Modelblocks Overview

February 28, 2025

Contents

1 Overview

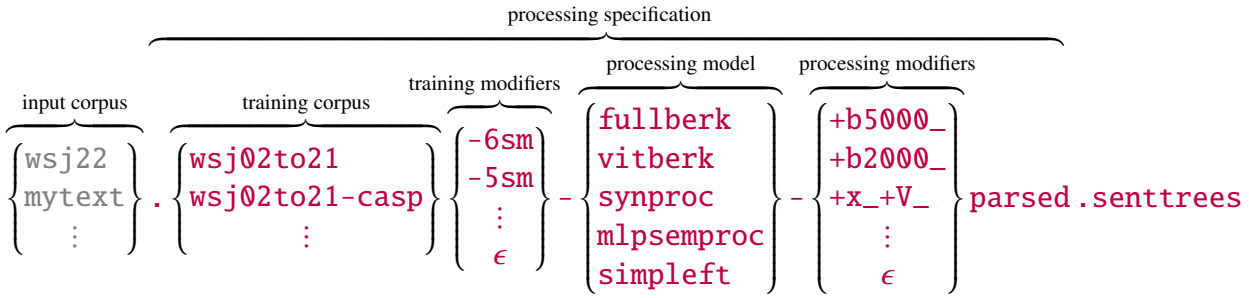
Modelblocks provides toolchains for prediction and evaluation of linguistic phenomena, like delimitation and classification of recursive signs (e.g. parsing), and psycholinguistic phenomena, like reading latencies and observations of neural activity or blood oxygenation, for the purpose of evaluating theories of sentence processing. These toolchains are organized into recursive sub-recipes for generating files with structured filenames, called **make targets**, using the *make* build manager. Make targets whose filenames conform to the Modelblocks sub-recipe syntax can be automatically generated by a *make* command from any directory containing a recipe file that includes pointers to relevant Modelblocks sub-recipe files. The resulting make targets may contain automatically-generated theoretically-motivated predictions or reports that evaluate the linguistic and psycholinguistic accuracy of these predictions, and thus provide evidence for or against the theories that motivate them.

2 Linguistic prediction and evaluation

2.1 Linguistic prediction

Filenames of make targets that contain linguistic predictions each consist of a specification of an **input corpus** followed by a **processing specification**. Each processing specification consists of a specification of a **training corpus**, followed by a specification of zero or more **training modifiers**,

followed by a specification of a **processing model**, followed by a set of zero or more **processing modifiers**:



Input corpus. Input corpora should be sentence-segmented and tokenized and formatted as matrices consisting of sequences of newline-delimited sentences (rows), each consisting of a sequence of space-delimited tokens (columns). Filenames of sentence-segmented and tokenized input corpora should end with the suffix `.senttoks`. Input files can also be automatically segmented and tokenized from unstructured text files with filenames ending in the suffix `.txt`.

Linguistically-annotated training corpus. Modelblocks supports the following linguistically-annotated training corpora:

- `wsj02to21`: The Penn Treebank (Marcus et al., 1994) standard training set consists of about 40,000 hand bracketed sentences of newspaper text from the Wall Street Journal. Projects using this resource should include the `resource-treebank` make file.
- `wsj02to21-casp`: The Penn Treebank (Marcus et al., 1994) standard training set may also be re-annotated into cued-association sentence processing markup (Nguyen et al., 2012). Projects using this resource should include the `resource-treebank` and `resource-casp` make files.
- `chtb`: The Penn Chinese Treebank (Xue et al., 2005) standard training set consists of about 40,000 hand bracketed sentences of newspaper text from various sources. Projects using this resource should include the `resource-chtb` make file.
- `chtb-casp`: The Penn Chinese Treebank (Xue et al., 2005) standard training set may also be re-annotated into cued-association sentence processing markup (Duan and Schuler, 2015). Projects using this resource should include the `resource-chtb` and `resource-chgcg` make files.
- `ontowsj02to21`: The Ontonotes (Pradhan et al., 2007) standard training set consists of about 40,000 hand bracketed sentences of newspaper text from the Wall Stree Journal. Projects using this resource should include the `resource-ontonotes` make file.
- `ontowsj02to21-casp`: The Ontonotes (Pradhan et al., 2007) standard training set may also be re-annotated into cued-association sentence processing markup. Projects using this resource should include the `resource-ontonotes` make file.

- `simplewiki-casp`: Modelblocks defines a training set of about 1,000,000 automatically bracketed sentences of encyclopedia text from Simple English Wikipedia. Projects using this resource should include the `resource-simplewiki` make file.

Linguistic processing model, training modifiers, and processing modifiers. Modelblocks supports the following processing models:

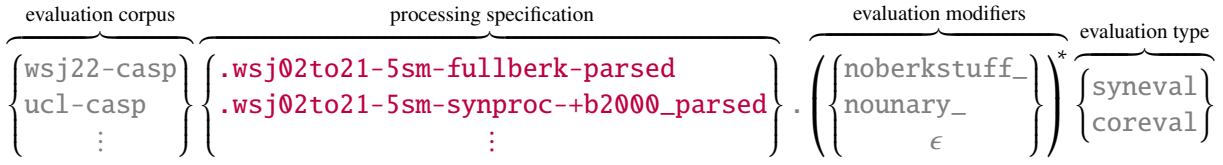
- `fullberk` and `vitberk`: Modelblocks supports the use of full-reranking and Viterbi-only versions of the Berkeley parser (Petrov and Klein, 2007), which provides unsupervised latent-variable annotation for categories through a number of iterations of a split-merge-smooth algorithm. This model requires a training option '`-`$\langle N \rangle$`sm`' specifying $\langle N \rangle$ iterations of the split-merge-smooth algorithm. Projects using this resource should include the `resource-lvpcfg` make file.

