Package 'ACEP'

May 10, 2026

Type Package

Title Análisis Computacional de Eventos de Protesta

Version 0.1.0

Author Agustín Nieto [aut, cre] (ORCID:
<https://orcid.org/0000-0003-4467-873X>)

Maintainer Agustín Nieto <agustin.nieto77@gmail.com>

Description La librería 'ACEP' contiene funciones específicas para desarrollar análisis computacional de eventos de protesta. Asimismo, contiene bases de datos con colecciones de notas sobre protestas y diccionarios de palabras conflictivas. La colección de diccionarios reune diccionarios de diferentes orígenes.
The 'ACEP' library contains specific functions to perform computational analysis of protest events. It also contains a database with collections of notes on protests and dictionaries of conflicting words. Collection of dictionaries that brings together dictionaries from different sources.

License MIT + file LICENSE

URL https://github.com/agusnieto77/ACEP,
https://agusnieto77.github.io/ACEP/

BugReports https://github.com/agusnieto77/ACEP/issues

Depends R (>= 3.5.0)

Imports graphics, stats, httr, jsonlite, stringr, utils, magrittr

Suggests furrr, ollamar, future, covr, knitr, rmarkdown, reticulate, rsyntax, spacyr, tidygeocoder, udpipe, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Language es

LazyData true

RoxygenNote 7.3.3

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2026-05-10 18:42:04 UTC

Contents

`acep_bases` *Coleccion de notas y recursos de prueba*

Description

Lista con fuentes de datos y muestras preprocesadas utilizadas en los ejemplos del paquete. Incluye enlaces de descarga para distintos portales, resumenes estadisticos y conjuntos anotados manualmente que permiten evaluar diccionarios, extraccion de tripletes y desempeno de modelos generativos.

Usage

```
data(acep_bases)
```

Format

Lista con 11 objetos:

la_nueva URL para descargar una muestra del corpus de notas del diario La Nueva Provincia de Bahia Blanca.

rev_puerto URL para descargar el corpus de notas de Revista Puerto.

rp_procesada data frame con indicadores de conflictividad construidos a partir de Revista Puerto.

lc_mdp URL para descargar el corpus de notas del diario La Capital (Mar del Plata).

rp_mdp URL para descargar el corpus de notas de Revista Puerto (edicion Mar del Plata).

ed_neco URL para descargar el corpus de notas del diario Ecos Diarios (Necochea).

ln_bb URL para descargar el corpus de notas de La Nueva (Bahia Blanca).

ln_arg URL para descargar un subconjunto de notas de La Nacion.

lc_720 data frame con 720 notas de La Capital publicadas entre 2016 y 2019, curado y documentado por Guillermina Laitano. Contiene etiquetas binarias manuales que permiten evaluar el diccionario de conflictividad, la extraccion de tripletes semanticos y la capacidad de distintos modelos generativos para tareas de clasificacion binaria y extraccion estructurada de eventos de protesta.

spacy_postag data frame con una oracion procesada con **spacyr** que sirve como ejemplo para funciones de dependencias y SVO.

titulares vector con titulares sinteticos referidos a conflictos sociales.

Source

Revista Puerto

La Nueva

References

Nieto, Agustin 2020 "Intersecciones entre historia digital e historia social: un ejercicio de lectura distante sobre la conflictividad maritima en la historia argentina reciente". Drassana: revista del Museu Maritim (28):122-42. (Revista Drassana)

Examples

```
acep_bases$rp_procesada[1:6, ]
```

`acep_claude` *Interaccion con modelos Claude usando Structured Outputs*

Description

Funcion para interactuar con la API de Anthropic Claude utilizando Tool Calling para garantizar respuestas en formato JSON que cumplen estrictamente con un esquema predefinido. A diferencia de OpenAI, Anthropic utiliza "forced tool use" para lograr structured outputs, definiendo el esquema deseado como input_schema de una herramienta y forzando al modelo a usarla. Compatible con todos los modelos Claude.

Usage

```
acep_claude(
  texto,
  instrucciones,
  modelo = "claude-sonnet-4-20250514",
  api_key = Sys.getenv("ANTHROPIC_API_KEY"),
  schema = NULL,
  parse_json = TRUE,
  temperature = 0,
  max_tokens = 2000,
  top_p = 0.2,
  top_k = NULL
)
```

Arguments

`texto`	Texto a analizar con Claude. Puede ser una noticia, tweet, documento, etc.
`instrucciones`	Instrucciones en lenguaje natural que indican al modelo que hacer con el texto. Ejemplo: "Extrae todas las entidades nombradas", "Clasifica el sentimiento".
`modelo`	Modelo de Claude a utilizar. Modelos disponibles (ordenados por potencia): - Claude 4.5: ‘"claude-sonnet-4-5-20250929"‘ (mas reciente y potente), ‘"claude-haiku-4-5-20251001"‘ (rapido) - Claude 4.1: ‘"claude-opus-4-1-20250805"‘ (razonamiento excepcional) - Claude 4: ‘"claude-opus-4-20250514"‘, ‘"claude-sonnet-4-20250514"‘ - Claude 3.7: ‘"claude-3-7-sonnet-20250219"‘ - Claude 3.5: ‘"claude-3-5-haiku-20241022"‘ (rapido y economico) - Claude 3: ‘"claude-3-opus-20240229"‘, ‘"claude-3-haiku-20240307"‘ Por defecto: ‘"claude-sonnet-4-20250514"‘. Ver: https://docs.anthropic.com/en/docs/about-claude/models

api_key	Clave de API de Anthropic. Si no se proporciona, busca la variable de entorno 'ANTHROPIC_API_KEY'. Para obtener una clave: https://console.anthropic.com/
schema	Esquema JSON que define la estructura de la respuesta. Puede usar 'acep_gpt_schema()' para obtener esquemas predefinidos o crear uno personalizado. Si es 'NULL', usa un esquema simple con campo "respuesta".
parse_json	Logico. Si 'TRUE' (por defecto), parsea automaticamente el JSON a un objeto R (lista o data frame). Si 'FALSE', devuelve el JSON como string.
temperature	Parametro de temperatura (0-1). Valores bajos (0-0.3) generan respuestas mas deterministas y consistentes. Valores altos (0.7-1) mas creativas. Por defecto: 0 (maxima determinismo). NOTA: Anthropic no permite usar 'temperature' y 'top_p' simultaneamente.
max_tokens	Numero maximo de tokens en la respuesta. Por defecto: 2000.
top_p	Parametro top-p para nucleus sampling (0-1). Controla la diversidad de la respuesta. Por defecto: 0.2. NOTA: Solo se usa si 'temperature' es 0 (valor por defecto).
top_k	Parametro top-k (solo disponible en Claude). Limita la seleccion a los K tokens mas probables. Por defecto: NULL (deshabilitado).

Details

Diferencias importantes entre modelos Claude:

- **Claude Sonnet 4.5** ('claude-sonnet-4-5'): NO permite 'temperature' y 'top_p' simultaneamente. La funcion solo envia uno de los dos si fue modificado del default.

- **Otros modelos Claude** ('claude-sonnet-4-20250514', 'claude-3-5-haiku-20241022', 'claude-3-opus-20240229', etc.): SI permiten ambos parametros simultaneamente.

La funcion detecta automaticamente el modelo y aplica la logica correcta.

Value

Si 'parse_json=TRUE', devuelve una lista o data frame con la respuesta estructurada segun el esquema. Si 'parse_json=FALSE', devuelve un string JSON.

Examples

```
## Not run:
# Extraer entidades de un texto
texto <- "El SUTEBA convoco a un paro en Buenos Aires el 15 de marzo."
instrucciones <- "Extrae todas las entidades nombradas del texto."
schema <- acep_gpt_schema("extraccion_entidades")
resultado <- acep_claude(texto, instrucciones, schema = schema)
print(resultado)

# Analisis de sentimiento
texto <- "La protesta fue pacifica y bien organizada."
schema <- acep_gpt_schema("sentimiento")
resultado <- acep_claude(texto, "Analiza el sentimiento del texto", schema = schema)
print(resultado$sentimiento_general)
```

```
# Clasificar noticia
texto <- "Trabajadores reclamaron mejoras salariales."
schema <- acep_gpt_schema("clasificacion")
resultado <- acep_claude(texto, "Clasifica esta noticia", schema = schema)
print(resultado$categoria)

## End(Not run)
```

---

acep_clean *Limpieza de texto.*

Description

Función que limpia y normaliza las notas/textos.

Usage

```
acep_clean(
  x,
  tolower = TRUE,
  rm_cesp = TRUE,
  rm_emoji = TRUE,
  rm_hashtag = TRUE,
  rm_users = TRUE,
  rm_punt = TRUE,
  rm_num = TRUE,
  rm_url = TRUE,
  rm_meses = TRUE,
  rm_dias = TRUE,
  rm_stopwords = TRUE,
  rm_shortwords = TRUE,
  rm_newline = TRUE,
  rm_whitespace = TRUE,
  other_sw = NULL,
  u = 1
)
```

Arguments

x	vector de textos al que se le aplica la función de limpieza de texto.
tolower	convierte los textos a minúsculas.
rm_cesp	remueve caracteres especiales.
rm_emoji	remueve los emojis.
rm_hashtag	remueve los hashtags.
rm_users	remueve las menciones de usuarixs de redes sociales.
rm_punt	remueve la puntuación.

rm_num	remueve números.
rm_url	remueve las url.
rm_meses	remueve los meses del año.
rm_dias	remueve los dias de la semana.
rm_stopwords	remueve palabras vacías.
rm_shortwords	remueve las palabras cortas.
rm_newline	remueve los saltos de linea.
rm_whitespace	remueve los espacios en blanco.
other_sw	su valor por defecto es NULL, sirve para ampliar el listado de stopwords con un nuevo vector de palabras.
u	umbral de caracteres para la función rm_shortwords.

Value

Si todas las entradas son correctas, la salida sera un vector de textos normalizados.

Examples

```
acep_clean("El SUTEBA fue al paro. Reclaman mejoras salariales.", rm_stopword = FALSE)
acep_clean("El SUTEBA fue al paro. Reclaman mejoras salariales.", rm_stopword = TRUE)
```

---

acep_clear_regex_cache

Limpiar caché de expresiones regulares

Description

Elimina todos los patrones regex almacenados en el caché interno de ‘acep_count()‘. Útil para liberar memoria cuando se han procesado muchos diccionarios diferentes o cuando se quiere forzar la recompilación de patrones.

Usage

```
acep_clear_regex_cache()
```

Value

Devuelve ‘NULL‘ invisiblemente.

Examples

```
# Limpiar el caché
acep_clear_regex_cache()
```

acep_context *Función para extraer contexto de palabras o frases.*

Description

Versión optimizada que usa vectorización en lugar de bucles anidados. Mejora de rendimiento de 70-80

Usage

```
acep_context(texto, clave, izq = 1, der = 1, ci = "\\b", cd = "\\S*")
```

Arguments

texto	vector con los textos a procesar.
clave	vector de palabras clave a contextualizar.
izq	número de palabras de la ventana hacia la izquierda.
der	número de palabras de la ventana hacia la derecha.
ci	expresión regular a la izquierda de la palabra clave.
cd	expresión regular a la derecha de la palabra clave.

Value

Data frame con id de textos y contexto de palabras/frases.

Examples

```
texto <- "El SOIP para por aumento de salarios"
texto_context <- acep_context(texto = texto, clave = "para")
texto_context
```

acep_corpus *Constructor de corpus para analisis de texto*

Description

Crea un objeto de clase ‘acep_corpus‘ que encapsula una coleccion de textos junto con su metadata asociada. Este objeto esta disenado para trabajar con las funciones pipeline de ACEP (‘pipe_clean‘, ‘pipe_count‘, etc.), permitiendo un flujo de trabajo encadenado y manteniendo trazabilidad de las transformaciones aplicadas.

