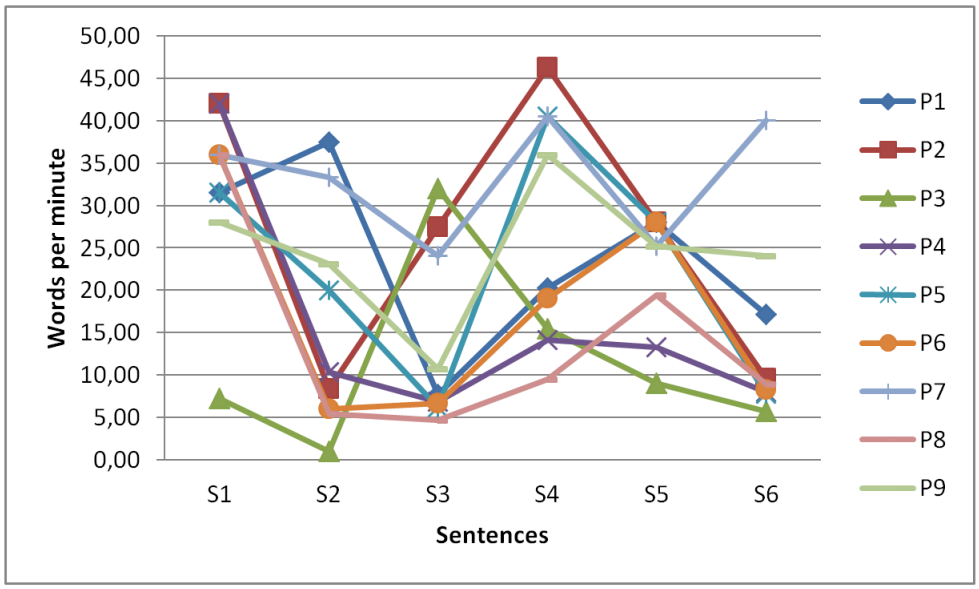


Figure 5.9: Typewriter text entry speed per sentence for each user

S1	S2	S3	S4	S5	S6
32,24	16,11	14,01	26,85	22,67	14,37

Table 5.7: Mean WPM for each sentence using Typewriter

All sentences were entered in between 0,98 and 46,29 WPM. We notice that sentence S2 has the slowest mean entry rate and sentence S1 the highest. Again participant P2 achieved a better speed for sentence S2 than the others just like in the results from the previous prototypes. For each sentence the mean entry speed is much higher than in previous prototypes. This leads us to believe that if the same amount of errors occur, they are corrected more efficiently in this prototype.

Words error rate

The word error rate per sentence for each user are displayed in Figure 5.10. The mean WER per sentence is displayed in Table 5.8.

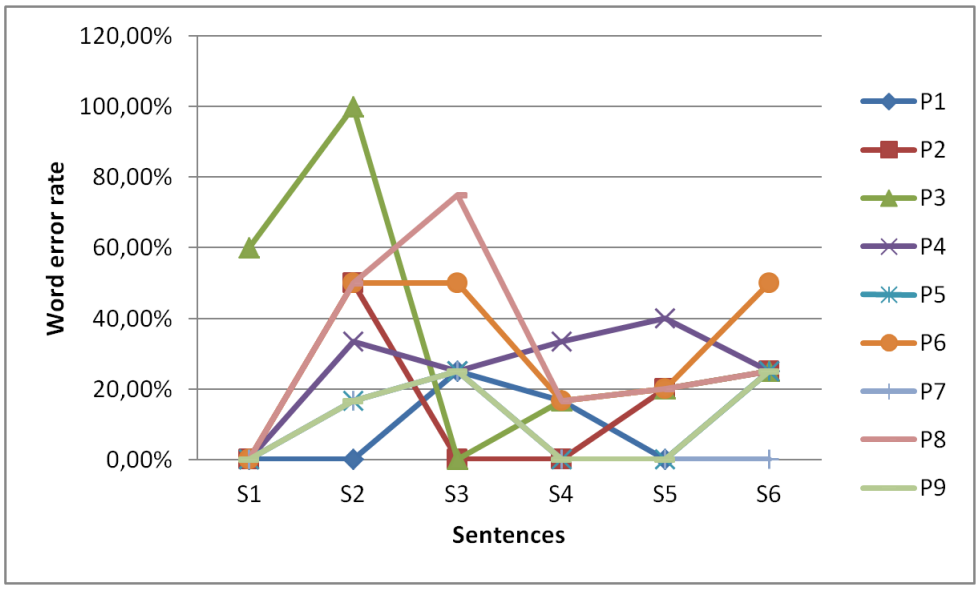


Figure 5.10: Typewriter prototype word error rate per sentence for each user

S1	S2	S3	S4	S5	S6
6,67%	37,04%	27,78%	11,11%	13,33%	25,00%

Table 5.8: Mean WER for each sentence using Typewriter

The lowest mean WER occurs for S1. In this sentence participant P3 had a WER of 60% while all others had a 0% WER. It resulted in him having the slowest entry speed by far for that sentence. Sentence S2 again contains the most errors, just like in all previous WER figures. Again a strong correlation is noticed between WPM and WER. In this prototype we see that entry speeds can still be reasonably high when some errors occur. The tight correlation between WER and WPM seems to fade a little, leading to believe that errors can be corrected more efficiently.

Correction methods

The used correction methods for the Typewriter prototype are displayed in Figure 5.11.

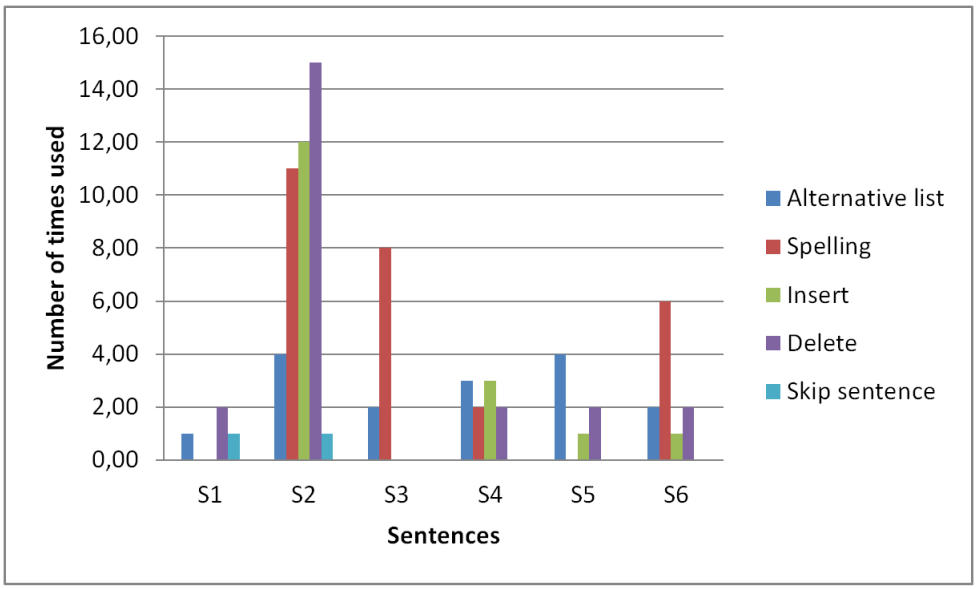


Figure 5.11: Typewriter prototype correction methods used

The alternative list was used in all sentences. Spelling was used in all sentences except S1 and S5. Overall spelling was used more than the alternative list with the exception of sentences S1 and S5. The insert feature was used in sentences S2, S4, S5 and S6. Deleting a word was done in all sentences. The skip sentence feature was only used in sentence S2 and S5. This feature was used only 2 times and both by participant P3.

Overall we notice that the speech recognition hypothesis contained the right amount of words because most modification were done using spelling or the alternative list. Participant P3 used a strange strategy to correct errors in sentence S2. Even though the correct amount of words were recognized he used the insert feature to re-speak a wrong word and insert it. These inserted words were wrongly recognizer some times so he had to use the spelling mode to correct them. After this he had to delete the remaining words. This was done more times than there are words in the sentence. He used the insert feature 11 times, the spelling feature 3 times and the delete feature 14 times for sentence S2 which contains 6 words.

Discussion

All participants used the prototype successfully. A limitation of the Typewriter prototype is that unrecoverable errors can be made. In the training

phase most participants made the error of accidentally triggering the prototype to skip words that were not corrected yet. However after the training phase all participants were able to produce the asked sentences without remaining errors.

We saw that participant P3 used a strategy that is far from optimal to correct sentence S2. We feel that this was due to the sentence being recognized completely wrong. Because of this the participant was unsure how to correct the sentence and experimented with this strategy. His strategy was not efficient because the slowest time out of all prototypes (0,98 WPM) was recorded for him correcting this sentence.

5.2.5 Typewriter Drag

Words per minute

The text entry speed per sentence for each user are displayed in Figure 5.12. The mean speeds per sentence are displayed in Table 5.9.

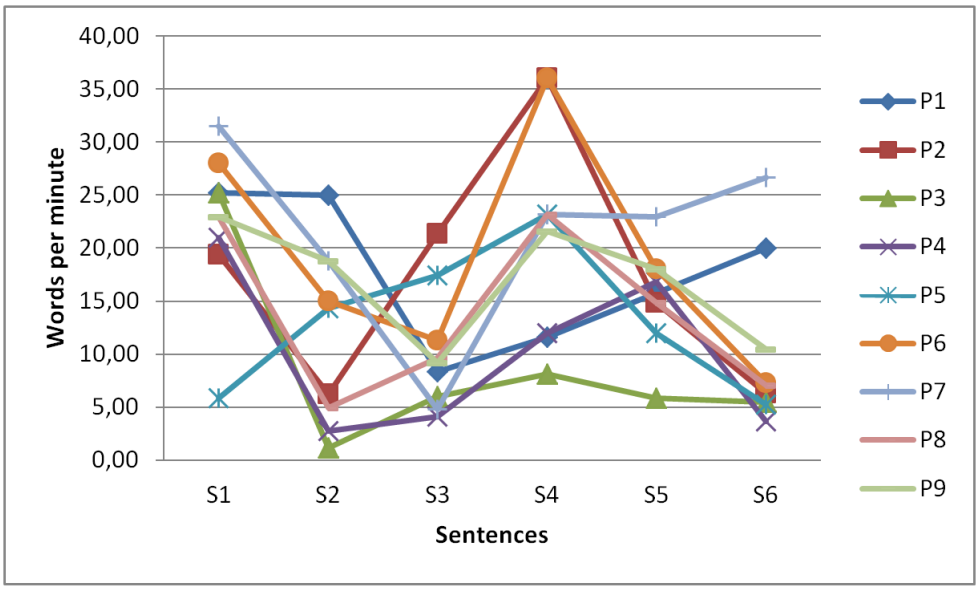


Figure 5.12: TypewriterDrag text entry speed per sentence for each user

S1	S2	S3	S4	S5	S6
22,44	11,87	10,23	21,63	15,44	10,23

Table 5.9: Mean WPM for each sentence using TypewriterDrag

All sentences were entered in between 1,17 and 36 WPM. Sentence S2 no longer has the slowest mean entry rate. This time it took the participants the most time to enter sentences S3 and S6. Again participant P2 achieved a better speed for sentence S2 than the others just like in the results from the previous prototypes. For all bot one sentence the mean entry speed is higher than both Scroller and ScrollerAuto prototypes, but lower than the Typewriter prototype.

Words error rate

The word error rate per sentence for each user are displayed in Figure 5.10. The mean WER per sentence is displayed in Table 5.10.

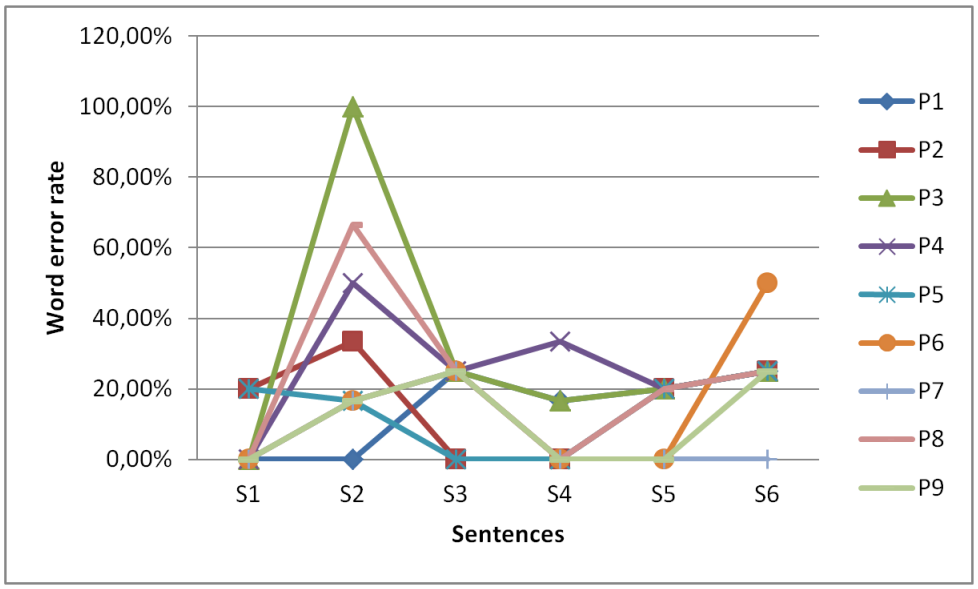


Figure 5.13: TypewriterDrag prototype word error rate per sentence for each user

S1	S2	S3	S4	S5	S6
4,44%	35,19%	19,44%	7,41%	13,33%	25,00%

Table 5.10: Mean WER for each sentence using TypewriterDrag

The lowest mean WER occurs for S1 and S4. Because of this they both have a faster mean entry speed. In sentence S2 participant P3 had a WER of 100% which is relfected in his low entry speed of 1,17 WPM. Sentence S2

again contains the most errors, just like in all previous WER figures. Again a strong correlation is noticed between WPM and WER.

