Robust Tracking for Automatic Reading Tutors

Emre Yılmaz, Dirk Van Compernolle, and Hugo Van hamme

Dept. ESAT, KU Leuven, Leuven, Belgium

{emre.yilmaz, dirk.vancompernolle, hugo.vanhamme}@esat.kuleuven.be

Abstract

Reading tutor software uses automatic speech recognition technology to support children in developing their reading skills. In many forms of exercise and evaluation, tracking the reading position is a relevant task or even a prerequisite, e.g. to provide assistance on the pronunciation of a word or to advance the screen to the next page. In this paper, we introduce a new robust tracking algorithm, which measures the similarity between the recognized phones and the phonetic transcription of words displayed on a screen using an efficient dynamic programming algorithm. The criteria for accepting a word reading attempt and thus advancing the cursor can hence be expressed phonetically. In addition, the most likely state of the Hidden Markov Model (HMM) used to decode the speech serves as a fallback for cases of phone matching failure. The new tracker's performance is compared with two other trackers which use either the most likely HMM state or phone matching. The evaluation metrics quantify both the frequency of timely movements and loss of tracking synchronicity. The proposed approach performs significantly better than the others achieving a Timing Accuracy of Tracking of 81.03% compared to 50.63% of the phone matching approach and 32.36% of the state-based approach.

Index Terms: automatic reading tutors, reading position tracking, tracking evaluation

1. Introduction

Automatic reading tutors (ART) are valuable tools to assess and improve the reading levels of elementary school children [1, 2, 3, 4, 5]. They are also used in the diagnosis and treatment of reading difficulties such as dyslexia. As the conventional methods require considerable time and effort, reading tutors with automated reading assessment have become more viable in recent years. Moreover, automated reading assessment does not suffer from observer bias which is a serious problem in conventional methods.

Developing an ART involves several tasks such as tracking the reading position, detecting and classifying the reading miscues, providing help when the reader struggles with a word and generating feedback on the reading during or immediately after the task. The work presented in this paper is mainly related with tracking the reading position. This involves highlighting the word that should be read and advancing the screen automatically for tasks that are presented on consecutive screens. The tracking process can be considered as a remedial tool since it guides the reader by setting a target word to read and stimulates the continuity of reading by automated screen advancement. Tracking is also required to provide help initiated by the system. The baseline reading tutor tracker is developed as a part of the SPACE project. A detailed explanation is provided in [1]. Alternative tracking algorithms developed for ARTs can be found in [6] and [7]. Several metrics for evaluating the tracking accuracy are presented in [8].

The tracking task in an ART is quite challenging, since the tracker has to follow the reader without affecting reading speed. Moreover, a decision has to be made when a reading attempt is good enough (even if it is wrong) to advance the cursor. If the cursor moves to the next word even for the most incomplete reading attempts, it will have a jumpy behavior which may distract the child. If the word remains highlighted after a valid reading attempt, the child might keep reading the same word which affects assessment accuracy. In this work, we propose a phone matching-based criterion to cope with mistimed movements. This criterion for evaluating how good the partial match between the utterance and phonetic transcription of the expected words imposes at least one vowel in a valid reading attempt taking into account that some phones are more easily inserted due to noise.

This paper is organized as follows. Section 2 gives a detailed explanation of the tracker which is based on the most likely HMM state as well as of the proposed phone matching (PM) tracker. Section 3 discusses the development of the evaluation tools. The results are given in Section 4. Section 5 concludes the paper.

2. Tracking Algorithms

The trackers presented below use the information generated by the recognizer to estimate the current reading position. We now give a top-level description of the recognizer (described in [9]). The expected utterance is modeled using word-level and sentence-level finite-state transducers (FST). Each state in the sentence-level FST is associated with a word in the reading task and these states are connected with several arcs corresponding to a correct reading, (multiple) word skips, (multiple) word recaptures and inserted speech. The states in the word-level FST correspond to phones and contain arcs for correct pronunciation as well as for retries. Finally, the phone models are HMMs, such that likelihoods can be generated for each input signal frame. A single pass Viterbi beam search then finds the best path through the composition of sentence-level FST, word-level FST and HMM. At regular time intervals (150 ms), the search engine infers the most likely HMM state. Since this state belongs to some phone and some word, it can also produce an estimate of the phone and the word currently read (though this is theoretically not necessarily the best candidate in the maximum likelihood sense). This will be called "the recognizer's estimate". In addition, the optimal path to reach this state can be computed. It is split in two parts: an initial part that is common to all backtrace paths ending in any state with a likelihood within the search beam (henceforth "the stable path") and which is followed by the state-specific part ending in the winning state (henceforth "the most likely hypothesis").

