SpeechSearcher Manual

Jon Dehdari, Tim Weale, Eric Fosler-Lussier, and DJ Hovermale

August 18, 2009

Contents

Chapter 1

Installation

The SpeechSearcher software has been tested and installed on Windows, Mac, and Linux platforms.

1.1 Windows Installation

SpeechSearcher has been verified for Windows XP and Windows Vista. You will need the Buckeye Speech Corpus in order to listen to audio files within SpeechSearcher. Follow the instructions in section 1.5 (below) to ensure directory organization of the corpus has the proper tree structure. The software expects this organization.

1. Download and install Tcl/Tk for Windows at:
   `http://www.activestate.com/activetcl/downloads`

2. Download and unzip Snack for Windows at:
   `http://www.speech.kth.se/snack/download.html`
   Click on `install.tcl`

3. Unzip Iwidgets.
   Place the unzipped folder at `C:\Tcl\lib\iwidgets-4.0.1`

4. Unzip SpeechSearcher, and place `BSC.db` in the same unzipped folder.

5. Click on `speechsearcher.tcl`

1.2 Mac Installation

SpeechSearcher has been verified for OS X 10.4 and 10.5. You will need the Buckeye Speech Corpus in order to listen to audio files within SpeechSearcher. Follow the instructions in section 1.5 (below) to ensure directory organization of the corpus has the proper tree structure. The software expects this organization.

1. Download and install Tcl/Tk for Mac at:
   `http://www.activestate.com/activetcl/downloads`
2. Download and unzip Snack for Mac at:
   `http://www.speech.kth.se/snack/download.html`
   Click on `install.tcl`
3. Unzip Iwidgets.
   Place the unzipped folder under the lib directory of the Tcl installation
4. Unzip SpeechSearcher, and place `BSC.db` in the same unzipped folder.
5. Click on `speechsearcher.tcl`

1.3 Linux Installation

SpeechSearcher has been verified for Red Hat Enterprise 4 and Debian 5. You will need the Buckeye Speech Corpus in order to listen to audio files within SpeechSearcher. Follow the instructions in section 1.5 (below) to ensure directory organization of the corpus has the proper tree structure. The software expects this organization.

Use your package manager (eg. `synaptic`, `apt-get`, `aptitude`, or `yum`) to install the following packages:

- `tcl8.4`
- `tk8.4`
- `iwidgets4`
- `libsnack2`

- libsqlite3-tcl
- libtktable2.9

For example, Debian / Ubuntu users can issue the following command in a shell:

```
sudo apt-get install tcl8.4 tk8.4 unzip iwidgets4 libsnack2 libsqlite3-tcl
libtktable2.9
```

Unzip SpeechSearcher, and download BSC.db into the same unzipped folder.

If you do not have sufficient privileges to install packages, contact your local systems administrator. Version 8.5 of Tcl/Tk may be used instead of version 8.4, although this version has not been thoroughly tested with SpeechSearcher.

Start the application by typing the command "./speechsearcher.tcl" in the directory where SpeechSearcher is located.
If you encounter an error "bad interpreter: No such file or directory", alternatively try the command "wish8.4 speechsearcher.tcl".

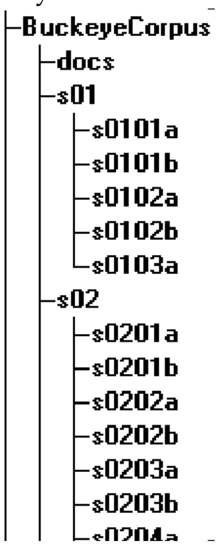
Basic queries may be entered either from the command-line or from the graphical interface, although results are displayed only in the graphical interface. Use the "--help" command-line argument for further details.

1.4 After Installation

After clicking on speechsearcher.tcl to start the application, you will be prompted to tell SpeechSearcher where the speech corpus resides on your computer. If this dialog does not appear, or if you wish to change this setting later, go to "Edit" → "Preferences", then specify the path to the corpus using the "Browse" button. Leave the Wavesurfer configuration file setting alone.

1.5 Corpus Organization

Ensure the corpus is organized properly on your computer. SpeechSearcher expects the corpus to be organized hierarchically, with each speaker (eg., s01) having a separate directory below a main corpus directory, and each speech file for that speaker (eg., s0101a, s0101b, s0102a, etc.) being in a separate subdirectory along with the accompanying files (eg. s0101a.wav, s0101a.words, s0101a.phons, etc). There should be three levels in the hierarchy, as shown in the picture below. With any other arrangement, SpeechSearcher may not work properly.



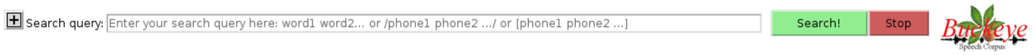
```
├BuckeyeCorpus
│ ├docs
│ ├s01
│ │ ├s0101a
│ │ ├s0101b
│ │ ├s0102a
│ │ ├s0102b
│ │ └s0103a
│ ├s02
│ │ ├s0201a
│ │ ├s0201b
│ │ ├s0202a
│ │ ├s0202b
│ │ ├s0203a
│ │ ├s0203b
│ │ └s0204a
```

The corpus has been reorganized in this way on the Buckeye corpus web site if you would prefer to download it in the preferred format. We apologize for any inconvenience this requirement causes.

