Deep Learning and Lexical, Syntactic and Semantic Analysis

Wanxiang Che (HIT)
Yue Zhang (SUTD)
# Outline

<table>
<thead>
<tr>
<th>Timeline</th>
<th>Content</th>
<th>Speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00-09:30</td>
<td>1. Introduction to Tasks</td>
<td>Wanxiang Che</td>
</tr>
<tr>
<td>09:30-10:00</td>
<td>2. Deep Learning Background</td>
<td>Wanxiang Che</td>
</tr>
<tr>
<td>10:00-10:10</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>10:10-10:40</td>
<td>3. Greedy Decoding</td>
<td>Yue Zhang</td>
</tr>
<tr>
<td>10:40-11:20</td>
<td>4. Dynamic Programming Decoding</td>
<td>Wanxiang Che</td>
</tr>
<tr>
<td>11:20-12:00</td>
<td>5. Beam-search Decoding</td>
<td>Yue Zhang</td>
</tr>
</tbody>
</table>
Part 1: Introduction to Lexical, Syntactic and Semantic Analysis
NLP Pipeline

- Word Segmentation
- POS Tagging
- Named Entity Recognition
- Syntactic
- Semantic
Part 1.1: Background
Word Segmentation

• Words are fundamental semantic units
• Chinese has no obvious word boundaries
• Word segmentation
  – Split Chinese character sequence into words
• Ambiguities in word segmentation
  – E.g. 严守一把握手机关了
    • 严守一/把/手机/关/了
    • 严守/一把手/机关/了
    • 严守/一把/手机/关/了
    • 严守一/把手/机关/了
    • ……
Part-of-speech (POS) Tagging

• A POS is a category of words which have similar grammatical properties
  – E.g. noun, verb, adjective

• POS tagging
  – Marking up a word in a text as a particular POS
  – based on both its definition and its context

• Ambiguities in POS Tagging
  – Time *flies* like an arrow.
  – 制服了敌人 vs. 穿着制服
Named Entity Recognition (NER)

- Named Entities
  - Persons, locations, organizations, expressions of times, quantities, monetary values, percentages, etc
- Locating and classifying named entities in text into pre-defined categories
- Ambiguities in NER

Kerry to visit Jordan, Israel Palestinian peace on agenda.
Syntactic Parsing

• Analyzing a natural language string conforming to the rules of a formal grammar, emphasizing subject, predicate, object, etc.

  – Constituency and Dependency Parsing
Semantic Role Labeling

• Recognizing predicates and corresponding arguments
  – Yesterday time, Mary buyer bought a shirt bought thing from Tom seller
  – Whom buyer did Tom seller sell a shirt bought thing to, yesterday time

• Answer “Who did what to whom when and where”
  – Question Answering
  – Information Extraction
  – ......
Combinatory Categorial Grammars (CCG)

- **CCG Lexical Entries**
  - Pair words and phrases with meaning by a CCG category

- **CCG Categories**
  - Basic building block
  - Capture syntactic and semantic information jointly
Structured Prediction

• Predicting structured objects, rather than scalar discrete or real values
• Outputs are influenced each other
• For example
  – Sequence labeling/tagging
    • Given an input sequence, produce a label sequence of equal length. Each label is drawn from a small finite set
  – Parsing
    • Given an input sequence, build a tree whose structure obeys some grammar (compositional rules)
Part 1.2: Sequence Labeling
Sequence Labeling/Tagging

• Given an input sequence, produce a label sequence of equal length
• Each label is drawn from a small finite set
• Labels are influenced each other
• For example: POS tagging
  – Input
    • Profits soared at Boeing Co., easily topping forecasts on Wall Street, ...
  – Output
    • Profits/N soared/V at/P Boeing/N Co./N ,/, easily/ADV ...
NER

• Input
  – Profits soared at Boeing Co., easily topping forecasts on Wall Street, ...

• Output
  – Profits soared at [Boeing Co. \texttt{ORG}], easily topping forecasts on [Wall Street \texttt{LOC}], ...

• Alternative Output (Tagging)
  – Profits/\texttt{O} soared/\texttt{O} at/\texttt{O} Boeing/B-\texttt{ORG} Co./I-\texttt{ORG} ,/\texttt{O} easily/\texttt{O} topping/\texttt{O} forecasts/\texttt{O} on/\texttt{O} Wall/B-\texttt{LOC} Street/I-\texttt{LOC} ,/\texttt{O} ...

• Where
  – B: Begin of entity XXX; I: Inside of entity XXX; O: Others
Word Segmentation

• Input
  – 严守一把手机关了

• Output
  – 严守一/把/手机/关/了/

• Alternative Output (Tagging)
  – 严/B守/I一/I把/B手/B机/I关/B了/B

• Where
  – B: Begin of a word; I: Inside of a word
Semantic Role Labeling

• Input
  – Yesterday, Mary bought a shirt from Tom

• Output
  – [Yesterday \textit{time}], [Mary \textit{buyer}] bought/pred [a shirt \textit{bought thing}] from [Tom \textit{seller}]

• Alternative Output (Tagging)
  – Yesterday/B-time ,/O Mary/B-buyer bought/pred a/B-bought thing shirt/I-bought thing from/O Tom/B-seller

