Manipulating Word Lattices to Incorporate Human Corrections

Yashesh Gaur[1], *Florian Metze*[1], *and Jeffrey P. Bigham*[1,2]

[1]Language Technologies Institute, Carnegie Mellon University
[2]Human Computer Interaction Institute, Carnegie Mellon University
{yashesh, fmetze, jbigham}@cs.cmu.edu

Abstract

Automatic Speech Recognition (ASR) is not perfect and even advanced statistical models make errors that render its output difficult to understand. We are therefore interested in having Humans correct ASR output efficiently. A naive approach, in which Humans manually "edit" the ASR output, may work when the recognition is done offline, but fails in on-line scenarios when Humans cannot keep up. To address this problem, our prior work introduced an approach that combines ASR and keyword search (KWS) to allow Humans to simply type corrections for the errors they observe, while the system positioned each correction using KWS and then "stitches" in the correction. In this paper, we present an improved "stitching" algorithm that works at the lattice level (rather than on the first-best string). We show that this algorithm drastically improves the word error rate (WER) of a TED system when applied to a new corpus of CS lectures that has not been carefully prepared for ASR experiments. We also show that the system can fix annoying repeat errors from just a single correction, making it suitable for post-processing of large amounts of data from limited corrections.

Index Terms: speech recognition, human computation, word lattices, keyword spotting.

1. Introduction

Automatic Speech Recognition (ASR) has received considerable attention from the research community for the past several decades, and its performance continues to improve [1, 2, 3]. However, even today's state-of-the-art ASRs produce unacceptable errors, especially in noisy environments or in the presence of many out of vocabulary words.

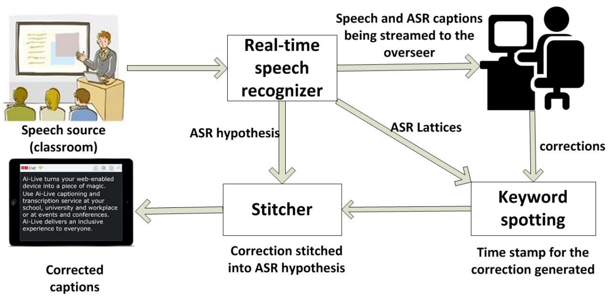
A use case that is very important and challenging is real-time captioning. The goal of real-time captioning is to convert aural speech to visual text with a very low latency. It provides immediate access to spoken content in lectures/talks and is a crucial accommodation for Deaf and Hard of Hearing (DHH). The most popular real-time (and reliable) captioning service is Communications Access Real-time Translation (CART), where extensively trained stenographers type on specialized keyboards and are able to caption up to 231 words per minute [4] with accuracy generally above 95%. However, stenographers are expensive and must booked well in advance. Consequently, they can not be used to caption for events that have been scheduled at the last minute or learning opportunities like discussions after the class. ASR, on the other hand, is relatively inexpensive and can provide on demand captioning but it still can't generate very high fidelity captions in real world scenarios. This is especially true for speech that has specialized content, *e.g.* a classroom computer science lectures about operating systems. The reasons for the unsatisfactory ASR performance are many. Poor microphones, noise from audience, reverberation and echo from the classroom/hall are some of them. More importantly, such talks are usually decoded using off an shelf recognizer. Such recognizers work with a generic language model (LM) as it is difficult to create adapted LMs for every class and/or discipline [5, 6]. Often the dictionary of recognizers do not contain the technical jargon that is found in abundance in such talks. Therefore, they get recognized incorrectly every time. These words, as our analysis will later show, contain important meaning and are salient features of the talk/lecture.

In our previous work, we introduced a new approach to real-time captioning that combines human effort with machine intelligence to generate captions with improved quality. This approach allowed humans to simply type in corrections for the errors that they observed in the ASR caption stream. The system used KWS to find the most suitable place for the correction in the caption and then "stitch" it in. This saved the human workers the substantial time required to position the correction correctly in the output stream, making them more efficient. The "stitching" approach we used was limited to replacing and inserting words in the ASR hypothesis. This paper presents a new stitching approach that incorporates human corrections directly into the word lattices generated by the recognizer. This new stitching mechanism yields better post correction transcripts. It is also able to correct mistakes that the human did not explicitly fix, but which follow from word lattice once the corrections are considered.

