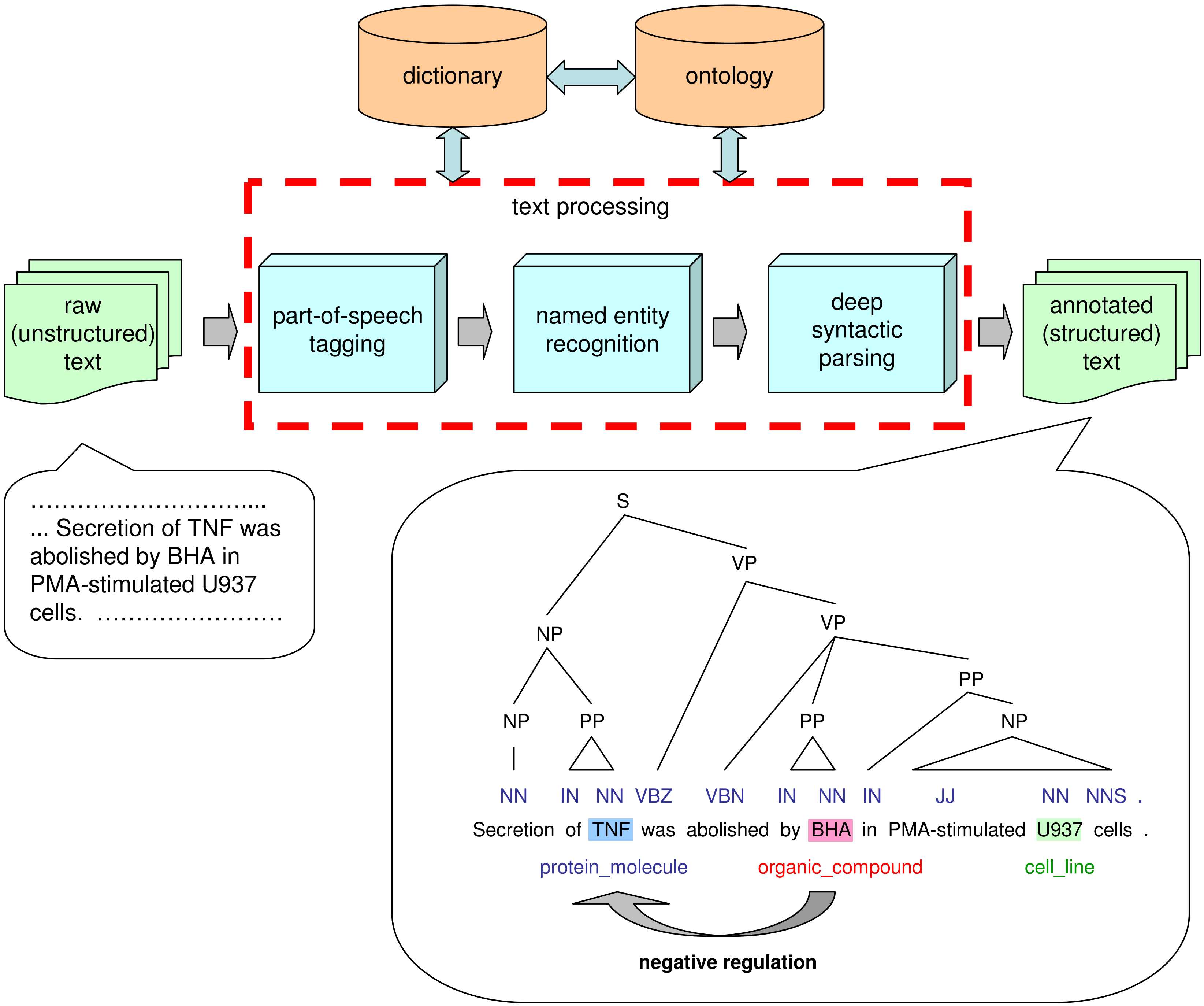
Sentence splitting, tokenization, language modeling, and part-of-speech tagging with hidden Markov models

Yoshimasa Tsuruoka

Topics

- Sentence splitting
- Tokenization
- Maximum likelihood estimation (MLE)
- Language models
  - Unigram
  - Bigram
  - Smoothing
- Hidden Markov models (HMMs)
  - Part-of-speech tagging
  - Viterbi algorithm


dictionary
ontology
text processing
raw (unstructured) text
part-of-speech tagging
named entity recognition
deep syntactic parsing
annotated (structured) text
…………………………...
... Secretion of TNF was abolished by BHA in PMA-stimulated U937 cells. ……………………
S
VP
NP
VP
PP
NP
PP
PP
NP
NN IN NN VBZ VBN IN NN IN JJ NN NNS .
Secretion of TNF was abolished by BHA in PMA-stimulated U937 cells .
protein_molecule
organic_compound
cell_line
negative regulation

