07-125

Automatic testing of speech understanding

Tom Francart[a)]

Exp ORL,

Dept. Neurosciences,

Katholieke Universiteit Leuven,

O & N 2,

Herestraat 49 bus 721,

B-3000 Leuven,

Belgium

Marc Moonen[b)]

SCD,

Dept. Electrical Engineering/ESAT,

Katholieke Universiteit Leuven,

Kasteelpark Arenberg 10,

bus 2446,

B-3001 Heverlee,

Belgium

Jan Wouters[c)]

Exp ORL,

Dept Neurosciences,

Katholieke Universiteit Leuven,

O & N 2,

Herestraat 49 bus 721,

B-3000 Leuven,

Belgium

(Dated: July 6, 2007)

Abstract

Speech understanding tests are commonly performed by manually scoring the oral response of the test subject. This is a time consuming and error prone process that can be avoided by having the subject type the response on a computer keyboard and automatically scoring the response. However, spelling errors may then be counted as recognition errors, and hence influcence the rest results. In this paper we report on an autocorrection approach based on two scoring algorithms to cope with such spelling errors. The first algorithm deals with sentences and is based on word score. The second algorithm deals with single words and is based on phoneme score. Both algorithms are evaluated with a corpus of typed answers based on three different Dutch speech materials. The percentage of differences between the autocorrection algorithm and the manual correction was calculated, in addition to the mean difference in speech recognition threshold between automatic and manual scoring. The sentence correction algorithm performed better than the accuracy that is commonly assumed for the used speech materials. The word correction algorithm performed better than the human operator. Therefore both algorithms can be used in practice and can in some situations even be preferred over a human operator.

PACS numbers: 71.Gv

I. INTRODUCTION

Both in clinical practice and in research, speech understanding tests are widely used to assess speech recognition performance in patients under varying conditions. While speech understanding tests in silence or at a fixed noise level are easy to conduct, they are very time consuming and prone to human errors. Speech understanding tests using an adaptive procedure (Levitt, 1971) or even more complex procedures are harder to conduct manually because interaction is needed to change the signal to noise ratio after each trial.

Human errors can be due to plain scoring mistakes on the part of the operator, but also to unclear pronunciation by the subject. This is an issue in hearing impaired subjects or subjects with a strong dialect and when doing tests with single words instead of with sentences.

Both issues can be addressed by using a computer program to automatically conduct the speech test, where subjects have to enter their response on a computer keyboard. The computer program can then compare the response to the stimulus and select the next stimulus to be presented. Implementation of such a simple program is very straightforward. However, subjects make typing errors, due to both mechanical difficulties operating the keyboard and cognitive errors, which may be counted as recognition errors, and hence influence the test results. Therefore the computer should take into account the possibility of spelling errors and possibly distinguish between such spelling errors and true recognition errors.

Current automatic word correction research may be divided into three broad classes of increasingly hard problems: (1) isolated word error detection, (2) isolated word error correction and (3) context-dependent word correction (Kukich, 1992). In the first class, errors are only detected, not corrected, mainly by looking up words or N-grams in a dictionary or frequency table. In the current state of the art, this class of problems is mainly solved.

[a)]Tom.Francart@med.kuleuven.be
[b)]Marc.Moonen@esat.kuleuven.be
[c)]Jan.Wouters@med.kuleuven.be

The second class, isolated-word error correction, consists of the generation of correction candidates and the ranking of the candidates. A certain input string is to be compared with many entries in a dictionary and amongst the matches, the best match is to be selected. An overview of practical techniques is given by Navarro (2001) and Kukich (1992).

In the third class, context-dependent word correction, not only every individual word is considered but also the words or even sentences around it in the text. Using different approaches such as language models, the noisy channel model, frequency tables and large corpora, the algorithm can then suggest a correction. Reynaert (2005) gives a review of such algorithms. A lot of research is still being done on this type of problem and while many solutions exist to subsets of this problem, in general—correct any spelling error within any context—the problem is essentially unsolved.

Spelling correctors from word processing software typically do the detection task using a dictionary and then suggest a number of possible corrections. They solve problem (1) and part of problem (2). It is clear that this approach is not sufficient for automatic correction of speech understanding tests, because in this case the error must not only be detected but also automatically corrected, without interaction of the user. On the other hand, in the case of speech understanding tests, the hard problem of context-dependent automatic correction can be simplified by using extra information that is readily available: the user does not type a random sentence, but is trying to repeat the sentence that was presented.

In this paper we describe two algorithms for autocorrection of sentences and single words respectively, in the context of speech understanding tests. Both algorithms are evaluated using a custom corpus of manually corrected speech understanding tests and are compared to a simple algorithm that does not take into account spelling errors.

The use of automated speech understanding tests has only been reported a few times in literature, e.g., in Stickney *et al.* (2005, 2004), and autocorrection was never used. However, it has many practical applications in a clinical setting as well as in a research environment. It can even be used for conducting speech understanding tests over the internet.

This paper consists of two main sections. In section II the two algorithms are described

and in section III the development of a test corpus is described and both algorithms are evaluated using that corpus.

