Evaluating the Effect of Predicting Oral Reading Miscues

Satanjeev Banerjee, Joseph E. Beck, and Jack Mostow

Project LISTEN (http://www.cs.cmu.edu/~listen)
School of Computer Science, Carnegie Mellon University
{satanjeev.banerjee,joseph.beck,mostow}@cs.cmu.edu

Abstract

This paper extends and evaluates previously published methods for predicting likely miscues in children's oral reading in a Reading Tutor that listens. The goal is to improve the speech recognizer's ability to detect miscues but limit the number of "false alarms" (correctly read words misclassified as incorrect). The "rote" method listens for specific miscues from a training corpus. The "extrapolative" method generalizes to predict other miscues on other words. We construct and evaluate a scheme that combines our rote and extrapolative models. This combined approach reduced false alarms by 0.52% absolute (12% relative) while simultaneously improving miscue detection by 1.04% absolute (4.2% relative) over our existing miscue prediction scheme.

1. Introduction

Project LISTEN's Reading Tutor helps children in grades 1 – 4 (typically ages 6 – 10) learn to read by using automatic speech recognition (ASR) to listen to them read aloud and giving spoken and graphical feedback. It displays text one sentence at a time, called the *target sentence*, and uses speech recognition to detect reading mistakes, called *miscues*. Many miscues are *real word miscues* where the child replaces the correct (*target*) word with a different word [1]. It is important for the Reading Tutor to detect miscues but also to avoid flagging correctly read words as miscues and providing negative feedback on them.

Unlike systems [3] where children read one word at a time, Project LISTEN's Reading Tutor allows children to read the whole sentence. But children skip words, jumping forward or back in the sentence. This phenomenon rules out using forced alignment of the utterance against the target text as a way to detect miscues. Miscues might be detected by using an all-phone decoder to find sequences of phonemes that do not match any of the target words (as in [4]), or by using confidence metrics (as in [5]). However none of these methods utilize knowledge about the kinds of real word miscues that students make. Previous work [1] utilized such knowledge to predict words that a student is likely to substitute for the target word. The goal of this paper is to evaluate the effect of such predicted miscues on recognition accuracy.

2. Experimental Data

Previous work [1, 2] in predicting likely miscues used a database of oral reading miscues developed by Richard Olson, Helen Datta, and Jacqueline Hulslander at the University of Colorado. This database contains over 100,000 transcribed oral reading miscues uttered by 868 children mostly aged 8 to 12 years.

To evaluate our miscue prediction algorithms, we use children's recorded utterances collected during the 2001-2002 school year by 99 Reading Tutors deployed at 8 schools in the area of Pittsburgh, PA, and one school in North Carolina. Each utterance is a single attempt of a student to read all or part of a sentence. Our speech data analyst, John Helman, has transcribed 11,797 of these utterances. Words are transcribed orthographically and non-words phonetically. For example, an imperfect reading of the target sentence "dreams about competitors" would be transcribed as "deems about /K AA M P IY T AX R Z/" because "deems" is a real word, whereas "/K AA M P IY T AX R Z/" does not represent a real word.

The 11,797 transcribed utterances were spoken by a total of 125 children. For this paper we use data from 50 children as a training set and from 25 other children as an independent test set, leaving untouched data from 50 students for further experimentation later on.

3. Detecting Miscues

The Reading Tutor's goal is to detect miscues, not necessarily identify them. For example, in Table 1, the target sentence was "little deems my royal dame" which the child read as *little dreams my real dame*. The ASR output hypothesis is `little deems my deems day`. We align both the transcript and the recognizer's output hypothesis to the target text, and then find the miscues that the recognizer correctly detects. These three token sequences are aligned as shown. ASR correctly detects that the child has misread the word "royal" (although it incorrectly identifies "deems" as the word the child uttered instead). Simply aligning the hypothesis to the transcript and reporting word error rate would not give credit for detecting such a miscue. ASR fails to detect that the child misread the word "deems".

Table 1: *Alignment of transcript, target text, and hypothesis.*

Transcript	*little*	*dreams*	*my*	*real*	*dame*
Target	little	deems	my	royal	dame
Hypothesis	`little`	`deems`	`my`	`deems`	`day`
		miscue undetected		miscue detected	false alarm

4. Evaluation Metrics

To evaluate the performance of a miscue detector, we define *miscue detection rate* as the number of miscues detected divided by the total number of miscues the child made. We define *false alarm rate* as the number of text tokens erroneously flagged as read incorrectly, divided by the total number of text tokens the child read correctly. In the example in Table 1, the number of miscues is 2 (*dreams* for "deems" and *real* for "royal"), one of which is detected. Thus the miscue detection rate is 50%. The number of false alarms is 1 (`day` for "dame") out of three correctly read words, giving a false alarm rate of 33%. We seek to detect more miscues with fewer false alarms. As in [7], we reduce our false alarm rate by half by ignoring miscues when the target word is any of 36 function words (e.g. *a*, *the*, etc.) on which miscues seldom affect comprehension. In our data, 5981 content and 3166 function tokens were read correctly, and 335 content and 147 function tokens had miscues.

