

# Package ‘ulex’

July 22, 2025

**Title** Unique Location Extractor

**Version** 0.1.0

**Description** Extracts coordinates of an event location from text based on dictionaries of landmarks, roads, and areas. Only returns the location of an event of interest and ignores other location references; for example, if determining the location of a road traffic crash from the text “crash near [location 1] heading towards [location 2]”, only the coordinates of “location 1” would be returned. Moreover, accounts for differences in spelling between how a user references a location and how a location is captured in location dictionaries.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**Imports** dplyr, tidyr, readr, purrr, tidytext, stringr, stringi, ngram, hunspell, stringdist, tm, raster, parallel, sf, quanteda, geodist, spacyr, utils

**URL** <https://dime-worldbank.github.io/ulex/>

**NeedsCompilation** no

**Author** Robert Marty [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-3164-3813>>)

**Maintainer** Robert Marty <[rmarty@worldbank.org](mailto:rmarty@worldbank.org)>

**Repository** CRAN

**Date/Publication** 2024-06-17 18:20:02 UTC

## Contents

|                             |          |
|-----------------------------|----------|
| augment_gazetteer . . . . . | 2        |
| locate_event . . . . .      | 6        |
| <b>Index</b>                | <b>9</b> |

---

|                   |                                    |
|-------------------|------------------------------------|
| augment_gazetteer | <i>Augments Landmark Gazetteer</i> |
|-------------------|------------------------------------|

---

## Description

Augments Landmark Gazetteer

## Usage

```
augment_gazetteer(
  landmarks,
  landmarks.name_var = "name",
  landmarks.type_var = "type",
  grams.min_words = 3,
  grams.max_words = 6,
  grams.skip_gram_first_last_word_match = TRUE,
  grams.add_only_if_name_new = FALSE,
  grams.add_only_if_specific = FALSE,
  types_rm = c("route", "road", "toilet", "political", "locality", "neighborhood",
    "area", "section of populated place"),
  types_rm.except_with_type = c("flyover", "round about", "roundabout"),
  types_rm.except_with_name = c("flyover", "round about", "roundabout"),
  parallel.sep_slash = TRUE,
  parallel.rm_begin = c(tm::stopwords("en"), c("near", "at", "the", "towards", "near")),
  parallel.rm_end = c("bar", "shops", "restaurant", "sports bar", "hotel", "bus station"),
  parallel.word_diff = "default",
  parallel.word_diff_iftype = list(list(words = c("stage", "bus stop", "bus station"),
    type = "transit_station")),
  parallel.rm_begin_iftype = NULL,
  parallel.rm_end_iftype = list(list(words = c("stage", "bus stop", "bus station"), type
    = "transit_station")),
  parallel.word_begin_addtype = NULL,
  parallel.word_end_addtype = list(list(words = c("stage", "bus stop", "bus station"),
    type = "stage")),
  parallel.add_only_if_name_new = FALSE,
  parallel.add_only_if_specific = FALSE,
  rm.contains = c("road", "rd"),
  rm.name_begin = c(tm::stopwords("en"), c("near", "at", "the", "towards", "near")),
  rm.name_end = c("highway", "road", "rd", "way", "ave", "avenue", "street", "st"),
  pos_rm.all = c("ADJ", "ADP", "ADV", "AUX", "CCONJ", "INTJ", "NUM", "PRON", "SCONJ",
    "VERB", "X"),
  pos_rm.except_type = list(pos = c("NOUN", "PROPN"), type = c("bus", "restaurant",
    "bank"), name = ""),
  close_thresh_km = 1,
  quiet = TRUE
)
```

