ISCAN Documentation

Release 0.0.1

Michael McAuliffe

Aug 22, 2021

Contents:

CHAPTER 1

Introduction

If you are a new ISCAN user, please go directly to *ISCAN Tutorials*.

ISCAN (Integrated Speech Corpus ANalysis) is a tool to manage and analyze corpora on a single server with support for multiple clients enriching and querying the corpora as needed. The ISCAN server contains tools to install/start/stop the databases that PolyglotDB (the Python API) uses.

Additionally, the ISCAN server is used to manage multiple different Polyglot databases, so users can isolate corpora as needed, and start and stop databases as they need access to them.

CHAPTER 2

ISCAN Tutorials

The ISCAN system is a system for going from a raw speech corpus to a data file (CSV) ready for further analysis (e.g. in R), which conceptually consists of a pipeline of four steps:

1. **Importing the corpus into ISCAN** Result: a structured database of linguistic objects (words, phones, sound files).
2. **Enriching the database** Result: Further linguistic objects (utterances, syllables), and information about objects (e.g. speech rate, word frequencies).
3. **Querying the database** Result: A set of linguistic objects of interest (e.g. utterance-final word-initial syllables),
4. **Exporting the results** Result: A CSV file containing information about the set of objects of interest

2.1 Preliminaries

2.1.1 Access

Before you can begin the tutorial, you will need access to log in to the ISCAN server via your web browser. To log in to the McGill ISCAN server via your web browser visit https://roquefort.linguistics.mcgill.ca, press the 'Log in' button on the top right of the screen and enter the username and password provided.

NWAV 2018 Workshop: the workshop organizers will give you a username and password.

After NWAV 2018 Workshop: Please contact savanna.willerton@mail.mcgill.ca to request access to one of the ISCAN tutorial accounts.

2.1.2 Questions, Bugs, Suggestions

If at any point while using ISCAN you get stuck, have a question, encounter a bug (like a button which doesn't work), or you see some way in which you believe the user interface could be improved to make usage more clear/smooth/straightforward/etc, then please file an issue on the ISCAN GitHub repository.

There is also a Slack channel you can join if you have quick questions or would like real-time help. Please contact savanna.willerton@mail.mcgill.ca for access.

2.1.3 Dataset

These tutorials use a tutorial corpus, which is (as of Oct 15, 2018) a small subset of ICE-Canada containing speech from all "S2A" files (two male Canadian English speakers). These files can be downloaded from the ICE Canada site, but doing so is not necessary for these tutorials!

2.2 Tutorial 1: Polysyllabic shortening

2.2.1 Motivation

Polysyllabic shortening refers to the "same" rhythmic unit (syllable or vowel) becoming shorter as the size of the containing domain (word or prosodic domain) increases. Two classic examples:

- English: stick, sticky, stickiness (Lehiste, 1972)
- French: pâte, pâté, pâtisserie (Grammont, 1914)

Polysyllabic shortening is often – but not always – defined as being restricted to accented syllables. (As in the English, but not the French example.) Using ISCAN, we can check whether a simple version of polysyllabic shortening holds in the tutorial corpus, namely:

- Considering all utterance-final words, does the initial vowel duration decrease as word length increases?

2.2.2 Step 1: Import

This tutorial will use the tutorial corpus available for you, available under the title 'iscan-tutorial-X' (where X is a number). The data for this corpus was parsed using the Montreal Forced Aligner, with the result being one Praat TextGrid per sound file, aligned with word and phone boundaries. These files are stored on a remote server, and so do not require you to upload any audio or TextGrid files.

The first step of this analysis is to create a *Polyglot DB* object of the corpus which is suitable for analysis. This is performed in two steps:

- *Importing* the dataset using ISCAN, using the phone, word, and speaker information contained in the corpus
- *Enriching* the dataset to include additional information about (e.g., syllables, utterances), as well as properties about these objects (e.g., speech rate)

To import the corpus into ISCAN, select 'iscan-tutorial-x' corpus (replacing "x" with the number you're using) from the dropdown menu under the 'Corpora' tab in the navigation bar. Next, click the 'Import' button. This will import the corpus into ISCAN and return a structured database of objects: words, phones, and sound files), that will be interacted with in the following steps.

2.2.3 Step 2: Enrichment

Now that the corpus has been imported as a database, it is now necessary to *enrich* the database with information about linguistic objects, such as word frequency, speech rate, vowel duration, and so on. You can see the Enrichment page to learn more about what enrichments are possible, but in this tutorial we will just use a subset.

First, select the 'iscan-tutorial-x' under the 'Corpora' menu, which presents all of the current information available for this specific corpus. To start enrichment, click the 'create, edit, and run enrichments' button. This page is referred to as the *Enrichment view*. At first, this page will contain an empty table - as enrichments are added, this table will be populated to include each of these enrichment objects. On the right hand side of the page are a list of new enrichments that can be created for this database.

Here, we will walk through each enrichment that is necessary for examining vowel duration to address our question ("Considering all utterance final…").

Syllables

Syllables are encoded in two steps. First, the set of syllabic segments in the phonological inventory have to be specified. To encode the syllabic segments:

1. Select 'Phone Subset' button under the 'Create New Enrichments' header
2. Select the 'Select Syllabics' preset option
3. Name the environment 'syllabics'
4. Select 'Save subset'

This will return you to the Enrichment view page. Here, press the 'Run' button listed under 'Actions'. Once syllabic segments have been encoded as such, you can encode the syllables themselves.

1. Under the 'Annotation levels' header, press the 'Syllables' button
2. Select *Max Onset* from the Algorithm dropdown menu
3. Select *syllabics* from the Phone Subset menu
4. Name the enrichment 'syllables'
5. Select 'Save enrichment'

Upon return to the Enrichment view, hit 'Run' on the new addition to the table.

Speakers

To enrich the database with speaker information:

1. Select the 'Properties from a CSV' option
2. Select 'Speaker CSV' from the 'Analysis' dropdown menu. The CSV for speaker information is available for download.
3. Upload the tutorial corpus 'speaker_info.csv' file from your local machine.
4. Select 'Save Enrichment' and then 'Run' from the Enrichment view.

Lexicon

As with the speaker information, lexical information can be uploaded in an analogous way. Download the Lexicon CSV, for the tutorial corpus, select 'Lexicon CSV' from the dropdown menu, save the enrichment, and run it.

Utterances

For our purposes, we define an utterance as a stretch of speech separated by pauses. So now we will specify a minimum duration of pause that separates utterances (150ms is typically a good default).

First, select 'pauses' from 'Annotation levels', and select '<SIL>' as the unit representing pauses. As before, select 'Save enrichment' and then 'run'.

With the positions of pauses encoded, we are now able to encode information about utterances:

1. Under the 'Annotation levels' header, select 'utterances'.
2. Name the new addition 'utterance'
3. Enter *150* in the box next to 'Utterance gap(ms)'

4. Select ‘Save enrichment’, and then ‘Run’ in the Enrichment view.

Speech rate

To encode speech rate information, select ‘Hierarchical property’ from the Enrichment view. This mode allows you to encode rates, counts or positions, based on certain hierarchical properties (e.g., utterances, words). (For example: number of syllables in a word; Here select the following attributes:

1. Enter “speech_rate” as the property name
2. From the Property type menu, select *rate*
3. From the Higher annotation menu, select *utterance*
4. From the Lower annotation menu, select *syllable*

And then, as with previous enrichments, select ‘Save enrichment’ and then run.

Stress

Finally, to encode the stress position within each word:

- Select ‘Stress from word property’ from the Enrichment view menu.
- From the ‘wordproperty’ dropdown box, select ‘stress_pattern’.
- Select ‘Save enrichment’ and run the enrichment in the Enrichment view.

2.2.4 Step 3: Query

Now that the database has been enriched with all of the properties necessary for analysis, it is now necessary to construct a **query**. Queries enable us to search the database for particular set of linguistic objects of interest.

First, return to the Corpus Summary view by selecting ‘iscan-tutorial-x’ from the top navigation header. In this view, there is a series of property categories which you can navigate through to add filters to your search.

In this case, we want to make a query for:

- Word-initial stressed syllables
- only in words at the end of utterances (fixed prosodic position)

Here, find the selection titled ‘Syllables’ and select ‘New Query’. To make sure we select the correctly positioned syllables, apply the following filters:

Under **syllable** properties:

- Left aligned with: *word*
- Under ‘add filter’, select ‘stress’ from the dropdown box, and enter ‘1’ in the text box. This will only select syllables with primary stress in this position.

Under **word** properties:

- Right aligned with: *utterance*

Warning: Note that if right alignment with utterances is specified for syllables in this query, this will inadvertently restrict the query to monosyllabic words, as aligning with a higher linguistic type (in this case, utterances) implicitly aligns it to an intermediate linguistic type (in this case, words).

Provide a name for this query (e.g., ‘syllable_duration’) and select ‘Save and run query’.

2.2.5 Step 4: Export

This query has found all word-initial stressed syllables for words in utterance-final position. We now want to export information about these linguistic objects to a CSV file. We want it to contain everything we need to examine how vowel duration (in seconds) depends on word length. Here we may check all boxes which will be relevant to our later analysis to add these columns to our CSV file. The preview at the bottom of the page will be updated as we select new boxes:

1. Under the **SYLLABLE** label, select:
   - label
   - duration
2. Under the **WORD** label, select:
   - label
   - begin
   - end
   - num_syllables
   - stress_pattern
3. Under the **UTTERANCES** label, select:
   - speech_rate
4. Under the **SPEAKER** label, select:
   - name
5. Under the **SOUND FILE** label, select:
   - name

Once you have checked all relevant boxes, select 'Export to CSV'. Your results will be exported to a CSV file on your computer. The name will be the one you chose to save plus "export.csv". In our case, the resulting file will be called "syllable_duration export.csv".

2.2.6 Examining & analysing the data

Now that the CSV has been exported, it can be analyzed to address whether polysyllabic shortening holds in the tutorial corpus. This part does not involve ISCAN, so it's not necessary to actually carry out the steps below unless you want to (and have R installed and are familiar with using it).

In **R**, load the data as follows:

```
library(tidyverse)
dur <- read.csv('syllable_duration export.csv')
```

You may need to first install the *tidyverse* library using `install.packages('tidyverse')`. If you are unable to install tidyverse, you may also use `library(ggplot2)` instead (note: if you do this, please use `subset()` instead of `filter()` for the remaining steps).

