# Package 'cleanNLP'

November 11, 2016

**Type** Package

**Title** A Tidy Data Model for Natural Language Processing

**Version** 0.24

**Date** 2016-11-10

**Author** Taylor B. Arnold

**Maintainer** Taylor B. Arnold <taylor.arnold@acm.org>

**Description** Provides a set of fast tools for converting a textual corpus into a set of normalized tables. The underlying natural language processing pipeline utilizes the Stanford's CoreNLP library. Exposed annotation tasks include tokenization, part of speech tagging, named entity recognition, entity linking, sentiment analysis, dependency parsing, coreference resolution, and information extraction. Several datasets containing token unigram, part of speech tag, and dependency type frequencies are also included to assist with analyses. Currently supports parsing text in English, French, German, and Spanish.

**Depends** dplyr, magrittr, R (>= 2.10)

**Imports** readr, rJava, RCurl

**SystemRequirements** Java (>= 7.0); Stanford CoreNLP <http://nlp.stanford.edu/ software/corenlp.shtml> (>= 3.5.2)

**License** GPL-3

**LazyData** true

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2016-11-11 12:09:27

## R topics documented:

---

`annotate` *Run the annotation pipeline on a set of documents*

---

## Description

Runs the clean_nlp annotators over a given corpus of text. The details for which annotators to run and how to run them are specified by using one of: `set_language` or `set_properties` (the former being the most user-friendly).

## Usage

```
annotate(input, file = NULL, output_dir = NULL, load = TRUE,
  keep = TRUE, as_strings = FALSE, doc_id_offset = 0L)
```

## Arguments

| | |
|---|---|
| `input` | either a vector of file names to parse, or a character vector with one document in each element. Specify the latter with the as_string flag. |
| `file` | character. Location to store a compressed R object containing the results. If NULL, the default, no such compressed object will be stored. |
| `output_dir` | path to the directory where the raw output should be stored. Will be created if it does not exist. Files currently in this location will be overwritten. If NULL, the default, it uses a temporary directory. Not to be confused with `file`, this location stores the raw csv files rather than a compressed dataset. |
| `load` | logical. Once parsed, should the data be read into R as an annotation object? |
| `keep` | logical. Once parsed, should the files be kept on disk in `output_dir`? |

| | |
|---|---|
| as_strings | logical. Is the data given to input the actual document text rather than file names? |
| doc_id_offset | integer. The first document id to use. Defaults to 0. |

## Value

if load is true, an object of class annotation. Otherwise, a character vector giving the output location of the files.

## Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

## References

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.*

## Examples

```
## Not run:
annotation <- annotate("path/to/corpus/directory")

## End(Not run)
```

---

cleanNLP *cleanNLP: A Tidy Data Model for Natural Language Processing*

---

## Description

Provides a set of fast tools for converting a textual corpus into a set of normalized tables. The underlying natural language processing pipeline utilizes the Stanford's CoreNLP library.

## Details

To quickly start using the **cleanNLP** package, first download the CoreNLP jar files using the function download_clean_nlp. Properties can then be set to change the behavior of the pipeline using the function set_language. There is also set_properties, which provide lower-level mechanisms for setting pipeline properties. The output of all of these functions are cached between R sessions and only need to be run once.

Once the package is set up, run init_clean_nlp to start the java engine and to load the CoreNLP pipeline. The latter may take up to several minutes depending on how which annotators are being used. After this function is done loading, use annotate to run the annotation engine over a corpus of text. Functions are then available to extract data tables from the annotation object:

get_token, get_dependency, get_document, get_coreference, get_entity, get_sentiment, and get_triple. See their documentation for further details.

If loading annotation that have previously been saved to disk, these can be pulled back into R using read_annotation. This does not require downloading the java jar files or initializing the annotation pipeline.

## Examples

```
## Not run:
# download files and set properties of the annotation engine (only need to do this once)
download_clean_nlp()
set_properties()
set_language("en", speed = 2)

# load the annotation engine (only do once per session)
init_clean_nlp()

# annotate your text
annotation <- annotate("path/to/corpus/directory")

# pull off data tables
token <- get_token(annotation)
dependency <- get_dependency(annotation)
document <- get_document(annotation)
coreference <- get_coreference(annotation)
entity <- get_entity(annotation)
sentiment <- get_sentiment(annotation)
triple <- get_triple(annotation)

## End(Not run)
```

---

# combine_annotators *Combine a set of annotations*

---

## Description

Takes an arbitrary set of annotations and efficiently combines them into a single object. All document ids are reset so that they are contiguous integers starting at zero.

