Natural Language Processing

Ivan Corneillet

Data Scientist

Learning Objectives

After this lesson, you should be able to:

- Define natural language processing
- List common tasks associated with
  - Use-cases
  - Tokenization
  - Tagging and parsing
  - Stemming and lemmatization
- Demonstrate how to classify documents using *sklearn*

Here's what's happening today:

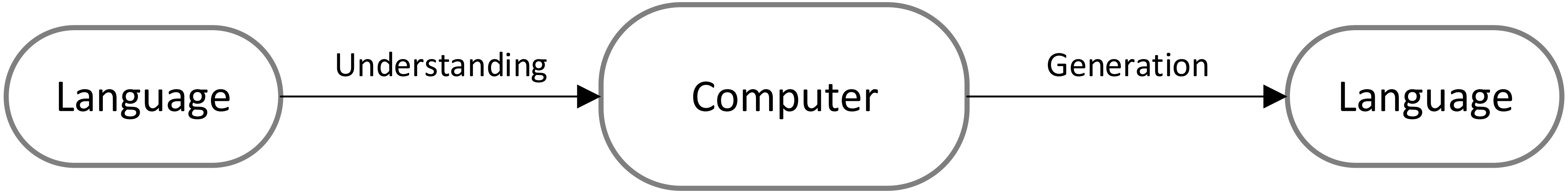
- Natural Language Processing
  - Understanding and generation
  - NLP is hard…
  - Tokenization
  - Tagging and parsing
  - Stemming and lemmatization
- Text Classification
  - Bag-of-words classification
  - Text Processing with *sklearn*
  - Term-Frequency and Inverse-Document-Frequency (TF-IDF)

Natural Language Processing

What is Natural Language Processing?

- Natural Language Processing (NLP) is the study of the computational treatment of natural (human) language, i.e., teaching computers how to understand (and generate) human language

Basic NLP Pipeline | Understanding and Generation



Understanding

- For most tasks, a fair amount of pre-processing is required to make the text digestible for our algorithms. We typically need to *add structure* to our *unstructured* data

Generation

- These tasks may range from simple classification tasks, such as deciding what category a piece of text falls into, to more complex tasks like translating or summarizing text

What are some real-world examples of NLP?

- Search engines (E.g., Google and Bing)
- Natural language assistants (E.g., Apple's Siri uses voice recognition to record a command and then various fairly advanced NLP engines to identify the question asked and possible answers)
- Machine translation (E.g., Google Translate)
- Question answering (E.g., IBM's Watson)
- News digest (E.g., Yahoo!)

Computers are confused by (human) language

- E.g., “Children make delicious snacks”

- *Are* Children delicious snacks?

- Do children *prepare* delicious snacks?

Each genre of text (e.g., blogs, emails, press releases, chats) presents different challenges to NLP

- E.g., newspapers news headlines
  - “Red tape holds up new bridges”
  - “Government head seeks arms”
  - “Blair wins on budget, more lies ahead”

Natural Language Processing

Tokenization

Tokenization is the task of separating a sentence into its constituent parts or *tokens*

- Determining the “words” of a sentence seems easy but can quickly become complicated with unusual punctuation (common in social media) or different language conventions

- What sort of difficulties may there be with the following sentence?
  - “The L.A. Lakers won the NBA championship in 2010, defeating the Boston Celtics”

“The L.A. Lakers won the NBA championship in 2010, defeating the Boston Celtics”

- To perform a proper analysis, we need to be able to identify that:
  - The periods in “L.A.” don’t mark the end of a sentence but an abbreviation
  - “L.A. Lakers” and “Boston Celtics” are one concept.
  - “2010” is the word used, not “2010,”

