From segmentation bootstrapping to transcription-to-word conversion

Uwe D. Reichel

Institute of Phonetics and Speech Processing, University of Munich

reichelu@phonetik.uni-muenchen.de

Abstract

The mapping of a raw phonetic transcription to an orthographic word sequence is carried out in three steps: First, a syllable segmentation of the transcription is bootstrapped, based on unsupervised subtractive learning. Then, the syllables are grouped to word entities guided by non-linguistic distributional properties. Finally, the phonetic word segmentations are mapped onto entries of a canonic pronunciation dictionary by means of a co-occurrence based aligner. For syllable segmentation accuracies between 89 and 96% are obtained, and for word segmentation accuracies between 92 and 98%. The transcription to word conversion performance amounts 77%.

Index Terms: segmentation, bootstrap, distributional learning, subtractive learning, alignment

1. Introduction

Phonetic or linguistic examinations of raw transcriptions usually require a segmentation of these transcriptions into syllables and/or words. Numerous studies about human segmentation of connected speech have revealed strategies like using boundary cues and subtractive learning [1]. Among the identified boundary cues are stress (see [2] for a review), segment transition probabilities [3] subsuming phonotactic constraints [4], and phonetic variation related to coarticulation [5]. In computational approaches segmentation is addressed in terms of encoding optimization like in the Minimum description length algorithm of [6] or by utilizing information theoretic findings of low segment information due to high occurrence frequencies and high entropy values across segment boundaries [7]. [8] carries out an iterative dynamic programming segmentation, augmenting a segment lexicon and a frequency table from unparsed utterance parts which in turn are used for re-segmentation of these parts. [9] accompany this algorithm by phonotactic n-gram models of word boundaries.

The algorithm introduced here incorporates subtractive learning and probabilistic aspects mentioned above. In section 2 the grouping of phonemes to syllables is addressed, and in section 3 the grouping of syllables to words. Section 4 introduces an approach to map the word-segmented transcription onto a sequence of lexical entries.

2. Syllable Segmentation

2.1. Data

Parts of the Kiel Corpus of Spontaneous Speech [10] were taken for this study containing transcribed spontaneous German dialogue speech and corresponding canonic transcriptions in German SAMPA. The data comprised 51 K words, the canonic transcription contained 229 K phones and 90 K syllables the spontaneous speech transcription 201 K phones and 87 K syllables.

2.2. Learning Algorithm

Syllable segmentation is carried out by acquiring knowledge about possible syllable onsets and codas to find the correct split point between syllable nuclei. The bootstrap learner operates in three steps: (1) initialization of syllable onset and coda counters for transcription substrings, (2) iterative increment of these counters by means of subtractive learning, and (3) finalization by deriving the sets of onsets and codas from the counter values. In addition to this iterative learning procedure, phoneme bigram probabilities are stored for fallback purposes.

In application, the optimal split point is derived from the stored onset and coda frequencies or by fallback from the phoneme bigram probabilities.

2.2.1. Learner prerequisites

The following assumption about the bootstrap learner are made: it is able to distinguish between speech and pause, it can detect syllable nuclei, and it can make use of the distributional properties of the input data.

The learner is conservative by constraining its onset and coda assignments to frequency thresholds. This is required since wrong constituent classifications would spread exponentially during the iteration phase. The threshold to be exceeded is set to the logarithm of the number of nuclei and pauses, which is 4 for our data. This definition is arbitrary but fulfills the purpose to link the threshold to the size of the data.

2.2.2. Notations

In the following pseudocode passages [P] stands for speech pause, [N] for syllable nucleus, S, X, Y are transcription substrings including the empty string ϵ, COD and ONS refer to the set of codas and onsets, $n_{cod}(\mathbf{X})$ and $n_{ons}(\mathbf{Y})$ are counters for X and Y occurring as a coda or onset respectively, while $m_{cod}(\mathbf{X}|\mathbf{Y})$ and $m_{ons}(\mathbf{Y}|\mathbf{X})$ are conditional counters, saying that $m_{cod}(\mathbf{X}|\mathbf{Y})$ is added to $n_{cod}(\mathbf{X})$ if Y has been identified as an onset (analogously for $m_{ons}(\mathbf{Y}|\mathbf{X})$). b is the counter threshold to be exceeded.