Correction methods

The used correction methods for the Typewriter Drag prototype are displayed in Figure 5.14.

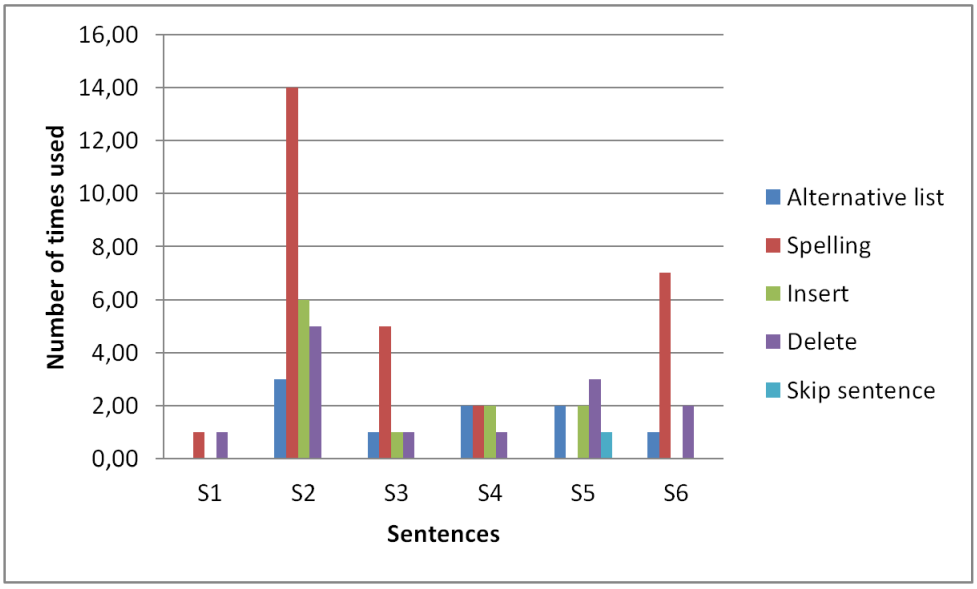


Figure 5.14: Typewriter Drag prototype correction methods used

The alternative list was used in all sentences except sentence S1 because of the limited amount of errors that occurred in that sentence (2 win total). Spelling was used in all sentences except S5. Overall spelling was used more than the alternative list. The insert feature was used in sentences S2, S3, S4 and S5. Deleting a word was done in all sentences. The skip sentence feature was only used once, in sentence S5 again by participant P3.

Overall we notice that the speech recognition hypothesis contained the right amount of words because most modification were done using spelling. We expected to see more participants use the alternative list like in previous prototypes. This is not a usability issue of the prototype. The alternative list just did not contain the correct alternative hypothesis.

Discussion

All participants used the prototype successfully. With the Typewriter Drag prototype all errors can be corrected by dragging back or forth. However dragging was accidentally triggered by some participants causing a slower entry speed.

5.2.6 Overview

Words per minute

The mean performance for each prototype per sentence is highlighted in Figure 5.15 and Table 5.11.

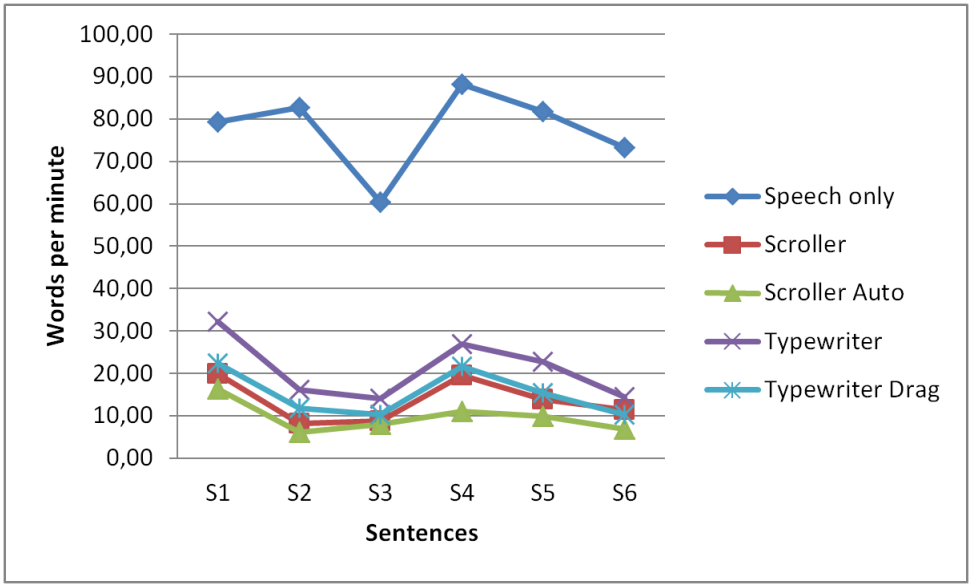


Figure 5.15: The mean WPM for each sentence and prototype

Speech only	Scroller	Scroller Auto	Typewriter	Typewriter Drag
77,63 WPM	13,59 WPM	9,69 WPM	21,04 WPM	15,31 WPM

Table 5.11: Average entry speed for each prototype

In the speech only test the greatest mean WPM was reached. This is logical because no corrections were made to the test. Thus the WER produced from this test is the WER after correction. When using the prototypes all participants corrected the sentences successfully so a WER after correction

of 0% was reached for all participants using any prototype.

The Typewriter prototype was the fastest out of the four prototypes to enter text with a mean text entry speed of 21,04 WPM. It was also the fastest for most participants individually, which can be seen in Chapter C. The second fastest prototype was the Typewriter Drag prototype with a mean entry speed of 15,31 WPM. After that came the Scroller (13,59 WPM) and Scroller Auto (9,69 WPM) was the slowest to enter text with. This trend is also noticeable for each person individually, although the biggest impact on speed was the WER for a specific sentence.

All SpeeG v2 prototypes show greater performance than the SpeeG v1 prototype (5,18 WPM). None of the prototypes has a mean entry speed that reaches the mean speed from Speech Dasher which is 40 WPM [Vertanen and MacKay, 2010]. However the non-native user in the Speech Dasher user study had a mean entry speed of 23 WPM [Vertanen and MacKay, 2010]. In Parakeet a expert user reached 24,43 WPM. However he used a gender specific acoustic model and was a native user, so he must have had a lower WER, which impacts performance.

Words error rate

The mean WER for each prototype per sentence is highlighted in Figure 5.16 and Table 5.12.

Speech only	Scroller	Scroller Auto	Typewriter	Typewriter Drag
17,72 %	20,43 %	25,68 %	20,15 %	17,47 %

Table 5.12: Mean WER for each prototype

The WER before correction was highest while using the Scroller Auto prototype (25,68%) and lowest using the Typewriter Drag prototype (17,47%). The difference between them is comparable to one word being wrongly recognized in a sentence of twelve words. However the amount of errors has a lot of impact. In sentence S4 the high WER for the Scroller Auto prototype made a diffence in Figure 5.16 compared to the other prototypes.

Speech recognition accuracy varied from person to person and greatly depends on the accent and pronunciation of the user. Participant P3 suffered from bad accuracy frequently reaching a WER of 100% for sentence S2. However the mean WER before correction for all prototypes and participants was 20,29%. This is comparable to one error occurring in a five word

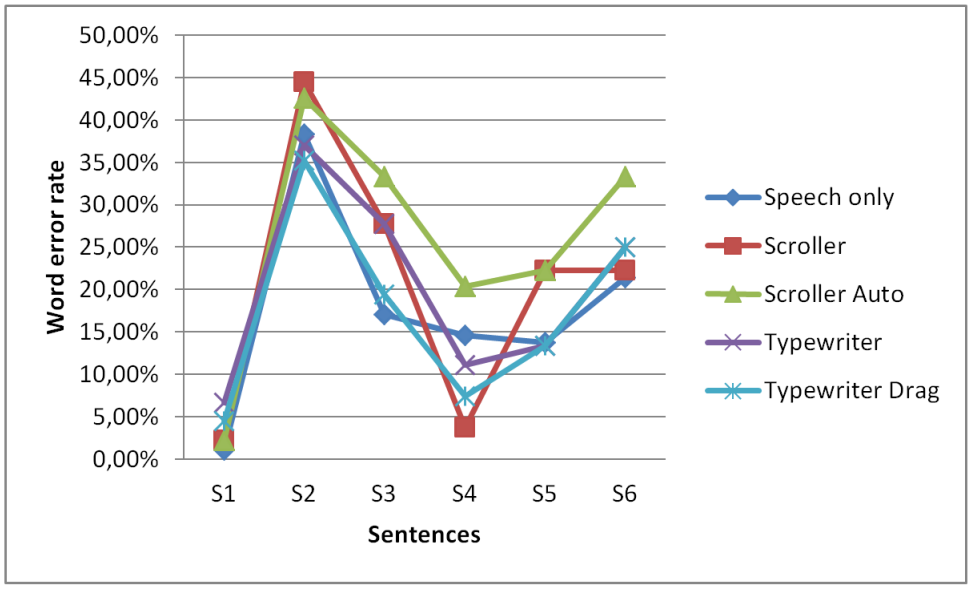


Figure 5.16: The mean WER for each sentence and prototype

sentence. Considering all participants were non-native English speakers this is a normal result.

In speech Dasher the mean WER before correction was 22%, which is similar to our result. In their study the two native participants had a WER of 7,8% and 12,4%. The non-native participant had a WER of 46,7%. Considering that in our study everyone was a non-native participant, our speech recognizer was more accurate.

In Parakeet the mean WER before correction was 8,46%. We have to note that in the Parakeet study only one expert participant native speaker was recorded. The acoustic model used was also trained to be gender specific, adding to the accuracy. Because of the difference in models it is hard to compare their results to ours.

It is also not possible to compare our WER before correction to that of SpeeG v1 because they counted and computed the amount of errors differently which can be seen in Figure 3.8.

Correction methods

The mean number of used correction methods are displayed in Figure 5.17.

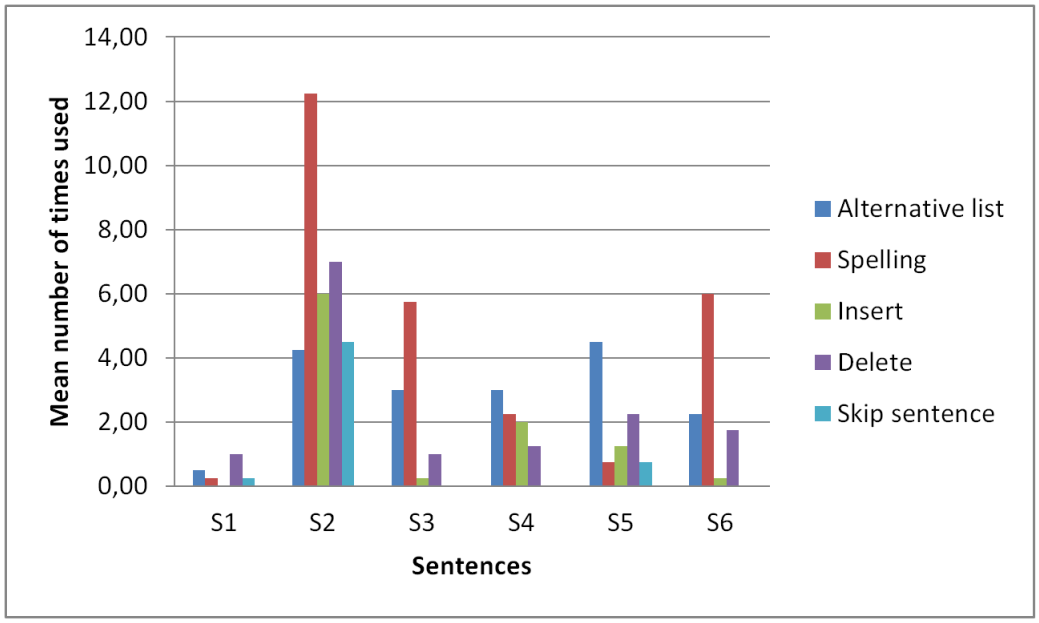


Figure 5.17: Mean number of correction methods used

We see that most corrections were made on sentence S2. This sentence also contained the most errors. Sentence S1 was the best recognized and thus little corrections were needed. The most used correction methods were the alternative list and spelling.

