Listen, Attend and Spell

A neural network for large vocabulary conversational speech recognition

by William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals

Summary and critical analysis by Arnaud Capitan

Summary and description of the approach used:

Speech recognizers often rely on Deep Neural Networks (DNNs) to improve the efficiency and accuracy of the algorithms used. Common models mix DNNs with Hidden Markov Models (HMM), which essentially means that for each phoneme of a sound, the model extract features and returns the transition and emission probabilities: based on that, it maps the best path for an audio through the model to words or sentences.

This paper introduces LAS (Listen, Attend and Spell), a neural network based on sequence-to-sequence learning with attention, that doesn't rely on HMMs: it consists of an encoder, the *listener*, which is a pyramidal RNN that maps speech into features, and a decoder, another RNN that does the opposite and outputs a probability distribution for utterances using attention mechanism. Both RNNs are trained jointly. The model relies on the attention mechanism, which allows to focus on specific parts of the input to produce step-by-step the best output in terms of probability. The training of such sequence-to-sequence framework works by giving groundtruth labels as inputs for the decoder. During inference, the model finds the most likely candidates for the next steps. We have a formal description of the model, which states how the two modules work, both the listener and the speller.

The first one, whose goal is to **listen**, transforms the input sequence of filter bank spectra features (of length T) into a high-level representation vector (of length $U \leq T$) using a pyramidal Bidirectional Long Short-Term Memory (BLSTM) RNN. This pyramid structure allows the model to converge relatively quickly, otherwise the error rates would remain high enough to judge the model inefficient. Concatenating the outputs at consecutive steps of each layer allows to reduce the time resolution by a factor of 2 before feeding into the next layer: which means that training multiple layers will allow to significantly reduce the initial length of the input signal into very few high-level features (hundreds compared to tens of thousands initial frames). Most importantly, nonlinear features emerge from those multiple layers, which allows to represent better the data.

The second one, whose goal is to both **attend and spell**, uses an attention-based LSTM transducer to predict the next character. It is based on:

- The decoder state, which relies on a 2-layer LSTM RNN
- The context vector, computed with the attention mechanism using scalar energy, itself computed using Multi-Layer Perceptron (MLP) Neural Networks applied on both the decoder state and the listener features and linearly blending the listener features at different time steps
- The previously generated characters to deduce from the decoder state and the context vector the distribution of the next character, which uses a MLP with softmax outputs over characters

The attention mechanism allows the model not to overfit the training data despite the large training set of three million utterances mentioned in the paper.

This describes globally how the LAS speech recognizer model works. Now we need to assess how the model is trained, and how inference from speech signal is done.

Now, for the training: one of the key elements is that both the encoder and the decoder are trained simultaneously using a sequence-to-sequence objective, using the groundtruth of previous characters to maximize the log probability of the next character. To reduce overfitting of the speller, meaning that it only sees during training correctly guessed previous characters, they used a sampling trick that sometimes (10% of the time) took samples from the previous character distribution to simulate the missing previous groundtruth for inference.

To apply the model, for inference, we want to find the most likely character sequence given the input acoustics, using left-to-right beam search. By only keeping a fixed (beta) number of the most likely sequences at each step, the algorithm expands the partial hypothesis at each step with the most likely possible characters. Once the <end of signal> token is reached, the most likely sequence is kept as output. The paper mentions a constraint over the search space thanks to a dictionary, but it was seemingly not needed as the model is efficient enough and spell real words “almost all the time”.

In the end, they use language models to rescore the beams since its easier to train them as they have a lot more text data than transcribed speed utterances to train the LAS model. Using those efficiently trained language models, they noticed a bias for shorter utterances, which they corrected by normalizing the probabilities by the number of characters.

Finally, the paper presents experiments, conducted on Google voice search data, with 3 million utterances for around 2000 hours of audio, and 22’000 test utterances. There are performances comparation between LAS, LAS and Language Model vs the state-of-the-art (at this time) CLDNN-HMM, and they respectively get 14.1% WER, 10.3 WER% and 8.0% WER, thus showing that it needs to achieve so more progress to be the best speech recognizer model.