II. DESCRIPTION OF THE ALGORITHMS

Two algorithms are described in this section, one for correcting sentences based on a word score and one for correcting words based on a phoneme score. The scoring rules that were used are given in appendix A. The basic principle is that in a sentence test a sentence is considered correct if all keywords are correct. According to the same principle, in a word test a word is considered correct if all phonemes are correct. Some sentence materials are scored based on keywords. The words that are important to get the meaning of the sentence (thus excluding articles, etc.) are considered keywords. This process requires a list of keywords per sentence. If no keywords are defined for a certain speech material and only sentence scores are required, all words are considered keywords by our algorithms. The speech tests that were used to evaluate the algorithms have been normalized using the same scoring rules.

A. The sentence algorithm

1. General

We consider the case where a subject hears a sentence and then has to type this sentence on the computer keyboard. In what follows, the *user input* is the sentence that the test subject types on the computer keyboard, i.e., the sentence to be corrected. The *gold standard* is the correct sentence, the sentence that was presented to the subject.

The algorithm processes two input strings: the user input and the gold standard. A sentence consists of words separated by white space. For each word of the gold standard it is manually indicated whether it is a keyword or not and whether it is part of a splitted keyword. Splitted keywords are keywords that consist of two separate words, but only count as one word when it comes to word score. In English, an example would be "The man wrapped up the package", where "wrapped up" would count as only one keyword.

FIG. 1. Flowchart of the sentence algorithm. An arrow signifies that the output from the source block is used as the input for the target block.

Figure 1 shows the general structure of the algorithm. In what follows, we briefly describe the different blocks:

Input normalization: The punctuation characters , ; . : are replaced by spaces, all remaining non-alphanumeric characters are removed, all diacritics are removed (e.g., ä becomes a, è becomes e), all letters are converted to lower case (e.g., `cApiTAL` becomes `capital`) and multiple sequential white space characters are simplified to a single white space character.

Split into words: The *sentence* is split into *words* by using the space character as a delimiter.

Space correction: Extra spaces in the middle of a word or missing spaces are corrected using the algorithm described in section II.A.2.

Dictionary check: Each word is checked against a dictionary and the results of this check are stored in memory. For our tests, we used the freely available Dutch OpenTaal dictionary (`http://opentaal.org/`).

Number to text: "Words" that consist of only numbers or of a series of numbers followed by a suffix are converted to text using a language specific number to text algorithm. We used a custom algorithm that was manually verified for all numbers from 0 to 10020. Larger numbers did not occur in the speech materials that were used in the evaluation. While subjects were encouraged to always use the numeric form when typing numbers, this step is still necessary in case they did not follow this rule.

List specific rules: Some language and speech-material specific rules in the form of regular expressions (Friedl, 2006) are applied. If for example the sentence material contains

words that can officially be spelled in different ways, one way is selected as the default and the other possibilities are converted to the default. In this stage also some very common spelling mistakes for a language can be corrected. The rules that were used for correction of the Dutch LIST and VU sentence test materials are given in table I. These rules are applied to both the user input and the gold standard. Note that the result does not necessarily correspond to the "correct spelling" any more. Therefore the dictionary check is performed on the data before this transformation.

Bigram correction: The sentence, the results from the dictionary correction and the gold standard are sent to the bigram correction algorithm for the actual spelling correction, as described in section II.A.3.

Word and sentence scores: The word score and sentence score are calculated, as described in section II.A.4.

2. Spacing correction

The input to the space correction algorithm is a typed sentence that was split into words by using the space character as a delimiter. The algorithm then operates on all unigrams and bigrams that can be created based on these words. A bigram is a combination of any two sequential words in a string. If for example the string is "The quick fox jumps", then the bigrams are "The quick", "quick fox" and "fox jumps". In the same way, a single word can be called a unigram. The output of the spacing correction algorithm is a sentence, that is again split into words where however the spacing may have changed.

First the basic approximate string matching mechanism is described, then the operation of the entire space correction algorithm is specified. Approximate string matching is done using the concept of anagram hashing (Reynaert, 2004, 2005). A hash function $H(W)$ is defined that converts a text string W into a numerical value.

$$H(W) = \sum_{i=1}^{|W|} f(w_i)^n \tag{1}$$

Here, W is the input string, $|W|$ the length of W and w_1 to $w_{|W|}$ are the characters in W. The function $f(c)$ gives the ASCII value of character c, i.e., a numerical value between 0 and 255. For example, the values for a, z and the space character are respectively 97, 122 and 32.

Reynaert (2004) found empirically that $n = 5$ is a good value, by considering collisions between values of the hash function on very large corpora. To give an idea of the range of $H(w)$, a few examples are given in table II.

The hash function is used to compare an input string I against a gold string G. If $H(I) = H(G)$, the strings are assumed to be equal.

It is clear that transpositions of characters can still be present if $H(I) = H(G)$. To allow for character insertions in I, we can iterate over the characters of I and check whether $H(I) - H(i_q) = H(G)$, where i_q is the qth character of I for $1 \leq q \leq |I|$. To allow for character deletions from I, we can iterate over the characters of G and check whether $H(I) + H(g_r) = H(G)$, where g_r is the rth character of G for $1 \leq r \leq |G|$. To allow for character substitutions, we can perform a nested iteration over the characters of I and G and check whether $H(I) - H(i_q) = H(G) - H(g_r)$ where i_q is the qth character of I and g_r is the rth character of G for $1 \leq q \leq |I|$ and $1 \leq r \leq |G|$.

