Package 'ndl'

July 23, 2025

Type Package

Title Naive Discriminative Learning

Version 0.2.18

Date 2018-09-09

Maintainer Tino Sering <konstantin.sering@uni-tuebingen.de>

Description Naive discriminative learning implements learning and classification models based on the Rescorla-Wagner equations and their equilibrium equations.

License GPL-3

Depends R (>= 3.0.2)

Imports Rcpp (>= 0.11.0), MASS, Hmisc

LinkingTo Rcpp

NeedsCompilation yes

RoxygenNote 6.1.0

Author Antti Arppe [aut],
Peter Hendrix [aut],
Petar Milin [aut],
R. Harald Baayen [aut],
Tino Sering [aut, cre],
Cyrus Shaoul [aut]

Repository CRAN

Date/Publication 2018-09-10 13:40:02 UTC

Contents

`ndl-package` *Naive Discriminative Learning*

Description

Naive discriminative learning implements learning and classification models based on the Rescorla-Wagner equations and their equilibrium equations.

Naive discriminative learning implements classification models based on the Rescorla-Wagner equations and the equilibrium equations of the Rescorla-Wagner equations. This package provides three kinds of functionality: (1) discriminative learning based directly on the Rescorla-Wagner equations, (2) a function implementing the naive discriminative reader, and a model for silent (single-word) reading, and (3) a classifier based on the equilibrium equations. The functions and datasets for the naive discriminative reader model make it possible to replicate the simulation results for Experiment 1 of Baayen et al. (2011). The classifier is provided to allow for comparisons between machine learning (svm, TiMBL, glm, random forests, etc.) and discrimination learning. Compared to standard classification algorithms, naive discriminative learning may overfit the data, albeit gracefully.

Details

The DESCRIPTION file:

Package:	ndl
Type:	Package
Title:	Naive Discriminative Learning
Version:	0.2.18
Date:	2018-09-09
Authors@R:	c(person("Antti Arppe", role = "aut", email = "arppe@ualberta.ca"), person("Peter Hendrix", role = "aut
Maintainer:	Tino Sering <konstantin.sering@uni-tuebingen.de>
Description:	Naive discriminative learning implements learning and classification models based on the Rescorla-Wag
License:	GPL-3
Depends:	R (>= 3.0.2)
Imports:	Rcpp (>= 0.11.0), MASS, Hmisc
LinkingTo:	Rcpp
NeedsCompilation:	yes
Packaged:	2015-11-10 10:28:58 UTC; kfs-studium
RoxygenNote:	6.1.0
Author:	Antti Arppe [aut], Peter Hendrix [aut], Petar Milin [aut], R. Harald Baayen [aut], Tino Sering [aut, cre],

Index of help topics:

lexample	Lexical example data illustrating the Rescorla-Wagner equations
modelStatistics	Calculate a range of goodness of fit measures for an object fitted with some multivariate statistical method that yields probability estimates for outcomes.
ndl-package	Naive Discriminative Learning
ndlClassify	Classification using naive discriminative learning.
ndlCrossvalidate	Crossvalidation of a Naive Discriminative Learning model.
ndlCuesOutcomes	Creation of dataframe for Naive Discriminative Learning from formula specification
ndlStatistics	Calculate goodness of fit statistics for a naive discriminative learning model.
ndlVarimp	Permutation variable importance for classification using naive discriminative learning.
numbers	Example data illustrating the Rescorla-Wagner equations as applied to numerical cognition by Ramscar et al. (2011).
orthoCoding	Code a character string (written word form) as letter n-grams
plot.RescorlaWagner	Plot function for the output of 'RescorlaWagner'.
plot.ndlClassify	Plot function for selected results of 'ndlClassify'.
plurals	Artificial data set used to illustrate the Rescorla-Wagner equations and naive discriminative learning.
predict.ndlClassify	Predict method for ndlClassify objects
random.pseudoinverse	Calculate an approximation of the pseudoinverse of a matrix.
serbian	Serbian case inflected nouns.
serbianLex	Serbian lexicon with 1187 prime-target pairs.
serbianUniCyr	Serbian case inflected nouns (in Cyrillic Unicode).
serbianUniLat	Serbian case inflected nouns (in Latin-alphabet Unicode).
summary.ndlClassify	A summary of a Naive Discriminatory Learning Model
summary.ndlCrossvalidate	A summary of a crossvalidation of a Naive Discriminatory Reader Model
think	Finnish 'think' verbs.

For more detailed information on the core Rescorla-Wagner equations, see the functions `RescorlaWagner` and `plot.RescorlaWagner`, as well as the data sets `danks`, `numbers` (data courtesy of Michael Ramscar), and `lexample` (an example discussed in Baayen et al. 2011).

The functions for the naive discriminative learning (at the user level) are estimateWeights and estimateActivations. The relevant data sets are serbian, serbianUniCyr,serbianUniLat, and serbianLex. The examples for serbianLex present the full simulation for Experiment 1 of Baayen et al. (2011).

Key functionality for the user is provided by the functions orthoCoding, estimateWeights, and estimateActivations. orthoCoding calculates the letter n-grams for character strings, to be used as cues. It is assumed that meaning or meanings (separated by underscores if there are more then one) are available as outcomes. The frequency with which each (unique) combination of cues and outcomes occurs are required. For some example input data sets, see: danks, plurals, serbian, serbianUniCyr and serbianUniLat.

The function estimateWeights estimates the association strengths of cues to outcomes, using the equilibrium equations presented in Danks (2003). The function estimateActivations estimates the activations of outcomes (meanings) given cues (n-grams).

The Rcpp-based learn and learnLegacy functions use a C++ function to compute the conditional co-occurrence matrices required in the equilibrium equations. These are internally used by estimateWeights and should not be used directly by users of the package.

The key function for naive discriminative classification is ndlClassify; see data sets think and dative for examples.

Author(s)

NA

Maintainer: Tino Sering <konstantin.sering@uni-tuebingen.de>

Author Contributions: Initial concept by R. Harald Baayen with contributions from Petar Milin and Peter Hendrix. First R coding done by R. Harald Baayen.

Initial R package development until version 0.1.6 by Antti Arppe. Initial documentation by Antti Arppe. Initial optimizations in C by Petar Milin and Antti Arppe.

Classification functionality developed further by Antti Arppe.

In version 0.2.14 to version 0.2.16, improvements to the NDL algorithm by Petar Milin and Cyrus Shaoul. In version 0.2.14 to version 0.2.16, improved performance optimizations (C++ and Rcpp) by Cyrus Shaoul.

From version 0.2.17 onwards bug fixes and cran compliance by Tino Sering.

References

Baayen, R. H. and Milin, P. and Filipovic Durdevic, D. and Hendrix, P. and Marelli, M., An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

Baayen, R. H. (2011) Corpus linguistics and naive discriminative learning. Brazilian Journal of Applied Linguistics, 11, 295-328.

Arppe, A. and Baayen, R. H. (in prep.) Statistical classification and principles of human learning.

Examples

```
## Not run:
# Rescorla-Wagner
data(lexample)

lexample$Cues <- orthoCoding(lexample$Word, grams=1)
lexample.rw <- RescorlaWagner(lexample, nruns=25, traceCue="h",
   traceOutcome="hand")
plot(lexample.rw)
mtext("h - hand", 3, 1)

data(numbers)

traceCues <- c( "exactly1", "exactly2", "exactly3", "exactly4", "exactly5",
   "exactly6", "exactly7", "exactly10", "exactly15")
traceOutcomes <- c("1", "2", "3", "4", "5", "6", "7", "10", "15")

ylimit <- c(0,1)
par(mfrow=c(3,3), mar=c(4,4,1,1))

for (i in 1:length(traceCues)) {
  numbers.rw <- RescorlaWagner(numbers, nruns=1, traceCue=traceCues[i],
     traceOutcome=traceOutcomes[i])
  plot(numbers.rw, ylimit=ylimit)
  mtext(paste(traceCues[i], " - ", traceOutcomes[i], sep=""), side=3, line=-1,
    cex=0.7)
}
par(mfrow=c(1,1))

# naive discriminative learning (for complete example, see serbianLex)
# This function uses a Unicode dataset.
data(serbianUniCyr)
serbianUniCyr$Cues <- orthoCoding(serbianUniCyr$WordForm, grams=2)
serbianUniCyr$Outcomes <- serbianUniCyr$LemmaCase
sw <- estimateWeights(cuesOutcomes=serbianUniCyr,hasUnicode=T)

desiredItems <- unique(serbianUniCyr["Cues"])
desiredItems$Outcomes=""
activations <- estimateActivations(desiredItems, sw)$activationMatrix
rownames(activations) <- unique(serbianUniCyr[["WordForm"]])

syntax <- c("acc", "dat", "gen", "ins", "loc", "nom", "Pl",  "Sg")
activations2 <- activations[,!is.element(colnames(activations), syntax)]
head(rownames(activations2),50)
head(colnames(activations2),8)

image(activations2, xlab="word forms", ylab="meanings", xaxt="n", yaxt="n")
mtext(c("yena", "...", "zvuke"), side=1, line=1, at=c(0, 0.5, 1),  adj=c(0,0,1))
mtext(c("yena", "...", "zvuk"), side=2, line=1, at=c(0, 0.5, 1),   adj=c(0,0,1))

# naive discriminative classification
data(think)
```

```
think.ndl <- ndlClassify(Lexeme ~ Person + Number + Agent + Patient + Register,
   data=think)
summary(think.ndl)
plot(think.ndl, values="weights", type="hist", panes="multiple")
plot(think.ndl, values="probabilities", type="density")

## End(Not run)
```

`acts2probs` *Calculate probability matrix from activation matrix, as well as predicted values*

Description

`acts2probs` takes the activation matrix returned by `ndlClassify` and calculates the matrix of probabilities for the estimated activation matrix, as well as the predicted values of the response variable.

Usage

```
acts2probs(acts)
```

Arguments

`acts`	A matrix of activations (number of observations by number of levels of the response variable).

Details

Probabilities in `p` are obtained by adding the absolute value of the minimum activation to the activation matrix, and renorming. The selection rule used to produce `predicted` is to choose for each instance in the data the outcome value that has received the highest probability estimate.

Value

A list with the following components:

`p` a matrix with the probabilities of the levels of the response variable for each observation.

`predicted` a character vector with predicted values.

Author(s)

Harald Baayen and Antti Arppe

References

Arppe, A. and Baayen, R. H. (in prep.). Statistical classification and principles of human learning.

See Also

See also `ndlClassify`.

Examples

```
data(think)
think.ndl <- ndlClassify(Lexeme ~ Person + Number + Agent + Register, data=think)
pdata <- acts2probs(think.ndl$activationMatrix)
```

`anova.ndlClassify` *Analysis of Model Fit for Naive Discriminatory Reader Models*

Description

Compute an analysis of individual variable contributions or model comparisons for one or more Naive Discriminatory Reader model fits.