Basic Steps of Natural Language Processing

- Sentence splitting
- Tokenization
- Part-of-speech tagging
- Shallow parsing
- Named entity recognition
- Syntactic parsing
- (Semantic Role Labeling)

Sentence splitting

Current immunosuppression protocols to prevent lung transplant rejection reduce pro-inflammatory and T-helper type 1 (Th1) cytokines. However, Th1 T-cell pro-inflammatory cytokine production is important in host defense against bacterial infection in the lungs. Excessive immunosuppression of Th1 T-cell pro-inflammatory cytokines leaves patients susceptible to infection.

Current immunosuppression protocols to prevent lung transplant rejection reduce pro-inflammatory and T-helper type 1 (Th1) cytokines.

However, Th1 T-cell pro-inflammatory cytokine production is important in host defense against bacterial infection in the lungs.

Excessive immunosuppression of Th1 T-cell pro-inflammatory cytokines leaves patients susceptible to infection.

A heuristic rule for sentence splitting

sentence boundary

= period + space(s) + capital letter

Regular expression in Perl

```
s/\. +([A-Z])/\.\n$1/g;
```

Errors

IL-33 is known to induce the production of Th2-associated cytokines (e.g. IL-5 and IL-13).

IL-33 is known to induce the production of Th2-associated
cytokines (e.g.

IL-5 and IL-13).

- Two solutions:
  - Add more rules to handle exceptions
  - Machine learning

Adding rules to handle exceptions

```
#!/usr/bin/perl

while(<STDIN>) {
    $_ =~ s/([\.\?]) +([\(0-9a-zA-Z])/$1\n$2/g;

    $_ =~ s/(\WDr\.)\n/$1 /g;
    $_ =~ s/(\WMr\.)\n/$1 /g;
    $_ =~ s/(\WMs\.)\n/$1 /g;
    $_ =~ s/(\WMrs\.)\n/$1 /g;
    $_ =~ s/(\Wvs\.)\n/$1 /g;
    $_ =~ s/(\Wa\.m\.)\n/$1 /g;
    $_ =~ s/(\Wp\.m\.)\n/$1 /g;
    $_ =~ s/(\Wi\.e\.)\n/$1 /g;
    $_ =~ s/(\We\.g\.)\n/$1 /g;

    print;
}
```

Tokenization

``We’re heading into a recession.’’

``	**We**	**’re**	**heading**	**into**	**a**	**recession**	**.**	**’’**

- Tokenizing general English sentences is relatively straightforward.
- Use spaces as the boundaries
- Use some heuristics to handle exceptions

Exceptions

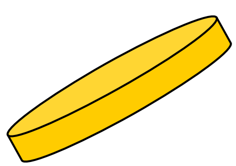
- separate possessive endings or abbreviated forms from preceding words:
  - Mary’s $\rightarrow$ Mary ’s
    Mary’s $\rightarrow$ Mary is
    Mary’s $\rightarrow$ Mary has
- separate punctuation marks and quotes from words **:**
  - Mary. $\rightarrow$ Mary .
  - “new” $\rightarrow$ “ new ”

Tokenization

- Tokenizer.sed: a simple script in *sed*
  - http://www.cis.upenn.edu/~treebank/tokenization.html
- Undesirable tokenization
  - original: “1,25(OH)2D3”
  - tokenized: “1 , 25 (OH) 2D3”
- Tokenization for biomedical text
  - Not straight-forward
  - Needs dictionary? Machine learning?

Maximum likelihood estimation

- Learning parameters from data
- Coin flipping example
  - We want to know the probability that a biased coin comes up heads.
  - Data: {H, T, H, H, T, H, T, T, H, H, T}



$$P(\text{Head}) = \frac{6}{11}$$

Why?

Maximum likelihood estimation

- Likelihood
  - Data: {H, T, H, H, T, H, T, T, H, H, T}

$$P(\text{Data}) = \theta \cdot (1-\theta) \cdot \theta \cdot \theta \cdot (1-\theta) \cdot \theta \cdot (1-\theta) \cdot (1-\theta) \cdot \theta \cdot \theta \cdot (1-\theta)$$
$$= \theta^6 (1-\theta)^5$$

- Maximize the (log) likelihood

$$\log P(\text{Data}) = 6 \log \theta + 5 \log(1-\theta)$$

$$\frac{d \log P(\text{Data})}{d\theta} = \frac{6}{\theta} - \frac{5}{1-\theta} = 0$$

$$\theta = \frac{6}{11}$$

Language models

- A language model assigns a probability to a word sequence $P(w_1 \ldots w_n)$

- Statistical machine translation

English sentence

$$P(\mathrm{e} \mid \mathrm{f}) = \frac{P(\mathrm{f} \mid \mathrm{e}) P(\mathrm{e})}{P(\mathrm{f})} \propto P(\mathrm{f} \mid \mathrm{e}) P(\mathrm{e})$$

French sentence

Language model for English

  – Noisy channel models: machine translation, part-of-speech tagging, speech recognition, optical character recognition, etc.

