Automatic Speech Recognition of the Swedish Language

Alexandros Ferles ferles@kth.se
Anastasios Lamproudis analam@kth.se
Leonidas Valavanis leovala@kth.se

Abstract

Automatic Speech Recognition deals with the problem of translating spoken language into text. Machine Learning techniques have been mainly adapted to such problems, and with the advancements made in the field of Deep Learning many such methods have been also proposed with high efficiency the latter years. *Deep Speech* is a state of the art deep neural network that provides speech-to-text translation with great accuracy for the English language. Based on the implementation of this network as a framework from the Mozilla foundation, we adapt this framework on the Swedish language. We present our results, conclusions and discuss further improvements that can be made to build an optimal Swedish Language ASR framework.

1 Introduction

It has been proven that deep neural networks perform quite well when used in Speech Recognition tasks. Under this scope, the Deep Speech[1] architecture was implemented and it provided state of the art performance in translating English spoken language into text. This network was further improved[2], while including a fine-tuned version for the Mandarin language, proving that after certain operations are performed it can be generalised to different spoken languages with a variety of different characteristics among them. For the original network, an open source framework in the form of a TensorFlow implementation has been provided by the Mozilla foundation[1]. With the intention of adapting this framework in the Swedish spoken language translation to text, we made use of the NST Acoustic database that includes several recordings of a variety of speakers drawn from both sexes, different areas of Sweden and different ages. After thorough work in pre-processing the available data to fit the requirements of this framework, we performed several experiments from scratch under many distinct configurations. We present and discuss the results of our work, where we focus on how training can be optimised through its training parameters and appropriate selection from the data pool, as well as the qualitative characteristics of the Swedish language that affected these results.

The rest of this paper is organised as follow: In Section 2, we provide a brief description of how the Deep Speech architecture works, while on Section 3 we focus on the NST dataset and describe all the pre-processing steps that were made. In Section 4, we present a full description of the experiments that were performed and the criteria based on which we evaluated these experiments along with the results of this evaluation. Discussion on these results can be found in Section 5, while in Section 6 we provide our final conclusions and suggestions on future work that can be done for further optimizing the ASR model for the Swedish language.

[1] `https://github.com/mozilla/DeepSpeech`

2 The Deep Speech Framework

The following figure gives us a visual description of the Deep Speech architecture:

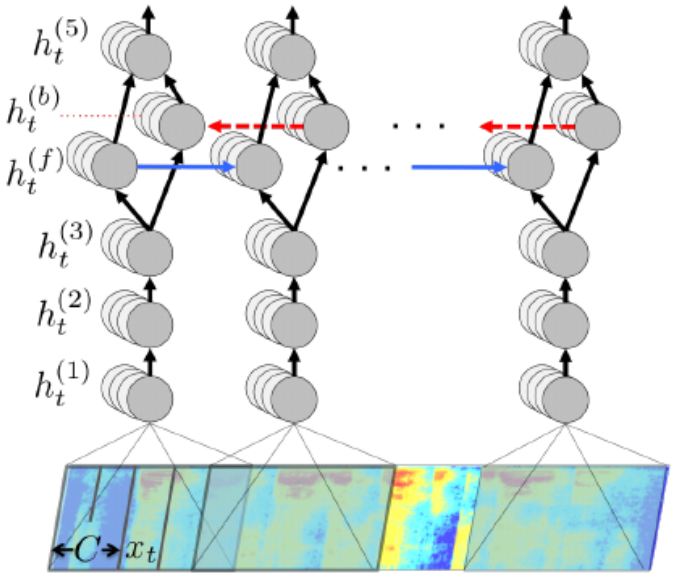

Figure 1: The Deep Speech network[1]

Before the training procedure starts, a spectrogram is created from the original sound files. Thus, the audio is converted to MFCC features that are fed to the network. Given an audio file, DeepSpeech calculates 26 MFCC features at every 0.02s time step with a window length of 0.032s. Knowing that the sample rate is 16000 kHz the number of features varies depending on the length of the recording.

