# Package 'idiolect'

August 28, 2024

**Type** Package

**Title** Forensic Authorship Analysis

**Version** 1.0.1

**Description** Carry out comparative authorship analysis of disputed and undisputed texts within the Likelihood Ratio Framework for expressing evidence in forensic science. This package contains implementations of well-known algorithms for comparative authorship analysis, such as Smith and Aldridge's (2011) Cosine Delta <doi:10.1080/09296174.2011.533591> or Koppel and Winter's (2014) Impostors Method <doi:10.1002/asi.22954>, as well as functions to measure their performance and to calibrate their outputs into Log-Likelihood Ratios.

**License** GPL (>= 2)

**URL** https://github.com/andreanini/idiolect, https://andreanini.github.io/idiolect/

**BugReports** https://github.com/andreanini/idiolect/issues

**Depends** quanteda, R (>= 3.5.0)

**Imports** caret, dplyr, fdrtool, ggplot2, kgrams, pbapply, pROC, proxy, quanteda.textstats, spacyr, stringr, textclean

**Suggests** knitr, readtext, rmarkdown, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Andrea Nini [aut, cre, cph] (<https://orcid.org/0000-0003-4218-5130>), David van Leeuwen [cph] (Author of some bundled functions from package ROC)

**Maintainer** Andrea Nini <andrea.nini@manchester.ac.uk>

**Repository** CRAN

**Date/Publication** 2024-08-28 09:00:02 UTC

# Contents



---

`calibrate_LLR` *Calibrate scores into Log-Likelihood Ratios*

---

## Description

This function is used to transform the scores returned by any of the authorship analysis functions into a Log-Likelihood Ratio (LLR).

## Usage

```
calibrate_LLR(calibration.dataset, dataset, latex = FALSE)
```

## Arguments

| | |
|---|---|
| `calibration.dataset` | A data frame containing the calibration dataset, typically the output of an authorship analysis function like `impostors()`. |
| `dataset` | A data frame containing the scores that have to be calibrated into LLRs using the calibration dataset. This is typically the result of applying a function like `impostors()` to the Q texts. |
| `latex` | A logical value. If FALSE (default), then the hypothesis labels are printed as plain text (Hp/Hd). If TRUE the labels are written to be read in LaTeX ($H_p$/$H_d$). |

## Value

The function returns a data frame with the LLRs (base 10), as well as the verbal label according to Marquis et al (2016) and a verbal interpretation of the results.

## References

Marquis, Raymond, Alex Biedermann, Liv Cadola, Christophe Champod, Line Gueissaz, Geneviève Massonnet, Williams David Mazzella, Franco Taroni & Tacha Hicks. 2016. Discussion on how to implement a verbal scale in a forensic laboratory: Benefits, pitfalls and suggestions to avoid misunderstandings. Science & Justice 56(5). 364–370. https://doi.org/10.1016/j.scijus.2016.05.009.

## Examples

```
calib <- data.frame(score = c(0.5, 0.2, 0.8, 0.01, 0.6), target = c(TRUE, FALSE, TRUE, FALSE, TRUE))
q <- data.frame(score = c(0.6, 0.002))
calibrate_LLR(calib, q)
```

---

# chunk_texts *Chunk a corpus*

---

## Description

This function can be used to chunk a corpus in order to control sample sizes.

## Usage

```
chunk_texts(corpus, size)
```

## Arguments

| | |
|---|---|
| corpus | A quanteda corpus. |
| size | The size of the chunks in number of tokens. |

## Value

A quanteda corpus object where each text is a chunk of the size requested.

## Examples

```
corpus <- quanteda::corpus(c("The cat sat on the mat", "The dog sat on the chair"))
quanteda::docvars(corpus, "author") <- c("A", "B")
chunk_texts(corpus, size = 2)
```

concordance *Qualitative examination of evidence*

## Description

This function uses quanteda::kwic() to return a concordance for a search pattern. The function takes as input three datasets and a pattern and returns a data frame with the hits labelled for authorship.

