Semi-automatic acoustic model generation from large unsynchronized audio and text chunks

Michele Alessandrini, Giorgio Biagetti, Alessandro Curzi, Claudio Turchetti

Dipartimento di Ingegneria Biomedica, Elettronica, e Telecomunicazioni (DIBET),
Università Politecnica delle Marche, Ancona, Italy

m.alessandrini@univpm.it, g.biagetti@univpm.it, a.curzi@univpm.it, c.turchetti@univpm.it

Abstract

In this paper an effective technique to train an acoustic model from large and unsynchronized audio and text chunks is presented. Given such a speech corpus, an algorithm to automatically segment each chunk into smaller fragments and to synchronize those to the corresponding text is defined. These smaller fragments are more suitable to be used in standard model training algorithms for usage in automatic speech recognition systems. The proposed approach is particularly suitable to bootstrap language models without relying neither on specialized training material nor borrowing from models trained for other similar languages. Extensive experimentation using the CMU Sphinx 4 recognizer and the SphinxTrain model generator in a setting designed for large-vocabulary continuous speech recognition shows the effectiveness of the approach.

Index Terms: speech recognition, acoustic model, model bootstrapping, automatic segmentation.

1. Introduction

Automatic speech recognition (ASR) engines are increasingly being perceived as a commodity feature that more and more people, speaking different languages and/or dialects, expect to be embedded in many consumer electronics devices or interactive systems. Since ASR systems need, among other things, good quality acoustic and language models to be able to operate satisfactorily, porting them to new languages, for which a suitable corpus of speech data to train the model might not be readily available, is a challenging task.

To tackle the lack of a large corpus, different approaches have already been proposed. It is possible, for instance, to use a model initially trained for another language as a starting point [1]. Although promising, this technique might raise a few concerns about the possibility of a residual contamination of the model by the original language.

In our work, we propose a technique that only uses material in the target language, even recorded for purposes other than making models. The typical use case involves collecting several, possibly long, audiobooks in the target language, accompanied by their textual transcription. The main roadblock to the use of this material for model generation is that the audio and text are usually unsynchronized, making them unsuitable as a direct input to a standard acoustic model generator.

In fact, using long chunks of audio and text to train a model is computationally demanding. Various efforts have been done to reduce this complexity, but they require that the audio be already labelled [2], or can only deal with one aspect of the computability problem, such as memory [3].

Since manually synchronizing large text and audio corpora is in principle a feasible, but tedious job, the main outcome of our work is a technique that, starting from very few and coarsely segmented audio and text blocks, algorithmically segments and synchronizes the whole corpus. Indeed, there is also a very useful byproduct of the segmentation process, that is a high-quality acoustic model of the target language, which can then be further processed by dedicated algorithms to obtain more refined, possibly speaker-independent, models.

2. Automatic segmentation algorithm

We assume that the material to be used to train the new model is available in the form of N large chunks of audio, with a textual transcription corresponding to each chunk. In the proposed scenario, a chunk can be as large as a whole audiobook chapter, or even more.

2.1. Acoustic model training

Traditionally, creating an acoustic model means training hidden Markov models (HMMs) of the speech features, which is done by means of a maximum likelihood estimation of the parameter vector λ that defines the HMMs themselves. The maximum-likelihood estimation is performed with a specialized expectation-maximization (EM) procedure, i.e. the Baum-Welch algorithm [4].

Given a vector of observed features $f = (f_1, \ldots, f_T)$, let $s = (s_1, \ldots, s_S)$ be the hidden state sequence and $P(f, s|\lambda)$ the incomplete-data likelihood function. The procedure is then composed of two phases: E) An objective function $Q(\lambda, \lambda')$ must be determined from the current set of parameters λ', by computing the likelihood conditional expectation

$$Q(\lambda, \lambda') = \sum_{s \in \mathcal{S}} \log P(f, s|\lambda)\, P(f, s|\lambda') \qquad (1)$$

where $\mathcal{S}$ is the set of all the possible state sequences. During this phase, a forward-backward procedure is employed to compute the *a posteriori* probabilities of all hidden state variables and to implicitly perform an automatic phone alignment. M) The objective function is maximized, yielding a new current set of parameters λ. The two phases are then iterated until a suitable stop condition is met.