Usage

```
acep_corpus(texto, metadata = NULL, id = NULL)
```

Arguments

texto	Vector de caracteres con los textos a almacenar en el corpus.
metadata	Lista opcional con informacion adicional sobre el corpus (ej: fuente, fecha de recoleccion, categorias tematicas).
id	Vector opcional de identificadores unicos para cada texto. Si no se proporciona, se asignan IDs secuenciales (1, 2, 3, ...).

Value

Objeto de clase 'acep_corpus' con la siguiente estructura:

- texto_original: Vector con los textos originales sin procesar
- texto_procesado: Vector con textos procesados (NULL inicialmente)
- id: Identificadores de cada texto
- metadata: Metadata adicional del corpus
- procesamiento: Registro de transformaciones aplicadas

Examples

```
# Crear corpus simple
textos <- c("El SUTEBA va al paro", "SOIP protesta por salarios")
corpus <- acep_corpus(textos)
print(corpus)

# Crear corpus con metadata e IDs personalizados
corpus <- acep_corpus(
  texto = c("Noticia 1", "Noticia 2"),
  metadata = list(fuente = "Diario La Nacion", year = 2024),
  id = c("LN001", "LN002")
)
```

---

acep_count *Conteo de menciones de palabras de un diccionario*

Description

Cuenta el número de veces que aparecen palabras de un diccionario en cada texto. Utiliza expresiones regulares con límites de palabra (word boundaries) para evitar coincidencias parciales. Incluye un sistema de caché que almacena los patrones regex compilados para acelerar ejecuciones repetidas con el mismo diccionario.

Usage

```
acep_count(texto, dic, use_cache = TRUE)
```

Arguments

texto	vector de textos al que se le aplica la función de conteo.
dic	vector de palabras del diccionario utilizado.
use_cache	logical, usar caché de regex (default TRUE).

Value

Vector con frecuencia de palabras del diccionario.

Examples

```
df <- data.frame(texto = c("El SUTEBA fue al paro. Reclaman mejoras salariales.",
"El SOIP lleva adelante un plan de lucha con paros y piquetes."))
diccionario <- c("paro", "lucha", "piquetes")
df$detect <- acep_count(df$texto, diccionario)
df
```

---

acep_db *Frecuencia, menciones e intensidad.*

Description

Función que usa las funciones acep_frec, acep_count y acep_int y devuelve una tabla con tres columnas nuevas: numero de palabras, número de menciones del diccionario, indice de intensidad.

Usage

```
acep_db(db, t, d, n)
```

Arguments

db	data frame con los textos a procesar.
t	columna de data frame que contiene el vector de textos a procesar.
d	diccionario en formato vector.
n	cantidad de decimales del indice de intensidad.

Value

Si todas las entradas son correctas, la salida sera una base de datos en formato tabular con tres nuevas variables.

Examples

```
df <- data.frame(texto = c("El SUTEBA fue al paro. Reclaman mejoras salariales.",
"El SOIP lleva adelante un plan de lucha con paros y piquetes."))
diccionario <- c("paro", "lucha", "piquetes")
acep_db(df, df$texto, diccionario, 4)
```

acep_detect *Detección de menciones de palabras.*

Description

Función que detecta de menciones de palabras que refieren a conflictos en cada una de las notas/textos.

Usage

```
acep_detect(x, y, u = 1, tolower = TRUE)
```

Arguments

x	vector de textos al que se le aplica la función de detección de menciones de palabras del diccionario.
y	vector de palabras del diccionario utilizado.
u	umbral para atribuir valor positivo a la detección de las menciones.
tolower	convierte los textos a minúsculas.

Value

Si todas las entradas son correctas, la salida sera un vector numérico.

Examples

```
df <- data.frame(texto = c("El SUTEBA fue al paro. Reclaman mejoras salariales.",
"El SOIP lleva adelante un plan de lucha con paros y piquetes."))
diccionario <- c("paro", "lucha", "piquetes")
df$detect <- acep_detect(df$texto, diccionario)
df
```

acep_diccionarios *Colección de diccionarios.*

Description

Colección de diccionarios que reúne diccionarios de diferentes orígenes. El diccionario dicc_confl_acep fueron construidos en el marco del Observatorio de Conflictividad de la UNMdP. Los diccionarios dicc_confl_gp y dicc_viol_gp fueron extraídos de Albrieu y Palazzo (2020).

Usage

```
data(acep_diccionarios)
```

Format

Es un objeto de clase 'list' con 3 componentes.

dicc_confl_gp es un vector con palabras de un diccionario de términos que refieren a conflictos

dicc_viol_gp es un vector con palabras de un diccionario de términos que refieren a violencia

dicc_confl_sismos es un vector con palabras de un diccionario de términos que refieren a conflictos

Source

Revista Puerto

La Nueva

References

Albrieu, Ramiro y Gabriel Palazzo 2020 «Categorización de conflictos sociales en el ámbito de los recursos naturales: un estudio de las actividades extractivas mediante la minería de textos». Revista CEPAL (131):29-59. (Revista CEPAL)

Laitano, Guillermina y Agustín Nieto «Análisis computacional de la conflictividad laboral en Mar del Plata durante el gobierno de Cambiemos». Ponencia presentado en VI Workshop - Los conflictos laborales en la Argentina del siglo XX y XXI: un abordaje interdisciplinario de conceptos, problemas y escalas de análisis, Tandil, 2021.

Examples

```
diccionario <- acep_load_base(acep_diccionarios$dicc_viol_gp)
diccionario
```

---

acep_extract *Función para buscar y extraer palabras en un texto.*

Description

Esta función busca palabras clave en un texto y extrae los resultados en un formato especifico.

Usage

```
acep_extract(texto, dic, sep = "; ", izq = "\\b\\w*", der = "\\w*\\b")
```

Arguments

texto	El texto en el que se buscaran las palabras clave.
dic	Un vector con las palabras clave a buscar.
sep	El separador utilizado para concatenar las palabras clave encontradas (por defecto: " | ").
izq	expresión regular para incorporar otros caracteres a la izquierda de los términos del vector 'dic'.
der	expresión regular para incorporar otros caracteres a la derecha de los términos del vector 'dic'.

Value

Si todas las entradas son correctas, la salida sera un vector de tipo caracter. procesado y el contexto de las palabras y/o frases entradas.

Examples

```
texto <- "Los obreros del pescado, en el marco de una huelga y
realizaron una manifestación con piquete en el puerto de la ciudad."
dicc <- c("huel", "manif", "piq")
acep_extract(texto, dicc)
```

---

acep_frec *Frecuencia de palabras totales.*

Description

Función que cuenta la frecuencia de palabras totales en cada una de las notas/textos.

Usage

```
acep_frec(x)
```

Arguments

x	vector de textos al que se le aplica la función de conteo de la frecuencia de palabras.

Value

Si todas las entradas son correctas, la salida sera un vector con una frecuencia de palabras.

Examples

```
acep_frec("El SUTEBA fue al paro. Reclaman mejoras salariales.")
```

acep_gemini *Interaccion con modelos Gemini usando Structured Outputs*

Description

Funcion para interactuar con la API de Google Gemini utilizando Structured Outputs nativos. Gemini soporta generacion de JSON estructurado mediante el parametro ‘responseSchema‘ que garantiza que las respuestas cumplan con el esquema definido. Compatible con todos los modelos Gemini 2.5 y Gemini 2.0. Acceso gratuito para uso limitado disponible en Google AI Studio.

Usage

```
acep_gemini(
  texto,
  instrucciones,
  modelo = "gemini-2.5-flash",
  api_key = Sys.getenv("GEMINI_API_KEY"),
  schema = NULL,
  parse_json = TRUE,
  temperature = 0,
  max_tokens = 2000,
  top_p = 0.95,
  top_k = 40
)
```

Arguments

texto	Texto a analizar con Gemini. Puede ser una noticia, tweet, documento, etc.
instrucciones	Instrucciones en lenguaje natural que indican al modelo que hacer con el texto. Ejemplo: "Extrae todas las entidades nombradas", "Clasifica el sentimiento".
modelo	Modelo de Gemini a utilizar. Opciones recomendadas: - Gemini 2.5: ‘"gemini-2.5-flash"‘ (rapido, multimodal, por defecto), ‘"gemini-2.5-flash-lite"‘ (mas economico), ‘"gemini-2.5-pro"‘ (mas potente) - Gemini 2.0: ‘"gemini-2.0-flash"‘, ‘"gemini-2.0-flash-lite"‘ Por defecto: ‘"gemini-2.5-flash"‘. Ver: https://ai.google.dev/gemini-api/docs/models
api_key	Clave de API de Google Gemini. Si no se proporciona, busca la variable de entorno ‘GEMINI_API_KEY‘. Para obtener una clave: https://aistudio.google.com/apikey
schema	Esquema JSON que define la estructura de la respuesta. Puede usar ‘acep_gpt_schema()‘ para obtener esquemas predefinidos o crear uno personalizado. Si es ‘NULL‘, usa un esquema simple con campo "respuesta". NOTA: Gemini usa un subconjunto de OpenAPI 3.0 Schema para definir estructuras.
parse_json	Logico. Si ‘TRUE‘ (por defecto), parsea automaticamente el JSON a un objeto R (lista o data frame). Si ‘FALSE‘, devuelve el JSON como string.
temperature	Parametro de temperatura (0-2). Valores bajos (0-0.3) generan respuestas mas deterministas. Valores altos (0.7-1) mas creativas. Por defecto: 0. Valor recomendado por Google: 1.0.

max_tokens	Numero maximo de tokens en la respuesta. Por defecto: 2000.
top_p	Parametro top-p para nucleus sampling (0-1). Controla la diversidad de la respuesta. Por defecto: 0.95 (valor tipico para Gemini).
top_k	Parametro top-k. Limita la seleccion a los K tokens mas probables. Por defecto: 40 (valor tipico para Gemini).

Details

La API de Gemini usa un enfoque diferente para structured outputs: - Define ‘responseMimeType: "application/json"‘ en ‘generationConfig‘ - Usa ‘responseSchema‘ con formato OpenAPI 3.0 Schema - Soporta tipos: string, integer, number, boolean, array, object - Campo opcional ‘propertyOrdering‘ controla orden de propiedades en respuesta

Diferencias importantes con OpenAI: - No requiere campo ‘additionalProperties: false‘ (se maneja automaticamente) - Los campos son opcionales por defecto (usar ‘required‘ para campos obligatorios) - El esquema cuenta como tokens de entrada

Value

Si ‘parse_json=TRUE‘, devuelve una lista o data frame con la respuesta estructurada segun el esquema. Si ‘parse_json=FALSE‘, devuelve un string JSON.

