Lexical processing strongly affects reading times but not skipping during natural reading, Heilbron et al. *Open Mind* (2023)

September 4, 2023

1 Code and data for Lexical processing strongly affects reading times but not skipping during natural reading -- Heilbron et al. *Open Mind* (2023)

This PDF was compiled from the notebook, `README_info.ipynb`. Since you need to open/run jupyter notebooks anyway, it is recommended to look at that notebook rather than this textfile.

1.1 Main results per main figures

- **Figure 2** --> `regression_analyses/word_skipping.ipynb`
- **Figure 3** --> `regression_analyses/reading_times.ipynb`
- **Figure 4** --> `regression_analyses/reading_times_cognitive_vs_occulomotor.ipynb` & `word_skipping_cognitive_vs_occulomotor.ipynb`
- **Figure 5** --> `regression_analyses/reading_times.ipynb`
- **Figure 6** --> `regression_analyses/contextual_vs_non_contextual_prior.ipynb`

Other data-describing figures from the Appendix are found in `Additional_data_descriptives.ipynb`, and `cloze_vs_surp_RT.ipynb`

1.2 Structure of corpora

There are two directories with data: - `materials/` - `resources/`

Materials contains the pre-loaded, preprocessed corpora, in a dataframe/csv format, preprocessed to compute additional metrics (like word eccentricity / launch distance, for each word), and lexical statistics (like linguistic/parafoveal entropy and surprisal. The eventual actually used datsets are saved as dataframes/csvs under the name: `materials/[CORPUS]/[corpus]_df_preloaded.csv`. For intance, for geco corpus: `materials/GECO/geco_df_preloaded.csv`.

Resources contain the original files of each corpus, and some derived files extracted from the raw corpora (such as the texts), that together are the ingredients to creat the `_preloaded.csv` dataframes. For most corpora the `resources` folder contains the texts already, but for Dundee these are unfortunately protected by copyright so I did not share these extracted ones here.

Starting from scratch To reproduce going from the raw corpora to the `_df_preloaded.csv` dataframes, there are two files to check out. First run the cells in `generate_files.ipynb`. Then you can run the cell in `prepare_preloaded_corpora.ipynb` to actually save the `_df_preloaded.csv` files.

After you have run the routines in `generate_files.ipynb`, you can construct the preprocessed `df_preloaded` from scratch using the functions `[corpus]_construct_df` (e.g. `geco_construct_df`) from `resources/load_resources.py`. One important caveat is the attribute `distowrd`. This is what we call the launch distance (i.e. distance to previous leftward fixation position, i.e. the fixation during the previous *progressive, rightward* fixation), which we define for each word, i.e. including for skipped words. This variable is critical for many analyses including the exclusion of words and the computation of parafoveal entropy. **By default the foutines in `resources/load_resources.py` compute this in *PIXELS*. But for downstream analyses this has to be converted to *characters*.** This doesnot happen in the load_resources functions, but it has to happen at some point. In the archived code this happens in the `_load_df` functions in `predread_io.py`. If you want to work with corpora you build yourself from `resources/load_resources.py`, this is something you have to do yourself at some point.

1.3 Information about the parafoveal Ideal Observer

A demo of the Ideal Observer can be found in - bayes_demo/ideal_obser_demo.ipynb

this notebook also renders the plots from figure A3

1.4 Python dependencies

For repeating the analyses, you need Jupyter to run/open the notebooks (like this one). Additionally, I used the following packages (with associated versions):

```
numpy: 1.18.1
scipy: 1.4.1
sklearn: 0.22.1
pandas: 1.0.1
matplotlib: 3.1.3
matplotlib_venn: 0.11.6
seaborn: 0.11.1
```

If you would additionally like to repeat the text analyses and start from scratch you also need the following packages:

```
spacy: 2.3.2
transformers: 3.0.0
torch: 1.4.0
```