## Usage

```
combine_annotators(...)
```

## Arguments

| | |
|---|---|
| ... | annotation objects to combine; either a single list item or all of the objects as individual inputs |

### Value

a single annotation object containing all of the input documents

### Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

### Examples

```
## Not run:
annotation <- combine_annotators(anno01, anno02, anno03)

## End(Not run)
```

---

## dep_frequency *Universal Dependency Frequencies*

---

### Description

The language-specific frequency of universal dependency codes as given in standard treebank corpora. Frequencies are multiplied by 100.

### References

http://universaldependencies.org/

---

## doc_id_reset *Reset document ids*

---

### Description

Given an annotation object, this function changes all of the document ids so that they are all contiguous integers, starting at the parameter `start_id`.

### Usage

```
doc_id_reset(annotation, start_id = 0L)
```

### Arguments

| | |
|---|---|
| `annotation` | annotation object to reset the IDs of |
| `start_id` | the starting document id. Defaults to 0. |

## Value

an annotation object with document ids updated across all tables.

## Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

## Examples

```
## Not run:
doc_id_reset(annotation, 10)

## End(Not run)
```

---

`download_clean_nlp` *Download java files needed for cleanNLP*

---

## Description

The cleanNLP package does not supply the raw java files provided by the Stanford NLP Group as they are quite large. This function downloads the libraries automatically, by default into the directory where the package was installed.

## Usage

```
download_clean_nlp(type = c("default", "base", "en", "fr", "de", "es"),
  output_loc, url = NULL, url_core = TRUE)
```

## Arguments

| | |
|---|---|
| `type` | type of files to download. The base backage is always required. Other jars include model files for French, German, and Spanish. These can be installed in addition to the base package. By default, the function downloads the base package and English model files. |
| `output_loc` | a string showing where the files are to be downloaded. If missing, will try to download files into the directory where the package was original installed. |
| `url` | the url to try to download components from. Setting to NULL uses the default location on the Stanford NLP server, but you can set this manually by using this option. It also allows for local files, but note that the path must include the prefix `file://`. For details, see `download.file`. |
| `url_core` | if `url` is not null, this flag indicates whehter the path given to url points to the core nlp files (which are zipped) or to model files (which are unzipped). |

## Examples

```
## Not run:
download_clean_nlp()
download_clean_nlp(type="spanish")

## End(Not run)
```

---

`get_coreference` *Access coreferences from an annotation object*

---

## Description

Coreferences are collections of expressions that all represent the same person, entity, or thing. For example, the text "Lauren loves dogs. She would walk them all day.", there is a coreference consisting of the token "Lauren" in the first sentence and the token "She" in the second sentence. In the output given from this function, a row is given for any mention of an entity; these can be linked using the `rid` key.

## Usage

```
get_coreference(annotation)
```

## Arguments

| | |
|---|---|
| `annotation` | an annotation object |

## Value

Returns an object of class `c("tbl_df", "tbl", "data.frame")` containing one row for every coreference in the corpus.

The returned data frame includes the following columns:

- "id" - integer. Id of the source document.
- "rid" - integer. Relation ID.
- "mid" - integer. Mention ID; unique to each coreference within a document.
- "mention" - character. The mention as raw words from the text.
- "mention_type" - character. One of "LIST", "NOMINAL", "PRONOMINAL", or "PROPER".
- "number" - character. One of "PLURAL", "SINGULAR", or "UNKNOWN".
- "gender" - character. One of "FEMALE", "MALE", "NEUTRAL", or "UNKNOWN".
- "animacy" - character. One of "ANIMATE", "INANIMATE", or "UNKNOWN".
- "sid" - integer. Sentence id of the coreference.
- "tid" - integer. Token id at the start of the coreference.
- "tid_end" - integer. Token id at the start of the coreference.
- "tid_head" - integer. Token id of the head of the coreference.

## Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

## References

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.*

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. The Life and Death of Discourse Entities: Identifying Singleton Mentions. In: *Proceedings of NAACL 2013*.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu and Dan Jurafsky. Deterministic coreference resolution based on entity-centric, precision-ranked rules. Computational Linguistics 39(4), 2013.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky. Stanford's Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task. In: *Proceedings of the CoNLL-2011 Shared Task, 2011*.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, Christopher Manning A Multi-Pass Sieve for Coreference Resolution. EMNLP-2010, Boston, USA. 2010.

## Examples

```
data(obama)

# how often are references made to males versus female in each speech?
get_coreference(obama) %$%
  table(gender, id)
```

---

# get_dependency *Access dependencies from an annotation object*

---

## Description

This function grabs the table of dependencies from an annotation object. These are binary relationships between the tokens of a sentence. Common examples include nominal subject (linking the object of a sentence to a verb), and adjectival modifiers (linking an adjective to a noun). While not included in the underlying data, the function has an option for linking these dependencies to the raw words and lemmas in the table of tokens. Both language-agnostic and language-specific universal dependency types are included in the output.

## Usage

```
get_dependency(annotation, get_token = FALSE)
```

## Arguments

| | |
|---|---|
| annotation | an annotation object |
| get_token | logical. Should words and lemmas be attached to the returned dependency table. |

## Value

Returns an object of class c("tbl_df", "tbl", "data.frame") containing one row for every dependency pair in the corpus.

The returned data frame includes at a minimum the following columns:

- "id" - integer. Id of the source document.
- "sid" - integer. Sentence id of the source token.
- "tid" - integer. Id of the source token.
- "sid_target" - integer. Sentence id of the source token.
- "tid_target" - integer. Id of the source token.
- "relation" - character. Language-agnostic universal dependency type.
- "relation_full" - character. Language specific universal dependency type.

If get_token is set to true, the following columns will also be included:

- "word" - character. The source word in the raw text.
- "lemma" - character. Lemmatized form of the source word.
- "word_target" - character. The target word in the raw text.
- "lemma_target" - character. Lemmatized form of the target word.

## Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

## References

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.*

Danqi Chen and Christopher D Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In: *Proceedings of EMNLP 2014*

Spence Green, Marie-Catherine de Marneffe, John Bauer, and Christopher D. Manning. 2010. Multiword Expression Identification with Tree Substitution Grammars: A Parsing tour de force with French. In: *EMNLP 2011*.

Spence Green and Christopher D. Manning. 2010. Better Arabic Parsing: Baselines, Evaluations, and Analysis. In: COLING 2010.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D. Manning. 2009. Discriminative Reordering with Chinese Grammatical Relations Features. In: Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation.

Anna Rafferty and Christopher D. Manning. 2008. Parsing Three German Treebanks: Lexicalized and Unlexicalized Baselines. In: *ACL Workshop on Parsing German.*

## Examples

```
data(obama)

# find the most common noun lemmas that are the syntactic subject of a clause
get_dependency(obama, get_token = TRUE) %>%
  filter(relation == "nsubj") %>%
  use_series(lemma_target) %>%
  table() %>%
  sort(decreasing = TRUE) %>%
  head(n = 40)
```

---

# get_document *Access document meta data from an annotation object*

---

## Description

Access document meta data from an annotation object

## Usage

```
get_document(annotation)
```

## Arguments

| | |
|---|---|
| annotation | an annotation object |

## Value

Returns an object of class `c("tbl_df", "tbl", "data.frame")` containing one row for every document in the corpus.

The returned data frame includes at least the following columns:

- "id" - integer. Id of the source document.
- "time" - date time. The time at which the parser was run on the text.
- "version" - character. Version of the CoreNLP library used to parse the text.
- "language" - character. Language of the text, in ISO 639-1 format.
- "uri" - character. Description of the raw text location. Set to `NA` if parsed from in-memory character vector.

Other application specific columns may be included as additional variables.

## Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

## References

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.*

## Examples

```
data(obama)

get_document(obama)
```

---

`get_entity` *Access named entities from an annotation object*

---

## Description

Named entity recognition attempts to find the mentions of various categories within the corpus of text. Common example include proper references to location (e.g., "Boston", or "England") or people (e.g., "Winston Churchill"), as well as specific dates (e.g., "tomorrow", or "September 19th") times, or numbers.

## Usage

```
get_entity(annotation)
```

## Arguments

| | |
|---|---|
| `annotation` | an annotation object |

## Value

Returns an object of class `c("tbl_df", "tbl", "data.frame")` containing one row for every named entity mention in the corpus.