Examples

```
## Not run:
# Configurar API key
Sys.setenv(GEMINI_API_KEY = "tu-api-key")

# Extraer entidades con Gemini 2.5 Flash
texto <- "El SUTEBA convoco a un paro en Buenos Aires el 15 de marzo."
resultado <- acep_gemini(texto, "Extrae las entidades nombradas",
                         schema = acep_gpt_schema("extraccion_entidades"))

# Analisis de sentimiento con modelo economico
resultado <- acep_gemini(texto, "Analiza el sentimiento",
                         modelo = "gemini-2.5-flash-lite",
                         schema = acep_gpt_schema("sentimiento"))

# Usar Gemini 2.0 Flash Lite (mas rapido)
resultado <- acep_gemini(texto, "Extrae entidades",
                         modelo = "gemini-2.0-flash-lite",
                         schema = acep_gpt_schema("extraccion_entidades"))

## End(Not run)
```

acep_gpt *Interaccion con modelos GPT usando Structured Outputs*

Description

Funcion para interactuar con la API de OpenAI utilizando Structured Outputs, una funcionalidad que garantiza respuestas en formato JSON que cumplen estrictamente con un esquema predefinido. Esto elimina la necesidad de parseo y validacion manual, haciendo las respuestas mas confiables y estructuradas. Compatible con modelos ‘gpt-4o‘ y ‘gpt-4o-mini‘.

Usage

```
acep_gpt(
  texto,
  instrucciones,
  modelo = "gpt-4o-mini",
  api_key = Sys.getenv("OPENAI_API_KEY"),
  schema = NULL,
  parse_json = TRUE,
  temperature = 0,
  max_tokens = 2000,
  top_p = 0.2,
  frequency_penalty = 0.2,
  seed = 123456
)
```

Arguments

texto	Texto a analizar con GPT. Puede ser una noticia, tweet, documento, etc.
instrucciones	Instrucciones en lenguaje natural que indican al modelo que hacer con el texto. Ejemplo: "Extrae todas las entidades nombradas", "Clasifica el sentimiento".
modelo	Modelo de OpenAI a utilizar. Compatible con Structured Outputs: ‘"gpt-4o-mini"‘ (mas rapido y economico), ‘"gpt-4o"‘, ‘"gpt-4o-2024-08-06"‘ (mas potente), ‘"gpt-4.1"‘, ‘"gpt-5-nano"‘, ‘"gpt-5-mini"‘, ‘"o1-mini"‘, ‘"o4-mini"‘, entre otros. Por defecto: ‘"gpt-4o-mini"‘. Ver: https://platform.openai.com/docs/guides/structured-outputs
api_key	Clave de API de OpenAI. Si no se proporciona, busca la variable de entorno ‘OPENAI_API_KEY‘. Para obtener una clave: https://platform.openai.com/api-keys
schema	Esquema JSON que define la estructura de la respuesta. Puede usar ‘acep_gpt_schema()‘ para obtener esquemas predefinidos o crear uno personalizado. Si es ‘NULL‘, usa un esquema simple con campo "respuesta".
parse_json	Logico. Si ‘TRUE‘ (por defecto), parsea automaticamente el JSON a un objeto R (lista o data frame). Si ‘FALSE‘, devuelve el JSON como string.

`temperature` Parametro de temperatura (0-2). Valores bajos (0-0.3) generan respuestas mas deterministas y consistentes. Valores altos (0.7-1) mas creativas. Por defecto: 0 (maxima determinismo). NOTA: Los modelos gpt-5, o1 y o4 solo aceptan temperature = 1 (default de OpenAI).

`max_tokens` Numero maximo de tokens en la respuesta. Por defecto: 2000.

`top_p` Parametro top-p para nucleus sampling (0-1). Controla la diversidad de la respuesta. Por defecto: 0.2. NOTA: Ignorado en modelos gpt-5, o1 y o4.

`frequency_penalty` Penalizacion por repeticion de tokens frecuentes (-2 a 2). Por defecto: 0.2. NOTA: Ignorado en modelos gpt-5, o1 y o4.

`seed` Semilla numerica para reproducibilidad. Usar el mismo seed con los mismos parametros genera respuestas identicas. Por defecto: 123456. NOTA: Ignorado en modelos gpt-5, o1 y o4.

Details

Diferencias entre modelos:

- **Modelos GPT-4o/GPT-4.1**: Soportan todos los parametros (temperature, top_p, frequency_penalty, seed). Usan 'max_tokens'.

- **Modelos GPT-5/o1/o4**: Solo aceptan temperature = 1 (default). Los parametros temperature, top_p, frequency_penalty y seed son automaticamente omitidos. Usan 'max_completion_tokens' en lugar de 'max_tokens'.

La funcion maneja estas diferencias automaticamente segun el modelo especificado.

Value

Si 'parse_json=TRUE', devuelve una lista o data frame con la respuesta estructurada segun el esquema. Si 'parse_json=FALSE', devuelve un string JSON.

Examples

```
## Not run:
# Extraer entidades de un texto
texto <- "El SUTEBA convoco a un paro en Buenos Aires el 15 de marzo."
instrucciones <- "Extrae todas las entidades nombradas del texto."
schema <- acep_gpt_schema("extraccion_entidades")
resultado <- acep_gpt(texto, instrucciones, schema = schema)
print(resultado)

# Analisis de sentimiento
texto <- "La protesta fue pacifica y bien organizada."
schema <- acep_gpt_schema("sentimiento")
resultado <- acep_gpt(texto, "Analiza el sentimiento del texto", schema = schema)
print(resultado$sentimiento_general)

# Clasificar noticia
texto <- "Trabajadores reclamaron mejoras salariales."
schema <- acep_gpt_schema("clasificacion")
resultado <- acep_gpt(texto, "Clasifica esta noticia", schema = schema)
```

```
print(resultado$categoria)

## End(Not run)
```

`acep_gpt_schema` *Esquemas JSON predefinidos para analisis de texto con GPT*

Description

Proporciona esquemas JSON predefinidos y validados para casos de uso comunes en analisis de texto con GPT. Estos esquemas garantizan respuestas estructuradas y consistentes para tareas como extraccion de entidades, clasificacion, analisis de sentimiento, resumen, pregunta-respuesta, extraccion de tripletes y analisis de acciones de protesta.

Usage

```
acep_gpt_schema(tipo = "extraccion_entidades")
```

Arguments

`tipo`	Tipo de esquema a devolver. Opciones:

- `"extraccion_entidades"`: Extrae personas, organizaciones, lugares, fechas y eventos
- `"clasificacion"`: Clasifica el texto en categorias con nivel de confianza
- `"sentimiento"`: Analiza sentimiento general y por aspectos especificos
- `"resumen"`: Genera resumenes cortos y detallados con puntos clave
- `"qa"`: Responde preguntas con citas textuales y nivel de confianza
- `"tripletes"`: Extrae relaciones sujeto-predicado-objeto
- `"protesta_breve"`: Extrae informacion basica de acciones de protesta (fecha, sujeto, accion, objeto, lugar)
- `"protesta_detallada"`: Extrae informacion detallada de multiples acciones de protesta con 9 campos por accion
- `"verdadero_falso"`: Devuelve una respuesta booleana simple (TRUE o FALSE) con nivel de confianza (0 a 1) y justificacion opcional

Value

Lista con esquema JSON compatible con OpenAI Structured Outputs. Puede usarse directamente en el parametro ‘schema‘ de ‘acep_gpt()‘ o ‘acep_ollama()‘.

Examples

```
# Obtener esquema para extraccion de entidades
schema_entidades <- acep_gpt_schema("extraccion_entidades")
names(schema_entidades$properties)  # personas, organizaciones, lugares, fechas, eventos

# Obtener esquema para clasificacion
schema_clasif <- acep_gpt_schema("clasificacion")
names(schema_clasif$properties)  # categoria, confianza, justificacion

# Obtener esquema para analisis de sentimiento
schema_sent <- acep_gpt_schema("sentimiento")
names(schema_sent$properties)  # sentimiento_general, puntuacion, aspectos

# Obtener esquema para analisis breve de protestas
schema_protesta <- acep_gpt_schema("protesta_breve")
names(schema_protesta$properties)  # fecha, sujeto, accion, objeto, lugar

# Obtener esquema para analisis detallado de protestas
schema_protesta_det <- acep_gpt_schema("protesta_detallada")
names(schema_protesta_det$properties)  # acciones (array con 9 campos cada una)

# Obtener esquema para respuesta verdadero/falso
schema_bool <- acep_gpt_schema("verdadero_falso")
names(schema_bool$properties)  # respuesta, nivel_confianza, justificacion
```

---

acep_int *Índice de intensidad.*

Description

Función que elabora un indice de intensidad en base a la relación entre palabras totales y palabras del diccionario presentes en el texto.

Usage

```
acep_int(pc, pt, decimales = 4)
```

Arguments

pc	vector numérico con la frecuencia de palabras conflictivas presentes en cada texto.
pt	vector de palabras totales en cada texto.
decimales	cantidad de decimales, por defecto tiene 4 pero se puede modificar.

Value

Si todas las entradas son correctas, la salida sera un vector numérico.

Examples

```
conflictos <- c(1, 5, 0, 3, 7)
palabras <- c(4, 11, 12, 9, 34)
acep_int(conflictos, palabras, 3)
```

`acep_load_base` *Carga los corpus y las bases creadas por el Observatorio.*

Description

Función para cargar bases de datos disponibles online. Por ahora están disponibles las siguientes bases: Revista Puerto 'rp_mdp'; La Nueva 'ln_bb', La Capital 'lc_mdp', Ecos Diarios 'ed_neco', La Nación 'ln_arg'

Usage

```
acep_load_base(tag)
```

Arguments

`tag`	etiqueta identificatoria del data frame a cargar: acep_bases$rp_mdp, acep_bases$ln_bb, acep_bases$lc_mdp, acep_bases$ed_neco, acep_bases$ln_arg

Value

Si todas las entradas son correctas, la salida sera una base de datos en formato tabular con un corpus de notas.

Examples

```
bd_sismos <- acep_bases$rev_puerto
head(acep_load_base(tag = bd_sismos))
```

`acep_may` *Convierte caracteres a mayusculas.*

Description

Esta función toma un vector de texto y convierte todas las letras minúsculas en mayúsculas, manteniendo el resto de los caracteres sin cambios.

Usage

```
acep_may(x)
```

Arguments

x	es un vector de texto (caracteres) que se desea convertir a mayúsculas.

Value

Devuelve un nuevo vector con todas las letras en mayúsculas.

Examples

```
vector_texto <- c("soip", "cGt", "Sutna")
acep_may(vector_texto)
vector_numeros <- c(1, 2, 3, 4, 5)
acep_may(vector_numeros)
vector_mezclado <- c("sutna", 123, "Ate")
acep_may(vector_mezclado)
```

---

acep_min *Convierte caracteres a minúsculas.*

Description

Esta función toma un vector de texto y convierte todas las letras mayusculas en minúsculas, manteniendo el resto de los caracteres sin cambios.

Usage

```
acep_min(x)
```

Arguments

x	Un vector de texto (caracteres) que se desea convertir a minúsculas.

Value

Devuelve un nuevo vector con todas las letras en minúsculas.

Examples

```
vector_texto <- c("SUTEBA", "Sindicato", "PEN")
acep_min(vector_texto)
vector_numeros <- c(1, 2, 3, 4, 5)
acep_min(vector_numeros)
vector_mezclado <- c("Soip", 123, "CGT")
acep_min(vector_mezclado)
```

acep_ollama *Interaccion con modelos Ollama locales y cloud usando Structured Outputs*

Description

Funcion para interactuar con modelos de lenguaje usando Ollama. Soporta tanto modelos locales (ejecutados en tu computadora sin costos) como modelos cloud de Ollama (modelos grandes como DeepSeek 671B, Qwen3 Coder 480B, Kimi 1T que se ejecutan en la nube sin necesidad de GPU local). Utiliza structured outputs para garantizar respuestas en formato JSON que cumplen con un esquema predefinido.

