



# ***DICTIONARYMAKER* USER MANUAL**

## **VERSION 2.16**

M Tempest and M Davel

12 October 2009

# Table of Contents

<b>1 OVERVIEW.....</b>	<b>4</b>
<b>2 CITING DICTIONARYMAKER.....</b>	<b>4</b>
<b>3 KEY CONCEPTS.....</b>	<b>4</b>
3.1 MAIN FUNCTIONALITY.....	4
3.2 MAIN DATA COMPONENTS.....	5
3.3 DICTIONARY PROJECT.....	5
3.4 BOOTSTRAPPING PROCESS.....	5
3.4.1 User perspective.....	5
3.4.2 System perspective .....	6
3.5 NEXT WORD SELECTION.....	7
3.6 AUDIO SUPPORT.....	7
3.7 UPDATING MODES.....	7
3.8 ERROR DETECTION.....	7
3.9 VERIFICATION STATUS .....	8
3.10 GRAPHEME-TO-PHONEME RULE EXTRACTION.....	8
<b>4 FILE FORMATS.....</b>	<b>8</b>
4.1 PHONEME FILE FORMAT.....	8
4.2 GRAPHEME FILE FORMAT.....	9
4.3 DICTIONARY FILE FORMAT.....	10
4.3.1 Native format.....	10
4.3.2 Exported format: HTK.....	10
4.3.3 Exported format: Festival.....	10
4.3.4 Exported format: semicolon-delimited.....	10
4.4 SOUND FILE FORMAT.....	11
4.5 WORD LIST.....	11
4.6 PROJECT FILE.....	11
<b>5 INSTALLING AND RUNNING DICTIONARYMAKER ON WINDOWS.....</b>	<b>11</b>
5.1 JAVA RUNTIME ENVIRONMENT.....	11
5.2 INSTALLING THE BINARY RELEASE.....	11
5.3 RUNNING DICTIONARYMAKER.....	12
5.4 BUILDING THE SOURCE RELEASE.....	12
<b>6 INSTALLING AND RUNNING DICTIONARYMAKER ON LINUX.....</b>	<b>12</b>
6.1 JAVA RUNTIME ENVIRONMENT.....	12
6.2 INSTALLING THE BINARY RELEASE.....	13
6.3 RUNNING DICTIONARYMAKER.....	13
6.4 BUILDING THE SOURCE RELEASE.....	14
<b>7 USING DICTIONARYMAKER.....</b>	<b>14</b>
7.1 STARTING A NEW DICTIONARY.....	14
7.2 BUILDING ON AN EXISTING PROJECT.....	15
7.3 BUILDING A DICTIONARY.....	16
7.4 CHANGING GRAPHEME SET/PHONEME SET/WORD LIST.....	17
7.5 IMPORTING A DICTIONARY.....	17
7.6 CHANGING THE UPDATE AND WORD SELECTION MODE .....	18
7.7 EXPORTING DATA.....	18
7.8 ANALYSING A PROJECT, DICTIONARY OR RULE SET STATUS.....	18
7.9 SYNCHRONISING THE RULE SET.....	18
<b>8 FUTURE FUNCTIONALITY.....</b>	<b>18</b>

<b>9</b>	<b>DICTIONARYMAKER LICENSING.....</b>	<b>19</b>
----------	---------------------------------------	-----------

## 1 Overview

The DictionaryMaker system facilitates the creation of an electronic pronunciation dictionary in a source language, as originally described in Davel and Barnard, 2003<sup>[1]</sup>.

A pronunciation dictionary consists of a list of words, each associated with one or more pronunciations. The developed pronunciation dictionary can be formatted for use by various speech processing applications, such as speech synthesis and speech recognition systems. Along with the pronunciation dictionary, a related set of grapheme-to-phoneme (g-to-p) rules is automatically created.

The system is designed to allow a speaker, who is fluent in the source language, to develop a pronunciation dictionary without requiring expert linguistic knowledge or programming expertise. The system balances machine learning and human intervention in order to simplify and minimise the human intervention required during the dictionary creation process. To do this, the system utilises a bootstrapping approach which involves improving models according to a controlled set of increments, at each increment utilising the previous model to generate the next.

Only a word list, a grapheme set and phoneme set for the source language are required as inputs to the system. A predefined dictionary can also be used to initiate a new project. Once initialised with these items, the system guides the source language speaker through the dictionary creation process.

## 2 Citing DictionaryMaker

When referring to DictionaryMaker in a publication, please cite the following article:

*M. Davel and E. Barnard, "Pronunciation Prediction with Default&Refine", Computer Speech and Language, vol 22, pp. 374-393, October 2008.*

If adding a footnote to note the source, please use the following URL:

<http://dictionarymaker.sourceforge.net/>

For example:

DictionaryMaker<sup>1</sup> v2.16 was used to extend the existing dictionary with 10 000 additional words through an interactive bootstrapping process <sup>[2]</sup>.

## 3 Key concepts

### 3.1 Main functionality

The system's main mode of operation is an interactive one that requires user input during the execution of each task.