Tokenization Examples

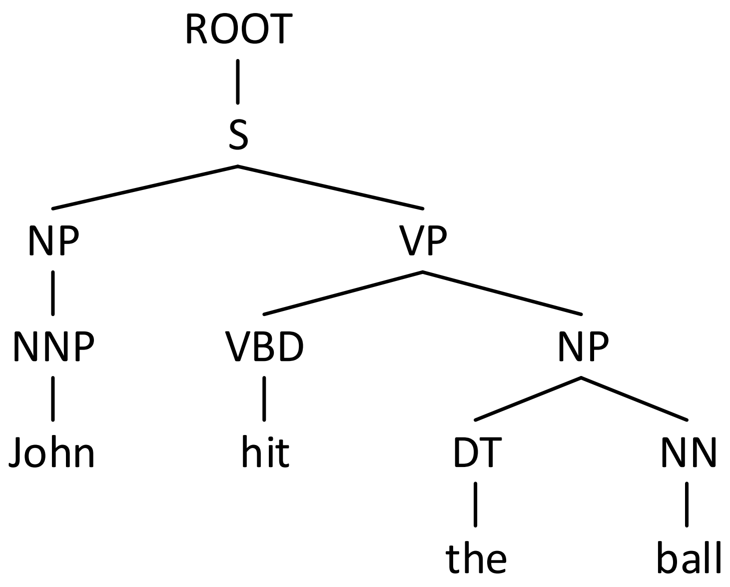
Sentence	Tokens
My house is located in Uptown.	[My, house, is, located, in, Uptown]
The Lakers are my favorite team.	[The, Lakers, are, my, favorite, team]
Data Science is the future!	[Data, Science, is, the, future]
GA has many locations.	[GA, has, many, locations]

Natural Language Processing

Tagging and Parsing

Tagging and Parsing

- In order to understand the various elements of a sentence, we need to *tag* important topics and *parse* their dependencies



DT - Determiner
NN - Noun, singular or mass
NNP - Proper noun, singular
NP - Noun phrase
S - Simple declarative clause
VBD - Verb, past tense
VP - Verb phrase

Tagging and Parsing (cont.)

- Our goal is to identify the *actors* and *actions* in the text in order to make informed decisions

- E.g., if we are processing financial news, we might need to identify which companies are involved and any actions they are taking
- E.g., if we are writing an assistant application, we might need to identify specific command phrases in order to determine what is being asked (e.g. “Siri, when is my next appointment?”)

Tagging and parsing is made up of a few overlapping subtasks

- Parts of speech tagging: What are the parts of speech in a sentence? (e.g. noun, verb, adjective)
- Named entity recognition: Can we identify *specific* proper nouns? Can we pick out people and locations?
- Chunking: Can we identify the pieces of the sentence that go together in meaningful chunks? (e.g. noun or verb phrases)

Tagging

```
John/NNP hit/VBD the/DT ball/NN
```

Parsing

```
(ROOT
  (S
    (NP (NNP John))
    (VP (VBD hit)
      (NP (DT the) (NN ball)))))
```

These subtasks are very difficult because language is complex and ever changing

- Most often, we are looking for heuristics to search through large amounts of text data
  - The results may not be perfect and that's okay
- These techniques rely on rule-based systems but more recent research has focused on more flexible systems, focusing on words used rather than on the structure of the sentence

Natural Language Processing

Stemming and Lemmatization

Stemming and lemmatization help identify common roots of words

- How would you describe the relationship between the terms ‘bad’ and ‘badly’?
- How about ‘different’ and ‘differences’?

Stemming is a crude process of removing common endings from words

- To stem a word is to reduce it to a base form, called the *stem*, after removing various suffixes and endings and, sometimes, performing some additional transformations
- In practice, prefixes are sometimes preserved, so ‘rescan’ will not be stemmed to ‘scan’

- E.g.,
  - badly → bad
  - computing → comput
  - computed → comput
  - wipes → wip
  - wiped → wip
  - wiping → wip

Lemmatization is a more refined process that uses specific language and grammar rules to derive the root of a word

- This is useful for words that do not share an obvious root such as ‘best’ and ‘better’

- E.g.,
  - best → good
  - better → good
  - good → good
  - wiping → wipe
  - hidden → hide
  - shouted → shout

Text Classification

Text Classification

- Text classification is the task of predicting what category or topic a piece of text is from

- For example, we may want to identify whether an article is a sports or business story
- Or whether has positive or negative sentiment

Text Classification (cont.)

