MSiA490 SEC20/28
Text Analytics
Lab 1 - Tokenization and Beyond

Timo Wang

Northwestern University

September 17, 2020

Overview

- Who is Timo?
- Tokenization with common libraries
- Hands on
- Quiz
- Thoughts & feedbacks

Who is Timo?

- Third-year PhD student in Computer Science
- Academic experience
  - Unsupervised multi-Document summarization
  - Knowledge graph completion through inference
  - Integrated frontend interface for AI systems
- Industry experience
  - Front-end developer

Tokenization with common libraries

Overview

- NLTK
- Spacy
- Stanford Stanza

Tokenization with common libraries

NLTK – Installation

Shell

```
pip install nltk
```

Python Console

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

Tokenization with common libraries

NLTK – Sentence-level

Python Console

```
from nltk.tokenize import sent_tokenize

text = 'I am happy. I am sleepy. I am dreamy.'
sents = sent_tokenize(text)
```

Tokenization with common libraries

NLTK – Word-level

Python Console

```
from nltk.tokenize import word_tokenize

text = 'I am happy. I am sleepy. I am dreamy.'
words = word_tokenize(text)
```

Tokenization with common libraries

Spacy – Installation

Shell

```
pip install spacy
python -m spacy download en_core_web_sm
```

Tokenization with common libraries

Spacy – Sentence-level

Python Console

```
import spacy

nlp = spacy.load('en_core_web_sm')
nlp.add_pipe(nlp.create_pipe('sentencizer'))

text = 'I am happy. I am sleepy. I am dreamy.'
doc = nlp(text)

sents = [sent.string.strip() for sent in doc.sents]
```

Tokenization with common libraries

Spacy – Word-level

Python Console

```
import spacy

nlp = spacy.load('en_core_web_sm')
nlp.add_pipe(nlp.create_pipe('sentencizer'))

text = 'I am happy. I am sleepy. I am dreamy.'
doc = nlp(text)

words = [token.text for token in doc]
```

Tokenization with common libraries

Stanford Stanza – Installation

Shell

```
pip install stanza
```

Python Console

```
import stanza
stanza.download('en')
```

Tokenization with common libraries

Stanford Stanza – Sentence-level

Python Console

```
import stanza

text = 'I am happy. I am sleepy. I am dreamy.'

nlp = stanza.Pipeline('en')
doc = nlp(text)
sents = [' '.join([token.text for token in sentence.
    tokens]).strip() for sentence in doc.sentences]
```

Tokenization with common libraries

Stanford Stanza – Word-level

Python Console

```
from functools import reduce

import stanza

text = 'I am happy. I am sleepy. I am dreamy.'

nlp = stanza.Pipeline('en')
words_by_sentence = [[token.text for token in sentence.
    tokens] for sentence in doc.sentences]
words = reduce(lambda lst1,lst2: lst1 + lst2,
    words_by_sentence)
```

Tokenization with common libraries

Extra resources

- Miniconda: https://docs.conda.io/en/latest/miniconda.html.
- NLTK: https://www.nltk.org/
- Spacy: https://spacy.io/usage/spacy-101
- Stanford Stanza: https://stanfordnlp.github.io/stanza/

Hands on

Tokenization, stemming and POS tagging

Task

1. Install and evaluate 2 to 3 text preprocessing libraries (in python or Java).
2. Download a publicly available text corpus.
3. Apply tokenization, stemming and POS tagging on the full corpus.

Note

Keep an eye out for the time/memory consumption of each library as well as their ease of use.

Quiz

Task 1

Which library do you find easiest to use for tokenization?

A NLTK

B Spacy

C Stanford Stanza

D Other

Quiz

Task 2

Which library runs fastest for POS tagging?

A NLTK

B Spacy

C Stanford Stanza

D Other

Quiz

Task 3

Which library appears most memory efficient on your machine/OS of choice?

A NLTK

B Spacy

C Stanford Stanza

D Other

Thoughts & feedbacks

- Make slides available before each lab.