



# Text Preprocessing in Python: Steps, Tools, and Examples



Data Monsters

[Follow](#)

Oct 15, 2018 · 7 min read

by Olga Davydova, Data Monsters

In this paper, we will talk about the basic steps of text preprocessing. These steps are needed for transferring text from human language to machine-readable format for further processing. We will also discuss text preprocessing tools.

After a text is obtained, we start with text normalization. Text normalization includes:

- converting all letters to lower or upper case
- converting numbers into words or removing numbers
- removing punctuations, accent marks and other diacritics

- removing white spaces
- expanding abbreviations
- removing stop words, sparse terms, and particular words
- text canonicalization

We will describe text normalization steps in detail below.

## Convert text to lowercase

### Example 1. Convert text to lowercase

Python code:

```
input_str = "The 5 biggest countries by population in 2017  
are China, India, United States, Indonesia, and Brazil."  
input_str = input_str.lower()  
print(input_str)
```

Output:

```
the 5 biggest countries by population in 2017 are china,  
india, united states, indonesia, and brazil.
```

## Remove numbers

Remove numbers if they are not relevant to your analyses. Usually, regular expressions are used to remove numbers.

### Example 2. Numbers removing

Python code:

```
import re
input_str = 'Box A contains 3 red and 5 white balls, while
Box B contains 4 red and 2 blue balls.'
result = re.sub(r'\d+', '', input_str)
print(result)
```

### Output:

```
Box A contains red and white balls, while Box B contains
red and blue balls.
```

## Remove punctuation

The following code removes this set of symbols [!"#\$%&'()\*+,./:;  
<=>?@[\\]^\_`{|}~]:

### Example 3. Punctuation removal

#### Python code:

```
import string
input_str = "This &is [an] example? {of} string. with.??
punctuation!!!!" # Sample string
result = input_str.translate(string.maketrans("", ""), 
string.punctuation)
print(result)
```

### Output:

```
This is an example of string with punctuation
```

## Remove whitespaces

To remove leading and ending spaces, you can use the `strip()` function:

### Example 4. White spaces removal

#### Python code:

```
input_str = " \t a string example\t "
input_str = input_str.strip()
input_str
```

#### Output:

```
'a string example'
```

## Tokenization

Tokenization is the process of splitting the given text into smaller pieces called tokens. Words, numbers, punctuation marks, and others can be considered as tokens. In [this table](#) (“Tokenization” sheet) several tools for implementing tokenization are described.

Name	Developer, Initial release	Features	Programming languages	License	Project link
Natural Language Toolkit (NLTK)	The University of Pennsylvania, 2001	Mac/Unix/Windows support Contains many corpora, toy grammars, trained models, etc [1].	Python	Apache License Version 2.0.	<a href="http://www.nltk.org/index.html">http://www.nltk.org/index.html</a>
TextBlob	Steven Loria, 2013	Splitting text into words and sentences WordNet integration [2]	Python	<a href="http://textblob.readthedocs.io/en/dev/license.html">http://textblob.readthedocs.io/en/dev/license.html</a>	<a href="http://textblob.readthedocs.io/en/dev/">http://textblob.readthedocs.io/en/dev/</a>
Spacy	Explosion AI, 2016	Runs on Unix/Linux, MacOS/OS X, and Windows. Neural network models multi-language support [3]	Python	MIT License	<a href="https://spacy.io/">https://spacy.io/</a>
Gensim	RaRe Technologies, 2009	Can process large, web-scale corpora Runs on Linux, Windows and OS X Vector space modeling and topic modeling [4]	Python	GNU GPLv2.1 license	<a href="https://radimrehurek.com/gensim/">https://radimrehurek.com/gensim/</a>
Apache OpenNLP	Apache Software Foundation, 2004	Contains a large number of pre-built models for a variety of languages Includes annotated text resources [5]	Java	Apache License, Version 2.0	<a href="https://opennlp.apache.org/">https://opennlp.apache.org/</a>
OpenNMT	Yoon Kim, harvardnlp, 2016	Is a generic deep learning framework mainly specialized in sequence-to-sequence models Can be used either via command line applications, client-server, or libraries. [6]	Python	MIT License	<a href="http://opennmt.net/">http://opennmt.net/</a>
		Has currently 3 main implementations ( <a href="#">OpenNMT-ta</a> , <a href="#">OpenNMT-py</a> , <a href="#">OpenNMT-tf</a> )	Lua		
		Includes an information extraction system Multiple languages support Accepts input in various formats [7]	Java		
Apache UIMA	IBM, Apache Software Foundation, 2006	Contains Addons and Sandbox Cross-platform REST requests support [8]	Java, C++	Apache License 2.0	<a href="https://uima.apache.org/">https://uima.apache.org/</a>
Memory-Based Shallow Parser (MBSP)	Vincent Van Asch, Tom De Smedt, 2010	Client-server architecture includes binaries (TiMBL, MBT and MBLEM) Precompiled for Mac OS X <a href="#">Cygwin usage for Windows</a> [9]	Python	GPL	<a href="https://www.clips.ugent.be/pages/M_BSP#tokenizer">https://www.clips.ugent.be/pages/M_BSP#tokenizer</a>
RapidMiner	RapidMiner, 2006	Unified platform Visual workflow design Breadth of functionality Broad connectivity [10]	RapidMiner provides a GUI to design and execute analytical workflows	AGPL	<a href="https://rapidminer.com/">https://rapidminer.com/</a>