The first 3 layers of the network are fully connected serving the purpose of feature extraction from the input spectrogram. Afterwards, a bi-directional Recurrent Neural Network (RNN) layer is used in order to take advantage from sequence information from both ends of the data (essentially exploiting separately the dependence from the start of the recording up to a time point *t* that is under investigation and similarly from the end of this recording to the same time point) passing this information to a final fully connected layer. The output of the network is then combined with a language model and an N-gram model in order to generate the corresponding text transcription.

The neural network training is guided by the Connectionist Temporal Classification (CTC) loss function to evaluate the model. Currently, CTC loss optimization outperforms all other approaches in recognizing text.

3 Data

Our work was mainly based on the NST Acoustic database for Swedish. The following table[3] describes the total distribution of the Swedish recordings from this dataset:

set	#speakers	#recordings	length (hours)
training	965	307568	420.8
test	76	73046	103.4

Table 1: NST dataset statistics

3.1 Preprocessing

The original format of the audio files is binary NIST and as such reformatting in a WAVE format was expected from the model to train on it. The DeepSpeech framework takes as audio input only WAVE files with 16-bit, 16 kHz, single channel, so we had to convert the data to the desired format using SoX. The Deep Speech framework expects to know 3 attributes for each data point:

- The location of the file in the local path
- The size of each WAV file
- The corresponding text transcription to this sound file

These attributes are delivered in a csv file format for all data points. Furthermore, as we experimented with the DeepSpeech we found that small WAVE files are not compatible with the implementation and produce training errors. To find small files, DeepSpeech compares the extracted MFCC features subtracted by $2 * N_f$, where N_f stands for the number of frames in the context with the number of characters in the transcription, and if the number of features are less than the number of characters then the WAVE file is defined as short for transcription, and thus excluded from the training process. Following this procedure, approximately 5% of all data were removed.

3.2 Swedish Corpus

In order to train a Deep Speech model from scratch we also need a language model and N-gram model which can be extracted from a sufficiently accurate and general Corpus of the Swedish language. Although Wikipedia and newspapers may provide enough information, they are highly likely to be inaccurate as they do not represent everyday speech. Instead we used the Familjeliv.se (Family Life) General Thread from Språkbanken in Gothenburg as a corpus from which we extracted $\sim 5,000,000$ sentences which we then normalised, removing symbols, punctuations and substituting foreign characters (vowels) in their best corresponding Swedish ones.

Following that process and in order to avoid unnecessary variance for our model, we agreed with the Facebook group training the *wav2letter++* model to use $\sim 1,000,000$ sentences from the above forum thread. With the help of KenLM[2], we used that corpus to create the leanguage model. We decided to proceed with a language model which has order 3, so our N-gram models are at most 3-gram ones. Additionally, singletons of order 3 are pruned. For faster extraction, a binary-model is created.

The **small corpus** was created from the transcriptions of the NST dataset, summing up to approximately 189,000 transcriptions. Note that only the training transcriptions were used for this corpus. In addition, we created the **full corpus**, which is a combination of the 189,000 transcriptions and the 1,000,000 sentences extracted from Språkbanken in Gothenburg. As we will discuss in more detail in (4.2), we will make use of both the *small corpus* and the *full corpus* to create two language models for testing. The small corpus takes place in both settings so that the existence of all the words that need to be predicted in the corpus is guaranteed, and generalization is preserved.

The following table shows the number of n-grams created by the language model :

Order	small corpus	big corpus
1	86995	249801
2	606534	2395874
3	112131	1132554

Table 2: Count of n-grams used by the language models

[2] `https://github.com/kpu/kenlm`

3.3 Datasets

For our work, we created the following datasets:

- The **Common Dataset**, which is the common dataset used by us and the ASR Facebook group.
- The **Full Dataset**, which is a dataset created only by us in order to exploit a maximum number of data and get optimal results.

