M907 Speech Recognition and Speech Synthesis Technologies

(Summer Semester 2023)

ASR-Assignment

Report

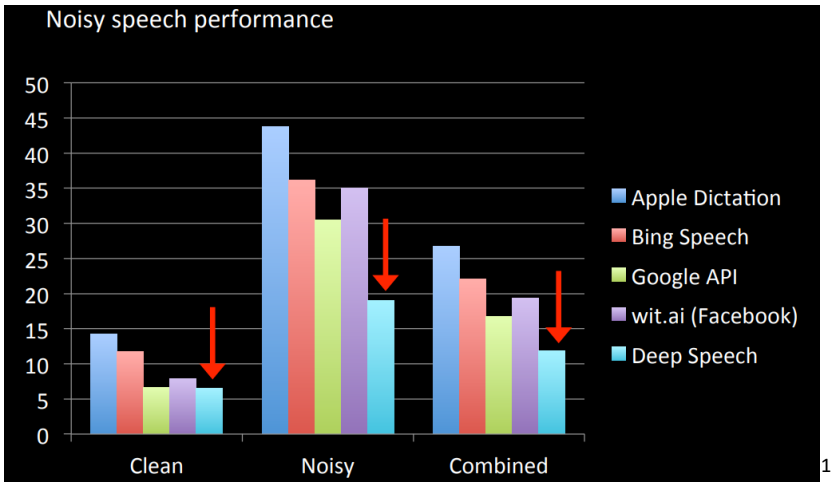
Kleopatra Karapanagiotou (lt12200010)

Table of Contents

Mozilla DeepSpeech

Model Overview

Mozilla DeepSpeech is an open-source, end-to-end speech recognition engine launched in 2017 by the Mozilla Corporation and it was developed based on deep learning algorithms. The key advantage of this system is that it can efficiently handle background noise, reverberation, and speaker variation by its robust model, and it successfully out-performs traditional Speech Recognition systems. In the following figure we see, the Word Error Rate of DeepSpeech and of other traditional SR systems and we confirm the outstanding performance of DeepSpeech in environments with background noise.



[1]

Opposing to traditional SR systems, the concept of phoneme or a phoneme dictionary is skipped by employing a well-optimized RNN model that runs on several GPUs and is pretrained by Mozilla based on the original DeepSpeech architecture[2] (training with CTC loss function with beam search and n-gram language model) with changed neural network topology.

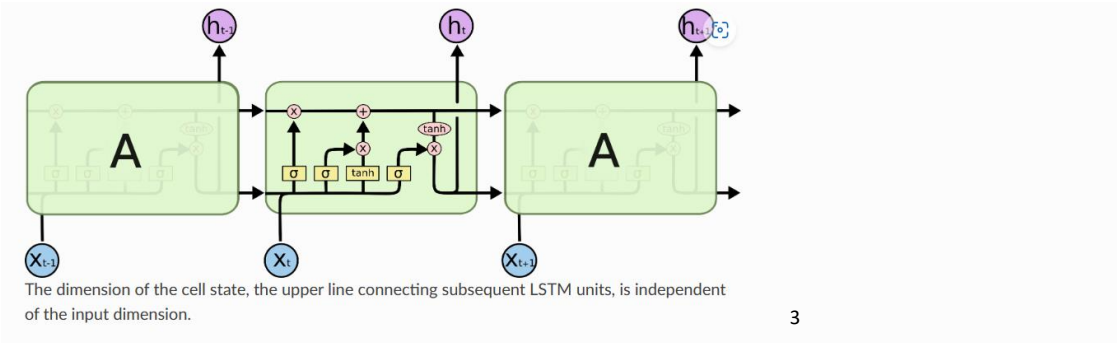
Architecture

The core of DeepSpeech is a deep recurrent neural network that receives audio features as input and outputs the corresponding transcription. The RNN consists of an LSTM RNN that works "forward in time" (unidirectional). This enables the model to be used for real-time audio transcriptions. In the following figure, we see the forward LSTM topology for the first three exemplary frames of the input feature sequence $\mathbf{X}$.

---

[1] [YouTube *Deep Speech: Scaling up end-to-end speech recognition*]. https://www.youtube.com/watch?v=P9GLDezYVX4

[2] Hannun et al., n.d.(https://arxiv.org/pdf/1412.5567.pdf)



The dimension of the cell state, the upper line connecting subsequent LSTM units, is independent of the input dimension.

