# Package 'politeness'

October 12, 2019

**Type** Package

**Title** Detecting Politeness Features in Text

**Version** 0.4.1

**Author** Mike Yeomans, Alejandro Kantor, Dustin Tingley

**Maintainer** Mike Yeomans <myeomans@hbs.edu>

**Description** Detecting markers of politeness in English natural language. This package allows researchers to easily visualize and quantify politeness between groups of documents. This package combines prior research on the linguistic markers of politeness (Brown & Levinson, 1987 <http://psycnet.apa.org/record/1987-97641-000>; Danescu-Niculescu-Mizil et al., 2013 <arXiv:1306.6078>; Voigt et al., 2017 <doi:10.1073/pnas.1702413114>). We thank the Spencer Foundation, the Hewlett Foundation, and Harvard's Institute for Quantitative Social Science for support.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 2.10)

**Imports** tm, quanteda, ggplot2, parallel, spacyr, textir, glmnet, data.table, stringr

**RoxygenNote** 6.1.1

**Suggests** knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2019-10-12 07:00:08 UTC

## R topics documented:

---

`bowl_offers` *Purchase offers for bowl*

---

## Description

A dataset containing the purchase offer message and a label indicating if the writer was assigned to be warm (1) or tough (0)

## Usage

```
bowl_offers
```

## Format

A data frame with 70 rows and 2 variables:

**message** character of purchase offer message

**condition** binary label indicating if message is warm or tough

## Source

Jeong, M., Minson, J., Yeomans, M. & Gino, F. (2019).

"Communicating Warmth in Distributed Negotiations is Surprisingly Ineffective." Study 3.

Study 3. https://osf.io/t7sd6/

## feature_table *Table of Politeness Features*

### Description

This table describes all the text features extracted in this package. See vignette for details.

### Usage

```
feature_table
```

### Format

A data.frame with information about 36 politeness features.

## findPoliteTexts *Find polite text*

### Description

Finds examples of most or least polite text in a corpus

### Usage

```
findPoliteTexts(text, df_polite, covar, type = c("most", "least",
  "both"), num_docs = 5L, ...)
```

### Arguments

| | |
|---|---|
| text | a character vector of texts. |
| df_polite | a data.frame with politeness features, as outputed by politeness, used to train model. |
| covar | a vector of politeness labels, or other covariate. |
| type | a string indicating if function should return the most or least polite texts or both. If length > 1 only first value is used. |
| num_docs | integer of number of documents to be returned. Default is 5. |
| ... | additional parameters to be passed to politenessProjection. |

### Details

Function returns a data.frame ranked by (more or least) politeness. If type == 'most', the num_docs most polite texts will be returned. If type == 'least', the num_docs least polite texts will be returned. If type == 'both', both most and least polite text will be returned. if num_docs is even, half will be most and half least polite else half + 1 will be most polite.

df_polite must have the same number of rows as the length(text) and length(covar).

## Value

data.frame with texts ranked by (more or least) politeness. See details for more information.

## Examples

```
data("phone_offers")
polite.data<-politeness(phone_offers$message, parser="none",drop_blank=FALSE)

findPoliteTexts(phone_offers$message,
                polite.data,
                phone_offers$condition,
                type = "most",
                num_docs = 5)

findPoliteTexts(phone_offers$message,
                polite.data,
                phone_offers$condition,
                type = "least",
                num_docs = 10)
```

---

# hedge_list *Hedge Words List*

---

## Description

Hedges

## Usage

```
hedge_list
```

## Format

A list of 99 hedging words.

---

# negative_list *Negative Emotions List*

---

## Description

Negative words.

## Usage

```
negative_list
```

### Format

A list of 4783 negatively-valenced words

---

## phone_offers *Purchase offers for phone*

---

### Description

A dataset containing the purchase offer message and a label indicating if the writer was assigned to be warm (1) or tough (0)

### Usage

```
phone_offers
```

### Format

A data frame with 355 rows and 2 variables:

**message** character of purchase offer message

**condition** binary label indicating if message is warm or tough

### Source

Jeong, M., Minson, J., Yeomans, M. & Gino, F. (2019).

"Communicating Warmth in Distributed Negotiations is Surprisingly Ineffective."

