Speech Emotion Recognition using Recurrent Neural Networks

Pierre Cournut
cournut@kth.se

Abdullah Murat Buldu
buldu@kth.se

Abstract

In this paper, we suggest a speech emotion recognition system using recurrent neural networks. Our method comprises three steps: data collection, preprocessing and model design. First, we gathered a dataset from an open source database. Then we analyzed this dataset and built functions in order to extract features and organize it into feedable inputs. Finally, we built a recurrent neural network that takes features vector as input, extract high-level features from it, and interpret it by performing classification among the different emotions. We compare several recurrent architectures and different feature types as input in order to determine the model that performs best.

Index Terms: Speech emotion recognition, recurrent neural network, deep neural network

1 Introduction

1.1 Speech Emotion Recognition

With the progress of machine learning, especially with deep neural networks, came great improvements in speech recognition. But in speech emotion recognition, it is hard to find correct features to represent emotional states.

We denote several common approaches for speech emotion recognition. One of them is frame level and in this approach low level features are directly used for emotion recognition. Gaussian mixture models are used to generate the distributions for each emotional state[5]. An other approach is utterance level and in this approach some statistical functions are used to obtain global characteristic of each utterance from their low level features and some discriminative classifiers are used with these statistical values[4]. With the popularity of the deep neural networks over the last few years, some studies started using the deep neural networks for speech emotion recognition, which will be mentioned in the following section.

In this paper, we focus on recurrent neural networks as these networks are more appropriate for sequential data. With using Mel-frequency centrum coefficients (MFCC) feature from speech signals, we are aiming to reach the emotional information of speeches. We also made experiments with filterbank features (MSPEC) to see the impact of correlation between features when they are fed to recurrent neural networks. In addition to this, to include time evaluation of the feature vectors, we made experiments with overlapping windows of the features, we tried on both MFCC and filterbank features.

In the next section, we will mention previous works on speech emotion recognition with deep neural networks and recurrent neural networks. We will continue in the next part by describing our proposed framework. Then we will detail our experimental settings before presenting our experimental results. Lastly, we will give an interpretation of our results and discuss possible improvements in our framework. [3]

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

1.2 Related work

In the beginning, deep neural networks were used on the utterance level statistical features, so deep neural networks substitutes discriminative classifiers such as support vector machines. After those models, a combination of deep neural network and extreme learning machine (DNN-ELM) models were used for speech emotion recognition[1]. In these models, acoustic features are used on deep neural networks to get high level features. Then, some statistical methods are applied to high level features to extract utterance level global characteristics. A discriminative classifier is used to make the predictions with the utterance level characteristics. Extreme learning machine (ELM) is selected due to its simple form and less computation requirements. In ELM, first input data is projected into high dimensional space and after that one hidden layer is applied to get the emotional states.

In the recurrent neural network models for speech emotion recognition[3], recurrent neural networks are used to get high-level features. Then, similarly some statistical functions are applied and extreme learning machines are used to get emotional states. The main difference between DNN-ELM and RNN-ELM models is in DNN-ELM models, it is assumed that in an utterance every frame has the same emotional state but in RNN-ELM models, the emotional state of each frame is considered as a random variable.

2 Method

2.1 Dataset

To train our model and evaluate its performance, we used the Berlin Database of Emotional Speech (Emo-DB), which contains 535 utterances spoken by actors in a happy, angry, anxious, fearful, bored, disgusted and neutral way. Utterances can be chosen from 10 different actors (male and female) and 10 different texts (in German).

Each utterance consists of a *.wav* file which name contains information about the speaker, the text spoken, the conveyed emotion and the version number. In order to constitute our dataset, we built a function to extract those information from the filename. At this stage, the loaded data consists of:

- a sample
- a speakerID
- a textID
- an emotionID
- a repetitionID

2.2 Preprocessing

2.2.1 Features extraction

After loading the data, we extract MFCC and MSPEC features from each sample using the following steps:

1. Each sample is converted into an array of constant-sized frame samples
2. The Fourier transform is computed on each frame sample. We thus obtain at each time step, a graph representing the weight of each frequency in the total spectrum.
3. Mel filters are applied on the obtained power spectrum. They consist of triangular filters centered on different frequencies that constitute the Mel frequency scale. Log values of the amplitudes summed over frequencies for each triangular filter consist of MSPEC features (mel-frequency spectrum coefficients) .
4. The Discrete Cosine transform is then applied on MSPEC features in order to get uncorrelated features. Log values of the amplitudes of the obtained spectrum consist of MFCC features (mel-frequency cepstrum coefficients).

