A Universal System for Automatic Text-to-Phonetics Conversion

Chen Gafni
Bar-Ilan University
chen.gafni@gmail.com

Abstract

This paper describes an automatic text-to-phonetics conversion system. The system was constructed to primarily serve as a research tool. It is implemented in a general-purpose linguistic software, which allows it to be incorporated in a multifaceted linguistic research in essentially any language. The system currently relies on two mechanisms to generate phonetic transcriptions from texts: (i) importing ready-made phonetic word forms from external dictionaries, and (ii) automatic generation of phonetic word forms based on a set of deterministic linguistic rules. The current paper describes the proposed system and its potential application to linguistic research.

1 Introduction

There are currently many commercial and academic works dealing with automatic conversion of text to phonetics (G2P). Existing solutions are based on some combination of language-specific phonetic dictionaries, corpus-based statistical models (e.g., Hidden Markov Models), deterministic models based on linguistic rules, and machine learning techniques (see review in Tomer, 2012). Importantly, most available tools are closed-source, support only a limited number of languages, and are often available only for commercial use (e.g., Baytukalov, 2019).

The current paper describes a new, open-source system that generates phonetic transcriptions from texts. Its design allows it to apply, in principle, to any language. At present, the system relies on two mechanisms to generate the transcriptions: (i) importing ready-made phonetic word forms from external dictionaries, and (ii) automatic generation of phonetic word forms based on a set of deterministic linguistic rules. The following sections describe the transcription mechanisms and additional relevant tools.

2 Preliminaries

The described system is implemented in the *Child Phonology Analyzer* software (CPA; Gafni, 2015)[1], which was built in MS Excel due to its popularity and user-friendly interface.[2] Nevertheless, the concepts behind the system are general and can be implemented in various environments. The following subsections describe the general organization of the system and guidelines for working with the data.

2.1 Tables

The system uses a set of tables of definitions and rules to guide its operation. The tables are stored in separate spreadsheets in the CPA file and can be edited according to the properties of the language in question. Moreover, variants of these tables can be stored in separate files and imported by the system when needed. This feature allows users to maintain sets of definitions and rules for multiple languages.[3]

2.2 Organizing the Data

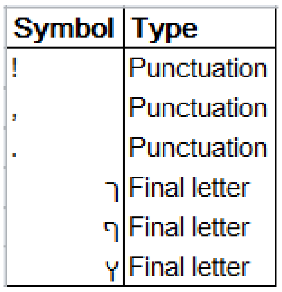
The transcription procedures ("macros") operate on a vector of words. Thus, the input text should be converted into a vector format prior to running the transcription macros. This can be done using the "Corpus tokenization" option of CPA's "Data

[1] Website: https://chengafni.wordpress.com/cpa/
[2] The code is written in Visual Basic for Applications for MS Excel and is available under the GNU General Public License. A version of this system for LibreOffice Calc is planned to appear in the future in order to free it from dependency on proprietary software.
[3] The system is accompanied by an external resource containing proposed sets of rules for several languages.

https://doi.org/10.26615/978-954-452-056-4_042

preparation” macro, which segments the text into words.

Segmentation is performed on the basis of blank spaces and additional word-dividing characters, which can be defined in CPA’s “Word dividers” table (Figure 1). There are two types of word dividers, which can be used for separating words even at the absence of a blank space: *punctuation* marks (e.g., comma) are deleted during segmentation, while *final letters* are not (final letters are special letter forms appearing only at word endings. See some examples from Hebrew in Figure 1). Once the text is transformed into a vector format, transcription can be performed. One of CPA tools (“Reconstruct corpus”) can then be used to recombine the phonetic word forms according to the structure of the original text.



Symbol	Type
!	Punctuation
,	Punctuation
.	Punctuation
ך	Final letter
ף	Final letter
ץ	Final letter

Figure 1: Word dividing symbols

3 Phonetic Dictionaries

For languages with irregular spelling or high proportion of homographs, such as English, Hebrew, and Arabic, automatic phonetic transcription requires a source of ready-made phonetic forms (i.e., a phonetic dictionary) for irregular and ambiguous words. CPA has a built-in macro that can import such ready-made forms. The macro receives as input a vector of written words to-be-transcribed and a phonetic dictionary – a table of written word forms and corresponding phonetic forms. The macro matches phonetic forms from the dictionary to written words in the vector. For words that are not found in the phonetic dictionary, transcription needs to be generated, either manually or with the automatic linguistic model (see next section). However, once the additional phonetic word forms are supplied, CPA can add them to the phonetic dictionary for future use.

