The transLectures-UPV toolkit

M. A. del-Agua, A. Giménez, N. Serrano, J. Andrés-Ferrer, J. Civera, A. Sanchis, and A. Juan

MLLP, DSIC, Universitat Politècnica de València (UPV),
{mdelagua,agimenez,nserrano,jandres,jcivera,josanna,ajuan}@dsic.upv.es

Abstract. Over the past few years, online multimedia educational repositories have increased in number and popularity. The main aim of the transLectures project is to develop cost-effective solutions for producing accurate transcriptions and translations for large video lecture repositories, such as VideoLectures.NET or the Universitat Politècnica de València's repository, poliMedia. In this paper, we present the transLectures-UPV toolkit (TLK), which has been specifically designed to meet the requirements of the transLectures project, but can also be used as a conventional ASR toolkit. The main features of the current release include HMM training and decoding with speaker adaptation techniques (fCMLLR). TLK has been tested on the VideoLectures.NET and poliMedia repositories, yielding very competitive results. TLK has been released under the permissive open source Apache License v2.0 and can be directly downloaded from the transLectures website.

Keywords: TLK, ASR toolkit, transLectures, HMM

1 Introduction

Online multimedia repositories are on the rise and becoming evermore consolidated as key knowledge assets. This is particularly true in the educational arena where large repositories of video lectures are being established on the back of increasingly available and standardized infrastructures. A well-known example of this is VideoLectures.NET, a free and open access web portal that has so far published more than 15K educational videos. VideoLectures.NET is a major player in the diffusion of the open source Matterhorn platform currently being adopted by many institutions and organizations within the Opencast community [3]. Other examples include massive open online course (MOOCs) aggregators, such as Coursera, Udacity, EdX, Udemy, iVersity, UPV[x] and others.

The generation of subtitles for these repositories is a costly task, both in terms of time and money, which prohibits many repositories from having their videos transcribed. Most of the video lectures available on VideoLectures.NET and MOOC aggregators, for instance, are not transcribed, despite the obvious benefits of doing so, including the incorporation of search and analysis functions. In order to overcome this deficit, the transLectures project aims to develop innovative, cost-effective solutions for producing accurate transcriptions and translations for video lectures. The project has two case studies: the aforementioned

VideoLectures.NET, and poliMedia, a Spanish and Catalan video lecture repository developed at the Universitat Politècnica de València (UPV).

An important area of work at transLectures is to develop solutions that can be easily transferred to other repositories beyond VideoLectures.NET and poliMedia. With this in mind, the transLectures-UPV team has developed a whole series of transferable tools, including online applications. This paper is focused on just one of these tools, the transLectures-UPV toolkit (TLK). TLK implements all the functionalities required to develop an automatic speech recognition (ASR) system. Although developed as part of the transLectures project to meet the specific requirements of video lecture transcription, it can also be used as a conventional ASR toolkit, like HTK [20], RASR [17] or KALDI [15]. In this paper, we go into more detail about this toolkit, which can be freely downloaded [6] under the permissive (for research and commercial purposes alike) Apache License v2.0.

This paper is organised as follows. Section 2 describes the different tools forming part of TLK that can be used either to build an ASR system or simply to transcribe input media files. A practical guide to the development of an ASR system using TLK is given in Section 3. Finally, the performance of TLK is assessed in Section 4, and some conclusions are given in Section 5.

2 Overview of the Toolkit

TLK can be divided into three major components: the library, the basic command line tools and the high-level command line tools. The library, named `libTLK`, is an ANSI C library and implements the core functionalities of TLK (feature extraction, parameter estimation, decoding, adaptation, etc.). A set of basic command line tools have been defined to use `libTLK`. Based on these basic tools, high-level command line tools have also been developed in order to carry out the main steps involved in building an HMM-based ASR system: preprocessing, training and decoding.

2.1 Building an ASR System Using TLK Tools

As illustrated in Fig. 1, an ASR system can be built using three high-level TLK tools: `tLtask-preprocess`, `tLtask-train` and `tLtask-recognise`.

tLtask-preprocess This tool takes time-segmented audio signals and the corresponding transcriptions as input and performs feature extraction and phonetic annotation. It also extracts clusters from the input audio, which can be used for speaker or video adaptation, and other useful data like the original or non-punctuated text.