The main interactive functions supported by the system are the following:

- Creating a new electronic pronunciation dictionary.
- Manipulating and updating an existing pronunciation dictionary.

In addition, some supporting (non-interactive) functionality is provided:

- Verifying that imported data is in the correct format.
- Creating a grapheme or phoneme set from an existing dictionary.
- Creating a g-to-p rule set from an existing dictionary.
- Creating a word list from an existing dictionary.
- Identifying possible errors in an existing dictionary.

---

<sup>1</sup>The DictionaryMaker software can be downloaded from <http://dictionarymaker.sourceforge.net/>.

- Providing statistics on a dictionary project:
  - The number of words in the dictionary, and number of words per category.
  - The number of rules in the dictionary, and a list of all rules generated from the dictionary.
- Exporting dictionary data in the following file formats:
  - DictionaryMaker
  - HTK [3]
  - Festival [4]
  - Semicolon-delimited.

## 3.2 Main data components

The main (abstract) data components of the system are the following:

1. A grapheme set  $G$ : a list of allowed graphemes (letters).
2. A phoneme set  $P$ : a list of allowed phonemes (sounds).
3. A rule set, describing how a grapheme in a specific graphemic context maps to a specific phoneme. The rule set format:

$$g_0..g_{i-1} - g_i - g_{i+1} ..g_n \rightarrow p, g_j \in G, p_i \in P.$$

The rule specifies that grapheme  $g_i$ , when found in the left context  $g_0..g_{i-1}$  and right context  $g_{i+1} ..g_n$  is realised as phoneme  $p_i$ .

4. A pronunciation dictionary: a dictionary consisting of word/pronunciation pairs currently being manipulated. For each pair a status indicator specifies whether a word has been verified, and if so, what the verdict is. (See section 4.3 for more information on this aspect.)
5. An activity log: a log of all significant activity associated with a specific dictionary, including timing information.

## 3.3 Dictionary project

A dictionary project consists of all the data components and the history (log) associated with a specific dictionary creation project. A single dictionary is often created through a sequence of projects, with each project improving or manipulating the dictionary in a specific way.

## 3.4 Bootstrapping process

### 3.4.1 User perspective

Before an experiment can be run, a list of source words, a grapheme set, a phoneme set and an associated set of sound samples are defined by the user.

The system runs through the words in the word list one by one, predicts a pronunciation and presents the user with an audio version of the word. The user acts as a “verifier” and provides a verdict with regard to the accuracy of the word/pronunciation pair. The first decision which the verifier makes is whether or not the word is valid:

- invalid - the word is not a valid word (e.g. it is a URL, it is spelled incorrectly or it is only part of a word). In such a case no pronunciation is captured, and the verifier continues on to the next word.
- If the word should be captured in the current dictionary, i.e. is a valid word, the verifier then

considers the pronunciation carefully, possibly correcting the pronunciation of the word on a phone-by-phone basis. Once the pronunciation has been updated, the verifier can select one of the following verdicts:

- correct – the word is valid and its pronunciation as displayed is correct.
- ambiguous – there is more than one valid pronunciation of the word, i.e. there are pronunciation variants. Only the most general form is added.
- proper noun – the word is a proper noun.
- foreign – the word is a valid word from a foreign language, but not a valid word in the source language.

At any stage, the verifier can choose to indicate that he/she is uncertain about the status of either a word or a pronunciation:

- uncertain – the verifier is unable to decide whether the word and/or its pronunciation is valid. This is a way for a verifier to flag words that need to be rechecked. These words will not be presented again for verification during the same session.
- skip – the verifier would like to skip the specific word for the time being. Skipped words will be presented again for verification during the same session.

These categories are currently not configurable, but the intention is to make them more flexible in future. For now, the following behaviour is associated with the above categories:

Category	Pronunciation saved	Used to extract g2p rules	Category saved	Seen as a completed word	Comment
correct	Yes	Yes	Yes	Yes	
invalid	No	No	Yes	Yes	
uncertain	Yes	No	Yes	Yes	
ambiguous	Yes	Yes	Yes	Yes	Exactly the same as “correct”: adding additional variants currently needs to be dealt with as an off-line process.
proper noun	Yes	No	Yes	Yes	
foreign	Yes	No	Yes	Yes	Same behaviour as “proper noun”, just a different flag.
unverified	No	No	No	No	Initially all words are unverified. Skipped words remain unverified.

This process is repeated (with increasingly accurate predictions) until a pronunciation dictionary of sufficient size is obtained.

### 3.4.2 System perspective

During each bootstrapping cycle, grapheme-to-phoneme rules are automatically extracted from the current version of the pronunciation dictionary, and used to generate additional word/pronunciation pairs based on the word list provided. The new word pairs are presented to the user for correction, and better rules are extracted from the corrected words.

The system initially predicts empty pronunciations, which, when corrected, form the basis for further bootstrapping.

