Natural Language Processing

Tagging

Based on slides from Michael Collins, Yoav Goldberg, Yoav Artzi
Plan

• What are part-of-speech tags?
• What are tagging models?
• Models for tagging
## Model zoo

<table>
<thead>
<tr>
<th></th>
<th>Tagging</th>
<th>Parsing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generative</td>
<td>HMMs</td>
<td>PCFGs</td>
</tr>
<tr>
<td>Log-linear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greedy</td>
<td></td>
<td>Transition-based parsing</td>
</tr>
<tr>
<td>Locally normalized</td>
<td>Viterbi</td>
<td>CKY</td>
</tr>
<tr>
<td>Globally normalized</td>
<td>forward-backward</td>
<td>Inside-outside</td>
</tr>
</tbody>
</table>

And also deep learning methods!
Part-of-speech tags

The/DT stuffy/JJ room/NN is/AUX filled/VBN with/IN students/NNS
Cross-language variations

• Some languages don’t have adjectives or adverbs (use verbs instead)
  • “I see well”: “eye me-clear”

• Some language have noun classifiers
  • Mandarin ("three —long and flat— knives")
  • Thai
What is it good for

- Characterizes the behavior of a word in context
- Useful for downstream applications/tasks
  - Parsing
    - what are parse trees good for?
  - Question answering (if the question is “when” then the answer should contain a numeral)
  - Articulation (object/object)
  - Punctuation (Hebrew)
- But with a lot of data can often do without it.
How to determine what it is

• Morphology
  • -ly
  • ה’ הידיעה
  • Can you add possesives?

• Context
  • Appears after “the”
  • Before a verb

The yinkish dripner blorked quastofically into the nindin with the pidbis.
How to determine what it is

The yinkish dripner blorked quastofically into the nindin with the pidibs.

- yinkish: adjective (or noun?)
- dripner: noun
- blorked: verb
- quastofically: adverb
- nindin: noun
- pidibs: noun
conjunction—>preposition

I study nlp because
I like science

I study nlp because science
Nouns

• In English:
  • Take 's, ness, ment
  • Occur after “the”
  • Refer to objects and entities in the real world
  • But also “smoking is forbidden”

• Types:
  • Proper nouns: Israel, Tel-Aviv University
  • Common nouns
    • Count nouns: classroom
    • Mass nouns: sand
Verbs

- Usually describe actions, states and processes
  - eat, talk
- Can have rich morphology (tense, aspect, …)
Adjectives and adverbs

- Adjectives:
  - Describe a noun, a property of some entity or object (color, texture, state, ...)

- Adverbs:
  - Describe an action (direction, degree, manner, time, location)
Prepositions vs. particles

• Prepositions
  • *on* the table

• Particles:
  • *interested in* music
# Penn Treebank tag set

![Penn Treebank part-of-speech tags (including punctuation)](image)

**Figure 10.1** Penn Treebank part-of-speech tags (including punctuation).
Universal tag set (>20 languages)

- **ADJ**: adjective
- **ADP**: adposition
- **ADV**: adverb
- **AUX**: auxiliary
- **CCONJ**: coordinating conjunction
- **DET**: determiner
- **INTJ**: interjection
- **NOUN**: noun
- **NUM**: numeral
- **PART**: particle
- **PRON**: pronoun
- **PROPN**: proper noun
- **PUNCT**: punctuation
- **SCONJ**: subordinating conjunction
- **SYM**: symbol
- **VERB**: verb
- **X**: other
Part-of-speech tagging

**Input:**
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

**Output:**
Profits/N soared/V at/P Boeing/N Co./N ,/, easily/Adv topping/V forecasts/N on/P Wall/N Street/N ,/, as/P their/POSS CEO/N Alan/N Mulally/N announced/V first/Adj quarter/N results/N ./.

N: noun  
V: verb  
P: preposition  
Adv: adverb  
Adj: adjective  
...

