ROVER Enhancement with Automatic Error Detection

Kacem Abida, Fakhri Karray

Department of Electrical and Computer Engineering, University of Waterloo, Ontario, CANADA
mkabida@engmail.uwaterloo.ca, karray@uwaterloo.ca

Abstract
In this paper, an approach is presented to improve the existing performance of the Recognizer Output Voting Error Reduction (ROVER) procedure used for speech decoders’ combination in automatic speech transcription. A contextual analysis is injected within the ROVER process to detect and eliminate erroneous words. This filtering is carried out through the combination of automatic error detection techniques. Experiments showed it is possible to outperform the ROVER baseline, and that combining it with error detection methods leads to an even lower Word Error Rate (WER) in the final ROVER composite output. 

Index Terms: ROVER, LSI, PMI, Error Detection, Combination

1. Introduction
ROVER[1] is one of the most successful recognizers’ combination method used in automatic speech transcription. Since it was first introduced by NIST back in 1997, researchers have been working hard to improve on ROVER baseline performance. However, ROVER’s WER has reached a plateau. In fact, no substantial WER reduction has been observed in the literature. The authors have presented in [2] an approach to solve this problem from a different perspective. The idea is to carry out a semantic analysis during the composite Word Transition Network (WTN) building stage, in order to spot errors. This is done by involving an error detection module within ROVER. Once erroneous tokens have been filtered out, a smoother voting is expected, which will hopefully ensure lower WER. Experiments in [2] highlighted the poor performance of the current error detection techniques, which led to a small WER reduction. This led us to investigate ways to combine error detection techniques in [3]. This paper is a continuity to the work previously described in the sense that it integrates the newly combined and therefore optimized error filtering techniques within ROVER. This paper is organized as follows. Background material is presented in section 2. Our proposed approach is then outlined in section 3, where we present the error detection algorithms and their combination, then we describe the integration within the ROVER process. In section 4, an experimental analysis and interpretations are reported. We conclude the paper with a summary of findings.

2. Background Material
ROVER [1], is a two-step process. First, it combines the multiple outputs into a single, minimal cost WTN. Second, the composite WTN is browsed by a voting process to select the best output sequence. At each slot in the network, a score is computed for each word using the frequency of word, along with its ASR confidence value. In the case of an insertion or a deletion, the NULL transition is used during the WTN building stage. Three voting mechanisms have been presented, one of which, namely Frequency of Occurrence, involves only the use of occurrence information to select the winner. The remaining two voting mechanisms make use of the confidence scores. Even though the WER reduction, achieved by ROVER can be considered outstanding throughout the literature, a few problems remain. In fact, the ROVER approach can only succeed if and only if the errors produced by each ASR system are different from one another. The iterative combination of word transition networks does not guarantee the optimal composite output. In [6], it has been reported that the best results are obtained when systems are ordered by increasing WER. In this research work, we aim at enhancing ROVER through the use of an error filtering stage. The most common non-probabilistic technique is pattern matching. However, this technique suffers from lot of issues such as the inability to cope with unseen patterns [7]. According to [8], transcription errors tend to occur in regular patterns rather than at random. Therefore, given a large training data, collected frequencies can be very well extended beyond the training corpus to cover a whole domain. The most commonly used techniques in this regard are Point-wise Mutual Information (PMI) and Latent Semantic Indexing (LSI)-based techniques. LSI is a powerful tool that has first been tailored to error detection in [4]. PMI-based error filtering technique was first introduced in [5]. LSI is an information retrieval tool that relies on the terms co-occurrences to identify the degree of similarity between them. PMI computes similarity scores between terms using word frequencies from a given corpus. Experiments in [4, 5] showed that it’s possible to achieve high precision, but with low recall ratio. Because of this poor performance, researchers have investigated ways to combine different error detectors to compensate for the low recall ratio. Few attempts already have been successful in combining error detection techniques to improve both precision and recall ratios[3, 4, 7]. Authors in [3] mainly tackled error detection techniques that relies on thresholding a confidence score to decide whether or not a given token from the speech transcription is an erroneous or correct output. In the next section, our approach to augment ROVER with an improved error filtering classifier is described.

3. Proposed Approach
We first present the LSI and PMI-based error detection procedures, then we describe the combination process of these error detectors, along with their integration within ROVER.

