Behavior Research Methods (2020) 52:264–278
https://doi.org/10.3758/s13428-019-01223-3

WordSeg: Standardizing unsupervised word form segmentation from text

Mathieu Bernard[1,2] · Roland Thiolliere[1] · Amanda Saksida[3] · Georgia R. Loukatou[1] · Elin Larsen[1,2] · Mark Johnson[4] · Laia Fibla[1,5] · Emmanuel Dupoux[1,2] · Robert Daland[6] · Xuan Nga Cao[1,2] · Alejandrina Cristia[1]

Published online: 1 April 2019

Abstract

A basic task in first language acquisition likely involves discovering the boundaries between words or morphemes in input where these basic units are not overtly segmented. A number of unsupervised learning algorithms have been proposed in the last 20 years for these purposes, some of which have been implemented computationally, but whose results remain difficult to compare across papers. We created a tool that is *open source*, enables *reproducible results*, and encourages *cumulative science* in this domain. WordSeg has a modular architecture: It combines a set of corpora description routines, multiple algorithms varying in complexity and cognitive assumptions (including several that were not publicly available, or insufficiently documented), and a rich evaluation package. In the paper, we illustrate the use of this package by analyzing a corpus of child-directed speech in various ways, which further allows us to make recommendations for experimental design of follow-up work. Supplementary materials allow readers to reproduce every result in this paper, and detailed online instructions further enable them to go beyond what we have done. Moreover, the system can be installed within container software that ensures a stable and reliable environment. Finally, by virtue of its modular architecture and transparency, WordSeg can work as an open-source platform, to which other researchers can add their own segmentation algorithms.

Keywords Unsupervised word discovery · First language acquisition · Natural language processing · Cumulative science

Introduction

One of the key tasks facing the language learning infant involves finding the minimal recombinable units present in the input. Since there are no systematic silences between words or morphemes, learners may need to carve them out from the running speech, a process known as segmentation. To do this, they may use a few universal and unambiguous cues (such as lengthy pauses), as well as a host of probabilistic cues. The latter can be classified into sublexical (e.g., which sound sequences tend to be found at word edges, and seldom within words) and lexical (e.g., certain words are more likely to follow each other than expected by chance). A number of computational algorithms building on subsets of such cues have been proposed, and several have been implemented in a variety of computer languages and applied to corpora so as to model infants' word form discovery processes. Typically, these models take as input a text-based, phonological representation of the input. To mimic the word discovery process, known word or morpheme boundaries are removed, and the algorithm is applied to try to make decisions on where breaks may occur, which are then compared against the original (gold) boundaries.

These studies are informative for a host of learnability questions, such as to test the sheer feasibility of a proposed word segmentation solution (Gambell & Yang, 2005), to compare alternative algorithms (Goldwater, Griffiths, & Johnson, 2009; Pearl, Goldwater, & Steyvers, 2010), to see whether languages differ in their intrinsic segmentability (Fourtassi, Börschinger, Johnson, & Dupoux, 2013), or whether child-directed speech is intrinsically easier to

✉ Alejandrina Cristia
alecristia@gmail.com

1 LSCP, Département d'études Cognitives, ENS, EHESS, CNRS, PSL Research University, 29, rue d'Ulm, 75005, Paris, France

2 INRIA, Villers-lès-Nancy, France

3 Institute for Maternal and Child Health - IRCCS "Burlo Garofolo" - Trieste, Trieste, Italy

4 Macquarie University, Sydney, Australia

5 University of East Anglia, Norwich, UK

6 University of California, Los Angeles, Los Angeles, CA, USA

segment than adult-directed speech (Ludusan, Mazuka, Bernard, Cristia, & Dupoux, 2017). Additionally, there is emergent evidence suggesting that computational word segmentation results may also be relevant for infant psycholinguistics, by predicting the contents of infants' long-term vocabulary better than lexical status (Ngon et al., 2013) or pure frequency (Larsen, Cristia, & Dupoux, 2017). These results provide initial validation to the cognitive modeling approaches to word segmentation that have enjoyed a fair amount of attention for the last several decades (e.g., Brent & Cartwright, 1996; Gambell & Yang, 2005; Goldwater et al., 2009; Harris, 1955), as they reveal that the latter may be close enough to infants' segmentation to make predictions that can be validated via direct experimental or correlational tests. In this context, it becomes crucial for the field to standardize segmentation methodology, so as to better explore the phenomenon of segmentation and make empirically informed predictions for infant experimental work.

In this paper, we present WordSeg, a software package conceived to allow this field of research to do cumulative science. The last few decades have seen a surge of interest in open-science methods, where researchers' choices are rendered transparent, enabling others to replicate and extend results more easily. One could imagine this is even easier for computational modeling than, say, live experimentation, since typically modeling involves the creation of scripts which can be run time and again, are blind to the person executing them, and seem more context-independent than animals, and yet, recent articles continue to alert us on the unavailability of key research materials (including code) even of modeling work (Gundersen & Kjensmo, 2018). The first step towards cumulative science is thus to favor open source code, that is, code that is both available publicly and tagged for public re-use. However, this is not enough. Even if the source code is made publicly available, it is often not set up to run in some other machine or operating system; and it is not sufficiently documented that it can be launched by some other user in an informed fashion so as to reproduce the original results (Stodden, Seiler, & Ma, 2018). Thus, the second step towards cumulative science involves providing appropriate documentation as well as taking steps to make sure reproducibility can be achieved outside the native context. The final ingredient is to enable other researchers to directly build on previous work in a cumulative fashion.

With all of these considerations in mind, we created a tool that has a modular architecture (see Fig. 1), combining a set of corpora description routines, several algorithms varying in complexity and cognitive assumptions, and a rich evaluation package, all integrated into a seamless pipeline. We have made our package openly accessible, and complemented it with supplementary materials allowing readers to reproduce every result in the current paper, as well as detailed online instructions further enabling them to go beyond what we have done. With this, we meet the first desideratum. Additionally, the whole system can be installed using Docker, ensuring that the environment will be stable across operating systems (Hung, Kristiyanto, Lee, & Yeung, 2016)—a requirement for reproducibility. Finally, by virtue of its modular architecture (and by clearly restricting and documenting e.g., input and output formats), the suite can work as an open-source platform to which researchers can add their own segmentation algorithms. This allows algorithm developers to benchmark their results against previously available segmentation algorithms, and should greatly facilitate making their own segmentation algorithm public—thus fitting the last desideratum, cumulativity. We believe this approach is extremely novel in our field: We cannot name one tool in psycholinguistics (or in another subfield of psychology) that attempts to provide a framework for *every* researcher to integrate and test their own model against others'.

We see two main use cases. The first involves fellow modelers, who are developing alternative unsupervised word segmentation algorithms. As just mentioned, our package can serve as a common platform that standardizes input and evaluation, and provides a set of alternative algorithms against which developers can benchmark their own tool. Moreover, they can then profit from the effort that has gone into making this package widely deployable by simply adapting their tool to the WordSeg architecture and adding it as a new WordSeg module. The second set of users is linguists and other cognitive scientists interested in early language acquisition. This second group would not develop additional code, but rather make use of the standardized user interface to describe and analyze their child language corpora, or respond to specific scientific questions. For instance, a user may be curious about the ease of segmentation of social words (such as "mommy" and the baby's name) in different languages. This user could apply all segmentation algorithms, and then estimate with what frequency these words appear as such (i.e., are not obscured by under- or over-segmentation) in the segmented output. Such WordSeg uses are extremely straightforward for anyone who knows how to interact with a terminal (and for readers who do not, we recommend Software Carpentry's introduction http://swcarpentry.github.io/shell-novice/).