5. Experimental Methodology

We compare various methods for miscue prediction in the following way. For each target sentence, we construct its language model as described in [7]. Next we predict miscues for each content word in the sentence and add them to the lexicon. Next we run the speech recognizer, and align its hypothesis against the target text to detect miscues. In this paper we evaluate the miscue detection and false alarm rate of two approaches reported in [1], as well as of a combined algorithm that we propose here.

For a baseline, we report the miscue detection rate and false alarm rate when no predicted miscues are used. That is, for each utterance the recognizer listens only for those words that are already in the sentence, and detects a miscue only if it sounds more like some other word in the sentence than like the target word. Row 1 of Table 2 shows the miscue detection rate and the false alarm rate for this method. Of the 562 text words the children misread, 21.58% were detected, all of which were content words (since miscues on function words are ignored), while of the 9477 sentence words read correctly, 2.42% were incorrectly flagged by the recognizer as having been read incorrectly.

6. Rote Prediction of Miscues

The rote method described in [1] predicts that miscues that at least two children have made in the past will be repeated by other students in the future. This method utilizes the Colorado miscue database to find such miscues. Our procedure for selecting rote miscues differs from [1] in the following ways. First, in [1], miscues were restricted to having the same first letter as the sentence word. For example, for the target word "accord" we would predict miscues like "accords" but not "cords". We now allow distractors to start with a different first letter than the sentence word. Second, we consider as possible distractors only the 25,000 most frequent words in our corpus of children's stories. Although it is possible for a student to say an obscure word such as "enervate," it is unlikely. Therefore, we exclude such words from the rote model.

For each target word, we sort the miscues recorded in the Colorado database according to the number of students who made that particular error. Our intuition is that the more students who made the same mistake, the more likely it is that other students will too. We select miscues for target words from this database in two ways – for each target word, we either pick the top m ranked miscues, or we pick all miscues that at least n students uttered.

Table 2 shows the miscue detection rates and false alarm rates for the various threshold values we tested. This table is sorted on false alarm rate. Observe that "at least 10 students" - the situation in which miscues are selected for a word only if at least 10 students uttered the same miscue - performs worse (lower miscue detection, higher false alarm rate) than the simpler choice of taking only the most frequent miscue for each target word. Similarly, observe that "at least 5 students" performs worse than simply picking the top 3 most frequent miscues. This trend suggests that in general picking a constant number of miscues performs better than relying on the number of students who uttered the miscue. We conjecture that this is because the latter method predicts a large number of distractors for the common words, leading to a higher false alarm rate.

Table 2: *Rote model performance*

Model type	Miscue detection rate (in %)	False alarm rate (in %)
Baseline	21.58	2.42
Top 1	22.82	2.88
At least 10 students	22.61	3.04
Top 2	24.90	3.27
Top 3	25.73	3.54
At least 5 students	24.9	3.62
Top 4	26.35	3.86
Top 5	26.14	4.02
All literal miscues	27.39	5.12

7. Extrapolative Prediction

[1] noted that the rote method does not perform very well on rarer words, and therefore described an extrapolative method that uses a classifier learning approach to generalize from the miscues in the Colorado miscue database. In this paper we reuse a few features and introduce a few new features. In particular we reuse:

- Frequencies of the target word and the miscue word in English.
- Frequency of the miscue word in the corpus of miscues.
- Edit distance between the spelling of the target word and the miscue word.
- Phonemic edit distance between the target word and the miscue word.
- Phonemic edit distance between the target word and the miscue word, normalized by the number of phonemes in the target word.
- A feature that has a value of 1 if the target word and the miscue word both start with the same phoneme, and 0 otherwise.

We removed the following features from our model because they had little predictive power:

- Grade level the student is at.
- A feature that has a value of 1 if the target word and the miscue word both end with the same phoneme, and 0 otherwise.

We added the following new features:

- Edit distance between the spelling of the target word and the miscue word normalized by the length of the spelling of the target word.

- A feature that has a value of 1 if the target word and the miscue word both start with the same letter, and 0 otherwise.
- A feature that has a value of 1 if the miscue word is a substring of the target word, and 0 otherwise.
- The position of the miscue word as a substring of the target word (e.g. position of “where” in “nowhere” is 3), or 0 if the miscue word is not a substring of the target word.
- The position of the target word as a substring of the miscue word, or 0 if the target word is not a substring of the miscue word (e.g. “nowhere” is not a substring of “where”, and so gets a value of 0.).
- A feature that has a value of 1 if the target word is a substring of the miscue word, and 0 otherwise.