The skip sentence was almost exclusively used by participant P3. The reason for this was because recognition accuracy was worst for him. Therefore he used the skip sentence and re-spoke in the hope a better result would arise. However the same errors kept re-occurring.

Discussion

We notice that in general spelling was used more than the alternative list. We saw that it took participants more time in spelling mode to correct a word than selecting an alternative. Increasing the number of alternatives could potentially speed up the selection process if the correct alternative is included.

All participants used the prototypes successfully and no participant had remaining errors in the processed sentences. Thus a final WER of 0% is noted for all prototytpes.

5.2.7 Questionnaire

Each participant was asked to fill in a questionnaire about using the prototypes. The most important results of this qualitative study are described here. All results can be viewed in Chapter B. The author of this dissertation, participant (P1) did not fill in the questionnaire to avoid biased results.

Usage of speech recognition interfaces

Participants were asked if they used Speech recognition regularly. About 63% of the participant have never used speech recognition and nobody uses it on a frequent basis. Those of them who did use it have only limited experience. Participants P2 and P3 have pariticipated in the evaluation study from SpeeG v1.

Do you feel comfortable with speech recognition

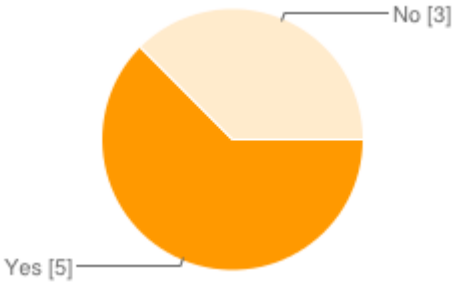


Figure 5.18: Result for the question: Do you feel comfortable with speech recognition

When the participants were asked if they felt comfortable using speech recognition 38% answered No. Although the majority felt comfortable using speech recognition this result was not expected. Speech is the primary form of communication, so one would think everyone to find it natural to talk to a machine. Participants said that the uncertainty that a sentence would be recognized correctly made them feel uncomfortable.

Quality of the speech recognizer

Participants were asked to rate the speech recognition on a scale from one to six. One being very bad and six being very good. Everyone found the speech recognition to be good, but nobody found it very good.

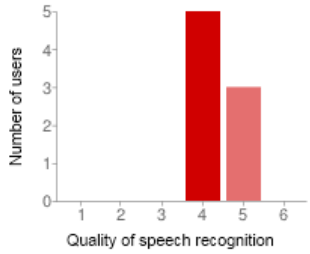


Figure 5.19: Result for the question: Quality of the speech recognizer

I experienced physical strain after the evaluation

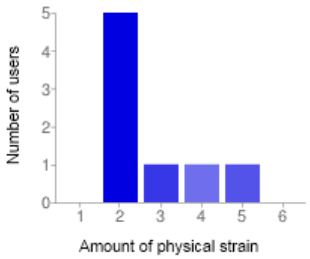


Figure 5.20: Result for the question: I experienced physical strain after the evaluation

Participants were asked if they experienced physical strain after the user study. Two participants said they felt some strain. The others did not experience physical strain.

Which interface did you find the easiest to learn

Half of the participants found the Scroller prototype to be the easiest to learn.

Which interface did you find the easiest to use (after the learning phase)

Six participants found the Typewriter prototype to be the easiest to use. The remaining two found the Typewriter Drag prototype the easiest to use.

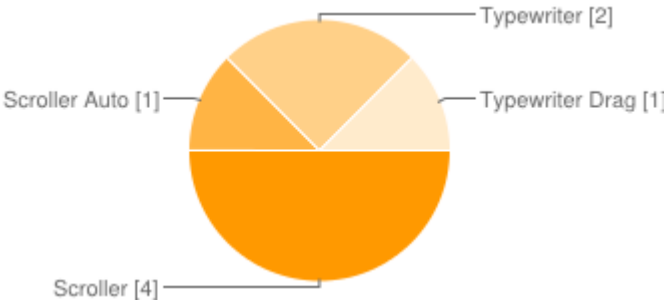


Figure 5.21: Result for the question: Which interface did you find the easiest to learn

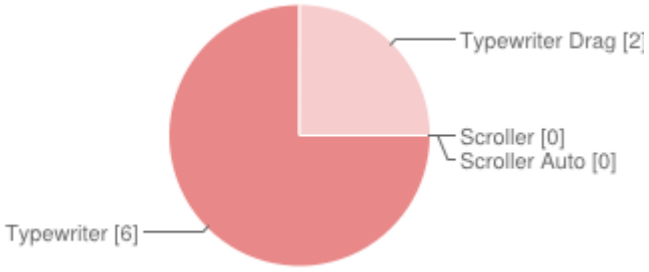


Figure 5.22: Result for the question: Which interface did you find the easiest to use (after the learning phase)

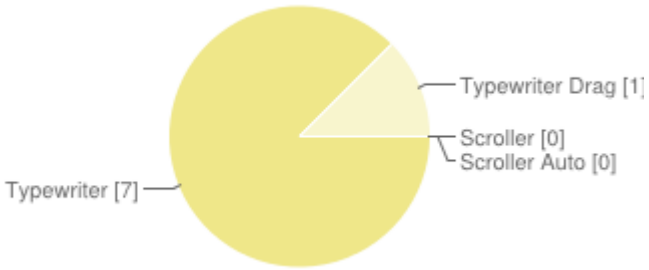


Figure 5.23: Result for the question: Which interface did you find was the quickest to enter text

Which interface did you find was the quickest to enter text

The Typewriter prototype was considered to be the quickest prototype to enter text. Only one participant disagreed and found the Typewriter Drag faster. This participant (P6) was the only user that was in fact faster using the Typewriter Drag prototype which can be seen in Figures 5.9 and 5.12.

Which interface did you prefer

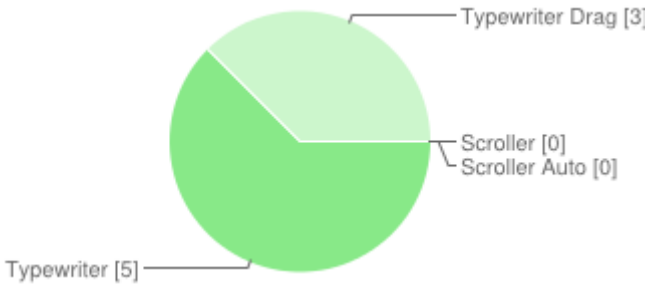


Figure 5.24: Result for the question: Which interface did you prefer

Five participants prefer using the Typewriter prototype. The remaining three prefer the Typewriter Drag prototype.

I find it important to improve the following features of SpeeG

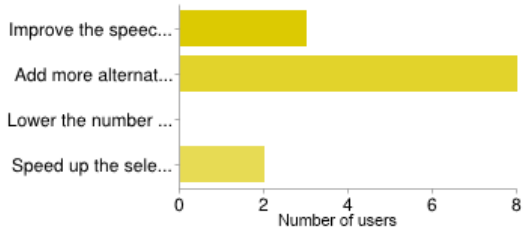


Figure 5.25: Result for the question: I find it important to improve the following features of SpeeG

All participants suggest to add more alternative word choices to the prototypes. Three participants want to improve speech recognition.

5.3 Conclusion

To investigate text input speed for the prototypes a quantitative and qualitative user study was conducted. This section is the conclusion of what was

noticed in the results of those studies.

More errors implies more corrections needed
In each test that was conducted a clear correlation between WPM and WER was noticed. When the speech recognizer makes more errors it is harder to correct them, and this is confirmed in the test results for each prototype. We observed that speech recognition accuracy had the biggest impact on performance. Participant P3 had worse speech recognition results than other participants and he therefore he had worse performance results. He was also the only participant that made frequent use of the skip sentence feature when the recognition result did not please him. Because of a WER of 100% he experimented with correction methods that were not optimal in an attempt to get better recognition results. The correlation between performace and errors was also observed in Speech Dasher [Vertanen and MacKay, 2010] and Parakeet [Vertanen and Kristensson, 2009].

Increased speed and accuracy in speech recognition
Comparing the speech recognizers from SpeeG v1 to the one we used we observe both a rise in accuracy and speed. Input speed has increased from a mean value of 10,96 WPM in the speech only test from SpeeG v1 to a mean value of 77,63 WPM in our speech only test. The big difference is largely because a sentence is uttered completely and recognized as a whole whereas before users only uttered one word at a time. This makes a difference in speed and interaction. The interaction becomes more natural because speaking at a higher speed is more like talking in a conversation. The continuous nature of speech and sentence-level recognition have had a positive effect on performance.

For examining if speech recognition accuracy has improved we look at the mean number of errors in the speech only test from SpeeG v1 highlighted in Figure 3.8. The same sentences were used. A good comparison with this figure can not be made because the errors are computed in a very different way. In SpeeG v1 each word was uttered separately and every wrong recognition was noted as a error. In SpeeG v2 a sentence was repeated if it contained one or more errors and the mean amount of errors was recorded. Even thought we are unable to compare the numbers participants P2 and P3 who participated in the study of SpeeG v1 explicitly said that the speech recognition accuracy had improved.

In our quantitative user study the mean WER before correction from all

tests was 20,29%. Speech Dasher featured a mean WER before correction of 22% [Vertanen and MacKay, 2010]. This result is very good considering two out of three participants were native speakers in the Speech Dasher study and we had all non-native speakers. The difference is greater if we consider only the non-native user in the Speech Dasher study. He had a WER before correction of 46,7% which is very high compared to our mean WER before correction of 20,29%. The highest mean WER before correction for one specific participant(P3) was 32,19%, wich is also significantly lower than the non-native user from the Speech Dasher study. We can conclude that speech recognition was more accurate for non-native users than in related work.

Typewriter was the fastest

The Typewriter prototype was the fastest out of the four prototypes to enter text with a mean text entry speed of 21,04 WPM. The mean speed recorded for SpeeG v1 was 5,18 WPM [Hoste et al., 2012]. It was also the fastest for most participant individually which can be seen in Chapter C. The fastest speed for entering a sentence was recorded by participant P2 entering sentence S4 at a rate of 46,29 WPM. The second fastest prototype was the Typewriter Drag prototype with a mean entry speed of 15,31 WPM. After that came the Scroller (13,59 WPM) and Scroller Auto (9,69 WPM) was the slowest to enter text with. This trend is also noticeable for each person individually which can be seen in Chapter C. We expected the Typewriter prototype to be the fastest because it needs less interaction than all others to process a sentence. For instance in Typewriter Drag you need to drag in order to process a word sequence. This dragging motion is not needed in Typewriter which saves the user some time. It is these kind of gestures that positively influence input speed.

All SpeeG v2 prototypes show greater performance than the SpeeG v1 prototype. All of the prototypes outperform the Controler (8,38 WPM) and Kinect only (1,83 WPM) input methods that were tested to compare with SpeeG v1 [Hoste et al., 2012]. The mean values for Typewriter (21,04 WPM), Typewriter Drag (15,31 WPM) and Scroller (13,59 WPM) are even better than the Speech only test (10,96 WPM) from SpeeG v1 [Hoste et al., 2012].

None of the SpeeG v2 prototypes prototypes has a mean entry speed that reaches the mean value from Speech Dasher which is 40 WPM.[Vertanen and MacKay, 2010]. However the non-native user in the Speech Dasher user study had a mean entry speed of 23 WPM [Vertanen and MacKay,

2010]. This is only just more than the mean entry speed for the Typewriter prototype (21,04 WPM). The potential for performance is present because the non-native user in the Speech Dasher study had a maximum speed of 40 WPM and participant P2 reached 46,29 WPM on sentence S4 using the Typewriter prototype.

The Typewriter prototype can be faster than Parakeet. In the expert study in Parakeet the mean entry rate was 24,43 WPM [Vertanen and Kristensson, 2009]. The participant had a mean WER before correction of 8,46%. In this study a native English speaking user that designed the user interface was tested. The fastest mean value for a prototype was recorded for participant P7 using the Typewriter prototype. He produced all sentences with a rate of 33,17 WPM and a WER before correction of 6,94%. He was not familiar with the user interface and is a non-native English speaker.

Participants preferred Typewriter and Typewriter Drag
Most participants preferred using the Typewriter prototype over the others (5.24). The remaining participants chose the Typewriter Drag prototype to be their favorite. Participant (P6) chose it because in his case he entered text faster with that prototype. The reason for that was because the WER was significantly lower (15,28% and 31,11%) when he was using the Typewriter Drag prototype. A general trend is that participants prefer the prototype that is fastest to enter text. Participants noted that the Typewriter and Typewriter Drag prototypes were much more intuitive because they did not require waiting for the interface to move to the next word. More control is given to the user which resulted in a increased input speed.

Limited physical strain
Most participants did not experience physical strain 5.20 with the SpeeG v2 prototypes even though they tested the four prototypes in one sequence. Physical strain was observed in SpeeG v1 which is why it was of great importance in SpeeG v2. In SpeeG v2 physical strain is kept at a minimum.

