
# Rosette Entity Extractor Application Developer's Guide

## Java Edition

7.53.3.c67.0

Publication date 2022-06-23


Basis Technology Corp.
1060 Broadway
Somerville, MA 02144
T 617.386.2000
F 617.386.2020
E support@rosette.com
http://support.rosette.com

# Table of Contents

# 1. Introduction

## 1.1. Overview

Entities are the key actors in your text data: the organizations, people, locations, products, and dates mentioned in documents. Rosette uncovers these entities, delivering structure, clarity, and insight to your data with adaptability, easy deployment, and consistent accuracy and performance across a broad range of languages [5] and text genres.

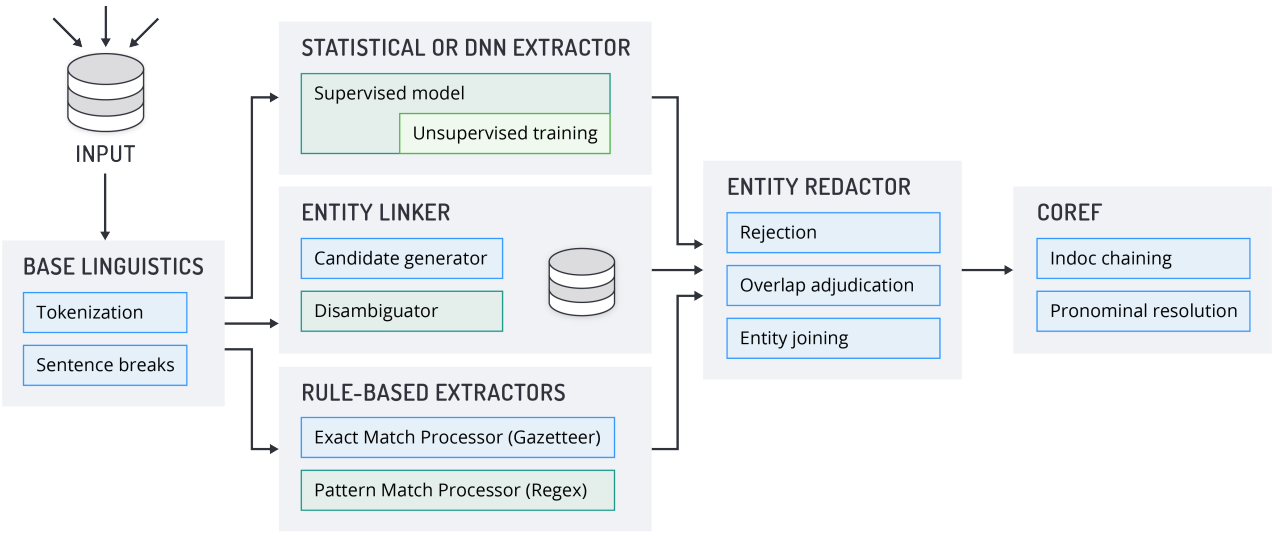
Rosette Entity Extractor (REX) ingests text and identifies people, locations, and organizations, in addition to many other entity types including product, date/time, URL, and email. These entities can be used to add structured metadata to a document or in downstream natural language processing (NLP) tasks, such as extracting themes and ideas, sentiment analysis, and relationship extraction.

**Entity Extraction** REX comes with multiple entity extraction processors along with a linker processor to link entities to a knowledge base. In case of conflicting entities, a redactor decides which entity extraction result “wins.” REX has extensive customization features, including adding new entity patterns to the pattern-matching processor and new entity lists to the exact match processor. You can add a custom processor [29] to systematically process REX results. Numerous configuration settings [11] let you fit REX to your specific use case.

**Entity Linking** REX has an entity linking processor which can identify the real-world entities extracted from the text as well as disambiguating between different entities with the same name. Entity linking can determine not only that "Tim Cook" is a person, but it can also determine *who* "Tim Cook" is and disambiguate between multiple possibilities. For example, is he the CEO of Apple or a political science scholar? The entity linking processor looks at the context of each extracted entity to link entities against Wikidata. REX supports linking to other public knowledge bases as well as your organization's custom knowledge bases.

**Adaptation & Customization** REX gives you a good start, but as with any natural language processor, Basis Technology assumes you will need to adapt REX to your specific task for best results. There is a field training kit (FTK) to optimize REX’s performance on your specific data or to add new entity types to the statistical model. The statistical model is context-sensitive, meaning it identifies entities based on the context it appears in and thus can find names of people even if the name has been misspelled. See Customizing the Statistical Models with the FTK [42]. The FTK also enables you to perform entity linking against your own entity knowledge base [34].

## 1.2. Architecture




Basic Entity Extraction with REX:

1. Using Rosette base linguistics, REX processes plain text input into sentences and tokens.
2. Entities are extracted by running the tokens through the statistical processor or DNN, regexes, and gazetteers. If the linker is enabled, the tokens are also run through the linker processor to link entities to a knowledge base.
3. Reject regexes and gazetteers may remove entities from the output. Some adjacent entities may be combined by the joiner into a single result. The final entities are selected by running the extractor results through the redactor.
4. The final extracted entities are returned as output.

### 1.2.1. Processors for Entity Extraction

REX uses multiple complementary methods to identify entity mentions in the input text: statistical models, pattern matching, and exact matching. With REX version 7.32, we added a deep neural network model which is currently in beta. Pattern-matching and exact matching processors can run in parallel with the statistical or the deep neural network processors, but the statistical and deep neural network processors cannot be used simultaneously.

- **Statistical Processor:** The statistical processor that uses contextual features of the input to identify entities. Using computational linguistics, it has been trained on a body of annotated news stories to extract a variety of entities in a number of languages.
- **Pattern Matching Processor (regular expressions):** Regular expressions (regexes) are a good way to identify language-specific entities and generic entities that appear in a variety of languages. You can modify the standard regexes that we supply, and add your own regexes.
- **Exact Matching Processor (gazetteers):** Gazetteers (entity lists) return exact matches to a predefined list. The REX distribution includes gazetteers for each language and a number of entity types, and a cross-language gazetteer for corporation names (as the name of the corporation does not generally change when it enters international markets). You can modify the standard gazetteers that we supply and add your own gazetteers to extract new entities or entity types.

- **Deep Neural Network Processor:** This processor uses a model trained using a deep neural network. It is slower than the statistical processor, but has shown an error reduction of about 10% for English and Arabic and 30% for Korean, as measured by F-Score, for extracting person, location, organization, and titles. The model is trained on the same data as the statistical model. The model is based on an LSTM neural network and is backed by the TensorFlow library.

**NOTE**
The deep neural network processor is currently available in English, Arabic, Hebrew, and Korean.

- **Name Classifier Processor:** This processor predicts entity types for text that lacks the syntactic context of complete sentences. It can extract entities from structured text, such as list items and tables, which typically contains text fragments instead of full sentences.

**Redaction:** When two processors return the same or overlapping entities, the redactor [26] chooses an entity based on the length of the competing entity strings. You can also configure the redactor to choose which same-length mention to return based on entity type and/or processor.

### 1.2.2. Processors to Customize Results

These processors run on the extracted entities to further customize the results.

**Joining:** You can use a configuration file and the API to establish rules for joining adjacent entities into one (such as joining titles with personal names).

**Rejections:** You can define regexes and gazetteers to reject entities that otherwise may be returned.

**Indoc Coref:** In a single document, REX chains together mentions that refer to the same entity (i.e., in-document coreference).

### 1.2.3. Optional Functionality

**Linker Processor:** This processor extracts and links entity mentions to a knowledge base of known entities, each with a unique ID. This processor is disabled by default. REX is shipped with a prepackaged default knowledge base linking entity mentions to a Wikidata QID. You can replace the default entity knowledge base with a custom knowledge base [34].

**NOTICE**
Currently, the linker performs its own entity extraction and does NOT use entities found by the default entity extraction processors (statistical, pattern-matching, exact-matching). Therefore, the linker processor's entities will not necessarily match those from the default entity extraction processors.

**Pronominal Resolver:** REX tries to resolve pronouns with their antecedent entities. This processor is disabled by default. The pronominal resolver is only available for English.

**Custom Processor by User:** REX allows the user to write custom processors and insert them [29] into the REX pipeline prior to the input phase or the entity redactor phase.

## 1.3. Standard Entity Types

REX is pre-trained to extract the following entity types.

- **LOCATION**
  - A city, state, country, region, or other location that contains both a population and a government.
  - A geographic place such as a body of water, mountain, park, or address.
  - A structure such as a building or monument.
- **ORGANIZATION**
  - A corporation, institution, government agency, or other group of people defined by an established organizational structure.
- **PERSON**
  - A human identified by name, nickname, or alias.
- **TITLE**
  - Appellation associated with a person by virtue of occupation, office, birth, or as an honorific.
- **NATIONALITY**
  - Reference to a country or region of origin, such as American or Swiss.
- **RELIGION**
  - Reference to an organized religion or theology as well as its followers.
- **IDENTIFIER:CREDIT_CARDNUM**
- **IDENTIFIER:DISTANCE***
- **IDENTIFIER:EMAIL**
- **IDENTIFIER:LATITUDE_LONGITUDE***
- **IDENTIFIER:MONEY**
- **IDENTIFIER:PERSONAL_ID_NUM**
- **IDENTIFIER:PHONE_NUMBER**
- **IDENTIFIER:URL**
- **IDENTIFIER:UTM***
  - Geographical coordinates, expressed with the Universal Transverse Mercator System.
- **TEMPORAL:DATE**
- **TEMPORAL:TIME**

Entity types marked with a * are not returned by default. Activate them by instructing REX to load the supplemental regexes [24] in each language's supplemental directory.

When the call includes `{"options": {"includeDBpediaTypes": true}`, Rosette supports additional top-level entity types and over 700 additional types drawn from the DBpedia [33] ontology. Entity linking must be enabled to return DBpedia entity types.

## 1.4. Adding New Entity Types

There are several ways to train REX to extract entity types beyond the standard set.

1. Create new gazetteers [19] (i.e., entity lists).
2. Create new regexes [23] for entities that fit a pattern, such as telephone numbers.
3. Retrain the statistical processor [42] on your data to adapt it to the syntax and vocabulary of your text and domain with the FTK.

**NOTE**
The FTK is not included with the standard REX distribution but is free to any licensee of REX. Contact support@rosette.com to get the FTK package.

## 1.5. Language Support

The following tables describe the entity types returned by the different REX processors for each supported language.

Key to processor used to identify each entity type:

- **S** = statistical processor
- **G** = exact matching processor (gazetteer)
- **R** = pattern matching processor (regex)
- **L** = entity linking available
- **D** = deep neural network processor

**Statistical, Exact Match (Gazetteer) Extracted Entities, and Linked Entities**

| Language | Entity Type | | | | | | |
|---|---|---|---|---|---|---|---|
| (ISO code) | LOC | ORG | PER | PROD | TTL | NAT | REL |
| Arabic `ara` | S/G/D/L | S/G/D/L | S/D/L | L | S | G | G |
| Chinese, Script-insensitive `zho` | S/G/L | S/G/L | S/L | L | S | G | G |
| Chinese, Simplified `zhs` | S/G/L | S/G/L | S/L | L | S | G | G |
| Chinese, Traditional `zhs` | S/G/L | S/G/L | S/L | L | S | G | G |
| Dutch `nld` | S/L | S/G/L | S/L | L | G | | |
| English `eng` | S/G/L/D | S/R/G/L/D | S/L/D | S/L | S | G | G |
| French `fra` | S/L | S/G/L | S/L | L | S | | |
| German `deu` | S/L | S/G/L | S/L | L | S | | |
| Hebrew `heb` | S/L/D | S/G/L/D | S/L/D | L | | | |

| Language (ISO code) | Entity Type | | | | | | |
|---|---|---|---|---|---|---|---|
| | LOC | ORG | PER | PROD | TTL | NAT | REL |
| Hungarian `hun` | S/G/L | S/G/L | S/G/L | S/L | S | | |
| Indonesian `ind` | S/G/L | S/G/L | S/L | L | | | |
| Italian `ita` | S/L | S/G/L | S/L | L | S | | |
| Japanese `jpn` | S/L | S/G/L | S/L | L | S | G | G |
| Korean `kor` | S/D/L | S/G/D/L | S/D/L | L | S | G | G |
| Malay, Standard `zsm` | S/G/L | S/G/L | S/L | L | | | |
| Pashto `pus` | S/L | S/G/L | S/L | L | S | | |
| Persian `fas` | S/L | S/G/L | S/L | L | G | G | G |
| Portuguese `por` | S/L | S/G/L | S/L | L | S | | |
| Russian `rus` | S/L | S/G/L | S/L | L | S | G | G |
| Spanish `spa` | S/L | S/G/L | S/L | L | S | | |
| Swedish `swe` | S/L | S/G/L | S/L | L | S | S/G | S/G |
| Urdu `urd` | S/L | S/G/L | S/L | L | G | | |
| Vietnamese `vie` | S/L | S/L | S/L | L | G | G | G |

The following entity types are not returned by default

### Rule-based Extracted Entities

| Language (ISO Code) | Entity Type | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | CC# | Dist[a] | EM | LATLNG[a] | MONEY | PERS ID | TEL# | URL | UTM[a] | DATE[a] | TIME |
| Arabic `ara` | R | R | R | R | R | R | R | R | R | R | R |
| Chinese, Script-insensitive `zho` | R | R | R | R | R | R | R | R | R | R | R |
| Chinese, Simplified `zhs` | R | R | R | R | R | R | R | R | R | R | R |
| Chinese, Traditional `zhs` | R | R | R | R | R | R | R | R | R | R | R |
| Dutch `nld` | R | R | R | R | R | R | R | R | R | R | R |
| English `eng` | R | R | R | R | R | R | R | R | R | R | R |
| French `fra` | R | R | R | R | R | R | R | R | R | R | R |
| German `deu` | R | R | R | R | R | R | R | R | R | R | R |
| Hebrew `heb` | R | R | R | R | R | R | R | R | R | R | R |
| Hungarian `hun` | R | R | R | R | R | R | R | R | R | R | R |
| Indonesian `ind` | R | R | R | | R | R | R | R | R | R | R |
| Italian `ita` | R | R | R | R | R | R | R | R | R | R | R |
| Japanese `jpn` | R | R | R | R | R | R | R | R | R | R | R |
| Korean `kor` | S | S/G | S | | S | G | G | | | | |
| Malay, Standard `zsm` | R | R | R | | R | R | R | R | R | R | R |
| Pashto `pus` | R | R | R | R | R | R | R | R | R | R | R |
| Persian `fas` | R | R | R | R | R | R | R | R | R | R | R |
| Portuguese`por` | R | R | R | R | R | R | R | R | R | R | R |
| Russian `rus` | R | R | R | R | R | R | R | R | R | R | R |
| Spanish `spa` | R | R | R | R | R | R | R | R | R | R | R |

| Language (ISO Code) | Entity Type | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | CC# | Dist[a] | EM | LATLNG[a] | MONEY | PERS ID | TEL# | URL | UTM[a] | DATE[a] | TIME |
| Swedish `swe` | R | R | R | R | R | R | R | R | R | R | R |
| Urdu `urd` | R | | R | | R | R | R | R | R | | |
| Vietnamese `vie` | R | R | R | | R | R | R | R | | R | R |

# 2. Getting Started

## 2.1. Minimum System Requirements

**IMPORTANT**
The amount of disk space required depends on your use case and the languages installed.

- x86_64 CPU with 4 or more physical cores
- Minimum 16 GB RAM
- Disk Space
  - Minimum (English only, no linker): 5 GB
  - All languages: 42 GB
- 64-bit JDK 11 or 17 (tested with OpenJDK)
- Ant 1.8.2 or later (optional - required to run included samples)

## 2.2. Installing REX

Your installation of REX will include the following files:

1. The SDK package: `rex-je-<VERSION>.zip`, where `<VERSION>` is the version of REX you are installing, e.g., `rex-je-7.28.1.c59.0.zip`. When you unzip the SDK package, the root directory is `rex-je-<VERSION>`. It contains the following subdirectories:

   **data** — Contains the statistical models, gazetteers, regexes, and configuration files for the redactor and joiner. You can add regex files and text gazetteers to the data tree and edit the configuration files.

   **lib** — Contains the .jar files that the SDK uses. Add these .jar files to your classpath.

   **licenses** — Default location for your Rosette license. The default configuration and samples require `rlp-license.xml` to be in this directory.

   **samples** — Contains sample data and applications to examine, modify, and run. `samples/lib` contains the third-party jar files used by the samples.

**language files** Language files for each language [1]. The files are named `rex-je-7.41.0.c60.0-<LANGUAGE>.zip`, where LANGUAGE is the three-letter ISO 639-3 code indicating the language of the file contents. For example, `rex-je-7.41.0.c60.0-eng.zip` is the file for the English language. The files are unzipped into the SDK's root directory, `rex-je-<VERSION>`.

2. An installer: `rex-je-<VERSION>-installer-zip` to unpackage the SDK package and language packs. Unzip the installer package and run `install-rex.sh` to begin. The installer will prompt you for:

   - The location of the other packages if they're not detected in the current directory.
   - The language packs you want to install. The installer defaults to all detected packs.
   - The installation directory (defaults to current).

   It will then unzip all necessary files to their correct locations.

   **NOTE**
   The installer does not copy in the license file.

3. The Rosette License: `rlp-license.xml`. Copy this file to the `licenses` subdirectory.

4. The documentation package: `rex-je-<VERSION>-doc.zip` When unzipped, the root directory contains a `doc` subdirectory with the following components:

   - REX Application Developer's Guide (this document, `rex-je-appdev-guide.pdf`)
   - Release Notes (`rex-je-<VERSION>-release-notes.pdf`)
   - Java API documentation (`apidocs/index.html`).

Entity linking [32] (the linker processor) [2] is provided within the standard REX SDK package. No other files are required for linking.

You may also receive packages for field training kit [42], to train the statistical processor or add a custom knowledge base.

## 2.3. A Quick Look at REX: Running the Sample Program

After you install REX and the license file, try running the sample application. The sample processes an input file and reports information about each entity that it extracts from the input.

Ant provides arguments for the root directory, a language code (eng), an input file, and an output file. The root directory provides the path to the Rosette license and to the data tree. The data tree includes the statistical model, default gazetteers, default regex files, and a redaction configuration file.

1. In a Bash shell (Unix) or Command Prompt (Windows), navigate to rex-je-<VERSION>/samples.

[1] Language files are packaged individually as of version `rex-je-7.41.0.c60.0`.
[2] As of version `rex-je-7.41.0.c60.0`.

2. Use the Ant build script to compile and run `EntityAnnotatorSample`

```
ant run.EntityAnnotatorSample
```

`EntityAnnotatorSample` instantiates an Annotator to process a UTF-8 input file and report the entities it finds in an output file.

3. The sample reads an input file in rex-je-<VERSION>/samples/data:

```
General George Washington (February 22, 1732 – December 14, 1799) was the
dominant military and political leader of the new United States of America
from 1775 to 1799. He led the American victory over Britain in the American
Revolutionary War as commander in chief of the Continental Army in 1775–1783,
and he presided over the writing of the Constitution in 1787.
```

Source: http://en.wikipedia.org/wiki/George_washington

4. The sample writes output to `rex-je-<VERSION>/samples/EntityAnnotatorSampleOut-eng.txt`. For each entity it finds, the output includes entity type, offsets for the location of the entity in the input document, the normalized entity (1 space between each word), and the source (statistical, regex, gazetteer, or joiner).

```
TITLE, [0, 7), General (statistical)
PERSON, [8, 25), George Washington (statistical)
LOCATION, [124, 148), United States of America (gazetteer:/pathto/data/gazetteer/eng/accept/gaz-LE.bin)
NATIONALITY, [179, 187), American (gazetteer:/pathto/data/gazetteer/eng/accept/gaz-LE.bin)
LOCATION, [201, 208), Britain (gazetteer:/pathto/data/gazetteer/eng/accept/gaz-LE.bin)
NATIONALITY, [216, 224), American (gazetteer:/pathto/data/gazetteer/eng/accept/gaz-LE.bin)
TITLE, [246, 264), commander in chief (statistical)
ORGANIZATION, [272, 288), Continental Army (statistical)
IDENTIFIER:URL, [374, 420), http://en.wikipedia.org/wiki/George_washington (regex:xxx_9)
```

**To process a sample document in a different language**, include the ISO 639-3 Language Code [54] when you call Ant. For example, to process a German document:

```
ant -Dlang=deu run.EntityAnnotatorSample
```

The source for this sample is in `rex-je-<VERSION>/samples/src/EntityAnnotatorSample.java`.

## 2.4. Removing Extra Language Files

The REX release package includes your licensed language models. If you won't be using all of your licensed languages, you can repackage REX so that only desired language models are included.