- Why is this hard?
Information sources

• Local:
  
  • *can* is usually a modal verb (MD) but is sometimes a noun (N)

• Contextual
  
  • a noun is more likely than a verb after a noun
    
    • DT NN is common but DT VB is not

• Conflict:
  
  • *The trash can is in the garage*
Distribution of POS tags

- Brown corpus
- “most common tag” baseline gets ~90%
- SOTA: 97%
- one error per 30 words

<table>
<thead>
<tr>
<th>#tags</th>
<th>#word types</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35340</td>
</tr>
<tr>
<td>2</td>
<td>3760</td>
</tr>
<tr>
<td>3</td>
<td>264</td>
</tr>
<tr>
<td>4</td>
<td>61</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>
Transformation-based tagger

- Start with most frequent tag per word
- Then apply rules that change it based on context
  - *If VB and DT before change VB to NN*
  - ...
- Learn from data which rules work well
Named entity recognition

**Input:**
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

**Output:**
Profits soared at [Boeing Co.]org, easily topping forecasts on [Wall Street]loc, as their CEO [Alan Mulally]per announced first quarter results.
NER as tagging

**Input:**
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

**Output:**
Profits/O soared/O at/O Boeing/B-org Co./I-org ,/O easily/O topping/O forecasts/O on/O Wall/B-loc Street/I-loc ,/O as/O their/O CEO/O Alan/B-per Mulally/I-per announced/O first/O quarter/O results/ O ./O
Chunking

• NER is an example for chunking, which is useful in general information extraction
  • Find companies
  • Diseases
  • Genes
  • …
Training set

1 Pierre/NNP Vinken/NNP ./, 61/CD years/NNS old/JJ ./, will/MD join/VB the/DT board/NN as/IN a/DT nonexecutive/JJ director/NN Nov./NNP 29/CD ./.

2 Mr./NNP Vinken/NNP is/VBZ chairman/NN of/IN Elsevier/NNP N.V./NNP ./, the/DT Dutch/NNP publishing/VBG group/NN ./.

3 Rudolph/NNP Agnew/NNP ./, 55/CD years/NNS old/JJ and/CC former/JJ chairman/NN of/IN Consolidated/NNP Gold/NNP Fields/NNP PLC/NNP ./, was/VBD named/VBN a/DT nonexecutive/JJ director/NN of/IN this/DT British/JJ industrial/JJ conglomerate/NN ./.

... ... ...

38,219 That/DT could/MD cost/VB him/PRP the/DT chance/NN to/TO influence/VB the/DT outcome/NN and/CC perhaps/RB join/VB the/DT winning/VBG bidder/NN ./.

Learn a function from sentences to tag sequences
Supervised learning

• Input: sentence-tag sequence pairs

\[
\{x^{(i)}, y^{(i)}\}_{i=1}^{m}
\]

• Output: function \( f \) from sentence \( x \) to tag sequence \( y \)

• Example:

\[
\begin{align*}
x^{(i)} & = \text{the} \quad \text{dog} \quad \text{laughs} \\
y^{(i)} & = \text{DT} \quad \text{NN} \quad \text{VBZ}
\end{align*}
\]
Structured prediction

• The size of the output space depends on the size of the input
  • Compare to language modeling
  • Not binary nor multi-class
• The size of the output is exponential in the size of the input
  • What is the size of the output space?
Conditional model

• Learn from data a model for $p(y \mid x)$

• *Given a sentence* $x$ *return:*

$$\arg\max_y p(y \mid x)$$

But how do you define a model $p(y \mid x)$?
Generative model

- Learn a model over \( p(x, y) \)
  - Express \( p(x, y) = p(y)p(x \mid y) \)
  - Generative story: generate tags, and then produce words from tags
    - \( p(y) \): prior on tags (generate a sequence of tags)
    - \( p(x \mid y) \): data conditioned on tags (generate words from tags)
- Use Bayes rule to express \( p(y \mid x) \):
  \[
p(y \mid x) = \frac{p(y)p(x \mid y)}{\sum_{y'} p(y')p(x \mid y')}
  \]
- **Output**: \( f(x) = \arg \max_y p(y \mid x) = \arg \max_y \frac{p(y)p(x \mid y)}{\sum_{y'} p(y')p(x \mid y')} = \arg \max_y p(y)p(x \mid y) \)
Hidden Markov Models

• Goal: define a tagging model $p(x, y) = p(y)p(x \mid y)$

• $p(y)$ is a trigram model over tag sequences

  • Model tag **transitions**

  $$p(y) = \prod_{i=1}^{n+1} q(y_i \mid y_{i-2}, y_{i-1}), \ y_0 = y_{-1} = *, \ y_{n+1} = \text{STOP}$$
Hidden Markov Models

• $p(x \mid y)$ models emissions

• Assumption: a word depends only on its tag (is this reasonable?)

