BUILDING CORPORA OF TRANSCRIBED SPEECH FROM OPEN ACCESS SOURCES

O.O. Iakushkin [a], G.A. Fedoseev, A.S. Shaleva, O.S. Sedova

Saint Petersburg State University, 7/9 Universitetskaya nab., St. Petersburg, 199034, Russia

E-mail: [a] o.yakushkin@spbu.ru

Currently there are hardly any open access corpora of transcribed speech in Russian that could be effectively used to train those speech recognition systems that are based on deep neural networks—e.g., DeepSpeech. This paper examines the methods to automatically build massive corpora of transcribed speech from open access sources in the internet, such as radio transcripts and subtitles to video clips. Our study is focused on a method to build a speech corpus using the materials extracted from the YouTube video hosting. YouTube provides two types of subtitles: those uploaded by users and those generated by speech recognition algorithms. Both have their specifics: user-provided subtitles may have timing inaccuracies, while auto-generated subtitles may have recognition errors. We used the YouTube Search API to obtain the links to various Russian-language video clips with subtitles available—words from a Russian dictionary served as an input. We examined two strategies to extract audio recordings with their corresponding transcripts: by using both types of subtitles or only those that were produced through automatic recognition. The voice activity detector algorithm was employed to automatically separate the segments. Our study resulted in creating transcribed speech corpora in Russian containing 1000 hours of audio recordings. We also assessed the quality of obtained data by using a part of it to train a Russian-language automatic speech recognition system based on the DeepSpeech architecture. Upon training, the system was tested on a data set consisting of audio recordings of Russian literature available on voxforge.com — the best WER demonstrated by the system was 18%.

Keywords: Deep Neural Network, Speech, Data Aggregation

© 2018 Oleg O. Iakushkin, George A. Fedoseev, Anna S. Shaleva, Olga S. Sedova

CEUR-WS.org/Vol-2267/475-479-paper-91.pdf

1. Introduction

Deep neural networks that perform speech recognition require an extensive amount of data for training—that is, hundreds of thousands of hours. The collection of such amount of data is the most resource-intensive task when it comes to building a speech recognition system for the Russian language, since there are no large datasets available for this language in open access. This paper is focused on the methods used to automatically collect and filter data obtained from open sources. The paper provides a description of several such methods and weighs the advantages and drawbacks of each method.

2. Dataset format

Speech recognition systems that are based on deep neural networks (e.g., DeepSpeech) should be trained on aligned audio-transcript pairs. The data include a set of three text files in the 'csv' format (train.csv, dev.csv, test.csv) and the corresponding directories with 'wav' audio files containing speech segments. The 'wav' files have one audio channel (mono), the sample rate of 16,000 Hz, and the depth of 16 bit for each value. The ratio between the audio length and the number of symbols in the transcript is subject to a restriction that follows from the CTC matrix's decoding algorithm [3]—namely, the number of steps in the CTC matrix must exceed the number of symbols in the transcript.

Each of the 'csv' files contains a table with data arranged in three columns: 'wav_filename', 'wav_filesize', and 'transcript'. The 'wav_filename' column contains the full paths to 'wav' files; the 'transcript' column has the text transcribed from the respective audio file; and the 'wav_filesize' column stores the sizes of audio files in bytes. The file size information is used to sort samples for neural network training.

We started out by collecting data available at the voxforge.org website that has a set of transcribed speech created by user community. The data has the duration of nearly 25 thousand hours and comprises parts of Russian literary works read aloud by users.

3. Getting speech data from YouTube service

3.1. Matching auto-generated and user-provided subtitles

We used YouTube API to search for YouTube video files, which was the first step of collecting audio files and their transcripts. We restricted our search only to videos with user-provided subtitles. A file with text queries was created to perform a looped search across multiple queries. The queries corresponding to a certain topic were included in the file manually. We used the Yandex WordStat service and the Yandex Wordstat Assistant extension to obtain various queries that fell under a specific topic. After the videos were retrieved using the queries, their identifiers were saved in a list for further processing.

Then we attempted to download YouTube's auto-generated captions (subtitles) in the 'vtt' format for each of the videos in the list. The Youtube-dl utility was used to extract subtitles and audio from YouTube. In automatic captions, there are timestamps marking the beginning of each of the recognized words. We employed a python script to use this data to match the words with their corresponding audio segments.