- `synproc`: Modelblocks provides an incremental left-corner parser (van Schijndel et al., 2013) based on a transform of the Berkeley grammar inducer (Petrov et al., 2006). This model requires a training option '`-`$\langle N \rangle$`sm`' specifying $\langle N \rangle$ iterations of the split-merge-smooth algorithm. This model also requires a processing option '`+b`$\langle N \rangle$`_`' specifying a beam width of $\langle N \rangle$ elements. With the '`+c_`' option, this model produces a predictor named `totsurp_`. Projects using this resource should include the `resource-lvpcfg` and `resource-incrsem` make files.

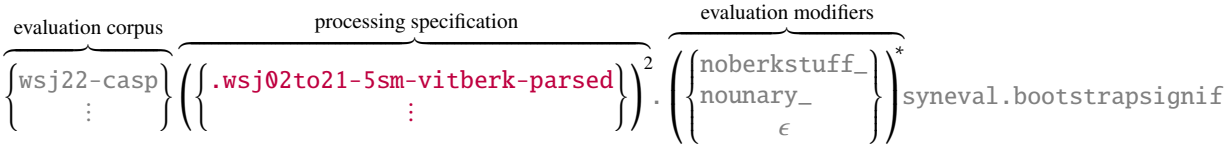
- `mlpsemproc`: Modelblocks provides an incremental left-corner parser augmented with predictors defined over contexts in a cued-association graph (Oh et al., 2021). This model requires a processing option '`+b`$\langle N \rangle$`_`' specifying a beam width of $\langle N \rangle$ elements, a processing option '`+u`$\langle N \rangle$`_`' specifying the maximum number of training corpus instances for unknown predicates and a processing option '`+c`$\langle N \rangle$`_`' specifying a coreference window of $\langle N \rangle$ time steps (Jaffe et al., 2020). Projects using this resource should include the `resource-incrsem` make file.

2.2 Linguistic accuracy evaluation

Filenames of make targets that contain evaluations of linguistic predictions each consist of an **evaluation corpus** followed by a **processing specification**, followed by zero or more **evaluation modifiers**, followed by an **evaluation type**:



Filenames of make targets that report significant differences between evaluations of linguistic predictions each consist of an **evaluation corpus** followed by two **processing specifications**, followed by zero or more **evaluation modifiers**:



Linguistically-annotated evaluation corpus. Evaluation corpora are partitioned into development sets, on which exploratory experiments should be conducted (e.g. parameter tuning), and test corpora, on which confirmatory experiments should be conducted (with appropriate multiple trials correction). Modelblocks supports the following evaluation corpora:

- `wsj22-nodashtags` and `wsj23-nodashtags`: Standard development and test partitions of the Penn Treebank (Marcus et al., 1994) each consist of about 2000 bracketed sentences of newspaper text from the Wall Stree Journal. Projects using this resource should include the `resource-treebank` make file.

- `wsj22-casp` and `wsj23-casp`: Standard development and test partitions of the Penn Treebank (Marcus et al., 1994) may also be re-annotated into cued-association sentence processing markup (Nguyen et al., 2012). Projects using this resource should include the `resource-treebank` and `resource-casp` make files.

- `chtb-dev-nodashtags` and `chtb-test-nodashtags`: Standard development and test partitions of the Penn Chinese Treebank (Xue et al., 2005) each consist of about 2000 bracketed sentences of newspaper text from the Wall Stree Journal. Projects using this resource should include the `resource-chtb` make file.

- `chtb-dev-casp` and `chtb-test-casp`: Standard development and test partitions of the Penn Chinese Treebank (Xue et al., 2005) may also be re-annotated into cued-association sentence processing markup (Duan and Schuler, 2015). Projects using this resource should include the `resource-chtb` and `resource-casp` make files.

- `dundee-casp`: Modelblocks defines development and test sets of the Dundee Corpus (Kennedy et al., 2003) each consisting of about 1000 bracketed sentences of newspaper text from the Independent. Projects using this resource should include the `resource-dundee` make file.

- `ucl-casp`: Modelblocks defines development and test partitions of the UCL Corpus (Frank et al., 2013) each consisting of about 100 bracketed sentences of narrative text from amateur stories. Projects using this resource should include the `resource-ucl` make file.

- `naturalstories-casp`: Modelblocks defines development and test partitions of the Natural Stories Corpus (Futrell et al., 2018) each consisting of about 500 bracketed sentences of narrative text from stories constructed to test memory in human sentence processing. Projects using this resource should include the `resource-naturalstories` make file.

Processing specifications. Processing specifications specify training data, models and modifiers as described in Section 2.1.

Evaluation modifiers. Modelblocks supports the following evaluation modifiers:

- `noberkstuff_`: This modifier removes latent variables induced by the Berkeley latent variable inducer from both predicted and attested trees.

- `nounary_`: This modifier removes unary projections from both predicted and attested trees.

- `nol_`: This modifier removes casp operator tags from both predicted and attested trees.
- `nopunc_`: This modifier removes punctuation from both predicted and attested trees, for syntactic evaluation.