Previous computational modeling work

It is beyond the scope of the present article to provide a comprehensive review on computational models of infant word segmentation, and thus we refer interested readers to Daland (2009) for a fuller introduction to the basic issues

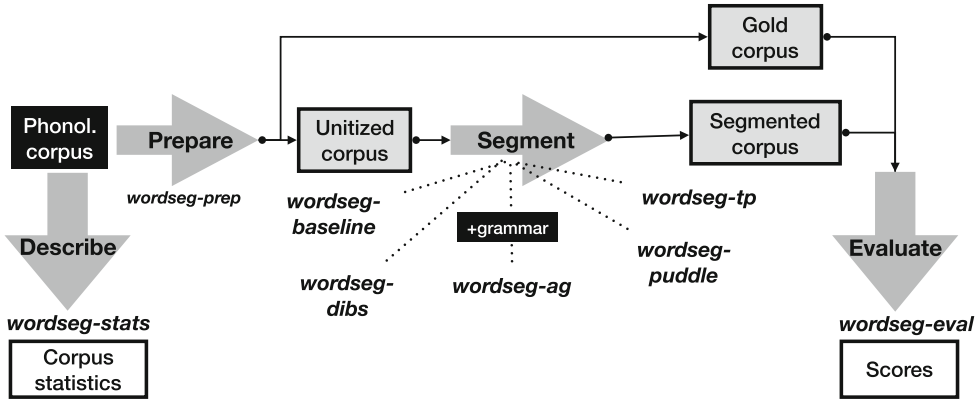

Fig. 1 Overview of the WordSeg suite. *Black boxes* represent input from the user; *other boxes* represent the output of a given stage; *arrows* represent the general description of procedures, most of which are implemented with a single command. The exception is the segmentation, where multiple segmentation processes are possible (parametric variation not shown)

surrounding computational models of infant word form segmentation, Brent (1999) for a historical classification of models, and Phillips (2015) for a recent literature review on the topic. It suffices here to state that this phenomenon has garnered considerable attention, but researchers have used varying methodologies in a way that compromises comparability. "Classes of algorithms currently represented in the package" lays out the main approaches that are currently represented in the package. Our package sought to also systematize "irrelevant" variation, as explained in "Keeping other aspects constant".

Classes of algorithms currently represented in the package

A systematic literature review[1] of 46 journal articles or theses that contained modeling results on word form segmentation published between 1993 and 2015 revealed that there are more postulated algorithms than papers, particularly when free parameters are taken into account. Thus, it was simply impossible to attempt to incorporate all previous algorithms. Our selection aimed at representing a few key dimensions of variation across open-source algorithms, and it was constrained by the availability of code and quality of the documentation.

One key distinction among included models pertains to whether they rely purely on local cues for word segmentation such as transitional probabilities between sounds or syllables. We will call this class sublexical. The lexical alternative involves aiming to reparse the input stream in terms of minimal recombinable units, or, put otherwise, building the lexicon that would be ideal to generate the corpus. This conceptual distinction does not prevent the existence of models that are hybrid. For instance, one of the models included in the suite is PUDDLE (Monaghan & Christiansen, 2010), which uses both lexical and phonotactic cues (see PUDDLE).

Additionally, some previous work has argued strongly for algorithms that process information **incrementally**, compared to others that do so in a **batch** mode (e.g., Monaghan & Christiansen, 2010). Although we believe that, to a certain extent, the dichotomy can be ill-posed, our sampling reflects both batch and incremental learners. We return to this topic in the discussion.

Two additional classes of models are not represented in the WordSeg suite. Unsupervised segmentation models that use raw speech as input and can fully parse a corpus are uncommon ithe speech technology literature (Versteegh et al., 2015), and not at all represented in work modeling infant word segmentation. The only exceptions we know of are closer to keyword discovery than full segmentation (e.g., Ludusan et al., 2014). Additionally, neural network type models are not represented either, mainly because this is an area of rapid technological development as neural networks are increasingly used for natural language processing in a wide range of applications including word segmentation (e.g., Cai et al., 2017).

Keeping other aspects constant

Most previous work uses only one or a very limited set of models, so that to decide which model performs better one often needs to compare performance across papers. However, our systematic review revealed a host of dimensions that varied across papers, and which prevent direct comparison across published work. Most saliently, it is not uncommon to observe extremely large variations

[1]The last author performed a search with the terms *infant "word segmentation" "computational model"* in scholar.google.com on August 10, 2015. The top 220 items were extracted automatically using Zotero. They were thereafter inspected manually, excluding as off-topic 174 on the basis of title, abstract, or full-text.

in the size of the corpus used as input (e.g., Saksida, Langus, & Nespor, 2017 based on around 10,000 words versus Daland & Pierrehumbert, 2011 drawing on 750,000 words). Moreover, previous work investigating the effect of input quantity among Adaptor Grammars found effects that were non-linear and dependent on the grammar itself (Börschinger, Demuth, & Johnson, 2012), making it all the more difficult to compare model performance across studies (see also Daland & Pierrehumbert, 2011; Johnson, Christophe, Dupoux, & Demuth, 2014; Monaghan & Christiansen, 2010, for further discussions of corpus size effects).

In early modeling work, it was not uncommon to use artificial corpora, and even in some current work the input consists of transcripts from broadcast speech or adult-directed speech (such as the Buckeye corpus, Pitt, Johnson, Hume, Kiesling, & Raymond, 2005). Using such input is no longer warranted, since corpora on the CHILDES (MacWhinney, 2009b) repository contains hundreds of transcriptions that are child-centered. These are likely to be ecologically valid, because recordings were gathered in children's natural environments, and often with a recording device worn by the child, thus capturing both child-directed and child-overheard speech available to the child.

For studies using CHILDES corpora, there are some sources of variation whose impact has not been sufficiently considered. Although it would seem that corpora are sure to be homogeneous if drawn from the CHILDES repository, different contributors actually use different criteria to define sentences. We have noticed that some corpus contributors are probably using a "breath group" or even "conversational turn" definition, since there may be 10–20 words in a given sentence. In contrast, others are probably using a syntactically or prosodically defined sentence, with overall shorter utterances, averaging three words in length. Additionally, researchers studying word segmentation often mix together various corpora from children of diverse ages without controlling for the possibility that the length and complexity of sentences and the lexical diversity in them varies as a function of the child's age. Despite the fact that they probably explain variation in segmentation performance, such characteristics are seldom thoroughly reported.

An additional source of variation relates to whether phones or syllables are the basic units at the phonological level. For example, Phillips and Pearl (2015) report better performance when the basic units were syllables, rather than phones, and argued in favor of syllables on plausibility grounds. Evaluating plausibility is not within the scope of the present paper. As for performance, Larsen and colleagues fully crossed basic unit against algorithm drawing from the sublexical, lexical, and hybrid types, and although in general F-scores were higher for syllables than phones, some exceptions remained (Larsen et al., 2017). Moreover, ranking across algorithms also depended on representational unit.

Finally, nearly every research paper on computational models of infant word segmentation contains arguments for and against the range of evaluation metrics that are typically used, prioritizing precision over recall, arguing that type statistics are more interesting than token statistics, or vice versa.

All of this variation seriously impedes direct comparison across published studies, and makes it difficult for researchers to decide how to set up their preprocessing and analysis pipelines to optimize comparability with previous work.

The WordSeg suite

The WordSeg suite allows the use of several algorithms drawn from previous literature in a controlled environment that standardizes input and allows users to easily report the full range of input and output statistics allowing cross-paper comparison. The overall process is represented in Fig. 1. Detailed instructions for use are available as online materials, which are updated as issues arise (https://wordseg.readthedocs.io). The version used for the current work is 0.7.1.[2]

Technical characteristics

The package is distributed from https://github.com/bootphon/wordseg, with a GPL-3.0 re-use license, from where it can be cloned or downloaded as a zip. In all cases, WordSeg requires several additional pieces of software (e.g., Python 3) to function. Installation instructions are provided covering how to download and install this ancillary software, as well as how to install WordSeg itself. The user can install WordSeg such that it will be available anywhere within the system, or only in a virtual environment via the use of Docker™ (Hung et al., 2016). WordSeg has native support for Linux and has been thoroughly tested on MacOS and Windows. Once the system is installed, users can use WordSeg as a command line interface from a Bash terminal or as a library from Python, with both series of commands described and exemplified in the online documentation https://wordseg.readthedocs.io. The code contained in WordSeg is mostly Python and C++, with variability being mainly due to the included segmentation algorithms.