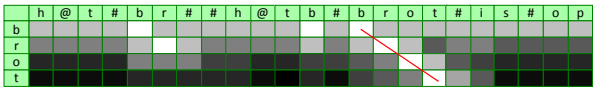

Figure 1: One way of performing search for the target transcription anywhere in the recognized phone string.

2.1. State-based algorithm

The state-based tracker estimates the current reading position by using the recognizer's estimate. The cursor is advanced if the recognizer's estimate is in a word or silence beyond the current cursor position. The cursor is never moved backwards. This simple approach works well when the recognizer detects silence in between the words. For other cases, the cursor is always late as the recognizer's estimate is updated after the child starts reading the next word. Moreover, abrupt changes in the recognizer's estimate either due to reading miscues or hesitant speech may result in an early cursor movement.

2.2. Simplified PM algorithm

A more robust criterion for the acoustic match is proposed to reduce the number of mistimed movements and mitigate the effect of recognition errors. Rather than directly relating the cursor to the recognizer's estimate, we propose a new framework based on deciding how close the uttered phones and phonetic transcription of the expected words match. The comparison is made by calculating the distance between the phonetic transcription of the expected word (the target transcription) and the recognized phone string (the phones in the stable path and in the most likely hypothesis). The latter may also contain phones belonging to the preceding words and unacceptable variants of the target transcription. The proposed distance measure returns a cost depending on the number of deletions, insertions and substitutions. An example is given in Figure 1. The Dutch word *brood* (bread) with a phonetic transcription *brot* is aligned with the recognized phone string, *h@t#br##h@tb#brot#is#op*. Here, the YAPA phonetic symbol set is used to represent the phonetic symbols [10]. # refers to silence. Our goal is to find the target transcription *brot* anywhere in the recognized phone string. In the first row and column of Figure 1, the recognized phones and the target phones are listed respectively. Darker tones refer to larger costs as explained below. The costs at the last row reveal the degree of the match anywhere in the recognized phone string. This string matching is realized using a dynamic programming (DP) algorithm with a careful initialization such that a match can start anywhere in the first row. For this purpose, the cost function $q(t,r)$, is defined as

$$q(t,r) = \begin{cases} c(t,r) & \text{for } t = 1 \\ min(q(t,r-1)+IC, q(t-1,r)+ & \text{otherwise.} \\ DC, q(t-1,r-1)) + c(t,r) & \end{cases}$$

Here, t and r are phone indices in the target transcription and recognized phone string respectively. Horizontal moves are penalized with an insertion cost (IC) and vertical moves are penalized with a deletion cost (DC). $c(t,r)$ is the local cost which is set to a match cost if the r^{th} element of the recognized phone string and t^{th} element of the target transcription are the same, or set to a substitution cost otherwise. The proposed initialization additionally avoids that the costs would grow without bound with the recognized phone string length.

The main aim of this DP algorithm is to estimate the right instant to move the cursor irrevocably rather than finding the optimal path as in the global alignment or the shortest path problem. Therefore, the costs can be calculated by updating a single distance vector. Storing the previous distance vector, $q(t,r-1)$, and the current local cost vector, $c(t,r)$, is adequate to obtain the current distance vector, $q(t,r)$.

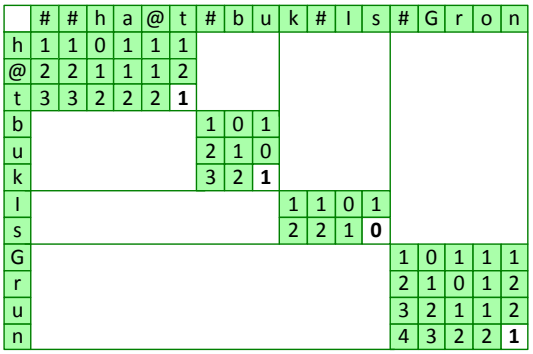

Figure 2: Evolution of the distance vector for the Dutch sentence *Het boek is groen.*

The total distance, which is equal to the last element of the distance vector, is compared with a threshold in order to accept or reject a reading attempt. The threshold has a profound impact on the jumpiness of the tracker and a significant improvement on the timing accuracy can be seen when it is well-chosen. We set the threshold depending on the number of phones in the target transcription (N):

$$\text{threshold} = \begin{cases} \text{An exact match} & \text{for } N = 2 \\ \text{Levenshtein distance} \leq 1 & \text{otherwise.} \end{cases}$$

This threshold is suitable for reading tasks that mostly consist of short words. As a result of this choice, the cursor does not wait for a complete pronunciation of words with three or more phones. Moreover, recognition errors and reading miscues can be handled to some extent.