Unfortunately, this training procedure is very likely to fail when fed with large chunks of audio, mainly because of its computational complexity. For instance, the forward phase, for a sequence of length T, has a time complexity of $O(S^2\, T)$ and a memory complexity of $O(S\, T)$. This means that, for training ASR systems, the worst-case complexities are typically $O(T^3)$ and $O(T^2)$ respectively [2].

fragments [#]	average duration [words]	memory usage [MB]	total CPU time [s]
100	60	31	234
50	120	92	304
20	300	235	381
10	600	295	363
5	1200	—	fail
2	3000	>8000	fail
1	6000	>8000	fail

Table 1: *Resource usage of the SphinxTrain model generation tool vs. audio chunk length. Machine used: Intel Xeon X3460 @ 2.8 GHz, 8 GiB RAM.*

Hence, feeding large chunks of audio to a conventional acoustic model training software may result in prohibitively large time and, even more importantly in today's computing environments, memory requirements. For instance, Tab. 1 reports empirical memory consumption and CPU usage versus chunk length for the SphinxTrain [5] model trainer, which employs the just described algorithm. Moreover, for large chunks the algorithm may also fail, either because of memory exhaustion (last two lines), or because of catastrophic errors in phone alignment (third from last line), which prevented the convergence of the model altogether.

Since the ideal chunk length would be limited to a few utterances, the need arises to segment the available audio and text chunks into shorter, but matching, fragments.

2.2. The automatic segmentation problem

While there can be simple means to separately segment audio and text, such as silence detection and punctuation analysis, the two must be synchronized. To do so, the following procedure can be successfully employed.

Of course, for good results the textual representation of the audio material must match what is spoken as best as possible. Abbreviations, digits, and other ideographic symbols should all be replaced with the corresponding plain words, so that an automatic phonetic transcription of the text can then be possible. If the text writer adhered to a predefined style, it is often easy to mostly automate this procedure.

The segmentation algorithm also needs a bootstrapping task:

- Manually segment chunk 1 into a few "largish" fragments. The only constraint is that they be processable by the training software. It will likely use quite some computing resources and produce a low-quality model, but it is a one-time job and the result should be enough for now.

Then, for each chunk n from 2 to N:

a) Train an acoustic model using segmented material from chunks $1, \ldots, n-1$. We'll call this "acoustic model $n-1$".

b) Train a language model using the textual material from chunks $1, \ldots, n$. We'll call this "language model n".

c) Use the acoustic model $n-1$ together with the language model n to recognize audio chunk n. The ensuing recognized text will likely be highly erroneous, but it is accompanied by timestamps for each word, which are what we are looking for.

d) Automatically segment audio and text chunk n, using slicing points derived from the timestamps obtained in c), and aligned to the text as will be shown next.

Finally, step a) can be repeated one last time to obtain the acoustic model N. The whole procedure is sketched in Fig. 1.

The number of segments into which step d) will be able to fragment the original audio is highly dependent upon the quality of the recognition performed during step c), so it can be expected to improve as more material is added and progression is made towards other chunks. It may be worthy to note that we use a language model that comprises all the text up to and including the chunk being recognized, so as to maximize the probability of correct recognition of the chunk in question.

2.3. Automatic slicing point detection

The core of the algorithm lies in step d), which must be able to find suitable slicing points for both audio and text. Using the rough and possibly highly-inaccurate time-aligned transcription of the audio made with the recognizer, it is possible to compare it with the known-good text to select matching points where it is possible to coherently cut both audio and text.

The algorithm proceeds as follows.