Usage

```
 ## S3 method for class 'ndlClassify'
anova(object, ..., statistic = "deviance", test = "Chisq")
```

Arguments

`object, ...`	Object(s) of class `"ndlClassify"`, typically the result of a call to `ndlClassify`, or a list of `objects` for the `ndlClassifylist` method.
`statistic`	A character string specifying the statistic describing the fit that is to be compared, by default `deviance`, which is obtained from the object(s).
`test`	A character string, determining the statistical method by which the significance of the comparison are done, by default the Chi-squared test (`Chisq`).

Details

Currently, comparison of the terms of a single model or multiple models is only implemented based on the `deviance` statistic.

Specifying a single object gives a sequential analysis of deviance table for that fit. That is, the reductions in the residual deviance as each term of the formula is added in turn are given in as the rows of a table, plus the residual deviances themselves.

If more than one object is specified, the table has a row for the residual degrees of freedom and deviance for each model. For all but the first model, the change in degrees of freedom and deviance is also given. (This only makes statistical sense if the models are nested.) It is conventional to list the models from smallest to largest, but this is up to the user.

The table will contain test statistics (and P values) comparing the reduction in deviance for the row to the residuals. Only a comparison of models or contributions of their components by the chi-squared test has been implemented.

The comparison between two or more models by `anova` or `anova.ndlClassifylist` will only be valid if they are fitted to the same dataset. If `anova.ndlClassifylist` detects this, it will stop and report an error.

Value

An object of class "anova" inheriting from class "data.frame".

Author(s)

Antti Arppe

References

Arppe, A. and Baayen, R. H. (in prep.) Statistical classification and principles of human learning.

See Also

ndlClassify

Examples

```
data(think)
set.seed(314)
think <- think[sample(1:nrow(think),500),]

think.ndl1 <- ndlClassify(Lexeme ~ Agent * Person, data=think)
anova(think.ndl1)

think.ndl2 <- ndlClassify(Lexeme ~ Agent * Person + Patient, data=think)
anova(think.ndl1, think.ndl2)
```

---

crosstableStatistics *Calculate statistics for a contingency table*

Description

crosstableStatistics takes a contingency table of observed vs. predicted values for a binary or polytomous response variable as input, and calculates a range of statistics about prediction accuracy.

Usage

```
crosstableStatistics(ctable)
```

Arguments

ctable	A contingency table cross-classifying observed and predicted values.

Value

A list with the following components:

`accuracy` Overall prediction accuracy

`recall.predicted` Recall of prediction for each outcome value

`precision.predicted` Precision of prediction for each outcome value

`lambda.prediction` lambda for prediction accuracy (improvement over baseline of always predicting mode)

`tau.classification` tau for classification accuracy (improvement over baseline of homogeneous distribution of predicted outcomes)

`d.lambda.prediction` d(lambda): used for calculating `P(lambda)`

`d.tau.classification` d(tau): used for calculating `P(tau)`

`p.lambda.prediction` P(lambda): probability of reaching `lambda` by chance

`p.tau.classification` P(tau): probability of reaching `tau` by chance

Author(s)

Antti Arppe and Harald Baayen

References

Arppe, A. 2008. Univariate, bivariate and multivariate methods in corpus-based lexicography – a study of synonymy. Publications of the Department of General Linguistics, University of Helsinki, No. 44. URN: http://urn.fi/URN:ISBN:978-952-10-5175-3.

Arppe, A. and Baayen, R. H. (in prep.). Statistical classification and principles of human learning.

Menard, Scott (1995). Applied Logistic Regression Analysis. Sage University Paper Series on Quantitative Applications in the Social Sciences 07-106. Thousand Oaks: Sage Publications.

See Also

See also `modelStatistics`, `ndlStatistics`, `ndlClassify`.

Examples

```
ctable <- matrix(c(30, 10, 5, 60), 2, 2)
crosstableStatistics(ctable)
```

cueCoding *code a vector of cues as n-grams*

Description

cueCoding codes a vector of cues into unigrams, bigrams, . . . , n-grams, with unigrams as default.

Usage

```
cueCoding(cues = c("hello", "world"), maxn=1, adjacent=FALSE)
```

Arguments

cues	A vector of cues (represented by strings) to be recoded as unigrams, bigrams, . . . , ngrams.
maxn	The longest n-gram to be encoded, by default maxn=1.
adjacent	A logical indicating whether only adjacent bigrams should be included when maxn=2. If adjacent=TRUE and maxn!=2, maxn is forced to 2.

Value

A vector of cue n-grams, one for each word in the input vector cues. Each n-gram vector lists the constituent unigrams, bigrams, etc., separated by underscores.

Author(s)

Antti Arppe and Harald Baayen

References

Arppe, A. and Baayen, R. H. (in prep.). Statistical classification and principles of human learning.

See Also

See also ndlClassify, ndlCuesOutcomes, ndlVarimp, ndlCrossvalidate.

Examples

```
# Cues from the \code{think} data: Person, Number, Register
cues <- c("First", "Plural", "hs95")
cueCoding(cues, maxn=1)
cueCoding(cues, maxn=2)
```

danks *Example data from Danks (2003), after Spellman (1996).*

Description

Data of Spellman (1996) used by Danks (2003) to illustrate the equilibrium equations for the Rescorla-Wagner model. There are two liquids (red and blue) that are potentially fertilizers, and the experimental participant is given the rates at which flowers bloom for the four possible conditions (no liquid, red liquid, blue liquid, and both liquids).

Usage

```
data(danks)
```

Format

A data frame with 8 observations on the following 3 variables.

`Cues` A character vector specifying the cues. The pots in which the flowers are grown, and the color of the fertilizer. Individual cues are separated by underscores.

`Outcomes` A character vector specifying whether plants flowered (y or n).

`Frequency` A numeric vector specifying the frequency of flowering.

Details

For details, see Danks (2003: 112).

Source

B. A. Spellman, (1996). Conditionalizing causality. In Shanks, D. R., Holyoak, K. J., & Medin, D. L. (Eds.), Causal learning: the psychology of learning and motivation, Vol. 34 (pp. 167-206). San Diego, CA: Academic Press.

References

D. Danks (2003), Equilibria of the Rescorla-Wagner model. Journal of Mathematical Psychology 47, 109-121.

B. A. Spellman, (1996). Conditionalizing causality. In Shanks, D. R., Holyoak, K. J., & Medin, D. L. (Eds.), Causal learning: the psychology of learning and motivation, Vol. 34 (pp. 167-206). San Diego, CA: Academic Press.

Examples

```
data(danks)
estimateWeights(cuesOutcomes=danks)
```

dative *Dative Alternation*

Description

Data describing the realization of the dative as NP or PP in the Switchboard corpus and the Treebank Wall Street Journal collection.

Usage

```
data(dative)
```

Format

A data frame with 3263 observations on the following 15 variables.

`Speaker` a factor coding speaker; available only for the subset of spoken English.

`Modality` a factor with levels `spoken`, `written`.

`Verb` a factor with the verbs as levels.

`SemanticClass` a factor with levels `a` (abstract: 'give it some thought'), `c` (communication: 'tell, give me your name'), `f` (future transfer of possession: 'owe, promise'), `p` (prevention of possession: 'cost, deny'), and `t` (transfer of possession: 'give an armband, send').

`LengthOfRecipient` a numeric vector coding the number of words comprising the recipient.

`AnimacyOfRec` a factor with levels `animate` and `inanimate` for the animacy of the recipient.

`DefinOfRec` a factor with levels `definite` and `indefinite` coding the definiteness of the recipient.

`PronomOfRec` a factor with levels `nonpronominal` and `pronominal` coding the pronominality of the recipient.

`LengthOfTheme` a numeric vector coding the number of words comprising the theme.

`AnimacyOfTheme` a factor with levels `animate` and `inanimate` coding the animacy of the theme.

`DefinOfTheme` a factor with levels `definite` and `indefinite` coding the definiteness of the theme.

`PronomOfTheme` a factor with levels `nonpronominal` and `pronominal` coding the pronominality of the theme.

`RealizationOfRecipient` a factor with levels `NP` and `PP` coding the realization of the dative.

`AccessOfRec` a factor with levels `accessible`, `given`, and `new` coding the accessibility of the recipient.

`AccessOfTheme` a factor with levels `accessible`, `given`, and `new` coding the accessibility of the theme.

References

Bresnan, J., Cueni, A., Nikitina, T. and Baayen, R. H. (2007) Predicting the dative alternation, in Bouma, G. and Kraemer, I. and Zwarts, J. (eds.), *Cognitive Foundations of Interpretation*, Royal Netherlands Academy of Sciences, 69-94.

Examples

```
## Not run:
data(dative)
out <- which(is.element(colnames(dative), c("Speaker","Verb")))
dative <- dative[,-out]
dative.ndl <- ndlClassify(RealizationOfRecipient ~ ., data=dative)
ndlStatistics(dative.ndl)


## End(Not run)
```

`estimateActivations` *Estimation of the activations of outcomes (meanings)*

Description

`estimateActivations` is used to estimate the activations for outcomes (meanings) using the equilibrium association strengths (weights) for the Rescorla-Wagner model.

Usage

```
estimateActivations(cuesOutcomes, weightMatrix, unique=FALSE, ...)
```

Arguments

`cuesOutcomes`	A data frame with three variables specifying frequency, cues, and outcomes: `Cues` A character vector specifying the cues. When there is more than one cue, the cues should be separated by underscores. `Outcomes` A character vector specifying the outcomes. When there is more than one outcome, the outcomes should be separated by underscores. `Frequency` A numeric vector specifying the frequency with which a combination of cues and outcomes occurs.
`weightMatrix`	A numeric matrix with as dimensions the number of cues (horizontal) and number of outcomes (vertical). Rows and columns should be labeled with cues and outcomes.
`unique`	A logical that, if `=TRUE`, removes duplicate rows from the activation matrix.
`...`	Control arguments to be passed along from `ndlClassify` and/or `ndlCrossvalidate`.

Details

The activation of an outcome is defined as the sum of the weights on the incoming links from active cues. When the input (the `Cues` in `cuesOutcomes`) contain elements that are not present in the rownames of the `weightMatrix`, such new cues are added to the `weightMatrix` with zero entries. The set of exemplars in `cuesOutcomes` may contain rows with identical cue sets but different outcome sets. Consequently, for such rows, identical vectors of activations of outcomes are generated. In the activation matrix returned by `estimateActivations`, such duplicate entries are removed.

For examples of how the cuesOutcomes data frame should be structured, see the data sets danks, plurals, and serbian. For examples of how the weightMatrix should be structured, see the corresponding output of estimateWeights.

Value

A list with the following components:

activationMatrix A matrix with as dimensions, for rows, the number of exemplars (by-row cue sets, typically word forms), and for columns, the number of unique outcomes (meanings), specifying the activation of a meaning given the cues in the input for a given exemplar.

newCues A vector of cues encountered in cuesOutcomes which were not present in weightMatrix.

... Control arguments to be passed along from ndlClassify, and/or ndlCrossvalidate.