Study 1. https://osf.io/t7sd6/

---

## politeness *Politeness Features*

---

### Description

Detects linguistic markers of politeness in natural language. This function is the workhorse of the `politeness` package, taking an N-length vector of text documents and returning an N-row data.frame of feature counts.

### Usage

```
politeness(text, parser = c("none", "spacy"), metric = c("count",
  "binary", "average"), drop_blank = TRUE, num_mc_cores = 1,
  binary = FALSE)
```

## Arguments

| | |
|---|---|
| text | character A vector of texts, each of which will be tallied for politeness features. |
| parser | character Name of dependency parser to use (see details). Without a dependency parser, some features will be approximated, while others cannot be calculated at all. |
| metric | character What metric to return? Raw feature count totals, Binary presence/absence of features, or feature counts per word Default is "count". |
| drop_blank | logical Should features that were not found in any text be removed from the data.frame? Default is TRUE |
| num_mc_cores | integer Number of cores for parallelization. Default is 1, but we encourage users to try parallel::detectCores() if possible. |
| binary | logical Deprecated! Do not use (see metric parameter). |

## Details

Some politeness features depend on part-of-speech tagged sentences (e.g. "bare commands" are a particular verb class). To include these features in the analysis, a POS tagger must be initialized beforehand - we currently support SpaCy which must be installed separately in Python (see example for implementation).

## Value

a data.frame of politeness features, with one row for every item in 'text'. Possible politeness features are listed in feature_table

## References

Brown, P., & Levinson, S. C. (1987). Politeness: Some universals in language usage (Vol. 4). Cambridge university press.

Danescu-Niculescu-Mizil, C., Sudhof, M., Jurafsky, D., Leskovec, J., & Potts, C. (2013). A computational approach to politeness with application to social factors. arXiv preprint arXiv:1306.6078.

Voigt, R., Camp, N. P., Prabhakaran, V., Hamilton, W. L., ... & Eberhardt, J. L. (2017). Language from police body camera footage shows racial disparities in officer respect. Proceedings of the National Academy of Sciences, 201702413.

## Examples

```
data("phone_offers")

politeness(phone_offers$message, parser="none",drop_blank=FALSE)

colMeans(politeness(phone_offers$message, parser="none", metric="binary", drop_blank=FALSE))
colMeans(politeness(phone_offers$message, parser="none", metric="count", drop_blank=FALSE))

dim(politeness(phone_offers$message, parser="none",drop_blank=FALSE))
dim(politeness(phone_offers$message, parser="none",drop_blank=TRUE))
```

```
## Not run:
# Detect multiple cores automatically for parallel processing
politeness(phone_offers$message, num_mc_cores=parallel::detectCores())

# Connect to SpaCy installation for part-of-speech features
install.packages("spacyr")
spacyr::spacy_initialize(python_executable = PYTHON_PATH)
politeness(phone_offers$message, parser="spacy",drop_blank=FALSE)


## End(Not run)
```

---

`politenessPlot` *Politeness plot*

---

## Description

Plots the prevalence of politeness features in documents, divided by a binary covariate.

## Usage

```
politenessPlot(df_polite, split = NULL, split_levels = NULL,
  split_name = NULL, split_cols = c("firebrick", "navy"),
  top_title = "", drop_blank = 0.05, middle_out = 0.5)
```

## Arguments

| | |
|---|---|
| `df_polite` | a data.frame with politeness features calculated from a document set, as output by `politeness`. |
| `split` | a vector of covariate values. must have a length equal to the number of documents included in `df_polite`. No NA values allowed. |
| `split_levels` | character vector of length 2 default NULL. Labels for covariate levels for legend. If NULL, this will be inferred from `split`. |
| `split_name` | character default NULL. Name of the covariate for legend. |
| `split_cols` | character vector of length 2. Name of colors to use. |
| `top_title` | character default "". Title of plot. |
| `drop_blank` | Features less prevalent than this in the sample value are excluded from the plot. To include all features, set to `0` |
| `middle_out` | Features less distinctive than this value (measured by p-value of t-test) are excluded. Defaults to 1 (i.e. include all). |

## Details

Length of `split` must be the same as number of rows of `df_polite`. Typically `split` should be a two-category variable. However, if a continuous covariate is given, then the top and bottom terciles of that distribution are treated as the two categories (while dropping data from the middle tercile).