2. Related Work

Humans are very good at converting speech to text and taking help from them to produce or correct speech transcriptions seems intuitive. In recent years, human computation via crowdsourcing has become a popular and inexpensive means to transcribe audio. [7] used it to transcribe podcasts, [8] used it to collect data from speech dialog systems and [9, 10] have used it to transcribe conversational speech. Crowdsourcing has also been used to correct/edit transcripts from classroom lectures [11]. These approaches, however, are meant to work in an offline setup and consequently can not be used for real-time captioning because humans can not keep up with natural speaking rates. [12] tries to address this issue by synchronously coordinating a group of humans to caption different parts of the audio, which are stitched together to yield complete transcripts with latency of less than 5 seconds. However, the coordination required is unwieldy in practice.

In our previous work [13], we presented an approach for a human powered system for the purpose of real-time captioning. As opposed to [12], our approach can work with a single human who does not directly produce work but "oversees" the real-time ASR to quickly identify and correct its mistakes. Prior approaches have had humans manually edit the hypoth-


Speech and ASR captions being streamed to the overseer
Real-time speech recognizer
Speech source (classroom)
ASR hypothesis
ASR Lattices
corrections
Keyword spotting
Stitcher
Correction stitched into ASR hypothesis
Time stamp for the correction generated
Corrected captions
live

Ai-Live turns your web-enabled device into a piece of magic. Use Ai-Live captioning and transcription service at your school, university and workplace or at events and conferences. Ai-Live delivers an inclusive experience to everyone.

Figure 1: *System Architecture: Speech and ASR captions are streamed to the overseer in real-time. Corrections are entered by the overseer if mistakes are observed. KWS quickly finds the time at which correction word must have been spoken and then passes it to the stitcher, which stitches the correction into the ASR captions/lattice.*

esis from the ASR to produce correct captions. This is only faster than typing from scratch for low word error rates, due to the time required to both type and position a correction in the output stream []. Our approach eliminates the slow positioning process and instead uses KWS techniques [14] to predict where the entered correction word(s) were spoken and stitch them into the hypothesis as the correct location.

The goal of Keyword Search (KWS), also known as Keyword Spotting or Spoken Term detection, is to determine when a given search term was spoken. Popular ways to implement it are to search through the phonetic/syllable/word lattices produced by the ASR system. Hence, we ensure that our search goes beyond the 1-best hypothesis. Moreover, KWS can also detect words that are out of vocabulary and is therefore not constrained by ASR's dictionary. Figure 1 shows the architecture of our system. Audio is fed to an online ASR. The audio and the ASR captions play in front of the human. The human looks out for mistakes in the captions and enters the corresponding correction if one is required. The correction is fed to the KWS module which tries to detect the word in the recently decoded speech. The time-stamps of the detection are then sent to the "stitcher" module which tries to stitch the correction in. Previously this module worked by replacing and inserting words in the ASR hypothesis. In this paper, we go beyond this simplistic scheme by stitching the correction into the word lattice generated by the ASR.