Note that $H(A) + H(B) = H(A \oplus B)$, where A and B are strings and $\oplus$ denotes the concatenation operation. We prefer to write $H(A) + H(B)$, because in a concrete implementation, the values of $H(A)$ and $H(B)$ only have to be calculated once and can then simply be added.

For the space correction algorithm, this check is extended to $H(I) - H(i_q) = H(G) - H(g_r) \pm H(s)$ where s is the space character, i.e., it is checked whether the letters in I correspond to the letters in G plus some other character, minus some other character and plus or minus the space character.

Now that the basic approximate string matching mechanism is clear, we can describe

the overall function of the space correction algorithm. It consists of the following steps.

Gold hashing: Every hashed gold word and gold bigram is stored in a hash list.

Bigram checking: The hash value of every user input bigram is checked against the hash list.

Unigram checking: The hash value of every user input word is checked against the hash list.

Space correction: If in the previous steps a matching unigram or bigram is found, it is aligned to the input uni/bigram and spaces from the found uni/bigram are inserted in the input uni/bigram.

Checking against the hash list works by looking up $H(I) - H(i_q) + H(c_x) \mp H(s)$, with c_x any character from the alphabet, in the hash list and checking if the found value corresponds to I without taking into account any space characters.

The hashing-part of the space correction algorithm in its current form is similar to calculating an error measure such as the Levenshtein distance (Levenshtein, 1965) between the user input and gold string and allowing a certain number of errors, but it is faster and easily extensible. A Levenshtein distance calculation algorithm, implemented using dynamic programming has a complexity of $O(|S_1| \cdot |S_2|)$, where $|S_1|$ and $|S_2|$ are the length of the input strings S_1 and S_2, whereas the hash-function approach only requires a few hash value calculations, hash table lookups and additions.

When a new bigram is found, it is determined if and where spaces should be inserted. The process is illustrated in figure 2. First all spaces are removed from the user input. Then it is aligned to the gold standard bigram using a dynamic programming method (Cormen *et al.*, 2001). If the number of corresponding letters is larger than 90%, spaces are inserted in the corresponding places in the user input bigram.

FIG. 2. Example of string alignment. Spaces are marked by empty boxes. In this case the gold string is "word score" and the user input string "woldsc re". First all spaces are removed from the input string. Then both strings are aligned. The space character marked with the single arrow could be inserted into the input string as shown. However, as the percentage of correctly aligned characters ($100 \cdot 7/10 = 70\%$) is smaller than 90%, no space will be inserted because the strings are in this case not considered sufficiently alike.

3. Bigram correction

The bigram correction algorithm takes as input the result from the space correction algorithm that is again split into words using the space character as a delimiter. It operates in the same way as the space correction algorithm, with two differences. The comparison function is now $H(I) - H(i_q) + H(c_x) = H(G)$, i.e., without the extra $\pm H(c_s)$, and words are only considered for correction if they are not in the dictionary.

The result is a corrected list of words in the input sentence that is then sent to the word score determination block.

4. Word score determination

The word score is calculated by comparing the result from the bigram correction to the gold standard (after the transformations previously described). It is the number of corrected keywords in the user input that correspond to gold keywords. The corresponding words must occur in the same order in both strings.

To decide if whether two words are the same, the following rules are followed:

- If the user input and gold word are numeric, they must match exactly.
- If the double metaphone representation of the user input and gold word differ, they are considered different. The double metaphone algorithm (Phillips, 2000) was built for doing phonetic comparisons across languages.

- If the Levenshtein distance (Levenshtein, 1965) between the user input and gold word is larger than 1, they are considered different. The Levenshtein distance is also called the edit distance and is the sum of the minimum number of insertions, deletions and transpositions necessary to transform one string into the other.

B. Example of the sentence algorithm

We illustrate the function of the entire algorithm by means of an example. Let the user input be

`Theboy fel from the windaw.`

and the correct answer

`The` **boy fell from** `the` **window**

We will use the words in bold as keywords. The user input is transformed by the different functional blocks as follows.

Input normalization: `theboy fel from the windaw`

Correct spacing: `the boy fel from the windaw`

Dictionary check: The words `fel` and `windaw` are not in the dictionary and can thus be corrected.

Bigram correction: The gold bigrams are given in table III. The bigrams that can be corrected (according to the dictionary check) are given in table IV. Looking up hash values in the hash list and replacing words where appropriate, yields the string: `the boy fell from the window.`

Word score: The gold standard and the corrected input sentence are exactly equal, the word score algorithm yields a keyword score of 4/4 and a corresponding sentence score of 1.

FIG. 3. General structure of the word correction algorithm.

C. The word algorithm

1. General

Speech understanding tests can also be done with single words. Typically words with a well defined structure, such as Consonant-Vowel-Consonant (CVC) are used and scores are given based on the number of phonemes correctly identified. In the following sections, an algorithm is described for automated scoring of these word-tests.

The organisation of the word algorithm is illustrated in figure 3, where the main steps are:

Input normalization: The input is transformed to lower case and diacritics and non-alphanumeric characters are removed.

Number to text: If the input consists of only digits, the number is converted to text (using the same number to text algorithm as used in the sentence correction algorithm, section II.A.1).

