Tutorial for the Spoken CALL Shared Task

Mario Kunstek[1], *Cem Freimoser*[1], *Dominik Jülg*[1], *Kay Berkling*[1], *Mengjie Qian*[2]

[1]Cooperative State University, Karlsruhe (CSU-K), Germany
[2]Department of Electronic, Electrical and Systems Engineering, University of Birmingham

kunstek.mario@googlemail.com, freimoser.c@gmail.com, djuelg@gmx.de, berkling@dhbw-karlsruhe.de, mxq486@bham.ac.uk

Abstract

This paper presents a tutorial and tool-box for working with the shared task for spoken Computer Aided Language Learning (CALL) for English as a Second Language (ESL). This is a computer-based dialogue system for Swiss school children learning English that should be able to judge the correctness of syntax and semantics in their responses. In order to work with this task, a number of files and baseline systems are provided to the researchers. The number of files, the syntax of ID's and the scripts as well as their usage in a pipe-line architecture can be confusing to a novice researcher in the field. In order to facilitate work with this task, this paper proposes a tutorial set-up and an architecture. The resulting tool-kit is an open source project on Github that walks the user through the corpus, setting up the system for training and comparing the results against human judgment. The contribution of this paper is the description of the tool-box in order to enable new research groups to find their way around the problem statement more easily, avoiding many of the typical start-up pit-falls.

Index Terms: CALL, speech recognition, ESL, tutorial, Corpus

1. Introduction

This paper represents a tutorial for the shared task described in [1, 2, 3]. For the purpose of completion, we will briefly review the basic idea behind the CALL application; further details can be found in the above publications.

The exercise to be recognized and scored is of the type prompt-response, where the German-speaking student is prompted to either respond to a request or translate a request or sentence into L2, which is English. The automated system should ideally accept a correct response or reject a student response if faulty and offer relevant support or feedback. There are a number of prompts (given as text in German, preceded by a short animated clip in English), namely to make a statement or ask a question regarding a particular item.

A baseline system is provided by the website of the project as well as output files of the system to attempt only the second task of the judgment, disregarding the speech recognition system. The final output of the system should be a judgment call as to the correctness of the utterance. A wide range of answers is to be allowed in response, adding to the difficulty of giving automated feedback. Incorrect responses are due to incorrect vocabulary usage, incorrect grammar, or bad pronunciation and quality of recording.

Part of the problem for any novice in getting started with the package are the conventions used and the changes that have to be made to use the data, retrain the system and understand the judgment. The goal of this tutorial is to ease the start-up time for new researchers. This tutorial is written by Bachelor students at Cooperative State Universiy, supported by PhD student Mengjie Qian from Martin Russell's laboratory at the University of Birmingham in a PhD2Bachelor knowledge-transfer project supported by an ISCA scholarship.

The rest of the paper will describe the given system and then introduce a tool-kit that supports a faster boot-strapping for new teams with the task based on the The CSU-K DNN-Based System for the 2nd Edition Spoken CALL SharedTask[4].

Section 2 describes the input files of the task from the point of view of the developer. Section 3 walks the reader through the tool-kit that follows the described architecture. The final section has some thoughts on future work.

2. Description of the Input Files and Systems

The downloadables for the Spoken Call Shared Task consists basically of three components: The Corpus, the baseline system (ASR and Text Classification), a number of file and scripts to work with the data.

2.1. Corpus

The data for the shared task was collected in 15 school classes at 7 different schools in the German speaking areas during a series of experiments in 2014 and early 2015. To compare automated system performance, human annotators judge each interaction in order to determine whether or not the utterance should have been accepted by the system. The training corpus contains 5,000 utterances. There are corpora from the years 2017 and 2018 and for 2018 a new evaluation test set was defined for comparison of systems. The corpus for the shared task has been annotated with correct transcription and a correct/incorrect tag regarding grammar, vocabulary, pronunciation and fluency. The corpus for the 2nd Edition of the Spoken CALL System combines the 1st Edition data with newly released data as shown in Table 1. The table enumerates the number of available training and test utterances for both editions (ST1, ST2). Examples of the data are given in Table 2.