First, by checking how many word (types) there are for each number of syllables in the CSV, we can see that only 1 word has 4 syllables:

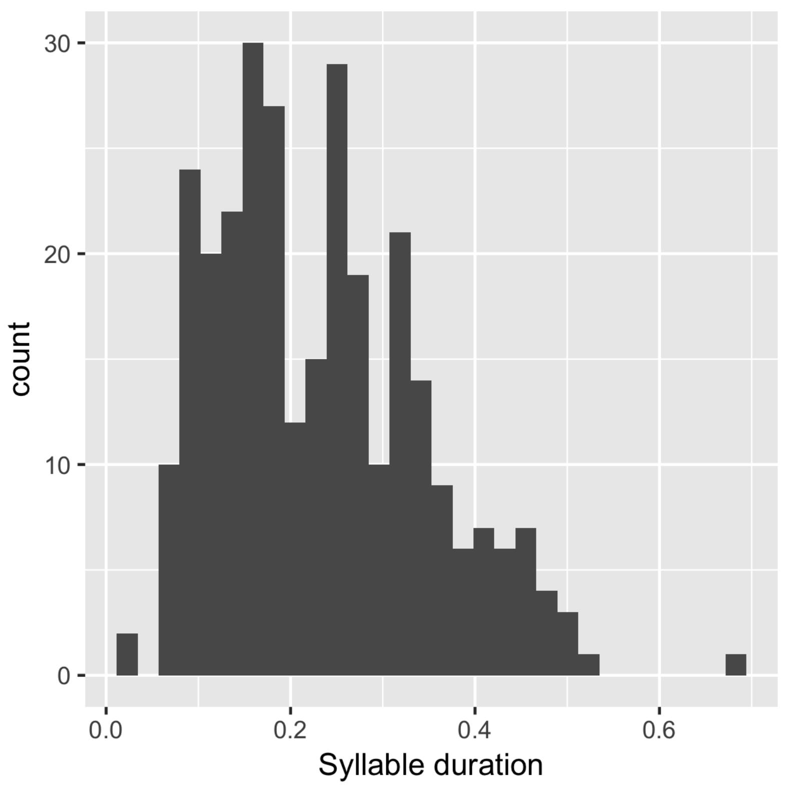
```
group_by(dur, word_num_syllables) %>% summarise(n_distinct(word_label))

#       word_num_syllables `n_distinct(word_label)`
#                    <int>                    <int>
# 1                      1                      109
# 2                      2                       34
# 3                      3                        7
# 4                      4                        1
```

We remove the word with 4 syllables, since we can't generalize based on one word type:

```
dur <- filter(dur, word_num_syllables < 4)
```

Similarly, it is worth checking the distribution of syllable durations to see if there are any extreme values:

```
ggplot(dur, aes(x = syllable_duration)) +
geom_histogram() +
xlab("Syllable duration")
```

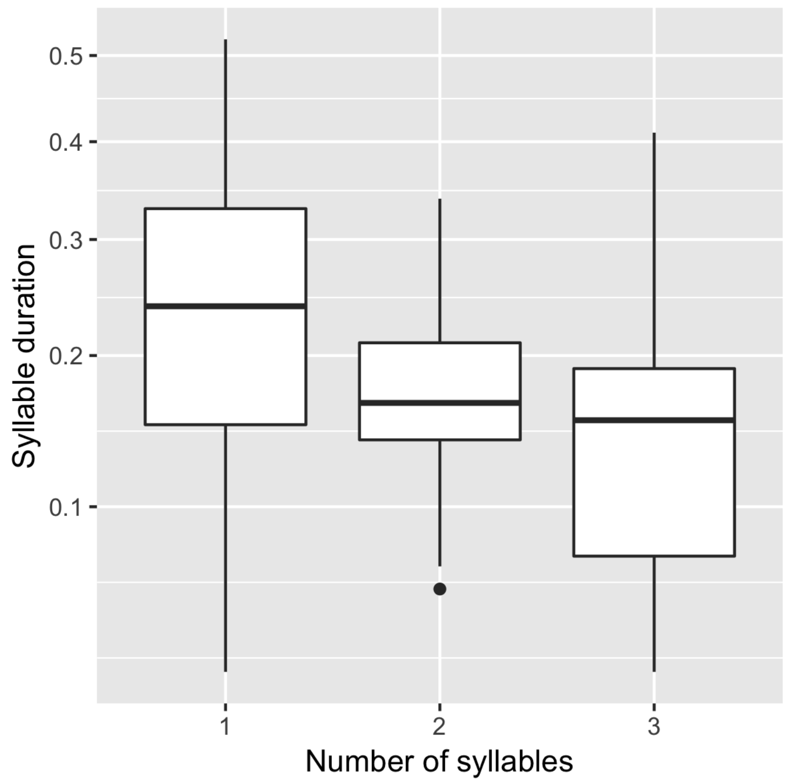


As we can see here, there is one observation which appears to be some kind of outlier, which perhaps are the result of pragmatic lengthening or alignment error. To exclude this from analysis:

```
dur <- filter(dur, syllable_duration < 0.6)
```

Plot of the duration of the initial stressed syllable as a function of word duration (in syllables):

```
ggplot(dur, aes(x = factor(word_num_syllables), y = syllable_duration)) +
geom_boxplot() +
xlab("Number of syllables") + ylab("Syllable duration") +
scale_y_sqrt()
```



Here it's possible to see that there is a consistent shortening effect based on the number of syllables in the word, where the more syllables in a word the shorter the initial stressed syllable becomes.

2.3 Tutorial 2: Vowel formants

Vowel quality is well known to vary according to a range of social and linguistic factors (Labov, 2001). The precursor to any sociolinguistic and/or phonetic analysis of acoustic vowel quality is the successful identification, measurement, extraction, and visualization of the particular vowel variables for the speakers under consideration. It is often useful to consider vowels in terms of their overall patterning together in the acoustic vowel space.

In this tutorial, we will use ISCAN to measure the first and second formants for the two speakers in the tutorial corpus, for the following vowels (keywords after Wells, 1982): FLEECE, KIT, FACE, DRESS, TRAP/BATH, PRICE, MOUTH, STRUT, NURSE, LOT/PALM, CLOTH/THOUGHT, CHOICE, GOAT, FOOT, GOOSE. We will only consider vowels whose duration is longer than 50ms, to avoid including reduced tokens. This tutorial assumes you have completed the *import* and *enrichment* sections from the previous tutorial, and so will only include the information specific to analysing formants.

2.3.1 Step 1: Enrichment

Stressed vowels

First, the set of stress vowels in the phonological inventory have to be specified. To encode these:

1. Select 'Phone Subset' button under the 'Create New Enrichments' header
2. Select the 'Select Stressed Vowels' preset option
3. Name the environment 'stressed_vowels'

4. Select 'Save subset'

This will return you to the Enrichment view page. Here, press the 'Run' button listed under 'Actions'.

Acoustics

Now we will compute vowel formants for all stressed syllables using an algorithm similar to FAVE.

For this last section, you will need a vowel prototype file. For the purposes of this tutorial, the file for the tutorial corpus is provided here:

ISCAN_Prototypes

Please save the file to your computer.

From the Enrichment View, under the 'Acoustics' header, select 'Formant Points'. As usual, this will bring you to a new page. From the **Phone class** menu, select *stressed_vowels*. Using the 'Choose Vowel Prototypes CSV' button, upload the ICECAN_prototypes.csv file you saved. For **Number of iterations**, type 3 and for **Min Duration (ms)** type 50ms.

Finally, hit the 'Save enrichment' button. Then click 'Run' from the Enrichment View.

2.3.2 Step 2: Query

The next step is to search the dataset to find a set of linguistic objects of interest. In our case, we're looking for all stressed vowels, and we will get formants for each of these. Let's see how to do this using the **Query view**.

First, return to the 'iscan-tutorial-x' Corpus Summary view, then navigate to the 'Phones' section and select **New Query**. This will take you to a new page, called the Query view, where we can put together and execute searches. In this view, there is a series of property categories which you can navigate through to add filters to your search. Under 'Phone Properties', there is a dropdown menu with search options labelled 'Subset'. Select 'stressed_vowels'. You may select 'Add filter' if you would like to see more options to narrow down your search.

The selected filter settings will be saved for further use. It will automatically be saved as 'New phone query', but let's change that to something more memorable, say 'Tutorial corpus Formants'. When you are done, click the 'Save and run query' button. The search may take a while, especially for large datasets, but should not take more than a couple of minutes for this small subset of the ICE-Can corpus we're using for the tutorials.

2.3.3 Step 3: Export

Now that we have made our query and extracted the set of objects of interest, we'll want to export this to a CSV file for later use and further analysis (i.e. in R, MatLab, etc.)

Once you hit 'Save query', your search results will appear below the search window. Since we selected to find all stressed vowels only, a long list of phone tokens (every time a stressed vowel occurs in the dataset) should now be visible. This list of objects may not be useful to our research without some further information, so let's select what information will be visible in the resulting CSV file using the window next to the search view.

Here we may check all boxes which will be relevant to our later analysis to add these columns to our CSV file. The preview at the bottom of the page will be updated as we select new boxes:

Under the Phone header, select:

- label
- begin
- end
- F1

- F2
- F3
- B1 (The bandwidth of Formant 1)
- B2 (The bandwidth of Formant 2)
- B3 (The bandwidth of Formant 3)
- num_formants

Under the Syllable header, select:

- stress
- position_in_word

Under the Word header, select:

- label
- stress_pattern

Under the Utterance header, select:

- speech_rate

Under the Speaker header, select:

- name

Under the Sound File header, select:

- name

Once you have checked all relevant boxes, select 'Export to CSV'. Your results will be exported to a CSV file on your computer. The name will be the one you chose to save plus "export.csv". In our case, the resulting file will be called "Tutorial Formants export.csv".

2.3.4 Step 4. Examining & analysing the data

With the tutorial complete, we should now have a CSV file saved on our personal machine containing information about the set of objects we queried for and all other relevant information.

We now examine this data. This part doesn't use ISCAN, so it's not necessary to actually carry out the steps below unless you want to.

In R, load the data as follows:

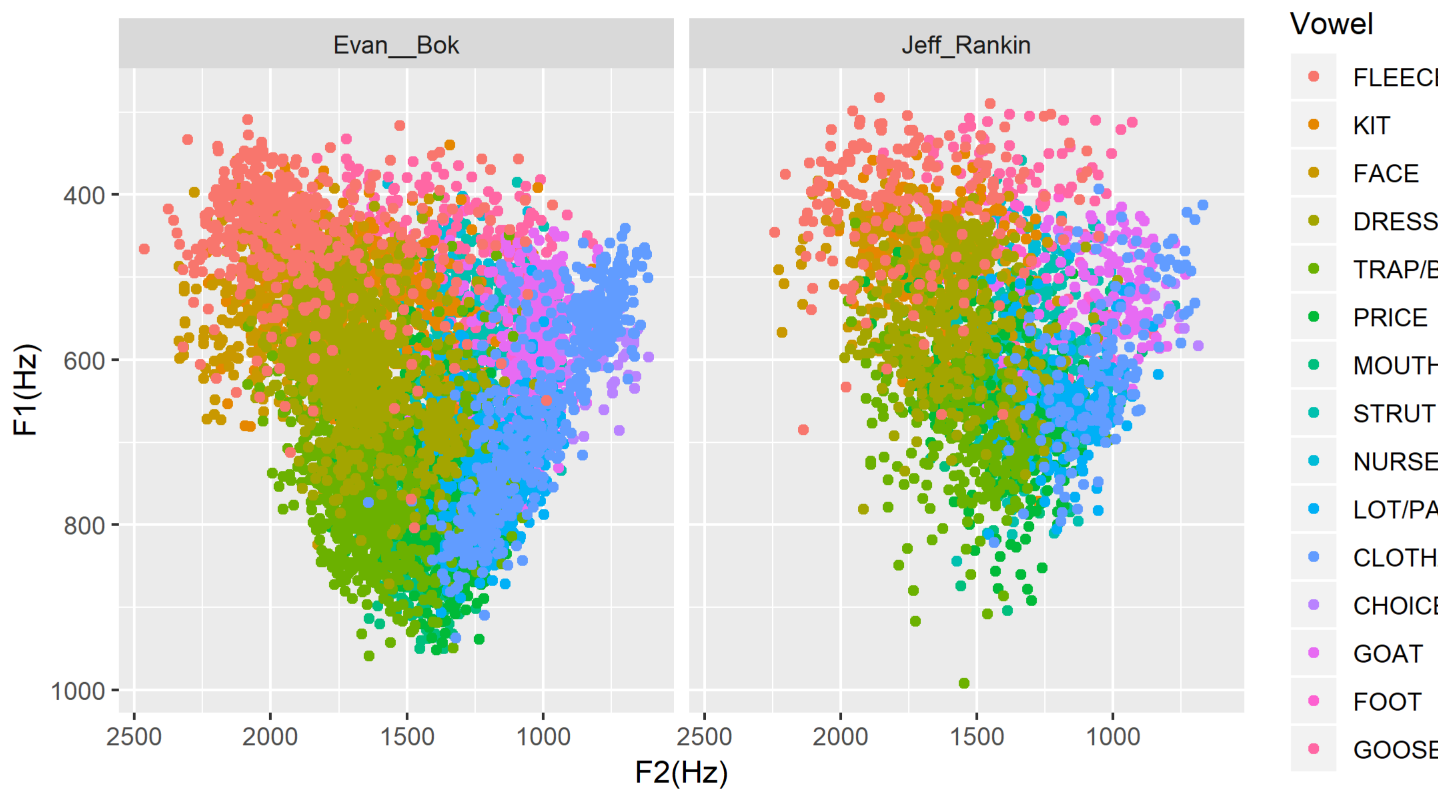
```
library(tidyverse)
v <- read.csv("Tutorial Formants export.csv")
```

