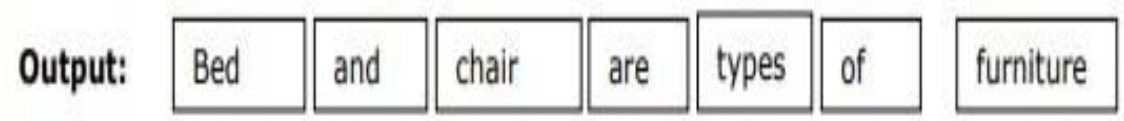
Text Preprocessing basics

1. Tokenization
2. Removing stopwords
3. Stemming
4. Lemmatization

1.Tokenization

- Tokenization may be defined as the process of breaking the given text, into smaller units called tokens.
- Words, numbers or punctuation marks can be tokens.
- It may also be called word segmentation.
  - **Input** – Bed and chair are types of furniture.



Output:	Bed	and	chair	are	types	of	furniture

- **nltk.tokenize** is the package provided by NLTK module to achieve the process of tokenization.

Definitions

- Word – A delimited string of characters as it appears in the text.
- Term – A “normalized” word (case, morphology, spelling etc); an equivalence class of words.
- Token – An instance of a word or term occurring in a document.
- Type – The same as a term in most cases: an equivalence class of tokens.

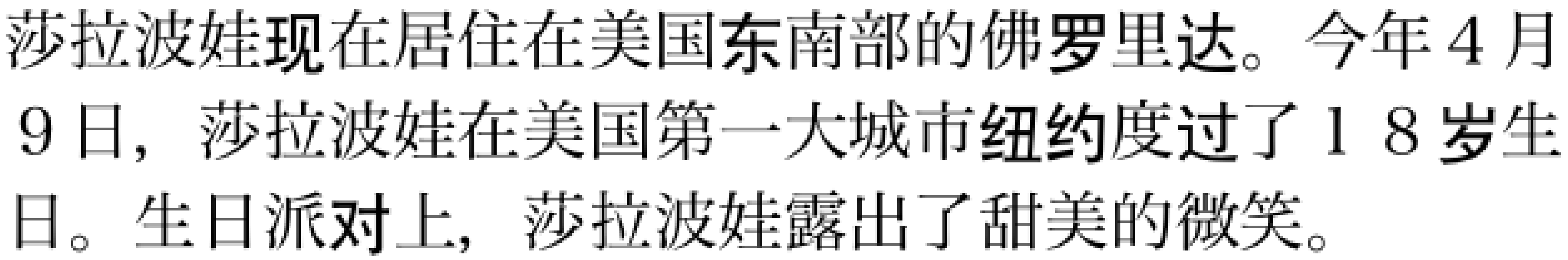
Tokenization problems: One word or two? (or several)

- Hewlett-Packard
- State-of-the-art
- co-education
- the hold-him-back-and-drag-him-away maneuver
- data base
- San Francisco
- Los Angeles-based company
- cheap San Francisco-Los Angeles fares
- York University vs. New York University

Numbers

- 3/20/91
- 20/3/91
- Mar 20, 1991
- B-52
- 100.2.86.144
- (800) 234-2333
- 800.234.2333
- Older IR systems may not index numbers . . .
- . . . but generally it’s a useful feature.