Files in the following directories are common to all languages and required:

```
bin/*
lib/*
data/etc/*
data/etc/regex/xxx/* (Language-neutral regex)
data/flinx/data/kb/basis/  (Default linker knowledge base, files that aren't listed below are common)
licenses/*
rbl-je/*
```

Each language requires files in the following directories:

```
data/gazetteer/{lang}/accept/gaz-LE.bin
data/regex/{lang}/*
data/statistical/{lang}/*
data/flinx/data/lang-vectors   (Linker vectors for specific languages go here)
data/flinx/data/kb/basis/{lang}  (Default linker knowledge base models for specific languages)
data/flinx/data/kb/basis/etc  (Default linker knowledge base vectors for specific languages go here)
data/flinx/data/kb/basis/{xxxx}-aliases.bin  (Default linker knowledge base data for specific scripts)
```

To create minimal package for the relevant languages, run the python script `bin/repack-rex.py` with python 2.7 or later.

```
% python repack-rex.py
Copyright (c) 2017 Basis Technology Corporation All Rights Reserved.
Support@basistech.com
http://www.basistech.com

This script repacks REX for languages specified in the script arguments.
usage: rex-distro-zip output-zip rlp-license-file lang [lang ...]
To create English-only REX distribution package zip:
  python repack-rex.py downloaded-rex-distro.zip eng-only.zip rlp-license.xml eng
```

# 3. Configuring the Entity Extractor

To run REX, you need to define an entity extractor. The `createDefaultExtractor` method creates an extractor holding all the default data. The license file and data files must be in their default [7] locations under the `rex-je-<VERSION>` directory.

1. Create an extractor using the default configuration.

   ```
   EntityExtractor extractor = EntityExtractor.createDefaultExtractor(rootDirectory);
   ```

2. (Optional) Add your own extraction rules [22].

3. (Optional) Set the processors to be used [29]

4. Use the extractor `createAnnotator(LanguageCode)` method to create an Annotator for processing the input text and extracting entities.

   ```
   extractor.createAnnotator(LanguageCode.ENGLISH);
   ```

## 3.1. Starting with an Empty Extractor

If you do not want to use the default configuration, you can start with an empty extractor and build from there.

1. Create an empty extractor, an extractor with no rules.

   ```
   EntityExtractor extractor = EntityExtractor.createEmptyExtractor();
   ```

2. Designate the Rosette license, by designating the Rosette license file or designating the text contained in the Rosette license file.

```
extractor.setLicense(new File("path/to/rlp-license.xml"));
```

or

```
extractor.setLicense("content-of-rlp-license-file");
```

3. Designate the RBL directory. For some languages, REX requires an RBL root directory to provide data used in segmentation and/or morphological analysis. A suitable installation of RBL is provided as part of the REX distribution, but is not set when using an empty extractor.

```
extractor.setBaseLinguisticsRootDirectory("path/to/RBLroot");
```

4. Add your own extraction rules [22].

5. (Optional) Set the processors to be used [29]

6. Use the extractor `createAnnotator(LanguageCode)` method to create an Annotator for processing the input text and extracting entities.

```
extractor.createAnnotator(LanguageCode.ENGLISH);
```

## 3.2. General Configuration Options

| Parameter | Description | Default |
|---|---|---|
| `rootDirectory` | A REX root directory contains language models and necessary configuration files. | `${rex-root}` |
| `rblRootDirectory` | The directory containing the RBL root for REX to use. | `${rex-root}/rbl-je` |
| `snapToTokenBoundaries` | Regular expressions and gazetteers may be configured to match tokens partially independent from token boundaries. If true, reported offsets correspond to token boundaries. | `true` |
| `maxEntityTokens` | The maximum number of tokens allowed in an entity returned by Statistical Entity Extractor. Entity Redactor discards entities from Statistical Entity Extractor with more than this number of tokens. | `8` |
| `customProcessors` | Custom processors to add to annotators. See Creating Custom Processors [29] for more details on custom processors. | `null` |
| `customProcessorClasses` | Register a custom processor class. | `null` |
| `excludedEntityTypes` | Entity types to be excluded from extraction. | `null` |
| `dataOverlayDirectory` | An overlay directory [13] is a directory shaped like the `data` directory. REX will look for files in both the overlay directory and the root directory, using files from both locations. However, if a file exists in both places (as identified by its path relative to the overlay or root data directory), REX prefers the version in the overlay directory. If REX finds a zero-length file in the overlay directory, it ignores both that file and any corresponding file in the root data directory. | `null` |
| `calculateSalience` | If true, entity chain salience values are calculated. Can be overridden by specifying `calculateSalience` in the API call. | `false` |
| `retainSocialMediaSymbols` | The option to retain social media symbols ('@' and '#') in normalized output | `false` |
| `keepEntitiesInInput` | The option to keep existing annotated text entities. | `false` |
| `structuredRegionsProcessingType` | Configures how structured regions will be processed. | `none` |

## 3.3. Handling Structured Regions

The REX statistical model is trained to extract entities from unstructured text, where the model uses the syntactic context in sentences to help identify entities and entity types. But not all data is unstructured. Often input documents contain some sections of structured text, such as tables and lists, along with the unstructured text. Structured text usually does not contain full sentences and is often missing the syntactic context that REX expects. This can lead to noisy results and false positives.

In addition to sentences and token, the Rosette Base Linguistics (RBL) processor identifies structured and unstructured regions. For structured regions, REX disables the statistical processor. The text in structured regions is still processed by the rule-based processors (gazetteers and regexes) and the linker. Additionally, for some languages, another extractor, the name classifier [18], can extract entities from structured regions of text.

By default, structured regions are processed the same as unstructured regions.

To change how structured text is processed, set the enum `structuredRegionsProcessingType` when configuring the entity extractor. It has three values:

- **none**: Disables the statistical/DNN models from processing structured regions. When set to `none`, REX does not attempt to extract entities from structured regions using the statistical processor or DNN models. The rule-based extractors (gazetteers, regex) and the linker are used to process structured regions.
- **nerModel**: (default) Processes the entire document as unstructured text. Structured regions are processed the same as unstructured regions.
- **nameClassifier**: Disables the statistical/DNN models from processing structured regions and enables the name classifier on the structured regions.

Some structured regions may contain enough syntactic context for the statistical/DNN models to accurately extract entities. You can set a minimum number of tokens required in a structured region to override the structured region processor setting. If the number of tokens in the region exceeds this minimum, the region will be processed with the statistical/DNN models. The default value is 0. With this default, all structured regions are processed as defined by the `structuredRegionsProcessingType`.

```
public REXFactoryConfiguration
EntityExtractor.setstructuredRegionProcessingSentenceTokensMin(Integer tokensMin);
```

## 3.4. Fragment Boundary Detector

**NOTE**
Disabling the fragment boundary detector will classify the entire text as unstructured. This has a similar effect to setting `structuredRegionsProcessingType` to `nerModel`.

REX detects entities within sentences. By default, REX uses a fragment boundary detector to identify structured regions, adding sentence boundaries at tabs, newlines, and multiple whitespace characters (such as 3 or more spaces) in text fragments, such as lists and tables. This enables the detection of multiple entities in text fragments that do not form standard sentences. Consider the following text:

```
George Washington
John Adams
Thomas Jefferson
```

Without the fragment boundary detector, the statistical model identifies the preceding text as a single PERSON entity. With the fragment boundary detector, the statistical model identifies three separate PERSON entities.

Turn off the fragment boundary detector using the `EntityExtractor#setUseFragmentBoundaryDetector(boolean useFragmentBoundaryDetector)` method.

**NOTE**
While the fragment boundary detector improves REX's performance on tables, lists, and other non-prose content, REX is, by design, tuned for prose and may not return high accuracy results on content with significant non-prose elements.

## 3.5. Overlay Data Directory

If your project has a set of unique data files that you would like to keep separate from other data files, you can put them in their own directory, also known as an overlay directory. This is an additional data directory, which takes priority over the default REX data directory.

The overlay directory must have the same directory tree as the provided `data` directory. If an overlay directory is set, REX searches both it and the default `data` directory.

- If a file exists in both places, the version in the overlay directory is used.
- If there is an empty file in the overlay directory, REX will ignore the corresponding file in the default `data` directory.
- If there is no file in the overlay directory, REX will use the file in the default directory.

To specify the overlay directory use:

```
EntityExtractor#setOverlayDataDirectory(Path overlayDataDirectory)
```

**Turn Off a Specific Language Gazetteer**

```
EntityExtractor extractor = EntityExtractor.createDefaultExtractor(new File("/path/to/root"));
// 'my-data' has an empty file at "gazetteer/eng/accept/gaz-LE.bin"
// so 'American' will not be extracted as a Nationality
extractor.setOverlayDataDirectory(Paths.get("my-data"));
String input = "George Washington was an American president.";
Annotator ann = extractor.createAnnotator(LanguageCode.ENGLISH);
AnnotatedText anText = ann.annotate(input);
for (Entity e : anText.getEntities()) {
    System.out.println(e.toString())
}

Output:
Entity{extendedProperties={}, type=PERSON, Mentions=[Mention{extendedProperties={},
startOffset=0, endOffset=17, source=statistical,
subsource=/path/to/root/data/statistical/eng/model-LE.bin, normalized=George Washington}]}
Entity{extendedProperties={}, type=TITLE, mentions=[Mention{extendedProperties={},
startOffset=34, endOffset=43, source=statistical,
subsource=/path/to/root/data/statistical/eng/model-LE.bin, normalized=president}]}
```

The default English gazetteer will not be used in calls.

## 3.6. Entity Salience

REX can return a salience score for each extracted entity. Salience indicates whether the entity is important to the overall scope of the document. Returned salience scores are binary, either 0 (not salient) or 1 (salient). The decision is made according to several parameters, such as frequency, distance from document start, etc. Salience is not calculated by default.

To enable entity salience use the method:

```
EntityExtractor#setCalculateSalience(boolean calculateSalience)

Entity.getSalience()
```

To return the salience value:

```
Entity.getSalience()
```

## 3.7. Invalidating Internal Data Caches

For the most port, REX uses memory-mapping techniques to keep Java heap memory usage low. However, there are certain cases, such as specific hardware configurations or heavy use of dynamic data, where REX's internal caches might cause memory problems. In such cases, REX provides some APIs to invalidate its caches so Java can reclaim the memory.

Cache eviction is currently supported for the statistical and gazetteer processors. A list of currently cached language data for a specific processor can be retrieved with either the `getCachedLanguagesForProcessor()` function in `com.basistech.rosette.rex.EntityExtractor` or `com.basistech.rosette.rex.REXAnnotatorFactory`.

To invalidate the cache:

- For specific language/processor pairs, use `invalidateProcessorCacheForLanguage()`.

- To invalidate all data for specific languages, use `invalidateCacheForLanguage`.
- To invalidate all of REX's internal caches, use `invalidateCache()`.

Invalidating cached data simply removes references from REX's caches so that Java's garbage collector can reclaim the memory. If there are still other references to an annotator for a specific language in the user process, memory won't be freed until those references are also disposed of.

## 3.8. Retrieving Base Linguistics Configuration

REX internally uses Rosette Base Linguistics (RBL) to analyze the text before processing it. If the user application already uses Base Linguistics for other purposes, it's possible to save processing time and have REX annotate pre-toxenized documents by passing REX's annotator `annotate` function a tokenized `AnnotatedText` instance instead of a string. However, if the user's instance of Base Linguistics and REX's internal instance of Base Linguistics are configured differently, REX's results might be affected.

To solve the problem, `EntityExtractor` provides a `getBaseLinguisticsParameters` function that returns the set of Rosette Base Linguistics options REX uses internally, given a language. This function should be called *after* the `EntityExtractor` has been otherwise configured. It returns an `EnumSet` of keys to the values REX configures them to.

**TIP**

REX provides a sample (`rex-je-<version>/samples/RBLParametersSample.java`) which demonstrates how to retrieve RBL parameters from REX and use RBL directly to process documents before running the REX extractor.

# 4. Modifying Entity Extraction Processors

REX provides multiple processors for extracting entities. You can optimize REX for your entity extraction tasks by configuring the processors. Examples of the modifications you can make include:

- Removing one or more processors [15]
- Adding gazetteers [19] or gazetteer entries for selecting or rejecting entities
- Adding regex [23] files or individual regex entries
- Adding custom processors [29]
- Customizing the statistical model with the FTK [42]

Each processor has its own set of parameters to customize its behavior.

## 4.1. Selecting Processors

By default, REX uses all the processors. You can select to use a subset of the processors. For example, you can decide to return only entities extracted by statistical analysis.

REX includes the following processors:

- `statistical`: Entity extractor processor using a statistically-trained model
- `deepNeuralNetwork` Entity extractor processor using a model trained using a deep neural network
- `acceptGazetteer` Rule-based entity extractor based on gazetteers
- `acceptRegex`: Rule-based entity extractor based on regular expressions
- `kbLinker`: Entity extractor based on a knowledge base of known entities
- `redactor` Chooses an entity when multiple processors extract the same or overlapping entities
- `joiner` Joins adjacent entities into a single entity
- `rejectGazetteer` Rule-based entity rejector based on gazetteers
- `rejectRegex`: Rule-based entity rejector based on regular expressions
- `indocCoref` Chains together mentions that refer to the same entity (in-document coreference)
- `pronominalResolver` Pronomial resolver

The order of execution of the processors is determined internally and cannot be changed. Some processors are prerequisites for other processors. REX will throw an exception if the processor list is missing a required processor.

Use the extractor's `setProcessors` method to enable a subset of the processors before you create the `EntityAnnotator`.

**Return Statistical Entities Only**

```
EntityExtractor extractor = EntityExtractor.createDefaultExtractor(new File("/path/to/root/"));
// Only use the statistical processor.
extractor.setProcessors(EnumSet.of(ProcessorType.statistical));
String input = "George Washington's birthday is an important event.";
Annotator ann = extractor.createAnnotator(LanguageCode.ENGLISH);
AnnotatedText anText = ann.annotate(input);
for (Entity e : anText.getEntities()) {
     System.out.println(e.toString());
}

Output:
Entity{extendedProperties={}, type=PERSON,
mentions=[Mention{extendedProperties={}, startOffset=0, endOffset=17,
source=statistical, subsource=/path/to/root/data/statistical/eng/model-LE.bin,
normalized=George Washington}]}
```

**NOTE**
The redactor chooses among the entities when processors extract the same or overlapping entities. Turning off the redactor will return all entities found by all processors. This can cause overlapping and unsorted entities to be returned.

## 4.2. Statistical Processor

The statistical processor uses models based on computational linguistics and human-annotated training documents. You can add other statistical models to improve extraction for your use case. Extract new entity types or improving the results of the statistical model requires training a new model with using the field training kit [42] (FTK).

Statistical model based extractions can return confidence scores for each entity. Confidence scores correlate well with precision and may be used for thresholding and removal fo false positives. Confidence is calculated by default if linking is enabled. Otherwise, use the `calculateConfidence` parameter to enable confidence scores. To set a threshold value, use the `confidenceThreshold` parameter.

**Statistical Processor Parameters**

| Parameter | Description | Default |
|---|---|---|
| `calculateConfidence` | If true, entity confidence values are calculated. Can be overridden by specifying `calculateConfidence` in the API call. | `false` |
| `confidenceThreshold` | The confidence value threshold below which entities extracted by the statistical processor are ignored. | `-1.0` |
| `statisticalModels` | Additional files used to produce statistical entities for the given language.<br><br>You may pass multiple statistical models. The parameter should be formatted in trios of values specfying language, case-sensitivity and the model file, separated by commas. Case-sensitivity can be `automatic`, `caseInsensitive` or `caseSensitive`. For example, setting two models for case-sensitive English and Japanese might look like : `eng,caseSensitive,english-model.bin,jpn,automatic,japanese-model.bin` | `null` |
| `caseSensitivity` | The capitalization (aka 'case') used in the input texts. Processing standard documents requires caseSensitive, which is the default. Documents with all-caps, no-caps or headline capitalization may yield higher accuracy if processed with the caseInsensitive value.<br><br>Can be `automatic`, `caseSensitive` or `caseInsensitive` | `caseSensitive` |

## 4.3. Deep Neural Network Processor

REX has a deep neural network (DNN) model that can be used in place of the statistical model for selected languages. By default, REX uses the statistical models rather than the DNN model. You can customize which model Rosette uses.

To set the DNN processor, either:

- Use the method `setUseDeepNeuralNetworkProcessor()` in `com.basistech.rosette.rex.EntityExtractor`
  OR
- Provide `ProcessorType.deepLearning` for the method `setProcessors`.

The deep neural network processor is using TensorFlow 2.3.1 (Java version 0.2.0). Ubuntu Linux 14.04+, Windows 7+, and MacOS 10.11+ are fully supported, but you should be able to run the processor successfully on other modern Linux flavors as well. To use the processor on platforms which are not otherwise supported, or to improve the speed on supported platforms, you can replace the TensorFlow library shipped with the product with one that's built from source.

To make use of GPUs, you should download tensorflow-core-platform-gpu and add it to the top of your classpath.

For optimal performance, you can build the Java bindings from source. See the TensorFlow documentation for the full list of software and hardware requirements and the TensorFlow Java documentation for instructions on building the Java bindings from source.

Currently, REX has DNN models for the following languages:

- Arabic (`ara`)
- English (`eng`)
- Hebrew (`heb`)
- Korean (`kor`)

**IMPORTANT**
The deep neural network model and the statistical model cannot be used together. When selected, the DNN replaces the statistical model.

## 4.4. Name Classifier

REX has a name classifier which can be used in place of the statistical model for structured regions. The name classifier is a machine learning model that tries to predict an entity type for an input string. It processes the entire structured region (the input string) as a single entity, predicting a label (PERSON, LOCATION, ORGANIZATION, or NONE) for the string. It works best on tables cells or list items where the entire entry is a single entity. If a structured region contains more text than the entity mention itself, the name classifier will usually label it as NONE.

To enable the name classifier for structured regions set the enum `structuredRegionsProcessingType` to `nameClassifier` when configuring the entity extractor.

Currently, REX supports the name classifier processor for the following languages:

- Arabic
- English
- French
- German
- Hebrew
- Japanese

Each language has its own configuration file, `data/name_classifier/<lang>/<lang>_config.yaml`, where `<lang>` is the 3 letter language code. The `labelScoreThresholds` field determines the chance that a classifier will label a phrase with a given entity type. Lowering the threshold will label more phrases, which will find more true positives, but may also identify more false positives.

To disable an entity type completely, remove or comment out the corresponding entry from the `<lang>_config.yaml` file. Example:

```
# labelScoreThresholds
# Set the model score thresholds for each entity type.
# To turn off an entity from the model, comment it out.
# The accuracy of the current ORG model is too low and so it is better to turn it off for now.
labelScoreThresholds:
  PER: 1.2
  LOC: 3.2
#  ORG: 5.2
```

**NOTE**
Currently, the ORG entity type is excluded for all languages. LOC is enabled for English and Japanese only.

## 4.5. Accept Gazetteer

A gazetteer is a list of exact matches in a predefined closed class. For example, you can use a gazetteer to match all the countries in the world, as there is a precise and unambiguous list of countries. An entry would count as ambiguous if it has multiple possible meanings, such as "Apple", which could be either an ORGANIZATION or a fruit. The gazetteers are very fast at extracting entities. If you are searching for specific words or phrases in your data, a custom gazetteer is a good way to find them quickly.

REX is shipped with default gazetteer files which you can modify, Gazetteer files are located in a subdirectory of the `data` directory, defined by language using the three-letter ISO-639-3 language code. A directory which applies to all languages, uses `xxx` for the language code. For example:

```
<install-directory>/roots/rex-<version>/data/gazetteer/eng/reject/
<install-directory>/roots/rex-<version>/data/gazetteer/xxx/accept/
```

By default, the data files are located in the `<install-directory>/roots/rex-<version>` directory. If you want your custom files to be in a separate location, use an Overlay Data Directory [13].

**Accept Gazetteer Parameters**

| Parameter | Description | Default |
|---|---|---|
| `allowPartialGazetteerMatches` | The option to allow partial gazetteer matches. For the purposes of this setting, a partial match is one that does not line up with token boundaries as determined by the internal tokenizer. This only applies to accept gazetteers. | `false` |
| `acceptGazetteers` | Additional gazetteer files used to produce entities for the given language. | `null` |

### 4.5.1. Creating a Custom Gazetteer

You can create your own, custom gazetteers. To create a custom gazetteer, put the new file in the appropriate location in the `data/gazetteer` tree.

- language-specific: `data/gazetteer/<lang>/accept`
- all languages: `data/gazetteer/xxx/accept`

A gazetteer file:

- Is a .txt file encoded in UTF-8.
- Each comment line is prefixed with #.
- The first non-comment line is `TYPE[:SUBTYPE]`, where TYPE is required and SUBTYPE is optional. The type is applied to the entire gazetteer and defines the entity type name for output. TYPE and SUBTYPE may be predefined or user-defined.