## Usage

```
concordance(
  q.data,
  k.data,
  reference.data,
  search,
  token.type = "word",
  window = 5,
  case_insensitive = TRUE
)
```

## Arguments

| | |
|---|---|
| q.data | A quanteda corpus object, such as the output of create_corpus(). |
| k.data | A quanteda corpus object, such as the output of create_corpus(). |
| reference.data | A quanteda corpus object, such as the output of create_corpus(). This is optional. |
| search | A string. It can be any sequence of characters and it also accepts the use of * as a wildcard. |
| token.type | Choice between "word" (default), which searches for word or punctuation mark tokens, or "character", which instead uses a single character search. |
| window | The number of context items to be displayed around the keyword (a quanteda::kwic() parameter). |
| case_insensitive | Logical; if TRUE, ignore case (a quanteda::kwic() parameter). |

## Value

The function returns a data frame containing the concordances for the search pattern.

## Examples

```
concordance(enron.sample[1], enron.sample[2], enron.sample[3:49], "wants to", token.type = "word")

#using wildcards
concordance(enron.sample[1], enron.sample[2], enron.sample[3:49], "want * to", token.type = "word")
```

```
#searching character sequences with wildcards
concordance(enron.sample[1], enron.sample[2], enron.sample[3:49], "help*", token.type = "character")
```

---

contentmask *Content masking*

---

## Description

This function offers three algorithms for topic/content masking. In order to run the masking algorithms, a spacy tokenizer or POS-tagger has to be run first (via spacyr). For more information about the masking algorithms see Details below.

## Usage

```
contentmask(
  corpus,
  model = "en_core_web_sm",
  algorithm = "POSnoise",
  fw_list = "eng_halvani",
  replace_non_ascii = TRUE
)
```

## Arguments

| | |
|---|---|
| corpus | A quanteda corpus object, typically the output of the create_corpus() function. |
| model | The spacy model to use. The default is "en_core_web_sm". |
| algorithm | A string, either "POSnoise" (default), "frames", or "textdistortion". |
| fw_list | The list of function words to use for the textdistortion algorithm. This is either the default ("eng_halvani") for the same list of function words used for POSnoise or it can be a vector of strings where each string is a function word to keep. |
| replace_non_ascii | A logical value indicating whether to remove non-ASCII characters (including emojis). This is the default. |

## Details

The default algorithm for content masking that this function applies is POSnoise (Halvani and Graner 2021). This algorithm only works for English and it transforms a text by masking tokens using their POS tag if these tokens are: nouns, verbs, adjectives, adverbs, digits, and symbols while leaving all the rest unchanged. POSnoise uses a list of function words for English that also includes frequent words belonging to the masked Part of Speech tags that tend to be mostly functional (e.g. make, recently, well).

Another algorithm implemented is Nini's (2023) `frames` or `frame n-grams`. This algorithm does not involve a special list of tokens and therefore can potentially work for any language provided that the correct `spacy` model is loaded. This algorithm consists in masking all tokens using their POS tag only when these are nouns, verbs, or personal pronouns.

Finally, the last algorithm implemented is a version of `textdistortion`, as originally proposed by Stamatatos (2017). This version of the algorithm is essentially `POSnoise` but without POS tag information. The default implementation uses the same list of function words that are used for `POSnoise`. In addition to the function words provided, the function treats all punctuation marks and new line breaks as function words to keep. The basic tokenization is done using `spacyr` so the right model for the language being analysed should be selected.

If you have never used `spacyr` before then please follow the instructions to set it up and install a model before using this function.

The removal of non-ASCII characters is done using the `textclean` package.

## Value

A `quanteda` corpus object only containing functional tokens, depending on the algorithm chosen. The corpus contains the same docvars as the input. Email addresses or URLs are treated like nouns.

