Natural Language Processing

Syntactic Parsing

Based on slides from Yoav Goldberg, Jason Eisner, Michael Collins, Shuli Wintner
Tagging wrap-up

- HMMs:
  - Training: counting
  - Decoding: Viterbi
Tagging wrap-up

• Log-linear models:
  • Motivation: richer features
  • Use (history, tag) pairs for training

\[ p_\theta(y \mid x) = \frac{e^{f(x,y)^\top \theta}}{\sum_{y' \in \mathcal{Y}} e^{f(x,y')^\top \theta}} \]
Tagging wrap-up

• Given training set of sentences and tag sequences $w^{(j)}, t^{(j)} \ j = 1 \ldots n$

• Build a training set of the form $x^{(i)}, y^{(i)}$ by constructing all history/label pairs.

• Maximize the L2-regularized maximum-likelihood estimate

• Gradient is the difference between the empirical expected feature vector and model expected feature vector

$$L(\theta) = \sum_{i=1}^{n} \log p_{\theta}(y^{(i)} | x^{(i)}) + \lambda \cdot ||\theta||^2$$
Tagging wrap-up

• Decoding:
  • If features don’t decompose:
    • Greedy
    • Beam-search

Fed raises interest rates 0.5 percent
Tagging wrap-up

• Decoding:
  • If features decompose: Viterbi
Tagging wrap-up

- Globally-normalized models:

  - Motivation:
    - Optimize what you care about
    - More flexibility in defining features (to be seen)
    - Under some conditions more expressive model (label bias)

\[
p_\theta(y \mid x) = \frac{\exp(f(x, y)\top \theta)}{\sum_{y' \in \mathcal{Y}} \exp(f(x, y')\top \theta)}
\]

\(\mathcal{Y} : \) all tag sequences for \(x\)
Training data:

\[
\begin{align*}
ABC & \quad ADE \\
abc & \quad abe
\end{align*}
\]

\[
q(x_i, y_i, \ldots, y_i) = \alpha \times \prod (x_i, y_i) \in \{AB, BC, AD, DE\} + \alpha \times \sum (x_i, y_i) \in \{AA, BB, CC, DD, EE\}
\]

\[
\mathbb{P}_\alpha (ABC | abc) = \frac{\exp(5\alpha)}{\sum \exp(5\alpha)} = \text{softmax}(5\alpha) \approx \alpha
\]

\[
\lim_{\alpha \to 0} \mathbb{P}_\alpha (ABC | abc) = 1
\]

For any definition of \( q(x_i, y_i, \ldots, y_i) = q(x_i, y_i) \):

\[
\begin{align*}
\mathbb{P}_L (ABC | abc) & \leq \mathbb{P}_L (A | a) \times \mathbb{P}_L (B | A, a, b) \times \mathbb{P}_L (C | A, B, a, b, c) \\
\mathbb{P}_L (ADE | ade) & \leq \mathbb{P}_L (A | a) \times \mathbb{P}_L (D | A, a, b) \times \mathbb{P}_L (E | A, B, a, b, e)
\end{align*}
\]

\[
\begin{align*}
\mathbb{P}_L (ABC | abc) + \mathbb{P}_L (ADE | ade) & \leq \mathbb{P}_L (B | A, a, b) + \mathbb{P}_L (D | A, a, b) \leq 1
\end{align*}
\]

on the other hand for large enough \( \alpha \):

\[
\begin{align*}
\mathbb{P}_\alpha (ABC | abc) + \mathbb{P}_\alpha (ADE | ade) > 1
\end{align*}
\]
Tagging wrap-up

- Globally-normalized models
  - A feature function $f$ mapping a pair $(x,y)$ to a feature vector $f(x,y)$
  - A generating function $GEN$ enumerating all candidate outputs
  - A parameter vector $\theta$

$$F(x) = \arg \max_y f(x, y)^\top \theta$$
Tagging wrap-up

\[ p_\theta(y \mid x) = \frac{\exp(f(x, y)^\top \theta)}{\sum_{y' \in \text{GEN}(x)} \exp(f(x, y')^\top \theta)} \]

\[ = \frac{\exp(\sum_i g(x, i, y_{i-2}, y_{i-1}, y_i)^\top \theta)}{\sum_{y' \in \text{GEN}(x)} \exp(\sum_i g(x, i, y'_{i-2}, y'_{i-1}, y'_i)^\top \theta)} \]