Chapter 2

Tutorial

This chapter provides a short tutorial on searching the speech corpus by the orthographic spelling of words (§ 2.1), and by the phonetic transcription of words (§ 2.2). After starting SpeechSearcher, the basic search window will appear.


⊞ Search query:
Enter your search query here: word1 word2... or /phone1 phone2 .../ or [phone1 phone2 ...]
Search!
Stop
Buckeye
Speech Corpus

2.1 Orthographic Searches

To perform a search on the orthographic spelling of words, such as "books", or "los angeles", simply type this into the search field and hit the enter key or click on the "Search!" button. Searches should be lowercase.

2.2 Phonetic Searches

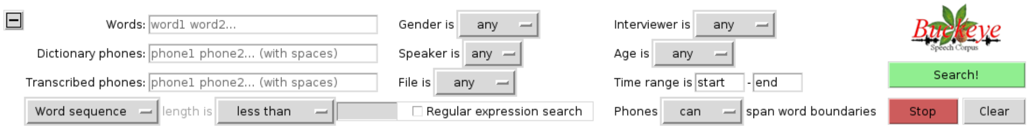
Phonetic searches come in two kinds: dictionary and transcribed. Dictionary searches query the corpus based the canonical, dictionary pronunciation of a word or segment, regardless of the actual pronunciation by speakers. They are surrounded by forward slashes `/.../` . Transcribed searches query the corpus based on the actual pronunciation of a word or segment. See section 3.7 for a description of the phonetic transcription scheme.

For example, to search for /bʊks/, type "`/b uh k s/`" (without quotation marks) into the basic search field and hit the enter key. To search for /tuə/, which could potentially match *to a*, *to another*, *two other*, ..., type "`t uw ah`" into the basic search field.

If we wish to see if anyone in the corpus said [bɔks], type "`[b ao k s]`" into the basic search field.

2.3 Advanced Searches

These basic searches allow us to search word segments and phone segments, but SpeechSearcher also supports much more advanced queries. To enable advanced searches, click on the "**+**" button to the left of the basic search field. You will see the advanced search window:



Words: word1 word2...

Dictionary phones: phone1 phone2... (with spaces)

Transcribed phones: phone1 phone2... (with spaces)

Word sequence — length is less than — □ Regular expression search

Gender is any

Speaker is any

File is any

Interviewer is any

Age is any

Time range is start - end

Phones can span word boundaries

This allows you to search for specific orthographic words that are uttered a specific way, among other things. So if we want to find out if anyone pronounced *books* as [bɔks], we would type "`books`" (without quotation marks) in the field "Words:", and also type "`b ao k s`" (again without quotation marks) in the field "Transcribed phones:". You do not need to surround phonetic searches with either /.../ or [...] in advanced mode.

2.4 Results

Once you have performed a search, you can save various things from this query:

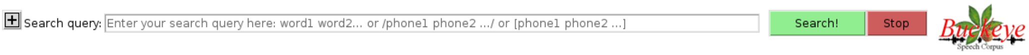
- You can save the formulation of a query to a file that can later be reloaded into SpeechSearcher: File → Save Query
- You can bookmark the formulation of a query: Bookmarks → Bookmark Query
- You can save the results to a file that can later be reloaded into SpeechSearcher: Actions → Result Set → Save Result Set
- You can save the results to a comma-separated-values (CSV) file: Actions → Save Results to CSV
- You can save a subset of the actual corpus, including the sound files, etc.: Actions → Save Resulting Corpus

Chapter 3

Searching

3.1 Simple Queries

Simple queries are done by entering a query string in the default search box:



Search query: Enter your search query here: word1 word2... or /phone1 phone2 .../ or [phone1 phone2 ...]

Simple queries may be done on words, dictionary phones or transcribed phones.

- Words are indicated by typing a word in the search box:
  `an example`

- Dictionary phones are indicated by surrounding each phone with a slash '/':
  `/ae n eh g z ae m p el/`

- Transcribed phones are indicated by surrounding each phone with a brace '[' and ']':
  `[ah n ih g z aen p el]`

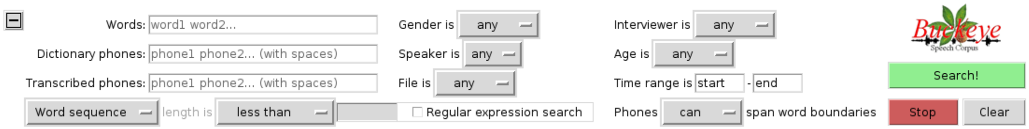
- Simple queries search the entire database for the given query parameters.

- Simple queries may not mix query types. If you want to mix the three types of queries, use the Advanced Queries option.

- OR queries may be used within simple queries

3.2 Advanced Queries

Advanced queries are done by entering a query string in the default search box.



Words: word1 word2...
Dictionary phones: phone1 phone2... (with spaces)
Transcribed phones: phone1 phone2... (with spaces)
Word sequence length is less than
□ Regular expression search
Gender is any
Speaker is any
File is any
Interviewer is any
Age is any
Time range is start - end
Phones can span word boundaries

Advanced queries may be done on words, dictionary phones or transcribed phones.