Rename the variable containing the vowel labels to 'Vowel', and reorder the vowels so that they pattern according to usual representation in acoustic/auditory vowel space:

```
v$Vowel <- v$phone_label
v$Vowel = factor(v$Vowel, levels = c('IY1', 'IH1', 'EY1', 'EH1', 'AE1', 'AY1','AW1',
↪'AH1', 'ER1', 'AA1', 'AO1', 'OY1', 'OW1', 'UH1', 'UW1'))
```

Plot the vowels for the two speakers in this sound file:

```
ggplot(v, aes(x = phone_F2, y = phone_F1, color=Vowel)) +
geom_point() +
facet_wrap(~speaker_name) +
scale_colour_hue(labels = c("FLEECE", "KIT", "FACE", "DRESS", "TRAP/BATH", "PRICE",
↪"MOUTH", "STRUT", "NURSE", "LOT/PALM", "CLOTH/THOUGHT", "CHOICE", "GOAT", "FOOT",
↪"GOOSE")) +
scale_y_reverse() +
scale_x_reverse() +
xlab("F2(Hz)") +
ylab("F1(Hz)")
```



2.4 Tutorial 3: Sibilants

Sibilants, and in particular, /s/, have been observed to show interesting sociolinguistic variation according to a range of intersecting factors, including gendered, class, and ethnic identities (Stuart-Smith, 2007; Levon, Maegaard and Pharao, 2017). Sibilants - /s z / - also show systematic variation according to place of articulation (Johnson, 2003). Alveolar fricatives /s z/ as in send, zen, are formed as a jet of air is forced through a narrow constriction between the tongue tip/blade held close to the alveolar ridge, and the air strikes the upper teeth as it escapes, resulting in high pitched friction. The post-alveolar fricatives / /, as in 'sheet', 'Asia', have a more retracted constriction, the cavity in front of the constriction is a bit longer/bigger, and the pitch is correspondingly lower. In many varieties of English, the post-alveolar fricatives also have some lip-rounding, reducing the pitch further.

Acoustically, sibilants show a spectral 'mountain' profile, with peaks and troughs reflecting the resonances of the cavities formed by the articulators (Jesus and Shadle, 2002). The frequency of the main spectral peak, and/or main area of acoustic energy (Centre of Gravity), corresponds quite well to shifts in place of articulation, including quite fine-grained differences, such as those which are interesting for sociolinguistic analysis: alveolars show higher frequencies, more retracted post-alveolars show lower frequencies.

As with the previous tutorials, we will use ISCAN to select all sibilants from the imported sound file for the two speakers in the tutorial corpus, and take a set of acoustic spectral measures including spectral peak, which we will then export as a CSV, for inspection.

2.4.1 Step 1: Enrichment

It is not necessary to re-enrich the corpus with the elements from the previous tutorial, and so here will only include the enrichments necessary to analyse sibilants.

Sibilants

Start by looking at the options under 'Create New Enrichments', press the 'Phone Subset' button under the 'Subsets' header. Here we select and name subsets of phones. If we wish to search for sibilants, we have two options for this corpus:

- For our subset of ICE-Can we have the option to press the pre-set button 'Select sibilants'.
- For some datasets the 'Select sibilants' button will not be available. In this case you may manually select a subset of phones of interest.

Then choose a name for the subset (in this case 'sibilants' will be filled in automatically) and click 'Save subset'. This will return you to the Enrichment view where you will see the new enrichment in your table. In this view, press 'Run' under 'Actions'.

Acoustics

For this section, you will need a special praat script saved in the MontrealCorpusTools/SPADE GitHub repository which takes a few spectral measures (including peak and spectral slope) for a given segment of speech. With this script, ISCAN will take these measures for each sibilant in the corpus. A link is provided below, please save the `sibilant_jane_optimized.praat` file to your computer: Praat script

From the Enrichment View, press the 'Custom Praat Script' button under the 'Acoustics' header. As usual, this will bring you to a new page. First, upload the saved file 'sibilant_jane_optimized.praat' from your computer using 'Choose Praat Script' button. Under the **Phone class** dropdown menu, select *sibilant*.

Finally, hit the 'Save enrichment' button, and 'Run' from the Enrichment View.

Hierarchical Properties

Next, from the **Enrichment View** press the 'Hierarchical property' button under 'Annotation properties' header. This will bring you to a page with four drop down menus (Higher linguistic type, Lower linguistic type, Subset of lower linguistic type, and Property type) where we can encode speech rates, number of syllables in a word, and phone position.

While adding each enrichment below, remember to choose an appropriate name for the enrichment, hit the 'save enrichment' button, and then click 'Run' in the Enrichment View.

Syllable Count 1 (Number of Syllables in a Word)

1. Enter "num_syllables" as the property name
2. From the Property type menu, select *count*
3. From the Higher annotation menu, select *word*
4. From the Lower annotation menu, select *syllable*

Syllable Count 2 (Number of Syllables in an Utterance)

1. Enter "num_syllables" as the property name
2. From the Property type menu, select *count*
3. From the Higher annotation menu, select *utterance*

4. From the Lower annotation menu, select *syllable*

Phone Count (Number of Phones per Word)

1. Enter "num_phones" as the property name
2. From the Property type menu, select *count*
3. From the Higher annotation menu, select *word*
4. From the Lower annotation menu, select *phone*

Word Count (Number of Words in an Utterance)

1. Enter "num_words" as the property name
2. From the Property type menu, select *count*
3. From the Higher annotation menu, select *utterance*
4. From the Lower annotation menu, select *word*

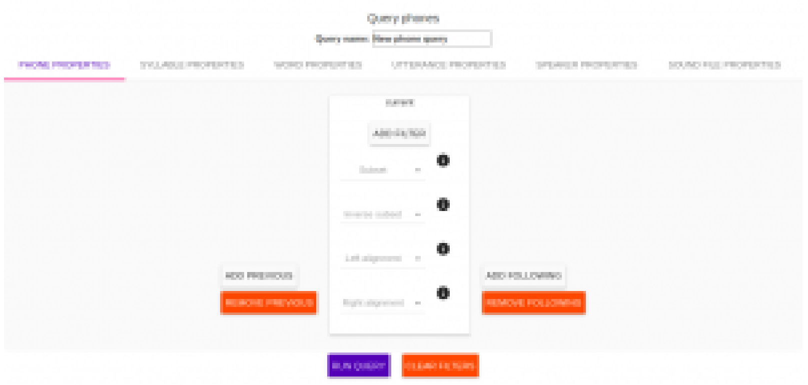
Phone Position

1. Enter "position_in_syllable" as the property name
2. From the Property type menu, select *position*
3. From the Higher annotation menu, select *syllable*
4. From the Lower annotation menu, select *phone*

2.4.2 Step 2: Query

The next step is to search the dataset to find a set of linguistic objects of interest. In our case, we're looking for all sibilants. Let's see how to do this using the **Query view**.

First, return to the 'iscan-tutorial-X' Corpus Summary view, then navigate to the 'Phones' section and select **New Query**. This will take you to a new page, called the Query view, where we can put together and execute searches. In this view, there is a series of property categories which you can navigate through to add filters to your search. Under 'Phone Properties', there is a dropdown menu labelled **'Subset'**. Select 'sibilants'. You may select 'Add filter' if you would like to see more options to narrow down your search.

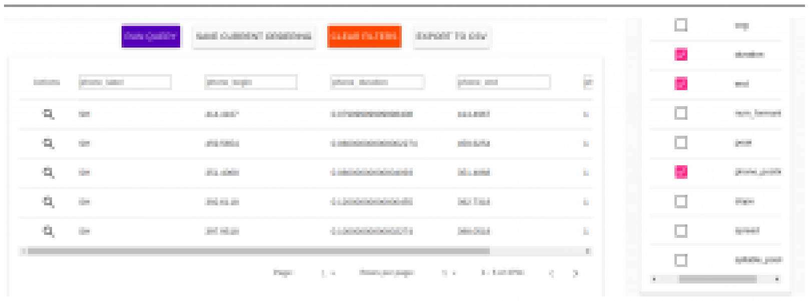

Query name: New phone query

The selected filter settings will be saved for further use. It will automatically be saved as 'New phone query', but let's change that to something more memorable, say 'SibilantsTutorial'. When you are done, click the 'Run query' button. The search may take a while, especially for large datasets.

2.4.3 Step 3: Export

Now that we have made our query and extracted the set of objects of interest, we'll want to export this to a CSV file for later use and further analysis (i.e. in R, MatLab, etc.)

Once you hit 'Run query', your search results will appear below the search window. Since we selected to find all sibilants only, a long list of phone tokens (every time a sibilant occurs in the dataset) should now be visible. This list of sibilants may not be useful to our research without some further information, so let's select what information will be visible in the resulting CSV file using the window next to the search view. G Here we may check all boxes which will be relevant to our later analysis to add these columns to our CSV file. The preview at the bottom of the page will be updated as we select new boxes:



Under the Phone header, select:

- label
- begin
- end
- cog
- peak
- slope
- spread

Under the Syllable header, select:

- stress

Under the Word header, select:

- label

Under the Utterance header, select:

- speech_rate

Under the Speaker header, select:

- name

Under the Sound File header, select:

- name

Once you have checked all relevant boxes, click the 'Export to CSV' button. Your results will be exported to a CSV file on your computer. The name will be the one you chose to save for the Query plus "export.csv". In our case, the resulting file will be called "SibilantsTutorial export.csv".

2.4.4 Step 4: Examining & analysing the data

With the tutorial complete, we should now have a CSV file saved on our personal machine containing information about the set of objects we queried for and all other relevant information.

We now examine this data. This part doesn't use ISCAN, so it's not necessary to actually carry out the steps below unless you want to.

First, open the CSV in R:

```
s <- read.csv("SibilantsTutorial export.csv")
```

Check that the sibilants have been exported correctly:

```
levels(s$phone_label)
```

Change the column name to 'sibilant':

```
s$sibilant <- s$phone_label
```

Check the counts for the different voiceless/voiced sibilants - // is rare!

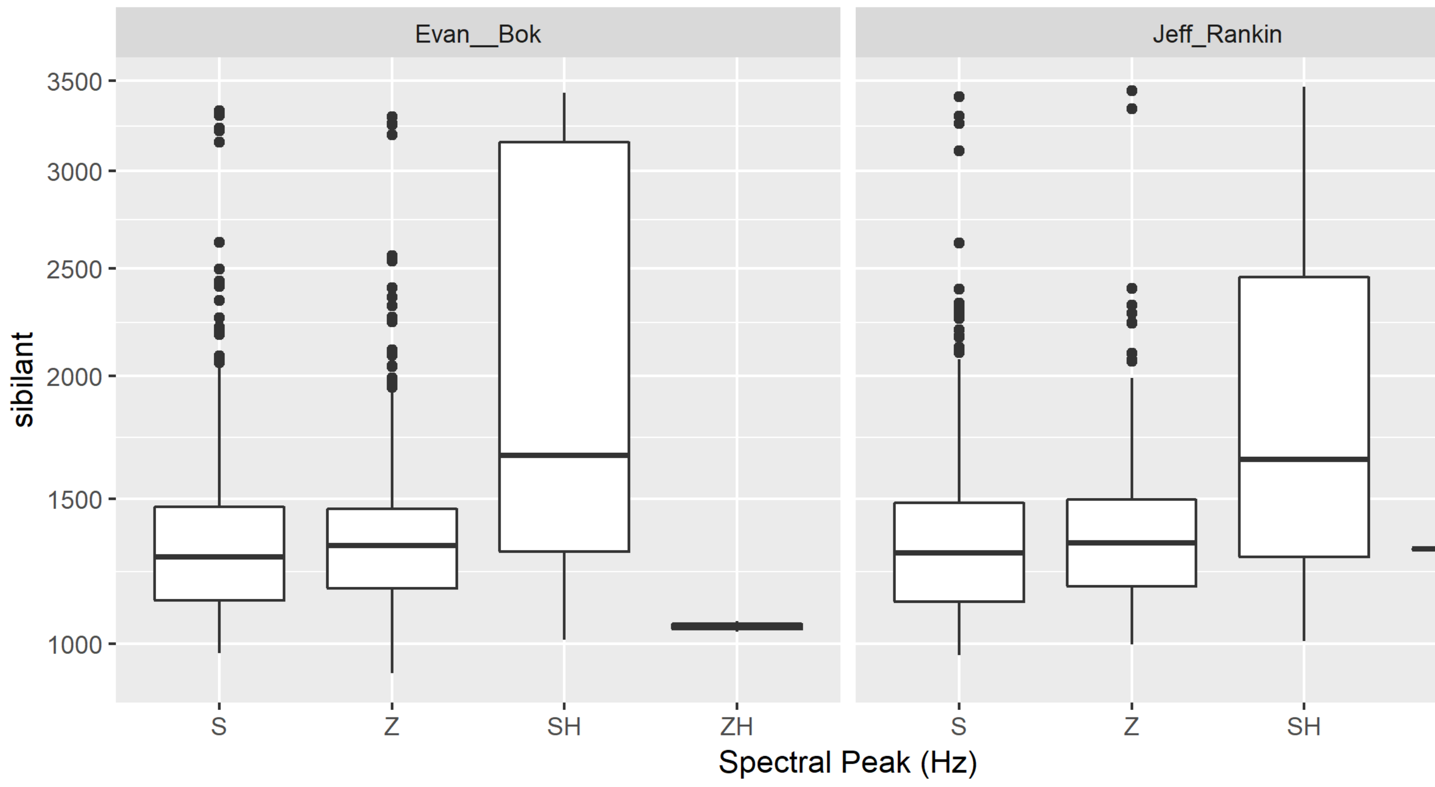
```
summary(s$sibilant)

# S    SH    Z     ZH
# 2268  187 1296  3
```