As in [1], positive examples of miscues are taken from the Colorado miscue database, while negative examples are words that are among the 25,000 most frequent words in English, are within a spelling edit distance of 3 and normalized phonemic distance of 3.75 from the target word (where distance between two maximally different phonemes is defined as 5), and do not belong to the positive examples.

In [1], logistic regression was used as a classification technique. Using the features described above, and testing on a separate set of students than those used to build the models, logistic regression had an 86.5% classification accuracy for distinguishing real miscues that students actually said from the negative examples created as described above. LogitBoost had a classification accuracy of 90.2%. Therefore, we used LogitBoost for this round of experiments. We used the default settings for LogitBoost in Weka [6]: 10 iterations, with 100% of the data kept across boosting rounds.

The classifier takes as input a target word and a miscue word, and outputs an estimate of the probability that the miscue is uttered instead of the target word. For a single target word, we sorted all its candidate miscues according to this probability estimate, and then selected miscues from this sorted list in one of two ways, either picking the top n miscues, or picking all miscues above a fixed probability threshold. Table 3 shows the performance of miscues picked using each of these various selection criteria, where the rows are sorted according to the false alarm rate. Observe once again that the simple algorithm of picking the single most probable miscue strictly outperforms picking miscues that have a probability value of more than 0.95 on both miscue detection rate as well as miscue detection rate. Similarly “top 2” strictly outperforms “$p \geq 0.9$”.

To evaluate whether using the information about real word miscues through the various features described above helps the classifier pick useful miscues, we ran an experiment where for each target word we picked its “worst 2” miscues, that is, the two miscues that the classifier gave the lowest probabilities of being likely miscues of the target word. Observe from rows 4 and 5 of Table 3 that the “top 2” had a false alarm rate that was 0.19% worse than the “worst 2” but 1.66% better in miscue detection rate absolute. We also wanted to see how much of the false alarm and miscue detection rate improvements over the no–distractor baseline can be attributed to just the effect of having any distractors in the language model. Accordingly, we predicted two random words as miscues for each target word. Row 7 of Table 3 shows that this “random 2” experiment performed pretty poorly and that the “top 2” algorithm was strictly better. This result illustrates that the knowledge that goes into building the distractor sets is indeed yielding better false alarm and miscue detection rates.

Table 3: *Extrapolative model performance*

Model type	Miscue detection rate (in %)	False alarm rate (in %)
$P \geq 0.99$	23.03	2.77
Top 1	25.52	3.59
$P \geq 0.95$	24.48	3.67
Worst 2	27.80	4.01
Top 2	29.46	4.20
$P \geq 0.9$	28.63	4.53
Random 2	25.93	4.83
Top 3	32.78	5.16
Top 4	34.44	6.01
Top 5	35.48	6.40
$P > 0.7$	40.25	9.15

8. Combining Rote and Extrapolative Prediction

An obvious next step is to combine these two methods to take advantage of their different strengths. In our evaluation of the two methods above we observed that simply ranking predicted miscues and predicting a fixed number of the most likely miscues for the target word performs better than thresholding. One simple way of combining these two methods is as follows. Assume we wish to find at least n miscues for a given target word. We use the top n most likely miscues for this word as predicted by the rote method. If the rote method predicts fewer than n miscues, we make up the difference by adding miscues predicted by the extrapolative method. Observe that this method gives preference to miscues predicted by the rote method in that miscues predicted by the extrapolative method are resorted to *only* when the rote method does not predict enough miscues. The intuition is that we use the more accurate predictions of the rote method for high frequency words, and the predictions of the extrapolative method for low frequency words for which the rote method may not predict any miscue due to a lack of sufficient training data. Table 4 shows miscue detection rates and false alarm rates obtained by the combined miscue prediction algorithm.

Table 4: *Combined model performance*

Model type	Miscue detection rate (in %)	False alarm rate (in %)
Cheating	42.53	2.90
Top 1	25.73	3.77
Truncation	24.69	4.29
Top 2	31.54	4.62
Top 3	33.61	5.54
Top 4	34.85	6.25
Top 5	36.72	6.96

To establish an upper bound on miscue detection performance we performed the following “cheating” experiment. For each target word in the test data we predicted only those words as miscues that the child actually uttered according to the transcripts of the test data, thereby creating a set of “perfect” distractors. The first row of table 4 lists the miscue detection rate and false alarm rate of this experiment. 30.5% of all miscues occur on function words that we ignore as mentioned before. Thus the cheating experiment successfully detects 61.2% of miscues on content words, and fails to detect the remaining miscues despite a "perfect" set of distractors. A visual inspection of 10 random instances of undetected miscues showed that very often the target word was phonetically very close to the transcribed miscue, e.g. "fog" and "frog", "can" and "can't", "set" and "sent", and ASR chose to recognize the correct word because its language model was biased towards accepting correct reading over incorrect reading. The low false alarm rate of this experiment is due to the fact that there was only a very small set of distractors that ASR was listening for, leading to less chance of hallucinating miscues.