Table 1: *# of utterances in the raining and test data of Shared Task 1 and Shared Task 2.*

	train	test
ST1	5222	996
ST2	6698	1000
Total	12916	1000

Possible problems with the presentation of the corpus are the identification of each utterance by either numbers of wave files. Especially, when working with neural networks, these

Table 2: *Sample Data-Set. L=Language, M=Meaning, NBad=difficulty of audio, I=Incorrect, C=Correct.*

ID	Prompt	Transcription	L	M
4596	Frag: mein Steak durchgebraten	i would like my steak well down	I	C
4596	Frag: mein Steak durchgebraten	i would like my steak well down	I	C
3709	Sag: Ich habe 2 jüngere Brüder	i have two young brothers	I	I

ID's have to be reformatted as numbers. The toolkit will address these issues. Problems with processing can be posed by the prompts being in German, while the responses are in English. Furthermore, the tasks requires handing in reject or accept of the utterance, while the file in Table 2 gives two columns of correctness, language and meaning. The actual correct or not statements are composed of these two and the D-measure combines the mis-judgments differently based on language and meaning intermediary judgments. This distinction causes some delay in grasping the problem statement.

2.2. Files

The website for the shared task includes all data and baseline which were used in ST1 and ST2 [5]. Figure 1 depicts the file structure of the downloaded task files.

2.3. Kaldi - System

A baseline system is provided for the shared task in the form of the open source speech recognizer Kaldi and a corresponding language model.

2.3.1. Setup and Structure

Kaldi is a toolkit for speech recognition.[1] The project has to be checked out from the git-project [6]. This project contains some folders which are briefly explained in the following.

- egs - under this folder contains almost all scripts which are needed to train the recognition models for dnn, rnn, lstm,´and sequence training. especially the scripts in the sub folder egs/wsj are used. This folder also contains examples and demonstrations for working with Kaldi.
- misc - various files and some related papers
- scripts - some example scripts from Kaldi which explaines how to train RNN models that may not be needed when training with SRILM.
- src - contains all needed classes, modules, methods and other files which Kaldi needs. In the /cudamatrix can be tested whether Kaldi and Cuda are installed correctly by compiling it with 'make test'.
- tools - Important tools for Kaldi, most of them are used to build the recognition systems. There are also many scripts which were used by the examples in egs.
- windows - instructions for Windows

[1]`http://kaldi-asr.org/`

One can test whether Kaldi has been installed correctly under the src dir, cd src/cudamatrix, make test, ./cu-vector-test. The recognition system will be based on the scripts in egs, especially egs/wsj.

2.3.2. Cuda

Kaldi uses parallel processing to train neuronal networks it's recommended to use single or multiple GPUs. For this reason, the graphic cards need additional drivers. For NVIDIA graphic cards Cuda [7] has to be installed to enable parallel processing by the installed GPU. Make sure to install the correct version of Cuda. Ensure that the Kaldi version you are using matches the Cuda version. For this tutorial the version number is CUDA-9.0. Also take note of the version required by the tensorflow library. There are specified versions that work together. This may not be the newest version.

2.3.3. cuDNN

Cuda 2.3.2 does not enable training of dnn models. To train dnn, the NVIDIA graphic card needs the cuDNN driver. It provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers [8]. Note that the version of cuDNN dependents on the installed cuda version. First, it is important to check which versions are intended to be used together with the installed Cuda version. Using CUDA-9.0 the corresponding cuDNN version is 7.0. Furthermore, check which versions are working well together with Tensorflow, keeping in mind that its not always best to have the newest version, because they might not be supported.

2.3.4. Experiments

In the egs folder see 2.3.1 are some example experiments from Kaldi. All of them have a README file that explains what this experiment does. The examples are tutorials for building different kinds of recognition. These experiments can be run with particular data chosen for training. These experiments help to learn how the different recognition systems work. Some of these have speaker recognition models.

Kaldi installation can easily be validated using the yes/no corpus stored in the folder: kaldi-folder/egs/yesno by following the steps written in the README. For example, run: kaldi-folder/egs/yesno/s5/run.sh. The installation is correct when no errors are displayed.

2.3.5. Scripts

In the scripts folder 2.3.1 are scripts that can be used for recurrent neuronal networks (RNN). These Scripts can be used for training the kaldi model. For this tutorial as well as in the referenced system [4] these scripts are not needed, because another model will be trained (see Section 3.3).