## References

Halvani, Oren & Lukas Graner. 2021. POSNoise: An Effective Countermeasure Against Topic Biases in Authorship Analysis. In Proceedings of the 16th International Conference on Availability, Reliability and Security, 1–12. Vienna, Austria: Association for Computing Machinery. https://doi.org/10.1145/3465481.3470050. Nini, Andrea. 2023. A Theory of Linguistic Individuality for Authorship Analysis (Elements in Forensic Linguistics). Cambridge, UK: Cambridge University Press. Stamatatos, Efstathios. 2017. Masking topic-related information to enhance authorship attribution. Journal of the Association for Information Science and Technology. https://doi.org/10.1002/asi.23968.

## Examples

```
## Not run:
text <- "The cat was on the chair. He didn't move\ncat@pets.com;\nhttp://quanteda.io/. i.e. a test "
toy.corpus <- quanteda::corpus(text)
contentmask(toy.corpus, algorithm = "POSnoise")
contentmask(toy.corpus, algorithm = "textdistortion")

## End(Not run)
```

---

`create_corpus` *Create a corpus*

---

## Description

Function to read in text data and turn it into a `quanteda` corpus object.

## Usage

```
create_corpus(path)
```

## Arguments

| | |
|---|---|
| path | A string containing the path to a folder of plain text files (ending in .txt) with their name structured as following: authorname_textname.txt (e.g. smith_text1.txt). |

## Value

A quanteda corpus object with the authors' names as a docvar.

## Examples

```
## Not run:
path <- "path/to/data"
create_corpus(path)

## End(Not run)
```

---

# delta *Delta*

---

## Description

This function runs a *Cosine Delta* analysis (Smith and Aldridge 2011; Evert et al. 2017).

## Usage

```
delta(
  q.data,
  k.data,
  tokens = "word",
  remove_punct = FALSE,
  remove_symbols = TRUE,
  remove_numbers = TRUE,
  lowercase = TRUE,
  n = 1,
  trim = TRUE,
  threshold = 150,
  features = FALSE,
  cores = NULL
)
```

## Arguments

| | |
|---|---|
| q.data | The questioned or disputed data, either as a corpus (the output of create_corpus()) or as a quanteda dfm (the output of vectorize()). |
| k.data | The known or undisputed data, either as a corpus (the output of create_corpus()) or as a quanteda dfm (the output of vectorize()). |
| tokens | The type of tokens to extract, either "word" (default) or "character". |
| remove_punct | A logical value. FALSE (default) keeps punctuation marks. |
| remove_symbols | A logical value. TRUE (default) removes symbols. |
| remove_numbers | A logical value. TRUE (default) removes numbers |
| lowercase | A logical value. TRUE (default) transforms all tokens to lower case. |
| n | The order or size of the n-grams being extracted. Default is 1. |
| trim | A logical value. If TRUE (default) then only the most frequent tokens are kept. |
| threshold | A numeric value indicating how many most frequent tokens to keep if trim = TRUE. The default is 150. |
| features | Logical with default FALSE. If TRUE, then the output will contain the features used. |
| cores | The number of cores to use for parallel processing (the default is one). |

## Value

If features is set to FALSE then the output is a data frame containing the results of all comparisons between the Q texts and the K texts. If features is set to TRUE then the output is a list containing the results data frame and the vector of features used for the analysis.

## References

Evert, Stefan, Thomas Proisl, Fotis Jannidis, Isabella Reger, Steffen Pielström, Christof Schöch & Thorsten Vitt. 2017. Understanding and explaining Delta measures for authorship attribution. Digital Scholarship in the Humanities 32. ii4–ii16. https://doi.org/10.1093/llc/fqx023. Smith, Peter W H & W Aldridge. 2011. Improving Authorship Attribution: Optimizing Burrows' Delta Method*. Journal of Quantitative Linguistics 18(1). 63–88. https://doi.org/10.1080/09296174.2011.533591.

## Examples

```
Q <- enron.sample[c(5:6)]
K <- enron.sample[-c(5:6)]
delta(Q, K)
```

## density_plot *Plot density of TRUE/FALSE distributions*

### Description

Plot density of TRUE/FALSE distributions

### Usage

```
density_plot(dataset, q = NULL)
```

### Arguments

| | |
|---|---|
| dataset | A data frame containing the calibration dataset, typically the output of an authorship analysis function like `impostors()`. |
| q | This optional argument should be one value or a vector of values that contain the score of the disputed text(s). These are then plotted as lines crossing the density distributions. |

### Value

A `ggplot2` plot with the density distributions for the scores for TRUE (typically, 'same-author') vs. FALSE (typically, 'different-author').