The returned data frame includes the following columns:

- "id" - integer. Id of the source document.
- "sid" - integer. Sentence id of the entity mention.
- "tid" - integer. Token id at the start of the entity mention.
- "tid_end" - integer. Token id at the end of the entity mention.
- "entity_type" - character. When using default models, one of "LOCATION", "PERSON", "ORGANIZATION", "MONEY", "PERCENT", "DATE", "TIME".
- "entity" - character. Raw words of the named entity in the text.
- "entity_normalized" - character. Normalized version of the entity.

## Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

## References

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.*

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In: *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363-370.*

## Examples

```
data(obama)

# what are the most common entity types used in the addresses?
get_entity(obama) %>%
 use_series(entity_type) %>%
 table()

# what are the most common locations mentioned?
get_entity(obama) %>%
  filter(entity_type == "LOCATION") %>%
  use_series(entity) %>%
  table() %>%
  sort(decreasing = TRUE) %>%
  head(n = 25)

# what are the most common organizations mentioned?
get_entity(obama) %>%
  filter(entity_type == "ORGANIZATION") %>%
  use_series(entity) %>%
  table() %>%
  sort(decreasing = TRUE) %>%
  head(n = 25)
```

---

# get_sentiment *Access sentiment scores from an annotation object*

---

## Description

Sentiment analysis attempts to extract the attitudes of the narrator or speaker towards their object of study. This function extracts a sentiment score from 0 to 4 from each sentence in the annotation.

## Usage

```
get_sentiment(annotation)
```

## Arguments

| | |
|---|---|
| annotation | an annotation object |

## Value

Returns an object of class `c("tbl_df", "tbl", "data.frame")` containing one row for every sentence in the corpus.

The returned data frame includes the following columns:

- "id" - integer. Id of the source document.
- "sid" - integer. Sentence id.
- "pred_class" - integer. Predicted sentiment class of the sentence, from 0 (worst) to 4 (best).
- "p0" - double. Predicted probability of the sentence being class 0.
- "p1" - double. Predicted probability of the sentence being class 1.
- "p2" - double. Predicted probability of the sentence being class 2.
- "p3" - double. Predicted probability of the sentence being class 3.
- "p4" - double. Predicted probability of the sentence being class 4.

## Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

## References

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.*

Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." Proceedings of the conference on empirical methods in natural language processing (EMNLP). Vol. 1631. 2013.

## Examples

```
# how do the predicted sentiment scores change across the years?
get_sentiment(obama) %>%
  group_by(id) %>%
  summarize(mean(pred_class), se = sd(pred_class) / sqrt(n()))
```

`get_token` *Access tokens from an annotation object*

## Description

This function grabs the table of tokens from an annotation object. There is exactly one row for each token found in the raw text. Tokens include words as well as punctuation marks. A token called `ROOT` is also added to each sentence; it is particularly useful when interacting with the table of dependencies.

## Usage

```
get_token(annotation)
```

## Arguments

| | |
|---|---|
| `annotation` | an annotation object |

## Value

Returns an object of class `c("tbl_df", "tbl", "data.frame")` containing one row for every token in the corpus. The root of each sentence is included as its own token.

The returned data frame includes the following columns:

- "id" - integer. Id of the source document.
- "sid" - integer. Sentence id, starting from 0.
- "tid" - integer. Token id, with the root of the sentence starting at 0.
- "word" - character. Raw word in the input text.
- "lemma" - character. Lemmatized form the token.
- "upos" - character. Universal part of speech code.
- "pos" - character. Language-specific part of speech code; uses the Penn Treebank codes.
- "speaker" - character. Identity of the speaker, with `PER0` denoting the narratorial voice.
- "wiki" - character. Link to Wikipedia entity.
- "cid" - integer. Character offset at the start of the word in the original document.
- "cid_end" - integer. Character offset pointing one past the character at the end of the word.

## Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

## References

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.*

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In: *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70.*

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In: *Proceedings of HLT-NAACL 2003, pp. 252-259.*

## Examples

```
data(obama)

# find average sentence length from each address
get_token(obama) %>%
  group_by(id, sid) %>%
  summarize(sentence_length = max(tid)) %>%
  summarize(avg_sentence_length = mean(sentence_length))
```

---

# get_triple *Access triples from an annotation object*

---

## Description

Relationship triples contain a subject, object, and relationship, all extracted directly from the text. They represent factual statements whereby the subject and object are related by the words consituting the relation.