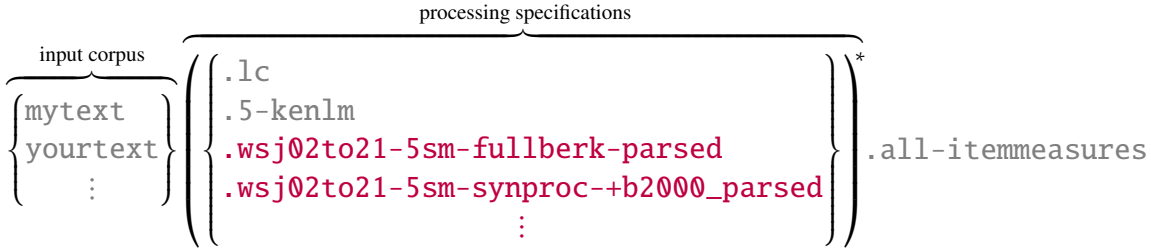
Evaluation types. Modelblocks supports the following evaluation types:

- `syneval`: This evaluates the parsing accuracy of a model, reporting the percent of hypothesized constituents that match to gold attested constituents (as recall) and vice versa (as precision).
- `coreval`: This evaluates the coreference accuracy of a model, reporting the percent of hypothesized antecedents of anaphors that match to gold attested antecedents (as recall) and vice versa (as precision).

3 Psycholinguistic prediction and evaluation

3.1 Psycholinguistic prediction

Psycholinguistic predictions are estimates of responses obtained by regressing sets of predictors against human comprehension effort. Psycholinguistic predictors in `all-itemmeasures` files are organized into matrices consisting of sequences of words (rows), each with a sequence of predictors (columns). Filenames of make targets that contain psycholinguistic predictors each consist of a specification of an **input corpus** followed by one or more **processing specifications**:



Modelblocks provides standard column names for the following predictors: subject id (`subject_`), word type (`word_`), word position in sentence (`sentpos_`), sentence position in document (`sentid_`), document id (`docid_`, word length (`wlen_`) in all corpora. For all self-paced reading corpora, Modelblocks additionally provides standard column names for the reading time response (`fdur_`). For all eye-tracking corpora, Modelblocks additionally provides standard column names for the following responses: first-pass fixation durations (`fdurFP_`), go-past fixation durations (`fdurGP_`), scan-path fixation durations (`fdurSP_`), and the following predictor: saccade distance in words (`wdelta_`). For all fMRI corpora, Modelblocks additionally provides standard column names for the following predictors: repitition time (sample) number (`tr_`) and functional region of interest (`fROI_`), and the following by-region responses: BOLD (`BOLD_`). Duplicate predictors are renamed with a numerical suffix according to the order of the predictor, e.g. the second instance of `totsurp_` is listed as `totsurp2_`.

Input corpus. Modelblocks supports segmented and tokenized `.senttoks` or unstructured text `.txt` files as described in Section 2.1.

Psycholinguistic processing specifications. Each processing specification adds one or more additional predictor columns to the matrix of item measures. Processing specifications specify training data, models and modifiers as described in Section 2.1. Modelblocks additionally supports the following processing specifications which are not derived from parsing:

- `.lc`: Modelblocks supports predictors derived from left-corner parsing operations in gold-standard annotations, including: end of constituent (`noF_` or `yesL_`), end of multi-word embedding (`yesJ_` or `yesG_`), and center-embedding depth (`embd_`).
- `.dlt`: Modelblocks supports predictors derived from Dependency Locality Theory (Gibson, 2000) in gold-standard annotations: standard DLT (`dlt_`) and the modified version of Shain et al. (2022) (`dltcvm_`).
- `.syncat`: (For naturalstories only) Modelblocks supports the generalized categorial grammar (Nguyen et al., 2012) syntactic categories associated with each word.
- `.semop`: Modelblocks supports the generalized categorial grammar (Nguyen et al., 2012) semantic operators (`opAa_`, `opAb_`, `opMa_`, `opMb_`, `opCa_`, `opCb_`, `opE_`, `opV_`, `opZ_`) associated with CASP lexical and grammatical inference rules (see CASP documentation on this web site).
- `.`$\langle N \rangle$`-kenlm`: Modelblocks also supports surprisal estimates from smoothed backed-off $\langle N \rangle$-gram language models, using the KenLM toolkit (Heafield et al., 2013), as predictor '`fwprob`$\langle N \rangle$`_`'.
- `.unigram`: Modelblocks supports unigram predictors from KenLM (Heafield et al., 2013) as a separate predictor '`unigramsurp_`'.
- `.glstm`: Modelblocks also supports surprisal estimates from the Gulordava et al. (2018) LSTM implementation as predictor '`glstmsurp_`'. This processing model requires an initial pre-processing run with a non-standard conda environment: first deactivate your current envrionment (if you had been using one) with 'conda deactivate', then create the non-standard environment with 'make glstm_env', then activate it with 'conda activate glstm_env', then execute the pre-processing run with 'make genmodel/<corpus>.glstm.itemmeasures', then deactivate the non-standard environment with 'conda deactivate', then switch back to your modelblocks environment (if you had been using one) with 'conda activate mb', and execute your modelblocks target as usual (it will use the pre-processed `.itemmeasures` target).
- `.jlstm`: Modelblocks also supports surprisal estimates from the Jozefowicz et al. (2016) LSTM implementation as predictor '`jlstmsurp_`'. This processing model requires an initial pre-processing run with a non-standard conda environment: first deactivate your current envrionment (if you had been using one) with 'conda deactivate', then create the non-standard environment with 'make jlstm_env', then activate it with 'conda activate jlstm_env', then execute the pre-processing run with 'make genmodel/<corpus>.jlstm.itemmeasures', then

deactivate the non-standard environment with 'conda deactivate', then switch back to your modelblocks environment (if you had been using one) with 'conda activate mb', and execute your modelblocks target as usual (it will use the pre-processed `.itemmeasures` target).