Common errors
Common mistakes for participants using the Scroller and Scroller Auto prototypes was accidental selection of a word. This mistake was caused by two things. The imprecise input from the skeletal tracking was sometimes the cause. However most of all the participants were unaware when the interface moved to the next word and tried to see if it did unsuccessfully.

Overall we noticed little cascading errors because the spelling mode offered speech recognition but with a limited vocabulary. In spelling mode cascading errors could occur and this was largely aided with the natural language feature from the speech recognizer. Participant P3 overall had the highest WER and was more prone to cascading errors than other participants.

Visualization and Continuous use

The user interface was clear to the participants. All participants had no errors remaining in their final text resulting in a WER after correction of 0% for each participant and each prototype. In Speech Dasher [Vertanen and MacKay, 2010] the WER after correction was 1,3% for participants using Dasher and 1,8% using Speech Dasher. Parakeet [Vertanen and Kristensson, 2009] showed a WER after correction of 0,94%.

6 Conclusion

SpeeG v1 [Hoste et al., 2012] was a first attempt to combine speech recognition and skeletal tracking in a text input system. However the evaluation demonstrated a lot of points of improvement. SpeeG v2 features four new prototypes that are designed to visualize hypothesis space in a clean and organized way to decrease training time and increase performance.

6.1 Conclusion

We investigated speech recognition and related text entry systems that use speech recognition. From important aspects and observations in previous work we have designed four prototypes: Scroller, Scroller Auto, Typewriter and Typewriter Drag. We have conducted a quantitative and qualitative user study to test both performance and usability.

Speech recognition accuracy was better than related work and much better if we consider only non-native users. Our mean WER before correction was 20,29%. In Speech Dasher [Vertanen and MacKay, 2010] the mean WER before correction was 22%. The non-native participant had a WER before correction of 46,7%.

The best mean performance was recorded for the Typewriter prototype with a mean text entry speed of 21,04 WPM. The Typewriter Drag had a mean

entry speed of 15,31 WPM. After that came the Scroller (13,59 WPM) and Scroller Auto (9,69 WPM) was the slowest to enter text with.

All of our prototypes outperformed all text input methods compared with SpeeG v1 in [Hoste et al., 2012]. The SpeeG v1 prototype had a mean entry speed of 5,18 WPM. The main reason for improved performance is that speech recognition performance was increased. In our speech only test text was produced at a rate of 77,63 WPM. In a similar test in [Hoste et al., 2012] speech was produced at 10,96 WPM. This increase in performance is due to the use of sentence-level recognition.

The fastest speed recorded for entering a sentence was with the Typewriter prototype at a rate of 46,29 WPM. The fastest speed for entering all six sentences was produced by a different participant at a mean rate of 33,17 WPM.

The WER before correction varied from sentence to sentence and the performance depended largely on the WER. The WER after correction was 0% for each prototype. This was unexpected because the Typewriter prototype is prone to errors. This is also an improvement over SpeeG v1 because it still had remaining errors after correction.

We observed that spelling was the most accessed correction method. This means that most of the times the alternative list did not contain a correct hypothesis. We would have liked to have seen the alternative list used more because it can be beneficial to performance because more time was wasted using spelling. The insert, delete and skip sentence correction methods were used fewer times.

From our qualitative user study we concluded that most participants preferred using the Typewriter prototype. The remaining participants preferred the Typewriter Drag prototype. A general trend was that participants preferred the prototype that was fastest to enter text. Participants noted that the Typewriter and Typewriter Drag prototypes were much more intuitive.

The participants experience very little physical strain with the SpeeG v2 prototypes even though they tested four prototypes in one sequence. Physical strain was observed when the SpeeG v1 prototype was evaluated in [Hoste et al., 2012].

6.2 Contributions

Contribution 1: Continuous text entry systems using speech
After researching speech recognition and input systems that use speech recognition we proposed a series of prototypes. These share characteristics with an existing input system that uses a touch screen to correct speech recognition errors. From these characteristics we designed and implemented four prototypes and evaluated their performance in a user study.

Contribution 2: Investigated accuracy of Microsoft Speech 8.0
We proposed using Microsoft Speech 8.0 as a speech recognizer instead of Sphinx 4 that was previously used by SpeeG v1. The speech recognizer was tested and found to be more accurate according to two participants who participated in both studies. It was also more accurate compared to related work. The speed at which speech input was entered was also improved on because sentences were spoken in one utterance compared to uttering each word separately.

Contribution 3: Investigated efficiency of prototypes
We evaluated the proposed prototypes and they all showed greater input speed compared to the SpeeG v1 prototype. They are thus more efficient to enter text using the same modalities. The fastest was the Typewriter prototype.

Contribution 4: Investigated usability of prototypes
We evaluated the usability of the proposed prototypes and found that the Typewriter and Typewriter Drag prototypes were easiest to use by the participants. All participants thus preferred one of these two over the others.

6.3 Future work

Although the SpeeG v2 prototypes were are all faster than the prototype of SpeeG v1, improvements can still be made to increase efficiency and usability.

6.3.1 Technical improvements

A lot of users had to spell their word when only one letter was wrong. Participants felt that more alternative word choices could improve speed and avoid spelling mode for small errors. As more alternative words are put on the screen, the hit zone for a button decreases. This could hinder usability

since using the Kinect sensor as input is an imprecise input method. This was also a problem in the Speech Dasher project [Vertanen and MacKay, 2010]. Just like us they chose to only show the most promising results and hide others behind a interacting feature. It might help is to make the button height correspond to the likelihood of that word occurring. This can free up screen size to include more alternative.

Sometimes the speech recognizer does not have alternative hypotheses. In that case a word confusion network or some form of language analysis can be performed on the output to increase alternatives. The downside of this approach is that recognition time will increase which has a negative impact on text entry speed.

A frustration for some users was that when the speech recognizer perfectly recognized their sentence they still needed to process it. Users asked for some sort of gesture to confirm a correctly recognize sentence. A feature like this can greatly improve entry speed and should be looked into in future work.

6.3.2 Further research

In this dissertation we chose to design a different user interface because some parts of the Dasher interface had a negative effect on usability. The biggest one was that no visual feedback was given during speech recognition and queuing speech recognition results was not visualized. However it should be interesting to see how a improved speech recognizer will impact text entry speeds in a Dasher-like interface like in SpeeG v1, provided the aforementioned aspects are addressed.

In the user study we chose to not train the speech recognizer. This helped us see what sentences where more difficult to recognize and provided a clear way to compare test results to related work. However the functionality for training is included in the speech recognizer and it would be interesting to see how training affects recognition accuracy and text input speed of the SpeeG v2 prototypes. We can also observe how well the speech recognizer adapts itself to a non-native accent.

The SpeeG v2 prototypes correct speech recognition results, these corrections can be fed back to the speech recognizer. In future work this should be researched because it provides an automatic training procedure to the users without having to go trough a explicit training procedure.

Bibliography

Chou, W. and Juang, B., editors (2002). *Pattern Recognition in Speech and Language Processing*. CRC Press Inc., Boca Raton, FL, USA.

CMU Sphinx (2011a). Pocketsphinx. `http://cmusphinx.sourceforge.net/wiki/download/`.

CMU Sphinx (2011b). Sphinx 4 Javadoc: Class JSGFGrammar. `http://cmusphinx.sourceforge.net/sphinx4/javadoc/edu/cmu/sphinx/jsgf/JSGFGrammar.html`.

Cohen, P., Johnston, M., McGee, D., Oviatt, S., Clow, J., and Smith, I. (1998). The efficiency of multimodal interaction: A case study.

Forsberg, M. (2003). Why is speech recognition difficult? *Department of Computing Science, Chalmers University of Technology, Gothenburg*.

Gaikwad, S., Gawali, N., and Yannawar, P. (2010). A review on speech recognition technique. *International Journal of Computer Applications*, 10:16–24.

Garofolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallett, D., and Dahlgren, N. (1993). Timit acoustic phonetic continuous speech corpus.

Gold, B. and Morgan, N. (2000). *Speech and audio signal processing: processing and perception of speech and music*. John Wiley.

Hirtle, D. (2004). Speech variability: The biggest hurdle for recognition. *Faculty of Computer Science, University of New Brunswick*.

Hoste, L., Dumas, B., and Signer, B. (2012). SpeeG: A Multimodal Speech- and Gesture-based Text Input Solution. In *Proceedings of AVI 2012, 11th International Working Conference on Advanced Visual Interfaces*, AVI '12, Naples, Italy.

Huang, X., Acero, A., Acero, A., and Hon, H. (2001). *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall PTR.

Huang, X. and Deng, L. (2010). An overview of modern speech recognition. In Indurkhya N., D. F., editor, *Handbook of Natural Language Processing, Second Edition*. CRC Press, Taylor and Francis Group, Boca Raton, FL. ISBN 978-1420085921.

Karat, C., Halverson, C., Horn, D., and Karat, J. (1999). Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, CHI '99, pages 568–575. ACM.

Lee, K., Hon, H., and Hwang, M. (1989). Recent progress in the sphinx speech recognition system. In *Proceedings of the workshop on Speech and Natural Language*, HLT '89, pages 125–130, Stroudsburg, PA, USA. Association for Computational Linguistics.

MacKenzie, I. and Soukoreff, R. (2003). Phrase sets for evaluating text entry techniques. In *Extended abstracts on Human factors in computing systems*, CHI EA '03, pages 754–755, New York, NY, USA. ACM.

Martin, A. (1972). A new keyboard layout. *Appl Ergon*, 3(1):48–51.

Oracle (1998). Grammar format specification. `http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/`.

Ostrach, T. (1997). Typing speed: How fast is average. 4,000 typing scores statistically analyzed and interpreted.

Seltzer, M. (2003). *Microphone Array Processing for Robust Speech Recognition*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University.

Suhm, B., Myers, B., and Waibel, A. (2001). Multimodal error correction for speech user interfaces. *ACM Trans. Comput.-Hum. Interact.*, 8(1):60–98.

Vertanen, K. (2007). English gigaword language model training recipe. `http://www.keithv.com/software/giga/`.

Vertanen, K. (2009). *Efficient Correction Interfaces for Speech Recognition*. PhD thesis, University of Cambridge, Cambridge, UK.

Vertanen, K. and Kristensson, P. (2009). Parakeet: A continuous speech recognition system for mobile touch-screen devices. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, IUI '09, pages 237–246. ACM.

Vertanen, K. and MacKay, D. (2010). Speech dasher: Fast writing using speech and gaze. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 595–598.

Ward, D., Blackwell, A., and MacKay, D. (2000). Dasher - a data entry interface using continuous gestures and language models. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, UIST '00, pages 129–137. ACM.

Yuan, J., Liberman, M., and Cieri, C. (2006). Towards an integrated understanding of speaking rate in conversation. In *Proceedings of Interspeech*, pages 541–544.

A

Questionnaire

This is the questionnaire that participants filled in so that usability and user satisfaction could be measured.

SpeeG Questionnaire

* Required

Usage of speech recognition interfaces *

- I never used speech recognition interfaces
- I use speech recognition interfaces occasionally
- I use speech recognition interfaces often/frequently

Do you feel comfortable with speech recognition *

- Yes
- No

Quality of the speech recogniser *

	1	2	3	4	5	6	
Very bad							Very good

I am ... handed *

- Left
- Right

I like the "left" hand up gesture to switch to spelling *

	1	2	3	4	5	6	
Not agree at all							Completely agree

I like that the "right" hand controls movement on the screen *

	1	2	3	4	5	6	
Not agree at all							Completely agree

I experienced physical strain after the evalutaion *

I had pain in the right arm from holding it up

	1	2	3	4	5	6	
Not agree at all							Completely agree

Which interface did you find the easiest to learn? *

- Scroller
- Scroller Auto
- Typewriter

- Typewriter Drag

Which interface did you find the easiest to use (after the learning phase)? *

- Scroller
- Scroller Auto
- Typewriter
- Typewriter Drag

Which interface did you find was the quickest to enter text? *

- Scroller
- Scroller Auto
- Typewriter
- Typewriter Drag

Which interface did you prefer? *

- Scroller
- Scroller Auto
- Typewriter
- Typewriter Drag

I liked the SpeeG Interface in general (the layout). *

	1	2	3	4	5	6	
Not agree at all	○	○	○	○	○	○	Completely agree

Why did you (not) like it?

Quality of the Scroller interface *

	1	2	3	4	5	6	
Very bad	○	○	○	○	○	○	Very good

I liked the SpeeG Scroller Interface. *

	1	2	3	4	5	6

Not agree at all ○ ○ ○ ○ ○ ○ Completely agree

Why did you (not) like it?

Quality of the Scroller Auto interface *

1 2 3 4 5 6

Very bad ○ ○ ○ ○ ○ ○ Very good

I liked the SpeeG Scroller Aurto Interface. *

1 2 3 4 5 6

Not agree at all ○ ○ ○ ○ ○ ○ Completely agree

Why did you (not) like it?

