Lattice re-scoring during manual editing for automatic error correction of ASR transcripts

Anna V. Rúnarsdóttir, Inga R. Helgadóttir, Jón Guðnason

Reykjavik University
Language and Voice Lab
Menntavegur 1, 101 Reykjavik, Iceland
annaviru@gmail.com, ingarun@ru.is, jg@ru.is

Abstract

Automatic speech recognition (ASR) systems are increasingly used to transcribe text for publication or official uses. However, even the best ASR systems make mistakes that can change the meaning of the recognition results. The results from these systems are therefore often reviewed by human editors, who fix the errors that arise. Offering automatic updates of utterances, with lattice re-scoring, could decrease the manual labor needed to fix errors from these systems. The research presented in this paper is conducted within an ASR-based transcription system with human post-editing for the Icelandic parliament, *Althingi*, and aims to automatically correct down-stream errors once the first error of a sentence has been manually corrected. After manually correcting the first error of the utterances, a new path is computed through the correction, using the lattice created during the ASR decoding process. With re-scoring, the sentence error rate (SER) for utterances containing two errors (and hence with SER=100%) drops to 82.77% and for utterances containing three errors drops to 95.88%. This paper demonstrates that the trade-off between automatically fixed errors and new errors introduced in the re-scoring heavily favours adding this process to the transcription system.

Index Terms: automatic speech recognition, lattice re-scoring, post-processing of ASR transcripts

1. Introduction

Automatic speech recognition (ASR) is often used in settings where it is necessary to have the text output as accurate as possible. Thus, the results are often post-processed by human editors, who fix the errors that may occur in the transcribed utterances and prepares the text for publication. For that reason, it is interesting to investigate ways to reduce the manual post-processing needed to verify ASR results. In this paper, we investigate whether recalculating the new best path through the utterance lattice, after manually fixing the first error that occurs, has an effect on the errors following later in the same utterance. For that purpose, we propose a lattice re-scoring method by search updating. A lattice, in this paper, refers to a representation of the alternative word-sequences for a particular utterance that exist with a likelihood higher than a set threshold [1]. The lattices are created during the decoding process and are usually re-scored automatically using a larger language model than used in the decoding. In this paper, the additional information contained in the lattices is employed to calculate new best-scoring paths for utterances that were not correctly transcribed by an ASR system.

The recent effort for developing speech recognition for Icelandic started with a collection of the open source acoustic database Malromur [2, 3] which was collected using techniques described in [4, 5]. An open source speech recognizer, based on Kaldi [6] and this data set, was presented in [7]. This acoustic data set, the recent publication of a Gigaword corpus for Icelandic [8], and an improved pronunciation dictionary [9] has made the development of high quality ASR systems for Icelandic more achievable[1].

The ASR system for the Icelandic parliament, *Althingi*, is based on a similar implementation to that of [7] where the acoustic data set for ASR training was aligned and prepared using 1500 hours of parliamentary speeches [10]. All speeches given in Althingi are recorded and made available on its website[2], along with the full transcription of the speeches. At Althingi, an ASR [10] is used to transcribe the speeches and the transcripts are then read through and manually corrected before being published. A system like the one at Althingi is therefore a good example of a system where lattice re-scoring using search updates could be useful. The word lattice search updating method itself, however, can be applied to any ASR system that generates word lattices.

Lattice re-scoring is employed by most ASR systems today. It is usually applied in the decoding process with no human input [11, 12]. Performing additional lattice re-scoring on ASR results based on manual edits have, to the best of our knowledge, not been considered in the literature before.

2. The speech recognizer

The ASR system used in this paper was based on an open ASR for Icelandic, developed by the Language and Voice Lab at Reykjavik University [7]. The acoustic model training is aligned to a recipe developed for the Switchboard corpus[3]. The model is a sequence model based on seven *time delay deep neural network* (TD-DNN) [13] layers and three *long-short term memory* (LSTM) layers [14]. The network takes 40 dimensional LDA feature vectors, and a 100 dimensional ivector, as input. Speed perturbations are applied in our DNN setup. The training data used for this work was about 100 hour (40,000 utterances) random selection from the Althingi's speech recognition cor-

This project was made possible through the support of Althingi's information and publications departments. We would also like to thank Dr. Hrafn Loftsson for valuable feedback during the writing of this paper.