[3]

The deep recurrent neural network has overall **6 layers**. MFCC features are extracted from the input audio signal and fed into three fully connected (FC) layers with rectified linear unit (ReLU) activation function. The fourth layer is a unidirectional long short-term memory (LSTM) unit with tanh activation function. This is followed again by an FC layer with ReLU activation function. The output layer of the network gives a matrix of character probabilities over a certain period of time, which is a fully connected layer with softmax activation function. At each time step of 20 milliseconds the network outputs a probability $P(ci\ |\mathbf{x}t)$ of each character ci in the character set $\mathcal{C}$, corresponding to that particular input audio frame t. DeepSpeech uses the CTC-loss function, which is useful for performing supervised learning on sequence data and this function allows us to maximize the likelihood that the character chosen by the network is a correct match. The gradients with respect to all model parameters are computed via back-propagation, and the Adam optimizer is used for training, because, generally, it requires less fine-tuning. The alphabet of the possible output characters consists of the number of characters in the target language plus one, for the blank. For example, in English it is the cardinality of the set:

$$\{a, b, c, \ldots, z, space, apostrophe, blank\}$$

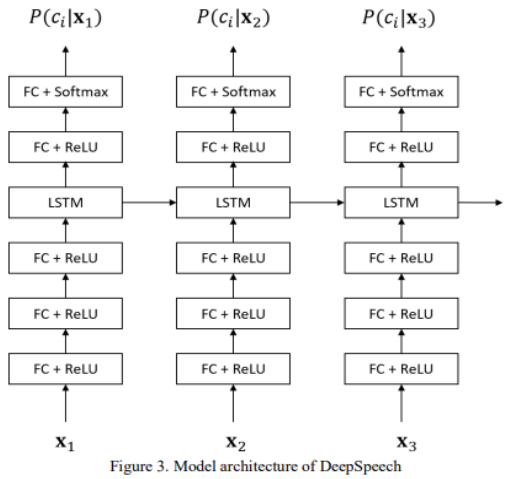
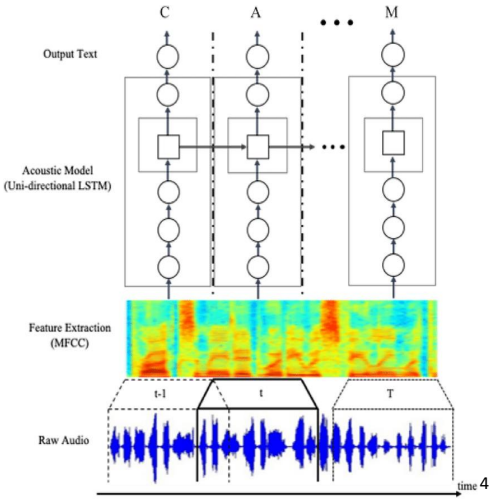


Figure 3. Model architecture of DeepSpeech



[4]

[3] Representation of an LSTM cell | Image by Christopher Olah

[4] Figure - available from: International Journal of Speech Technology

External Scorer

DeepSpeech can use an additional language model optionally (referred to as "scorer"), to improve the accuracy of the predicted transcripts, because it helps in rectifying the spelling errors which are correct phonetically, but not in line with the language spellings and grammar. The scorer is used to compute the likelihood of sequences of words or characters in the output, to guide the decoder towards more likely results. The scorer consists of two components, the

• KenLM language model, which is a probabilistic language model that uses n-gram models, which assign probabilities to sequences of words based on their frequencies in a training corpus. These probabilities are used to estimate the likelihood of a given sequence of words occurring in a language.

• and a vocabulary as a text file.

The probability estimate from the language model is included as a factor in the inference problem, which is given by:

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmax}} P(\mathbf{W}|\mathbf{X}) \cdot P(\mathbf{W})^{\alpha} \cdot R_{\mathbf{W}}^{\beta}.$$

Where:

P(W|X)= the CTC conditional probability

$P(W)^{\alpha}$= the language model probability

R_W^{β}= the "word" insertion bonus

The same calculation in log space is :

$\log(P_{CTC}(w|x)) + \alpha \log(P_{LM}(w)) + \beta \text{wordcount}(w)$

The per-frame probabilities and the word insertion term are to be further decoded with a CTC decoder with beam search.

Task Description

Evaluating the speech recognition accuracy of Mozilla DeepSpeech under the effect of noise and by adjusting the weight of the language model. The accuracy metric I chose for this task is the Word Error Rate, which is the sum of word substitutions, deletions and insertions divided by the number of words in the reference sentence.