- Typically, this is done by using the text as features and the label as the target output. This is referred to as *bag-of-words* classification
- To include text as features, we usually create a binary feature for each word, i.e., does this piece of text contain that word?

- As we do this, we need to consider several things
  - Does order of words matter?
  - Does punctuation matter?
  - Does upper or lower case matter?

To create binary text features, we first create a vocabulary to account for all possible words in our universe

$$x = \left(x_{aardvark}, \ldots, x_{ball}, \ldots, x_{hit}, \ldots, x_{John}, \ldots, x_{the}, \ldots, x_{zyzzogeton}\right)$$

Two simple bag-of-words approaches

- Mark the vocabulary used in each document. E.g., "John hit the ball"

$$x = \left(\underbrace{\dots}_{False}, \underbrace{x_{ball}}_{True}, \underbrace{\dots}_{False}, \underbrace{x_{hit}}_{True}, \underbrace{\dots}_{False}, \underbrace{x_{John}}_{True}, \underbrace{\dots}_{False}, \underbrace{x_{the}}_{True}, \underbrace{\dots}_{False}\right)$$

- Count the vocabulary used in each document

$$x = \left(\underbrace{\dots}_{0}, \underbrace{x_{ball}}_{1}, \underbrace{\dots}_{0}, \underbrace{x_{hit}}_{1}, \underbrace{\dots}_{0}, \underbrace{x_{John}}_{1}, \underbrace{\dots}_{0}, \underbrace{x_{the}}_{1}, \underbrace{\dots}_{0}\right)$$

Text Classification

Term Frequency and Inverse Document Frequency (TF-IDF)

Term Frequency – Inverse Document Frequency (TF-IDF)

- An alternative bag-of-words approach is a Term Frequency – Inverse Document Frequency (TF-IDF) representation
- TF-IDF uses the product of two intermediate values, the Term Frequency and Inverse Document Frequency

Term Frequency (TF)

$$tf(t,d) = \frac{number\ of\ occurences\ of\ term\ t\ in\ document\ d}{number\ of\ terms\ in\ document\ d}$$

- *Term Frequency* assigns high weight to frequent words (words that appear frequently) in a document

Inverse Document Frequency (IDF)

$$idf(t, D) = \frac{total\ number\ of\ documents\ D}{number\ of\ documents\ term\ t\ appears\ in\ them}$$

- *Document Frequency* is the percentage of documents that a particular word appears in
- *Inverse Document Frequency* is *Document Frequency*'s inverse

- *Inverse Document Frequency* assigns high weight to rare words in all the documents

Term Frequency – Inverse Document Frequency (TF-IDF)

$$tf{-}idf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

- The intuition behind *TF-IDF* is to assign high weight to words that either
  - appear frequently in this document or
  - appear rarely in other documents (and are therefore specific to this document)

Activity | Term Frequency – Inverse Document Frequency (TF-IDF)

EXERCISE

DIRECTIONS (15 minutes)

1. Let's consider the following set of documents *D* after tokenization (to keep the words readable, we'll skip stemming/lemmatization):
   a. ['flu', 'depression', 'flu', 'arthritis', 'flu']
   b. ['flu', 'flu', 'flu']
   c. ['arthritis', 'depression']
   d. ['depression', 'flu', 'depression']
   e. ['flu', 'flu']
   f. ['measles', 'flu', 'measles']
2. Calculate the TF-IDF matrix
3. When finished, share your answers with your table

DELIVERABLE

Answers to the above questions

Activity | Term Frequency – Inverse Document Frequency (TF-IDF)

EXERCISE