**Arguments**

|                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>landmarks</code>                             | sf spatial points data.frame of landmarks.                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>landmarks.name_var</code>                    | Name of variable indicating name of landmark. (Default: "name").                                                                                                                                                                                                                                                                                                                                                                         |
| <code>landmarks.type_var</code>                    | Name of variable indicating type of landmark. (Default: "type").                                                                                                                                                                                                                                                                                                                                                                         |
| <code>grams.min_words</code>                       | Minimum number of words in name to make n/skip-grams out of name. (Default: 3).                                                                                                                                                                                                                                                                                                                                                          |
| <code>grams.max_words</code>                       | Maximum number of words in name to make n/skip-grams out of name. Setting a cap helps to reduce spurious landmarks that may come out of really long names. (Default: 6).                                                                                                                                                                                                                                                                 |
| <code>grams.skip_gram_first_last_word_match</code> | For skip-grams, should first and last word be the same as the original word? (Default: TRUE).                                                                                                                                                                                                                                                                                                                                            |
| <code>grams.add_only_if_name_new</code>            | When creating new landmarks based on n- and skip-grams, only add an additional landmark if the name of the landmark is new; i.e., the name doesn't already exist in the gazetteer. (Default: FALSE).                                                                                                                                                                                                                                     |
| <code>grams.add_only_if_specific</code>            | When creating new landmarks based on n- and skip-grams, only add an additional landmark if the name of the landmark represents a specific location. A specific location is a location where most landmark entries with the same name are close together (within <code>close_thresh_km</code> kilometers). (Default: FALSE).                                                                                                              |
| <code>types_rm</code>                              | If landmark has one of these types, remove - unless <code>types_rm.except_with_type</code> or <code>types_rm.except_with_name</code> prevents removing. (Default: <code>c("route", "road", "toilet", "political", "locality", "neighborhood", "area", "section of populated place")</code> ).                                                                                                                                            |
| <code>types_rm.except_with_type</code>             | Landmark types to always keep. This parameter only becomes relevant in cases where a landmark has more than one type. If a landmark has both a <code>"types_rm"</code> and a <code>"types_always_keep"</code> landmark, this landmark will be kept. (Default: <code>c("flyover", "round about", "roundabout")</code> ).                                                                                                                  |
| <code>types_rm.except_with_name</code>             | Landmark names to always keep. This parameter only becomes relevant in cases where a landmark is one of <code>"types_rm"</code> . Here, we keep the landmark if <code>"names_always_keep"</code> is somewhere in the name. For example, if the landmark is a road but has flyover in the name, we may want to keep the landmark as flyovers are small spatial areas. (Default: <code>c("flyover", "round about", "roundabout")</code> ). |
| <code>parallel.sep_slash</code>                    | If a landmark contains a slash, create new landmarks before and after the slash. (Default: TRUE).                                                                                                                                                                                                                                                                                                                                        |