- Decoding: if features decompose can still use Viterbi
Tagging wrap-up

• Main difference: learning

\[ L(\theta) = \sum_i \log p_\theta(y_i \mid x_i) \]

\[ \nabla L(\theta)_i = f(x_i, y_i) - \sum_{y'} p_\theta(y' \mid x)f(x, y') \]

How to compute the second term?
Tagging wrap-up

- Let’s look at bigram features only

\[
\sum_{y} p_\theta(y \mid x) f(x, y) = \sum_{y} \sum_{i} p_\theta(y \mid x) g(x, i, y_{i-1}, y_i)
\]

\[
= \sum_{i} \sum_{a,b} \sum_{y:y_{i-1}=a,y_i=b} p_\theta(y \mid x) g(x, i, y_{i-1}, y_i)
\]

\[
= \sum_{i} \sum_{a,b} g(x, i, a, b) \sum_{y:y_{i-1}=a,y_i=b} p_\theta(y \mid x)
\]

\[
= \sum_{i} \sum_{a,b} g(x, i, a, b) q_i(a, b)
\]

- q terms are the probability that the tag sequence has a and b in positions i-1, i
Tagging wrap-up

• Deep models
  • Greedy taggers
    • Arbitrary feed-forward network
  • Viterbi taggers
    • Arbitrary feed-forward network over current and limited horizon tags
Tagging wrap-up

• Compute the above with forward-backward
  • analogous to Viterbi with a backward pass, and replacing max with sum
Tagging wrap-up

He/N eats/V

He eats|1|N
He eats|1|V
He eats|2|NN
He eats|2|NV
He eats|2|VV
He eats|2|VN

\[ \alpha(1, N) \psi(2, N, N) \]
\[ \alpha(1, V) \psi(2, V, N) \]
\[ \alpha(1, N) \psi(2, N, V) \]
\[ \alpha(1, V) \psi(2, V, V) \]

\[ \alpha(2, N) \]
\[ \alpha(2, V) \]

\[ \text{score(He eats|NV)} \]

\[ \log P(N \, V \mid \text{He eats}) \]

\[ z \]
Tagging wrap-up

$x_i$: one-hot rep. for word $i$

$e_i = W^{\text{emb}} x_i$: word embedding

$h_i^f = \text{LSTM}(h_{i-1}, e_i)$

$h_i^b = \text{LSTM}(h_{i+1}, e_i)$

$c_i = \sigma(W[h_i^f; h_i^b] + b)$

$p(y_i | x) = \text{softmax}(W^s c_i + b)$

$L(\theta) = -\sum_i \log p(y_i = y^* | x)$

- All tag decisions are independent!! Works OK for POS tagging
  - 96.97 acc. compared to 97.32 with a linear CRF
  - Doesn’t work for NER tagging. Why?

Ling et al., 2015
Tagging wrap-up

$x_i$: one-hot rep. for word $i$

$e_i = W^{emb} x_i$: word embedding

$l_i = \text{LSTM}(l_{i-1}, e_i)$

$r_i = \text{LSTM}(r_{i+1}, e_i)$

$c_i = [l_i; r_i]$

$p_i = W^{(proj)} \sigma(Wc_i + b)$

$s(x, y) = \sum_{i=1}^{n} p_{i,y_i} + \sum_{i=0}^{n} A_{y_i,y_{i+1}}$

- $A$ is a learned parameter matrix ($\#\text{tags}^2$).
- SOTA or close to that on POS tagging, NER, and other tasks
So far

- Word vectors
- Language modeling
  - n-gram models
  - neural models
- Tagging
  - HMMs
  - locally-normalized linear models
  - globally-normalized linear models
- Deep learning models
Future

• Parsing: trees over sentence

• Generation: generate text/structure conditioned on textual input
Plan for today - syntax

- Grammars
- Parsing
- Context-free grammars
- The syntax of English
Grammars
What are grammars?

