Tokenization and Lemmatization

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

DataCamp

Text sources

- News articles
- Tweets
- Comments

Making text machine friendly

- `Dogs` , `dog`
- `reduction` , `REDUCING` , `Reduce`
- `don't` , `do not`
- `won't` , `will not`

Text preprocessing techniques

- Converting words into lowercase
- Removing leading and trailing whitespaces
- Removing punctuation
- Removing stopwords
- Expanding contractions
- Removing special characters (numbers, emojis, etc.)

Tokenization

```
"I have a dog. His name is Hachi."
```

Tokens:

```
["I", "have", "a", "dog", ".", "His", "name", "is", "Hachi", "."]
```

```
"Don't do this."
```

Tokens:

```
["Do", "n't", "do", "this", "."]
```

Tokenization using spaCy

```python
import spacy

# Load the en_core_web_sm model
nlp = spacy.load('en_core_web_sm')

# Initiliaze string
string = "Hello! I don't know what I'm doing here."

# Create a Doc object
doc = nlp(string)

# Generate list of tokens
tokens = [token.text for token in doc]
print(tokens)
```

```
['Hello','!','I','do',"n't",'know','what','I',"'m",'doing','here','.']
```

Lemmatization

- Convert word into its base form
  - `reducing`, `reduces`, `reduced`, `reduction` → `reduce`
  - `am`, `are`, `is` → `be`
  - `n't` → `not`
  - `'ve` → `have`

Lemmatization using spaCy

```
import spacy

# Load the en_core_web_sm model
nlp = spacy.load('en_core_web_sm')
# Initiliaze string
string = "Hello! I don't know what I'm doing here."
# Create a Doc object
doc = nlp(string)


# Generate list of lemmas
lemmas = [token.lemma_ for token in doc]
print(lemmas)
```

```
['hello','!','-PRON-','do','not','know','what','-PRON','be','do','here', '.']
```

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

DataCamp

Text cleaning

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

DataCamp

Text cleaning techniques

- Unnecessary whitespaces and escape sequences
- Punctuations
- Special characters (numbers, emojis, etc.)
- Stopwords

isalpha()

```
"Dog".isalpha()
```

```
True
```

```
"3dogs".isalpha()
```

```
False
```

```
"12347".isalpha()
```

```
False
```

```
"!".isalpha()
```

```
False
```

```
"?".isalpha()
```

```
False
```

A word of caution

- Abbreviations: `U.S.A` , `U.K` , etc.
- Proper Nouns: `word2vec` and `xto10x` .
- Write your own custom function (using regex) for the more nuanced cases.

Removing non-alphabetic characters

```
string = """
OMG!!!! This is like    the best thing ever \t\n.
Wow, such an amazing song! I'm hooked. Top 5 definitely. ?
"""
import spacy

# Generate list of tokens
nlp = spacy.load('en_core_web_sm')
doc = nlp(string)
lemmas = [token.lemma_ for token in doc]
```

Removing non-alphabetic characters

```
...
...
# Remove tokens that are not alphabetic
a_lemmas = [lemma for lemma in lemmas
            if lemma.isalpha() or lemma == '-PRON-']

# Print string after text cleaning
print(' '.join(a_lemmas))
```

```
'omg this be like the good thing ever wow such an amazing song -PRON- be hooked top definitely'
```

Stopwords

- Words that occur extremely commonly
- Eg. articles, be verbs, pronouns, etc.

Removing stopwords using spaCy

```
# Get list of stopwords
stopwords = spacy.lang.en.stop_words.STOP_WORDS
string = """
OMG!!!! This is like    the best thing ever \t\n.
Wow, such an amazing song! I'm hooked. Top 5 definitely. ?
"""
```

Removing stopwords using spaCy

```
...
...
# Remove stopwords and non-alphabetic tokens
a_lemmas = [lemma for lemma in lemmas
            if lemma.isalpha() and lemma not in stopwords]
# Print string after text cleaning
print(' '.join(a_lemmas))
```

```
'omg like good thing wow amazing song hooked definitely'
```

Other text preprocessing techniques

- Removing HTML/XML tags
- Replacing accented characters (such as é)
- Correcting spelling errors

A word of caution

Always use only those text preprocessing techniques that are relevant to your application.

Let's practice!

Part-of-speech tagging

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

DataCamp

Applications

- Word-sense disambiguation
  - `"The bear is a majestic animal"`
  - `"Please bear with me"`
- Sentiment analysis
- Question answering
- Fake news and opinion spam detection

POS tagging

- Assigning every word, its corresponding part of speech.

```
"Jane is an amazing guitarist."
```

- **POS Tagging:**
  - `Jane` → **proper noun**
  - `is` → **verb**
  - `an` → **determiner**
  - `amazing` → **adjective**
  - `guitarist` → **noun**

POS tagging using spaCy

```
import spacy

# Load the en_core_web_sm model
nlp = spacy.load('en_core_web_sm')
```

```
# Initiliaze string
string = "Jane is an amazing guitarist"
```

```
# Create a Doc object
doc = nlp(string)
```

POS tagging using spaCy

```
...
...
# Generate list of tokens and pos tags
pos = [(token.text, token.pos_) for token in doc]
print(pos)
```

```
[('Jane', 'PROPN'),
 ('is', 'VERB'),
 ('an', 'DET'),
 ('amazing', 'ADJ'),
 ('guitarist', 'NOUN')]
```

POS annotations in spaCy

- `PROPN` → proper noun
- `DET` → determinant
- spaCy annotations at https://spacy.io/api/annotation

POS	DESCRIPTION	EXAMPLES
`ADJ`	adjective	big, old, green, incomprehensible, first
`ADP`	adposition	in, to, during
`ADV`	adverb	very, tomorrow, down, where, there
`AUX`	auxiliary	is, has (done), will (do), should (do)
`CONJ`	conjunction	and, or, but
`CCONJ`	coordinating conjunction	and, or, but
`DET`	determiner	a, an, the

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

Named entity recognition

FEATURE ENGINEERING FOR NLP IN PYTHON

Rounak Banik
Data Scientist

DataCamp

Applications

- Efficient search algorithms
- Question answering
- News article classification
- Customer service

Named entity recognition

- Identifying and classifying named entities into predefined categories.
- Categories include person, organization, country, etc.

```
"John Doe is a software engineer working at Google. He lives in France."
```

- **Named Entities**
- `John Doe` $\rightarrow$ person
- `Google` $\rightarrow$ organization
- `France` $\rightarrow$ country (geopolitical entity)

NER using spaCy

```
import spacy
string = "John Doe is a software engineer working at Google. He lives in France."

# Load model and create Doc object
nlp = spacy.load('en_core_web_sm')
doc = nlp(string)

# Generate named entities
ne = [(ent.text, ent.label_) for ent in doc.ents]
print(ne)
```

```
[('John Doe', 'PERSON'), ('Google', 'ORG'), ('France', 'GPE')]
```

NER annotations in spaCy

- More than 15 categories of named entities
- NER annotations at https://spacy.io/api/annotation#named-entities

TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.

A word of caution

- Not perfect
- Performance dependent on training and test data
- Train models with specialized data for nuanced cases
- Language specific

Let's practice!

FEATURE ENGINEERING FOR NLP IN PYTHON

DataCamp