Improve Speech to Text application with transformers Layer

CS6536 Guided study in Data Science

Malo Ferriol

40101063

Supervisor: Dr. Antoni Chan

Department of Computer Science

City University of Hong Kong

Abbreviations

Automatic Speech Recognition	ASR
Bidirectional Encoder Representations from Transformers	BERT
Language Model	LM
Hidden Markov Model	HMM
Masked Language Model	MLM
Word Error Rate	WER

Contents

1 Introduction

Humans have developed the spoken language as their primitive mean of communication. Later the written language was created to record and share knowledge. Computer science is using written text and information. Since the beginning of computer research where conducted to automatically transcript the speech into written text.

The ASR system have different component for specific task, first it analyse the signal then it model a text sequence to output a sound result.

The early ASR system used pronunciation and acoustic model to recognize phoneme or words. It required an in-depth knowledge of the problem and very specialized components to obtain good result. Those system where not well robust to noise. In addition they used Hidden Markov Model. Once the phoneme identified the HMM is used as statistical model to calculate the most likely word composed from the phoneme. The HMM is trained with a vocabulary of words of possible output but a language model such as n-gram is required to obtain a sound sentence.

The issue with this system is that the components are distinct and cannot be trained together. The acoustic model identifying the phoneme requires labeled data identified by specialist.

Those acoustic model where then replaced by Deep Neural Network. This allowed a better automatization and required less in-depth knowledge. New feature could be used such as MFCC, filter banks or spectrogram.

The breakthrough in ASR came with the first End-to-End system. Its performance were not superior to the HMM based model but it was the first model to be able to train all together. Baidu research released a paper about their DeepSpeech model [1]. It is one of the first open source End-to-End ASR system.

This was made possible by the use of the Connectionist Temporal Classification algorithm. The system is able to learn the pronunciation and acoustic model for both DNN and CTC. The DNN is usually composed a RNN. However the CTC does not output sentences but only character. The model learns to map audio signal to character. Therefor a language model is required to improve the model and correct simple mistakes.

The language model often used is the n-gram model. The main issue is that a word will be contextualized only to its n neighbour and not the entire sentence.

In the field of Natural Language Processing (NLP) there has been many important achievement over the last few years. Google research shared a paper on the model BERT [2]. The model is not a language model but it is the most performing Natural Language Understanding model. One of its key advantages is its self-attention mechanism. It allows the model to be able to train on unlabeled data. The BERT model has been trained on more than 2 Terabytes of raw text. The Masked Language Model (MLM) of BERT is trained to contextualize and identify a masked word in a sentence. With the Bidirectional Transformers it is capable of analyzing the context of a word in regards of every other word in the sentence.

The aim of the project is to use an End-to-End ASR system with the MLM BERT model as a re-scoring language model.

The ASR system with RNN and CTC will output the distribution of the character recognize rather than a sentence. The number of possible distribution can become very large so we will select the n-most likely distribution using beam search.

The adaptation of the NLU model is not a straight forward task because it is not adapted for it. A recent paper "Effective sentence scoring method using BERT for speech recognition."[3] propose a solution in order to use BERT as a language model. A research team from AWS later published a paper [4] and an implementation of this scoring method.

The scoring result of the BERT model will be used to rank the output of the ASR.

The transformer based encoder decoder model is very powerful. In recent years it has been used in Neural Machine Translation task. Transforming an input sequence into another. It is widely used for language translation. We will use this architecture for another purpose. We will try to develop a new model that will correct the output of the ASR system in order to improve its result.

With both the implementation of DeepSpeech from Mozilla and of the scoring method from AWS we can look at the viability of using out-of-the-box models to created an ASR system. We will also compare the performance of the system in regards of the pre-trained BERT model we choose. We will use the original BERT model but also the multilingual version and the Large version.

In addition we will look at

In this project we experimented on the rescoring task using BERT model. The result improved the performance compared to the ASR system without the use of Language Model. However the result remained inferior to the original model with the n-gram model.

In addition the experiment showed that the use of transformer based encoder decoder to correct the output of the ASR reranked output did not improved the performance. On the contrary the performance drop of around 1%.

2 Background

2.1 History

The first apparition of a speech recognition system could be find as early as 1922 when an American company commercialized a product which answer to a single word (Rex). The system actually only detect a single frequency and not the word itself. The research for speech recognition started as early as the mid 20th century. The first system could be found in the Bell lab in 1952. It was aimed at recognizing digit in a closed environment free of noise. During that time the fundamentals of the speech recognition task were set. Progress were made in the 1960s when systems were able to recognize small vocabularies (10 to 100s of words). It required that the words be uttered one by one, and the recognition was based on the acoustic and phonetic properties of the speech sound. They used filter banks and other methods such as time normalization. In the next decade progress was made using patter recognition, the size of the vocabulary grew to 100s to a 1000 of words.

In the 1980s a breakthrough was made possible using statistical models. The vocabularies were not limited anymore from 1000 to unlimited number of words. It was made possible using Hidden Markov Model (HMM) in addition to stochastic language model (n-grams model). The systems were powerful enough to handle continuous speech. During the next decade the acoustic models and language models were largely improved. The hardware and their implementation as well. This allowed the better use of larger vocabularies.

In the 2000s the speech recognition systems performance achieved such good results that they could be commercially used. The most notorious example is Siri the vocal assistant developed by Apple.

Automatic Speech Recognition system are still a very active field of research and only recently some paper have demonstrated that their performance are comparable or even better than human.

2.2 Type of speech

The audio sample used for ASR are usually resumed by four types.

2.2.1 Isolated Words

This is when the speaker pause between the utterance of each word, marking a significant silence. The system is still able to analyse sample with multiple words however it requires each word to be uttered separately.

2.2.2 Connected Words

This is when the speaker mark small pause between the utterance of each word. This pause is short. The word are uttered separately.

2.2.3 Continuous Speech

This is when the user is speaking naturally. The speech is dictated carefully and distinctly.

2.2.4 Spontaneous Speech

This is when the speech is not rehearsed and natural. It could be a flawed speech containing shutters such as "euh", "uhm", "ah".

2.3 Characteristic of an Automatic Speech Recognition system

ASR systems have multiple property depending on the purpose and performance. It is defined by the vocabulary of words it can decode. The system can be speaker dependant if it works only one known speaker or independent if it can achieve result regardless of the speaker. The type of speech that can recognize is characteristic to the system whether it is isolated or continuous speech. Another property of a system is its capabilities to recognize a speech in adverse condition or not. The last characteristic of a system is the purpose for which it was built, is it a specialized or task dependent system or is it built with a general purpose.