2.2.3. Initialization of the constituent counters

In the initialization step, seed syllable constituents are extracted from environments trivially to parse. These trivial cases are given by transcription substrings occurring between syllable nuclei and pauses.

[N] **X** [P] $\longrightarrow n_{cod}(\mathbf{X})$++
[P] **Y** [N] $\longrightarrow n_{ons}(\mathbf{Y})$++

Examples

- [t r **a** k **t** <P>] $\longrightarrow n_{cod}(\mathbf{k\ t})$++
- [<P> **t r a** k t] $\longrightarrow n_{ons}(\mathbf{t\ r})$++

2.2.4. Iterative counter increment

The onset and coda counters are iteratively updated using subtractive learning, that is, given an inter-nucleus sequence $S = XY$, then X is considered as a potential coda in case Y has already been established as a reliable onset (and vice versa). Reliability is derived from the constituent counters, i.e. Y is treated as a reliable onset, if the onset counter $n_{ons}(\mathbf{Y})$ is already greater than the threshold b.

First iteration step: undivided inter-nucleus sequences

In the first iteration step already initialized counters of undivided inter-nucleus sequences are updated together with the complementary counter of the remainder empty sequence.

data := $\{[N_1]\ S_1\ [N_2], [N_2]\ S_2\ [N_3], \dots\}$

foreach $[N]$ **S** $[N] \in$ *data*
  if $n_{ons}(\mathbf{S}) \wedge \neg\, n_{cod}(\mathbf{S})$
    $n_{ons}(\mathbf{S})$++
    $n_{cod}(\epsilon)$++

  if $n_{cod}(\mathbf{S}) \wedge \neg\, n_{ons}(\mathbf{S})$
    $n_{cod}(\mathbf{S})$++
    $n_{ons}(\epsilon)$++
end

Example

- [**a:6 S t r a:**]:
- **if S t r** initialized (only) as onset
- **then** onset count increment for **S t r**: $n_{ons}(\mathbf{S\ t\ r})$++ and coda count increment for ϵ: $n_{cod}(\epsilon)$++

Further iterations steps: split inter-nucleus sequences

In the further iteration steps, an inter-nucleus sequence S for which exists only one possible segmentation with respect to subtractive learning into subsequences X and Y results in the following counter updates: if X is already established as a coda, i.e. $n_{cod}(\mathbf{X}) > b$, then the onset counter of Y $n_{ons}(\mathbf{Y})$ is incremented as well as the conditional coda counter of X $m_{cod}(\mathbf{X}|\mathbf{Y})$. The latter counter will be added to $n_{cod}(\mathbf{X})$ if Y turns out to be an onset in the end. After these updates, S is removed from the data.

repeat until no more update of n_{cod}(*) and n_{ons}(*)
  foreach $[N]$ **S** $[N] \in$ *data*
    S $\leftarrow$ **X Y**
    if $n_{cod}(\mathbf{X}) > b \wedge \neg\exists$ **X'** $\neq$ **X**: $n_{cod}(\mathbf{X'}) > b$
      $n_{ons}(\mathbf{Y})$++
      $m_{cod}(\mathbf{X}|\mathbf{Y})$++
      $data \leftarrow data - [N]\ \mathbf{S}\ [N]$

    if $n_{ons}(\mathbf{Y}) > b \wedge \neg\exists$ **Y'** $\neq$ **Y**: $n_{ons}(\mathbf{Y'}) > b$
      $n_{cod}(\mathbf{X})$++
      $m_{ons}(\mathbf{Y}|\mathbf{X})$++
      $data \leftarrow data - [N]\ \mathbf{S}\ [N]$
  end
end

Example

- [**I m t r a**]:
- **if m – t r** is the only possible segmentation, i.e. only $n_{ons}(\mathbf{t\ r}) > b$
- **then**
  1. coda count increment for **m**: $n_{cod}(\mathbf{m})$++
  2. dependent onset count increment for **t r**: $m_{ons}(\mathbf{t\ r}|\mathbf{m})$++, which will be added to $n_{ons}(\mathbf{t\ r})$ later on, given that **m** turns out as a coda in the end, i.e. if $n_{cod}(\mathbf{m})$ will finally exceed b
  3. remove the sequence from the data