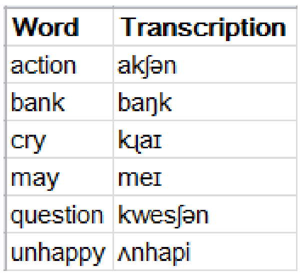
4 The Linguistic Model

For languages with some degree of regular mapping between spelling and sound, phonetic transcription of text can be generated automatically on the basis of deterministic rules. This section describes a multi-stage model of automatic phonetic transcription, guided by linguistic principles. Underlying this model is the assumption that, for any regular orthographic system, phonetic transcription rules can be defined in terms of a small set of general operations. The general operations themselves are hard-coded in the software, but an unlimited number of language-specific rules can be defined on the basis of these operations. This flexible method allows the system to produce automatic phonetic transcription for every language that has, at least partly, regular orthography.

The proposed model has four components, which will be described in the following subsections. The components operate independently of one another, but they should be applied in the order in which they are listed. The first component alone produces sufficient results for most purposes. If needed, the additional three components can be used, together, for fine-tuning.

4.1 Pre-Prosody Transcription

This component takes as input the vector of words to-be-transcribed, a table of pre-prosody transcription rules (Table 1), and a table containing sets of symbols and strings, called “phono-orthographic groups” (Table 2).[4] The pre-prosody transcription[5] applies the transcription rules in successive order to the list of words. Entities



Word	Transcription
action	akʃən
bank	baŋk
cry	kɹaɪ
may	meɪ
question	kwesʃən
unhappy	ʌnhapi

Figure 2: Transcribed words

[4] The tables of rules are generated manually, in principle. CPA has a set of editable tables that contain proposed rules for several languages (see also 2.1).

[5] The term ‘pre-prosody’ indicates that the transcription rules applied by this component disregard the prosodic properties of the word, including syllable structure and stress pattern.

	Target	Output	Type	Preceding environment	Following environment
1	ph	f			
2	c				k
3	ay	eɪ		#?	#
4	[cg]	[sj]			[Front_vowel]
5	[Consonant][Consonant]	[Consonant]	Degemination		
6	◌ّ		Lengthening		
7	[V_diac][C_diac]	[C_diac][V_diac]	Metathesis		

Table 1: Pre-prosody transcription rules

defined in the table of phono-orthographic groups may be called by transcription rules. The output of the process is a vector of phonetic forms corresponding to the written words (Figure 2).

The table of pre-prosody transcription rules has five fields (Table 1): (1) Target: the input to the rule, i.e., the string to be converted. All rules must have a value for the target string. The other fields are optional. (2) Output: the string replacing the target; if left empty, the target string will be deleted. (3) Type: the type of operation to be performed by the rule; if left empty, simple substitution will be performed (see below for other types of rules). (4) Preceding environment, and (5) Following environment: these fields are used for formulating context-sensitive rules. When either field is not empty, the transcription rule will apply only to words in which the target string is preceded by the 'preceding environment string' and/or followed by the 'following environment string'.

Substitution rules can be used for simple grapheme-to-phoneme conversions, which can be either context-free or context-sensitive. For example, rule 1 in Table 1 is a context-free substitution of *ph* by *f* in words such as *phone* (/foʊn/). Rule 2 in Table 1 is an example for context-sensitive rule – deleting *c* before *k* in words such as *back* (/bæk/).

Substitution rules may include wildcards to define more general entities. Three types of wildcards are defined in the software: a question mark (?) stands for any single character in the target, output or environment strings; an asterisk (*) stands for any number of successive characters; and, a hash sign (#) represents word boundaries. Wildcards can be used for defining phonological and morphological words patterns. For example, rule 3 in Table 1 captures the pronunciation of *ay* at the end of three-letter English words, such as *bay* (/beɪ/). The question mark in this rule indicates that the rule applies to *ay* sequences preceded by a single character. The hash signs suggest that the rule applies only when word boundaries are present at both edges.