Quality of the Typewriter interface *

1 2 3 4 5 6

Very bad ○ ○ ○ ○ ○ ○ Very good

I liked the SpeeG Typewriter Interface. *

1 2 3 4 5 6

Not agree at all ○ ○ ○ ○ ○ ○ Completely agree

Why did you (not) like it?

Quality of the Typewriter Drag interface *

	1	2	3	4	5	6	
Very bad	○	○	○	○	○	○	Very good

I liked the SpeeG Typewriter Drag Interface. *

	1	2	3	4	5	6	
Not agree at all	○	○	○	○	○	○	Completely agree

Why did you (not) like it?

I would replace my Wii/Xbox/PS3/Digibox on-screen keyboard with this interface today *

	1	2	3	4	5	6	
Not agree at all	○	○	○	○	○	○	Completely agree

Why would you (not) like to replace your current Wii/Xbox/PS3/Digibox on-screen keyboard with this interface?

I find it important to improve the following features of SpeeG *

- ☐ Improve the speech recognition

- [ ] Add more alternative word choices
- [ ] Lower the number of alternative word choices
- [ ] Speed up the selection (pointing) process

Do you have any other proposals to improve the usability? (controls, interface, performance...)

Submit

B

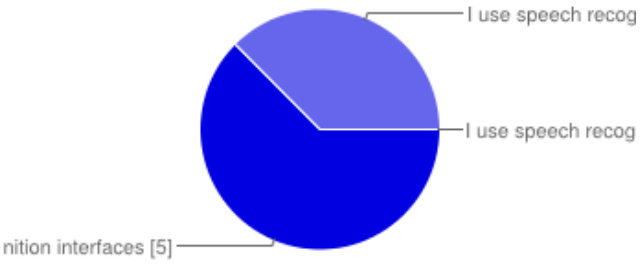
Questionnaire Results

This are the results for each question from the questionnaire that participants filled in to test usability and user satisfaction.

8 responses

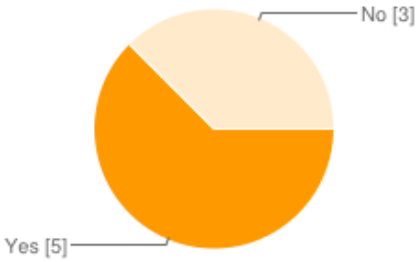
Summary See complete responses

Usage of speech recognition interfaces



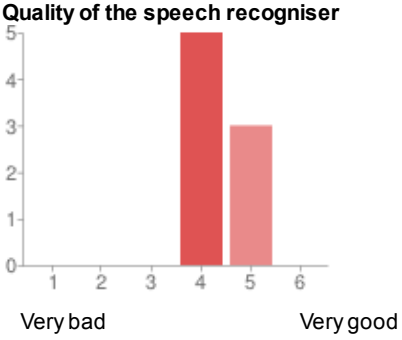
I never used speech recognition interfaces	**5**	63%
I use speech recognition interfaces occasionally	**3**	38%
I use speech recognition interfaces often/frequently	**0**	0%

Do you feel comfortable with speech recognition



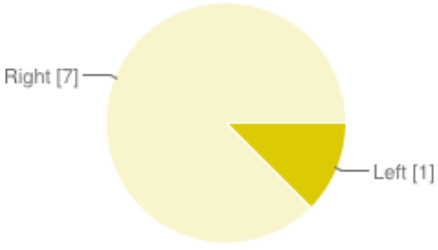
Yes	**5**	63%
No	**3**	38%

Quality of the speech recogniser



1 - Very bad	**0**	0%
2	**0**	0%
3	**0**	0%
4	**5**	63%
5	**3**	38%
6 - Very good	**0**	0%

I am ... handed

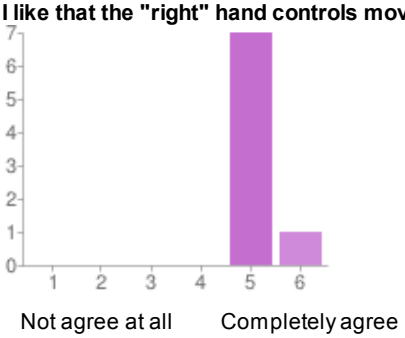


Left	**1**	13%
Right	**7**	88%

I like the "left" hand up gesture to switch to spelling

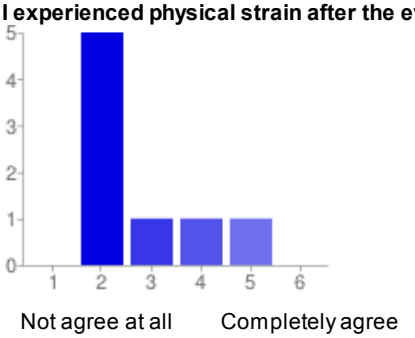
1 -	Not agree at all	**0**	0%
2		**0**	0%
3		**0**	0%
4		**2**	25%
5		**2**	25%
6 -	Completely agree	**4**	50%

I like that the "right" hand controls movement on the screen



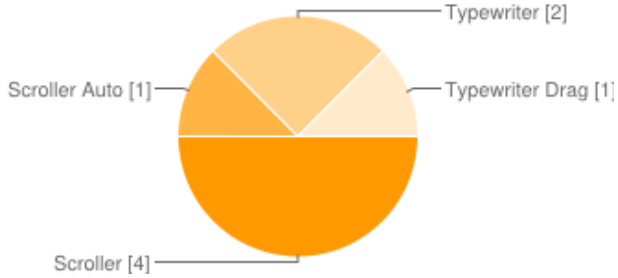
1 -	Not agree at all	**0**	0%
2		**0**	0%
3		**0**	0%
4		**0**	0%
5		**7**	88%
6 -	Completely agree	**1**	13%

I experienced physical strain after the evalutaion



1 -	Not agree at all	**0**	0%
2		**5**	63%
3		**1**	13%
4		**1**	13%
5		**1**	13%
6 -	Completely agree	**0**	0%

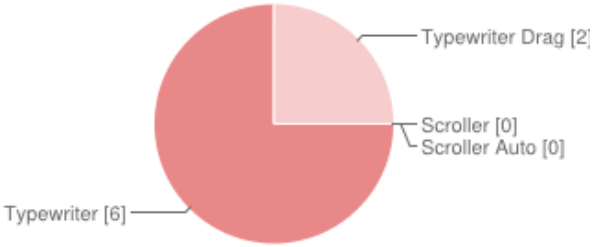
Which interface did you find the easiest to learn?



Scroller		**4**	50%
Scroller Auto		**1**	13%
Typewriter		**2**	25%
Typewriter Drag		**1**	13%

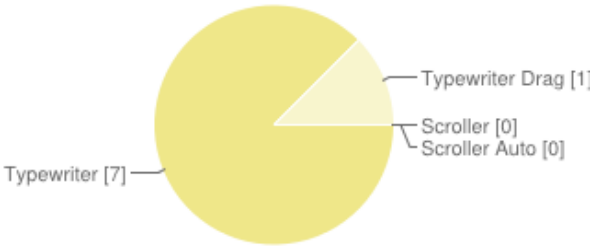
Which interface did you find the easiest to use (after the learning phase)?

Scroller		**0**	0%
Scroller Auto		**0**	0%
Typewriter		**6**	75%



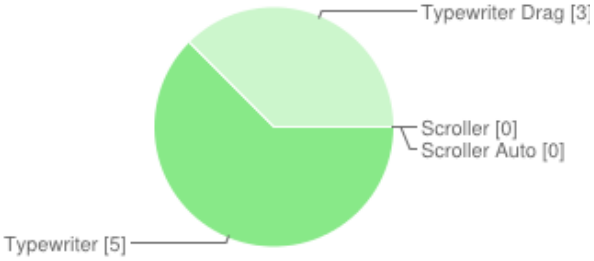
Typewriter Drag	**2**	25%

Which interface did you find was the quickest to enter text?



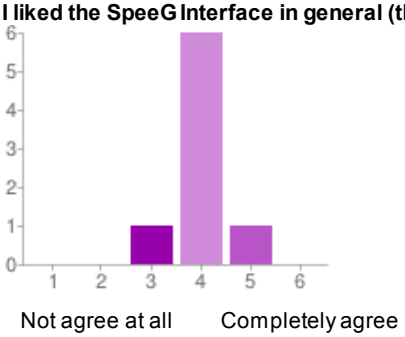
Scroller	**0**	0%
Scroller Auto	**0**	0%
Typewriter	**7**	88%
Typewriter Drag	**1**	13%

Which interface did you prefer?



Scroller	**0**	0%
Scroller Auto	**0**	0%
Typewriter	**5**	63%
Typewriter Drag	**3**	38%

I liked the SpeeG Interface in general (the layout).

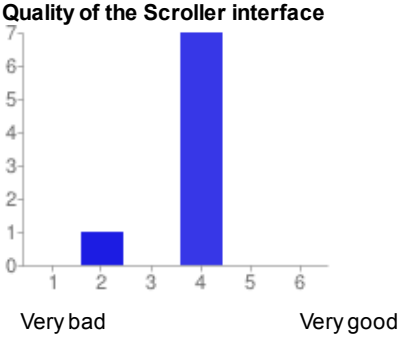


1 - Not agree at all	**0**	0%
2	**0**	0%
3	**1**	13%
4	**6**	75%
5	**1**	13%
6 - Completely agree	**0**	0%

Why did you (not) like it?

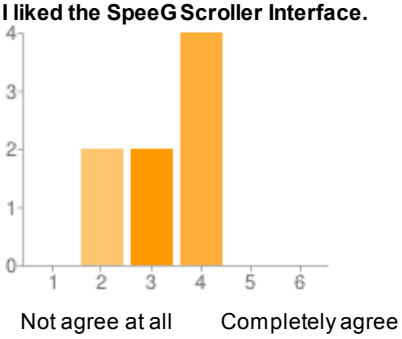
several enhancements should be added: Additional visual feedback Smoother selection of the boxes to reduce accidental selections More alternative words would reduce the need for the letter mode More clear distinction between consecutive sentences Lot of space is wasted, but otherwise it is rather clean The layout of the Interface was good but can be made more attractive (other colors, other font, etc.). Aanduiding van huidig woord, vervelende sensitiviteit very basic, a nicely designed interface would probably help to understand and use the interface. And perhaps an onscreen tutorial would be ea ...

Quality of the Scroller interface



1 -	Very bad	**0**	0%
2		**1**	13%
3		**0**	0%
4		**7**	88%
5		**0**	0%
6 -	Very good	**0**	0%

I liked the SpeeG Scroller Interface.

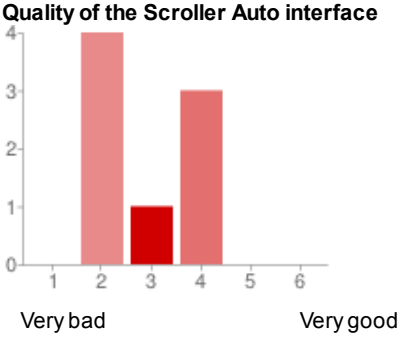


1 -	Not agree at all	**0**	0%
2		**2**	25%
3		**2**	25%
4		**4**	50%
5		**0**	0%
6 -	Completely agree	**0**	0%

Why did you (not) like it?

the user feels more in control compared to the Auto version - It was not always clear what word was 'active'. - Having one insertion point which inserts between the active word and the next (and which is kind of situated there) makes sense in theory, but in practice it's cumbersome to have to move the text to the right spot to insert a word (With current user-interaction systems, you just move the cursor to the right spot; here you must put the cursor in the right spot until the text is in the right spot, and only then you can insert/edit. This feels like a step down from current user-interf ...

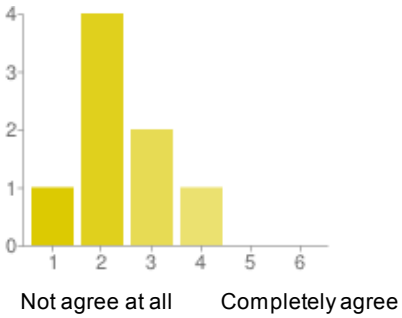
Quality of the Scroller Auto interface



1 -	Very bad	**0**	0%
2		**4**	50%
3		**1**	13%
4		**3**	38%
5		**0**	0%
6 -	Very good	**0**	0%

I liked the SpeeG Scroller Aurto Interface.

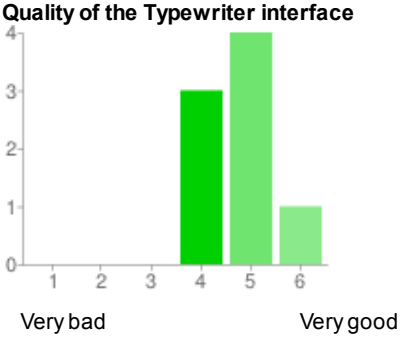
1 -	Not agree at all	**1**	13%
2		**4**	50%
3		**2**	25%
4		**1**	13%
5		**0**	0%
6 -	Completely agree	**0**	0%



Why did you (not) like it?