[1] Open source language resources for Icelandic are available on https://www.malfong.is, and the Icelandic speech recognition recipe is available on https://github.com/cadia-lvl/ice-asr

[2] http://www.althingi.is

[3] https://github.com/kaldi-asr/kaldi/blob/master/egs/swbd/s5c/local/chain/tuning/run_tdnn_lstm_1e.sh

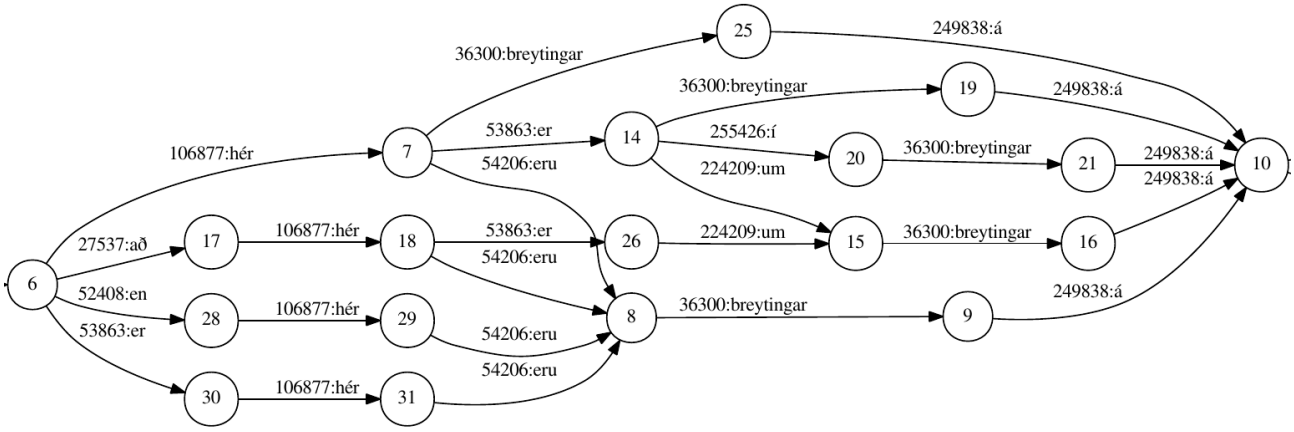

Figure 1: *A visualization of a part of a Kaldi word lattice for one utterance. The lattice has been converted to the non-compact form, epsilon arcs have been removed and acoustic and language model costs have been added together into the normal tropical cost.*

pus, collected and prepared by [10]. The test set, used both to evaluate the ASR and to test the lattice re-scoring method, is a combination of the development and evaluation set created by [10] and consists of 8,545 utterances, or ca 22 hours of acoustic data. The average length of the utterances in the test set is 22 words.

The lexicon and language models (LMs) used were created by [10]. The lexicon is based on a pronunciation dictionary from the Icelandic speech recognition project Hjal [15] and expanded with additional words from the LM training set, using the Sequitur G2P toolkit [16] to find their phonetic transcriptions, resulting in 270k words. The LM training set contained speech transcripts from Althingi, about 55M word tokens. To provide faster decoding of the TDNN-LSTM model, a pruned trigram LM was used in the decoding and the results were then re-scored using a larger trigram LM. The LMs were modified Kneser-Ney smoothed and built using the KenLM toolkit [17].

The word error rate (WER) of the ASR used in the lattice re-scoring tests is 14.24% and the sentence error rate (SER) is 80.75%.

3. Proposed method

The penultimate step of the speech recognition process is to produce a word lattice that describes the most probable sentences according to the acoustic and language models. The last step of the speech recognition decoding process is to calculate the best path through the lattice and thus the most likely sentence given the input signal and the model. The proposed method re-calculates the best path through the lattice by forcing it do go through the initial manual correction of the first error in a sentence. In what follows, we describe the lattice data structure in Kaldi, the scoring process and the proposed re-scoring approach.