[2] https://zenodo.org/record/1471532, https://github.com/bootphon/wordseg/releases/tag/v0.7.1

Input selection, cleaning, and phonologization

The suite does not directly support full pre-processing and phonologization of corpora, but we provide some pointers for users. For most researchers, the starting stage will be a CHILDES style `.cha` file, which contains comments as well as transcribed content. These first stages of cleaning will be dependent on the particular corpus because they vary somewhat across CHILDES corpora, and on the research question, since researchers may want to include or exclude specific speakers or utterances. Sample scripts we have used in the past can serve as inspiration (see the `/data/cha/` section of the package). Additionally, the WordSeg suite assumes that the input has already been phonemized and syllabified. For corpora in which this has not been done, we recommend readers look into the Phonemizer package (https://github.com/bootphon/phonemizer), which provides tools to convert text to phonemes. Another option is the WebMaus automatic segmentation tool (https://www.clarin-d.net/en/webmaus-basic-), which converts text files to phonemic transcriptions based on trained statistical models. For languages with a transparent orthography, hand-crafted rules can be used to derive the phonemic representation of words. Examples are provided in the `/data/phonorules/` section. Finally, users may want to employ a syllabification routine using the Maximize Onset Principle, a rule of thumb whereby a sequence of phones will be parsed such that the onset cluster will be as heavy as the language allows. For instance, the sequence /estra/ will be broken up into /es.tra/ in Spanish and /e.stra/ in English. We have adapted Perl code that does so from Phillips and Pearl (2015) and provide examples in the `/data/syllabification/` section and the `wordseg-syll` tool.

Preparing the input

For the rest of the processes, the package assumes that the input file contains only the transcribed utterances in phonological form, one utterance per line. Additionally, it is assumed that word boundaries and basic units are coded in the input text. The input text can have one or both of the following basic units: phones, syllables.

The `wordseg-prep` tool in the package allows users to convert the input text from the input form where syllable and word boundaries are tagged to the input to be provided to the models. This tool outputs a unitized version and a gold version of the text. A unitized version contains spaces between phones or syllables (as chosen by the user). The gold version only has spaces between words. The gold text will be used later to evaluate the output of segmentation.

Describing the corpus

The package also contains `wordseg-stats`, a tool to describe the input corpora. This description tool prints out the number of all of the following units: sentences or lines, single-word utterances, and number of tokens, types, and hapaxes (i.e., types with token frequency of exactly one) for words, syllables, and phones. Additionally, a measure of lexical diversity that controls for corpus length is extracted, namely a moving average type to token ratio similar to that available in the CHILDES tools (MacWhinney, 2009a), where a window of 10 word tokens are considered at a time, moved one token at a time. Finally, `wordseg-stats` returns a measure of entropy, i.e., the intrinsic ambiguity found in a text (see Fourtassi et al. 2013 for details). In a nutshell, given a set of utterances and the lexicon found in the gold segmentation, this measure of entropy assesses to what extent there are many versus few possible parses of the utterances (i.e., in a corpus with two sentences, “ice cream” and “icecream”, both utterances are ambiguous between “ice cream” and “icecream” segmentations).

Segmenting

All of the algorithms are called with variants of `wordseg-X`, where X is the short name for the algorithm (as shown on Table 1), together with the necessary parameters and ancillary files, both of which depend on the specific algorithm. The input for all algorithms is plain text as built by `wordseg-prep`, where only unit tokens (syllables or phonemes) are available and separated by single spaces (that is, the word boundaries have been removed), but some of them additionally require a training set or a configuration file. In the rest of this section, we provide a general description of each algorithm, parametrization and required files. We have not incorporated standardized measurements of memory requirements or length of processing, because these, we believe, could largely relate to details of implementation, which may not affect fundamentally the results found.

Baseline

Researchers might be interested in comparing baseline results to those of the word segmentation algorithms. The WordSeg package provides tools for word segmentation baselines based on the insertion of word boundaries in random positions in the text, explained for instance by Lignos (2012).

The Random Baseline assigns word boundaries with a probability parameter p specified by the researcher. By default, a random segmentation consists in adding word boundaries with $p = 0.5$ to each unit token. The

Table 1 Segmentation algorithm families currently included in WordSeg

Acronym	Class	Processing	Key units
baseline	sublexical	batch	units
dibs	sublexical	batch	unit bigrams
tp	sublexical	batch	unit bigrams
puddle	hybrid	incremental	unit n-grams, words
ag	lexical	batch	words

We say "families" because each has a set of parameters that allows further variation. Class indicates the main class the algorithm belongs to; Processing whether the input is processed in batch or incrementally; and Key units the crucial representations that the algorithm uses for segmentation

user can specify a random seed, to ensure reproducibility. Alternatively, the researcher can choose $p = 0$ to generate an "Utterance Baseline", considering each utterance as a single word; and $p = 1$, to insert all possible boundaries and treat each unit token (phones or syllables) as a word. The researcher can also inspect the statistics mentioned in "Describing the corpus" to calculate the true p of word boundaries given the basic unit (e.g., for a corpus unitized into syllables, $p = \frac{nw}{ns}$, where nw is number of words and ns is number of syllables). This number can then be provided by the user as the p parameter, in which case, this would be an Oracle Random Baseline (Lignos, 2012, "oracle" because it is given the true p by the researcher; random because it will insert the correct number of boundaries to match p, without knowing where they should occur).

Diphone-based segmenter (DiBS)

Daland's DiBS (short for Diphone-based segmentation, Daland and Pierrehumbert, 2011) uses phone bigram probabilities to decide whether a specific sequence is likely to span a word boundary (typically because the phone bigram is rare) or not. A DiBS model is any model which assigns, for each phrase-medial phone bigram, a value between 0 and 1 inclusive, representing the probability the model assigns that there is a word boundary between the two phones. In practice, these probabilities are mapped to hard decisions (break or no break).

Making these decisions requires knowing the chunk-initial and chunk-final probability of each phone, as well as all phone bigram probabilities; and additionally the probability of a sentence-medial word boundary. In our package, these four sets of probabilities are estimated from a training corpus also provided by the user, where word boundaries are marked. Please note we say chunk-initial and chunk-final because the precise chunk depends on the type of DiBS used, as explained in the next paragraph.

Three versions of DiBS are available. DiBS-gold is supervised in that "chunks" are the gold words. It is thus supposed to represent the optimal performance possible. DIBS-phrasal uses phrases (sentences) as chunks. Finally, DIBS-lexical uses as chunks the components of a seed lexicon provided by the user (who may want to input e.g. high frequency words, or words often said in isolation, or words known by young infants).

By default, the sentence-medial probability of word boundary is calculated in the same way for all three DiBS, and it is the actual gold probability (i.e., the number of words minus number of sentences, divided by the number of phones minus number of sentences). Via a parameter, users can also provide the algorithm with a probability of word boundary calculated in some other way they feel is more intuitive.

DiBS was initially designed with phones as basic units. However, for increased flexibility we have rendered it possible to use syllables as input.

Transitional probabilities (TP)

Like DiBS, the next family of algorithms attempts to distinguish between more or less *internally cohesive* phone/syllable sequences. In the implementation we have adopted (Saksida et al., 2017), transitional probabilities (TPs) are calculated in one of three ways:

- Forward TPs for XY are defined as the frequency of the sequence XY divided by the frequency of X;
- Backward TPs for XY are defined as the frequency of the sequence XY divided by the frequency of Y;
- Mutual information for XY is the log (base 2) of the frequency of the sequence XY divided by the product of frequency of X and that of Y

This direction parameter is crossed with another, defining a cut-off for how low TPs must be to signal a boundary, and which also has two settings. In the first, a boundary is posited when a relative dip in TP is found. That is, given the syllable or phone sequence WXYZ, there will be a boundary posited between X and Y if the TP for XY is lower than both that for WX and that for YZ. The second setting uses the average of the TP over the whole corpus as the threshold. Notice that both of these are unsupervised: Knowledge of word boundaries is not necessary to compute any of the parameters.

TP was initially designed with syllables as basic units, but has been adapted to accept either phones or syllables as input in this package.