- `.gpt{2,2-medium,2-large,2-xl}`: Modelblocks also supports surprisal estimates from GPT-2 (Radford et al., 2019) as predictor '`totsurp_`'. This processing model requires an initial pre-processing run with a non-standard conda environment: first deactivate your current envrionment (if you had been using one) with 'conda deactivate', then create the non-standard environment with 'make hf_env', then activate it with 'conda activate hf_env', then execute the pre-processing run with 'make genmodel/<corpus>.gpt{2,2-medium,2-large,2-xl}.itemmeasures', then deactivate the non-standard environment with 'conda deactivate', then switch back to your modelblocks environment (if you had been using one) with 'conda activate mb', and execute your modelblocks target as usual (it will use the pre-processed `.itemmeasures` target).

- `.opt-{125m,350m,1300m,2700m,6700m,13000m,30000m,66000m}`: Modelblocks also supports surprisal estimates from OPT (Zhang et al., 2022) as predictor '`totsurp_`'. This processing model requires an initial pre-processing run with a non-standard conda environment: first deactivate your current envrionment (if you had been using one) with 'conda deactivate', then create the non-standard environment with 'make hf_env', then activate it with 'conda activate hf_env', then execute the pre-processing run with 'make genmodel/<corpus>.opt-{125m,350m,1300m,2700m,6700m,13000m,30000m,66000m}.itemmeasures', then deactivate the non-standard environment with 'conda deactivate', then switch back to your modelblocks environment (if you had been using one) with 'conda activate mb', and execute your modelblocks target as usual (it will use the pre-processed `.itemmeasures` target).

- `.gpt-neo{-125m,-1300m,-2700m,x-20000m}`: Modelblocks also supports surprisal estimates from GPT-neo (Black et al., 2021, 2022) as predictor '`totsurp_`'. This processing model requires an initial pre-processing run with a non-standard conda environment: first deactivate your current envrionment (if you had been using one) with 'conda deactivate', then create the non-standard environment with 'make hf_env', then activate it with 'conda activate hf_env', then execute the pre-processing run with 'make genmodel/<corpus>.gpt-neo{-125m,-1300m,-2700m,x-20000m}.itemmeasures', then deactivate the non-standard environment with 'conda deactivate', then switch back to your modelblocks environment (if you had been using one) with 'conda activate mb', and execute your modelblocks target as usual (it will use the pre-processed `.itemmeasures` target).

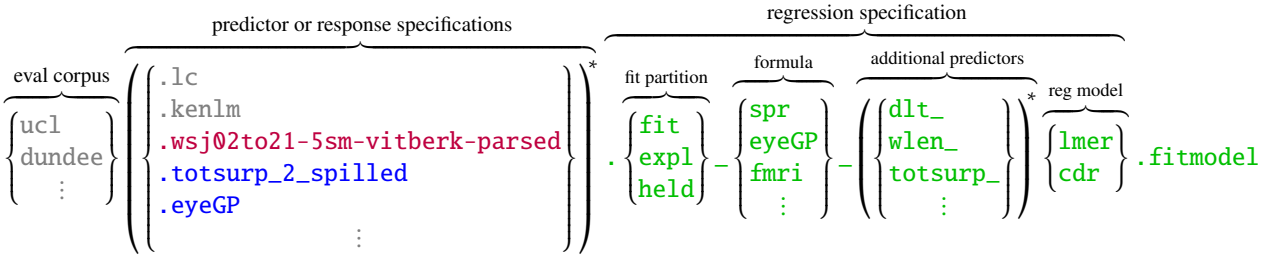
- `.gpt-j-6000m`: Modelblocks also supports surprisal estimates from GPT-j (Wang and Komatsuzaki, 2021) as predictor '`totsurp_`'. This processing model requires an initial pre-processing run with a non-standard conda environment: first deactivate your current envrionment (if you had been using one) with 'conda deactivate', then create the non-standard environment with 'make hf_env', then activate it with 'conda activate hf_env', then execute the pre-processing run with 'make genmodel/<corpus>.gpt-j-6000m.itemmeasures', then deactivate the non-standard environment with 'conda deactivate', then switch back to your modelblocks environment (if you had been using one) with 'conda activate mb', and

execute your modelblocks target as usual (it will use the pre-processed `.itemmeasures` target).