2.4 Challenges of speech recognition

A speech is an audio signal. A speaker in different condition will produce different waveform. It could be due to the variation of space, speaking rate, volume, accent, pitch. In addition different person can have different accent or dialect. The variation can also come from the microphone source or the noise in the background. The key objective of an ASR is to be robust to such conditions. Further more once the audio is converted the ASR also needs to be robust to the decoder/Language model mismatches to produce sentences. Finally the most challenging task will be to be able to develop a multilingual model.

2.5 Components of Automatic Speech Recognition system

2.5.1 Audio sample

Digital version of an audio sample. It is a discrete signal.

2.5.2 Acoustic Model

An acoustic model will take an audio sample as an input and will output a statistical distribution of the possible transcript as an outcome. It maps the signal to linguistic unit. Until recently the output were phoneme or words. Most models now will be character based or bites based. For a language with the latin alphabet a character will be sufficient but for more oriental language such as Mandarin in order to output chinese character a bite based system is better suited. It will first extract feature from the signal, using for example MFCC or the spectrogram of the signal. Then it could use a statistical model such as HMM or a Deep Neural Network (DNN).

2.5.3 Decoder

The final task of an ASR is to convert the output of the Acoustic Model to a sound sentence. Most modern decoder use several algorithm combined to perform the task. First it uses the Connectionist Temporal Classification algorithm to map the character to a distribution of words. Then perform a Beam search on the distribution probability with an n-gram language model to find the most likely.

3 Overview

3.1 DeepSpeech

The model DeepSpeech has been developed by Baidu research. They published a first paper describing one of the first End-to-End Deep Learning Automatic Speech Recognition model. Out model is an implementation done by Mozilla Research inspired by the Baidu paper. The architecture is slightly different.

3.1.1 Acoustic Model

The model apply feature extraction on a window and on C adjacent window. The input uses MFCC feature. The feature size depends on the sample rate 13 feature if its 8 kHz and 26 feature if its 16 kHz. Then apply 3 layer Fully Connected NN which will feed an LSTM cell. The recurrent layer is computed sequentially. It is a uni-directionnal feed forward in time LSTM layer from the next window (time-step). Then for each time frame it feeds a FC layer which use a Relu function to transform the result. The last layer is used to find the probability for each possible outcome at each time step. The outcome is either one of the 26 character of the alphabet, a space, an apostrophe or a blank $\{a, b, c, ..., z, space, apostrophe, blank\}$. We will see later the use of the blank character.

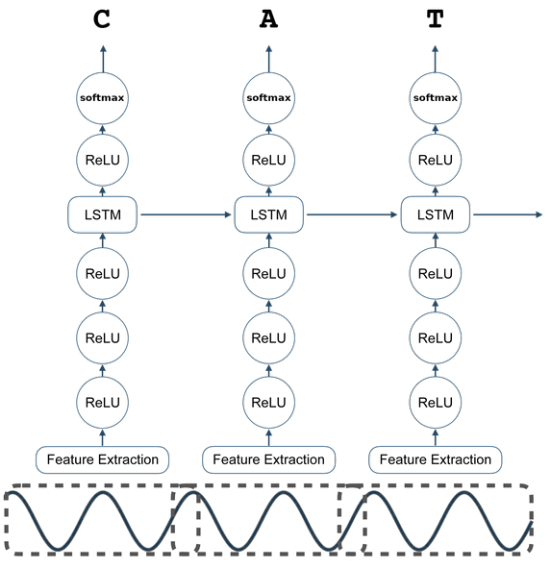

Figure 1: DeepSpeech architecture

3.1.2 Connectionist Temporal Classification

The output is a sequence of character which is used as input for the Connectionist Temporal Classification (CTC) in addition to a beam-search which will produce the output of the system. It is usually used with a Language model in order to improve the output. The Mozilla implementation of DeepSpeech uses the KenML stochastic language model using n-gram model.

During the training phase the audio sample and its transcript are not align. The speed of the speech can diverge and a speaker can vary its speed during the same audio recording. It makes the task much more difficult because the length of input and output can vary, and their alignment is unknown. Those difficulties prevent us from using simpler model. The CTC goal is to overcome those challenges and provide all the possible distribution of character for a given audio sample.

The loss function of CTC is the conditional probability that a distribution Y is the right answer for a given input X.

$$p(Y|X) \tag{3.1}$$

The loss function is required to be differentiable in order to be able to use gradient descent for training.

The objective function is :

$$Y^* = argmaxp(Y|X) \tag{3.2}$$

The output of the Acoustic Model is a list of character. Each character can be found multiple times consecutively. The simplest way to get around that problem is to concatenate the same character into one when they follow each other in the sequence. This creates a new problem when a word contains two repeating character in a row such as "hello". It could also yields issue when the end character of a word is the same as the starting character of the following word it could end up in a loss of information.
This is why the "blank" character was introduced.

First the algorithm collapse the character that follow each other and then it removes the blank character that does not correspond to anything in the output.
The algorithm uses Dynamic Programming in order to reduce the computational complexity. CTC merge the distribution when they produce the same output. Then the Beam search is performed in order to find the outcome with the highest probability.

Again the complexity is reduced for the distribution probability by reducing the size of the search tree. Rather than following all the possible arc it goes through the most likely and tries to find the depth as fast as possible.

3.1.3 Language model

We can add a language model to obtain a better result. A Language Model (LM) is a stochastic model which calculate the probability of a word based on the previous one.

Let's note Y^* the optimal solution of Y

The objective function become the distribution probability and a N-gram Language Model.

$$Y^* = \operatorname{argmax} \underbrace{p(Y|X)}_{\text{The CTC conditional probability}} * \alpha \overbrace{p(Y)}^{\text{The Language Model probability}} \tag{3.3}$$

where

α is a hyperparameters.

The language model improve the outcome and helps produce better sentences.

3.2 BERT

The model BERT is a Bidirectional Encoder Representations from Transformers model. It is a Natural Language Processing technique. The aim is to understand the language. Usually NLP task are trained on large labeled data set.

Previously to contextualise a word embedding model was used, such as word2vec. It would map every word in a vector representation. The idea behind it is to associate words with close meaning. The main issue is due to the fact that word can have different meaning therefor the representation should differs when the context does. Another constraint is that such representation requires labeled data. It is not adapted for all NLP task.

From its conception BERT is different. The BERT model is trying to predict a word in sentence. Therefor it does not require labeled data. The original model is pre-trained on text corpus such as the entire English Wikipedia and more.