Reorder the sibilants into a more intuitive order (alveolars then post-alveolars):

```
s$sibilant <- factor(s$sibilant, levels = c('S', 'Z', 'SH', 'ZH'))
```

Finally, plot the sibilants for the two speakers:

```
ggplot(s, aes(x = factor(sibilant), y = phone_peak)) +
geom_boxplot() +
xlab("Spectral Peak (Hz)") +
ylab("sibilant") +
scale_y_sqrt() +
facet_wrap(~speaker_name)
```



2.5 Tutorial 4: Custom scripts

Often in studies it is necessary to perform highly specialized analyses. As ISCAN can't possibly provide every single analysis that anyone could ever want, there is a way to perform analyses outside of ISCAN, and then bring them in. This is the purpose of the 'Custom Properties from a Query-generated CSV' enrichment. Using it is relatively straightforward, although it requires some prelimanary steps to get the data in the right format before using. It also requires access to the original sound files of a corpus if you wish to use these in your analysis.

In this tutorial we will be using an R script, but you can use any script or software that you so choose.

2.5.1 Step 1: Necessary Enrichments

The only necessary enrichment to do in this tutorial is to encode a sibilant subset of phones. To do this, start at the 'iscan-tutorial-X' corpus summary view, and click on the 'Create, run and edit enrichments' button in the centre column. Then, click on 'Phone Subset'.

At the enrichment page, click on the select sibilants button, then name the subset 'sibilants' and save the subset.

Finally, at the main enrichment page, run the sibilant enrichment.

2.5.2 Step 2: Running a phone query

Now that we have all the enrichments we need, we can go to the **Query View**.

Starting at the 'iscan-tutorial-X' corpus summary view, navigate to the phones section of the left-most column and click "new Phone Query". From there, the you'll have the option to choose various different filters to select a subset of phones. For this tutorial we're looking at sibilants, so all you need to do is select the sibilants subset from the first drop-down menu from the centre menu.

Feel free to also re-name the query to anything you'd like, for example 'sibilant ID query'. From there, click on 'Run query' and wait for the query to finish.

2.5.3 Step 3: Exporting phone IDs

Once the query has finished, a new pane will appear to the right of the window. This pane will contain a list of different properties of the phones found, and properties of the syllables, words, and utterances that a phone is in. These are the columns that will be included in the CSV that you will download from ISCAN.

For the script, we will need a couple different columns.

Under the Phone header, select:

- label
- begin
- end
- id

Under the Sound File header, select:

- name

Once all these columns have been selected, click the "Generate CSV Export File" in the row of buttons in the centre of the screen, above the results of the query. This may take a second or two to run, then once it's available, click on the "Save CSV export file" and save the file somewhere convenient on your computer.

An important thing to note for this section is that while you can rename columns for export, you should not rename the ID column if you intend on importing this CSV later. By default, a phone ID column will be labeled "phone_id". When importing, ISCAN looks for a column that ends with "_id" and then uses the first half of the name of that column to know what these IDs represent(in this case, phones). You also should not have multiple ID columns in your import CSV, although if you do, ISCAN will use the first ID column only.

2.5.4 Step 4: Running the R script

The script and associated files can be downloaded here. This script estimates spectral features in R (Reidy, 2015).

In order to get this script running on your computer, you will have to make a few minor edits once you have extracted the ZIP file. Open up 'iscan-token-spectral-demo.R' in your text editor.

At the top of the file, there will be two file paths defined. Change 'sound_file_directory' to the file path of where you have the sound files of the tutorial corpus. Then, change 'corpus_directory' to be the file path of the CSV that you downloaded from ISCAN. I have it as a relative path, but you can of course make it an absolute path.

```r
sound_file_directory <- "/home/michael/Documents/Work/test_corpus"
corpus_data <- read.csv("../sibilants_export.csv")
```

This script also assumes you have not renamed any of the columns that you exported. If you did change any columns' name, you will have to look through the script and change the following lines to the names of the corresponding columns.

```r
sound_file <- paste(corpus_data[row, "sound_file_name"], '.wav', sep="")
begin <- corpus_data[row, "phone_begin"]
end <- corpus_data[row, "phone_end"]
```

Finally, you can run the script in R, and it will create a new CSV file, 'spectral_sibilants.csv' that we will import to ISCAN.

2.5.5 Step 4: Importing the CSV

Back in ISCAN, go to the enrichments page for your tutorial corpus.

Under 'Annotation properties', click on the 'Custom Properties from a Query-generated CSV' button.

From this page, click on the "browse" button and navigate to the 'spectral_sibilants.csv' generated in the last step. Select it for upload, then click on "Upload CSV". This may take a second or two, so be patient.

After this, a new list of properties will appear which come from the columns of the CSV. Scroll down, and select all the features that start with 'spectral'.

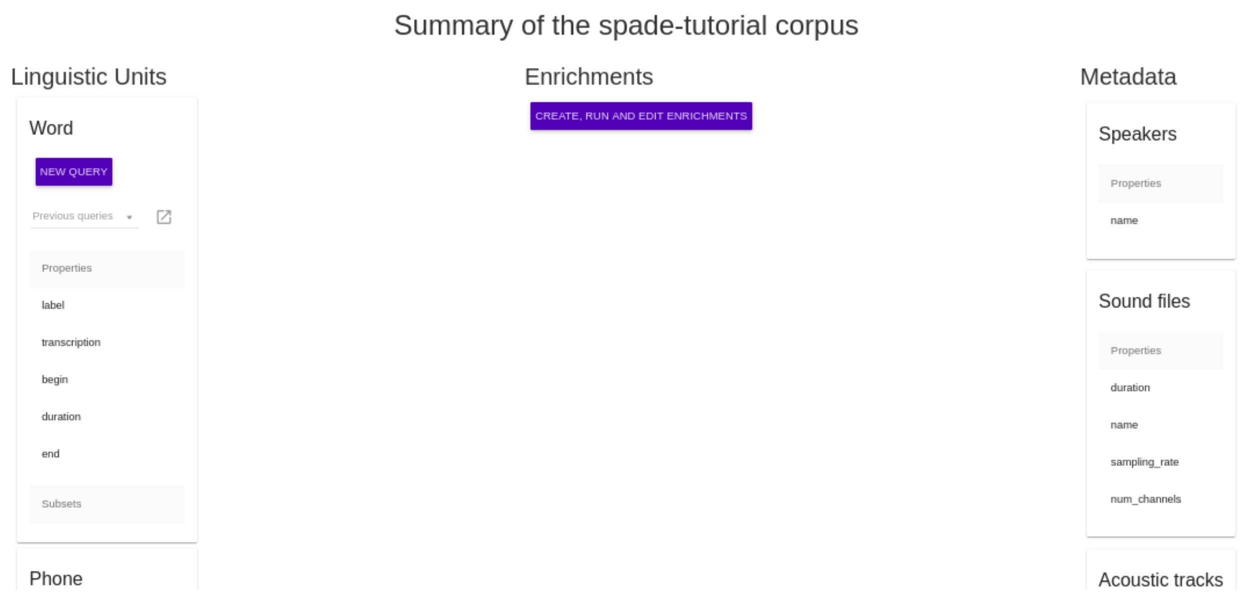
Then, click 'save enrichment', and from the main enrichment page, run the enrichment labelled 'Enrich phone from "spectral_sibilants.csv"'.

Now you're done! ISCAN will now have all of the values calculated by the R script associated with all the sibilants in the corpus. You can test this out by going to the phone query you created earlier. You should see all these new properties in the column selection pane, although you may need to click "Refresh Query" before the values appear.

CHAPTER 3

Corpus View

The Corpus View is the first page a user sees after making a selection from the *Corpora* dropdown menu in the ISCAN banner (given that the corpus has been imported. See the *ISCAN Tutorials* for directions on importing.) This page provides an overview and summary of the selected corpus.



3.1 Linguistic Units

A linguistic unit can be an utterance, word, syllable, or phone.

1. **Utterance** An utterance is (loosely) a group of sounds delimited by relatively long pauses on either side. This could be a clause, sentence, or phrase. Note that utterances need to be encoded before they appear in the Corpus View. See *Enrichment View* for information on encoding utterances.

2. **Word** A word is a collection of phones that form a single meaningful element.
3. **Syllable** A syllable is a unit of pronunciation having one vowel sound. Syllables must be encoded before they appear in the Corpus View. See *Enrichment View* for information on encoding syllables.
4. **Phone** A phone is a single speech segment.

For each of the linguistic units, the Corpus View provides a summary of the encoded *properties* and *subsets*, and allows the user to begin a new query over that unit, or to view and continue editing previous queries.

3.2 Enrichments

This section includes only a button to bring the user to the *Enrichment View*, where databases can be enriched by encoding various elements. Usually, the database starts off with just words and phones, but by adding enrichments a diverse range of information will become available to the user for searching with the *Query View*.

note: This button will be available only to users with permission to enrich. See :ref:‘administration‘ if a user does not have the correct permissions.

3.3 Metadata

The right side bar of Corpus View summarizes any metadata available for the selected corpus.

1. **Speaker Properties** Specifies the metadata properties known about the speakers of a corpus. These are taken from a speaker information metadata file which can be uploaded in the *Enrichment View*. This might include the name, gender, ethnicity, education level, and age of each speaker, among many other properties.
2. **Sound File Properties** Specifies properties of the corpus sound files, such as the name, the duration, and the times that speech begins and ends for each file.
3. **Acoustic Tracks** Specifies which tracks, if any, have been encoded.

