IP9

Daniel Tiefenauer

September 25, 2018

Abstract

tbd.

Contents

1 Introduction

This report documents the progress of the project *Forced Alignment with a Recurrent Neural Network*. The project serves as a master thesis at University of Applied Sciences (FHNW) (IP9). Some preliminary work has been done in a previous project (IP8). The overall goal, project situation and background information are described in detail in the project report for IP8 and are not repeated here. Instead, a quick recap is given for the sake of completeness of this documentation.

1.1 Scope and overall goal

ReadyLingua is a Switzerland based company that develops tools and produces content for language learning. Some of this content consists of audio/video data with an accompanying transcript. The overall goal is to enrich this data with temporal information, so that for each part of the transcript the corresponding point in the audio/video data can be found. This process is called *acfa*. An *InnoSuisse* project was started in 2018 to research how this could be achieved. The *InnoSuisse* project foresees three different approaches, one of which is followed in this project.

1.2 Chosen approach and previous work

The approach chosen in this project is based on speech pauses, which can be detected using *Voice Activity Detection (VAD)*. The utterances in between are transcribed using *Automatic Speech Recognition (ASR)*, for which a *Recurrent Neural Network (RNN)* is used. The resulting partial transcripts contain the desired temporal information and can be matched up with the full transcript with a process called *Local Sequence Alignment (LSA)*.

In the IP8 project, VAD, ASR and LSA were treated as part of a pipeline which split a given audio file into individual utterances, transcribe them and localize them in the original transcript. Since the quality of the ASR stage has an imminent impact on the downstream LSA stage, the quality of the alignments is heavily dependent on the quality of the partial transcripts. However, ASR is highly prone to external influences like background noise, properties of the speaker (gender, speaking rate, pitch, loudness). Apart from that, language is inherently abiguous (e.g. accents), inconsistent (e.g. linguistic subtleties like homonyms or homophones) and messy (stuttering, unwanted repetitions, mispronunciation).

1.2.1 Previous results

For the VAD stage, an implementation of WebRTC was used which was shown to be capable of detecting utterances with very high accuracy within reasonable time. For the LSA stage a combination of the Smith-Waterman algorithm and the Levenshtein distance was used. This combination included tunable parameters and proved to be able to be able to localize partial transcript within the full transcript pretty well, provided the similarity between actual and predicted text was high enough.

Because the final pipeline should be language-agnostic, the IP8 project proposed the use of *DeepSpeech* for the ASR stage, which uses Connectionist Temporal Classification (CTC) [1] as its cost function. It included some experiments on what features could be used to train a RNN for the ASR stage. Possible features were raw power-spectrograms (as stipulated by the *DeepSpeech* paper), Mel-Spectrograms and Mel-Frequency Cepstral Coefficients (MFCC). It was found that training on MFCC features would probably require the least amount of training data because. An RNN using a simplified version of the *DeepSpeech* architecture was trained on data from the *LibriSpeech* project (containing only English samples). However, developing a fully-fletched ASR system is extremely time-consuming and could not be done within the project time. For that reason a state-of-the-art *Speech-To-Text (STT)* engine (*Google Cloud Speech*) was embedded in the pipeline as the ASR stage. Using this engine, the pipeline was able to produce very good (although not perfect)

transcripts for the individual utterances. Therefore the chosen approach was validated and the pipeline could shown to be generally functional.

1.3 Goal of this project

In this project, the chosen pipelined approach shall further be refined. Because the VAD and the LSA stage already work pretty well, the focus in this project is on the ASR stage. By researching if and how an own STT engine can be obtained the pipeline shall become self-contained by using an ASR stage that is not dependent on a proprietary solution and can be individually tuned. Moreover the quality of the produced alignment shall be assessed. Since this is the overall goal, it shall become clear how this quality changes with varying properties of the ASR system.

Concretely, the following aspects shall be examined more closely:

- **How does the quality of the simplified *DeepSpeech*-RNN change with increasing training data?** By plotting the learning curve we should be able to see whether the RNN is able to learn something useful at all and also get some intuition about how much training data is needed to get reasonably accurate partial transcripts.
- **How does the quality of the partial transcripts change when using synthesized training data?** Neural Network usually require large amounts of training data and often improve with increasing size of the training set. However, labelled training data is usually scarce and expensive to acquire. For the purpose of Forced Alignment however, synthesized training data can be easily obtained by adding some distortion to the original signal (reverb, change of pitch, change of tempo).
- **How does the quality of the partial transcript change when integrating a Language Model (LM)?** STT-engines traditionally use a LM that models the probabilities of characters, words or sentences. A LM can help producing valid transcripts by mapping transcripts (that may sound similar to what was actually said) to orthographically correct sentences.
- **How can we assess the quality of the alignments?** This should give us some insight about how the quality of the alignment changes with varying capability of the STT-engine and what quality of transcripts is required.

The answers to above questions should help in estimating the effort to create a generic solution (amount of training data, architecture, etc.). [1]

[1] Because ASR is highly dependent on the language that should be recognized, a different STT system has to be trained for each language.