Group	Members
cg	c,g
sj	s,ʤ
Front_vowel	e,i,y
Consonant	b,d,f,g,l,m,n,p,r,s,t,z
C_diac	◌ּ,◌ׂ,◌ׁ,'
V_diac	◌ְ,◌ֱ,◌ֹ,◌ָ,◌ַ,◌ֶ,◌ֵ,◌ִ

Table 2: Phono-orthographic groups

Rules can also be generalized by the inclusion of phono-orthographic groups, defined in a separate table. Each entry in the table of phono-orthographic groups has two fields (Table 2): the name of the group, and its members. For example, the term *Front_vowel* can be used for grouping *e*, *i*, and *y*.

Transcription rules can include phono-orthographic groups by enclosing the name of the group between brackets (e.g., [Front_vowel]). When a group is embedded in a transcription rule, the algorithm converts the compact rule into a set of simple rules, each applying to a different member of the group. For example, rule 4 in Table 1 uses groups to capture the pronunciation of *c* and *g* before front vowels (/s/ in *cent* /sent/ and /ʤ/ in *gene* /ʤiːn/, respectively). This single, compact rule stands for six simple rules: c→s/_e, c→s/_i, c→s/_y, g→ʤ/_e, g→ʤ/_i, g→ʤ/_y (where the formula A→B/_X is read: A becomes B before X).

In addition to substitution rules, several types of special operations can be used by specifying the name of the operation in the 'Type' field in the table of rules. Three types of operations are defined in the software: *degemination*, *lengthening*, and *metathesis*.

Degemination is used for collapsing a sequence of two identical phones when pronounced as a single, short sound. For example, rule 5 in Table 1 collapses sequences of identical consonants (e.g., *mm* is pronounced as a single *m* in *hammer*).

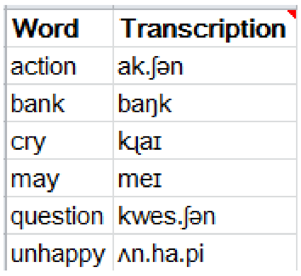
Lengthening realizes the function of diacritical marks of lengthening/gemination. For example, rule 6 in Table 1 realizes the function of the Arabic Shaddah (e.g., the letter م is pronounced /m/ in its plain form, but as /mm/ when modified by a Shaddah, i.e.,مّ).

Finally, *metathesis* switches the order of elements in sequences of two phono-orthographic groups (i.e., if the target contains a member of group 1 followed by a member of group 2, they are switched in the output). For example, in pointed Hebrew scripts, diacritics are used for indicating vowels as well as for modifying the phonetic value of consonants. A single letter can host both consonant and vowel diacritics (the C_diac and V_diac groups in Table 2, respectively). Thus, the string בַּ, pronounced /ba/, is composed of the letter ב (representing the consonants /b/ and /v/), a consonant diacritic ◌ּ (specifying the consonant /b/), and a vowel diacritic ◌ַ (representing the vowel /a/). Although the order in which these diacritics are attached to the letter does not affect the visual form of the text, it is important for the purpose of phonetic transcription – consonant diacritics must be attached before vowel diacritics. For example, the string בַּ can be formed by combining the three elements in two ways: ב+◌ּ+◌ַ, or ב+◌ַ+◌ּ. However, only the first order reflects the phonological structure of the string. A rule of metathesis can be defined to switch the order in sequences of vowel diacritics + consonant diacritic to guarantee correct ordering (rule 7 in Table 1).

After performing the pre-prosody transcription, certain modifications might be needed due to phonological processes related to prosodic structure. This post-prosody transcription (see 4.4) requires that the phonetic forms be parsed into syllables and have stress markers assigned to them. These components are described below.

4.2 Syllabification

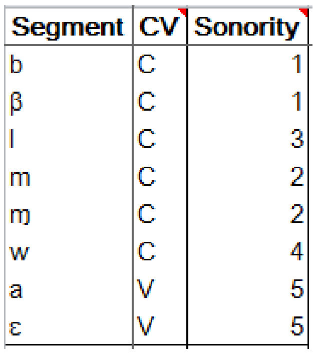
This component takes as input a vector of phonetic word forms, a list of binary parameters and parameter weights, and a phonetic table. The output is a vector of syllabified phonetic word forms (Figure 3). The basic sites of syllable boundaries are around local sonority minima (e.g., the boldfaced consonants in fæ**m**ə**l**i 'family' → fæ.**m**ə.**l**i).