- the original text is compared to the recognized text, using any of the well-established algorithms for approximate string matching [6], so as to obtain matching intervals of contiguous words, which have a very high probability of having been correctly recognized. These matching intervals allow the association of timestamps, coming from the matching recognized text, to intervals of words in the text to be segmented.
- a map of the pauses is built. Pauses are located in the text using punctuation analysis, and in the audio by silence detection algorithms. This is done because we only want to cut audio chunks in fragments that start and end with a pause, so as not to alter the prosody and affect the recognition of coarticulated phones at word boundaries.
- a set of candidate fragments is built using the result of the comparison between the recognized and original text. Each contiguous span of matching words is considered a candidate fragment, which will be subjected to further processing later on. Intervals between those are also considered candidate fragments, but since the recognition failed there, we do not have timestamps inside of these fragments, so they cannot be subjected to further processing and must be kept whole, as only the starting and ending times are known (from the surrounding correctly recognized words).
- candidates that have associated timestamps are broken down at the pauses previously mapped, as the timestamps allow the pauses detected in the audio to also be located in the text and vice versa.
- because of the presence of unrecognized segments, candidates may not end on a pause, so the set of fragments need to be polished. Consecutive short intervals are merged together (e.g., when they are less than a predefined number of words long), and if one end does not align with a pause, the fragment is enlarged to cover the adjoining interval, until one is found that properly ends with a pause.

With this procedure, it is possible to coherently fragment both audio and text, with a granularity that can be defined as

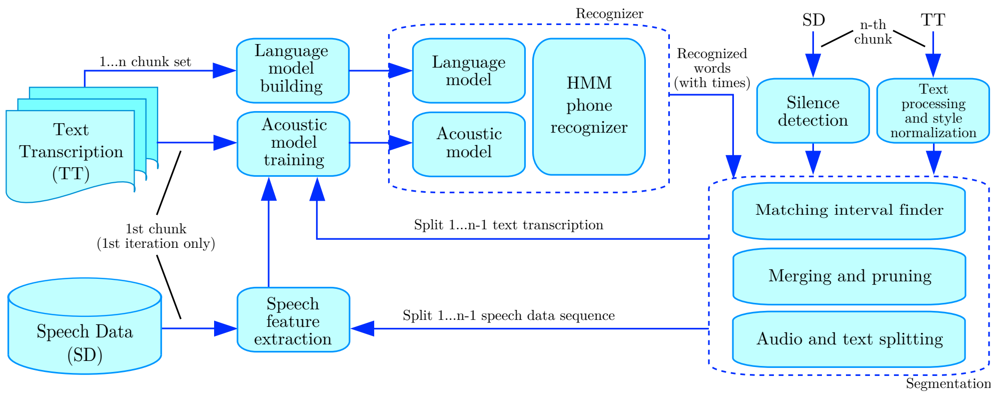

Figure 1: *The proposed procedure for chunk segmentation and alignment.*

desired within the recognized intervals, while unrecognized intervals must be preserved as they are. Nevertheless, as the accuracy of the model improves as more training material is added, unrecognized intervals will become shorter and rarer, so that fragmentation will soon only be limited by the natural occurrences of pauses in the language that is being processed.

2.4. Model refinement

When the automatic segmentation algorithm is applied to all the available audio chunks, we also implicitly trained the acoustic model $N-1$. Acoustic model N can also be easily generated, and, at least theoretically, it should be the best of the models generated so far.

It should then be possible to use this last model to improve the segmentation on the first chunks, which were likely to suffer from the lack of a good model and have thus been split into overly large fragments.

In this way it should be possible to improve the segmentation, and even obtain better models if the errors occurred during the first iteration were large enough to cause a misalignment that hampered the training.

In principle, this iteration could be performed many times. Nevertheless, in our experience, if the chunks are large enough, only the first few are likely to suffer, so that doing more than two passes does not produce measurable improvements. More on this will be shown in the next section.

3. Experimental results

To test the effectiveness of the proposed technique, we performed an extensive experimentation using the CMU Sphinx 4 ASR system, coupled with an advanced ETSI ES 202 050-compliant feature extractor. The main parameters used are listed in Tab. 2. As for the test material, we selected one quite long audiobook in Italian, whose audio and text transcriptions are readily and freely available.