Author(s)

R. H. Baayen & Antti Arppe

References

Baayen, R. H. and Milin, P. and Filipovic Durdevic, D. and Hendrix, P. and Marelli, M., An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

See Also

estimateWeights, danks, plurals, serbian

Examples

```
data(serbian)
serbian$Cues <- orthoCoding(serbian$WordForm, grams=2)
serbian$Outcomes <- serbian$LemmaCase
sw <- estimateWeights(cuesOutcomes=serbian)
sw[1:5,1:6]
activations <- estimateActivations(unique(serbian["Cues"]), sw)$activationMatrix
rownames(activations) <- unique(serbian[["WordForm"]])
activations[1:5,1:6]

syntax <- c("acc", "dat", "gen", "ins", "loc", "nom", "Pl", "Sg")
activations2 <- activations[,!is.element(colnames(activations),syntax)]
head(rownames(activations2), 50)
head(colnames(activations2), 8)
image(activations2, xlab="word forms", ylab="meanings", xaxt="n", yaxt="n")
mtext(c("yena", "...", "zvuke"), side=1, line=1, at=c(0, 0.5, 1), adj=c(0,0,1))
mtext(c("yena", "...", "zvuk"), side=2, line=1, at=c(0, 0.5, 1), adj=c(0,0,1))
```

`estimateWeights` *Estimation of the association weights using the equilibrium equations of Danks (2003) for the Rescorla-Wagner equations.*

Description

A function to estimate the weights (associative strengths) for cue-outcome pairs when learning is in equilibrium, using the equilibrium equations for the Rescorla-Wagner model of Danks (2003).

Usage

```
estimateWeights(cuesOutcomes, removeDuplicates=TRUE, saveCounts=FALSE,
verbose=FALSE, trueCondProb=TRUE, addBackground=FALSE, hasUnicode=FALSE, ...)
```

Arguments

`cuesOutcomes`	A data frame with three variables specifying frequency, cues, and outcomes, that may be created with `ndlCuesOutcomes` or with the accessory script in the inst/scripts directory: `Cues` A character vector specifying the cues. When there is more than one cue, the cues should be separated by underscores. `Outcomes` A character vector specifying the outcomes. When there is more than one outcome, the outcomes should be separated by underscores. `Frequency` A numeric vector specifying the frequency with which a combination of cues and outcomes occurs.
`removeDuplicates`	A logical specifying whether multiple occurrences of a Cue in conjunction with an individual instance of an Outcome shall each be counted as a distinct occurrence of that Cue (`FALSE`: default), or only as a single occurrence (`TRUE`).
`saveCounts`	A logical specifying whether the co-occurrence matrices should be saved. If set equal to `TRUE`, the files `coocCues.rda` and `coocCuesOutcomes.rda` will be saved in the current working directory.
`verbose`	If set to `TRUE`, display diagnostic messages.
`addBackground`	If you would like to add a background rate for all your cues and outcomes, but did not include an general environment cue to all your events, one will be added for you to the matrices, as specified in Danks (2003). If changed from the default (FALSE) to TRUE, background cues will be added. The name used for the background rates is "Environ", and will be included in the output weight matrix.
`trueCondProb`	The conditional probability calculations used will be those specified in Danks (2003). If changed from the default (TRUE) to FALSE, the normalization specified in Baayen, et al (2011) is used.
`hasUnicode`	A logical specifying whether to apply a UTF-8 to integer conversion to the names of the cues. This was implemented to solve issues with differences Unicode cue names.
`...`	Control arguments to be passed along from `ndlClassify` and/or `ndlCrossvalidate`.

Details

Using Rcpp, a C++ based implementation processes all of the data in RAM. The module will check the amount of RAM you have available in your system and warn you if the amount of RAM is insufficient to build your model.

For examples of how the `cuesOutcomes` data frame should be structured, see the data sets `danks`, `plurals`, and `serbian`. N.B. Empty `Cues` or `Outcomes` (effectively having `length = 0`), e.g. `Cues` or `Outcomes` strings with an initial or final underscore or two immediately adjacent underscores, will result in an error.

Value

A matrix with cue-to-outcome association strengths. Rows are cues, and columns are outcomes. Rows and columns are labeled. If addBackground=T, a row named "Environ" will be added to the output.

Acknowledgements

The assistance of Uwe Ligges in getting the C function `cooc` to work within the R framework is greatly appreciated. This C function was removed in version 0.2.0 and replaced with the C++ function by Cyrus Shaoul.

Note

Add a note here.

Author(s)

Cyrus Shaoul, R. H. Baayen and Petar Milin, with contributions from Antti Arppe and Peter Hendrix.

References

Baayen, R. H. and Milin, P. and Filipovic Durdevic, D. and Hendrix, P. and Marelli, M. (2011), An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

See Also

`estimateActivations`, `ndlCuesOutcomes`, `danks`, `plurals`, `serbian`

Examples

```
data(danks)
estimateWeights(cuesOutcomes=danks)

data(plurals)
plurals$Cues <- orthoCoding(plurals$WordForm, grams=1)
round(estimateWeights(cuesOutcomes=plurals),2)

data(serbian)
```

```
serbian$Cues <- orthoCoding(serbian$WordForm, grams=2)
serbian$Outcomes <- serbian$LemmaCase
sw <- estimateWeights(cuesOutcomes=serbian)
round(sw[1:5,1:6],2)
```

`estimateWeightsCompact`

Estimation of the association weights using the equilibrium equations of Danks (2003) for the Rescorla-Wagner equations using a compact binary event file.

Description

A function to estimate the weights (associative strengths) for cue-outcome pairs when learning is in equilibrium, using the equilibrium equations for the Rescorla-Wagner model of Danks (2003) using a compact binary event file.

Usage

```
estimateWeightsCompact(datasource, removeDuplicates=TRUE,
saveCounts=FALSE, verbose=FALSE, MaxEvents=100000000000000,
trueCondProb=TRUE, addBackground=FALSE, ...)
```

Arguments

`datasource`	A data source that is linked with a file naming convention. If the datasource is the string "source", then the following resources will need to exist in the current working directory: `source.events` A directory that contains binary event files in the format specified in learn.module.cpp `source.cues` A text file that contains the full list of cues in the first column, and separated by a tab, the CueID for each cue. Must be encoded in UTF8. `source.outcomes` A text file that contains the full list of outcomes in the first column, and separated by a tab, the OutcomeID for each outcome. Must be encoded in UTF8.
`removeDuplicates`	A logical specifying whether multiple occurrences of a Cue in conjunction with an Outcome shall each be counted as a distinct occurrence of that Cue (`FALSE`), or only as a single occurrence (`TRUE`: default).
`saveCounts`	A logical specifying whether the co-occurrence matrices should be saved. If set equal to `TRUE`, the files `coocCues.rda` and `coocCuesOutcomes.rda` will be saved in the current workspace. Default is FALSE.
`verbose`	If set to `TRUE`, display diagnostic messages.
`MaxEvents`	If changed from the default value, the learning algorithm will stop learning after using the first N events in the training data. This actually number of events used may be slightly higher than the number specified.

addBackground	If you would like to add a background rate for all your cues and outcomes, but did not include an general environment cue to all your events, one will be added for you to the matrices, as specified in Danks (2003). If changed from the default (FALSE) to TRUE, background cues will be added. The name used for the background rates is "Environ", and will be included in the output weight matrix.
trueCondProb	The conditional probability calculations used will be those specified in Danks (2003). If changed from the default (TRUE) to FALSE, the normalization specified in Baayen, et al (2011) is used.
...	Control arguments to be passed along from ndlClassify and/or ndlCrossvalidate.

Details

Using Rcpp, a C++ based implementation processes all of the data RAM. The module will check the amount of RAM you have available in your system and warn you of RAM is insufficient to build your model.

Value

A matrix with cue-to-outcome association strengths. Rows are cues, and columns are outcomes. Rows and columns are labeled. If addBackground=T, a row named "Environ" will be added to the output.

Acknowledgements

Thanks to all the beta testers of the ndl package.

Note

Add a note here.

Author(s)

Cyrus Shaoul, R. H. Baayen and Petar Milin, with contributions from Antti Arppe and Peter Hendrix.

References

Baayen, R. H. and Milin, P. and Filipovic Durdevic, D. and Hendrix, P. and Marelli, M., (2011) An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

See Also

estimateActivations

Examples

```
  message("This module requires data in a non-portable format to
demonstrate how it works.")
```

`learn` *Count cue-outcome co-occurences needed to run the Danks equations.*

Description

An internal function to count cue-outcome co-occurrences.

Usage

```
learn(data,RemoveDuplicates,verbose,MaxEvents,addBackground)
```

Arguments

`data`	A directory where the binary event data files are located.
`RemoveDuplicates`	A logical specifying whether multiple occurrences of a Cue in conjunction with an Outcome shall each be counted as a distinct occurrence of that Cue (`FALSE`), or only as a single occurrence (`TRUE`: default).
`verbose`	Display diagnostic messages or not.
`MaxEvents`	The total number of events to learn from before stopping learning. Checked one time per compact data file.
`addBackground`	Option to add background rates.

Details

This function calls an Rcpp function of the same name to process the data in the compact data format.

Value

A list of two matrices with cue-cue coocurrences and cue-outcome cooccurrences and a vector with background rates.

Acknowledgements

Thanks to all the testers!

Note

No temporary files are used.

Author(s)

Cyrus Shaoul

References

Baayen, R. H. and Milin, P. and Filipovic Durdevic, D. and Hendrix, P. and Marelli, M., An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

See Also

estimateActivations, ndlCuesOutcomes, estimateWeightsCompact, danks, plurals, serbian

Examples

```
#None (internal function)
```

`learnLegacy` *Count cue-outcome co-occurrences needed to run the Danks equations.*

Description

An internal function to count cue-outcome co-occurrences.

Usage

```
learnLegacy(DFin,RemoveDuplicates,verbose)
```

Arguments

`DFin`	A dataframe, as defined in the documentation for estimateWeights.
`RemoveDuplicates`	A logical specifying whether multiple occurrences of a Cue in conjunction with an Outcome shall each be counted as a distinct occurrence of that Cue (`FALSE`), or only as a single occurrence (`TRUE`: default).
`verbose`	Display diagnostic messages or not.

Details

This function calls an Rcpp function of the same name to process the data in the DFin data frame.

Value

A list of two matrices with cue-cue co-occurrences and cue-outcome co-occurrences.

Acknowledgements

Thanks to all the testers out there! Martijn, you know who you are.

Note

No temporary files are used.

Author(s)

Cyrus Shaoul

References

Baayen, R. H. and Milin, P. and Filipovic Durdevic, D. and Hendrix, P. and Marelli, M., An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

See Also

`estimateActivations`, `ndlCuesOutcomes`, `estimateWeights`, `danks`, `plurals`, `serbian`

Examples

```
#None (internal function)
```

---

lexample *Lexical example data illustrating the Rescorla-Wagner equations*

Description

Ten monomorphemic and inflected English words with fictive frequencies, and meanings.