Data Preparation

For the evaluation I used the development set of the MiniLibriSpeech ASR corpus. The corpus consists of 26 folders, one for each speaker. I kept 5 random .wav files from each speaker folder. Some speaker folders contained 2 or 3 subfolders with .wav files, so from these speaker folders I kept 10 or 15 .wav files accordingly. The number of subfolders chosen for all speakers was 37, so the cardinality of the .wav files is 185. For this task I had to add noise to these 185 files. For each of these I generated 3 noisy files with different Signal to Noise Ratio (3, 6 and 9 dB). The noisy environment I chose is from a meeting recording (foldername: **OMEETING**) with sampling rate 16000 Hz and was chosen from the DEMAND database. The process for adding background noise to the clean .wav files can be found in the Jupyter Notebook with name **"Adding Noise"**). For the effect of the language model weight, I generated for each speaker folder 4 new .wav files out of the original(clean) file. In each of them the weight of the language model was set to the following 4 values:

lm_alpha=0: Which means we assign no weight to the language model, disabling its influence during decoding. The model will rely solely on the acoustic model predictions.

lm_alpha=1: We assign equal weight to the language model and the acoustic model predictions. This weight implies that we have a high level of confidence in the language model and want it to have a significant impact on the decoding process.

lm_alpha=1.5, lm_alpha=2: The next two weights we experiment with, imply that we overtrust the language model and want to check how biased the results will be.

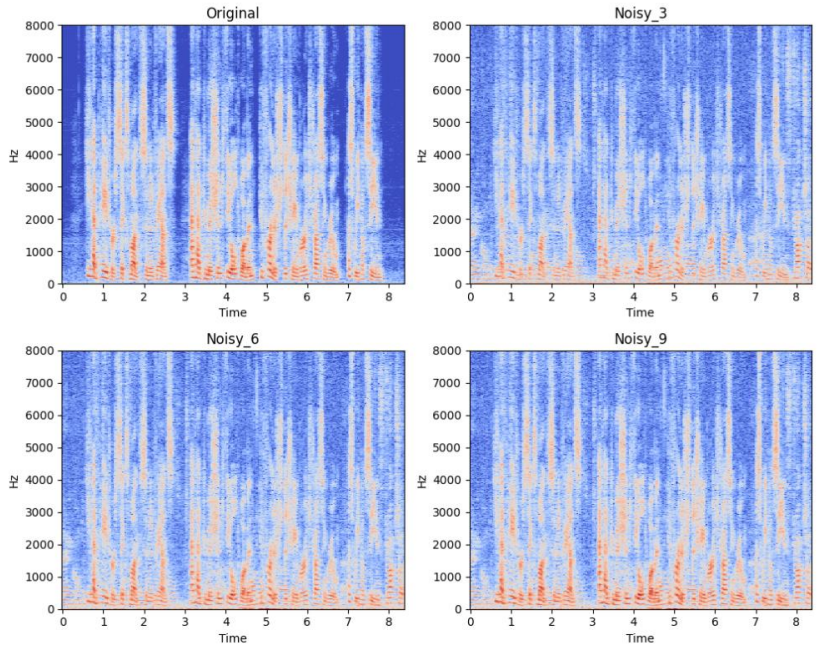
So by using the scripts from the Notebook with name "**Transcriptions with Mozilla DeepSpeech"** I generated 8 experiment transcription settings with the following names: Clean, Clean_lm_0, Clean_lm_1, Clean_lm_1.5, Clean_lm_2, Noisy_3, Noisy_6, Noisy_9 .

During the visualization of the WER results, one file with WAV_ID 8842-304647-0001 seemed to be affecting the statistics, due to its outstandingly high WER result. After review, I confirm that this sentence does not contain English words, so I decided to remove it from all the 8 generated transcription settings. Further details about the effect of this outlier in the datasets can be found in the Notebook **"Comparing WER results".**

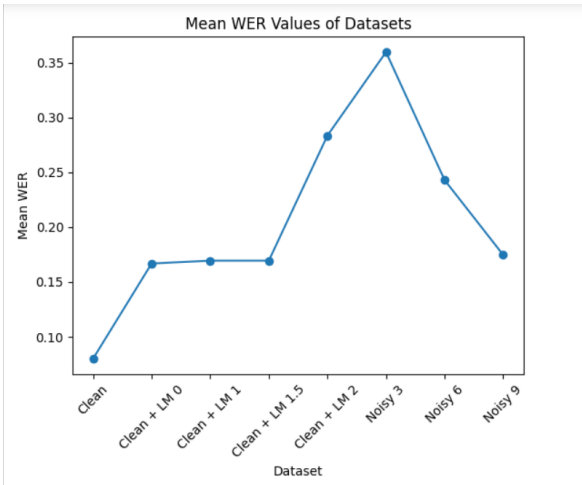
Experiment Results

(a)Effect of noise in the performance of the Speech Recognition system

Already during the process of generating noisy files out of an original clean file, we got the intuition/suspicion, that the higher the Signal to Noise Ratio, the more accurate the transcription will be, because the spectrogram of the noisy file with SNR equal to 9dB seems to be maintaining the original speech patterns that we see in the spectrogram of the original file but with some additional diffuse energy due to the noise.