### Tokenization tools

## Remove stop words

“Stop words” are the most common words in a language like “the”, “a”, “on”, “is”, “all”. These words do not carry important meaning and are usually removed from texts. It is possible to remove stop words using [Natural Language Toolkit \(NLTK\)](#), a suite of libraries and programs for symbolic and statistical natural language processing.

### Example 7. Stop words removal

**Code:**

```
input_str = "NLTK is a leading platform for building Python
programs to work with human language data."
stop_words = set(stopwords.words('english'))
from nltk.tokenize import word_tokenize
tokens = word_tokenize(input_str)
result = [i for i in tokens if not i in stop_words]
print (result)
```

## Output:

```
['NLTK', 'leading', 'platform', 'building', 'Python',
'programs', 'work', 'human', 'language', 'data', '..']
```

A [scikit-learn](#) tool also provides a stop words list:

```
from sklearn.feature_extraction.stop_words import
ENGLISH_STOP_WORDS
```

It's also possible to use [spaCy](#), a free open-source library:

```
from spacy.lang.en.stop_words import STOP_WORDS
```

## Remove sparse terms and particular words

In some cases, it's necessary to remove sparse terms or particular words from texts. This task can be done using stop words removal techniques considering that any group of words can be chosen as the stop words.

## Stemming

Stemming is a process of reducing words to their word stem, base or root form (for example, books—book, looked—look). The main two algorithms are [Porter stemming algorithm](#) (removes common morphological and inflexional endings from words [14]) and [Lancaster stemming algorithm](#) (a more aggressive stemming algorithm). In the “[Stemming](#)” sheet of the table some stemmers are described.