Usage

```
data(lexample)
```

Format

A data frame with 10 observations on the following 3 variables:

`Word` A character vector specifying word forms

`Frequency` A numeric vector with the – fictive – frequencies of occurrence of the words

`Outcomes` A character vector specifying the meaning components of the words, separated by underscores

Details

This example lexicon is used in Baayen et al. (2011) (table 8, figure 4) to illustrate the Rescorla-Wagner equations.

References

Baayen, R. H., Milin, P., Filipovic Durdevic, D., Hendrix, P. and Marelli, M. (2011), An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

See Also

RescorlaWagner, orthoCoding

Examples

```
## Not run:
data(lexample)
lexample$Cues <- orthoCoding(lexample$Word, grams=1)
par(mfrow=c(2,2))
lexample.rw <- RescorlaWagner(lexample, nruns=25, traceCue="h",traceOutcome="hand")
plot(lexample.rw)
mtext("h - hand", 3, 1)

lexample.rw <- RescorlaWagner(lexample, nruns=25, traceCue="s",traceOutcome="plural")
plot(lexample.rw)
mtext("s - plural", 3, 1)

lexample.rw <- RescorlaWagner(lexample, nruns=25, traceCue="a",traceOutcome="as")
plot(lexample.rw)
mtext("a - as", 3, 1)

lexample.rw <- RescorlaWagner(lexample, nruns=25, traceCue="s",traceOutcome="as")
plot(lexample.rw)
mtext("s - as", 3, 1)
par(mfrow=c(1,1))

## End(Not run)
```

`modelStatistics` *Calculate a range of goodness of fit measures for an object fitted with some multivariate statistical method that yields probability estimates for outcomes.*

Description

`modelStatistics` calculates a range of goodness of fit measures.

Usage

```
modelStatistics(observed, predicted, frequency=NA, p.values,
   n.data, n.predictors, outcomes=levels(as.factor(observed)),
   p.normalize=TRUE, cross.tabulation=TRUE,
   p.zero.correction=1/(NROW(p.values)*NCOL(p.values))^2)
```

Arguments

observed	observed values of the response variable
predicted	predicted values of the response variable; typically the outcome estimated to have the highest probability
frequency	frequencies of observed and predicted values; if NA, frequencies equal to 1 for all observed and predicted values
p.values	matrix of probabilities for all values of the response variable (i.e outcomes)
n.data	sum frequency of data points in model
n.predictors	number of predictor levels in model
outcomes	a vector with the possible values of the response variable
p.normalize	if TRUE, probabilities are normalized so that sum(P) of all outcomes for each datapoint is equal to 1
cross.tabulation	if TRUE, statistics on the crosstabulation of observed and predicted response values are calculated with crosstableStatistics
p.zero.correction	a function to adjust slightly response/outcome-specific probability estimates which are exactly P=0; necessary for the proper calculation of pseudo-R-squared statistics; by default calculated on the basis of the dimensions of the matrix of probabilities p.values.

Value

A list with the following components:

loglikelihood.null Loglikelihood for null model

loglikelihood.model Loglikelihood for fitted model

deviance.null Null deviance

deviance.model Model deviance

R2.likelihood (McFadden's) R-squared

R2.nagelkerke Nagelkerke's R-squared

AIC.model Akaike's Information Criterion

BIC.model Bayesian Information Criterion

C index of concordance C (for binary response variables only)

crosstable Crosstabulation of observed and predicted outcomes, if cross.tabulation=TRUE

crosstableStatistics(crosstable) Various statistics calculated on crosstable with crosstableStatistics, if cross.tabulation=TRUE

Author(s)

Antti Arppe and Harald Baayen

References

Arppe, A. 2008. Univariate, bivariate and multivariate methods in corpus-based lexicography – a study of synonymy. Publications of the Department of General Linguistics, University of Helsinki, No. 44. URN: http://urn.fi/URN:ISBN:978-952-10-5175-3.

Arppe, A., and Baayen, R. H. (in prep.) Statistical modeling and the principles of human learning.

Hosmer, David W., Jr., and Stanley Lemeshow 2000. Applied Regression Analysis (2nd edition). New York: Wiley.

See Also

See also ndlClassify, ndlStatistics, crosstableStatistics.

Examples

```
data(think)
think.ndl <- ndlClassify(Lexeme ~ Agent + Patient, data=think)
probs <- acts2probs(think.ndl$activationMatrix)$p
preds <- acts2probs(think.ndl$activationMatrix)$predicted
n.data <- nrow(think)
n.predictors <- nrow(think.ndl$weightMatrix) *
   ncol(think.ndl$weightMatrix)
modelStatistics(observed=think$Lexeme, predicted=preds, p.values=probs,
   n.data=n.data, n.predictors=n.predictors)
```

---

ndlClassify *Classification using naive discriminative learning.*

Description

ndlClassify uses the equilibrium equations of Danks (2003) for the Rescorla-Wagner model (1972) to estimate association strengths (weights) for cues (typically levels of factorial predictors) to outcomes (typically a binary or polytomous response variable). Given the association strengths, the probability of a response level is obtained by summation over the weights on active incoming links.

Usage

```
ndlClassify(formula, data, frequency=NA, variable.value.separator="", ...)

## S3 method for class 'ndlClassify'
print(x, max.print=10, ...)
```

Arguments

formula	An object of class formula (or one that can be coerced to that class): a symbolic description of the model to be fitted.
data	A data frame containing the variables in the model.
frequency	A numeric vector (or the name of a column in the input data frame) with the frequencies of the exemplars. If absent, each exemplar is assigned a frequency equal to 1.
x	An object of the class "ndlClassify" fitted with ndlClassify to be printed with print.ndlClassify.
max.print	The maximum number of rows of the weightMatrix to be output when printing with print.ndlClassify; by default equal to 10; if set to NA all rows will be output.
variable.value.separator	A character string which will separate variable names from variable values in their combination as cue values; by default an empty character string (="").
...	Control arguments to be passed along to ndlCuesOutcomes, estimateWeights, estimateActivations, and/or print.ndlClassify.

Details

Classification by naive discriminative learning.

Value

A list of the class "ndlClassify" with the following components:

activationMatrix A matrix specifying for each row of the input data frame the activations (probabilities) of the levels of the response variable (nrow observations by nlevels of response variable).

weightMatrix A matrix specifying for each cue (predictor value) the association strength (weight) to each outcome (level of the response variable) (number of distinct predictor values by number of response levels).

cuesOutcomes The input data structure for naive discriminative learning created by ndlCuesOutcomes based on the data argument (number of observations by 3: Frequency, Cues, Outcomes).

call The call matched to fit the resulting "ndlClassify" object.

formula The formula specified for fitting the resulting "ndlClassify" object.

data The supplied data argument, excluding all elements not specified for the modeling task in formula and frequency.

Author(s)

R. H. Baayen and Antti Arppe

References

Baayen, R. H. and Milin, P. and Filipovic Durdevic, D. and Hendrix, P. and Marelli, M., An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

Danks, D. (2003). Equilibria of the Rescorla-Wagner model. Journal of Mathematical Psychology, 47 (2), 109-121.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In Black, A. H., & Prokasy, W. F. (Eds.), Classical conditioning II: Current research and theory (pp. 64-99). New York: Appleton-Century-Crofts.

Arppe, A. and Baayen, R. H. (in prep.) Statistical classification and principles of human learning.

See Also

summary.ndlClassify, plot.ndlClassify, anova.ndlClassify, predict.ndlClassify, ndlCuesOutcomes, estimateWeights, cueCoding

Examples

```
data(think)
set.seed(314)
think <- think[sample(1:nrow(think),500),]
think.ndl <- ndlClassify(Lexeme ~ (Person * Number * Agent) + Register,
   data=think)
summary(think.ndl)

## Not run:
think.ndl.SA <- ndlClassify(Lexeme ~ (Polarity + Voice + Mood + Person +
  Number + Covert + ClauseEquivalent + Agent + Patient + Manner + Time +
  Modality1 + Modality2 + Source + Goal + Quantity + Location +
  Duration + Frequency + MetaComment + ReasonPurpose + Condition +
  CoordinatedVerb)^2 + Author + Section, data=think)
summary(think.ndl.SA)

## End(Not run)

## Not run:
data(dative)
out <- which(is.element(colnames(dative), c("Speaker","Verb")))
dative <- dative[-out]
dative.ndl <- ndlClassify(RealizationOfRecipient ~ ., data=dative)
summary(dative.ndl)


## End(Not run)
```

ndlCrossvalidate *Crossvalidation of a Naive Discriminative Learning model.*

Description

ndlCrossvalidate undertakes a crossvalidation of a Naive Discriminative Learning model fitted using ndlClassify.

Usage

```
ndlCrossvalidate(formula, data, frequency=NA, k=10, folds=NULL, ...)
```

Arguments

formula	An object of class formula (or one that can be coerced to that class): a symbolic description of the model to be fitted. If alternatively set to =NA, the data argument is expected to be in the ndl internal format as generated by ndlCuesOutcomes, and ndlCrossvalidate will check that this is the case.
data	A data frame (as in ndlClassify) containing the variables in the formula specifying the model.
frequency	A numeric vector (or the name of a column in the input data frame) with the frequencies of the exemplars. If absent, each exemplar is assigned a frequency equal to 1.
k	The number of folds, by default equal to 10.
folds	A list of user-defined folds, each item on the list representing a vector of indices indicating lines in the data frame to be used for testing a model fitted with the rest of the data. By default NULL, so that the folds are determined with random selection by the function ndlCrossvalidate.
...	Control arguments to be passed along to auxiliary functions, in specific estimateWeights and/or estimateActivations.

Details

Crossvalidation of a Naive Discriminative Learning model.

Value

A list of the class "ndlCrossvalidate" with the following components:

call The call matched by ndlCrossvalidate

formula The formula specified for ndlCrossvalidate

fits A list of individual fits resulting from ndlCrossvalidate

k The number of folds, by default equal to 10

n.total The sum frequency of data points

`n.train` The size of the training set

`n.test` The size of of the testing set

`folds` A list with the folds used in the crossvalidation; either selected at random by `ndlCrossvalidate` or provided by the user.

Author(s)

Antti Arppe

References

Baayen, R. H. and Milin, P. and Filipovic Durdevic, D. and Hendrix, P. and Marelli, M., An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

Arppe, A. and Baayen, R. H. (in prep.). Statistical modeling and the principles of human learning.

See Also

`summary.ndlCrossvalidate`, `ndlStatistics`, `ndlCuesOutcomes`, `cueCoding`, `estimateWeights`, `estimateActivations`

Examples

```
data(think)
set.seed(314)
think <- think[sample(1:nrow(think),500),]
think.cv5 <- ndlCrossvalidate(Lexeme ~ Agent + Patient, data=think, k=5)
summary(think.cv5)
rm(think)

## Not run:
data(think)
think.cv10 <- ndlCrossvalidate(Lexeme ~ Person + Number + Agent + Patient + Register,
   data=think, k=10)
summary(think.cv10)

## End(Not run)
## Not run:
library(languageR)
data(finalDevoicing)
finDev.cv10 <- ndlCrossvalidate(Voice ~ Onset1Type + Onset2Type + VowelType *
   ConsonantType * Obstruent + Nsyll + Stress, data=finalDevoicing, k=10)
summary(finDev.cv10)


## End(Not run)
```

ndlCuesOutcomes *Creation of dataframe for Naive Discriminative Learning from formula specification*

Description

ndlCuesOutcomes creates a dataframe for fitting a naive discriminative classification model with ndlClassify, using the specified formula and provided data.