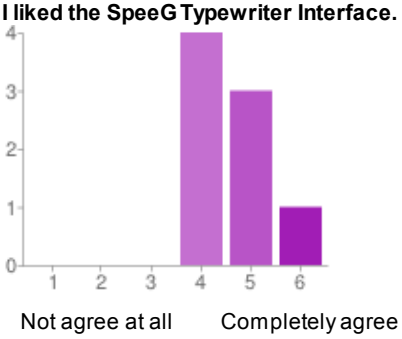
It is too slow, the user has a hard tome knowing which word is the next It was even harder in Scroller Auto to keep track of the active word/letter, because they could be 'in between columns'. It was difficult to select the correct word in the list of words per column. The reason for this was that often other columns were altered unintended. De volledige zin accepteren, aanduiding van actief te selecteren woord Althought the loading bar improves my earlier complaint about the scroller interface, it doesnt solve the problem. The loading bar should be more clear ? bigger or more noticeable colors ...

Quality of the Typewriter interface



1 -	Very bad	**0**	0%
2		**0**	0%
3		**0**	0%
4		**3**	38%
5		**4**	50%
6 -	Very good	**1**	13%

I liked the SpeeG Typewriter Interface.

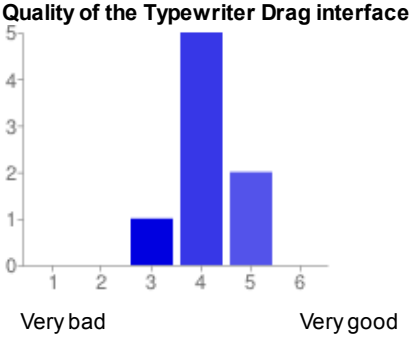


1 -	Not agree at all	**0**	0%
2		**0**	0%
3		**0**	0%
4		**4**	50%
5		**3**	38%
6 -	Completely agree	**1**	13%

Why did you (not) like it?

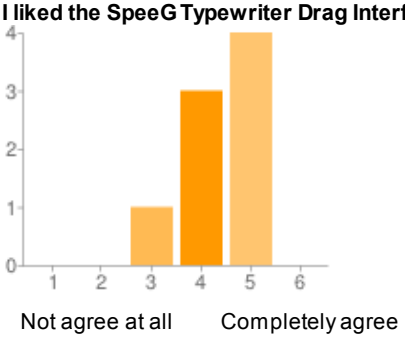
Better than the two Scroller interfaces, however the fact that you cannot cancel a wrong issue is a killing issue Worked relatively fluently; accepting letters/words/phrases that didn't have to be corrected was very fluent, which probably even makes this the most future-proof (as the rate of correct entries is likely to improve). But correcting was by far the easiest in Typewriter as well: The interface was simple; the chance of accidentally doing something you didn't mean to do is low, and if you do it anyway you can quickly undo that action (except for accidentally accepting the words on th ...

Quality of the Typewriter Drag interface



1 -	Very bad	**0**	0%
2		**0**	0%
3		**1**	13%
4		**5**	63%
5		**2**	25%
6 -	Very good	**0**	0%

I liked the SpeeG Typewriter Drag Interface.



1 -	Not agree at all	**0**	0%
2		**0**	0%
3		**1**	13%
4		**3**	38%
5		**4**	50%
6 -	Completely agree	**0**	0%

Why did you (not) like it?

Accidental triggers Requires additional features and testing Good combination of efficiency and usability, with however a few glaring issues which _should_ be corrected ASAP. The drag gestures were confusing, stopping to drag by moving in the opposite direction did not feel intuitive; perhaps using depth to distinguish "dragging the words" from "moving the cursor" would be more effective (if that can be done with the Kinect, that is). The selection of the correct words per sentence was easy and the dragging option to process a sentence was good but difficult to handle at some times. Vervelende ...

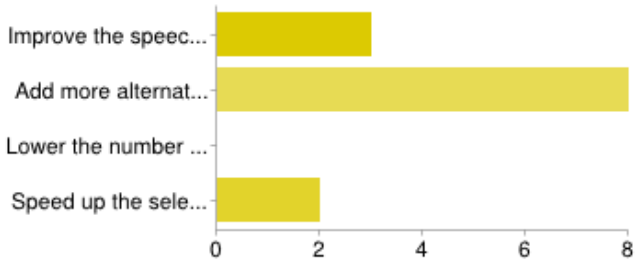
I would replace my Wii/Xbox/PS3/Digibox on-screen keyboard with this interface today

1 -	Not agree at all	**1**	13%
2		**0**	0%
3		**5**	63%
4		**0**	0%
5		**1**	13%
6 -	Completely agree	**1**	13%

Why would you (not) like to replace your current Wii/Xbox/PS3/Digibox on-screen keyboard with this interface?

The XBox interface based on gestures is one of the most inefficient I have ever seen... I have no Wii/Xbox/PS3, and no keyboard on my Digibox, so I can't quite put myself in that position, but I don't think I want to talk to those devices. The data-channel of 'sound-waves through the air' cannot sufficiently be directed at specific targets, shielded from other targets, etc. You run into the same problems humans have when speaking: Sometimes you don't know to whom someone is talking. (And "Soundscape as a shared commodity", NoiseTube, "noise pollution & tragedy of the commons", bla-bla) I don' ...

I find it important to improve the following features of SpeeG



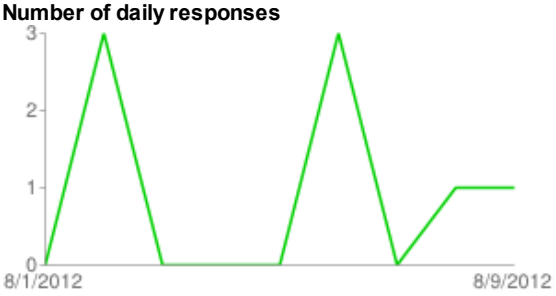
Improve the speech recognition	**3**	38%
Add more alternative word choices	**8**	100%
Lower the number of alternative word choices	**0**	0%
Speed up the selection (pointing) process	**2**	25%

People may select more than one checkbox, so percentages may add up to more than 100%.

Do you have any other proposals to improve the usability? (controls, interface, performance...)

Insertion mode should also support alternative words These comments concern the Typewriter Drag interface: - slightly different colour for the currently focussed word - re arrange the layout to have larger space between the columns, so that users can do more fluid movements - for the drag gesture, the delay before being able to actually drag should be clearly indicated (with a "bubble"?) - Also, the zone that activates the dragging is poorly indicated. Maybe instead of having the most likely word at the bottom, have the most likely words aligned around the center. In case of 5 options, the be ...

Number of daily responses



C

Results for every participant

C.1 P1

An overview of the performance in WPM from participant P1 can be seen in Figure C.1.

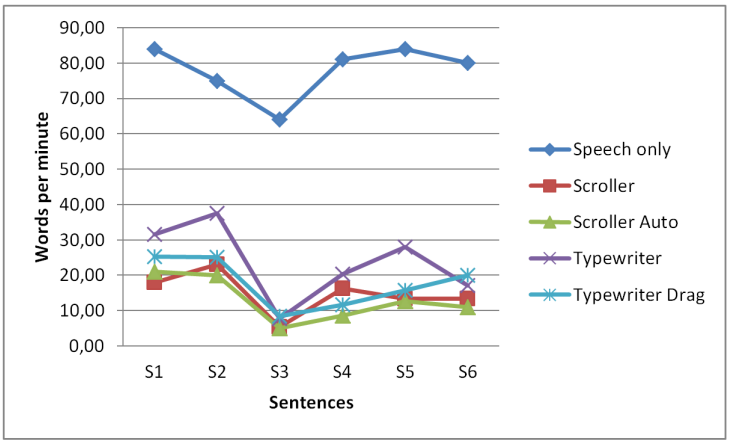


Figure C.1: WPM of each sentence and prototype from participant P1

An overview of the WER before correction from participant P1 can be seen in Figure C.2.

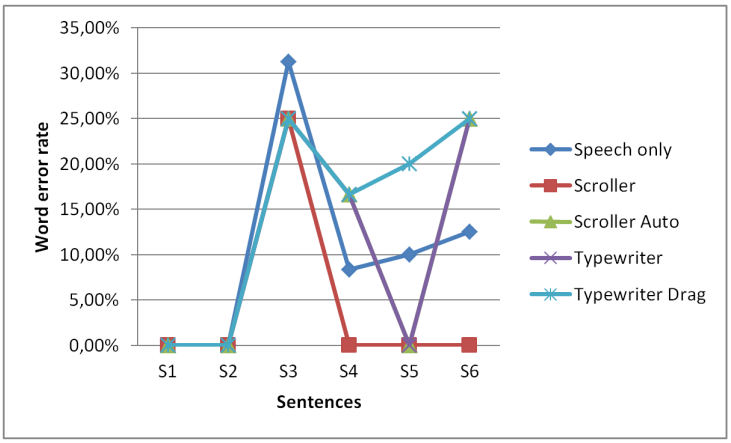


Figure C.2: WER for each sentence and prototype from participant P1

An overview of what correction methods praticipant P1 used can be seen in Figure C.3.

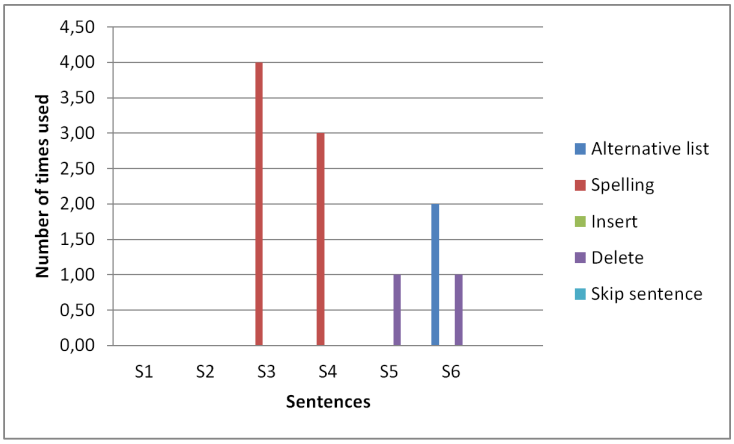


Figure C.3: The total number of correction methods used for each sentence from participant P1

C.2 P2

An overview of the performance in WPM from participant P2 can be seen in Figure C.4.

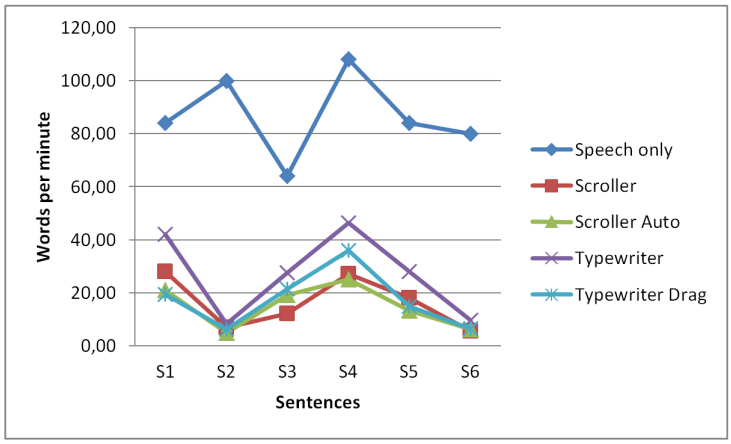


Figure C.4: WPM of each sentence and prototype from participant P2

An overview of the WER before correction from participant P2 can be seen in Figure C.5.

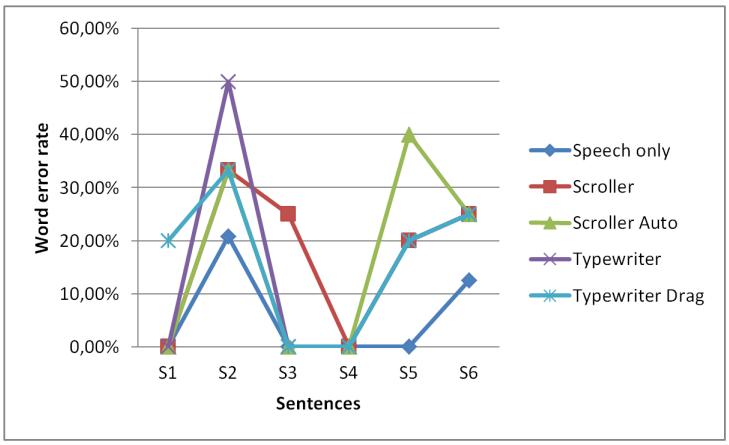


Figure C.5: WER for each sentence and prototype from participant P2

An overview of what correction methods praticipant P2 used can be seen in Figure C.6.

