Package: pseudobibeR (via r-universe)

February 18, 2025

Title Aggregate Counts of Linguistic Features

Version 1.2

Date 2024-11-18

Description Calculates the lexicogrammatical and functional features described by Biber (1985) <doi:10.1515/ling.1985.23.2.337> and widely used for text-type, register, and genre classification tasks.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

LazyData TRUE

Depends R (>= 3.5.0)

Imports dplyr, purrr, quanteda, quanteda.textstats, rlang, stringr, tibble, magrittr

Suggests testthat (>= 3.0.0), udpipe

Config/testthat/edition 3

NeedsCompilation no

Author David Brown [aut, cre] (<https://orcid.org/0000-0001-7745-6354>), Alex Reinhart [aut] (<https://orcid.org/0000-0002-6658-514X>)

Maintainer David Brown <dwb2@andrew.cmu.edu>

Date/Publication 2024-11-19 12:20:06 UTC

Config/pak/sysreqs libicu-dev libxml2-dev

Repository https://browndw.r-universe.dev

RemoteUrl https://github.com/cran/pseudobibeR

RemoteRef HEAD

RemoteSha c8e0dabdae6bd2d037ab4b1d5cb6fe2e97321ca2

Contents

`biber` *Extract Biber features from a document parsed and annotated by spacyr or udpipe*

Description

Takes data that has been part-of-speech tagged and dependency parsed and extracts counts of features that have been used in Douglas Biber's research since the late 1980s.

Usage

```
biber(
  tokens,
  measure = c("MATTR", "TTR", "CTTR", "MSTTR", "none"),
  normalize = TRUE
)

## S3 method for class 'spacyr_parsed'
biber(
  tokens,
  measure = c("MATTR", "TTR", "CTTR", "MSTTR", "none"),
  normalize = TRUE
)

## S3 method for class 'udpipe_connlu'
biber(
  tokens,
  measure = c("MATTR", "TTR", "CTTR", "MSTTR", "none"),
  normalize = TRUE
)
```

Arguments

`tokens`	A dataset of tokens created by `spacyr::spacy_parse()` or `udpipe::udpipe_annotate()`
`measure`	Measure to use for type-token ratio. Passed to `quanteda.textstats::textstat_lexdiv()` to calculate the statistic. Can be the Moving Average Type-Token Ratio (MATTR), ordinary Type-Token Ratio (TTR), corrected TTR (CTTR), Mean Segmental Type-Token Ratio (MSTTR), or `"none"` to skip calculating a type-token ratio. If a statistic is chosen but there are fewer than 200 token in the smallest document, the TTR is used instead.

`normalize` If `TRUE`, count features are normalized to the rate per 1,000 tokens.

Details

Refer to `spacyr::spacy_parse()` or `udpipe::udpipe_annotate()` for details on parsing texts. These must be configured to do part-of-speech and dependency parsing. For `spacyr::spacy_parse()`, use the `dependency = TRUE`, `tag = TRUE`, and `pos = TRUE` arguments; for `udpipe::udpipe_annotate()`, set the `tagger` and `parser` arguments to `"default"`.

Feature extraction relies on a dictionary (included as `dict`) and word lists (`word_lists`) to match specific features; see their documentation and values for details on the exact patterns and words matched by each. The function identifies other features based on local cues, which are approximations. Because they rely on probabilistic taggers provided by spaCy or udpipe, the accuracy of the resulting counts are dependent on the accuracy of those models. Thus, texts with irregular spellings, non-normative punctuation, etc. will likely produce unreliable outputs, unless taggers are tuned specifically for those purposes.

The following features are detected. Square brackets in example sentences indicate the location of the feature.

Tense and aspect markers:

f_01_past_tense Verbs in the past tense.

f_02_perfect_aspect Verbs in the perfect aspect, indicated by "have" as an auxiliary verb (e.g. *I [have] written this sentence.*)"

f_03_present_tense Verbs in the present tense.