Language modeling

$w_1 \ldots w_n$	$P(w_1 \ldots w_n)$
He opened the window .	0.0000458
She opened the window .	0.0000723
John was hit .	0.0000244
John hit was .	0.0000002
:	:

- How do you compute $P(w_1 \ldots w_n)$?

  ⇨ Learn from data (a corpus)

Estimate probabilities from a corpus

Corpus

Humpty Dumpty sat on a wall . Humpty Dumpty had a great fall .

$$P(\text{Humpty Dumpty sat on a wall .}) = \frac{1}{2}$$

$$P(\text{Humpty Dumpty had a great fall .}) = \frac{1}{2}$$

$$P(\text{Humpty Dumpty had a fall .}) = 0$$

Sequence to words

- Let's decompose the sequence probability into probabilities of individual words

Chain rule

$$P(x,y) = P(x)P(y \mid x)$$

$$\begin{aligned}
P(w_1 \ldots w_n) &= P(w_1)P(w_2 \ldots w_n \mid w_1) \\
&= P(w_1)P(w_2 \mid w_1)P(w_3 \ldots w_n \mid w_1 w_2) \\
&= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1 w_2)P(w_4 \ldots w_n \mid w_1 w_2 w_3) \\
&= \ldots \\
&= \prod_{i=1}^{n} P(w_i \mid w_1 \ldots w_{i-1})
\end{aligned}$$

Unigram model

- Ignore the context completely

$$P(w_1 \ldots w_n) = \prod_{i=1}^{n} P(w_i \mid w_1 \ldots w_{i-1})$$

$$\approx \prod_{i=1}^{n} P(w_i)$$

- Example

$$P(\text{Humpty Dumpty sat on a wall .})$$
$$\approx P(\text{Humpty})P(\text{Dumpty})P(\text{sat})P(\text{on})P(\text{a})P(\text{wall})P(.)$$

MLE for unigram models

Humpty Dumpty sat on a wall .
Humpty Dumpty had a great fall .

- Count the frequencies

$$P(w_k) = \frac{C(w_k)}{\sum_w C(w_k)}$$

$$P(\text{Humpty}) = \frac{2}{14}$$

$$P(\text{on}) = \frac{1}{14}$$

Unigram model

- Problem

$$P(\text{Humpty Dumpty sat on a wall .}) \approx P(\text{Humpty})P(\text{Dumpty})P(\text{sat})P(\text{on})P(\text{a})P(\text{wall})P(\text{.})$$

$$P(\text{Dumpty Humpty sat on a wall .}) \approx P(\text{Humpty})P(\text{Dumpty})P(\text{sat})P(\text{on})P(\text{a})P(\text{wall})P(\text{.})$$

- Word order is not considered.

Bigram model

- 1st order Markov assumption

$$P(w_1 \ldots w_n) = \prod_{i=1}^{n} P(w_i \mid w_1 \ldots w_{i-1})$$

$$\approx \prod_{i=1}^{n} P(w_i \mid w_{i-1})$$

- Example

$$P(\text{Humpty Dumpty sat on a wall .})$$

$$\approx P(\text{Humpty} \mid \text{<s>}) P(\text{Dumpty} \mid \text{Humpty}) P(\text{sat} \mid \text{Dumpty})$$

$$P(\text{on} \mid \text{sat}) P(\text{a} \mid \text{on}) P(\text{wall} \mid \text{a}) P(. \mid \text{wall})$$

MLE for bigram models

Humpty Dumpty sat on a wall .
Humpty Dumpty had a great fall .

