NLP For Economists

Text Extraction, Cleanup and Preprocessing

Sowmya Vajjala

Munich Graduate School of Economics - LMU Munich

Guest Course, October 2020

Goals for this session

- What is text extraction and cleanup?
- Why should we preprocess text?
- What are some common pre-processing steps?
- What are some problems with existing solutions?

source material: Chapter 2 from `https://practicalnlp.ai`

Text extraction and cleanup

- text extraction and cleanup refers to the process of extracting raw text from the input data
- ... by removing all the other non-textual information, such as markup, metadata, etc.,
- .... and converting the text to the required encoding format.

Relevance of text extraction

- Text data can appear in various forms: PDF, HTML, XML or text, scanned text, twitter stream etc.
- Text extraction is a standard data-wrangling step, and we don't usually employ any NLP-specific techniques during this process.
- Yet, all our NLP depends on whether we managed to extract text properly in this step.

- we discussed some part of this issue in the previous video

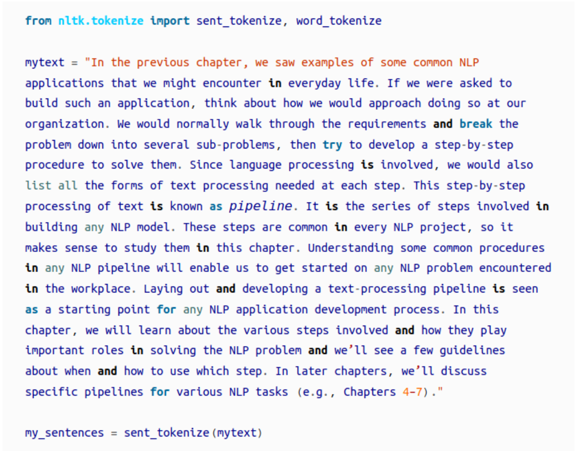
What does "cleanup" include?

- Unicode normalization: umlauts in German, smileys in social media text etc should be handled appropriately early on, to avoid errors later.
- Spelling correction/normalization, especially when dealing with social media text
- If it is a html webpage: remove all tags, java script code snippets, boiler plate content etc.

... and so on.

Pre-processing: Preliminaries

- Sentence splitting
- Word segmentation/tokenization



```
from nltk.tokenize import sent_tokenize, word_tokenize

mytext = "In the previous chapter, we saw examples of some common NLP
applications that we might encounter in everyday life. If we were asked to
build such an application, think about how we would approach doing so at our
organization. We would normally walk through the requirements and break the
problem down into several sub-problems, then try to develop a step-by-step
procedure to solve them. Since language processing is involved, we would also
list all the forms of text processing needed at each step. This step-by-step
processing of text is known as pipeline. It is the series of steps involved in
building any NLP model. These steps are common in every NLP project, so it
makes sense to study them in this chapter. Understanding some common procedures
in any NLP pipeline will enable us to get started on any NLP problem encountered
in the workplace. Laying out and developing a text-processing pipeline is seen
as a starting point for any NLP application development process. In this
chapter, we will learn about the various steps involved and how they play
important roles in solving the NLP problem and we'll see a few guidelines
about when and how to use which step. In later chapters, we'll discuss
specific pipelines for various NLP tasks (e.g., Chapters 4-7)."

my_sentences = sent_tokenize(mytext)
```

Pre-processing: Preliminaries

```
for sentence in my_sentences:
   print(sentence)
   print(word_tokenize(sentence))

Output:
In the previous chapter, we saw a quick overview of what is NLP, what are some
of the common applications and challenges in NLP, and an introduction to
different tasks in NLP.
['In', 'the', 'previous', 'chapter', ',', 'we', 'saw', 'a', 'quick',
'overview', 'of', 'what', 'is', 'NLP', ',', 'what', 'are', 'some', 'of', 'the',
'common', 'applications', 'and', 'challenges', 'in', 'NLP', ',', 'and', 'an',
'introduction', 'to', 'different', 'tasks', 'in', 'NLP', '.']
```

Caveats

- While readily available solutions work for most of our needs and most NLP libraries will have a tokenizer and sentence splitter bundled with them, it's important to remember that they're far from perfect.
- Sometimes, specialized tokenizers may be needed (e.g., TweetTokenizer in NLTK)
- Sometimes, during text extraction, we miss the structure of the text (e.g., extracting from a two column pdf). This will affect sentence/word splitting.

Frequent Preprocessing steps

- It is common to remove too frequent words (known as "stop words") while doing some NLP tasks (e.g., visualization, text classification, topic modeling etc)
- We also see people removing punctuation, digits, any special characters etc.
- however, depending on the task, they may be significant. e.g., smileys may be irrelevant for identifying events, but relevant for identifying sentiment.

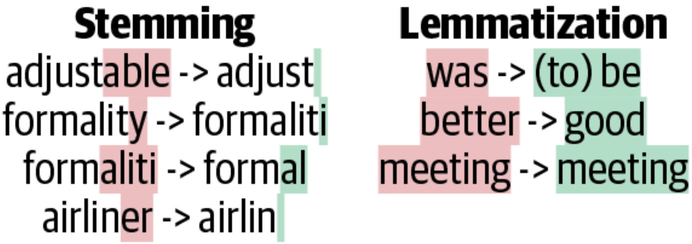
Frequent Preprocessing steps

The code example below shows how to remove stop words, digits, and punctuation and lowercase a given collection of texts, using a stopword list from nltk.

```
from nltk.corpus import stopwords
From string import punctuation
def preprocess_corpus(texts):
    mystopwords = set(stopwords.words("english"))
    def remove_stops_digits(tokens):
       return [token.lower() for token in tokens if token not in mystopwords
               and not token.isdigit() and token not in punctuation]
    return [remove_stops_digits(word_tokenize(text)) for text in texts]
```

Stemming and Lemmatization

- Stemming applies a fixed set of rules (e.g., if the word ends in "-es," remove "-es") to remove suffixes at the end of words.
- Lemmatization is the process of mapping all the different forms of a word to its base word, or lemma.



Stemming	Lemmatization
adjustable -> adjust	was -> (to) be
formality -> formaliti	better -> good
formaliti -> formal	meeting -> meeting
airliner -> airlin	

Stemming and Lemmatization

```
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()
word1, word2 = \cars", \revolution"
print(stemmer.stem(word1), stemmer.stem(word2))

from nltk.stem import WordNetLemmatizer
lemmatizer = WordnetLemmatizer()
print(lemmatizer.lemmatize("better", pos="a")) #a is for adjective

import spacy
sp = spacy.load('en_core_web_sm')
token = sp(u'better')
for word in token:
   print(word.text,  word.lemma_)
#outputs from different lemmatizers may be different.
```

Conclusion

- Many other steps such as language detection, code-mixing/transliteration, part of speech tagging, syntactic parsing etc.
- We choose what we want based on the scenario.
- Lot of existing NLP libraries and tools. None perfect, but in many cases, they meet our needs.
- Tools are specific to languages. You can explore German tools if you want.