## Usage

```
get_triple(annotation)
```

## Arguments

| | |
|---|---|
| annotation | an annotation object |

## Value

Returns an object of class `c("tbl_df", "tbl", "data.frame")` containing one row for every triple found in the corpus.

The returned data frame includes the following columns:

- "id" - integer. Id of the source document.

- "subject" - character. Raw text of the triple's subject.
- "object" - character. Raw text of the triple's object.
- "relation" - character. Raw text of the triple's relation.
- "confidence" - double. Confidence score from 0 (least confident) to 1 (most confident).
- "be_prefix" - integer. Equals 1 if the triple's relationship has a form of "to be" as a prefix.
- "be_suffix" - integer. Equals 1 if the triple's relationship has a form of "to be" as a suffix.
- "of_suffix" - integer. Equals 1 if the triple's relationship has a form of "of" as a suffix.
- "tmod" - integer. Equals 1 if the triple is a temporal modifier.
- "sid" - integer. Sentence id of the triple.
- "tid_subject" - integer. Token id at the start of the triple's subject.
- "tid_subject_end" - integer. Token id at the end of the triple's subject.
- "tid_object" - integer. Token id at the start of the triple's object.
- "tid_object_end" - integer. Token id at the end of the triple's object.
- "tid_relation" - integer. Token id at the start of the triple's relation.
- "tid_relation_end" - integer. Token id at the end of the triple's relation.

## Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

## References

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.*

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. Leveraging Linguistic Structure For Open Domain Information Extraction. In: *Proceedings of the Association of Computational Linguistics (ACL), 2015*.

## Examples

```
# what are the most common relations in the text?
get_triple(obama) %>%
  use_series(relation) %>%
  table() %>%
  sort(decreasing = TRUE) %>%
  head(n = 40L)
```

`init_clean_nlp` *Initialize the cleanNLP java object*

## Description

This must be run prior to calling any other cleanNLP functions. Options that control the behavior of the pipeline must be set using one of: `set_language` or `set_properties`. Options may be reset during a given R session; in order to take effect, make a new call to this function.

## Usage

```
init_clean_nlp(lib_location = NULL, mem = "12g", verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `lib_location` | a string giving the location of the CoreNLP java files. This should point to a directory which contains, for example the file "stanford-corenlp-*.jar", where "*" is the version number. If missing, the function will try to find the library in the environment variable CORENLP_HOME, and otherwise will fail. |
| `mem` | a string giving the amount of memory to be assigned to the rJava engine. For example, "6g" assigned 6 gigabytes of memory. At least 2 gigabytes are recommended at a minimum for running the CoreNLP package. On a 32bit machine, where this is not possible, setting "1800m" may also work. This option will only have an effect the first time `initCoreNLP` is called, and also will not have an effect if the java engine is already started by a separate process. |
| `verbose` | boolean. Should messages from the pipeline be written to the console or suppressed? |

## Author(s)

Taylor B. Arnold, `<taylor.arnold@acm.org>`

## Examples

```
## Not run:
init_clean_nlp()

## End(Not run)
```

## obama *Annotation of Barack Obama's State of the Union Addresses*

### Description

Parsed text from all eight State of the Union addresses given by Barack Obama.

### References

http://www.presidency.ucsb.edu/sou.php

## pos_frequency *Universal Part of Speech Code Frequencies*

### Description

The language-specific frequency of universal part of speech codes as given in standard treebank corpora. Frequencies are multiplied by 100.

### References

http://universaldependencies.org/

## print.annotation *Print a summary of an annotation object*

### Description

Print a summary of an annotation object

### Usage

```
## S3 method for class 'annotation'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | an annotation object |
| ... | other arguments. Currently unused. |

### Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

## read_annotation *Read annotation files from disk*

### Description

Loads an annotation that has been stored as a set of csv files in a local directory. This is typically created by a call to annotate or write_annotation.