2.2.2 Dynamic features

In order to provide the model with information about the temporal context of each features vector, we computed first and second time derivatives of both MSPEC and MFCC features using the following formula:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{1}$$

where N is respectively equal to 1 and 2 in the case of first and second time derivatives and c_t denotes the feature to be derived at time step t.

We then stacked those additional features to each input vector. To measure the effect of dynamic features on emotion classification, we considered models trained on raw features, raw features with first time derivatives and raw features with first and second time derivatives.

2.3 Model

Our model takes as input a sequence of feature vectors and outputs an emotion ID which consists of an integer between 0 and 6. To perform this classification task from a sequence which can be of variable length, we considered a recurrent architecture for our neural network model.

2.3.1 High-level features extraction using a Recurrent Neural Network

We make use of a Recurrent Neural Network (RNN) to extract high-level features from input vectors sequence. The size of a sequence being variable forces us to feed the network one element of the sequence at a time. We thus have to make each iteration of the process interact with the others, which is where RNNs come into play.

Even though a RNN is fed one element of a sequence at a time, it achieves to capture the sequential aspect of the whole sequence by performing the same task for every element with the output being dependent on the previous computations. That sequential information is preserved in the RNN's hidden state that can be seen as the memory of the RNN given that it captures information about what has been calculated so far.

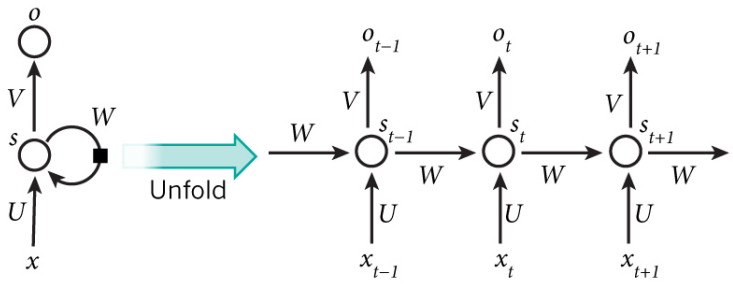

Figure 1: A RNN and its unfolded equivalent (Source: Nature)

The above diagram shows a RNN being unfolded into a full network, each layer corresponding to the processing of one element of the sequence. At time step t:

- x_t is the input.
- U, V and W are parameters matrices shared by all time steps.
- s_t is the hidden state or, as seen above, the memory of the network at that moment of the process. It is calculated recursively in the following way: $s_t = \phi(Ux_t + Ws_{t-1})$, where ϕ is a nonlinearity such as $\tanh$.
- o_t is the output at time step t, often calculated as follows: $o_t = softmax(Vs_t)$

However, an RNN can run very deep, each increase in the size of the sequence implying a new layer. This greatly amplifies the adverse effect of the vanishing gradient problem which led us to introduce Long Short Term Memory cells (LSTM cells).

LSTM cells [2] have a specific architecture that allows them to escape vanishing gradients and keep track of information over long time periods. They differ from basic RNN cells with the presence of a cell state and how hidden states are computed. With basic RNN cells, we have a single layer where $s_t = \phi(Ux_t + Ws_{t-1})$. With LSTM cells, there are four layers that interplay to form a gating mechanism which removes or adds information from or to the cell state. This gives the network the ability to remember or forget specific information about preceding elements.

2.3.2 Classification using a fully connected layer

On top of this RNN structure, we built a fully connected layer to interpret high-level features. It takes as input the output of the last RNN cell, and outputs a probability distribution over the different classes.

It consists of a dense layer where each node is linked to the nodes from the previous layer using trainable weights. To normalize the raw output vector z to a probability distribution p, we use a *softmax* activation.

$$\sigma(z) = (\frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}})_{1 \leq j \leq K} \tag{2}$$

where K denotes the number of classes.

To compute the accuracy of our model, we only consider the class with the highest probability.

2.3.3 Regularization

Regularization is commonly used in machine learning algorithms to avoid overfitting which happens when a model is too closely fit to the training data. We combined two different regularization techniques.

First, we added a regularization term in the cost function. This positive term which grows with the complexity of the network, becomes a burden on the cost function which we want to minimize and preserves the generalization of the model. We use L2 regularization on the fully connected layer.