3.1. The word lattice structure

The word lattices in Kaldi are weighted finite state acceptors and are implemented using the OpenFst toolkit [18] as a transducer (WFST) with a matching input and output symbol sequences [19]. A visualization of a part of a Kaldi word lattice for one utterance is shown in Fig. 1. Associated with each arc in the acceptor is a weight and the symbol. Here, the symbol is a proposed word in the sentence and the weight is a cost associated with the given arc. The final step of the decoder is therefore to minimize the accumulated cost along the proposed path.

Table 1: *A small Kaldi lattice in text mode. There is one WFST for each utterance. The first line of each WFST is the utterance ID. Each line, after the utterance ID, describes an edge. The first two columns are the start and end state of the edge, respectively. The third one shows the arc symbol, i.e. a word or <eps>. Columns four and five denote the acoustic and graph costs, respectively. The last column is a string of integers that represent the transition IDs, or the frame alignment for the word symbol. The last line of each FST is the end state.*

BN-rad20160504T163103_00032					
0	1	<eps>	17.3189	-41.5142	2_1_1_1_1_1_1_1
1	2	til	15.4052	-38.6993	11588_11587_...
2	3	að	0	0	1866_13196_13195
3	4	koma	16.1535	-47.1775	6190_6189_...
4	5	<eps>	8.10594	-19.5778	2_1_1_1_1
5	6	í	2.74915	-16.3408	5312_5311_5311_...
6	7	veg	20.2384	-84.7704	12390_...
7	8	fyrir	0	0	3750_3749_...
8	9	<eps>	5.40018	0	2_1_1_1_1_...
9					

There are various ways of defining the weight in OpenFst but word lattices in Kaldi are implemented using two data structures called *Lattice* and *CompactLattice*. The latter one is used in this work[4] [1].

Table 1 lists an example of a small Weighted Finite-State Transducers (WFSTs) from Kaldi. ϵ-transitions, denoted as "<eps>", allow the WFST to switch to another state, without consuming an input symbol. Note that the graph cost in the WFSTs in Kaldi is defined as the sum of the LM cost, the weighted transition probabilities, and any pronunciation cost.

3.2. Lattice re-scoring

The motivation behind the lattice re-scoring method presented in this paper, is to facilitate the manual post-processing of ASR transcripts. It is done by manually fixing the first error that ap-

[4] http://kaldi-asr.org/doc/lattices.html

Table 2: *A lookup table, constructed from the lattice shown in Table 1. The keys represent the nodes in the lattice. For each key, the corresponding value is a list of tuples, where each tuple contains a node, connected by a direct arc from the key node, along with the cost and the word associated with the path.*

```
{
      '0':  [('1', -24.1953, '<eps>')],
      '1':  [('2', -23.2941, 'til')],
      '2':  [('3', 0.0, 'að')],
      '3':  [('4', -31.0240, 'koma')],
      '4':  [('5', -11.47186, '<eps>')],
      '5':  [('6', -13.5916, 'í')],
      '6':  [('7', -64.5320, 'veg')],
      '7':  [('8', 0.0, 'fyrir')],
      '8':  [('9', 5.4002, '<eps>')],
      '9':  [('9', 0.0)],
}
```

pears in an utterance and finding a new hypothesis by forcing a new best-path search through the corrected word in the utterance. In practice, a human editor would make the initial correction to an utterance, but in this paper we find it by comparing the Althingi test set reference and the ASR hypothesis. The possible types of errors are substitutions, insertions and deletions. A substitution error occurs when the correct word is substituted for a different one, a deletion error when it is omitted in the ASR hypothesis, and an insertion error when the recognition result contains additional words that were not spoken.