Usage

```
acep_ollama(
  texto,
  instrucciones,
  modelo = "qwen3:1.7b",
  schema = NULL,
  parse_json = TRUE,
  temperature = 0,
  max_tokens = 4000,
  host = "http://localhost:11434",
  api_key = Sys.getenv("OLLAMA_API_KEY"),
  seed = 123456
)
```

Arguments

texto	Texto a analizar. Puede ser una noticia, tweet, documento, etc.
instrucciones	Instrucciones en lenguaje natural que indican al modelo que hacer con el texto. Ejemplo: "Extrae todas las entidades nombradas", "Clasifica el sentimiento".
modelo	Modelo de Ollama a utilizar. - Para Ollama local: "qwen3:1.7b", "llama3.2:latest", "mistral", "phi3", "gemma2" (debe estar previamente descargado con 'ollama pull nombre_modelo') - Para Ollama Cloud API: modelos cloud especificos disponibles sin GPU local: "deepseek-v3.1:671b-cloud", "gpt-oss:20b-cloud", "gpt-oss:120b-cloud", "kimi-k2:1t-cloud", "qwen3-coder:480b-cloud", "glm-4.6:cloud", "minimax-m2:cloud" Por defecto: "qwen3:1.7b"
schema	Esquema JSON que define la estructura de la respuesta. Puede usar 'acep_gpt_schema()' para obtener esquemas predefinidos o crear uno personalizado. Si es NULL, usa un esquema simple con campo "respuesta".
parse_json	Logico. Si TRUE (por defecto), parsea automaticamente el JSON a un objeto R (lista o data frame). Si FALSE, devuelve el JSON como string.
temperature	Parametro de temperatura (0-2). Valores bajos (0-0.3) generan respuestas mas deterministas. Valores altos (0.7-1) mas creativas. Por defecto: 0.

max_tokens	Numero maximo de tokens en la respuesta. Por defecto: 4000.
host	URL del servidor Ollama. Por defecto: "http://localhost:11434" para uso local. Para usar Ollama Cloud API, especificar "https://ollama.com" (sin /api, se agrega automaticamente).
api_key	API key para Ollama API remota. Solo requerido si usas un servidor remoto. Por defecto busca la variable de entorno OLLAMA_API_KEY. Para uso local (localhost) no es necesario.
seed	Semilla numerica para reproducibilidad. Por defecto: 123456.

Value

Si parse_json=TRUE, devuelve una lista o data frame con la respuesta estructurada segun el esquema. Si parse_json=FALSE, devuelve un string JSON.

Examples

```
## Not run:
# Primero, instalar Ollama y descargar un modelo:
# Terminal: ollama pull llama3.1

# Extraer entidades de un texto
texto <- "El SUTEBA convoco a un paro en Buenos Aires el 15 de marzo."
instrucciones <- "Extrae todas las entidades nombradas del texto."
schema <- acep_gpt_schema("extraccion_entidades")
resultado <- acep_ollama(texto, instrucciones, schema = schema)
print(resultado)

# Analisis de sentimiento
texto <- "La protesta fue pacifica y bien organizada."
schema <- acep_gpt_schema("sentimiento")
resultado <- acep_ollama(texto, "Analiza el sentimiento del texto", schema = schema)
print(resultado$sentimiento_general)

# Usar Ollama Cloud API (requiere API key)
# Los modelos cloud se ejecutan sin necesidad de GPU local
Sys.setenv(OLLAMA_API_KEY = "tu-api-key")
resultado_remoto <- acep_ollama(
  texto = texto,
  instrucciones = "Extrae entidades",
  modelo = "deepseek-v3.1:671b-cloud",  # Modelo cloud de 671B parametros
  host = "https://ollama.com",
  schema = acep_gpt_schema("extraccion_entidades")
)


## End(Not run)
```

acep_ollama_setup *Guia de instalacion y uso de Ollama*

Description

Imprime instrucciones para instalar y configurar Ollama en tu sistema.

Usage

```
acep_ollama_setup()
```

acep_openrouter *Interaccion con modelos de IA usando OpenRouter*

Description

Funcion para interactuar con multiples proveedores de IA (OpenAI, Anthropic, Google, Meta, etc.) a traves de la API unificada de OpenRouter. Soporta Structured Outputs para modelos compatibles (OpenAI GPT-4o+, Fireworks, y otros). OpenRouter normaliza las diferencias entre proveedores, permitiendo acceder a 400+ modelos con una sola API. Ideal para comparar modelos o usar fallbacks automaticos.

Usage

```
acep_openrouter(
  texto,
  instrucciones,
  modelo = "openai/gpt-4o-mini",
  api_key = Sys.getenv("OPENROUTER_API_KEY"),
  schema = NULL,
  parse_json = TRUE,
  temperature = 0,
  max_tokens = 2000,
  top_p = 0.2,
  app_name = NULL,
  site_url = NULL,
  use_fallback = FALSE,
  fallback_provider_order = NULL,
  fallback_models = NULL
)
```

Arguments

texto	Texto a analizar. Puede ser una noticia, tweet, documento, etc.
instrucciones	Instrucciones en lenguaje natural que indican al modelo que hacer con el texto. Ejemplo: "Extrae todas las entidades nombradas", "Clasifica el sentimiento".
modelo	Modelo a utilizar con formato "proveedor/modelo". Ejemplos populares: - OpenAI: ‘"openai/gpt-4o-mini"‘ (rapido y economico), ‘"openai/gpt-4o"‘ (potente) - Anthropic: ‘"anthropic/claude-sonnet-4.5"‘, ‘"anthropic/claude-3.5-haiku"‘ - Google: ‘"google/gemini-2.5-flash"‘, ‘"google/gemini-2.0-flash-001"‘ - Meta: ‘"meta-llama/llama-3.3-70b-instruct"‘, ‘"meta-llama/llama-4-maverick:free"‘ - Qwen: ‘"qwen/qwen3-next-80b-a3b-instruct-2509"‘ - DeepSeek: ‘"deepseek/deepseek-chat-v3-0324:free"‘, ‘"deepseek/deepseek-r1:free"‘ Por defecto: ‘"openai/gpt-4o-mini"‘. Ver lista completa: https://openrouter.ai/models
api_key	Clave de API de OpenRouter. Si no se proporciona, busca la variable de entorno ‘OPENROUTER_API_KEY‘. Para obtener una clave: https://openrouter.ai/settings/keys
schema	Esquema JSON que define la estructura de la respuesta. Puede usar ‘acep_gpt_schema()‘ para obtener esquemas predefinidos o crear uno personalizado. Si es ‘NULL‘, usa un esquema simple con campo "respuesta". NOTA: Structured Outputs solo funciona con modelos compatibles (OpenAI GPT-4o+, Fireworks). Para otros modelos, se usara JSON mode basico.
parse_json	Logico. Si ‘TRUE‘ (por defecto), parsea automaticamente el JSON a un objeto R (lista o data frame). Si ‘FALSE‘, devuelve el JSON como string.
temperature	Parametro de temperatura (0-2). Valores bajos (0-0.3) generan respuestas mas deterministas. Valores altos (0.7-1) mas creativas. Por defecto: 0.
max_tokens	Numero maximo de tokens en la respuesta. Por defecto: 2000.
top_p	Parametro top-p para nucleus sampling (0-1). Por defecto: 0.2.
app_name	Nombre de tu aplicacion (opcional). Se muestra en openrouter.ai/activity.
site_url	URL de tu aplicacion (opcional). Para estadisticas en OpenRouter.
use_fallback	Logico. Si ‘TRUE‘, OpenRouter usara modelos alternativos si el proveedor primario falla. Por defecto: FALSE (sin fallbacks).
fallback_provider_order	Vector opcional de slugs de proveedores para forzar un orden especifico de enrutamiento (ej.: ‘c("openai", "anthropic")‘). Requiere ‘use_fallback = TRUE‘ para habilitar intentos sucesivos.
fallback_models	Vector opcional de modelos alternativos (en formato ‘"proveedor/modelo"‘) que se probaran en orden si el modelo principal devuelve un error recuperable (429, 5xx, timeouts). Ideal para definir variantes pagas cuando la version ‘:free‘ alcance su limite.

Details

OpenRouter abstrae las diferencias entre proveedores, mapeando automaticamente los parametros a la interfaz nativa de cada modelo. Los parametros no soportados por un modelo son ignorados silenciosamente. Esto permite usar la misma funcion para cualquier modelo sin preocuparse por las especificidades de cada API.

Cuando 'use_fallback = TRUE', la funcion configura el objeto 'provider' de OpenRouter para conservar la resiliencia ante errores transitorios y, si se define 'fallback_models', intenta llamar secuencialmente a cada modelo alternativo ante codigos recuperables (429, 5xx, timeouts). Esto evita depender del campo 'route', ya deprecado en la API.

Para Structured Outputs estrictos, recomendamos usar modelos OpenAI (gpt-4o+) o Fireworks. Otros modelos intentaran seguir el esquema pero sin garantias estrictas.

Value

Si 'parse_json=TRUE', devuelve una lista o data frame con la respuesta estructurada segun el esquema. Si 'parse_json=FALSE', devuelve un string JSON.

MODELOS COMPATIBLES Y PROBADOS (ACTUALIZADO)

OpenAI - Familia GPT-5 (Ultima Generacion):

- `openai/gpt-5` - Modelo principal GPT-5
- `openai/gpt-5-pro` - Version Pro, maxima precision
- `openai/gpt-5-mini` - Version mini, economica
- `openai/gpt-5-nano` - Version nano, ultrarrapida
- `openai/gpt-5-chat` - Optimizado para chat

OpenAI - Familia GPT-4.1:

- `openai/gpt-4.1` - Modelo principal GPT-4.1
- `openai/gpt-4.1-mini` - Version mini
- `openai/gpt-4.1-nano` - Version nano

OpenAI - Familia GPT-4:

- `openai/gpt-4o` - GPT-4 optimizado
- `openai/gpt-4o-mini` - Economico, rapido, ideal para produccion
- `openai/gpt-4-turbo` - Version turbo
- `openai/gpt-4` - Modelo base GPT-4

OpenAI - Familia GPT-3.5:

- `openai/gpt-3.5-turbo` - Version turbo, economica

OpenAI - Modelos OSS (Open Source Style):

- `openai/gpt-oss-120b` - Modelo grande (120B parametros)
- `openai/gpt-oss-120b:exacto` - Version exacta
- `openai/gpt-oss-20b` - Modelo pequeno (20B parametros)
- `openai/gpt-oss-20b:free` - Version gratuita

xAI - Familia Grok 4 (Ultima Generacion):

- `x-ai/grok-4` - Modelo principal Grok 4
- `x-ai/grok-4-fast` - Version rapida, optimizada

xAI - Familia Grok 3:

- `x-ai/grok-3` - Modelo principal Grok 3
- `x-ai/grok-3-mini` - Version mini, economica
- `x-ai/grok-3-beta` - Version beta
- `x-ai/grok-3-mini-beta` - Version mini beta

DeepSeek - Familia V3 (Ultima Generacion):

- `deepseek/deepseek-v3.2-exp` - Version experimental 3.2
- `deepseek/deepseek-v3.1-terminus` - Version terminus 3.1
- `deepseek/deepseek-v3.1-terminus:exacto` - Version terminus exacta

DeepSeek - Familia R1 (Razonamiento):

- `deepseek/deepseek-r1-0528` - Version R1 de mayo 2028
- `deepseek/deepseek-r1` - Modelo principal R1
- `deepseek/deepseek-r1:free` - Version R1 gratuita

DeepSeek - R1 Distilled (Basado en Llama):

- `deepseek/deepseek-r1-distill-llama-70b` - R1 destilado en Llama 70B
- `deepseek/deepseek-r1-distill-llama-70b:free` - Version gratuita

DeepSeek - Chat y Otros:

- `deepseek/deepseek-chat` - Optimizado para chat

DeepCogito - Modelos Especializados:

- `deepcogito/cogito-v2-preview-deepseek-671b` - Modelo cogito 671B basado en DeepSeek

Meta Llama - Familia 4:

- `meta-llama/llama-4-maverick` - Llama 4 Maverick

Meta Llama - Familia 3.3:

- `meta-llama/llama-3.3-70b-instruct` - Version de pago
- `meta-llama/llama-3.3-70b-instruct:free` - Version gratuita

Meta Llama - Familia 3.2:

- `meta-llama/llama-3.2-90b-vision-instruct` - Con capacidades de vision

Meta Llama - Familia 3.1:

- `meta-llama/llama-3.1-405b-instruct` - Modelo grande 405B
- `meta-llama/llama-3.1-70b-instruct` - Modelo mediano 70B

NousResearch - Modelos Hermes:

- `nousresearch/hermes-3-llama-3.1-405b` - Hermes 3 basado en Llama 405B

Mistral AI - Familia Magistral:

- `mistralai/magistral-medium-2506` - Magistral medium
- `mistralai/magistral-medium-2506:thinking` - Con razonamiento extendido

Mistral AI - Otros Modelos:

- `mistralai/mistral-large-2407` - Mistral large
- `mistralai/mixtral-8x22b-instruct` - Mixtral 8x22B (MoE)

MoonshotAI - Familia Kimi K2:

- `moonshotai/kimi-k2` - Modelo principal Kimi K2
- `moonshotai/kimi-k2-0905` - Version 09/05
- `moonshotai/kimi-k2-0905:exacto` - Version exacta
- `moonshotai/kimi-k2-thinking` - Con razonamiento extendido

Qwen - Familia Qwen3 (235B - Modelos Grandes):

- `qwen/qwen3-235b-a22b-2507` - Modelo grande 235B (version 2507)
- `qwen/qwen3-235b-a22b` - Modelo grande 235B
- `qwen/qwen3-235b-a22b-thinking-2507` - Con razonamiento extendido
- `qwen/qwen3-max` - Version maxima

Qwen - Familia Qwen3 (30B-80B - Modelos Medianos):

- `qwen/qwen3-next-80b-a3b-instruct` - Modelo next 80B
- `qwen/qwen3-32b` - Modelo 32B
- `qwen/qwen3-30b-a3b` - Modelo 30B
- `qwen/qwen3-30b-a3b-instruct-2507` - Version instruct 2507
- `qwen/qwen3-30b-a3b:free` - Version 30B gratuita

Qwen - Familia Qwen3 (Modelos Pequenos):

- `qwen/qwen3-14b:free` - Modelo 14B gratuito
- `qwen/qwen3-4b:free` - Modelo 4B gratuito, ultrarrapido

Qwen - Familia Qwen 2.5:

- `qwen/qwen-2.5-72b-instruct` - Modelo 2.5 generacion anterior
- `qwen/qwen-plus` - Version plus

Google Gemini - Familia 2.5 (Ultima Generacion):

- `google/gemini-2.5-flash` - Rapido, ultima generacion
- `google/gemini-2.5-pro` - Mayor precision, ultima generacion
- `google/gemini-2.5-flash-lite` - Ultrarrapido, ligero
- `google/gemini-2.5-flash-preview-09-2025` - Preview version septiembre
- `google/gemini-2.5-flash-lite-preview-09-2025` - Preview lite septiembre

Google Gemini - Familia 2.0:

- `google/gemini-2.0-flash-001` - Version estable 2.0
- `google/gemini-2.0-flash-lite-001` - Version ligera 2.0

Google Gemini - Familia 1.5:

- `google/gemini-pro-1.5` - Version anterior, estable

Anthropic Claude - Familia Haiku (Rapidos y Economicos):

- `anthropic/claude-3.5-haiku` - Version 3.5, muy rapido
- `anthropic/claude-3-haiku` - Version 3, economico
- `anthropic/claude-haiku-4.5` - Ultima version, mas preciso

Anthropic Claude - Familia Sonnet (Equilibrados):

- `anthropic/claude-3.5-sonnet` - Popular, buen balance
- `anthropic/claude-3.7-sonnet` - Version mejorada
- `anthropic/claude-3.7-sonnet:thinking` - Con razonamiento extendido
- `anthropic/claude-sonnet-4` - Generacion 4
- `anthropic/claude-sonnet-4.5` - Ultima version, mas preciso

Anthropic Claude - Familia Opus (Maxima Precision):

- `anthropic/claude-3-opus` - Version 3, muy preciso
- `anthropic/claude-opus-4` - Generacion 4
- `anthropic/claude-opus-4.1` - Ultima version disponible

Importante: los modelos etiquetados como ‘:free‘ operan con cuotas comunitarias y suelen estar sometidos a limites de tasa estrictos por parte de OpenRouter. Es frecuente recibir respuestas HTTP 429 (Too Many Requests) cuando la demanda supera la cuota disponible; este codigo indica que el proveedor rechazo la peticion para proteger la infraestructura compartida. Si ocurre, espera unos segundos y reintenta, o selecciona la variante de pago equivalente (sin sufijo ‘:free‘) o activa ‘use_fallback‘ para que OpenRouter cambie automaticamente a un modelo disponible.

Examples

```
## Not run:
# Configurar API key
Sys.setenv(OPENROUTER_API_KEY = "tu-api-key")

# Usar GPT-4o mini (rapido y economico)
texto <- "El SUTEBA convoco a un paro en Buenos Aires el 15 de marzo."
resultado <- acep_openrouter(texto, "Extrae las entidades nombradas",
                             modelo = "openai/gpt-4o-mini",
                             schema = acep_gpt_schema("extraccion_entidades"))

# Comparar con Claude
resultado_claude <- acep_openrouter(texto, "Extrae las entidades nombradas",
                                    modelo = "anthropic/claude-sonnet-4.5",
                                    schema = acep_gpt_schema("extraccion_entidades"))

# Usar modelo gratuito
resultado_free <- acep_openrouter(texto, "Clasifica el sentimiento",
                                  modelo = "meta-llama/llama-4-maverick:free",
                                  schema = acep_gpt_schema("sentimiento"))

# Definir fallback hacia variantes pagas o proveedores alternativos
resultado_resiliente <- acep_openrouter(
  texto,
```

```
  "Extrae las entidades nombradas",
  modelo = "meta-llama/llama-4-maverick:free",
  schema = acep_gpt_schema("extraccion_entidades"),
  use_fallback = TRUE,
  fallback_models = c("meta-llama/llama-4-maverick", "openai/gpt-4o-mini")
)


## End(Not run)
```

`acep_pipeline` *Pipeline completo de análisis de conflictividad*

Description

Ejecuta un flujo de trabajo completo de análisis de texto que incluye: limpieza opcional, conteo de menciones de un diccionario, y cálculo de intensidad. Esta función encadena automáticamente las funciones ‘pipe_clean()‘, ‘pipe_count()‘ y ‘pipe_intensity()‘ para facilitar análisis rápidos.

Usage

```
acep_pipeline(texto, dic, clean = TRUE, ...)
```

Arguments

`texto`	Vector de caracteres con los textos a analizar.
`dic`	Vector de caracteres con las palabras del diccionario de conflictividad (o cualquier otro diccionario temático) a buscar en los textos.
`clean`	Lógico. Si ‘TRUE‘ (por defecto), aplica limpieza y normalización al texto antes del análisis usando ‘acep_clean()‘.
`...`	Argumentos adicionales para pasar a ‘acep_clean()‘ cuando ‘clean = TRUE‘. Por ejemplo: ‘rm_stopwords = TRUE‘, ‘rm_num = TRUE‘, ‘tolower = TRUE‘.

Value

Objeto de clase ‘acep_result‘ con tipo ‘"intensidad"‘ que contiene:

- `id`: Identificadores de cada texto
- `texto`: Textos analizados (limpios si ‘clean = TRUE‘)
- `frecuencia`: Número de menciones del diccionario por texto
- `n_palabras`: Número total de palabras por texto
- `intensidad`: Índice normalizado de intensidad (frecuencia/n_palabras)

Examples

```
## Not run:
# Pipeline completo con limpieza
textos <- c("El SUTEBA va al paro por mejoras salariales",
            "SOIP en lucha contra despidos")
dic_conflictos <- c("paro", "lucha", "reclamo", "protesta")
resultado <- acep_pipeline(textos, dic_conflictos,
                           clean = TRUE, rm_stopwords = TRUE)
print(resultado)

# Pipeline sin limpieza
resultado <- acep_pipeline(textos, dic_conflictos, clean = FALSE)

## End(Not run)
```

`acep_plot_rst` *Resumen visual de la serie temporal de los indices de conflictividad.*

Description

Función que devuelve un panel visual de cuatro gráficos de barras con variables proxy de los indices de conflictividad agrupados por segmento de tiempo.

Usage

```
acep_plot_rst(datos, tagx = "horizontal")
```

Arguments

`datos`	data frame con datos procesados.
`tagx`	orientación de las etiquetas del eje x ('horizontal' | 'vertical').

Value

Si todas las entradas son correctas, la salida sera una imagen de cuatro paneles.

Examples

```
datos <- acep_bases$rp_procesada
datos_procesados_anio <- acep_sst(datos, st = 'anio')
acep_plot_rst(datos_procesados_anio, tagx = 'vertical')
```

acep_plot_st *Gráfico de barras de la serie temporal de indices de conflictividad.*

Description

Función que devuelve un gráfico de barras con la serie temporal de indices de conflictividad por dia, mes o anio.

Usage

```
acep_plot_st(
  x,
  y,
  t = "",
  ejex = "",
  ejey = "",
  etiquetax = "horizontal",
  color = "mint"
)
```

Arguments

x	vector de valores del eje x (por ejemplo, fechas).
y	vector de valores numéricos del eje y (por ejemplo, menciones).
t	titulo del gráfico.
ejex	nombre del eje x.
ejey	nombre del eje y.
etiquetax	orientación de las etiquetas del eje x ('horizontal' | 'vertical').
color	color de las barras.

Value

Si todas las entradas son correctas, la salida sera una imagen de un panel.

Examples

```
datos <- acep_bases$rp_procesada
dpa <- acep_sst(datos, st = 'anio')
acep_plot_st(
dpa$st, dpa$frecm,
t = 'Evoluci\u00f3n de la conflictividad en el sector pesquero argentino',
ejex = 'A\u00f1os analizados',
ejey = 'Menciones de t\u00e9rminos del diccionario de conflictos',
etiquetax = 'horizontal')
```

acep_postag *Etiquetado POS adaptativo con optimizaciones avanzadas*

Description

Version optimizada de acep_postag que se adapta automaticamente al tamano del input. Implementa procesamiento por lotes (chunking) para grandes volumenes, cache de geocodificacion para evitar consultas repetidas, y estrategias de procesamiento adaptativas segun la cantidad de textos. Puede procesar desde 10 hasta millones de textos de forma eficiente.