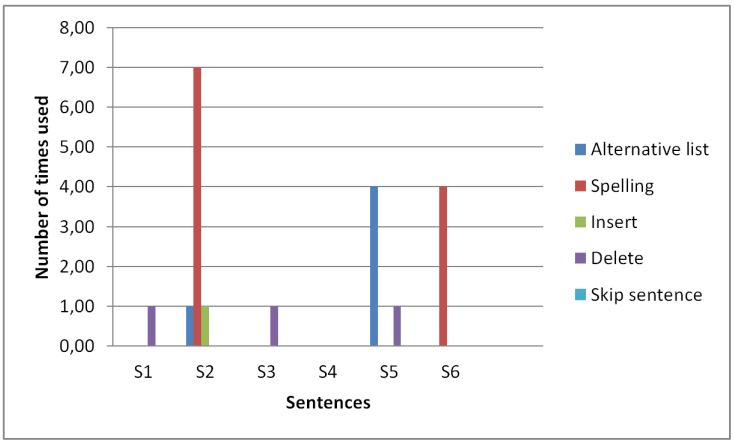


Figure C.6: The total number of correction methods used for each sentence from participant P2

C.3 P3

An overview of the performance in WPM from participant P3 can be seen in Figure C.7.

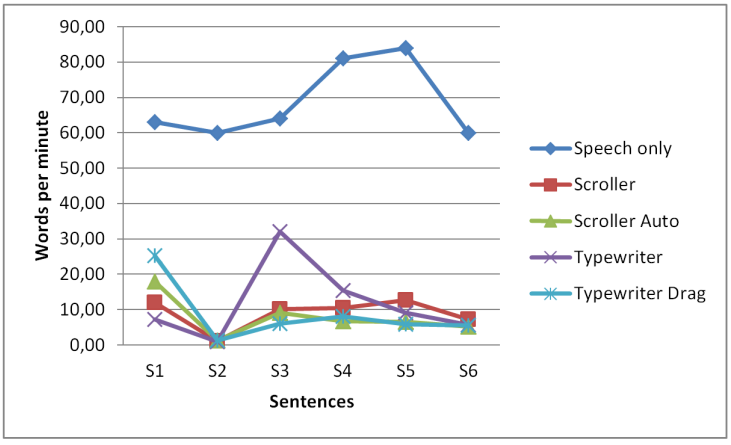


Figure C.7: WPM of each sentence and prototype from participant P3

An overview of the WER before correction from participant P3 can be seen in Figure C.8.

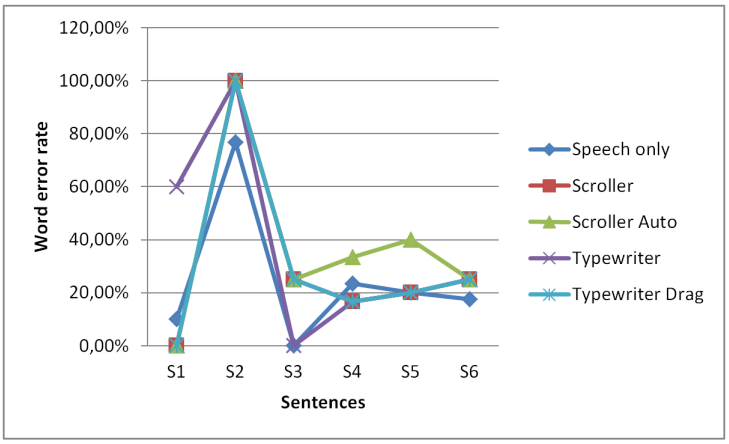


Figure C.8: WER for each sentence and prototype from participant P3

An overview of what correction methods praticipant P3 used can be seen in Figure C.9.

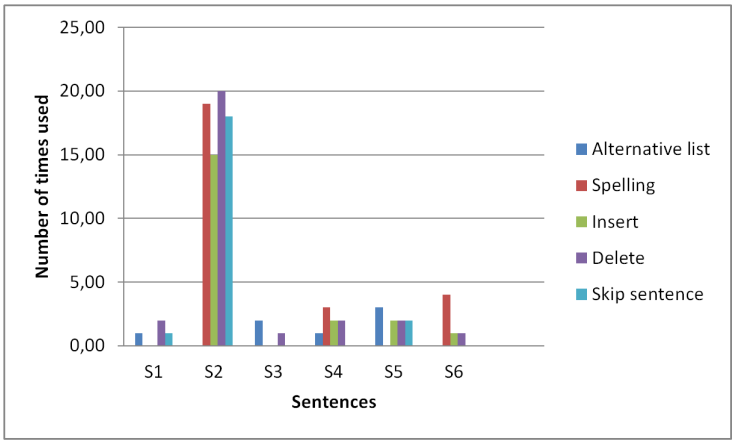


Figure C.9: The total number of correction methods used for each sentence from participant P3

C.4 P4

An overview of the performance in WPM from participant P4 can be seen in Figure C.10.

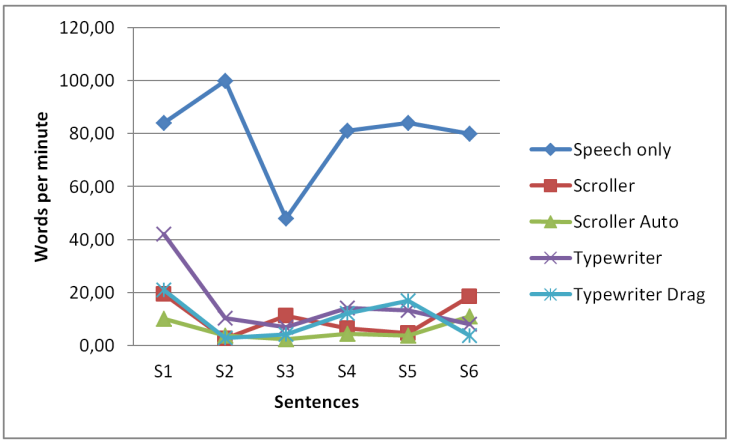


Figure C.10: WPM of each sentence and prototype from participant P4

An overview of the WER before correction from participant P4 can be seen in Figure C.11.

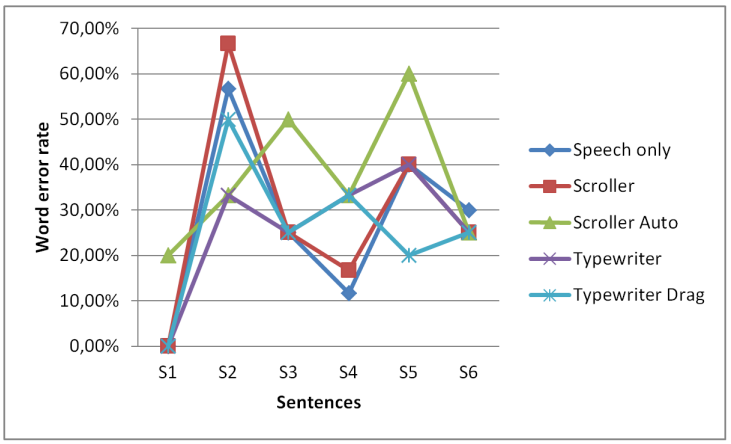


Figure C.11: WER for each sentence and prototype from participant P4

An overview of what correction methods praticipant P4 used can be seen in Figure C.12.

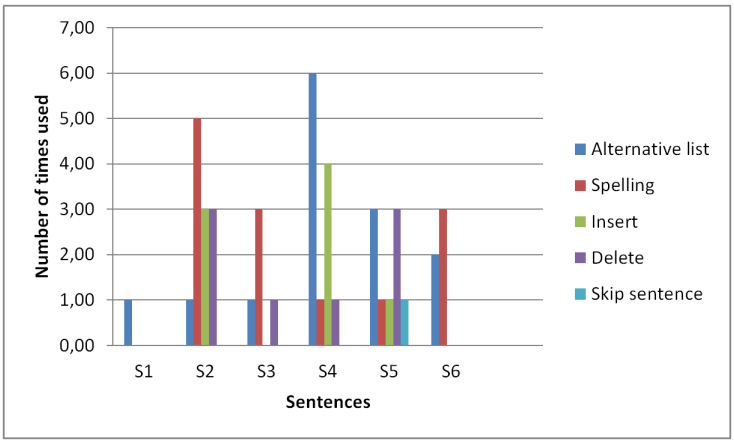


Figure C.12: The total number of correction methods used for each sentence from participant P4

C.5 P5

An overview of the performance in WPM from participant P5 can be seen in Figure C.13.

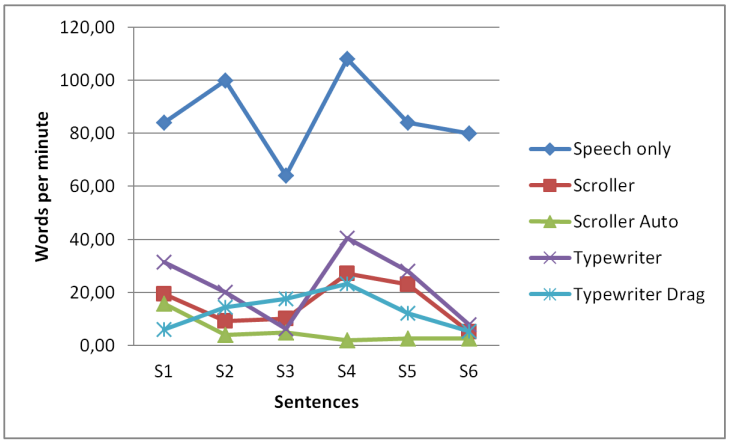


Figure C.13: WPM of each sentence and prototype from participant P5

An overview of the WER before correction from participant P5 can be seen in Figure C.14.

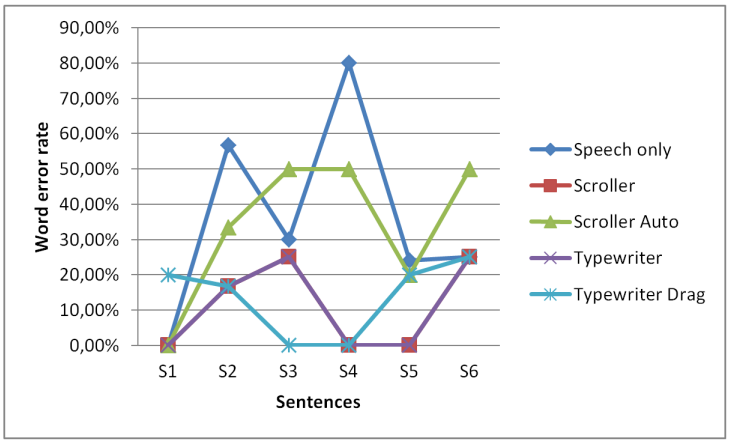


Figure C.14: WER for each sentence and prototype from participant P5

An overview of what correction methods praticipant P5 used can be seen in Figure C.15.

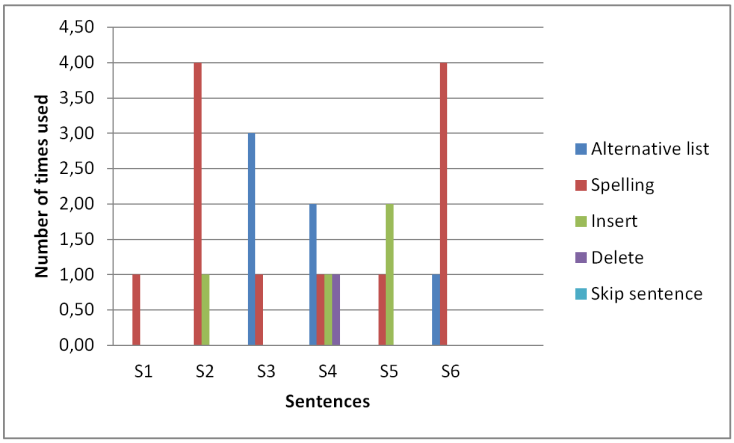


Figure C.15: The total number of correction methods used for each sentence from participant P5

C.6 P6

An overview of the performance in WPM from participant P6 can be seen in Figure C.16.

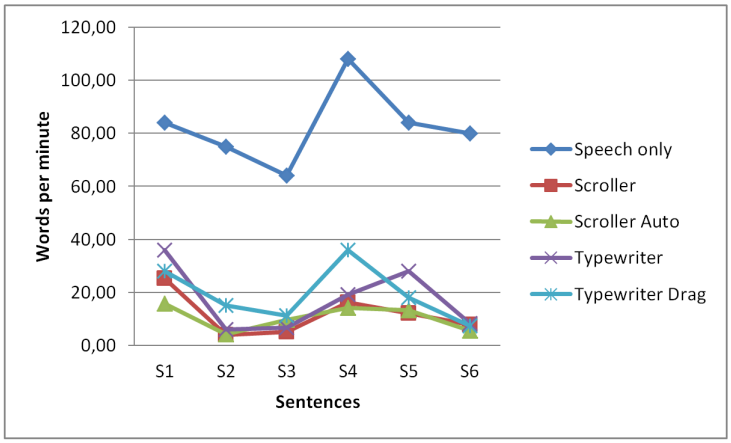


Figure C.16: WPM of each sentence and prototype from participant P6

An overview of the WER before correction from participant P6 can be seen in Figure C.17.

