Introduction to Natural Language Processing

Vitalii Duk

Agenda

- NLP tasks
- Basics of Computational Linguistics
- Raw text pre-processing
- Text classification
- Topic modelling and text clustering
- Vector representation of words
- Python examples

Major NLP tasks

- Machine translation
- Sentiment analysis
- Text summarization
- Topic segmentation
- Named Entity Recognition
- Text-to-speech

NLP problematics

- Context-free grammar
- Different linguistic typologies (subject, object, verb): *SOV, SVO, VSO, VOS, others*
- Different writing systems: *Arabic, Latin, Cyrillic, Chinese, others*

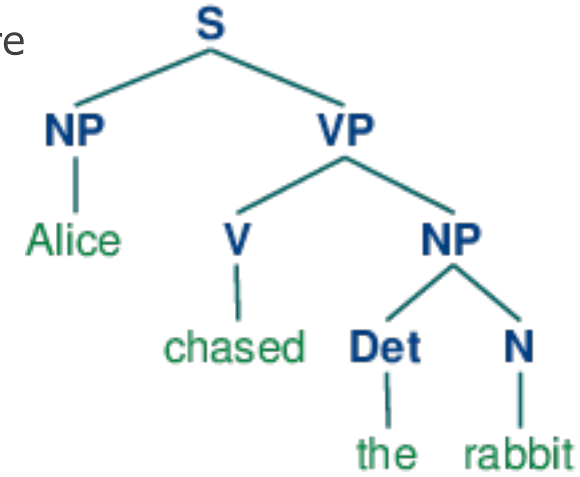
Word order	Example in English	Languages
SOV	She him loves.	*Sanskrit, Hindi, Ancient Greek, Latin, Japanese*
SVO	She loves him.	*English, French, Indonesian, Malay, Mandarin, Russian*
VSO	Loves she him.	*Arabic, Irish, Filipino, Welsh*

Corpora

- **Brown Corpus**: *500* examples, *15* genres, *2000* words with POS tags.
- **Quranic Arabic Corpus**: *77 430* words present in Quran with morphological and syntactic annotations.
- **WordNet**: *155 287* words organized into *117 659* synsets and *206 941* word-sense pairs.

Treebank

- text corpus with annotated syntactic and semantic structure
- based on top of corpus with annotated POS tags
- in practice usually used to determine relations between objects in a sentence
- more than 20 treebanks for majority of languages
  wikipedia.org/wiki/Treebank



Pre-processing: stop words

- most common words in a language
- bring a little value to the sense of a text
- there is no universal list of stop words
- typical examples: ***a, the, of, it, as, in, at***

Pre-processing: tokenization

- **Word tokenization** – split raw text into a set of words, typically using white space.
- **Sentence tokenization** – split raw text in to a set of sentences, typically using period.

Typical problems:

- word tokenization: $\text{Let's visit New } York. \rightarrow \; [Let's,\ visit,\ New,\ York]$
- sentence tokenization: $\text{It } was\ Mr.Holmes. \rightarrow \; ['It\ was\ Mr.'],\ ['Holmes.']$

Pre-processing: stemming & lemmatization

- **Stem** – part of the word that never changes even when morphologically inflected.

$$swimming \rightarrow swim$$
$$university \rightarrow univers$$

- **Lemma** – the base form of the word.

$$went \rightarrow go$$

- Python NLTK stemmers: *PorterStemmer, SnowballStemmer*
- Python NLTK lemmatization: *WordNetLemmatizer*

Pre-processing: n-grams

- Example: *It was raining in Dubai yesterday*.
- 1-grams or **unigrams**:

$$[(It,),(was,),(raining,),(in,),(Dubai,),(yesterday,)]$$

- 2-grams or **bigrams**:

$$[(It,was),(was,raining),(raining,in),(in,Dubai),(Dubai,yesterday)]$$

Part-of-speech tagging

- **Verb** - show an action or a state of being: *go, write, exist, be*
- **Noun** - refer to people, animals, objects, states, events: *John, lion, table, freedom, love*
- **Adjective** - used to describe or specify a noun or pronoun: *good, beautiful, nice, my*
- **Adverb** - used to modify a verb, adjective and other adverbs: *completely, never, there*
- Others: **Pronoun**, **Preposition**, **Conjunction**, **Interjection**.

Part-of-speech tagging workflow

- Get manually annotated corpus with POS tags for each word.
- Derive features which will be used to predict POS tag for word $\boldsymbol{\omega_i}$:
  - previous *k* words: $\boldsymbol{\omega_{i-1}} \; \dots \; \boldsymbol{\omega_{i-k}}$
  - POS tags of previous *k* words: $\boldsymbol{t_{i-1}} \; \dots \; \boldsymbol{t_{i-k}}$
  - next *k* words: $\boldsymbol{\omega_{i+1}} \; \dots \; \boldsymbol{\omega_{i+k}}$
  - POS tags of previous *k* words: $\boldsymbol{t_{i+1}} \; \dots \; \boldsymbol{t_{i+k}}$
- Use one of the classification algorithms to train mode:
  Hidden Markov Models, Neural Network, etc.