2 Exploring the DeepSpeech architecture

The implementation of the RNN used for STT in the IP8 project was done in Python using Keras[2]. The following simplifications were made

- No LM
- No convolution in first layer

Although the RNN used a simpler architecture and no computational power is needed by querying a LM, performance was still a big issue. Training on aligned speech segments from the *LibriSpeech* corpus was not possible within project time because it would have taken approximately two months when using a single acGPU. However, this is constent with the experience made by the Machine Learning team at Mozilla Research, which used a cluster of 16 GPUs that required about a week [3] to train a variant [3] of the RNN originally proposed in [1].

An open source implementation of a DeepSpeech model is available from Mozilla [4]. Since the implementation uses a LM, the quality of the model is measured as the percentage of misspelled or wrong words (called Word Error Rate (WER)) or as the edit distance (also called Levenshtein similarity or Label Error Rate (LER)). A pre-trained model for inference of English transcript can be downloaded, which achieves a WER of just 6.5%, which is close to human level performance [3].

An own model could be trained by providing training-, validation- and test-data in the format expected by the implementation. However, this is not the preferred procedure for this project for various reasons:

1. The model's intended application is ASR. In such settings a low WER is desirable. However, for this project ASR is just one stage in a pipeline that has to be *good enough* for the LSA stage to make the alignments. As a result, although sensible for pure ASR tasks, the architecture of the Mozilla implementation might be overly complicated for this project. The problem with this is that more complex models usually require more training data, which might not be available for the target languages used by *ReadyLingua*.
2. The implementation requires an (optional) LM which might not be available for the target languages.

The second point becomes less crucial, since the LM is optional for training and inference. However, without a LM the quality of transcription drops perceivably, as the following example shows:

transcript	value	LER
actual transcript	`and i put the vice president in charge of mission control`	1.00
inference without LM	`ii put he bice president in charge of mission control`	0.89
inference with LM	`i put the vice president in charge of mission control`	0.92

Table 1: examples of infered transcripts with pre-trained DeepSpeech model with and without LM (sample `20161203potusweeklyaddress` from the ReadyLingua corpus

For these reasons the limits of the simplified implementation are explored first. The training of an own model using the Mozilla implementation can be used as a second resort if the simplified model is found to be not accurate enough.

[2] `https://keras.io`
[3] the variant used MFCC as features whereas the original paper proposed raw spectrograms
[4] `https://github.com/mozilla/DeepSpeech`

3 Creating a Language Model

The learning curve above was plotted using the results from training on English samples. To compare get an intuition about whether the conclusions made are transferable to training on samples in other languages, the training was done again on German samples. In order to do this, a language model for German had to be created first. A 4-gram LM was trained on a corpus of 2.844.448 articles and pages from Wikipedia (104.327.682 sentences, xxx words, yyy unique words) using KenLM [2]. The corpus was normalized by removing Wiki markup, any punctuation and making everything lowercase. Accentuated characters like `è,é,ê, ...` and special characters like the German *ß* were translated to their most similar ASCII-equivalent (`e` resp. `ss`) to account for possible spelling errors and reduce the number of words. Umlauts (although not part of the ASCII codeset) were kept as-is however. n-grams can be represented with a tree structure [5], which allows for pruning. The n-grams used to train the LM have been pruned by setting the minimal threshold for any n-Gram ($n \in 1..4$) to 100. Pruning the unigrams was done to get rid of obvious spelling mistakes. Pruning higher-order n-grams was done to increase performance (both in space and time).

[5] note that *KenLM* offers a s.c. *PROBING* data structure, which is fundamentally a hash table combined with interpolation search, a more sophisticated variant of binary search, which allows for constant space complexity and linear time complexity. This does however not change the fact that n-grams can conceptually be thought as a tree of grams

List of Figures

List of Tables

References

[GFG06] Alex Graves, Santiago Fernández, and Faustino Gomez. “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks”. In: *In Proceedings of the International Conference on Machine Learning, ICML 2006*. 2006, pp. 369–376.

[Hea11] Kenneth Heafield. “KenLM: Faster and Smaller Language Model Queries”. In: *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland, United Kingdom, July 2011, pp. 187–197. URL: https://kheafield.com/papers/avenue/kenlm.pdf.

[Mor17] Reuben Morais. *A Journey to $<$10% Word Error Rate*. 2017. URL: https://hacks.mozilla.org/2017/11/a-journey-to-10-word-error-rate (visited on 09/14/2018).

Acronyms used in this document

ASR Automatic Speech Recognition

CTC Connectionist Temporal Classification

FA Forced Alignment

FHNW University of Applied Sciences

GPU Graphics Processing Unit

LER Label Error Rate

LM Language Model

LSA Local Sequence Alignment

MFCC Mel-Frequency Cepstral Coefficients

RNN Recurrent Neural Network

STT Speech-To-Text

VAD Voice Activity Detection

WER Word Error Rate

4 Ehrlichkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende schriftliche Arbeit selbstständig und nur unter Zuhilfenahme der in den Verzeichnissen oder in den Anmerkungen genannten Quellen angefertigt habe. Ich versichere zudem, diese Arbeit nicht bereits anderweitig als Leistungsnachweis verwendet zu haben. Eine Überprüfung der Arbeit auf Plagiate unter Einsatz entsprechender Software darf vorgenommen werden.
Würenlingen, September 25, 2018

Daniel Tiefenauer