Conversion into graphemes: The input is converted into a series of grapheme codes (section II.C.2)

Compare graphemes: The user input and gold standard grapheme codes are compared (section II.C.3), resulting in a phoneme score, from which the word score can be derived.

2. Conversion into graphemes

This module operates on both the user input word and the gold standard word. It makes use of a language-specific list of graphemes. Every grapheme corresponds to a numeric grapheme code. Graphemes that sound the same receive the same code. The list that is

currently used for Dutch is given in table V. Some graphemes correspond to the same phoneme only if they occur at the end of a word and not if they occur in the middle of a word. Therefore, if a `g` or `d` occurs at the end of the word, it is converted to the code of `ch` or `t`.

The algorithm looks for the longest possible grapheme in the string. For example `boot` would be converted into [2 40 19] and not into [2 14 14 19].

3. Compare graphemes

The phoneme score is calculated by comparing the two arrays of grapheme codes. First, graphemes that do not occur in the user input grapheme list are removed from the gold grapheme list and graphemes that do not occur in the gold grapheme list are removed from the user input grapheme list. Then the score is calculated as the number of corresponding graphemes for the best alignment of both arrays. The best alignment is defined as the alignment that yields the highest score.

D. Example of the word algorithm

As an example, the word `kieuw` is presented to the test subject. If the typed word (user input) is `kiew`, the autocorrection proceeds as follows:

Grapheme conversion: `kieuw` is converted to [10 31 37] and `kiew` is converted to [10 31 37]

Grapheme comparison: Correlation of the 5 different alignment positions of both arrays yields [0 0 3 0 0], thus the score becomes 3

As a second example, the word `dijk` is presented to the test subject. If the typed word (user input) is `bij`, the autocorrection proceeds as follows:

Grapheme conversion: `dijk` is converted to [4 35 10] and `bij` is converted to [2 35]

Grapheme comparison: As the graphemes number 4, 10 and 2 only occur in one of both arrays, they are removed. The resulting arrays are [35] and [35]. Cross correlation of both arrays yields [1], thus the score becomes 1.

III. EVALUATION OF THE ALGORITHMS

To asses the feasibility of the usage of both algorithms in practice, a corpus of typed responses to speech understanding tests was developed and used to evaluate the algorithms. The difference in score between the autocorrection algorithm and the manual score is determined and compared to the score error introduced by manually scoring the speech tests.

A. Development of a test corpus: procedures

To develop a test corpus, a clinical test setup was reproduced. However, in addition to repeating the speech token they heard, the subjects also typed the token on a computer keyboard. The operator then scored the oral response using the standard procedures for each speech material. Two final year university level audiology students conducted the experiments and the analyses described in this paper were performed by a third person.

For each speech material, clear rules were established for obtaining the score, corresponding to the rules that were used for the normalization of the speechmaterials and described in the corresponding papers. All subject responses and manual corrections as described in the next paragraph were combined into a test corpus that can be used to fine-tune or evaluate an autocorrection algorithm.

A corpus entry consists of the following elements:

Correct sentence: The sentence as it was presented to the subject, annotated with keywords and splitted keywords

Subject response: The string as it was typed on the computer keyboard by the test subject.

Manual score: (MO) The score that was given by the audiologist using the *oral* response. This was done by indicating correctly repeated words on a printed copy of the speech token lists and manually calculating the word score and sentence score on the spot.

Corrected manual score: (MOC) In a first iteration, typed responses were run though the autocorrection algorithm and every difference in score between the algorithm and the manual score of the oral response was analyzed by the operator using the notes he made during the experiment. When the operator appeared to have made an error while scoring the oral response, it was corrected.

Manual score based on typed response (MT) Every string entered by the subject was manually scored, ignoring spelling errors.

If only the pure autocorrection aspect of the algorithm is to be evaluated, the MT scores are the most relevant. It corresponds to giving typed input sentences to a human operator and to the autocorrection algorithm and have both of them calculate the word score and sentence score.

To assess performance in real-life situations, the MOC scores have to be considered. Here "the perfect operator" is used as a reference. Differences between the MOC and MT scores are due to differences between the oral response and the typed response.

Finally, differences between the MO and MOC scores correspond to errors made by the operator, and that will be present in any real experiment. Thus, to assess real-life performance of the algorithm, it is useful to consider the difference between the errors made by the operator and errors made by the algorithm, i.e., the extra errors introduced by the algorithm.

B. Materials

Three different Dutch speech materials were used to develop three different corpora:

NVA words: (Wouters *et al.*, 1994) 15 lists of 12 consonant-vowel-consonant (CVC) words, uttered by a male speaker.

LIST sentences: (van Wieringen and Wouters, 2006) 35 lists of 10 sentences, uttered by a female speaker. Each list contains 32 or 33 keywords. A sentence is considered correct if all keywords are repeated correctly and in the right order. Both a keyword score and sentence score are defined. This sentence material is intended for testing cochlear implant users.

VU sentences: (Versfeld *et al.*, 2000) 39 lists of 13 sentences, uttered by a male speaker. A sentence is considered correct if all words, not only keywords, are repeated correctly. Usually, only a sentence score is used with the VU sentences.

The NVA words were presented in silence at 3 different sound pressure levels (well audible, around 50% performance and below 50% performance, ranging from 20dBSPL up to 65dBSPL).