Gazetteer entries and potential matches are space normalized to treat any whitespace between words as a single space. This enables the gazetteer to match entities with differences in whitespace.

**TIP**

To improve performance, text gazetteers can be compiled to a binary gazetteer using build-binary-gazetteer with the REX Field Training Kit [42]. The binary gazetteer file name must end with "-LE.bin".

#### Gazetteers to Track Infectious Diseases

To track common infectious diseases, create a gazetteer like this:

```
# File: infectious-diseases-gazetteer.txt
#
DISEASE:INFECTIOUS
tuberculosis
e. coli
malaria
influenza
```

A single gazetteer may not be enough; you can create as many gazetteers as you need. To search for the scientific names of the infectious disease, you can create a file like this:

```
# File: latin-infectious-gazetteer.txt
#
DISEASE:INFECTIOUS
Mycobacterium tuberculosis
Escherichia coli
Plasmodium malariae
Orthomyxoviridae
```

To track certain diseases by their causes:

```
# File: infectious-bacterial-gazetteer.txt
#
DISEASE:BACTERIAL
Escherichia coli
E. coli
Staphylococcus aureus
Streptococcus pneuminiae
Salmonella
```

Or to track the drugs used to treat them:

```
# File: antimicrobial-drugs-gazetteer.txt
#
DRUG:ANTIMICROBIAL
methicillin
vancomycin
macrolide
fluoroquinolone
```

**TIP**

By default, the data files are located in the `<install-directory>/roots/rex-<version>` directory. To install custom gazetteer files in a separate directory, use an Overlay Data Directory [13].

### 4.5.2. Partial Gazetteer Matches

By default, gazetteer matches must match token boundaries in the input text. You can enable partial matches that do not start and/or do not end on token boundaries. You can also set individual regexes to return partial matches by including `allow-partial-matches="yes"` in a regex.

Use the `EntityExtractor#setAllowPartialGazetteerMatches(boolean allowPartialGazetteerMatches)` method to enable partial matches. You can also set individual regexes to return partial matches by including `allow-partial-matches="yes"` in a regex.

Partial matches require in-document coreference to be disabled. As a result, the mentions will not be grouped into entities.

```
EntityExtractor#setIndoc(IndocTypes.NULL)
```

**TIP**

We do not recommend that you enable partial matches. It adds processing time and may match more than you expect. An entry such as "red" in a COLOR gazetteer will match "Frederick" in the input text.

### 4.5.3. Chinese Gazetteers

REX can analyze both simplified and traditional Chinese language documents. The following three language codes for are all used for Chinese:

- Chinese (`zho`)
- Simplified Chinese (`zhs`)

- Traditional Chinese (`zht`)

`zho` is the Chinese language code; it applies to both simplified and traditional Chinese. Gazetteers using `zho` as the language code apply to documents with a language code of `zhs` or `zht`. Users should include both simplified and traditional Chinese words in the `zho` gazetteer, so that it will work for all Chinese language codes.

**Adding a Simplified and Traditional Word for "lion"**

```
{"language": "zho",
"configuration": {
"entities": { "ANIMAL": [ "狮子", "獅子" ] }
}
}
```

## 4.5.4. Adding Extraction Rules

Whether you have created a default extractor or an empty extractor, you can use the API to add extraction rules. `EntityExtractor` provides a number of methods for setting rules.

For the details, see the Javadoc for `EntityExtractor` (`apidocs/com/basistech/rosette/rex/EntityExtractor.html`).

**Add Custom Rules to the Default Configuration**

```
EntityExtractor extractor = EntityExtractor.createDefaultExtractor(new File("path/to/root"));
// Add a gazetteer entity.
extractor.addGazetteerEntity(LanguageCode.lookupByISO639("eng"),  "birthday", "MYTYPE");
// Add a gazetteer rule to reject "George Washington" as PERSON.
extractor.addGazetteerEntity(LanguageCode.ENGLISH, "George Washington", "PERSON", false);
String input = "George Washington's birthday was a joyous event.";
Annotator ann = extractor.createAnnotator(LanguageCode.ENGLISH);
AnnotatedText anText = ann.annotate(input);
for (Entity e : anText.getEntities()) {
    System.out.println(e.toString());
}
```

```
Output:
Entity{extendedProperties={}, type=MYTYPE, mentions=[Mention{extendedProperties={},
startOffset=20, endOffset=28, source=gazetteer, subsource=dynamic, normalized=birthday}]}
```

**Use Custom Rules with no Defaults**

```
EntityExtractor extractor = EntityExtractor.createEmptyExtractor();
extractor.setLicense(new File("../rex-je/licenses/rlp-license.xml"));
// Add a gazetteer entity.
extractor.addGazetteerEntity(LanguageCode.ENGLISH, "birthday", "MYTYPE");
// Add a  and a regular expression entity that returns 4 digits as a year. In a
// Java string, the regex escape (\) must itself be escaped.
extractor.addRegularExpression(LanguageCode.ENGLISH, "(\\d{4})", "YEAR");
String input = "George Washington's birthday in 1789 was a joyous event.";
Annotator ann = extractor.createAnnotator(LanguageCode.ENGLISH);
AnnotatedText anText = ann.annotate(input);
for (Entity e : anText.getEntities()) {
    System.out.println(e.toString());
}

Output:
Entity{extendedProperties={}, type=MYTYPE, mentions=[Mention{extendedProperties={},
startOffset=20, endOffset=28, source=gazetteer, subsource=dynamic, normalized=birthday}]}
Entity{extendedProperties={}, type=YEAR, mentions=[Mention{extendedProperties={},
startOffset=32, endOffset=36, source=regex, subsource=eng_0, normalized=1789}]}
```

## 4.6. Accept Regex

Regular expressions (regexes) are used for finding entities which follow a strict pattern with a rigid form and infinite combinations, such as URLs and credit card numbers. In the default REX installation the regex files are:

- **language specific:** `data/regex/<lang>/accept/regexes.xml` where <lang> is the ISO 693-3 language code
- **cross-language:** `data/regex/xxx/accept/regexes.xml`

You can modify these files to add new patterns to extract the same entity type.

**Accept Regex Parameters**

| Parameter | Description | Default |
|---|---|---|
| `acceptRegularExpressionSets` | Additional files used to produce regex entities. | `null` |
| `supplementalRegularExpressionPaths` | The option to add supplemental regex files, usually for entity types that are excluded by #default. The supplemental regex files are located at `data/regex/<lang>/accept/supplemental` and are not used unless specified. | `null` |

To extract new entity types that have predictable patterns, add a new XML regex file, either the language-specific (`<lang>` ) or generic (`xxx`) location. REX uses the Tcl regex format for defining the regex patterns.

REX modifies the regex matcher so that `\n` in a regex expression matches straight new lines (`\n`\), carriage returns (`\r`), or a combination of both (`\r\n`). Regardless of what is matches, offsets and lengths in the result will match the input document.

By default, the data files are located in the `<install-directory>/roots/rex-<version>` directory. If you want your custom files to be in a separate location, use an Overlay Data Directory [13].

### 4.6.1. Creating a New Regex

Each regex is defined in a **regexp**, which may contain a **lang** attribute and may refer to **define** elements.

The **lang** attribute designates the language for the regex. If the regex applies to text in any language, there is no lang attribute. For example, all the regexes in `data/regex/eng` should include `lang="eng"`. The regexes in `data/regex/xxx` do not include the `lang` attribute, since they apply to text in any language.

A **define** element contains a regex and a **name** attribute. By naming the regex, you can include the regex in multiple regexp files.

**Defining a Regular Expression: `time_ampm`**

1. Define the regular expression in a `define` statement:

```
<define lang="eng" name="time_ampm">(?:[pa]\.?\s?m\.?)</define>
```

2. Use the regular expression in a `regexp` statement:

```
<regexp lang="eng" type="TEMPORAL:TIME">...${time_ampm}...<regexp>
```

When REX evaluates the `regexp` statement, it follows these steps:

1. When `${time_ampm}` appears in a `regexp lang="eng"` element, REX looks for a `define name="time_ampm" lang="eng"` statement.
2. If it does not find the element, Rosette REX looks for a `define name="time_ampm"` element without the lang attribute.
3. If it does not find such an element, an error occurs.

If you include an `id` attribute setting, that value is returned as the "subsource" of an entity returned by this regexp.

## 4.6.2. Supplemental Regexes

REX is shipped with supplemental regexes which are not activated by default. The supplemental regexes are located in the `data/regex/<lang>/accept/supplemental` directory.

To activate a supplemental regex, move the file into the appropriate `data/regex/<lang>/accept` directory. For example, to activate the German date regex, copy or move the file `date-regex.xml` from `data/regex/deu/accept/supplemental` directory to the `data/regex/deu/accept` directory.

**Supplemental Regexes by Language**

| Language (ISO Code) | Currency | Date | Distance | Geo | License-Plate | Numbers | Org | Personal-ID | Phone | Time |
|---|---|---|---|---|---|---|---|---|---|---|
| Arab `ara` | | X | X | X | X | X | | | | X |
| German `deu` | | X | X | X | | X | | | | X |
| English `eng` | | X | X | X | | X | X | | | X |
| Farsi `fas` | | X | X | X | | X | | | | X |
| French `fra` | | X | X | X | | X | | | | X |
| Hebrew `heb` | | X | X | X | | X | | | | X |
| Hungarian `hun` | | X | X | X | | X | | | | X |

| Language (ISO Code) | Currency | Date | Distance | Geo | License-Plate | Numbers | Org | Personal-ID | Phone | Time |
|---|---|---|---|---|---|---|---|---|---|---|
| Hindu `ind` | | X | X | | | | | | | X |
| Italian `ita` | | X | X | X | | X | | | | X |
| Japanese `jpn` | | X | X | X | | X | | | | X |
| Korean `kor` | | X | X | X | | X | | | | X |
| Dutch `nld` | | X | X | X | | X | | | | X |
| Portuguese `por` | | X | X | X | | X | | | | X |
| Pursian `pus` | | X | X | X | | X | | | | X |
| Russian `rus` | | X | X | X | | X | | | | X |
| Spanish `spa` | | X | X | X | | X | | | | X |
| Swedish `swe` | | X | X | X | | X | | | | X |
| Upper-case English `uen` | | X | X | X | | X | | | | X |
| Vietnamese `vie` | X | X | X | | | | | | | X |
| Simplified Chinese `zhs` | | X | X | X | | X | | | | X |
| Traditional Chinese `zht` | | X | X | X | | X | | | | X |
| Malay, Standard `zsm` | | X | X | | | | | | | X |

## 4.7. Joiner

The joiner combines adjacent entities into a single entity, based on the joiners rules. REX then returns the single entity.

The configuration file for joining adjacent entities is in `data/etc`.

### Joiner Parameters

| Parameter | Description | Default |
|---|---|---|
| `joinerRuleFiles` | File containing additional joiner rules. | `null` |
| `runJoinerPostRedactor` | Run the joiner after the redactor, instead of before. | `false` |

The file `neredact-config.xml` specifies the rules for joining adjacent entities. Adjacent TITLE entities are joined into a single TITLE entity. The `joiner` elements for joining TITLE and PERSON entities into a PERSON entity are commented out by default.

```
<neredactconfig>
  <joiners>
    <joiner left='TITLE' right='TITLE' joined='TITLE'/>
<!-- Not joined by default
    <joiner language='eng' left='TITLE' right='PERSON' joined='PERSON'/>
    <joiner language='jpn' left='PERSON' right='TITLE' joined='PERSON'/>
```

```
-->
  </joiners>
</neredactconfig>
```

Rules can optionally specify a language, in which case they will apply only to entities of that specific language. If a language is not specified, the rule will apply for any language.

Entities are considered adjacent if they are separated by no more than 5 whitespace characters.

For example, to join "Barack Obama" and "President" in "Barack Obama, President", the joiner rule is:

```
<joiner left='PERSON' adjacency-regex=',\s+' right='TITLE' joined='PERSON'/>
```

The joiner runs before the redactor, as of release 7.46.2. To run the joiner after the redactor, set the parameter `runJoinerPostRedactor` to `true`.

## 4.8. Redactor

The redactor determines which entity to choose when multiple mentions for the same entity are extracted. The redactor first chooses longer entity mentions over shorter ones. If the length of the mentions are the same, the redactor uses weightings to select an entity mention.

Different processors can extract overlapping entities. For example, a gazetteer extracts "Newton", Massachusetts as a LOCATION, and the statistical processor extracts "Isaac Newton" as a PERSON. When two processors return the same or overlapping entities, the redactor chooses an entity based on the length of the competing entity strings. By default, a conflict between overlapping entities is resolved in favor of the longer candidate, "Isaac Newton".

**TIP**

The correct entity mention is almost always the longer mention. There can be examples, such as the example of "Newton" above, where the shorter mention is the correct mention. While it might seem that turning off the option to prefer length is the easiest fix, it usually just fixes a specific instance while reducing overall accuracy. We strongly recommend keeping the default `redactorPreferLength` as true.

The redactor can be configured to set weights by:

- entity type
- processor

### 4.8.1. Configuring the Redactor

The configuration file for setting redactor weights is in `<data/etc`.

#### Set weight by entity type

Each of the `ne-type` elements in `ne-types.xml` defines weightings for a specified entity type. For example, to assign weights for IDENTIFER entities:

```
<ne_type>
    <name>IDENTIFIER</name>
    <subtypes>
        <name>EMAIL</name>
        <name>URL</name>
        <name>DOMAIN_NAME</name>
        <name>IP_ADDRESS</name>
        <name>PHONE_NUMBER</name>
        <name>FAX_NUMBER</name>
        <name>PERSONAL_ID_NUM</name>
        <name>CREDIT_CARD_NUM</name>
        <name>MONEY</name>
        <name>PERCENT</name>
        <name>NUMBER</name>
    </subtypes>
    <weight name="statistical" value="9" />
    <weight name="gazetteer" value="10" />
    <weight name="regex" value="10" />
</ne_type>
```

This assigns weights for the IDENTIFIER entities. They are also weighted by processor.

**Set weights by processor**

The processor weights are relative values; they do not have to add up to any specific value. For example, to favor gazetteer entries over regexes, and favor both over values returned by statistical analysis, you could set the weights as follows:

```
    <weight name="statistical" value="1" />
    <weight name="gazetteer" value="10" />
    <weight name="regex" value="5" />
```

Some processors offer subsources to identify specific instances. The kb-linker processor returns a subsource indicating the knowledge base the extraction originated in. To set a weight to a specific subsource set the `name` property to `PROCESSOR:SUBSOURCE`. For example, to favor your custom knowledge base (myKB) over other extractions but keep other linker extractions low, you could set the weights as follows:

```
    <weight name="kb-linker:MyKB" value="20" />
    <weight name="kb-linker" value="1" />
```

When you define new entity types for gazetteers and regexes, you should add those entity types to `ne-types.xml` if you want to control how the redactor resolves conflicts. Types that do not appear in this file (or are not assigned weights through the API) receive weights of 10 for all three processors.

For an entity type with subtypes, the settings apply to all the subtypes.

You can use the `EntityExtractor#setRedactorWeight` method to assign different weights to individual subtypes.

## 4.9. Reject Gazetteer

Instead of adding entities to extract when matched you can define a list of entities to reject when matched. These are reject gazetteers.

The format of a reject gazetteer is identical to the format of an accept gazetteer except the wildcard (`*`) is allowed in the entity type. As with accept gazetteers, they are arranged by language.

- language-specific: `data/gazetteer/<lang>/reject`
- all languages: `data/gazetteer/xxx/reject`

If, for example, it is for rejecting German entities, put it in `data/gazetteer/deu/reject`. If it is for rejecting entities in multiple languages, put it in `data/gazetteer/xxx/reject`.

#### Pronominal Resolution Parameters

| Parameter | Description | Default |
|---|---|---|
| `rejectGazetteers` | Additional gazetter files used to reject entities for the given language. | `null` |

#### Reject Gazetteer

The following .txt file in `data/gazetteer/eng/reject`, rejects the PERSON entity named "George Watson" when processing English documents.

```
PERSON
George Watson
```

A wildcard entity type would match any types. The value "George Watson" would be rejected from all entity types, not just PERSON.

```
*
George Watson
```

## 4.10. Reject Regex

A typical regex is used to identify entities of a specified entity type. You can also define a regex to reject entities; that is whenever the pattern is identified, the entity is rejected as the defined type. Reject regexes follow the same format as accept regexes with the addition that the wildcard (*) is allowed for the entity type.

Place your reject regex files in the following directories:

- language-specific: `data/regex/<lang>/reject`
- all languages: `data/regex/xxx/reject`

#### Reject Regex Parameters

| Parameter | Description | Default |
|---|---|---|
| `rejectRegularExpressionSets` | Additional regex files used to reject entities. | `null` |

For example, a file to reject German entities, is named `data/regex/deu/reject`. Files rejecting entities in multiple languages go in `data/regex/xxx/reject`.

#### Regex to Reject a Location

The following .xml file in data/regex/eng/reject rejects **Baltimore** as a **LOCATION** entity when processing English documents.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE regexps PUBLIC "-//basistech.com//DTD RLP Regular Expression Config 7.1//EN"
                         "urn:basistech.com:7.1:rlpregexp.dtd">