### Examples

```
res <- data.frame(score = c(0.5, 0.2, 0.8, 0.01, 0.6), target = c(TRUE, FALSE, TRUE, FALSE, TRUE))
q <- c(0.11, 0.7)
density_plot(res, q)
```

## enron.sample *Enron sample*

### Description

A small sample of the *Enron* corpus comprising ten authors with approximately the same amount of data. Each author has one text labelled as 'unknown' and the other texts labelled as 'known'. The data was pre-processed using the *POSnoise* algorithm to mask content (see `contentmask()`).

### Usage

```
enron.sample
```

### Format

A `quanteda` corpus object.

## Source

Halvani, Oren. 2021. Practice-Oriented Authorship Verification. Technical University of Darmstadt PhD Thesis. https://tuprints.ulb.tu-darmstadt.de/19861/

---

`impostors` *Impostors Method*

---

## Description

This function runs the *Impostors Method* for authorship verification. The Impostors Method is based on calculating a similarity score and then, using a corpus of impostor texts, perform a bootstrapping analysis sampling random subsets of features and impostors in order to test the robustness of this similarity.

## Usage

```
impostors(
  q.data,
  k.data,
  cand.imps,
  algorithm = "RBI",
  coefficient = "minmax",
  k = 300,
  m = 100,
  n = 25,
  features = FALSE,
  cores = NULL
)
```

## Arguments

| | |
|---|---|
| `q.data` | The questioned or disputed data, either as a corpus (the output of `create_corpus()`) or as a `quanteda` dfm (the output of `vectorize()`). |
| `k.data` | The known or undisputed data, either as a corpus (the output of `create_corpus()`) or as a `quanteda` dfm (the output of `vectorize()`). More than one sample for a candidate author is accepted for all algorithms except IM. |
| `cand.imps` | The impostors data for the candidate authors, either as a corpus (the output of `create_corpus()`) or as a `quanteda` dfm (the output of `vectorize()`). This can be the same object as `k.data` (e.g. to recycle impostors). |
| `algorithm` | A string specifying which impostors algorithm to use, either "RBI" (deafult), "KGI", or "IM". |
| `coefficient` | A string indicating the coefficient to use, either "minmax" (default) or "cosine". This does not apply to the algorithm KGI, where the distance is "minmax". |
| `k` | The *k* parameters for the RBI algorithm. Not used by other algorithms. The default is 300. |

| | |
|---|---|
| m | The *m* parameter for the IM algorithm. Not used by other algorithms. The default is 100. |
| n | The *n* parameter for the IM algorithm. Not used by other algorithms. The default is 25. |
| features | A logical value indicating whether the important features should be retrieved or not. The default is FALSE. This only applies to the RBI algorithm. |
| cores | The number of cores to use for parallel processing (the default is one). |

## Details

There are several variants of the Impostors Method and this function can run three of them:

1. IM: this is the original Impostors Method as proposed by Koppel and Winter (2014).
2. KGI: Kestemont's et al. (2016) version, which is a very popular implementation of the Impostors Method in stylometry. It is inspired by IM and by its generalized version, the General Impostors Method proposed by Seidman (2013).
3. RBI: the Rank-Based Impostors Method (Potha and Stamatatos 2017, 2020), which is the default option as it is the most recent and as it tends to outperform the original. The two data sets `q.data`, `k.data`, must be disjunct in terms of the texts that they contain otherwise an error is returned. However, `cand.imps` and `k.data` can be the same object, for example, to use the other candidates' texts as impostors. The function will always exclude impostor texts with the same author as the Q and K texts considered.

## Value

The function will test all possible combinations of Q texts and candidate authors and return a data frame containing a score ranging from 0 to 1, with a higher score indicating a higher likelihood that the same author produced the two sets of texts. The data frame contains a column called "target" with a logical value which is TRUE if the author of the Q text is the candidate and FALSE otherwise.