Place and time adverbials:

f_04_place_adverbials Place adverbials (e.g., *above*, *beside*, *outdoors*; see list in `dict$f_04_place_adverbials`)

f_05_time_adverbials Time adverbials (e.g., *early*, *instantly*, *soon*; see `dict$f_05_time_adverbials`)

Pronouns and pro-verbs:

f_06_first_person_pronouns First-person pronouns; see `dict$f_06_first_person_pronouns`

f_07_second_person_pronouns Second-person pronouns; see `dict$f_07_second_person_pronouns`

f_08_third_person_pronouns Third-person personal pronouns (excluding *it*); see `dict$f_08_third_person_pronouns`

f_09_pronoun_it Pronoun *it*, *its*, or *itself*

f_10_demonstrative_pronoun Pronouns being used to replace a noun (e.g. *[That] is an example sentence.*)

f_11_indefinite_pronouns Indefinite pronouns (e.g., *anybody*, *nothing*, *someone*; see `dict$f_11_indefinite_pronouns`

f_12_proverb_do Pro-verb *do*

Questions:

f_13_wh_question Direct *wh-* questions (e.g., *When are you leaving?*)

Nominal forms:

f_14_nominalizations Nominalizations (nouns ending in *-tion*, *-ment*, *-ness*, *-ity*, e.g. *adjustment*, *abandonment*)

f_15_gerunds Gerunds (participial forms functioning as nouns)

f_16_other_nouns Total other nouns

Passives:

f_17_agentless_passives Agentless passives (e.g., *The task [was done].*)

f_18_by_passives *by*- passives (e.g., *The task [was done by Steve].*)

Stative forms:

f_19_be_main_verb *be* as main verb

f_20_existential_there Existential *there* (e.g., *[There] is a feature in this sentence.*)

Subordination features:

f_21_that_verb_comp *that* verb complements (e.g., *I said [that he went].*)

f_22_that_adj_comp *that* adjective complements (e.g., *I'm glad [that you like it].*)

f_23_wh_clause *wh*- clauses (e.g., *I believed [what he told me].*)

f_24_infinitives Infinitives

f_25_present_participle Present participial adverbial clauses (e.g., *[Stuffing his mouth with cookies], Joe ran out the door.*)

f_26_past_participle Past participial adverbial clauses (e.g., *[Built in a single week], the house would stand for fifty years.*)

f_27_past_participle_whiz Past participial postnominal (reduced relative) clauses (e.g., *the solution [produced by this process]*)

f_28_present_participle_whiz Present participial postnominal (reduced relative) clauses (e.g., *the event [causing this decline]*)

f_29_that_subj *that* relative clauses on subject position (e.g., *the dog [that bit me]*)

f_30_that_obj *that* relative clauses on object position (e.g., *the dog [that I saw]*)

f_31_wh_subj *wh*- relatives on subject position (e.g., *the man [who likes popcorn]*)

f_32_wh_obj *wh*- relatives on object position (e.g., *the man [who Sally likes]*)

f_33_pied_piping Pied-piping relative clauses (e.g., *the manner [in which he was told]*)

f_34_sentence_relatives Sentence relatives (e.g., *Bob likes fried mangoes, [which is the most disgusting thing I've ever heard of].*)

f_35_because Causative adverbial subordinator (*because*)

f_36_though Concessive adverbial subordinators (*although*, *though*)

f_37_if Conditional adverbial subordinators (*if*, *unless*)

f_38_other_adv_sub Other adverbial subordinators (e.g., *since*, *while*, *whereas*)

Prepositional phrases, adjectives, and adverbs:

f_39_prepositions Total prepositional phrases

f_40_adj_attr Attributive adjectives (e.g., *the [big] horse*)

f_41_adj_pred Predicative adjectives (e.g., *The horse is [big].*)

f_42_adverbs Total adverbs

Lexical specificity:

f_43_type_token Type-token ratio (including punctuation), using the statistic chosen in `measure`, or TTR if there are fewer than 200 tokens in the smallest document.

f_44_mean_word_length Average word length (across tokens, excluding punctuation)

Lexical classes:

f_45_conjuncts Conjuncts (e.g., *consequently*, *furthermore*, *however*; see `dict$f_45_conjuncts`)

f_46_downtoners Downtoners (e.g., *barely*, *nearly*, *slightly*; see `dict$f_46_downtoners`)

f_47_hedges Hedges (e.g., *at about*, *something like*, *almost*; see `dict$f_47_hedges`)

f_48_amplifiers Amplifiers (e.g., *absolutely*, *extremely*, *perfectly*; see `dict$f_48_amplifiers`)

f_49_emphatics Emphatics (e.g., *a lot*, *for sure*, *really*; see `dict$f_49_emphatics`)

f_50_discourse_particles Discourse particles (e.g., sentence-initial *well*, *now*, *anyway*; see `dict$f_50_discourse_parti`

f_51_demonstratives Demonstratives (*that*, *this*, *these*, or *those* used as determiners, e.g. *[That] is the feature*)