Name	Developer, Initial release	Features	Programming languages	License	Project link
Natural Language Toolkit (NLTK)	The University of Pennsylvania, 2001	Includes Porter stemmer, Snowball stemmer, and Lancaster stemmer Contains a stemmer that uses regular expressions to identify morphological affixes <a href="#">multiple languages support [15]</a>	Python	<a href="#">Apache License Version 2.0.</a>	<a href="http://www.nltk.org/api/nltk.stem.html">http://www.nltk.org/api/nltk.stem.html</a>
Snowball	Martin Porter, 2002	Multiple languages support <a href="#">Small string processing language designed for creating stemming algorithms [16]</a>	Java Python Wrappers are provided	<a href="#">the 3-clause BSD License</a>	<a href="https://snowballstem.org/">https://snowballstem.org/</a>
PyStemmer	Richard Boulton, 2006	Efficient algorithms for calculating a "stemmed" form of a word Provides algorithms for several (mainly European) languages <a href="#">Porter stemming algorithm for English [17]</a>	MPython	MIT license	<a href="https://github.com/snowballstem/pystemmer">https://github.com/snowballstem/pystemmer</a>
Hunspell stemmer	łopusz	Polish language support <a href="#">Dictionary-based stemming [18]</a>	Java	<a href="#">Apache License, Version 2.0</a>	<a href="https://www.elastic.co/guide/en/elasticsearch/guide/current/hunspell.html">https://www.elastic.co/guide/en/elasticsearch/guide/current/hunspell.html</a>
CoreNLP Stemmer	The Stanford Natural Language Processing Group, 2010	<a href="#">A special class Stemmer was created</a>	Java	<a href="#">GNU General Public License</a>	<a href="https://github.com/stanfordnlp/CoreNLP/blob/master/src/edu/stanford/nlp/process/Stemmer.java">https://github.com/stanfordnlp/CoreNLP/blob/master/src/edu/stanford/nlp/process/Stemmer.java</a>
Apache Lucene	Apache Software Foundation, 1999	Scalable, high-performance indexing Powerful, accurate and efficient search algorithms <a href="#">cross-platform solution [19]</a>	Java ported to Object Pascal, Perl, C#, C++, Python, Ruby and PHP	<a href="#">Apache License, Version 2.0</a>	<a href="https://lucene.apache.org/">https://lucene.apache.org/</a>
DKPro Core	The Ubiquitous Knowledge Processing Lab (UKP) at the Technische Universität Darmstadt, 2009	Various models covering different languages accompany the components Supports various data formats <a href="#">Easy integration into Python projects [20]</a>	Java Groovy Python	<a href="#">Apache Software License (ASL) version 2</a>	<a href="https://dkpro.github.io/dkpro-core/">https://dkpro.github.io/dkpro-core/</a>

### Stemming tools

#### Example 8. Stemming using NLTK:

##### Code:

```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
stemmer= PorterStemmer()
input_str="There are several types of stemming algorithms."
input_str=word_tokenize(input_str)
for word in input_str:
    print(stemmer.stem(word))
```

##### Output:

There are sever type of stem algorithm.

## Lemmatization

The aim of lemmatization, like stemming, is to reduce inflectional forms to a common base form. As opposed to stemming, lemmatization does

not simply chop off inflections. Instead it uses lexical knowledge bases to get the correct base forms of words.

Lemmatization tools are presented libraries described above: [NLTK](#), [\(WordNet Lemmatizer\)](#), [spaCy](#), [TextBlob](#), [Pattern](#), [gensim](#), [Stanford CoreNLP](#), [Memory-Based Shallow Parser \(MBSP\)](#), [Apache OpenNLP](#), [Apache Lucene](#), [General Architecture for Text Engineering \(GATE\)](#), [Illinois Lemmatizer](#), and [DKPro Core](#).

### Example 9. Lemmatization using NLTK:

Code:

```
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
lemmatizer=WordNetLemmatizer()
input_str="been had done languages cities mice"
input_str=word_tokenize(input_str)
for word in input_str:
    print(lemmatizer.lemmatize(word))
```

Output:

```
be have do language city mouse
```

## Part of speech tagging (POS)

Part-of-speech tagging aims to assign parts of speech to each word of a given text (such as nouns, verbs, adjectives, and others) based on its definition and its context. There are many tools containing POS taggers including [NLTK](#), [spaCy](#), [TextBlob](#), [Pattern](#), [Stanford CoreNLP](#), [Memory-Based Shallow Parser \(MBSP\)](#), [Apache OpenNLP](#), [Apache Lucene](#), [General Architecture for Text Engineering \(GATE\)](#), [FreeLing](#), [Illinois Part of Speech Tagger](#), and [DKPro Core](#).