Once a manual edit has been made, the path from the beginning of the sentence to the corrected word is considered to be correct. The rest of the word lattice for that utterance is searched recursively using a depth-first search algorithm, in order to find all possible paths through the lattice that contain the initial correct path. The search algorithm determines all paths between two nodes. The same node will only occur once in each path so the path will not contain any cycles. Instead of doing a full search in the lattice, the search along a path is stopped as soon as a word does not match the correct beginning, and the algorithm moves to the pruned path's siblings. In the end, a list of all paths in the graph, that have the correct beginning are returned. Associated with each path is the path cost up to the end node in the path.

The lattices do not always contain a path that has the correction, either because the correction is missing from the lexicon of the ASR system, or it was not considered sufficiently likely in the utterance being examined. In those cases, it is not possible to calculate a new best scoring path, hence a new hypothesis will not be found. Lattice re-scoring by updating the search can also not fix errors later in the utterances if the correct words do not exist in the lexicon.

A lookup table was created for each utterance lattice so that the WFSTs could be searched. The lookup table is in the form of a Python dictionary. For each state in the WFST, there is an entry in the lookup table for the state and the state edges. With each edge, the edge word, cost and end node are listed. The lookup table for the lattice in Table 1 is shown in Table 2. This simple lattice only contains one path from every node, but in larger lattices a node will usually have a couple of different paths.

Since Kaldi uses negative-log likelihoods, or costs, for the

Table 3: *An example of an utterance in the* `per_utt` *file. The first two lines show the alignment between the reference and the hypothesis. The third line shows the types of errors. The fourth line shows the number of errors, in the utterance, per error type. First, the number of correct words (C) is displayed, then, the number of substitutions (S), insertions (I) and deletions (D), respectively.*

ref	það	hefur	hann	reyndar	gert	án	***	allra
hyp	það	***	er	reyndar	gert	án	allrar	allra
op	C	D	S	C	C	C	I	C
#csid	5 1 1 1							

Table 4: *The same utterance as in Table 3 after a new hypothesis has been found for the utterance, by applying the proposed lattice re-scoring method.*

ref	það	hefur	hann	reyndar	gert	án	***	allra
hyp	það	hefur	hann	reyndar	gert	án	allrar	allra
op	C	C	C	C	C	C	I	C
#csid	7 0 1 0							

weights, the best path through a WFST is the one with the lowest cost. Semiring is used in Kaldi to acquire the pair of weights, which has the lowest cost (graph + acoustic), when two pairs are compared [1]. The $\oplus$ operation is used to return the lowest cost. The two costs are combined into one cost in the lookup table. The total cost for a path in the graph is simply the sum of the costs of all edges comprising the path. The Bellman-Ford algorithm [20] is used to find the shortest path.

4. Evaluation methodology

To evaluate whether recalculating a new best path through an utterance lattice, after manually fixing its first error, has an effect on the errors following later in the same utterance, we use word error rates (WERs) and sentende error rates (SERs). We want to investigate whether the new hypothesis search results in lower WERs and SERs. For that purpose Kaldi's `per_utt` file was used. The file is automatically created by Kaldi when ASR results are evaluated on a test set. It contains the alignment of the references and hypotheses in the test set, along with information about the errors in the hypotheses. Once new hypotheses had been created for the utterances in the test set, a new `per_utt` file was created, now with alignments between the references and the new hypotheses. Tables 3 and 4 show an example of one utterance in `per_utt` before and after lattice re-scoring. For this utterance the third error was not fixed, but the second one was.

The error details of the utterance in the `per_utt` file was extracted before and after corrections so that the errors could be compared. The information extracted, is returned as a lookup table, which contains the position and types of errors in the hypothesis. Two dictionaries are returned per utterance, one containing information about errors present before the lattice re-scoring is applied and one with information about the errors after lattice re-scoring. For the utterance in Table 3, the error details returned would be:

```
{1:  ['D', 1], 2:  ['S', 2], 3:  ['I', 6]}
```

and for the corrected utterance in Table 4, the returned details would be:

```
{1:  ['I', 6]}
```

The error detail dictionaries are then compared, in order to get information about the errors fixed. The lattice re-scoring method is explained in more detail in [21].