Second, we added dropout to some of our layers to prevent our model from overfitting. Dropout is applied to a specific layer and consist in randomly deactivate units in that layer during a particular forward and backward pass. We used dropouts in our input and hidden layers.

2.4 Training

In our training we aimed to minimize the categorical cross-entropy with the regularizations mentioned in the previous sections. We used Adam optimizer to minimize the cost. For training we used batches with size 128.

3 Experiments

3.1 Data

3.1.1 Validation and testing set

We divided the 535 utterances that constitute our dataset into a training, a validation and a testing set. To ensure the validity of our generalization performance, each speaker is only included in one of the three sets.

Set	Number of utterances	Part of the entire dataset
Training	408	76.3%
Validation	56	10.4%
Testing	71	13.3%

Table 1: Dataset constitution

3.1.2 Sequence representation and data augmentation

As in [], we stack 25 consecutive features vector in order to obtain sequences that consists of context window with the size of 250 ms. We obtain sequences of input vectors which sizes are presented in Figure 2.

Feature type	Vector size
MFCC	(25, 13)
MFCC + First time derivatives	(25, 26)
MFCC + First and second time derivatives	(25, 39)
MSPEC	(25, 40)
MSPEC + First time derivatives	(25, 80)
MSPEC + First and second time derivatives	(25, 120)

Table 2: Input size for the different feature types

In order to increase the size of our dataset, we introduced a shift parameter that controls the overlapping between consecutive sequences. We considered different values for this shift parameter and finally set it to 3 which gives us an overlapping of 88%. It can seem as a big value but due to the small size of the initial dataset, the effect of that data augmentation on final performance outweighed overfitting problems that surged with such redundancy in our opinion.

3.2 Model

We implemented our model using Keras 1.2.0.

Our model contains 2 long short-term memory layer with 128 nodes and between the layers we added dropout to avoid overfitting.
On top of those 2 hidden layers, we added a fully connected layer with softmax activation to get the probabilities for each emotion.
We used Adam optimizer to train our network.

After several cross-validation experiments, this parameterization of the model is the one that gave us the best results and we thus used it as a basis for all experiments:

- Dropout probability $= 0.2$
- L2 regularization term $= 0.0001$

4 Results

4.1 Evaluation

First, we will present two tests that we did to select our model and their associated results. The tests concern the feature type used as input and the architecture of the model. We evaluated the performance of our different models by measuring the accuracy on the testing set. Given that we had multiple classes (7 in total), we considered over 50% accuracy to be already convincing results.

Then, we will go over the settings of our best model and detail its performance and results statistics.

4.2 Impact of feature type

In order to measure the impact of dynamic features, we trained several models with the same architecture (2 LSTM hidden layers + 1 fully connected layer) and the same hyper-parameters but with different input feature types:

- raw MFCC and MSPEC features
- raw features and their first time derivative
- raw features and their first and second time derivative

Models were trained for 10 epochs and we considered the final accuracy on the testing set.

Feature type	Accuracy
MFCC	53.1%
MFCC +Δ	53.1%
MFCC +Δ+$\Delta\Delta$	54%
MSPEC	54.5%
MSPEC +Δ	56.6%
MSPEC +Δ+$\Delta\Delta$	57.8%

Table 3: Accuracy on testing set for different feature types

The use of dynamic information contained in first and second time derivatives helps the network train and perform better. Even if we did not detail the results in this report, we also tried stacking raw features in order for a feature vector to contain information about the past and the future but we obtained poorer performance than when using derivatives.

4.3 Impact of the model architecture

In order to measure the impact of the model architecture, we trained several recurrent models with the same number of nodes in the two hidden layers but with different cells. We made experiment with:

- simple Vanilla cells with 128 nodes
- LSTM cells with 128 nodes
- bidirectional LSTM cells with 64 nodes

Feature type	Vanilla	LSTM	Bidirectional
MFCC+Δ+$\Delta\Delta$	48.4%	54%	57.7%
MSPEC +Δ+$\Delta\Delta$	51.7%	57.8%	57.7%

Table 4: Test accuracy using Vanilla cells, LSTM cells and bidirectional LSTM cells

The use of Long Short-Term Memory cells and a Bidirectional architecture also helps the network perform better. As the use of Bidirectional LSTM cells did not improve results significantly, we did not retain it for our further tests.

4.4 Best model

Following the tests about input feature type and model architecture described above, our best model takes as input MSPEC features and their first and second time derivatives. It then comprises 2 hidden layers that consists in LSTM cells and that extract high-level features. Those high-level features are then interpreted by a fully connected layer that produces a probability distribution over the different output classes (possible emotions).

