TECHNISCHE UNIVERSITÄT DRESDEN

SEMANTIC COMPUTING

Lecture 3: Natural Language Processing and Language Modeling

Dagmar Gromann

International Center For Computational Logic

TU Dresden, 2 November 2018

TECHNISCHE UNIVERSITÄT DRESDEN

Overview

- NLP pipeline continued
- NLP applications
- Language Modeling

NLP pipeline continued

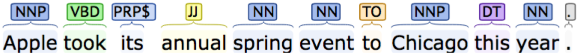
Basic NLP pipeline - Syntactic Analysis

Input: Apple took its annual spring event to Chicago this year.

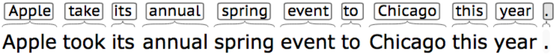
Tokenization

Apple / took / its / annual / spring / event / to / Chicago / this / year

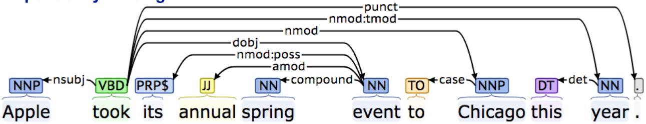
Part-of-Speech Tagging



Lemmatization



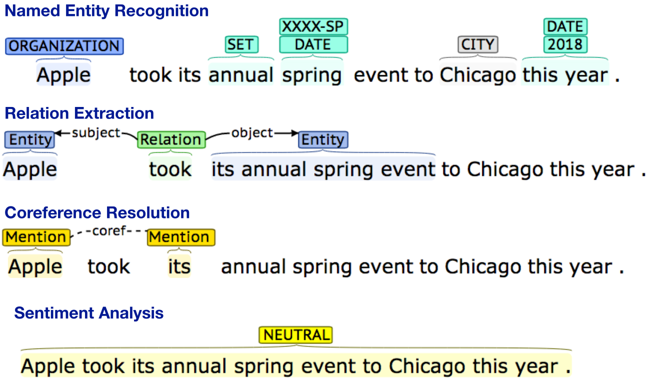
Dependency Parsing



Examples generated with the Stanford Core NLP toolset (`http://corenlp.run/`).

Basic NLP pipeline - Semantic Analysis

Input: Apple took its annual spring event to Chicago this year.



Examples generated with the Stanford Core NLP toolset (`http://corenlp.run/`).

Named Entity Recognition

Subtask of information extraction that locates and classifies named entities, i.e., a real-world object that can be denoted with a proper name - person, organization, location, products, etc.

```
from nltk.tag.perceptron import PerceptronTagger
tagger = PerceptronTagger()

sent = "Apple took its annual spring event to Chicago this year."
tags = tagger.tag(nltk.word_tokenize(sent))
sent = nltk.ne_chunk(tags, binary=True) #
print(sent)
```

(S
(NE Apple/NNP)
took/VBD
its/PRP$
annual/JJ
spring/NN
event/NN
to/TO
(NE Chicago/NNP)
this/DT
year/NN
./.)

Relation Extraction from Text

Also a subtask of information extraction with two main processes:

1. extraction of entities (NER)
   - People, organizations, locations, times, dates, prices, etc.
2. extraction of relations between those entities
   - Located in, employed by, part of, etc.

How?

- lexico-syntactic patterns (X is_a Y: “A dog is_a mammal.”)
- patterns and rules (PERSON [be]? (born) PREP PLACE, “Trump was born in New York City.")
- Machine learning (supervised, unsupervision,...)
- Deep learning (all potential architectures)

