Department of Electrical Engineering
IIT Bombay

EE691: R&D Project

Deep Learning for Prominence Detection in Children's Read Speech

Author:
Mithilesh Vaidya (17D070011)

Guide:
Prof. Preeti Rao

Collaborator:
Kamini Sabu

Spring 2020/2021

Abstract

Expressive reading, considered the defining attribute of oral reading fluency, comprises the prosodic realization of phrasing and prominence. The automatic detection of prominence in speech has a number of practical applications. In the context of evaluating oral reading, it helps to establish the speaker's comprehension of the text. We consider a labelled dataset of children's recordings for the speaker-independent detection of prominent words using acoustic-prosodic and lexico-syntactic features. A previous well-tuned random forest ensemble predictor is replaced by an RNN sequence classifier to exploit potential context dependency across the longer utterance. Further, deep learning is applied to obtain word-level features from low-level acoustic contours of fundamental frequency, intensity and spectral shape in an end-to-end fashion. Performance comparisons are presented across the different feature types and across different feature learning architectures for prominent word prediction to draw insights wherever possible.

1 Introduction

The main goal of the project is to predict the degree of prominence for each word in a given utterance, based on prosodic features. This task is considered as an intermediate step in predicting the oral reading abilities of children.
An auxiliary goal is to replace the 34-handcrafted features proposed in the journal paper [4] with a CNN feature extractor which takes as input raw acoustic contours. They are easy to compute. If such learned filters can attain the same (or better) performance, we can do away with complicated time-consuming hand-crafted feature extraction.
Note that this report is a short summary of our work which was submitted to Interspeech 2021. Please refer to [5] for a more detailed analysis.

2 Dataset

We have 807 recordings and a total of 42,100 words, which gives us about 52 words per recording on average. 35 unique speakers are present in this labelled dataset. For each word, we have the following hand-crafted features:

- Acoustic (159) from RFECV analysis
- Lexical (17) which include PoS, phrasing and prominence structure
- Pause and duration (12)

For more details about these word-level features, please refer to [4]. Thus, each word will be represented by a fixed-dimensional vector, depending on which of the above features are chosen.
We also have 15 (normalised and unnormalised) acoustic contours for the entire recording. With the help of word boundaries derived from forced alignment, we can extract a T_i x 15 matrix of features. Here, T_i need not be the number of frames in the word. Refer to section 3.2 for more details.
For each word, we have an integer label between 0 and 7 (both inclusive) which serves as the ground truth during training. These are manual annotations by experts and our goal is to predict these labels. We trained a regression model by predicting an output between 0 and 1 and scaled it back to the original scale. One potential future direction is to instead predict a 8-way classification output.

3 DNN models

3.1 RNN encoder

We use a simple RNN which takes as input the word-level feature embedding as input. The RNN output at each time step is fed to a simple feed-forward layer of input dimension H/2H (depending on whether the encoder is bidirectional) and output dimension 1, which is passed through a Sigmoid layer to get a final prediction between 0 and 1. We experimented with the popular RNN architectures: GRU and LSTM. The number of layers and hidden dimension were the two main hyperparameters which were tuned.

3.2 CNN feature extractor

3.2.1 Input to CNN

Recall from section 2: we have a matrix of acoustic contours for each utterance (call it $feat$). From the word boundaries (extracted from forced alignment), one can extract a slice for the i^{th} word depending on the following schemes (see figure 1):

1. If c_i is the centre frame of the i^{th} word, extract the slice $feat[c_i - k : c_i + k]$. Depending on the length of the word, some neighbouring frames might get included if the length of the word $< 2k$. Otherwise, we might lose some frames at the beginning and the end. k is a hyperparameter. Since the maximum length of the word in our dataset is 80, $k = 40$ is an ideal choice.
2. Let s_i and e_i denote the start and end frame of the i^{th} word. Extract the slice $feat[s_i - k : e_i + k]$. Such a scheme ensures that the entire word is captured, along with a neighbouring context of k frames. We tuned the value of k in the range: [10, 15, 25].
3. Let s_{-k} and end_{+k} denote the start frame of the k^{th} preceding word and k^{th} following word respectively. Extract the slice $feat[s_{-k} : e_{+k}]$. Thus, the surrounding k words are given as context.