CHAPTER 4

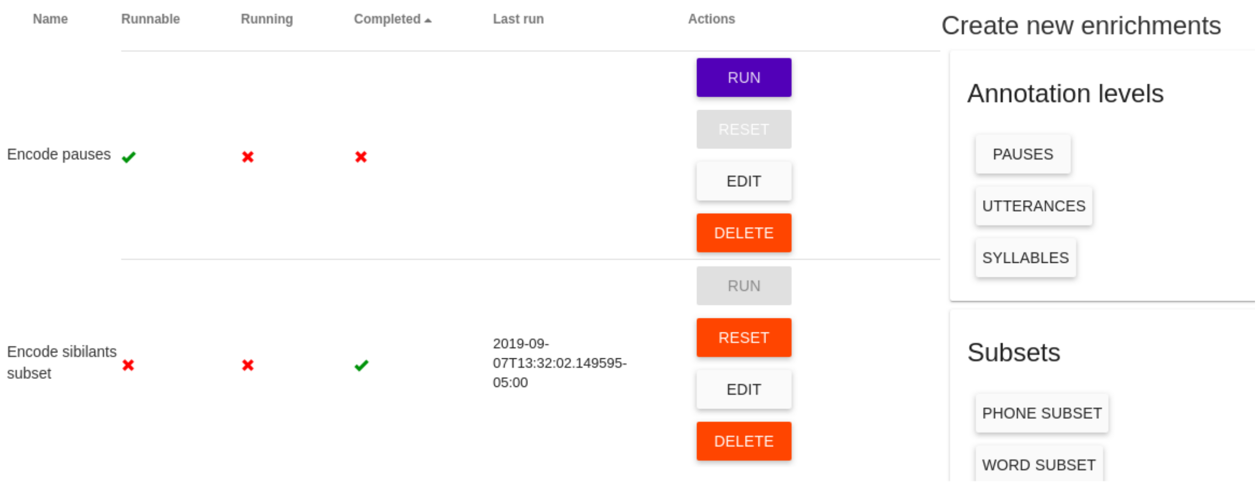
Enrichment View

Databases can be enriched by encoding various elements. Usually, the database starts off with just words and phones, but by adding enrichments a diverse range of information will become available to the user for searching with the Query View later. All enrichments are added in the Enrichment View. Here are some details about this View and the Enrichment options available to the user.

4.1 Actions

Once an enrichment has been saved, a number of possible actions will become available for it. The actions buttons are available as a column in the table visible at the top of the Enrichment View

1. *Run* If the enrichment is runnable, the user may run it to apply this enrichment to the corpus
2. *Reset* The user may reset an enrichment to remove changes added to the corpus by running that enrichment. The state of the database will be as it was before running this enrichment
3. *Edit* The user may make changes to the saved enrichment. For the changes to be applied to the corpus, hit *Run* again after editing
4. *Delete* This has the same effect as *Reset* but also removes this enrichment from the Enrichment View. To re-run a deleted enrichment, the user must start over and create a new enrichment



Name	Runnable	Running	Completed ▲	Last run	Actions
Encode pauses	✓	✗	✗		RUN RESET EDIT DELETE
Encode sibilants subset	✗	✗	✓	2019-09-07T13:32:02.149595-05:00	RUN RESET EDIT DELETE

4.2 Create New Enrichments

To the right of the table of saved enrichments and actions are several options for enriching a corpus.

4.2.1 Annotation levels

1. **Pauses** This allows the user to specify for a given database what segments should count as pauses instead of speech. These are typically among the most common words in a corpus, so top 25 words are provided (25 word default may be changed). If not found, custom pause labels may also be entered in the search bar.
2. **Utterances** Define utterances here as segments of speech separated by a gap of a specified length (typically between .15-.5 seconds).
3. **Syllables** If the user has encoded syllabic segments, syllables can now be encoded using a syllabification algorithm (e.g. maximum attested onset).

4.2.2 Subsets

1. **Phone subset** Here the user may encode a subset of phones, such as vowels, stops, or sibilants, by selecting from the table which phone labels to include. All phones belonging to this subset will be labeled as such in the corpus. For some corpora, predefined subsets ‘sibilants’, ‘syllabics’ and/or ‘stressed vowels’ may be selected automatically.
2. **Word subset** The user may choose to group a subset of words together by selecting from the list of all words appearing in the corpus.

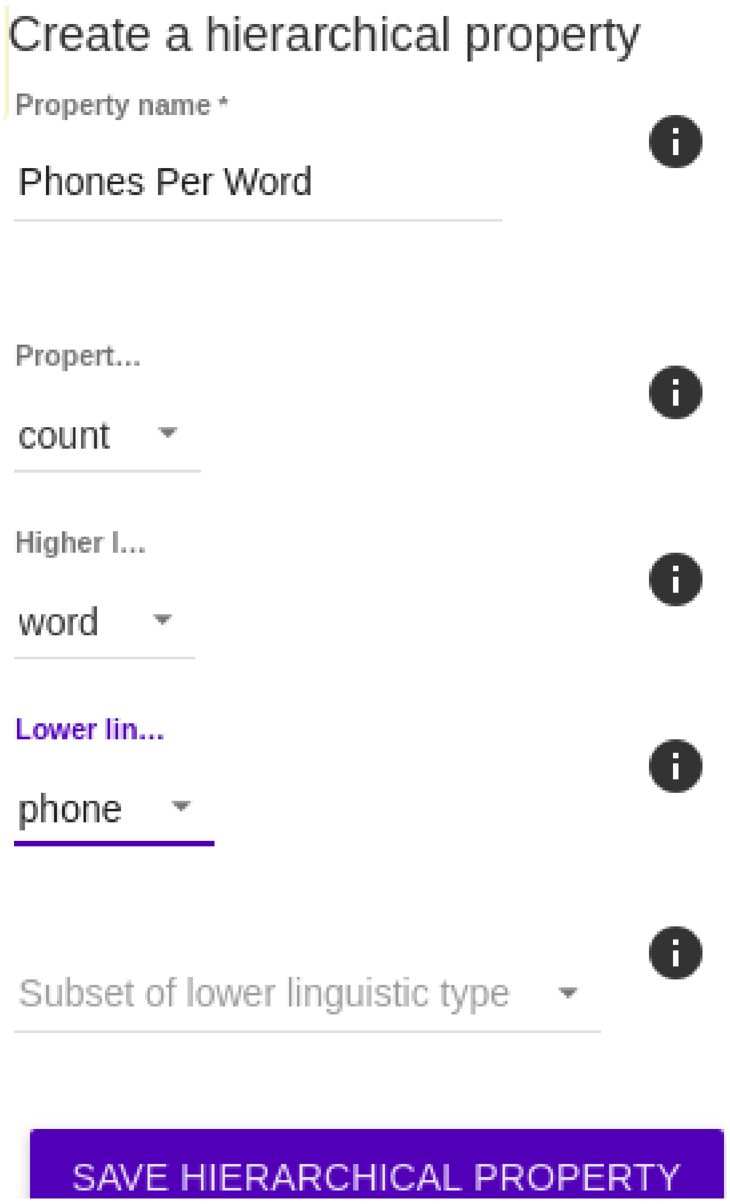
4.2.3 Annotation properties

1. **Hierarchical property** This selection allows the user to encode properties involving two levels, such as number of syllables in each utterance, or rate of syllables per second.
   - *Property type* Target property can be rate, count, or position. Rate will encode on the higher linguistic type number of lower linguistic type units per second (i.e., speech rate on utterances could be the rate of syllables per second). Count will encode on the higher linguistic type the number of lower linguistic

type units that it contains. Position will encode on the lower linguistic type its position within the higher linguistic unit (i.e., position of syllables within a word).

- *Higher linguistic type* Upper level property in hierarchy
- *Lower linguistic type* Lower level property in hierarchy
- *Subset of lower linguistic type* The user may specify a subset of the lower linguistic type to use. If specified, units outside of this subset are ignored. For instance, speech rate for a utterance could be calculated as the rate of "syllabic" phones per second, rather than needing syllables encoded. Another example would be if certain words are of interest in an experiment, that subset can be specified for the position of words of interest within each utterance.

Example: To encode the number of phones per word, set *Property type* to **count**, *Higher linguistic type* to **word**, and *Lower linguistic type* to **phones**. You may leave the subset of a lower linguistic type blank, but the other fields must be filled as hierarchical properties involve some relation between two linguistic types.


Create a hierarchical property

Property name *

Phones Per Word

Propert...

count

Higher l...

word

Lower lin...

phone

Subset of lower linguistic type

1. **Stress from word property** Enriches syllables with stress information ('stressed' or 'unstressed', coded as '1'

or '0') via a listing for each *word* in the lexical enrichment CSV. The user must choose which property to use for encoding syllable stress. The property should have syllable stress separated by dashes (i.e., "1-0" for "dashes"). If there is a mismatch in the number of syllables in this property and in the database, the word's syllables will not have any stress encoded.

2. **Properties from a CSV** Here the user can import information to the corpus using CSV files saved locally
   - *Lexicon CSV* Allows the user to assign certain properties to specific words using a CSV file. For example the user might want to encode word frequency. This can be done by having words in one column and corresponding frequencies in the other column of a column-delimited text file.
   - *Speaker CSV* Allows the user to enrich speakers with information by adding speaker metadata, such as sex and age, from a CSV.
   - *Phone CSV* Allows the user to add certain helpful features to phonological properties. For example, adding 'fricative' to 'manner_of_articulation' for some phones.
   - *Sound File CSV* Sound file properties may include notes about noise, recording environment, etc.
3. **Custom properties from a query-generated CSV** This option allows users to perform analyses outside of ISCAN, and then bring them in. For detailed information on how to use this enrichment, see Tutorial 4 in *ISCAN Tutorials*.
4. **Relativize a property** This permits the user to encode certain statistics
   - *Linguistic type* Specifies which linguistic type (for example, phone or word) will be relativized.
   - *Property* Specifies which property of the linguistic type to relativize.

The user may also decide whether relativization should be performed within speaker (using by-speaker means and standard deviations). If this option is not selected, means and standard deviations will be calculated across the the whole corpus.

4.2.4 Acoustics

1. **Pitch tracks** Here you can specify the program to use to generate pitch tracks (for example, PRAAT)
2. **Voice onset time** Here a user may enrich the corpus with AutoVOT
   - *Stop subset* This menu presents options for the subset of phones that will have their VOTs calculated
   - *Use custom classifier* If this option is selected, you may choose your own classifier (The file format for classifier is a zip file containing both the pos and neg files from an AutoVOT trained classifier). Otherwise it will default to a classifier trained on voiceless word-initial VOTs in SOTC
   - *VOT Min/Max(ms)* These values represent the minimum and maximum values of the VOT calculated. A minimum value of 15 ms will ensure that the difference between the closure and onset of voicing will be at least 15 ms.
   - *Window Min/Max(ms)*
   - *Overwrite manually edited VOTs?* Select this option to overwrite any VOTs that were manually edited in the inspection view

For convenience, default settings for voiced and voiceless stops are available

1. **Formant tracks** Here you can specify the program to use to generate formant tracks (for example, PRAAT)
2. **Intensity tracks** Here you can specify the program to use to generate intensity tracks (for example, PRAAT)
3. **Refined formant points or tracks** This option is for generating and refining formant point measures. The user must specify:
   - The subset of phones representing segments over which the formant analysis will be run.

- The number of refinement iterations. Increasing the amount of iterations will significantly increase the amount of time it takes to run the analysis, but it may improve convergence and accuracy of measurements.

And the user may optionally specify:

- The minimum duration of a phone for it to be analyzed.
- A CSV file containing formant measure prototypes to seed the algorithm before the first refinement iteration. If no CSV file is selected, the prototypes are generated from the data.
- If you prefer to save tracks rather than a single point.

4. **Custom Praat script** This options allows you to run a custom Praat script over a specified type of annotation, or some subset of a type of annotation.
5. **Relativize an acoustic track** For this enrichment, acoustic tracks must already have been encoded. If multiple have been encoded, you may select which acoustic track will be relativized. You may also specify whether relativization should be performed within speaker (using by-speaker means and standard deviations). If this option is not selected, means and standard deviations will be calculated across the the whole corpus.