PUDDLE

PUDDLE stands for Phonotactics from Utterance Determines Distributional Lexical Elements. This algorithm was proposed by Monaghan and Christiansen (2010); the original awk rendering (shared with us by Monaghan) was reimplemented in Python for this package. PUDDLE takes the opposite strategy of algorithms such as DiBS and TPs that focus on local events to posit breaks. In contrast, PUDDLE takes in whole utterances and tries to break them apart into relatively large chunks. The system has three long-term storage units: a "lexicon", a set of onset bigrams, and a set offset bigrams. At the beginning, all three are empty. The lexicon will be fed as a function of input utterances, and the bigrams will be fed by extracting onset and offset bigrams from the lexicon. The algorithm is incremental, as follows.

The model scans each utterance, one at a time and in the order of presentation, looking for a match between every possible sequence of units in the utterance and items in the lexicon. We can view this step as a search made by the learner as he tries to retrieve from memory a word to match it against the input. If, for a considered sequence of phones, a match is found, then the model checks whether the two units preceding and following the candidate match belong to the list of ending and beginning bigrams, respectively. Imagine a target utterance like "thisisacutebaby", unitized at the phone level; a lexicon containing the item "this"; possible bigrams thus being "th" for onsets and "is" for offsets. Although "this" is found in the target utterance, the utterance will not be split because the remainder, "**is**acutebaby", does not begin with a permissible onset. It should be born in mind that this constraint is crucial for the model to avoid over-segmentation: If not applied, the model will ultimately segment the corpus to the basic unit level (e.g., phones). If a substring match is not found, then the utterance is stored in the long-term lexicon as it is, and its onset and offset bigrams will be added to the relevant buffers. Thus, in the running example, the lexicon will end up containing two items, "this" and "thisisacutebaby"; the onset buffer will have the item "th" with a frequency of 2; and the offset buffer will have "is" and "by", each with a frequency of 1.

In our implementation of PUDDLE, we have rendered it more flexible by assuming that users may want to use syllables, rather than phones, as basic units. Additionally, users may want to set the length of the onset and offset n-grams. Some may prefer to use trigrams rather than bigrams; conversely, when syllables are the basic unit, it may be more sensible to use unigrams for permissible onsets and offsets.

Adaptor grammars (AG)

In the adaptor grammar framework (Goldwater et al., 2009; Johnson & Goldwater, 2009), parsing a corpus involves inferring the probabilities with which a set of rewrite rules (a "grammar") may have been used in the generation of that corpus. The WordSeg suite natively contains the capacity to generate one grammar, the most basic and universal one. Users can also create their own and/or change extant ones to fit the characteristics of the language they are studying (see the `/data/ag/` section of the package for more examples).

The simplest grammar, automatically generated with the call `wordseg-ag`, can be conceived as having one rewrite rule to the effect that "sentences are one or more words", one rewrite rule to the effect that "words are one or more basic units", and a set of rewrite rules that spell out basic units into all of the possible terminals. Imagine a simple language with only the sounds a and b, the abstract rules would then be:

- Sentence → Word (Word)+
- Word → Sound (Sound)+
- Sound → a
- Sound → b

A key aspect of adaptor grammar is that it can also generate subrules that are stocked and re-used. For instance, imagine "ba ba abab", a corpus in the above-mentioned simple language. As usual, we remove word boundaries, resulting in "babaabab" as the input to the system. A parse of that input using the rules above might create a stored subrule "Word → ba"; or even two of them, as the system allows homophones. The balance between creating such subrules and reusing them is governed by a Pitman-Yor process, which can be controlled by the user by setting additional parameters. For instance, one of these parameters, often called "concentration," determines whether subrules are inexpensive and thus many of them are created, or whether they are costly and therefore the system will prefer reusing rules and subrules rather than creating new ones.

The process of segmenting a corpus with this algorithm will in fact contain three distinct subprocesses. The first, as described above, is to parse a corpus given a set of rules and a set of generated subrules. This will be repeated a number of times ("sweeps"), as sometimes the parse will be uneconomical or plain wrong, and therefore the first and last sweeps in a given run will be pruned, and among the rest one in a few will be stored and the rest discarded.

The second subprocess involves applying the parses that were obtained in the first subprocess onto the corpus again,

which can be thought of as an actual segmentation process. Remember that in some parses of the "ba ba abab" corpus (inputted as "babaabab"), the subrule "Word → ba" might have been created 0, 1, or 2 times. Moreover, even if we ignore this source of variation, the subrules may be re-used or not, thus yielding multiple possible segmentations ("baba abab" with no subrule, "ba ba a ba b" with one "Word → ba" subrule or the same with 3 "Word → ba" subrules, etc.)

The third and final subprocess involves choosing among these alternative solutions. To this end, minimum Bayes risk is used to find the most common sample segmentations.

As this description shows, there are many potential free parameters, some that are conceptually crucial (concentration) and others that are closer to implementation (number of sweeps). By default, all of these parameters are set to values that were considered as reasonable for experiments (on English, Japanese, and French adult and child corpora Fourtassi et al., 2013; Johnson et al., 2014) running at the time the package started emerging, and that we thus thought would be a fair basis for other general users. The full list can be accessed by typing `wordseg-ag --help`. The following is a selection based on what is often reported in adaptor grammar papers:

- number of runs: 8
- number of sweeps per run: 2000
- number of sweeps that are pruned: 100 at the beginning and end, 9 in every 10 in between
- Pitman-Yor a parameter: 0.0001
- Pitman-Yor b parameter: 10000
- Rule probability (theta) is estimated using Dirichlet prior

Evaluation

An objective way to measure the performance of word segmentation algorithms is to compare the segmented corpus with the gold one, which corresponds to a perfect segmentation as would be done by a literate adult. This comparison can be done at different levels: word token, word type, and boundary. We provide two boundary scores, one counting utterance edges and the other not counting edges (since these will always be correct, by definition).

At a particular level, the evaluation looks at two different criteria: precision, the probability that a segmented boundary/token/type is correct; and recall, the probability a correct boundary/token/type has been segmented. Concretely, the precision P and recall R are calculated as follows:

$$P = \frac{\text{True positives}}{\text{True positives} + \text{False positives}} \quad (1)$$

$$R = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}} \quad (2)$$

The harmonic mean between precision and recall is computed to give the F1, which we will call F-score.

To take an example, imagine a corpus 'the dog bites the dog'; the segmented output is 'the dog bites thedog'. This will yield the following performance:

- token precision: 0.75, recall: 0.6, F-score: 0.67
- type precision: 0.75, recall: 1, F-score: 0.86
- boundary precision: 1, recall: 0.83, F-score: 0.91
- boundary no edge precision: 1, recall: 0.75, F-score: 0.86

Two additional evaluation outputs are provided at the user's request. First, users can obtain the Rand Index RI, which captures both true positives and negatives. It is calculated as follows:

$$RI = \frac{\text{True positives} + \text{True negatives}}{\text{True positives} + \text{True negatives} + \text{False positives} + \text{False negatives}} \quad (3)$$

Our evaluation actually provides the Adjusted Rand Index, where both numerator and denominator have been adjusted for chance agreement via resampling.

Second, some readers may be specifically interested in finding out which lexical items come to be correctly segmented, or else segmented incorrectly in one of these three ways: undersegmented (i.e., joined with a neighboring word); oversegmented (i.e., broken down into subparts); or plain mis-segmented. An optional parameter yields an evaluation summary file being returned, which contains all words in the gold corpus and the number of times with which they were found in each of those four groups.

One important consideration pertains to incremental algorithms, in which performance is changing throughout the corpus. To make their evaluation comparable to that of the others, we implemented a system of corpus folding, with a default of five folds (which can be parametrized by the user). For the first fold, a given algorithm is run in the whole corpus. Next, the final 20% of the corpus is moved to the onset of the corpus, and the algorithm is run again, such that this time the final 20% will in fact be the utterances that start at the 60% point in the corpus and end at the 80% point. This process repeats for the remaining three folds (40–60%, 20–40%, 0–20%). At this point, the final 20% of the corpora outputted in each of the five runs is concatenated in the right order, and the whole is evaluated. Please note that this is not an instance of cross-validation, since the models may continue learning over the last 20%.