A set of structural rules governing the composition of clauses, phrases, and words in any given natural language...

- **Formalism**
  - A method for describing the structure of language (CFG, TAG, HPSG, LFG, …)

- **Instance**
  - An implementation of a formalism in a particular language
  - Defines the infinite set of grammatical sentences
John saw Mary.
I ate sushi with tuna.

I want you to go there.
Did you go there?

I ate the cake that John had made for me yesterday.

John made some cake.

Did you went there?

...
Sequence model?

Subject → Verb → Object

{l, you, we, …} → {eat, drink, …} → {sushi, pizza, …}
Sequence model?

Subject \rightarrow Verb \rightarrow Object

\{I, you, we, \ldots\} \rightarrow \{eat, drink, \ldots\} \rightarrow \{sushi, pizza, \ldots\}

Undergeneration: I sleep
Sequence model?

Subject → Verb → Object

{l, you, we, ...} → {eat, drink, ...} → {sushi, pizza, ...}

Undergeneration: I sleep
Overgeneration: I sleep sushi, I drink pizza
Sequence model?

Undergeneration: *I sleep*
Overgeneration: *I sleep sushi, I drink pizza*

- **Subcategorization**: different verbs take a different number of arguments
- **Selectional preference**: verbs take certain types for semantic arguments
Sequence model?

{sleep, yawn, ...}

Intrans. verb

Subject

{l, you, we, ...} {eat, drink, ...} {sushi, pizza, ...}

Trans. verb

Object
Language is recursive

the ball
the big ball
the big red ball
the big red heavy ball

Nouns can take an infinite number of adjectives
Language is recursive

the ball
the big ball
the big red ball
the big red heavy ball

Nouns can take an infinite number of adjectives

*English is weird about adjective order (opinion, size, shape, age, color, nationality material)
  • the big round old red house
  • the red old round big house
Recursive sequence model

Det. → Adj. → Noun

{the, a, …} → {red, big, …} → {house, chair, …}
Hierarchical recursive structure

\[ \text{the cat likes tuna} \]
\[ \text{the cat the dog chased likes tuna} \]
\[ \text{the cat the dog the rat bit chased likes tuna} \]
\[ \text{the cat the dog the rat the elephant admired bit chased likes tuna} \]

\[ a^n b^n \] construction (not a regular language)

Competence vs. performance (Chomsky):
- Competence: idealized capacity
- Performance: what we actually utter

Example from Shuli Wintner
Context-free?

Chomsky (1957):
“English is not a regular language. As for context-free languages, I do not know whether or not English is itself literally outside the range of such analyses”

In swiss-german there are constructions like:
\[(\text{acc-noun})^n (\text{dative-noun})^m (\text{acc-verb})^n (\text{dative-verb})^m\]
Language has structure

- Subjects asked to memorize sentences
- Probability of error related to phrase structure

Conclusion: our representation will be hierarchical
Strong vs. weak capacity

- Formal language theory:
  - Language is a set of strings
  - We care about generating the right set
- Formal syntax:
  - Language is a set of strings with structure
  - We care about strings having the right structure
Constituency vs. dependency

Constituency: words are leaves with part-of-speech tags as parents. Other nodes are syntactic categories.

Dependency: All nodes are words. Each word is a modifier to a single head.
Parsing
Goal

• Input: sentence
  • *Fruit flies like a banana*

• Output: constituency parse tree

• Method: supervised learning
  • Given (x,y) pairs of sentences and parse trees, learn a mapping from sentences to parse trees
What is it good for?

Information extraction

- trained: exiles, agent, fighters, army, issues, Farm, facilities
- used: buildings, dealers, seal, missile, methods, processes
- provided: proof, lists, leftists, communists, Factbook, training

Question answering

- Tokyo was hit by powerful earthquakes in 1703, 1782, 1812, 1855 and 1923.
- The 2004 Indian Ocean earthquake. The European nation hardest hit may have been Sweden, whose death toll was 543. ... The deadliest earthquakes since 1900 were the Tangshan, China earthquake of 1976, in which at least 255,000 were killed; the earthquake of 1927 in Xining, Qinghai, China (200,000); the Great Kanto earthquake which struck Tokyo in 1923 (143,000); and the Gansu, China, earthquake of 1920 (200,000).
What is it good for?