3. Manipulating Word Lattices

Our prior word-based stitching technique reduced WER by about 50% [13], assuming all corrections were provided. The reason for such a poor performance can be attributed to both poor detection by keyword spotting and poor stitching. In our approach, detection and stitching are independent steps, so that we are relying on a state-of-the-art Open Source solution for detection keywords [14] and focus our efforts on developing the stitching technique described in this paper. Still, manual error analysis suggests that further improvements can be achieved with improved keyword spotting. One of our observations early on is that to get better stitching results, we need to stitch the word into the lattice rather than replace words in the final transcript. The manner in which stitching is done would depend on if the correction word is already present in the lattice at the correct position but couldn't be part of the final path or if it is absent from the lattice altogether. In the following sub-sections we describe these schemes in detail.

```
Data: lattice, correction, startTime, endTime,
      searchBuffer=$\epsilon$, insertBuffer=$\delta$
Result: lattice with correction stitched in.
$\mathbf{S}$ = all states in lattice;
$\mathbf{S_R}$ = all states in $\mathbf{S}$ s.t. $s$.time $\in$ [startTime, endTime];
foreach $s \in \mathbf{S_R}$ do
    $\mathbf{A}$ = {arcs that emit from $s$};
    foreach $a \in \mathbf{A}$ do
        if $a$.word is keyword then
            boost the acoustic weight of arc $a$;
        end
    end
end
if keyword is not found then
    $\mathbf{S}_{start}$ = {$s$ s.t. $s \in \mathbf{S}$ and $s$.time $\in$
    [startTime $- \delta$, startTime $+ \delta$]};
    $\mathbf{S}_{end}$ = {$s$ s.t. $s \in \mathbf{S}$ and $s$.time $\in$
    [endTime $- \delta$, endTime $+ \delta$]};
    foreach $(s_1, s_2)$ s.t. $s_1 \in \mathbf{S}_{start}$ and $s_2 \in \mathbf{S}_{end}$ do
        $a$ = new arc;
        $a$.word = correction;
        boost acoustic score of $a$;
        add $a$ to lattice with from $s_1$ to $s_2$;
    end
end
Rescore the lattice using LM to assign graph costs to
new arcs;
```

Algorithm 1: Lattice Stitcher

3.1. Correction word present in the lattice

ASR lattices contain a lot more information than the best path and it is likely that we will find our correction word(s) already present in the lattices. These corrections couldn't make it to the best path because their acoustic and graph costs were not strong enough. After we get a detection from the KWS system (in terms of start timing and the duration of the keyword), we iterate through the lattice and try to find the arcs that correspond to our correction word. We only consider an arc if it represents the correction word and its begin and end timestamps are within a reasonable range of the time stamps given by the KWS system. We boost the acoustic costs of these arcs by a reasonable margin making sure that the new best path would pass through one of these transitions. We leave the graph costs of these transitions untouched which ensures that a correct transition is chosen amongst the transitions whose acoustic costs were just boosted.

3.2. Correction word absent from the lattice

There will be cases when the correction word is absent from the lattice, but KWS will find and return approximate start and end time stamps for it. In such cases, our stitching scheme will need to add appropriate arcs/transitions with reasonable weights into the lattices such that when the lattice is re-scored, the correction word is present in the lattice. Hence, when no arcs corresponding to our correction word are found in the lattice, we consider the start and end time produced by the KWS and try to predict the states which might act as the start state and end state for the new arcs. We tried 2 different approaches here. Our first approach was to try and estimate different start and end states and then introduce arcs arcs between every pair of the above start and end state. The second approach takes all the states whose time stamp lies between the start and end time returned by the KWS system in introduced arcs between every state pair i and j such that $time(i) < time(j)$. Although the second approach introduces more arcs than the first one, we found the first one worked better. Hence, we decided to continue with this method. The acoustic scores in the arcs introduced was chosen to be the maximum acoustic score observed in the lattice. The graph costs were left blank. The lattice was latter re-scored using a language model which assigned appropriate LM costs to the newly introduced transitions.

4. Experimental Setup

To evaluate the performance of our stitching technique, we performed two sets of experiments, using Kaldi's keyword search implementation [14] for spotting keywords.

Our first experiment is performed on the TEDLIUM dataset [15] using a Kaldi [16] system trained with PDNN [17] and using the standard TEDLIUM language model. The goal for this experiment is to investigate the improvement when going from word-based stitching to lattice-based stitching. The baseline WER is 19.2% on the TED development data. TED talks resemble classroom settings to some extent, but the language model matches the test data quite well.

The second experiment thus uses an Eesen-based [18] TED system with a pruned "Cantab" language model [19], tested on a corpus of Stanford CS classes, for which we have generated reference alignments from available transcripts [1]. This setup resembles a real use case, because no adaptation of the system to the target domain has been performed. The system has a baseline WER of 15.6% on the TED development data, and has been trained with the Connectionist Temporal Classification (CTC) criterion. The keyword search implementation has been carried over from Kaldi.

In both cases, the results are reported for a simulation experiment where corrections were simulated to be coming from a human who is able to report all the incorrectly recognized words within 10 seconds.

The out-of-vocabulary (OOV) rate for the TED test set is virtually 0%, while for the Stanford CS test set it is 3.4%, both with a vocabulary size of about 150 k words.