Code Example Relex

Running Stanford CoreNLP from the command line [1].

```
java -cp "*" -Xmx2g edu.stanford.nlp.pipeline.StanfordCoreNLP
-annotators tokenize,ssplit,pos,lemma,ner,parse,relation -file input.txt
Java 9: java --add-modules java.se.ee
Alternative:   java -mx2g -cp "*" edu.stanford.nlp.naturalli.OpenIE
```

```
<MachineReading>
  <entities>
    <entity id="EntityMention-1">LOCATION
      <span start="0" end="1"/>
      <probabilities/>
    </entity>
    <entity id="EntityMention-2">O
      <span start="1" end="2"/>
      <probabilities/>
    </entity>
    <entity id="EntityMention-3">O
      <span start="5" end="6"/>
      <probabilities/>
    </entity>
  </entities>
  <relations/>
</MachineReading>
Alternative: TU Dresden       is located in     Germany
```

[1] https://stanfordnlp.github.io/CoreNLP/cmdline.html

Coreference Resolution

Coreference resolution is the task of identifying all expressions (mentions) in a text that refer to the same real-world entity, such as

"She has not told her friend about that story because it is too embarrassing for her."

Code Example Coref

Running StanfordCoreNLP from the command line [1].

"She has not told her friend about that story because it is too embarrassing for her."

```
java -cp "*" -Xmx3g edu.stanford.nlp.pipeline.StanfordCoreNLP
-annotators tokenize,ssplit,pos,lemma,ner,parse,dcoref -file input.txt
Java 9: java --add-modules java.se.ee
```

```
<coreference>
  <coreference>
    <mention representative="true">
      <text>She</text>
      ...
    </mention>
    <mention>
      <text>her</text>
      ....
    </mention>
  </coreference>
```

```
  <coreference>
    <mention representative="true">
                ...
      <text>that story</text>
    </mention>
    <mention>
      <text>it</text>
      ....
    </mention>
  </coreference>
</coreference>
```

[1] https://stanfordnlp.github.io/CoreNLP/cmdline.html

Sentiment Analysis

Computational study of opinions, sentiments, evaluations, attitudes, affects, emotions, etc. found in text. Also called opinion mining.

- Polarity detection: positive, negative, neutral or on a scale of 1 to 5 how positive, negative or neutral
- Valence detection: valence is the "goodness" or "badness" of an emotion, which means it takes sentiment intensity into account (e.g. 0.83 negative on a scale from 0 to 1)
- Objectivity: how objective or subjective is a statement?
- Emotion classification: anger, fear, sadness, joy, etc.
- Stance classification: for or against a position

Sentiment Analysis - Example

Massive business value for all sentiment analysis applications - complaint management, product improvement, word-of-mouth marketing analysis, brand awareness, etc.

Movie reviews

- “Get off the screen.” 👎
- “I watched the screening tonight and I really loved it.” 👍

Product rating

- ★☆☆☆☆“The echo dot turned Alexa into a douchebag salesman.”
- ★★★☆☆“A fun gadget, but the jury is still out on how useful it actually is."
- ★★★★★“The Smartest of Them All!!!”

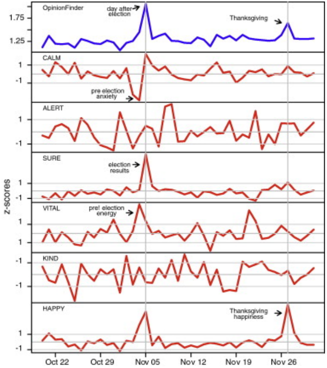
Sentiment Analysis on Twtitter

Twitter analysis

Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. Journal of computational science, 2(1), 1-8.

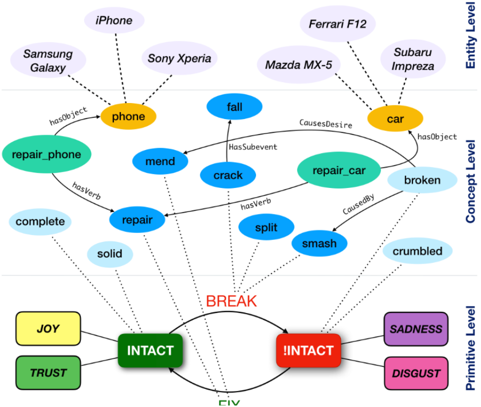
Measurement of the collective mood state based on large-scale Twitter feeds analysis and its correlation to the value of the Dow Jones Industrial Average (DJIA) over time.