|                                            |                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>parallel.rm_begin</code>             | If a landmark name begins with one of these words, add a landmark that excludes the word. (Default: <code>c(tm::stopwords("en"), c("near", "at", "the", "towards", "near"))</code> ).                                                                                                                                                                                                                                    |
| <code>parallel.rm_end</code>               | If a landmark name ends with one of these words, add a landmark that excludes the word. (Default: <code>c("bar", "shops", "restaurant", "sports bar", "hotel", "bus station")</code> ).                                                                                                                                                                                                                                  |
| <code>parallel.word_diff</code>            | If the landmark includes one of these words, add a landmark that swaps the word for the other word (e.g., "center" with "centre"). By default, uses a set collection of words. Users can also manually specify different word versions. Input should be a data.frame with the following variables: <code>version_1</code> (for one spelling of the word) and <code>version_2</code> (for a second spelling of the word). |
| <code>parallel.word_diff_iftype</code>     | If the landmark includes one of these words, add a landmark that swaps the word for the other word (e.g., "bus stop" with "bus station"). Enter a named list of words, with <code>words = c()</code> and <code>type = c()</code> . (Default: <code>list(list(words = c("stage", "bus stop", "bus station"), type = "transit_station"))</code> ).                                                                         |
| <code>parallel.rm_begin_iftype</code>      | If a landmark name begins with one of these words, add a landmark that excludes the word if the landmark is a certain type. (Default: <code>NULL</code> ).                                                                                                                                                                                                                                                               |
| <code>parallel.rm_end_iftype</code>        | If a landmark name ends with one of these words, add a landmark that excludes the word if the landmark is a certain type. (Default: <code>list(list(words = c("stage", "bus stop", "bus station"), type = "transit_station"))</code> ).                                                                                                                                                                                  |
| <code>parallel.word_begin_addtype</code>   | If the landmark begins with one of these words, add the type. For example, if landmark is "restaurant", this indicates the landmark is a restaurant. Adding the "restaurant" to landmark ensures that the type is reflected. (Default: <code>NULL</code> ).                                                                                                                                                              |
| <code>parallel.word_end_addtype</code>     | If the landmark ends with one of these words, add the type. For example, if landmark is "X stage", this indicates the landmark is a bus stage. Adding the "stage" to landmark ensures that the type is reflected. (Default: <code>list(list(words = c("stage", "bus stop", "bus station"), type = "stage"))</code> ).                                                                                                    |
| <code>parallel.add_only_if_name_new</code> | When creating parallel landmarks using the above parameters, only add an additional landmark if the name of the landmark is new; i.e., the name doesn't already exist in the gazetteer. (Default: <code>FALSE</code> ).                                                                                                                                                                                                  |
| <code>parallel.add_only_if_specific</code> | When creating parallel landmarks using the above parameters, only add an additional landmark if the name of the landmark represents a specific location. A specific location is a location where most landmark entries with the same name are close together (within <code>close_thresh_km</code> kilometers). (Default: <code>FALSE</code> ).                                                                           |
| <code>rm.contains</code>                   | Remove the landmark if it contains one of these words. Implemented after N/skip-grams and parallel landmarks are added. (Default: <code>c("road", "rd")</code> ).                                                                                                                                                                                                                                                        |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rm.name_begin      | Remove the landmark if it begins with one of these words. Implemented after N/skip-grams and parallel landmarks are added. (Default: <code>c(tm::stopwords("en"), c("near", "at", "the", "towards", "near"))</code> ).                                                                                                                                                                                                                                                                                                                                                                                     |
| rm.name_end        | Remove the landmark if it ends with one of these words. Implemented after N/skip-grams and parallel landmarks are added. (Default: <code>c("highway", "road", "rd", "way", "ave", "avenue", "street", "st")</code> ).                                                                                                                                                                                                                                                                                                                                                                                      |
| pos_rm.all         | Part-of-speech categories to remove. Part-of-speech determined by Spacy. (Default: <code>c("ADJ", "ADP", "ADV", "AUX", "CCONJ", "INTJ", "NUM", "PRON", "SCONJ", "VERB", "X")</code> ).                                                                                                                                                                                                                                                                                                                                                                                                                     |
| pos_rm.except_type | When specify part-of-speech categories to remove in <code>pos_rm.all</code> , when to override <code>pos_rm.all</code> and keep the word. Names list with: (1) <code>pos</code> (if the word is also another type of part-of-speech); (2) <code>type</code> (if the word is also a certain type of place); and (3) <code>name</code> (if the word includes certain text). Example: <code>list(pos = c("NOUN", "PROPN"), type = c("bus", "restaurant", "bank"), name = c("parliament"))</code> . (Default: <code>list(pos = c("NOUN", "PROPN"), type = c("bus", "restaurant", "bank"), name = "")</code> ). |
| close_thresh_km    | When to consider locations close together. Used when determining if a landmark name with multiple locations are specific (close together) or general (far apart). (Default: 1).                                                                                                                                                                                                                                                                                                                                                                                                                            |
| quiet              | Print progress of function. (Default: TRUE).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

## Value

sf spatial point data.frame of landmarks.

## Examples

```
library(ulx)
library(spacyr)
spacy_install()

lm_sf <- data.frame(name = c("white house",
                           "the world bank group",
                           "the george washington university"),
                  lat = c(38.897778,
                        38.89935,
                        38.9007),
                  lon = c(-77.036389,
                        -77.04275,
                        -77.0508),
                  type = c("building", "building", "building")) |>
sf::st_as_sf(coords = c("lon", "lat"),
             crs = 4326)

lm_aug_sf <- augment_gazetteer(lm_sf)
```

---

locate\_event

*Locate Event*


---

## Description

Locate Event

## Usage

```
locate_event(
  text,
  landmark_gazetteer,
  landmark_gazetteer.name_var = "name",
  landmark_gazetteer.type_var = "type",
  roads,
  roads.name_var = "name",
  areas,
  areas.name_var = "name",
  event_words,
  prepositions_list = list(c("at", "next to", "around", "just after", "opposite", "opp",
    "apa", "hapa", "happened at", "just before", "at the", "outside", "right before"),
    c("near", "after", "toward", "along", "towards", "approach"), c("past", "from",
    "on")),
  junction_words = c("intersection", "junction"),
  false_positive_phrases = "",
  type_list = NULL,
  clost_dist_thresh = 500,
  fuzzy_match = TRUE,
  fuzzy_match.min_word_length = c(5, 11),
  fuzzy_match.dist = c(1, 2),
  fuzzy_match.ngram_max = 3,
  fuzzy_match.first_letters_same = TRUE,
  fuzzy_match.last_letters_same = TRUE,
  quiet = TRUE,
  mc_cores = 1
)
```

## Arguments

`text` Vector of texts to be geolocated.