`tLtask-preprocess` uses the `tLextract` basic command tool to perform the Mel-Frequency Cepstral Coefficients (MFCC) feature extraction process as described in [20]. `tLextract` supports a large number of audio file formats since it uses the `libsox` library. The parameters involved in the extraction process

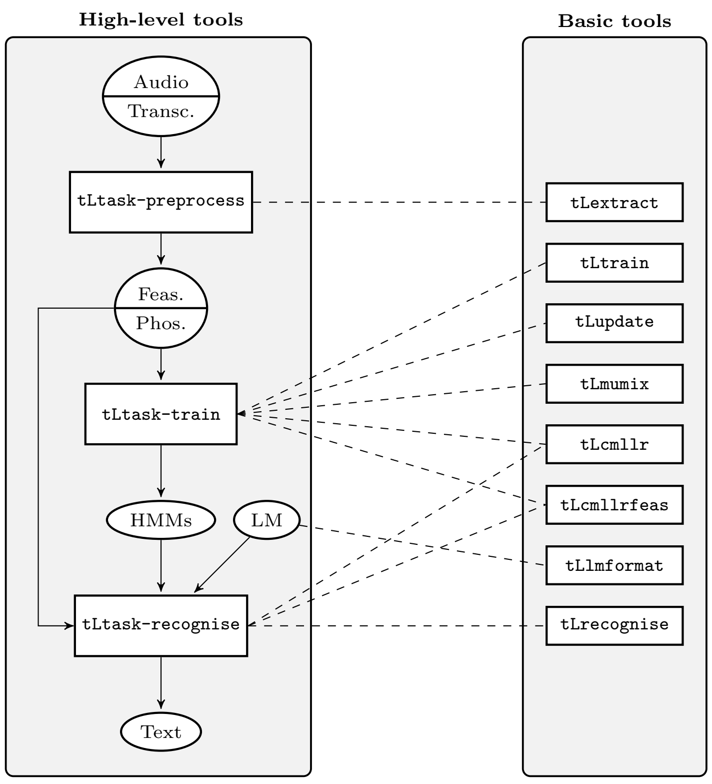

Fig. 1. Building an ASR system using TLK tools.

are easy to configure: sampling frequency, duration of the extraction window, number of cepstral coefficients, etc. Furthermore, `tLextract` also allows the application of a mean variance normalization to the input samples.

The phonetic transcription is obtained using different auxiliary scripts depending on the input language. The current release supports Spanish and Catalan.

tLtask-train. This tool takes the output from `tLtask-preprocess` and performs the following training schema to estimate the HMMs:

1. Standard model training: monophone training, triphone training, transformation of the triphone model to a tied phoneme model, tied phoneme training.
2. Estimation of CMLLR matrices and CMLLR features.
3. CMLLR model training: CMLLR monophone training, CMLLR triphone training, CMLLR transformation of the triphone model to a tied phoneme model, CMLLR tied phoneme training.

This is the training schema for a two-step recognition system using fCMLLR features [9], and tied-state triphone HMMs. The final standard and CMLLR models are made up of Gaussian mixture distributions estimated following on an iterative training schema in which mixture components are mixed at each iteration (mixing is performed using `tLmumix`). Tied-state triphone HMMs are estimated following a phonetic decision tree approach [21]. This technique is implemented as an auxiliary Python script based on predefined linguistic rules. These rules are implemented as regular expressions in Python and can be easily defined by users. The current release includes rules for English, Spanish and Catalan.

`tLtask-train` uses the `tLtrain` basic command tool which implements Baum-Welch and Viterbi algorithms for parameter estimation [7, 19]. `tLtrain` has been designed to be able to properly manage large corpora by scaling in cluster environments. Specifically, `tLtrain` is used by `tLtask-train` following a Map-Reduce approach. That is, training is split into two stages: a first stage in which `tLtrain` is used to compute statistics, which can be split over several independent processes; and a second stage where the statistics computed in the previous stage are merged using the basic command line tool `tLupdate`. It is worth noting that `tLupdate` has support for linear interpolation of counts which might be useful in an online learning schema. Additionally, `tLtrain` allows samples to be packed into tar files for a better I/O latency in a cluster environment.