The LIST and VU sentence materials were masked by four different noise materials: speech shaped noise, a competing speaker in Dutch, a competing speaker in Swedish and the ICRA5-250 speech shaped noise modulated with a speech envelope (Wagener *et al.*, 2006). The operator was instructed to measure at least 3 signal to noise ratios (SNR) for each condition with the purpose of determining the speech reception threshold (SRT) by fitting a psychometric curve through these points later on. The number of measured SNR's per condition varied between 3 and 5 and the used SNR varied between −20dB and 10dB. The SNRs used for each subject were recorded.

As the sentence algorithm is based on keywords, we marked keywords for the VU sentences and used these for the algorithm. They were marked according to the same rules that were used while developing the LIST sentence material. In simplified form, this means that all words are keywords except pronouns, adpositions, auxiliary verbs and articles. This condition is labelled (word).

In clinical practice however, the VU sentences are scored differently: a sentence is only counted as correct if all words, not only keywords, are repeated correctly. Therefore, we also performed autocorrection using all words in the gold standard sentence as keywords instead of only the keywords that were marked. This condition is labelled (sent).

C. Subjects

To obtain a diverse corpus, 20 students of the university of Leuven were recruited (group 1), as well as 13 subjects who reported having problems with spelling (group 2).

D. Evaluation

We evaluated both autocorrection (Ac) algorithms using our different corpora. In a first stage, we measured the number of false positives and negatives. In a second stage, we assessed the influence on the obtained SRT, the value that is traditionally derived from speech understanding tests.

For comparison, we also performed autocorrection on our corpora using a very simple algorithm that counts the number of keywords that are exactly the same in the input sentence and in the gold standard, and that also occur in the same order. This algorithm is labeled SIMPLE. While it has a very high false negative rate, the results give an impression of the number of spelling mistakes that were made in each condition.

To evaluate of the number of errors made by the human operators, the percentages of modifications between the MO and MOC conditions were calculated.

1. Percent correct

In a first phase, we calculated the difference between the manual scores (for word score and sentence score) and the automatically generated scores. The difference is given as percentage errors of the autocorrector, with the manual score as a reference. As there are

different manual scores (cf. section III.A), several scores are given for each condition in table VI.

For sentences, results are given for word score and sentence score. The word score is the number of keywords that were incorrectly scored per total number of keywords. The sentence score is based on the word score and is a realistic indicator for real-life performance, as sentence scores are commonly used for the LIST sentences and always used for the VU sentences.

For words, results are given for phoneme score and word score. The word score is based on the phoneme score. Here, the phoneme score is the most realistic indicator, as phoneme scores are commonly used in practice.

2. Influence on SRT

The SRT is commonly determined by fitting a two-parameter logistic function through the percent-correct values found at different SNR's recorded during subject testing.

We assessed the influence of the autocorrection algorithm on the estimated SRT by calculating the difference between the SRT determined by fitting the percent correct values found by on the one hand the manual scoring (MOC) and on the other hand the autocorrection algorithm (Ac).

There were always three or more data points (SNR values) per condition (speech material/noise type/subject). The average difference in SRT between manual and automatic scoring for each speech material is given in the last column of table VI. As the accuracy of the SRT determined by this method is usually not better than $\pm$1dB (van Wieringen and Wouters, 2006; Versfeld *et al.*, 2000), our algorithm will have no significant impact on an estimated SRT value if the difference stays below this value.

IV. RESULTS

Table VI shows percentages of errors of the autocorrection algorithm versus the different manually scored entries in the corpus. It reveals several interresting tendencies.

First, considering the number of tokens presented, 3280 (group 1) or 1310 (group 2) LIST sentences correspond to 328 or 131 lists of 10 sentences. This means that overall each of the 35 lists of sentences was presented at least 3 times to each group of subjects and often more. For the VU sentences, similarly, 258 or 134 lists were presented corresponding to at least 3 presentations to each group. Similarly, for the NVA words 63 and 50 lists were presented. This means that each of the 15 NVA word lists was presented at least 3 times to each group of subjects.

Comparing the results for group 1 and 2, the "good" and the "bad" spellers, shows that the simple algorithm (column 4) made many more errors in group 2. The results from the simple algorithm are of course the same for the VU (word) and VU (sent) conditions, but are shown twice for clarity. Comparing the autocorrection performance between the two groups, shows that it made slightly more errors for group 2, on average 0.5 % difference in wordscore errors for the LIST sentences and 0.3 % for the VU sentences.

The differences between the MO and MOC scores correspond to errors introduced by manually scoring the speech tests, either by misunderstanding the oral response or by miscalculating the resulting score. Comparing column 5-6 (MO-Ac) and 7-8 (MOC-Ac) shows that both the word scores and sentence scores improve on average by 1.0 % less errors of the autocorrection algorithm. Furthermore, the difference between the MOC scores and MT scores is related to the difference between the typed response and the oral response. It gives an indication of how hard it was for the subjects to combine the typing task with the speech perception task. The average difference between the MOC-Ac scores and the MT-Ac scores is 0.5 %.