Chinese: No whitespace



莎拉波娃现在居住在美国东南部的佛罗里达。今年４月９日，莎拉波娃在美国第一大城市纽约度过了１８岁生日。生日派对上，莎拉波娃露出了甜美的微笑。

Ambiguous segmentation in Chinese

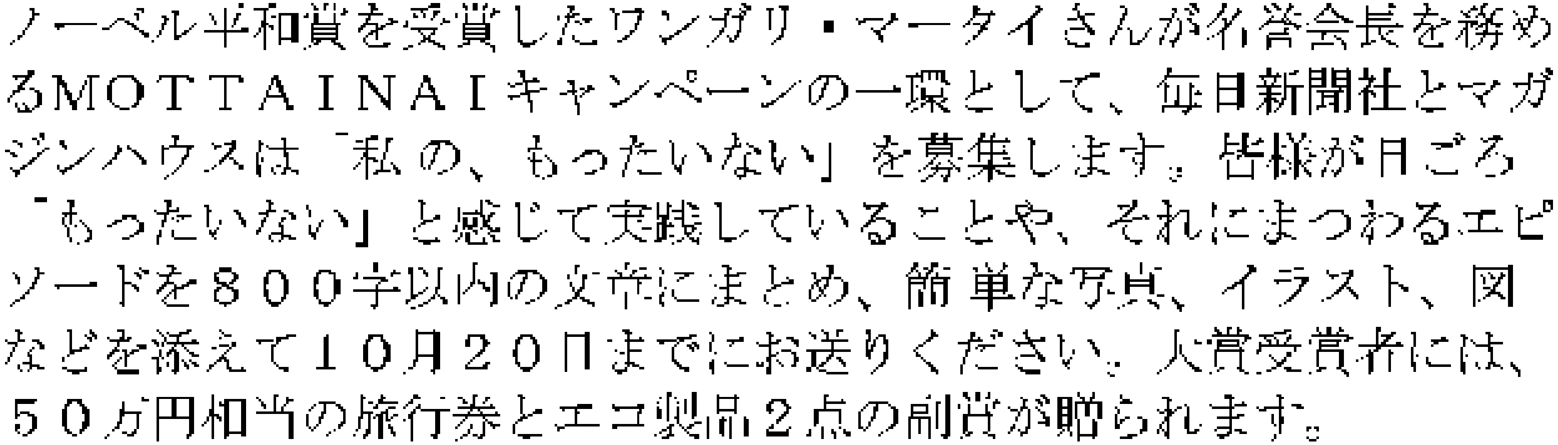
和尚

The two characters can be treated as one word meaning ‘monk’ or as a sequence of two words meaning ‘and’ and ‘still’.

Other cases of “no whitespace”

- Compounds in Dutch, German, Swedish
- Computerlinguistik → Computer + Linguistik
- Lebensversicherungsgesellschaftsangestellter
- → leben + versicherung + gesellschaft + angestellter
- Inuit: tusaatsiarunnanngittualuujunga (I can’t hear very well.)
- Many other languages with segmentation difficulties: Finnish, Urdu, . . .

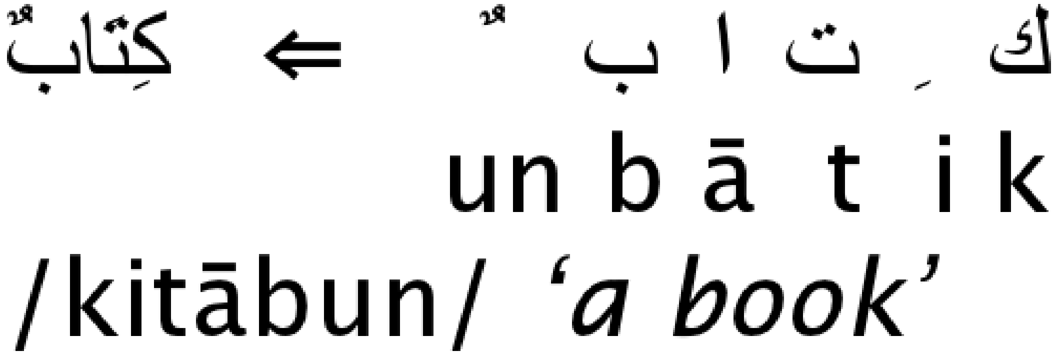
Japanese



ノーベル平和賞を受賞したワンガリ・マータイさんが名誉会長を務めるＭＯＴＴＡＩＮＡＩキャンペーンの一環として、毎日新聞社とマガジンハウスは「私の、もったいない」を募集します。皆様が日ごろ「もったいない」と感じて実践していることや、それにまつわるエピソードを８００字以内の文章にまとめ、簡単な写真、イラスト、図などを添えて１０月２０日までにお送りください。入賞受賞者には、５０万円相当の旅行券とエコ製品２点の副賞が贈られます。