<regexps>
  <regexp lang="eng" type="LOCATION">Baltimore</regexp>
</regexps>
```

**NOTE**
Lookbehind assertions are not supported.

## 4.11. In-document Coreference

Documents often contain multiple references to a single entity. In-document coreference (indoc coref) chains together all mentions to the same entity.

### Configuration Parameters

| Parameter | Description | Default |
|---|---|---|
| `indocType` | An option for document entity resolution (also known as entity chaining). Valid values are: `HIGH`, `STANDARD`, `STANDARD_MINUS` or `NULL` | `STANDARD` |
| `maxResolvedEntities` | The maximum number of entities for in-document coreference resolution (a.k.a. chaining). | `2000` |

## 4.12. Pronominal Resolution

If `resolvePronouns` is enabled (it is disabled by default), REX will try to resolve pronouns with the corresponding antecedent entities.

### Pronominal Resolution Parameters

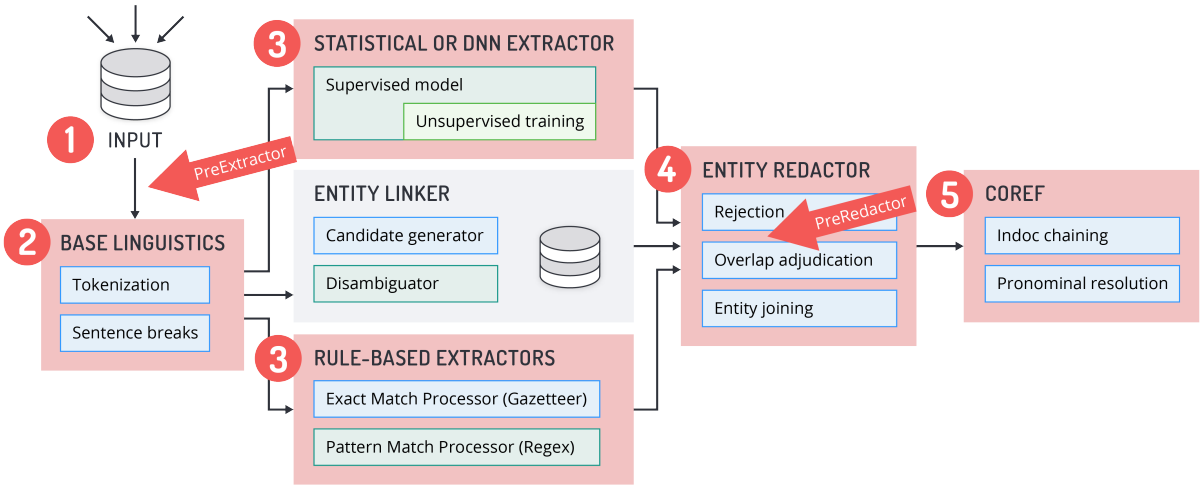
| Parameter | Description | Default |
|---|---|---|
| `resolvePronouns` | When true, resolve pronouns to person entities. | `false` |

# 5. Creating Custom Processors

REX has a plugin architecture that allows users to create custom processors that can be inserted into the REX pipeline at two points.

1. At the `preExtractor` phase - a custom processor may insert additional text pre-processing after input, but before tokenization and sentence breaks (either provided by Rosette Base Linguistics or the user's own tokenizer and sentence breaker).

2. At the `preRedaction` phase - a custom processor may insert corrections or modifications to output from the default extractors (statistical, regex, gazetteer), using the full information and context that the default extractors have access to (e.g., plain text data, sentence boundaries, tokens, full list of entities extracted and the source extractors which found them, boundaries, and processor types, etc.)




## 5.1. Pre-Extraction Custom Processors: For Additional Text Pre-Processing

Custom processors at the `preExtractor` phase can provide additional text pre-processing. For example, if the files contain boilerplate, footers, and navigation bar text that are not the target of the analysis, including these parts of the document in the analysis may trip up the tokenization process and thus decrease the overall quality of extraction results. A `preExtractor` custom processor can strip footers of emails or add metadata to the target files.

The code sample at `samples/src/SampleCustomProcessor.java` includes a custom processor called `MetadataAnnotator`, which adds metadata to each file.

## 5.2. Pre-Redaction Custom Processors: For Correcting/Modifying Extractor Output

Custom processors at the `preRedaction` phase are run after the default processors (statistical, gazetteer, regex) and any filters (reject files for regex and gazetteer) have run, but before the redactor. A custom processor at the `preRedaction` phase receives all information and context of the intermediate results from the output of the default extraction processors, and can make modifications to those results before the redactor phase adjudicates conflicts between the results from statistical, gazetteer, and regex processors.

Only entities and metadata attributes fields can be updated with the pre-redaction custom processor. If the custom processor attempts to make changes in forbidden fields, specifically data (input), token, or sentence attributes, the specified changes will be ignored and a warning will be logged.

Examples of cases that are correctable with a custom processor include:

1. Reject a mention as an entity: Cases where REX incorrectly extracts a mention that is not an entity can be excluded from the new list of entity results.

2. Correcting the entity type: If, for example, your dataset consists of personal letters, and you have high confidence that after a closing such as “Love,” or “Sincerely yours,” the entity that follows should be a PERSON, but REX is identifying it as an ORGANIZATION.

3. Modifying entity boundaries: If, for example, REX is incorrectly extracting “Hi” as part of a PERSON entity, as in “Hi Joe” instead of just extracting “Joe”.

**NOTE**

**Filters (reject files) vs. Pre-Redaction Custom Processors**

The reject files for regexes [23] and gazetteers [19] simply filter out a list of words or a pattern-matched set of words the user does not want to extract as entities. These reject functions operate without considering the context in which these words appear. By contrast, custom processors at the preRedaction phase have access to the entire context in which an extracted entity appears, and thus can implement smarter rules.

## 5.3. Implementing the Custom Processor

You can implement the `CustomProcessor` and `Annotator` interfaces in Java in your own JAR and register them via the extractor's `setCustomProcessors`. Your custom processor is the factory of the annotator implementation and thus should be familiar with the requirements of your annotators, and provide them with the correct parameters for the language and the phase requests. The `Annotator` is the interface to the ADM (i.e., annotated text) and based on the custom processor it manipulates the ADM and outputs it to the next phase.

## 5.4. Walk-Through Example of preRedaction Phase Custom Processors

A custom `preRedaction` annotator receives entity mentions from all extraction processors, after reject processors run and before redactor and coref processors run. It can reject (remove) entity mentions, modify entity types or adjust entity mention offsets. These modifications will affect the input of the next processors in the pipeline. For example coref would not consider chaining together PERSON and ORGANIZATION mentions into the same entity, so a mention whose entity type was changed from ORGANIZATION to PERSON by a custom processor would only be chained to other PERSON entities. After the `Redactor` phase, the rest of the pipeline runs as usual.

The code samples at `samples/src/SampleCustomProcessor.java` demonstrate a sample custom processor and how it might be used in an application.

1. Create a default entity annotator (to compare its output with the modified entity annotator's output).
2. Register the custom processor by `registerCustomProcessorClass()`.
3. Configure the entity extractor to use the custom processor by `setCustomProcessors()`.
4. Create an entity annotator with three custom processors
5. First, `personContextAnnotator`, specifies that after the letter closing, "Love," the entity that follows is PERSON. Note that the annotator changes the "source" (indicating which processor produced the result) from "statistical" to "custom processor". Remember to edit the redactor configuration file, `data/etc/ne-types.xml`, to give greater weight [26] to the custom processor if you would like it to "win" when there is a conflict with results from a default processor at the `preRedaction` phase.
6. Second processor, `boundaryAdjustAnnotator`, corrects a REX mention boundary issue in which "Hi Joe" is extracted instead of just "Joe".
7. Third process is an example of a custom processor at the `preExtraction` phase that adds metadata to `AnnotatedText`.

8. Test (serves as the “application”)

# 6. Entity Linking

Entity linking provides a mechanism for disambiguating the identity of similarly named entities mentioned in a document. For example, “Rebecca Cole” is the second African-American woman to become a doctor in the United States and also the name of an Australian professional basketball player. Linking helps establish the identity of the entity by disambiguating common names and matching a variety of names, such as nicknames and formal titles, with an entity ID.

## 6.1. Linker Processor

To link entities to a knowledge base, REX uses a statistical disambiguation model trained on a knowledge base. The linker processor is delivered with a model based on a default Wikidata knowledge base. If the entity exists in Wikidata, then REX returns the Wikidata QID, such as Q1 for the Universe, in the `entityId` field. Once enabled, the linker can also return:

- DBpedia Types [33]
- Refinitiv PermIDs [33]

If the linker is disabled (the default), a random string is returned as the entityId. The string starts with a "T" (temporary id) followed by a random number, which is unique per document.

In addition to the default Wikidata knowledge base, you can train a disambiguation model for a custom knowledge base. The custom knowledge base [34] model can replace or run in parallel with the default knowledge base.

Once the custom model has been trained, you can add new entries without retraining the model, as long as the new entries are similar to the ones used for training.

**Linker Processor Files** The linker processor is packaged as part of the standard REX distribution. The linker files are in the subdirectory `data/flinx`.

**IMPORTANT**

The linker processor both extracts and links entities. These functions are separate from the default REX entity extraction performed by the statistical, pattern-matching, and exact-matching processors, thus **entities from the linker processor may differ** from those returned by the statistical, pattern-matching, and exact-matching processors. The redactor will resolve any overlapping or conflicting entity results.

### Linker Configuration Parameters

| Parameter | Description | Default |
|---|---|---|
| `kbs` | Custom list of Knowledge Bases for the linker, in order of priority | `null` |

| Parameter | Description | Default |
|---|---|---|
| `linkEntities` | The option to link mentions to knowledge base entities with disambiguation model. Enabling this option also enables `calculateConfidence`. | `false` |
| `calculateConfidence` | If true, entity confidence values are calculated. Can be overridden by specifying `calculateConfidence` in the API call. | `false` |
| `useDefaultConfidence` | The option to assign default confidence value 1.0 to non-statistical entities instead of null. | `false` |
| `linkingConfidenceThreshold` | The confidence value threshold below which linking results by the kbLinker processor are ignored. | `-1.0` |

Entity linking is enabled by setting the `EntityExtractor linkEntities(boolean lnk)` to `true`, and disabled by setting it to `false`.

### 6.1.1. Selecting a Knowledge Base for Linking

By default, all knowledge bases under the `data/flinx/data/kb` directory inside the REX installation will automatically be used for linking. Any custom knowledge bases placed in this directory will be loaded each time REX launches.

You can enable dynamic loading, controlling which custom knowledge bases will be loaded with `EntityExtractor setKbDirs`, which takes a `List` of `Path`s to knowledge bases.

The list is in priority order; the match from the highest knowledge base on the list will be returned.

**IMPORTANT**

Setting the list of knowledge bases completely overwrites the list of knowledge bases the linker uses. If you want the default Wikidata knowledge base to be included, it must be on the list of knowledge bases.

### 6.1.2. DBpedia Types for Linked Entities

The linker processor can associate entities with types drawn from the DBpedia ontology [59], which provides over 700 types at up to seven levels of granularity.

Providing both primary and secondary entity types increases the usability of the linker processor's results for many NLP applications. For example, classifying Pheobe Buffay (QID: Q682396) as PERSON is a necessary first step towards effective pronominal resolution, whereas the secondary type path Agent/FictionalCharacter/SoapCharacter paves the way for identifying the relationship of Pheobe Buffay with Lisa Kudrow (Q179041).

Turning on the `includeDBpediaType` flag increases the recall of the linker processor's results. When the flag is enabled, the linker will return both non-named entities like "guitar" (Q6607), type MISC, or named entities of new types, such as "2018 World Cup" (Q170645), type EVENT.

By default, providing DBpedia types is turned off. To enable it; `EntityExtractor includeDBpediaTypes (boolean includeDBpediaTypes)` must be set to `true`.

### 6.1.3. PermIDs

The linker processor can return the Refinitiv PermID for a subset of entities which are identified with a QID. By default, linking to PermIDs is turned off.

Entity linking to PermIDs is enabled by setting the `EntityExtractor includePermID(boolean includePermID)` to true, and disabled by setting it to false. In order to activate PermID linking, both `EntityExtractor includePermID(boolean includePermID)` and `EntityExtractor linkEntities(boolean lnk)` must be set to `true`. When PermID linking isn't explicitly set, its default value is `false`.

## 6.2. Creating a Custom Knowledge Base for Linking

**NOTE**
Linking to a custom knowledge base is licensed separately. Contact Rosette support for a license for this functionality. Custom knowledge base linking is currently only available in the Java Edition of REX.

The linker supports linking from multiple knowledge bases. In addition to the default Wikidata knowledge base, you can train a disambiguation model for a custom knowledge base. Your custom knowledge base model can replace or run in parallel with the default Wikidata knowledge base.

To create a new disambiguation model:

1. Define [35] the custom knowledge base.
2. Train [37] the disambiguation model with the REX FTK.

To use the new model for linking:

1. Copy the new knowledge base folder `<rex-field-training-home>/asset/custom-kb` to be a subdirectory of `<rex-home>/data/flinx/data/kb/`. Restart REX to use the new model.
2. Customize [33] (*optional*) the set and priority of linking knowledge bases. REX does not have to be restarted to use the new model, it will be uploaded dynamically.

Once the model has been trained, you can add new entries without retraining the model, as long as the new entries are similar to the ones used for training.

You can add Custom Knowledge Base Connectors [41] that retrieve linking information directly from an external data source.

### 6.2.1. Overview of Linking and Custom Seeding

**How does entity linking work?**

The linker processor first identifies possible entity mentions (“candidates”) through exact matching. It walks through the input text token by token, looking for matches in an `aliases.bin` file, which contains the alias phrases in the json input files. Then each candidate mention is resolved to a known entity with an ID or labeled as “NONE". NONE here means the candidate mention could not be linked to an entry in the knowledge base.

The disambiguation phase applies a machine-learned SVM ranking model on each candidate mention in isolation. That is, the linking of one candidate doesn’t affect the linking of another. Features for each

mention can look at context, but the model is still applied separately to each mention. Context is provided by `contextWords` and `relatedEntityIds`.

#### Generating binaries

In this phase we read and process the knowledge base's json files, generating easy-to-load binary files. The binary files are used both for candidate generation and selecting features for the disambiguation model.

Different knowledge bases may provide different details about the entities. For example, most knowledge bases can provide information about the prevalence score of each entity, but very few knowledge bases indicate if a phrase is likely to be an entity or not. Those details are used for disambiguation, so a byproduct of this phase is the `config` file, which lists the available features for the training process.

#### Training the disambiguation model

After producing a list of the available features, the actual training is performed on a pre-annotated corpus provided by Basis Technology for training the disambiguation model. The first step of the process extracts all available features for each annotated entity sample. While there are over a dozen features that could be used, not all may be usable if the required data is not available in the user's knowledge base. The second step trains the SVM model.

At the end of this phase, both the binaries and the new model are found in the `/asset/custom-kb` directory, ready to be copied to `<rex-home>/data/flinx/data/kb/`.

#### Custom knowledge bases without a disambiguation model

Passing `-d` as an argument to `train-linker-model` creates a custom knowledge base without a disambiguation model. Such knowledge bases act as "enhanced gazetteers"; like a gazetteer they will always extract any aliases contained in them. They will also add an ID provided by the knowledge base. Entities extracted from a knowledge base without a disambiguation model will have `1.0` as their linking confidence value. If several entities in the knowledge base have the same alias, one will be returned if the alias is encountered in the text, but which is returned is indeterminate. Overlapping aliases will return the longest one.

### 6.2.2. Defining a Custom Knowledge Base

A custom knowledge base for the linker processor requires the following files:

- A file `kb.json` containing language-agnostic information, such as entity ID or entity type. Each knowledge base typically has a single file of this type.
- One or more files `<lang>-kb.json` containing language-dependent information, such as an entity's primary name. `<lang>` is the three-letter ISO 639-3 code indicating the language of the file contents.

For example, if the knowledge base contains entities with information for Japanese and Korean languages, the expected input files are: `kb.json`, `jpn-kb.json` and `kor-kb.json`. If a knowledge base contains only a single language, the kb.json file is not required; all data can go into a single file, the language-specific file (e.g. `jpn-kb.json`).

### kb.json file

The kb.json file lists language-agnostic entity definitions.

| Field | Required | Description |
|---|---|---|
| `entityId` | Y | Unique identifier for the entity. In the default knowledge base Basis Technology provides, the entityID is the Wikidata QID. |
| `entityType` | Y | The entity's REX entity type, either standard [4] (e.g., PERSON, LOCATION, ORGANIZATION) or custom [5] (e.g., DRUG, BACTERIA, FRUIT). |
| `prevalence` | N | A popularity score showing the relative frequency of this entity in the training corpus, represented as an unbounded non-negative integer. In the default knowledge base, the prevalence is defined as the number of times the entry is a destination link from other Wikipedia pages. |
| `relatedEntityIds` | N | List of IDs of entities related to this entity in some way. In the default knowledge base, a related entity is one that appears in the first paragraph of the entity's Wikipedia page. Note that it is possible for an entity to be mentioned in a Wikipedia article but for the entity to not have a QID. |

### Language-Agnostic Definitions (`kb.json`)

```
[
  {
    "entityId": "B1",
    "entityType": "ORGANIZATION",
    "prevalence": 100,
    "relatedEntityIds": [
      "B2"
    ]
  },
  {
    "entityId": "B2",
    "entityType": "PERSON",
    "prevalence": 95,
    "relatedEntityIds": [
      "B1"
    ]
  }
]
```

## <lang>-kb.json file

This file provides a list of entity definitions, not necessarily corresponding to the list in kb.json. Information for each entity may duplicate any of the language-agnostic fields in kb.json, and also includes language-dependent information:

| Field | Required | Description |
|---|---|---|
| `entityId` | Y | A unique identifier of Latin alphanumeric characters starting with a capital letter. |
| `entityType` | Y (unless specified elsewhere) | The entity's REX entity type, either standard [4] (e.g., PERSON, LOCATION, ORGANIZATION) or custom [5] (e.g., DRUG, BACTERIA, FRUIT). |
| `prevalence` | N | The relative frequency of this entity in the training corpus, represented as an unbounded non-negative integer. In the default knowledge base, the prevalence is the number of times the entry appears as a destination link from other Wikipedia pages. |
| `relatedEntityIds` | N | List of IDs of entities related to this entity in some way. In the default knowledge base, a related entity is one that appears in the first paragraph of the entity's Wikipedia page. Note that it is possible for an entity to be mentioned in a Wikipedia article but not have a QID. |
| `entityName` | N | Primary language-specific name of entity. In the default knowledge base, this is the name of the entity's Wikipedia page. |

| Field | | Required | Description |
|---|---|---|---|
| `aliases` | | N | List of language-specific aliases for this entity. Each alias is defined by a JSON record with the following 3 fields: |
| | `phrase` | Y | The actual alias name or phrase referring to the entity. |
| | `aliasProbability` | N | A number from 0 to 1 describing the likelihood that when the associated entity is mentioned in a particular language, the mention matches this phrase. |
| | `isEntityProbability` | N | A number from 0 to 1 describing the likelihood that this phrase refers to an entity (although not necessarily this one). In our default knowledge base, this value is the likelihood the phrase was used as a Wikipedia anchor tag. |
| `contextWords` | | Y | List of language-specific words that are strongly related to this entity. In the default knowledge base, a context word is one that appears in the first paragraph of the entity's Wikipedia page. For a custom knowledge base, it is up to the author to decide how/if this field gets populated. However, `contextWords` should be chosen judiciously, limited to only words that have a very strong connection. For example, for film actor Kevin Bacon, list which school he attended, but not every person who has ever met him. |

**Language-Specific Definitions (`eng-kb.json`):**

```
[
 {
    "entityId": "B1",
    "entityName": "Basis Technology",
    "entityType": "ORGANIZATION",
    "aliases": [ {
        "phrase": "Basis Technology"
      }, {
        "phrase": "Basis Tech"
      }
    ],
    "contextWords": [ "Software",  "NLP", "Rosette", "Text Analytics" ]
  }, {   "entityId": "B2",
    "entityName": "Carl W. Hoffman",
    "entityType": "PERSON",
    "aliases": [ {
        "phrase": "Carl Hoffman"
      }, {
        "phrase": "Carl"
     }],
    "contextWords": [ "Software",  "NLP", "Rosette", "Text Analytics" ]
} ]
```

If the knowledge base contains only a single language, all necessary information may be represented in a single file. This file, `eng-kb.json`, is the only file necessary.

### 6.2.3. Generating Linker Files with the FTK

The FTK converts your knowledge base definition files into the linker input format, generating binaries from the data, and training the disambiguation model to your custom knowledge base.

**NOTE**

The REX field training kit (FTK) is not part of the standard REX distribution. Contact `<support@rosette.com>` to obtain the FTK package.

## Setup the FTK and Training Files

Once you have defined the custom knowledge base in the prescribed format [35], you are ready to setup the FTK in preparation for training the model. These directions assume you have successfully installed REX and have the docker version of the REX FTK package.

Unzip the Linker data (`rex-je-linker-data-<version>.zip`) into the rex-je root directory.

### Load the Docker image

1. Create a working directory, which we will refer to as `<rex-field-training-home>`.
2. Load the image:

```
docker load -i rex-training-docker-<version>.tar.gz
```

3. Validate that the image is loaded:

```
docker images
```

   Confirm there is an image named `rex-field-training` in the images list.

### Prepare the training files directory

1. Create the directory `asset/seeding-input/` in `<rex-field-training-home>`.