2.2.5. Finalization

In the finalization step, the conditional constituent counters are added to the onset and coda counters in order to derive the final frequency values. All constituents occurring more often than b are stored together with their frequency information.

repeat until no more update of n_{cod}(*) and n_{ons}(*)
  foreach Y: $n_{ons}(\mathbf{Y}) > b$
    foreach X: $n_{cod}(\mathbf{X}) > 0$
      $n_{cod}(\mathbf{X}) \leftarrow n_{cod}(\mathbf{X}) + m_{cod}(\mathbf{X}|\mathbf{Y})$
      $m_{cod}(\mathbf{X}|\mathbf{Y}) \leftarrow 0$
    end
  end

  foreach X: $n_{cod}(\mathbf{X}) > b$
    foreach Y: $n_{ons}(\mathbf{Y}) > 0$
      $n_{ons}(\mathbf{Y}) \leftarrow n_{ons}(\mathbf{Y}) + m_{ons}(\mathbf{Y}|\mathbf{X})$
      $m_{ons}(\mathbf{Y}|\mathbf{X}) \leftarrow 0$
    end
  end
end

$ONS \leftarrow \{\mathbf{Y} : n_{ons}(\mathbf{Y}) > b\}$
$COD \leftarrow \{\mathbf{X} : n_{cod}(\mathbf{X}) > b\}$

2.2.6. Phoneme transition probabilities

Additionally to the constituent information extracted as described above, phoneme bigram probabilities are calculated in terms of maximum likelihood estimates. Transition probabilities are considered to be useful for syllable segmentation since they have been found to be significantly higher within than across syllables.

2.3. Application

Given an inter-nucleus sequence S the optimal split point $\hat{s}$ splitting S into the sub-sequences X and Y is the one which maximizes the sum $n_{cod}(\mathbf{X})$+$n_{ons}(\mathbf{Y})$. If no split into a reliable onset and coda can be found, the split point is set to the phoneme transition probability minimum.

given: [N] **S** [N]

$T \leftarrow [\epsilon\ \mathbf{S}\ \epsilon]$
$\hat{s} \leftarrow \arg\max_s \left[n_{ons}(t_{1\ldots s}) + n_{cod}(t_{s+1\ldots n})\right]$

if $\neg\exists\hat{s}$

$T \leftarrow$ [[N] **S** [N]]
$\hat{s} \leftarrow \arg\min_s \left[P(t_{s+1}|t_s)\right]$

Examples

- [**a n t r aI**]
  - **n – t r**: n_{cod}(**n**)+n_{ons}(**t r**) = 2288+21 = 2309
  - **n t – r**: n_{cod}(**n t**)+n_{ons}(**r**) = 172+42 = 214

  $\longrightarrow$ [**a n – t r aI**]
- [**I s C @**]
  - **s C**: no possible segmentation by COD and ONS
  - $P(s|I) = 0.16$
    $\mathbf{P(C|s) = 0.01}$
    $P(@|C) = 0.05$
  - transition probability minimum between [s] and [C]

  $\longrightarrow$ [**I s – C @**]

2.4. Results

2.4.1. Syllable constituent retrieval

Table 1 shows the evaluation results for syllable constituent retrieval for the entire data described in section 2.1.

Table 1: *Evaluation of syllable constituent retrieval.*

constituent	measure	canonic	spontaneous
onset	precision	100	92.86
	recall	51.90	24.07
	token coverage	92.22	87.27
coda	precision	100	100
	recall	50.00	29.37
	token coverage	95.66	87.92

The conservative behavior of the learner forced by the frequency threshold b results in high precision and low recall values. Since precision and recall refer to onset and coda *types*, low recall values are justifiable since only the rarely occurring types have been missed. This can be read from the high token coverage values.

2.4.2. Segmentation performance

For evaluation only non-trivial cases have been considered (no speech pauses, and no pause-adjacent phones). So the results shown in table 2 are to be interpreted as lower performance boundaries.

No significant performance difference on training and test data is observed (Wilcoxon rank sum tests, $-0.26 < z < 1.17$, $p > 0.25$) indicating a good generalization capability of the algorithm.