Word	Transcription
action	ak.ʃən
bank	baŋk
cry	kɹaɪ
may	meɪ
question	kwes.ʃən
unhappy	ʌn.ha.pi

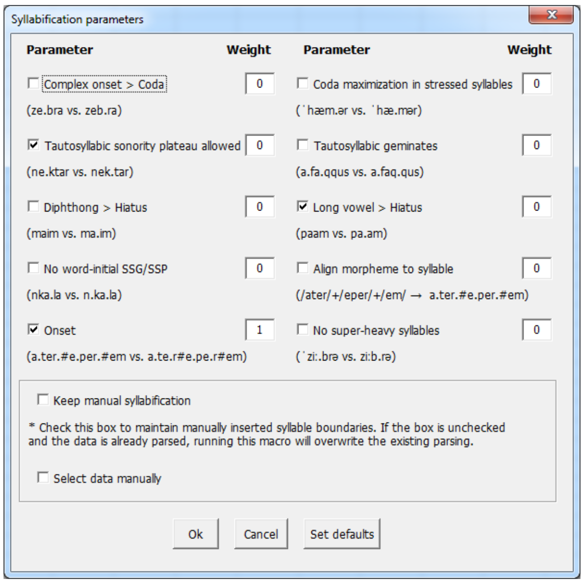
Figure 3: Syllabified phonetic word forms

In order to determine sites of sonority minimum, the syllabification procedure converts the phonetic word forms into strings of sonority levels. Sonority levels are non-negative integers specified for each phone in the phonetic table of CPA (Figure 4). For example, if fricatives, nasals, liquids, and vowels have sonority levels of 1, 2, 3, and 5, respectively, the sonority-level representation of *fæməli* will be 152535. In this representation, 2 and 3 are local sonority minima (Sonority minima at word edges are ignored).



Segment	CV	Sonority
b	C	1
β	C	1
l	C	3
m	C	2
ɱ	C	2
w	C	4
a	V	5
ɛ	V	5

Figure 4: The phonetic table



Syllabification parameters

Parameter	Weight	Parameter	Weight
☐ Complex onset > Coda (ze.bra vs. zeb.ra)	0	☐ Coda maximization in stressed syllables (ˈhæm.ər vs. ˈhæ.mər)	0
☑ Tautosyllabic sonority plateau allowed (ne.ktar vs. nek.tar)	0	☐ Tautosyllabic geminates (a.fa.qqus vs. a.faq.qus)	0
☐ Diphthong > Hiatus (maim vs. ma.im)	0	☑ Long vowel > Hiatus (paam vs. pa.am)	0
☐ No word-initial SSG/SSP (nka.la vs. n.ka.la)	0	☐ Align morpheme to syllable (/ater/+/eper/+/em/ → a.ter.#e.per.#em)	0
☑ Onset (a.ter.#e.per.#em vs. a.te.r#e.pe.r#em)	1	☐ No super-heavy syllables (ˈzi:.brə vs. zi:b.rə)	0

☐ Keep manual syllabification

* Check this box to maintain manually inserted syllable boundaries. If the box is unchecked and the data is already parsed, running this macro will overwrite the existing parsing.

☐ Select data manually

Figure 5: Syllabification parameters

	Trigger	Tier	Position	Process	Result
1	[+STRID][+STRID]	Features	Coda	Vowel epenthesis	ə
2	Unstressed vowel	CV		Vowel reduction	ə
3	Sonority decrease	Sonority	Onset	Vowel epenthesis	ə

Table 3: Post-prosody transcription rules

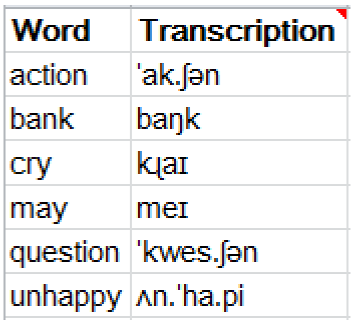
The basic sites of syllable boundaries can be adjusted by a set of binary parameters, which handle various cases, such as consonant sequences (e.g., whether /dɪspleɪ/ ‘display’ should be parsed as dɪs.pleɪ or dɪ.spleɪ). The system currently has 10 built-in parameters, which can be switched on and off according to the properties of the examined language (Figure 5). Some of the parameters require phones to be recognized as vowels or consonants. This information is also specified in CPA’s phonetic table (Figure 4). For example, if the *Complex onset > Coda* parameter is switched on, consonant sequences will be parsed as complex onsets (dɪ.spleɪ). If the parameter is switched off, consonant sequences will be split between coda and onset positions (dɪs.pleɪ).