Examples of use

This section has three goals. First and foremost, we aim to illustrate the package and show its flexibility. This example

allows users to have a benchmark when they themselves use the package. Since we expect that we and others will continue improving it, however, we recommend users check https://github.com/alecristia/wordseg-brm-analyses for an up-to-date version of these results as well as reproducible code. Second, we would like to inform researchers working on this domain on the impact of key methodological and conceptual decisions, such as what input and evaluation units are used. Finally, we try to assess the conditions in which performance is *stable and replicable*.

Crucially, we would like to make it clear from the start that the goal is not to compare performance across algorithms to find the best-performing one. Best performance against orthographic standards does not mean that the algorithm represents human performance, let alone infant performance. For instance, Larsen et al. (2017) found essentially a zero correlation between algorithms' F-scores against adult segmentation and the proportion of variance explained in infant word knowledge in an English sample. Thus, we consider that, at present, there is insufficient evidence to determine which algorithm best captures human (infant and adult) performance, and that they may all be valuable and informative to the computational modeler interested in the psychological phenomena surrounding word form segmentation. We want to provide the research community with an array of algorithms which, given the uncertainty regarding the information that infants and other learners incorporate, has a high likelihood of capturing at least some behaviors, or at the very least allows the researcher to focus on findings that are true *regardless of which algorithm is used as a proxy*.

Methods

Corpus

We used the Providence corpus (Börschinger et al., 2012; Demuth, Culbertson, & Alter, 2006), available from CHILDES (MacWhinney, 2009b) because it is commonly used and large enough to allow us to break it down into several subparts, and apply inferential statistics to assess whether certain factors truly explain significant proportions of variance. It contains transcriptions of recordings gathered from six American English-speaking children. Recordings started when children spoke at least four words according to parental report, which happened when they were around one year of age. About one hour of child-context interactions were recorded every 1–2 weeks until they were around 3 years of age. For the present study, we focus on the 74 transcripts (from five children) meeting the following desiderata:

– children were two years of age or younger
– there was only one adult present (which lowers the likelihood of including adult-directed speech)
– there were at least 300 utterances spoken by the adult

These transcripts were cleaned using custom bash scripts, which removed all comment lines and all sentences uttered by children. The resulting orthographic representations were phonologized using FESTIVAL (Taylor, Black, & Caley, 1998), which yields a representation including syllable boundaries. FESTIVAL uses a dictionary look-up system, complemented with grapheme-to-phoneme conversion rules for words not in the dictionary. The following is an example of the resulting "Tags" representation, which contains spaces to mark phone boundaries, ;s for syllable boundaries, and ;w for word boundaries.

1. Orthographic: you wanna sit with mommy
2. Tags: y uw ;s ;w w aa ;s n ax ;s ;w s ih t ;s ;w w ih dh ;s ;w m aa ;s m iy ;s ;w
3. Gold: yuw waanax siht wihdh maamiy

Processing with WordSeg

We generated the results for all the experiments below with a single Bash script (although we could have used Python

Table 2 Corpus characteristics of individual transcripts

Characteristics	Mean	SD
N phone tokens	11,463.22	3,414.43
N phone types	39.62	0.51
N syllable tokens	4,581.11	1,350.05
N syllable types	688.54	163.65
N words tokens	3,720.28	1,075.26
N words types	670.99	178.71
N word hapax	293.15	93.43
MATTR	0.89	0.04
Entropy	0.018	0.002
N SWU	102.81	55.33
N utts	700.27	200.38
	Derived metrics	
Prop. SWU	0.14	0.04
Prop. hapax	0.43	0.04
Avg. phones/word	3.08	0.08
Avg. syllables/word	1.23	0.03
Avg. words/utt	5.38	0.93

Tokens refers to unique instances, types to abstract units. *Hapax* stands for types that occur exactly once. *MATTR* stands for moving average type to token ratio, a TTR calculated over ten consecutive words so as to control for overall corpus size. *Entropy* is a measure of ambiguity in segmentability; a higher number means more ambiguity. *Utt*(s) stands for utterances; *SWU* for Single Word Utterance

instead). The following is a version of that Bash script, simplified for ease of inspection:

```
#!/bin/bash
# segment independent transcripts
FOLDER="/Providence/"
for tag in $FOLDER/*tags.txt; do
  # compute statistics on the unitized input text
  cat $tag | wordseg-stats --json > ${tag}_stats.json
  # prepare the input for segmentation and generate the gold text
  cat $tag | wordseg-prep --unit $unit --gold gold.txt > prep.txt
  # segment the prepared text with different algorithms
  # sublexical
  cat prep.txt | wordseg-baseline --probability 0.0 > ${tag}_seg.base00.txt
  cat prep.txt | wordseg-tp --threshold relative > ${tag}_seg.tprel.txt
  cat prep.txt | wordseg-dibs --type phrasal --unit $unit $tags > ${tag}_seg.dibs.txt
  # lexical
  cat prep.txt | wordseg-ag  > ${tag}_seg.AGu.txt
  # hybrid
  cat prep.txt | wordseg-puddle --window 2 > ${tag}_seg.puddle.txt
  # evaluate against the gold file
  for segmented in ${tag}_seg.*.txt; do
    algo=$(echo $segmented | sed 's/.*seg.//' | sed 's/.txt//')
    cat $segmented | wordseg-eval gold.txt > ${tag}_out.${algo}.txt
  done
done
```

The sample script above represents the following conceptual decisions:

- All algorithms are fed with a phone-unitized version of the corpus,
- The baseline is that which segments at utterance level only,
- The TP version uses the forward TP (default), with a relative threshold,
- The version of DiBS chosen in this example is the phrasal type, using the full corpus to extract phone bigram statistics,
- For AG, since no grammar was provided, the simple one mentioned above is automatically generated,
- For PUDDLE, we used bigrams (window of 2)

The full script can be retrieved from https://github.com/alecristia/wordseg-brm-analyses/blob/master/do_prov.sh. It actually feeds all algorithms with both phone- and syllable-unitized input, contains three baselines (cut at utterance boundary, at every unit boundary, and at half of them); and TP is run with both an absolute and a relative threshold.

Corpus statistics

Our call to wordseg-stats allowed us to describe the analyzed transcripts. Table 2 shows means and SDs of various corpus characteristics that are calculated by the statistics package, as well as some that can be derived from the output. The most important message we would like to convey here is that the standard deviations are quite high, particularly for sentence length. This is despite the fact that we focused on a single corpus, and further restricted inclusion to transcripts collected when children were younger than 2 years of age. Nonetheless, there are sizable changes in average sentence length, which may impact segmentation performance.

Effects of processing unit and algorithm

As mentioned above, we have analyzed each transcript within a subset of the Providence corpus separately, encoded in terms of phones and syllables, with a set of algorithms. In this section, we report on analyses aimed at assessing to what extent performance is affected by these two factors and their interaction. As shown in Supplementary Materials (https://github.com/alecristia/wordseg-brm-analyses/blob/master/supmat.pdf), all performance metrics are highly correlated with each other. Therefore, we focus here exclusively on token F-scores. Figure 2 shows that performance varies enormously as a function of algorithm and basic unit, with important interactions between the two. Next, we highlight