This happens, because as we increase the value of SNR, we give more power on the desired signal relative to the power of the background noise. So, with a higher SNR, the background noise gets weaker. The above assumption was confirmed during the evaluation with Word Error Rate, where we see the descending trend of the WER-line as we reach higher SNR.



(b)Effect of the weight of the language model

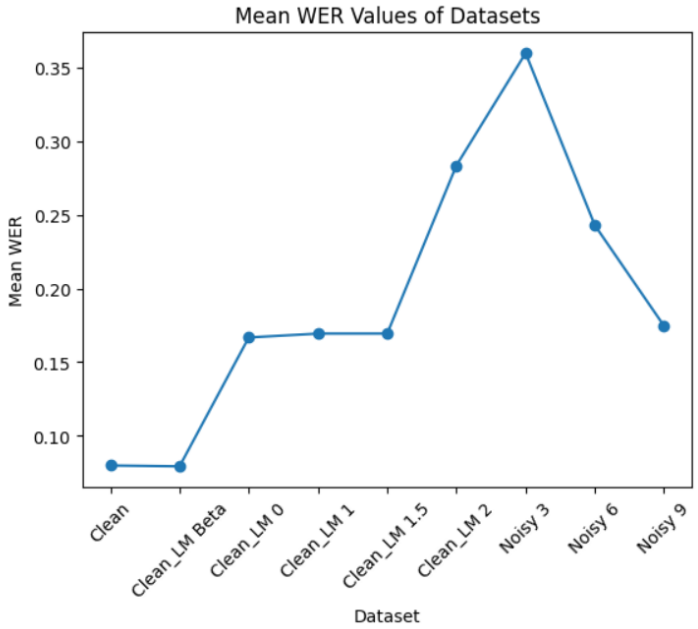
From the above graph, we also receive information about the effect of a series of values for the language model weight. We see, that, either we assign no weight to the language model (lm=0), or we set it equal to the weight of the acoustic model (lm=1) or we double it from the default value (lm=1.5), we get quite similar WER results. Results get tremendously worse if we set the language model weight equal to 2. We can say, that with that weight on the language model, the acoustic model does not exist. We rely only on the language model probabilities for the final transcription. This behavior can be described as "overfitting" to the language model. All these 4 cases increase the WER performance of the clean dataset, which means that changing the default weight of the language model (0.75) does not really improve the model's performance.

For a more thorough overview of the WER results of all the transcription settings, you can refer to the folder named **"Excel sheets"** and to the .csv file with title **"WER_Results".** For the generation of the WER results, you can refer to the python scripts in the folder **"Python_scripts_wer_calculations".**

Conclusions and further thoughts

By reviewing the .wav files and their transcriptions, I noticed that there was a lot of uncommon vocabulary. In scenarios where there is out-of-vocabulary words, increasing lm_beta encourages the model to produce transcriptions that include a wider range of vocabulary. This can be the reason why, adjusting the language model did not improve the transcription performance. Instead, it would be interesting if we adjusted the word insertion bonus (lm_beta) higher.

I attempted to increase the value of lm_beta to 2, whereas the default is 1.85. The WER results did not indicate a significant difference from the clean transcription setting. However, there was a decrease from 0.08 to 0.074, so experimenting with even higher lm_beta values both in clean and noisy transcription settings may lead to better WER results.



References

- Building a neural speech recognizer for quranic recitations - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Mozillas-DeepSpeech-model-structure-over-the-time_fig5_362507405 [accessed 11 Jun, 2023]
- End to End Automatic Speech Recognition: State of the Art. (2022, September 9). Paperspace Blog. https://blog.paperspace.com/end-to-end-automatic-speech-recognition-state-of-the-art/
- Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., & Ng, A. (n.d.). Deep Speech: Scaling up end-to-end speech recognition. https://arxiv.org/pdf/1412.5567.pdf
- Mahajan, Jagesh. (2020, February 26). *Running Mozilla's Tensorflow Implementation of DeepSpeech*. Delvify. https://medium.com/delvify/running-mozillas-tensorflow-implementation-of-deepspeech-1f157417f2c5
- Hannun,A.(2015).Stanford Seminar-DeepSpeech: Scaling up end-to-end speech recognition. In *YouToube* https://www.youtube.com/watch?v=P9GLDezYVX4
- Scheidl, H. (2021, July 7). An Intuitive Explanation of Connectionist Temporal Classification. Medium. https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c