The overall process consists of the following steps:

1. The system analyses its current understanding of the task (based on the word list characteristics and the type and status of pronunciations from the overall word list) and chooses the next word or set of words to be considered.
2. The system then generates a new pronunciation for each of the words in the above list, using its current grapheme-to-phoneme rule set.
3. The system creates a “sounded” version of each word using the predicted pronunciation and the user-specified sound samples, and records the verifier's response.
4. Based on the status of each of the words in the newly verified word/pronunciation list, the system extracts a new grapheme-to-phoneme rule set. Only word/pronunciation pairs marked as “correct” are used when extracting a new rule set.
5. This process is repeated until a sufficient number of correct words is obtained.

The system logs the history of all activities per project.

### 3.5 Next word selection

The next word selection is set in the “Project > Properties” menu option, and defaults to “system select”. In “system select” mode, the next word to be verified is randomly selected from the set of unverified words. In “user select” mode, the next word to be verified is the next unverified word found in the word list uploaded by the user, as displayed in the word list panel.

### 3.6 Audio support

A user can listen to the phoneme set at any time. When a new word is predicted, a “sounded” version of the word is generated. This sounded version consists of a concatenation of the phoneme samples. Currently, sound samples must be added by the user. Sound files need to be recorded in a different application and saved separately. (We recommend *Praat* for this purpose, which is available at <http://www.fon.hum.uva.nl/praat/>.)

The user can choose to play the sounded version as many times as required. In auto-play mode, the word is played once when predicted, without requiring user interaction.

It is strongly recommended that the audio support be utilised when building a pronunciation dictionary, especially if the user building the dictionary has limited linguistic experience or is unfamiliar with the phone set being used [5].

### 3.7 Updating modes

Rules are updated during the bootstrapping process according to the update mode chosen, i.e. “continuous update” or “batch update” mode, as follows:

- In “continuous update” mode a new rule set is extracted every time a word verified as “correct” is added to the system. (The updated rules are then used to predict the subsequent word.)
- In “batch update” mode the rule set is only updated after a user-specified number of words (batch size) have been verified as “correct”.

The user can choose to switch modes at any time.

### 3.8 Error detection

The system has the ability to identify possible errors in a pronunciation dictionary by flagging words that create a large number of exception rules. The error indicators can be seen in the word list display panel by utilising the “show by status” option.

### 3.9 Verification status

For each word/pronunciation pair in a pronunciation dictionary, a status indicator specifies whether a word has been verified, and if so, whether the verdict was that the word is “correct”, “invalid”, “uncertain”, “ambiguous”, a “proper noun” or a “foreign” word.

Only “correct” words are used during grapheme-to-phoneme rule extraction.

### 3.10 Grapheme-to-phoneme rule extraction

The Default&Refine algorithm is used for rule extraction and pronunciation prediction. Default&Refine [6] is a greedy search algorithm that extracts a list of increasingly specialised context-sensitive rules from a dictionary. Prior to rule extraction, word/pronunciation pairs are aligned using an optimised version of Viterbi alignment [7].

The rule set format is similar to most rewrite rule schemes:

$$g_0..g_{i-1} - g_i - g_{i+1} ..g_n \rightarrow p, g_j \in \text{Graphemes}, p_i \in \text{Phonemes}$$

where this rule specifies that grapheme  $g_i$ , when found in the left context  $g_0..g_{i-1}$  and right context  $g_{i+1} ..g_n$  should be realised as phoneme  $p_i$ . Rules are ordered explicitly. When a new word is being predicted, the graphemes are processed one at a time. Each grapheme and its left and right context is compared to the rules in the rule set, and the first matching rule is applied.

## 4 File formats

The following files are stored together in the same directory, as a DictionaryMaker project:

- Phoneme definition (.pho)
- Grapheme list (.gra)
- Pronunciation dictionary (.dict)
- Word list (.wdl)
- Project (.proj)
- History (.log)

Apart from the history file, all files can either be created by DictionaryMaker, or be manipulated manually without using DictionaryMaker. The history file is a system-generated file that cannot be edited.

The .wav files for the various phonemes listed in the phoneme file, need to be placed in the “sounds” directory.

### 4.1 Phoneme file format

The phoneme file is a text file of the following layout for each line:

<phoneme symbol><tab><sound file><tab><category>

For example:

<i>a</i>	<i>a.wav</i>	<i>vowels</i>
<i>ch</i>	<i>ch.wav</i>	<i>consonants</i>
<i>k_h</i>	<i>k_h.wav</i>	<i>consonants</i>

One of the following phoneme sets can be used in each phoneme file:



- IPA
- SAMPA
- HTK

The supported subset of the IPA phoneset is the following:

tɸ, tɸ', tɸ<sup>h</sup>, tʃ, tʃ', tʃ<sup>h</sup>, ts', ts<sup>h</sup>, kx', kx<sup>h</sup>, kɬ, ps', ps<sup>h</sup>, pʃ<sup>h</sup>, pʃ', dʒ, dz, dβ, dz, dz<sup>h</sup>, dʒ<sup>h</sup>, f, v, θ, ð, s, z, ʃ, ʒ, x, h, ɦ, ɬ, ɮ, dɬ, dʒ, ɣ, β, ɸ, ɤ, sw, zw, ɸs, ɸS, βʒ, fʃ, ɿ, ɿgɿ, ɿ<sup>h</sup>, ɿɿ, ɿɿgɿ, ɿɿ<sup>h</sup>, ɿ!, ɿgɿ!, ɿ<sup>h</sup>, p, p<sup>h</sup>, p', pʃ', pʃ<sup>h</sup>, b, ɓ, by, t, t<sup>h</sup>, t', tʃ, tʃ<sup>h</sup>, d, dj, dɦ, c', c<sup>h</sup>, ʈ, k, k<sup>h</sup>, k', g, gh, tɿ, tɿ<sup>h</sup>, m, n, ɲ, ŋ, ŋ, mɦ, nɦ, ɱ, ɲɦ, ɿ, i:, y, u, u:, e, e:, ø:, ɛ, ɜ:, ɔ, ɔ:, a, a:, ə, æ, o, o:, œ, ɒ, əi, œy, ai, ɔi, əu, au, iə, eə, uə, r, rɦ, l, j, w, ɭ

The supported subset of the SAMPA phoneset is the following:

tp\, tK\_>, tK\_h, tS, tS\_>, tS\_h, ts\_>, ts\_h, kx\_>, kx\_h, kK\_>, ps\_>, ps\_h, pS\_h, pS\_>, d\_0Z, dz, dB, dz', dz'h\, dZh\, v, T, D, s, z, S, Z, x, h, h\, K, K\, dK, dK\, G, B, p\, s', sw, zw, p\, p\ S, BZ, fS, ɿ, ɿg\_0, ɿh, ɿɿ, ɿɿg\_0, ɿɿh, ɿ!, ɿ!g\_0, ɿ!h, p, p\_h, p\_>, pj\_>, pj\_h, b, b\_<, bj, t, t\_h, t\_>, tj, tj\_h, d, dj, dh\, c\_>, c\_h, J\, k, k\_h, k\_>, g, gh\, tl\_>, tl\_h, m, n, J, N, n', m\_h, n\_h, m\_j, J\_h, ɿ, i:, y, u, u:, e, e:, 2:, E, 3:, O, O:, a, A:, @, {, o, o:, 9, Q, @i, 9y, ai, Oi, @u, au, i@, e@, u@, r, rh\, l, j, w, l',

The subset of the HTK phoneset is the following:

tp\_b, tK\_>, tK\_h, tS, tS\_>, tS\_h, ts\_>, ts\_h, kx\_>, kx\_h, kK\_>, ps\_>, ps\_h, pS\_h, pS\_>, d\_0Z, dz, dB, dz\_a, dz\_ah\_b, dZh\_b, v, T, D, s, z, S, Z, x, h, h\_b, K, K\_b, dK, dK\_b, G, B, p\_b, s\_a, sw, zw, p\_bs, p\_bS, BZ, fS, ɿ\_b, ɿ\_bg, ɿ\_bh, ɿ\_b|\_b, ɿ\_b|\_bg, ɿ\_b|\_bh, ɿ\_b, ɿ\_bg, ɿ\_bh, p, p\_h, p\_>, pj\_>, pj\_h, b, b\_<, bj, t, t\_h, t\_>, tj, tj\_h, d, dj, dh\_b, c\_>, c\_h, J\_b, k, k\_h, k\_>, g, gh\_b, tl\_>, tl\_h, m, n, J, N, n\_a, m\_h, n\_h, m\_j, J\_h, ɿ, i:, y, u, u:, e, e:, eu\_:, E, E\_:, O, O:, a, A:, @, {, o, o:, u\_, Q, @i, u\_y, ai, Oi, @u, au, i@, e@, u@, r, rh\_b, l, j, w, l\_a

Phoneme symbols that match the following regular expression are allowed:

[\\D]\*+

All sound files referenced in the phoneme file should be placed in a *sounds directory*. A maximum of four phoneme categories can be specified in the phoneme file. These categories are user-defined, and are used by the system to organise the phoneme display panels in a way that makes it easier for the verifier to find the correct phoneme.

## 4.2 Grapheme file format

The grapheme file is a text file containing the graphemes used in the source language, with one grapheme per line. Only a single character is currently allowed per grapheme. Diacritic marks are supported in UTF-8 formatted files.

For example:

a  
b  
š

## 4.3 Dictionary file format

### 4.3.1 Native format

The dictionary file is a text file of the following layout for each line:

<transcription><tab><pronunciation><tab><word status><tab><error level>

For example:

bangwe	b a N E	1	0
botha	b O t_> a	5	0

The word transcription only uses graphemes found in the .gra file. Diacritic marks are supported in UTF-8 formatted files.

The pronunciation is captured using phonemes from the .pho file, with individual phonemes separated by spaces.

The word status is assigned by the verifier through the DictionaryMaker interface, and is one of the following numbers which represent the categories defined in section 3:

- 0 – unverified
- 1 – correct
- 2 – invalid
- 3 – uncertain
- 4 – ambiguous
- 5 – proper noun
- 6 – foreign

The error level number is generated by the DictionaryMaker system.