```
mkdir -p <rex-field-training-home>/asset/seeding-input
```

   The `asset` directory is used for transferring data in and out of it. In the case of docker container FTK, this is your mounted `asset` directory.

2. Copy your knowledge base json input files to `asset/seeding-input/`. After placing them, the directory should look like this:

```
ls <rex-field-training-home>/asset/seeding-input
eng.json jpn.json  kb.json  kor.json
```

### Run the Docker Container

```
docker run -it --rm -v <local-asset-dir-full-path>:/asset -v <rex-installation-path>:/basis/rex rex-field-training
```

A startup message similar to the one below, should now appear in your console window, confirming that your Docker is set up correctly:

```
REX field training tools (docker) version
Please refer to the legal notices in /basis/ftk/dependencies/ThirdPartyLicenses.txt.
Copyright (c) 2016 Basis Technology Corporation All Rights Reserved.
Support@basistech.com
 http://www.basistech.com

brat instance started on port 8080 in this container. brat data: /asset/bratdata

Available commands:

Statistical model:
```

```
  generate-ngram, cluster-ngram, train-model, evaluate

Advanced tools: learning-curve

Linker model:
 generate-linker-binaries, train-linker-model

Binary gazetteer:
 build-binary-gazetteer
```

## Train the Model

The following is a brief rundown of the steps for custom seeding. For additional information about what is happenings behind the scenes as these steps are carried out, please refer to Overview of Linking and Custom Seeding [34].

1. From your console, generate the binary files:

   ```
   generate-linker-binaries
   ```

2. Edit the existing configuration file for each language located at `<rex-field-training-home>/asset/config/features/` to show only the features that you will be using for training your disambiguation model. The feature set will depend partly on what data is available inside your knowledge base's json files.

   `NO_ENTITY_THRESHOLD` **DO NOT REMOVE this required item.** This item does not represent a feature, but rather is the minimum required score for linking to happen.

   `IS_ENTITY_PROBABILITY` Use this feature if your json file has the `isEntityProbability` field filled out. This feature is the probability that a matching candidate is an entity.

   `ALIAS_LOWERCASE_PROBABILITY` Use this feature if your json file has the `isEntityProbability` field filled out. This feature uses the probability that a matching candidate name in lowercase is still an entity.

   `ALIAS_TITLECASE_PROBABILITY` Use this feature if you want your model to be case-sensitive in disambiguation. This feature uses the probability that the original candidate is titlecased, based on all aliases (pre-computed per candidate).

   `MENTION_TOKEN_COUNT` An optional feature that may or may not be helpful in your model. Given the number of tokens in a mention, how likely is it to be linked to an entity.

   `IS_MENTION_TITLECASE` A feature helpful in languages which have case sensitivity (e.g., English but not Japanese). Looks to see if the mention at runtime is in title case.

   `IS_MENTION_ACRONYM` A feature helpful in languages which use acronyms, or if your knowledge base contains acronyms. Looks to see if a mention is an acronym.

   `CANDIDATE_PREVALENCE` Use this feature if your json file has the `prevalence` field filled out. There are cases where you may not want to use this feature, such as if you don't want your model to be biased against entities which have low prevalence.

   `contextWords = 8` This feature enables context-sensitive entity linking. This feature is required if you have the contextWords field filled out for 5% or more of your knowledge base. If contextWords data doesn't exist or is sparse, turn this feature off. A high value for this feature indicates your context

words are reliable, i.e., high quality context words that really help identify the entity; while a low value indicates lower quality, i.e., every place this person ever visited or person they ever met.

**Choosing a value for the `CONTEXT_WORDS` feature** The entities in the knowledge base are in a continuum from an entity that has very few context words (for whom every connection is thus precious and distinguishing), such as a near hermit, to someone that has a great many context words (for whom every connection is less precious and distinguishing), such as for Barack Obama. During training, context words are divided into "bins" and more bins means a finer grained distinction in this area. We recommend starting with 8 and then trying different values to see what produces the best results for you.

3. From your console, train the disambiguation model

```
train-linker-model
```

4. Copy the new knowledge base folder `<rex-field-training-home>/asset/custom-kb` to be a subdirectory of `<rex-home>/data/flinx/data/kb/`. Restart REX to use the new model.

5. Run `REXCmd` to observe new model results on sample documents from your domain, with entities that exist in the knowledge base. If there are too few results, update the `NIL_BIAS` coefficent value in `custom-kb/<lang_code>/params` file to with a number between 0 and 1. If you want to improve the results further, try to update the feature list in `/asset/config/features/<lang_code>` and train again.

**NOTE**
We recommend that after you run through the whole process once, try repeating steps 2 and 3 to see how different features and configurations affect the performance of your disambiguation model, until you arrive at a model that seems to work for you. Feel free to reach out to the customer engineering group at `<support@rosette.com>` at Basis Technology if you have questions.

### 6.2.4. Adding Entries to a Custom Knowledge Base

Once the disambiguation model for a custom knowledge base has been created, you can add entries without retraining the disambiguation model. The new entries must be similar to the entries that the model was trained on.

1. Create JSON files [35] containing the new entries.

2. Add the files to the directory: `<rex-je-root>/data/flinx/data/kb/<custom_kb_name>/`

3. Modify the entity extractor at runtime to use the files:

```
EntityExtractor#addKbEntities(String disambiguationModel,
LanguageCode language, File file)
EntityExtractor#addKbEntities(String disambiguationModel, Path path)
```

**NOTE**
If the new knowledge base entries **have different attributes** (such as having a prevalence score that the existing knowledge base entries do not have), then you should retrain the disambiguation model with the new entries.

### 6.2.5. Custom Knowledge Base Connectors

You can add a custom Knowledge Base Connector that retrieves linking information from a knowledge base backed by any external data source or code. Knowledge Base Connectors are an advanced customization feature, and should be used with care; unoptimized connector implementations can greatly affect REX's performance.

Custom Knowledge Base Connectors are created by implementing the `com.basistech.rosette.flinx.api.internal.KnowledgeBase interface`. The interface contains functions that provide information about entities to the linker disambiguation model features, as described in Train the Model [39]. At a minimum, a connector must implement the `findCandidates` candidates that identifies potential candidates for linking in a document, the `lookupEntityType` and `getLabels` functions that return basic information about entities supported by the Knowledge Base, and the `getContextVector` function that provides a general context vector for entities.

#### SQLite Connector Sample

A complete example of a custom Knowledge Base Connector that uses a knowledge base backed by a SQLite database is provided in the `samples` directory of your REX installation.

**WARNING**
The sample uses a SQLite database which is simple and easy to install for demonstration purposes, but is not optimized for performance. Your installation will depend on your requirements and existing knowledge bases.

The file `SQLiteKnowledgeBase.java` is well commented and can be used as an example for how to build your own connector.

To build and run the sample, follow the directions in the `README.md` file in the repository. The files are dependent on the version of Rosette Server installed.

To build and run the sample, make sure you have a local version of SQLite installed, and then:

1. From the `samples/sqlite-kb-connector` directory, run `mvn install`.
2. Copy the built `sqlite-kb-connector-1.0.jar` from the `target` folder to `{rex-installation}/lib`.
3. Create a SQLite knowledge base: copy the contents of the sample's `kb` directory into a new directory under `{rex-installation}/data/flinx/data/kb`. You may name this new folder anything you like.

4. Inside the knowledge base directory, run `create-kb-database.sh custom-kb.db`. This will create a new SQLite database with the correct schema.

5. To put data in the knowledge base, use a database browser or connect to the database with the SQLite command-line utility, then add lines to the tables as follows to add an entity. The example code provides the commands using the SQLite command-line utility.

   - From a command-line, start the sqlite3 program and open the sample database.

     ```
     sqlite3
     .open custom-kb.db
     ```

   - In the `entities` table, add a line with your desired entity ID in the `entityId` field (for example, `E1`). The `entityNum` column should auto increment.

     ```
     insert into entities(entityNum, entityId) VALUES(1, 'E1');
     ```

   - In the `aliases` table, add lines for any alias you want the entity to have. For example, add a line with `entityId` set to `E1` and `alias` set to `John Doe`.

     ```
     insert into aliases(alias, entityId) VALUES ('John Doe', 'E1');
     ```

   - In the `canonicalNames` table, set the canonical name for the entity. For example, add a line with `entityId` set to `E1` and `canonicalName` set to `John Doe`.

     ```
     insert into canonicalNames(entityId, canonicalName) VALUES ('E1', 'Jonathan Doe');
     ```

   - In the `entityTypes` table, set the entity type for the entity. For example, add a line with `entityId` set to `E1` and `entityType` set to `PERSON`.

     ```
     insert into entityTypes(entityId, entityType) VALUES ('E1', 'PERSON');
     ```

   - In the `contextWords` table, set context words to help the linker recognize the entity, separated by spaces. For example, if John Doe works at Basis Technology and you expect them to appear in documents related to the company, you might add a line with `entityId` set to `E1` and `contextWords` set to `basis technology nlp language rex`.

     ```
     insert into contextWords(entityId, contextWords) VALUES ('E1', 'basis technology nlp language rex');
     ```

6. Once the database is in place, any extractions with the linker activated will use the connector to search the new SQLite-based knowledge base in addition to the standard Wikidata knowledge base.

# 7. Customizing Statistical Models with the Field Training Kit (FTK)

REX has a field training system to customize and retrain the statistical models to improve the extraction results in your domain. The two primary types of customization are:

- improving accuracy on data in your specific domain
- extracting additional entity types

The REX FTK is also used to create custom knowledge base models for linking, as described in Creating a Custom Knowledge Base for Linking [34].

If your domain of information is similar to the default REX domain, news stories, then we recommend that you first use Unsupervised Training [48] and unannotated text to retrain the models. If your domain is very different from the default REX, then we recommend Supervised Training [51] with annotated text to retrain the statistical models.

You can customize the models to statistically extract new types of entities. The default models extract the following entity types: Person, Location, Organization, Product, and Title. The models use linguistic context disambiguate between "Apple Inc." and an "apple". You can extract car parts, medical terms, weapons, and other entity types specific to your use case.

The FTK can be shipped as a Docker container or as a package of scripts (supported by only CentOS currently). The requisites and setup instruction of the two package is somewhat different, as discussed in Requirements [43] and the installation instructions for each package.

# 7.1. Installing the Field Training Kit

## 7.1.1. Requirements

### Basis Technology software

- Installed REX package
- Field training kit, one of:
  - `rex-training-docker-<version>.tar.gz`
  - `rex-training-<version>.tar.gz`.
- Field training language resources, `rex-training-lang-<lang>.tar.gz`, where `<lang>` is the language code in ISO 693-3 Language Codes [54].

### Third party software and hardware

- 64-bit processor architecture
- 50 GB of storage, minimum

For base Field Training Kit

- 12 GB of RAM, minimum
- Java 11 or above
- Python 3.5
- CentOS 6+

For Docker Field Training Kit

- Docker Engine (http://docs.docker.com/engine/installation)
- 12 GB of RAM on the Docker Machine

For example, if using Oracle VirtualBox as the VM platform that hosts Docker, run this command:

```
docker-machine create --driver virtualbox --virtualbox-memory "12288" default
```

**WARNING**
**Windows with Docker.**

Due to Docker limitations, the customization process may be slower and more complicated when running on Windows than with Mac or Linux. Plan your time and hardware resources accordingly.

### Data Training Files

Corpus of text in UTF-8, without markup.

This corpus may be annotated for Named Entities or left in its plain-text form, depending on the type of statistical customization (supervised or unsupervised) and whether you require an objective quantitative accuracy estimation.

We measure a corpus' size by the number of tokens, or words, that comprise it. This includes both entities and non-entities.

We recommend that for both customization methods that you annotate a corpus that meets the minimum required size, which depends on the type of customization. Annotate content from your domain that closely resembles the input that REX will process.

If you train REX with supervised training (or add a new entity type), you must have a separate annotated corpus for evaluation. You should annotate a large enough corpus for both training and evaluation.

## 7.1.2. Installing without Docker

The following section will guide you through the common setup steps that are needed for either kind of statistical training, annotation or evaluation.

**IMPORTANT**
Before installing the FTK, REX must already be installed in your environment.

Install the FTK:

1. Unzip the field training kit `rex-training-kit-<version>.tar.gz`. A directory named `basis` will be created where you unzipped the file. Let `<basis-ftk-path>` be the absolute path to the directory where the `basis` directory was created.

2. Create a directory named `asset`. Unzip the field training language resource `rex-training-lang-<lang>.tar.tz` within the `asset` directory.

From within the unzipped field training language resource:

1. Create a directory for each language `asset/input/<lang>`:

```
mkdir -p asset/input/<lang>
```

2. Set environment variables with `<rex-field-training-home>` being one directory above the `asset` directory and `<rex-installation-path>` being the root directory of the unzipped REX package `rex-je-<version>.zip`:

```
PATH=<basis-ftk-path>/basis/ftk/bin:$PATH
export REX_JE_ROOT=<rex-installation-path>
export ASSET=<rex-field-training-home>/asset
```

3. Copy your example text files to `<rex-field-training-home>/asset/input/<lang>`. Reminder:

> **IMPORTANT**
> The training process requires clean utf-8 input documents with no markup. If acquiring text from the web please make sure to remove HTML tags, javascript, CSS, metadata etc.

4. To display the usage menu:

```
ftk help
```

### 7.1.3. Installing with Docker

**Load the Docker image**

1. Create a working directory, which we will refer to as `<rex-field-training-home>`.

2. Load the image:

```
docker load -i rex-training-docker-<version>.tar.gz
```

3. Validate that the image is loaded:

```
docker images
```

Confirm there is an image named `rex-field-training` in the images list.

**Prepare the training files directory**

1. Make the directories `asset/input/<lang>` in `<rex-field-training-home>`.

```
mkdir -p asset/input/<lang>
```

   The `asset` directory will be mounted to the Docker container and is used for transferring data in and out of it.

2. Extract the language resource to `<rex-field-training-home>/asset`.

```
cd asset
tar xvf ../rex-training-lang-<lang>.tar.gz
```

   After extracting the files, make the `asset` directory and sub-directories writable by anyone.

3. Copy your example text files to `<rex-field-training- home>/asset/input/<lang>`.

> **IMPORTANT**
> The training process requires clean utf-8 input documents with no markup. If acquiring text from the web please make sure to remove HTML tags, javascript, CSS, metadata etc.

**Run the Docker container**

```
docker run -it --rm -v <local-asset-dir-full-path>:/asset
-v <rex-installation-path>:/basis/rex rex-field-training
```

*Optional*: If you're planning to perform any annotations using the bundled Brat server (see Annotating [49] below), please add port mapping to the `run` command:

```
docker run -it --rm -v <local-asset-dir-full-path>:/asset -v <rex-installation-path>:/basis/rex
-p <local_port_number>:8080 rex-field-training
```

Windows: Note the special syntax used on Windows machines to denote mounted path names:

```
docker run -it --rm -v '//<rex-field-training-home>:/asset'
-v '//<rex-installation-path>:/basis/rex' rex-field-training
```

An example filepath for Windows is `//c/Users/basis/asset:/asset`

> **TIP**
> You may find it convenient to run the container inside a `screen` session (or another terminal multiplexer of your choice), so you can later detach and reattach to your session from a terminal.

A startup message similar to the one below, should now appear in your console window, confirming that your Docker is set up correctly:

```
REX field training tools (docker) version
Please refer to the legal notices in /basis/ftk/dependencies/ThirdPartyLicenses.txt.
Copyright (c) 2016 Basis Technology Corporation All Rights Reserved.
Support@basistech.com
 http://www.basistech.com

brat instance started on port 8080 in this container. brat data: /asset/bratdata

Available commands:

Statistical model:
  generate-ngram, cluster-ngram, train-model, evaluate

Advanced tools: learning-curve

Linker model:
 generate-linker-binaries, train-linker-model

Binary gazetteer:
 build-binary-gazetteer
```

The previous command placed you at a shell prompt inside a running field training docker container. From here, you may proceed to perform supervised training, unsupervised training, evaluation, or annotation.

## 7.2. Using the Field Training Kit

The field training system enables you to customize and retrain the statistical models with your input to improve the extraction results in your domain. This customization expands the extraction to include entities REX has not encountered.

You have the option of retraining the models on unannotated (unsupervised) or annotated text (supervised). If your domain of information is similar to the default REX domain, news stories, then we recommend that you first use Unsupervised Training [48] and unannotated text to retrain the models. If your domain is very different from the default REX, then we recommend Supervised Training [51] with annotated text to retrain the statistical models.

The major benefit of unsupervised training is that the process does not require the human-intensive effort to annotate example data. The model will discover entities using the context of words within the plain text input. It will generate groupings of words that appear in similar contexts and assign them to the same cluster, like "Boston", "Texas", and "France". The model then uses that cluster information to extract entities from your input.

If your domain is very different from the news stories that REX was trained on, after performing Unsupervised Training, you can use Supervised Training [51] to improve the extraction results using annotated data from your domain. To perform supervised training, you need to annotate a corpus of text. You can use any annotation tool. The REX training system includes the Brat Rapid Annotation Tool [49].

## 7.3. Retraining Rex Models

**Generating N-Gram Distributions**

Taking the customer-provided example text from `/asset/input`, the system splits it into unigram and bigram counts. It breaks the input into sentences, then tokenizes and normalizes the input to begin generating the n-grams. Using Rosette Base Linguistics, we determine the normalized form for each input token. For some languages we use lemmatization to determine the dictionary-form of a token, and then disambiguate to return the correct meaning of the word from multiple lemmas.

Next, the system scans the normalized input to calculate the distribution of unigrams and bigrams. The new distributions are combined with the previously trained and annotated corpus in `/asset/corpora/<lang>/news/train`. The combined n-grams are placed in `/asset/combined/<lang>.ftk.{uni,bi}.gz`. The n-gram distributions are the building blocks for the clusters, which are created in the next step.

**Creating Clusters**

Reading the n-grams produced in `/asset/combined/<lang>.ftk.{uni,bi}.gz`, the system applies a mutual-information clustering algorithm to determine the correlation between n-grams. The system creates the final word clusters and stores them in `/asset/wordclasses/<lang>.gz`.

The algorithm groups up to one thousand words into a cluster, empirically designed to yield the optimal extraction accuracy.

**Training Models**

Using the newly created word clusters from the previous step, the system can now retrain the model. The system reads the word clusters generated and produces the files `model-{LE,BE}.bin` and `model_uc-{LE,BE}.bin` in `/asset/models/<lang>`.

The supervised training algorithm creates a sequence labeler, a structured, averaged perceptron. It is informed by the annotated text and the unlabeled clusters generated in the previous step.

The field training language resources from Basis Technology contain annotated news documents used for training the default REX statistical model. These documents are intentionally encrypted for Basis Technology use only.

The model is then compiled into a binary format, which can be read by the REX system. Once the binary model is deployed, REX will use it to extract entities from your domain.

## 7.4. Unsupervised Training

**Customer input**

- 100MB (or more) of text in UTF-8 without markup.
  This is the recommended minimum to improve the model's accuracy. The amount of input you need depends on how different the target domain is from the original text used to train the model.

**IMPORTANT**
Creating word clusters can take a significant amount of time to complete. Our tests indicate that retraining the models on high-resource languages like English, Spanish, and Chinese could take up to a few days to complete, when using server-grade machines.

### 7.4.1. Performing Unsupervised Training

1. Generate the n-gram distributions:

```
generate-ngram <lang>
```

The local `asset/generated` directory now contains compressed word ngrams statistics derived from your corpus.

2. Create the new clusters:

```
cluster-ngram <lang>
```

The local `<rex-field-training-home>/asset/wordclasses/` directory now contains the results of the clustering algorithm applied on the word ngrams.

3. Train the models on the new clusters:

```
train-model <lang>
```

The output is a one or more binary files that comprise the retrained statistical model (for languages in Latin script there will be two files created: a case-sensitive and a case-insensitive one.)

4. Copy the new models from `<rex-field-training-home>/asset/models/<lang>` to `<rex-home>/data/statistical/<lang>`, and remove the default models. Keep the naming convention of the models.
Alternatively, call `EntityExtractor.setStatisticalModels` method and point the system at the new model(s) for that language.

5. Continue to Evaluating [54] to measure the accuracy of the new statistical models on your input.

## 7.5. Annotating Documents

### 7.5.1. Annotating with Brat

Manual annotation of documents is required for the following use cases:

- When improving REX's accuracy via supervised training [51].
- When adding support for additional entity types [53].
- For obtaining a "golden" data set for use as an objective, quantitative evaluation [54] of REX's accuracy.

**TIP**
We strongly recommend that you first perform Unsupervised Training [48], which does not require any annotation, before attempting supervised training in order to improve the extraction results at a minimal human effort.

To help with the annotation process, the Brat Rapid Annotation Tool is included in the Docker field training container. Once you map to a port on your machine, you can access Brat with your web browser. See the Brat Manual for more information on using and customizing Brat.

When you've finished annotating the corpus, this guide will instruct you on how to convert the files to Rosette's Annotated Data Model (ADM) which REX subsequently uses to retrain the statistical models.

The following section guides you through the process of annotating data files using the bundled Brat server.

**TIP**

The bundled Brat server is configured for Left-To-Right languages by default. To annotate Right-To-Left languages, (Hebrew, Arabic, etc), edit `visual.conf` in your Brat collection (default `bratdata`) to include:

```
[options]
Text direction:rtl
```

**Instructions**

1. Create a world-writable directory in `<rex-field-training-home>/asset/` named `bratdata`:

   ```
   mkdir <rex-field-training-home>/asset/bratdata
   ```

2. Copy your corpus of plain, unannotated text (`<filename>.txt`) files into the `bratdata` directory. This can be a subset of the corpus you had previously placed in `asset/input/<lang>`.

3. For each `filename.txt` file, create an empty `<filename>.ann` file in the `bratdata` directory.

4. Make all of the individual files (`<filename>.txt` and `<filename>.ann`) as well as the directory itself readable and writable by anyone. For example:

   ```
   chmod -R ugo+rw <rex-field-training-home>/asset/bratdata
   ```

5. If you haven't done so already, run the Docker container while mapping the Brat port (8080) to a free port on your machine. See the General Setup Instructions [44] above.
   The container startup scripts populate the empty `asset/bratdata` directory with configuration files.

6. Open a browser and go to `http://<docker_machine_ip>:<local_port_number>`.
   Note that on systems that use a docker-machine you'll need to obtain its IP to be able to access it. See the Docker documentation for additional details.
   Follow the instructions to select the files to annotate.

7. In the top, right-hand corner, click Login. You will need to login to Brat every time you start the Docker container.

   - Username: brat
   - Password: brat

8. **If you're annotating for a new entity type**, open the `asset/bratdata/annotation.conf` file. Add the name of the new entity type to the list on a new line, for example "MEDICAL".

9. Annotate the examples with Brat in the browser.

The Brat output will be in the <filename>.ann files. Brat uses a 'standoff' annotation method where the original `.txt` files are read-only and the annotations are stored in the readable/writable `.ann` files, which represent the annotation in simple tabular format. Below is an excerpt of an `.ann` file that represents annotations of the entity type `FRUIT` in some Spanish text file:

```
T1      FRUIT 277 284   Manzana
T2      FRUIT 313 320   toronja
T3      FRUIT 571 579   manzanas
```

10. Create a directory for the ADM output files