After extracting the slice, we augment it with the following features:

- Positional Encoding: since the feature slice for our word in focus may contain frames from the surrounding words, it is essential to distinguish them. We experimented with:
  - 1-bit encoding which is 1 for the current word and 0 for the rest as proposed in [6].
  - A 3-bit encoding which is 100 for the frames of the previous word, 010 for the current and 001 for the following words.
  - 5-bit encoding which is similar to the previous one, except the pauses between words are also distinguished.
- Syllable numbering: From syllable boundaries, each frame is augmented with a 7-hot encoding where the i^{th} element is set to 1 if the frames correspond to the i^{th} syllable in the word. The maximum number of syllables in a word in our dataset is 7.
- We also experimented with a transformer-style encoding [3] which is a continuous extension of the one-hot encoded version. However, performance degraded substantially.

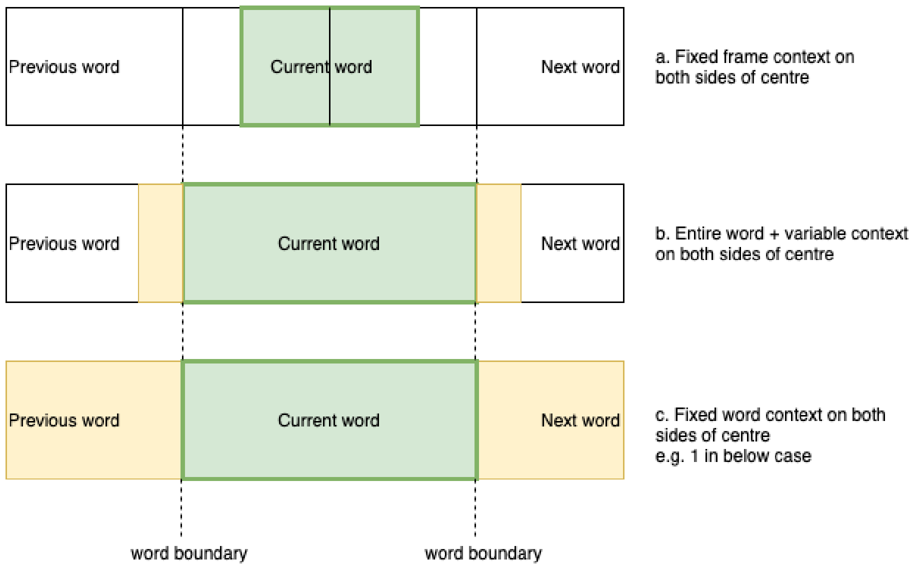

Previous word
Current word
Next word
a. Fixed frame context on both sides of centre
Previous word
Current word
Next word
b. Entire word + variable context on both sides of centre
Previous word
Current word
Next word
c. Fixed word context on both sides of centre e.g. 1 in below case
word boundary
word boundary

Figure 1: Various schemes for choosing word context, as discussed in section 3.2.1. We have time along the horizontal axis and features along the vertical. a, b and c refer to schemes 1, 2 and 3 respectively. Green refers to frames chosen from the current word of interest while frames in yellow denote the ones from neighbouring words which are included in the feature matrix of the current word.

3.2.2 Architecture

We have 1D convolutional filters of k different widths which slides over the feature slice for each word and output a T_i x N matrix where T_i is the number of input frames and N is the number of channels in the filter of a given width. The output is then max-pooled across time to get a N-dimensional vector for each of the k kernels. The vectors are concatenated to get a single Nk-dimensional encoding for each word. We tried out various configurations of filter widths: [5, 10, 25, 50] to capture varying levels of context and [25, 50] to capture only phone and syllable-level context. Since the results were similar for both, we decided to stick to [25, 50] since it has lower number of parameters. The number of channels N was varied in the range [8, 16, 32] and we found that 8 gave the best results. The stride was set to 1 for all filters. The core architecture is similar to that proposed in [7].