5. Results and Discussion

Table 1 shows the reduced WER after applying our stitching algorithm. We also compare it with our previous stitching algorithm which worked on a sentence level.

Table 1: *WER on TEDLIUM for Kaldi ASR baseline, WER after stitching based on word level replacement, and stitching based on lattice manipulation.*

Error	ASR Captions	word stitcher	lattice stitcher
WER	19.2 %	9.6 %	4.33 %
Substitution	12.8 %	3.1 %	1.9 %
Insertion	2.3 %	1.7 %	1.5%
Deletion	4.1 %	4.8 %	0.8 %

[1]introduction to computer science – programming paradigms. http://see.stanford.edu/see/lecturelist.aspx?coll=2d712634-2bf1-4b55-9a3a-ca9d470755ee

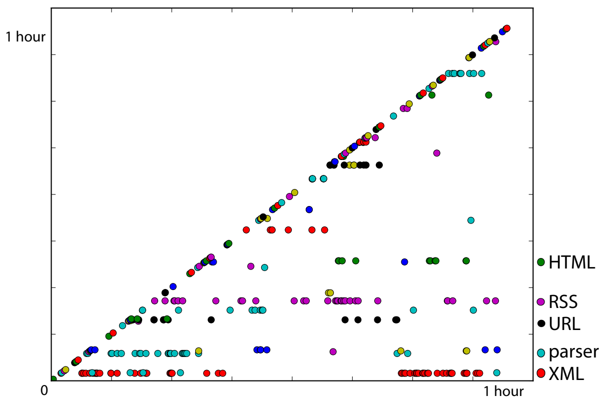


Figure 2: *Pattern of out-of-vocabulary (OOV) word occurrences and repetitions in a typical lecture from the Stanford computer science course: about 60% of OOV tokens repeat, showing up as horizontal lines in this plot. The 5 most repeated OOV words are labeled, and correspond to the primary content of the lecture*

Table 2: *WER on Stanford CS for Eesen ASR baseline, WER post stitching based on lattice manipulation.*

Error	TEDLIUM based recognizer	Post Corrections
WER	35.0 %	17.9 %
Substitution	21.4 %	9.3 %
Insertion	7.7 %	5.2 %
Deletion	5.9 %	3.2 %

Table 1 shows that the lattice stitching algorithm provides a significant improvement over our previous word-based approach, and is able to reduce deletions in particular.

Table 2 shows the results on the Stanford CS corpus. Again, there is a significant reduction of errors, although the overall efficacy of the approach is less. Our analysis shows that timing information found in lattices that have been generated with a CTC acoustic model is less reliable than timing information from "conventional" acoustic models that have been trained with a frame-based criterion and do not include the "blank" symbol, resulting in less effective "stitching". A detailed description of the CTC loss function during training can be found in [20], while [18] discusses the Weighted Finite State Transducer (WFST) based decoding approach that we are using here.

Figure 2 shows the recurring pattern of out of vocabulary (OOV) words in one of the lectures from the Stanford computer science talks. These OOV words carry the essence of the lecture. Our approach is particularly fruitful in this scenario because the human has to enter the OOV only once and our system keeps searching for the OOV word repeatedly potentially correcting the mistake even before it has been spotted. More importantly, this is done without regenerating the language model and the decoding graph.

6. Conclusion and Future Work

This paper describes a new lattice-based "stitching" algorithm, which can be used to correct ASR output from sparse user "corrections." We are currently working on an in-depth analysis of the differences between lattices generated with a CTC acoustic model and those that have been generated with a DNN. Early results indicate that we will be able to further reduce the error rate on CTC acoustic models. This is attractive because CTC acoustic models employ context independent (CI) phonetic states, and no sub-phonetic states (conventional models use begin-, middle-, and end-states), which should allow for better confidence measures and thus better keyword search. They are also about 2 times faster during decoding (not including skipping frames), and use significantly less memory because of the smaller CI search graph. These experiments should be concluded within the next few weeks.

7. Acknowledgements

This work was supported by National Science Foundation Award #IIS-1218209.

8. References

[1] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, Nov 2012.