This mechanism also make the model improve and train on the unsupervised and unlabeled data used in production. The pre-trained model is considered as a base for Natural Language Understanding. It can be used for fine-tuning for a specific task

3.2.1 Transformers

The architecture differs from the Deep Learning model usually used for NLP by its use of the transformer. It enables the model to use a self-attention mechanism. It makes BERT very powerful by analysing a word in a sentence with regards to every other words in this sentence. The transformer helps the model understand the word in its context which is a challenge for most DL model.

The use of bidirectionnal Transformers made it possible for BERT to understand the meaning of a word in its context which can change drastically for the same word.

3.2.2 Masked Language Modelling

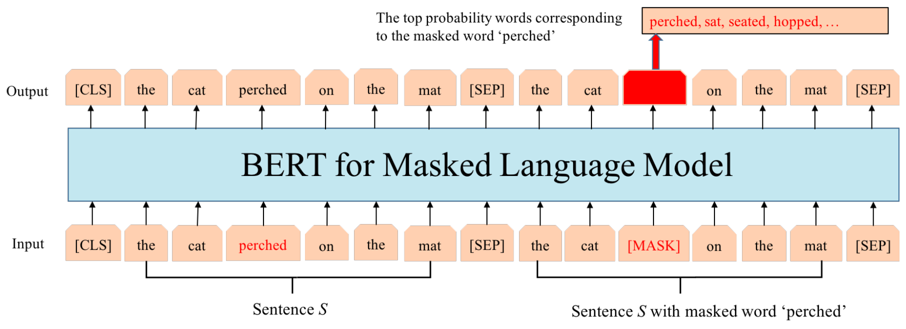
The top probability words corresponding to the masked word 'perched'
perched, sat, seated, hopped, …
Output
[CLS]
the
cat
perched
on
the
mat
[SEP]
the
cat
on
the
mat
[SEP]
BERT for Masked Language Model
Input
[CLS]
the
cat
perched
on
the
mat
[SEP]
the
cat
[MASK]
on
the
mat
[SEP]
Sentence S
Sentence S with masked word 'perched'

Figure 2: BERT for Masked Language Model

Masked Language Modelling is a method used by BERT to identify the context of a word without knowing the word by masking it. Then BERT has the task to identify the word while masked with only its surrounding word. The context is key successfully perform the task as no prior information is given to the model.

3.3 Transformer

The Transformer is a model. It is mainly used for NLP task. The model is composed of an Encoder and a Decoder.

3.3.1 Encoder

The first component is actually a stack of encoder. All of them are made of the same structure with a self-attention layer as input which feeds a feed-forward neural network. The input is one sentence at a time. However each words have its own path.The self-attention layer will look at the other words and its position in the sentence while it encodes the word. Then the feed forward neural network will apply the same operation to every input independently. This is a property of the transformer.

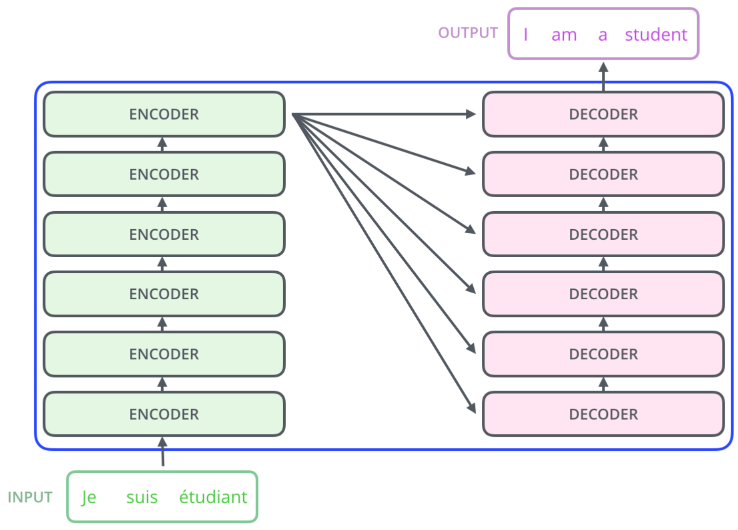

Figure 3: The transformer encoder decoder stack

The input of the model is text so it has an embedding layer. Its output is the input of the first encoder. Then each encoder has for input the output of the previous encoder.

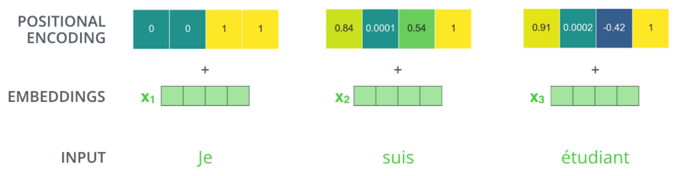

Figure 4: Transformer embedding and positional encoding

3.3.2 Self-Attention mechanism

The self-attention mechanism takes place in several steps.

The self-attention mechanism takes place in several steps. The first is to calculate several vectors for each input (words) a Query, Key and Value vectors. Each of those vector is the result of the multiplication of the input by their respective matrix. Those matrix have been set during training. Those vector are usually smaller than the embedding vector.

The second step is to calculate the score for each input in comparison to the other input. This is calculated for each position. The score is the dot product of the query vector and the key vector. Each input score itself and the other compared to himself. Therefor their is a N times N score. For example for an three input A, B and C, at the position of A the score is its query vector dot its key vector, compared to the query vector of A dot the key vector of B and the query vector of A dot the Key vector of C. This mechanism determines the influence of the other parts of the sentence at this position.

The third and fourth step divided the score in order to make the gradient more stable usually with the square root of the vector dimension. Then perform a softmax operation on the score value to normalize their sum between 0 and 1. The softmax helps reduce the score of the input irrelevant at this position.

The fifth step multiply the score and the values of each words. The softmax of the score acts as a weight here.

The sixth step calculates the sum of the values from the fifth step. It is the output of the attention layer for each position.

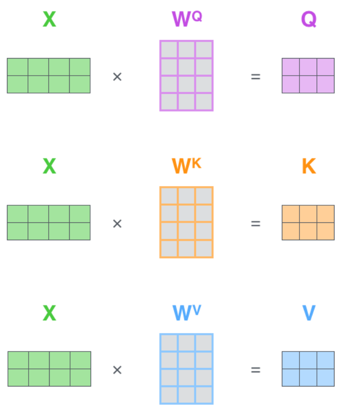

Figure 5: Calculation for self-attention mechanism

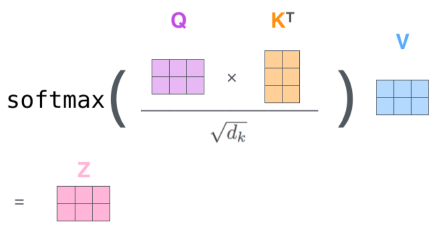

Figure 6: Calculation for self-attention mechanism with Matrix

When the input are concatenated in a matrix with each input as a row the six step can be performed at once with matrix calculation. This is shown in Figure 5, the green vector is the input, the purple Q is the Query, the orange K is the Key and lastly the blue V is the Value the W matrix are the respective weight.

