Further Topics in Text Mining

Konstantin Todorov

todorov at lirmm dot fr

University of Montpellier

March 2019

Outline

Outline

Outline

Introduction

What is part of speech tagging?

POS = Part of speech

The process of assigning automatically a part of speech label to each word in a given sentence.

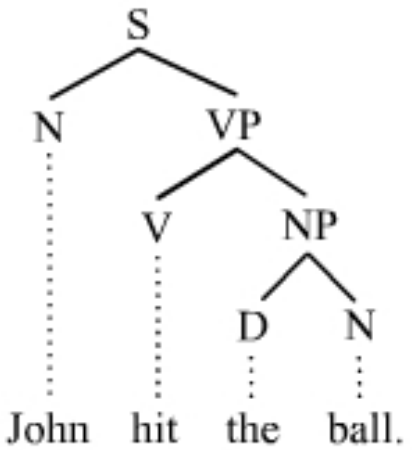
Example:

Heat water in a large vessel. –>

Heat (verb) water (noun) in (preposition) a (determiner) large (adjective) vessel (noun).

Applications

- Information retrieval
- Text classification (e.g., opinion mining)
- Word sens disambiguation
- Parsing



Constituency-based parse tree

Source: Wikipedia.

Introduction

POS Tags

Standardized POS tags, e.g., Penn Treebank POS tags:

POS Tag	Description	Example
CC	coordinating conjunction	and
CD	cardinal number	1, third
DT	determiner	the
EX	existential there	*there* is
FW	foreign word	d'hoevre
IN	preposition/subordinating conjunction	in, of, like
JJ	adjective	big
JJR	adjective, comparative	bigger
JJS	adjective, superlative	biggest
LS	list marker	1)
MD	modal	could, will
NN	noun, singular or mass	door
NNS	noun plural	doors
NNP	proper noun, singular	John
NNPS	proper noun, plural	Vikings
PDT	predeterminer	*both* the boys
POS	possessive ending	friend*'s*
PRP	personal pronoun	I, he, it

Source: http://www.monlp.com/2011/11/08/part-of-speech-tags/

Choice of a tag set:

- a tag per part of speech
- 5 basic tags: adj, noun, verb, adv, prep
- Penn Treebank has 36 tags
- ...

Be consistent in the use of labels.

According to Penn's tagging, our example from the start looks like that:

- Heat /VB water /NN in /IN a /DT large /JJ vessel /NN.

Introduction

What makes tagging difficult?

Any ideas?

Can’t we just label all words once and for all?

How about ambiguity?

Example:
"Scream like a savage."

- scream: verb, noun?
- like: adverb, verb...?
- savage: adjective, noun?

We want to tag words automatically.

...So, what kind of information we have to consider?

Outline

Probabilistic Tagging

Estimate the probability that a tag occurs, given
a word, a set of previously known tags, a neighborhood of words...

Probability estimation is often based on frequency counts.

- Remember estimating probabiliy of a class in Bayes Classification?

To count frequencies we need data.

- a training corpus of tagged sentences.
- Brown Corpus: about 1 million words and a set of POS tags assigned to each word

Probabilistic Tagging

A simple approach

Let the word w has a set of tags $\{t_1, t_2, ..., t_k\}$.

- Estimate $P(t_i|w), \forall i = 1, ..., k$
- From training data: $P(t_i|w) = \frac{N(w,t_i)}{N(w,t_1)+N(w,t_2)+...+N(w,t_k)}$
- — The most popular wins.

Works in more than 80 percent of the cases.
Example: heat/NN (89%), heat/VB (5%) –> but is heat in our sentence a noun?

"Heat water in a large vessel."

Probabilistic Tagging

Bayesian Approach

Now let's look at the whole sentence, $W = w_1, w_2, \ldots, w_n$.

- Example: W = heat, water, in, a, large, vessel

Goal: predict a tag sequence $T = t_1, \ldots, t_n$ for W

- Look for the sequence T that maximizes $P(T|W)$.
- Estimations come from training data, where every word w_i has a set of tags $\{t_1, t_2, \ldots, t_k\}$.

Probabilistic Tagging

Bayesian Approach

Bayes theorem tells us that $P(T|W) = P(W|T)P(T)/P(W)$.

The T that maximizes $P(W|T)P(T)/P(W)$, also maximizes $P(W|T)P(T)$. We want to find that T.

Let's look at $P(T)$ first.

- $P(T) = P(t_1)P(t_2|t_1)P(t_3|t_2,t_1)...P(t_n|t_{n-1},...,t_2,t_1)$
- In approximation, $P(T) \approx P(t_1)P(t_2|t_1)P(t_3|t_2)...P(t_n|t_{n-1})$

Now let's look at $P(W|T)$.

- Assume a word depends on its own POS tag only and not on the other words or their POS tags.
- $P(W|T) = P(w_1|t_1)P(w_2|t_2)...P(w_n|t_n)$

We have $P(W|T)P(T) \approx P(w_1|t_1)P(w_2|t_2)...P(w_n|t_n)P(t_1)P(t_2|t_1)P(t_3|t_2)...P(t_n|t_{n-1})$

Probabilistic Tagging

Bayesian Approach

Bayes theorem tells us that $P(T|W) = P(W|T)P(T)/P(W)$.