Figure 2: Diagram of our best model architecture

Our best model achieves 57.7% accuracy on the testing set and we also assessed its classification performance using its confusion matrix drawn below.

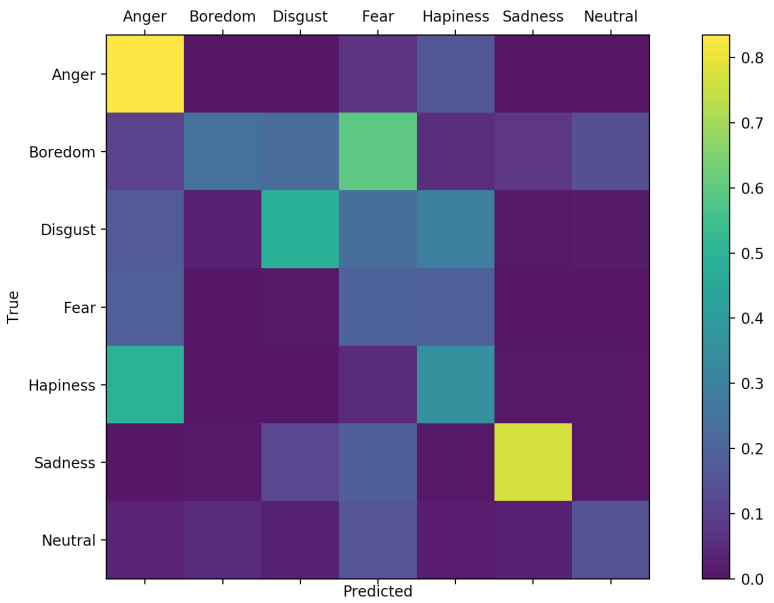

Figure 3: Confusion matrix of our best model

As expected given the accuracy performed on the testing set, this confusion matrix has a quite strong diagonal that traduces the ability of our model to predict correct classes most of the time. More precisely, anger and sadness are the most correctly classified emotions with more than 80% accuracy. However, we observe that our model struggles to distinguish happiness from anger and boredom from fear.

5 Discussion and Conclusions

5.1 Results interpretation

Concerning the overall results we obtained when evaluating our different models, it seems like Recurrent Neural Networks are a good way to extract and interpret high-level features from audio samples in order to recognize speech emotion. Since we performed classification over 7 different classes, accuracy of the order of 58% are already convincing results in our opinion.

More precisely, our different tests lead us to select a recurrent architecture based on LSTM cells and taking as input MSPEC features and their first and second time derivatives.

5.2 Improvements

Even though we were satisfied with our overall results, we really struggled to prevent our model from overfitting. In order to avoid this phenomenon and to improve our framework, we could have enquire the use of following techniques in further developments:

- extract and add new features such as the pitch, the voice probability or the zero-crossing rate
- use the Extreme Learning Machine paradigm as in [1]
- try other data augmentation techniques such as adding random noise to samples
- fine tuning our best model and our optimizer using a grid-search

6 Appendix

Following the review our draft report received, we made the following improvements:

- To provide a better idea of our model results and classification performance, we included the confusion matrix of our best model which detail results and statistics as suggested.
- To enhance the understanding of our model architecture, we described our best model in more details and added some graphical interpretation (diagram) as suggested.
- Even if we agree with the reviewer on the fact that we did not bring much novelty in the domain of speech emotion recognition, we were unfortunately not able to totally change our model and report to make new experiments.

References

[1] Han, Kun, Dong Yu, and Ivan Tashev (2014), "Speech emotion recognition using deep neural network and extreme learning machine." In *Fifteenth Annual Conference of the International Speech Communication Association*.

[2] Hochreiter, Sepp and Jürgen Schmidhuber (1997), "Long short-term memory." *Neural computation*, 9, 1735–1780.

[3] Lee, Jinkyu and Ivan Tashev (2015), "High-level feature representation using recurrent neural network for speech emotion recognition."

[4] Mower, Emily, Maja J Mataric, and Shrikanth Narayanan (2011), "A framework for automatic human emotion classification using emotion profiles." *IEEE Transactions on Audio, Speech, and Language Processing*, 19, 1057–1070.

[5] Neiberg, Daniel, Kjell Elenius, and Kornel Laskowski (2006), "Emotion recognition in spontaneous speech using gmms." In *Ninth International Conference on Spoken Language Processing*.