Table 5: *The 1^{st} column, displays the number of errors present in the utterances, before the first error is manually corrected. The 2^{nd} one shows the number of utterances containing the number of errors specified in the 1^{st} column, before lattice re-scoring. The 3^{rd} column, shows the number of utterances where all errors were fixed and no new errors were added. The 4^{th} one shows the number of utterances where the next error after the correction was fixed, and the last column, gives the number of utterances where the lattice re-scoring introduced new errors.*

#Errors	#Utterances	#All errors fixed	# Next error fixed	#New errors added
0	1645	-	-	-
1	1124	1083	-	41
2	865	149	229	102
3	729	30	235	110
4	593	3	189	125
5	444	1	148	76
6	317	0	106	64
> 6	735	0	284	178
Total:	**6452**	**1266**	**1191**	**696**

5. Results

Of the $8,545$ utterances in the test set, $1,645$ were already correctly transcribed, and $2,093$ did not have the correction in its word lattice. The lattice re-scoring was thus only applied to $4,807$ of the utterances in the test set. Table 5 gives an overview of the test set, after lattice re-scoring was applied, where the $2,093$ utterances, which did not contain the word in its lattices are excluded.

In practise, the human editors do not know which utterances are correctly transcribed by the ASR and which are not. Therefore, in a text editor, the search updating would be applied to all edited recognition results.

By only re-scoring the utterances in the test set, which contained one error before the manual corrections, gave an insight into whether new errors would be added by the new method. That was the case in 3.65% $(41/1124)$ of them.

Every subsequent error in an an utterance is automatically fixed in 17.23% $(149/865)$ of those utterances that originally contained two errors, and in 4.12% $(30/729)$ of the ones that had three errors. Only in 0.39% of the utterances, which had four and five errors combined, were all the errors fixed by lattice re-scoring and in none of those that contained six or more. Hence, as the number of errors increases in the utterances, the ability of the lattice re-scoring method to fix all the remaining errors without introducing new ones, decreases. However, the ratio of utterances, where the next error after the correction is fixed, is similar among the utterances, regardless of the amount of errors they contain. Analysis regarding how the distance between the first error and the next affects the benefits of the proposed lattice re-scoring method has not been done.

$3,683$ utterances in the test set contained two errors or more and had the correction of the first error in the decoding lattice. By applying the re-scoring method, all errors were fixed in 4.97% $(183/3,683)$ of those. The next error was fixed in 32.34% $(1,191/3,683)$ and new errors introduced in 17.78% $(655/3,683)$. Table 6, presents the total number of errors and the WER and SER in those utterances, before and after applying the lattice re-scoring. It shows that applying the method decreases the total number of errors by 5.5%.

Table 6: *Total number of errors, WER and SER, of the utterances which contained two or more errors. The Before row displays the values, after the manual corrections of the first error have been performed, and the After row present the same values after re-scoring.*

	# Errors	WER (%)	SER (%)
Before	13588	14.15	100
After	12837	13.37	95.03

Looking into the utterances where the error following the manual correction, was not fixed, we see that for 8.2% of them the correction of the next error was an out of vocabulary word (OOV), i.e. not in the ASR lexicon. In these cases the new method did not have a chance of providing a correction. This is not that high percentage, so in most cases, the new best hypothesis simply did not contain the correction.

6. Conclusion

Our goal with this project was to find a way to reduce the manual labor put into editing ASR outputs. We have shown that updating the lattice search for an utterance, after a manual correction is made, decreases the total number of errors present in the rest of the utterance. The total number of errors removed from the test set by applying lattice re-scoring is 5.5%. Furthermore, 32.34% of errors standing closest to manually fixed errors are automatically fixed. By applying the proposed method, the SER for utterances containing two errors drops by 17.23%, i.e. from 100% to 82.77%, and for utterances containing three errors it drops by 4.12%, or to 95.88%. Therefore, for an ASR system, such as the one built for the Icelandic parliament, it is promising to provide lattice re-scoring in the manual editing phase of ASR outputs.