2 More detailed corpora characteristics

2.0.1 Dundee

- Length = 56,212 tokens
- Pixels per character ~ *8*
- Pixels per degree ~ *26.67*
- Characters per visual degree ~ *3.33*
- Character size = 0.3°
- Viewing distance = 500mm
- Font = 8x16 monospaced font (white-on-black)

2.0.2 Geco

- Length = 56,466 tokens -- except particpant `pp25` (sub_i=3), which misses text-4
- Pixels per character = *10*
- Pixels per degree ~ *30*
- Characters per visual degree = *3*
- Character size ~ 0.333°
- Viewing distance = XXXmm
- Font = 14 point Courier New (on light grey background)

2.0.3 Provo

- Length = 2,689 tokens
- Pixels per character ~ *13.33*
- Pixels per degree ~ *40*
- Characters per visual degree ~ *3*
- Character size = 0.333°
- Viewing distance = 600mm

2.1 Raw files to processed dataframes

Within analyses we relied on three large naturalistic reading corpora: + The Dundee corpus + The GeCo corpus + The Provo corpus

The first task was to gather the raw corpura files and make them: 1. Easily readable by Python Pandas 2. Have all the necessary computed columns (e.g. compute launch distance) -- simple maths 3. Bring the three corpora in one common frame

Point 3 is especially important in order to have a 1 button solution for regression, we dont want different naming, formatting or whatever.

2.1.1 Processing structure

1. Raw files > constructed csv files `[generate_files.ipynb]`

2. Integrate linguistic aspects into previously constructed csv files `prepare_preloaded_corpora.ipynb`

2.2 Dataframe naming convensions

The overall majority of all three dataframes are brought into one common dataframe, column naming convension is very similair with some small exceptions.

2.2.1 `xxx_load_df` [minimally processed]

- `IA_Area` = The size of the current interest area around the word in pixels.
- `IA_Bottom` = The bottom side pixel position of the current interest area.
- `IA_Left` = The left side pixel position of the current interest area.
- `IA_Left_Real` = The left side pixel position of the current word (where the first alphabetic character of that word starts)
- `IA_Left_Real` = The right side pixel position of the current word (where the last alphabetic character of that word ends)
- `IA_Right` = The right side pixel postition of the current interest area.
- `IA_Top` = The top side pixel position of the current interest area.
- `Launch_Duration` = The duration of the launch site fixation (i.e. the previous /leftward/ fixated word).
- `Launch_duration_all` = The duration of the last fixation of the previous fixated word, for both fixated and skipped words.
- `Launch_X` = The horizontal coordinate position of the launch site fixation (i.e. the previous /leftward/ fixated word).
- `Launch_X_all` = Horizontal launch position (position of last previous progressive fixation), for both fixated and skipped words.
- `Launch_Y` = The vertical coordinate position of the launch site fixation (i.e. the previous /leftward/ fixated word).
- `PP` = The participant identificator.
- `Sentence_Length` = The number of words in the current sentence.
- `Text_Nr` = The text identifier number.
- `Trial` = The number of the trial.
- `Trial_Reading_Time` = Summation of all fixation durations in the current trial.
- `Word` = The word contained in the current interest area (including punctuation and other non-letter characters).