```
mkdir /asset/<admoutput>
```

11. When you're done annotating, convert the Brat files to ADM using the `corpuscmd` utility:

```
corpuscmd Brat2Adm --bratInput /asset/bratdata --output /asset/<admoutput>
```

The output json-serialized-ADM files created will be have the `.txt.adm.json` extension.

12. Continue to Supervised Trainingt [51] to retrain the models, to Add a New Entity Type [53], or to Evaluate the Restrained Statistical Model [54].

### 7.5.2. Alternate Ways to Generate Annotated Data

You can also generate annotated data in ADM json format directly. This is the representation and serialization used by REX for training and evaluation. The output generated by `REXCmd`, REX's command-line utility, also uses json-serialized ADMs as its output. You may simply generate (or convert) your corpus following the `REXCmd` generated json as an example.

Some users find Brat's standoff annotation format (comprised of `.ann` and `.txt` files) a convenient way to represent annotations. If you wish to write Brat files yourself (without using the bundled Brat server) you may do so and then start the process discussed above from step 10.

Please contact `<support@basistech.com>` for more information about the Rosette Annotated Data Model and ADM files.

## 7.6. Supervised Training to Improve Accuracy in a Domain

You can improve the extraction of the default entity types from your input by customizing the statistical processor with annotated data. If your domain is drastically different from the default REX domain, then it will have a larger impact on the REX results.

### 7.6.1. Annotating Instructions

1. Setup Brat and annotate data. [49]
   For Supervised Training, we recommend that you annotate a corpus containing a minimum of 60,000 tokens. However, if your target domain is very different from the default REX domain then we recommend a larger corpus. The greater the difference between the domains, the more tokens necessary to create a new statistical model.

2. Copy your corpora of ADM files in a directory (`<train corpus path>`) accessible from the container. (You may want to replace the Basis-provided training corpus for "New Model Only" training explained below. To do so, copy your adm files to `/asset/corpora/<lang>/news/train`, and remove *.enc files in the directory. Please make sure your corpus contains at least 1000 unique tokens.)

3. Put all available raw, unannotated plain text into `<rex-field-training-home>/asset/input/<lang>/`.

4. Setup Docker [45]

**Supervised Training**

1. From your Docker console, generate the n-gram distributions:

```
generate-ngram <lang>
```

2. From your Docker console, create the new clusters:

```
cluster-ngram <lang>
```

**Instructions for New or Mixed Statistical Models**

You must choose whether to extract entities using both the new and the default statistical models together, which we call model mixing, or if you want to exclusively use the new statistical Model.

With model mixing, REX runs both the new and the default models in parallel and uses the redactor module to adjudicate the overlapping results.

**Model Mixing**

1. From your Docker console, train the model on the new clusters:

```
train-model -T <entity types> -t <train corpus path> <lang>
```

   where `<entity types>` is a comma-delimited list of entity types in the annotated corpus, e.g. `-T FRUIT,DRUGS` The output is a binary file of the retrained statistical model. For languages in Latin script there will be two models: case-sensitive and case-insensitive.

2. Copy the new models from `<rex-field-training-home>/asset/models/<lang>` to `<rex-home>/data/statistical/<lang>` with the existing models. Keep the naming convention of the models.
   Alternatively, call `EntityExtractor.setStatisticalModels` method and point the system at both the new and default model(s) for that language. Only pointing to the new model(s) will overwrite the default model(s).

**NOTE**
You can Customize the Redactor [26] to favor output from the new statistical model(s).

3. Once you have retrained the models, proceed to Evaluating the Retrained Statistical Model [54].

**Using New Model Only**

1. From your Docker console, train the model on the new clusters:

```
train-model -w /asset/wordclasses/<lang>.customer.gz -T <entity types>  -t <train corpus path> <lang>
```

The output is a binary file of the retrained statistical model. For languages in Latin script there will be two statistical models: case-sensitive and case-insensitive.

2. Copy the new models from `<rex-field-training-home>/asset/models/<lang>` to `<rex-home>/data/statistical/<lang>`, and remove the default models. Keep the naming convention of the models.
   Alternatively, call `EntityExtractor.setStatisticalModels` method and point the system at the new model(s) for that language.

3. Once you have retrained the models, proceed to Evaluating the Retrained Statistical Model [54].

**TIP**

**Model Naming Convention**

The prefix must be `model.` and the suffix must be `-LE.bin`. Any alphanumeric ASCII characters are allowed in between.

Example valid model names:

- model.fruit-LE.bin
- model.customer4-LE.bin

## 7.7. Adding New Statistical Entity Types

The REX statistical processor can also be retrained to extract types of entities that are specific to your domain. This customization functions best with a distinct category of entities which do not have a set pattern, or which have an unlimited number of possibilities. If the entities appear in a pattern, such as license plate numbers, then you should Create a Regex [23] for extracting plate numbers. If there is a finite number of possible entities, such as movies that have won an Academy Award, then you should Create a Gazetteer [19] to extract the movie titles.

When you annotate content, you can add new entity types to the Brat configuration. Then REX will take the annotated content and statistically extract domain-specific entities, such as medical terms.

### 7.7.1. Instructions

1. Setup Brat and annotate data [49]
   If you are adding a new entity type, we recommend that you annotate a corpus with a minimum of 100,000 tokens. The corpus is much larger as REX needs more data to accurately extract a new entity type and to avoid conflicts with similar entities. More annotated data increases accuracy for that new entity type.

2. Retrain the model, following the steps in Supervised Training [51].

3. Continue to Evaluation [54] to review the new model's performance.

## 7.8. Evaluating the Retrained Statistical Model

### 7.8.1. Initial Accuracy Quick Test

When the statistical training is complete, the system will evaluate the resulting model against a standard data set and report F-scores. These numbers are part of the output generated by `train-model` and measure the accuracy of the freshly retrained model on the original evaluation input from Basis Technology. They are a useful quick test to ensure that the models are functioning properly. If you encounter dramatically low or unexpectedly high F-scores, it could be a sign that something went wrong in the training process.

Since the model has been adapted to your target domain, these F-scores are not a valid reflection of the actual accuracy of the new model's performance. It is expected that the scores may be moderately lower because the retrained model is no longer adapted for news documents.

### 7.8.2. Measuring Accuracy on Your Data

To perform a methodical evaluation of the performance of the default or customized models on **your** dataset, you must first annotate a corpus of text from your domain. Do not reuse documents that were already used for training.

**Instructions**

1. Annotate data. [49]
   For evaluating the performance of REX, we recommend that you annotate a corpus with a minimum of 30,000 tokens. A larger corpus will produce more statistically significant F-scores.

2. Replace the files in `/asset/corpora/<lang>/news/eval` with the new ADM files.

3. If you haven't done so already, make sure the Field Training Docker [45] container is ready for use.

4. From your Docker console, evaluate the models on your domain:

   ```
   evaluate <lang>
   ```

   As in `train-model`, you can use a specific set of entity types with `-T <entity types>`.
   The `-v` flag will produce verbose output, which is especially useful when using custom entity types

5. The Docker console will report the F-scores, precision, and recall.

# 8. ISO 639-3 Language Codes

REX uses ISO 639-3 codes to specify the language of the input text.

| Language | Code |
|---|---|
| Arabic | `ara` |
| Chinese, Script-insensitive** | `zho**` |
| Chinese, Simplified | `zhs` |
| Chinese, Traditional | `zht` |
| Dutch | `nld` |
| English | `eng` |

| Language | Code |
|---|---|
| English Uppercase** | uen |
| French | fra |
| German | deu |
| Hebrew | heb |
| Hungarian | hun |
| Indonesian | ind |
| Italian | ita |
| Japanese | jpn |
| Korean | kor |
| Malay, Standard | zsm |
| Persian (Iranian Persian, Afghan Persian) | fas |
| Portuguese | por |
| Pashto | pus |
| Russian | rus |
| Spanish | spa |
| Swedish | swe |
| Urdu | urd |
| Vietnamese | vie |
| Generic, Cross-Language | xxx |

# 9. Tcl Regex Format

The Pattern Matching Processor [23] uses the Tcl regular expression engine to identify named entities in input text. To see the named entity types that the Pattern Matching Processor with the shipped regexes file returns, see Language Support [5] of Named Entities. For background information about adding your own regexes, see Accept Regex [23].

For information on Tcl syntax, see the Tcl re_syntax Manual Page .

REX modifies the regex matcher so that `\n` in a regex expression matches straight new lines (`\n`\), carriage returns (`\r`), or a combination of both (`\r\n`). Regardless of what is matches, offsets and lengths in the result will match the input document.

## 9.1. Tcl License

This software is copyrighted by the Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, ActiveState Corporation and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS

# 10. Solr Plugin

A Solr plugin is available for REX. The plugin integrates into Solr's update chain, processing requested text fields and creating new fields containing extracted entities.

You can find a short tutorial walking through a basic installation and configuration of a simple REX sample in the `doc` directory inside the REX Solr Plugin package.

## 10.1. Installing the Solr Plugin

To install the Solr plugin, copy all files from the `lib` directory inside the REX Solr Plugin package into the `lib` directory of your Solr core. In addition, a reference to your REX installation is required in your core's `solrconfig.xml` file:

```
   <lib dir="${rexje.root}/lib" regex=".*\.jar"/>
```

Either set `rexje.root` in your Solr installation to point to your REX installation directory, or change `${rexje.root}` in the line above directly.

To use the plugin, a processor chain using it should be configured in `solrconfig.xml`. You can either create a special processor chain or integrate it into an existing one. The simplest configuration looks like this:

```
   <updateRequestProcessorChain name="rex">
   <processor class="com.basistech.rosette.solr.EntityExtractorUpdateProcessorFactory">
       <str name="rootDirectory">${rexje.root}</str>
       <str name="fields">text_eng</str>
   </processor>
   <processor class="solr.LogUpdateProcessorFactory"/>
   <processor class="solr.RunUpdateProcessorFactory"/>
</updateRequestProcessorChain>
```

`rootDirectory` and `fields` are mandatory parameters. `rootDirectory` should point to your REX installation directory. `fields` instructs the plugin which document fields to process, as described below.

The REX plugin outputs new fields once it is run. Make sure your schema allows for dynamic fields to be created, or configure it ahead of time with the dynamic fields required. REX output fields are multi-valued string fields. They are named the same as the field they're processing, postfixed with `_REX_{ENTITYTYPE}`. For example, in the simple processor chain above, a field called `text_eng_REX_PERSON` might be created. A schema entry for this (and other types for the `text_eng` field) can be set up like this in your schema definition:

```
<dynamicField name="text_eng_REX_*" type="string" indexed="true" stored="true" multiValued="true"/>
```

## 10.2. Solr Plugin Docker Container

The REX SOLR plugin is also available as a docker container. To install it, unzip the REX SOLR plugin container distribution package and run:

```
docker load -i rex-solr-docker.tar.gz
```

To run the image with its default configuration containing the demo SOLR project, locate your REX license file (rlp-license.xml), and \ run

```
REX_LICENSE_PATH=[path_to_license_file]/rlp-license.xml docker-compose up
```

The SOLR instance inside the image should run on its default port, 8983, with the REX SOLR demo active.

The default `docker-compose.yaml` provided loads the image with English data files. To use other languages, edit `docker-compose.yaml`\ before running `docker-compose up` to run the image:

1. Add a service reference to the desired language, following this template (English should already be defined in the default configur\ ation and can be copied and modified): `root-rex-[lang]-[version]: image: rosette/root-rex-[lang]:[version] volumes: - rosette-roots-vol:/roots-vol`

2. Add the new service reference to the `depends_on` section of the `rex-solr-demo` reference, as follows: `rex-solr-demo: image: rex-solr-demo:latest depends_on: - root-rex-[lang]-[version] - root-rex-[lang]-[version] ... - root-rex-root-[version]`

To modify the SOLR configuration to use your own instead of the demo, connect to the running image: identify its process id by running\ `docker ps` and then run

```
docker exec -it [pid] /bin/bash
```

The SOLR installation can be found in `var/solr`. You may also mount external database storage locations using regular Docker commands\ and reference them from SOLR configuration.

## 10.3. Using the plugin

When the plugin is run as part of a Solr processor chain, REX processes all fields listed in the `fields` parameter in the plugin configuration, and populates multi-valued entity fields for every entity type extracted.

Fields processed by the plugin *must* conform to a naming convention, and be post fixed with an underscore followed by the ISO639 language code the field's text is in. For example, REX can run on fields named `text_eng` or `article_content_jpn`. You should either set up your fields with these names in the original schema, or use other update processor plugins to identify a field's language and copy its content to a dynamic field with a compatible name.

Every configuration option available in the SDK for the `EntityExtractor` class is available in the plugin, save for those related to entity linking, which is not currently supported. Also not supported are the addition of dynamic individual regex expressions or gazetteer entries. Reference the chart below for the appropriate parameter to add to the processor definition in `solrconfig.xml`. All parameters are strings.

For example, setting up the plugin to use confidence threshold might look like this:

```
   <processor class="com.basistech.rosette.solr.EntityExtractorUpdateProcessorFactory">
   <str name="rootDirectory">${rexje.root}</str>
   <str name="fields">text_eng</str>
   <str name="calculateConfidence">True</str>
   <str name="confidenceThreshold">0.8</str>
</processor>
```

| **`EntityExtractor` API** | **Solr processor parameter name** | **Comments** |
|---|---|---|
| `setOverlayDataDirectory` | `overlayDataDirectory` | |
| `setLicense` | `license` | A valid license file. Passing a isn't supported. |
| `setTruecaserRootDirectory` | `truecaserRootDirectory` | |
| `setBaseLinguisticsRootDirectory` | `baseLinguisticsRootDirectory` | |
| `setMaxEntityTokens` | `maxEntityTokens` | |
| `setCalculateConfidence` | `calculateConfidence` | `True` or `False` |
| `setCalculateSalience` | `calculateSalience` | `True` or `False` |
| `disableStatisticalResultCleaner` | `disableStatisticalResultCleaner` | `True` or `False` |
| `setConfidenceThreshold` | `confidenceThreshold` | |
| `setStatisticalModel` | `statisticalModels` | You may pass multiple statist parameter should be formatt values specfying language, c and the model file, separated Case-sensitivity can be `auto` `caseInsensitive` or `case` For example, setting two mo sensitive English and Japane like : `eng,caseSensitive` `model.bin,jpn,automat` `model.bin` |

| `EntityExtractor` API | Solr processor parameter name | Comments |
|---|---|---|
| `addGazetteer` | `addGazetteers` | Multiple gazetteer files can b with this parameter. Gazette defined by four comma-sepa language, file, accept/reject ( accept, `False` for reject), an sensitivity ('True' for case-se for case-insensitive). For exa two accept gazetteer files fo and Japanese might look like `gaz.bin,True,True,jpn gaz,True,False`. |
| `setAllowPartialGazetteerMatches` | `allowPartialGazetteerMatches` | `True` or `False` |
| `addRegularExpressions` | `addRegularExpressions` | Multiple regular expression f added with this parameter. E pair of comma-separated val and whether this is an accep file (`True` for accept, `False` For example, adding two acc files for English and Japanese look like: `eng-regex.xml, regex.xml,True`. |
| `setRedactorWeights` | `redactorWeights` | |
| `setIndoc` | `indoc` | Can be `HIGH`, `STANDARD`, `STANDARD_MINUS` or `NULL` |
| `setSnapToTokenBoundaries` | `snapToTokenBoundaries` | `True` or `False` |
| `setCaseSensitivity` | `caseSensitivity` | Can be `automatic`, `caseSe` `caseInsensitive` |
| `setMaxResolvedEntities` | `maxResolvedEntities` | |
| `setProcessors` | `processors` | List of processors separated Can be `statistical`, `acce` `acceptRegex`, `rejectGaz` `rejectRegex`, `kbLinker`, `pronominalResolver`, `re` `joiner`, or `deepNeuralNe` |
| `setUseDeepNeuralNetworkProcessor` | `useDeepNeuralNetworkProcessor` | `True` or `False` |
| `addJoinerRules` | `addJoinerRules` | |
| `setResolvePronouns` | `resolvePronouns` | `True` or `False` |
| `setCustomProcessors` | `customProcessors` | |
| `registerCustomProcessorClass` | `registerCustomProcessorClass` | |
| `setUseDefaultConfidence` | `useDefaultConfidence` | `True` or `False` |
| `setStructuredRegionProcessingType` | `structuredRegionProcessingType` | Can be one of `nerModel`, `ne` or `none` |
| `setStructuredRegionProcessingSentenceTokensMin` | `structuredRegionProcessingSentenceTokensMin` | |

# 11. Entity Types and DBpedia Types

## 11.1. Entity Types and DBpedia Types

| Entity type | DBpedia type |
|---|---|
| ACTIVITY | Activity |
| ACTIVITY | Activity/Game |

| Entity type | DBpedia type |
|---|---|
| ACTIVITY | Activity/Sales |
| ACTIVITY | Activity/Sport |
| ACTIVITY | Activity/Sport/Athletics |
| ACTIVITY | Activity/Sport/TeamSport |
| ACTIVITY | Activity |
| ANATOMY | AnatomicalStructure |
| ANATOMY | AnatomicalStructure/Artery |
| ANATOMY | AnatomicalStructure/BloodVessel |
| ANATOMY | AnatomicalStructure/Bone |
| ANATOMY | AnatomicalStructure/Brain |
| ANATOMY | AnatomicalStructure/Embryology |
| ANATOMY | AnatomicalStructure/Ligament |
| ANATOMY | AnatomicalStructure/Lymph |
| ANATOMY | AnatomicalStructure/Muscle |
| ANATOMY | AnatomicalStructure/Nerve |
| ANATOMY | AnatomicalStructure/Vein |
| DISEASE | Disease |
| EVENT | Event |
| EVENT | Event/Competition |
| EVENT | Event/Competition/Contest |
| EVENT | Event/LifeCycleEvent |
| EVENT | Event/LifeCycleEvent/PersonalEvent |
| EVENT | Event/NaturalEvent |
| EVENT | Event/NaturalEvent/Earthquake |
| EVENT | Event/NaturalEvent/SolarEclipse |
| EVENT | Event/NaturalEvent/StormSurge |
| EVENT | Event/PenaltyShootOut |
| EVENT | Event/SocietalEvent |
| EVENT | Event/SocietalEvent/AcademicConference |
| EVENT | Event/SocietalEvent/Attack |
| EVENT | Event/SocietalEvent/Convention |
| EVENT | Event/SocietalEvent/Election |
| EVENT | Event/SocietalEvent/FilmFestival |
| EVENT | Event/SocietalEvent/HistoricalEvent |
| EVENT | Event/SocietalEvent/Meeting |
| EVENT | Event/SocietalEvent/MilitaryConflict |
| EVENT | Event/SocietalEvent/MusicFestival |
| EVENT | Event/SocietalEvent/Rebellion |
| EVENT | Event/SocietalEvent/SpaceMission |
| EVENT | Event/SocietalEvent/SportsEvent |
| EVENT | Event/SocietalEvent/SportsEvent/CyclingCompetition |
| EVENT | Event/SocietalEvent/SportsEvent/FootballMatch |
| EVENT | Event/SocietalEvent/SportsEvent/GrandPrix |
| EVENT | Event/SocietalEvent/SportsEvent/InternationalFootballLeagueEvent |
| EVENT | Event/SocietalEvent/SportsEvent/MixedMartialArtsEvent |
| EVENT | Event/SocietalEvent/SportsEvent/NationalFootballLeagueEvent |
| EVENT | Event/SocietalEvent/SportsEvent/Olympics |

| Entity type | DBpedia type |
|---|---|
| EVENT | Event/SocietalEvent/SportsEvent/Olympics/OlympicEvent |
| EVENT | Event/SocietalEvent/SportsEvent/Race |
| EVENT | Event/SocietalEvent/SportsEvent/Race/CyclingRace |
| EVENT | Event/SocietalEvent/SportsEvent/Race/HorseRace |
| EVENT | Event/SocietalEvent/SportsEvent/Race/MotorRace |
| EVENT | Event/SocietalEvent/SportsEvent/Tournament |
| EVENT | Event/SocietalEvent/SportsEvent/Tournament/GolfTournament |
| EVENT | Event/SocietalEvent/SportsEvent/Tournament/SoccerTournament |
| EVENT | Event/SocietalEvent/SportsEvent/Tournament/TennisTournament |
| EVENT | Event/SocietalEvent/SportsEvent/Tournament/WomensTennisAssociationTournament |
| EVENT | Event/SocietalEvent/SportsEvent/WrestlingEvent |
| EVENT | Holiday |
| EVENT | SportsSeason |
| EVENT | SportsSeason/MotorsportSeason |
| EVENT | SportsSeason/SportsTeamSeason |
| EVENT | SportsSeason/SportsTeamSeason/BaseballSeason |
| EVENT | SportsSeason/SportsTeamSeason/FootballLeagueSeason |
| EVENT | SportsSeason/SportsTeamSeason/FootballLeagueSeason/NationalFootballLeagueSeason |
| EVENT | SportsSeason/SportsTeamSeason/NCAATeamSeason |
| EVENT | SportsSeason/SportsTeamSeason/SoccerClubSeason |
| EVENT | SportsSeason/SportsTeamSeason/SoccerLeagueSeason |
| EVENT | Statistic |
| EVENT | TimePeriod |
| EVENT | TimePeriod/CareerStation |
| EVENT | TimePeriod/CareerStation/MilitaryService |
| EVENT | TimePeriod/GeologicalPeriod |
| EVENT | TimePeriod/HistoricalPeriod |
| EVENT | TimePeriod/PeriodOfArtisticStyle |
| EVENT | TimePeriod/PrehistoricalPeriod |
| EVENT | TimePeriod/ProtohistoricalPeriod |
| EVENT | TimePeriod/Reign |
| EVENT | TimePeriod/Tenure |
| EVENT | TimePeriod/Year |
| EVENT | TimePeriod/YearInSpaceflight |
| EVENT | UnitOfWork/Case |
| EVENT | UnitOfWork/Case/LegalCase |
| EVENT | UnitOfWork/Case/LegalCase/SupremeCourtOfTheUnitedStatesCase |
| EVENT | UnitOfWork/Project |
| EVENT | UnitOfWork/Project/ResearchProject |
| EVENT | E4_Period |
| FOOD | Food |
| FOOD | Food/Beverage |
| FOOD | Food/Beverage/Beer |
| FOOD | Food/Beverage/Vodka |
| FOOD | Food/Beverage/Wine |
| FOOD | Food/Beverage/Wine/ControlledDesignationOfOriginWine |
| FOOD | Food/Cheese |

| Entity type | DBpedia type |
|---|---|
| IDENTIFIER | Identifier |
| IDENTIFIER | Identifier/TopLevelDomain |
| LANGUAGE | Language |
| LANGUAGE | Language/ProgrammingLanguage |
| LOCATION | ElectricalSubstation |
| LOCATION | Place |
| LOCATION | Place/ArchitecturalStructure |
| LOCATION | Place/ArchitecturalStructure/AmusementParkAttraction |
| LOCATION | Place/ArchitecturalStructure/AmusementParkAttraction/RollerCoaster |
| LOCATION | Place/ArchitecturalStructure/AmusementParkAttraction/WaterRide |
| LOCATION | Place/ArchitecturalStructure/Arena |
| LOCATION | Place/ArchitecturalStructure/Building |
| LOCATION | Place/ArchitecturalStructure/Building/Casino |
| LOCATION | Place/ArchitecturalStructure/Building/Castle |
| LOCATION | Place/ArchitecturalStructure/Building/Factory |
| LOCATION | Place/ArchitecturalStructure/Building/HistoricBuilding |
| LOCATION | Place/ArchitecturalStructure/Building/Hospital |
| LOCATION | Place/ArchitecturalStructure/Building/Hotel |
| LOCATION | Place/ArchitecturalStructure/Building/Museum |
| LOCATION | Place/ArchitecturalStructure/Building/Prison |
| LOCATION | Place/ArchitecturalStructure/Building/ReligiousBuilding |
| LOCATION | Place/ArchitecturalStructure/Building/ReligiousBuilding/Church |
| LOCATION | Place/ArchitecturalStructure/Building/ReligiousBuilding/Monastery |
| LOCATION | Place/ArchitecturalStructure/Building/ReligiousBuilding/Mosque |
| LOCATION | Place/ArchitecturalStructure/Building/ReligiousBuilding/Shrine |
| LOCATION | Place/ArchitecturalStructure/Building/ReligiousBuilding/Synagogue |
| LOCATION | Place/ArchitecturalStructure/Building/ReligiousBuilding/Temple |
| LOCATION | Place/ArchitecturalStructure/Building/Restaurant |
| LOCATION | Place/ArchitecturalStructure/Building/ShoppingMall |
| LOCATION | Place/ArchitecturalStructure/Building/Skyscraper |
| LOCATION | Place/ArchitecturalStructure/Building/Venue |
| LOCATION | Place/ArchitecturalStructure/Building/Venue/Cinema |
| LOCATION | Place/ArchitecturalStructure/Building/Venue/Stadium |
| LOCATION | Place/ArchitecturalStructure/Building/Venue/Theatre |
| LOCATION | Place/ArchitecturalStructure/Infrastructure |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/Airport |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/Dam |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/Dike |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/LaunchPad |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/Lock |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/Port |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/PowerStation |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/PowerStation/NuclearPowerStation |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/RestArea |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/RouteOfTransportation |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/RouteOfTransportation/Bridge |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/RouteOfTransportation/RailwayLine |

| Entity type | DBpedia type |
|---|---|
| LOCATION | Place/ArchitecturalStructure/Infrastructure/RouteOfTransportation/RailwayTunnel |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/RouteOfTransportation/Road |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/RouteOfTransportation/RoadJunction |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/RouteOfTransportation/RoadTunnel |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/RouteOfTransportation/WaterwayTunnel |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/Station |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/Station/MetroStation |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/Station/RailwayStation |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/Station/RouteStop |
| LOCATION | Place/ArchitecturalStructure/Infrastructure/Station/TramStation |
| LOCATION | Place/ArchitecturalStructure/MilitaryStructure |
| LOCATION | Place/ArchitecturalStructure/MilitaryStructure/Fort |
| LOCATION | Place/ArchitecturalStructure/Mill |
| LOCATION | Place/ArchitecturalStructure/Mill/Treadmill |
| LOCATION | Place/ArchitecturalStructure/Mill/Watermill |
| LOCATION | Place/ArchitecturalStructure/Mill/WindMotor |
| LOCATION | Place/ArchitecturalStructure/Mill/Windmill |
| LOCATION | Place/ArchitecturalStructure/Monument |
| LOCATION | Place/ArchitecturalStructure/Monument/GraveMonument |
| LOCATION | Place/ArchitecturalStructure/Monument/Memorial |
| LOCATION | Place/ArchitecturalStructure/Pyramid |
| LOCATION | Place/ArchitecturalStructure/SportFacility |
| LOCATION | Place/ArchitecturalStructure/SportFacility/CricketGround |
| LOCATION | Place/ArchitecturalStructure/SportFacility/GolfCourse |
| LOCATION | Place/ArchitecturalStructure/SportFacility/RaceTrack |
| LOCATION | Place/ArchitecturalStructure/SportFacility/RaceTrack/Racecourse |
| LOCATION | Place/ArchitecturalStructure/SportFacility/SkiArea |
| LOCATION | Place/ArchitecturalStructure/SportFacility/SkiArea/SkiResort |
| LOCATION | Place/ArchitecturalStructure/Square |
| LOCATION | Place/ArchitecturalStructure/Tower |
| LOCATION | Place/ArchitecturalStructure/Tower/Lighthouse |
| LOCATION | Place/ArchitecturalStructure/Tower/WaterTower |
| LOCATION | Place/ArchitecturalStructure/Tunnel |
| LOCATION | Place/ArchitecturalStructure/Zoo |
| LOCATION | Place/CelestialBody |
| LOCATION | Place/CelestialBody/Asteroid |
| LOCATION | Place/CelestialBody/Constellation |
| LOCATION | Place/CelestialBody/Galaxy |
| LOCATION | Place/CelestialBody/Planet |
| LOCATION | Place/CelestialBody/Satellite |
| LOCATION | Place/CelestialBody/Satellite/ArtificialSatellite |
| LOCATION | Place/CelestialBody/Star |
| LOCATION | Place/CelestialBody/Star/BrownDwarf |
| LOCATION | Place/CelestialBody/Swarm |
| LOCATION | Place/CelestialBody/Swarm/Globularswarm |
| LOCATION | Place/CelestialBody/Swarm/Openswarm |
| LOCATION | Place/Cemetery |

| Entity type | DBpedia type |
|---|---|
| LOCATION | Place/ConcentrationCamp |
| LOCATION | Place/CountrySeat |
| LOCATION | Place/Garden |
| LOCATION | Place/HistoricPlace |
| LOCATION | Place/Mine |
| LOCATION | Place/Mine/CoalPit |
| LOCATION | Place/NaturalPlace |
| LOCATION | Place/NaturalPlace/Archipelago |
| LOCATION | Place/NaturalPlace/Beach |
| LOCATION | Place/NaturalPlace/BodyOfWater |
| LOCATION | Place/NaturalPlace/BodyOfWater/Bay |
| LOCATION | Place/NaturalPlace/BodyOfWater/Lake |
| LOCATION | Place/NaturalPlace/BodyOfWater/Ocean |
| LOCATION | Place/NaturalPlace/BodyOfWater/Sea |
| LOCATION | Place/NaturalPlace/BodyOfWater/Stream |
| LOCATION | Place/NaturalPlace/BodyOfWater/Stream/Canal |
| LOCATION | Place/NaturalPlace/BodyOfWater/Stream/River |
| LOCATION | Place/NaturalPlace/Cape |
| LOCATION | Place/NaturalPlace/Cave |
| LOCATION | Place/NaturalPlace/Crater |
| LOCATION | Place/NaturalPlace/Crater/LunarCrater |
| LOCATION | Place/NaturalPlace/Desert |
| LOCATION | Place/NaturalPlace/Forest |
| LOCATION | Place/NaturalPlace/Glacier |
| LOCATION | Place/NaturalPlace/HotSpring |
| LOCATION | Place/NaturalPlace/Mountain |
| LOCATION | Place/NaturalPlace/MountainPass |
| LOCATION | Place/NaturalPlace/MountainRange |
| LOCATION | Place/NaturalPlace/Valley |
| LOCATION | Place/NaturalPlace/Volcano |
| LOCATION | Place/Park |
| LOCATION | Place/PopulatedPlace |
| LOCATION | Place/PopulatedPlace/Agglomeration |
| LOCATION | Place/PopulatedPlace/Community |
| LOCATION | Place/PopulatedPlace/Continent |
| LOCATION | Place/PopulatedPlace/Country |
| LOCATION | Place/PopulatedPlace/Country/HistoricalCountry |
| LOCATION | Place/PopulatedPlace/GatedCommunity |
| LOCATION | Place/PopulatedPlace/Intercommunality |
| LOCATION | Place/PopulatedPlace/Island |
| LOCATION | Place/PopulatedPlace/Island/Atoll |
| LOCATION | Place/PopulatedPlace/Locality |
| LOCATION | Place/PopulatedPlace/Region |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/ClericalAdministrativeRegion |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/ClericalAdministrativeRegion/Deanery |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/ClericalAdministrativeRegion/Diocese |

| Entity type | DBpedia type |
|---|---|
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/ClericalAdministrativeRegion/Parish |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/Arrondissement |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/Canton |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/Department |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/Department/OverseasDepartment |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/District |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/District/HistoricalDistrict |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/DistrictWaterBoard |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/MicroRegion |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/Municipality |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/Municipality/FormerMunicipality |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/Prefecture |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/Province |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/Province/HistoricalProvince |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/Regency |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/GovernmentalAdministrativeRegion/SubMunicipality |
| LOCATION | Place/PopulatedPlace/Region/AdministrativeRegion/HistoricalAreaOfAuthority |
| LOCATION | Place/PopulatedPlace/Region/HistoricalRegion |
| LOCATION | Place/PopulatedPlace/Region/NaturalRegion |
| LOCATION | Place/PopulatedPlace/Settlement |
| LOCATION | Place/PopulatedPlace/Settlement/City |
| LOCATION | Place/PopulatedPlace/Settlement/City/Capital |
| LOCATION | Place/PopulatedPlace/Settlement/City/CapitalOfRegion |
| LOCATION | Place/PopulatedPlace/Settlement/CityDistrict |
| LOCATION | Place/PopulatedPlace/Settlement/HistoricalSettlement |
| LOCATION | Place/PopulatedPlace/Settlement/Town |
| LOCATION | Place/PopulatedPlace/Settlement/Village |
| LOCATION | Place/PopulatedPlace/State |
| LOCATION | Place/PopulatedPlace/Street |
| LOCATION | Place/PopulatedPlace/Territory |
| LOCATION | Place/PopulatedPlace/Territory/OldTerritory |
| LOCATION | Place/ProtectedArea |
| LOCATION | Place/SiteOfSpecialScientificInterest |
| LOCATION | Place/WineRegion |
| LOCATION | Place/WorldHeritageSite |
| MEASURE | Altitude |
| MEASURE | Area |
| MEASURE | Depth |
| MEASURE | GrossDomesticProduct |
| MEASURE | GrossDomesticProductPerCapita |
| MEASURE | Population |
| MEASURE | SportCompetitionResult |
| MEASURE | SportCompetitionResult/OlympicResult |
| MISC | Thing |

| Entity type | DBpedia type |
|---|---|
| MISC | Agent |
| MISC | Agent/Employer |
| MISC | Agent/Organisation/TermOfOffice |
| MISC | Award |
| MISC | Award/Decoration |
| MISC | Award/NobelPrize |
| MISC | Blazon |
| MISC | ChartsPlacements |
| MISC | Colour |
| MISC | Demographics |
| MISC | ElectionDiagram |
| MISC | EthnicGroup |
| MISC | Flag |
| MISC | GeneLocation |
| MISC | GeneLocation/HumanGeneLocation |
| MISC | GeneLocation/MouseGeneLocation |
| MISC | List |
| MISC | List/TrackList |
| MISC | Media |
| MISC | MedicalSpecialty |
| MISC | Medicine |
| MISC | Name |
| MISC | Name/GivenName |
| MISC | Name/Surname |
| MISC | PublicService |
| MISC | Relationship |
| MISC | SportCompetitionResult/SnookerWorldRanking |
| MISC | TopicalConcept |
| MISC | TopicalConcept/AcademicSubject |
| MISC | TopicalConcept/CardinalDirection |
| MISC | TopicalConcept/Fashion |
| MISC | TopicalConcept/Genre |
| MISC | TopicalConcept/Genre/ArtisticGenre |
| MISC | TopicalConcept/Genre/LiteraryGenre |
| MISC | TopicalConcept/Genre/MovieGenre |
| MISC | TopicalConcept/Genre/MusicGenre |
| MISC | TopicalConcept/Ideology |
| MISC | TopicalConcept/MathematicalConcept |
| MISC | TopicalConcept/PhilosophicalConcept |
| MISC | TopicalConcept/PoliticalConcept |
| MISC | TopicalConcept/ScientificConcept |
| MISC | TopicalConcept/Standard |
| MISC | TopicalConcept/SystemOfLaw |
| MISC | TopicalConcept/Tax |
| MISC | TopicalConcept/Taxon |
| MISC | TopicalConcept/TheologicalConcept |
| MISC | TopicalConcept/TheologicalConcept/ChristianDoctrine |

| Entity type | DBpedia type |
|---|---|
| MISC | TopicalConcept/Type |
| MISC | TopicalConcept/Type/DocumentType |
| MISC | TopicalConcept/Type/GovernmentType |
| MISC | UnitOfWork |
| MISC | Unknown |
| MISC | Unknown/WikimediaTemplate |
| MISC | Work |
| MISC | Work/Document |
| MISC | Work/Document/File |
| MISC | Work/Document/Image |
| MISC | Work/Document/Image/MovingImage |
| MISC | Work/Document/Image/StillImage |
| MISC | Work/Document/Sound |
| MISC | Work/MusicalWork/NationalAnthem |
| MISC | Work/MusicalWork/Song/EurovisionSongContestEntry |
| MISC | Work/WrittenWork |
| MISC | Work/WrittenWork/Annotation |