Dictionary entries that match the following regular expression are valid:

```
^([\S]++)([\s]+)([D]*+)([d]++)([\s]+)([d]++)([\s]*)$
```

### 4.3.2 Exported format: HTK

The HTK dictionary file is a text file of the following layout for each line:

<transcription><space><pronunciation>

For example:

<i>mutsendo</i>	<i>m u t s_&gt; i n d O</i>
<i>mutshireledzi</i>	<i>m u t s_&gt; i r E I E d z i</i>

### 4.3.3 Exported format: Festival

The Festival dictionary file is a text file of the following layout for each line:

("<transcription>"< nil > (<pronunciation>))

For example:

<i>("tsikana" nil (ts_&gt; i k_&gt; a n a))</i>
<i>("kona" nil (k_&gt; O n a))</i>

### 4.3.4 Exported format: semicolon-delimited

The dictionary file can also be exported as a semicolon-delimited text file.

## 4.4 Sound file format

The sound files need to be .wav files recorded at a rate of 44 100 bps, sampled as 16 bit mono. Inconsistent sound file formats lead to sound playback problems.

## 4.5 Word list

The Word list file contains a list of words to be verified, with one word per line in the text file.

For example:

bangwe
botha

## 4.6 Project file

The project file contains project information and settings, and is used to launch a project. It is created and maintained by DictionaryMaker.

# 5 Installing and Running DictionaryMaker on Windows

DictionaryMaker is available as both binary and source releases, and a Java Runtime Environment is needed for either of these releases to work.

## 5.1 Java Runtime Environment

DictionaryMaker requires that a Java Runtime Environment, equal to or newer than Sun Java 5.0.10, be installed. Most Windows installations come with this already installed, but if not installed the latest version is available from:

<http://java.com/en/download/index.jsp>

If an older version is required by other applications, JRE and JDK 5.0.10 is available from:

<http://java.sun.com/products/archive/>

On Windows, the Java version can be determined through the Java Control Panel.

## 5.2 Installing the binary release

The latest available binary release for Windows is *dictionarymaker-bin-2.16.beta.zip*. The binary release downloaded from Sourceforge will have to be unzipped/unpacked with an archiving utility into a directory of your choice. When the archive file is unpacked, the following directory structure is created:

```
dictionarymaker-2.16.beta\  
  dictionarymaker-2.16.beta.jar - the DictionaryMaker application  
  run.bat  
  VERSION  
  CONTRIBUTORS  
  INSTALL_WINDOWS.txt  
  LICENSE.txt  
  README.txt  
  DMTutorial\  
    Data\  
      setswana.dict  
      setswana.initial.dict  
      setswana.pho  
      setswana.wdl
```

```
setswana.gra
sounds \ (containing .wav files for the Setswana phonemes)
Documentation\
    dictionarymaker-2.16.beta.tutorial.odt
    dictionarymaker-2.16.beta.manual.odt
release_notes\ (containing current and previous release notes)
```

The file “dictionarymaker-2.16.beta.jar” is the DictionaryMaker application, which is executed by running the batch file “run.bat”.

### 5.3 Running DictionaryMaker

DictionaryMaker has been tested on Windows XP and can be run by executing the created run.bat file in the same way that any other batch file is executed. This batch file executes the Java application, using the applicable Java memory settings.

### 5.4 Building the source release

Note that for most users, the binary release should be sufficient. This section includes information for users who wish to build a new release from source.

The latest available source release for Windows is *dictionarymaker-src-2.16.beta.zip*. To build the “dictionarymaker-2.16.beta.jar”-file:

- Copy “build.properties.example” to “build.properties” and edit it so that the necessary paths are valid for the local computer.
- From the command-line, change directory to the DictionaryMaker folder.
- Run: ant jar

DictionaryMaker will need Java-1.5 to compile the source code, and the build script uses the following:

- Apache Ant v1.6.2
- FindBugs 0.9.1
- Checkstyle v3.5-1
- Quilt v0.45
- java v1.5.0\_06
- Junit v3.8.1-4
- JFCUnit

## 6 Installing and Running DictionaryMaker on Linux

### 6.1 Java Runtime Environment

DictionaryMaker requires that a Java Runtime Environment, equal to or newer than Sun Java 5.0.10, be installed. Most Linux installations come with this already installed, but if not installed, the latest version is available from:

<http://java.com/en/download/index.jsp>

If an older version is required by other applications, JRE and JDK 5.0.10 is available from:

<http://java.sun.com/products/archive/>

On Linux, the Java version can be determined by typing

```
java -version
```

on the command line. The directory in which the Java version that is being used is installed can be determined by typing

```
printenv JAVA_HOME
```

on the command line.

If multiple versions of Java are installed on a unix system, the version to be used can be changed by setting the `java_home` environment variable, again by typing

```
export JAVA_HOME='/opt/jdk1.5.0_10'
```

on the command line.

Once Java is in place, DictionaryMaker can be installed through either of the two releases:

- The binary release provides the application, its documentation and example data.
- The source release provides the Java source code and the Jar file needs to be built from it.

## 6.2 Installing the binary release

The latest available binary release for Linux is *dictionarymaker-bin-2.16.beta.tar.gz*.