- `Word_Cleaned` = The word contained in the current interest area (cleaned-up, excluding punctuation).
- `Word_Cont_or_Func` = Factor denoting whether the current word is a content word (**1**) or a function word (**0**).
- `Word_F1_Duration` = The duration of the first fixation that was within the current word.
- `Word_F1_Fixationindex` = The ordinal sequence of the first fixation that was within the current word.
- `Word_F1_Progressiveness` = Boolean checks whether later interest areas have been vistited before current word first fixation. (**1**) if NO higher IA ID in earlier fixation, (**0**) if higher IA ID in earlier fixation (i.e. regression back to current word).
- `Word_F1_Starttime` = Start time of the first fixation to enter the current interest area.
- `Word_F1_Visited` = The number of different words visited before the first fixation is made into the current word.
- `Word_F1_X` = The horizontal coordinate position of the first fixation that was within the current word.
- `Word_F1_Y` = The vertical coordinate position of the first fixation that was within the current word.
- `Word_Fixation_Count` = Total number of fixation falling within the current word.
- `Word_Function` = The syntactic function of the current word in the sentence context.
- `Word_Gopasttime` = Summation of all fixation durations from when the current word is first fixated until the eyes enter a word with a higher word identification number.
- `Word_Length` = The length of the current word, in letters.
- `Word_Nr` = The ordinal position of the word in the text.
- `Word_Nr_Sentence` = The ordinal position of the current word within the current sentence.
- `Word_Nr_Text` = The ordinal position of the current word within the current text.
- `Word_Nr_Trial` = The ordinal position of the current word within the current trial.
- `Word_Pupil` = Average pupil size across all fixation in the current word.
- `Word_R1_Count` = The number of all fixation in a trial falling in the first run of the current word.
- `Word_R1_Endtime` = The end time of the first run of fixations in the current word.
- `Word_R1_Gazeduration` = Summation of all fixation durations in the first run within the current word.
- `Word_R1_Last_X` = The horizontal coordinate postion of the last fixation (of the first run) that was within the current word.
- `Word_R1_Last_Y` = The vertical coordinate postion of the last fixation (of the first run) that was within the current word.

- `Word_R1_Starttime` = The start time of the first run of fixations in the current word.
- `Word_Run_Count` = The number of times the current word was entered and left (runs).
- `Word_Skip` = A word is considered skipped (i.e. Word_Skip = **1**) if no fixation occurred in first-pass reading.
- `distowrd` = The launch distance, the distance from the last fixation in a previous word to the start of the current word.
- `endline` = Boolean indicator whether a word or multiple words at the end of the sentence were skipped (e.g. if the last 3 words in a sentence were skipped **endline** would be set to (**1**) for all 3, if the last word was fixated all words in that sentence will be indicated as (**0**)).
- `firstfixated` = Boolean indicator indicating the first fixated word of each sentence.
- `nonalpha_start` = The number of non-alphabetic characters before a word starts (e.g. **"-hallo** counts **2**).
- `nonalpha_end` = The number of non-alphabetic characters after a word ends (e.g. puncuations: **ok...** counts as **3**).
- `sentence_fixated` = Boolean indicator indicating whether a sentence had at least **one** fixation. (**1**) when a sentence was fixated at least ones, (**0**) if the sentence was completly skipped.
- `startline` = Boolean indicator of where a line starts.

Some slight differences between resulting dataframes exist, these veriations consist mostly of temporary variables used for corpus specific parsing.

Dundee The Dundee corpus focuessed on letter by letter fixational data, the available data lacks any precise vertical measurements. For this reason, we do not have any `Launch_Y`, `Word_F1_Y`, `Word_R1_Last_Y`. Furthermore, the Dundee corpus lacks any pupil data (i.e. `Word_Pupil`). + `Blinked` = Boolean indicater whether a blinke occured (**1**) before or aften a fixation within the current word. + `EMARKS` = Number of additional character following text word (e.g. closing punctuation). + `FDUR_y` = Specific duration for combining dataframes - please ignore (to be dropped). + `INLET_POSLINE` = The initial letter position on the text word on an 81-character line (the space to the left of each word counts as its initial letter). + `LINE` = Line processed (pet text/screen number), 1-5. + `LINE_WPOS` = The position of the text word on its line. + `OLEN` = Length of fixated `object' (i.e. word plus any additional characters). + `OMARKS` = Number of additional characters before text word. + `PUNCTCODE` = Punctuation code. Punctuation codes are listed in Appendix of original resource. + `SERIAL_SCRNUM` = The serial number of the text word in a screen. + `Sentence_Index` = The serial number of the sentence within the text. + `TXFR` = Local text frequency. + `Word_F1_X_Object` = First fixation horizontal position in characters, per screen (per line, per char) counting punctuation fixations. + `Word_F1_X_Word` = First fixation horizontal position in characters, per screen (per line, per char) discounting punctuation fixations. + `endtime` = Generated timestamp end of trial. + `starttime` = Generated timestap start of trial. #### Geco The geco already included nearly all of the necessary data, resulting in minimum required parsing. All variables are as described above (underneed `xxx_load_df` [minimally processed]). #### Provo All additional variables contained in the provo dataframe are a result of parsing artifacts. + `IA_FIRST_FIXATION_RUN_INDEX` = Count of how many runs of fixations have occurred when a first fixation is made to an interest area. The current run is also included in the tally. + `index`