- Words are indicated by typing a word in the “Words:” search box
- Dictionary phones are indicated by typing a segment in the “Dictionary phones:” box
- Transcribed phones are indicated by typing a segment in the “Transcribed phones:” box
- OR queries may be used within advanced queries
- Regular expression and metacharacter queries, by checking the “Regular expression search” box.

3.2.1 OR Queries

“OR” queries are queries that allow several words or phones to be combined into a single query. These queries may be done in the default search box or the advanced search box.

OR queries are done by using an SQL union on all combinations of the parts of the queries. That is, all OR-ed portions are combined with all non-OR portions of the query. For example:

- **words**: `this OR that`
  - Queries: $\{\text{this} \cup \text{that}\}$
- **words**: `this OR that`, **trans**: `[dh]`
  - Queries: $\{\text{this} \cap [\text{dh}] \cup \text{that} \cap [\text{dh}]\}$
- **words**: `this OR that`, **trans**: `[dh OR ah]`
  - Queries: $\{\text{this} \cap [\text{dh}] \cup \text{this} \cap [\text{ah}] \cup \text{that} \cap [\text{dh}] \cup \text{that} \cap [\text{ah}]\}$

The queries are separated by the literal string “OR”. Strings such as “or” and “Or” will not be recognized.

For more information, see the FAQ.

3.2.2 Regular Expression and Metacharacter Queries

Regular expression queries provide a powerful way of searching corpora, allowing for higher-level descriptions than just literal words or characters. For example, to search for either “car” or “cars”, enter `cars?` in the Words field of the advanced query area (first click the “+” button). Then check the checkbox labeled “Regular expression search” in the advanced query area to enable regular expression searches. The `?` allows the previous character (“s”) to occur either 0 or 1 time.

Pattern	**Description**	**Example query**
`.`	Match **any** character	`b .h t`
`?`	Match the previous character **zero or one** time	`car.?`
`*`	Match the previous character **zero or more** times	`car.*`
`+`	Match the previous character **one or more** times	`car.+`
`{3,7}`	Match the previous character **between** 3 and 7 times	`.{10,12}`
`{3,}`	Match the previous character **at least** 3 times	`.{14,}`
`{,7}`	Match the previous character **no more than** 7 times (including 0)	`.{,2}`
`|`	Match the previous character **or** the subsequent character	`ab|cd`
`(...)`	**Group** the characters within the parentheses	`talk(ed|ing)`
`[ptk]`	Match **either** `p`, `t`, or `k`	`[mb]et`
`[^ptk]`	Match anything **except** `p`, `t`, or `k`	`[^mb]et`
`^a`	Match `a` only at the **beginning** of an entire query	`^met`
`a$`	Match `a` only at the **end** of an entire query	`met$`

Note that regular expression searches require more time to perform than basic searches.

For advanced details, see the Tcl Regular Expression Manual at:
http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm

Metacharacters / Phonological Features

You can also include metacharacters, such as `V` for vowels and `C` for consonants, in your regular expression searches on dictionary/transcribed phones. The vowel class `V` also includes diphthongs and nasalized vowels. These metacharacters are simply predefined regular expressions, so you can search by other distinctive features by formulating your own regular expression.

Remember to separate metacharacters with spaces. Searches that include metacharacters are very time-consuming.
Consonant metacharacters include the following:

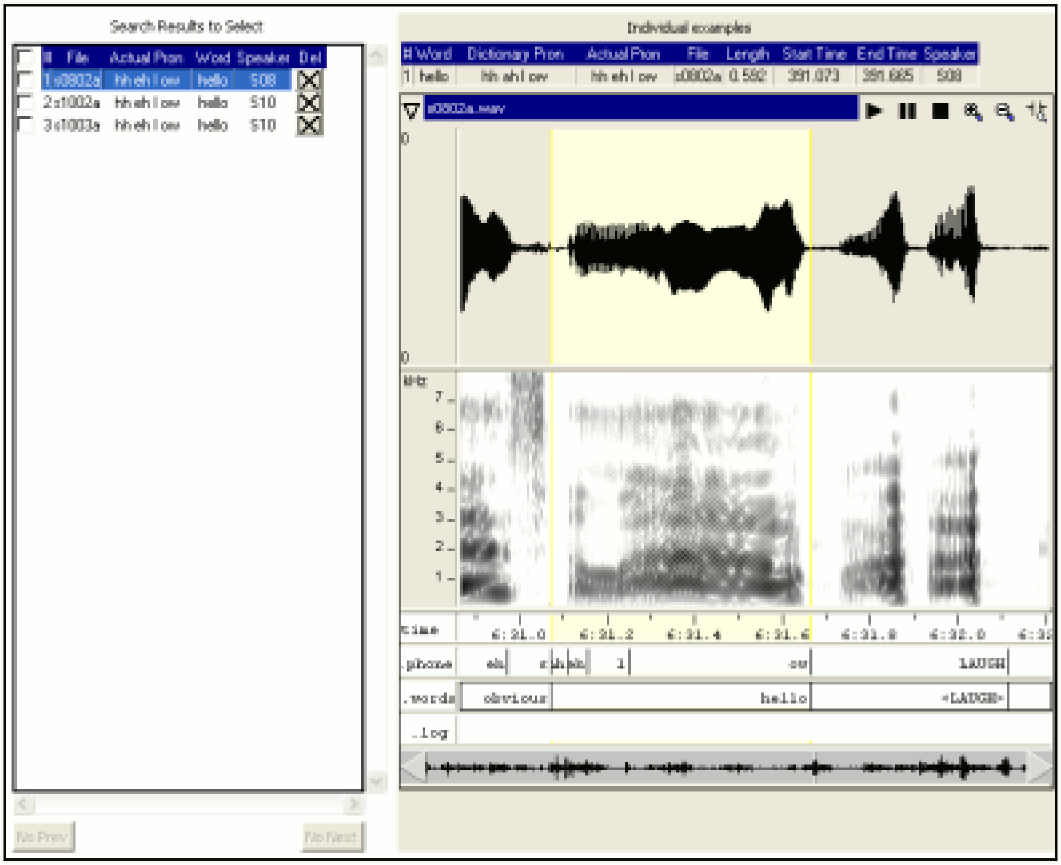
- `C` - All consonants. See section 3.7 for details.
- `P` - Plosives (=stops): `p, t, k, b, d, g, tq`
- `J` - Affricates: `ch, jh`
- `N` - Nasals: `m, n, ng`
- `F` - Fricatives: `f, v, th, dh, s, z, sh, zh, hh`
- `L` - Liquids: `l, el, r, er, ern`
- `Y` - Glides (=semivowels): `w, y`
- `LAB` - Labials: `p, b, m, f, v`
- `COR` - Coronals: `th, dh, t, d, n, s, z, sh, zh, r, er, ern, l, el`
- `DOR` - Dorsals: `y, k, g, ng`
- `LAR` - Laryngeals: `hh`