Table 2: *Evaluation of syllable segmentation (for non-trivial cases only). Means and standard deviations after 10-fold cross validation. Accuracy is given on the phoneme level, precision and recall for detected syllable boundaries.*

measure	data	canonic	spontaneous
accuracy	train	96.02 (0.18)	89.19 (0.17)
	test	95.89 (0.46)	89.15 (0.70)
precision	train	95.02 (0.21)	88.27 (0.21)
	test	94.86 (0.68)	88.23 (0.88)
recall	train	94.88 (0.24)	86.47 (0.21)
	test	94.71 (0.66)	86.42 (1.17)

3. Word Segmentation

In principle, for word segmentation the same subtractive learning approach as for the syllable segmentation could be employed, but since the number of word types by far exceeds the number of syllable constituent types, it is not expected to perform well. Therefore, word segmentation is carried out in a supervised learning framework by C4.5 decision trees [11] predicting for each syllable s_i whether it is in word-initial position, thus right-adjacent to a word boundary, or not.

3.1. Features

The C4.5 tree was trained on the following distributional and bootstrapping features:

- **Distributional:**
  - Transition probabilities $P(s_i|s_{i-1})$, $P(s_{i+1}|s_i)$, and $P(s_{i-1}|s_{i-2})$.
- **Bootstrapping-based:**
  - STANDALONE(s_i)
  - STANDALONE(s_{i-1})
  - $P([P]|s_{i-1})$
  - $P(s_i|[P])$

STANDALONE(*) means, that * occurs at least once in an inter-pausal position in the training data.

3.2. Results

As with syllable segmentation only non-trivial cases have been considered for the word segmentation evaluation ignoring speech pauses and pause-adjacent syllables.

The results are presented in table 3. As with syllable segmentation, no significant performance difference on training and test data is observed (Wilcoxon rank sum test, $-1.47 < z \leq 0$, $p > 0.15$) again indicating a good generalization capability of the presented method. All accuracies are significantly higher than the baseline performance of 64.19% achieved by the model "each syllable is a word" (Wilcoxon rank sum test, $z > 4.00$, $p < 0.001$).

4. Transcription to word conversion

Each word entity t of the transcription derived by the word segmentation is mapped to the appropriate lexical entry e of a canonical pronunciation dictionary L by finding the most similar canonical pronunciation $w(e)$. Similarity is expressed in

Table 3: *Evaluation of word segmentation (for non-trivial cases only). Means and standard deviations after 10-fold cross validation. Accuracy is given on the syllable level, precision and recall for detected word boundaries.*

measure	data	canonic	spontaneous
accuracy	train	97.85 (0.18)	95.54 (0.20)
	test	97.86 (0.62)	95.56 (1.05)
precision	train	97.69 (0.22)	95.99 (0.16)
	test	97.77 (0.55)	96.02 (0.71)
recall	train	98.99 (0.11)	97.41 (0.17)
	test	98.95 (0.42)	97.42 (0.87)

terms of the Levenshtein distance which is derived from the *PermA* sequence aligner [12].

$$\hat{e} \quad = \quad \arg\min_{e \in L}[\text{levdist}(w(e), t)]$$

4.1. PermA Alignment

The Levenshtein distance is defined as the minimum total edit cost to convert the sequence w (short for $w(e)$, the canonical transcription in the pronunciation dictionary) to the sequence t (the transcription of the entity derived from word segmentation). *PermA* uses the standard edit operations *substitution, deletion* and *insertion*. Their costs are defined in terms of conditional probabilities which are calculated by maximum likelihood estimation (i and j refer to character indices within the sequences w and t respectively):

- **substitution:**

$$c(w_i, t_j) = \begin{cases} 0 & : \text{ equal}(w_i, t_j) \\ 1 - P(t_j|w_i) & : \text{ else.} \end{cases}$$

- **deletion:** $c(w_i, \_\_) = 1 - P(\_\_|w_i)$
- **insertion:** $c(\_\_, t_j) = 1 - P(t_j|\_\_)$

When generating the cost function all (w_i, t_j) co-occurrence count increments are distributed within a triangular window of length 3 and area 1 centered on w_i. $(w_i, \_\_)$-counts are derived the same way from training instants where $|t| < |w|$. In these cases, sequence t is set to equal length as w by padding __–symbols. Counts are then incremented for all t-permutations with respect to possible __–symbol positions, and all increments are finally normalized to the number of permutations. $(\_\_, t_j)$-counts for the insertion costs are derived analogously. This way *PermA* allows for a uniform modeling of all editing costs.