The *Common Dataset* which we share with the Facebook group, includes approximately 30,000 samples split in a training (27,000 samples), validation (3,000 samples) and test set (2,000 samples). It was constructed under the scope of balancing both sexes (male and female subjects share similar contribution in all sets) as well as contain subjects that were both born and raised in Stockholm (the attributes *region of birth* and *region of youth* both have 'Stockholm med omnejd' as their value). We believe that this dataset is an excellent starting point, since its samples are invariant in terms of spoken accent of the Swedish language, and additionally come from an area that holds the biggest percentage of the Swedish population. Thus, it helped us gain useful insight on the training configurations of the framework, which can be easily generalised to a broader set. The distribution of male and female speakers in the Common Dataset can be found in the following table:

#data	Total	Training set	Validation set	Test set
Male	14073	11796	1340	937
Female	18039	15204	1660	1175

Table 3: Distribution by sex on the Common Dataset

The *Full Dataset* includes in total 150,000 data split in 90,000 training, 10,000 validation and 50,000 test samples.The dataset is sampled from the NST with the intention of preserving a balanced contribution of both sexes in the samples used in all the separate subsets (training and validation sets, the test is already separated and independent of the other sets). Moreover, we wish to include samples drawn from all areas of Sweden. Stockholm and Göteborg have slightly more data as they cover a greater percentage of the population. The distribution of the data drawn by area of youth distribution by sex can be found in the following tables:

Area	Training	Validation	Test
Stockholm med omnejd	18000	2000	5400
Göteborg med omnejd	9000	1000	5324
Östra sydsverige	8100	900	5324
Västra sydsverige	8100	900	5324
Norrland	8100	900	5324
Västergötland	8100	900	4437
Östergötland	8100	900	5085
Mellansverige	8100	900	5400
Västsverige	8100	900	5324
Dalarna med omnejd	8100	900	4437

Table 4: Distribution by area of youth on the Full Dataset

#data	Total	Training set	Validation set	Test set
Male	72340	40082	6078	26180
Female	81354	51718	4122	25514

Table 5: Distribution by sex on the Full Dataset

Our original intention was to include the whole NST dataset and split it accordingly, but it was proven impossible given our computational resources. We were not in position to complete an experiment under this setting due to memory errors. As a result, we come with a variety of subsets that are tested towards optimal efficiency of the derived test set results in addition to executing training operations that can be completed to the end.

4 Evaluation

4.1 Evaluation Metrics

The following metrics are used to evaluate the performance of our work:

1. *Word Error Rate* (**WER**) of our results, measuring the mis-predicted words of the generated text transcriptions.

2. *Character Error Rate* (**CER**), of our results, measuring the mis-predicted characters of the generated text transcriptions.

This way we can understand the drawbacks of the model in two directions; in what degree it was able to understand the pronounced characters on the sound files and whether it was able to build the correct words from these characters. As we also discuss to the next section, the performance between these rates is different, which in our opinion can be attributed to the fact that Swedish pronounced words have short pauses between their syllables which are misinterpreted as different words.

4.2 Experiments

We focus our experiments using the following language models:

- Language model generated by what we call the **full corpus**, which is a combination of the *General Thread* corpus drawn from *www.familjeliv.se* and the corpus that we created from all the transcriptions in the NST training dataset.

- Language model generated by what we call the **small corpus**, which is only the corpus that we created from all the transcriptions in the NST training dataset

and of course using both the *Common Dataset* and *Full Dataset*.

We observed that there was almost no difference between the language models when used for the decoding part of the framework, where predictions from the trained model were generated. Thus, we decided in most of our experiments to proceed with the language model generated by the **full corpus** ,which is more general, unless mentioned otherwise. Moreover the default learning rate for our experiments is 0.0001.

When using the *Common Dataset*, which we share with the Facebook group, we observe a memory usage of up to 30%. As a result, there is room for us to explore training settings that include more data than the common ones. However, training by using the whole training set in a mini-batch gradient-descent approach is impossible in our machine due to constraints from our GPU card.

