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
Secretion of TNF was abolished by BHA in PMA-stimulated U937 cells.
Basic Steps of Natural Language Processing

• Sentence splitting
• Tokenization
• Part-of-speech tagging
• Shallow parsing
• Named entity recognition
• Syntactic parsing
• (Semantic Role Labeling)
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 =~ s/. +([A-Z])/. \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

```perl
#!/usr/bin/perl

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

    print;
}
```
Tokenization

```
```
```
```

```
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 → Mary ’s
  – Mary’s → Mary is
  – Mary’s → Mary has

• separate punctuation marks and quotes from words:
  – Mary. → Mary .
  – “new” → “ new ”
Tokenization

- Tokenizer.sed: a simple script in `sed`
  - [http://www.cis.upenn.edu/~treebank/tokenization.html](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.

\[
P(\text{Head}) = \frac{6}{11}
\]
Maximum likelihood estimation

• Likelihood

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

• 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

\[ P(e \mid f) = \frac{P(f \mid e)P(e)}{P(f)} \propto P(f \mid e)P(e) \]

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

<table>
<thead>
<tr>
<th>$w_1...w_n$</th>
<th>$P(w_1...w_n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>He opened the window .</td>
<td>0.0000458</td>
</tr>
<tr>
<td>She opened the window .</td>
<td>0.0000723</td>
</tr>
<tr>
<td>John was hit .</td>
<td>0.0000244</td>
</tr>
<tr>
<td>John hit was .</td>
<td>0.0000002</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

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

\[
P(w_1...w_n) = P(w_1)P(w_2...w_n | w_1)
\]

\[
= P(w_1)P(w_2 | w_1)P(w_3...w_n | w_1w_2)
\]

\[
= P(w_1)P(w_2 | w_1)P(w_3 | w_1w_2)P(w_4...w_n | w_1w_2w_3)
\]

\[
= ...
\]

\[
= \prod_{i=1}^{n} P(w_i | w_1...w_{i-1})
\]

Chain rule: \( P(x, y) = P(x)P(y | x) \)
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

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

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

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

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

• Word order is not considered.
Bigram model

• 1\textsuperscript{st} order Markov assumption

\[
P(w_1...w_n) = \prod_{i=1}^{n} P(w_i | w_1...w_{i-1})
\]

\[
\approx \prod_{i=1}^{n} P(w_i | w_{i-1})
\]

• Example

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

\[
\approx P(\text{Humpty} | <s>) P(\text{Dumpty} | \text{Humpty}) P(\text{sat} | \text{Dumpty})
\]

\[
P(\text{on} | \text{sat}) P(\text{a} | \text{on}) P(\text{wall} | \text{a}) P(\text{.} | \text{wall})
\]
MLE for bigram models

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

- Count frequencies

\[ P(w_k | w_{k-1}) = \frac{C(w_{k-1}w_k)}{C(w_{k-1})} \]

\[ P(\text{sat} | \text{Dumpty}) = \frac{1}{2} \]

\[ P(\text{had} | \text{Dumpty}) = \frac{1}{2} \]
N-gram models

• $n$-1 order Markov assumption

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

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

• Modeling with a large $n$
  – Pros: rich contextual information
  – Cons: sparseness (unreliable probability estimation)
Web 1T 5-gram data

• Released from Google
  – 1 trillion word tokens of text from Web pages

• Examples

<table>
<thead>
<tr>
<th>serve as the incoming</th>
<th>92</th>
</tr>
</thead>
<tbody>
<tr>
<td>serve as the incubator</td>
<td>99</td>
</tr>
<tr>
<td>serve as the independent</td>
<td>794</td>
</tr>
<tr>
<td>serve as the index</td>
<td>223</td>
</tr>
<tr>
<td>serve as the indication</td>
<td>72</td>
</tr>
<tr>
<td>serve as the indicator</td>
<td>120</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>Humpty</th>
<th>Dumpty</th>
<th>sat</th>
<th>on</th>
<th>a</th>
<th>wall</th>
<th>had</th>
<th>great</th>
<th>fall</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humpty</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dumpty</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sat</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>on</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wall</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>had</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>great</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>fall</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Add-One Smoothing

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

- Add one to all counts

<table>
<thead>
<tr>
<th></th>
<th>Humpty</th>
<th>Dumpty</th>
<th>sat</th>
<th>on</th>
<th>a</th>
<th>wall</th>
<th>had</th>
<th>great</th>
<th>fall</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humpty</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Dumpty</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>sat</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>on</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>wall</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>had</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>great</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>fall</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>.</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Add-One Smoothing

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

- $V$: the size of the vocabulary

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

$$P(\text{sat} | \text{Dumpty}) = \frac{1+1}{2+10}$$
Part-of-speech tagging

Paul Krugman, a professor at Princeton University, was awarded the Nobel Prize in Economics on Monday.