### Example 10. Part-of-speech tagging using TextBlob:

Code:

```
input_str="Parts of speech examples: an article, to write,  
interesting, easily, and, of"  
from textblob import TextBlob  
result = TextBlob(input_str)  
print(result.tags)
```

**Output:**

```
[('Parts', u'NNS'), ('of', u'IN'), ('speech', u'NN'),  
('examples', u'NNS'), ('an', u'DT'), ('article', u'NN'),  
('to', u'TO'), ('write', u'VB'), ('interesting', u'VBG'),  
('easily', u'RB'), ('and', u'CC'), ('of', u'IN')]
```

## Chunking (shallow parsing)

Chunking is a natural language process that identifies constituent parts of sentences (nouns, verbs, adjectives, etc.) and links them to higher order units that have discrete grammatical meanings (noun groups or phrases, verb groups, etc.) [23]. Chunking tools: [NLTK](#), [TreeTagger chunker](#), [Apache OpenNLP](#), [General Architecture for Text Engineering \(GATE\)](#), [FreeLing](#).

**Example 11. Chunking using NLTK:**

The first step is to determine the part of speech for each word:

**Code:**

```
input_str="A black television and a white stove were bought  
for the new apartment of John."  
from textblob import TextBlob  
result = TextBlob(input_str)  
print(result.tags)
```

**Output:**

```
[('A', u'DT'), ('black', u'JJ'), ('television', u'NN'),
 ('and', u'CC'), ('a', u'DT'), ('white', u'JJ'), ('stove',
 u'NN'), ('were', u'VBD'), ('bought', u'VBN'), ('for',
 u'IN'), ('the', u'DT'), ('new', u'JJ'), ('apartment',
 u'NN'), ('of', u'IN'), ('John', u>NNP)]
```

The second step is chunking:

**Code:**

```
reg_exp = "NP: {<DT>?<JJ>*<NN>}"
rp = nltk.RegexpParser(reg_exp)
result = rp.parse(result.tags)
print(result)
```

**Output:**

```
(S (NP A/DT black/JJ television/NN) and/CC (NP a/DT
white/JJ stove/NN) were/VBD bought/VBN for/IN (NP the/DT
new/JJ apartment/NN)
of/IN John>NNP)
```

It's also possible to draw the sentence tree structure using code

```
result.draw()
```



## Named entity recognition

Named-entity recognition (NER) aims to find named entities in text and classify them into pre-defined categories (names of persons, locations,

organizations, times, etc.).

Named-entity recognition tools: [NLTK](#), [spaCy](#), [General Architecture for Text Engineering \(GATE\)](#)—[ANNIE](#), [Apache OpenNLP](#), [Stanford CoreNLP](#), [DKPro Core](#), [MITIE](#), [Watson Natural Language Understanding](#), [TextRazor](#), [FreeLing](#) are described in the “NER” sheet of the table.