The T that maximizes $P(W|T)P(T)/P(W)$, also maximizes $P(W|T)P(T)$. We want to find that T.

Let's look at $P(T)$ first.

- $P(T) = P(t_1)P(t_2|t_1)P(t_3|t_2,t_1)...P(t_n|t_{n-1},...,t_2,t_1)$
- In approximation, $P(T) \approx P(t_1)P(t_2|t_1)P(t_3|t_2)...P(t_n|t_{n-1})$

Now let's look at $P(W|T)$.

- Assume a word depends on its own POS tag only and not on the other words or their POS tags.
- $P(W|T) = P(w_1|t_1)P(w_2|t_2)...P(w_n|t_n)$

We have $P(W|T)P(T) \approx P(w_1|t_1)P(w_2|t_2)...P(w_n|t_n)P(t_1)P(t_2|t_1)P(t_3|t_2)...P(t_n|t_{n-1})$

Probabilistic Tagging

Bayesian Approach

Bayes theorem tells us that $P(T|W) = P(W|T)P(T)/P(W)$.

The T that maximizes $P(W|T)P(T)/P(W)$, also maximizes $P(W|T)P(T)$. We want to find that T.

Let's look at $P(T)$ first.

- $P(T) = P(t_1)P(t_2|t_1)P(t_3|t_2,t_1)...P(t_n|t_{n-1},...,t_2,t_1)$
- In approximation, $P(T) \approx P(t_1)P(t_2|t_1)P(t_3|t_2)...P(t_n|t_{n-1})$

Now let's look at $P(W|T)$.

- Assume a word depends on its own POS tag only and not on the other words or their POS tags.
- $P(W|T) = P(w_1|t_1)P(w_2|t_2)...P(w_n|t_n)$

We have $P(W|T)P(T) \approx P(w_1|t_1)P(w_2|t_2)...P(w_n|t_n)P(t_1)P(t_2|t_1)P(t_3|t_2)...P(t_n|t_{n-1})$

Probabilistic Tagging

Bayesian Approach

To compute
$P(W|T)P(T) \approx P(w_1|t_1)P(w_2|t_2)...P(w_n|t_n)P(t_1)P(t_2|t_1)P(t_3|t_2)...P(t_n|t_{n-1})$

We estimate from the training corpus

- $P(w_i|t_i) \approx \frac{N(w_i,t_i)}{N(t_i)}$
- $P(t_i|t_{i-1}) \approx \frac{N(t_{i-1},t_i)}{N(t_{i-1})}$,

where $N(t_i)$ counts the number of appearances of t_i in the corpus,
$N(w_i, t_i)$ – the number of appearances of the couple (w_i, t_i),
and $N(t_{i-1}, t_i)$ – the frequency of the tag sequence (t_{i-1}, t_i).

Outline

TreeTagger

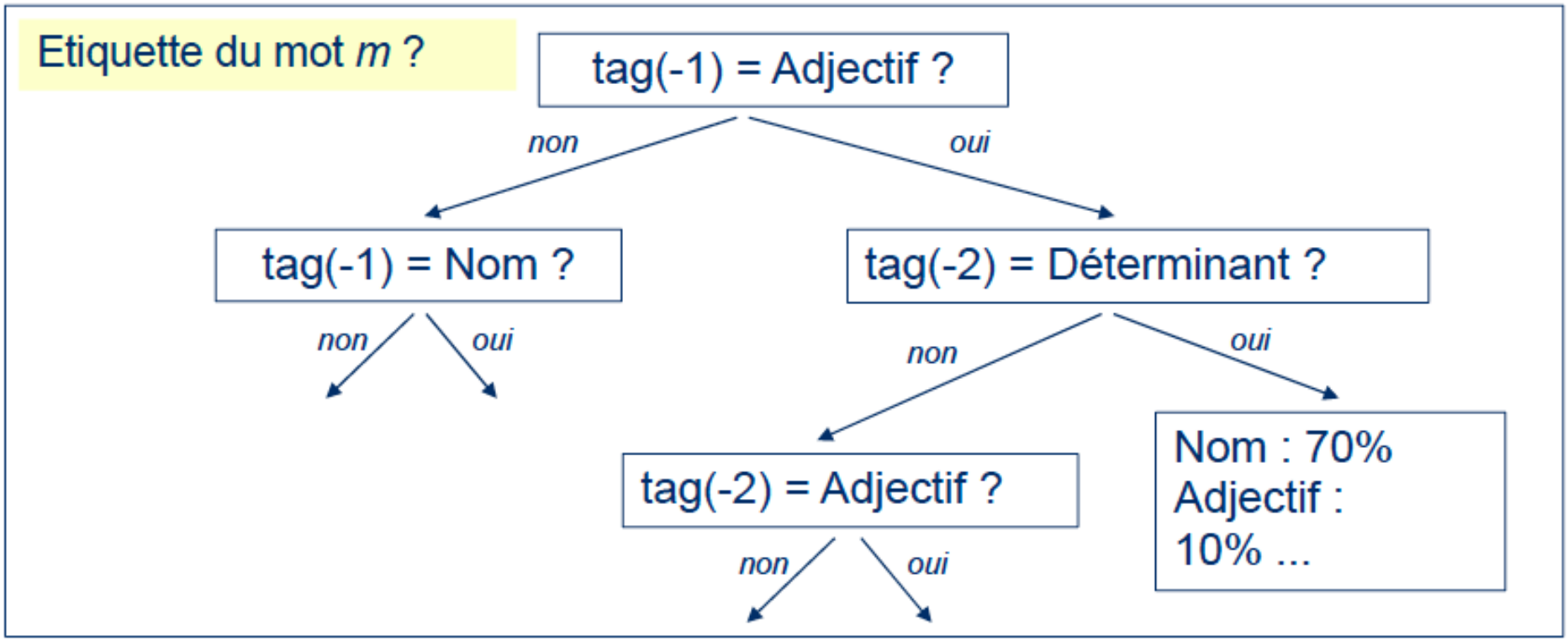
Introduced by H. Schmid in the 90s[1].