Comparison: presidential election and Thanksgiving (as baseline)



SenticNet: Concept-Level Sentiment Analysis

Cambria, E., Poria, S., Hazarika, D., & Kwok, K. (2018). SenticNet 5: discovering conceptual primitives for sentiment analysis by means of context embeddings. In AAAI.



Basic Code Example using NLTK Vader

VADER = Valence Aware Dictionary and sEntiment Reasoner

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sia = SentimentIntensityAnalyzer()

sentences = ["Get off the screen.", "I watched the screening
        tonight and I really loved it.", "The Smartest of Them All",
        "Very bad movie!"]

for sentence in sentences:
    print(sentence)
    ss = sia.polarity_scores(sentence)
    for k in sorted(ss):
        print('{0}: {1}, '.format(k, ss[k], end=''))
```

Get off the screen.
compound: 0.0, neg: 0.0, neu: 1.0, pos: 0.0
I watched the screening tonight and I really loved it.
compound: 0.6361, neg: 0.0, neu: 0.625, pos: 0.375
The Smartest of Them All
compound: 0.6124, neg: 0.0, neu: 0.5, pos: 0.5
Very bad movie!

compound: -0.623, neg: 0.671, neu: 0.329, pos: 0.0

NLP tasks

Each of the presented processing steps in the NLP pipeline is a whole research field in its own right with many different approaches to tackle its core problems. Some more:

- Word Sense Disambiguation: identify the correct sense of a word in a context, e.g. Tutorial 1 Exercise on WordNet
- Semantic Role Labeling (shallow parsing): assigning labels to elements of a sentence that indicate their role, e.g. agent, goal, means. Demo: Curator
- Spelling correction: automatically correct spelling mistakes
- Many more...

Language Modeling

Prediction

Humans are incredibly good at predicting:

- Once upon a ?
- And the haters gonna hate, Baby, I’m just gonna?
- Don’t stop me know, I’m having ?
- Shall I compare thee to ?

Prediction

Humans are incredibly good at predicting:

- Once upon a **time**
- And the haters gonna hate, Baby, I'm just gonna **shake**
- Don't stop me know, I'm having **such a good time**
- Shall I compare thee to **a summer's day**

What comes before "computing"?
Grid computing 207011
parallel computing 101732
performance computing 229510
etc.
We can predict the next word given its history using language models. Source: `http://norvig.com/ngrams/count_2w.txt`

Language Modeling

Specify a language model that learns from examples rather than specifying the rules of a language using formal grammar.

Language Model

Models that assign probabilities to sequences of words are called language models: $P(w_1, w_2, w_3, ..., w_n)$

Useful in real-world applications, for example:

- machine translation
  $P(I\ didn't\ do\ anything) > P(I\ didn't\ do\ nothing)$
- speech recognition
  $P(I\ ramble) > P(I\ Rambo)$
- spelling correction
  $P(Please\ pay\ before\ exiting) > P(Please\ pai\ before\ existing)$

Traditional Language Models

Probability is usually conditioned on a window of n previous words:

- We can calculate the probability of a sentence by calculating the joint probability of each element in the sentence:
  $P(S) = P(w_1, w_2, ...w_n)$
- Chain rule: Any member of a joint distribution of random variables can be calculated using conditional probabilities:
  $P(S) = P(w_1), P(w_2|w_1)P(w_3|w_1, w_2)...P(w_n|w_1, ..., w_{n-1})$
- Markov assumption: only the last n words are considered in the history and can be utilized to approximate the probability
  $P(w_1, ..., w_m) \approx \prod_{i=1}^{m} P(w_i|w_i - (n-1), ..., w_i)$

N-Gram Models

The simplest type of language model is the N-gram model. The N specifies the number of swords in a sequence: 2-gram (bigrams), 3-gram (trigrams), etc.

- to estimate the probabilities for **unigrams** (probabilities only depend on the probability of the word): $p(w_1) = \frac{count(w_1)}{\sum_w count(w)}$
- to estimate the probabilities for **bigrams** (conditioning on one previous word): $p(w_2|w_1) = \frac{count(w_1,w_2)}{count(w_1)}$
- to estimate the probabilities for **trigrams** (conditioning on two previous words): $p(w_3|w_1,w_2) = \frac{count(w_1,w_2,w_3)}{count(w_1,w_2)}$

This is why those models are usually today referred to as **count-based models**.

Example

<s>I live in Dresden</s>
<s>Dresden is a city</s>
<s>I do not like pigeons in the city</s>

- Unigram? $P(live) = \frac{1}{22} = 0.04$
- Bigram? $P(Dresden| < s >) = \frac{1}{3} = 0.33$
- Trigram? $P(Dresden|live \quad in) = \frac{1}{2} = 0.5$

TECHNISCHE UNIVERSITÄT DRESDEN

In practice

- Trigrams are more common than bigrams
- Log probabilities are used to avoid underflow (the more probabilities we multiply, the smaller the product)
- Model based on frequency counts only do not perform well on unseen items. Instead:
  - **back-off** (e.g. if 4-gram not found, use 3-gram, etc.)
  - **Laplace smoothing** (add-one: $p(w_2|w_1) = \frac{count(w_1,w_2)+1}{count(w_1)+Vocab}$)
- Computation: Recent example of a Kneser-Ney language model training was 140 GB Ram in 2.8 days for one model of 128 billion tokens

Bigram Model in Python