- Count frequencies

$$P(w_k \mid w_{k-1}) = \frac{C(w_{k-1}w_k)}{C(w_{k-1})}$$

$$P(\text{sat} \mid \text{Dumpty}) = \frac{1}{2}$$

$$P(\text{had} \mid \text{Dumpty}) = \frac{1}{2}$$

N-gram models

- *n*-1 order Markov assumption

$$P(w_1 \dots w_n) = \prod_{i=1}^{n} P(w_i \mid w_1 \dots w_{i-1})$$
$$\approx \prod_{i=1}^{n} P(w_i \mid w_{i-n+1} \dots w_{i-1})$$

- Modeling with a large *n*
  - Pros: rich contextual information
  - Cons: sparseness (unreliable probability estimation)

Web 1T 5-gram data

- Released from Google
  - http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13
  - 1 trillion word tokens of text from Web pages
- Examples

serve as the incoming	92
serve as the incubator	99
serve as the independent	794
serve as the index	223
serve as the indication	72
serve as the indicator	120
:	:

Smoothing

- Problems with the maximum likelihood estimation method
  - Events that are not observed in the training corpus are given zero probabilities
  - Unreliable estimates when the counts are small

Add-One Smoothing

Humpty Dumpty sat on a wall .
Humpty Dumpty had a great fall .

- Bigram counts table

	Humpty	Dumpty	sat	on	a	wall	had	great	fall	.
Humpty	0	2	0	0	0	0	0	0	0	0
Dumpty	0	0	1	0	0	0	1	0	0	0
sat	0	0	0	1	0	0	0	0	0	0
on	0	0	0	0	1	0	0	0	0	0
a	0	0	0	0	0	1	0	1	0	0
wall	0	0	0	0	0	0	0	0	0	1
had	0	0	0	0	1	0	0	0	0	0
great	0	0	0	0	0	0	0	0	1	0
fall	0	0	0	0	0	0	0	0	0	1
.	0	0	0	0	0	0	0	0	0	0

Add-One Smoothing

Humpty Dumpty sat on a wall .
Humpty Dumpty had a great fall .

- Add one to all counts

	Humpty	Dumpty	sat	on	a	wall	had	great	fall	.
Humpty	1	3	1	1	1	1	1	1	1	1
Dumpty	1	1	2	1	1	1	2	1	1	1
sat	1	1	1	2	1	1	1	1	1	1
on	1	1	1	1	2	1	1	1	1	1
a	1	1	1	1	1	2	1	2	1	1
wall	1	1	1	1	1	1	1	1	1	2
had	1	1	1	1	2	1	1	1	1	1
great	1	1	1	1	1	1	1	1	2	1
fall	1	1	1	1	1	1	1	1	1	2
.	1	1	1	1	1	1	1	1	1	1

Add-One Smoothing

Humpty Dumpty sat on a wall .
Humpty Dumpty had a great fall .

- *V:* the size of the vocabulary

$$P(w_k \mid w_{k-1}) = \frac{C(w_{k-1}w_k)+1}{C(w_{k-1})+V}$$

$$P(\text{sat} \mid \text{Dumpty}) = \frac{1+1}{2+10}$$

Part-of-speech tagging

Paul Krugman, a professor at Princeton University, was
NNP NNP , DT NN IN NNP NNP , VBD
awarded the Nobel Prize in Economics on Monday.
VBN DT NNP NNP IN NNP IN NNP

- Assigns a part-of-speech tag to each word in the sentence.
- Part-of-speech tags

NN: Noun	IN: Preposition
NNP: Proper noun	VBD: Verb, past tense
DT: Determiner	VBN: Verb, past participle
:	:

Part-of-speech tagging is not easy

- Parts-of-speech are often ambiguous

I have to go there.
verb

I had a go at it.
noun

- We need to look at the context
- But how?