[2] M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, and M. Federico, "Report on the 10th iwslt evaluation campaign," in *Proc. IWSLT*, Heidelberg; Germany, 2013, http://www.eu-bridge.eu/87_282.php.

[3] J. Glass, T. J. Hazen, S. Cyphers, I. Malioutov, D. Huynh, and R. Barzilay, "Recent Progress in the MIT Spoken Lecture Processing Project," in *Proc. Interspeech*, 2007. [Online]. Available: http://groups.csail.mit.edu/sls/publications/2007/Interspeech07-glass-lecture.pdf

[4] C. Jensema, R. McCann, and S. Ramsey, "Closed-captioned television presentation speed and vocabulary," *American Annals of the deaf*, vol. 141, no. 4, pp. 284–292, 1996.

[5] H. Yamazaki, K. Iwano, K. Shinoda, S. Furui, and H. Yokota, "Dynamic language model adaptation using presentation slides for lecture speech recognition," in *In Proc. INTERSPEECH*, 2007, pp. 2349–2352.

[6] C. Munteanu, G. Penn, and R. Baecker, "Web-based language modelling for automatic lecture transcription," in *Proc. INTERSPEECH*, 2007.

[7] J. Ogata, M. Goto, and K. Eto, "Automatic transcription for a web 2.0 service to search podcasts," in *INTERSPEECH 2007, 8th Annual Conference of the International Speech Communication Association, Antwerp, Belgium, August 27-31, 2007*, 2007, pp. 2617–2620.

[8] G. Parent and M. Eskenazi, "Toward better crowdsourced transcription: Transcription of a year of the let's go bus information system data," in *Spoken Language Technology Workshop (SLT), 2010 IEEE*. IEEE, 2010, pp. 312–317.

[9] M. Marge, S. Banerjee, and A. I. Rudnicky, "Using the amazon mechanical turk for transcription of spoken language," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5270–5273.

[10] S. Novotney and C. Callison-Burch, "Cheap, fast and good enough: Automatic speech recognition with non-expert transcription," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California: Association for Computational Linguistics, June 2010, pp. 207–215. [Online]. Available: http://www.aclweb.org/anthology/N10-1024

[11] H. Kolkhorst, K. Kilgour, S. Stüker, and A. Waibel, "Evaluation of interactive user corrections for lecture transcription," in *2012 International Workshop on Spoken Language Translation, IWSLT 2012, Hong Kong, December 6-7, 2012*, 2012, pp. 217–221.

[12] W. Lasecki, C. Miller, A. Sadilek, A. Abumoussa, D. Borrello, R. S. Kushalnagar, and J. Bigham, "Real-time captioning by groups of non-experts," in *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*. New York, New York, USA: ACM Press, Oct. 2012, pp. 23–34. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2380116.2380122

[13] Y. Gaur, F. Metze, Y. Miao, and J. P. Bigham, "Using keyword spotting to help humans correct captioning faster," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[14] J. Trmal, G. Chen, D. Povey, S. Khudanpur, P. Ghahremani, X. Zhang, V. Manohar, C. Liu, A. Jansen, D. Klakow, D. Yarowsky, and F. Metze, "A keyword search system using open source software," in *Proc. IEEE Workshop on Spoken Language Technology*. South Lake Tahoe, NV; USA: IEEE, Dec. 2014, to appear.

[15] A. Rousseau, P. Deléglise, and Y. Estève, "Ted-lium: an automatic speech recognition dedicated corpus." in *LREC*, 2012, pp. 125–129.

[16] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011.

[17] Y. Miao, "Kaldi+pdnn: Building dnn-based ASR systems with kaldi and PDNN," *CoRR*, vol. abs/1401.6984, 2014. [Online]. Available: http://arxiv.org/abs/1401.6984

[18] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-End Speech Recognition using Deep RNN Models and WFST-based Decoding," in *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*. Scottsdale, AZ; U.S.A.: IEEE, Dec. 2015, https://github.com/srvk/eesen.

[19] W. Williams, N. Prasad, D. Mrva, T. Ash, and T. Robinson, "Scaling recurrent neural network language models," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. Brisbane; Australia: IEEE, May 2015.

[20] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented seq uence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine Learning*. ACM, 2006, pp. 369–376.