Lecture 13- Natural Language Processing

Hamed Hasheminia

Agenda

- What is Natural Language Processing?
- NLP Applications
- Basic NLP Practice

Natural Language Processing

- The interface between human and computer language
- Natural language processing is the task of extracting meaning and information from text documents.
- 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.
- Can you come up with some real-world examples of NLP? (I think some of you are already using it!)

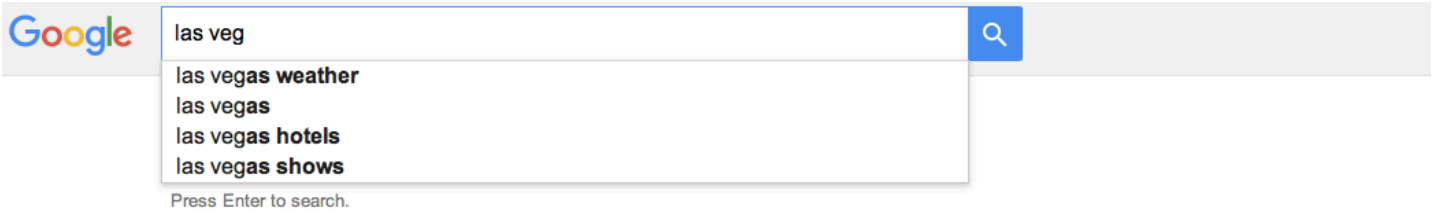
How do we teach computers to understand human language?

- How do we discern the meaning of **sea** in the following sentences?
  - “The driftwood floated in from the **sea**”
  - My cousin is dealing with a **sea** of troubles.”
- Large body of water: **the sea**
- Figurative large quantity **sea of**

NLP Applications

- Speech Recognition

- Machine Translation
- Auto Correction
- Sentiment analysis
- Topic Modeling



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.

Tokenization Examples

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.]

Lemmatization and Stemming

- 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 ‘better’ and ‘best’.
- What are some other examples of words that do not share an obvious root?

Lemmatization and Stemming

Lemmatization

shouted → shout

best → good

better → good

good → good

Stemming

badly → bad

computing → comput

computed → comput

Stop Words

- Some words are so common that they provide no information to a statistical language model.
- Here are some examples of stop words “the, in, a, ….”
- We should remove these stop words.

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.
- 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?

Text Classification

"It's a great advantage not to drink among hard drinking people."

Feature	Value	Feature	Value
it's	1	people	1
great	1	withhold	0
good	0	random	0
advantage	1	smoke	0
not	1	among	1
think	0	whenever	0
drink	1	thoughtful	0
from	0	inexhaustible	0
hard	1	men	0
drinking	1	Nick	0

Regularizing the models

- **REMEMBER**: Using all of the words can be useful, but we may need to use regularization to avoid overfitting. Otherwise, rare words may cause the model to overfit and not generalize.
- We can use CV and Lasso/Ridge type of modification to regularize our models.
- A better way to lower the high dimension of our input data is using Principle Component Method.

Term Frequency – Inverse Document Frequency

- TF-IDF uses the product of two intermediate values, the *Term Frequency* and *Inverse Document Frequency*.

- *Term Frequency* is just the number of times a word appears in the document (i.e. count).

- *Document Frequency* is the percentage of documents that a particular word appears in.

- For example, Document Frequency of “the” would be 100% while “Syria” is much lower.

- *Inverse Document Frequency* is just 1/Document Frequency.

TF-IDF (Term Frequency-inverse document frequency)

- TF-IDF formula in Python

- TF-IDF(t, d) = tf(t,d) * (idf(t,d) + 1)
- tf(t, d) How many times t is repeated in document d
- idf(t, d) = log((1 + n_d) / (1 + df(t, d))
  - n_d = total number of documents
  - df(t, d) = Among all documents, how may of them have t in them
- In python – TF-IDF vector is automatically normalized.

Example

- I love Data Science.
- I am a GA student
- I am a student and a GA member.

- Tf-idf(t = love, d = 1)
  - tf(love, 1) = 1 ("Love" is repeated once in document 1 – first sentence)
  - $n_d = 3$ (We have three documents/3 sentences)
  - Df(t = love, d) = 1 (among all sentences "Love" only appeared once)
  - TF-idf = 1 * (log((1+3)/(1+1))+ 1) = 1 *(1 + log(2)) = 1.69

- TF-idf(t = love , d = 1) = 0 * (1 + log(4/2)) = 0
- TF-idf(t = I , d = 2) = 1 * (1 + log(4/4)) = 1

TF-IDF

- The intuition is that the words that have high weight are those that either appear frequently in this document or appear rarely in other documents (and are therefore unique to this document).
- This is a good alternative to using a static set of stop words.

Summary

- We learned about Natural Language Processing
- Tokenization/Lemmatization/Stemming
- Parsing and tagging trees
- Stop Words
- Bag-of-Words
- TF-IDF