When two parameters are potentially in conflict, they can be ranked relative to each other by assigning different integer weights to them. For example, if the *Onset* parameter is on, onsetless syllables will be dispreferred (e.g., ˈmʌni → ˈmʌ.ni ‘money’). This can be overridden (e.g., ˈmʌn.i) by switching on the *Coda maximization in stressed syllables* parameter and giving it a higher weight than the onset parameter (this requires that stress would be marked on the word before running syllabification).

4.3 Stress Assignment

When the phonetic transcription requires modifications due to processes related to stress (e.g., reduction of unstressed vowels), stress markers should be added to the phonetic word forms. The stress assignment component of the software takes as input a vector of syllabified words and the desired stress pattern. The output is a vector of syllabified words with stress markers inserted at the appropriate positions (e.g., for the input word ak.ʃən ‘action’ and penultimate stress pattern, the output will be ˈak.ʃən; Figure 6).

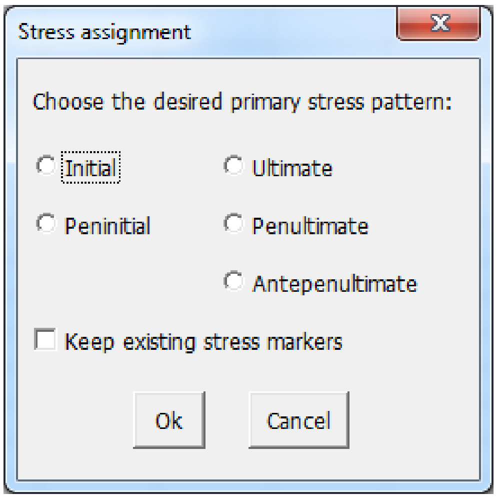
The software has five built-in stress patterns (at present, only primary stress is handled): Initial, Peninitial, Ultimate, Penultimate, and Antepenultimate (Figure 7). For languages with a non-fixed stress pattern (e.g., English), this procedure can be used for applying the most



Word	Transcription
action	'ak.ʃən
bank	baŋk
cry	kɻaɪ
may	meɪ
question	'kwes.ʃən
unhappy	ʌn.'ha.pi

Figure 6: Phonetic word forms with stress

frequent stress pattern. Manual corrections can be made afterwards. If stress position depends on the number of syllables, it is possible to run stress assignment multiple times, starting with the rule for the longer words. Checking the ‘Keep existing stress markers’ option will prevent stress rules for shorter words from applying to longer words, for which stress has been assigned already.



Stress assignment

Choose the desired primary stress pattern:

- ○ Initial
- ○ Ultimate
- ○ Peninitial
- ○ Penultimate
- ○ Antepenultimate

☐ Keep existing stress markers

Figure 7: Stress parameters

4.4 Post-Prosody Transcription

This components modifies phonetic word forms according to phonological rules related to prosodic structure. It takes as input a vector of phonetic word forms, a table of post-prosody transcription rules (Table 3), and a phonetic table. The procedure applies the transcription rules in successive order to the list of words.

The table of post-prosody transcription rules has five fields (Table 3): (1) Trigger: the phonological structure triggering the required modification. A trigger can be a specific element or a sequence of elements defined in the phonetic table (e.g., [+STRID][+STRID] stands for a sequence of two stridents such as /s/ and /z/). In addition, there are several types of special pre-defined triggers: The *Sonority decrease*, *Sonority increase*, and *Sonority plateau* triggers handle phone sequences in which the sonority level decreases, increases, or remains unchanged, respectively (see Figure 4). The *No vowel* trigger handles syllables with no vowels. The *Unstressed vowel* trigger handles vowels in unstressed syllables.

(2) Tier: the phonological tier relevant to the trigger. *Features* tier is used with phonological features triggers (e.g., [+STRID]), while *CV* tier is used with *No vowel* and *Unstressed vowel* triggers. *Sonority* tier is used with all sonority-related triggers.

(3) Position: for triggers applying to consonants (sonority and feature triggers), this field indicates the prosodic position (*Onset* or *Coda*) in which the trigger must be found in order to trigger the modification.