Name	Developer, Initial release	Features	Programming languages	License	Project link
Baleen	Defence Science and Technology Laboratory (Dstl), 2014	Works with unstructured and semi-structured data sources Includes a built-in server [25]	Java	Apache License 2.0	<a href="https://github.com/dstl/baleen">https://github.com/dstl/baleen</a>
CogComp NER Tagger (Illinois Named Entity Tagger)	L. Ratinov, D. Roth, Cognitive Computation Group, 2009	Tags plain text with named entities 4-label type set (people / organizations / locations / miscellaneous) <a href="#">18-label type set (based on the OntoNotes corpus)</a> [26]	Java	<a href="#">Licensing Agreement</a>	<a href="https://github.com/CogComp/cogcomp-nlp/tree/master/ner">https://github.com/CogComp/cogcomp-nlp/tree/master/ner</a>
Minimal Named-Entity Recognizer (MER)	LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal, 2017	Returns the list of terms recognized in the text, including their exact location (annotations) <a href="#">Only requires a lexicon (text file) with the list of terms representing the entities of interest RESTful Web service</a> [27]	GNU awk	-	<a href="https://github.com/lasigeBioTM/MER">https://github.com/lasigeBioTM/MER</a>
ParallelDots	ParallelDots	Uses deep learning technology to determine representations of character groupings Discovers the most relevant entities in textual content Accurate, real-time, customizable [28] demo	<a href="#">excel add-in</a> <a href="#">AI APIs</a>	Pricing	<a href="https://www.paralleldots.com/named-entity-recognition">https://www.paralleldots.com/named-entity-recognition</a>
Open Calais	Thomson Reuters Corporation	<a href="#">Extracts entities (companies, people, places, products, etc.), relationships, facts, events, topics.</a> [29]	API	<a href="#">Terms of Service</a>	<a href="http://www.opencalais.com/about-open-calais/">http://www.opencalais.com/about-open-calais/</a>
LingPipe	Breck Baldwin, 1999	Finds the names of people, organizations, or locations Source code and unit tests <a href="#">Multi-lingual, multi-domain, multi-genre models</a> [30]	Java	<a href="#">License Matrix</a>	<a href="http://alias-i.com/lingpipe/index.html">http://alias-i.com/lingpipe/index.html</a>
Named Entity Recognition Tool	Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, Chris Dyer, 2016	A neural architecture <a href="#">state-of-the-art performance in NER on the 4 CoNLL datasets (English, Spanish, German and Dutch) without resorting to any language-specific knowledge or resources such as gazetteers</a> [31]	Python	Apache License 2.0	<a href="https://github.com/glample/tagger">https://github.com/glample/tagger</a>

## NER Tools

### Example 12. Named-entity recognition using NLTK:

#### Code:

```
from nltk import word_tokenize, pos_tag, ne_chunk
input_str = "Bill works for Apple so he went to Boston for a conference."
print ne_chunk(pos_tag(word_tokenize(input_str)))
```

## Output:

```
(S (PERSON Bill/NNP) works/VBZ for/IN Apple/NNP so/IN
he/PRP went/VBD to/T0 (GPE Boston/NNP) for/IN a/DT
conference/NN ./.)
```

## Coreference resolution (anaphora resolution)

Pronouns and other referring expressions should be connected to the right individuals. Coreference resolution finds the mentions in a text that refer to the same real-world entity. For example, in the sentence, “Andrew said he would buy a car” the pronoun “he” refers to the same person, namely to “Andrew”. Coreference resolution tools: [Stanford CoreNLP](#), [spaCy](#), [Open Calais](#), [Apache OpenNLP](#) are described in the [“Coreference resolution” sheet of the table.](#)