3. The CSU-K Tool-Kit

The CSU-K Toolkit helps with guiding the researcher through the installation and testing process of various tools. It then supports data preparation to avoid some of the tricky parts that can make this process frustrating. Training and testing of Kaldi system is then supported with a number of explanatory scripts followed by the back-end, that will take the text-output from Kaldi and present it to the classifier that decides whether the utterance is correct. Finally, the D-value that is used for comparing any

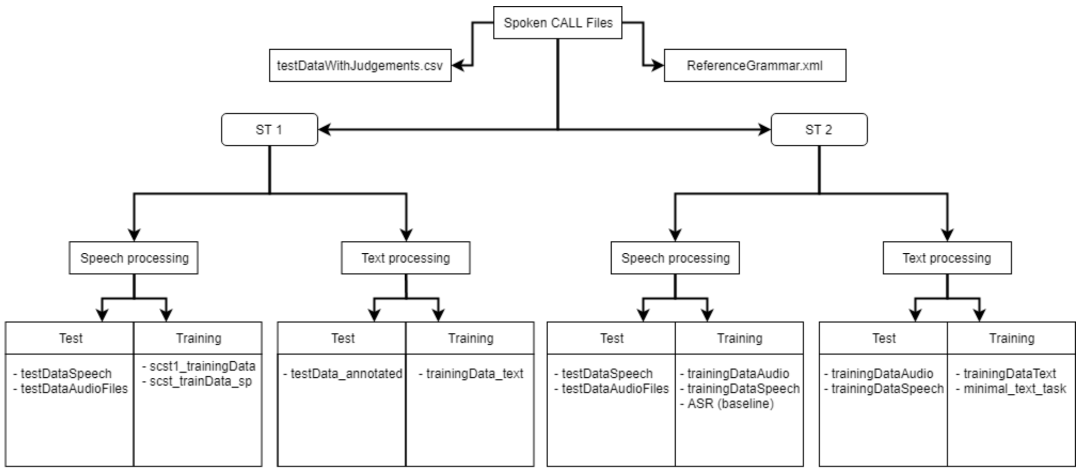

Figure 1: *File structure*

new system to previous ones is explained, showing how to use the toolkit script.

Using the CSU-K toolkit, researchers can tackle the problems of the shared task more quickly. It provides all necessary scripts to build the CSU-K DNN-based system independently Researchers are slowed down with a series of boiler flat tasks. For example, to drain the CSU-K DNN system itself, a separate test and train set must first be created. The CSU-K toolkit provides a number of scripts that can help to separate the data. Among others, separation can be performed based on a simple modulo function, but separation based on IDs is also possible with the CSU-K toolkit. In addition, a script is given with which one can check again to ensure disjoint sets. After separation, the data must still be converted into a format that can be used for the CSU-K DNN. The CSU-K Toolkit also offers scripts for this. Thus, the X-vectors and the Y-vectors can be generated using the CSU-K toolkit. The CSU-K toolkit also offers scripts that we have used to follow different approaches. Researchers can therefore generate grammar and language from the training data from the shared task. Moreover, the CSU-K toolkit can help to retrain a Doc2Vec model. Scripts for troubleshooting are also provided here in order to check which matches there are regarding the ID between different input files. (An overview table could be shown in an appendix on a fifth page if allowed by Interspeech. In the meantime it is visible in the git project.[2])

An overview of the system is depicted in Figure 2. The components will be described next.

3.1. Data Preparation

The design of the IDs differs fundamentally between ST1 and ST2. For ST1, for example, only a numerical design of the ID was chosen. During this time, an alphanumeric design was selected for ST2. Furthermore, the data from ST1 does not provide any information about the speaker. This means there are no special IDs for the speaker. Therefore a 1:1 relation is often used. This means UtterenceID equals SpeakerID. With the data from ST2, however, the speaker was coded into the Utteance ID. Thus, the alphanumeric string contains a part of the Utterence assigned to a speaker. What has not been taken into account