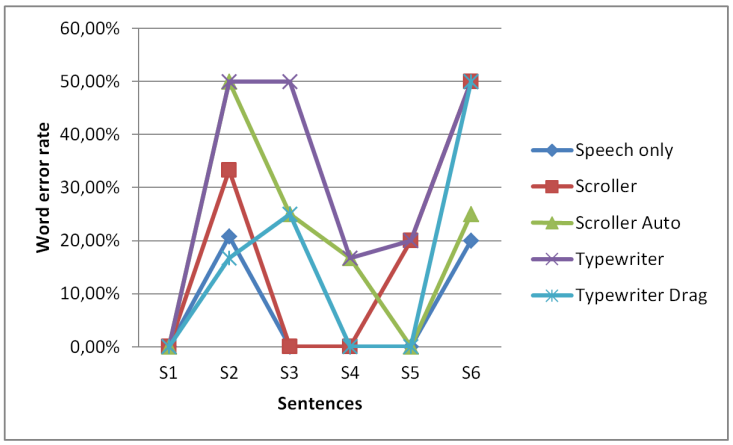


Figure C.17: WER for each sentence and prototype from participant P6

An overview of what correction methods praticipant P6 used can be seen in Figure C.18.

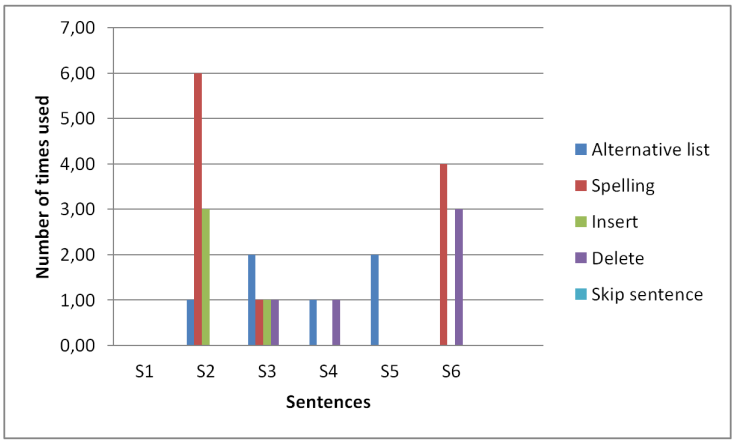


Figure C.18: The total number of correction methods used for each sentence from participant P6

C.7 P7

An overview of the performance in WPM from participant P7 can be seen in Figure C.19.

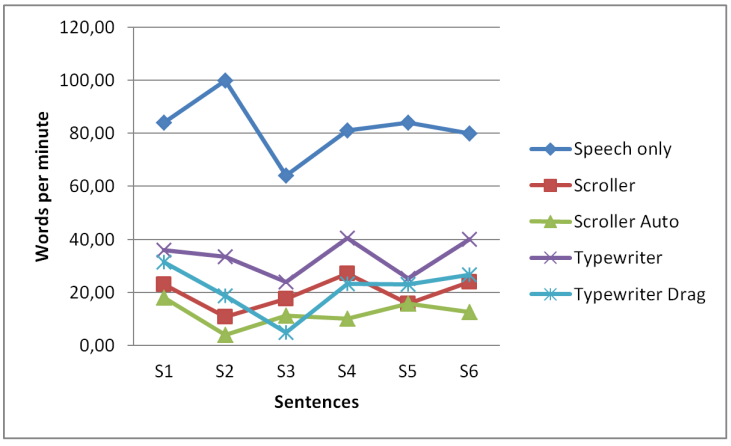


Figure C.19: WPM of each sentence and prototype from participant P7

An overview of the WER before correction from participant P7 can be seen in Figure C.20.

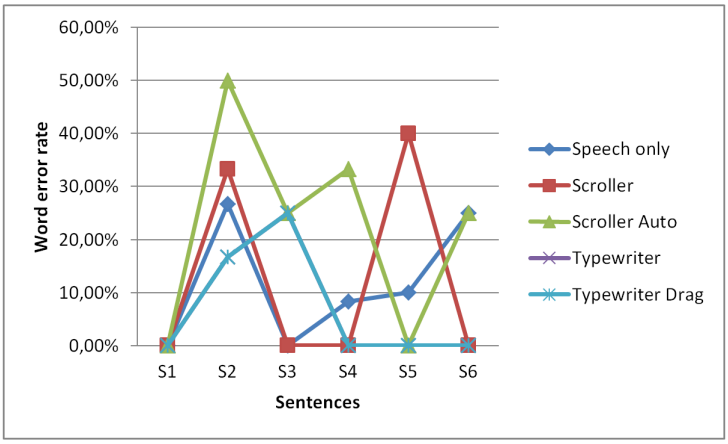


Figure C.20: WER for each sentence and prototype from participant P7

An overview of what correction methods praticipant P7 used can be seen in Figure C.21.

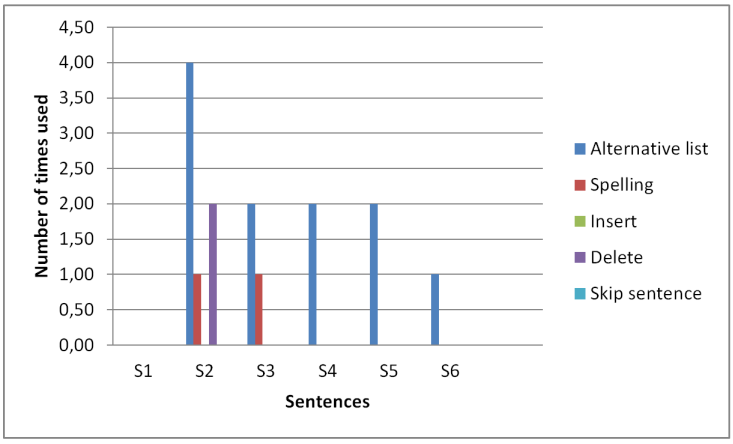


Figure C.21: The total number of correction methods used for each sentence from participant P7

C.8 P8

An overview of the performance in WPM from participant P8 can be seen in Figure C.22.

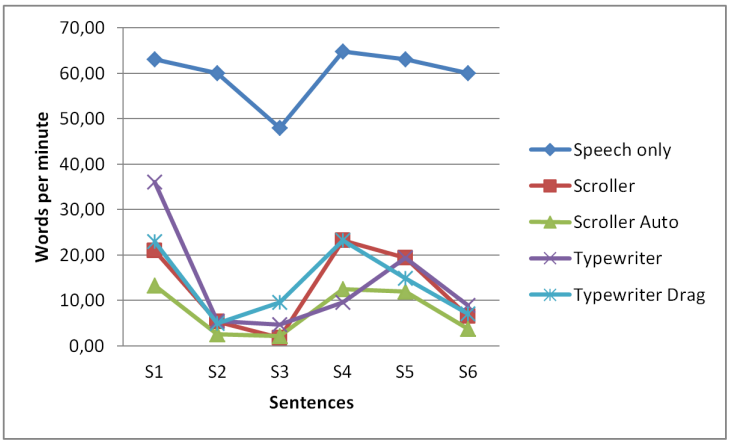


Figure C.22: WPM of each sentence and prototype from participant P8

An overview of the WER before correction from participant P8 can be seen in Figure C.23.

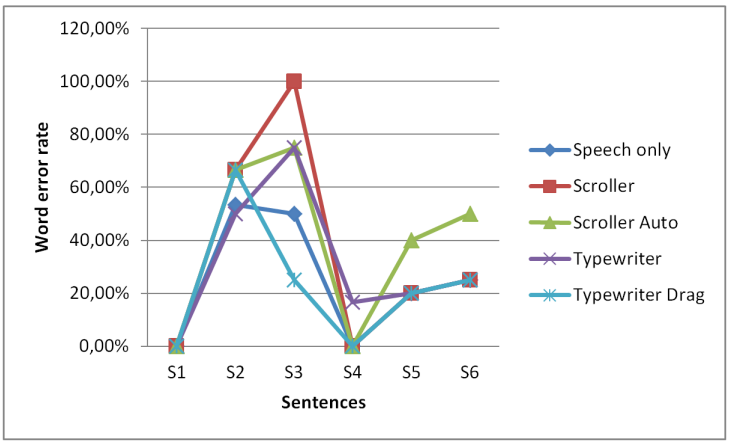


Figure C.23: WER for each sentence and prototype from participant P8

An overview of what correction methods praticipant P8 used can be seen in Figure C.24.

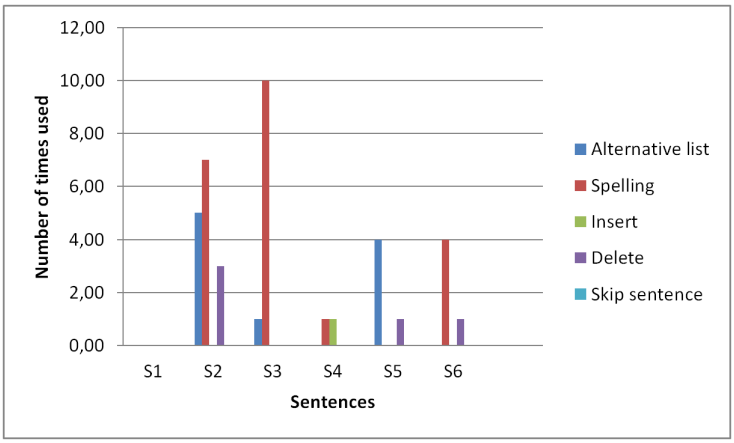


Figure C.24: The total number of correction methods used for each sentence from participant P8

C.9 P9

An overview of the performance in WPM from participant P9 can be seen in Figure C.25.

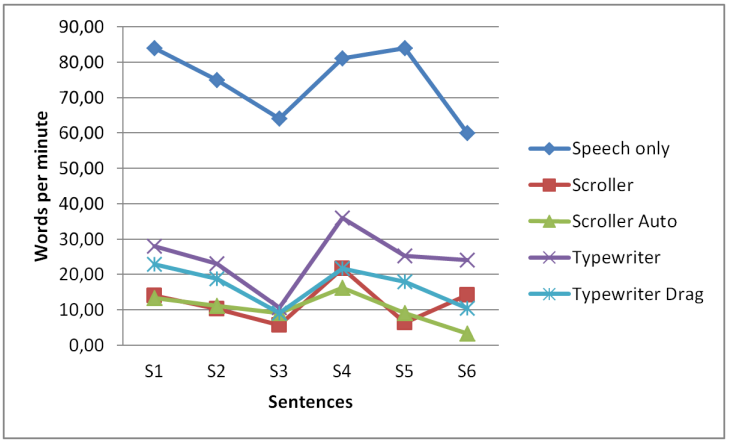


Figure C.25: WPM of each sentence and prototype from participant P9

An overview of the WER before correction from participant P9 can be seen in Figure C.26.

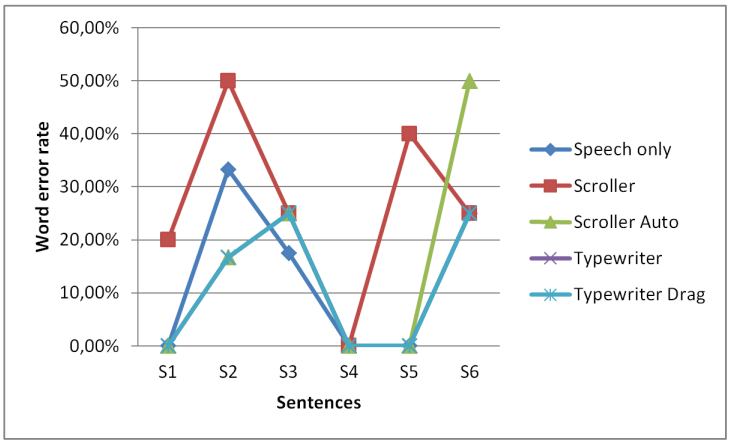


Figure C.26: WER for each sentence and prototype from participant P9

An overview of what correction methods praticipant P9 used can be seen in Figure C.27.

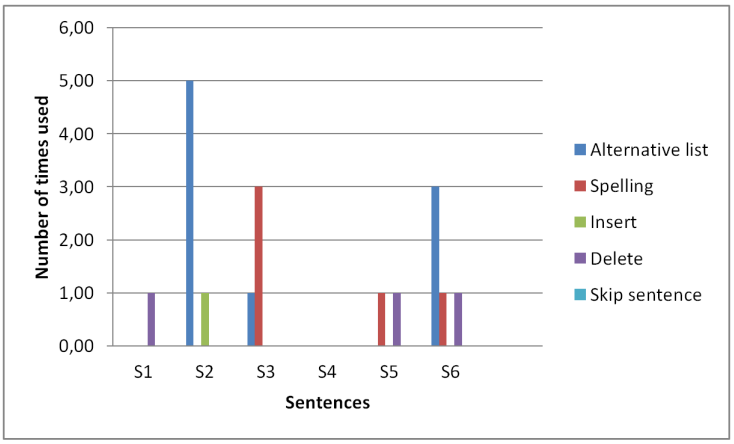


Figure C.27: The total number of correction methods used for each sentence from participant P9