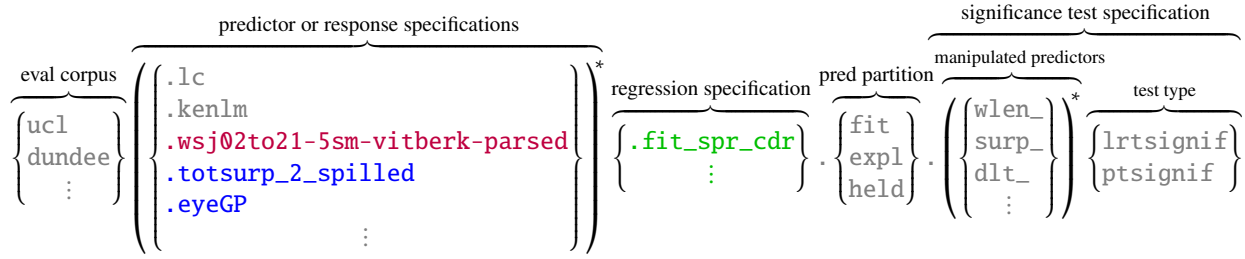
- `.pythia-`⟨M⟩`-`⟨N⟩: Modelblocks also supports surprisal estimates from Pythia (Biderman et al., 2023) as predictor '`totsurp_`', where ⟨M⟩ ∈ {`70m`, `160m`, `410m`, `1000m`, `1400m`, `2800m`, `6900m`, `12000m`} is the models size and ⟨N⟩ ∈ {`0`, `1`, `2`, `4`, `8`, `16`, `32`, `64`, `128`, `256`, `512`, `1000`, `2000`, ...} is the number of training epochs. This processing model requires an initial pre-processing run with a non-standard conda environment: first deactivate your current envrionment (if you had been using one) with 'conda deactivate', then create the non-standard environment with 'make hf_env', then activate it with 'conda activate hf_env', then execute the pre-processing run with 'make genmodel/<corpus>.pythia-*M*-*N*.itemmeasures', then deactivate the non-standard environment with 'conda deactivate', then switch back to your modelblocks environment (if you had been using one) with 'conda activate mb', and execute your modelblocks target as usual (it will use the pre-processed `.itemmeasures` target).

3.2 Psycholinguistic accuracy evaluation

Psycholinguistic predictors are evaluated by the magnitude of their coefficients when used as predictors in regression to reading times or other dependent measures. Filenames of make targets that contain evaluations of psycholinguistic predictions each consist of an **evaluation corpus** followed by a set of one or more **predictor or response specifications**, followed by a **regression specification**. The set of predictor or response specifications consists of zero or more processing specifications, followed by zero or more additional predictor or response specifications. Each regression specification consists of a fit partition, followed by a **base formula**, followed by one or more additional predictors, followed by a **regression model**:



Filenames of make targets that report significant differences in evaluations of psycholinguistic predictions each consist of an evaluation corpus followed by a set of one or more predictor or response specifications, followed by a regression specification, followed by a prediction partition, followed by a **significance test specification**. Each significance test specification consists of one or two manipulated predictor names, followed by a **significance test type**:



a)

model	F
wsj22-gcg15.nol.goldfailerr.wsj02to21-casp-nol-4sm-fullberk-parsed.noberkstuff_nodelim_nopunc_nounary_syneval	86.5
wsj22-gcg15.nol.goldfailerr.wsj02to21-gcg15-nol-prtrm-3sm-synproc-+u_+b5000_parsed.noberkstuff_nodelim_nopunc_nounary_syneval	84.1
wsj22-gcg15.nol.goldfailerr.wsj02to21-gcg15-nol-prtrm-4sm-synproc-+u_+b5000_parsed.noberkstuff_nodelim_nopunc_nounary_syneval	85.2
wsj22-casp.goldfailerr.wsj02to21-casp-+u10_+c0_bestmlp_mlpsemproc-+c0_+b2000_parsed.nodelim_nopunc_nounary_nol_syneval	84.4
corconllwsjdev-casp.corontowsj02to21-casp-+u5_+c100_ontocorfix_bestmlp_mlpsemproc-+c100_+b2000_parsed.corefeval	
corconllwsjdev-casp.corontowsj02to21-casp-+u5_+c100_ontocorfix_bestmlp_mlpsemproc-+c100_+b2000_parsed.coreval	46.1

b)

```
dundee.wsj02to21-0sm-synproc-+c_+u_+b2000_parsed.5-kenlm.totsurp_1_spilled.eyeGP.fit_eyeGP_fwprob5surp_lmer.fit.totsurpS1_lrtsignif
naturalstories.wsj02to21-0sm-synproc-+c_+u_+b2000_parsed.5-kenlm.unigram.spr.fit_spr_cdr.expl.totsurp_ptsignif
naturalstoriesfmri_Lang.t.wsj02to21-casp-+u5_+c0_+rp_bestmlp_mlpsemproc-+c0_+b2000_+rp_parsed.dlt.fmri-bywrd.fit_fmri_cdr.fit.dltcvm_totsurp_ctsignif
```

Table 1: Sample working parse and weighted pronoun link evaluations using cued-association sentence processing model (a), diamond likelihood ratio test significance report item, comparing word length against total surprisal with spillover 1 (b).

An example make target for a diamond likelihood ratio test significance report is shown in Table 1.

Psycholinguistically-annotated evaluation corpus. Modelblocks supports the following evaluation corpora, which provide psycholinguitic annotations from human subjects:

- `alice`: Modelblocks supports experimentation on the Alice Corpus (Brennan et al., 2016) consisting of about 2100 words of narrative text from *Alice in Wonderland*, annotated with fMRI BOLD responses from 29 subjects. Projects using this resource should include the `resource-alice` make file and may use the `fmri` baseline regression formula. This resource defines a `fit` partition consisting of responses whose summed subject and sentence number have modulo four equal to zero or one, an `expl` partition consisting of responses whose summed subject and sentence number have modulo four equal to two, and a `held` partition consisting of all other responses.

- `brown`: Modelblocks supports experimentation on the Brown Corpus (Smith and Levy, 2013) consisting of about 500 sentences of text from books and periodicals, annotated with self-paced reading responses from approximately 35 subjects. Projects using this resource should include the `resource-brown` make file and may use the `spr` baseline regression formula. This resource defines a `fit` partition consisting of responses whose summed subject and sentence number have modulo four equal to zero or one, an `expl` partition consisting of responses whose summed subject and sentence number have modulo four equal to two, and a `held` partition consisting of all other responses.