CHAPTER 5

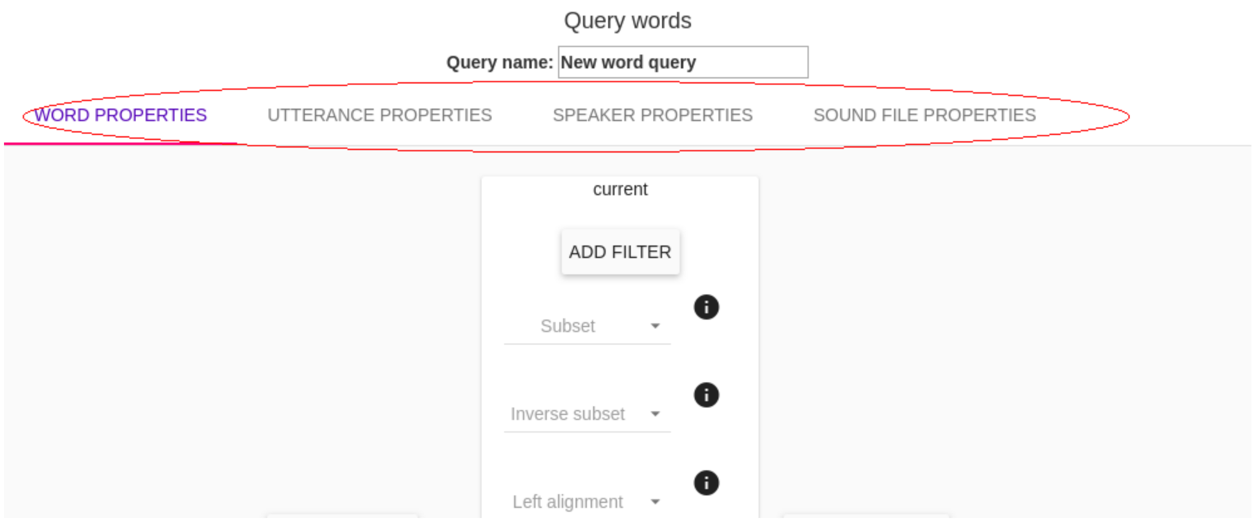
Query View

In this view, the user constructs queries by specifying a particular set of data of interest, and exports a CSV file of relevant linguistic information. For example uses of this page, please see the *ISCAN Tutorials*.

From the *Corpus View*, the user selects a Linguistic Unit over which to query. A linguistic unit can be an utterance, word, syllable, or phone. By selecting a linguistic unit, the user is specifying the set of elements over which the query is to be made. For example, selecting "phones" will cause the program to look for phones with properties specified by the user (if "words" were selected, then the program would look for words, etc.) Go to *Corpus View* for more information about linguistic units.

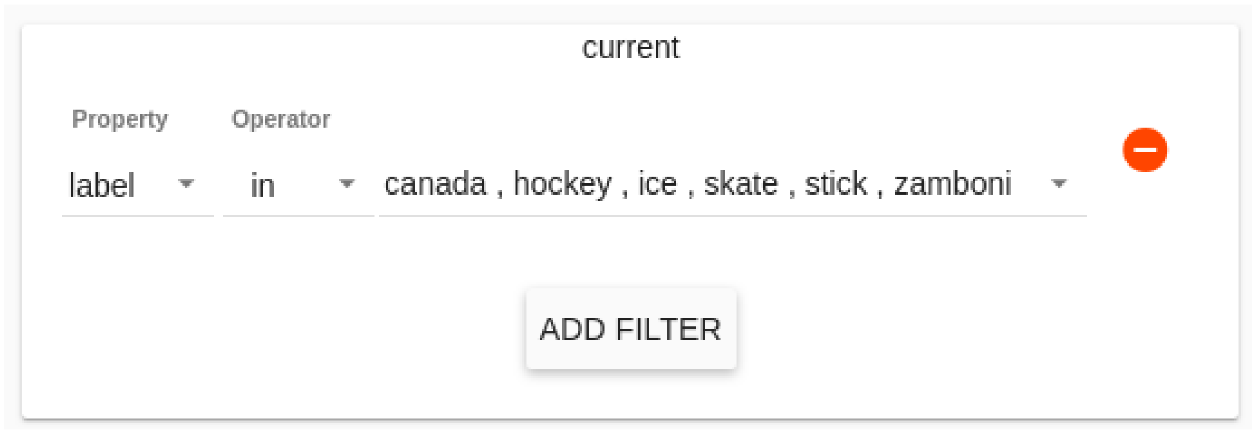
5.1 Building Queries

The user builds queries by specifying filters, subsets, and/or alignments for some linguistic unit. The user can specify particular properties for target units, as well as the units preceding and following the targets. Filters, alignments, and subsets may be specified on any linguistic type at or higher than the linguistic unit of interest. For example, if a user is querying words, filters may be specified over properties of words, over the properties of the utterances those words appear in, over the properties of the sound files in which the words appear, and over the speakers who produced the words. These can be added by selecting the linguistic type of interest in the top bar of the Query View.



5.1.1 Filters

Filters are statements that limit the data returned to a specific set. Each filter added provides another constraint on the data. A filter can be specified by choosing the 'ADD FILTER' button. The filters a dropdown menu for *Property* and *Operator*, as well as a space to type in a *Value*.



Property

The first dropdown menu when creating a filter is used to target a property.

Phone Properties

Without enrichment, the following properties are available for phones:

- *id* The identification key for the phone
- *label* The orthographic content of the phone
- *begin* The start time (seconds) of the phone
- *duration* The amound of time the phone occupies
- *end* The end time (seconds) of the phone

Word Properties

Without enrichment, the following properties are available for words:

- *id* The identification key for the word
- *label* The orthographic content of the word
- *transcription* The phonetic transctiption of the word in the corpus
- *begin* The start time (seconds) of the word
- *duration* The amound of time the word occupies
- *end* The end time (seconds) of the word

Syllable Properties

Without enrichment, no properties are available for syllables as they must first be encoded (see *ISCAN Tutorials* for directions to encode syllables).

Utterance Properties

Without enrichment, no properties are available for utterances as they must first be encoded (see *ISCAN Tutorials* for directions to encode utterances).

Speaker Properties

Without enrichment, the only property available for speakers is the speaker *name*.

After enrichment, the properties available will depend on the metadata CSV file used as input for the enrichment. The speaker information files may include such demographic information as age, ethnicity, gender, place of birth, education level, native language, and so on, but the particular properties available will vary by corpus.

Sound File Properties

Without enrichment, the following properties are available for sound files:

- *duration* The amound of time of the whole sound file
- *name* The name of the sound file
- *sampling_rate* The number of times the audio is sampled per second
- *num_channels* The number of data channels contained in the file

Operator

The second dropdown menu when creating a filter selects an operator.

- **==** means 'equals'. The property selected must match the value specified in the text box following this menu.
- **!=** means 'does not equal'. The property selected must *not* match the value specified in the text box following this menu.
- **in** (for lists of values) The property selected must appear in the specified subset of Values. For example, in a filter on *Speaker Properties*, over the property *name*, operator *in*, selecting a subset of the names in the *Value* dropdown menu will force the query only to output results for which the speaker name is one of those selected.
- **not in** (for lists of values) The property selected must *not* appear in the specified subset of Values.
- **<** (for numeric values) The property selected must be less than the specified value. For example, with this operator you may specify that the *duration* of a *word* be less than 190 ms.
- **>** (for numeric values) The property selected must be greater than the specified value.
- **<=** (for numeric values) The property selected must be less than or equal to the specified value.

- **>=** (for numeric values) The property selected must be greater than or equal to the specified value.

Value

The third field in building a filter is the Value. This may be a text box in which the user inputs a numeric value (for example representing miliseconds for durations, or time at the beginning/end) or a string value (for example representing the label of some phone, or the transcription of a word). In some cases this field may also be a list menu from which the user can select a subset (this will be the case when the operators *in* and *not in* are used).

5.1.2 Subsets

If the user has encoded subsets of linguistic units (for example, *sibilants* may be defined as a subset of phones), then these will be available to further limit the scope of queries to relevant segments. The user may specify one or more subsets that the linguistic unit must belong to. If multiple subsets are selected, the query will look for units that match *all* of them.

Additionally, inverse subsets can be defined, specifying one or more subsets that the linguistic unit must *not* belong to. If multiple subsets are selected, the query will look for units that do not match *any* of them.

5.1.3 Alignments

The user may specify linguistic types that will be aligned at the left and right edges.

Left alignment specifies higher linguistic types that the left edges will be aligned. For example, left aligning words to utterances will get all utterance-initial words.

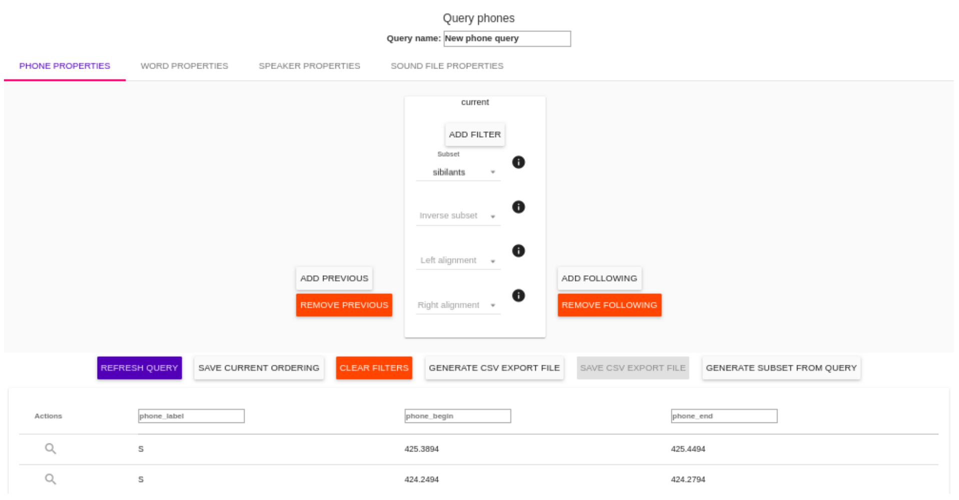
Right alignment Specifies higher linguistic types that the right edges will be aligned. For example, right aligning words to utterances will get all utterance-final words.

5.2 Exporting Queries

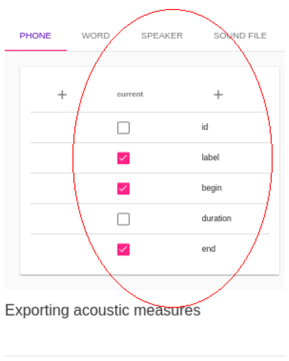
Once all of the desired filters, subsets, and alignments have been selected to build the Query, several options are available at the bottom of this page.

- **Run Query** This will generate the results of the query. The results will be displayed on the Query View page. The user can rename and reorder the displayed results by clicking on the column headers.
- **Refresh Query** At any time after running a query, the user may update/add/remove filters, subsets, and alignments. Refreshing the query will output the results of the query after the changes. This will also return the ordering of the columns of results to the default unless 'Save Current Ordering' has been selected.
- **Save Current Ordering** Clicking on the column headers in the displayed results will reorder them. These changes to the display are typically reverted to default when a query is refreshed. Selecting 'Save Current Ordering' saves the columns as they are displayed across refreshes.
- **Clear Filters** This button will clear all filters built by the user.
- **Generate CSV Export File** This will generate a CSV file in the format shown on the page as a result of running a query. This may take some time, depending on the size of the corpus.
- **Save CSV Export File** This allows you to download the generated CSV file and save it to your local machine.
- **Generate Subset from Query** Allows the user to name and identify results of the query to use in future queries. For example: the user may find all word initial phones in one query, then use *Generate Subset from Query* in order to easily query **'word-initial'** in future queries.

ISCAN shows the results of running a query on the page. This can be a quick way to visualize data, but most often the user will want to further manipulate the data (i.e. in R, MatLab, etc.) To this end, the user has the option to export query results. ISCAN allows the user to specify the information that is exported by adding columns to the final output file using the window which appears next to the search view.



Actions	phone_label	phone_begin	phone_end
🔍	s	425.3894	425.4494
🔍	s	424.2494	424.2794



☐	id
✓	label
✓	begin
☐	duration
✓	end

The same properties appearing during query building will be available, and the user may tick the boxes next to the features to add them as columns to the output CSV.

CHAPTER 6

Administration and installation

Note: At the moment, iscan-server is only supported on Ubuntu.

Warning: Running analyses via script currently do not work with the docker installation. If you plan on writing/running automated scripts, rather than interacting with the data via the web interface, use the non-Dockerized version.

Much of this documentation is meant to be more technical and is geared towards advanced users interested in setting up their own ISCAN servers, either on a desktop computer for local use or on a dedicated server for communal use. Please see the *ISCAN Tutorials* section for more information and a walk-through on how to use ISCAN once it has been set up. Please see the troubleshooting section below for a continuously-updated set of solutions to common installation or maintenence issues.