Vowel metacharacters include the following:

- `V` - All vowels. See section 3.7 for details.
- `DP` - Diphthongs: (± nasal) `ay, ey, oy, aw, ow`
- `LV` - Long vowels: (± nasal) `iy, aa, ao, uw`
- `SV` - Short vowels: (± nasal) `ih, eh, ae, ah, uh`
- `HV` - High vowels: (± nasal) `iy, ih, uh, uw`
- `MV` - Mid vowels: (± nasal) `eh, ah, ao`
- `LWV` - Low vowels: (± nasal) `ae, aa`
- `FV` - Front vowels: (± nasal) `iy, ih, ey, eh, ae`
- `CENV` - Central vowels: (± nasal) `ah`
- `BV` - Back vowels: (± nasal) `uw, uh, ow, ah, aa`
- `ER` - Syllabic consonants: (± nasal) `em, en, el, er`

3.3 Using Query Results

The results of the SpeechSearcher query are displayed below the search fields. There are two parts to the Query Result box. The first is a list of the results returned by the query. Selecting one of the result fields then brings up more detailed information about the individual result, including a widget to allow the playback of the selected utterance. For more information see sections 3.3.1 and 3.3.2 below.



Search Results to Select

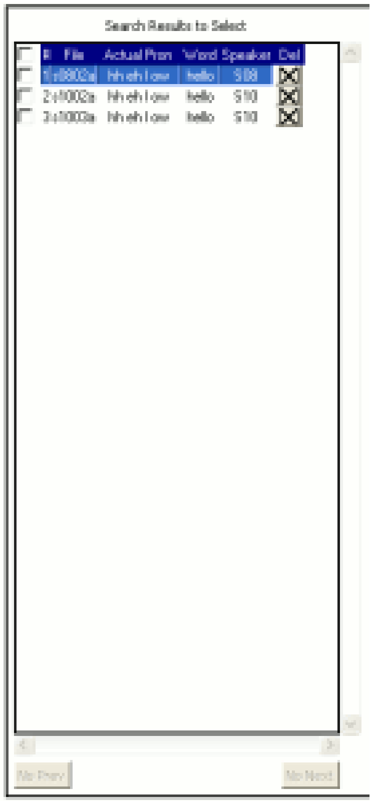
	#	File	Actual Pron	Word	Speaker	Del
□	1	s0802a	hh eh l ow	hello	508	☒
□	2	s1002a	hh eh l ow	hello	510	☒
□	3	s1003a	hh eh l ow	hello	510	☒

Individual examples

#	Word	Dictionary Pron	Actual Pron	File	Length	Start Time	End Time	Speaker
1	hello	hh ah l ow	hh eh l ow	s0802a	0.592	391.073	391.665	508

3.3.1 Query Result List

The query result list contains the results of the SpeechSearcher query.



