Introduction to Natural Language Processing

Lecture 3. POS tagging. Key word and phrase extraction.

Ekaterina Chernyak, Dmitry Ilvovsky

echernyak@hse.ru, dilvovsky@hse.ru

National Research University
Higher School of Economics (Moscow)

July 16, 2015

Part of speech (POS)

Part of speech [Manning, Shuetze, 1999]

Words of a language are grouped into classes which show similar syntactic behavior. These word classes are called parts of speech (POS). Three important parts of speech are noun, verb, and adjective. The major types of morphological process are inflection, derivation, and compounding.

There are around 9 POS according to different schools:

- Nouns (NN, NP), pronouns (PN, PRP), adjectives (JJ): number, gender, case
- Adjective (JJ): comparative, superlative, short form
- Verbs (VB): subject number, subject person, tense, aspect, modality, participles, voice
- Adverbs (RB), prepositions (IN), conjunctions (, CS), articles (AT) and particles (RP): nothing

POS ambiguation

Ship (noun or verb?)

- a luxury cruise *ship*
- Both products are due to *ship* at the beginning of June
- A new engine was *shipped* over from the US
- The port is closed to all **shipping**

Contest (noun or verb?)

- Stone decided to hold a contest to see who could write the best song.
- She plans to contest a seat in Congress next year.

POS taggers

- Corpus- or dictionary-based VS rule-based
- Ngram-based taggers:
  - unigram tagging: assign the most frequent tag
  - ngram tagging: look at the context of *n* previous words (requires a lot of training data)
- Trade-off between the accuracy and the coverage: combine different taggers

NLTK POS default tagger

```
In[1]:  from nltk.tag import pos_tag
In[2]:  print pos_tag(['ship'])
Out[1]:  [('ship', 'NN')]
In[3]:  print pos_tag(['shipping'])
Out[2]:  [('shipping', 'VBG')]
```

See `http://www.nltk.org/api/nltk.tag.html` for more details on learning taggers.

Exercise 3.1 Genre comparison

Text genre [Santini, Sharoff, 2009]

The concept of genre is hard to agree upon. Many interpretations have been proposed since Aristotles Poetics without reaching any definite conclusions about the inventory or even principles for classifying documents into genres. The lack of an agreed definition of what genre is causes the problem of the loose boundaries between the term "genre" with other neighbouring terms, such as "register", "domain", "topic", and "style".

Exercise 3.1

Input: Two texts of different genre (for example, Wikipedia article and blog post)
Output: rank all of POS tags for both texts

How can you describe the difference between two genres?

Key word and phrase extraction

There are many definitions of key word and phrase. Thus there are many methods for their extraction:

- supervised VS unsupervised
- frequency-based VS more complex
- from individual text VS from text collection
- word (unigram) VS bigram VS ngram
- term VS named entity VS collocation
- sequential words VS using window

Supervised methods for key word and phrase extraction

I am a word. Am I a key word? Let us build a classifier.

- Am I in the beginning or in the end of the sentence?
- Am I capitalized?
- How many times do I occur?
- Am I used in Wikipedia as a title of a category or an article?
- Am I a term?
- Am I a NE?
- etc.

But we need a collection of marked up texts!

Unsupervised methods for key word and phrase extraction from a single text

- POS patterns
- Association measures: PMI, T-Score, LLR
- Graph methods: TextRank [Mihalcea, Tarau, 2004]
- Syntactic patterns

Exercise 3.2

Input: sif1.txt (or your own text)
Key word: top n_1 NN
Key phrase: top n_2 phrases, that satisfy the following patterns: JJ + NN, NN + NN, NN + IN + NN
Output: list of key words and phrases

Hint: use `nltk.ngrams` to get ngrams.

Bigram association measures (1)

w_1, w_2 - two words
$f(w_1), f(w_2)$ – their individual frequency
$f(w_1, w_2)$ – their co-occurrence frequency
$PMI(w_1, w_2) = \log \frac{f(w_1,w_2)}{f(w_1)f(w_2)}$

Pointwise Mutual Information [Manning, Shuetze, 1999]

PMI measures the reduction of uncertainty about the occurrence of one word when we are told about the occurrence of the other one.

Bigram association measures (2)

w_1, w_2 - two words
$f(w_1), f(w_2)$ – their individual frequency
$f(w_1, w_2)$ – their co-occurrence frequency
N – number of words
$T - score(w_1, w_2) = \frac{f(w_1,w_2) - f(w_1) * f(w_2)}{f(w_1,w_2)/N}$

T-Score [Manning, Shuetze, 1999]

T-Score is a statistical t-test applied to finding collocations. The t-test looks at the difference between the observed and expected means, scaled by the variance of the data. The T-score is most useful as a method for ranking collocations. The level of significance itself is less useful.