Usage

```
ndlCuesOutcomes(formula, data, frequency=NA,
  numeric2discrete=function(x) Hmisc::cut2(x,g=g.numeric), g.numeric=2,
  check.values=TRUE, ignore.absent=NULL, variable.value.separator="", ...)
```

Arguments

formula	An object of class formula (or one that can be coerced to that class): a symbolic description of the model to be fitted.
data	A data frame containing the variables in the model.
frequency	A numeric vector (or the name of a column in the input data frame) with the frequencies of the exemplars. If absent, each exemplar is assigned a frequency equal to 1.
numeric2discrete	A function to transform a continuous numeric predictor into a number of discrete classes, by default cut2 from the Hmisc package. If set to NULL, each value of each numeric predictor will be treated as a discrete class of its own.
g.numeric	A parameter to be passed to the numeric2discrete function (parameter g for Hmisc::cut2(..., g=g.numeric, ...), or a user-defined function), determining the desired number of discrete categories for each numeric predictor; by default equal to 2.
check.values	A logical specifying whether underscores '_' in predictor values should substituted with periods '.'; if =FALSE, the predictor values will be only checked and an error message will result if any underscores are discovered.
ignore.absent	A character vector specifying one or more values for any predictor (e.g. NIL, None and/or Other) which may be considered truely absent cues in terms of the Rescorla-Wagner equations; by default set to NULL so that all values of all predictors will be treated as present cues.
variable.value.separator	A character string which will separate variable names from variable values in their combination as cue values; by default an empty character string (=""").
...	Control arguments to be passed along to estimateWeights.

Details

Creates a dataframe to be used for fitting a Naive Discriminatory Learning classifier model.

Value

A dataframe with the following columns:

`Frequency` Frequency with which the specific Cues and Outcomes co-occur.

`Cues` A character vector of sets of Cues per instance, with Cues separated by underscore '_'.

`Outcomes` A character vector of Outcomes per instance.

Author(s)

R. H. Baayen and Antti Arppe

References

Arppe, A. and Baayen, R. H. (in prep.) Statistical modeling and the principles of human learning.

See Also

`cueCoding`, `ndlClassify`

Examples

```
data(think)
set.seed(314)
think <- think[sample(1:nrow(think),500),]
think.CuesOutcomes <- ndlCuesOutcomes(Lexeme ~ (Person * Number * Agent) + Register,
data=think)
head(think.CuesOutcomes)

## Not run:
data(dative)
dative.cuesOutcomes <- ndlCuesOutcomes(RealizationOfRecipient ~ LengthOfRecipient +
   LengthOfTheme, data=dative, numeric2discrete=NULL)
table(dative.cuesOutcomes$Cues)

dative.cuesOutcomes1 <- ndlCuesOutcomes(RealizationOfRecipient ~ LengthOfRecipient +
   LengthOfTheme, data=dative)
table(dative.cuesOutcomes1$Cues)

dative.cuesOutcomes2 <- ndlCuesOutcomes(RealizationOfRecipient ~ LengthOfRecipient +
   LengthOfTheme, data=dative, g.numeric=3)
table(dative.cuesOutcomes2$Cues)


## End(Not run)
```

`ndlStatistics` *Calculate goodness of fit statistics for a naive discriminative learning model.*

Description

`ndlStatistics` takes an Naive Discriminary Learning model object as generated by `ndlClassify` and calculates a range of goodness of fit statistics using `modelStatistics`.

Usage

```
ndlStatistics(ndl, ...)
```

Arguments

`ndl`	A naive discriminative learning model fitted with `ndlClassify`.
`...`	Control arguments to be passed along to `modelStatistics`.

Value

A list with the following components:

`n.data` sum frequency of data points

`df.null` degrees of freedom of the Null model

`df.model` degrees of freedom of the fitted model

`statistics` a list of various measures of goodness of fit calculated with `modelStatistics`

Author(s)

Antti Arppe and Harald Baayen

References

Arppe, A. and Baayen, R. H. (in prep.) Statistical modeling and the principles of human learning.

See Also

See also `ndlClassify`, `modelStatistics`.

Examples

```
data(think)
set.seed(314)
think <- think[sample(1:nrow(think),500),]
think.ndl <- ndlClassify(Lexeme ~ Agent + Patient, data=think)
ndlStatistics(think.ndl)

## Not run:
```

```
data(dative)
dative.ndl <- ndlClassify(RealizationOfRecipient ~ AnimacyOfRec + DefinOfRec +
   PronomOfRec + AnimacyOfTheme + DefinOfTheme + PronomOfTheme, data=dative)
ndlStatistics(dative.ndl)

## End(Not run)
```

`ndlVarimp` *Permutation variable importance for classification using naive discriminative learning.*

Description

`ndlVarimp` uses permutation variable importance for naive discriminative classification models, typically the output of `ndlClassify`.

Usage

```
ndlVarimp(object, verbose=TRUE)
```

Arguments

`object`	An object of class `"ndlClassify"` (or one that can be coerced to that class); typically a model object as produced by `ndlClassify`.
`verbose`	A logical (default TRUE) specifying whether the successive predictors being evaluated should be echoed to stdout.

Details

Variable importance is assessed using predictor permutation. Currently, conditional permutation variable importance (as for `varimp` for random forests in the `party` package) is not implemented.

Value

A list with two numeric vectors:

`concordance` For binary response variables, a named vector specifying for each predictor the index of concordance when that predictor is permuted. For polytomous response variables, NA.

`accuracy` A named vector specifying for each predictor the accuracy of the model with that predictor permuted.

Author(s)

R. H. Baayen and Antti Arppe

References

R. Harald Baayen (2011). Corpus linguistics and naive discriminative learning. Brazilian journal of applied linguistics, 11, 295-328.

Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin and Achim Zeileis (2008). Conditional Variable Importance for Random Forests. BMC Bioinformatics, 9, 307.

See Also

summary.ndlClassify, plot.ndlClassify, anova.ndlClassify, ndlCuesOutcomes, estimateWeights, cueCoding

Examples

```
## Not run:
data(dative)
dative <- dative[!is.na(dative$Speaker),-2]
dative.ndl <- ndlClassify(RealizationOfRecipient ~ ., data=dative)
dative.varimp <- ndlVarimp(dative.ndl)

library(lattice)
dotplot(sort(summary(dative.ndl)$statistics$accuracy-dative.varimp$accuracy),
   xlab="permutation variable importance")

## End(Not run)
```

---

numbers *Example data illustrating the Rescorla-Wagner equations as applied to numerical cognition by Ramscar et al. (2011).*

Description

The data used in simulation 3 of Ramscar et al. (2011) on numerical cognition.

Usage

```
data(lexample)
```

Format

A data frame with 10 observations on the following 3 variables.

`Cues` A character vector specifying cues for quantities, separated by underscores.

`Frequency` The frequencies with which the numbers appear in the COCA corpus.

`Outcomes` A character vector specifying numerical outcomes associated with the input quantities.

Details

The cues represent learning trials with objects of the same size, shape and color. The numeric cues represent the presence of at least one subset of the specified size. The cues `exactlyn` represent the presence of exactly `n` objects. We are indebted to Michael Ramscar to making this data set available for inclusion in the package.

References

Michael Ramscar, Melody Dye, Hanna Muenke Popick & Fiona O'Donnell-McCarthy (2011), The Right Words or Les Mots Justes? Why Changing the Way We Speak to Children Can Help Them Learn Numbers Faster. Manuscript, Department of Psychology, Stanford University.

Examples

```
data(numbers)

traceCues=c( "exactly1", "exactly2", "exactly3", "exactly4",
"exactly5", "exactly6", "exactly7", "exactly10", "exactly15")
traceOutcomes=c("1", "2", "3", "4", "5", "6", "7", "10", "15")

ylimit=c(0,1)
par(mfrow=c(3,3),mar=c(4,4,1,1))

for (i in 1:length(traceCues)){
  numbers.rw = RescorlaWagner(numbers, nruns=1,
    traceCue=traceCues[i],traceOutcome=traceOutcomes[i])
  plot(numbers.rw, ylimit=ylimit)
  mtext(paste(traceCues[i], " - ", traceOutcomes[i], sep=""),
    side=3, line=-1, cex=0.7)
}
par(mfrow=c(1,1))
```

---

orthoCoding *Code a character string (written word form) as letter n-grams*

Description

`orthoCoding` codes a character string into unigrams, bigrams, ..., n-grams, with as default bigrams as the substring size. If tokenization is not at the letter/character level, a token separator can be provided.

Usage

```
orthoCoding(strings=c("hel.lo","wor.ld"), grams = c(2), tokenized = F, sepToken = '.')
```

Arguments

strings	A character vector of strings (usually words) to be recoded as n-grams.
grams	A vector of numbers, each one a size of ngram to be produced. For example a vector like grams=c(1,3) will create the unigram and trigram cues from the input.
tokenized	If tokenzied is FALSE (the default), the input strings are split into letters/characters. If it is set to TRUE, the strings will be split up based on the value of sepToken.
sepToken	A string that defines which character will be used to separate tokens when tokenized is TRUE. Defaults to the "." character.

Value

A vector of grams (joined by underscores), one for each word in the input vector *words*.

Author(s)

Cyrus Shaoul, Peter Hendrix and Harald Baayen

References

Baayen, R. H. and Milin, P. and Filipovic Durdevic, D. and Hendrix, P. and Marelli, M., An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

See Also

See also estimateWeights.

Examples

```
#Default
orthoCoding(tokenize=FALSE)
#With tokenizing on a specific character
orthoCoding(tokenize=TRUE)

#Comparing different n-gram sizes
data(serbian)
serbian$Cues=orthoCoding(serbian$WordForm, grams=2)
head(serbian$Cues)
serbian$Cues=orthoCoding(serbian$WordForm, grams=c(2,4))
head(serbian$Cues)
```

plot.ndlClassify *Plot function for selected results of* ndlClassify.

