DLI Accelerated Data Science Teaching Kit

Lecture 20.1 - Basics: Preprocessing, Representation, Word Importance

Text is everywhere

We use documents as primary information artifact in our lives

Our access to documents has grown tremendously thanks to the Internet

- *WWW:* webpages, Twitter, Facebook, Wikipedia, Blogs, ...
- *Digital libraries:* Google books, ACM, IEEE, ...
- Lyrics, closed caption... (youtube)
- Police case reports
- Legislation (law)
- Reviews (products, rotten tomatoes)
- Medical reports (EHR - electronic health records)
- Job descriptions

Big (Research) Questions

... in understanding and gathering information from text and document collections

- establish authorship, authenticity; plagiarism detection
- classification of genres for narratives (e.g., books, articles)
- tone classification; sentiment analysis (online reviews, twitter, social media)
- code: syntax analysis (e.g., find common bugs from students' answers)

Popular **Natural Language Processing** (NLP) libraries

- **Stanford NLP**
- **OpenNLP**
- **NLTK** (python)

tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing

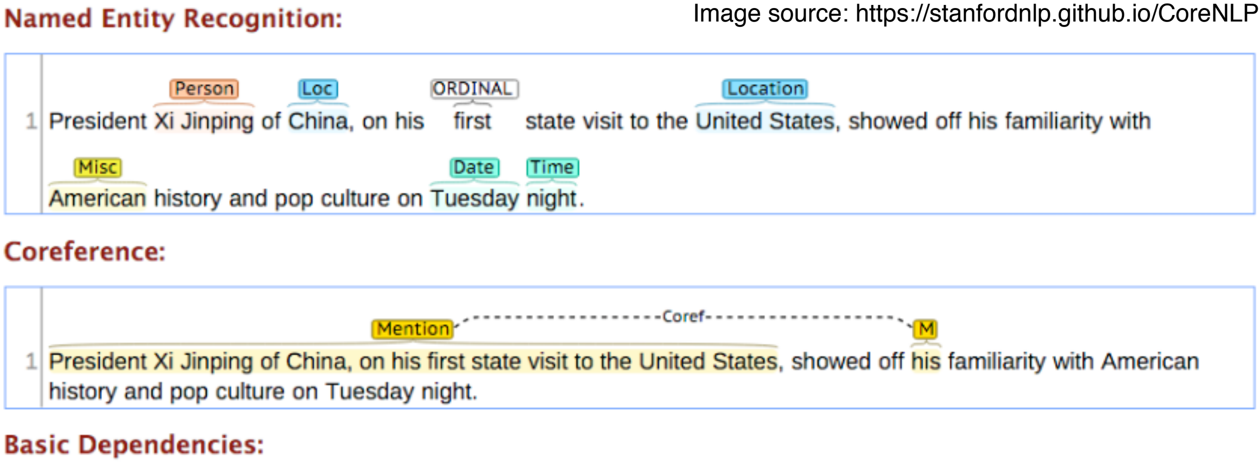

Image source: https://stanfordnlp.github.io/CoreNLP/

Named Entity Recognition:

1 President Xi Jinping of China, on his first state visit to the United States, showed off his familiarity with American history and pop culture on Tuesday night.

Coreference:

1 President Xi Jinping of China, on his first state visit to the United States, showed off his familiarity with American history and pop culture on Tuesday night.

Basic Dependencies:

Outline

- **Preprocessing** (e.g., stemming, remove stop words)
- **Document representation** (most common: bag-of-words model)
- **Word importance** (e.g., word count, TF-IDF)
- **Latent Semantic Indexing** (find “concepts” among documents and words), which helps with **retrieval**

Stemming

Reduce words to their **stems** (or base forms)

Words: compute, computing, computer, ...

Stem: comput

Several classes of algorithms to do this:

- Stripping suffixes, lookup-based, etc.

http://en.wikipedia.org/wiki/Stemming
Stop words: http://en.wikipedia.org/wiki/Stop_words

Bag-of-words model

Represent each **document** as a **bag of words**, ignoring words’ ordering. Why? For **simplicity**.

Unstructured text becomes **a vector of numbers**

e.g., docs: “I like visualization”, “I like data”.

1 : “I”

2 : “like”

3 : “data”

4 : “visualization”

“I like visualization” ➡ [1, 1, 0, 1]

“I like data” ➡ [1, 1, 1, 0]

TF-IDF

A word’s importance score in a **document**, among **N documents**

When to use it? Everywhere you use “word count”,
you can likely use TF-IDF.

TF: **term** frequency
= #appearance in **document**
(high, if terms appear many times)

IDF: inverse document frequency
= log(**N** / #document containing the **term**)
(penalize “common” words appearing in almost any documents)

Final score = TF * IDF
(higher score ➡ word is more “characteristic” of document)

Example: http://en.wikipedia.org/wiki/Tf–idf#Example_of_tf.E2.80.93idf

Vector Space Model

Why?

Each document ➡ vector

Each query ➡ vector

Search for documents ➡ find “similar” vectors

Cluster documents ➡ cluster “similar” vectors

DLI Accelerated Data Science Teaching Kit

Thank You