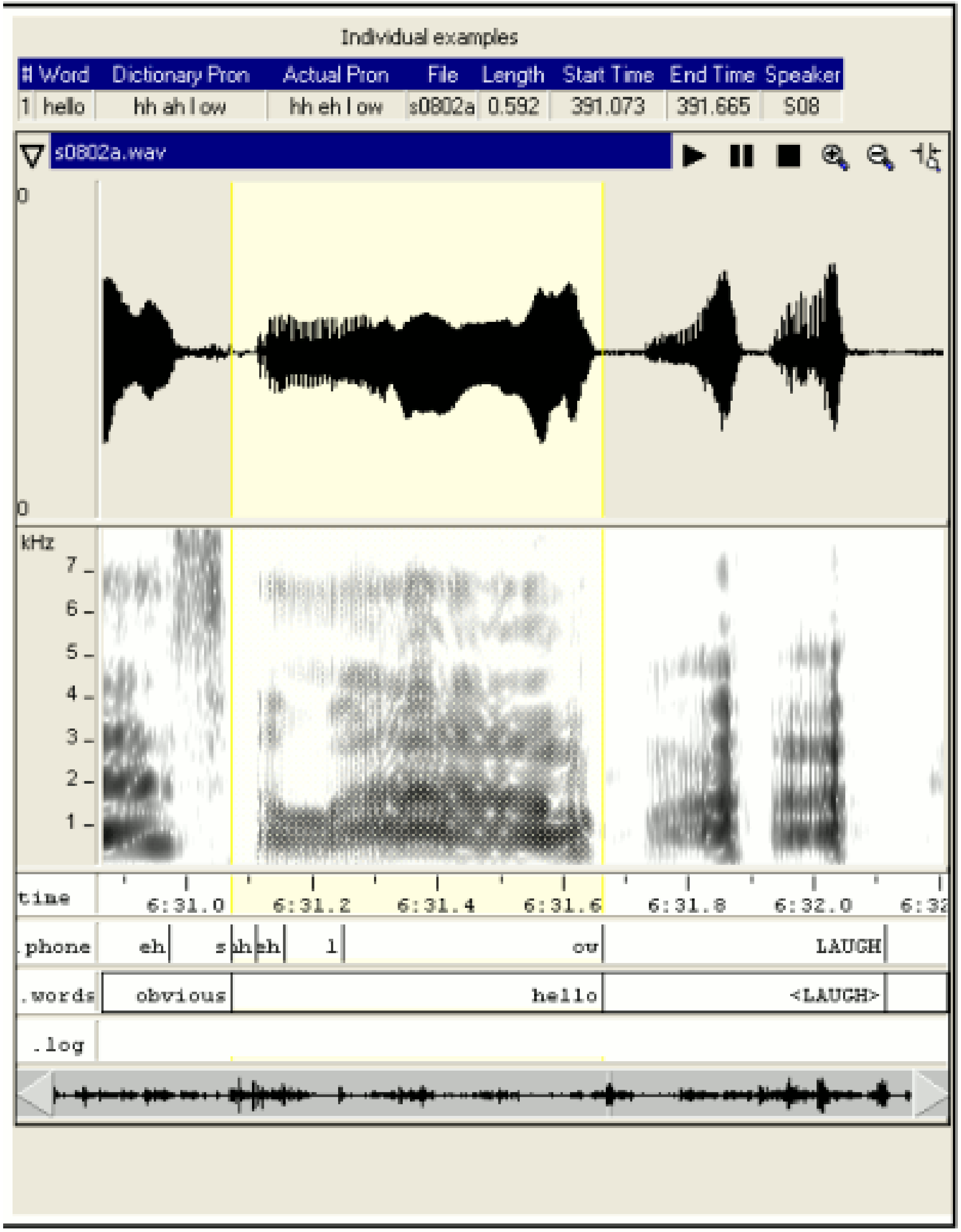
□	#	File	Actual Pron	Word	Speaker	Del
□	1	s0802a	hh eh l ow	hello	S08	☒
□	2	s1002a	hh eh l ow	hello	S10	☒
□	3	s1003a	hh eh l ow	hello	S10	☒

The displayed result columns can be customized via the preferences panel (§ 5).

Clicking on an individual result brings up more detailed information in the Individual Result box (§ 3.3.2).

3.3.2 Individual Result Information

The individual information about an individual utterance from the query can be found in the Individual Result box.



Individual examples

#	Word	Dictionary Pron	Actual Pron	File	Length	Start Time	End Time	Speaker
1	hello	hh ah l ow	hh eh l ow	s0802a	0.592	391.073	391.665	S08

The top portion of the box contains more detailed information about the result while the lower portion contains a WaveSurfer interface to allow starting and stopping of the .wav file.

3.4 Saving Queries

This option saves the query parameters to a file for opening later, which is described in section 3.5. Changes to the result set for the query are not recorded.

In order to save the subsequent result set, choose the "Save Result Set" option, described in section 4.1.2.