`tLtask-train` uses additional basic command tools to complete the CMLLR model training. `tLcmllr` is used to calculate a transformation matrix over all Gaussian mixtures of a simple HMM using the Constrained MLLR algorithm (CMLLR), while `tLcmllrfeas` transforms samples into fCMLLR features using a CMLLR transformation matrix.

tLtask-recognise. This tool transcribes audio samples produced by `tLtask-preprocess` using HMM models estimated by `tLtask-train` following a two-step recognition schema:

1. Recognition using the standard tied phoneme HMMs.
2. Estimation of CMLLR matrices.
3. CMLLR transformation of input samples.
4. Recognition using the CMLLR tied phoneme HMMs.

`tLtask-recognise` uses the basic tool `tLrecognise`, which implements the well-known Viterbi algorithm, to obtain the most probable hypothesis [19]. In addition to HMMs, a language model and a pronunciation dictionary must be provided for decoding. `tLrecognise` allows two different language model representations. If the language model is a wordnet (without back-off), decoding is carried out over a huge finite state model built by embedding HMMs into the states of the wordnet [20]. In contrast, if the language model is in ARPA format (back-off), the decoder follows a word-conditioned tree search approach [12]. Specifically, a prefix tree with all the possible pronunciations is pre-calculated. To speed up the process, prior to decoding (`tLtask-recognise` or `tLrecognise`), the language model must be transformed into an internal format. This transformation is carried out by the basic tool `tLlmformat`. `tLrecognise` implements several well-known pruning techniques: beam search, histogram pruning, word end pruning and look-ahead. Although look-ahead is not exactly a pruning technique, its use is highly recommended when pruning techniques are applied in conjunction with a prefix tree approach [13]. `tLrecognise` also supports the generation of lattices following the technique described in [14]. Two formats for lattices are supported: the TLK format and the HTK format [20]. If desired, lattices can be generated including information related to time alignment at phoneme level.

As with `tLtask-train`, `tLtask-recognise` has been designed to work well in cluster environments. Specifically, it can be configured to split recognition into parallel processes, and cache big files (like models) on host machines.

The output of `tLtask-recognise` is given in different formats: plain text, recognise output, CTM format [4], etc.

2.2 Using TLK Tools For Decoding Only

TLK includes a high-level tool named `tLtranscribe` that allows users to directly transcribe media files. This tool reads a preinstalled system, freeing the user from all the technical details. As illustrated in Fig. 2, `tLtranscribe` makes use of the high-level tools `tLtask-recognise` and `tLtask-segment`. The tool `tLtask-segment` uses `tLextract` to automatically perform the segmentation of the audio signal. For the purposes of testing the `tLtranscribe` tool, a system for Spanish transcription has been released under a Creative Commons Attribution 4.0 International License.

3 Using TLK

This section describes how an ASR system can be built using TLK following the process depicted in Fig. 1. A more detailed version of this tutorial is available on the transLectures website [6].

1. TLK installation and data preparation.
   - The current version of TLK runs on Linux and Mac OS X, and can be easily installed from the transLectures website.
   - Acoustic data is also available on the transLectures website and can be downloaded by executing:
     ```
     wget translectures.eu/files/tlk/tlk-tutorial-data.tgz
     tar -xzvf tlk-tutorial-data.tgz
     ```
     This will create the directory `tlk-tutorial-data`, which itself contains several directories. The `train` directory contains the data that will be used to train HMMs, while the `test` directory contains the data that will be used to asses the system. These data correspond to Spanish lectures recorded at Universitat Politècnica de València and their annotations in .trs and .dfxp format.