With the example sentence “How much would a woodchuck chuck”, the authors showed that the model learns monotonic alignment (based on the attention mechanism) without explicit constraints. Since we can see it with the graphic representation, it also shows the interpretability of the attention mechanism, while still revealing confusion between, similar words (here, woodchuck and chuck).

Such confusion dilutes the attention of the model, which may lead to an accumulation of errors in the case of phonetically complex sentences. They also studied the impacts of beam width, utterance length, a word frequency analysis, and some interesting decoding examples (‘triple a’ vs ‘aaa’).

Overall the model performs great.

<u>Critical analysis of the adopted method focusing on the pros and the cons:</u>

The pyramidal structure allows to reduce the length of the input (which can be thousands of frames long) to only key high-level features for the model to predict the output sequence, and with a high accuracy doing so. This allows both faster computation times as it reduces the dimension, and more complex features as there are multiple layers in the pBLTSM of the encoder.

They found out by experimentation that despite the learning issue mentioned – when during the training inference some groundtruth were missing – they did not require pretraining, which saves computation time. No improvements were made using a multi-frame phoneme state prediction, as well as using the phonemes as a joint objective target. All of this means that the researchers who came

up with this LAS method explored diverse ways to improve both the accuracy and the computation time of the algorithm, eventually without success, but they still pushed their algorithm to its best

The LAS model remains with a word error rate of 10.3%, compared to the best CLDNN-HMM (Compute Library Deep Neural Network – Hidden Markov Models) algorithm available which score 8%-word error rate on clean test data. One thing I noticed is that they did not say using a dictionary to constraint the search space was inefficient, unlike they mentioned with the multi-frame phoneme state prediction or the phonemes as a joint objective target; they just brushed it away saying that the model was efficient enough not to require a dictionary. Maybe it is in fact needed, and the few percent of difference between the best model and this one comes from this slight difference in training.

The impact of the model:

This LAS paper was released in 2015. This kind of framework, according to the paper, has also been extended for machine translation, image captioning, parsing and conversational modelling. But most importantly, was it used for speech recognition as intended? Was the algorithm efficient enough despite its slight worse error rate to be used and upgraded for today's use?

LAS algorithm is mentioned in the Google's Universal Speech Model (USM), referred to as the USM-LAS. It uses a conformer-based encoder to process speech into high level features (like in this model), and it uses different algorithms for decoding, one of which is LAS. USM-LAS is said to achieve state-of-the-art word error rate across many languages, having relatively a 32.7% less WER than Whisper, the model used for speech recognition by OpenAI.

We can thus say that this model was pretty useful and efficient, as it is one component of surely the most efficient speech recognizer in the world (currently according to the data presented by Google, but of course they are going to say their algorithm outperforms everyone else's).

https://research.google/blog/universal-speech-model-usm-state-of-the-art-speech-ai-for-100-languages/

Possible improvements:

They mentioned the coefficients obtained by the softmax being typically very sharp, and focused on only a few frames on the features of h : If we were to activate more features of h, or on the contrary reduce the number of frames of h, we could use the function softmax temperature, which could change the probability distribution to either trigger more features, or reduce its number, depending on the goal (quick convergence or exact reconstruction with more features, careful with overfitting as it was already an issue as mentioned in the pros and cons).

As mentioned previously, the dictionary to constraint the search space may be in fact efficient, and make the model better during the training : they did say that the model was efficient enough to output real words "almost all the time", and that looking at the examples provided at the end of the paper the mistakes come from wrong words and not inexistent words, but maybe the accuracy of the model would improve by using a dictionary.

I chose this paper because it was a direct application on DNN we learnt in TSIA 207 applied in the growing context of speech recognition and speech synthesis. Here we do not have speech synthesis in this paper, but in the course and during the lab session we saw how hard it was to correctly predict phonemes in a sound to then apply such model to predict the words.

This paper presents a seemingly efficient method to correctly do speech-recognition (as shown in the article by Google, it is in fact really good), but the month-long training time mentioned in the paper,

as well as the data required, makes me realize that those kinds of models are only available for big corporations with lots of data and computation power / time.

The formal description of the model is available in the Jupyter Notebook.