The binary release downloaded from Sourceforge will have to be unzipped/unpacked with an archiving utility into a directory of your choice. When the archive file is unpacked, the following directory structure is created:

```
dictionarymaker-2.16.beta\
  dictionarymaker-2.16.beta.jar - the DictionaryMaker application
  run.sh
  VERSION
  CONTRIBUTORS
  INSTALL-LINUX.txt
  LICENSE.txt
  README.txt
  DMTutorial\
    Data\
      setswana.dict
      setswana.initial.dict
      setswana.pho
      setswana.wdl
      setswana.gra
      sounds \ (containing .wav files for the Setswana phonemes)
  Documentation\
    dictionarymaker-2.16.beta.tutorial.odt
    dictionarymaker-2.16.beta.manual.odt
  release_notes\ (containing current and previous release notes)
```

The file “dictionarymaker-2.16.beta.jar” is the DictionaryMaker application, which is executed by running the shell script “run.sh”.

## 6.3 Running DictionaryMaker

DictionaryMaker can be run on Linux in a number of ways:

1. In a console window, change directory to the directory in which the dictionarymaker-2.16.beta.jar is located, and execute the jar with:

```
java -Xms128m -Xmx512m -Dfile.encoding=utf-8 -jar  
dictionarymaker-2.16.beta.jar
```

2. Make the “run.sh” script executable in the directory in which un.sh is located using the chmod command:

```
chmod +x run.sh
```

Then type

```
./run.sh or sh run.sh
```

to run the shell script and let it launch the DictionaryMaker application, using the applicable Java memory settings.

3. Set run.sh to Executable in its File Explorer “Properties” tab, and execute it by double-clicking on it.

## 6.4 Building the source release

The program had been tested on Fedora Core 4, Debian, and Mandriva systems. *DictionaryMaker* will need Java-1.5 to compile the source code, and the build script uses the following:

- Apache Ant v1.6.2
- FindBugs 0.9.1
- Checkstyle v3.5-1
- Quilt v0.45
- java v1.5.0\_06
- Junit v3.8.1-4
- JFCUnit

To build the “\*.jar”-file:

- Copy “build.properties.example” to “build.properties” and edit it so that the necessary paths are valid for the local computer DictionaryMaker is to be run on.
- From the command-line, change directory to the DictionaryMaker folder.
- Run: `ant jar`

## 7 Using DictionaryMaker

This section provides a quick-start guide to using DictionaryMaker. A full tutorial is also available<sup>[8]</sup>. Details of all file formats are specified in section 4.

### 7.1 Starting a new dictionary

Starting a new dictionary is done by starting a new project; which can be found under the “File” menu. You will need to provide the following to create a new project:

- *Name and Location*: Name of the new project and where it will be saved.
- *Graphemes*: The list of graphemes (alphabetical symbols) the source language uses. These can be added manually, or imported from a text-based “\*.gra”-file.

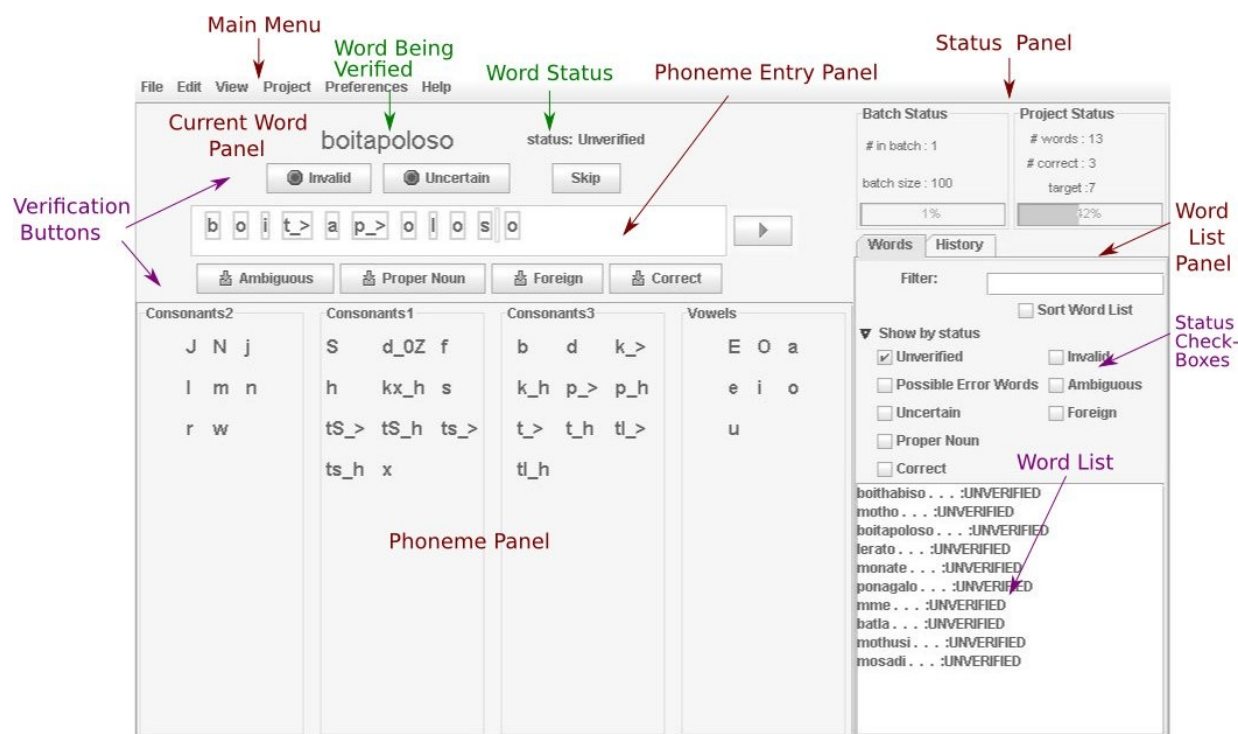
- *Phonemes*: The list of phonemes (vocal sounds) used by the source language. Again, these can be added manually, or imported from a text-based “\*.pho”-file. When adding sounds, however, a phoneme name, category, and the location of the appropriate sound file must be provided:
  - *Categories* are used in DictionaryMaker to sort the available phonemes into groups for ease of use. All phonemes allocated to the same group are displayed together in one panel.
  - The *sound files* enable DictionaryMaker to play back a sound or pronunciation to the verifier, so that the word can be verified. Sound files need to be recorded in a different application and saved separately. (We recommend *Praat*, available at <http://www.fon.hum.uva.nl/praat/>, for this purpose.)
- *Word list/Dictionary*: The words in the project are now defined, using either a raw word list, or a starting dictionary:
  - *Word list*: A list of words which DictionaryMaker will use to bootstrap the new dictionary. These can be added manually, or imported from a text-based “\*.wdl”- or “\*.txt”-file.
  - *Dictionary*: The bootstrapping process can be accelerated by beginning with another, smaller dictionary.