Writing rules for part-of-speech tagging

I have to go there.
verb

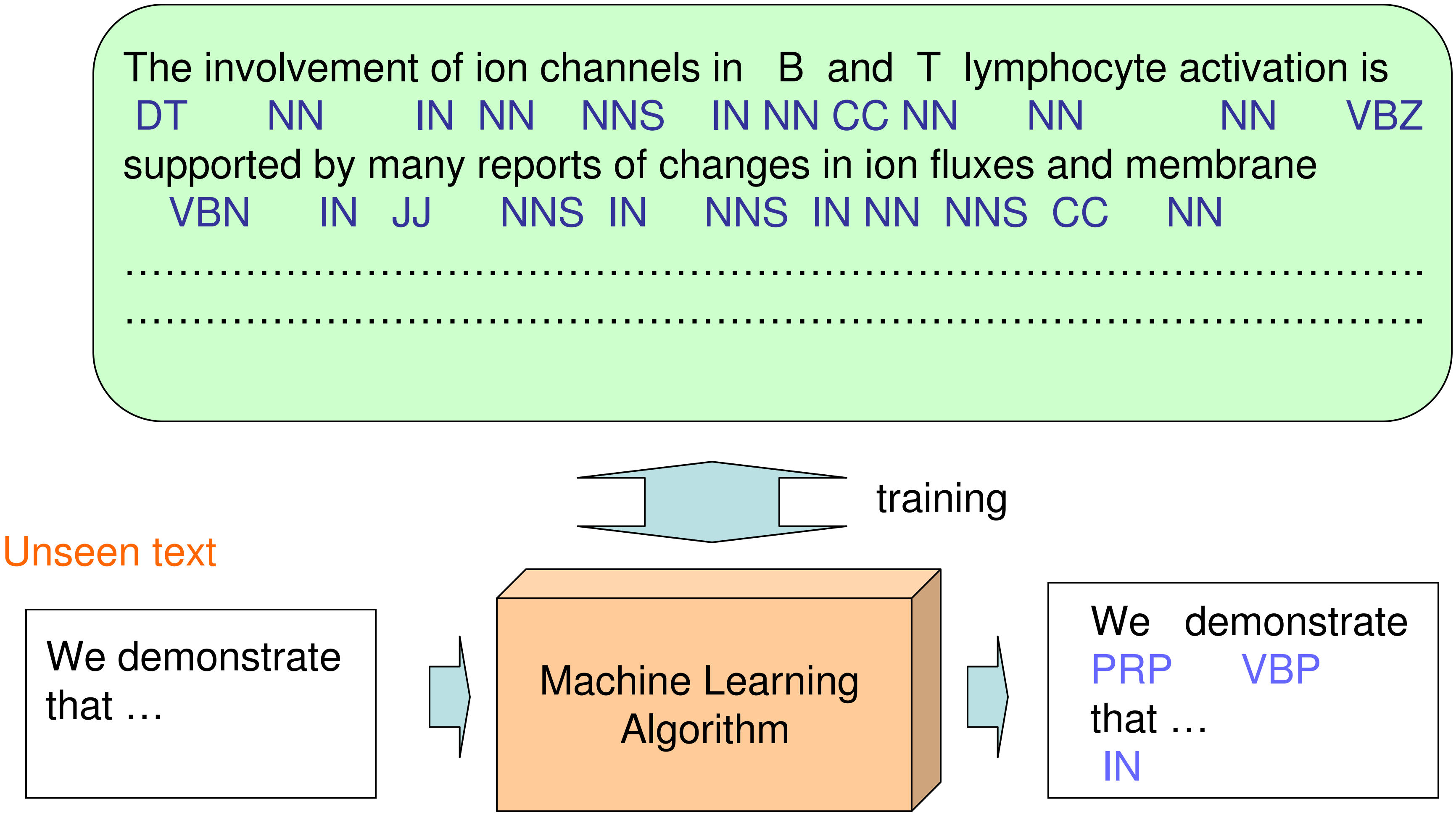
I had a go at it.
noun

- If the previous word is “to”, then it’s a verb.
- If the previous word is “a”, then it’s a noun.
- If the next word is …

:

Writing rules manually is impossible

Learning from examples



Part-of-speech tagging

- Input:
  - Word sequence $W = w_1 \ldots w_n$
- Output (prediction):
  - The most probable tag sequence $T = t_1 \ldots t_n$

$$\hat{T} = \underset{T \in \tau}{\operatorname{argmax}} P(T \mid W)$$

$$= \underset{T \in \tau}{\operatorname{argmax}} \frac{P(T)P(W \mid T)}{P(W)}$$

constant

$$= \underset{T \in \tau}{\operatorname{argmax}} P(T)P(W \mid T)$$

Assumptions

$$
\begin{aligned}
P(W \mid T) &= P(w_1 \ldots w_n \mid t_1 \ldots t_n) \\
&= P(w_1 \mid t_1 \ldots t_n) P(w_2 \ldots w_n \mid w_1 t_1 \ldots t_n) \\
&= P(w_1 \mid t_1 \ldots t_n) P(w_2 \mid w_1 t_1 \ldots t_n) P(w_3 \ldots w_n \mid w_1 w_2 t_1 \ldots t_n) \\
&= \ldots \\
&= \prod_{i=1}^{n} P(w_i \mid w_1 \ldots w_{i-1} t_1 \ldots t_n)
\end{aligned}
$$

- Assume the word emission probability depends only on its tag

$$
P(W \mid T) = \prod_{i=1}^{n} P(w_i \mid t_i)
$$

Part-of-speech tagging with Hidden Markov Models

- Assume the first-order Markov assumption for tags

$$P(T) \approx \prod_{i=1}^{n} P(t_i \mid t_{i-1})$$

- Then, the most probably tag sequence can be expressed as

$$\hat{T} = \operatorname*{argmax}_{T \in \tau} P(T) P(W \mid T)$$

$$= \operatorname*{argmax}_{T \in \tau} \prod_{i=1}^{n} P(t_i \mid t_{i-1}) P(w_i \mid t_i)$$

transition probability   emission probability

Learning from training data

The involvement of ion channels in B and T lymphocyte activation is
DT NN IN NN NNS IN NN CC NN NN NN VBZ
supported by many reports of changes in ion fluxes and membrane
VBN IN JJ NNS IN NNS IN NN NNS CC NN