\[
p(x \mid y) = p(x_1 \mid y) \\
\times p(x_2 \mid x_1, y) \\
\times \ldots \\
\times p(x_n \mid x_1, \ldots, x_{n-1}, y)
\]

\[
= \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1}, y) \approx \prod_{i=1}^{n} e(x_i \mid y_i)
\]
Hidden Markov Models

\[ p(x_1, \ldots, x_n, y_1 \ldots, y_n, \text{STOP}) = \]

\[ q(\text{STOP} \mid y_{n-1}, y_n) \times \prod_{i=1}^{n} q(y_i \mid y_{i-2}, y_{i-1}) \cdot e(x_i \mid y_i) \]

\[ y_0 = y_{-1} = * \]
Example

\[ p(\text{the, dog, laughs, DT, N, V}) \]
\[ = q(\text{DT} \mid *, *) \times q(\text{N} \mid *, \text{DT}) \times q(\text{V} \mid \text{DT, N}) \times q(\text{STOP} \mid \text{N, V}) \]
\[ \times e(\text{the} \mid \text{DT}) \times e(\text{dog} \mid \text{N}) \times e(\text{laughs} \mid \text{V}) \]

- In practice work in log space
Parameter estimation

• Transition parameters:

  • Get Maximum likelihood (ML) from training data (like language models)

    \[ q(y_i | y_{i-2}, y_{i-1}) = \frac{c(y_{i-2}, y_{i-1}, y_i)}{c(y_{i-2}, y_{i-1})} \]

  • Smoothing to handle rare tag trigrams and bigrams

    \[ q(y_i | y_{i-2}, y_{i-1}) = \lambda_1 \cdot \frac{c(y_{i-2}, y_{i-1}, y_i)}{c(y_{i-2}, y_{i-1})} + \lambda_2 \cdot \frac{c(y_{i-1}, y_i)}{c(y_{i-1})} + \lambda_3 \cdot \frac{c(y_i)}{M} \]

    \[ \sum_{i} \lambda_i = 1, \lambda_i \geq 0 \]
Parameter estimation

• Emission parameters

• Again, ML estimates from training data

\[ e(x | y) = \frac{c(x, y)}{c(y)} \]

• Can we have zero counts in the test?
Rare words problems

Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

- We encounter new words frequently in real text
  - Names, rare words, typos, etc. etc.
  - \( p(y \mid x) = 0 \) for all \( y \)
Common solution

• Partition the vocabulary to two sets
  • **frequent**: occurring more than $K(=5)$ times
  • **rare**: other words

• Cluster rare words to a small number of clusters manually

• Preprocess the training data to replace rare words with their cluster ID.
## Example (named entity recognition)

<table>
<thead>
<tr>
<th>Class</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>1234</td>
</tr>
<tr>
<td>NumberAndSlash</td>
<td>29/3/2017</td>
</tr>
<tr>
<td>allCaps</td>
<td>NLP</td>
</tr>
<tr>
<td>initCap</td>
<td>Jonathan</td>
</tr>
<tr>
<td>firstWord</td>
<td></td>
</tr>
<tr>
<td>lowercase</td>
<td>table</td>
</tr>
<tr>
<td>other</td>
<td>;</td>
</tr>
</tbody>
</table>
Example (named entity recognition)

**Input:**
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

**Output:**

```plaintext
firstWord/O soared/O at/O initCap/B-org Co./I-org ,/O easily/O lowercase/O forecasts/O on/O initCap/B-loc Street/I-loc ,/O as/O their/O CEO/O Alan/B-per initCap/I-per announced/O first/O quarter/O results/O ./O
```
Inference/decoding
The problem

• The decoding problem is to compute
  
  \[ \text{argmax}_y p(x, y) = \text{argmax}_y p(y \mid x) \]

• This is necessary in HMMs at test time
Brute force

- Score all possible tag sequences $y$

- With sequences of length $n$ and a tag set $S$, this takes exponential time $|S| \times |S| \times \ldots = |S|^n$
Greedy

• choose $y_1$ that maximizes $e(x_1 \mid y_1)q(y_1 \mid *, *)$

• choose $y_2$ that maximizes $e(x_2 \mid y_2)q(y_2 \mid *, y_1)$

• ...