The first experiment made use of the first chapter only, which is almost 6000 words long (about 2600 s), i.e., it is clearly way too short to train a large-vocabulary acoustic model from scratch, but too large to be processed as a single chunk by a traditional acoustic model trainer.

The first chapter was then manually split in just 10 chunks, and these fed to the proposed algorithm. Despite the obvious shortcomings with such little material, the algorithm performed almost flawlessly. Fig. 2 shows the statistical distribution of

training material:
- language: Italian
- book title: “I promessi sposi”, by Alessandro Manzoni
- duration: $\sim$22 h
- source: Liber Liber (`http://www.liberliber.it/`)

frontend:
- standard: ETSI ES 202 050 compliant
- audio: 8 kS/s, mono, 16 bit
- dithering: yes
- features: 13 MFCC + Δ + $\Delta\Delta$

language model:
- 3-gram statistical model

acoustic model:
- states per HMM: 3 + final state
- gaussians per state: 8
- tied states: 1000

auto-split:
- min. fragment duration: 4 words or 1 s

Table 2: *Parameters used in the experiments.*

the duration of the generated fragments: whiskers are the minimum and maximum values, boxes span the range from the first (Q_1) to the third (Q_3) quartile, and the dash in the box is the median (the second quartile, Q_2). As can be seen, the first chunk was used as a single entity to bootstrap the system. Fig. 3 shows the recognition accuracy in the decoding of each chunk. As could be expected, decoding of chunk #2 with an acoustic model trained with only about 500 words from chunk #1 failed almost entirely, so #2 could not be split. Decoding of #3, which could count on a slightly better acoustic model, went a little better, and it allowed the chunk to be split. Henceforward the model continued to improve, as well as the resulting segmentation.

Figs. 4 and 5 report the same graphs for a more extensive test done throughout the whole audiobook. This time the 10 fragments manually cut from the first chapter were used to bootstrap the system, which, with much longer chunks and audio material to train upon, reached convergence[1] almost immediately.

[1]The accuracy data reported in the figures is of course only meaningful to evaluate convergence, as at least for the the language model the test set overlapped the training set.

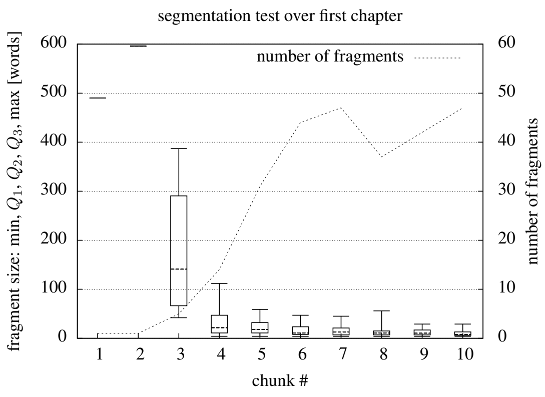

Figure 2: *Statistical distribution of the duration, in words, of the automatically segmented fragments for each chunk. Each chunk was approximately 1/10th of the first chapter of the audiobook, and the system was bootstrapped without any further manual segmentation.*

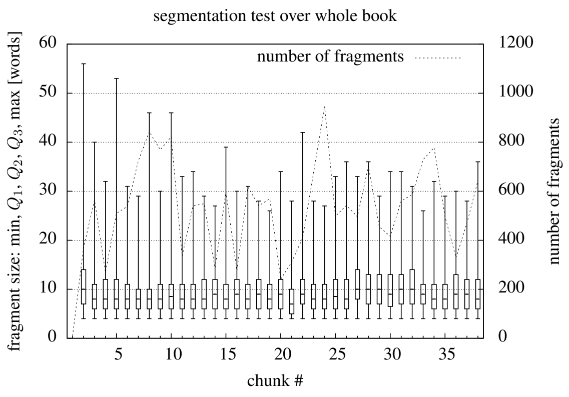

Figure 4: *Statistical distribution of the duration, in words, of the automatically segmented fragments for each chunk. Each chunk was one full chapter of the audiobook and the system was bootstrapped with 10 manually segmented fragments from chunk #1 (not shown because out of scale).*