Instead of a single filterbank, we also tried splitting the 15 contours in 3 feature groups: pitch (4), intensity (4) and energy (7). The above architecture is used for each of the feature groups i.e. we get a Nk-dimensional vector from each of the feature groups, which is further concatenated to get a 3Nk-dimensional embedding. This splitting of features gave better performance than a single filterbank. Max-pooling across these 3 Nk-dimensional vectors to get a single Nk-dimensional vector did not give good performance.

Instead of a single convolutional filter, we also experimented with deeper architectures with the following structure: Conv1D, BatchNorm, Activation (Tanh/ReLU). These 3 units were repeated M times where M is the number of layers and skip connections were used to improve the gradient flow. However,

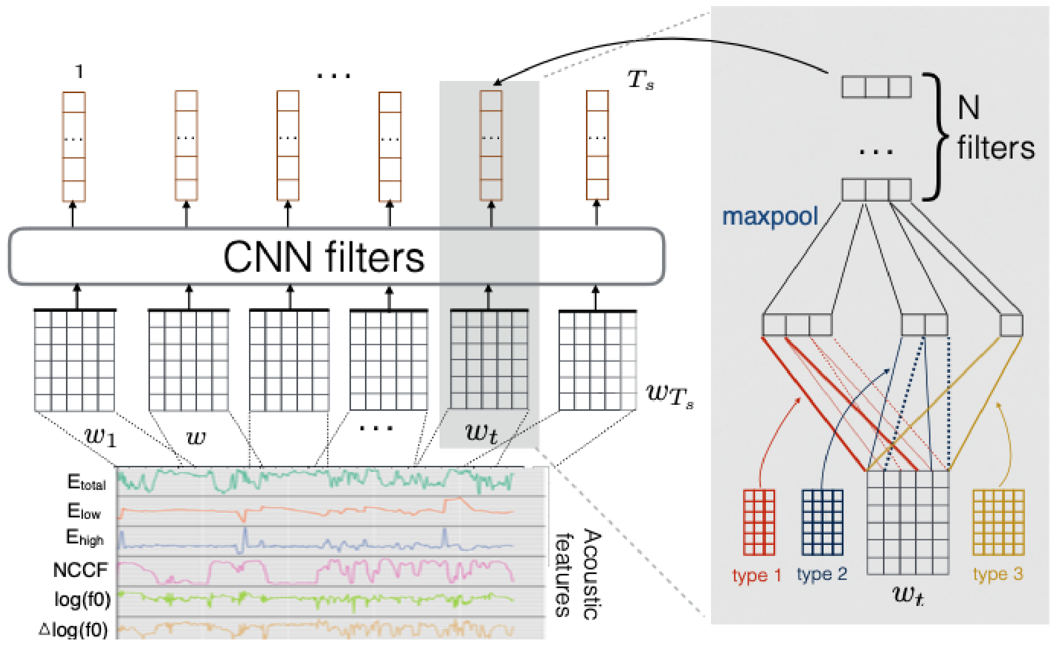


Figure 2: The zoomed in block on the right shows 3 filters of different widths. The result of sliding each of them across the input feature matrix is max-pooled to get a scalar value. They are concatenated to get a single Nk-dimensional encoding of the acoustic features. Figure reproduced from [7].

the results were not promising.
The final architecture is given in figure 3.