6.1 Installation via Docker

Note: Currently only Ubuntu is a supported and tested system. We hope to support other operating systems in the future, but please bear in mind that things may not work. If you are on Windows 10, you can install the Linux subsystem. ISCAN servers are known to run using the Bash shell in Windows 10, but the Docker does not work on the Linux subsystem for Windows. If this is the only option for you, please see *Installation without using Docker*.

Warning: Running analyses via script currently do not work with the docker installation. If you plan on writing/running automated scripts, rather than interacting with the data via the web interface, use the non-Dockerized version.

6.1.1 Prerequisites

ISCAN server uses Docker. This containerization means that the only dependency the user must install is Docker itself.

Preparing Docker

First, install Docker for Ubuntu. It is easiest to install using the *Install from repository* method.

Next, complete Docker for Ubuntu's postinstallation instructions. This will make it unnecessary to prepend Docker commands with `sudo`.

Then, install Docker Compose, the tool for defining and running multi-container Docker applications.

6.1.2 Installation

First, clone the iscan-spade-server repository to your machine:

```
git clone https://github.com/MontrealCorpusTools/iscan-spade-server.git
```

Configuration

Two files need to be updated for your specific machine. First, navigate to `iscan-spade-server/iscan_server/settings` and copy/rename the `local_settings.py.template` to `local_settings.py` and add your host's IP address and domain name.

Second, if you would like to use the built-in simple HTTP server (nginx) using the production mode (see *Development and Production modes* for more details on the different modes), edit the `iscan-spade-server/docker-utils/nginx/nginx.conf` file. Update the `server_name` field to reflect your server's IP address/domain name (as with the `local_settings.py` file above).

Build

Included are a `Dockerfile` and a *docker-compose.yml*. In order to build a Docker image from these files, navigate to the root of the repository and run:

```
docker-compose build
```

Then, run:

```
docker-compose up
```

This will launch the containers.

Initial migrations

The first time you use iscan-spade-server, you will need to make database migrations. In another terminal, while the containers are up, run:

```
docker-compose run app init
```

The needed migrations to perform will be detected and made.

Superuser creation

The first time you use iscan-spade-server, you will need to set up a username and password to log in with. In another terminal, while the containers are up, run:

```
docker-compose run app python3 manage.py createsuperuser
```

This will begin a prompt that asks you for a username, email address, and password. Once you have filled them out, the prompt will close.

Then, you should be able to log in with your credentials. You should only need to perform this step once; from now on, whenever you start the server, you should be able to log in with your defined username and password. When finished, press `Ctrl+C` to end the current server run.

6.1.3 Development and Production modes

By default, running the build commands in *Build* will use the "production" environment.

6.2 Use & workflow via Docker

6.2.1 Starting and stopping the server

To start the server and its containers, run:

```
docker-compose up
```

In your web browser, navigate to `localhost:8080`. You should see the ISCAN web page.

To stop the server, press `Ctrl+C` only once. The terminal should show a `Gracefully stopping...` message and then exit.

6.2.2 Mounted volumes

This Docker instance is configured so that the contents of certain directories persist between runs of the server, and so that contents are constant between the local directory and the directory in the container. These local directories, located in the root of the repository, are:

- `polyglot_source/` - the directory containing corpora to be loaded.
- `polyglot_data/` - the directory where corpus metadata will be stored
- `iscan/` - the directory where the front-end code is stored
- `iscan_server/` - the directory containing the Django project for the server

Changes you make locally in these folders should persist into the container without needing to re-build the Docker image.

6.2.3 Cleaning

The `docker-compose up` command usefully regenerates fresh containers each time it is run, but old containers can take up space. To clean up containers on your machine, first stop all of them:

```
docker stop $(docker ps -a -q)
```

Then, remove them:

```
docker rm $(docker ps -a -q)
```

6.3 Installation without using Docker

Note: Currently only Ubuntu is a supported and tested system. We hope to support other operating systems in the future, but please bear in mind that things may not work. If you are on Windows 10, you can install the Linux subsystem. ISCAN servers are known to run using the Bash shell in Windows 10. Also note that `sudo` ability is required for installation and running services that ISCAN depends upon.

6.3.1 Prerequisites

ISCAN servers have the following prerequisites.:

1. Python 3
2. Java 8
3. RabbitMQ
4. SQL database (PostGreSQL recommended)
5. NodeJS and NPM
6. Praat
7. Reaper (optional)

Java 8

To install Java 8 on Ubuntu, you can install the Oracle version via:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

RabbitMQ

For working with asynchronous tasks, a message queue is needed, RabbitMQ is the default, installed as follows

```
sudo apt-get install rabbitmq-server
sudo service rabbitmq-server start
```

See https://simpleisbetterthancomplex.com/tutorial/2017/08/20/how-to-use-celery-with-django.html#installing-rabbitmq-on-ubuntu-1604 for more details.

Relational Database

ISCAN server can use a SQLite database, but in general a PostGreSQL database is recommended. It can be installed via:

```
sudo apt-get install postgresql postgresql-contrib libpq-dev
sudo service postgresql start
```

The database will have to be set up with a user/password as well, see https://www.digitalocean.com/community/tutorials/how-to-use-postgresql-with-your-django-application-on-ubuntu-14-04 for more instructions.

Praat

ISCAN requires the barren version of the Praat executable to run its acoustic analysis scripts (no graphical interface).

You can get this from http://www.fon.hum.uva.nl/praat/download_linux.html.

Once you extract it, make sure that the command `praat` points to this executable (either through an alias or renaming `praat_barren` and making sure the parent directory is included in `$PATH`)

Reaper

Reaper is a program for pitch analysis, so you'll only need to install it if you want to use Reaper's pitch estimation in place of Praat's.

Follow the instructions on Reaper's GitHub repository (https://github.com/google/REAPER) to install it and put the resulting executable somewhere on the system path so that Polyglot can find it easily.

AutoVOT

AutoVOT is a program for automatically calculating Voice Onset Times(VOT). It's necessary in order to run any enrichment related to VOTs

Follow the instructions on AutoVOT's GitHub repository (https://github.com/mlml/AutoVOT) to install it and put the resulting executable somewhere on the system path so that Polyglot can find it easily.

NodeJS

Installation of the front end JavaScript and dependencies is handled by NPM, which is installed as follows:

```
sudo apt-get install nodejs npm
```

6.3.2 Installation

Start by cloning the GitHub repository

```
git clone https://github.com/MontrealCorpusTools/iscan-server.git
```

Once there, look in the `iscan-server/iscan_server/settings` directory and create a file named `local_settings.py`.

Add the following to it, replacing any paths with relevant paths for your system, as well as information for the PostGreSQL database (i.e., whatever database name, user name and password you used when setting up the PostGreSQL database):

```
SOURCE_DATA_DIRECTORY = '/path/for/where/corpora/should/be/loaded/from'

POLYGLOT_DATA_DIRECTORY = '/path/to/store/all/polyglot/data'

DATABASES = {
     'default': {
         'ENGINE': 'django.db.backends.postgresql_psycopg2',
         'NAME': 'database_name',
         'USER': 'user_name',
         'PASSWORD': 'password',
         'HOST': 'localhost',
         'PORT': '5433',
     }
}
```

From the root of the server directory, install all of the server's dependencies:

```
pip install -r requirements.txt
```

For development, getting the latest version of PolyglotDB is recommended via:

```
pip install https://github.com/MontrealCorpusTools/PolyglotDB/archive/master.zip
```

Then set up the server's database:

```
python manage.py migrate
```

To install all of the JavaScript dependencies for the front end, run:

```
npm install
```

To generate a superuser admin account for the server:

```
python manage.py createsuperuser
```

In a separate terminal, start the celery process (from the root of the iscan-server repository):

```
celery -A iscan_server worker -l info
```

Finally, run the server:

```
python manage.py runserver 8080
```

6.4 Server administration

This page is to be used a reference for common tasks in administration of an ISCAN server. The workflow for many of these are not currently as streamlined as they would ideally be.

6.4.1 Updating ISCAN and PolyglotDB

As ISCAN and PolyglotDB are both under active development, updating to the latest version is necessary to fix issues that crop up. To perform an update for the Docker version, run following command from the root of the `iscan-spade-server` directory:

```
./update.sh
```

Which will then fetch the latest changes from the GitHub repositories of both packages. If it generates a permissions error, run `chmod +x update.sh` and rerun the above command.

For the non-Docker version, updating the ISCAN and PolyglotDB packages can be done via the following command from the root of the `iscan-spade-server` repo:

```
pip install -U -r requirements.txt
```

6.4.2 Getting a tutorial corpus

At the moment, ISCAN hosted on https://roquefort.linguistics.mcgill.ca uses an aligned version of ICE-Canada corpus. However, this tutorial corpus is not distributable currently, so we plan to make another one based on a subset of LibriSpeech available in the near future. For the purpose of adding tutorial corpora for each user to run through tutorials, any smaller corpus aligned using the Montreal Forced Aligner will work once renamed to `spade-tutorial` and added to the system.

For ISCAN hosted on https://roquefort.linguistics.mcgill.ca, site administrators can create tutorial corpora from the new *User View*, by going to *Users* in the navigation bar directly in ISCAN.

6.4.3 Adding new corpora

To add any new corpus, simply put its folder in the `polyglot_source` directory (see *Mounted volumes* for its location in the Docker installation). When not using Docker, this is configuration setting in the `local_settings.py` of the Django configuration (`SOURCE_DATA_DIRECTORY` in the non-Docker *Installation*).

Once the corpus is in the directory, performing a reload on the home page of the ISCAN server will update ISCAN's records of databases and corpora. If the corpus does not use forced aligned TextGrids as transcripts, then the corpus format will have to be changed in the admin page (i.e. go to https://hostname.com/admin/iscan/corpus/, select the corpus and select the appropriate supported format from the Input format dropdown).

Supported formats

The primary supported format is the output TextGrids from commonly used forced aligners:

- Montreal Forced Aligner
- FAVE align

In addition, the following formats are supported as well:

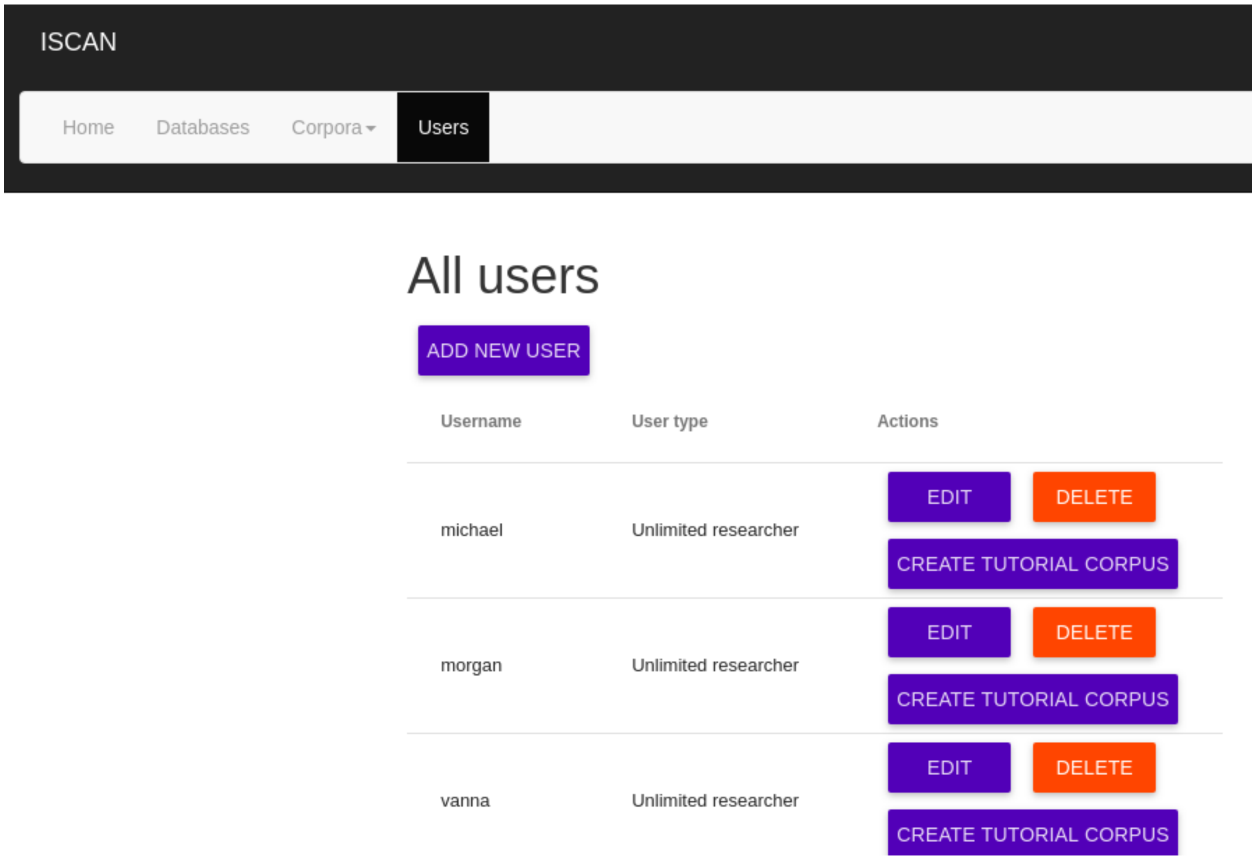
- TextGrid output from LaBB-Cat
- BAS Partitur formatted files
- Buckeye Speech Corpus
- TIMIT

In regards to how the corpus should be structured, where possible files should be divided into speaker directories. If the force aligned TextGrid has speaker information (i.e., word and phone tiers for multiple speakers), this is not necessary.