In the Figure 6 we see the matrix calculation which result in the matrix Z with each row being the result of the six step calculation.

3.3.3 Multi head

Transformer model usually uses what is called "multiheaded attention". It is simply the same attention mechanism performed several time on the same sentences. In order to obtain different output the weighted matrix are initialized differently.

It improves the models by creating different subspace representation of the same input.

In order to rescale the output for the feed-forward neural network layer a new weigthed matrix is created. The output of the self-attention head are concatenated into one matrix. The new weigthed matrix is multiplicated to the result matrix. It is designed to obtain a result of the size of the expected output. The weighed matrix is trained with the model.

3.3.4 Position

Another key property of the Transformer is its ability to account for the position of the word in the sentence.

After the embedding layer and before the first encoder the model adds a vector. The position is determined with a pattern that the model can learn. This helps the model determine the position the words or their relative distance. The purpose is to add the information in the embedding vector in order to transmit the information in the self-attention layer. In the Figure 4 we see the positional vector with the embedding one. They will both constitute the input of the first encoder.

3.3.5 Residual

The last specificity of the encoder is its residual connection. Both sub layer (self-attention and feed-forward) outputs feeds a normalization layer. The input of this layer is the sum of the output of a sub-layer (self-attention or feed-forward) with the input the sub-layer.

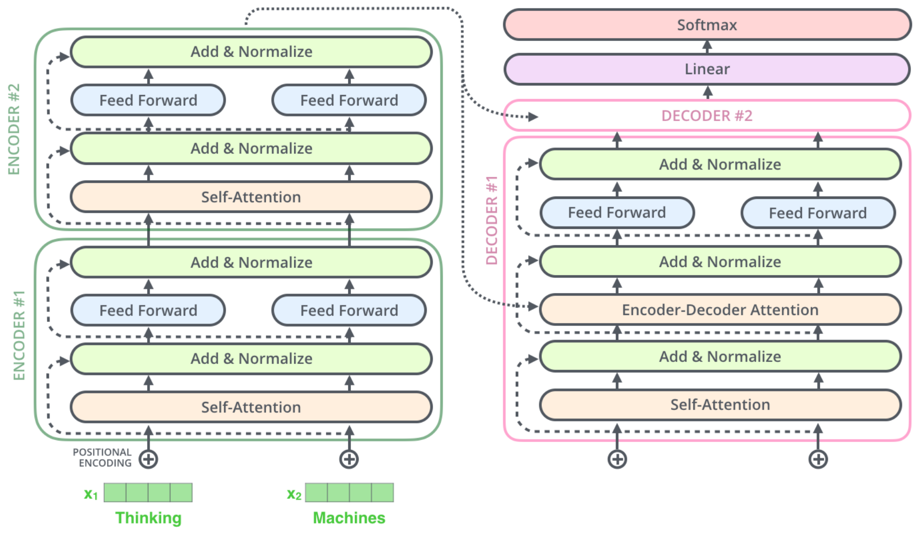

Figure 7: Detailed cells architecture with residual transfer

3.3.6 Decoder

The decoding component of a Transformer is much like the encoding one a stack of decoders. The architecture differs from the encoders with the addition of a sub-layer the encoder-decoder attention layer. It helps the decoder to turn its attention on the right part of the sentence. This is shown in the Figure 7 where another layer has been added compared to the encoder.

The output of the encoding component is two vectors used as attention vector. They are the result of the transformation of the output of the last encoder. Those two vector will then feed the encoder-decoder attention layer in each decoder of the decoder component.

The decoding process will not stop until its outputs reach a special symbol meaning the termination of the process. This implies that the decoding steps can be greater than the length of the input sequence.

Allowing the model to produce sequence of a different size of the input. This is important for translation task for example.

The output of each step is then feeding the first decoder of the stack. Like in the encoding step a positional vector and a embedding vector are used for each word.

Unlike in the encoder the decoder cannot see the next position in the sequence. This is possible by replacing its values by applying a mask.

The new encoder-decoder attention layer works like a self-attention layer except it uses the Query vector from the previous layer and the Key and Value vector are the output of the encoding component.

3.3.7 Linear-SoftMax layer

The Transformer model output words. It needs a layer to transform the output of the decoder component.

The linear layer is a fully connected neural network that transform the output of the decoders into a vector of the dimension of the vocabulary. Each cell of the vector correspond to the score of a word.

The softmax layer transform the output of the previous layer into a probability. Each cell between 0 and 1 and their sum equals to 1. The word with the highest probability is selected as the output for this pass.

3.4 N-gram Language Model

3.4.1 Uni-gram

The most basic ngram model is the uni-gram language model. It is a simple probabilistic model. The probability of a word is calculated based on the number of its occurrence in the corpus.

$$P(word) = \frac{\text{number of occurence of (word)}}{\text{total number of words}} \tag{3.4}$$

3.4.2 Higher N-gram

The probability of a word will depend on the $n - 1$ previous words.

This probability for a word is estimated with the number of time the n-gram appears in the corpus divided by the total number of occurrence of the sequence of the $n - 1$ words.

for a n-gram of three words w_1, w_2, w_3 the probaility of the word w_3 with the condition to be preceded by w_1, w_2 is

$$P(w_3|w_1w_2) = \frac{\text{number of occurence of } (w_1w_2w_3)}{\text{number of occurence of } (w_1w_2)} \tag{3.5}$$

For the words that are at the begining of a sentence we simply add a symbol to replace the blank space. This symbol can be the only words in an n-gram. For example the probability of a word to start a sentence is the number of times it has been found at the begining of a sentence devided by the total number of sentence. This is because the n-1 word before the start of the sentence are this symbol.

4 Related Work

4.1 Ken LM

The KenLM model [5] differs from the traditional n-gram model. It is a fast and reliable language model. One of its advantages is that it can be integrated into the model decoding. Because the probability of a word is calculated in regards of the previous word in the sequence it is possible to add the probability to the beam search decoder.