Name	Developer, Initial release	Features	Programming languages	License	Project link
<b>Beautiful Anaphora Resolution Toolkit (BART)</b>	Massimo Poesio, Simone Ponazzo, Yannick Versley, Johns Hopkins Summer Workshop, 2007	Incorporates a variety of machine learning approaches Uses several machine learning toolkits <a href="#">Exports the result as inline XML [36]</a>	REST-based web service	<a href="#">Apache license v2.0</a>	<a href="http://www.bart-coref.org/">http://www.bart-coref.org/</a>
<b>JavaRAP</b>	Long Qiu, 2004	Resolves third person pronouns, lexical anaphors, Identifies pleonastic pronouns  Output format: anaphor - antecedent pairs; text with in-place substitutions  Accuracy: 57.9% (MUC6) around 1,500 words per second [37]	Java	-	<a href="https://wing.comp.nus.edu.sg/~qiu/NLPTools/JavaRAP.html">https://wing.comp.nus.edu.sg/~qiu/NLPTools/JavaRAP.html</a>
<b>A General Tool for Anaphora Resolution - GuiTAR</b>	University of Essex, 2007	Takes as an input a MAS-XML compliant file and adds new markup holding anaphoric information (elements). <a href="#">Evaluation module is also included [38]</a>	Java	<a href="#">GPL License</a>	<a href="http://cswww.essex.ac.uk/Research/nle/GuiTAR/">http://cswww.essex.ac.uk/Research/nle/GuiTAR/</a>
<b>Reconcile</b>	Cornell University, The University of Utah, Lawrence Livermore National Labs, 2009	Runs on common data sets or unlabeled texts  Utilizes supervised machine learning classifiers from the Weka toolkit, the Berkeley Parser and Stanford Named Entity Recognition system.  MUC Score (MUC-6): recall 67.23; precision 65.54; F-measure 66.38 <a href="#">[39]</a>	Java	<a href="#">GPL License</a>	<a href="https://www.cs.utah.edu/nlp/reconcile/">https://www.cs.utah.edu/nlp/reconcile/</a>
<b>ARKref</b>	Brendan O'Connor, Michael Heilman, 2009	Deterministic, rule-based system  Uses syntactic information from a constituent parser, and semantic information from an entity recognition component <a href="#">Precision: 0.657617, recall: 0.552433, f1: 0.600454 [40]</a>	Java	<a href="#">GPL, MIT</a>	<a href="http://www.cs.cmu.edu/~ark/ARKref/">http://www.cs.cmu.edu/~ark/ARKref/</a>
<b>Illinois Coreference Package</b>	Dan Roth, Eric Bengtson, 2008	Coreference related features:  Gender and number match, WordNet relations including synonym, hypernym, antonym, and ACE entity types (person, organization, and geopolitical entity) <a href="#">Features anaphoricity classifier [41]</a>	Java	-	<a href="https://cogcomp.org/page/software_view/Coref">https://cogcomp.org/page/software_view/Coref</a>
<b>Neural coref</b>	Hugging Face, 2017	Uses neural nets and spaCy <a href="#">Adds various speakers in the conversation when computing the features and resolving the coreferences [42]</a> Online demo	Python	<a href="#">MIT License</a>	<a href="https://github.com/huggingface/neuralcoref">https://github.com/huggingface/neuralcoref</a>

Coreference resolution tools

An example of coreference resolution using xrenner can be found [here](#).

## Collocation extraction

Collocations are word combinations occurring together more often than would be expected by chance. Collocation examples are “break the rules,” “free time,” “draw a conclusion,” “keep in mind,” “get ready,” and so on.

Name	Developer, Initial release	Features	Programming languages	License	Project link
Termex	Text Analysis and Knowledge Engineering Lab, University of Zagreb, 2009	UTF-8 formatted input text uses 14 association measures Processes of n-grams up to length four Hand selection of candidate n-grams for terminology lexica Windows and Linux support Fast and memory efficient processing of large corpora <a href="#">[47]</a>	Front end GUI	Freely available for research purposes upon request.	<a href="http://ktlab.fer.hr/termex/">http://ktlab.fer.hr/termex/</a>
Collocate	Athelstan	Uses statistical analyses (t-score, log likelihood, Mutual Information) and frequency information to present a list of candidate collocations  Searches for a word (phrase) within a set span (e.g. 4 words).  Produces an n-gram list  <a href="#">Extracts collocations using thresholds and either mutual information [48]</a>	Application	<a href="#">Educational Price. Single user: \$45</a> <a href="#">Site license (2-year, 15 users) \$395</a>	<a href="https://www.athel.com/colloc.html">https://www.athel.com/colloc.html</a>
CollTerm	Faculty of Humanities and Social Sciences at University of Zagreb; Research Institute for Artificial Intelligence at Romanian Academy	Results based on five different co-occurrence measures for multiword units (i.e. collocations) or distributional differences from large representative corpus by application of the TF-IDF measurement on single word units  <a href="#">Language independent [49]</a>	Python	<a href="#">ApacheLicense 2.0</a>	<a href="https://github.com/accurat-toolkit/CollTerm">https://github.com/accurat-toolkit/CollTerm</a>
Collocation Extractor	Dan Ștefănescu, 2012	Collocation words features in this approach: – the distance between them is relatively constant; – they appear together more often than expected by chance (Log-Likelihood)  Language independent  <a href="#">output annotation format: Text output with one collocation per line and annotations separated by tab [50]</a>	Application	Restrictions: Inform Licensor, No User Nature: Academic,	<a href="http://metashare.ilsp.gr:8080/repository/browse/collection-extractor/7a2432acdc7311e5aa0b00237df3e35819ec25a2cee244cd8782b24eddcad3c8/">http://metashare.ilsp.gr:8080/repository/browse/collection-extractor/7a2432acdc7311e5aa0b00237df3e35819ec25a2cee244cd8782b24eddcad3c8/</a>