6.4.4 Creating and Modifying Users

User creation and permissions editing can be done in two ways. As previously, everything can be done throught the Django admin interface (i.e., https://hostname.com/admin/auth/user/). Alternatively, there is now a method for superusers to access this through ISCAN directly, by selecting the *Users* tab in the navigation bar.

In the Users View, a user with administrative permissions can choose the 'Add New User' button to add a user with any role (roles explained below).



All users

Username	User type	Actions
michael	Unlimited researcher	EDIT DELETE CREATE TUTORIAL CORPUS
morgan	Unlimited researcher	EDIT DELETE CREATE TUTORIAL CORPUS
vanna	Unlimited researcher	EDIT DELETE CREATE TUTORIAL CORPUS

User Permissions

When creating a new user, you can choose whether to grant the following permissions:

- *Can edit*: allows the user to edit and correct aspects of the corpus data, such as acoustic measurements like pitch/formants/ tracks or time boundaries of segments/words (this functionality is currently not implemented in ISCAN)
- *Can annotate*: allows the user to add their own annotations to linguistic items, within an annotation framework specified in the admin interface (this functionality isn't fully featured yet, and has primarily only been implemented for annotating utterances)
- *Can view annotations*: allow the user to see annotations in the corpus
- *Can listen*: allows the user to play audio on the corpus detail page
- *Can view detail*: allows the user access to the query detail view, otherwise the user can query but not see the full context of each result
- *Can enrich*: allows the user to create/run/reset/delete enrichments

- *Can access database*: allows the user to start/stop the database for this corpus

User Roles

Some predefined user types are available to easily create new users with the correct permissions.

- **Guest** These users have permissions to query any public corpora. They have no other permissions, and they cannot query any restricted/private corpora.
- **Annotator** These users have permissions to query, view detail, listen, and add annotations to all public corpora. They *do not* have database access, permission to enrich corpora, edit, or view other annotations.
- **Researcher** These researchers have database access, permission to enrich and query all corpora. Additionally, for *public* corpora only, they can view detail, edit, listen, and view annotations. They do not have permission to add annotations.
- **Unlimited researcher** These users correspond to what we otherwise call *superusers*. They have all permissions for all corpora, as well as database access and administrative permissions. This is reserved for McGill internal team members.

In addition to these predefined roles, individual permissions can all be edited manually in the User View by finding the user and selecting the *Edit* button in the *Actions* column. More specific per-corpus permissions can be given to uses through the Django admin interface as well (i.e. https://hostname.com/admin/iscan/corpuspermissions/).

6.4.5 Enable running of SPADE scripts

Warning: This feature is experimental and may not work reliably. Particularly, the way corpora are created via script does not interact with the rest of the ISCAN ecosystem, so these corpora cannot be viewed in the web browser.

Running SPADE scripts is an optional functionality in ISCAN. This allows users to run scripts which automatically do all necessary enrichments and output a specific CSV for a given corpus.

In order to set up SPADE scripts, you must first enable SPADE scripts running, by changing `SPADE_SCRIPTS_ENABLED` to `True` in `setings.py`. Then you should clone the SPADE script repository. The path to this repository must then be set in `settings.py` as `SPADE_SCRIPT_DIRECTORY`.

For the ISCAN Docker repository, the SPADE repo should be cloned in the directory directly above `iscan-spade-server`.

If you also have access to the UNISYN SPADE respository, you should put it in the same directory that you put the SPADE script repository as well.

Necessary SPADE repo changes

auth_token

Since the SPADE scripts are ran as a subprocess of the SPADE server, it is necessary to provide a token for permissions.

Using the ISCAN Docker repository, this can be done by runing `./generate_token.sh USER` where `USER` is the username of an admin account. This will create a file called `auth_token` which must be put in the SPADE script repository.

Installing corpora for SPADE scripts

Once a corpus has been put in `polyglot_source`, you must update the paths in the `config.yml` file for that corpus in the SPADE scripts repo.

If you are using the docker installation with default settings, this can easily be done by switching to the `docker-paths` branch of the SPADE scripts repo. This will also set the IP in `common.py` for Docker.

Feel free to delete any directories for corpora that you do not have access to. This will prevent users from running scripts over corpora that do not exist(which will naturally cause errors).

Installing new scripts

To install a new script, simply put it in the SPADE repository. This must be done by an administrator by hand for security reasons. The script should output a CSV in the corpus that it runs over.

6.4.6 Reporting errors and issues

Some issues can be worked around in the admin interface. For instance, running an enrichment locks the corpus as `busy`, which can cause issues with rare exceptions during their running to cause the corpus to become locked. This `busy` status can be fixed by changing this property on the admin page for that corpus object.

Additionally, databases can be reset to their original non-imported state by deleting the database on the admin page for databases (i.e., https://hostname.com/admin/iscan/database/).

If any issues are encountered, please post them along with the exception message found either in the runserver window or the celery window to the GitHub issues page.

6.5 Apache server configuration

The current recommended configuration is to have a forward facing web server (Apache/NGINX) proxy pass to a locally running server (Gunicorn/Django dev server). The following all assumes that there is a locally running server running on port 8080, and uses Apache as the example configs listed below, as well as the hostname https://roquefort.linguistics.mcgill.ca/.

Additionally, since there is sensitive data involved, we heavily recommend using HTTPS rather than HTTP.

Note: All commands assume Ubuntu 16.04. Commands may differ depending on other operating systems.

6.5.1 Enabling prerequisite Apache modules

```
sudo apt-get install apache2
sudo service apache2 stop
sudo a2enmod rewrite
sudo a2enmod ssl
sudo a2enmod proxy
sudo a2enmod proxy_http
```

6.5.2 HTTP server config

The HTTP server config uses the rewrite module to change any HTTP requests into HTTPS ones, so that there is never any use of http://roquefort.linguistics.mcgill.ca/ over https://roquefort.linguistics.mcgill.ca/. The following config would be saved to a file named `roquefort.linguistics.mcgill.ca.conf` in `/etc/apache2/sites-available/`.

```
<VirtualHost *:80>

    # Update for other hostname
    ServerName roquefort.linguistics.mcgill.ca
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html


    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    RewriteEngine on
    # Update for other hostname
    RewriteCond %{SERVER_NAME} =roquefort.linguistics.mcgill.ca
    RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

Enable the site via:

```
sudo a2ensite roquefort.linguistics.mcgill.ca.conf
```

6.5.3 HTTPS server config

The primary configuration file for the Apache server is the HTTPS one below. SSL certificates are easily generated through Let's encrypt. The Proxy module for Apache is used to forward all requests to the locally running ISCAN server. The following config would be saved to a file named `roquefort.linguistics.mcgill.ca-ssl.conf` in `/etc/apache2/sites-available/`.

```
<IfModule mod_ssl.c>
<VirtualHost *:443>

    # Update for other hostname
    ServerName roquefort.linguistics.mcgill.ca

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # Update for actual location
    SSLCertificateFile /etc/letsencrypt/live/roquefort.linguistics.mcgill.ca/
↪fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/roquefort.linguistics.mcgill.ca/
↪privkey.pem
    Include /etc/letsencrypt/options-ssl-apache.conf

    <Location "/">
```

(continues on next page)

(continued from previous page)

```
            ProxyPass http://localhost:8080/
            ProxyPassReverse http://localhost:8080/
            ProxyPreserveHost On
            RequestHeader unset X-Forwarded-Proto

            RequestHeader set X-Forwarded-Proto https env=HTTPS
     </Location>

</VirtualHost>
</IfModule>
```

Enable the site via:

```
sudo a2ensite roquefort.linguistics.mcgill.ca-ssl.conf
```

Once the configuration files are set up, the Apache server can be rebooted via:

```
sudo service apache2 restart
```

6.6 Troubleshooting common issues

6.6.1 Connection refused

If you run into an error where a SPADE server returns the following connection error:

```
Traceback (most recent call last):
File "/home/linguistics/XXX/miniconda3/lib/python3.6/site-packages/neobolt-1.7.0rc5-
↪py3.6-linux-x86_64.egg/neobolt/direct.py", line 793, in _connect
s.connect(resolved_address)
ConnectionRefusedError: [Errno 111] Connection refused
```

1. First check that your webserver is running. Assuming you are connecting via localhost, run:

```
telnet localhost 8080
```

or

```
netstat | grep 8080
```

And you should see that port 8080 is listening. If either of these are not working, run:

```
python manage.py runserver 8080
```

from the root of your `iscan-spade-server` directory.

2. Check Neo4j is listening on the right ports. This can be checked at `iscan-spade-server/polyglot_data/CORPUS/neo4j.log`. Specifically, ISCAN Neo4j should be using 7400 (compared to a default Neo4j install which uses 7474 and 7687).

In this case, the easiest thing to do is to reset the database. Inside your `SPADE` repository, run:

```
python reset_database.py CORPUS
```

Where `CORPUS` refers to the particular corpus you are trying to use. Which will delete the database files from your polyglot_data directory. If you then run:

```
python SPADE_SCRIPT CORPUS -r
```

This will rebuild your database from scratch.

6.6.2 Neo4j PID error on stopping

Sometimes there may be issues in how ISCAN tracks the Polyglot databases. It does this by storing the PID from the system for both Neo4j and InfluxDB. Sometimes however, these can be empty but the database is started and the database processes are running. As a workaround for this issue, you can reset the database fully by doing the following:

1. Stop the Neo4j process (`kill -9 <pid>`, where `<pid>` can be gotten from a `ps -l` command or `top -n 5` command, the neo4j will be listed as `java`)
2. Navigate to the ISCAN admin page (locally http://localhost:8080/admin/)
3. Log in with a superuser's credentials
4. Click on `Databases` under `ISCAN`
5. Click on the database name that is having issues
6. Change the status to `Stopped`
7. Retry whatever operation was having issues previously (either start the database again or run the script).

6.7 Moving parts

The ISCAN server uses several components, which are all managed by Docker (see *Installation via Docker* and *Use & workflow via Docker* for more information). The components are:

- A web interface, with which the user can manage and analyze corpora, written in Angular
- A Python API, PolyglotDB, which communicates between the web interface and the back-end web framework, using Django
- A relational database, which manages metadata about Polyglot databases, using PostgreSQL
- Message and job queues for working with asynchronous tasks, using RabbitMQ and Celery

CHAPTER 7

Indices and tables

- genindex
- modindex
- search