4.1.1 Modified Kneser–Ney smoothing

Kneser-Ney Probability equation with interpolation :

$$P_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{max(c_{KN}(w_{i-n+1}^{i} - d, 0))}{c_{KN}(w_{i-n+1}^{i-1})} + \lambda(w_{i-n+1}^{i-1})P_{KN}(w_i|w_{i-n+2}^{i-1}) \tag{4.1}$$

where

c_{KN} the count equation in high order and the continuous count in low order.

d is the discount factor

λ is the normalize discount function

$P_{KN}(w_i|w_{i-n+2}^{i-1})$ it the continuation probability

The modified Kneser-Ney smoothing algorithm introduce recursion. In Lower order the count becomes the continuous count and the discount factor becomes greater than 0. The last factor introduce the interpolation with the recursion. It drop down the order of the n-gram.

The main advantage of the Kneser-Ney smoothing is in the lower order. In a traditional n-gram model we only look at the probability of a word in the context of the given preceding words. The continuous function rather focus on the word in all the context it can be found. It is the fraction of the number of different previous string for the word divided by the total number of n-gram.

This is a very good improvement on the traditional n-gram model. When a word has not yet been seen in a context a word seen in many context is most likely going to be selected over a word seen in a few context.

Its main disadvantages is that it is not a multilingual model contrary to the BERT model. Having one multilingual model enable a better integration, no need to detect the language of the input.

4.2 Attention based ASR

Attention based model are a hot topic in research. We see that the research uses existing model with state-of-the-art result in other fields such as NLP and Computer Vision and tries to adapt them to their research.

4.2.1 Joint CTC-Attention Decoder

In the paper *Advances in Joint CTC-Attention based End-to-End Speech Recognition with a Deep CNN Encoder and RNN-LM* the author experiment with a new architecture using both the CTC algorithm and an attention based mechanism. Similarly to the Transformer architecture it uses a form of Encoder-Decoder.

The acoustic model is a Deep Neural Network based on the VGG network with Convolutional Neural Network (CNN). It is a model that is primary used in image processing. It feeds a Bilateral LSTM layer.Both network compose the encoder.

Sitting on top of the encoder is the joint decoder. This is where the CTC and the attention layer are combined. Both layer are fed from the encoder. In addition we find a RNN Language Model.

The objective of this architecture is to solve the issue of CTC and attention based layer for ASR. The CTC is very efficient at solving the problem of character alignment, but it makes a few conditional independence assumption. This creates an issue when trying to find the correct sequence of character and words. The attention based layer does not make prior assumption. This is very useful for sequence modeling. However it can cause alignment issues.

In order to solve the issue of the Attention Decoder the CTC objective function is used during training to enforce alignment.

The objective function becomes :

$$L = \lambda log p_{ctc}(C|X) + (1 - \lambda) log p_{att}(C|X) \tag{4.2}$$

where

p_{ctc} the CTC objective function

p_{att} the Attention layer objective function

λ a tunable parameter

This architecture uses the same CTC as our proposed solution but improved it with the attention mechanism. The methods used in the paper cannot be used with the BERT model because only the decoder layer is used. One possible experiment would be to initialize the weight of the Attention decoder with the BERT Decoder. Training would still be necessary since the decoder works with character and not words but it could have an impact.

Compared to our solution the integration of the CTC objective function with the Attention Decoder is an improvement. However it does not make use of out-of-the-box model to improve its language understanding unlike our project.

4.2.2 Transformer-based End-To-End ASR

The Transformer model attracts a lot of research for the NLP task. Now new research are done using the model for ASR system. In the previous paper the focus was made on the attention mechanism.

In this paper "Improving Transformer-Based End-to-End Speech Recognition with Connectionist Temporal Classification and Language Model Integration" [7] the focus is on the Transformer architecture.

The model uses the log-Melfilterbank speech features. Then it uses a two layer CNN to subsample the input. It feeds the Transformer encoder.

The encoder is based attention and feed-forward neural network layer. The output of the encoder component is an encoded vector.

For the decoder similarly to the architecture in the previous paper their is two branch. The Transformer Decoder and the CTC. It is a joint decoding approach. The output of the CTC and of the Transformer Decoder are joint with the log likelihood. In addition a Language model is introduced in the objective function.

The loss function is :

$$\hat{Y} = arg\, max_{Y \in y^*} (\lambda\, log\, p_{transformer}(Y|X_e) + (1-\lambda)\, log\, p_{ctc}(Y|X_e) + \gamma\, log\, p_{LM}(Y)) \quad (4.3)$$

where

p_{ctc} the CTC objective function

$p_{transformer}$ the Transformer decoder objective function

Y the decoded character sequence

X_e the encoder output sequence

γ a parameter named LM weight

λ a parameter named CTC weight

This new research has a similar aim to use the attention mechanism for decoding. In addition it uses the same mechanism for the encoding of the speech feature.

5 System modeling and structure – the proposed solution

In this section we will give a detail structure of the proposed solution. For this project we will us the DeepSpeech implementation by Mozilla. It is an english version of an ASR system. For the Language model we will use the AWS implementation of a scorer for BERT. In addition we created an Encoder Decoder based on the framework NeMo (Neural Modul) develloped by Nvidia. This model will perform a correction task on the output of the ASR in order to try to correct it.

5.1 Acoustic Model

For the Acoustic system we opted for an End-to-End system. We used the DeepSpeech implementation made by Mozilla. The first layer is a feature extraction that calculates the MFCC vector for a subsection of the signal for 10ms. Then it feeds a five RNN layers. The output is processed by a CTC based decoder using Beam Search algorithm.

We have decided to use such an implementation because it is an open source project with a widely used architecture.

The architecture only uses a language model as an optional module. The CTC algorithm has two property that are interesting when associated with a Transformer based Language Model such as BERT. The first is the conditional independence of the output. It is an issue for sequence to sequence modeling. The Transformer model can solve this problem with its attention mechanism. The second is the alignment property. The algorithm can enforce a monotonic alignment which most Sequence to Sequence model are not able to.

The choice of this acoustic model is interesting when associated with a Transformer model such as BERT.

Another reason we chose this model is that the structure does not embark a Language model. Our LM will be the only one and we can have a better understanding of the efficiency.

5.2 Language Model

For the LM we decided to use the output of the Beam Search from the ASR system. We used the 100 best Beam Search result.

5.2.1 Re-Scoring Model

The BERT model cannot be used out-of-the-box as a Language Model. We decided not to try to integrate it in the ASR system but to rather use it as a re-scoring model.

The language model outputs a score for the given n outputs. The score is calculated based upon the score of the LM and of the Beam Search.

For each sentence a score is attributed. Then on the development dataset the result are interpolated with the score of the ASR.