Available here: http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/

- A probabilistic tagger
- Use binary decision tree to estimate the POS tag of a word.
- Construct trees recursively by starting with a set of known tags.
- Usually, one needs to know three preceding consecutive tags - the context.
- Intuition: closer tags provide more information

[1] http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/data/tree-tagger2.pdf

TreeTagger



- $P(tag_m = Nom \mid tag(-2)=déterminant, tag(-1) = Adjectif) = 70\%$
- $P(tag_m = Adjectif \mid tag(-2)=déterminant, tag(-1) = Adjectif) = 10\%$

Source: a course by M. Roche.

Probabilities are estimated from a training set.

Outline

N-grams

An n-gram is a contiguous sequence of n items from a given sequence of text or speech: uni-grams, bi-grams, 4-grams, etc.

An n-gram model is a type of probabilistic language model for predicting the next item in a sequence of items in the form of a (n-1)-order Markov model.

Given a sequence of letters, what is the likelihood of the next letter?
— Use conditional probabilities and estimate them from corpora.
— "For ex" –> likelihood of "ample"?

Independence assumptions are made so that each word depends only on the last n-1 words.

N-grams

Example

The cow jumps over the moon.

- Bi-grams: the cow, cow jumps, jumps over, over the, the moon
- Tr-grams: the cow jumps, cow jumps over, jumps over the, over the moon.
- Uni-gram model – singletons containing one word.

Several applications:

- Indexing large corpora (the web)
- Creating a feature model for machine learning (SVM, Naive Bayes, ...)
- Similarity measures on documents

Outline

Feature Selection

Feature selection (aka "Variable selection"): summarization of text content for better indexing!

- High dimensional feature space (high number of unique terms)
- —> a problem for most of the learning algorithms
- Automatic feature selection: removal of uninformative terms / construction of new terms as a combination of existing ones

Document frequency thresholding

- a simple feature selection technique
- based on the document frequency for each term in the training dataset

Two ways to go:

- terms with very rare occurrences are considered as unimportant for the regrouping of documents into categories (noise)
- terms with very frequent occurrences are considered as unimportant for the regrouping of documents into categories (stop-words).

Mutual information

One considers the co-occurrences of a term and a category. Let t be a term and c – a category and let A be the number of times t and c co-occur, B – the number of times t occurs alone, C – the number of times c occurs alone and m – the total number of documents in the dataset. The mutual information criterion is estimated by

$$I(t,c) = log\frac{A \times m}{(A+C) \times (A+B)}. \quad (1)$$

χ^2 criterion

A related approach uses the χ^2 **- statistics** by testing the lack of independence between t and c. The criterion is estimated by

$$\chi^2(t,c) = \frac{m \times (AN - CB)^2}{(A+C) \times (B+N) \times (A+B) \times (C+N)},$$

where N is the number of times neither t, nor c occurs.

Term strength

Assumptions:

1. Documents with many shared words are assumed to be related.
2. Terms in the overlapping area of related documents are assumed to be important.

Let d_1 and d_2 be two documents – distinct, but assumed to be highly similar, and let t be a term. The strength of t is given by

$$s(t) = P(t \in d_1 | t \in d_2).$$

—> A criterion based on the conditional probability that a term appears in a document, given that it appears in another, related document. Estimated from training data of similar documents and frequency counts.

A take away message...

Preparing data (choosing the right descriptors, or features) for the machine learning task in view can be 80 percent of the job!

Sources and further reading

This course uses references and examples from the following tutorial
http://www.cs.umd.edu/ nau/cmsc421/part-of-speech-tagging.pdf,

as well as from a course by Mathieu Roche:
http://agents.cirad.fr/index.php/Mathieu+ROCHE.

Here's a link to the paper of H. Schmid on TreeTagger:
http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/data/tree-tagger2.pdf
and a link to a nice paper on feature selection:
https://pdfs.semanticscholar.org/c3eb/cef26c22a373b6f26a67934213eb0582804e.pdf