Description

This function presents visually the estimated weights or expected probabilities for a model fitted with ndlClassify

Usage

```
## S3 method for class 'ndlClassify'
plot(x, values="weights", ...)

## S3 method for class 'ndlWeights'
plot(x, type="density", predictors=NULL, outcomes=NULL,
panes="single", lty=NULL, col=NULL, mfrow=NULL, main=NULL,
legend.position="topright", ...)

## S3 method for class 'ndlProbabilities'
plot(x, type="density", select="all",
panes="single", lty=NULL, col=NULL, pch=NULL, mfrow=NULL,
main=NULL, legend.position="topright", ...)
```

Arguments

x	A object of the class "ndlClassify" produced by ndlClassify, consisting of a list including estimated weights for predictors and association strengths for outcome-predictor combinations.
values	A character string specifiying whether estimated weights (default) or expected probabilities should be plotted.
type	A character string spefifying the type of plot to be drawn; density is available for both value types as default, while a histogram (hist) is available only for plot.ndlWeights and sorted values (sort) only for plot.ndlProbabilities.
panes	A character string specifying whether a single pane (default) integrating all component plots, or multiple panes for each individual component plot are to be plotted. If multiple panes are selected, the number or rows and columns is specified automatically. Alternatively, one can invoke the plotting of multiple panes by explicitly specifying the appropriate number of rows and columns with the parameter mfrow (N.B. this overrides panes="single").
predictors	A regular expression specifying which predictors and their values should be included in the plot(s); by default =NULL so that all predictors incorporated in the ndlClassify model will be included.
outcomes	A list of outcomes to be included in the plot; by default =NULL so that all outcomes will be considered.

`select` For `plot.ndlProbabilities`, a character string specifying which instance-wise probability estimates should be plotted; by default `all`, other values are `max` for instance-wise maximum probabilities, `min` for instance-wise minimum probabilities, `maxmin`, `minmax` for both maximum and minimum instance-wise probabilities. Alternatively, a numeric vector `c(1,2,...)` specifying selected ranks of the instance-wise probability estimates can be provided, with `1` corresponding to the instance-wise maximum probability estimates.

`lty`, `col`, `pch`, `mfrow`, `main`, `legend.position`
Specifications of various graphical parameters (see `par`) to be used in the plots; if any of these is set to `=NULL` default settings will be used (for `legend.position`, the default value is `topright`). Note that `lty` is relevant only to `plot.ndlWeights(..., type="density", ...)` and `plot.ndlProbabilities(..., type="density", ...)`, and `pch` only to `plot.ndlProbabilities(..., type="sort", ...)`.

`...` Arguments to be passed to methods, such as graphical parameters (see `par`).

Value

A plot of the selected type is produced on the graphics device.

Author(s)

Antti Arppe and R. H. Baayen

References

Arppe, A. and Baayen, R. H. (in prep.)

See Also

`ndlClassify`, `acts2probs`

Examples

```
## Not run:

data(think)
think.ndl <- ndlClassify(Lexeme ~ Agent + Patient + Section, data=think)

plot(think.ndl, values="weights")
plot(think.ndl, values="weights", type="hist", panes="multiple")
plot(think.ndl, values="weights", type="density", panes="multiple")
plot(think.ndl, values="weights", type="density", panes="multiple",
   predictors="Section*")
plot(think.ndl, values="weights", type="density", panes="multiple",
   predictors="Patient*")
plot(think.ndl, values="weights", type="hist", panes="multiple", col=1:4)
plot(think.ndl, values="weights", type="density", panes="single",
   outcomes=c("ajatella","miettia","pohtia","harkita"))

plot(think.ndl, values="probabilities")
plot(think.ndl, values="probabilities", panes="multiple")
```

```
plot(think.ndl, values="probabilities", select="max")
plot(think.ndl, values="probabilities", select=c(1:3))
plot(think.ndl, values="probabilities", panes="multiple", select=c(1:3))
plot(think.ndl, values="probabilities", type="sort", legend.position="topleft")
plot(think.ndl, values="probabilities", type="sort", pch=".",
   legend.position="topleft")
plot(think.ndl, values="probabilities", type="sort", pch=".", panes="multiple")

## End(Not run)
```

`plot.RescorlaWagner` *Plot function for the output of* `RescorlaWagner`.

Description

This function graphs the Rescorla-Wagner weights for a cue-outcome pair against learning time.

Usage

```
## S3 method for class 'RescorlaWagner'
plot(x, asymptote=TRUE, xlab="t", ylab="weight", ylimit=NA, ...)
```

Arguments

`x`	A object of the class `"RescorlaWagner"` produced by `RescorlaWagner`, consisting of a list including estimated weights for the incremental and equilibrium stages.
`asymptote`	A logical specifying whether the equilibrium asymptotic weight should be added to the plot.
`xlab`	Label for x-axis, by default `"t"`.
`ylab`	Label for y-axis, by default `"weight"`.
`ylimit`	The range of values to be displayed on the Y axis. By default, this will be determined from the data itself.
`...`	Arguments to be passed to methods, such as graphical parameters (see `link{par}`).

Value

A plot is produced on the graphics device.

Author(s)

R. H. Baayen and Antti Arppe

References

Danks, D. (2003). Equilibria of the Rescorla-Wagner model. Journal of Mathematical Psychology, 47 (2), 109-121.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In Black, A. H., & Prokasy, W. F. (Eds.), Classical conditioning II: Current research and theory (pp. 64-99). New York: Appleton-Century-Crofts.

See Also

RescorlaWagner, orthoCoding

Examples

```
data(lexample)
lexample$Cues <- orthoCoding(lexample$Word, grams=1)
lexample.rw <- RescorlaWagner(lexample, nruns=25,
   traceCue="h", traceOutcome="hand")
plot(lexample.rw)
mtext("h - hand", 3, 1)

# Full example

## Not run:
par(mfrow=c(2,2))
lexample.rw <- RescorlaWagner(lexample, nruns=25,
   traceCue="h", traceOutcome="hand")
plot(lexample.rw)
mtext("h - hand", 3, 1)

lexample.rw <- RescorlaWagner(lexample, nruns=25,
   traceCue="s", traceOutcome="plural")
plot(lexample.rw)
mtext("s - plural", 3, 1)

lexample.rw <- RescorlaWagner(lexample, nruns=25,
   traceCue="a", traceOutcome="as")
plot(lexample.rw)
mtext("a - as", 3, 1)

lexample.rw <- RescorlaWagner(lexample, nruns=25,
   traceCue="s", traceOutcome="as")
plot(lexample.rw)
mtext("s - as", 3, 1)
par(mfrow=c(1,1))

## End(Not run)
```

plurals *Artificial data set used to illustrate the Rescorla-Wagner equations and naive discriminative learning.*

Description

Data set with 10 English words of different (ad hoc) frequencies, each with a lexical meaning and a grammatical meaning.

Usage

```
data(plurals)
```

Format

A data frame with 10 observations on the following 3 variables:

`WordForm` A character vector of word forms (cues).

`Frequency` A numeric vector of frequencies.

`Outcomes` A character vector of meanings (outcomes). Meanings are separated by underscores. The `NIL` meaning is ignored.

Source

Baayen, R. H. and Milin, P. and Filipovic Durdevic, D. and Hendrix, P. and Marelli, M., An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

References

Baayen, R. H. and Milin, P. and Filipovic Durdevic, D. and Hendrix, P. and Marelli, M., An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

Examples

```
data(plurals)
plurals$Cues <- orthoCoding(plurals$WordForm, grams=1)
estimateWeights(cuesOutcomes=plurals)
```

predict.ndlClassify *Predict method for ndlClassify objects*

Description

Obtains predictions on the basis of a fitted "ndlClassify" object on data already incorporated in the object or on new data with the same predictors as the originally fitted model object.

Usage

```
## S3 method for class 'ndlClassify'
predict(object, newdata=NULL, frequency=NA,
   type="choice", ...)
```

Arguments

object	objects of class "ndlClassify", typically the result of a call to ndlClassify.
newdata	optionally, a data frame in which to look for variables with which to predict. If omitted (i.e. set to NULL), the original data used to fit the object are used.
frequency	A numeric vector (or the name of a column in the (new) data frame newdata) with the frequencies of the exemplars. If absent, each exemplar is assigned a frequency equal to 1.
type	the type of prediction requested. The default option type="choice" produces the predicted individual discrete choices (i.e. Outcomes), given the predictor Cues selected for fitting the original object. The option type="acts" provides the sum activations for each Outcome given the Cue combinations in newdata (or in the original data in object, while the alternative type="probs" yields the distributions of predicted probabilities (based on the activations) over the Outcome responses.
...	further arguments passed to and from other functions.

Details

If newdata is omitted the predictions are based on the data used for the fit.

Value

a vector predicted, or matrix of activations activations, or a matrix of predictions probabilities.

Author(s)

Antti Arppe

References

Arppe, A. and Baayen, R. H. (in prep.) Statistical classification and principles of human learning.

See Also

ndlClassify, estimateActivations, acts2probs

Examples

```
data(think)
think.ndl <- ndlClassify(Lexeme ~ Agent + Patient, data=think[1:300,])
head(predict(think.ndl, type="choice"))
predict(think.ndl, newdata=think[301:320,], type="probs")
predict(think.ndl, newdata=think[301:320,], type="acts")
```

---

random.pseudoinverse *Calculate an approximation of the pseudoinverse of a matrix.*

Description

An internal function that uses an approximation of the SVD using the first k singular values of A to calculate the pseudo-inverse. Only used when the cue-cue matrix contains more than 20,000 cues.

Usage

```
random.pseudoinverse(m, verbose=F, k = 0)
```

Arguments

m	A matrix.
k	If k = 0, the default, k will be set to the size of 3/4 of the singular values. If not, the k-rank approximation will be calculated.
verbose	Display diagnostic messages or not.

Details

This idea was proposed by Gunnar Martinsson Associate Professor and Director of Graduate Studies Department of Applied Mathematics, University of Colorado at Boulder http://amath.colorado.edu/faculty/martinss/ And with ideas from: Yoel Shkolnisky and his Out-of-Core SVD code: https://sites.google.com/site/yoelshkolnisky/software

Value

The approximate pseudoinverse of the input matrix

Acknowledgements

Thanks to Gunnar for his help with this!

Note

No temporary files are used.

Author(s)

Cyrus Shaoul

References

"Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions" Nathan Halko, Per-Gunnar Martinsson, Joel A. Tropp http://arxiv.org/abs/0909.4061

See Also

estimateWeights, estimateWeightsCompact,

Examples

```
#None (internal function)
```

`RescorlaWagner` *Implementation of the Rescorla-Wagner equations.*

Description

`RescorlaWagner` implements an iterative simulation based on the Rescorla-Wagner equations. Given a data frame specifying cues, outcomes, and frequencies, it calculates, for a given cue-outcome pair, the temporal sequence of developing weights.

Usage

```
RescorlaWagner(cuesOutcomes, traceCue="h", traceOutcome="hand",
   nruns=1, random=TRUE, randomOrder = NA, alpha=0.1, lambda=1,
   beta1=0.1, beta2=0.1)
```

Arguments

`cuesOutcomes`	A data frame specifying cues, outcomes, and frequencies of combinations of cues and outcomes. In the data frame, cues and outcomes should be character vectors.
`traceCue`	A character string specifying the cue to be traced over time.
`traceOutcome`	A character string specifying the outcome to be traced over time.
`nruns`	An integer specifying the number of times the data have to be presented for learning. The total number of learning trials is `nruns*sum(cuesOutcomes$Frequency)`.
`random`	A logical specifying whether the order of the learning trials for a given run should be randomly reordered. Can be set to `FALSE` in case all frequencies are 1, and the sequence of learning trials in `cuesOutcomes` is given by the order of the rows.
`randomOrder`	If not `NA`, a vector specifying the (usually random) order of the learning trials.

alpha	The salience of the trace cue.
lambda	The maximum level of associative strength possible.
beta1	The salience of the situation in which the outcome occurs.
beta2	The salience of the situation in which the outcome does not occur.