Note again that each grapheme, or word in the above text-based files should be on a line of its own! The phoneme name, file, and category (in that order) must be listed on the same line – but again each phoneme should be on a new line.

## 7.2 Building on an existing project

Choose an existing dictionary under the “File > Open Recent” menu to continue building one of the most recently opened projects, or open the appropriate “\*.proj”-file from the “File > Open” menu.

## 7.3 Building a dictionary



DictionaryMaker has the following layout:

- Current Word Panel:** Displays the current word being added to the dictionary, its current status and the list of phonemes used to pronounce the word. This panel also contains a number of buttons with specific functions:
  - The Play button [▶] allows the verifier to listen to the phonemes in sequence.
  - The Skip button allows the verifier to move on to the next word to be verified without giving a verdict on the current word.
  - The Correct button marks a word as correct – the word is valid, and its displayed pronunciation is correct.
  - The Invalid button marks a word as invalid – the word is not a valid word (e.g. it is a URL), or it is spelled incorrectly, or it is only part of a word.
  - The Uncertain button marks a word as uncertain – the verifier is unable to decide whether the word and/or its pronunciation is valid.
  - The Proper Noun button marks a word as a proper noun – the word is a proper noun.
  - The Foreign button marks a word as foreign – the word is a valid word from a foreign language, but not a word in the source language.
  - The Ambiguous button marks a word as ambiguous – there are multiple valid pronunciations (pronunciation variants, e.g. the present and past tense of the word “read”, pronounced as /r iy d / and /r eh d/ respectively).
- Phoneme Panel:** Displays the set of phonemes DictionaryMaker associates with the source language. These phonemes are divided according to the categories provided when the phoneme-list was added. The phonemes are placed in the list displayed in the Phoneme Entry Panel to correct the predicted pronunciation:



- Right-clicking one of the phoneme labels will play the relevant sound-file.
- Left-clicking a phoneme adds it to the pronunciation list at the current cursor location. The cursor can be moved with the right and left arrow keys on the keyboard, or by moving the mouse over the phoneme entry panel. The backspace key deletes the phoneme to the left of the cursor.
- Phonemes can be dragged to the pronunciation list and dropped in location, and can also be dragged from the list to remove them.
- Left clicking on a phoneme in the pronunciation list pops up a context menu that lets you play, remove or change the phoneme.
- Right clicking on a phoneme in the pronunciation list lets you quickly replace it and gives a drop-down list of phonemes with which it can be replaced.
- *Word List Panel:* The “Words” tab displays the list of words which DictionaryMaker uses to create a dictionary. This list can display all words of a preferred status (“Unverified”, “Correct”, “Uncertain”, etc.) and can also be filtered to display only words that start with specific letters. The option to view previous words in the order they were verified and to correct possible mistakes made during verification, is available under the “History” tab.
- *Status Panel:* The status of the current project and the current batch (if DictionaryMaker is using “Batch” update-mode) are displayed. It shows the size, target-size, and percentage completed in each case.

After having created a new project, the DictionaryMaker system runs through all the unverified words (if a batch-size is specified then it will run through the first batch of words) one-at-a-time.

The verifier then updates or corrects the given pronunciation. Selecting the appropriate verdict button will process the current word and DictionaryMaker will continue with the next word.

## 7.4 Changing grapheme set/phoneme set/word list

The grapheme set, phoneme set and the word list are defined during the creation of the new project.