Bag-of-words

- **Main idea:** use word frequencies as features to classify text.
- Go through *N* documents in our dataset and build a dictionary of words used in a dataset
- Build matrix of frequencies with size *N* x *M*
- Run classifier using frequencies matrix

Bag-of-words: matrix example

1. Place was really good.
2. We had a good time.
3. Spent a great time there.

$$X\begin{pmatrix} place & was & really & good & we & had & time & spent & great & there \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

tf-idf

- term frequency–inverse document frequency
- one of the most popular term-weighting algorithms
- increases proportionally along with the word frequency
- adjusted to reduce importance of frequent words in general

$$tfidf(t, d, D) = f_{t,d} \cdot \log \frac{N}{n_t}$$

Bayes theorem

$$P(B \mid A) = \frac{P(A \mid B) \cdot P(A)}{P(B)}$$

- $P(A)$ and $P(B)$ are the probabilities to observe events A and B in our overall data
- $P(A \mid B)$ is a probability of observing event A given the fact that B is true
- $P(B \mid A)$ is a probability of observing event B given that A is true

Naïve Bayes

- assume independence between predictors
- suitable for a large datasets

Likelihood

Class prior probability

$$P(Positive, [good = 1, bad = 0]) = \frac{P([good = 1, bad = 0], Positive) \cdot P(Positive)}{P([good = 1, bad = 0])}$$

Posterior probability

Evidence prior probability

Latent Dirichlet allocation

- unsupervised statistical model
- represents documents as mixtures of topics
- each topic is represented by a set of weighted words

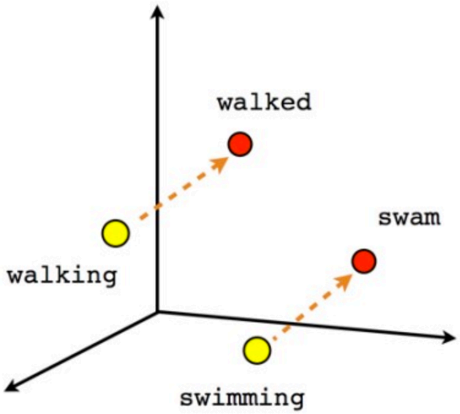
Topic 1		Topic 2		Topic 3	
word	**weight**	**word**	**weight**	**word**	**weight**
politics	3245	university	5443	environment	4554
government	2334	education	4435	pollution	3442
affairs	1545	degree	4322	ecosystem	1245

Latent Dirichlet allocation

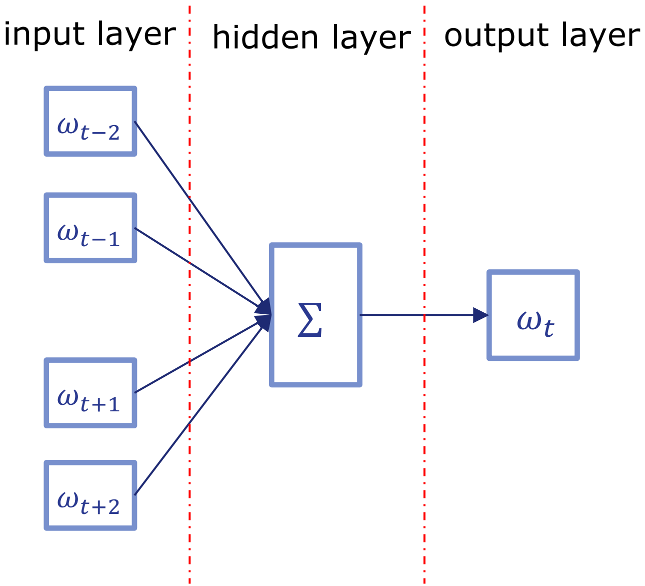
- Assume that documents are produced from a mixture of topics. Topics generate words based on their probability distribution.
- Determine the number of words in a document. Let’s say our document has 50 words.
- Determine the mixture of topics in that document. For example, the document might contain 1/2 the topic ***education*** and 1/2 the topic ***politics***.
- Using each topic’s multinomial distribution, output words to fill the document’s word slots.

Vector representation of words

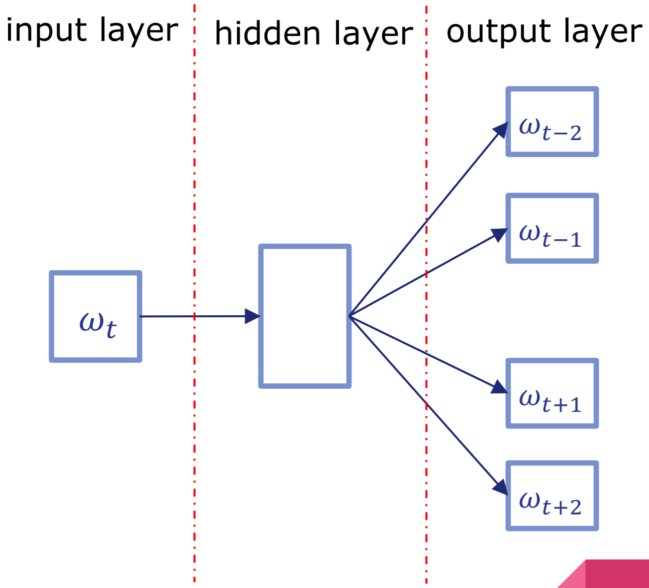
- build a vector for each word taking in account context in which particular word occurs
- don’t require labeled data
- simple feedforward neural network under the hood
- popular implementations: **word2vec, GloVe**



Vector representation of words



CBOW
Continuous Bag-of-Words



skip-gram

Common NLP tools

- NLTK – nltk.org
- gensim - radimrehurek.com/gensim
- SpaCy - spacy.io
- OpenNLP - opennlp.apache.org
- Stanford CoreNLP - stanfordnlp.github.io/CoreNLP

Python examples

github.com/root-ua/nlp-intro-meetup

Thanks!