## Value

a ggplot of the prevalence of politeness features, conditional on `split`. Features are sorted by variance-weighted log odds ratio.

## Examples

```
data("phone_offers")

polite.data<-politeness(phone_offers$message, parser="none", binary=FALSE, drop_blank=FALSE)

politeness::politenessPlot(polite.data,
                           split=phone_offers$condition,
                           split_levels = c("Tough","Warm"),
                           split_name = "Condition",
                           top_title = "Average Feature Counts")


polite.data<-politeness(phone_offers$message, parser="none", binary=TRUE, drop_blank=FALSE)

politeness::politenessPlot(polite.data,
                           split=phone_offers$condition,
                           split_levels = c("Tough","Warm"),
                           split_name = "Condition",
                           top_title = "Binary Feature Use")
```

---

`politenessProjection` *Politeness projection*

---

## Description

Training and projecting a regression model of politeness.

## Usage

```
politenessProjection(df_polite_train, covar = NULL,
  df_polite_test = NULL, classifier = c("glmnet", "mnir"), ...)
```

## Arguments

| | |
|---|---|
| df_polite_train | a data.frame with politeness features as outputed by politeness used to train model. |
| covar | a vector of politeness labels, or other covariate. |
| df_polite_test | optional data.frame with politeness features as outputed by politeness used for out-of-sample fitting. Must have same feature set as polite_train (most easily acheived by setting dropblank=FALSE in both calls to politeness). |
| classifier | name of classification algorithm. Defaults to "glmnet" (see glmnet) but "mnir" (see mnlm) is also available. |
| ... | additional parameters to be passed to the classification algorithm. |

## Details

List:

- train_proj projection of politeness model within training set.
- test_proj projection of politeness model onto test set (i.e. out-of-sample).
- train_coef coefficients from the trained model.

## Value

List of df_polite_train and df_polite_test with projection. See details.

## Examples

```
data("phone_offers")
data("bowl_offers")

polite.data<-politeness(phone_offers$message, parser="none",drop_blank=FALSE)

polite.holdout<-politeness(bowl_offers$message, parser="none",drop_blank=FALSE)

project<-politenessProjection(polite.data,
                              phone_offers$condition,
                              polite.holdout)

# Difference in average politeness across conditions in the new sample.

mean(project$test_proj[bowl_offers$condition==1])
mean(project$test_proj[bowl_offers$condition==0])
```

## polite_dicts *Feature Dictionaries*

### Description

Six dictionary-like features for the detector: Negations; Pauses; Swearing; Pronouns; Formal Titles; and Informal Titles.

### Usage

```
polite_dicts
```

### Format

A list of six `quanteda::dictionary` objects

## positive_list *Positive Emotions List*

### Description

Positive words.

### Usage

```
positive_list
```

### Format

A list of 2006 positively-valenced words

## receptiveness *Conversational Receptiveness*

### Description

Pre-trained model to detect conversational receptiveness

### Usage

```
receptiveness(texts, num_mc_cores = 1)
```

### Arguments

| | |
|---|---|
| texts | character A vector of texts, each of which will be tallied for politeness features. |
| num_mc_cores | integer Number of cores for parallelization. |

### Details

This is a wrapper around a pre-trained model of "conversational receptiveness". The model trained from Study 1 of that paper can be applied to new text with a single function. This model requires grammar parsing via SpaCy. Please see spacyr for details on installation.

### Value

a vector with receptiveness scores

### References

Yeomans, M., Minson, J., Collins, H., Chen, F. & Gino, F. (2019). Conversational Receptiveness: Improving Engagement with Opposing Views.

### Examples

```
## Not run:
data("phone_offers")

receptiveness(phone_offers$message)


## End(Not run)
```

---

## receptTrain *Receptivness Training Data*

---

### Description

A dataset containing responses to political arguments and an average rating of their receptiveness to opposing views.

### Usage

```
receptTrain
```

### Format

A data frame with 543 rows and 2 variables:

**message** character of response

**condition** numeric average rating of writer's receptiveness

## Source

Yeomans, M., Minson, J., Collins, H., Chen, F. & Gino, F. (working paper).
"Conversational Receptiveness: Improving Engagement with Opposing Views"
Study 1. https://osf.io/2n59b/

# Index