3.1. LSI-based Error Detection
The LSI procedure aims at extracting features that highlight the similarities between words. Given a large textual corpus, a term-document matrix is built where rows stand for words,
and columns stand for documents. Singular Values Decomposition (SVD) is applied on the large matrix to reduce dimensionality. Then reliable similarity measures between terms are derived using only these dimensions. The cosine similarity is used to quantify similarities between terms. Two different aggregations have been used to compute the semantic similarity score of a given word in an utterance of length $M$: the mean semantic scoring (MSS), $\text{MSS}_i = \frac{1}{M} \sum_{j=1}^{M} \text{cosine}(w_i, w_j)$ and the mean rank of the semantic scores (MR), $\text{MR}_i = \frac{1}{M} \sum_{j=1}^{M} \text{RANK}(\text{cosine}(w_i, w_j))$. The rank of the semantic score shown in MR is computed as follows. First, the set of semantic scores $L_i$ is computed. $L_i$ is the set of cosine scores between the word $w_i$ and all the remaining words $w_j$ in the corpus. The $\text{MR}_i$ score is then the mean of the rank of each $\text{cosine}(w_i, w_j)$ score in the set of $L_i$. The LSI-based error detection works as follows: given a recognizer’s transcription output, a word is tagged as erroneous if and only if its $\text{MSS}$ (respectively $\text{MR}$) is below a threshold $K$. The error filtering applied on a word $w_i$ in a given transcription output, is detailed in algorithm 1.

**Algorithm 1 LSI-based Error Detection**

1: Compute Cosine scores between $w_i$ and all other words in the transcription output.
2: Compute $\text{MSS}_i$ score or $\text{MR}_i$ score.
3: $w_i$ is an error if $\text{MSS}_i \leq K$ or $\text{MR}_i \leq K$.

The threshold $K$ is to be optimized through a training stage. The higher $K$ is, the more aggressive the error filtering is, and vice versa.

3.2. PMI-based Error Detection

The PMI-based semantic similarity is a measure of how similar and how close in meaning $w_i$ and $w_j$ are. It is computed as follows: $\text{PMI}(w_i, w_j) = \log \left( \frac{P(w_i, w_j)}{P(w_i)P(w_j)} \right)$. In a nutshell, the PMI score is defined as the probability of seeing both words ($w_i$ and $w_j$) together, divided by the probability of observing each of these words separately. Given a large textual corpus with size $N$ tokens, the probabilities can be computed using $P(w_i) = \frac{c(w_i)}{N}$ and $P(w_i, w_j) = \frac{c(w_i, w_j)}{N}$, where $c(w_i)$ and $c(w_i, w_j)$ are the frequency counts collected from the corpus. The process of detecting an error using the PMI-based technique[5] is described in algorithm 2.

**Algorithm 2 PMI-based Error Detection**

1: Identify the neighborhood $N(w)$.
2: Compute PMI scores $\text{PMI}(w_i, w_j)$ for all pairs of words $w_i \neq w_j$ in the neighborhood $N(w)$, including $w$.
3: Compute Semantic Coherence $\text{SC}(w_i)$ for every word $w_i$ in the neighborhood $N(w)$, by aggregating the $\text{PMI}(w_i, w_j)$ scores of $w_i$ with all $w_j \neq w_i$.
4: Define $\text{SC}_{avg}$ to be the average of all the semantic coherence measures $\text{SC}(w_i)$ in $N(w_i)$.
5: Tag the word $w$ as an error if $\text{SC}(w) \leq K.\text{SC}_{avg}$.

In step 3 of the algorithm, the semantic coherence can be computed using different aggregation variants: Harmonic mean: $\text{SC}(w_i) = \frac{1}{N} \sum_{i \neq j}^N \frac{1}{\text{PMI}(w_i, w_j)}$. Arithmetic mean: $\text{SC}(w_i) = \frac{1}{N} \sum_{i \neq j}^N \text{PMI}(w_i, w_j)$, Maximum: $\text{SC}(w_i) = \max_{i \neq j} \text{PMI}(w_i, w_j)$, and Sum: $\text{SC}(w_i) = \sum_{i \neq j}^N \text{PMI}(w_i, w_j)$.

The filtering parameter $K$ is used to control the error detection rate. The higher $K$ is, the more aggressive the error detection, and vice versa. If $K$ is set quite low, more erroneous tokens slip past the detector, and get tagged as correctly transcribed.