In practice, the recognized phone string is updated every 150 ms in case of a change in the most likely hypothesis and a comparison is made between the strings using the proposed DP algorithm. The distance vector is updated after each phone in the recognized phone string. The insertion, deletion and substitution costs are set to 1 while the match cost is set to 0. We further added the "recognition of the first vowel" condition to cope with the phones that are more easily inserted due to noise. The cursor moves to the next word if the total distance is less than or equal to the threshold and the first vowel in the target transcription is recognized. After every movement, the phones in the stable path are removed so that possible confusions with the earlier utterances of the same word are avoided. Similar to the state-based tracker, the simplified PM tracker moves only forward. If the child rereads or skips a word, the cursor is immobilized until the marked word is read.

To illustrate the use of the proposed distance measure, an example for an arbitrary utterance of the Dutch sentence *Het boek is groen* (The book is green) is given in Figure 2. In this example, the recognized phone string is *##ha@t#buk#Is#Gron* and it is given in the first row. The phonetic transcription of the given sentence is *h@t buk Is Grun* and it is given in the first column. The evolution of the distance vector over time is given in the columns. Each white cell indicates the movement of the cursor. After each movement, the length of the distance vector is altered according to the new target transcription and the values are reset. For the target transcription *h@t*, the recognized

phones are *ha@t*. The cursor moves to *boek* after *t* is recognized as the last element of the distance vector is equal to 1 (as there is only an insertion). The recognized phones and the target transcription match for the word *boek*. The cursor moves to *is* before the last phone, *k*, is recognized. The third word *is* consists of two phones only. After an exact match is detected, the cursor moves to *groen*. *Gron* is a valid reading attempt as there is only a substitution. The cursor moves after *n* is recognized.

2.3. Advanced PM algorithm

Tracking based on the PM gets stuck easily due to poor reading or skipping a word. The recognizer's estimate is therefore also used to compensate for the cases where a similarity between the recognized phone string and the target transcription cannot be detected due to erroneous reading or recognition. Every 150 ms, the recognizer's estimate is checked. When the recognizer's estimate is the same word located beyond the current reading position three times successively (corresponds to a duration of 450 ms), the cursor moves to the word coming after the recognizer's estimate. This condition for the duration increases the robustness against abrupt changes in the recognizer's estimate. In this way, the cursor keeps following the reader even though it loses track of the reading position.

A tracking scheme is adopted defining the behavior of the cursor in different situations. For the ordinary case, i.e. when the distance is less than or equal to the threshold, the cursor moves to the next word motivating the reader to proceed as in the simplified version. In case of a failure in the PM, the tracker is triggered by the changes in the recognizer's estimate. The cursor jumps twice or more aiming to compensate for the delay.

3. Evaluation and Results

To the authors' knowledge, the tracking error rate [6] and the perceived tracking error rate [8] are the only measures defined so far to evaluate tracker performance. These measures evaluate the timing implicitly in terms of insertions, deletions and substitutions. However, the distribution of time differences between the ideal movements and tracker results better quantifies the perceived timing accuracy. Thus, new performance measures are defined. The ideal position of the cursor is estimated using the information provided in the CHOREC database [11] containing reading sessions from 400 Dutch speaking elementary school children. We used 279 reading sessions where the children read 9 graded text stories ranging from 103 to 223 words in length. This database is manually annotated with various information including the word boundaries and the information about the reading miscues. The instants where the cursor should move are estimated using the word boundaries. For the segments with correct pronunciations, the boundary between that segment and the next segment is estimated to be the ideal instant. For incorrect pronunciations, it has to be decided whether the incorrect pronunciation can be counted as a valid reading attempt. For this purpose, we used the same criterion discussed in Section 2.2 to compare the target transcription with the manually verified phonetic transcription. If there is no valid reading attempt of a word, the start of the segment belonging to the next word is estimated to be the ideal instant. Finally, the difference between the tracker results and ideal instants is calculated and the distribution of the time differences is analyzed to visualize the timing performance of each tracker.