The ΔSRT values give the mean difference in SRT found from the psychometric function when using the MOC and Ac scores. While all ΔSRT values differ significantly from each

other, both between groups and between speech materials, the absolute differences are very small and there is no clear tendency of change. Note that for each condition only 3 or 4 SNRs were measured and that ΔSRT will decrease if more SNRs are measured per condition. For example for group 1 the percentage of errors for the LIST sentences is 0.6%. In the case of 3 SNRs measured per condition, the average number of errors per condition is $10 \times 3 \times 0.006 = 0.18$. This means that in most cases the SRT will not be influenced at all, but if there is an error present in any of the sentences of this condition, it may have a large influence on the SRT because the psychometric curve (with 2 parameters) is fit through only 3 datapoints.

The percentages of errors for the sentence scores tend to be somewhat larger than those for the word scores. This is due to the fact that any word of a sentence that was incorrectly scored leads to an error in the score of the entire sentence, while in the case of word scoring, it only leads to an error for one of the words of the sentence.

Comparing percentages of errors between the VU sentences with keyword scoring and sentence scoring (where every word is considered a keyword and thus influences the sentence score), shows that the algorithm performs better with keyword scoring. The reason is that a human operator tends to ignore or just mishears small errors in words that are irrelevant for the meaning of the sentence.

The MO-MOC scores indicate the number of errors the made by the human operator. The average human error for word scoring is 3.4 % and for sentence scoring it is 0.9 %. Comparing these values to the values in column MOC-Ac and MT-Ac, it can be seen that the average number of errors made by the autocorrection algorithm is nearly an order of magnitude smaller.

Comparison of the autocorrection performance between the LIST and VU sentences, reveals no significant difference using a paired t-test for any of both groups of subjects. However, comparing the scores for the VU sentences with sentence and keyword scoring respectively, does reveal a significant difference ($p < 0.01$).

V. DISCUSSION

The within-subjects standard deviation on the SRT determined using an adaptive procedure is 1.17dB for the LIST sentences in noise (van Wieringen and Wouters, 2006) and 1.07dB for the VU sentences in noise (Versfeld *et al.*, 2000). As the error introduced by using the autocorrection algorithm is an order of magnitude smaller, its use will have no noticeable influence on the test results.

For assessing real-life performance of the autocorrection algorithms, the MOC scores should be used as a reference, because these compare the algorithms to a well established standard, i.e., manual scoring of oral responses. As the errors in the condition when the percent correct score is used directly, without calculating the SRT, are very small, they will in most cases be below the expected accuracy of testing. Moreover, the number of errors made by the autocorrection algorithm is similar to or smaller than the number of errors made by the operator, so the results will not be influenced more by the use of autocorrection than by the errors made by a human operator, or by the possible bias of the human operator.

The simple non-autocorrecting algorithm (Simple) should not be used in practice, especially when the subjects are expected to have problems with spelling. Comparison of the results of the simple algorithm between group 1 and group 2 reveals that the subjects of group 2 indeed made many more spelling errors. Comparison of the results of the autocorrection algorithms between groups shows that the results are slightly worse for group 2, but still within the expected accuracy of speech understanding tests. It should however be noted that the subjects in group 2 were selected based on their self-reported problems with spelling. Therefore the results for group 2 should be observed as worst case results. In normal circumstances, these subjects would probably not be tested using an automated setup because they need a lot of encouragement and repeated instructions. Nevertheless, the percentages of errors of the autocorrection algorithm are still in the same range as the percentages of errors made by a human operator. The algorithm itself copes very well with this difficult task.

Comparison of the MOC en MT scores shows that there is a small difference between the operator's assessment of the oral response and the typed response. It is however never clear what the intended answer is: did the subject intend to answer what he said or what he typed?

In the case of the word correction algorithm, the results are even more extreme: here the autocorrection algorithm always performs better than the human operator. This is probably due to on the one hand unclear articulation by the subjects and on the other hand the difficulty of the task: the experimenter has to stay very well concentrated during the repetitive task and in a few seconds time, he has to decide which phonemes were correctly repeated. When the answer is approximately correct, there is a chance of positive bias and when the answer is wrong, it is not always straightforward to identify a single correctly identified phoneme using the strict rules.

Comparing the number of errors for the VU sentences with respectively keyword scoring and sentence scoring, indicates that while the percentages of errors for sentence scoring are still very acceptable, the algorithm is best used with keyword scoring. As a human experimenter tends to ignore small errors in words that do not contribute to the meaning of the sentence – even if according to the strict rules the response is incorrect – keyword scoring using the autocorrection algorithm comes closest to this situation.

Both algorithms were developed for the Dutch language. Applicability to other languages depends on the correspondence between phonemes and graphemes of the language. While in Dutch this correspondence is rather strict, this is not the case in other languages (e.g., English). In any case, we expect the autocorrection algorithms to perform very well in German, Swedish, Danish, French, Italian, Spanish, Polish, Turkish and many other languages. For converting the sentence algorithm to another language, the only blocks that have to be changed are the language and speech material specific rules and of course the list of keywords of the speech material. For converting the word algorithm to another language, the list of phonemes and phoneme codes has to be changed.

VI. CONCLUSION

The autocorrection algorithms for both sentence tests and word tests are very well suited for use in practice and will not introduce more errors than a human operator.