Modals:

f_52_modal_possibility Possibility modals (*can*, *may*, *might*, *could*)

f_53_modal_necessity Necessity modals (*ought*, *should*, *must*)

f_54_modal_predictive Predictive modals (*will*, *would*, *shall*)

Specialized verb classes:

f_55_verb_public Public verbs (e.g., *assert*, *declare*, *mention*; see `dict$f_55_verb_public`)

f_56_verb_private Private verbs (e.g., *assume*, *believe*, *doubt*, *know*; see `dict$f_56_verb_private`)

f_57_verb_suasive Suasive verbs (e.g., *command*, *insist*, *propose*; see `dict$f_57_verb_suasive`)

f_58_verb_seem *seem* and *appear*

Reduced forms and dispreferred structures:

f_59_contractions Contractions

f_60_that_deletion Subordinator *that* deletion (e.g., *I think [he went].*)

f_61_stranded_preposition Stranded prepositions (e.g., *the candidate that I was thinking [of]*)

f_62_split_infinitive Split infinitives (e.g., *He wants [to convincingly prove] that ...*)

f_63_split_auxiliary Split auxiliaries (e.g., *They [were apparently shown] to ...*)

Co-ordination:

f_64_phrasal_coordination Phrasal co-ordination (N and N; Adj and Adj; V and V; Adv and Adv)

f_65_clausal_coordination Independent clause co-ordination (clause-initial *and*)

Negation:

f_66_neg_synthetic Synthetic negation (e.g., *No answer is good enough for Jones.*)

f_67_neg_analytic Analytic negation (e.g., *That isn't good enough.*)

Value

A `data.frame` of features containing one row per document and one column per feature. If `normalize` is `TRUE`, count features are normalized to the rate per 1,000 tokens.

References

Biber, Douglas (1985). "Investigating macroscopic textual variation through multifeature/multidimensional analyses." *Linguistics* 23(2), 337-360. doi:10.1515/ling.1985.23.2.337

Biber, Douglas (1988). *Variation across Speech and Writing*. Cambridge University Press.

Biber, Douglas (1995). *Dimensions of Register Variation: A Cross-Linguistic Comparison.* Cambridge University Press.

Covington, M. A., & McFall, J. D. (2010). Cutting the Gordian Knot: The Moving-Average Type–Token Ratio (MATTR). *Journal of Quantitative Linguistics*, 17(2), 94–100. doi:10.1080/09296171003643098

See Also

dict, word_lists

Examples

```
# Parse the example documents provided with the package
biber(udpipe_samples)

biber(spacy_samples)
```

---

dict *Dictionaries defining text features*

Description

For Biber features defined by matching text against dictionaries of word patterns (such as third-person pronouns or conjunctions), or features that can be found by matching patterns against text, this gives the dictionary of patterns for each feature. These are primarily used internally by `biber()`, but are exported so users can examine the feature definitions.

Usage

```
dict
```

Format

A named list with one entry per feature. The name is the feature name, such as `f_33_pied_piping`; values give a list of terms or patterns. Patterns are matched to spaCy tokens using `quanteda::tokens_lookup()` using the `glob` valuetype.

udpipe_samples *Samples of parsed text*

Description

Examples of spaCy and udpipe tagging output from excerpts of several public-domain texts. Can be passed to `biber()` to see examples of its feature detection.

Usage

```
udpipe_samples

spacy_samples
```

Format

An object of class `udpipe_connlu` of length 3.

An object of class `spacyr_parsed` (inherits from `data.frame`) with 1346 rows and 9 columns.

Details

Texts consist of early paragraphs from several public-domain books distributed by Project Gutenberg https://gutenberg.org. Document IDs are the Project Gutenberg book numbers.

See `udpipe::udpipe_annotate()` and `spacyr::spacy_parse()` for further details on the data format produced by each package.

word_lists *Lists of words defining text features*

Description

For Biber features defined by matching texts against certain exact words, rather than patterns, this list defines the exact words defining the features. These lists are primarily used internally by `biber()`, but are exported so users can examine the feature definitions.

Usage

```
word_lists
```

Format

A named list with one entry per word list. Each entry is a vector of words.

Index