The objective function is :

$$\hat{W} = arg\,max_W(\lambda\, log\, p_{ctc}(W|X) + (1-\lambda)\, log\, p_{lm}(Y|X) \tag{5.1}$$

where

p_{ctc} the score of the CTC objective function

p_{lm} the score of the language model

W the target text token

X the source text token

λ a tunable parameter with value between 0 and 1

In order to obtain the best parameter λ for the interpolation we perform grid search with the linspace function function in python with 20 values.

5.2.2 BERT as language model for re scoring

The BERT model uses a Masked Language Model. Traditional Language Model predict a word based on the past word in the sentence. Because of the bidirectional representation it is not a Language Model *perse*. However in the paper *BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model* the researcher demonstrate that it can be seen as a *maximum pseudolikelihood estimation*.

In contract with an LM a word is predicted by replacing it with a [MASK] and using all word in the sentence.

For a given sentence : "I am a student in Hong Kong"

Seven input will be created, one for each masked word. Then the likelihod of each word will be calculated. The score of the sentence will be the sum over all log likelihood.

It is not a LM as the score is not a probability but it can be used for re-scoring purpose.

The main reason that influenced our choice of model is its state-of-the-art result on many NLP task. It is able to get a good understanding of the language and a great contextualisation.

Because the BERT model is trained on a very large corpus (2.5 TB of text) it gives a general understanding of the language regardless of the training set. Some language model are trained for a specific task which achieve great result in a limited context.

In addition BERT has also multilingual implementation that achieve great result on more than a hundred language. It will allow the development of the ASR system without having to change the LM.

5.3 ASR post-processor

We developed a Transformer based Encoder Decoder See Figure 3. This model is based on the paper *Correction of Automatic Speech Recognition with Transformer Sequence-to-sequence Model* [9]. The purpose of the model is to improve the output of the ASR system by using a Sequence2Sequence model to correct the possible mistakes.

The CTC algorithm considers words in a sentence from left to right. The probability of a word is heavily influenced by the previous words. The words at the beginning of the sentences were influence by fewer words and the first was not. The probability of errors at the beginning is higher. There is an accumulative effect of the error. A mistake early in the sentence will influence the rest of the decoding.

While a bidirectional Language model used for re-scoring can help decrease the influence of such error. If the ASR system did not succeed at finding the correct token or if its score is too low the re-scoring algorithm will not be able to compensate.

The objective is to improve the quality of the result by correcting the sentence. The aim is to use the attention mechanism of the Transformer to generate a sentence where each word is selected based on all the other word.

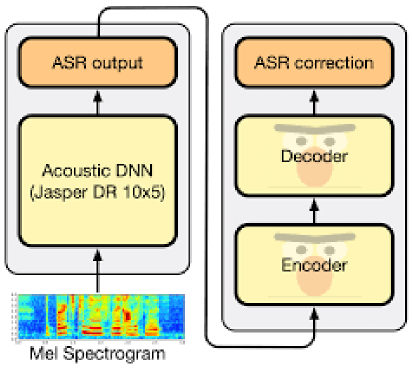

Figure 8: The ASR post-processor encoder decoder structure

The Figure 8 shows an exemple of the architecture of the model. In our project the Acoustic model is DeepSpeech and the output of the model can also be reranked first by the BERT MLM.

5.3.1 Encoder Decoder machine translation structure

The base of the model is a Neural Machine Translation model based on Transformer. The goal is similar to translation except that the language stays the same. An input sequence is encoded then decoded to produce a new sentence. It is not a BERT model but it is initialized with the BERT model weight.

The whole model is composed of 6 modules. The first is the BERT tokenizer which transform the text input in numerical data. The second is the BERT encoder. The third is a Transformer Decoder initialized with BERT weight. The fourth is a token classifier which transform the output into a vector of the vocabulary size. The fifth is the Beam Search modules which helps produce the best possible sequence of text, it is only used during evaluation and inference.

6 Experimental Set-up

6.1 Dataset

6.1.1 LibriSpeech

Our experiment was made on the Librispeech dataset [10].

The audio is sampled at 16kHz the same rate used in our acoustic model. The dataset is audio-book from 2484 speaker, 1283 males and 1201 females.

After calculating the WER with an ASR model the sample with a lower score were assigned a "clean" tag and the ones with a high WER were assigned the tag "other".

The training set is composed of three subset train-clean-100, train-clean-360 and train-other-500. Together they form a 960 hours of speech with their transcription. The text corpus is formed of around 281K sentences.

The development and test set are split in two with one "clean" and one "other" set. Each set is around 5 hours of speech.

6.1.2 VoxForge

The VoxForge [11] project is an open source initiative. It is a speech dataset available in different language. Due to hardware limitation we were not able to get an evaluation of the 500hours of speech from the LibriSpeech dataset. It is an inconvenient because the set is composed of sample labelled as "other". In other words using only the "clean" set would create a disparity.

In order to work around the problem we decided to use data from another set. Since it is not from the librispeech it is most likely that the model will yield result closer or similar to the one from the train-other-500.

We downloaded 100 hours of speech from the VoxForg project.

6.2 Hardware

For the training, testing and evaluation we used a VM in the cloud hosted on google colab. A GPU was provided, usualy the model was an nvidia P100 with 16GB of memory. It was enough to be able to evaluate the dataset containing 360 hours of speech but not enough for the one with 500hours of speech. The VM were available with a 24hours limit. The 360hours set took around 20hours for evaluation.

6.3 Implementation

6.3.1 DeepSpeech

The purpose of the project is to improve the output of the ASR system with the help of NLP models. The DeepSpeech model is only used to output the text sequence. We use two implementation of the model. The first is without the scorer. The output is a list of the 100 best scored sequence from the Beam Search.

6.3.2 BERT MLM scorer

The AWS research team has develop a library in python to create Masked Language Model with BERT [12]. The project is based on the PyTorch library. The pre-trained model weight are downloaded from the huggingface [13] repository. This will enable us to use the same BERT model over different library. We have decided not to fine-tune the model in order to obtain result on the out-of-the-box model. The interest here is to look at the model performance as is without fine-tuning it on a specific dataset.

6.3.3 Transformer based correction model

Nvidia has developed the NeMo toolkit [14] used to create model for Conversational AI. The aim is to create Neural Module that could be easily combined together to create complex model. They developed several examples for their project. We used one project to create an encoder decoder model initialized with BERT based uncased weight value. The weight were obtained from the huggingface [13] repository similarly to the MLM scorer.