Details

The equilibrium weights (Danks, 2003) are also estimated.

Value

An object of the class "RescorlaWagner", being a list with the following components:

weightvector A numeric vector with the weights for all nruns*sum(dat[,"Frequency"]) training trials.

equilibriumWeight The weight of the cue-outcome link at equilibrium.

traceCue A character string specifying the trace cue.

traceOutcome A character string specifying the trace outcome.

Author(s)

R. H. Baayen and Antti Arppe

References

Danks, D. (2003). Equilibria of the Rescorla-Wagner model. Journal of Mathematical Psychology, 47 (2), 109-121.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In Black, A. H., & Prokasy, W. F. (Eds.), Classical conditioning II: Current research and theory (pp. 64-99). New York: Appleton-Century-Crofts.

See Also

orthoCoding, plot.RescorlaWagner, numbers

Examples

```
data(lexample)
lexample$Cues <- orthoCoding(lexample$Word, grams=1)
lexample.rw <- RescorlaWagner(lexample, nruns=25,
   traceCue="h", traceOutcome="hand")
plot(lexample.rw)

data(numbers)
traceCues=c( "exactly1", "exactly2", "exactly3", "exactly4",
   "exactly5", "exactly6", "exactly7", "exactly10", "exactly15")
traceOutcomes=c("1", "2", "3", "4", "5", "6", "7", "10", "15")
ylimit=c(0,1)
```

```
par(mfrow=c(3,3),mar=c(4,4,1,1))

for(i in 1:length(traceCues)) {
   numbers.rw <- RescorlaWagner(numbers, nruns=1,
      traceCue=traceCues[i], traceOutcome=traceOutcomes[i])
    plot(numbers.rw, ylimit=ylimit)
    mtext(paste(traceCues[i], " - ", traceOutcomes[i], sep=""),
       side=3, line=-1, cex=0.7)
  }
par(mfrow=c(1,1))
```

`serbian` *Serbian case inflected nouns.*

Description

3240 case-inflected Serbian nouns and their frequencies, for 270 different masculine, feminine and neuter noun lemmas.

Usage

```
data(serbian)
```

Format

A data frame with 3240 observations on the following 3 variables:

`WordForm` A character vector specifying the inflected word forms.

`LemmaCase` A character vector specifying lemma (meaning), case, and number.

`Frequency` A numeric vector specifying the frequency of each word form.

Details

Frequencies were taken from the Frequency Dictionary of Contemporary Serbian Language (Kostic, 1999). The 270 lemmas comprise the set of nouns for which each different case form appears at least once in this resource.

Source

Kostic, D. (1999). Frekvencijski recnik savremenog srpskog jezika (Frequency Dictionary of Contemporary Serbian Language). Institute for Experimental Phonetics and Speech Pathology & Laboratory of Experimental Psychology, University of Belgrade, Serbia.

References

Kostic, D. (1999). Frekvencijski recnik savremenog srpskog jezika (Frequency Dictionary of Contemporary Serbian Language). Institute for Experimental Phonetics and Speech Pathology & Laboratory of Experimental Psychology, University of Belgrade, Serbia.

Baayen, R. H., Milin, P., Filipovic Durdevic, D., Hendrix, P. and Marelli, M. (2011), An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

See Also

See also `serbianLex`, `estimateActivations`.

Examples

```
data(serbian)
serbian$Cues <- orthoCoding(serbian$WordForm, grams=2)
serbian$Outcomes <- serbian$LemmaCase
sw <- estimateWeights(cuesOutcomes=serbian)
sw[1:5,1:5]
desiredItems <- unique(serbian["Cues"])
desiredItems$Outcomes=""
activations <- estimateActivations(desiredItems, sw)$activationMatrix
rownames(activations) <- unique(serbian[["WordForm"]])
activations[1:5,1:6]
```

---

serbianLex *Serbian lexicon with 1187 prime-target pairs.*

Description

The 1187 prime-target pairs and their lexical properties used in the simulation study of Experiment 1 of Baayen et al. (2011).

Usage

```
data(serbianLex)
```

Format

A data frame with 1187 observations on the following 14 variables:

`Target` A factor specifying the target noun form

`Prime` A factor specifying the prime noun form

`PrimeLemma` A factor specifying the lemma of the prime

`TargetLemma` A factor specifying the target lemma

`Length` A numeric vector with the length in letters of the target

`WeightedRE` A numeric vector with the weighted relative entropy of the prime and target inflectional paradigms

`NormLevenshteinDist` A numeric vector with the normalized Levenshtein distance of prime and target forms

`TargetLemmaFreq` A numeric vector with log frequency of the target lemma

`PrimeSurfFreq` A numeric vector with log frequency of the prime form

`PrimeCondition` A factor with prime conditions, levels: `DD`, `DSSD`, `SS`

`CosineSim` A numeric vector with the cosine similarity of prime and target vector space semantics

`IsMasc` A vector of logicals, `TRUE` if the noun is masculine.

`TargetGender` A factor with the gender of the target, levels: `f`, `m`, and `n`

`TargetCase` A factor specifying the case of the target noun, levels: `acc`, `dat`, `nom`

`MeanLogObsRT` Mean log-transformed observed reaction time

References

Baayen, R. H., Milin, P., Filipovic Durdevic, D., Hendrix, P. and Marelli, M. (2011), An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

Examples

```
# calculate the weight matrix for the full set of Serbian nouns
data(serbian)
serbian$Cues <- orthoCoding(serbian$WordForm, grams=2)
serbian$Outcomes <- serbian$LemmaCase
sw <- estimateWeights(cuesOutcomes=serbian)

# calculate the meaning activations for all unique word forms

desiredItems <- unique(serbian["Cues"])
desiredItems$Outcomes <- ""
activations <- estimateActivations(desiredItems, sw)$activationMatrix
rownames(activations) <- unique(serbian[["WordForm"]])
activations <- activations + abs(min(activations))
activations[1:5,1:6]

# calculate simulated latencies for the experimental materials

data(serbianLex)
syntax <- c("acc", "dat", "gen", "ins", "loc", "nom", "Pl", "Sg")
we <- 0.4 # compound cue weight
strengths <- rep(0, nrow(serbianLex))
for(i in 1:nrow(serbianLex)) {
   target <- serbianLex$Target[i]
   prime <- serbianLex$Prime[i]
   targetLemma <- as.character(serbianLex$TargetLemma[i])
   primeLemma <- as.character(serbianLex$PrimeLemma[i])
   targetOutcomes <- c(targetLemma, primeLemma, syntax)
   primeOutcomes <- c(targetLemma, primeLemma, syntax)
```

```
   p <- activations[target, targetOutcomes]
   q <- activations[prime, primeOutcomes]
   strengths[i] <- sum((q^we)*(p^(1-we)))
}
serbianLex$SimRT <- -strengths
lengthPenalty <- 0.3
serbianLex$SimRT2 <- serbianLex$SimRT +
  (lengthPenalty * (serbianLex$Length>5))

cor.test(serbianLex$SimRT, serbianLex$MeanLogObsRT)
cor.test(serbianLex$SimRT2, serbianLex$MeanLogObsRT)

serbianLex.lm <- lm(SimRT2 ~ Length +  WeightedRE*IsMasc +
      NormLevenshteinDist + TargetLemmaFreq +
      PrimeSurfFreq + PrimeCondition, data=serbianLex)
summary(serbianLex.lm)
```

`serbianUniCyr` *Serbian case inflected nouns (in Cyrillic Unicode).*

Description

3240 case-inflected Serbian nouns and their frequencies, for 270 different masculine, feminine and neuter noun lemmas, written using the Cyrillic alphabet and encoded in UTF-8.

Usage

```
data(serbianUniCyr)
```

Format

A data frame with 3240 observations on the following 3 variables:

`WordForm` A character vector specifying the inflected word forms encoded in UTF-8.

`LemmaCase` A character vector specifying lemma (meaning), case, and number.

`Frequency` A numeric vector specifying the frequency of each word form.

Details

Frequencies were taken from the Frequency Dictionary of Contemporary Serbian Language (Kostic, 1999). The 270 lemmas comprise the set of nouns for which each different case form appears at least once in this resource.

Source

Kostic, D. (1999). Frekvencijski recnik savremenog srpskog jezika (Frequency Dictionary of Contemporary Serbian Language). Institute for Experimental Phonetics and Speech Pathology & Laboratory of Experimental Psychology, University of Belgrade, Serbia.

References

Kostic, D. (1999). Frekvencijski recnik savremenog srpskog jezika (Frequency Dictionary of Contemporary Serbian Language). Institute for Experimental Phonetics and Speech Pathology & Laboratory of Experimental Psychology, University of Belgrade, Serbia.

Baayen, R. H., Milin, P., Filipovic Durdevic, D., Hendrix, P. and Marelli, M. (2011), An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

See Also

See also `serbian`, `serbianLex`, `estimateActivations`.

Examples

```
## Not run:
data(serbianUniCyr)
serbianUniCyr$Cues <- orthoCoding(serbianUniCyr$WordForm, grams=2)
serbianUniCyr$Outcomes <- serbianUniCyr$LemmaCase
sw <- estimateWeights(cuesOutcomes=serbianUniCyr)
sw[1:5,1:5]
desiredItems <- unique(serbianUniCyr["Cues"])
desiredItems$Outcomes=""
activations <- estimateActivations(desiredItems, sw)$activationMatrix
rownames(activations) <- unique(serbianUniCyr[["WordForm"]])
activations[1:5,1:6]

## End(Not run)
```

---

serbianUniLat *Serbian case inflected nouns (in Latin-alphabet Unicode).*

Description

3240 case-inflected Serbian nouns and their frequencies, for 270 different masculine, feminine and neuter noun lemmas, written using the Latin alphabet and encoded in UTF-8.

Usage

```
data(serbianUniLat)
```

Format

A data frame with 3240 observations on the following 3 variables:

`WordForm` A character vector specifying the inflected word forms encoded in UTF-8.

`LemmaCase` A character vector specifying lemma (meaning), case, and number.

`Frequency` A numeric vector specifying the frequency of each word form.

Details

Frequencies were taken from the Frequency Dictionary of Contemporary Serbian Language (Kostic, 1999). The 270 lemmas comprise the set of nouns for which each different case form appears at least once in this resource.

Source

Kostic, D. (1999). Frekvencijski recnik savremenog srpskog jezika (Frequency Dictionary of Contemporary Serbian Language). Institute for Experimental Phonetics and Speech Pathology & Laboratory of Experimental Psychology, University of Belgrade, Serbia.