Table 1: Evaluation criteria

Evaluation	Difference between tracker and ideal instants
Very late	$t_{trac} - t_{ideal} > 0.5s$
Late	$0.5s > t_{trac} - t_{ideal} > 0.1s$
OK	$0.1s > t_{trac} - t_{ideal} > -0.3s$
Early	$-0.3s > t_{trac} - t_{ideal} > -0.5s$
Very early	$-0.5s > t_{trac} - t_{ideal}$

Table 2: Comparison of the TAT, STAT>50% and STAT>70% values

Algorithm	TAT	STAT>50%	STAT>70%
State-bas.	3457/10683 (0.32)	25/279 (0.09)	3/279 (0.01)
Simp. PM	5409/10683 (0.51)	154/279 (0.55)	135/279 (0.48)
Adv. PM	8656/10683 (0.81)	269/279 (0.96)	244/279 (0.87)

3.1. Evaluation tools

The timing accuracy of the cursor movements is labeled according to the categories given in Table 1. The ideal instants are compared with the tracker movement from the same word to the next. The accuracy categories are chosen asymmetrically. The lower bound of the acceptable range (evaluated as 'OK') is kept relatively larger in absolute value. This is reasonable as the ideal instants of the movement are obtained using the word boundaries and the tracker should not wait until a word is completely pronounced before moving. Thus, a difference of -0.3 seconds does not necessarily mean that it is an early movement.

We define a performance measure called Timing Accuracy of Tracking (TAT) which is the ratio between the number of well-timed movements and all movements which are categorized according to Table 1.

$$\text{TAT} = \frac{\text{OK}}{\text{VL} + \text{L} + \text{OK} + \text{E} + \text{VE}} \quad (1)$$

VL, L, OK, E and VE are the total number of the words classified as very late, late, OK, early and very early respectively. The TAT within each session (STAT) is also calculated and used to evaluate the accuracy in each session. STAT>50% and STAT>70% are defined as the number of reading sessions having a STAT that is higher than 50% and 70% respectively. The number of very late (VL) and very early (VE) movements are used as measures of jumpiness. A high VL value implies that the tracker is rather immobile and tends to get stuck and a high VE value indicates that the tracker is very jumpy which can be distracting.

3.2. Results and discussion

The histograms illustrating the distribution of the time differences between the tracker results and the ideal instants are given in Figure 3. As seen in Figure 3a, the peak is shifted towards the right as the state-based tracker moves slightly later compared to the PM trackers. Furthermore, more outliers (either very late or very early movements) are observed in the distribution of the state-based tracker. The total number of samples for the simplified PM algorithm is much smaller as it often gets stuck before tracking up to the last word.

In Table 2, the timing accuracy related measures, namely the TAT, STAT>50% and STAT>70%, are given for each tracker. We are mainly interested in the first and the third rows of the table. The TAT results are presented in the first column. The advanced PM algorithm outperforms the state-based

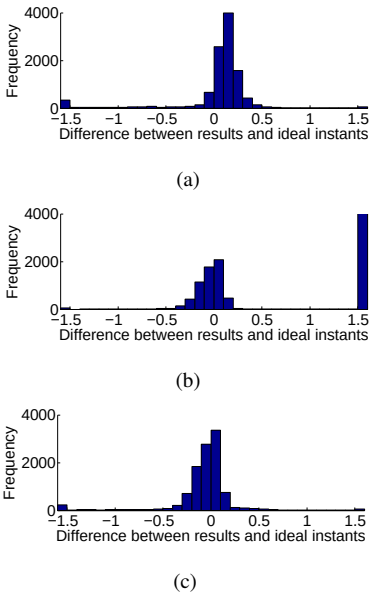

Figure 3: Distribution of the difference between the tracker results and ideal instants for the state-based (a), simplified PM (b), and advanced PM (c) trackers.

Table 3: Comparison of the VL and VE values

Algorithm	VL	VE
State-bas.	202/10683 (0.019)	768/10683 (0.072)
Simp. PM	4229/10683 (0.396)	145/10683 (0.014)
Adv. PM	189/10683 (0.018)	504/10683 (0.047)

one achieving 81.03% compared to 32.36%. In the next two columns, the STAT>50% and STAT>70% values are given. The advanced PM algorithm achieves good timing in at least 70 percent of the words (STAT>70%) in 244 of 279 sessions. The number of sessions with a STAT>70% is only 3 in the state-based algorithm.