……………………………………………………………………………………
……………………………………………………………………………………

Maximum likelihood estimation

transition probability $$P(t_i \mid t_{i-1}) = \frac{C(t_{i-1} t_i)}{C(t_{i-1})}$$

emission probability $$P(w_i \mid t_i) = \frac{C(w_i t_i)}{C(t_i)}$$

Examples from the Wall Street Journal corpus (1)

$$P(\mathrm{NN} \mid \mathrm{DT}) = 0.47$$
$$P(\mathrm{JJ} \mid \mathrm{DT}) = 0.22$$
$$P(\mathrm{NNP} \mid \mathrm{DT}) = 0.11$$
$$P(\mathrm{NNS} \mid \mathrm{DT}) = 0.07$$
$$P(\mathrm{CD} \mid \mathrm{DT}) = 0.02$$

$$P(\mathrm{IN} \mid \mathrm{NN}) = 0.25$$
$$P(\mathrm{NN} \mid \mathrm{NN}) = 0.12$$
$$P(, \mid NN) = 0.11$$
$$P(. \mid NN) = 0.08$$
$$P(\mathrm{VBD} \mid NN) = 0.05$$

POS Tag	Description	Examples
DT	determiner	a, the
NN	noun, singular or mass	company
NNS	noun plural	companies
NNP	proper nouns, singular	Nasdaq
IN	preposition/subordinating conjunction	in, of, like
VBD	verb, past tense	took
:	:	:

Examples from the Wall Street Journal corpus (2)

$$P(\% \mid \text{NN}) = 0.037$$
$$P(\text{company} \mid \text{NN}) = 0.019$$
$$P(\text{year} \mid \text{NN}) = 0.0167$$
$$P(\text{market} \mid \text{NN}) = 0.0155$$
$$P(\text{share} \mid \text{NN}) = 0.0107$$

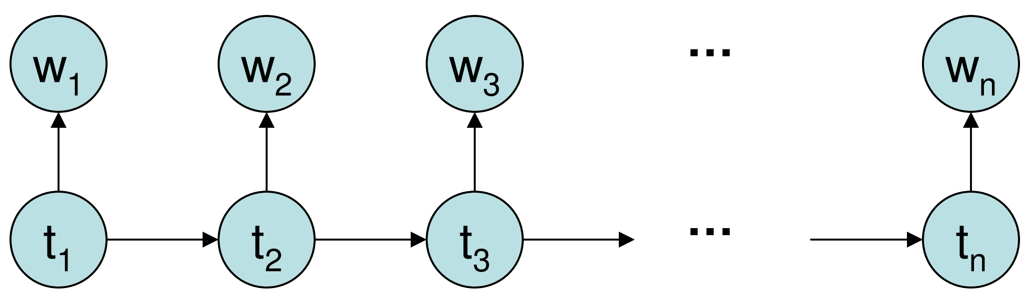
$$P(\text{said} \mid \text{VBD}) = 0.188$$
$$P(\text{was} \mid \text{VBD}) = 0.131$$
$$P(\text{were} \mid \text{VBD}) = 0.064$$
$$P(\text{had} \mid \text{VBD}) = 0.056$$
$$P(\text{rose} \mid \text{VBD}) = 0.022$$

POS Tag	Description	Examples
DT	determiner	a, the
NN	noun, singular or mass	company
NNS	noun plural	companies
NNP	proper nouns, singular	Nasdaq
IN	preposition/subordinating conjunction	in, of, like
VBD	verb, past tense	took
:	:	:

Part-of-speech tagging

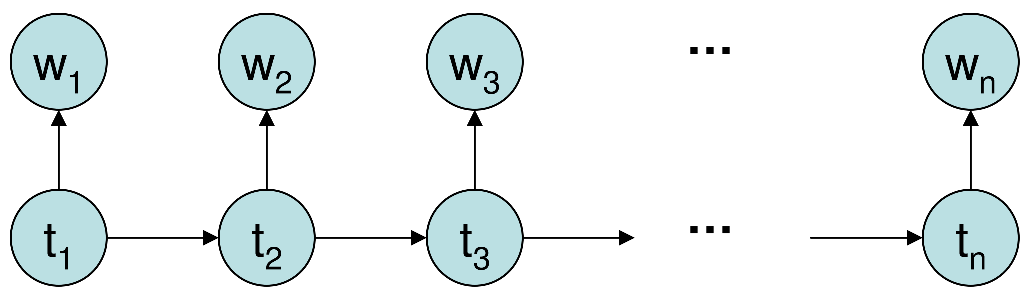
- Input:
  - Word sequence $W = w_1 \ldots w_n$
- Output (prediction):
  - The most probable tag sequence $T = t_1 \ldots t_n$

$$\hat{T} = \underset{T \in \tau}{\operatorname{argmax}} \prod_{i=1}^{n} P(t_i \mid t_{i-1}) P(w_i \mid t_i)$$