The next step of this work should be to measure whether the proposed method really decreases the editing time in the wild, or if the human editor is delayed by the new suggestions. Moreover, it has to be investigated how easy it is to integrate Kaldi lattice re-scoring into a text editor for an ASR system. The scripts used, to perform the lattice re-scoring are not heavy to run, however, the execution time of the scripts was not measured in detail. Also, the Kaldi lattices do need to be stored, so they can be accessed during the search process. Both these factors need to be considered when integrating the method into a text editor.

7. References

[1] D. Povey, M. Hannemann, G. Boulianne, L. Burget, A. Ghoshal, M. Janda, M. Karafiát, S. Kombrink, P. Motlíček, Y. Qian *et al.*, "Generating exact lattices in the WFST framework," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 4213–4216.

[2] J. Guðnason, O. Kjartansson, J. Jóhannsson, E. Carstensdóttir, H. H. Vilhjálmsson, H. Loftsson, S. Helgadóttir, K. M. Jóhannsdóttir, and E. Rögnvaldsson, "Almannaromur: An open Icelandic speech corpus," in *Spoken Language Technologies for Under-Resourced Languages*, 2012.

[3] S. Steingrímsson, J. Guðnason, S. Helgadóttir, and E. Rögnvaldsson, "Málrómur: A manually verified corpus of recorded Icelandic speech," in *Proceedings of the 21st Nordic Conference on Computational Linguistics*, 2017, pp. 237–240.

[4] M. Petursson, S. Klüpfel, and J. Gudnason, "Eyra-speech data acquisition system for many languages," *Procedia Computer Science*, vol. 81, pp. 53–60, 2016.

[5] J. Guðnason, M. Pétursson, R. Kjaran, S. Klüpfel, and A. B. Nikulásdóttir, "Building ASR corpora using Eyra." in *INTERSPEECH*, 2017, pp. 2173–2177.

[6] 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, iEEE Catalog No.: CFP11SRW-USB.

[7] A. B. Nikulásdóttir, I. R. Helgadóttir, M. Pétursson, and J. Guðnason, "Open ASR for Icelandic: Resources and a baseline system," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018.

[8] S. Steingrímsson, S. Helgadóttir, E. Rögnvaldsson, S. Barkarson, and J. Gudnason, "Risamálheild: A very large Icelandic text corpus," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018.

[9] A. B. Nikulásdóttir, J. Guðnason, and E. Rögnvaldsson, "An Icelandic pronunciation dictionary for TTS," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, Dec 2018, pp. 339–345.

[10] I. R. Helgadóttir, R. Kjaran, A. B. Nikulásdóttir, and J. Guðnason, "Building an ASR corpus using Althingi's parliamentary speeches," in *Proceedings of INTERSPEECH*, Stockholm, Sweden, August 2017.

[11] A. Ljolje, F. Pereira, and M. Riley, "Efficient general lattice generation and rescoring," in *Sixth European Conference on Speech Communication and Technology*, 1999.

[12] H. Sak, M. Saraclar, and T. Güngör, "On-the-fly lattice rescoring for real-time automatic speech recognition," in *Eleventh annual conference of the international speech communication association*, 2010.

[13] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, 1989.

[14] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *arXiv preprint arXiv:1402.1128*, 2014.

[15] E. Rögnvaldsson, "The Icelandic speech recognition project Hjal," *Nordisk Sprogteknologi. Årbog*, pp. 239–242, 2003.

[16] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, no. 5, pp. 434–451, 2008.

[17] K. Heafield, "KenLM: Faster and smaller language model queries," in *Proc. of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2011, pp. 187–197.

[18] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "Openfst: A general and efficient weighted finite-state transducer library," in *International Conference on Implementation and Application of Automata*. Springer, 2007, pp. 11–23.

[19] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[20] J. Bang-Jensen and G. Z. Gutin, *Digraphs: Theory, algorithms and applications*. Springer Science & Business Media, 2008.

[21] A. V. Rúnarsdóttir *et al.*, "Re-scoring word lattices from automatic speech recognition system based on manual error corrections," Master's thesis, Reykjavik University, 2018.