Usage

```
acep_postag(
  texto,
  core = "es_core_news_lg",
  bajar_core = TRUE,
  inst_spacy = FALSE,
  inst_miniconda = FALSE,
  inst_reticulate = FALSE,
  chunk_size = 1000,
  geocode_cache_file = "geocode_cache.json",
  use_cache = TRUE,
  show_progress = TRUE
)
```

Arguments

texto	Vector de caracteres con los textos a procesar.
core	Idioma del modelo de etiquetado POS del paquete spacyr. Opciones disponibles: 'es_core_news_sm', 'es_core_news_md', 'es_core_news_lg' (espanol), 'pt_core_news_sm', 'pt_core_news_md', 'pt_core_news_lg' (portugues), 'en_core_web_sm', 'en_core_web_md', 'en_core_web_lg', 'en_core_web_trf' (ingles). Default: "es_core_news_lg".
bajar_core	Parametro booleano que define si descargar o no el modelo de etiquetado POS. Default: TRUE.
inst_spacy	Parametro booleano que define si instalar o no spacy (Python). Default: FALSE.
inst_miniconda	Parametro booleano que define si instalar o no miniconda. Default: FALSE.
inst_reticulate	Parametro booleano que define si instalar o no el paquete reticulate. Default: FALSE.
chunk_size	Tamano de los lotes para procesamiento chunking. Ajustar segun RAM disponible: 500 para sistemas con 2-4 GB RAM, 1000 para 8 GB RAM (default), 2000-5000 para 16+ GB RAM. Default: 1000.
geocode_cache_file	Ruta del archivo JSON para guardar cache de geocodificacion. Permite evitar consultas repetidas a la API de Nominatim y compartir cache entre proyectos. Default: "geocode_cache.json".

use_cache	Parametro booleano que activa/desactiva el sistema de cache de geocodificacion. Desactivar para forzar re-geocodificacion de todas las ubicaciones. Default: TRUE.
show_progress	Parametro booleano que controla la visualizacion de mensajes de progreso durante el procesamiento. Util para operaciones largas. Default: TRUE.

Details

La funcion implementa dos estrategias de procesamiento automaticas:

- **Batch Processing** (<= 100 textos): Procesa todos los textos en una sola llamada para maxima velocidad.
- **Chunking** (> 100 textos): Divide los textos en lotes del tamano especificado en chunk_size para controlar el uso de memoria y permitir procesamiento de grandes volumenes.

El sistema de cache de geocodificacion guarda las coordenadas de ubicaciones ya consultadas en formato JSON, evitando consultas repetidas a la API de Nominatim (que tiene limite de 1 req/seg). Esto puede reducir el tiempo de procesamiento en 50-90

Para datasets muy grandes (>100,000 textos), se recomienda procesar en lotes usando la funcion auxiliar proporcionada en los ejemplos y guardar resultados incrementalmente.

Value

Lista con seis elementos en formato tabular:

- texto_tag: Data frame con tokens etiquetados (POS, lemas, dependencias, etc.)
- texto_tag_entity: Data frame con entidades nombradas consolidadas
- texto_only_entity: Data frame con solo las entidades extraidas
- texto_only_entity_loc: Data frame con entidades de tipo LOC geocodificadas (lat/long)
- texto_nounphrase: Data frame con frases nominales consolidadas
- texto_only_nounphrase: Data frame con solo las frases nominales extraidas

Examples

```
## Not run:
# Ejemplo basico con pocos textos
textos <- c(
  "En Mar del Plata el SOIP declara la huelga en demanda de aumento salarial.",
  "La manifestacion se realizo en Buenos Aires el 15 de marzo.",
  "El presidente visito Cordoba para inaugurar la nueva planta."
)
resultado <- acep_postag(texto = textos, bajar_core = FALSE)
head(resultado$texto_tag)

# Ejemplo con dataset mediano y configuracion personalizada
resultado <- acep_postag(
  texto = mis_1000_textos,
  bajar_core = FALSE,
  chunk_size = 500,
```

```
  geocode_cache_file = "cache/ubicaciones_argentina.json",
  use_cache = TRUE
)

# Ver ubicaciones geocodificadas
head(resultado$texto_only_entity_loc)

# Procesamiento incremental para datasets muy grandes
procesar_incremental <- function(textos, batch_size = 10000) {
  dir.create("resultados", showWarnings = FALSE)
  n_batches <- ceiling(length(textos) / batch_size)

  for (i in 1:n_batches) {
    start_idx <- (i - 1) * batch_size + 1
    end_idx <- min(i * batch_size, length(textos))
    batch <- textos[start_idx:end_idx]

    resultado <- acep_postag(
      texto = batch,
      bajar_core = FALSE,
      chunk_size = 2000,
      use_cache = TRUE,
      geocode_cache_file = "cache_global.json"
    )

    saveRDS(resultado, sprintf("resultados/batch_%04d.rds", i))
    message(sprintf("Batch %d/%d completado", i, n_batches))
  }
}

# Usar funcion incremental
procesar_incremental(mis_millones_de_textos, batch_size = 10000)

# Ver contenido del cache
cache <- jsonlite::read_json("geocode_cache.json", simplifyVector = TRUE)
print(paste("Ubicaciones en cache:", nrow(cache)))

## End(Not run)
```

---

acep_postag_hibrido *Etiquetado POS, lematizacion y extraccion de entidades con spacyr*

Description

Realiza analisis linguistico completo de textos usando la biblioteca spaCy a traves de spacyr. Incluye: etiquetado POS (Part-of-Speech), lematizacion, tokenizacion, extraccion de entidades nombradas, frases nominales y geocodificacion de ubicaciones. La funcion procesa automaticamente grandes volumenes de texto dividiendolos en lotes (chunks) y soporta procesamiento paralelo para acelerar el analisis

Usage

```
acep_postag_hibrido(
  texto,
  core = "es_core_news_lg",
  bajar_core = TRUE,
  inst_spacy = FALSE,
  inst_miniconda = FALSE,
  inst_reticulate = FALSE,
  chunk_size = 1000,
  parallel_chunks = FALSE,
  n_cores = NULL,
  geocode_cache_file = "geocode_cache.json",
  use_cache = TRUE,
  show_progress = TRUE
)
```

Arguments

texto	Vector de caracteres con los textos a analizar.
core	Modelo de lenguaje de spaCy a utilizar. Opciones: ‘"es_core_news_sm"‘, ‘"es_core_news_md"‘, ‘"es_core_news_lg"‘ (espanol), ‘"en_core_web_sm"‘, ‘"en_core_web_md"‘, ‘"en_core_web_lg"‘ (ingles), ‘"pt_core_news_sm"‘, ‘"pt_core_news_md"‘, ‘"pt_core_news_lg"‘ (portugues). Por defecto: ‘"es_core_news_lg"‘.
bajar_core	Logico. Si ‘TRUE‘, descarga automaticamente el modelo si no esta instalado.
inst_spacy	Logico. Si ‘TRUE‘, instala la biblioteca spaCy en el entorno Python.
inst_miniconda	Logico. Si ‘TRUE‘, instala Miniconda (necesario para spaCy).
inst_reticulate	Logico. Si ‘TRUE‘, instala el paquete reticulate de R.
chunk_size	Numero de textos a procesar por lote. Valores mas bajos consumen menos memoria pero tardan mas. Por defecto: 1000.
parallel_chunks	Logico. Si ‘TRUE‘, procesa los lotes en paralelo usando multiples nucleos del CPU. Requiere los paquetes ‘future‘ y ‘furrr‘. Por defecto: ‘FALSE‘.
n_cores	Numero de nucleos de CPU a usar en modo paralelo. Si es ‘NULL‘, detecta automaticamente el numero de nucleos disponibles menos uno.
geocode_cache_file	Ruta al archivo JSON donde se almacena el cache de geocodificacion para evitar consultas repetidas. Por defecto: ‘"geocode_cache.json"‘.
use_cache	Logico. Si ‘TRUE‘, usa y actualiza el cache de geocodificacion.
show_progress	Logico. Si ‘TRUE‘, muestra mensajes de progreso en la consola.

Value

Lista con 6 data frames que contienen diferentes niveles de analisis:

- texto_tag: Tokenizacion completa con etiquetas POS, lemas, dependencias sintacticas y atributos morfologicos para cada token
- texto_tag_entity: Tokens con entidades nombradas consolidadas (ej: "Mar del Plata" como una sola entidad en lugar de 3 tokens separados)
- texto_only_entity: Solo las entidades nombradas extraidas (personas, organizaciones, ubicaciones, fechas, etc.)
- texto_only_entity_loc: Entidades de tipo ubicacion (LOC) con coordenadas geograficas (latitud/longitud) obtenidas mediante geocodificacion
- texto_nounphrase: Tokens con frases nominales consolidadas
- texto_only_nounphrase: Solo las frases nominales extraidas

Examples

```
## Not run:
# Analisis basico de un texto
texto <- "El SUTEBA convoco a un paro en Mar del Plata el 15 de marzo."
resultado <- acep_postag_hibrido(texto)

# Ver tokens con etiquetas POS
head(resultado$texto_tag)

# Ver entidades nombradas
print(resultado$texto_only_entity)

# Ver ubicaciones geocodificadas
print(resultado$texto_only_entity_loc)

# Procesar multiples textos con procesamiento paralelo
textos <- c("Primera noticia sobre conflictos.",
            "Segunda noticia sobre protestas.",
            "Tercera noticia sobre reclamos.")
resultado <- acep_postag_hibrido(textos,
                                 parallel_chunks = TRUE,
                                 chunk_size = 100)

## End(Not run)
```

---

acep_process_chunks *Procesamiento de textos en lotes para optimizar memoria*

Description

Divide un vector grande de textos en lotes (chunks) mas pequenos y los procesa secuencialmente aplicando una funcion de ACEP. Esta estrategia permite analizar corpus extensos (millones de documentos) sin superar la capacidad de memoria RAM disponible. La funcion combina automaticamente los resultados de todos los lotes.

Usage

```
acep_process_chunks(
  texto,
  funcion,
  chunk_size = 1000,
  show_progress = TRUE,
  ...
)
```

Arguments

texto	Vector de caracteres con los textos a procesar.
funcion	Funcion de ACEP a aplicar a cada lote. Ejemplos: 'acep_clean', 'acep_token', 'acep_count', 'acep_upos', etc. Debe ser una funcion que acepte un vector de textos como primer argumento.
chunk_size	Numero de textos por lote. Valores mas bajos reducen el consumo de memoria pero aumentan el tiempo total de procesamiento. Por defecto: 1000.
show_progress	Logico. Si 'TRUE', muestra mensajes informativos sobre el progreso del procesamiento (que lote se esta procesando). Por defecto: 'TRUE'.
...	Argumentos adicionales que se pasan directamente a la funcion especificada en el parametro 'funcion'. Ejemplo: si 'funcion = acep_clean', puede pasar 'rm_stopwords = TRUE', 'tolower = TRUE', etc.

Value

El tipo de resultado depende de la funcion aplicada:

- Si la funcion retorna un vector, devuelve un vector combinado
- Si la funcion retorna un data frame, devuelve un data frame combinado (rbind)
- Si la funcion retorna una lista, devuelve una lista de listas

Examples

```
## Not run:
# Procesar 10,000 textos con limpieza en lotes de 1000
textos_limpios <- acep_process_chunks(
  texto = corpus_grande,
  funcion = acep_clean,
  chunk_size = 1000,
  rm_stopwords = TRUE
)

# Tokenizar corpus masivo
tokens <- acep_process_chunks(
  texto = corpus_masivo,
  funcion = acep_token,
  chunk_size = 500,
  tolower = TRUE
```

```
)

# Contar menciones en corpus grande
diccionario <- c("paro", "huelga", "protesta")
frecuencias <- acep_process_chunks(
  texto = corpus_grande,
  funcion = acep_count,
  chunk_size = 2000,
  dic = diccionario
)

## End(Not run)
```

`acep_prompt_gpt` *Colección de instrucciones para GPT.*

Description

Colección de instrucciones para interactuar con los modelos de OpenAI. Las instrucciones fueron testeadas en el marco de las tareas que realizamos en el Observatorio de Conflictividad Social de la Universidad Nacional de Mar del Plata.