References

Kostic, D. (1999). Frekvencijski recnik savremenog srpskog jezika (Frequency Dictionary of Contemporary Serbian Language). Institute for Experimental Phonetics and Speech Pathology & Laboratory of Experimental Psychology, University of Belgrade, Serbia.

Baayen, R. H., Milin, P., Filipovic Durdevic, D., Hendrix, P. and Marelli, M. (2011), An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. Psychological Review, 118, 438-482.

See Also

See also `serbian`, `serbianLex`, `estimateActivations`.

Examples

```
data(serbianUniLat)
serbianUniLat$Cues <- orthoCoding(serbianUniLat$WordForm, grams=2)
serbianUniLat$Outcomes <- serbianUniLat$LemmaCase
sw <- estimateWeights(cuesOutcomes=serbianUniLat)
sw[1:5,1:5]
desiredItems <- unique(serbianUniLat["Cues"])
desiredItems$Outcomes=""
activations <- estimateActivations(desiredItems, sw)$activationMatrix
rownames(activations) <- unique(serbianUniLat[["WordForm"]])
activations[1:5,1:6]
```

---

summary.ndlClassify *A summary of a Naive Discriminatory Learning Model*

Description

A summarization method for an object of the class `"ndlClassify"`.

Usage

```
## S3 method for class 'ndlClassify'
summary(object, ...)

## S3 method for class 'summary.ndlClassify'
print(x, digits = max(3, getOption("digits") - 3), max.print=10, ...)
```

Arguments

object	An object of class "ndlClassify", resulting from a call to ndlClassify.
x	An object of class "summary.ndlClassify", usually resulting from a call to summary.ndlClassify.
digits	The number of significant digits to use when printing.
max.print	The maximum number of rows of weights to be output when printing; by default equal to 10; ; if set to NA all rows will be output.
...	Control arguments passed to or from other methods, e.g. ndlStatistics and modelStatistics.

Details

Calculates descriptive statistics of a fitted Naive Discriminatory Learning model and prints a nice summary of the key results.

Value

summary.ndlClassify returns an object of the class "summary.ndlClassify", a list with the following components:

call The call matched to fit the "ndlClassify" object.

formula The formula specified for the "ndlClassify" object.

weights The estimated weights.

statistics A range of descriptive statistics calculated with ndlStatistics.

Author(s)

Antti Arppe

References

Arppe, A. and Baayen, R. H. (in prep.)

See Also

ndlClassify, ndlStatistics, modelStatistics

Examples

```
## For examples see examples(ndlClassify).
```

summary.ndlCrossvalidate *A summary of a crossvalidation of a Naive Discriminatory Reader Model*

Description

A summarization method for an object of the class "ndlCrossvalidate".

Usage

```
## S3 method for class 'ndlCrossvalidate'
summary(object, ...)

## S3 method for class 'summary.ndlCrossvalidate'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object	An object of class "ndlCrossvalidate", resulting from a call to ndlCrossvalidate.
x	An object of class "summary.ndlCrossvalidate", usually resulting from a call to summary.ndlCrossvalidate.
digits	the number of significant digits to use when printing.
...	further arguments passed to or from other methods.

Details

Calculates overall descriptive statistics of the crossvalidation of a fitted Naive Discriminatory Reader model and prints a nice summary of the key results.

Value

summary.ndlCrossvalidate returns an object of the class "summary.ndlCrossvalidate", a list with the following components:

call The call matched to fit the "ndlCrossvalidate" object.

formula The formula specified for the "ndlCrossvalidate" object.

statistics.summary The means, minima and maxima of a range descriptive statistics for the fit and performance of individual folds; see ndlStatistics.

crosstable.summary The means of the crosstabulation of observed and predicted outcomes for the held-out test data.

recall.predicted.summary The means of the recall values for the individual outcomes predicted with the held-out test data.

precision.predicted.summary The means of the precision values for the individual outcomes predicted with the held-out test data.

`statistics.all` All the values for a range descriptive statistics for the fit and performance of individual folds on the held-out test data; see `ndlStatistics`.

`k` The number of folds.

`n.total` The sum frequency of all data points in `data`.

`n.train` The sum frequency of data points used for training the individual models (excluding the individual folds).

`n.test` The sum frequency of data points in the individual held-out folds used for testing the individual models.

Author(s)

Antti Arppe

References

Arppe, A. and Baayen, R. H. (in prep.)

See Also

`ndlCrossvalidate`, `ndlClassify`, `ndlStatistics`

Examples

```
## For examples see examples(ndlCrossvalidate).
```

`think` *Finnish 'think' verbs.*

Description

3404 occurrences of four synonymous Finnish 'think' verbs ('ajatella': 1492; 'mietti\"a': 812; 'pohtia': 713; 'harkita': 387) in newspaper and Internet newsgroup discussion texts

Usage

```
data(think)
```

Format

A data frame with 3404 observations on the following 27 variables:

`Lexeme` A factor specifying one of the four 'think' verb synonyms

`Polarity` A factor specifying whether the 'think' verb has negative polarity (`Negation`) or not (`Other`)

`Voice` A factor specifying whether the 'think' verb is in the `Passive` voice or not (`Other`)

Mood A factor specifying whether the 'think' verb is in the Indicative or Conditional mood or not (Other)

Person A factor specifying whether the 'think' verb is in the First, Second, Third person or not (None)

Number A factor specifying whether the 'think' verb is in the Plural number or not (Other)

Covert A factor specifying whether the agent/subject of the 'think' verb is explicitly expressed as a syntactic argument (Overt), or only as a morphological feature of the 'think' verb (Covert)

ClauseEquivalent A factor specifying whether the 'think' verb is used as a non-finite clause equivalent (ClauseEquivalent) or as a finite verb (FiniteVerbChain)

Agent A factor specifying the occurrence of Agent/Subject of the 'think' verb as either a Human Individual, Human Group, or as absent (None)

Patient A factor specifying the occurrence of the Patient/Object argument among the semantic or structural subclasses as either an Human Individual or Group (IndividualGroup), Abstraction, Activity, Communication, Event, an 'etta' ('that') clause (etta_CLAUSE), DirectQuote, IndirectQuestion, Infinitive, Participle, or as absent (None)

Manner A factor specifying the occurrrence of the Manner argument as any of its subclasses Generic, Negative (sufficiency), Positive (sufficiency), Frame, Agreement (Concur or Disagree), Joint (Alone or Together), or as absent (None)

Time A factor specifying the occurrence of Time argument (as a moment) as either of its subclasses Definite, Indefinite, or as absent (None)

Modality1 A factor specifying the main semantic subclasses of the entire Verb chain as either indicating Possibility, Necessity, or their absense (None)

Modality2 A factor specifying minor semantic subclasses of the entire Verb chain as indicating either a Temporal element (begin, end, continuation, etc.), External (cause), Volition, Accidental nature of the thinking process, or their absense (None)

Source A factor specifying the occurrence of a Source argument or its absense (None)

Goal A factor specifying the occurrence of a Goal argument or its absence (None)

Quantity A factor specifying the occurrence of a Quantity argument, or its absence (None)

Location A factor specifying the occurrence of a Location argument, or its absence (None)

Duration A factor specifying the occurrence of a Duration argument, or its absence (None)

Frequency A factor specifying the occurrence of a Frequency arument, or its absence (None)

MetaComment A factor specifying the occurrence of a MetaComment, or its absence (None)

ReasonPurpose A factor specifying the occurrence of a Reason or Purpose argument (ReasonPurpose), or their absence (None)

Condition A factor specifying the occurrence of a Condition argument, or its absence (None)

CoordinatedVerb A factor specifying the occurrence of a Coordinated Verb (in relation to the 'think' verb: CoordinatedVerb), or its absence (None)

Register A factor specifying whether the 'think' verb occurs in the newspaper subcorpus (hs95) or the Internet newsgroup discussion corpus (sfnet)

Section A factor specifying the subsection in which the 'think' verb occurs in either of the two subcorpora

Author A factor specifying the author of the text in which the 'think' verb occurs, if that author is identifiable – authors in the Internet newgroup discussion subcorpus are anonymized; unidentifiable/unknown author designated as (None)

Details

The four most frequent synonyms meaning 'think, reflect, ponder, consider', i.e. 'ajatella, miettia, pohtia, harkita', were extracted from two months of newspaper text from the 1990s (Helsingin Sanomat 1995) and six months of Internet newsgroup discussion from the early 2000s (SFNET 2002-2003), namely regarding (personal) relationships (sfnet.keskustelu.ihmissuhteet) and politics (sfnet.keskustelu.politiikka). The newspaper corpus consisted of 3,304,512 words of body text (i.e. excluding headers and captions as well as punctuation tokens), and included 1,750 examples of the studied 'think' verbs. The Internet corpus comprised 1,174,693 words of body text, yielding 1,654 instances of the selected 'think' verbs. In terms of distinct identifiable authors, the newspaper sub-corpus was the product of just over 500 journalists and other contributors, while the Internet sub-corpus involved well over 1000 discussants. The `think` dataset contains a selection of 26 contextual features judged as most informative.

For extensive details of the data and its linguistic and statistical analysis, see Arppe (2008). For the full selection of contextual features, see the `amph` (2008) microcorpus.

Source

amph 2008. A micro-corpus of 3404 occurrences of the four most common Finnish THINK lexemes, 'ajatella, miettia, pohtia, and harkita', in Finnish newspaper and Internet newsgroup discussion texts, containing extracts and linguistic analysis of the relevant context in the original corpus data, scripts for processing this data, R functions for its statistical analysis, as well as a comprehensive set of ensuing results as R data tables. Compiled and analyzed by Antti Arppe. Available on-line at URL: http://www.csc.fi/english/research/software/amph/

Helsingin Sanomat 1995. ~22 million words of Finnish newspaper articles published in Helsingin Sanomat during January–December 1995. Compiled by the Research Institute for the Languages of Finland [KOTUS] and CSC – IT Center for Science, Finland. Available on-line at URL: http://www.csc.fi/kielipankki/

SFNET 2002-2003. ~100 million words of Finnish internet newsgroup discussion posted during October 2002 – April 2003. Compiled by Tuuli Tuominen and Panu Kalliokoski, Computing Centre, University of Helsinki, and Antti Arppe, Department of General Linguistics, University of Helsinki, and CSC – IT Center for Science, Finland. Available on-line at URL: http://www.csc.fi/kielipankki/

References

Arppe, A. 2008. Univariate, bivariate and multivariate methods in corpus-based lexicography – a study of synonymy. Publications of the Department of General Linguistics, University of Helsinki, No. 44. URN: http://urn.fi/URN:ISBN:978-952-10-5175-3.

Arppe, A. 2009. Linguistic choices vs. probabilities – how much and what can linguistic theory explain? In: Featherston, Sam & Winkler, Susanne (eds.) The Fruits of Empirical Linguistics. Volume 1: Process. Berlin: de Gruyter, pp. 1-24.

Examples

```
## Not run:
data(think)
think.ndl = ndlClassify(Lexeme ~ Person + Number + Agent + Patient + Register,
   data=think)
summary(think.ndl)
```

```
plot(think.ndl)

## End(Not run)
```

Index