Thereafter, the user-provided captions were also downloaded—these were always available since the videos had been searched for using the respective filter. Then we searched the auto-generated subtitles (word by word) for segments that precisely correspond to each of the timed segments in the user-provided subtitles. In some instances, no corresponding segments were present because certain words were not recognized or a user's subtitles did not match the speech. However, where the corresponding segments did exist, this step allowed to obtain more accurate timestamps (start and end times of a speech segment) and to make sure that a particular text is in fact present in the audio

(otherwise the automatic recognition system would not have picked it up). It is also important to obtain accurate timestamps so that the speech was not split across the middle of a word and so that no speech that is not in the transcript was included. An audio file was downloaded and segmented where the script identified at least one match between the auto-generated and the user-provided subtitles.

However, the timestamps obtained from the file containing automatic captions may split words in the audio. This is remedied by a 0.5 second correction of timestamps that moves them to the nearest silence interval. The silence intervals are identified by means of a Python shell for the Voice Activity Detector utility from WebRTC, an open source project.

A separate directory is created for each video to store all its data. For speech segments that underwent successful correction (where correction was needed), timed audio segments in the 'wav' format are cut from the audio file and saved in the 'parts' directory, while 'parts.csv' stores a full path to such 'wav' file, the audio segment's size in bytes, and the corresponding text.

This approach allowed the most accurate alignment of transcripts with audio recordings. However, our algorithm turned out to be too selective: where the YouTube speech recognition tool failed to recognize correctly as few as one word in a string of user-provided captions, this string was removed from the dataset. This resulted in a very slow process of corpus building—approximately two hours of recorded speech in one hour of real time. Furthermore, the data collection statistics revealed that only about 2.5% of all downloaded audio was included in the final corpus: for a couple of days, we collected only 140 hours of speech, while many works on speech recognition mention that the corpus required for a deep neural network training should comprise 400 to 1000 hours. Thus, the described approach did not seem to be optimal, taking into account the limited amount of subtitled videos in the Russian part of YouTube.

3.2. Silence intervals alignment based approach

This approach, similarly to the previous one, uses a list of YouTube video identifiers that are somehow obtained at the beginning of the algorithm. Then the audio files and the 'vtt' files are downloaded for those videos that have YouTube's automatic captions, while those videos that lack automatic captions are left out. The audio files are converted into the 'wav' format (16000 Hz, mono, 16 bit). The approximate (0.3-0.5 second margin) start and end times recognized by YouTube's transcribing tool are retrieved for each word from the 'vtt' file. The audio is cut into segments from 5 to 20 seconds in length separated by silence intervals that are determined by means of the WebRTC VAD speech detector. Then the words corresponding to an audio segment's time are selected for each segment. The selected words are sorted according to their start times and grouped in one string where they are separated by spaces. Thereafter, the audio is subject to volume adjustment (to the level of 10 dB), and the bandpass filter with the range of 200-3000 Hz is applied to it.

This approach allows to build the speech corpus twice as fast as the previous one, and there are two reasons for that. First, the entire audio file is accurately and sequentially separated into segments based on silence intervals at the very first stage, while the previous approach involves start and end time corrections for each segment. Second, this approach does not need user-provided captions, which gives access to a much larger collection of videos—though, at the expense of transcript quality. The processing of 100 hours of speech takes only one hour of real time, which is by 50 times faster compared to the previous approach.

4. Getting speech data from radio shows

The Echo of Moscow radio station's website became another source of training material. The website provides open access to at least 2,000 audio files (radio programmes) and their transcripts, which makes it an attractive source for building quite a large speech corpus. Each radio programme contains about 40 minutes of speech.

We identified several problems related to the extraction of the required data from the website in question. First, the transcripts are not linked to the audio recordings in any way, which makes it unclear how we could divide the audio into segments and align these segments with the transcribed

text in order to create training samples. Second, the radio programmes contain signature tunes and advertisements with speech segments that are not included in the transcript. Third, many programmes contain dialogues, and both participants often speak at the same time—the detection and removal of such segments is a complicated task. Fourth, the dialogue form is characterized by sounds (interjections) that reveal the interlocutors' mental process—these sounds are lacking in the transcript and should be removed from the audio. Fifth, speakers often reiterate words for two or more times in a row, which is also a challenge. We managed to solve the first two problems mentioned in this paragraph, while the last two remain unresolved.

4.1. Automatic audio-text alignment

There are no reliable speech recognition systems for the Russian language in open access. This prompted us to use Dynamic Time Warping (DTW) in order to compare two audios in the process of audio-text alignment [1].

