Cristiano Chesi (c. chesi @ unisi. it)

CREATING, EXPLORING AND QUERYING UNSTRUCTURED CORPORA
THE ROLE OF FREQUENCIES AND REGULAR EXPRESSIONS

Goals

(1) Creating a non-annotated corpus
(2) Exploring a semi-structured corpus (Childes)
(3) Appreciating frequencies
(4) Using Regular Expression to isolate morphological cues in your favorite language

Frequencies and Concordances analysis

(5) Download and install TextStat for Windows (http://neon.niederlandistik.fu-berlin.de/en/textstat/),
WordSmith (if you have a license for it) or AntConc (http://www.laurenceanthony.net/software.html)
(6) Create a new corpus (Menu "Corpus" > "New Corpus")
(7) Populate the corpus retrieving document from the web ("add files from the web")
remember "balancing" issues:
a. choose a target linguistic domain to analyze
(newspaper, twitter, Facebook, blog, specific author, specific slang, pathology…)
b. try to keep the sampling as ecological as possible
c. try to be as representative as possible (remember that a standard "target" is 1.000.000 token... for this workshop we can start with 10.000/100.000 tokens)
(8) Analyze frequencies ("Show word frequencies")... what are the most frequent items in all corpora?

Regular Expressions for querying unannotated text

(9) Regular expressions: A regular expression is a pattern that describes a set of strings. Regular expressions are constructed analogously to arithmetic expressions, by using various operators to combine smaller expressions (see below)

(10) See https://regexr.com/ for creating the first expressions

The you can use GREP (General Regular Expression Print) (for Windows users: Windows Grep: http://www.wingrep.com/)

From "manual-entry: GREP" in Gnu Emacs 20.7.4.

Grep understands two different versions of regular expression syntax: "basic" and "extended." In GNU grep, there is no difference in available functionality using either syntax. In other implementations, basic regular expressions are less powerful. The following description applies to extended regular expressions; differences for basic regular expressions are summarized afterwards.

The fundamental building blocks are the regular expressions that match a single character. Most characters, including all letters and digits, are regular expressions that match themselves. Any metacharacter with special meaning may be quoted by preceding it with a backslash.

A list of characters enclosed by [and] matches any single character in that list; if the first character of the list is the caret ^ then it matches any character not in the list. For example, the regular expression [0123456789] matches any single digit. A range of ASCII characters may be specified by giving the first and last characters, separated by a hyphen. Finally, certain named classes of characters are predefined. Their names are self explanatory, and they are: [:alnum:], [:alpha:], [:cntrl:], [:digit:], [:graph:], [:lower:], [:print:], [:punct:], [:space:], [:upper:], and [:xdigit:].
For example, [[:alnum:]] means [0-9A-Za-z], except the latter form is dependent upon the ASCII character encoding, whereas the former is portable. (Note that the brackets in these class names are part of the symbolic names, and must be included in addition to the brackets delimiting the bracket list.)
Most metacharacters lose their special meaning inside lists.
To include a literal] place it first in the list. Similarly, to include a literal ^ place it anywhere but first. Finally, to include a literal - place it last.

The period . matches any single character. The symbol \w is a synonym for [[:alnum:]] and \W is a synonym for [^[:alnum:]].

The caret ^ and the dollar sign $ are metacharacters that respectively match the empty string at the beginning and end of a line. The symbols \< and \> respectively match the empty string at the beginning and end of a word. The symbol \b matches the empty string at the edge of a word, and \B matches the empty string provided it's not at the edge of a word.

A regular expression may be followed by one of several repetition operators:

? The preceding item is optional and matched at most once.
* The preceding item will be matched zero or more times.
+ The preceding item will be matched one or more times.
{n} The preceding item is matched exactly n times.
{n,} The preceding item is matched n or more times.
{n,m} The preceding item is matched at least n times, but not more than m times.

Two regular expressions may be concatenated; the resulting regular expression matches any string formed by concatenating two substrings that respectively match the concatenated subexpressions.

Two regular expressions may be joined by the infix operator |; the resulting regular expression matches any string matching either subexpression.

Repetition takes precedence over concatenation, which in turn takes precedence over alternation. A whole subexpression may be enclosed in parentheses to override these precedence rules.

The backreference \n, where n is a single digit, matches the substring previously matched by the nth parenthesized subexpression of the regular expression.

In basic regular expressions the metacharacters ?, +, {, |, (, and) lose their special meaning; instead use the backslashed versions \?, \+, \{, \|, \(, and \).

Traditional egrep did not support the { metacharacter, and some egrep implementations support \{ instead, so portable scripts should avoid { in egrep patterns and should use [{] to match a literal {.