= Per participant index count (needed for participant wise parsing). + `level_0` = Per participant index count, excluding duplicates (needed for participant wise parsing).

2.2.2 `xxx_load_df_preloaded` [with added linguistic and parafoveal characteristics]

Note: `xxx_load_df_preloaded` is a subset of `xxx_load_df` with additional columns for lexical (implemented with gpt-2) and parafoveal (implemented with optimal reader) characteristics. Here I discribe only the additional columns.

- `BAD` = Boolean indicator, (**1**) notation is given for bad parsing instances. For complete explenation see below.
- `GPT2_LogProb` = Word-by-word log probability (optained using GPT-2).
- `GPT2_Rank` = GPT-2 rank within distribution.
- `GPT2_String` = GPT-2 string (mostly for parsing error detection).
- `GPT2_lexical_entropy` = Lexical entropy obtained using GPT-2.
- `GPT2_lexical_propability` = Lexical (non-log) probability obtained using GPT-2.
- `GPT2_lexical_surprisal` = Lexical surprisal obtained using GPT-2.
- `post_cohort_entropies` = Letter-by-letter list of post cohort entropies of each letter.
- `sigma(5.7)rnk(2)` = Parafoveal weighted trie entropy given a specific rank.
- `unigram_surprisal` = Word frequency / Unigram surprisal of a given word.
- `weighted_entr` = Parafoveal weighted trie entropy given a specific rank (same as `sigma(5.7)rnk(2)` if only one accuity distance and rank was provided).

2.3 BAD markings

BAD markings are given for rows where a parsing error occurred, all functions to give bad markings are integrated into the dataframes in `preprocessing_corpora.ipynb`. Three main function inside `predread_io.py` will handle the **BAD** markings, `geco_mark_bad(df)`, `provo_mark_bad(df)`, and `dundee_mark_bad(df)`. *Note that, most bad markings are due to parsing errors, with the exception of letter digit combination (which only occur in geco and Dundee).*

`geco_mark_bad(df)`: + nan words are marked as bad + words that have a letter digit combination (e.g. `c42') + words that contain spaces within a single line word (so two words parsed in a single row) + manually mark bad for errors in text 3 of participant 27 (measuring errors)

`provo_mark_bad(df)`: + parsing errors in text 55, last two words were set to empty (marked as bad) + mark bad for missed measurements in participant 1

`dundee_mark_bad(df)`: + mark bad for letter digit combinations (e.g. `c42') + rows that are adjacent to a blink are marked as bad

2.4 Settting restrictions for analysis

Restriction settings are handled by a function `predread_io.py` named `set_restrictions(df, analysis, maxdist=21, prevfix=False)`.

Input needs: + **df**: input dataframe + **analysis**: either *`skipping'* or *`readingtimes'* of the respective analysis + (optional) **maxdist**: the maximum distance (in chars) from launch site + (optional) **prevfix**: True will sellect only instances where the prev word was fixated, default = False

Function will return the restricted dataframe. For example `geco = set_restrictions(geco, "0027skipping"0027, prevfix=True)` will make restrictions in the geco dataframe looking at word skipping and only counting words where the previous word was fixated.

restrictions in place are as follow: + **BAD**: words with a bad marking are excluded + **Word_F1_Progressiveness**: words that are non-progressive are excluded (not counting skipped words for skipping option) + **distowrd**: words falling outside the maxdist window (words that excide the launch distance) are excluded + (optional): **prevfix**: words where the previous word is skipped are exluded