4.2.1 Common Dataset

We firstly present the results derived in the common dataset under various training parameter settings, where 3-gram model is used. Additionally, depending on the batch size which varies the training time, a training epoch lasts between 30 seconds and one minute. The following table sums up our results:

#epochs	Train batch size	Test batch size	#hidden nodes	Dropout rate	WER	CER
60	100	50	450	0.3	26%	11.83%
90	32	32	375	0.5	24.49%	10.98%
201	32	32	350	0.5	22.19%	**9.7**%
155	32	32	345	0.5	22.43%	10.03%
164	32	32	340	0.5	**21.96**%	9.73%
65	32	32	300	0.5	32.93%	15.32%

Table 6: Common Dataset Results

The default parameters of the deep speech implementation were avoided, since a batch equal to 1 is used for the train, test and validation data. That means that each data point is updated individually and the training becomes sequential resulting in a very slow training process. Modifications regarding the train/validation and test batch size, number of hidden nodes and amount of dropout rate applied to the network were performed.

Under another scope, we tried to use a different amount of the dropout rate in each fully connected layer in several experiments as shown in the following table:

Epochs	# hidden nodes	Layer #1 dropout	Layer #2 dropout	Layer #3 dropout	Output Layer dropout	WER	CER
114	350	0.2	0.3	0.5	0.5	**20.6**%	**9.1**%
120	340	0.2	0.3	0.5	0.5	20.64%	9.21%

Table 7: Results in the Common Dataset when varying the value of dropout rate in each fully connected layer

We also used the Full Dataset for 200 epochs of training and a dropout rate of 0.5 in order to pre-train our model. Afterwards, we used the pre-trained model on the common dataset again with different dropout rate values in each fully connected layer in the Common Dataset and got good error rate results as shown in the following table:

Layer #1 Dropout Rate	Layer #2 Dropout Rate	Layer #3 Dropout Rate	Output Layer Dropout Rate	WER	CER
0.2	0.3	0.5	0.5	**17.01**%	**7.45**%

Table 8: Results of a pre-trained model on the Full Dataset

4.2.2 Uneven Dataset

We also conducted a single experiment with unbalanced data: some areas were used solely in the test set while not participating in the train and validation set, and some other areas had an uneven participation in these sets. This way, we are in position to evaluate the performance of the framework in cases that the real, unseen data are dissimilar to the training data and test how it will perform in circumstances that enrisk bad generalization. The results on this experiment were encouraging, as seen in the following table:

#epochs	Train batch size	Test batch size	#hidden nodes	Dropout rate	WER	CER
51	32	32	350	0.5	29.81%	12.73%

Table 9: Uneven Dataset Results

4.2.3 Full Dataset

We also conducted some experiments using the full dataset described in section (3.3). Due to the large training time of the full dataset, we did not have time to explore the hyperparameters from the beginning, thus, we tried some parameters that achieved the best results in the common dataset. Depending on the number of epochs we used, the training varies at 2-3 minutes per epoch. At first, this dataset gave us a loss of *inf* throughout the training process, although the validation loss decreased. In the deep speech forum they commented that a possible reason is faulty audio data. So we found the data and removed them. Note that on the following table the parameter batch size applies to train, validation and test dataset. The results are shown below:

#epochs	Batch Size	#hidden nodes	Learning rate	Dropout rate	WER	CER
70	32	350	0.0001	0.5	32%	14.6%
80	32	350	0.0001	0.4	29.5%	12.8%
15	32	350	0.001	0.4	61.8%	33.7%

Table 10: Full Dataset Results

5 Discussion

5.1 Experimental Results

From the experiments performed in both datasets, we are in position to comment on the effect different hyperparameters have on the training process of our network. Increasing the dropout rate as high as 0.3 and 0.5 respectively provides us with some of the best results of the model, where configuring the number of hidden nodes to the correct amount is also an important factor in moving the word and character error rates towards their minimum value.

Regarding the *Common Dataset*, one of the best performances was drawn from 340 hidden nodes and an amount of dropout equal to 0.5 when using the same rate in all layers, which essentially provides us with an ensemble of up to 2^N networks[4] (*N* stands for the number of total hidden nodes) when training the network. We were in position to get even better results on this dataset when tuning different values of the dropout rate in each fully connected layer. Starting with a relatively low value of 0.2 and increasing it to as high as 0.5 in the last 2 FC layers gave us the lowest value of both the Word Error Rate (20.6%) and Character Error Rate (9.1%) respectively when training solely on the *Common Dataset*. However, training the model in a subset of the **Full Dataset** and tuning the pre-trained model in the *Common Dataset* brought both error rates to the lowest observed values as shown in table (8).