If the RBI algorithm is selected and the features parameter is TRUE then the data frame will also contain a column with the features that are likely to have had an impact on the score. These are all those features that are consistently found to be shared by the candidate author's data and the questioned data and that also tend to be rare in the dataset of impostors.

## References

Kestemont, Mike, Justin Stover, Moshe Koppel, Folgert Karsdorp & Walter Daelemans. 2016. Authenticating the writings of Julius Caesar. Expert Systems With Applications 63. 86–96. https://doi.org/10.1016/j.eswa.2016.0

Koppel, Moshe & Yaron Winter. 2014. Determining if two documents are written by the same author. Journal of the Association for Information Science and Technology 65(1). 178–187.

Potha, Nektaria & Efstathios Stamatatos. 2017. An Improved Impostors Method for Authorship Verification. In Gareth J.FALSE. Jones, Séamus Lawless, Julio Gonzalo, Liadh Kelly, Lorraine Goeuriot, Thomas Mandl, Linda Cappellato & Nicola Ferro (eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction (Lecture Notes in Computer Science), vol. 10456, 138–144. Springer, Cham. https://doi.org/10.1007/978-3-319-65813-1_14. (5 September, 2017).

Potha, Nektaria & Efstathios Stamatatos. 2020. Improved algorithms for extrinsic author verification. Knowledge and Information Systems 62(5). 1903–1921. https://doi.org/10.1007/s10115-019-01408-4.

Seidman, Shachar. 2013. Authorship Verification Using the Impostors Method. In Pamela Forner, Roberto Navigli, Dan Tufis & Nicola Ferro (eds.), Proceedings of CLEF 2013 Evaluation Labs and Workshop – Working Notes Papers, 23–26. Valencia, Spain. https://ceur-ws.org/Vol-1179/.

## Examples

```
Q <- enron.sample[1]
K <- enron.sample[2:3]
imps <- enron.sample[4:9]
impostors(Q, K, imps, algorithm = "KGI")
```

---

`lambdaG` *Apply the LambdaG algorithm*

---

## Description

This function calculates the likelihood ratio of grammar models, or $\lambda_G$, as in Nini et al. (under review). In order to run the analysis as in this paper, all data must be preprocessed using `contentmask()` with the "algorithm" parameter set to "POSnoise".

## Usage

```
lambdaG(q.data, k.data, ref.data, N = 10, r = 30, cores = NULL)
```

## Arguments

| | |
|---|---|
| `q.data` | The questioned or disputed data as a `quanteda` tokens object with the tokens being sentences (e.g. the output of `tokenize_sents()`). |
| `k.data` | The known or undisputed data as a `quanteda` tokens object with the tokens being sentences (e.g. the output of `tokenize_sents()`). |
| `ref.data` | The reference dataset as a `quanteda` tokens object with the tokens being sentences (e.g. the output of `tokenize_sents()`). This can be the same object as `k.data`. |
| `N` | The order of the model. Default is 10. |
| `r` | The number of iterations. Default is 30. |
| `cores` | The number of cores to use for parallel processing (the default is one). |

## Value

The function will test all possible combinations of Q texts and candidate authors and return a data frame containing $\lambda_G$, an uncalibrated log-likelihood ratio (base 10). $\lambda_G$ can then be calibrated into a likelihood ratio that expresses the strength of the evidence using `calibrate_LLR()`. The data frame contains a column called "target" with a logical value which is TRUE if the author of the Q text is the candidate and FALSE otherwise.

## References

Nini, A., Halvani, O., Graner, L., Gherardi, V., Ishihara, S. Authorship Verification based on the Likelihood Ratio of Grammar Models. https://arxiv.org/abs/2403.08462v1

## Examples

```
q.data <- enron.sample[1] |> quanteda::tokens("sentence")
k.data <- enron.sample[2:10] |> quanteda::tokens("sentence")
ref.data <- enron.sample[11:ndoc(enron.sample)] |> quanteda::tokens("sentence")
lambdaG(q.data, k.data, ref.data)
```

---

# lambdaG_visualize *Visualize the output of the LambdaG algorithm*

---

## Description

This function outputs a colour-coded list of sentences belonging to the input Q text ordered from highest to lowest $\lambda_G$, as shown in Nini et al. (under review).