Table 3 presents the VL and VE values. For the simplified PM algorithm, all of the words that are not highlighted (as the cursor gets stuck at a previous word) are counted as very late movements. The large difference between the VL values of the simplified PM and the other trackers justifies using the recognizer's estimate. The advanced PM tracker has slightly less VL and VE values than the state-based tracker. Thus, the advanced PM tracker moves more timely and has a better jumpiness as the tracking speed is closer to the actual reading speed compared to the state-based tracker.

The performance of both the state-based and advanced PM trackers is also evaluated in an online system. The tracking accuracy and jumpiness of the advanced PM tracker are perceived to be better not only for correct readings but also for readings with many skips, recaptures and hesitations.

4. Conclusions

In this paper, a new tracker for automatic reading tutors is discussed and compared with a simple state-based tracker. Tracking is a key task in a reading tutor as it guides the child by highlighting the target word and stimulates the continuity of reading by automatic screen advancement. This is achieved by using information generated by the speech recognizer. The state-based tracker uses the recognizer's estimate of the current reading position. However, this tracking approach responds late unless the words are separated by silence. Furthermore, sudden changes in the recognizer's estimate due to reading miscues result in a misleading jumpy behavior of the tracker. To alleviate these problems, we propose a tracking algorithm based on the comparison of the recognized phones and phonetic transcriptions of the expected words. The recognizer's estimate is also used to improve the tracking especially for cases of the tracker failing to find a sufficiently close *local* match between the expected and recognized phones due to recognition errors or reading miscues. Because of the *global* sentence model, this state machine can cope with many types of reading miscues, which explains the success of the combined method. The timing accuracy of tracking (TAT) and other measures are defined to evaluate tracker performance. The advanced PM algorithm performs significantly better than the state-based algorithm with a TAT of 81.03% compared to 32.36%.

5. Acknowledgments

This work has been supported by the Fund for Scientific Research Flanders, FWO, under grant G.0260.07 (TELEX).

6. References

[1] J. Duchateau, Y. O. Kong, L. Cleuren, L. Latacz, J. Roelens, A. Samir, K. Demuynck, P. Ghesquiere, W. Verhelst, and H. Van hamme, "Developing a reading tutor: Design and evaluation of dedicated speech recognition and synthesis modules," *Speech Communication*, vol. 51, no. 10, pp. 985–994, October 2009.

[2] J. Mostow, S. Roth, A. Hauptmann, and M. Kane, "A prototype reading coach that listens," in *Proc. of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, USA, 1994, pp. 785–792.

[3] A. Hagen, B. Pellom, and R. Cole, "Children's speech recognition with application to interactive books and tutors," in *IEEE Automatic Speech Recognition and Understanding (ASRU) Workshop*, St. Thomas, USA, 2003, pp. 186–191.

[4] X. Li, L. Deng, Y. C. Ju, and A. Acero, "Automatic children's reading tutor on hand-held devices," in *INTERSPEECH*, Brisbane, Australia, September 2008, pp. 1733–1736.

[5] M. P. Black, J. Tepperman, and S. S. Narayanan, "Automatic prediction of children's reading ability for high-level literacy assessment," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 1015–1028, May 2011.

[6] S. Banerjee, J. Mostow, J. Beck, and W. Tam, "Improving language models by learning from speech recognition errors in a reading tutor that listens," in *Second International Conference on Applied Artificial Intelligence*, Kolhapur, India, December 2003.

[7] A. Hagen, B. Pellom, and R. Cole, "Highly accurate children's speech recognition for interactive reading tutors using subword units," *Speech Communication*, vol. 49, no. 12, pp. 861–873, October 2007.

[8] M. H. Rasmussen, J. Mostow, Z. H. Tan, B. Lindberg, and Y. Li, "Evaluating tracking accuracy of an automatic reading tutor," in *Speech and Language Technology in Education (SLaTE)*, Venice, Italy, August 2011, pp. 17–20.

[9] J. Duchateau, M. Wigham, K. Demuynck, and H. Van hamme, "A flexible recognizer architecture in a reading tutor for children," in *Proc. of the ITRW on Speech Recognition and Intrinsic Variation*, Toulouse, France, May 2006, pp. 330–331.

[10] P. Mertens and F. Vercammen, "Fonilex manual," Tech. Rep., March 1998.

[11] L. Cleuren, J. Duchateau, P. Ghesquire, and H. Van hamme, "Children's oral reading corpus: Description and assessment of annotator agreement," in *Proc. 6th International Conference on Language Resources and Evaluation*, Marrakech, Morocco, May 2008.