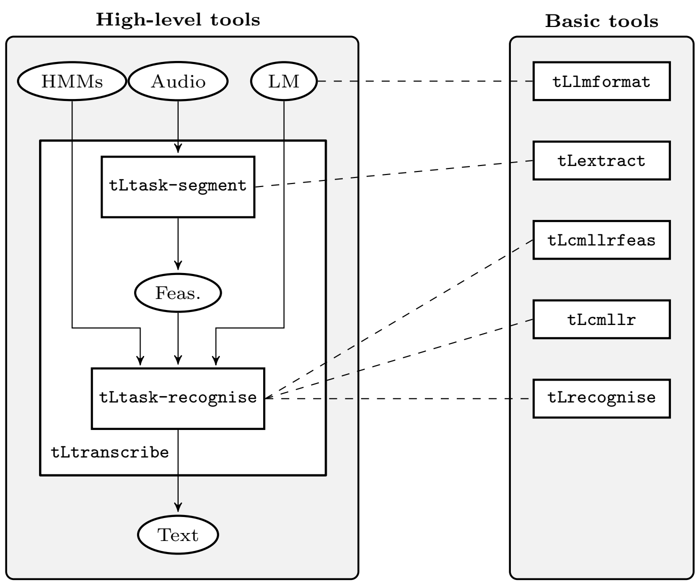

Fig. 2. Transcribing media files with `tLtranscribe`.

- Now, running `tLtask-preprocess` the data is preprocessed obtaining the required files for training and evaluation:
  ```
  tLtask-preprocess es dfxp tlk-tutorial-data/train preprocess-train
  tLtask-preprocess es dfxp tlk-tutorial-data/test preprocess-test
  ```
  Note that the configuration options (i.e. `es` and `dfxp`) indicate the language and the file format, respectively.

2. HMM training:
   - First of all, a directory should be created to store the training files:
     ```
     mkdir training; cd training
     ```
   - Then, the two directories inside `preprocess-train` need to be linked to the training directory:
     ```
     ln -s ../preprocess-train/samples ../preprocess-train/lists .
     ```
   - Next, a template of the tool's configuration file `tLtask-train` should be generated:
     ```
     tLtask-train --write-example-config-file > config-file.ini
     ```
     This configuration file contains the default parameters needed to train standard HMMs for the Spanish language. In order to use previously preprocessed acoustic data, the `Lists` section of this configuration file has to be changed:
     ```
     [Lists]
     set_name = lists/samples
     ...
     [General]
     ...
     prefix-name = training-tutorial
     ```

   – Finally, the following command runs the tool `tLtask-train` to perform the HMM training:
     ```
     tLtask-train config-file.ini --log-folder log
     ```
     The tool `tLtask-train` will execute all necessary commands to train HMMs following the training schema described in previous section Note that, although certain processes are executed in parallel depending on the computer, this process might take some time.

3. Automatic transcription:
   – As in the case of training, a directory should be created in the base directory for storing the automatic transcriptions:
     ```
     mkdir recognition; cd recognition
     ```
   – Also, some links must be created to the acoustic data and models:
     ```
     ln -s ../preprocess-test/samples ../preprocess-test/lists \
           ../preprocess-test/references ../training/models \
           ../tlk-tutorial-data/misc/mono.lex \
           ../tlk-tutorial-data/misc/mlm.gz .
     ```
   – The tool `tLtask-recognise` needs a configuration file, easily generated by running:
     ```
     tLtask-recognise --write-example-config-file > config-file.ini
     ```
     Some changes need to be made to this file in order to use previously preprocessed test data:
     ```
     [General]
     prefix-name = tutorial
     ...
     [HMM]
     prefix-name = training-tutorial
     ...
     [LM]
     language-model = mlm.gz
     lexicon = mono.lex
     ```
   – Finally, upon executing the following command, the test audio samples will be automatically transcribed following the two-step recognition schema described in previous section:
     ```
     tLtask-recognise config-file.ini --log-folder log
     ```

4. Measuring the transcription quality:
   – The sclite tool in SCTK is used to compute the Word Error Rate (WER) of the automatic transcriptions [4]:
     ```
     sclite -r references/035040d6-7fd4-ab4a-80ff-e87d3a5d84db.stm \
            stm -h tutorial/cmllr_step2/transcription.ctm ctm
     ```

4 Empirical Results

TLK has been developed within the framework of the transLectures project to deal with the transcription of video lectures. Specifically, ASR systems have been developed for three languages: English, Spanish and Catalan. The English ASR system has been developed for the transcription of English lectures from the VideoLectures.NET repository. The Spanish and Catalan ASR systems have been developed for the poliMedia repository. For training and evaluation purposes, three databases have been developed by manually transcribing video lectures from these repositories. The main statistics of these speech databases are shown in Table 1.