## Usage

```
lambdaG_visualize(
  q.data,
  k.data,
  ref.data,
  N = 10,
  r = 30,
  output = "html",
  print = "",
  scale = "absolute",
  cores = NULL
)
```

## Arguments

| | |
|---|---|
| q.data | A single questioned or disputed text as a quanteda tokens object with the tokens being sentences (e.g. the output of tokenize_sents()). |
| k.data | A known or undisputed corpus containing exclusively a single candidate author's texts as a quanteda tokens object with the tokens being sentences (e.g. the output of tokenize_sents()). |
| ref.data | The reference dataset as a quanteda tokens object with the tokens being sentences (e.g. the output of tokenize_sents()). |
| N | The order of the model. Default is 10. |
| r | The number of iterations. Default is 30. |

| | |
|---|---|
| output | A string detailing the file type of the colour-coded text output. Either "html" (default) or "latex". |
| print | A string indicating the path to the folder where to print a colour-coded text file. If left empty (default), then nothing is printed. |
| scale | A string indicating what scale to use to colour-code the text file. If "absolute" (default) then the raw $\lambda_G$ is used; if "relative", then the z-score of $\lambda_G$ over the Q data is used instead, thus showing relative importance. |
| cores | The number of cores to use for parallel processing (the default is one). |

## Value

The function outputs a list of two objects: a data frame with each row being a token in the Q text and the values of $\lambda_G$ for the token and sentences, in decreasing order of sentence $\lambda_G$ and with the relative contribution of each token and each sentence to the final $\lambda_G$ in percentage; the raw code in html or LaTeX that generates the colour-coded file. If a path is provided for the print argument then the function will also save the colour-coded text as an html or plain text file.

## References

Nini, A., Halvani, O., Graner, L., Gherardi, V., Ishihara, S. Authorship Verification based on the Likelihood Ratio of Grammar Models. https://arxiv.org/abs/2403.08462v1

## Examples

```
q.data <- corpus_trim(enron.sample[1], "sentences", max_ntoken = 10) |> quanteda::tokens("sentence")
k.data <- enron.sample[2:5]|> quanteda::tokens("sentence")
ref.data <- enron.sample[6:ndoc(enron.sample)] |> quanteda::tokens("sentence")
outputs <- lambdaG_visualize(q.data, k.data, ref.data, r = 2)
outputs$table
```

---

# most_similar *Select the most similar texts to a specific text*

---

## Description

Select the most similar texts to a specific text

## Usage

```
most_similar(sample, pool, coefficient, n)
```

## Arguments

| | |
|---|---|
| sample | This is a single row of a quanteda dfm representing the sample to match. |
| pool | This is a dfm containing all possible samples from which to select the top n. |
| coefficient | The coefficient to use for similarity. Either "minmax", "cosine", or "Phi". |
| n | The number of rows to extract from the pool of potential samples. |

## Value

The function returns a dfm containing the top n most similar rows to the input sample using the minmax distance.

## Examples

```
text1 <- "The cat sat on the mat"
text2 <- "The dog sat on the chair"
text3 <- "Violence is the last refuge of the incompetent"
c <- quanteda::corpus(c(text1, text2, text3))
d <- quanteda::tokens(c) |> quanteda::dfm() |> quanteda::dfm_weight(scheme = "prop")
most_similar(d[1,], d[-1,], coefficient = "minmax", n = 1)
```

---

# ngram_tracing *N-gram tracing*

---

## Description

This function runs the authorship analysis method called *n-gram tracing*, which can be used for both attribution and verification.