| MISC | Work/WrittenWork/Annotation/Reference |
| MISC | Work/WrittenWork/Law |
| MISC | Work/WrittenWork/Letter |
| MISC | Work/WrittenWork/Quote |
| MISC | Work/WrittenWork/Resume |
| MISC | Work/WrittenWork/StatedResolution |
| MISC | Work/WrittenWork/Treaty |
| MISC | Work/Document |
| MISC | Image |
| MISC | SpatialThing |
| MISC | _Feature |
| MISC | Property |
| MISC | Concept |
| MISC | OrderedCollection |
| MONEY | Currency |
| ORGANIZATION | Agent/Family |
| ORGANIZATION | Agent/Family/NobleFamily |
| ORGANIZATION | Agent/Organisation |
| ORGANIZATION | Agent/Organisation/Broadcaster |
| ORGANIZATION | Agent/Organisation/Broadcaster/BroadcastNetwork |
| ORGANIZATION | Agent/Organisation/Broadcaster/RadioStation |
| ORGANIZATION | Agent/Organisation/Broadcaster/TelevisionStation |
| ORGANIZATION | Agent/Organisation/Company |
| ORGANIZATION | Agent/Organisation/Company/Bank |
| ORGANIZATION | Agent/Organisation/Company/Brewery |
| ORGANIZATION | Agent/Organisation/Company/Caterer |
| ORGANIZATION | Agent/Organisation/Company/LawFirm |
| ORGANIZATION | Agent/Organisation/Company/PublicTransitSystem |
| ORGANIZATION | Agent/Organisation/Company/PublicTransitSystem/Airline |
| ORGANIZATION | Agent/Organisation/Company/PublicTransitSystem/BusCompany |

| Entity type | DBpedia type |
|---|---|
| ORGANIZATION | Agent/Organisation/Company/Publisher |
| ORGANIZATION | Agent/Organisation/Company/RecordLabel |
| ORGANIZATION | Agent/Organisation/Company/Winery |
| ORGANIZATION | Agent/Organisation/EducationalInstitution |
| ORGANIZATION | Agent/Organisation/EducationalInstitution/College |
| ORGANIZATION | Agent/Organisation/EducationalInstitution/Library |
| ORGANIZATION | Agent/Organisation/EducationalInstitution/School |
| ORGANIZATION | Agent/Organisation/EducationalInstitution/University |
| ORGANIZATION | Agent/Organisation/EmployersOrganisation |
| ORGANIZATION | Agent/Organisation/GeopoliticalOrganisation |
| ORGANIZATION | Agent/Organisation/GovernmentAgency |
| ORGANIZATION | Agent/Organisation/GovernmentAgency/GovernmentCabinet |
| ORGANIZATION | Agent/Organisation/Group |
| ORGANIZATION | Agent/Organisation/Group/Band |
| ORGANIZATION | Agent/Organisation/Group/ComedyGroup |
| ORGANIZATION | Agent/Organisation/InternationalOrganisation |
| ORGANIZATION | Agent/Organisation/Legislature |
| ORGANIZATION | Agent/Organisation/MilitaryUnit |
| ORGANIZATION | Agent/Organisation/Non-ProfitOrganisation |
| ORGANIZATION | Agent/Organisation/Non-ProfitOrganisation/RecordOffice |
| ORGANIZATION | Agent/Organisation/Parliament |
| ORGANIZATION | Agent/Organisation/PoliticalParty |
| ORGANIZATION | Agent/Organisation/ReligiousOrganisation |
| ORGANIZATION | Agent/Organisation/ReligiousOrganisation/ClericalOrder |
| ORGANIZATION | Agent/Organisation/SambaSchool |
| ORGANIZATION | Agent/Organisation/SportsClub |
| ORGANIZATION | Agent/Organisation/SportsClub/HockeyClub |
| ORGANIZATION | Agent/Organisation/SportsClub/RugbyClub |
| ORGANIZATION | Agent/Organisation/SportsClub/SoccerClub |
| ORGANIZATION | Agent/Organisation/SportsClub/SoccerClub/NationalSoccerClub |
| ORGANIZATION | Agent/Organisation/SportsLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/AmericanFootballLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/AustralianFootballLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/AutoRacingLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/BaseballLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/BasketballLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/BowlingLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/BoxingLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/CanadianFootballLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/CricketLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/CurlingLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/CyclingLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/FieldHockeyLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/FormulaOneRacing |
| ORGANIZATION | Agent/Organisation/SportsLeague/GolfLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/HandballLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/IceHockeyLeague |

| Entity type | DBpedia type |
|---|---|
| ORGANIZATION | Agent/Organisation/SportsLeague/InlineHockeyLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/LacrosseLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/MixedMartialArtsLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/MotorcycleRacingLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/PaintballLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/PoloLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/RadioControlledRacingLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/RugbyLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/SoccerLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/SoftballLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/SpeedwayLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/TennisLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/VideogamesLeague |
| ORGANIZATION | Agent/Organisation/SportsLeague/VolleyballLeague |
| ORGANIZATION | Agent/Organisation/SportsTeam |
| ORGANIZATION | Agent/Organisation/SportsTeam/AmericanFootballTeam |
| ORGANIZATION | Agent/Organisation/SportsTeam/AustralianFootballTeam |
| ORGANIZATION | Agent/Organisation/SportsTeam/BaseballTeam |
| ORGANIZATION | Agent/Organisation/SportsTeam/BasketballTeam |
| ORGANIZATION | Agent/Organisation/SportsTeam/CanadianFootballTeam |
| ORGANIZATION | Agent/Organisation/SportsTeam/CricketTeam |
| ORGANIZATION | Agent/Organisation/SportsTeam/CyclingTeam |
| ORGANIZATION | Agent/Organisation/SportsTeam/FormulaOneTeam |
| ORGANIZATION | Agent/Organisation/SportsTeam/HandballTeam |
| ORGANIZATION | Agent/Organisation/SportsTeam/HockeyTeam |
| ORGANIZATION | Agent/Organisation/SportsTeam/SpeedwayTeam |
| ORGANIZATION | Agent/Organisation/TradeUnion |
| PERSON | Agent/Deity |
| PERSON | Agent/FictionalCharacter |
| PERSON | Agent/FictionalCharacter/ComicsCharacter |
| PERSON | Agent/FictionalCharacter/ComicsCharacter/AnimangaCharacter |
| PERSON | Agent/FictionalCharacter/DisneyCharacter |
| PERSON | Agent/FictionalCharacter/MythologicalFigure |
| PERSON | Agent/FictionalCharacter/NarutoCharacter |
| PERSON | Agent/FictionalCharacter/SoapCharacter |
| PERSON | Agent/Person |
| PERSON | Agent/Person/Archeologist |
| PERSON | Agent/Person/Architect |
| PERSON | Agent/Person/Aristocrat |
| PERSON | Agent/Person/Artist |
| PERSON | Agent/Person/Artist/Actor |
| PERSON | Agent/Person/Artist/Actor/AdultActor |
| PERSON | Agent/Person/Artist/Actor/VoiceActor |
| PERSON | Agent/Person/Artist/Comedian |
| PERSON | Agent/Person/Artist/ComicsCreator |
| PERSON | Agent/Person/Artist/Dancer |
| PERSON | Agent/Person/Artist/FashionDesigner |

| Entity type | DBpedia type |
|---|---|
| PERSON | Agent/Person/Artist/Humorist |
| PERSON | Agent/Person/Artist/MusicalArtist |
| PERSON | Agent/Person/Artist/MusicalArtist/BackScene |
| PERSON | Agent/Person/Artist/MusicalArtist/ClassicalMusicArtist |
| PERSON | Agent/Person/Artist/MusicalArtist/Instrumentalist |
| PERSON | Agent/Person/Artist/MusicalArtist/Instrumentalist/Guitarist |
| PERSON | Agent/Person/Artist/MusicalArtist/MusicDirector |
| PERSON | Agent/Person/Artist/MusicalArtist/Singer |
| PERSON | Agent/Person/Artist/Painter |
| PERSON | Agent/Person/Artist/Photographer |
| PERSON | Agent/Person/Artist/Sculptor |
| PERSON | Agent/Person/Astronaut |
| PERSON | Agent/Person/Athlete |
| PERSON | Agent/Person/Athlete/ArcherPlayer |
| PERSON | Agent/Person/Athlete/AthleticsPlayer |
| PERSON | Agent/Person/Athlete/AustralianRulesFootballPlayer |
| PERSON | Agent/Person/Athlete/BadmintonPlayer |
| PERSON | Agent/Person/Athlete/BaseballPlayer |
| PERSON | Agent/Person/Athlete/BasketballPlayer |
| PERSON | Agent/Person/Athlete/Bodybuilder |
| PERSON | Agent/Person/Athlete/Boxer |
| PERSON | Agent/Person/Athlete/Boxer/AmateurBoxer |
| PERSON | Agent/Person/Athlete/BullFighter |
| PERSON | Agent/Person/Athlete/Canoeist |
| PERSON | Agent/Person/Athlete/ChessPlayer |
| PERSON | Agent/Person/Athlete/Cricketer |
| PERSON | Agent/Person/Athlete/Cyclist |
| PERSON | Agent/Person/Athlete/DartsPlayer |
| PERSON | Agent/Person/Athlete/Fencer |
| PERSON | Agent/Person/Athlete/GaelicGamesPlayer |
| PERSON | Agent/Person/Athlete/GolfPlayer |
| PERSON | Agent/Person/Athlete/GridironFootballPlayer |
| PERSON | Agent/Person/Athlete/GridironFootballPlayer/AmericanFootballPlayer |
| PERSON | Agent/Person/Athlete/GridironFootballPlayer/CanadianFootballPlayer |
| PERSON | Agent/Person/Athlete/Gymnast |
| PERSON | Agent/Person/Athlete/HandballPlayer |
| PERSON | Agent/Person/Athlete/HighDiver |
| PERSON | Agent/Person/Athlete/HorseRider |
| PERSON | Agent/Person/Athlete/Jockey |
| PERSON | Agent/Person/Athlete/LacrossePlayer |
| PERSON | Agent/Person/Athlete/MartialArtist |
| PERSON | Agent/Person/Athlete/MotorsportRacer |
| PERSON | Agent/Person/Athlete/MotorsportRacer/MotorcycleRider |
| PERSON | Agent/Person/Athlete/MotorsportRacer/MotorcycleRider/MotocycleRacer |
| PERSON | Agent/Person/Athlete/MotorsportRacer/MotorcycleRider/SpeedwayRider |
| PERSON | Agent/Person/Athlete/MotorsportRacer/RacingDriver |
| PERSON | Agent/Person/Athlete/MotorsportRacer/RacingDriver/DTMRacer |

| Entity type | DBpedia type |
|---|---|
| PERSON | Agent/Person/Athlete/MotorsportRacer/RacingDriver/FormulaOneRacer |
| PERSON | Agent/Person/Athlete/MotorsportRacer/RacingDriver/NascarDriver |
| PERSON | Agent/Person/Athlete/MotorsportRacer/RacingDriver/RallyDriver |
| PERSON | Agent/Person/Athlete/NationalCollegiateAthleticAssociationAthlete |
| PERSON | Agent/Person/Athlete/NetballPlayer |
| PERSON | Agent/Person/Athlete/PokerPlayer |
| PERSON | Agent/Person/Athlete/Rower |
| PERSON | Agent/Person/Athlete/RugbyPlayer |
| PERSON | Agent/Person/Athlete/SnookerPlayer |
| PERSON | Agent/Person/Athlete/SnookerPlayer/SnookerChamp |
| PERSON | Agent/Person/Athlete/SoccerPlayer |
| PERSON | Agent/Person/Athlete/SquashPlayer |
| PERSON | Agent/Person/Athlete/Surfer |
| PERSON | Agent/Person/Athlete/Swimmer |
| PERSON | Agent/Person/Athlete/TableTennisPlayer |
| PERSON | Agent/Person/Athlete/TeamMember |
| PERSON | Agent/Person/Athlete/TennisPlayer |
| PERSON | Agent/Person/Athlete/VolleyballPlayer |
| PERSON | Agent/Person/Athlete/VolleyballPlayer/BeachVolleyballPlayer |
| PERSON | Agent/Person/Athlete/WaterPoloPlayer |
| PERSON | Agent/Person/Athlete/WinterSportPlayer |
| PERSON | Agent/Person/Athlete/WinterSportPlayer/Biathlete |
| PERSON | Agent/Person/Athlete/WinterSportPlayer/BobsleighAthlete |
| PERSON | Agent/Person/Athlete/WinterSportPlayer/CrossCountrySkier |
| PERSON | Agent/Person/Athlete/WinterSportPlayer/Curler |
| PERSON | Agent/Person/Athlete/WinterSportPlayer/FigureSkater |
| PERSON | Agent/Person/Athlete/WinterSportPlayer/IceHockeyPlayer |
| PERSON | Agent/Person/Athlete/WinterSportPlayer/NordicCombined |
| PERSON | Agent/Person/Athlete/WinterSportPlayer/Skater |
| PERSON | Agent/Person/Athlete/WinterSportPlayer/Ski_jumper |
| PERSON | Agent/Person/Athlete/WinterSportPlayer/Skier |
| PERSON | Agent/Person/Athlete/WinterSportPlayer/SpeedSkater |
| PERSON | Agent/Person/Athlete/Wrestler |
| PERSON | Agent/Person/Athlete/Wrestler/SumoWrestler |
| PERSON | Agent/Person/BeautyQueen |
| PERSON | Agent/Person/BusinessPerson |
| PERSON | Agent/Person/Chef |
| PERSON | Agent/Person/Cleric |
| PERSON | Agent/Person/Cleric/Cardinal |
| PERSON | Agent/Person/Cleric/ChristianBishop |
| PERSON | Agent/Person/Cleric/ChristianBishop/Archbishop |
| PERSON | Agent/Person/Cleric/ChristianPatriarch |
| PERSON | Agent/Person/Cleric/Pope |
| PERSON | Agent/Person/Cleric/Priest |
| PERSON | Agent/Person/Cleric/Saint |
| PERSON | Agent/Person/Cleric/Vicar |
| PERSON | Agent/Person/Coach |

| Entity type | DBpedia type |
|---|---|
| PERSON | Agent/Person/Coach/AmericanFootballCoach |
| PERSON | Agent/Person/Coach/CollegeCoach |
| PERSON | Agent/Person/Coach/VolleyballCoach |
| PERSON | Agent/Person/Criminal |
| PERSON | Agent/Person/Criminal/Murderer |
| PERSON | Agent/Person/Criminal/Murderer/SerialKiller |
| PERSON | Agent/Person/Economist |
| PERSON | Agent/Person/Egyptologist |
| PERSON | Agent/Person/Engineer |
| PERSON | Agent/Person/Farmer |
| PERSON | Agent/Person/HorseTrainer |
| PERSON | Agent/Person/Journalist |
| PERSON | Agent/Person/Judge |
| PERSON | Agent/Person/Lawyer |
| PERSON | Agent/Person/Linguist |
| PERSON | Agent/Person/MemberResistanceMovement |
| PERSON | Agent/Person/MilitaryPerson |
| PERSON | Agent/Person/Model |
| PERSON | Agent/Person/Monarch |
| PERSON | Agent/Person/MovieDirector |
| PERSON | Agent/Person/Noble |
| PERSON | Agent/Person/OfficeHolder |
| PERSON | Agent/Person/OrganisationMember |
| PERSON | Agent/Person/OrganisationMember/SportsTeamMember |
| PERSON | Agent/Person/Philosopher |
| PERSON | Agent/Person/PlayboyPlaymate |
| PERSON | Agent/Person/Politician |
| PERSON | Agent/Person/Politician/Ambassador |
| PERSON | Agent/Person/Politician/Chancellor |
| PERSON | Agent/Person/Politician/Congressman |
| PERSON | Agent/Person/Politician/Deputy |
| PERSON | Agent/Person/Politician/Governor |
| PERSON | Agent/Person/Politician/Lieutenant |
| PERSON | Agent/Person/Politician/Mayor |
| PERSON | Agent/Person/Politician/MemberOfParliament |
| PERSON | Agent/Person/Politician/Minister |
| PERSON | Agent/Person/Politician/President |
| PERSON | Agent/Person/Politician/PrimeMinister |
| PERSON | Agent/Person/Politician/Senator |
| PERSON | Agent/Person/Politician/VicePresident |
| PERSON | Agent/Person/Politician/VicePrimeMinister |
| PERSON | Agent/Person/PoliticianSpouse |
| PERSON | Agent/Person/Presenter |
| PERSON | Agent/Person/Presenter/RadioHost |
| PERSON | Agent/Person/Presenter/TelevisionHost |
| PERSON | Agent/Person/Producer |
| PERSON | Agent/Person/Psychologist |

| Entity type | DBpedia type |
|---|---|
| PERSON | Agent/Person/Referee |
| PERSON | Agent/Person/Religious |
| PERSON | Agent/Person/RomanEmperor |
| PERSON | Agent/Person/Royalty |
| PERSON | Agent/Person/Royalty/BritishRoyalty |
| PERSON | Agent/Person/Royalty/BritishRoyalty/Baronet |
| PERSON | Agent/Person/Scientist |
| PERSON | Agent/Person/Scientist/Biologist |
| PERSON | Agent/Person/Scientist/Entomologist |
| PERSON | Agent/Person/Scientist/Medician |
| PERSON | Agent/Person/Scientist/Professor |
| PERSON | Agent/Person/SportsManager |
| PERSON | Agent/Person/SportsManager/SoccerManager |
| PERSON | Agent/Person/TelevisionDirector |
| PERSON | Agent/Person/TheatreDirector |
| PERSON | Agent/Person/Writer |
| PERSON | Agent/Person/Writer/Historian |
| PERSON | Agent/Person/Writer/MusicComposer |
| PERSON | Agent/Person/Writer/PlayWright |
| PERSON | Agent/Person/Writer/Poet |
| PERSON | Agent/Person/Writer/ScreenWriter |
| PERSON | Agent/Person/Writer/SongWriter |
| PERSON | PersonFunction |
| PERSON | PersonFunction/PoliticalFunction |
| PERSON | PersonFunction/Profession |
| PERSON | Person |
| PRODUCT | Activity/Game/BoardGame |
| PRODUCT | Activity/Game/CardGame |
| PRODUCT | Device |
| PRODUCT | Device/Battery |
| PRODUCT | Device/Camera |
| PRODUCT | Device/Camera/DigitalCamera |
| PRODUCT | Device/Engine |
| PRODUCT | Device/Engine/AutomobileEngine |
| PRODUCT | Device/Engine/RocketEngine |
| PRODUCT | Device/InformationAppliance |
| PRODUCT | Device/Instrument |
| PRODUCT | Device/Instrument/Guitar |
| PRODUCT | Device/Instrument/Organ |
| PRODUCT | Device/MobilePhone |
| PRODUCT | Device/Weapon |
| PRODUCT | Work/Artwork |
| PRODUCT | Work/Artwork/Painting |
| PRODUCT | Work/Artwork/Sculpture |
| PRODUCT | Work/Cartoon |
| PRODUCT | Work/Cartoon/Anime |
| PRODUCT | Work/Cartoon/HollywoodCartoon |

| Entity type | DBpedia type |
|---|---|
| PRODUCT | Work/CollectionOfValuables |
| PRODUCT | Work/CollectionOfValuables/Archive |
| PRODUCT | Work/Database |
| PRODUCT | Work/Database/BiologicalDatabase |
| PRODUCT | Work/Film |
| PRODUCT | Work/LineOfFashion |
| PRODUCT | Work/MusicalWork |
| PRODUCT | Work/MusicalWork/Album |
| PRODUCT | Work/MusicalWork/ArtistDiscography |
| PRODUCT | Work/MusicalWork/ClassicalMusicComposition |
| PRODUCT | Work/MusicalWork/Musical |
| PRODUCT | Work/MusicalWork/Opera |
| PRODUCT | Work/MusicalWork/Single |
| PRODUCT | Work/MusicalWork/Song |
| PRODUCT | Work/RadioProgram |
| PRODUCT | Work/Software |
| PRODUCT | Work/Software/VideoGame |
| PRODUCT | Work/TelevisionEpisode |
| PRODUCT | Work/TelevisionSeason |
| PRODUCT | Work/TelevisionShow |
| PRODUCT | Work/Website |
| PRODUCT | Work/WrittenWork/Article |
| PRODUCT | Work/WrittenWork/Book |
| PRODUCT | Work/WrittenWork/Book/Novel |
| PRODUCT | Work/WrittenWork/Book/Novel/LightNovel |
| PRODUCT | Work/WrittenWork/Comic |
| PRODUCT | Work/WrittenWork/Comic/ComicStrip |
| PRODUCT | Work/WrittenWork/Comic/Manga |
| PRODUCT | Work/WrittenWork/Comic/Manhua |
| PRODUCT | Work/WrittenWork/Comic/Manhwa |
| PRODUCT | Work/WrittenWork/Drama |
| PRODUCT | Work/WrittenWork/MultiVolumePublication |
| PRODUCT | Work/WrittenWork/PeriodicalLiterature |
| PRODUCT | Work/WrittenWork/PeriodicalLiterature/AcademicJournal |
| PRODUCT | Work/WrittenWork/PeriodicalLiterature/Magazine |
| PRODUCT | Work/WrittenWork/PeriodicalLiterature/Newspaper |
| PRODUCT | Work/WrittenWork/PeriodicalLiterature/UndergroundJournal |
| PRODUCT | Work/WrittenWork/Play |
| PRODUCT | Work/WrittenWork/Poem |
| SPECIES | Species |
| SPECIES | Species/Archaea |
| SPECIES | Species/Bacteria |
| SPECIES | Species/Eukaryote |
| SPECIES | Species/Eukaryote/Animal |
| SPECIES | Species/Eukaryote/Animal/Amphibian |
| SPECIES | Species/Eukaryote/Animal/Arachnid |
| SPECIES | Species/Eukaryote/Animal/Bird |

| Entity type | DBpedia type |
|---|---|
| SPECIES | Species/Eukaryote/Animal/Crustacean |
| SPECIES | Species/Eukaryote/Animal/Fish |
| SPECIES | Species/Eukaryote/Animal/Insect |
| SPECIES | Species/Eukaryote/Animal/Mammal |
| SPECIES | Species/Eukaryote/Animal/Mammal/Cat |
| SPECIES | Species/Eukaryote/Animal/Mammal/Dog |
| SPECIES | Species/Eukaryote/Animal/Mammal/Horse |
| SPECIES | Species/Eukaryote/Animal/Mollusca |
| SPECIES | Species/Eukaryote/Animal/Reptile |
| SPECIES | Species/Eukaryote/Fungus |
| SPECIES | Species/Eukaryote/Plant |
| SPECIES | Species/Eukaryote/Plant/ClubMoss |
| SPECIES | Species/Eukaryote/Plant/Conifer |
| SPECIES | Species/Eukaryote/Plant/CultivatedVariety |
| SPECIES | Species/Eukaryote/Plant/Cycad |
| SPECIES | Species/Eukaryote/Plant/Fern |
| SPECIES | Species/Eukaryote/Plant/FloweringPlant |
| SPECIES | Species/Eukaryote/Plant/FloweringPlant/Grape |
| SPECIES | Species/Eukaryote/Plant/Ginkgo |
| SPECIES | Species/Eukaryote/Plant/Gnetophytes |
| SPECIES | Species/Eukaryote/Plant/GreenAlga |
| SPECIES | Species/Eukaryote/Plant/Moss |
| SUBSTANCE | Biomolecule |
| SUBSTANCE | Biomolecule/Enzyme |
| SUBSTANCE | Biomolecule/Gene |
| SUBSTANCE | Biomolecule/Gene/HumanGene |
| SUBSTANCE | Biomolecule/Gene/MouseGene |
| SUBSTANCE | Biomolecule/Hormone |
| SUBSTANCE | Biomolecule/Lipid |
| SUBSTANCE | Biomolecule/Polysaccharide |
| SUBSTANCE | Biomolecule/Protein |
| SUBSTANCE | ChemicalSubstance |
| SUBSTANCE | ChemicalSubstance/ChemicalCompound |
| SUBSTANCE | ChemicalSubstance/ChemicalElement |
| SUBSTANCE | ChemicalSubstance/Drug |
| SUBSTANCE | ChemicalSubstance/Drug/CombinationDrug |
| SUBSTANCE | ChemicalSubstance/Drug/MonoclonalAntibody |
| SUBSTANCE | ChemicalSubstance/Drug/Vaccine |
| SUBSTANCE | ChemicalSubstance/Mineral |
| TITLE | Diploma |
| TRANSPORT | MeanOfTransportation |
| TRANSPORT | MeanOfTransportation/Aircraft |
| TRANSPORT | MeanOfTransportation/Aircraft/MilitaryAircraft |
| TRANSPORT | MeanOfTransportation/Automobile |
| TRANSPORT | MeanOfTransportation/Locomotive |
| TRANSPORT | MeanOfTransportation/MilitaryVehicle |
| TRANSPORT | MeanOfTransportation/Motorcycle |

| Entity type | DBpedia type |
|---|---|
| TRANSPORT | MeanOfTransportation/On-SiteTransportation |
| TRANSPORT | MeanOfTransportation/On-SiteTransportation/ConveyorSystem |
| TRANSPORT | MeanOfTransportation/On-SiteTransportation/Escalator |
| TRANSPORT | MeanOfTransportation/On-SiteTransportation/MovingWalkway |
| TRANSPORT | MeanOfTransportation/Rocket |
| TRANSPORT | MeanOfTransportation/Ship |
| TRANSPORT | MeanOfTransportation/SpaceShuttle |
| TRANSPORT | MeanOfTransportation/SpaceStation |
| TRANSPORT | MeanOfTransportation/Spacecraft |
| TRANSPORT | MeanOfTransportation/Train |
| TRANSPORT | MeanOfTransportation/TrainCarriage |
| TRANSPORT | MeanOfTransportation/Tram |