3.3. Combination of Error Detection Techniques

The low recall and precision ratios of the current automatic error detection techniques in speech transcription led researchers to investigate ways to improve both of these ratios. The idea is to combine different error detection approaches in the hope that the new technique achieves higher recall, without degrading the precision ratio. The ultimate goal is to be able to improve both ratios simultaneously upon the combination of the error detection techniques. The implicit assumption in this thinking, is that the error detection techniques have to have different performance in terms of these two ratios. In other words, when one approach achieves high recall and low precision, the other techniques to be combined with, needs to achieve high precision and low recall to ensure improvement in both ratios with the new technique. The logic of our proposed approach in [3] is to preserve each technique’s advantage or powerful characteristics in the final combination. Figure 1 describes the flow of the error detection combination approach. The scale of each error detection technique confidence score is different. Therefore a score normalization stage is needed to standardize all confidence score from the various detection techniques to lie between zero and one. To normalize the confidence scores, we use, $X_{\text{scaled}} = \frac{X - \text{min}}{\text{max} - \text{min}}$, where $X$ is the score to be normalized, and $\text{min}$, respectively $\text{max}$, is the minimum, respectively maximum value of the technique’s confidence score. Once all the confidence scores have been normalized, a score combination formula is then applied to build a new score. The classification threshold, $K$, is then applied on this new score to detect erroneous output. Two score combination formulas were used, namely Weighted Average (WA), $\text{Score}_{WA} = \sum_{i=1}^{N} \alpha_i \text{Score}_i$, and Harmonic Mean (HM), $\text{Score}_{HM} = \frac{N}{\sum_{i=1}^{N} \frac{1}{\text{Score}_i}}$, where $\sum_{i=1}^{N} \alpha_i = 1$ The weighting factors play an important role in realizing a trade off between various detection techniques to optimize recall and precision ratios. This coefficients needs to be optimized a priori during training. The algorithm 3 summarizes the combination procedure for $N$ different speech transcription error detection techniques.
In step 4 of algorithm 3, the threshold parameter $K$ is to be optimized through a training stage. It is used to control the error detection rate. The higher $K$ is, the more aggressive the error filtering, and vice versa. If $K$ is quite low, more erroneous words slip past the combined error detector. Besides the WA and HM combination schemes, a direct combination scenario has been attempted. When direct combination method is used, a word is considered erroneous, if at least one of the techniques tags it as an error. This type of combination is beneficial to increase the recall because it tends to classify tokens as errors as soon as one single technique tags it as such.

3.4. Integrating Error Filtering with ROVER

ROVER works as follows: first, the output of the different recognizers are combined into a composite transition network through a dynamic programming alignment procedure. Then a voting schema is applied at each slot of the network, to select the best hypothesis and build a new transcription output. The problem with this process is that errors contained in each recognizer output are kept in the composite network, which may trick the voting algorithm and lead to errors propagating into the final composite output. We introduce a pre-filtering stage right after the different outputs are aligned, and then eliminate the errors to facilitate the voting. This way we are hoping for fewer mistakes in the final output. To do this, each word’s surrounding context at each slot in the WTN is used, to determine whether that word is a semantic outlier and therefore should be deleted. The augmented ROVER with error filtering is detailed in algorithm 4.

Algorithm 3 Combination of Error Detection Techniques

1: Compute the score of the word $w$, $Score_w$, for each technique.
2: Scale the confidence score to the $[0,1]$ interval
3: Compute the new confidence score, $Score_{comb}$.
4: Tag the word $w$ as an error if $Score_{comb} \leq K$.

In step 4 of algorithm 3, the threshold parameter $K$ is to be optimized through a training stage. It is used to control the error detection rate. The higher $K$ is, the more aggressive the error filtering, and vice versa. If $K$ is quite low, more erroneous words slip past the combined error detector.

Algorithm 4 Augmenting ROVER with Error Detectors

1: Create the composite WTN by aligning the WTNs from the different recognizers
2: for all slots in the composite WTN do
3: Apply the error detection procedure
4: Remove the detected erroneous words from the slot, and replace them with the NULL transitions.
5: end for
6: Apply voting on the new WTN.

Instead of applying the voting mechanism at each slot, a pre-filtering stage is introduced. At each slot, the error detector is used to spot errors. If a token is flagged as an error, the algorithm updates the slot by removing the arc of the erroneous word, and replacing it by a NULL transition. This simulates a deletion, and the ROVER voting schema will handle it accordingly. Once all slots are pre-processed, the voting algorithms are used to select the most appropriate token at each slot in the new network.