## Usage

```
ngram_tracing(
  q.data,
  k.data,
  tokens = "character",
  remove_punct = FALSE,
  remove_symbols = TRUE,
  remove_numbers = TRUE,
  lowercase = TRUE,
  n = 9,
  coefficient = "simpson",
  features = FALSE,
  cores = NULL
)
```

## Arguments

| | |
|---|---|
| q.data | The questioned or disputed data, either as a corpus (the output of create_corpus()) or as a quanteda dfm (the output of vectorize()). |
| k.data | The known or undisputed data, either as a corpus (the output of create_corpus()) or as a quanteda dfm (the output of vectorize()). More than one sample for a candidate author is accepted but the function will combine them so to make a profile. |
| tokens | The type of tokens to extract, either "word" or "character" (default). |

| | |
|---|---|
| remove_punct | A logical value. FALSE (default) keeps punctuation marks. |
| remove_symbols | A logical value. TRUE (default) removes symbols. |
| remove_numbers | A logical value. TRUE (default) removes numbers. |
| lowercase | A logical value. TRUE (default) transforms all tokens to lower case. |
| n | The order or size of the n-grams being extracted. Default is 9. |
| coefficient | The coefficient to use to compare texts, one of: "simpson" (default), "phi", "jaccard", "kulczynski", or "cole". |
| features | Logical with default FALSE. If TRUE then the result table will contain the features in the overlap that are unique for that overlap in the corpus. If only two texts are present then this will return the n-grams in common. |
| cores | The number of cores to use for parallel processing (the default is one). |

## Details

N-gram tracing was originally proposed by Grieve et al (2019). Nini (2023) then proposed a mathematical reinterpretation that is compatible with Cognitive Linguistic theories of language processing. He then tested several variants of the method and found that the original version, which uses the Simpson's coefficient, tends to be outperformed by versions using the Phi coefficient, the Kulczynski's coefficient, and the Cole coefficient. This function can run the n-gram tracing method using any of these coefficients plus the Jaccard coefficient for reference, as this coefficient has been applied in several forensic linguistic studies.

## Value

The function will test all possible combinations of Q texts and candidate authors and return a data frame containing the value of the similarity coefficient selected called 'score' and an optional column with the overlapping features that only occur in the Q and candidate considered and in no other Qs (ordered by length if the n-gram is of variable length). The data frame contains a column called 'target' with a logical value which is TRUE if the author of the Q text is the candidate and FALSE otherwise.

## References

Grieve, Jack, Emily Chiang, Isobelle Clarke, Hannah Gideon, Aninna Heini, Andrea Nini & Emily Waibel. 2019. Attributing the Bixby Letter using n-gram tracing. Digital Scholarship in the Humanities 34(3). 493–512. Nini, Andrea. 2023. A Theory of Linguistic Individuality for Authorship Analysis (Elements in Forensic Linguistics). Cambridge, UK: Cambridge University Press.

## Examples

```
Q <- enron.sample[c(5:6)]
K <- enron.sample[-c(5:6)]
ngram_tracing(Q, K, coefficient = 'phi')
```

`performance` *Performance evaluation*

## Description

This function is used to the test the performance of an authorship analysis method.

## Usage

```
performance(training, test = NULL)
```

## Arguments

| | |
|---|---|
| `training` | The data frame with the results to evaluate, typically the output of an authorship analysis function, such as `impostors()`. If only training is present then the function will perform a leave-one-out cross-validation. |
| `test` | Optional data frame of results. If present then a calibration model is extracted from training and its performance is evaluated on this data set. |

## Details

Before applying a method to a real authorship case, it is good practice to test it on known ground truth data. This function performs this test by taking as input either a single table of results or two tables, one for training and one for the test, and then returning as output a list with the following performance statistics: the log-likelihood ratio cost (both $C_{llr}$ and $C_{llr}^{min}$), Equal Error Rate (ERR), the mean values of the log-likelihood ratio for both the same-author (TRUE) and different-author (FALSE) cases, the Area Under the Curve (AUC), Balanced Accuracy, Precision, Recall, F1, and the full confusion matrix. The binary classification statistics are all calculated considering a Log-Likelihood Ratio score of 0 as a threshold.