- `dundee`: Modelblocks supports experimentation on the Dundee Corpus (Kennedy et al., 2003) consisting of about 2000 sentences of newspaper text from the Independent, annotated with eye-tracking responses from 10 subjects. Projects using this resource should include the `resource-dundee` make file and may use the `eyeGP`, `eyeFP`, or `eyeSP` baseline regression formula. This resource defines a `fit` partition consisting of responses whose summed subject and sentence number have modulo four equal to zero or one, an `expl` partition consisting of responses whose summed subject and sentence number have modulo four equal to two, and a `held` partition consisting of all other responses.

- `geco`: Modelblocks supports experimentation on the GECO Corpus (Cop et al., 2016) consisting of an entire Agatha Christie novel, annotated with eye-tracking responses from 14

subjects. Projects using this resource should include the `resource-geco` make file and may use the `eyeGP`, `eyeFP`, or `eyeSP` baseline regression formula. This resource defines a `fit` partition consisting of responses whose summed subject and sentence number have modulo four equal to zero or one, an `expl` partition consisting of responses whose summed subject and sentence number have modulo four equal to two, and a `held` partition consisting of all other responses.

- `naturalstories`: Modelblocks supports experimentation on the Natural Stories Corpus (Futrell et al., 2018) consisting of about 500 sentences of narrative text from stories constructed to test memory in human sentence processing, annotated with self-paced reading responses from approximately 150 subjects. Projects using this resource should include the `resource-naturalstories` make file and may use the `spr` baseline regression formula. This resource defines a `fit` partition consisting of responses whose summed subject and sentence number have modulo four equal to zero or one, an `expl` partition consisting of responses whose summed subject and sentence number have modulo four equal to two, and a `held` partition consisting of all other responses.

- `naturalstories_Lang.t` and `naturalstories_MD.t`: Modelblocks also supports experimentation on the Natural Stories fMRI corpus (Shain et al., 2020) consisting of about 500 sentences of narrative text from stories constructed to test memory in human sentence processing, annotated with fMRI BOLD data from Language (Lang) and Multiple Demand (MD) networks of approximately 150 subjects. Projects using this resource should include the `resource-naturalstoriesfmri` make file and may use the `fmri` baseline regression formula. This resource defines a `fit` partition consisting of responses whose summed subject and sentence number have modulo four equal to zero or one, an `expl` partition consisting of responses whose summed subject and sentence number have modulo four equal to two, and a `held` partition consisting of all other responses.

- `pereira-sent`: Modelblocks supports experimentation on the Pereira et al. (2018) corpus, consisting of about 384 sentences, annotated with per-sentence BOLD fMRI responses from 16 subjects. Projects using this resource should include the `resource-pereira` make file and may use the `fmri-sent` baseline regression formula. This resource defines a `fit` partition consisting of responses whose summed subject and sentence number have modulo four equal to zero or one, an `expl` partition consisting of responses whose summed subject and sentence number have modulo four equal to two, and a `held` partition consisting of all other responses.

- `provo`: Modelblocks supports experimentation on the PROVO Corpus (Luke and Christianson, 2017) consisting of about 150 sentences, annotated with eye-tracking responses from 84 subjects. Projects using this resource should include the `resource-provo` make file and may use the `eyeGP`, `eyeFP`, or `eyeSP` baseline regression formula. This resource defines a `fit` partition consisting of responses whose summed subject and sentence number have modulo four equal to zero or one, an `expl` partition consisting of responses whose summed subject and sentence number have modulo four equal to two, and a `held` partition consisting of all other responses.

- `ucl`: Modelblocks supports experimentation on the UCL Corpus (Frank et al., 2013) consisting of about 100 sentences of narrative text from amateur stories, annotated with eye-tracking responses from about 40 subjects. Projects using this resource should include the `resource-ucl` make file and may use the `eyeGP`, `eyeFP`, or `eyeSP` baseline regression formula. This resource defines a `fit` partition consisting of responses whose summed subject and sentence number have modulo four equal to zero or one, an `expl` partition consisting of responses whose summed subject and sentence number have modulo four equal to two, and a `held` partition consisting of all other responses.

Predictor or response specifications. Psycholinguistic evaluations define predictor or response specifications, which subsume all psycholinguistic processing specifications as described in Section 3.1, as well as the following additional predictor or response specifications:

- `.`⟨*P*⟩`_`⟨*N*⟩`_spilled`: Modelblocks supports ⟨*N*⟩-word spillover of any predictor ⟨*P*⟩. This specification creates an additional predictor named ⟨*P*⟩`S`⟨*N*⟩`_`.
- `.`⟨*P*⟩`_`⟨*N*⟩`_futured`: Modelblocks supports ⟨*N*⟩-word future spillover of any predictor ⟨*P*⟩ (van Schijndel and Schuler, 2016). This specification creates an additional predictor named ⟨*P*⟩`F`⟨*N*⟩`_`.
- `.`⟨*P*⟩`_cumued`: Modelblocks supports accumulation of any predictor ⟨*P*⟩ (van Schijndel and Schuler, 2016). This specification creates an additional predictor named `cumu`⟨*P*⟩`_`.
- `.`⟨*P*1⟩`_`⟨*P*2⟩`_delta`: Modelblocks supports specification of predictors defined to be the difference between any two predictors ⟨*P*1⟩ - ⟨*P*2⟩ This specification creates an additional predictor named `d`⟨*P*1⟩⟨*P*2⟩`_`.
- `.hrf`: Modelblocks supports convolution of predictors by the canonical Hemodynamic Response Function (Boynton et al., 1996). This specification modifies all predictors to be convolved over time based on time stamps of the stimulus events.
- `.eyeFP`: Modelblocks supports first-pass durations as responses. This specification creates an additional response named `fdurFP`. This specification also excludes start-of-sentence, end-of-sentence, start-of-line, end-of-line, start-of-screen, end-of-screen, start-of-file, and end-of-file responses from evaluation, and any response resulting from a saccade that skips more than four words (Demberg and Keller, 2008).
- `.eyeGP`: Modelblocks supports go-past durations as responses. This specification creates a response named `fdurGP`. This specification also excludes start-of-sentence, end-of-sentence, start-of-line, end-of-line, start-of-screen, end-of-screen, start-of-file, and end-of-file responses from evaluation, and any response resulting from a saccade that skips more than four words (Demberg and Keller, 2008).
- `.eyeSP`: Modelblocks supports scan-path durations as responses. This specification creates a response named `fdurSP`. This specification also excludes start-of-sentence, end-of-sentence, start-of-line, end-of-line, start-of-screen, end-of-screen, start-of-file, and end-of-file responses from evaluation, and any response resulting from a saccade that skips more than four words (Demberg and Keller, 2008).