Table 1. Main statistics of the English, Spanish and Catalan speech databases used in the transLectures project.

	English	Spanish	Catalan
Videos	28	704	210
Speakers	104	83	53
Hours	26.6	114.2	25.8
Sentences	7.3K	41.6K	13.7K
Running Words	192K	1M	198K
Vocabulary Size	13K	35.9K	24.4K

From each database some lectures were selected for evaluation purposes: 3.4h for Spanish and English, and 2.1h for Catalan. However, since video lectures from VideoLectures.NET are longer ($\approx$ 50min) than poliMedia lectures ($\approx$ 10min), this means just 4 videos were selected for English in absolute terms, while 23 and 16 videos were selected for Spanish and Catalan, respectively. The remaining data were used for training and development. For tasks where there was a lack of training data, as was the case for English and Catalan, the training data was increased by out-of-domain corpora.

The progress of the ASR systems developed within the transLectures project using TLK for each language is depicted in Fig. 3. As can be observed, the performances of the three systems have improved continuously throughout the project. In particular, very high performance levels have been achieved in Spanish (12.8% WER). Work began on the English and Catalan systems later than on the Spanish system (specifically, one year later). However, big improvements in WER have been achieved in the six-month period (20.1% in Catalan and 22.7% in English). In all languages, the performance is close or below 20% WER, which has been reported as the threshold under which ASR output becomes useful for users [11]. All these improvements can in part be explained by the fact that TLK has been under active development since the beginning of the project. This includes some features currently being tested, for example, hybrid models with deep neural networks (DNNs) [16, 8, 18], and multilingual DNNs [10]. It is worth noting that, in all cases, the language model used has about 200K words. Moreover, the percentage of out-of-vocabulary words is below 2% (1.7% for Spanish). For further details on the development of these systems, please refer to the public transLectures reports [2, 5, 1].

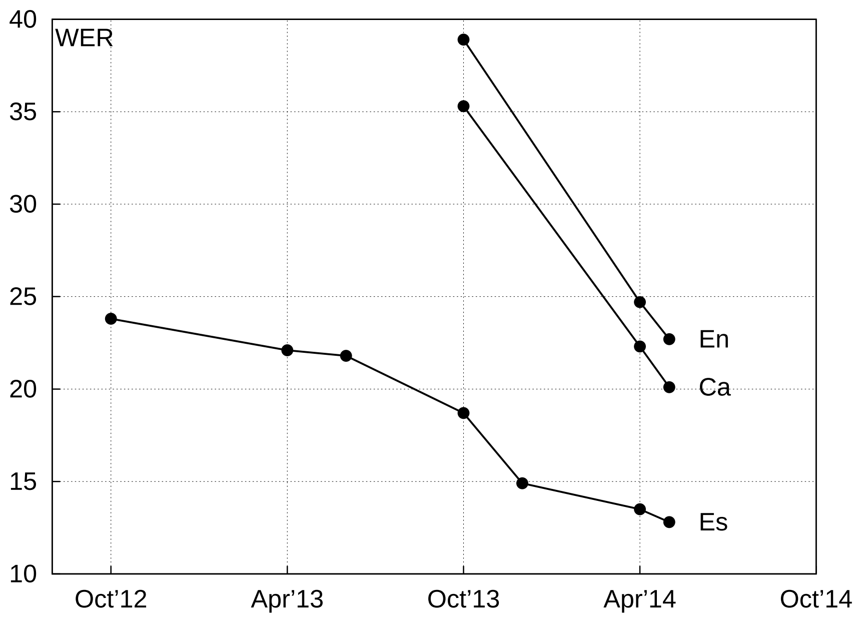

Fig. 3. Progress measured in WER of the TLK ASR systems developed within the transLectures project for Spanish (Es), English (En) and Catalan (Ca).