Finding the best sequence

- Naive approach:
  - Enumerate all possible tag sequences with their probabilities and select the one that gives the highest probability



DT	DT	DT	DT
NN	NN	NN	NN
NNS	NNS	NNS	NNS
NNP	NNP	NNP	NNP
VB	VB	VB	VB
:	:	:	:

→ exponential in the length of the sentence

Finding the best sequence

- if you write them down…

$$P(\mathrm{DT} \mid <\mathrm{s}>)P(w_1 \mid \mathrm{DT})P(\mathrm{DT} \mid \mathrm{DT})P(w_2 \mid \mathrm{DT})\Lambda\ P(\mathrm{DT} \mid \mathrm{DT})P(w_n \mid \mathrm{DT})$$

$$P(\mathrm{NN} \mid <\mathrm{s}>)P(w_1 \mid \mathrm{NN})P(\mathrm{DT} \mid \mathrm{DT})P(w_2 \mid \mathrm{DT})\Lambda\ P(\mathrm{DT} \mid \mathrm{DT})P(w_n \mid \mathrm{DT})$$

$$P(\mathrm{NNS} \mid <\mathrm{s}>)P(w_1 \mid \mathrm{NNS})P(\mathrm{DT} \mid \mathrm{DT})P(w_2 \mid \mathrm{DT})\Lambda\ P(\mathrm{DT} \mid \mathrm{DT})P(w_n \mid \mathrm{DT})$$

$$\vdots$$

$$P(\mathrm{DT} \mid <\mathrm{s}>)P(w_1 \mid \mathrm{DT})P(\mathrm{NN} \mid \mathrm{DT})P(w_2 \mid \mathrm{NN})\Lambda\ P(\mathrm{DT} \mid \mathrm{DT})P(w_n \mid \mathrm{DT})$$

$$P(\mathrm{NN} \mid <\mathrm{s}>)P(w_1 \mid \mathrm{NN})P(\mathrm{NN} \mid \mathrm{NN})P(w_2 \mid \mathrm{NN})\Lambda\ P(\mathrm{DT} \mid \mathrm{DT})P(w_n \mid \mathrm{DT})$$

$$P(\mathrm{NNS} \mid <\mathrm{s}>)P(w_1 \mid \mathrm{NNS})P(\mathrm{NN} \mid \mathrm{NNS})P(w_2 \mid \mathrm{NN})\Lambda\ P(\mathrm{DT} \mid \mathrm{DT})P(w_n \mid \mathrm{DT})$$

$$\vdots$$

A lot of redundancy in computation

There should be a way to do it more efficiently

Viterbi algorithm (1)

- Let $V_k(t_k)$ be the probability of the best sequence for $w_1 \ldots w_k$ ending with the tag t_k

$$V_k(t_k) \equiv \max_{t_1, t_2, \ldots t_{k-1}} \prod_{i=1}^{k} P(t_i \mid t_{i-1}) P(w_i \mid t_i)$$

$$= \max_{t_{k-1}} \max_{t_1, t_2 \ldots t_{k-2}} \prod_{i=1}^{k} P(t_i \mid t_{i-1}) P(w_i \mid t_i)$$

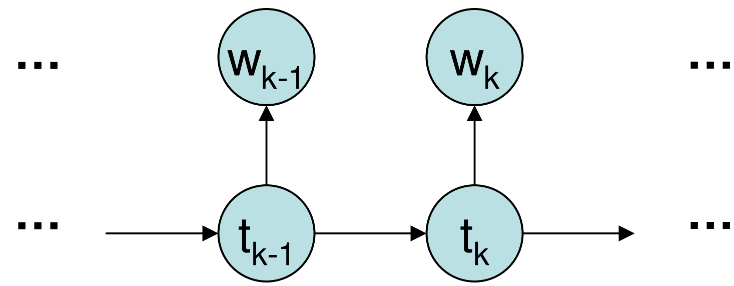
$$= \max_{t_{k-1}} P(t_k \mid t_{k-1}) P(w_k \mid t_k) \max_{t_1, t_2 \ldots t_{k-2}} \prod_{i=1}^{k-1} P(t_i \mid t_{i-1}) P(w_i \mid t_i)$$