- `.spr`: Modelblocks supports self-paced reading durations as responses. This specification creates a response named `fdur`. This specification also excludes start-of-sentence, end-of-sentence responses from evaluation, and any response less than 100ms or greater than 3000ms duration, and any response with a correct value of less than four.

- `.fmri-bywrd`: Modelblocks supports fMRI BOLD data as responses. This specification creates a response named `BOLD`.

Regression model, baseline formula. Modelblocks supports the following regression models:

- `lmer`: Modelblocks supports linear mixed effects regression (LMER; Baayen et al., 2008). LMER requires the designation of a formula file with suffix `.lmerform` in a project `scripts` directory, which specifies how predictors are used in the fixed and random effects structure of the model. In this file:

  - Line 1 specifies a single **response variable**, e.g. `log(fdur)`.
  - Line 2 specifies the predictor variables for which to calculate **fixed effects**, e.g. `z.(sentpos) + z.(wlen)`.
  - Line 3 specifies the predictor variables for which to calculate **by-subject random effects**, e.g. `z.(sentpos) + z.(wlen)`. This line cannot be empty.
  - Line 4 optionally specifies zero or more **additional random effects**, e.g. `(1 | word) + (1 | sentid:subject)`. This line must specify a grouping factor, or if empty, must not contain a trailing newline.

  Additional modifiers specify additional parameters to the model. Modelblocks provides simple baseline `spr`, `eyeFP`, `eyeGP`, `eyeSP` and `fmri .lmerform` files for self-paced reading, eye-tracking using first-pass durations, eye-tracking using go-past durations, eye-tracking using scan-path durations and fMRI data, which can be modified in your project `scripts` directory. Projects using this resource should include the `resource-lmefit` make file. Full documentation of LMER hyperparameters, formula syntax, and config file syntax can be found at `https://www.rdocumentation.org/packages/lme4/versions/1.1-23/topics/lmer/`.

- `nmrelmer`: Modelblocks supports linear mixed effects regression (LMER; Baayen et al., 2008) with no main random effects. This option is like `lmer` but does not add in a random slope for the main effect. Per-subject random intercepts are still added.

- `cdr`: Modelblocks supports continuous deconvolutional regression (CDR; Shain and Schuler, 2018). CDR requires the designation of a formula file with suffix `.cdrform` in a project `scripts` directory, which is a template for a CDR config file that specifies how predictors are used in the fixed and random effects structure of the model. Additional modifiers specify additional parameters to the model. Modelblocks provides simple baseline `spr`, `eyeFP`, `eyeGP`, `eyeSP` and `fmri .cdrform` files for self-paced reading, eye-tracking using first-pass durations, eye-tracking using go-past durations, eye-tracking using scan-path durations and fMRI data, which can be modified in your project `scripts` directory.

Projects using this resource should include the `resource-cdr` make file. Full documentation of CDR hyperparameters, formula syntax, and config file syntax can be found at `https://cdr.readthedocs.io/en/latest/`. Modelblocks assumes CDR commit 9b51 from 2022-05-06.

Significance test type. Modelblocks supports the following statistical significance testing methods:

- `lrtsignif`: Modelblocks supports likelihood ratio testing (LRT) as a means of determining the contribution of a predictor of interest by comparing the estimated likelihood of the modeled variables corpus given a baseline formula with and without that predictor as a fixed affect. LRT requires the designation of a predictor of interest. Additional modifiers specify additional parameters to the model. Projects using this resource should include the `resource-lr` make file. The LRT significance test cannot be used with CDR.

- `ptsignif`: Modelblocks supports permutation testing (PT) as a means of determining the contribution of a predictor of interest by comparing the estimated likelihood of the modeled variables corpus given a baseline formula with and without that predictor as a fixed effect. PT requires the designation of a predictor of interest. Additional modifiers specify additional parameters to the model.

- `ctsignif`: Modelblocks supports permutation testing using correlation (CT) as a means of determining the contribution of a predictor of interest by comparing the estimated likelihood of the modeled variables corpus given a baseline formula with and without that predictor as a fixed effect. CT requires the designation of a predictor of interest. Additional modifiers specify additional parameters to the model.