Collocation extraction tools

### Example 13. Collocation extraction using ICE [51]

Code:

```
input=["he and Chazz duel with all keys on the line."]
from ICE import CollocationExtractor
extractor =
CollocationExtractor.with_collocation_pipeline("T1" ,
bing_key = "Temp", pos_check = False)
```

```
print(extractor.get_collocations_of_length(input, length = 3))
```

## Output:

["on the line"]

# Relationship extraction

Relationship extraction allows obtaining structured information from unstructured sources such as raw text. Strictly stated, it is identifying relations (e.g., acquisition, spouse, employment) among named entities (e.g., people, organizations, locations). For example, from the sentence “Mark and Emily married yesterday,” we can extract the information that Mark is Emily’s husband.

Name	Developer, Initial release	Features	Programming languages	License	Project link
ReVerb	University of Washington's Turing Center	Automatically identifies and extracts binary relationships from English sentences Designed for Web-scale information extraction where the target relations cannot be specified in advance and speed is important. Inputs raw text <a href="#">Outputs (argument1, relation phrase, argument2) triples [53]</a>	Java	<a href="#">ReVerb Software License Agreement</a>	<a href="http://reverb.cs.washington.edu/">http://reverb.cs.washington.edu/</a>
EXEMPLAR	University of Alberta, 2013	Able to identify instances of any relation described in the text Extracts relations with two or more arguments <a href="#">Role of an argument can be SUBJ (subject), DOBJ (direct object) and POBJ (prepositional object) [54]</a>	Java	<a href="#">GNU General Public License v3.0</a>	<a href="https://github.com/U-Alberta/exemplar">https://github.com/U-Alberta/exemplar</a>
Toolkit for Exploring Text for Relation Extraction (TETRE)	Alisson Oldoni, 2017	Accepts raw text as an input Optimized for the task of information extraction in a corpus composed of academic papers Does data transformation, parsing, wraps tasks of third-party binaries <a href="#">Outputs the relations in HTML and JSON [55]</a>	Command line tool	<a href="#">MIT License</a>	<a href="https://github.com/aoldoni/tetre">https://github.com/aoldoni/tetre</a>
TextRazor	TextRazor, 2011	Uses state-of-the-art NLP and AI techniques Processes thousands of words per second per core <a href="#">Allows user to add product names, people, companies, custom classification rules and advanced linguistic patterns [56]</a>	Python PHP  Java  REST API	<a href="#">Pricing</a>	<a href="https://www.textrazor.com/">https://www.textrazor.com/</a>
Information Extraction in Python (IEPY)	Machinalis, 2014	Tries to predict relations using information provided by the user Aimed to perform Information Extraction (IE) on a large dataset created for scientific experiments with new IE algorithms <a href="#">Configured with convenient defaults [57]</a>	Python	<a href="#">BSD 3-Clause "New" or "Revised" License</a>	<a href="https://github.com/machinalis/iepy">https://github.com/machinalis/iepy</a>

An example of relationship extraction using [NLTK](#) can be found [here](#).

## Summary

In this post, we talked about text preprocessing and described its main steps including normalization, tokenization, stemming, lemmatization, chunking, part of speech tagging, named-entity recognition, coreference resolution, collocation extraction, and relationship extraction. We also discussed text preprocessing tools and examples. A [comparative table](#) was created.

After the text preprocessing is done, the result may be used for more complicated NLP tasks, for example, machine translation or natural language generation.