### Usage

```
read_annotation(input_dir)
```

### Arguments

| | |
|---|---|
| input_dir | path to the directory where the files are stored |

### Value

an object of class annotation

### Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

### Examples

```
## Not run:
annotation <- read_annotation("path/to/annotation")

## End(Not run)
```

## set_language *Easy interface for setting up the pipeline*

### Description

This function calls set_properties, setting all of the correct properties to parse the given language and given speed. This is the most user-friendly entry point for setting properties. The function set_properties provides more fine grained control. See Details for more information about the speed codes

### Usage

```
set_language(language, speed = 2)
```

## Arguments

| | |
|---|---|
| language | a character vector describing the desired language; should be one of: "ar", "de", "en", "es", "fr", or "zh". |
| speed | integer code. Sets which annotators should be loaded, based on on how long they take to load and run. Speed 0 is the fastest, and speed 8 is the slowest. See Details for a full description of the levels |

## Details

Currently available speed codes are integers from 0 to 8. Setting speed above 2 has no additional effect on the German and Spanish models. Setting above 1 has no effect on the French model. The available speed codes are:

- "0" runs just the tokenizer, sentence splitter, and part of speech tagger. Extremely fast.
- "1" includes the dependency parsers and, for English, the sentiment tagger. Often 20-30x slower than speed 0.
- "2" adds the named entity annotator to the parser and sentiment tagger (when available). For English models, it also includes the mentions and natlog annotators. Usually no more than twice as slow as speed 1.
- "3" adds the WikiDict annotator on top of the speed 2 annotators. This take a while to load (often upwards of 5 minutes), but once loaded is only about 50% slower than speed 2.
- "4" adds the OpenIE triples annotator on top of the speed 2 annotators. Generally takes about twice as long as speed 2.
- "5" add the coreference resolution annotator to the speed 2 annotators. Depending on the corpus, this takes about 2-4x longer than the speed 2 annotators
- "6" adds both the WikiDict and coreference resolution annotator to the speed 2 annotators. The speed difference compared to speed 5 is marginal, other than the increased start-up time.
- "7" adds both the OpenIE triples and coreference annotators to the speed 2 annotators. Can sometimes take several times longer than running them separately depending on your system resources.
- "8" add the WikiDict, OpenIE triples, and coreference annotators to the speed 2 annotators. The speed is usually similar to speed 7, as long as you have enough memory allocated to store the WikiDict tables and still have plenty of space for the OpenIE triples (14GB+).

For most users, speeds 0, 2, 5, and 6 will likely be the most useful. We suggest starting at speed 2 and down grading to 0 if your corpus is particularly large, or upgrading the 5 or 6 if you can sacrifice the slowdown. If your text is not formal written text (i.e., tweets or text messages), the speed 0 annotator should still work well but anything beyond that may be difficult. Semi-formal text such as e-mails or transcribed speech are generally okay to run for all of the levels, though you may get errors above speed 3 if the parser cannot figure out how to parse odd sections of text (the corefence and OpenIE triples annotators need the parser to order to run correctly).

## Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

## Examples

```
## Not run:
set_language("en")

## End(Not run)
```

---

`set_properties` *Set properties for the coreNLP pipeline*

---

## Description

This function allows for directly setting properties to be passed on to the Stanford CoreNLP pipeline. This will generally be of interest to experienced users who are already familiar with the general pipeline. See the function `set_language` for a more user-friendly approach.

## Usage

```
set_properties(keys, values, clear = FALSE)
```

## Arguments

| | |
|---|---|
| `keys` | a character vector of keys giving the names of the properties to set |
| `values` | a character vector the same length of keys giving the values to set each respective property to |
| `clear` | should the set of properties be cleared before setting these values |

## Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

## Examples

```
## Not run:
set_properties("annotators", "segment, ssplit, tokenize, pos")

## End(Not run)
```

## word_frequency *Most frequent English words*

### Description

A dataset of the 150k most frequently used English words, extracted by Peter Norvig from the Google Web Trillion Word Corpus. Frequencies are multiplied by 100.

### References

http://norvig.com/ngrams/

## write_annotation *Write annotation files to disk*

### Description

Takes an annotation object and stores it as a set of files in a local directory. These are stored as plain-text csv files with column headers. To save as a compressed format, instead directly call the function saveRDS.

### Usage

```
write_annotation(annotation, output_dir)
```

### Arguments

| | |
|---|---|
| annotation | annotation file being stored |
| output_dir | path to the directory where the files will be saved |

### Author(s)

Taylor B. Arnold, <taylor.arnold@acm.org>

### Examples

```
## Not run:
write_annotation(annotation, "/path/to/annotation")

## End(Not run)
```

# Index