Currently the Reading Tutor has a simple approach to modeling students' mispronunciations. In this approach, described in [7], *truncations* are defined for each target word as prefixes of the phoneme sequence that constitutes the pronunciation of the target word. All such truncations are listened for by the recognizer, except for the truncation that represents only the first phoneme of the target word (because a single phoneme "word" is easy to hallucinate) and the truncation that is one short of the full word itself (because it is too easily confusable with the target word itself). For example, the target word "abolish", whose pronunciation is /AX B AA L IX SH/, has the following truncation distractors: /AX B/, /AX B AA/, and /AX B AA L/. The intuition behind constructing such distractors was that they model false starts in children's reading. Moreover this method is simple and easy to implement.

To compare our new methods with the old truncation method, we computed false alarm rates and miscue detection rates using the truncation distractors, shown in the second row of Table 4. Observe that the "top 1" algorithm does better than the truncation algorithm in both false alarm and miscue detection. Thus the new method represents a slight improvement over the current approach in the Reading Tutor.

For all results in tables 2, 3, and 4, the miscue detection rate has a standard error rate of approximately $\pm$ 2% (absolute) while the false alarm rate has a standard error rate of approximately $\pm$ 0.2%. False alarm rates are better estimated than miscue detection rates because there are 5981 correctly read content word tokens, but only 335 miscues. However, this formula makes the simplifying assumption that miscue probabilities are independent, which ignores the fact that miscues are not distributed randomly, but concentrated among some readers and utterances.

9. Conclusions

Detecting miscues for a given target word is important for a reading tutor that listens. However, depending upon the choice of algorithm, listening for the "wrong" set of words may hurt more than help our cause. This paper evaluated the rote and extrapolative methods of miscue prediction, based on their ability to help the Reading Tutor detect student reading mistakes without too many false alarms. We constructed and evaluated a scheme that combines those two methods. This combined approach results in a reduction of false alarms by 0.52% absolute (12% relative) while simultaneously improving miscue detection by 1.04% absolute (4.2% relative) over the Reading Tutor's current truncation-based method of modeling miscues.

10. Acknowledgements

We thank Richard Olson, Helen Datta, and Jacqueline Hulslander for the University of Colorado miscues database, the Weka team at the University of Waikato, New Zealand, for the machine learning software used in the experiments in this paper, and Ted Pedersen at the University of Minnesota Duluth for the classification program WekaClassify.

This work was supported by National Science Foundation under Grant No. REC-9979894. Any opinions, findings, and conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science foundation or the official policies, either expressed or implied, of the sponsors or of the United States Government.

11. References

[1] Mostow, J., Beck, J., Winter, S. V., Wang, S., and Tobin, B., "Predicting oral reading miscues". In *Proceedings of the Seventh International Conference on Spoken Language Processing,* ICSLP'02, Denver, USA. p. 1221–1224.

[2] Fogarty, J., Dabbish, L., Steck, D., and Mostow, J., "Mining a database of reading mistakes: For what should an automated Reading Tutor listen?". In *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future,* Johnson, W. L., Editor. 2001, Amsterdam: IOS Press: San Antonio, Texas. p. 422–433.

[3] Russell, M., Brown, C., Skilling, A., Series, R., Wallace J., Bonham B., and Barker P. "Applications of automatic speech recognition to speech and language development in young children". In *Proceedings of the Fourth International Conference on Spoken Language Processing,* (ICSLP96), Philadelphia, USA, 3-6 October 1996.

[4] Witt, S. M., "Use of Speech Recognition in Computer Assisted Language Learning", PhD Thesis, University of Cambridge, 1999.

[5] Hazen, T. J., Seneff, S., and Polifroni, J. "Recognition confidence scoring and its use in speech understanding systems". In *Computer Speech and Language*, 2002, **16**, 49-67.

[6] Witten, I. H., and Frank, E., *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* 2000: Morgan Kaufmann.

[7] Mostow, J., Roth, S. F., Hauptmann, A. G., and Kane, M., "A prototype reading coach that listens [AAAI–94 Outstanding Paper Award]". In *Proceedings of the Twelfth National Conference on Artificial Intelligence.* 1994. Seattle, WA: American Association for Artificial Intelligence. P. 785–792.