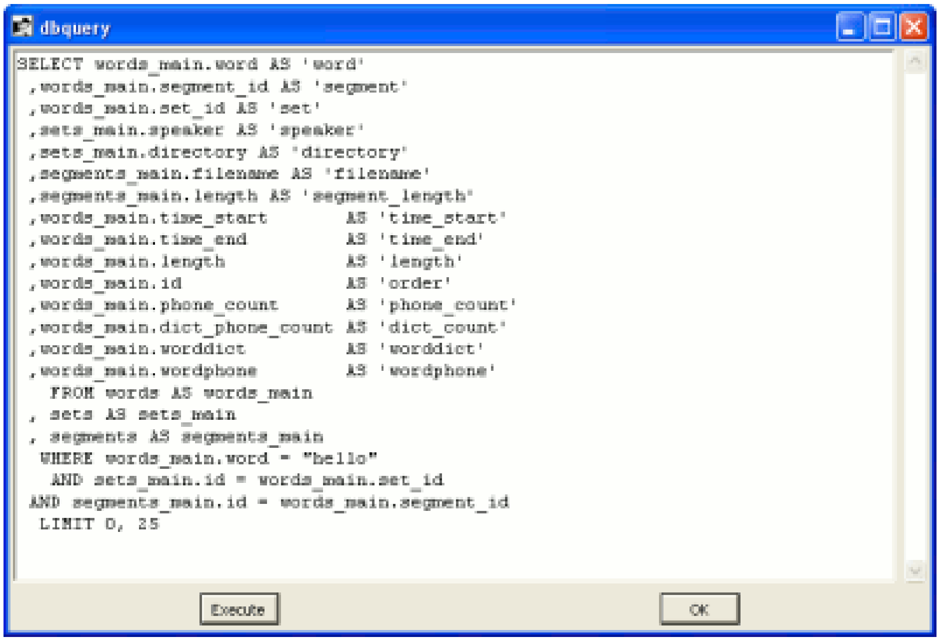
3.5 Opening Queries

Saved queries may be re-run by using the "Open Query..." option in the File menu. This option simply creates a new database query from the parameters in the file, so the execution time may be significant. All previous modifications to the query result set are not recorded.

To open saved result sets without having to re-run the database queries, choose the "Load Result Set" option, which is described in section 4.1.1.

3.6 Viewing Queries

This option allows the user to view the underlying SQL queries to the SpeechSearcher database.



```
SELECT words_main.word AS 'word'
 ,words_main.segment_id AS 'segment'
 ,words_main.set_id AS 'set'
 ,sets_main.speaker AS 'speaker'
 ,sets_main.directory AS 'directory'
 ,segments_main.filename AS 'filename'
 ,segments_main.length AS 'segment_length'
 ,words_main.time_start        AS 'time_start'
 ,words_main.time_end          AS 'time_end'
 ,words_main.length            AS 'length'
 ,words_main.id                AS 'order'
 ,words_main.phone_count       AS 'phone_count'
 ,words_main.dict_phone_count  AS 'dict_count'
 ,words_main.worddict          AS 'worddict'
 ,words_main.wordphone         AS 'wordphone'
   FROM words AS words_main
 ,  sets AS sets_main
 ,  segments AS segments_main
  WHERE words_main.word = "hello"
   AND sets_main.id = words_main.set_id
 AND segments_main.id = words_main.segment_id
  LIMIT 0, 25
```

There are two options available:

View: This option allows for the user to only view the query.

Edit: This option allows the user to modify and also execute the query for a new result set.

3.7 Phonetic Transcription Conventions

3.7.1 Vowels

Buckeye	IPA
aa	ɑ
aan	ɑ̃
ae	æ
aen	æ̃
ah	ə / ʌ
ahn	ə̃ / ʌ̃
ao	ɔ
aon	ɔ̃
aw	aʊ
awn	ãʊ
ay	aɪ
ayn	ãɪ
eh	ɛ
ehn	ɛ̃
el	l̩
em	m̩
en	n̩

Buckeye	IPA
er	ɹ̩
ern	ɹ̩̃
ey	eɪ
eyn	ẽɪ
ih	ɪ
ihn	ɪ̃
iy	i
iyn	ĩ
ow	oʊ
own	õʊ
oy	ɔɪ
oyn	ɔ̃ɪ
uh	ʊ
uhn	ʊ̃
uw	u
uwn	ũ

3.7.2 Consonants

Buckeye	IPA
b	b
ch	ʧ
d	d
dh	ð
dx	ɾ
f	f
g	ɡ
hh	h
jh	ʤ
k	k
l	l / ɫ
m	m
n	n

Buckeye	IPA
ng	ŋ
p	p
r	ɹ
s	s
sh	ʃ
t	t
th	θ
tq	ʔ
v	v
w	w
y	j
z	z
zh	ʒ

3.8 Part-of-Speech Tagging Conventions

3.8.1 Tagset

We used the same set of part-of-speech tags as is used by the Penn Treebank project (Marcus et al., 1993). These tags are listed below:

CC	Coordinating conjunction	PP$	Possessive pronoun
CD	Cardinal number	RB	Adverb
DT	Determiner	RBR	Adverb, comparative
EX	Existential *there*	RBS	Adverb, superlative
FW	Foreign word	RP	Particle
IN	Preposition/sub. conj.	SYM	Symbol (math. or scientific)
JJ	Adjective	TO	*to*
JJR	Adjective, comparative	UH	Interjection
JJS	Adjective, superlative	VB	Verb, base form
LS	List item Marker	VBD	Verb, past tense
MD	Modal	VBG	Verb, gerund/pres. part.
NN	Noun, singular or mass	VBN	Verb, past participle
NNS	Noun, plural	VBP	Verb, non-3rd sing. present
NNP	Proper Noun, singular	VBZ	Verb, 3rd sing. present
NNPS	Proper Noun, plural	WDT	*wh*-determiner
PDT	Predeterminer	WP	*wh*-pronoun
POS	Possessive ending	WP$	Possessive *wh*-pronoun
PRP	Personal pronoun	WRB	*wh*-adverb

The Buckeye Speech Corpus poses some unique challenges to part-of-speech tagging, and the tagset was modified to account for these challenges. First of all, the transcription conventions of the speech corpus do not include punctuation of any kind. Thus there are no tags for punctuation. Secondly, there are many comments and annotations inside brackets within the corpus. These are assigned a "NULL" tag. Because the corpus is organized into acoustic chunks it was desirable to tag the entire acoustic chunk rather than tag the component words separately in cases such as contractions and what the transcription conventions refer to as "collocations". This resulted in the creation of compound tags, which consist of the Penn Treebank tags for the individual words connected by an underscore. A list of the additional tags which were used to annotate the corpus along with typical words which were assigned these tags follows:

DT_VBZ	that's
EX_VBZ	there's
NULL	comments in <...> or {...}
PRP_MD	I'll, you'll, we'll, he'll, she'll, it'll, they'll, I'd, you'd, we'd, he'd, she'd, it'd, they'd
PRP_VBP	I'm, you're, we're, they're, I've, you've, we've, they've, yknow
PRP_VBZ	it's, he's, she's
VBG_TO	gonna
VBP_RB	don't, didn't, can't, haven't, weren't
VBP_TO	wanna
VBZ_RB	doesn't, hasn't, isn't
WP_VBZ	who's
WP_RB	wouldn't, shouldn't, couldn't, won't

3.8.2 Automated Tagging Method

With roughly 400 000 acoustic chunks to be tagged, it was necessary to automate the tagging process. We created two hand-tagged training sets of 2 000 acoustic chunks each and one test set of 3 000 acoustic chunks from two different speakers. We used the C&C part-of-speech tagger (Curran & Clark, 2003). Several pre-processing and post-processing steps were taken, since written English differs greatly from the transcribed spoken English of the corpus, which has repeated words, no punctuation, no capitalization, and bracketed text, among other differences. Before running the tagger on the corpus, we introduced capitalization. We then simulated utterance boundaries by creating a new line before and after each piece of bracketed information in the corpus. The maximum length of these utterances was set to 15 acoustic chunks. After running the tagger on the resulting corpus, we assigned the `NULL` tag to the bracketed information and the compound tags to the contractions and 'collocations'. We achieved roughly 91% accuracy on development sets, and 90% accuracy on the test set. The entire corpus was then tagged using this procedure.

Chapter 4

Exporting

4.1 Result Set Actions

Result sets contain the list of results returned by a query. They are able to be saved (§ 4.1.2), loaded (§ 4.1.1), or reset (§ 4.1.3).

4.1.1 Load Result Set

The "Load Result Set" takes a previously saved result set (§ 4.1.2) and loads it into the result list. Loaded result sets are limited to the results found in the saved result set and cannot make use of previous and next commands available when a traditional query is run.

Loaded result sets are not subject to the query maximum found in the preferences. That is, if the maximum number of queries returned is traditionally 25 and the saved result set contains 100 results, all 100 results will be displayed.

This command does not load the query used to generate the result set. For that, use the "Open Query" command (§ 3.5).

4.1.2 Save Result Set

There are two options when saving a result set:

Save All: Saves all results in the result set for later recall

Save Marked: Saves only those results that are marked for later recall

When saving a result set, the resulting file is a Tcl script that will re-create the effect of the query and any subsequent changes you have made to the result set. The saved result set may be re-loaded at a future time by the "Load Result Set" command (§ 4.1.1).

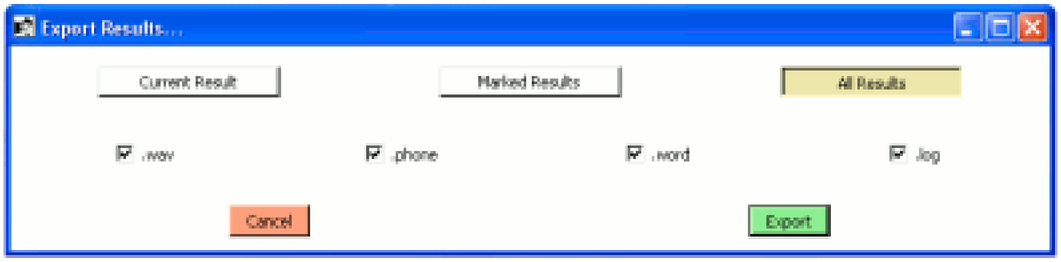
This command does not save the query used to get the result set. For that, use the "Save Query" command (§ 3.4).

4.1.3 Restore Result Set

This command restores the result set to its initial state immediately after the query's execution. All changes and modifications to the result set list are discarded.

4.2 Export Results

The Export Results menu option allows SpeechSearcher to export the results of the query as one or more of several file types.



☑ .wav ☑ .phone ☑ .word ☑ .log

4.2.1 Export scopes

Current Result: Export only the result found in the "Individual Result" (§ 3.3.2) portion of the SpeechSearcher interface. If no result is found in the "Individual Result" portion, SpeechSearcher will give an error message.

Marked Results: Export only those results marked in the Result List (§ 3.3.1)

All Results: Export all results in the Result List (§ 3.3.1)

4.2.2 File types available for exporting

.wav: A playable file format

.phone: A list of phones in the exported segment

.word: A list of words in the exported segment

.log: A log of the exporting procedure

4.2.3 Export Naming Conventions

`%u`: speaker

`%s`: start time

`%e`: end time

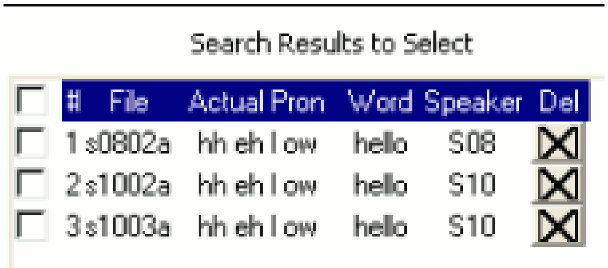
`%q`: query terms

`%x`: extension

4.3 CSV Export

This option exports the current search results as a comma-separated file (`.csv`). Fields exported are based on the displayed fields in the List Result Box.