Summarization/simplification

The first new product, ATF Protype, is a line of digital postscript typefaces that will be sold in packages of up to six fonts.

ATF Protype is a line of digital postscript typefaces that will be sold in packages of up to six fonts.
What is it good for?

**Machine translation**: re-ordering of parse trees for English-Japanese translation

[SUBJECT] + TIME + PLACE/IMPLEMENT + INDIRECT OBJECT + OBJECT + ACTION VERB

*Sources said that IBM bought Lotus yesterday*

*Sources yesterday IBM Lotus bought that said*
Why is it hard?

• Real sentences are long:

"Former Beatle Paul McCartney today was ordered to pay nearly $50M to his estranged wife as their bitter divorce battle came to an end."

"Welcome to our Columbus hotels guide, where you’ll find honest, concise hotel reviews, all discounts, a lowest rate guarantee, and no booking fees."
Why is it hard?

- **Ambiguities**: prepositional attachment
Why is it hard?

- **Ambiguities:**

  ![Diagram of sentence structure]
Why is it hard?

• Ambiguities:
  
  • *She announced a program to promote safety in trucks and vans*
Context-free grammars
Context-free grammars

A context-free grammar (CFG) is a 4-tuple $G = (N, \Sigma, R, S)$:

- $N$ is a set of non-terminal symbols
- $\Sigma$ is a set of terminal symbols
- $R$ is a set of rules $X \rightarrow Y_1 Y_2 \ldots Y_n$, $n \geq 0$, $X \in N, Y_i \in N \cup \Sigma$
- $S \in N$ is a special start symbol
Example

\( N = \{S, \text{NP}, \text{VP}, \text{PP}, \text{DT}, \text{Vi}, \text{Vt}, \text{NN}, \text{IN}\} \)

\( \Sigma = \{\text{sleeps, saw, man, woman, telescope, the, with, in}\} \)

R:

\[
\begin{array}{llllll}
S & \rightarrow & \text{NP VP} & \text{Vi} & \rightarrow & \text{sleeps} \\
\text{VP} & \rightarrow & \text{Vi} & \text{Vt} & \rightarrow & \text{saw} \\
\text{VP} & \rightarrow & \text{Vt NP} & \text{NN} & \rightarrow & \text{man} \\
\text{VP} & \rightarrow & \text{VP PP} & \text{NN} & \rightarrow & \text{woman} \\
\text{NP} & \rightarrow & \text{DT NN} & \text{NN} & \rightarrow & \text{telescope} \\
\text{NP} & \rightarrow & \text{NP PP} & \text{DT} & \rightarrow & \text{the} \\
\text{PP} & \rightarrow & \text{IN NP} & \text{IN} & \rightarrow & \text{with} \\
\end{array}
\]
Left-most derivations

• Sequence of strings $s_1 \ldots s_n$, where
  • $s_1 = S$
  • $s_n$ is a string in $\Sigma^*$
  • Each $s_i$ for $i=2 \ldots n$ is derived from $s_{i-1}$ by picking the left-most non-terminal $X$ in $s_{i-1}$ and replacing it by some $\beta$ where $X \rightarrow \beta$ is a rule in $R$
## Example

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$ $S$</td>
<td>$S \rightarrow NP \ VP$</td>
</tr>
<tr>
<td>$s_2$ $NP \ VP$</td>
<td>$NP \rightarrow DT \ NN$</td>
</tr>
<tr>
<td>$s_3$ $DT \ NN \ VP$</td>
<td>$DT \rightarrow$ the</td>
</tr>
<tr>
<td>$s_4$ the $NN \ VP$</td>
<td>$NN \rightarrow$ dog</td>
</tr>
<tr>
<td>$s_5$ the dog $VP$</td>
<td>$VP \rightarrow VB$</td>
</tr>
<tr>
<td>$s_6$ the dog $VBZ$</td>
<td>$VBZ \rightarrow$ laughs</td>
</tr>
<tr>
<td>$s_7$ the dog laughs</td>
<td></td>
</tr>
</tbody>
</table>
## Example