- Assigns a part-of-speech tag to each word in the sentence.
- Part-of-speech tags
  - **NN**: Noun
  - **NNP**: Proper noun
  - **DT**: Determiner
  - **IN**: Preposition
  - **VBD**: Verb, past tense
  - **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.  I had a go at it.

verb  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
The involvement of ion channels in B and T lymphocyte activation is supported by many reports of changes in ion fluxes and membrane.

We demonstrate that …
Part-of-speech tagging

• Input:
  – Word sequence \( W = w_1...w_n \)

• Output (prediction):
  – The most probable tag sequence \( T = t_1...t_n \)

\[
\hat{T} = \arg\max_{T \in \tau} P(T \mid W) \\
= \arg\max_{T \in \tau} \frac{P(T)P(W \mid T)}{P(W)} \\
= \arg\max_{T \in \tau} P(T)P(W \mid T)
\]
Assumptions

\[ P(W \mid T) = P(w_1 \ldots w_n \mid t_1 \ldots t_n) \]
\[ = P(w_1 \mid t_1 \ldots t_n)P(w_2 \ldots w_n \mid w_1 t_1 \ldots t_n) \]
\[ = P(w_1 \mid t_1 \ldots t_n)P(w_2 \mid w_1 t_1 \ldots t_n)P(w_3 \ldots w_n \mid w_1 w_2 t_1 \ldots t_n) \]
\[ = \ldots \]
\[ = \prod_{i=1}^{n} P(w_i \mid w_1 \ldots w_{i-1} t_1 \ldots t_n) \]

• 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

\[
\hat{T} = \arg\max_{T \in \tau} P(T)P(W \mid T)
\]

\[
= \arg\max_{T \in \tau} \prod_{i=1}^{n} P(t_i \mid t_{i-1})P(w_i \mid t_i)
\]

transition probability emission probability
Learning from training data

The involvement of ion channels in B and T lymphocyte activation is supported by many reports of changes in ion fluxes and membrane properties.
Examples from the Wall Street Journal corpus (1)

\[
P(\text{NN} \mid \text{DT}) = 0.47 \quad P(\text{IN} \mid \text{NN}) = 0.25 \\
P(\text{JJ} \mid \text{DT}) = 0.22 \quad P(\text{NN} \mid \text{NN}) = 0.12 \\
P(\text{NNP} \mid \text{DT}) = 0.11 \quad P(, \mid \text{NN}) = 0.11 \\
P(\text{NNS} \mid \text{DT}) = 0.07 \quad P(. \mid \text{NN}) = 0.08 \\
P(\text{CD} \mid \text{DT}) = 0.02 \quad P(\text{VBD} \mid \text{NN}) = 0.05
\]

<table>
<thead>
<tr>
<th>POS Tag</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>determiner</td>
<td>a, the</td>
</tr>
<tr>
<td>NN</td>
<td>noun, singular or mass</td>
<td>company</td>
</tr>
<tr>
<td>NNS</td>
<td>noun plural</td>
<td>companies</td>
</tr>
<tr>
<td>NNP</td>
<td>proper nouns, singular</td>
<td>Nasdaq</td>
</tr>
<tr>
<td>IN</td>
<td>preposition/subordinating conjunction</td>
<td>in, of, like</td>
</tr>
<tr>
<td>VBD</td>
<td>verb, past tense</td>
<td>took</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
Examples from the Wall Street Journal corpus (2)

\[ P(\% \mid \text{NN}) = 0.037 \]
\[ P(\text{company} \mid \text{NN}) = 0.019 \]
\[ P(\text{year} \mid \text{NN}) = 0.0167 \]
\[ P(\text{market} \mid \text{NN}) = 0.0155 \]
\[ P(\text{share} \mid \text{NN}) = 0.0107 \]
\[ P(\text{said} \mid \text{VBD}) = 0.188 \]
\[ P(\text{was} \mid \text{VBD}) = 0.131 \]