GNU egrep attempts to support traditional usage by assuming that { is not special if it would be the start of an invalid interval specification. For example, the shell command egrep '{1' searches for the two-character string {1 instead of reporting a syntax error in the regular expression. POSIX.2 allows this behavior as an extension, but portable scripts should avoid it.

OPTIONS

-A NUM, --after-context=NUM
Print NUM lines of trailing context after matching lines.

-B NUM, --before-context=NUM
Print NUM lines of leading context before matching lines.

-C [NUM], -NUM, --context[=NUM]
Print NUM lines (default 2) of output context.

-b, --byte-offset
Print the byte offset within the input file before each line of output.

-c, --count
Suppress normal output; instead print a count of matching lines for each input file. With the -v, --invert-match option (see below), count non-matching lines.

-E, --extended-regexp
Interpret PATTERN as an extended regular expression

-e PATTERN, --regexp=PATTERN
Use PATTERN as the pattern; useful to protect patterns beginning with -.

-F, --fixed-strings
Interpret PATTERN as a list of fixed strings, separated by newlines, any of which is to be matched.

-G, --basic-regexp
Interpret PATTERN as a basic regular expression. This is the default.

-i, --ignore-case
Ignore case distinctions in both the PATTERN and the input files.

-n, --line-number
Prefix each line of output with the line number within its input file.

-v, --invert-match
Invert the sense of matching, to select non-matching lines.

-x, --line-regexp
Select only those matches that exactly match the whole line.

DIAGNOSTICS
Normally, exit status is 0 if matches were found, and 1 if no matches were found. (The -v option inverts the sense of the exit status.) Exit status is 2 if there were syntax errors in the pattern, inaccessible input files, or other system errors.

(11) What to search for
1. Create a regular expression to capture all (and only) the articles in your language
2. Create a regular expression to capture all (and only) the prepositions in your language

Strumenti di analisi ed Espressioni Regolari per CHILDES

(12) What's CHILDES?

Childes (Child Language Data Exchange System) is an archive of spontaneous speech transcription between children and adults (each transcription is about 20-60 minutes long)..
Standard CHAT format:

Obligatory Headers
@Begin marks the beginning of a file
@End marks the end of the file
@ID: code for a larger database
@Participants: lists actors in a file

Constant Headers
@Age of XXX: marks a speaker's age
@Birth of XXX: shows date of birth of speaker
@Coder: people doing transcription and coding
@Coding: version of CHAT coding
@Education of XXX: indicates educational level of speaker
@Filename: shows name of file
@Font: sets the default font for the file
@Group of XXX: indicates the subject's group in group studies
@Language: the principal language of the transcript
@Language of XXX: language(s) spoken by a given participant
@SES of XXX: ndicates socioeconomic status of speaker
@Sex of XXX: indicates gender of speaker
@Stim: indicates stimulus for elicited production
@Transcriber: gives the transcriber's name or initials
@Warning: marks defects in file

Changeable Headers
@Activities: component activities in the situation
@Bg and @Bg: begin gem
@Bck: backgrounding information
@Comment: comments
@Date: date of the interaction
@Eg and @Eg: end gem
@g: simple gems
@Location: geographical location of the interaction
@New Episode: point at which a new episode begins and old one ends
@Room Layout: configuration of furniture in room
@Situation: general atmosphere of the interaction
@Tape Location: footage markers from tape
@Time Duration: beginning and end times
@Time Start: beginning time

Words
@ special form markers
xxx unintelligible speech, not treated as a word
xx unintelligible speech, treated as a word
yyy unintelligible speech transcribed on %pho line, not treated as a word
yy unintelligible speech transcribed on %pho line,treated as a word

www untranscribed material
0 actions without speech
& phonological fragment
[?] best guess
text(text)text noncompletion of a word
0word omitted word
0*word ungrammatical omission
00word (grammatical) ellipsis

Basic Utterance Terminators
. period
? question
! exclamation

Tone Unit Marking
-? rising final contour
-! final exclamation contour
-. falling final contour
-'. rise-fall final contour
-,. fall-rise final contour
-, level nonfinal contour
-_ falling nonfinal contour
- low level contour
-' rising nonfinal contour
, syntactic juncture
,, tag question
pause between words
-: previous word lengthened

Prosody Within Words
/ stress
// accented nucleus
/// contrastive stress
: lengthened syllable
:: pause between syllables
^ blocking

Special Utterance Terminators
+... trailing off
+..? trailing off of a question
+!? question with exclamation
+/. interruption
+/? interruption of a question
+//. self-interruption
+//? self-interruption of a question
+"/. quotation follows on next line
+". quotation precedes
+" quoted utterance follows
+^ quick uptake
+< "lazy" overlap marking
+, self-completion
++ other-completion
[c] clause delimiter