5 Conclusions and Further Remarks

In this paper we have presented the transLectures-UPV ASR toolkit (TLK) based on HMMs. TLK implements well-known ASR features and released under the open source Apache License 2.0. The functionality of TLK has been recently extended, adding a new component that supports Deep Neural Networks (DNNs) following a hybrid decoding approach [8]. Although the current release does not include DNN training, with this still being at an experimental stage, it does include DNN support for recognition. In fact, beside the standard Gaussian HMM based Spanish system, we have also released a Spanish system based on DNNs. Both systems can be downloaded from the transLectures website [6].

As future work, we plan to improve TLK further by adding new state-of-the-art features, such as convolutional NNs or recurrent NNs. Also, we plan to carry out extensive, comparative tests with other toolkits.

Acknowledgments. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 287755 (transLectures) and ICT Policy Support Programme (ICT PSP/2007-2013) as part of the Competitiveness and Innovation Framework Programme (CIP) under grant agreement no 621030 (EMMA), and the Spanish MINECO Active2Trans (TIN2012-31723) research project.

References

1. Final report on massive adaptation (M36). To be delivered on October 2014
2. First report on massive adaptation (M12), `https://www.translectures.eu/wp-content/uploads/2013/05/transLectures-D3.1.1-18Nov2012.pdf`
3. Opencast Matterhorn, `http://opencast.org/matterhorn/`
4. sclite - Score speech recognition system output, `http://www1.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.htm`
5. Second report on massive adaptation (M24), `https://www.translectures.eu//wp-content/uploads/2014/01/transLectures-D3.1.2-15Nov2013.pdf`
6. TLK: The transLectures-UPV Toolkit, `https://www.translectures.eu/tlk/`
7. Baum, L.E., Petrie, T., Soules, G., Weiss, N.: A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. The Annals of Mathematical Statistics 41(1), 164–171 (1970)
8. Dahl, G.E., Yu, D., Deng, L., Acero, A.: Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. IEEE Transactions on Audio, Speech, and Language Processing 20(1), 30–42 (2012)
9. Digalakis, V., Rtischev, D., Neumeyer, L., Sa, E.: Speaker Adaptation Using Constrained Estimation of Gaussian Mixtures. IEEE Transactions on Speech and Audio Processing 3, 357–366 (1995)
10. Huang, J.T., Li, J., Yu, D., Deng, L., Gong, Y.: Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In: Proc. of ICASSP (2013)
11. Munteanu, C., Baecker, R., Penn, G., Toms, E., James, D.: The Effect of Speech Recognition Accuracy Rates on the Usefulness and Usability of Webcast Archives. In: Proc. of CHI. pp. 493–502 (2006)
12. Ney, H., Ortmanns, S.: Progress in dynamic programming search for LVCSR. Proceedings of the IEEE 88(8), 1224–1240 (2000)
13. Ortmanns, S., Ney, H., Eiden, A.: Language-model look-ahead for large vocabulary speech recognition. In: Proc. of ICSLP. vol. 4, pp. 2095–2098 (1996)
14. Ortmanns, S., Ney, H., Aubert, X.: A word graph algorithm for large vocabulary continuous speech recognition. Computer Speech and Language 11(1), 43 – 72 (1997)
15. Povey, D., et al.: The Kaldi Speech Recognition Toolkit. In: Proc. of ASRU (2011)
16. Rumelhart, D., Hintont, G., Williams, R.: Learning representations by back-propagating errors. Nature 323(6088), 533–536 (1986)
17. Rybach, D., et al.: The RWTH Aachen University Open Source Speech Recognition System. In: In Proc. Interspeech. pp. 2111–2114 (2009)
18. Seide, F., Li, G., Chen, X., Yu, D.: Feature engineering in Context-Dependent Deep Neural Networks for conversational speech transcription. In: Proc. of ASRU. pp. 24–29 (2011)
19. Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory 13(2), 260–269 (1967)
20. Young, S., et al.: The HTK Book. Cambridge University Engineering Department (1995)
21. Young, S.J., Odell, J.J., Woodland, P.C.: Tree-based state tying for high accuracy acoustic modelling. In: Proc. of HLT. pp. 307–312 (1994)