4 Experiments

4.1 Training Methodology

For a detailed explanation of the train-val-test split, please refer to [4]. All features are normalised by subtracting the mean and dividing by the standard deviation for each feature across the entire dataset. We experimented with Adam and AdamW as the optimiser. The learning rate was set to 0.003. No learning scheduler was used since Adam has an adaptive learning rate for each parameter. But this can be explored in the future.
MSE loss between the scaled ground truth and the model output was minimised. A batch size of 500 was used, based on the memory capacity of the GPU. Fine-tuning of batch size is a possible direction for future work. Weights of all layers were initialised according to the default scheme in [2].
Pearson correlation was used as the early stopping metric. We tested the performance of the model every 8 epochs. As soon as the difference between two consecutive Pearson values dropped below 0.005, we tested the model after every epoch and stopped training once the performance on the validation dataset dropped.
We used a single NVIDIA GeForce RTX2080 GPU with 12 GB of graphics memory for all our exper-

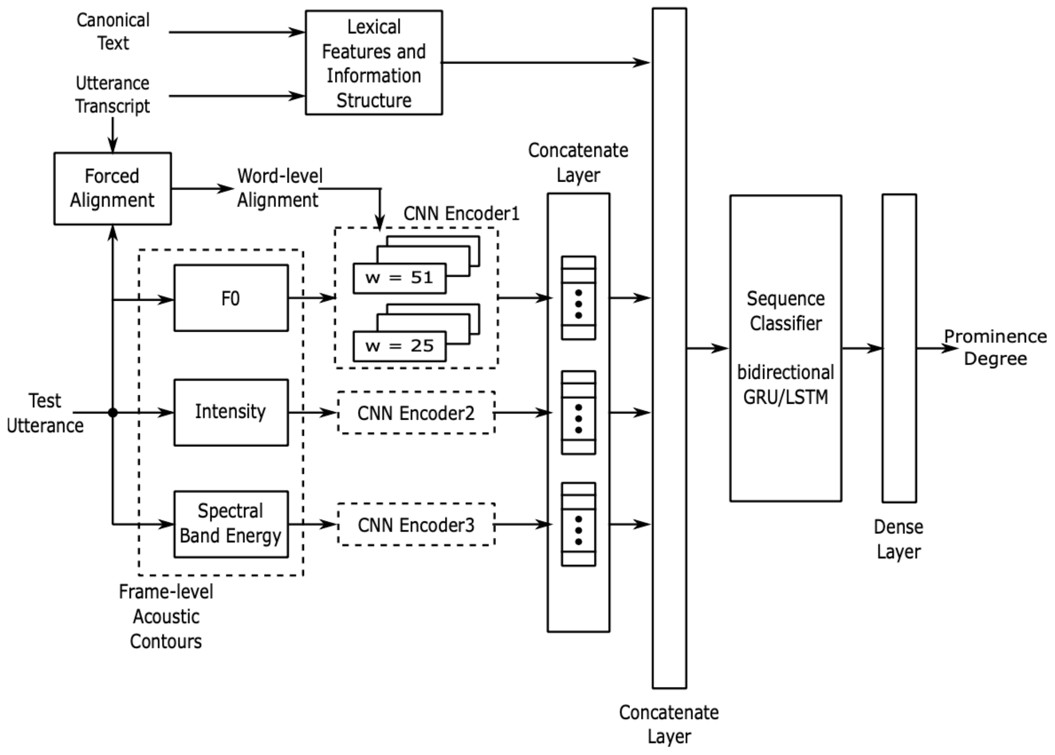
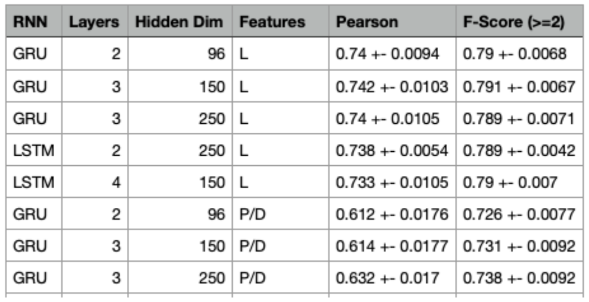


Figure 3: Figure reproduced from [5].

iments. The training time varied widely. For experiments involving only the word-level features and the RNN encoder, training on the 4 train folds took less than 120 seconds. On the other hand, after including the CNN feature extractor, the training time jumps to almost 45 minutes for the 4-fold CV on the train folds. This jump can be explained by the fact that each word in each utterance is treated as a distinct sample for obtaining the CNN output. As a result, if the i^{th} utterance has k_i words, the i^{th} batch has k_i samples. If M is the total number of utterances in our dataset, we will have M such batches (807 to be precise). Note that the weights of the CNN are **not** updated in this step. We are simply calculating the CNN outputs. Only the end-to-end training of RNN encoder and CNN leads to weight updates (which has a batch size of 500 as mentioned above).