## Resources:

1. <http://www.nltk.org/index.html>
2. <http://textblob.readthedocs.io/en/dev/>
3. <https://spacy.io/usage/facts-figures>
4. <https://radimrehurek.com/gensim/index.html>
5. <https://opennlp.apache.org/>
6. <http://opennmt.net/>
7. <https://gate.ac.uk/>
8. <https://uima.apache.org/>
9. <https://www.clips.uantwerpen.be/pages/MBSP#tokenizer>
10. <https://rapidminer.com/>
11. <http://mallet.cs.umass.edu/>
12. <https://www.clips.uantwerpen.be/pages/pattern>
13. <https://nlp.stanford.edu/software/tokenizer.html#About>

14. <https://tartarus.org/martin/PorterStemmer/>
15. <http://www.nltk.org/api/nltk.stem.html>
16. <https://snowballstem.org/>
17. <https://pypi.python.org/pypi/PyStemmer/1.0.1>
18. <https://www.elastic.co/guide/en/elasticsearch/guide/current/hunspell.html>
19. <https://lucene.apache.org/core/>
20. <https://dkpro.github.io/dkpro-core/>
21. <http://ucrel.lancs.ac.uk/claws/>
22. <http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/>
23. [https://en.wikipedia.org/wiki/Shallow\\_parsing](https://en.wikipedia.org/wiki/Shallow_parsing)
24. [https://cogcomp.org/page/software\\_view/Chunker](https://cogcomp.org/page/software_view/Chunker)
25. <https://github.com/dstl/baleen>
26. <https://github.com/CogComp/cogcomp-nlp/tree/master/ner>
27. <https://github.com/lasigeBioTM/MER>
28. <https://blog.paralleldots.com/product/dig-relevant-text-elements-entity-extraction-api/>
29. <http://www.opencalais.com/about-open-calais/>
30. <http://alias-i.com/lingpipe/index.html>
31. <https://github.com/glample/tagger>
32. <http://minorthird.sourceforge.net/old/doc/>
33. [https://www.ibm.com/support/knowledgecenter/en/SS8NLW\\_10.0.0/com.ibm.watson.wex.aac.doc/aac-tasystemt.html](https://www.ibm.com/support/knowledgecenter/en/SS8NLW_10.0.0/com.ibm.watson.wex.aac.doc/aac-tasystemt.html)
34. <https://www.poolparty.biz/>
35. <https://www.basistech.com/text-analytics/rosette/entity-extractor/>
36. <http://www.bart-coref.org/index.html>

37. <https://wing.comp.nus.edu.sg/~qiu/NLPTools/JavaRAP.html>
38. <http://cswww.essex.ac.uk/Research/nle/GuiTAR/>
39. <https://www.cs.utah.edu/nlp/reconcile/>
40. <https://github.com/brendano/arkref>
41. [https://cogcomp.org/page/software\\_view/Coref](https://cogcomp.org/page/software_view/Coref)
42. <https://medium.com/huggingface/state-of-the-art-neural-coreference-resolution-for-chatbots-3302365dcf30>
43. <https://github.com/smartschat/cort>
44. <http://www.hlt.utdallas.edu/~altaf/cherrypicker/>
45. <http://nlp.lsi.upc.edu/freeling/>
46. <https://corpling.uis.georgetown.edu/xrenner/#>
47. [http://takelab.fer.hr/termex\\_s/](http://takelab.fer.hr/termex_s/)
48. <https://www.athel.com/colloc.html>
49. <http://linghub.lider-project.eu/metashare/a89c02f4663d11e28a985ef2e4e6c59e76428bf02e394229a70428f25a839f75>
50. <http://ws.racai.ro:9191/narratives/batch2/Colloc.pdf>
51. <http://www.aclweb.org/anthology/E17-3027>
52. <https://metacpan.org/pod/Text::NSP>
53. <https://github.com/knowitall/reverb>
54. <https://github.com/U-Alberta/exemplar>
55. <https://github.com/aoldoni/tetre>
56. <https://www.textrazor.com/technology>
57. <https://github.com/machinalis/iepy>
58. <https://www.ibm.com/watson/developercloud/natural-language-understanding/api/v1/#relations>
59. <https://github.com/mit-nlp/MITIE>