Regarding the various forms of the *Full Dataset*, we observe that the best performance was drawn with dropout rate 0.4, which achieved a Word Error Rate of 29.5%. Moreover, the model is really sensitive to the learning rate as for a value of $lr = 0.001$ the model performed early stop at the 15th iteration and the results were significantly increased (61.8% WER).

In our early experiments, we also tried to remove data that where the duration of their sound files exceeded 10 seconds, in order to find out whether big samples play a negative role on the performance of the network. Our conclusion towards this direction is that no such effect is taking place.

Lastly, since the network and framework were tested and optimized in the English language, we expect to observe differences when applying the same architecture in a very different spoken language such as the Swedish language. For example, an artifact of the Swedish language is that in most spoken words, their syllables are verbally extended and the network understood them as separate words instead of consecutive syllables of the same word. In many cases, we observed an almost perfect prediction in terms of characters and syllables, which were placed as separate words leading the results to unexpected high errors. Some representative examples are the following:

True Transcription	Predicted Transcription	WER	CER
åsensbruk	*å sens bruk*	3	2
hermanson	*är man son*	3	4
hemmingsmark	*hem min smak*	3	3
nebraska	*när bra ska*	3	5

Table 11: Errors generated from the misinterpretation of syllables

While the whole word is interpreted correctly, understanding syllables as words leads to high Word Error Rate, and even Character Error Rate, since individual syllables are corrected from the language model to fit the N-gram models correctly. A possible reason for that result is the acoustic model. The acoustic model is responsible for representing the relationship between an audio signal and the phonemes. Thus, it may interpret the words wrong. Unfortunately the process is done by a Tensorflow method so we do not have access to see what happens when debugging.

5.2 Future Work

While the results that we extracted in both datasets are very encouraging, we feel there is room for improvements that can push the errors of the network predictions to their minimum amount. Future work should focus in the following directions:

- Further optimization of the hyperparameters of the network by using all the data from NST. This would mean access to heavier computational resources and more time available in order to be in position and execute a high number of experiments under several different configurations.
- Context-specific processing on the Swedish language. Finding the way to deal with the artifact mentioned in (5.1), will push the performance of the framework to optimal.
- Exploit the potentials of the *Deep Speech 2*[2] network in the same dataset, under a similar procedure with the one that was carried through this work.
- Exploit the potentials of Transfer Learning[5] and whether we can use and tune the trained weights for the English language in order to apply them in the Swedish language. This step would also have to deal with the the artifact of the syllables understood as separate words.

The first two suggestions can be integrated to our current work and build upon it, while the last two serve for a completely new approach on the same problem.

6 Conclusion

Through this work we presented the evaluation of the Deep Speech framework regarding its performance in speech-to-text operations on data drawn from the NST dataset for the Swedish language. This evaluation focused in two directions: in extracting results from training on data that are commonly used by the *wav2letter++* framework so that we can have a critical comparison between these frameworks and by evaluating Deep Speech in a big subset of NST that was possible under the constraints set by the available computational resources to us. Our work involved many tasks, ranging from data manipulation and pre-processing to hyperparameter tuning on the network for maximum efficiency. Our results in both ends are encouraging, and we believe that there is room for further improvement given more computational resources and time for experiments.

References

[1] A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep speech: Scaling up end-to-end speech recognition," *CoRR*, vol. abs/1412.5567, 2014.

[2] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Y. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Y. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu, "Deep speech 2: End-to-end speech recognition in english and mandarin," *CoRR*, vol. abs/1512.02595, 2015.

[3] N. Vanhainen and G. Salvi, "Free acoustic and language models for large vocabulary continuous speech recognition in swedish," in *LREC*, 2014.

[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jan. 2014.

[5] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, pp. 3320–3328, 2014.