Bigram association measures (3.1)

w_1, w_2 - two words
$O_{11} = f(w_1, w_2)$ – **observed** number of bigram (w_1, w_2) occurrences
$E_{11} = \frac{O_{11}+O_{12}}{N} \times \frac{O_{11}+O_{21}}{N} \times N$ – **expected** number of bigram (w_1, w_2) occurrences

	w_2	not w_2
w_1	O_{11}	O_{12}
not w_1	O_{21}	O_{22}

Table : 2-by-2 occurance table

$$chi^2 = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} = |i \in 1,2, j \in 1,2| =$$

$$= \frac{N(O_{11}O_{22} - O_{12}O_{21})^2}{(O_{11} + O_{12})(O_{11} + O_{21})(O_{12} + O_{22})(O_{21} + O_{22})}$$

Bigram association measures (3.2)

But χ^2 has many other interesting applications.

χ^2 [Manning, Shuetze, 1999]

In general, for the problem of finding collocation, the difference between the T-score and the χ^2 does not seem to be large.

Bigram association measures in NLTK

NLTK BigramCollocationFinder

```
In[1]:  from nltk.collocations import *
In[2]:  bigram_measures =
nltk.collocations.BigramAssocMeasures()
In[3]:  finder = BigramCollocationFinder.from_words(tokens)
In[4]:  finder.apply_freq_filter(3)
In[5]:  for i in finder.nbest(bigram_measures.pmi, 20):
...
```

Bigram measures:

- `bigram_measures.pmi`
- `bigram_measures.student_t`
- `bigram_measures.chi_sq`
- `igram_measures.likelihood_ratio`

See `http://www.nltk.org/_modules/nltk/metrics/association.html` for more more bigram association measures.

TextRank: using graph centrality measures for key word and phrase extraction (1) [Mihalcea, Tarau, 2004]

1. Add words as vertices in the graph.
2. Identify relations that connect words:
   - consequent words;
   - words inside (left or right) the window (2-5 words);

   and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.
3. Iterate the graph-based ranking algorithm until convergence (for example, PageRank).
4. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.
5. If two adjacent words are selected as potential keywords by TextRank, collapse them into one single key phrase.

See original paper: `http://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf`

TextRank (2)

Compatibility of systems of linear constraints over the set of natural numbers. Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types systems and systems of mixed types.

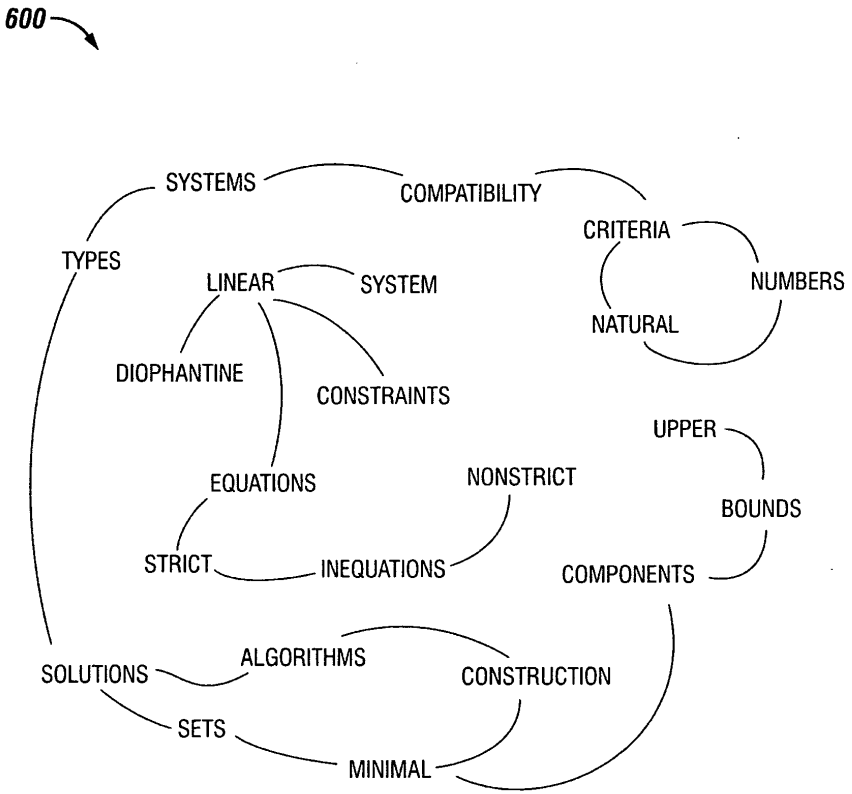

FIG. 6

TextRank (3)

$G = (V, E)$ – a graph, V – vertices, E – edges
$In(V_i)$ – the set of vertices that point to it
$Out(V_i)$ – the set of vertices that V_i points to
Graph centrality measures:

PageRank [Brin, Page, 1998]

$$PR(V_i) = (1 - d) + d \times \sum_{V_j \in In(V_i)} \frac{PR(V_j)}{|Out(V_j)|}$$

HITS [Kleinberg, 1999]

$$HITS_A(V_i) = \sum_{V_j \in In(V_i)} HITS_H(V_j) \; HITS_H(V_i) = \sum_{V_j \in Out(V_i)} HITS_A(V_j)$$

TextRank (4)

Key words and phrases, assigned by TextRank using PageRank centrality measure:
linear constraints; linear diophantine equations; natural numbers; nonstrict inequations; strict inequations; upper bounds

Exercise 3.3

Input: sif.txt (or your own text)
Output: key words and phrases computed by PageRank (using $PR(V_i)$, $HITS_A(V_i)$ and $HITS_H(V_i)$ as centrality measures)

Hint: use NetworkX for PageRank and HITS (`http://networkx.github.io/documentation/networkx-1.9.1/reference/algorithms.link_analysis.html`)

Unsupervised methods for key word and phrase selection from a text in a collection

The problem: given a collection of texts find those words and phrases (terms) that occur in this text significant frequently than in other texts.

Term frequency [Luhn, 1957]

The weight of a term that occurs in a document is simply proportional to the term frequency.

Inverse document frequency [Spaerck Jones, 1972]

The specificity of a term can be quantified as an inverse function of the number of documents in which it occurs.

$$tfidf(term, text, collection) = tf(term, document) \times idf(term, collection)$$

Variants of TF and IDF weights

$tf(term, document) =$

- binary: 1 *if* $term \in document$, 0 *otherwise*
- raw frequency: $f_{t,d}$
- log normalization: $log(1 + f_{t,d})$

$idf(term, document) =$

- unary: 1
- inverse frequency: $\log \frac{N}{n_t}$, where N – total number of texts in the collection, n_t – number of texts, that contain term t
- smoothed inverse frequency: $\log \frac{N}{n_t+1}$

The most common scheme: $f_{t,d} \times \log \frac{N}{n_t+1}$

TF-IDF in NLTK

NLTK TextCollection class

```
In[1]:  from nltk.text import TextCollection
In[2]:  collection = [WhitespaceTokenizer().tokenize(text)
for text in collection]
In[3]:  corpus = TextCollection(collection)
In[4]:  for i in collection[0]:  print i, corpus.tf_idf(i,
collection[0])
```

Exercise 3.4

Input: sif2.txt (or your own collection of texts)
Output: list of key words and phrases according to TF-IDF for one text

Hint: use `sorted(mylist,key=lambda l:l[1], reverse=True)` to sort `mylist` in descending order based on the second parameter

TF-IDF alternatives

Exercise 3.5

Write MI and χ^2 as an alternative for TF-IDF for measuring significance of a term in a text in a collection.

Check your ideas in [Sebastiani, 2001]
(`http://arxiv.org/pdf/cs/0110053.pdf`)

Using TF-IDF to measure text similarity

Let the text d_i in a collection of documents be represented as a vector: $d_i = (w_1^i, w_2^i, ..., w_{|V|}^i)$, where V is a vocabulary and $|V|$ is the number of types (terms). $|V|$ is also the number of dimensions, where every type (term) corresponds to its own direction. w_k^i is the weight of the type k in text i (usually – TF-IDF weight).
The similarity of two texts is defined as a cosine between corresponding vectors:

Cosine similarity [Salton et. al, 1975]

$$cos(d_i, d_j) = \frac{d_i \times d_j}{||d_i||||d_j||} = \frac{\sum_k w_k^i \times w_k^j}{\sqrt{(\sum_k w_k^i)^2}\sqrt{(\sum_k w_k^j)^2}}$$

This is an algebraic model for representing collections of texts, called **Vector Space Model** [Salton et. al, 1975].

References

1. Manning, Christopher D., and Hinrich Schuetze. Foundations of statistical natural language processing. MIT press, 1999.
2. Santini, Marina, and Serge Sharoff. Web Genre Benchmark Under Construction. JLCL, Volume 24 (1), 2009.
3. Luhn, Hans Peter. "A statistical approach to mechanized encoding and searching of literary information." IBM Journal of research and development 1.4 (1957): 309-317.
4. Sparck Jones, Karen. "A statistical interpretation of term specificity and its application in retrieval." Journal of documentation 28.1 (1972): 11-21.
5. Sebastiani, Fabrizio. "Machine learning in automated text categorization." ACM computing surveys (CSUR) 34.1 (2002): 1-47.
6. Salton, Gerard, Anita Wong, and Chung-Shu Yang. "A vector space model for automatic indexing." Communications of the ACM 18.11 (1975): 613-620.
7. Bird, Steven, Ewan Klein, and Edward Loper. Natural language processing with Python. " O'Reilly Media, Inc.", 2009.