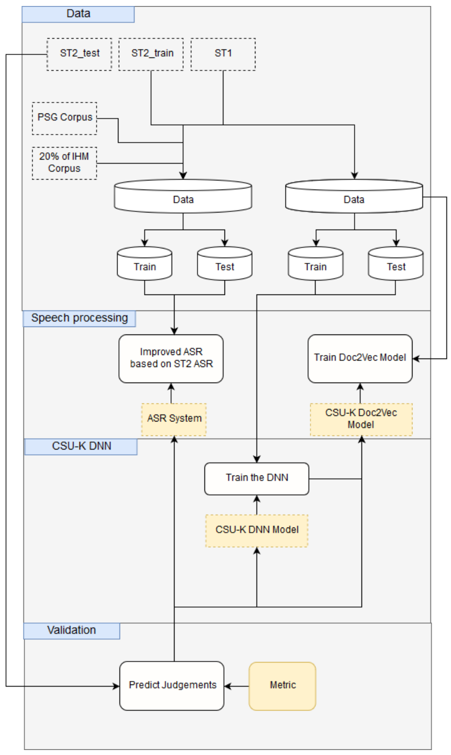

Figure 2: *System Overview*

here, however, is to fill in the empty places with zeros. Therefore, this ID is not well suited for Kaldi. A correction as shown in Table 3 is necessary. The CSU-K toolkit provides the script

After downloading all the files (see Sec. 2.2) from the web page of the Spoken CALL Shared Task[5] the train data, test data and validation data have to be chosen. In ST1 and ST2 there are csv files which are named A, B and C. The difference between these files is that native speakers already checked the spoken results from the children and made a judgment whether this will be accepted or rejected. A.csv contains "good" data where all native speaker and all used machines have the same opinion. The B.csv file contains data where all native speaker

[2]https://github.com/Snow-White-Group/CSU-K-Toolkit/blob/master/README.md

Table 3: *st2-Id*

Id	Wavfile
5	user-8_2014-05-11_17-57-14_utt_012.wav
user-8_2014-05-11_17-57-14_utt_012	user-8_2014-05-11_17-57-14_utt_012.wav
user-008_2014-05-11_17-57-14_utt_012	user-8_2014-05-11_17-57-14_utt_012.wav

have the same opinion but the machines have different ones. C.csv includes only the bad data where no good labeling is possible because the speakers haven't the same opinion. Reasons for this can be that a child in the record has pronounced it wrong where some speakers said that it is anyways correct and other rejected it. The wanted data has to be joined together and get split in a train set and a test set.

After creating a csv file containing all the utterances of interest, the split_file.py script creates a test and a training set. These files should be consistent in delimiters. The normalize_* scripts will perform this function so that the files can be merged.

3.2. Automatic Speech Recognition (ASR)

The speech processing part deals with generating feature files with the run_train.sh script of the Automatic Speech Recognition (ASR) system that trains the language model 3.2. It's important that each command in the run_train script is executed step-by-step. This script contains additional comments which explain the target step and whether this has to be executed or not. This script can be changed and adapted to one's needs. The following steps are only needed if no model exists or a new model should be trained.

The ASR needs four other files to enable training the LM. These are:

```
wav.scp - format:
    <recording-id> <path to wav-file>
utt2spk - format:
    <utterance-id> <speaker-id>
spk2utt - format:
    <speaker-id> <utterance-id1>
        <utterance-id2> <utterance-id3>
...
text - format:
    <utterance-id>
        <transcription of the sentence>
```

The next step is to train a model with the run_train script. Note that the run_train is not run at once but command by command.

init_params.sh: This is a script that was created to give some guidance to the order of tasks that are necessary to gain an understanding of how to use runtrain script. This script sets global parameters used for the training and executes the scripts cmd.sh and path.sh which also set global values and trigger some other procedures which are needed in the background.

run_train.sh: Script for training the language model (LM) and generating the features. This script has additional parameters which can be used to configure how to train the LM.

The system used here is provided with the task and is based on the winning Edition 1 ASR system [9].