(4) Process: the type of modification applied to phonetic word forms in which the trigger is found. Currently, the software can perform two types of modifications: *Vowel epenthesis* inserts a vowel to correct ill-formed sequences, and *Vowel reduction* replaces unstressed vowels with a default neutral vowel.

(5) Result: this field specifies inserted elements (epenthetic vowels and neutral vowels).

The following examples demonstrate the application of post-prosody rules. Rule 1 in Table 3 inserts an epenthetic ə to break sequences of two stridents in coda position (e.g., makss → maksəs ‘Max's’, where *makss* is the output of the pre-prosody transcription, which converted *M* to *m* and *x* to *ks*, and deleted the apostrophe in Max's). Rule 2 in Table 3 replaces vowels in unstressed syllables by ə (e.g., ˈe.le.fant → ˈe.lə.fənt ‘elephant’, where ˈe.le.fant is the result of pre-prosody transcription, syllabification and assignment of antepenultimate stress to elephant).

5 Discussion

This paper describes a system of text-to-phonetics conversion. The system is incorporated in a general-purpose linguistic software that includes tools for building dictionaries, as well as corpus analysis functions. Thus, the described system can help studying the phonological properties of text corpora and it is also useful for creating resources for under-resourced languages. For example, it was used for creating a phonological dictionary for Hebrew (Gafni, 2019). In addition, the linguistic model of the software, by itself, can be used as a research and educational tool. The pre-prosody transcription tool, in particular, can be used for exploring and demonstrating the effect of rule-ordering – a common practice in theoretical phonology. In fact, the studied language need not have a writing system at all; the input corpus can be a list of hypothesized phonological underlying representations, and the transcription rules can be phonological rules transforming the underlying representations to surface representations.

In addition, the linguistic model can be used for calculating indices of linguistic complexity by assessing the proportion of words that have regular spelling in a given language and the number of deterministic rules needed to capture the patterns of orthographic regularity in a language. Such measures of complexity can be valuable for literacy education (e.g., Smythe et al., 2008).

It should be noted that the described system is still under development. At its current state, the transcription system can perform perfectly on completely regular orthographies with a fixed stress pattern. Several planned improvements will allow the system to handle more complex cases. For example, the stress assignment component should handle secondary stress and stress rules that are sensitive to syllable weight. In addition, the post-prosody transcription should include more options, such as referring to pretonic syllables, which are relevant sites for certain phonological processes like vowel reduction in Russian (Asherov et al., 2016).

Furthermore, the generalizability of the system can greatly improve by adding machine learning procedures, such as sequence-to-sequence models with greedy decoding (Chae et al., 2018). This will allow the system to generate rules automatically based on examples. It will also be able to handle cases of homography (e.g., whether *wind* should be transcribed /wɪnd/ (noun) or /waɪnd/ (verb)) by analyzing token frequency and contextual effects (syntax and semantics). Such improvements will make the transcription system more powerful and reliable.

References

Daniel Asherov, Alon Fishman, and Evan-Gary Cohen. 2016. Vowel Reduction in Israeli Heritage Russian. *Heritage Language Journal*, 2:113–133.

Timur Baytukalov. 2019. EasyPronunciation.com: All-in-one solution to learn pronunciation online.

Moon Jung Chae, Kyubyong Park, Linhyun Bang, Soobin Suh, Longhyuk Park, Namju Kimt, and Longhun Park. 2018. Convolutional sequence to sequence model with non-sequential greedy decoding for grapheme to phoneme conversion. In *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018)*, pages 2486–2490.

Chen Gafni. 2015. Child Phonology Analyzer: processing and analyzing transcribed speech. In The Scottish Consortium for ICPhS 2015, editor, *Proceedings of the 18th International Congress of Phonetic Sciences.*, page 1–5, paper number 531, Glasgow, UK: the University of Glasgow.

Chen Gafni. 2019. General Lexicons of Hebrew: Resources for Linguistic and Psycholinguistic Research (version 1.0).

Ian Smythe, John Everatt, Nasser Al-Menaye, Xianyou He, Simone Capellini, Eva Gyarmathy, and Linda S. Siegel. 2008. Predictors of word-level literacy amongst Grade 3 children in five diverse languages. *Dyslexia*, 14(3):170–187.

Eran Tomer. 2012. *Automatic Hebrew Text Vocalization*. Ph.D. thesis, Ben-Gurion University of the Negev.