4.2 Reporting Results

At the end of each experiment, we store all the model hyperparameters (such as hidden dimension, number of layers, etc.), training parameters (such as learning rate, batch size, etc.), the final result (such as Pearson correlation) and the timestamp in a JSON file. This helps in proper documentation and reproducibility. Refer to figure 4 for an example.



RNN	Layers	Hidden Dim	Features	Pearson	F-Score (>=2)
GRU	2	96	L	0.74 +- 0.0094	0.79 +- 0.0068
GRU	3	150	L	0.742 +- 0.0103	0.791 +- 0.0067
GRU	3	250	L	0.74 +- 0.0105	0.789 +- 0.0071
LSTM	2	250	L	0.738 +- 0.0054	0.789 +- 0.0042
LSTM	4	150	L	0.733 +- 0.0105	0.79 +- 0.007
GRU	2	96	P/D	0.612 +- 0.0176	0.726 +- 0.0077
GRU	3	150	P/D	0.614 +- 0.0177	0.731 +- 0.0092
GRU	3	250	P/D	0.632 +- 0.017	0.738 +- 0.0092

Figure 4: A JSON file with the column header as hyperparameters. Each row is appended after an experiment is complete. Note: not all columns are included in the figure.

For comparing models across architectures, we compute the mean and standard deviation of the Pearson correlation coefficients across the 4 training folds or 3 test folds. The mean and standard deviation of this list of results is reported in the table.
Ideally, one should run each experiment multiple times with different random seeds and average out the results over such runs. We couldn't do it, owing to time constraints.

4.3 Results

Given below are a few select tables from [5] from which we can draw some important conclusions.

Model	# layers	# units	Correlation	F-score
RFC	-	-	0.69*	0.63*
GRU	2	96	0.68	0.63
LSTM	2	256	0.69	0.63
BGRU	2	96	0.70	0.64
BLSTM	2	256	0.71*	0.64*

Table 1: Performance of various models for a set of 34 acoustic features (chosen from 159 after further RFECV). * indicates standard deviation < 0.01. Note that BLSTM is the best performing RNN model but it outperforms the baseline Random Forest Classifier (RFC) by only 0.02 in terms Pearson Correlation.

Features	Correlation	F-score
A34	0.70	0.64
A34 + L	0.75*	0.67*
A34 + L + I	0.79*	0.69*

Table 2: Performance with addition of lexical and information structure features. We can see a clear jump in both correlation and F-score. This indicates that complementary information is being brought in by the lexical and information structure attributes.

Features	Correlation	F-score
A34	0.70	0.64
CNN	0.69	0.63
CNN + D-P12 + A10	0.71	0.64
CNN + D-P12 + A10 + L + I	0.77*	0.68
A34 + L + I	0.79*	0.69*

Table 3: Performance of CNN encoding concatenated with different word-level features as input. Although CNN performance is equivalent to the A34 performance (first 2 rows), the 0.02 gap in the Pearson Correlation between the last 2 rows indicates that there is scope for further improvement for the CNN feature extractor.

5 Other experiments

5.1 Unlabelled Data

We used the RF classifier from the journal paper to label some more data. Only the high-confidence outputs were used to generate this dataset. However, on augmenting the labelled dataset with this new set and training the CNN feature extractor, the model performance dropped.