In a clinical setting, the use of automated speech understanding tests may be rather limited because the subjects require clear instructions anyway and some subjects may not be able to efficiently use a computer keyboard. However, automated speech understanding tests can be very usefull in many other area's, including research, screening large groups of patients and remote tests (e.g., over the internet).

When test subject does not articulate clearly it can be very hard to manually score single words, especially when testing hearing impaired subjects. In this case automatic scoring using our autocorrection algorithm should be preferred over manual scoring.

Acknowledgments

This research was partly sponsored by Cochlear Ltd. and by the IWT (Institute for the Promotion of Innovation by Science and Technology in Flanders) project 050445. We thank our test subjects for their patient and enthusiastic participation to the speech tests. We also thank Inne Vanmoer and Evelyne Bekaert for their help in gathering data for the corpus.

APPENDIX A: SCORING RULES

1. CVC tests

1. Every phoneme that is repeated correctly results in 1 point
2. A phoneme must be exactly correct, even if the difference is small
3. The phonemes must be repeated in the right order
4. Extra phonemes before or after the correctly repeated phonemes have no influence on the score

2. Sentence tests

1. Every keyword that is repeated correctly results in 1 point
2. A keyword must be exactly correct, e.g. if the plural form is given when the singular form was expected, the word is considered incorrect.
3. Both parts of verbs that can be split must be repeated correctly for the verb to be scored as correct.

REFERENCES

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (**2001**). *Introduction to Algorithms*, chapter 15, 323–369, 2nd edition (MIT Press & McGraw-Hill).

Friedl, J. E. F. (**2006**). *Mastering Regular Expressions*, 3rd edition (O'Reilly).

Kukich, K. (**1992**). "Technique for automatically correcting words in text", ACM Comput. Surv. **24**, 377–439.

Levenshtein, V. (**1965**). "Binary codes capable of correcting spurious insertions and deletions or ones.", Problems of information transmission 8–17.

Levitt, H. (**1971**). "Transformed up-down methods in psychoacoustics.", J Acoust Soc Am **49**, Suppl 2:467+.

Navarro, G. (**2001**). "A guided tour to approximate string matching", ACM Computing Surveys **33**, 31–88.

Phillips, L. (**2000**). "The Double Metaphone Search Algorithm", C/C++ Users Journal .

Reynaert, M. (**2004**). "Multilingual text induced spelling correction", in *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*.

Reynaert, M. (**2005**). "Text-induced spelling correction", Ph.D. thesis, University of Tilburg.

Stickney, G., Zeng, F., Litovsky, R., and Assmann, P. (**2004**). "Cochlear implant speech recognition with speech maskers.", J Acoust Soc Am **116**, 1081–91.

Stickney, G. S., Nie, K., and Zeng, F. G. (**2005**). "Contribution of frequency modulation to speech recognition in noise.", J Acoust Soc Am **118**, 2412–20.

van Wieringen, A. and Wouters, J. (**2006**). "LIST en LINT: sentences and numbers for quantifying speech understanding in severely impaired listeners for Flanders and The Netherlands", Int J Audol (submitted) .

Versfeld, N. J., Daalder, L., Festen, J. M., and Houtgast, T. (**2000**). "Method for the selection of sentence materials for efficient measurement of the speech reception threshold.", J Acoust Soc Am **107**, 1671–84.

Wagener, K. C., Brand, T., and Kollmeier, B. (**2006**). "The role of silent intervals for sentence intelligibility in fluctuating noise in hearing-impaired listeners.", Int J Audiol **45**, 26–33.

Wouters, J., Damman, W., and Bosman, A. (**1994**). "Vlaamse opname van woordenlijsten voor spraakaudiometrie (Flemish recording of word-lists for speech audiometry).", Logopedie 28–33.

1. Replace cadeau by kado
2. Replace bureau by buro
3. Replace eigenaresse by eigenares
4. Replace any number of d and t's at the end of a word by a single t
5. Replace ei by ij

TABLE I. Description of regular expressions used for the Dutch LIST and VU sentence test materials

Word (w)	$H(w)$
a	8,587,340,257
z	27,027,081,632
aa	17,174,680,514
zz	54,054,163,264
autocorrection	219,976,507,191
the bigram	120,336,090,499

TABLE II. Example values of $H(w)$

ID	bigram	Hash value
1	the boy	95,540,814,653
2	boy fell	102,798,238,621
3	fell from	113,502,799,457
4	from the	106,245,375,489
5	the window	147,159,170,907

TABLE III. Bigrams and corresponding hash values for "the boy fell from the window"

ID	bigram	Hash value
1	boy fel	88,104,957,853
2	fel from	98,809,518,689
3	the windaw	138,895,929,613

TABLE IV. Bigrams and corresponding hash values for bigrams can be corrected from the user input sentence. (a bigram can only be corrected if it contains a word that is not in the dictionary)

[a]$_1$ [b]$_2$ [c]$_3$ [d]$_4$ [e]$_5$ [f]$_6$ [h]$_7$ [i]$_8$ [j]$_9$ [k]$_{10}$ [l]$_{11}$ [m]$_{12}$ [n]$_{13}$ [o]$_{14}$ [p]$_{15}$ [q]$_{16}$ [r]$_{17}$ [s]$_{18}$ [t]$_{19}$ [u]$_{20}$ [v]$_{21}$ [x]$_{22}$ [y]$_{23}$ [z]$_{24}$ [ch]$_{25}$ [g]$_{26}$ [oe]$_{27}$ [ui]$_{28}$ [aa]$_{29}$ [ee]$_{30}$ [ie]$_{31}$ [uu]$_{32}$ [ng]$_{33}$ [ij ei]$_{35}$ [uw w]$_{37}$ [ou au]$_{39}$ [oa oo]$_{40}$

TABLE V. Graphemes used for correction of Dutch CVC words. Graphemes with the same code are between square brackets and codes are given as subscripts.