Usage

```
data(acep_prompt_gpt)
```

Format

Es un objeto de clase 'list' con 4 componentes.

instruccion_breve_sao_es es un texto en castellano con instrucciones breves para extraer eventos de protesta y codificarlos con las siguientes claves: 'fecha', 'sujeto', 'accion', 'objeto', 'lugar'.

instruccion_larga_sao_es es un texto en castellano con instrucciones largas para extraer eventos de protesta y codificarlos con las siguientes claves: 'id', 'cronica', 'fecha', 'sujeto', 'organizacion', 'participacion', 'accion', 'objeto', 'lugar'.

instruccion_breve_sao_en es un texto en inglés con instrucciones breves para extraer eventos de protesta y codificarlos con las siguientes claves: 'date', 'subject', 'action', 'object', 'place'.

instruccion_larga_sao_en es un texto en inglés con instrucciones largas para extraer eventos de protesta y codificarlos con las siguientes claves: 'id', 'chronicle', 'date', 'subject', 'organization', 'participation', 'action', 'object', 'place'.

Examples

```
prompt01 <- acep_prompt_gpt$instruccion_larga_sao_es
prompt01
```

acep_regex_cache_size *Consultar tamaño del caché de regex*

Description

Devuelve el número de patrones regex almacenados actualmente en el caché interno de ‘acep_count()‘. Cada diccionario único genera una entrada en el caché.

Usage

```
acep_regex_cache_size()
```

Value

Número entero con la cantidad de patrones en caché.

Examples

```
# Ver cuántos patrones hay en caché
acep_regex_cache_size()
```

acep_result *Constructor de resultados de analisis*

Description

Crea un objeto de clase ‘acep_result‘ que encapsula los resultados de un analisis de texto realizado con funciones de ACEP. Este objeto proporciona metodos especializados para visualizacion (‘plot()‘), resumen (‘summary()‘) y conversion a data frame (‘as.data.frame()‘).

Usage

```
acep_result(data, tipo = "general", metadata = NULL)
```

Arguments

data	Data frame con los resultados del analisis.
tipo	Tipo de resultado que contiene el objeto. Valores comunes: ‘"frecuencia"‘, ‘"intensidad"‘, ‘"svo"‘, ‘"serie_temporal"‘, ‘"general"‘. Este parametro determina el comportamiento de los metodos de impresion y visualizacion.
metadata	Lista opcional con informacion sobre el analisis realizado (ej: diccionario utilizado, parametros aplicados, corpus de origen).

Value

Objeto de clase 'acep_result' con la siguiente estructura:

- `data`: Data frame con los resultados del analisis
- `tipo`: Etiqueta del tipo de resultado
- `metadata`: Informacion adicional del analisis
- `fecha_creacion`: Timestamp de creacion del objeto

Examples

```
# Crear resultado de analisis de frecuencias
datos <- data.frame(
  texto = c("El SUTEBA va al paro", "SOIP protesta"),
  frecuencia = c(5, 3)
)
resultado <- acep_result(datos, tipo = "frecuencia")
print(resultado)
summary(resultado)

# Convertir a data frame
df <- as.data.frame(resultado)
```

`acep_rs` *Cadenas de caracteres para limpiar y normalizar textos.*

Description

Cadenas de caracteres y expresiones regulares para limpiar y normalizar textos.

Usage

```
data(acep_rs)
```

Format

Son cadenas de caracteres.

sw1 es un string de palabras vacias.

sw1 es un string de palabras vacias.

dias es un string de dias.

meses es un string de meses.

emoji es un string con expresiones regulares para emojis.

sintildes es un string de letras sin tildes.

tildes es un string de letras con tildes.

punt es un string de puntuación.

num es una expresión regular para números.

hashtag es una expresión regular para hashtag.

espacios es una expresión regular para espacios.

saltos es una expresión regular para saltos de línea.

url es una expresión regular para urls.

users es una expresión regular para usuarixs.

Examples

```
print(acep_rs)
```

`acep_sst` *Serie temporal de índices de conflictividad.*

Description

Función que devuelve los indices de conflictividad agrupados por segmento de tiempo: 'dia', 'mes', 'anio'. Esta función viene a reemplazar a acep_rst. Simplifica los parámetros.

Usage

```
acep_sst(datos, st = "mes", u = 2, d = 4)
```

Arguments

`datos`	data frame con las variables 'fecha' (en formato Date), 'n_palabras' (numérica), conflictos' (numérica), 'intensidad' (numérica). Las ultimas tres se pueden construir en un solo paso con la función 'acep_db' o en tres pasos con las funciones 'acep_frec', 'acep_men', 'acep_int'.
`st`	parámetro para establecer el segmento temporal a ser agrupado: 'anio', 'mes', 'dia'.
`u`	umbral de menciones para contabilizar una nota como nota que refiere a un conflicto, por defecto tiene 2 pero se puede modificar.
`d`	cantidad de decimales, por defecto tiene 4 pero se puede modificar.

Value

Si todas las entradas son correctas, la salida sera una base de datos en formato tabular con nuevas variables.

Examples

```
datos <- acep_bases$rp_procesada
head(datos)
datos_procesados_anio <- acep_sst(datos, st='anio', u=4)
datos_procesados_mes <- acep_sst(datos)
datos_procesados_dia <- acep_sst(datos, st ='dia', d=3)
head(datos_procesados_anio)
head(datos_procesados_mes)
head(datos_procesados_dia)
```

---

acep_svo *Función para extraer tripletes SVO (Sujeto-Verbo-Objeto).*

Description

Función que devuelve seis objetos data.frame con etiquetado POS (modelo spacyr) y relaciones sintácticas (modelo rsyntax) que permiten reconstruir estructuras sintácticas como SVO y Sujeto-Predicado. Una vez seleccionadas las notas periodísticas referidas a conflictos, esta función permite extraer sujetos de la protesta, acción realizada y objeto(s) de la acción. También devuelve entidades nombradas (NER).

Usage

```
acep_svo(acep_tokenindex, prof_s = 3, prof_o = 3, u = 1)
```

Arguments

acep_tokenindex	data.frame con el etiquetado POS y las relaciones de dependencia generado con la función acep_postag.
prof_s	es un numero entero positivo que determina la profundidad a la que se buscan las relaciones dentro del sujeto. Este parámetro se hereda del la función children() del paquete {rsyntax}. Se recomienda no superar el valor 2.
prof_o	es un numero entero positivo que determina la profundidad a la que se buscan las relaciones dentro del objeto. Este parámetro se hereda del la función children() del paquete {rsyntax}. Se recomienda no superar el valor 2.
u	numero entero que indica el umbral de palabras del objeto en la reconstrucción SVO.

Value

Si todas las entradas son correctas, la salida sera una lista con tres bases de datos en formato tabular.

Source

Dependencias Universales para taggeo POS

Sobre el paquete rsyntax

References

Welbers, K., Atteveldt, W. van, & Kleinnijenhuis, J. 2021. Extracting semantic relations using syntax: An R package for querying and reshaping dependency trees. Computational Communication Research, 3-2, 1-16. doi:10.5117/CCR2021.2.003.WELB

Examples

```
## Not run:
acep_svo(acep_bases$spacy_postag)

## End(Not run)
```

`acep_together` *Interaccion con modelos de IA usando TogetherAI*

Description

Funcion para interactuar con modelos de IA a traves de la API de TogetherAI. TogetherAI proporciona acceso a modelos open-source de alta calidad como Llama, Qwen, Mistral, DeepSeek y muchos otros. Soporta JSON mode para respuestas estructuradas. La API es compatible con el formato de OpenAI, lo que facilita la integracion.

Usage

```
acep_together(
  texto,
  instrucciones,
  modelo = "meta-llama/Meta-Llama-3.1-70B-Instruct-Turbo",
  api_key = Sys.getenv("TOGETHER_API_KEY"),
  schema = NULL,
  parse_json = TRUE,
  temperature = 0,
  max_tokens = 2000,
  top_p = 0.2,
  top_k = 50,
  repetition_penalty = 1,
  stop = NULL,
  prompt_system = "json"
)
```

Arguments

`texto`	Texto a analizar. Puede ser una noticia, tweet, documento, etc.
`instrucciones`	Instrucciones en lenguaje natural que indican al modelo que hacer con el texto. Ejemplo: "Extrae todas las entidades nombradas", "Clasifica el sentimiento".

`modelo` Modelo a utilizar. Ejemplos populares: - Moonshot: ‘"moonshotai/Kimi-K2-Instruct-0905"‘ (128K context) - Meta Llama: ‘"meta-llama/Meta-Llama-3.1-405B-Instruct-Turbo"‘, ‘"meta-llama/Llama-3.3-70B-Instruct-Turbo"‘ - Qwen: ‘"Qwen/Qwen2.5-72B-Instruct-Turbo"‘, ‘"Qwen/QwQ-32B-Preview"‘ - Mistral: ‘"mistralai/Mixtral-8x22B-Instruct-v0.1"‘, ‘"mistralai/Mistral-7B-Instruct-v0.3"‘ - DeepSeek: ‘"deepseek-ai/DeepSeek-V3"‘, ‘"deepseek-ai/DeepSeek-R1"‘ - Google: ‘"google/gemma-2-27b-it"‘, ‘"google/gemma-2-9b-it"‘ Por defecto: ‘"meta-llama/Meta-Llama-3.1-70B-Instruct-Turbo"‘. Ver lista completa: https://docs.together.ai/docs/chat-models

`api_key` Clave de API de TogetherAI. Si no se proporciona, busca la variable de entorno ‘TOGETHER_API_KEY‘. Para obtener una clave: https://api.together.xyz/settings/api-keys

`schema` Esquema JSON que define la estructura de la respuesta. Puede usar ‘acep_gpt_schema()‘ para obtener esquemas predefinidos o crear uno personalizado. Si es ‘NULL‘, usa un esquema simple con campo "respuesta". NOTA: TogetherAI soporta JSON mode con `response_format: {type: "json_object"}` para modelos compatibles. Consulta la lista de modelos soportados en: https://docs.together.ai/docs/json-mode

`parse_json` Logico. Si ‘TRUE‘ (por defecto), parsea automaticamente el JSON a un objeto R (lista o data frame). Si ‘FALSE‘, devuelve el JSON como string.

`temperature` Parametro de temperatura (0-2). Valores bajos (0-0.3) generan respuestas mas deterministas. Valores altos (0.7-1) mas creativas. Por defecto: 0.

`max_tokens` Numero maximo de tokens en la respuesta. Por defecto: 2000.

`top_p` Parametro top-p para nucleus sampling (0-1). Por defecto: 0.2.

`top_k` Parametro top-k para muestreo. Limita las opciones a los k tokens mas probables. Por defecto: 50. Usar 0 o -1 para desactivar.

`repetition_penalty` Penalizacion por repeticion de tokens (0.1-2.0). Valores > 1 penalizan repeticiones. Por defecto: 1.

`stop` Secuencias de parada opcionales. Vector de strings que detienen la generacion. Por defecto: NULL.