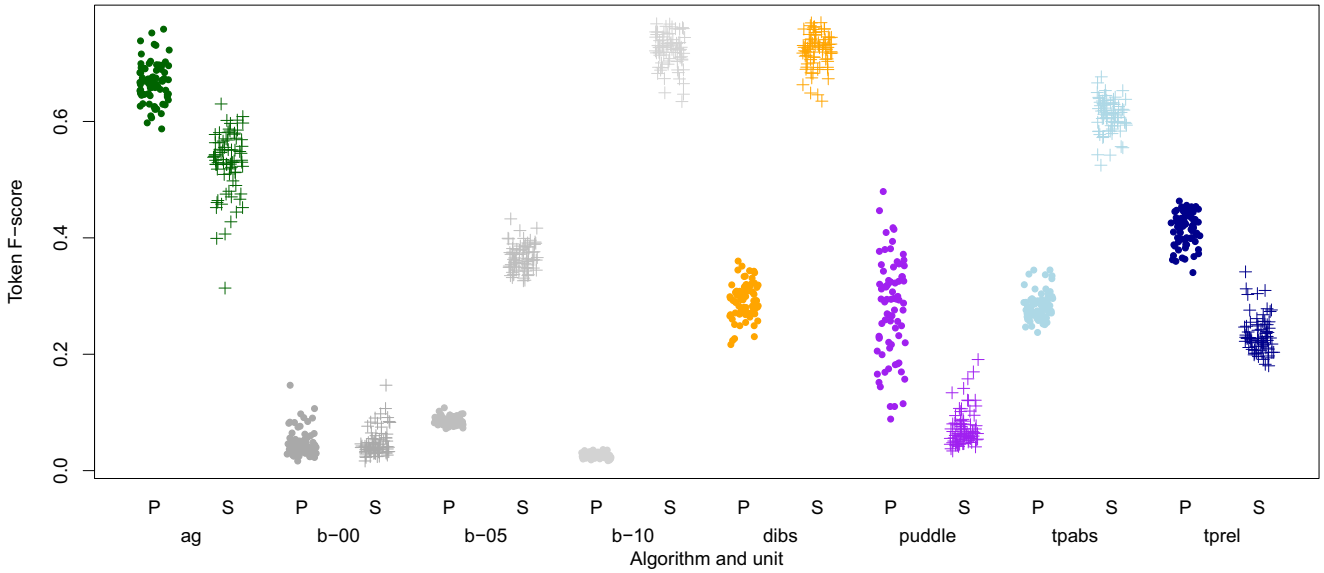

Fig. 2 Token F-scores as a function of unit and algorithm. Each point is the performance of a segmentation experiment on one of the 74 transcripts, using either phones (*circles*, *left*) or syllables (*crosses*, *right*) in combination with one of the eight algorithms (distinguished by position on the *x*-axis as well as color). In baselines, b-00 stands for $p = 0$; b-05 for $p = 0.5$; b-10 for $p = 1$

aspects of these results relevant to our three goals for "Examples of use".

The first result that may attract readers' attention is that performance varies greatly across algorithms. For instance, as has been discussed elsewhere (Gambell & Yang, 2005; Swingley, 2005), excellent scores can be achieved in English infant-directed speech samples like this one by simply segmenting every syllable, our baseline with $p = 1$ algorithm. Above and beyond the specific explanation, this observation highlights the usefulness of WordSeg's included baseline algorithms.

A second conclusion is that algorithm and unit interact. The reason is obvious for two cases: TP (absolute versus relative), and PUDDLE. For TP, performance is higher for syllable-as-unit than phone-as-unit when using an absolute threshold, but the opposite for a relative threshold. The reason is probably that the relative threshold algorithm requires at least 4 units in a row to be able to find a local dip (Gambell & Yang, 2005). Therefore, no boundary can be postulated in short sentences, with fewer than four syllables. In contrast, a boundary can be postulated in short sentences when these are represented in phones, because a local dip can be established when there are few syllables (provided these contain at least four phones).

A similar conclusion can be drawn from the PUDDLE performance, which was higher for phones than syllables. By setting the window for onset and offset buffers uniformly at 2, we effectively prevented the algorithm from breaking up more utterances when unitizing with syllables than with phones.

A third conclusion is that performance is enormously affected by unit and algorithm. To investigate this more precisely, we fit a regression with token F-scores as dependent measure, unit and algorithm as well as their interaction as fixed effects, and transcript identity as blocking factor.[3] This model explained 98% of the variance in performance, with both main effects and their interaction being highly significant.

Effects of corpus length

Although the analysis in the previous section showed that nearly all the variance in performance across transcripts was explained by algorithm, unit, and their interaction, it remains possible that transcript characteristics do affect word segmentation performance. As discussed in "Keeping other aspects constant", a good candidate for a factor that would affect performance is corpus length. Preliminary analyses revealed that PUDDLE's performance was changing as a function of corpus length within the sample studied in the previous subsection. Therefore, we carried out an additional experiment to extend the length coverage. We followed previous work (Börschinger et al., 2012; Daland & Pierrehumbert, 2011; Monaghan & Christiansen, 2010) by submitting concatenated versions of the transcripts to our segmentation procedure. That is, we first analyzed the first transcript; then, we concatenated

[3]This regression was preferred over a mixed model because there is disagreement as to how to estimate proportion of variance explained in the latter.

the first two by pasting the second transcript after the first and analyzed the resulting combined corpus; and proceeded in this manner until all included transcripts had been concatenated. Children vary in the number of included transcripts both because some were visited more regularly and from an earlier age (e.g., Naima), and because a different proportion of transcripts were excluded (due to being too short or containing more than one adult, see "Corpus"; e.g., only four out of 40 transcripts for William are included here).

Figure 3 portrays performance on Naima's transcripts. Since variation across children's data was very low, we have only included there results from one child to facilitate readers' visual inspection of results (see full figure in the online supplementary materials). All algorithms exhibit strong changes (upwards or downwards) in the 0–3k region, which may be associated to peculiarities of some of these transcripts, since they are visible even in the baselines. Afterwards, most algorithms remain fairly stable with very slight linear changes (if any), with one exception: PUDDLE. Indeed, we notice that PUDDLE-phones exhibits a non-linear pattern, with performance increasing rapidly between 1k and 4k sentences; peaking at 5–7k sentences; and slowly dropping (by little) thereafter. PUDDLE-syllables increases slowly and linearly throughout the range. In general terms, then, performance is stable for all algorithm-unit combinations (except for PUDDLE-syllables) in the 5–15k region.

We investigated the effects of corpus size more precisely by fitting a linear regression taking the last data point for each child (i.e., the concatenation of all the transcripts associated with that child). As before, the dependent measure was token F-score and the predictors were the algorithm in interaction with the unit (blocked within child). This regression explained 99.2% of the variance; addition of number of words in interaction with algorithm increased this to 99.3% (which was not significant in a Chi-square test).

In short, we have found that for most algorithm-unit combinations, performance is stable across a wide range of corpora sizes (roughly between 5,000 and 15,000 word tokens), and furthermore that corpora size affected performance very minimally once algorithm-unit effects were taken into account.

Discussion

This paper presents a package that allows the systematization of several key steps in the computational study of word form segmentation by infants and other agents. One of the

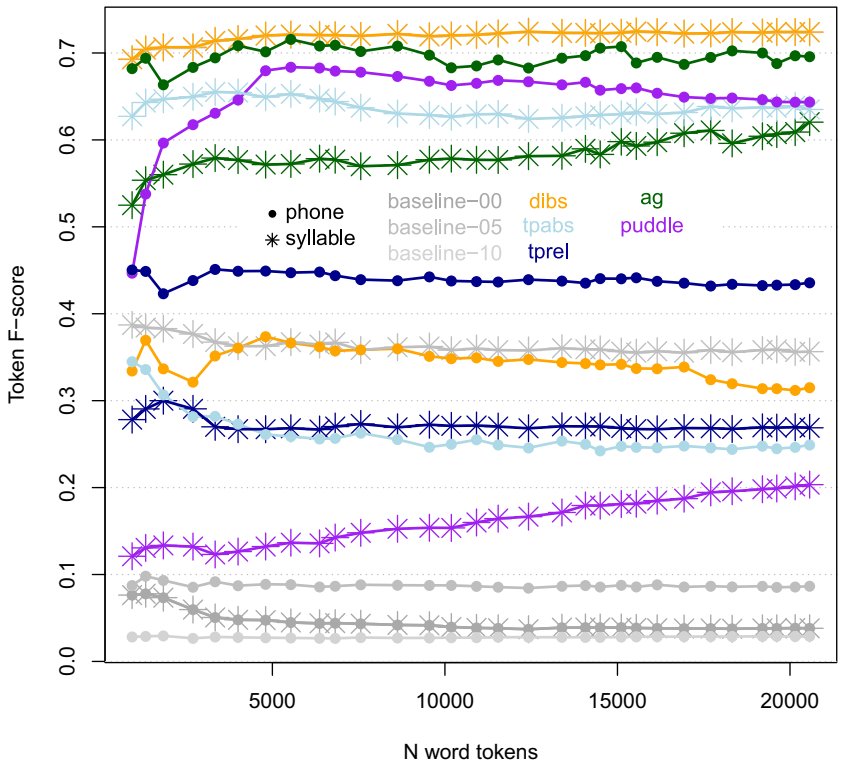

Fig. 3 Token F-scores as a function of unit and algorithm in cumulative transcripts. Each point is the performance of a segmentation experiment on a transcript that is the result of the concatenation of a given transcript and all preceding transcripts for a given child. Since variation across children was low, only Naima's data are shown here; see supplementary materials for the other curves

strengths of our package is that it contains a tool to describe the input. Our analyses of a CHILDES corpus demonstrates that there is wide variability in the input to children even within 1–2 years of age in terms of sentence length and lexical properties which may impact segmentation performance. The package also provides all basic performance measures. Our analyses suggest that these are by and large correlated.

