Introduction to NLP

Recommended Reading

Speech and Language Processing, by Dan Jurafsky and James H. Martin

Available here: https://web.stanford.edu/~jurafsky/slp3/

Natural Language Processing with Python, by Steven Bird, Ewan Klein, and Edward Loper

Available here: https://www.nltk.org/book/

Introduction

NLP: A set of methods for making human language accessible to computers.

Combines computational linguistics with statistical, machine learning, and deep learning models.

Introduction

Applications of NLP:

- Classification (eg. spam filtering or sentiment analysis)
- Automatic machine translation
- Dialogue systems (chatbots)
- Speech recognition/speech-to-text

Introduction

Python libraries for NLP

- NLTK
- Gensim
- SpaCy

Machine Learning vs. Linguistics

Computational linguistics (rule-based modeling of human language) - language is the object of study

NLP - focused on the design and analysis of computational algorithms and representations for processing natural human language.

See this article for more discussion on this topic.

Machine Learning vs. Linguistics

Can take a machine learning only approach, training end-to-end systems to go from raw text to a desired output (summary, database, or translation).

Or can take text and convert to general purpose linguistic structures (eg. morphemes, part-of-speech tags).

NLP Challenges

- Text is fundamentally discrete
- Distribution over words resembles a power law (see Zipf's law), so NLP systems have to be robust to observations that don't occur in the training data.
- Language is compositional. Words combine to create phrases and phrases combine to create larger phrases. That is, there is recursive structure in language.

Text Normalization

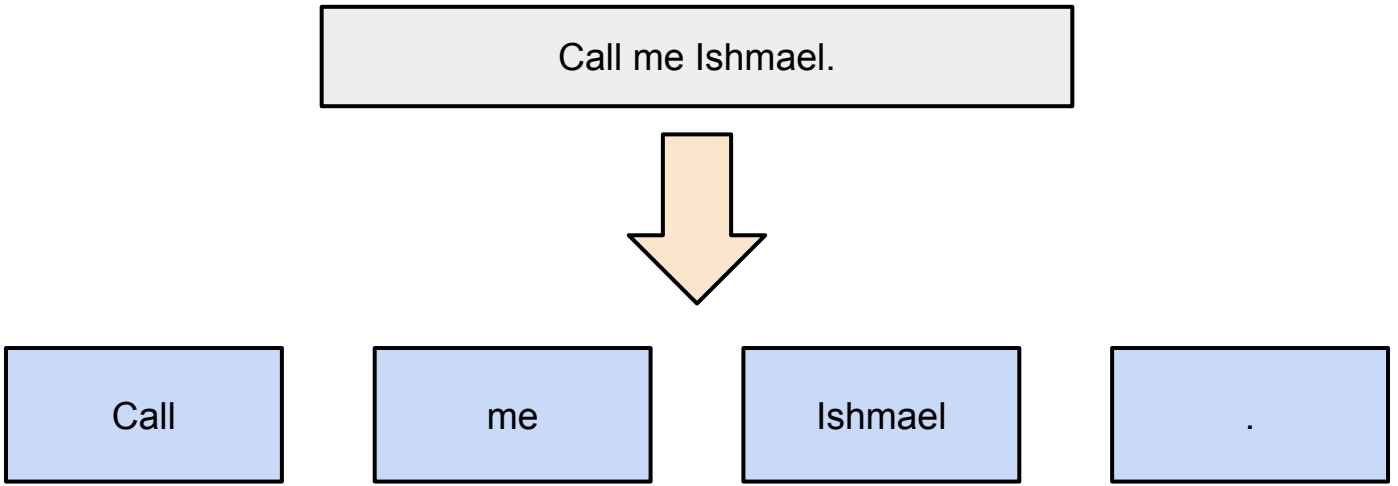
Any NLP task starts with converting from raw text to usable features.

This process is called **text normalization**.

It can include tokenization, sentence segmentation, stemming, and lemmatization.

Tokenization

Tokenization: Takes an input (string) and a token type (meaningful unit of text, like a word), and splits into tokens.



Tokenization

It’s not always obvious what constitutes a token.

Issues in tokenization:

- New York -> [“New”, “York”]
- forward-looking -> [“forward”, “-”, “looking”]

Tokenization

Sentence Tokenization (dividing text into sentences) can also be challenging.

CELLULAR COMMUNICATIONS INC. sold 1,550,000 common shares at $21.75 each yesterday, according to lead underwriter L.F. Rothschild & Co.

There are various techniques for this. For example, the Punkt tokenizer (which is used by the NLTK sent_tokenize function).

Tokenization

NLTK: Tokenize package (https://www.nltk.org/api/nltk.tokenize.html)

Includes various word tokenizers and sentence tokenizers.

Can also write your own using regex.

Text Normalization

Other ways to normalize:

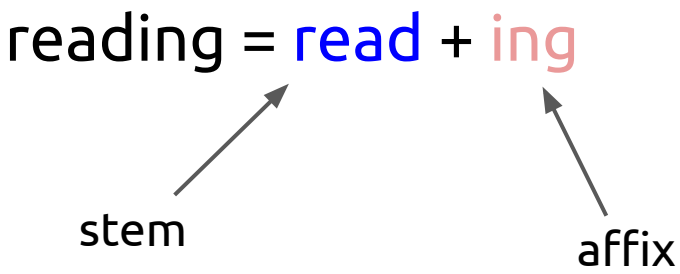
- Case folding (lowercasing all text)
  - Sometimes can lead to problems (Apple vs. apple)
  - Keeping case can be useful for tasks like sentiment analysis
- Stemming
- Lemmatization

Morphology

Morpheme: the small meaningful units that make up words

Stems: The core meaning-bearing units

Affixes: Bits and pieces that adhere to stems, often with grammatical functions.



Text Normalization - Lemma vs. Wordform

Lemma: Same stem, part of speech, rough word sense

- cat and cats = same lemma

Wordform: the full inflected surface form

- cat and cats = different wordforms

Morphology

Stemming: Reduce words to their stems

Stemming is the crude chopping of affixes, and is language dependent.

automate, automatic, automation -> automat

Morphology

Lemmatization: Uses knowledge about a language's structure to reduce words down to their lemmas, the canonical or dictionary forms of words.

Requires more information than stemming (which is typically rules-based).

Morphology

Should you stem or lemmatize?

It reduces the feature space of text data. It can be useful for text retrieval (eg. in a search application).

However, research shows that in some cases (topic modeling, specifically) that stemming can hurt performance.

By reducing the feature space, you are potentially discarding useful information.