References

Baayen, R. H., Davidson, D. J., and Bates, D. M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59:390–412.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., Skowron, A., Sutawika, L., and Van Der Wal, O. (2023). Pythia: A suite for analyzing large language models across training and scaling. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., Pieler, M., Prashanth, U. S., Purohit, S., Reynolds, L., Tow, J., Wang, B., and Weinbach, S. (2022). GPT-NeoX-20B: An open-source autoregressive language model. In Fan, A., Ilic, S., Wolf, T., and Gallé, M., editors, *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin. Association for Computational Linguistics.

Black, S., Gao, L., Wang, P., Leahy, C., and Biderman, S. (2021). GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Boynton, G. M., Engel, S. A., Glover, G. H., and Heeger, D. J. (1996). Linear systems analysis of functional magnetic resonance imaging in human v1. *Journal of Neuroscience*, 16(13):4207–4221.

Brennan, J., Stabler, E. P., Van Wagenen, S. E., Luh, W.-M., and Hale, J. T. (2016). Abstract linguistic structure correlates with temporal activity during naturalistic comprehension. *Brain and Language*, 157–158:81–94.

Cop, U., Dirix, N., Drieghe, D., and Duyck, W. (2016). Presenting geco: An eyetracking corpus of monolingual and bilingual sentence reading. *Behavior Research Methods*, 49.

Demberg, V. and Keller, F. (2008). Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.

Duan, M. and Schuler, W. (2015). Parsing chinese with a generalized categorial grammar. In *Proceedings of the Grammar Engineering Across Frameworks Workshop (GEAF'15)*, pages 25–32.

Frank, S. L., Fernandez Monsalve, I., Thompson, R. L., and Vigliocco, G. (2013). Reading time data for evaluating broad-coverage models of English sentence processing. *Behavior Research Methods*, 45:1182–1190.

Futrell, R., Gibson, E., Tily, H., Vishnevetsky, A., Piantadosi, S., and Fedorenko, E. (2018). The Natural Stories corpus. In *LREC 2018*.

Gibson, E. (2000). The dependency locality theory: A distance-based theory of linguistic complexity. In *Image, language, brain: Papers from the first mind articulation project symposium*, pages 95–126, Cambridge, MA. MIT Press.

Gulordava, K., Bojanowski, P., Grave, E., Linzen, T., and Baroni, M. (2018). Colorless green recurrent networks dream hierarchically. In *NAACL-HLT*, pages 1195–1205.

Heafield, K., Pouzyrevsky, I., Clark, J. H., and Koehn, P. (2013). Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

Jaffe, E., Shain, C., and Schuler, W. (2020). Coreference information guides human expectations during natural reading. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4587–4599, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. *CoRR*.

Kennedy, A., Pynte, J., and Hill, R. (2003). The Dundee corpus. In *Proceedings of the 12th European conference on eye movement*.

Luke, S. and Christianson, K. (2017). The provo corpus: A large eye-tracking corpus with predictability norms. *Behavior research methods*, 50.

Marcus, M., Kim, G., Marcinkiewicz, M. A., Bies, R. M. A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The {Penn} {T}ree{B}ank: Annotating predicate argument structure. In *Proceedings of the ARPA Human Language Technology Workshop*.

Nguyen, L., van Schijndel, M., and Schuler, W. (2012). Accurate unbounded dependency recovery using generalized categorial grammars. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING '12)*, pages 2125–2140, Mumbai, India.

Oh, B.-D., Clark, C., and Schuler, W. (2021). Surprisal estimators for human reading times need character models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3746–3757, Online. Association for Computational Linguistics.

Pereira, F., Lou, B., Pritchett, B., Ritter, S., Gershman, S. J., Kanwisher, N. G., Botvinick, M. M., and Fedorenko, E. (2018). Toward a universal decoder of linguistic meaning from brain activation. *Nature Communications*, 9.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL'06)*.

Petrov, S. and Klein, D. (2007). Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*, pages 404–411, Rochester, New York. Association for Computational Linguistics.

Pradhan, S. S., Hovy, E. H., Marcus, M. P., Palmer, M., Ramshaw, L. A., and Weischedel, R. M. (2007). Ontonotes: A unified relational semantic representation. In *ICSC*, pages 517–526.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *ArXiv*.

Shain, C., Blank, I. A., Fedorenko, E., Gibson, E., and Schuler, W. (2022). Robust effects of working memory demand during naturalistic language comprehension in language-selective cortex. *Journal of Neuroscience*, 42(39):7412–7430.

Shain, C., Blank, I. A., van Schijndel, M., Schuler, W., and Fedorenko, E. (2020). fMRI reveals language-specific predictive coding during naturalistic sentence comprehension. *Neuropsychologia*, 138.

Shain, C. and Schuler, W. (2018). Deconvolutional time series regression: A technique for modeling temporally diffuse effects. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP'18)*, pages 2679–2689.

Smith, N. J. and Levy, R. (2013). The effect of word predictability on reading time is logarithmic. *Cognition*, 128:302–319.

van Schijndel, M., Exley, A., and Schuler, W. (2013). A model of language processing as hierarchic sequential prediction. *Topics in Cognitive Science*, 5(3):522–540.

van Schijndel, M. and Schuler, W. (2016). Addressing surprisal deficiencies in reading time models. In *Proceedings of the Computational Linguistics for Linguistic Complexity Workshop*. Association for Computational Linguistics.

Wang, B. and Komatsuzaki, A. (2021). Gpt-j-6b: A 6 billion parameter autoregressive language model.

Xue, N., Xian, F., Chiou, F.-D., and Palmer, M. (2005). The Penn Chinese Treebank: Phrase Structure annotation of a large corpus. *Natural Language Engineering*, 11:207–238.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. (2022). Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.