• choose $y_n$ that maximizes $e(x_n \mid y_n)q(y_n \mid y_{n-2}, y_{n-1})$

**Complexity:** $O(|S|^n)$ but incorrect!

**Exercise:** Build an example (with $q$ and $e$ parameters) where the greedy algorithm fails
Viterbi

- Efficient and correct algorithm

**Intuition:** The HMM model decomposes the probability of the full tag sequence to a product of more local probabilities
Viterbi

• Let $n$ be the length of the sentence, and $S$ be the tag set

• $S_k$ is the tag set for position $k$, $-1 \leq k \leq n$

$$S_0 = S_{-1} = \{\ast\}, S_k = S$$

• Define the probability of a tag prefix:

$$r(y_{-1}, y_0, y_1, \ldots, y_k) = \prod_{i=1}^{k} q(y_i \mid y_{i-2}, y_{y-1}) \cdot e(x_i \mid y_i)$$
Viterbi

• Build a dynamic programming table for the highest tag sequence probability ending in a tag bigram at position $k$

\[ \pi(k, u, v) = \max \{ y_1, \ldots, y_k : y_{k-1} = u, y_k = v \} \]

• $\pi(7, P, D)$:

```
the man saw the dog with the telescope
```
Viterbi algorithm

• Base:

  • \( \pi(0, *, *) = 1 \) (all tag sequences begin with those)

• Recursively:

\[
\text{for } k \in \{1 \ldots n\}, \text{ for all } u \in S_{k-1}, v \in S_k : \\
\pi(k, u, v) = \max_{w \in S_{k-2}} \pi(k - 1, w, u) \times q(v \mid w, u) \times e(x_k \mid v)
\]
Correctness: induction on $k$

The man saw the dog with the telescope

Base: $\pi(0, *, *) = 1$

Assume $\pi(t, u, v)$ is correct for all $t < k$ and show it is correct for $k$

$$\pi(k, u, v) = \arg\max_w \left[ \pi(k-1, u, v) q(v \mid w, u) e(x_k \mid v) \right]$$

Claim: The highest probability tag sequence of length $k$ that ends with $(w,u,v)$ contains the highest probability tag sequence of length $k-1$ that ends with $(w,u)$.

Proof: Assume the highest probability sequence $y^*$ of length $k$ that ends with $(w,u,v)$ does not contain the highest probability tag sequence of length $k-1$ that ends with $(w,u)$. Then, we can construct a sequence $y_{\text{new}}$ that ends with $(w,u,v)$ with higher probability by adding the tag $v$ and then the first term will be higher and the others are unchanged. This contradicts the fact that $y^*$ is the highest probability sequence.

Corollary: $\pi(k, u, v)$ because we go over all possible $w$'s
Example

the man saw the dog with the telescope

\[ \pi(7, P, DT) = \max_{w \in \{DT, P, N, V\}} \pi(6, w, P) \times q(DT | w, P) \times e(\text{the} | DT) \]
Viterbi with back pointers

**Input:** a sentence $x_1, \ldots, x_n$, parameters $q, e$, and tag set $S$

**Base case:** $\pi(0, *, *) = 1$

**Definition:** $S_{-1} = S_0 = \{*\}, S_k = S$ for $k \in \{1 \ldots n\}$

**Algorithm:**

for $k \in \{1 \ldots n\}, u \in S_{k-1}, v \in S_k$:

$$\pi(k, u, v) = \max_{w \in S_{k-2}} \pi(k - 1, w, u) \times q(v \mid w, u) \times e(x_k \mid v)$$

$$bp(k, u, v) = \arg \max_{w \in S_{k-2}} \pi(k - 1, w, u) \times q(v \mid w, u) \times e(x_k \mid v)$$

$$(y_{n-1}, y_n) = \arg \max_{u, v} (\pi(n, u, v) \times q(\text{STOP} \mid u, v))$$

for $k = (n - 2) \ldots 1, y_k = \text{bp}(k + 2, y_{k+1}, y_{k+2})$

return $y_1, \ldots, y_n$
Viterbi runtime and memory

• Memory:
  • Dynamic programming table: $O(n|S|^2)$

• Runtime:
  • For every position and tag pair do max over all tags: $O(n|S|^3)$
  • Linear in $n$ as opposed to exponential
  • Correct
Popping up

• Training:
  • estimate parameters $q$ and $e$ from training data

• Test:
  • Apply Viterbi
Summary

• HMMs are simple to train: count stuff
• Reasonable performance for POS tagging
  • 96.46% token accuracy on WSJ (link)
• Unknown words can be troublesome