6.4 Methodology

We did not implement a pipeline for our proposed solution. The DeepSpeech model does not have a method for batch inference. The input of the MLM scorer differs from the output of the DeepSpeech model. The implementation of the Encoder-Decoder for correction does not have the methods necessary for inference yet. In order to obtain result for each step we had to perform the task independently and modify the input if necessary.

6.4.1 Benchmark on the ASR system

We evaluated the result for the ASR system with the use of the default Language model it is the KenLM n-gram model. The language model is created based on the normalized corpus of librispeech with the top 500,000 words. Unlike in our proposed solution the LM is integrated with into the system. It does not perform re-scoring but join scoring with the ASR decoder.

6.4.2 Get 100 Nbest result

For the re-scoring task we need a number of possible sequence that will be scored. The CTC algorithm uses Beam Search to find the n-best scored distribution. In order to obtain those result and their respective score we run inference on the DeepSpeech model with 100 best scored branch of the Beam Search Tree. We perform the task on the Dev and Test set from the LibriSpeech dataset.

The output is reformatted into a json file containing a python dictionary with the name of the audio file as key, the distribution and their score as value with the source text transcript.

6.4.3 Get score from the MLM

The library used for the MLM task contains two methods, one for scoring and another for re-scoring using the equation 5.1.

We perform the scoring task on the dev dataset.

Three different MLM models were used the bert-base-uncased, bert-large-uncased and the bert-base-multilingual-uncased. The output of the ASR is an uncased sequence.

6.4.4 Perform Re-scoring

We used the dev dataset to find the best hyperparameter for re-scoring using grid search. For each model 20 parameter are tested.

With the result of the rescoring we then perform the Word-Error-Rate metrics on it. Then we compare all the result to see if using an out-of-the-box model can improve the performance of an ASR system on the test dataset.

6.4.5 Get evaluation of training test

For the encoder decoder task we need to train our model on the output of the ASR system. We used the evaluation task to obtain the output text sequence in order to obtain the WER score in addition.

We obtained the evaluation result from 3 training set LibriSpeech train-100-clean train-360-clean and the train dataset from VoxForge. We obtained around 200K sentences.

6.4.6 Reformat data for NeMo

We created two dataset the first contains all of the text output regardless of their WER score, for the second we only selected the one with a score inferior to 50%. The reason behind the selection is that the sentences with too many mistakes will make the model perform too many changes on the sentences. The objective of the model is to correct the mistakes of the output that are very similar to the intended output. The input format for the model is two txt file containing list of string. It is a very elementary format. Their is no information about the origin of the string such as the id of the audio file. It did caused issues when trying to perform inference.

6.4.7 Training the Encoder Decoder

The team which developed this project at nvidia used a server with 8 GPU. We were only able to a single GPU. It was sufficient to train the model on our dataset but we had to make some changes on the hyperparameters. We reduced the batch by two from 4096 to 2048. The decision was a conservative one, we were afraid the RAM limit of the GPU would be attained and the training stop.

With a smaller batch size we had to increase the number of step to compensate. From the original 2000 step limit we increased it 10,000. We stored the checkpoint of the model every 1000 step in order to perform test on the model.

The most time consuming task was the evaluation around 1000 time longer than training steps. We decided to change the frequency of the evaluation every 200 steps to every 1000 steps.

6.4.8 Perform inference wih NeMo

The library is a recent project. The project was first committed in February of this year. The script only contains the training task. In order to use the model for performance analysis we had to find a way to perform inference.

First we tried to reproduce a solution similar to another project. We created each neural modul and initialized them with the trained checkpoint. We lacked the necessary knowledge to make this solution work. The format of some input were inconsistent with the one expected by some module.

Then we looked a the existing methods for inference in the documentation. The NeMo toolkit aggregate several Neural Module into one model. A global instance called Neural Module Factory is used for training. It has a method for inference as well. We created a function that would restore the model from the checkpoint then initialize the necessary callback function for inference. Then we performed the task on the dev-clean dataset. The task worked fine but the output was not usable. Due to the elementary format of the input data, a simple text file we could not trace back the output to the input data. The data was shuffled during inference. We could not do our performance analysis.

The last solution is similar to the previous one. We create the training instance for the model. Each module is restored from checkpoint. We do not perform a training task. We use the model to only perform a single evaluation task on the validation dataset. In order to be able to use this solution we had to instantiate a fake training set which will only be use for the initialization of the model. Not task is performed on it.

This work around solution enable us to get the WER score for a dataset on the trained model.

7 Preliminary performance analysis of system

The proposed solution is composed of three module ASR, BERT MLM and Encoder-Decoder for output correction. Each one is a standalone program. Their performance is analyzed independently. The result display in this section are the percentage of error calculated with WER metrics.

7.1 Performance of the rescoring LM

Performance of the LM scoring		
LM	Test-clean	Test-other
KenLM	5,99	19,19
Beam Search	12,16	29,27
base	**8,8**	26,76
large	8,96	27
multiligual	9,04	**26,7**

The base result from the ASR system with the n-gram language model yields the best results. This solution has a better integration in the system since the scorer is involve in the scoring mechanism with the Beam Search algorithm.

The result from the decoder without the LM scorer are the lowest. This is due to the absence of Language Model which incur a lack of language understanding.

The use of a rescoring model on top of the decoder result improves the performance. For the clean test set the percentage of WER diminishes by at most 3,3% with the BERT base model. For the test set with sample labeled as "other" the percentage of error diminishes by at most 2,57% with the BERT multilingual model. The use of a larger model such as BERT-large with 340M parameter does not yields better result compared to the base model with 110M parameter.

Surprisingly the use of a multilingual model performance are in pair with a monolingual model. It even obtain the best result for the most difficult test set. This could be due to the possible occurrence of other language in the test set.

7.2 Performance of the Encoder Decoder

Table 1: Performance of the Encoder-Decoder

	trained on all sample		sample with WER < 50 %	
LM	Test-clean	Test-other	Test-clean	Test-other
KenLM	6,5	20	6,5	20
base	9,53	**26,37**	**9,41**	27,6
large	**9,32**	27,6	9,52	27,89
multiligual	9,37	27,27	9,67	**27,5**

The Encoder-Decoder model to correct the output of an ASR does not improve the performance. We see a drop between 0.5% and 1% in the WER metrics for all input.

The only improvement is for the sequence from the test-other dateset which were re-scored with the BERT base model and corrected by the model trained on all sample. However that same model see the worse drop in performance on the test-clear dataset. This is certainly due to the model overfitting sample "other" at the expanse of the "clear" sample.