The alignment was made with the 'aeneas' utility. That is the point where we faced the problem of advertisements lacking in the transcript. This approach required us to delete advertisements and signature tunes from the audio.

4.2. Automatic advertisement recognition

The radio programmes contained two types of video to remove: signature tunes and advertisements. All programmes had the same signature tune, which allowed us to delete it through searching for an acoustic fingerprint in the database collected beforehand [2]. For this purpose, we used 'audfprint'—a Python library that allows to create a fingerprint database and detect a fingerprint in an audio file. Our database contained only one fingerprint: the signature tune. This method enabled us to determine the start and end times of the signature tune with a 1-2 second accuracy, even where a radio programme used only a tiny part of the tune.

We could not use the same workflow to delete advertisements, since they began in a variety of ways. We noticed that the speech rate and sound volume were higher in advertisements than in the rest of a programme. We analysed the amplitude of audio signals to locate high-density and high-volume regions—in almost all audios these corresponded to advertisements and signature tunes with a 5-10 second error. The signature tune was found in the beginning of a programme and after advertisements—in the latter instance, it marked the end of an advertisement and, thus, allowed to detect the end of advertisements using the acoustic fingerprint, which is a more accurate method.

Therefore, we managed to delete the advertisements and signature tunes from radio shows and, therefore, to prepare the audio files for processing by an automatic DTW aligner.

However, there were two other problems mentioned elsewhere in this paper: additional sounds (interjections) and overlapping remarks of two interlocutors. These problems did not allow the automatic aligner to align the bulk of audios, which prevented us from using this method in automatic building of a speech corpus.

5. Automatic data cleaning

Our study indicates that the method described in section 3.2 is the most efficient in terms of both resource-intensity and quality. This method involves the division of an entire audio file in segments using a silence detector, which is followed by the selection of words for each fragment from YouTube's automatic captions.

Though the method in 3.2 is the best among the automatic methods compared in this paper, it fails to provide the dataset quality comparable to that of datasets prepared by humans. The inaccuracies result due to poor detection of word boundaries in audio files. We assessed 1000 hours of data in the 'yt-vad-1k' dataset created by method 3.2 to reveal that over 30% of samples have additional or lacking words in the beginning or end of transcripts. We tackled this problem with an intermediary acoustic model trained on the 'yt-vad-1k' dataset containing 1000 hours of recordings with cutting errors. The model was used on the test sub-dataset 'yt-vad-1k-test' and returned the recognition errors of 35.2% WER and 11% CER. Transcript prediction was performed for each of the

samples. The Levenshtein distance was used to compare the first and the last 15 symbols between the expected and the predicted transcripts. The samples whose Levenshtein distance exceeded 0.5 at the beginning or end of transcripts were removed from the dataset. This allowed to automatically clean the datasets by deleting samples with inaccurate cutting and obtain two new datasets ‘voxforge-ru-clean’ and ‘yt-vad650-clean’ of 11.5 and 650 hours respectively.

6. Acknowledgement

This research was partially supported by the Russian Foundation for Basic Research grants (projects no. 17-29-04288, 16-07-01113). The authors would like to acknowledge the Reviewers for the valuable recommendations that helped in the improvement of this paper.

7. Conclusion

The paper describes several methods used to automatically collect data from various sources in order to train a neural network employed for speech recognition. We identified the most efficient data collection method that allows to promptly collect hundreds of thousands hours of speech transcripts from the YouTube video hosting.

This method was used to collect the dataset ‘yt-vad-1k’ comprising 1000 hours of transcribed recordings made by thousands of people in various recording conditions and with varying degree of background noise. We proposed a method to improve the quality of data by automatically removing the samples that contain cutting errors. The cleaning method was tested and allowed to obtain two more datasets: ‘voxforge-ru-clean’ и ‘yt-vad-650-clean’.

Further work is needed to improve the quality of data obtained from the Echo of Moscow radio station’s website. Specifically, we need to develop a method to detect repeated sounds or words and delete them from an audio file, as well as to clean files from segments containing overlapping speech. The solution of these problems will facilitate a qualitative expansion of the available dataset, which will result in a better recognition accuracy of the existing acoustic model.

References

[1] Muller M. Dynamic time warping // Information retrieval for music and motion. 2007. pp. 69-84.

[2] Haitsma J., Kalker T. A highly robust audio fingerprinting system. // Ismir., Vol. 2002., 2002., pp. 107-115.

[3] Graves A. Supervised sequence labelling // Supervised sequence labelling with recurrent neural networks., Springer, 2012., pp. 52—73.