4 different “alphabets”: Chinese characters, hiragana syllabary for inflectional endings and functional words, katakana syllabary for transcription of foreign words and other uses, and latin. No spaces (as in Chinese). End user can express query entirely in hiragana!

Arabic script



ك ِ ت ا ب ٌ ⇐ كِتَابٌ

un b ā t i k

/kitābun/ *‘a book’*

NLTK (Natural Language Toolkit)

- The Natural language toolkit (NLTK) is a collection of Python libraries designed especially for identifying and tag parts of speech found in the text of natural language like English.

- The **Natural Language Toolkit**, or more commonly **NLTK**, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.[4] It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania.[5] NLTK includes graphical demonstrations and sample data.

- The Natural language toolkit (NLTK) is a collection of Python libraries designed especially for identifying and tag parts of speech found in the text of natural language like English.

- Installing NLTK

```
pip install nltk
```

- If we are using Anaconda, then a Conda package for NLTK can be built by using the following command

```
conda install -c anaconda nltk
```

```
import nltk
Nltk.download()
```

Tokenizing sentences into words

word_tokenize module

- Used for basic word tokenization.

```
import nltk
nltk.download('punkt’)
from nltk.tokenize import word_tokenize
word_tokenize(‘This is my favourite book’)
```

WordPunktTokenizer Class

- An alternative word tokenizer that splits all punctuation into separate tokens.

```
from nltk.tokenize import WordPunctTokenizer
tokenizer = WordPunctTokenizer()
tokenizer.tokenize("won't")
```

Tokenizing text into sentences

- NLTK provides **sent_tokenize** module for this purpose.

```
  import nltk
  from nltk.tokenize import sent_tokenize
text = "Let us understand the difference between sentence & word tokenizer.
 It is going to be a simple example."
  sent_tokenize(text)
```

2.Remove Stop words

- Some common words that are present in text but do not contribute in the meaning of a sentence.
- The most common stop words are ‘the’ and ‘a’.
- NLTK stopwords corpus

```
from nltk.corpus import stopwords
```

```
 nltk.download('stopwords’)
 from nltk.corpus import stopwords
 stopwords.words('english’)
english_stops = set(stopwords.words('english’))
words = ['I', 'am', 'a', 'writer’]
[word for word in words if word not in english_stops]
```

Stop words

- stop words = extremely common words which would appear to be of little value in helping select documents matching a user need
- Examples: a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with
- But you need stop words for phrase queries, e.g. “King of Denmark”
- Most web search engines index stop words.

Stemming and Lemmatization

Stemming and Lemmatization both generate the root form of the inflected words.

In grammar, inflection is the modification of a word to express different grammatical categories such as tenses, person gender or mood.