4.2. Results

As shown in table 4 for the spontaneous speech transcriptions the mean word retrieval accuracy amounts 76.48%. The error is given by the sum of deletion and insertion rates, which were obtained from errors in syllable and word segmentation, and by the substitution rate obtained from errors of transcription to word mapping.

Table 4: *Word retrieval evaluation. Means and standard deviations after 10-fold cross validation.*

accuracy	76.48 (2.09)
substitution rate	18.63 (1.44)
deletion rate	1.95 (0.45)
insertion rate	2.94 (0.59)

5. Discussion

Syllable segmentation It has been shown that bootstrap learning with few and non-linguistic assumptions is capable to group phones to syllables. Not faced in this study is the issue of ambisyllabicity. In the evaluation data, ambisyllabic phones were attached to the second syllable.

Word segmentation It turned out, that word segmentation based solely on low-level non-linguistic features is quite successful. Given additional word stress information a further improvement can be expected as is pointed out for example in [13] who exploit this information in form of a one stress per word constraint. A problem not faced in this study is the treatment of clitics resulting in ambiword syllables as in [ha:p QIC $\longrightarrow$ ha:pC] (*have I*).

Transferability to human phonology inference As was pointed out, bootstrapping design and feature selection was to some extent inspired by findings of human word segmentation. Some other features which turned out to be useful here, could in turn be helpful to formulate hypotheses about human inference. In general, machine learning insights could be used as a starting point for psycholinguistic artificial language studies on aspects of human language acquisition.

6. Acknowledgments

The work of the author has been carried out within the CLARIN-D project [14] (BMBF-funded).

7. References

[1] D. Dahan and M. Brent, "On the discovery of novel wordlike units from utterances: An artificial-language study with implications for native-language acquisition," *J. Experimental Psychology: General*, vol. 128, pp. 165–185, 1999.

[2] A. Cutler, D. Dahan, and W. Donselaar, "Prosody in the comprehension of spoken language: A literature review," *Language and Speech*, vol. 40, pp. 141–202, 1997.

[3] J. Saffran, "Statistical language learning: Mechanisms and Constraints," *Current Directions in Psychological Science*, vol. 12, no. 4, pp. 110–114, 2003.

[4] P. Jusczyk, P. Luce, and J. Charles-Luce, "Infants' sensitivity to phonotactic patterns in the native language," *J. Memory and Language*, vol. 33, pp. 630–645, 1994.

[5] S. Mattys, L. White, and J. Melhorn, "Integration of Multiple Speech Segmentation Cues: A Hierarchical Framework," *J. Experimental Psychology: General*, vol. 134, no. 4, pp. 477–500, 2005.

[6] C. de Marcken, "The unsupervised acquisition of a lexicon from continuous speech," Technical Report AIM-1558, 1995.

[7] P. Cohen, N. Adams, and B. Heeringa, "Voting Experts: An unsupervised algorithm for segmenting sequences," *Journal of Intelligent Data Analysis*, vol. 11, no. 6, pp. 607–625, 2007.

[8] M. Brent, "An efficient, probabilistically sound algorithm for segmentation and word discovery," *Machine Learning*, vol. 34, no. 1–3, pp. 71–105, 1999.

[9] D. Blanchard and J. Heinz, "Improving word segmentation by simultaneously learning phonotactics," in *Proc. CoNLL*, Manchester, 2008, p. 6572.

[10] K. Kohler, "Labelled data bank of spoken standard German – the Kiel Corpus of Read/Spontaneous Speech," in *Proc. ICSLP*, 1996, pp. 1938–1941.

[11] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo: Morgan Kaufmann, 1993.

[12] U. Reichel, "Perma and Balloon: Tools for string alignment and text processing," in *Proc. Interspeech*, Portland, Oregon, 2012, p. paper no. 346.

[13] T. Gambell and C. Yang, "Mechanisms and constraints in word segmentation," Manuscript, 2005.

[14] "http://eu.clarin-d.de/index.php/en/," Clarin-D web page.