3.3. Text Processing

To process the RecResult of Kaldi with CSU-K DNN (see [4]) based classifier a pre-processing must take place beforehand. Pre-processing is the conversion of the character string into a format suitable for the CSU-K DNN based classifier. The CSU-K DNN based classifier can be considered as a simple function: $f(x) = y$. It tries to map an input x to a y. The x are a simple scalar values. They are between 0-1 and indicate how likely it is that the child's testimony will correctly solve the task. The x is a 10-dimensional vector. This vector describes the distance between the child's statement and the sample solutions. To generate this vector, the generateDistanceVectors script from the CSU-K toolkit is required. As a result of this script one obtains a CSV file. One possibility to modify this approach is by exchanging the Doc2Vec model. Besides the possibility to use one of the most popular models like FaceBook or Google, there is also the possibility to use the CSU-K Doc2Vec model [3]. However, one can also create a new model using the generateDoc2VecModel scripts from the CSU-K toolkit. It is important to understand that the format of the created feature vector (by the generateDistanceVectors script) is not changed. However, the values that the individual features represent can change significantly. This can be converted into an array using standard Numpy[4]. Consider that the first column represents the ID and is not needed. The CSU-K DNN Model can be downloaded from GitHub[5]. This is a deep neural network created with Keras [4]. Once the model has been loaded, the input can be passed to the DNN. Now you get the corresponding probabilities. A complete example can still be found on GitHub[6]. Of course it is also possible to train one's own DNN instead of using the CSU-K DNN.

4. Future Work and Conclusions

In this paper, we have attempted to assemble a tutorial and toolkit in order to facilitate a start for novice users of the system. It is an open source git project to support regular updates by the community. Possible improvements for future approaches would include a database instead of csv files and uniform IDs, perhaps with speaker ID in a separate column.

5. Acknowledgements

This work was performed by Bachelor students as part of their capstone project. In the absence of a local speech lab, the authors thank University of Birmingham Martin Russel's students Mengjie Qian and Xizi Wei for visiting our lab. We thank ISCA for providing scholarships for Mengjie the "Förderverein" for supporting student travel to the conference.

[3]https://github.com/Snow-White-Group/CSU-K-Toolkit/tree/master/models

[4]https://docs.scipy.org/doc/numpy/reference/generated/numpy.loadtxt.html

[5]https://github.com/Snow-White-Group/The-CSU-K-DNN-Based-System-for-the-2nd-Edition-Spoken-CALL-Shared-Task/tree/master/Dev3

[6]https://github.com/Snow-White-Group/The-CSU-K-DNN-Based-System-for-the-2nd-Edition-Spoken-CALL-Shared-Task/blob/master/Dev3/dev3.ipynb

6. References

[1] C. Baur, J. Gerlach, M. Rayner, M. Russell, and H. Strik, “A shared task for spoken call?” in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Paris, France: European Language Resources Association (ELRA), May 2016.

[2] C. Baur, C. Chua, J. Gerlach, M. Rayner, M. Russell, H. Strik, and X. Wei, “Overview of the 2017 spoken call shared task,” in *Proc. 7th ISCA Workshop on Speech and Language Technology in Education*, 2017, pp. 71–78. [Online]. Available: http://dx.doi.org/10.21437/SLaTE.2017-13

[3] A. Caines, “Spoken CALL Shared Task system description,” in *Proc. 7th ISCA Workshop on Speech and Language Technology in Education*, 2017, pp. 79–84. [Online]. Available: http://dx.doi.org/10.21437/SLaTE.2017-14

[4] C. Freimoser, M. Kunstek, D. Jülg, K. Berkling, and M. Qian, “The CSU-K DNN-Based System for the 2nd Edition Spoken CALL Shared Task,” in *Proc. Interspeech*, Hyderabad, India, September 2018.

[5] U. of Geneva. (2017) Spoken call shared task - second edition.

[6] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, “The kaldi speech recognition toolkit,” in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[7] NVIDIA. (2006) Cuda - computing on gpu. [Online]. Available: https://developer.nvidia.com/cuda-zone

[8] ——. (2013) cudnn - deep neuronal network library. [Online]. Available: https://developer.nvidia.com/cudnn

[9] M. Qian, X. Wei, P. Jančovič, and M. Russell, “The university of birmingham 2017 slate call shared task systems,” in *Proc. 7th ISCA Workshop on Speech and Language Technology in Education*, 2017, pp. 91–96. [Online]. Available: http://dx.doi.org/10.21437/SLaTE.2017-16