Another key strength of the package is the presence of a tool to unitize this input into phones or syllables as basic phonological units, and a third is that the package contains a range of conceptually diverse algorithms. Our analyses demonstrate that the crossing of these two factors (basic representational unit, and algorithm) has enormous effects on segmentation performance. In contrast, segmentation performance was rather stable across a wide range of corpus sizes, particularly for batch algorithms.

Limitations and future directions

The first direction in which we think the WordSeg suite should be improved is by providing users with solutions for phonologizing their texts, and facilitating informed choices for data selection from CHILDES. Previous researchers have used a range of pre-processing pipelines, making choices that could affect segmentation results. Some researchers remove repetition or mumbling within sentences, which obscures any dependence which may have been present previously. For instance, "she xxx baby girl" would become "she baby girl" (since xxx indicates untranscribed spoken material in the CHAT format), which misrepresents the sequence of words produced by the speaker. Sometimes material tagged as non-lexical or onomatopeic (with the CHAT tags &hey and choochoo@o, respectively) are similarly deleted from the input. Some go so far as dropping words that are not part of the finite dictionary being used. Since child-directed speech will often contain onomatopeia and other forms of non-standard words, such an analytic decision unduly simplifies the task of the word segmenter. The latter problem can be removed by using a text-to-speech system (or grapheme to phoneme conversion rules in languages with transparent orthography) on all potential child input. Such systems may also help make some strides towards making the phonologized input more realistic via the application of phonological processes of e.g. assimilation and reduction.

Although not illustrated in the examples above, the package is flexible enough to allow evaluation of segmentation at linguistic levels other than the word level. For instance, some users may desire to evaluate on morphemes rather than words (Börschinger et al., 2012; Phillips & Pearl, 2015). It has been previously discussed (Phillips & Pearl, 2015) that error evaluation based on the gold word standard might not be optimal when modeling infant segmentation of useful linguistic units. Evaluation on the morpheme level should also be considered, since segmenting out the constituent morphemes of a word could actually help infants acquire more lexical elements of their language (Kim, 2015; Shi, Werker, & Cutler, 2006). Similarly, one can imagine extensions assessing segmentations of yet other levels of the prosodic hierarchy, such as syllables, or syntactic units, such as phrases. A somewhat related issue is how to deal with plausible segmentation errors due to undersegmentation of sequences of words that are often produced together (collocations). To avoid penalizing for these, the user can simply create a version of the gold where word boundaries are removed in high frequency phrases. These extensions are all possible and easy to implement in WordSeg, since both the preparation and the evaluation steps allow the user to provide the code used in their text as separators. However, they all require that the user has exhaustively tagged morpheme boundaries (or whatever other unit they want to evaluate). Future developments could integrate a morphological parser to help users who lack this level of annotation, perhaps building on extant open-source, multilingual tools (e.g., CLAN, MacWhinney, 2009b).

All this said, most readers will agree with us that performance against the gold standard is not necessarily the ultimate goal of research on infant word segmentation. We have begun to investigate how the output of word segmentation algorithms may be related to human performance more directly. Specifically, we have been using parental reports of infant word comprehension as the variable to be predicted (Larsen et al., 2017). This code, although available from Larsen (2018) has not been prepared for public re-use as extensively as the WordSeg code has. Additionally, there is considerable conceptual and methodological work needed to extrapolate the method to corpora of other languages (see Baudet, 2018 for a first attempt). We hope others will find ways of employing the WordSeg package output to relate word segmentation results from computational models to human performance, and similarly document and share their code.

Another conceptual development we foresee involves breaking down the currently incorporated algorithms into recombinable modules. We have opted to reuse extant algorithms to allow users to connect with previous literature. Nonetheless, as word segmentation research advances, it would be ideal to reflect on the fact that some extant algorithms represent a set of conceptual choices, each of which is potentially combinable with others. For example, PUDDLE (Monaghan & Christiansen, 2010) incorporates a strategy that profits from single-word utterances or chunks. In that model, utterances that have not been segmented are encoded directly into long-term memory, and later used to break up new utterances. We could imagine a

model that encodes phonotactics like DiBS does (i.e., not with a list of permissible phone bigrams but rather as a probability distribution of the transition) together with a chunk memorization module as found in PUDDLE. It would also be interesting to explore parameters that have similarly been confounded with other design options, such as whether the model should treat differently phenomena occurring at utterance edges than utterance middles (Venkataraman, 2001), or saliently whether the processing is batch or incremental.

Finally, the modular architecture of WordSeg as well as the fact that it is open source should facilitate its integration with systems focusing on unsupervised learning of language structure at other levels. Recent research has begun to investigate word segmentation from raw speech (Versteegh et al., 2015), an interesting development given infant psycholinguistic research strongly suggesting young infants may build their earliest proto-lexicon using acoustic representations (e.g., Houston & Jusczyk, 2000). Although there are very few public corpora of child-directed speech with phonological transcriptions that are aligned well enough to be usable for this process, some recent work has made great strides towards standardizing and facilitating forced alignment (McAuliffe, Socolof, Mihuc, Wagner, & Sonderegger, 2017), including on CHILDES corpora (Elsner & Ito, 2017; Frermann & Frank, 2017). As to the integration of systems working on other levels of acquisition, it would be worthwhile to explore parsers allowing the discovery of morphological structure within words (such as the open-source Linguistica, see Lee & Goldsmith, 2016, section 5.2) as well as others that succeed in acquiring multi-word dependencies (and thus a form of shallow syntax, e.g., McCauley & Christiansen, 2017).

It is not feasible for us to promise to implement all such developments. Fortunately, having opted for a modular, open-source structure makes it easy for *others* to contribute these and other algorithms. As more and more cognitive scientists and psychologists use computational modeling, more and more students and researchers will have the necessary computer skills to make contributions via the GitHub system. These users would fork our repository from github.com/bootphon/wordseg, add their tool in the wordseg/algos section, and then either keep this improved version in their own repositories, or do a pull request so that the standard WordSeg comes to include their tool. Notice incidentally that the use of readthedocs.com allows us to harvest help sections from within python code, thus inviting tool developers to include statements of use that directly become available to WordSeg users. For readers who find this idea appealing but do not have previous experience with git, we recommend the excellent introduction to git offered by Software Carpentry (https://swcarpentry.github.io/git-novice/, followed by GitHub's tutorials for forking (https://help.github.com/articles/fork-a-repo/) and creating pull requests (https://help.github.com/articles/creating-a-pull-request-from-a-fork/). We provide further information in a dedicated section of our documentation https://wordseg.readthedocs.io/en/latest/contributing.html#contributing-to-the-code.

In conclusion, the present version of WordSeg greatly facilitates research on unsupervised word form segmentation by integrating multiple previous contributions into a modular architecture. We look forward to further improvements, inviting feedback and development.