`prompt_system` Prompt del sistema que define el comportamiento del modelo. Opciones: - ‘"json"‘ (por defecto): Usa un prompt estructurado que instruye al modelo a responder SOLO en formato JSON siguiendo el esquema proporcionado. Agrega `response_format: {type: "json_object"}` - ‘"texto"‘: Usa un prompt simple para respuestas en texto plano sin estructura. Elimina automaticamente el contenido de pensamiento (<think>...</think>) de modelos como Qwen3-Thinking - String personalizado: Cualquier texto que definas como prompt del sistema

Details

Sobre TogetherAI:

TogetherAI es una plataforma especializada en modelos open-source que ofrece: - Precios competitivos y modelos gratuitos - Alta velocidad de inferencia optimizada - Acceso a modelos de ultima generacion (Llama, Qwen, DeepSeek, etc.) - API compatible con formato OpenAI

JSON Mode:

La funcion utiliza JSON mode de TogetherAI para obtener respuestas estructuradas. Cuando 'prompt_system = "json"', la funcion: 1. Incluye el esquema JSON en el prompt del sistema (REQUERIDO por TogetherAI) 2. Agrega `response_format: {type: "json_object"}` al body de la peticion 3. Instruye explicitamente al modelo a responder SOLO en JSON

Esta combinacion de esquema textual + `response_format` asegura respuestas JSON validas y consistentes en cada llamada.

Modelos compatibles con JSON mode:

Los modelos mas recientes que soportan JSON mode incluyen: - Qwen3, Qwen2.5 (Instruct, Coder, VL, Thinking) - DeepSeek-R1, DeepSeek-V3 - Meta Llama 3.1, 3.3, 4 - Mistral 7B Instruct - Google Gemma

Ver lista completa: https://docs.together.ai/docs/json-mode

Validaciones:

La funcion incluye validacion de limite de tokens. Si la respuesta es truncada por 'max_tokens', devuelve un mensaje claro indicando que se necesitan mas tokens.

Value

Si 'parse_json=TRUE', devuelve una lista o data frame con la respuesta estructurada segun el esquema. Si 'parse_json=FALSE', devuelve un string JSON.

Examples

```
## Not run:
# Configurar API key
Sys.setenv(TOGETHER_API_KEY = "tu-api-key")

# Usar Llama 3.1 70B (rapido y potente)
texto <- "El SUTEBA convoco a un paro en Buenos Aires el 15 de marzo."
resultado <- acep_together(texto, "Extrae las entidades nombradas",
                           modelo = "meta-llama/Meta-Llama-3.1-70B-Instruct-Turbo",
                           schema = acep_gpt_schema("extraccion_entidades"))

# Usar Qwen para analisis de sentimiento
resultado_qwen <- acep_together(texto, "Clasifica el sentimiento",
                                modelo = "Qwen/Qwen2.5-72B-Instruct-Turbo",
                                schema = acep_gpt_schema("sentimiento"))

# Usar DeepSeek-V3
resultado_ds <- acep_together(texto, "Analiza el texto",
                              modelo = "deepseek-ai/DeepSeek-V3",
                              schema = acep_gpt_schema("clasificacion"))

# Usar Moonshot Kimi con 128K context
resultado_kimi <- acep_together(texto, "Resume el texto",
                                modelo = "moonshotai/Kimi-K2-Instruct-0905",
                                schema = acep_gpt_schema("resumen"))

# Usar modo texto plano (sin estructura JSON)
```

```
resultado_texto <- acep_together(texto, "Resume este texto en una frase",
                                 modelo = "meta-llama/Meta-Llama-3.1-70B-Instruct-Turbo",
                                  prompt_system = "texto",
                                  parse_json = FALSE)
print(resultado_texto)  # Devuelve string de texto plano

# Usar prompt del sistema personalizado
resultado_custom <- acep_together(
  texto,
  "Analiza el sentimiento",
  modelo = "Qwen/Qwen2.5-72B-Instruct-Turbo",
  prompt_system = paste(
    "Eres un experto en analisis de sentimientos politicos.",
    "Se objetivo y neutral."
  ),
  parse_json = FALSE
)

## End(Not run)
```

---

acep_token *Tokenizador.*

Description

Función que tokeniza las notas/textos.

Usage

```
acep_token(x, tolower = TRUE, cleaning = TRUE)
```

Arguments

x	vector de textos al que se le aplica la función de tokenización.
tolower	convierte los textos a minúsculas.
cleaning	hace una limpieza de los textos.

Value

Si todas las entradas son correctas, la salida será un data.frame con las palabras tokenizadas.

Examples

```
acep_token("Huelga de obreros del pescado en el puerto")
```

acep_token_plot *Gráfico de barras de palabras más recurrentes en un corpus.*

Description

Función que devuelve un gráfico de barras con las palabras mas recurrentes en un corpus textual.

Usage

```
acep_token_plot(x, u = 10, frec = TRUE)
```

Arguments

x	vector de palabras tokenizadas.
u	numero de corte para el top de palabras mas frecuentes.
frec	parámetro para determinar si los valores se visualizaran como frecuencia absoluta o relativa.

Value

Si todas las entradas son correctas, la salida sera un gráfico de barras.

Examples

```
tokens <- c(rep("paro",15), rep("piquete",25), rep("corte",20), rep("manifestación",10),
rep("bloqueo",5), rep("alerta",16), rep("ciudad",12), rep("sindicato",11), rep("paritaria",14),
rep("huelga",14), rep("escrache",15))
acep_token_plot(tokens)
```

acep_token_table *Tabla de frecuencia de palabras tokenizadas.*

Description

Función que cuenta la frecuencia de palabras tokenizadas.

Usage

```
acep_token_table(x, u = 10)
```

Arguments

x	vector de palabras tokenizadas.
u	número de corte para el top de palabras más frecuentes.

Value

Si todas las entradas son correctas, la salida sera una tabla con la frecuencia relativa y absoluta de palabras tokenizadas.

Examples

```
tokens <- c(rep("paro",15), rep("piquete",25), rep("corte",20), rep("manifestación",10),
rep("bloqueo",5), rep("alerta",16), rep("ciudad",12), rep("sindicato",11), rep("paritaria",14),
rep("huelga",14), rep("escrache",15))
acep_token_table(tokens)
```

---

acep_upos *Función para etiquetado POS, lematización, tokenización.*

Description

Función que devuelve un marco de datos objetos con etiquetado POS (modelo udpipe) para su posterior procesamiento con la función acep_postag.

Usage

```
acep_upos(texto, modelo = "spanish")
```

Arguments

texto	vector con los textos a procesar.
modelo	idioma del modelo de etiquetado POS del paquete udpipe.

Value

Si todas las entradas son correctas, la salida sera un marco de datos con 17 variables.

Source

Dependencias Universales para taggeo POS

Sobre el modelo UDPipe

Sobre el paquete rsyntax

References

Welbers, K., Atteveldt, W. van, & Kleinnijenhuis, J. 2021. Extracting semantic relations using syntax: An R package for querying and reshaping dependency trees. Computational Communication Research, 3-2, 1-16. doi:10.5117/CCR2021.2.003.WELB

Examples

```
## Not run:
texto <- "El SOIP declara la huelga en demanda de aumento salarial."
acep_upos(texto)

## End(Not run)
```

`pipe_clean` *Limpieza de texto en pipeline*

Description

Aplica limpieza y normalización de texto dentro de un flujo pipeline. Esta función actúa como adaptador de ‘acep_clean()‘ para trabajar con objetos ‘acep_corpus‘, registrando las transformaciones aplicadas.

Usage

```
pipe_clean(corpus, ...)
```

Arguments

`corpus`	Objeto ‘acep_corpus‘ o vector de caracteres. Si se pasa un vector, se crea automáticamente un objeto ‘acep_corpus‘.
`...`	Argumentos para ‘acep_clean()‘. Ejemplos: ‘rm_stopwords = TRUE‘, ‘rm_num = TRUE‘, ‘tolower = TRUE‘, ‘rm_punt = TRUE‘.

Value

Objeto ‘acep_corpus‘ con el campo ‘texto_procesado‘ actualizado y registro de la transformación en ‘procesamiento$limpieza‘.

Examples

```
# Crear corpus y limpiar
textos <- c("El SUTEBA va al paro!!!", "SOIP protesta 123")
corpus <- acep_corpus(textos)
corpus_limpio <- pipe_clean(corpus, rm_punt = TRUE, rm_num = TRUE)
print(corpus_limpio)
```

pipe_count *Conteo de menciones en pipeline*

Description

Cuenta las menciones de palabras de un diccionario dentro de un flujo pipeline. Esta función extrae los textos de un ‘acep_corpus‘ (procesados o originales) y aplica ‘acep_count()‘ para detectar ocurrencias del diccionario.

Usage

```
pipe_count(corpus, dic, ...)
```

Arguments

corpus	Objeto ‘acep_corpus‘. Debe ser un corpus válido creado con ‘acep_corpus()‘ o resultado de ‘pipe_clean()‘.
dic	Vector de caracteres con las palabras del diccionario a buscar.
...	Argumentos adicionales (actualmente no utilizados, reservado para futuras extensiones).

Value

Objeto ‘acep_result‘ con tipo ‘"frecuencia"‘ que contiene un data frame con: ‘id‘, ‘texto‘ y ‘frecuencia‘ de menciones por texto.

Examples

```
# Contar menciones en corpus
textos <- c("El SUTEBA va al paro", "SOIP en lucha y paro")
corpus <- acep_corpus(textos)
diccionario <- c("paro", "lucha", "protesta")
resultado <- pipe_count(corpus, diccionario)
print(resultado)
```

pipe_intensity *Cálculo de intensidad en pipeline*

Description

Calcula el índice de intensidad normalizado dentro de un flujo pipeline. La intensidad se define como la proporción de menciones del diccionario respecto al total de palabras: intensidad = frecuencia / n_palabras.

Usage

```
pipe_intensity(result, decimales = 4)
```

Arguments

result	Objeto ‘acep_result‘ que debe contener una columna ‘frecuencia‘. Típicamente proviene de ‘pipe_count()‘.
decimales	Número de decimales para redondear el índice de intensidad. Por defecto: 4.

Value

Objeto ‘acep_result‘ con tipo ‘"intensidad"‘ que incluye columnas adicionales: ‘n_palabras‘ e ‘intensidad‘.

Examples

```
# Calcular intensidad desde resultado de conteo
textos <- c("El SUTEBA va al paro", "SOIP en lucha y paro")
corpus <- acep_corpus(textos)
diccionario <- c("paro", "lucha")
resultado <- pipe_count(corpus, diccionario)
resultado_intensidad <- pipe_intensity(resultado, decimales = 4)
print(resultado_intensidad)
```

---

pipe_timeseries *Generación de series temporales en pipeline*

Description

Crea agregaciones temporales de índices de conflictividad dentro de un flujo pipeline. Agrupa los resultados por segmentos temporales (día, mes, año) y calcula estadísticas resumidas usando ‘acep_sst()‘.

Usage

```
pipe_timeseries(data, st = "mes", u = 2, d = 4)
```

Arguments

data	Data frame o objeto ‘acep_result‘ que contenga columnas: ‘fecha‘ (o variable temporal), ‘n_palabras‘, ‘conflictos‘, ‘intensidad‘.
st	Segmento temporal para agrupar. Valores: ‘"dia"‘, ‘"mes"‘, ‘"anio"‘. Por defecto: ‘"mes"‘.
u	Umbral para calcular métricas categóricas. Por defecto: 2.
d	Número de decimales para redondear. Por defecto: 4.

Value

Objeto ‘acep_result‘ con tipo ‘"serie_temporal"‘ que contiene agregaciones por período temporal.

Examples

```
## Not run:
# Crear serie temporal desde data frame con fechas
data <- data.frame(
  fecha = as.Date(c("2024-01-15", "2024-01-20", "2024-02-10")),
  n_palabras = c(100, 150, 120),
  conflictos = c(5, 8, 6),
  intensidad = c(0.05, 0.053, 0.05)
)
serie <- pipe_timeseries(data, st = "mes", u = 2)
print(serie)

## End(Not run)
```

Index