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>$S \rightarrow NP \ VP$</td>
</tr>
<tr>
<td>$s_2$</td>
<td>$NP \rightarrow DT \ NN$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$DT \ NN \ VP \rightarrow DT \ the$</td>
</tr>
<tr>
<td>$s_4$</td>
<td>$the \ NN \ VP \rightarrow NN \rightarrow dog$</td>
</tr>
<tr>
<td>$s_5$</td>
<td>$the \ dog \ VP \rightarrow VP \rightarrow VB$</td>
</tr>
<tr>
<td>$s_6$</td>
<td>$the \ dog \ VBZ \rightarrow VBZ \rightarrow laughs$</td>
</tr>
<tr>
<td>$s_7$</td>
<td>$the \ dog \ laughs$</td>
</tr>
</tbody>
</table>

![Diagram of tree structure](image-url)
Properties of CFGs

• A CFG defines a set of derivations

• A string is in the language if there is a derivations that yields it

• Ambiguity is when the same string can be derived in multiple ways with left-most derivations
The syntax of English
Product Details (from Amazon)
Hardcover: 1779 pages
Publisher: Longman; 2nd Revised edition
Language: English
ISBN-10: 0582517346
Product Dimensions: 8.4 x 2.4 x 10 inches
Shipping Weight: 4.6 pounds
English syntax

• Parts-of-speech (saw that already)
Noun phrase grammar

| Ñ | → | NN | NN | → | box |
| Ñ | → | NN Ñ | NN | → | car |
| Ñ | → | JJ Ñ | NN | → | mechanic |
| Ñ | → | Ñ Ñ | NN | → | pigeon |
| NP | → | DT Ñ | DT | → | the |
| | | | | → | a |
| | | | | → | fast |
| | | | | → | metal |
| | | | | → | idealistic |
| | | | | → | clay |

We can generate:
- *the car, the fast car, the fast metal car*
- *the car mechanic, the fast car mechanic*
We can generate:

- *the fast car mechanic under the pigeon in the box*
Verbs, verb phrases and sentences

• Verb types
  • Vi: intransitive verbs (*sleeps*, *walks*, *yawns*)
  • Vt: transitive verbs (*see*, *like*, *hug*, *kiss*)
  • Vd: ditransitive verbs (*give*, *send*)

• VP rule
  • VP —> Vi
  • VP —> Vt NP
  • VP —> Vd NP NP

• Sentence rule
  • S —> NP VP

*The dog gave the mechanic the fast car*
PPs modifying verb phrases

- VP $\rightarrow$ VP PP
  - *sleeps in the car, walks like the mechanic, gave the mechanic the fast car on Tuesday*
Complementizer and SBARs

- COMP $\rightarrow$ that $|$ which $|$ ...

- SBAR $\rightarrow$ COMP S

  - that the man sleeps, that the mechanic saw the dog
More verb types

- $V[5] \rightarrow$ said | reported
- $V[6] \rightarrow$ told | informed
- $V[7] \rightarrow$ bet

- $VP \rightarrow V[5] \text{ SBAR}$
- $VP \rightarrow V[6] \text{ NP SBAR}$
- $VP \rightarrow V[7] \text{ NP NP SBAR}$

- *said that the man sleeps*
- *told the dog that the mechanic likes the pigeon*
- *bet the pigeon 50$ that the mechanic owns a fast car*
Coordination

- CC $\rightarrow$ and | or | but | ...
- NP $\rightarrow$ NP CC NP
- Ñ $\rightarrow$ Ñ and Ñ
- VP $\rightarrow$ VP CC VP
- S $\rightarrow$ S CC S
- SBAR $\rightarrow$ SBAR CC SBAR
There’s more…

• Agreement
  
  • *the dog laughs* vs. *the dogs laugh*

• Wh-movement
  
  • *The dog that the cat liked* ____

• Active vs. passive
  
  • *the dog saw the cat* vs. *the cat was seen by the dog*