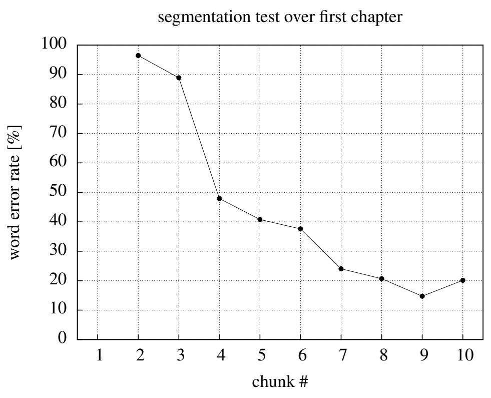

Figure 3: *Recognition accuracy achieved in the decoding of each chunk reported in Fig. 2.*

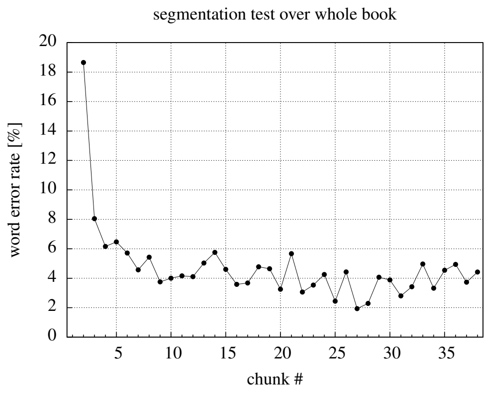

Figure 5: *Recognition accuracy achieved in the decoding of each chunk reported in Fig. 4.*

4. Conclusions

This paper proposes a technique for an almost automatic generation of acoustic models from unsynchronized audio and text material. The algorithm is particularly suitable to train models from non-specialized recorded material available in very large chunks, such as an audiobook collection.

By fully exploiting its iterative segmentation and synchronization capabilities, down to a single utterance level, the system can achieve a good recognition performance already after very few iterations. It only feeds the Viterbi phone aligner employed in the model trainer with well-cut utterance-level aligned audio fragments, much shorter than the original chunks, so as to attain the best possible training conditions. It also makes the best use of the linguistic information that can be derived from the corpus during the recognition step, by knowing in advance the text that should be recognized. As already cited, since the worst-case training by means of EM has a time complexity of $O(T^3)$ and a memory complexity of $O(T^2)$, the proposed technique, besides giving a significant improvement in the accuracy of the produced model built with non-specialized material, also allows a significant speed-up of the training by using smaller fragments.

5. References

[1] N. Chatzichrisafis, V. Diakoloukas, V. Digalakis, and C. Harizakis, "Gaussian mixture clustering and language adaptation for the development of a new language speech recognition system," *IEEE Trans. Audio, Speech, Language Process.*, vol. 15, no. 3, pp. 928–938, Mar. 2007.

[2] D. Huggins-Daines and A. I. Rudnicky, "A constrained Baum-Welch algorithm for improved phoneme segmentation and efficient training," in *Proc. of 9th International Conference on Spoken Language Processing (Interspeech 2006 - ICSLP)*, Pittsburgh, PA, Sep. 2006, pp. 1205–1208.

[3] W. Khreich, E. Granger, A. Miri, and R. Sabourin, "On the memory complexity of the forward-backward algorithm," *Pattern Recogn. Lett.*, vol. 31, pp. 91–99, Jan. 2010.

[4] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, Feb. 1970.

[5] M. Seltzer and R. Singh. Instructions for using the Sphinx3 trainer. [Online]. Available: http://www.speech.cs.cmu.edu/sphinxman/fr4.html

[6] E. W. Myers, "An O(ND) difference algorithm and its variations," *Algorithmica*, vol. 1, pp. 251–266, 1986.