The performance disparity in WER score between the input dataset and the output dataset does not change with the performance on the input data. The score with the KenLM is drop by 0,5% which indicates that the model does not perform better if the dataset is already well estimated.

On average the WER of the total training set was around 8%, for the filtered set the average was around 6%.

The model trained on all sample yields better result than the one with the selected sample. One of the reason could be that the dataset was too small and that the lack of error to be corrected on the selected training set perform worse.

This kind of model requires a large dataset in order to not overfit. In our training phase we used around 200K sentences. The performance shows that the model would need a better training set.

One way to augment the dataset would be to select the 10 most likely distribution from the Beam Search on the Librispeech training set. Using train-clear-100, train-clear-360 and train-other-500 will improve the quality of the set. Then filter the set with the sample with a WER inferior to a certain threshold such as 50%.

Using the Beam Search result have a disadvantages to show result that were not the output of the ASR. It also has the advantages to incorporate sample with errors from the ASR. The training set will contains different version of a sample with different mistakes. We emit the hypothesis that this would improve the performance of the model by seeing potential mistake from the ASR.

8 Conclusion and Future Work

The most common approach to decoding in an E2E ASR system is to use an n-gram language model in addition to the CTC algorithm. It will generate a the n-best distribution. It could viewed as a reduced sized search space. The beam search select the most likely path in a tree of distribution in order to reduce the size of the tree.

One of the way to improve the performance is to rescore the distribution with a Language Model. This is called a two-pass strategy. The main disadvantages is that the second LM is not integrated in the first decoder. This will result in the loss of impact from the model on the result. Potentially good distribution might be disregarded in the first-pass.

We have seen that the performance of the integrated n-gram model surpass any rescoring model used in the experiment.

However the use of Neural Language Model can have other benefits. The BERT multilingual model yields excellent performance in Natural Language Understanding task in over a hundred language. We have shown in the experiment that the use of this model improves the performance over a simple Beam Search on english sample. The integration of such model could improve the performance on multilingual ASR system.

Another benefits from the use of rescoring strategy is the use of state-of-the-art Bidirectional model. Their architecture allow the scoring model to improve its understanding of the sentence by contextualizing every word in regards to the rest of word in the sentence. The downfall is their architecture cannot be integrated in the first pass.

A less common approach to improve the decoding of the ASR system is to use an encoder-decoder to correct its mistakes. This kind of models requires a large dataset in order to obtain tangible result. In our experiment we did not do data augmentation. This resulted in a the creation of an overfitting model which actually degrades the performance. This solution would require more work in order to yield interesting result. None the less It is interesting to notice that the performance did not notice a decrease of the performance that important. This suggest that the model does not modify the sentence too much.

8.1 Reflection on the integration of the proposed solution

The architecture of the proposed solution can be improved. From the result of the experiment and our broader understanding of the problem we will suggest a new architecture.

The architecture presented in this paper seems more appropriated [7]. The encoder is based on the transformer architecture. Then the encoded tokens will feed two decoder. The first will be the CTC algorithm and the second a transformer based decoder. They perform the task jointly.

A Language Model can be added to the decoding task. A transformer based one such as Transformer-XL makes sens but several reports states that it incurs latency making the computational performance decrease.

In order to avoid latency a two pass decoder can be used with the integration of a model such as BERT multilingual. This would make the creation of a multilingual model possible.

8.2 Future work

The use of well-known model for Image Processing such as AlexNet have been evoked in some papers. The transformer architecture is the one that attracts the most attention for the speech encoding task and the character decoding task. An interesting research would be to look for the combination of both architecture.

The BERT architecture makes is not usable for an integration in the first-pass decoding structure. One of the possible way of transfering the learning from the BERT model to a decoder would be to use a transformer based LM decoder initialized with the BERT pretrained weight.

References

[Amo+15] Dario Amodei, Rishita Anubhai, Eric Battenberg, et al. *Deep Speech 2: End-to-End Speech Recognition in English and Mandarin.* 2015. arXiv: `1512.02595 [cs.CL]`.

[Dev+18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* 2018. arXiv: `1810.04805 [cs.CL]`.

[SLJ19] Joongbo Shin, Yoonhyung Lee, and Kyomin Jung. *Effective Sentence Scoring Method using Bidirectional Language Model for Speech Recognition.* 2019. arXiv: `1905.06655 [cs.CL]`.

[Sal+20] Julian Salazar, Davis Liang, Toan Q. Nguyen, et al. "Masked Language Model Scoring". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* Online: Association for Computational Linguistics, July 2020, pp. 2699–2712. URL: `https://www.aclweb.org/anthology/2020.acl-main.240`.

[Hea+13] Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, et al. "Scalable Modified Kneser-Ney Language Model Estimation". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers).* Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 690–696. URL: `https://www.aclweb.org/anthology/P13-2121`.

[Hor+17] Takaaki Hori, Shinji Watanabe, Yu Zhang, et al. *Advances in Joint CTC-Attention based End-to-End Speech Recognition with a Deep CNN Encoder and RNN-LM.* 2017. arXiv: `1706.02737 [cs.CL]`.

[Kar+19] Shigeki Karita, Nelson Enrique Yalta Soplin, Shinji Watanabe, et al. "Improving Transformer-Based End-to-End Speech Recognition with Connectionist Temporal Classification and Language Model Integration". In: *INTERSPEECH.* 2019.

[WC19] Alex Wang and Kyunghyun Cho. *BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model.* 2019. arXiv: `1902.04094 [cs.CL]`.

[HPG19] Oleksii Hrinchuk, Mariya Popova, and Boris Ginsburg. *Correction of Automatic Speech Recognition with Transformer Sequence-to-sequence Model.* 2019. arXiv: `1910.10697 [cs.CL]`.

[Pan+15] V. Panayotov, G. Chen, D. Povey, et al. “Librispeech: An ASR corpus based on public domain audio books”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pp. 5206–5210.

[Vox] Voxforge.org. *Free Speech... Recognition (Linux, Windows and Mac) - voxforge.org*. `http://www.voxforge.org/`. accessed 06/25/2014.

[aws20] awslabs. *mlm-scoring*. `https://github.com/awslabs/mlm-scoring`. 2020.

[Wol+19] Thomas Wolf, Lysandre Debut, Victor Sanh, et al. “HuggingFace’s Transformers: State-of-the-art Natural Language Processing”. In: *ArXiv* abs/1910.03771 (2019).

[Kuc+19] Oleksii Kuchaiev, Jason Li, Huyen Nguyen, et al. *NeMo: a toolkit for building AI applications using Neural Modules*. 2019. arXiv: `1909.09577 [cs.LG]`.