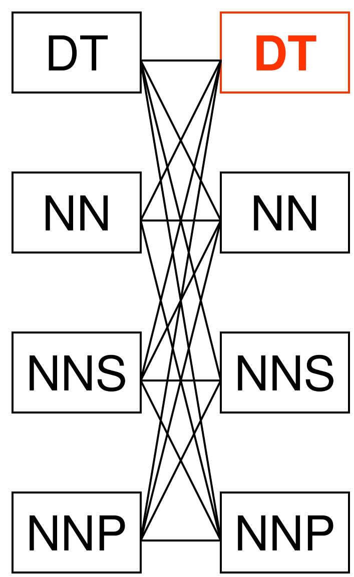
$$= \max_{t_{k-1}} P(t_k \mid t_{k-1}) P(w_k \mid t_k) \underline{V_{k-1}(t_{k-1})}$$

We can compute them in a recursive manner!

Viterbi algorithm (2)

$$V_k(t_k) = \max_{t_{k-1}} P(t_k \mid t_{k-1}) P(w_k \mid t_k) V_{k-1}(t_{k-1})$$





Beginning of the sentence

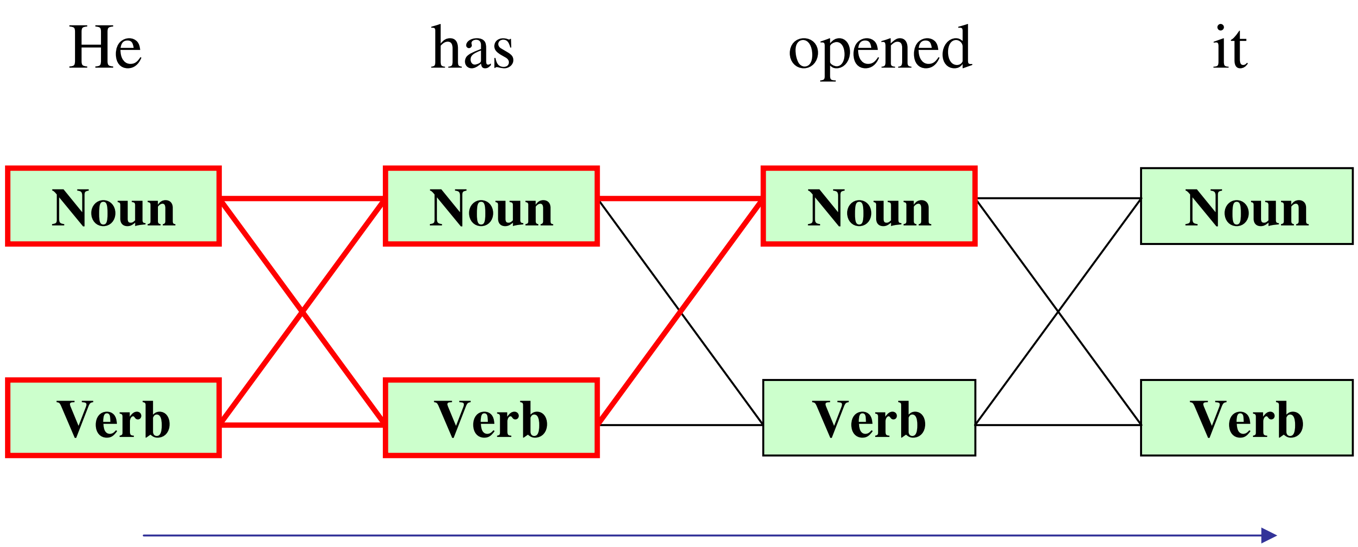
$$V_0(<\mathrm{s}>) = 1$$

$$P(\mathrm{DT} \mid \mathrm{DT}) P(w_k \mid \mathrm{DT}) V_{k-1}(\mathrm{DT})$$
$$P(\mathrm{DT} \mid \mathrm{NN}) P(w_k \mid \mathrm{DT}) V_{k-1}(\mathrm{NN})$$
$$P(\mathrm{DT} \mid \mathrm{NNS}) P(w_k \mid \mathrm{DT}) V_{k-1}(\mathrm{NNS})$$
$$P(\mathrm{DT} \mid \mathrm{NNP}) P(w_k \mid \mathrm{DT}) V_{k-1}(\mathrm{NNP})$$
$$\vdots$$

Viterbi algorithm (3)

- Left to right
  - Save backward pointers to recover the best tag sequence

Complexity: (number of tags)^2 x (length)



Topics

- Sentence splitting
- Tokenization
- Maximum likelihood estimation (MLE)
- Language models
  - Unigram
  - Bigram
  - Smoothing
- Hidden Markov models (HMMs)
  - Part-of-speech tagging
  - Viterbi algorithm