```
from nltk.corpus import reuters
from nltk import bigrams
from collections import Counter, defaultdict

first_sentence = reuters.sents()[0]
print(first_sentence)
#Output: ['ASIAN', 'EXPORTERS', 'FEAR', 'DAMAGE', 'FROM', 'U', '.', 'S', ...]
print(list(bigrams(first_sentence, pad_left=True, pad_right=True)))
#Output: [(None, 'ASIAN'), ('ASIAN', 'EXPORTERS'), ('EXPORTERS', 'FEAR'), ]

model = defaultdict(lambda: defaultdict(lambda : 0))

#Generate a dictionary of counts
for sentence in reuters.sents():
    for w1, w2 in bigrams(sentence, pad_right=True, pad_left=True):
        model[w1][w2] += 1

print(model["the"]["economists"])
# Output: "economist" follows "the" 8 times
print("Example why padding is useful", model[None]["The"])
# Output: "The" starts a sentence 8839 times
```

Bigram Model in Python - continued

```
#Transform counts into probabilities
for w1 in model:
    total_count = float(sum(model[w1].values()))
    for w2 in model[w1]:
        model[w1][w2] /= total_count


print(model["the"]["economists"]) #0.00013733669808243634
print(model[None]["The"]) #0.16154324146501936
```

Evaluation

Main two evaluation methods for most computational linguistic models:

- **Extrinsic evaluation**: measure how much a specific application improves by using your model as compared to the standard baseline (time-consuming!)
- **Intrinsic evaluation**: measure the quality of the model independent of any application

For the intrinsic evaluation, the corpus is split into a:

- Training set: data used to train the model
- Test set: data used to test the trained model using a specific accuracy measure

The model that more accurately predicts the test set is the better model.

Review of Lecture 3

- What is Named Entity Recognition?
- Which two processes are needed for relation extraction?
- What is sentiment analysis?
- What is the difference between emotion classification and polarity detection?
- What is a language model?
- How can the chain rule and the Markov assumption be used in a language model? What are they?
- What happens when we want to compute a bigram that a model has not seen before?
- How can a language model be evaluated?