For example, the results:



Search Results to Select

☐	#	File	Actual Pron	Word	Speaker	Del
☐	1	s0802a	hh eh l ow	hello	S08	☒
☐	2	s1002a	hh eh l ow	hello	S10	☒
☐	3	s1003a	hh eh l ow	hello	S10	☒

Get exported as:

```
s0802a, {hh ee l ow}, hello, S08
s1002a, {hh ee l ow}, hello, S10
s1003a, {hh ee l ow}, hello, S10
```

Chapter 5

Preferences

5.1 General

Corpus Directory: This points to the root of the SpeechSearcher directory and is initialized during installation.

Wavesurfer Config File: This is the location of the config file for Wavesurfer.

5.2 Search

Number of Saved Queries: The number of queries saved in the history for each session.

Maximum Number of Results: The maximum number of results returned for each query. Used to limit the number of results in order to avoid inordinately long queries.

Displayed Result Fields: Specifies which result fields are displayed in the left-hand result box.

5.3 Export

Exported Offset Time: The offset time for exporting.

Chapter 6

Frequently Asked Questions

How can I return the results of my `OR` query in the order of files?

While SQLite indicates that we should be able to utilize an ORDER BY command in our query, we have been unable to get it to recognize the command at this time. Until this problem is resolved, we return the results of the combined queries as a unification of the two separate queries.

Part of Speech Tagging

DJ Hovermale

August 10, 2009

0.1 Part-of-Speech Tagging Conventions

0.1.1 Tagset

We used the same set of part-of-speech tags as is used by the Penn Treebank project (Marcus et al., 1993). These tags are listed below:

CC	Coordinating conjunction	PP$	Possessive pronoun
CD	Cardinal number	RB	Adverb
DT	Determiner	RBR	Adverb, comparative
EX	Existential *there*	RBS	Adverb, superlative
FW	Foreign word	RP	Particle
IN	Preposition/sub. conj.	SYM	Symbol (math. or scientific)
JJ	Adjective	TO	*to*
JJR	Adjective, comparative	UH	Interjection
JJS	Adjective, superlative	VB	Verb, base form
LS	List item Marker	VBD	Verb, past tense
MD	Modal	VBG	Verb, gerund/pres. part.
NN	Noun, singular or mass	VBN	Verb, past participle
NNS	Noun, plural	VBP	Verb, non-3rd sing. present
NNP	Proper Noun, singular	VBZ	Verb, 3rd sing. present
NNPS	Proper Noun, plural	WDT	*wh*-determiner
PDT	Predeterminer	WP	*wh*-pronoun
POS	Possessive ending	WP$	Possessive *wh*-pronoun
PRP	Personal pronoun	WRB	*wh*-adverb

The Buckeye Speech Corpus poses some unique challenges to part-of-speech tagging, and the tagset was modified to account for these challenges. First of all, the transcription conventions of the speech corpus do not include punctuation of any kind. Thus there are no tags for punctuation. Secondly, there are many comments and annotations inside brackets within the corpus. These are assigned a "NULL" tag. Because the corpus is organized into acoustic chunks it was desirable to tag the entire acoustic chunk rather than tag the component words separately in cases such as contractions and what the transcription conventions refer to as "collocations". This resulted in the creation of compound tags, which consist of the Penn Treebank tags for the individual words connected by an underscore. A list of the additional tags which were used to annotate the corpus along with typical words which were assigned these tags follows:

DT_VBZ	that's
EX_VBZ	there's
NULL	comments in <...> or {...}
PRP_MD	I'll, you'll, we'll, he'll, she'll, it'll, they'll, I'd, you'd, we'd, he'd, she'd, it'd, they'd
PRP_VBP	I'm, you're, we're, they're, I've, you've, we've, they've, yknow
PRP_VBZ	it's, he's, she's
VBG_TO	gonna
VBP_RB	don't, didn't, can't, haven't, weren't
VBP_TO	wanna
VBZ_RB	doesn't, hasn't, isn't
WP_VBZ	who's
WP_RB	wouldn't, shouldn't, couldn't, won't

0.1.2 Automated Tagging Method

With roughly 400 000 acoustic chunks to be tagged, it was necessary to automate the tagging process. We created two hand-tagged training sets of 2 000 acoustic chunks each and one test set of 3 000 acoustic chunks from two different speakers. We used the C&C part-of-speech tagger (Curran & Clark, 2003). Several pre-processing and post-processing steps were taken, since written English differs greatly from the transcribed spoken English of the corpus, which has repeated words, no punctuation, no capitalization, and bracketed text, among other differences. Before running the tagger on the corpus, we introduced capitalization. We then simulated utterance boundaries by creating a new line before and after each piece of bracketed information in the corpus. The maximum length of these utterances was set to 15 acoustic chunks. After running the tagger on the resulting corpus, we assigned the NULL tag to the bracketed information and the compound tags to the contractions and 'collocations'. We achieved roughly 91% accuracy on development sets, and 90% accuracy on the test set. The entire corpus was then tagged using this procedure.