Test material	Group	#	Simple	MO-Ac		MOC-Ac		MT-Ac		MO-MOC		ΔSRT (dB)
				Word	Sent	Word	Sent	Word	Sent	Word	Sent	
LIST	1	3280	12.7	1.5	1.3	0.5	0.6	0.4	0.4	3.3	0.7	0.06
LIST	2	1310	26.6	2.3	3.7	1.1	2.2	0.2	0.3	4.3	1.5	0.23
VU (word)	1	3354	6.0	1.5	2.1	0.5	0.7	0.3	0.2	3.4	0.6	0.55
VU (word)	2	1742	16.5	1.7	3.0	1.0	1.9	0.4	0.4	2.9	1.0	0.18
VU (sent)	1	3354	6.0		3.9		3.3		2.9	3.4	0.6	0.66
VU (sent)	2	1742	16.5		6.3		5.4		3.9	2.9	1.0	0.61
				Phon	Word	Phon	Word	Phon	Word	Phon	Word	
NVA	1	756	32.8	1.3	4.0	0.3	0.9	0.1	0.3	3.0	3.0	
NVA	2	600	38.0	3.4	8.8	1.2	3.5	0.2	0.5	7.0	5.7	

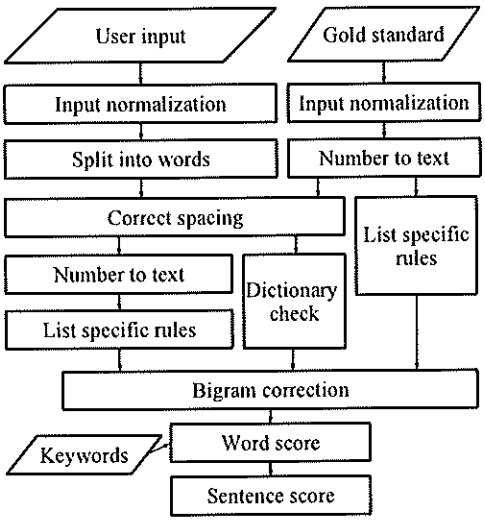
TABLE VI. Percentage of errors made by the autocorrection algorithm for each speech material, group of subjects, number of tokens in the corpus and corpus entry type. For the sentence materials, errors for keyword score and for sentence score are given. For the CVC materials, errors for phoneme score and for word score are given. # is the total number of sentences presented for the sentence tests and the total number of words presented for the CVC test. MO-MOC is the percentage of changes between the MO and MOC scores. ΔSRT is the mean of the differences in estimated SRT (in dB) between Ac and MOC for each condition.

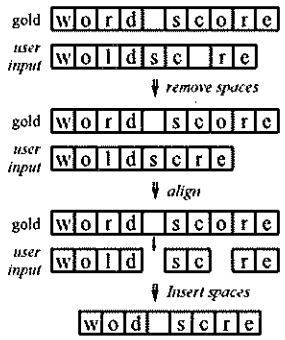
LIST OF FIGURES

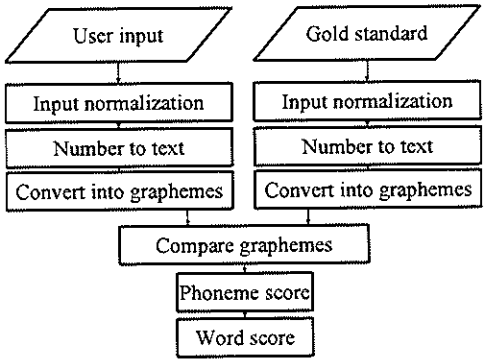
FIG. 1 Flowchart of the sentence algorithm. An arrow signifies that the output from the source block is used as the input for the target block. 7

FIG. 2 Example of string alignment. Spaces are marked by empty boxes. In this case the gold string is "word score" and the user input string "woldsc re". First all spaces are removed from the input string. Then both strings are aligned. The space character marked with the single arrow could be inserted into the input string as shown. However, as the percentage of correctly aligned characters ($100 \cdot 7/10 = 70\%$) is smaller than 90%, no space will be inserted because the strings are in this case not considered sufficiently alike. 11

FIG. 3 General structure of the word correction algorithm. 13


User input
Gold standard
Input normalization
Input normalization
Split into words
Number to text
Correct spacing
List specific rules
Number to text
Dictionary check
List specific rules
Bigram correction
Keywords
Word score
Sentence score


gold
word score
user input
wold sc re
remove spaces
gold
word score
user input
wold scre
align
gold
word score
user input
wold sc re
Insert spaces
wod scre


User input
Gold standard
Input normalization
Input normalization
Number to text
Number to text
Convert into graphemes
Convert into graphemes
Compare graphemes
Phoneme score
Word score