Acknowledgements This work was directly supported by the Agence Nationale de la Recherche (ANR-14-CE30-0003 MechELex) and the European Research Council (ERC-2011-AdG-295810 BOOTPHON). We also acknowledge funding from the Fondation de France, the Ecole de Neurosciences de Paris, the Region Ile de France (DIM cerveau et pensee); and the institutional support of Agence Nationale de la Recherche (ANR-10-IDEX-0001-02 PSL*, and ANR-10-LABX-0087 IEC). We benefited from code shared directly with AC by Padraic Monaghan. We also reused code stored on GitHub by Lawrence Phillips and originally written by Sharon Goldwater, Lisa Pearl, and/or Mark Johnson. We are grateful to Melanie Soderstrom for help with the systematic review mentioned in "Previous computational modeling work"; as well as to her and Nan Bernstein for comments on a previous version of this manuscript. Finally, we are grateful to members of the LAAC, CoML, and Language teams at the LSCP for helpful discussion.

References

Baudet, G. (2018). Xlingcorrelation. https://github.com/bootphon/XLingCorrelation.

Börschinger, B., Demuth, K., & Johnson, M. (2012). Studying the effect of input size for Bayesian word segmentation on the Providence corpus. In *Proceedings of COLING*, (pp. 325–340).

Brent, M. R. (1999). Speech segmentation and word discovery: A computational perspective. *Trends in Cognitive Sciences*, *3*(8).

Brent, M. R., & Cartwright, T. A. (1996). Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, *61*(1), 93–125.

Cai, D., Zhao, H., Zhang, Z., Xin, Y., Wu, Y., & Huang, F. (2017). Fast and accurate neural word segmentation for Chinese. arXiv:1704.07047.

Daland, R. (2009). Word segmentation, word recognition, and word learning: A computational model of first language acquisition. PhD, Northwestern University.

Daland, R., & Pierrehumbert, J. B. (2011). Learning diphone-based segmentation. *Cognitive Science*, *35*(1), 119–155.

Demuth, K., Culbertson, J., & Alter, J. (2006). Word-minimality, epenthesis and coda licensing in the early acquisition of English. *Language and Speech*, *49*(2), 137–173.

Elsner, M., & Ito, K. (2017). An automatically aligned corpus of child-directed speech. In *Proceedings of interspeech*, (pp. 1736–1740).

Fourtassi, A., Börschinger, B., Johnson, M., & Dupoux, E. (2013). Whyisenglishsoeasytosegment. In *Proceedings of CMCL*, (pp. 1–10).

Frermann, L., & Frank, M. C. (2017). Prosodic features from large corpora of child-directed speech as predictors of the age of acquisition of words. arXiv preprint arXiv:1709.09443.

Gambell, T., & Yang, C. (2005). Word segmentation: Quick but not dirty. Unpublished manuscript.

Goldwater, S., Griffiths, T. L., & Johnson, M. (2009). A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, *112*(1), 21–54.

Gundersen, O. E., & Kjensmo, S. (2018). State of the art: Reproducibility in artificial intelligence. In *Thirty-Second AAAI Conference on Artificial Intelligence*, (p. 17248).

Harris, Z. S. (1955). From phoneme to morpheme. *Language*, *31*(2), 190–222.

Houston, D. M., & Jusczyk, P. W. (2000). The role of talker-specific information in word segmentation by infants. *Journal of Experimental Psychology: Human Perception and Performance*, *26*(5), 1570.

Hung, L. H., Kristiyanto, D., Lee, S. B., & Yeung, K. Y. (2016). GUIdock: Using Docker containers with a common graphics user interface to address the reproducibility of research. *PloS One*, *11*(4), e0152,686.

Johnson, M., Christophe, A., Dupoux, E., & Demuth, K. (2014). Modelling function words improves unsupervised word segmentation. In *ACL*, (pp. 282–292).

Johnson, M., & Goldwater, S. (2009). Improving nonparametric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, (pp. 317–325).

Kim, Y. J. (2015). 6-month-olds' segmentation and representation of morphologically complex words. PhD University of California, Los Angeles.

Larsen, E. (2018). WordSegcomprehension. https://github.com/elinlarsen/WordSegComprehension

Larsen, E., Cristia, A., & Dupoux, E. (2017). Relating unsupervised word segmentation to reported vocabulary acquisition. In *Interspeech* (pp. 2198–2202).

Lee, J. L., & Goldsmith, J. A. (2016). Linguistica 5: Unsupervised learning of linguistic structure. In *Proceedings of NAACL-HLT 2016 (Demonstrations)* (pp. 22–26).

Lignos, C. (2012). Infant word segmentation: An incremental, integrated model. In *Proceedings of the West Coast Conference on Formal Linguistics* (Vol. 30, pp. 13–15).

Ludusan, B., Mazuka, R., Bernard, M., Cristia, A., & Dupoux, E. (2017). The role of prosody and speech register in word segmentation: A computational modelling perspective. In *Proceedings of the 55th annual meeting of the Association for Computational Linguistics* (Vol. 2, pp. 178–183).

Ludusan, B., Versteegh, M., Jansen, A., Gravier, G., Cao, X. N., Johnson, M., & Dupoux, E. (2014). Bridging the gap between speech technology and natural language processing: an evaluation toolbox for term discovery systems. In *Proceedings of LREC* (pp. 560–576).

MacWhinney, B. (2009). *The CHILDES Project part 1: The CHAT transcription format*. New York: Psychology Press.

MacWhinney, B. (2009). *The CHILDES Project part 2: The database*. New York: Psychology Press.

McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., & Sonderegger, M. (2017). Montreal Forced Aligner: Trainable text-speech alignment using Kaldi. In *Proceedings of interspeech* (pp. 498–502).

McCauley, S. M., & Christiansen, M. H. (2017). Computational investigations of multiword chunks in language learning. *Topics in Cognitive Science*, *9*(3), 637–652.

Monaghan, P., & Christiansen, M. H. (2010). Words in puddles of sound: Modelling psycholinguistic effects in speech segmentation. *Journal of Child Language*, *37*(03), 545–564.

Ngon, C., Martin, A., Dupoux, E., Cabrol, D., Dutat, M., & Peperkamp, S. (2013). (non) words, (non) words, (non) words: Evidence for a protolexicon during the first year of life. *Developmental Science*, *16*(1), 24–34.

Pearl, L., Goldwater, S., & Steyvers, M. (2010). Online learning mechanisms for Bayesian models of word segmentation. *Research on Language and Computation*, *8*(2–3), 107–132.

Phillips, L. (2015). The role of empirical evidence in modeling speech segmentation. PhD, University of California, Irvine.

Phillips, L., & Pearl, L. (2015). The utility of cognitive plausibility in language acquisition modeling: Evidence from word segmentation. *Cognitive Science*, *39*(8), 1824–1854.

Pitt, M. A., Johnson, K., Hume, E., Kiesling, S., & Raymond, W. (2005). The Buckeye corpus of conversational speech: Labeling conventions and a test of transcriber reliability. *Speech Communication*, *45*(1), 89–95.

Saksida, A., Langus, A., & Nespor, M. (2017). Co-occurrence statistics as a language-dependent cue for speech segmentation. *Developmental Science*, *20*(3), e12,390.

Shi, R., Werker, J. F., & Cutler, A. (2006). Recognition and representation of function words in English-learning infants. *Infancy*, *10*(2), 187–198.

Stodden, V., Seiler, J., & Ma, Z. (2018). An empirical analysis of journal policy effectiveness for computational reproducibility. *Proceedings of the National Academy of Sciences*, *115*(11), 2584–2589.

Swingley, D. (2005). Statistical clustering and the contents of the infant vocabulary. *Cognitive Psychology*, *50*(1), 86–132.

Taylor, P., Black, A. W., & Caley, R. (1998). The architecture of the FESTIVAL speech synthesis system. In *Proceedings of the 3rd ESCA Workshop on Speech Synthesis*, (pp. 147–151).

Venkataraman, A. (2001). A statistical model for word discovery in transcribed speech. *Computational Linguistics*, *27*(3), 351–372.

Versteegh, M., Thiolliere, R., Schatz, T., Cao, X. N., Anguera, X., Jansen, A., & Dupoux, E. (2015). The Zero Resource Speech Challenge 2015. In *Sixteenth Annual Conference of the International Speech Communication Association*.