## Value

The function returns a list containing a data frame with performance statistics, including an object that can be used to make a tippet plot using the `tippet.plot()` function of the `ROC` package (https://github.com/davidavdav/ROC).

## Examples

```
results <- data.frame(score = c(0.5, 0.2, 0.8, 0.01), target = c(TRUE, FALSE, TRUE, FALSE))
perf <- performance(results)
perf$evaluation
```

## posterior *Posterior prosecution probabilities and odds*

### Description

This function takes as input a value of the Log-Likelihood Ratio and returns a table that shows the impact on some simulated prior probabilities for the prosecution hypothesis.

### Usage

```
posterior(LLR)
```

### Arguments

| | |
|---|---|
| LLR | One single numeric value corresponding to a Log-Likelihood Ratio (base 10). |

### Value

A data frame containing some simulated prior probabilities/odds for the prosecution and the resulting posterior probabilities/odds after the LLR.

### Examples

```
posterior(LLR = 0)
posterior(LLR = 1.8)
posterior(LLR = -0.5)
posterior(LLR = 4)
```

## tokenize_sents *Tokenize to sentences*

### Description

This function turns a corpus of texts into a quanteda tokens object of sentences.

### Usage

```
tokenize_sents(corpus, model = "en_core_web_sm")
```

### Arguments

| | |
|---|---|
| corpus | A quanteda corpus object, typically the output of the create_corpus() function or the output of contentmask(). |
| model | The spacy model to use. The default is "en_core_web_sm". |

## Details

The function first split each text into paragraphs by splitting at new line markers and then uses spacy to tokenize each paragraph into sentences. The function accepts a plain text corpus input or the output of `contentmask()`. This function is necessary to prepare the data for `lambdaG()`.

## Value

A quanteda tokens object where each token is a sentence.

## Examples

```
## Not run:
toy.pos <- corpus("the N was on the N . he did n't move \n N ; \n N N")
tokenize_sents(toy.pos)

## End(Not run)
```

---

`vectorize` *Vectorize data*

---

## Description

This function turns texts into feature vectors.

## Usage

```
vectorize(
  input,
  tokens,
  remove_punct,
  remove_symbols,
  remove_numbers,
  lowercase,
  n,
  weighting,
  trim,
  threshold
)
```

## Arguments

| | |
|---|---|
| `input` | This should be a quanteda corpus object with the author names as a docvar called "author". Typically, this is the output of the `create_corpus()` function. |
| `tokens` | The type of tokens to extract, either "character" or "word". |
| `remove_punct` | A logical value. FALSE to keep the punctuation marks or TRUE to remove them. |

| | |
|---|---|
| remove_symbols | A logical value. TRUE removes symbols and FALSE keeps them. |
| remove_numbers | A logical value. TRUE removes numbers and FALSE keeps them. |
| lowercase | A logical value. TRUE transforms all tokens to lower case. |
| n | The order or size of the n-grams being extracted. |
| weighting | The type of weighting to use, "rel" for relative frequencies, "tf-idf", or "boolean". |
| trim | A logical value. If TRUE then only the most frequent tokens are kept. |
| threshold | A numeric value indicating how many most frequent tokens to keep. |

## Details

All the authorship analysis functions call `vectorize()` with the standard parameters for the algorithm selected. This function is therefore left only for those users who want to modify these parameters or for convenience if the same dfm has to be reused by the algorithms so to avoid vectorizing the same data many times. Most users who only need to run a standard analysis do not need use this function.

## Value

A dfm (document-feature matrix) containing each text as a feature vector. N-gram tokenisation does not cross sentence boundaries.

## Examples

```
mycorpus <- quanteda::corpus("The cat sat on the mat.")
quanteda::docvars(mycorpus, "author") <- "author1"
matrix <- vectorize(mycorpus, tokens = "character", remove_punct = FALSE, remove_symbols = TRUE,
remove_numbers = TRUE, lowercase = TRUE, n = 5, weighting = "rel", trim = TRUE, threshold = 1500)
```

# Index