`landmark_gazetteer` `sf` spatial data.frame representing landmarks.

`landmark_gazetteer.name_var` Name of variable indicating name of landmark.

`landmark_gazetteer.type_var` Name of variable indicating type of landmark.

|                                             |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>roads</code>                          | sf spatial data.frame representing roads.                                                                                                                                                                                                                                                                                                                                                                         |
| <code>roads.name_var</code>                 | Name of variable indicating name of road.                                                                                                                                                                                                                                                                                                                                                                         |
| <code>areas</code>                          | sf spatial data.frame representing areas, such as administrative areas or neighborhoods.                                                                                                                                                                                                                                                                                                                          |
| <code>areas.name_var</code>                 | Name of variable indicating name of area.                                                                                                                                                                                                                                                                                                                                                                         |
| <code>event_words</code>                    | Vector of event words, representing events to be geocoded.                                                                                                                                                                                                                                                                                                                                                        |
| <code>prepositions_list</code>              | List of vectors of prepositions. Order of list determines order of preposition precedence. (Default: <code>list(c("at", "next to", "around", "just after", "opposite", "opp", "apa", "hapa", "happened at", "just before", "at the", "outside", "right before"), c("near", "after", "toward", "along", "towards", "approach"), c("past", "from", "on"))</code> ).                                                 |
| <code>junction_words</code>                 | Vector of junction words to check for when determining intersection of roads. (Default: <code>c("intersection", "junction")</code> ).                                                                                                                                                                                                                                                                             |
| <code>false_positive_phrases</code>         | Common words found in text that include spurious location references (eg, <b>githurai bus</b> is the name of a bus, but <b>githurai</b> is also a place). These may be common phrases that should be checked and ignored in the text. (Default: <code>""</code> ).                                                                                                                                                |
| <code>type_list</code>                      | List of vectors of types. Order of list determines order or type precedence. (Default: <code>NULL</code> ).                                                                                                                                                                                                                                                                                                       |
| <code>closest_dist_thresh</code>            | Distance (meters) as to what is considered "close"; for example, when considering whether a landmark is close to a road. (Default: <code>500</code> ).                                                                                                                                                                                                                                                            |
| <code>fuzzy_match</code>                    | Whether to implement fuzzy matching of landmarks using levenstein distance. (Default: <code>TRUE</code> ).                                                                                                                                                                                                                                                                                                        |
| <code>fuzzy_match.min_word_length</code>    | Minimum word length to use for fuzzy matching; vector length must be the same as <code>fuzzy_match.dist</code> . (Default: <code>c(5,11)</code> ).                                                                                                                                                                                                                                                                |
| <code>fuzzy_match.dist</code>               | Allowable levenstein distances for fuzzy matching; vector length must be same as <code>fuzzy_match.min_word_length</code> . (Default: <code>c(1,2)</code> ).                                                                                                                                                                                                                                                      |
| <code>fuzzy_match.ngram_max</code>          | The number of n-grams that should be extracted from text to calculate a levensteing distance against landmarks. For example, if the text is composed of 5 words: <code>w1 w2 w3 w4</code> and <code>fuzzy_match.ngram_max = 3</code> , the function extracts <code>w1 w2 w3</code> and compares the levenstein distance to all landmarks. Then in checks <code>w2 w3 w4</code> , etc. (Default: <code>3</code> ). |
| <code>fuzzy_match.first_letters_same</code> | When implementing a fuzzy match, should the first letter of the original and found word be the same? (Default: <code>TRUE</code> ).                                                                                                                                                                                                                                                                               |
| <code>fuzzy_match.last_letters_same</code>  | When implementing a fuzzy match, should the last letter of the original and found word be the same? (Default: <code>TRUE</code> ).                                                                                                                                                                                                                                                                                |
| <code>quiet</code>                          | If <code>FALSE</code> , prints text that is being geocoded. (Default: <code>TRUE</code> ).                                                                                                                                                                                                                                                                                                                        |



# Index

augment\_gazetteer, [2](#)

locate\_event, [6](#)