4. Experimental Results

Nuance v9.0 and Sphinx-4 decoders were used with the HUB4 testing framework. Two language models have been used. LM-98T28 was built using HUB4 training data transcriptions, and LM-BN99, publicly available in the Sphinx website. The term-document matrix for the LSI-based detector, has been built using the latest Wikipedia XML dump. A total of 3.2M documents, and 100K unique terms have been identified. Google’s one trillion-token corpus[9] was used to collect uni-gram and bi-gram frequency counts required by the PMI-based error detector: the 13.5 million uni-grams and the 314.8 million bi-grams counts were used. Errors found in each of the decoders’ transcriptions were automatically flagged, and correct words were selected at random, to obtain a testing dataset to assess the error detection algorithm. For a two-class problem (Error/Not Error), we define: the precision, $Prec = \frac{TP}{TP + FP}$, and the recall, $Rec = \frac{TP}{TP + FN}$, where $TP$, $FP$, $TN$, and $FN$ represents true positives, false positives, true negatives, and false negatives respectively. To report the recognition performance, we used the WER metric.

4.1. Assessment of the Error Detectors

In this section, we study the impact of the combination of error detection techniques, namely weighted average, harmonic average and direct combination. Since we have in total three error detectors (MSS and MR under the LSI detector), we have chosen in this paper to only report results for the PMI and MSS detectors. Results for the remaining scenarios are similar. 500-long feature vectors have been selected to represent the tokens in the LSI based error detection technique. A neighborhood of 20 words has been selected to compute the PMI scores, and the maximum aggregation schema has been used to aggregate these scores. Figure 2 shows the precision vs recall graph for the different error classifiers. The precision vs recall graph highlights the impact of combining error detectors. In fact, all three combi-
nation scenarios outperform the individual error detection technique. The graph also shows that we cannot guarantee descent recall if precision is high with individual error classifiers. However, when we combine the classifiers, we could improve the recall ratio without sacrificing too much precision. Our main goal in error detection, is to capture as many errors as possible without false tagging correct tokens as erroneous output. Therefore, we want high precision with as high recall as possible. It appears now that when error detectors are combined, it’s possible to achieve better trade off between the recall and precision. The next section will study the impact of this improvement gained through error classifiers combination, on the ROVER process when augmented with the contextual analysis to filter out errors.

4.2. ROVER Augmented with Combined Error Detectors

In this paper, only the frequency-based voting mechanism of ROVER was used, because of the unreliable confidence scores of both Sphinx and Nuance decoders. A subset of the different combinations of both decoders have been reported due to the paper limitation: Sphinx-4 with LM-98T28, Sphinx-4 with LM-BN99, and Nuance v9.0. It is worth mentioning here that the decoders have been highly fine tuned, to achieve the lowest WER with the HUB4 framework. In fact, the ROVER baseline WER is at 22%. Therefore, it should be very difficult to improve on this performance. However experiments below show that with our proposed approach, it’s possible to outperform this highly optimized baseline. WER after the integration of error filtering, are lower than the baseline ROVER’s WER for most of the decoder’s combinations. Figure 3 illustrates the WER reduction compared to ROVER baseline, when it is augmented with single and combined error filtering techniques. Since the order of combination matters in ROVER’s performance (an inherent problem within the word transition network building stage), we have only reported three out of the six possible combination of three decoders’ settings. The remaining possibilities share similar findings. For each pair of decoders, we have reported the WER reduction achieved when integrating either single or combined error filtering techniques. The achieved reduction reached up to 2% compared to the original ROVER baseline. This is a promising result, because the current techniques have reached a plateau and even an apparently-small improvement in percentage is important. Results also show that combining error detectors ensures even more WER reduction, because it compensates for the low recall rates of the individual error classifiers. These findings have proven that if we are able to achieve high recalls and precision with the error detection technique, our approach of augmenting ROVER with error filtering could guarantee lower WER compared to the ROVER baseline.

5. Conclusion

In this work, a framework was proposed to improve on the ROVER performance. This is done by augmenting it with a semantic analysis procedure. An error-filtering process is introduced after building the composite network to remove the erroneous words at each slot. Augmenting ROVER with this contextual analysis to filter out erroneous words ensures lower WER compared to ROVER baseline. Besides, combining error detection techniques guarantees even a lower WER reduction. Experimental results have shown that the augmented ROVER with combined error filtering classifiers outperforms the baseline WER figures in most of the cases, reaching a drop of up to 2% in terms of absolute WER reduction.

6. References