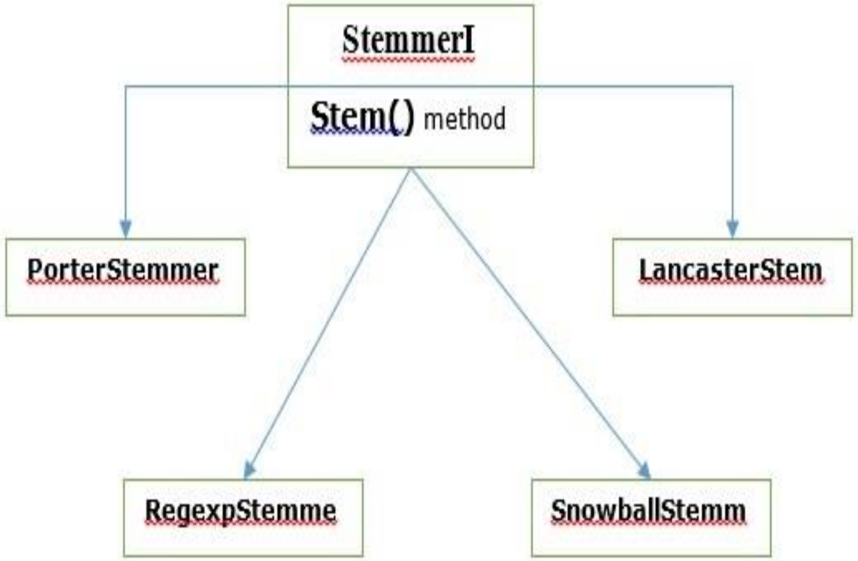
3.Stemming

- Stemming is a technique used to extract the base form of the words by removing affixes from them.

- It is just like cutting down the branches of a tree to its stems.

- For example, the stem of the words ***eating, eats, eaten*** is ***eat***.

- Various Stemming algorithms



Stemming

- Definition of stemming: Crude heuristic process that chops off the ends of words in the hope of achieving what “principled”
- lemmatization attempts to do with a lot of linguistic knowledge.
- Language dependent
- Often inflectional and derivational
- Example for derivational: automate, automatic, automation all reduce to automat

Porter‘s algorithm

- Most common algorithm for stemming English
- Results suggest that it is at least as good as other stemming options
- Conventions + 5 phases of reductions
- Phases are applied sequentially
- Each phase consists of a set of commands.
  - Sample command: Delete final ement if what remains is longer than 1 character
  - replacement → replac
  - cement → cement
- Sample convention: Of the rules in a compound command, select the one that applies to the longest suffix.

Porter stemmer: A few rules

Rule	**Example**
SSES → SS	caresses → caress
IES → I	ponies → poni
SS → SS	caress → caress
S →	cats → cat

Porter stemming algorithm

- Most common stemming algorithms
- Designed to remove and replace well-known suffixes of English words.
- PorterStemmer class

```
import nltk
from nltk.stem import PorterStemmer
word_stemmer = PorterStemmer()
word_stemmer.stem('writing')
```

4. Lemmatization

- Lemmatization technique is like stemming.
- The output we will get after lemmatization is called ‘lemma’, which is a root word.
- Reduce inflectional/variant forms to base form
    - Example: am, are, is → be
    - Example: car, cars, car’s, cars’ → car
    - Example: the boy’s cars are different colors → the boy car be different color
  - Lemmatization implies doing “proper” reduction to dictionary headword form (the lemma).
  - Inflectional morphology (cutting → cut) vs. derivational morphology (destruction → destroy)
- After lemmatization, we will be getting a valid word that means the same thing.

4. Lemmatization

- Lemmatization technique is like stemming.
- The output we will get after lemmatization is called ‘lemma’, which is a root word.
- After lemmatization, we will be getting a valid word that means the same thing.
- NLTK provides **WordNetLemmatizer** class which is a thin wrapper around the **wordnet** corpus.
- This class uses **morphy()** function to the **WordNet CorpusReader** class to find a lemma

```
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
lemmatizer.lemmatize('books')
```

Difference between Stemming & Lemmatization

- Stemming uses the stem of the word, while lemmatization uses the context in which the word is being used and the root form or lemma
- Stemming algorithms work by cutting off the end or the beginning of the word, taking into account a list of common prefixes and suffixes that can be found in an inflected word.
- Lemmatization, on the other hand, takes into consideration the morphological analysis of the words

```
import nltk

from nltk.stem import PorterStemmer

word_stemmer = PorterStemmer()

word_stemmer.stem('studies’)

Output : Studi
```

```
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
lemmatizer.lemmatize('studies’)
Output : study
```