Scoped Symbols
·%mov:"*"_0_1073· time alignment marker
[=! text] paralinguistics, prosodics
[!] stressing
[!!] contrastive stressing
["] quotation marks
[= text] explanation
[: text] replacement
[0 text] omission
[:=x text] translation
[=? text] alternative transcription
[%xxx: text] dependent tier on main line
[% text] comment on main line
[$text] code on main tier
[?] best guess
[>] overlap follows
[<] overlap precedes
<text> [<>] overlap follows and precedes
[>number][<number] overlap enumeration
[/] retracing without correction
word (*N) word repetition
[//] retracing with correction
[///] retracing with reformulation
[/-] false start without retracing
[/?] unclear retrace type
[*] error marking
[+ text] postcode
[+ bck] exclued utterance
[+ trn] inclued utterance

Dependent Tiers
%act: actions
%add: addressee
%alt: alternative transcription
%cod: general purpose coding
%coN: additional general coding categories, co1, co2
%coh: cohesion tier
%com: comments by investigator
%def: codes from SALT
%eng: English translation
%err: error coding
%exp: explanation
%fac: facial actions
%flo: flowing version
%gls: target language gloss for unclear utterance
%gpx: gestural and proxemic activity
%int: intonation
%lan: language
%mod: model or target phonology
%mor: morphemic semantics
%mov: movie tier
%par: paralinguistics
%pho: phonetic transcription
%sit: situation
%snd: sonic CHAT sound tier
%spa: speech act coding
%syn: syntactic structure notation
%ssy simple syntactic categories
%tim: time stamp coding

Dependent Tier Special Codes
$ indicates codes
$=N occurs for N following utterances
$sc=N-M codes refer to words N through M on the main tier
<bef> occurrence before an utterance
<aft> occurrence after an utterance

Error Coding
$= source of an error in the %err line

= placed between error and target
; separates errors on %err line

Morphosyntactic Coding
- suffix marker
prefix marker
+ compound or rote form marker
~ clitic marker
~# placed after separable prefix
~- placed before second part of discontinuous morpheme
& fusion marker

= English translation for the stem
-0 omitted affix
-0* incorrectly omitted affix
| follows part-of-speech on %mor line
& nonconcatenated morpheme in %mor line
# prefix delimiter on %mor line
+ (Plus) compound delimiter on %mor line
- (Dash) suffix delimiter on %mor line
: feature fusion on %mor line
~ (Tilde) clitic delimiter on %mor line
0 precedes omitted element
0* precedes incorrectly omitted element

(13) Short CHILDES transcription sample (CHAT format):

```
@UTF8
@Begin
@Participants:    CHI Cam Target_Child, DON Mother
@ID:  it|romance|CHI|3;4.9|female|||Target_Child||
@ID:  it|romance|DON|||||Mother||
@Age of CHI:      3;4.9
@Sex of CHI:      female
@Birth of CHI:    3-MAY-1988
@Date:      12-SEP-1991
@Filename:  cx40
@Situation: registrazione seduta
*DON: ma non si sente # prima si registra, e dopo # … come una cosa che
      prima si scrive, e dopo si legge .
%sit: come sempre, quando Camilla vede il registratore, le viene in mente
      di ascoltare le cassette, e vuole impossessarsene per premere i
tasti
*DON: ora bisogna registrarlo # adesso giochiamo un attimino, scusa !
*CHI: io io c'ho messo io ho preso quello che volevo .
*DON: quale volevi ?
*CHI: io volevo questo .
*DON: e e cosa quale, quello .
*CHI: … un nastrino .
*DON: si ma cosa, che canzoni ci sono, sopra .
*CHI: non lo so .
*DON: come non lo sai ?
*CHI: la scuola della &scuo e &dimin .
%act: canticchia, inventando un po' le parole
*DON: ah@i !
*CHI: puffi.
[…]
@End
```

(14) How can you use CHILDES?

CLAN (Computerized Language Analysis) tools:

CHAINS	Tracks sequences of interactional codes across speakers.
CHECK	Verifies the accuracy of CHAT conventions in files.
CHIP	Examines parent-child repetition and expansion.
CHSTRING	Changes words and characters in CHAT files.
COLUMNS	Reformats the transcripts into columnar form.
COMBO	Searches for complex string patterns.
COOCUR	Examines patterns of co-occurence between words.
DATES	Uses the date and birthdate of the child to compute age.
DIST	Examines patterns of separation between speech act codes.
DSS	Computes the Developmental Sentence Score.