\[ P(\text{were} \mid \text{VBD}) = 0.064 \]
\[ P(\text{had} \mid \text{VBD}) = 0.056 \]
\[ P(\text{rose} \mid \text{VBD}) = 0.022 \]

<table>
<thead>
<tr>
<th>POS Tag</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>determiner</td>
<td>a, the</td>
</tr>
<tr>
<td>NN</td>
<td>noun, singular or mass</td>
<td>company</td>
</tr>
<tr>
<td>NNS</td>
<td>noun plural</td>
<td>companies</td>
</tr>
<tr>
<td>NNP</td>
<td>proper nouns, singular</td>
<td>Nasdaq</td>
</tr>
<tr>
<td>IN</td>
<td>preposition/subordinating conjunction</td>
<td>in, of, like</td>
</tr>
<tr>
<td>VBD</td>
<td>verb, past tense</td>
<td>took</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
Part-of-speech tagging

• Input:
  – Word sequence \( W = w_1...w_n \)

• Output (prediction):
  – The most probable tag sequence \( T = t_1...t_n \)

\[
T = \arg\max_{T \in \tau} \prod_{i=1}^{n} P(t_i | t_{i-1}) P(w_i | t_i)
\]

\[
\begin{align*}
\text{w}_1 & \quad \text{w}_2 & \quad \text{w}_3 & \quad \ldots & \quad \text{w}_n \\
\text{t}_1 & \quad \text{t}_2 & \quad \text{t}_3 & \quad \ldots & \quad \text{t}_n
\end{align*}
\]
Finding the best sequence

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

$$w_1, t_1 \rightarrow w_2, t_2 \rightarrow w_3, t_3 \rightarrow \ldots \rightarrow w_n, t_n$$

- exponential in the length of the sentence
Finding the best sequence

- if you write them down...

\[
P(DT \mid < s >)P(w_1 \mid DT)P(DT \mid DT)P(w_2 \mid DT)\Lambda P(DT \mid DT)P(w_n \mid DT) \\
P(NN \mid < s >)P(w_1 \mid NN)P(DT \mid DT)P(w_2 \mid DT)\Lambda P(DT \mid DT)P(w_n \mid DT) \\
P(NNS \mid < s >)P(w_1 \mid NNS)P(DT \mid DT)P(w_2 \mid DT)\Lambda P(DT \mid DT)P(w_n \mid DT) \\
: \\
P(DT \mid < s >)P(w_1 \mid DT)P(NN \mid DT)P(w_2 \mid NN)\Lambda P(DT \mid DT)P(w_n \mid DT) \\
P(NN \mid < s >)P(w_1 \mid NN)P(NN \mid NN)P(w_2 \mid NN)\Lambda P(DT \mid DT)P(w_n \mid DT) \\
P(NNS \mid < s >)P(w_1 \mid NNS)P(NN \mid NNS)P(w_2 \mid NN)\Lambda P(DT \mid DT)P(w_n \mid DT) \\
:
\]

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 $w_1...w_k$ ending with the tag $t_k$

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

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

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

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

We can compute them in a recursive manner!
Viterbi algorithm (2)

\[ V_k(t_k) = \max_{t_k-1} P(t_k | t_{k-1}) P(w_k | t_k) V_{k-1}(t_{k-1}) \]

Beginning of the sentence

\[ V_0(<s>) = 1 \]

\[ P(DT | DT) P(w_k | DT) V_{k-1}(DT) \]
\[ P(DT | NN) P(w_k | DT) V_{k-1}(NN) \]
\[ P(DT | NNS) P(w_k | DT) V_{k-1}(NNS) \]
\[ P(DT | NNP) P(w_k | DT) V_{k-1}(NNP) \]
\[ : \]
Viterbi algorithm (3)

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

Complexity: \((\text{number of tags})^2 \times \text{(length)}\)
Topics

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