5.2 Speaker Embedding

We trained a separate neural network to predict the speaker (167-class classification) on both the labelled and unlabelled dataset (we do not need the ground truth prominence score, only the speaker identity for each recording). The activations of a bottleneck layer (whose dimension is a hyperparameter) were used as a speaker embedding. Note that this embedding was different for different utterances of the speaker. This embedding was concatenated with the word-level features and fed to the RNN encoder. There was no improvement in results.
A plausible explanation: we used the same 15 acoustic contours as input to predict the speaker. It is possible that the model learned similar filters as compared to the prominence score prediction case. Moreover, we can always expect some loss of information when we extract the bottleneck activations. This can be resolved by joint-training of the speaker classification model and the main prominence detection model. We can have 2 loss functions: MSE for the prominence score and CEL for speaker classification, which are appropriately weighted. In such cases, both branches can learn complementary information.

6 Conclusion

In this work, we have presented a deep learning model for predicting the degree of prominence for each word in a given recording of children's speech. We experimented with various schemes for providing context, positional encoding, CNN architectures and training methodologies.
The CNN feature extractor attained similar performance as compared to the hand-crafted features, which require extensive domain knowledge. This follows the general trend we see in today's deep learning landscape: replace hand-crafted features with more sophisticated architectures trained on more data. In our case, amount of high-quality training data may turn out to be a bottleneck and

hence semi-supervised approaches may be worth exploring.
As far as future work is concerned, the following directions seem promising:

- Replace RNNs with convolutional networks like TCNN or whatever the current state-of-the-art is for sequence-to-sequence models. Also, try models which feed the output y_t as input to the network for calculating y_{t+1} in an encoder-decoder fashion. Though this should be implicitly captured by RNNs, an explicit feedback might be more effective.
- Debug why deeper CNNs for acoustic contours failed to give better results. Ample literature talks about ways to circumvent the gradient issues e.g. skip connections, LayerNorm, etc.
- Unsurprisingly, the positional encoding gave a big jump in performance. Look for better encodings as compared to the simple one-hot encoded versions we've implemented.
- Try out different weight initialisation schemes, random seeds and other DL hacks to maximise performance.
- Jointly train the speaker embedding and prominence detection model as discussed in the previous section.
- Use multi-modal attention to weight the CNN acoustic features, word-level features, lexical feature since they all bring complementary information. Refer to [1] for more details
- Look for better stopping criterions and training methodologies in case of limited data.
- Try out classification instead of regression. Softmax probabilities give confidence and hence can be used for unlabelled data generation (similar to the RF case).
- Study sophisticated semi-supervised learning schemes to exploit unlabelled data more effectively.

References

[1] M. S. Grover et al. “Multi-modal Automated Speech Scoring using Attention Fusion”. In: *ArXiv* abs/2005.08182 (2020).

[2] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[3] “Posistional encoding in Transformers”. In: *kazemnejad.com* (). URL: `https://kazemnejad.com/blog/transformer_architecture_positional_encoding/`.

[4] Kamini Sabu and Preeti Rao. “Prosodic event detection in children’s read speech”. In: *Comput. Speech Lang.* 68 (2021), p. 101200. DOI: `10.1016/j.csl.2021.101200`. URL: `https://doi.org/10.1016/j.csl.2021.101200`.

[5] Kamini Sabu, Mithilesh Vaidya, and Preeti Rao. *Deep Learning for Prominence Detection in Children’s Read Speech*. 2021. arXiv: `2104.05488 [cs.CL]`.

[6] Sabrina Stehwien and Ngoc Thang Vu. “Prosodic Event Recognition Using Convolutional Neural Networks with Context Information”. In: *Proc. Interspeech 2017*. 2017, pp. 2326–2330. DOI: `10.21437/Interspeech.2017-1159`. URL: `http://dx.doi.org/10.21437/Interspeech.2017-1159`.

[7] Trn Trang et al. “Parsing Speech: a Neural Approach to Integrating Lexical and Acoustic-Prosodic Information”. In: Jan. 2018, pp. 69–81. DOI: `10.18653/v1/N18-1007`.