FLO	Reformats the file in simplified form.
FREQ	Computes the frequencies of the words in a file or files.
FREQMERG	Combines the outputs of various runs of FREQ.
FREQPOS	Tracks the frequencies in various utterance positions.
GEM	Finds areas of text that were marked with gem markers.
GEMFREQ	Computes frequencies for words inside gem markers.
GEMLIST	Lists the pattern of gem markers in a file or files.
KEYMAP	Lists the frequencies of codes that follow a target code.
KWAL	Searches for word patterns and prints the line.
MAKEDATA	Converts data formats for CHAT files across platforms.
MAKEMOD	Adds a %mod line for the target SAMPA phonology
MAXWD	Finds the longest words in a file.
MLT	Computes the mean length of turn.
MLU	Computes the mean length of utterance.
MODREP	Matches the child's phonology to the parental model.
MOR	Inserts a new tier with part-of-speech codes.
PHONFREQ	Computes the frequency of phonemes in various positions.
POST	Probabilistic disambiguator for the %mor line
POSTLIST	Displays the patterns learned by POSTTRAIN
POSTTRAIN	Trains the probabilistic network used by POST
RELY	Measures reliability across two transcriptions.
SALTIN	Converts SALT files to CHAT format.
STATFREQ	Formats the output of FREQ for statistical analysis.
TEXTIN	Converts straight text to CHAT format.
TIMEDUR	Uses the numbers in sonic bullets to compute overlaps.
VOCD	Computes the VOCD lexical diversity measure.
WDLEN	Computes the length of utterances in words.

(15) Some example (@ indicates .cha selected files):

MLU (usage: mlu @)

FREQ (usage: freq +tchi +s"casa" @)
+t specify tier +tchi, or +tmot, or +ttho, or +t%cod
+d includes words with frequency info, and line numbers
+d1 includes words, no frequency info, no line numbers
+f send output to file instead of output window
+s search string note: you can use wildcards and search lists
+u collapse output into one file

KWAL (usage: kwal +s"casa" @)
+t select tier
+d output: legal CHAT format
+d1 output: legal CHAT format +file names +line numbers
+d2 output: like d2, but once per file only
+s search string
+f send output to file
+u collapse all selected files
+n/-n +nchi: include/exclude all utterances for speaker chi when they follow a match to the search string
+wN provide a window of N utterances following the utterance
-wN provide a window of N utterances preceding the utterance
+o include additional tier (*mot, for example) for representational purposes ONLY

COMBO (usage: combo +t*CHI +w2 -w2 +s"mia^*a" @)

(16) Connect to PHPChildes (Childes interface with RE):
http://www.ciscl.unisi.it/childes/
login: guest
pass: unisi

(17) Regular expression to find imperfective forms in Italian:

comando base:
grep -i -n -B3 -A3 –E
"(([[:space:]]|[[:punct:]])[[:alpha:]]*[aei](vo|vi|va|vamo|vate|vano)([[:space:]]|[[:punct:]])|([[:space:]]|[[:punct:]])[eE]r(o|i|a|avamo|avate|ano)([[:space:]]|[[:punct:]]))";

Filter:
grep -i -v -E
"(([[:space:]]|[[:punct:]])[dbl][lrea](a)?v[oaei](te)?([[:space:]]|[[:punct:]])|([[:space:]]|[[:punct:]])([Ss]c|[Aa]r)riv[ioea]([[:space:]]|[[:punct:]])|([[:space:]]|[[:punct:]])[cC]attiv|([[:space:]]|[[:punct:]])(copri)?divan|([[:space:]]|[[:punct:]])(s)?c(h)?(i)?av[iaeo]([[:space:]]|[[:punct:]])|[ae]tevi([[:space:]]|[[:punct:]]))"

Looking for complex patterns

1. Find irregular forms in children productions (“chetto” instead of “questo”).
2. Can you adapt the previous pattern in order to isolate both regular and irregular forms (e.g. demonstratives)
3. Create a regular expression to capture local agreement (e.g. Determiner-Noun)
4. Try searching for over-regular forms (es. *Mario ha corruto qua!*)
5. Create regular expressions to find substitutions and omissions of auxiliaries! (e.g. *ho caduto*)

References

Create a corpus with
TextStat (Windows): http://neon.niederlandistik.fu-berlin.de/en/textstat/
Antconc (Mac & Windows): http://www.laurenceanthony.net/software.html

Learning how to use Regular Expressions
https://regexr.com/

Use Regular Expressions (Windows Grep)
http://www.wingrep.com/

CHILDES (CLAN software, linguistic data and documentations) can be freely browsed and downloaded from:
http://childes.psy.cmu.edu/

PHPChildes (Childes we interface for using Regular Expressions):
http://www.ciscl.unisi.it/childes/
login: guest
pass: unisi