However, graphemes, phonemes and words may be added, edited or removed during the development of the dictionary and from an existing project. This is done by choosing the “Project > Edit Dictionary” menu item and defining the new grapheme and phoneme sets and word list. Not all changes to the dictionary are allowed (for example a grapheme that is in use may not be removed). The system provides appropriate warnings if illegal changes are attempted.

## 7.5 Importing a dictionary

It is also possible to import an entire DictionaryMaker-format dictionary. When the menu option “Project > Import Dictionary” is used, the imported dictionary is merged with the current one in the following way:

- Words in the imported dictionary that do not occur in the current one are added to the current dictionary, along with their pronunciation and word status.
- Unverified words in the current dictionary are replaced with verified ones from the new dictionary.
- If a word was verified in both dictionaries, but assigned a different word status, the verifier is asked whether or not the word status and pronunciation of the word in the current dictionary should be replaced with that from the new dictionary.

- If a word has been assigned the same status in both dictionaries, the current dictionary's pronunciation is used.

## 7.6 Changing the update and word selection mode

By checking the “Use Custom Settings” under the “Project > Properties” menu item, the user can set the update mode used to extract the rules from newly-verified words:

- *Continuous*: The system continues through the whole list of words. A new rule set is extracted each time a word verified as “correct” is added to the system.
- *Batch*: The system runs through a user-specified number of words, and updates the rule set before continuing with the next batch.

DictionaryMaker also gives the user the opportunity to choose whether the *system* or the *user* should specify the next word to be verified:

- *System*: The next word to be verified is randomly selected from the list of unverified words.
- *User*: The next word to be verified is the next unverified word, as displayed in the word list panel.

These options are also available under the “Preferences > System Defaults” menu item. However, it is important to stress that this second option will set the default options for all future projects, whereas the first will only apply to the current project.

## 7.7 Exporting data

Various types of data may be exported from DictionaryMaker to text-based files, using the “File > Export” menu option. Exporting the dictionary, rule set, graphemes, phonemes, and word list will create files which may be used to create another project which may utilise the same data.

In addition, the dictionary itself can be exported in HTK, Festival or semicolon-delimited format. See section 4 for more detail on file formats.

## 7.8 Analysing a project, dictionary or rule set status

The current project statistics can be viewed under “Project > Statistics”. Here, the number of words can be viewed. The options for viewing the rules, and the number of rules per grapheme (and per grapheme per context size), are available here.

## 7.9 Synchronising the rule set

The rule set of the current project can be updated (by running a full rule extraction) without waiting for the specified batch to fill up. This event can be found under the “Project > Synchronise” menu.

# 8 Future functionality

Functionality likely to be added in the near future:

- Capturing pronunciations for multiple pronunciation variants.
- Creating a dictionary from an existing rule set and a given word list.
- Recording the sound clips from within the DictionaryMaker application.

- Additional tools for manipulating grapheme and phoneme sets, e.g. splitting and merging of phonemes (with subsequent re-bootstrapping of relevant words).
- Determining whether a formant synthesizer works better than our simple concatenative phoneme synthesiser.

If you would like to contribute to this process, please email [bmcalister@users.sourceforge.net](mailto:bmcalister@users.sourceforge.net).

## 9 DictionaryMaker Licensing

DictionaryMaker version 2.16.beta is freely available from <http://dictionarymaker.sourceforge.net> under an Open Source license.

Although the code is free, we offer no warranties. We will continue to develop, fix bugs and answer queries when we can, but are not in a position to guarantee this. A copy of the license is included in the software directory, and is reproduced below:

Copyright © 2009, Meraka Insitute, CSIR, South Africa.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the CSIR nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

1 M. Davel and E. Barnard, "*Bootstrapping for language resource generation*", in Proceedings of the Symposium of the Pattern Recognition Association of South Africa, South Africa, 2003, pp. 97–100.

- 2 M. Davel and E. Barnard, “*Pronunciation Prediction with Default&Refine*”, Computer Speech and Language, vol 22, pp. 374-393, October 2008.
- 3 The Hidden Markov Model Toolkit (HTK) is a portable toolkit for building and manipulating hidden Markov models, and is available at <http://htk.eng.cam.ac.uk/>.
- 4 Festival offers a general framework for building speech synthesis systems, and is available from <http://www.cstr.ed.ac.uk/projects/festival/>.
- 5 M. Davel and E. Barnard, “*The efficient creation of pronunciation dictionaries: human factors in bootstrapping*,” in Proceedings of Interspeech, Jeju, Korea, October 2004, pp. 2797–2800.
- 6 M. Davel and E. Barnard, “*A default-and-refinement approach to pronunciation prediction*,” in Proceedings of the Symposium of the Pattern Recognition Association of South Africa, South Africa, November 2004, pp. 119–123.
- 7 M. Davel and E. Barnard, “*The efficient creation of pronunciation dictionaries: machine learning factors in bootstrapping*”, in Proceedings of Interspeech, Jeju, Korea, October 2004, pp. 2781–2784.
- 8 DictionaryMaker Tutorial for Version 2.1.5.