• Where
  – B: Begin of an arg; I: Inside of an arg; O: Others
CCG Supertagging

He goes on the road with his piano

A bitter conflict with global implications

<table>
<thead>
<tr>
<th>frequency cut-off</th>
<th># cat types</th>
<th># cat tokens in 2-21 not in cat set</th>
<th># sentences in 2-21 with missing cat</th>
<th># cat tokens in 00 not in cat set</th>
<th># sentences in 00 with missing cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 225</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>409</td>
<td>1 933 (0.2%)</td>
<td>1 712 (4.3%)</td>
<td>79</td>
<td>69</td>
</tr>
</tbody>
</table>
Sequence Labeling Models

**HMM**

\[ P(y_{1:n}, x_{1:n}) \propto \prod_{t=1}^{n} P(y_t | y_{t-1}) P(x_t | y_t) \]

**MEMM**

\[ P(y_{1:n} | x_{1:n}) \propto \prod_{t=1}^{n} P(y_t | y_{t-1}, x_t) \]

\[ \propto \prod_{t=1}^{n} \frac{1}{Z_{y_{t-1}, x_t}} \exp \left( \sum_j \lambda_j f_j(y_t, y_{t-1}) + \sum_k \mu_k g_k(y_t, x_t) \right) \]

**CRF**

\[ P(y_{1:n} | x_{1:n}) \propto \frac{1}{Z_{y_{1:n}}} \prod_{t=1}^{n} \exp \left( \sum_j \lambda_j f_j(y_t, y_{t-1}) + \sum_k \mu_k g_k(y_t, x_t) \right) \]
Features of POS Tagging with CRF

• Assume only two feature templates
  – tag bigrams
  – word/tag pairs

\[ f_{100} = \begin{cases} 
1 & \text{if } \langle y_{i-1}, y_i \rangle = \langle n, v \rangle \\
0 & \text{otherwise}
\end{cases} \]

\[ g_{101} = \begin{cases} 
1 & \text{if } x_i \text{ is ended with “ing” and } y_i = v \\
0 & \text{otherwise}
\end{cases} \]
CRF Decoding

\[
\arg \max_{\gamma[1:n] \in \text{GEN}(x[1:n])} \sum_{i=1}^{n} w \cdot f(x[1:n], y_i, y_{i-1})
\]

where \( \text{GEN}(x[1:n]) \) is all possible tag sequences

• Dynamic Programming Algorithm
  – Viterbi Algorithm
Viterbi Algorithm

- Define a dynamic programming table
  \[ \pi(i, y) = \text{maximum score of a tag sequence ending in tag } y \text{ at position } i \]
- Recursive definition: \[ \pi(i, y) = \max_t \left( \pi(i - 1, t) + w \cdot f(x_{1:n}, y, t) \right) \]
Part 1.3: Parsing
Dependency Parsing

• A dependency tree is a tree structure composed of the input words and meets a few constraints:
  – Single-head
  – Connected
  – Acyclic
Graph-based Dependency Parsing

- Find the highest scoring tree from a complete dependency graph

\[
Y^* = \arg \max_{Y \in \Phi(X)} \text{score}(X, Y)
\]
First-order as an Example

- The first-order graph-based method assumes that arcs in a tree are independent from each other (arc-factorization)
- Maximum Spanning Tree (MST) Algorithm

\[
\text{score}(X,Y) = \sum_{(h,m)\in Y} \text{score}(X,h,m)
\]
How to Score an Arc

• Given an sentence, how to determine the score of each arc?

\[ \text{score}(2,4) = ? \]

• Feature based representation: an arc is represented as a feature vector \( f(2,4) \)

\[ \text{score}(2,4) = w \cdot f(2,4) \]
Features for an Arc

* As McGwire neared, fans went wild
Decoding for first-order model

- Eisner (2000) described a *dynamic programming* based decoding algorithm for bilexical grammar
- McDonald+ (2005) applied this algorithm to the search problem of the first-order model
Transition-based Dependency Parsing

• Gradually build a tree by applying a sequence of transition actions – shift/reduce (Yamada and Matsumoto, 2003; Nivre, 2003)

• The score of the tree is equal to the summation of the scores of the actions

\[
score(X, Y) = \sum_{i=0}^{m} score(X, h_i, a_i)
\]

- \(a_i\) → the action adopted in step \(i\)
- \(h_i\) → the partial results built so far by \(a_0...a_{i-1}\)
- \(Y\) → the tree built by the action sequence \(a_0...a_m\)
Transition-based Dependency Parsing

• The goal of a transition-based dependency parser is to find the highest scoring action sequence that builds a legal tree.

\[ Y^* = \arg \max_{Y \in \Phi(X)} \text{score}(X, Y) \]

\[ = \arg \max \sum_{i=0}^{m} \text{score}(X, h_i, a_i) \]
Transition-based Dependency Parsing

- Greedily predict a transition sequence from an initial parser state to some terminal states
- State (configuration)
  = Stack + Buffer + Dependency Arcs

Stack
- ROOT
- has_VBZ
- good_JJ
- He_PRP
- nsubj

Buffer
- Control_NN
- _

Configuration

Arc Standard algorithm
- LEFT-ARC(l)
- RIGHT-ARC(l)
- SHIFT

Classifier
Transition Action: LEFT-ARC ($l$)

Configuration

Stack
ROOT He_PRP has_VBZ

Buffer
good JJ Control NN ...

Operation:
- Add a left arc ($S_0$)
- Remove “He_PRP” from Stack

He_PRP

nsbj

Configuration

Stack
ROOT has_VBZ

Buffer
good JJ Control NN ...

He_PRP
Transition Action: SHIFT

Configuration

Stack
ROOT has_VBZ
He_PRP

Buffer
good_JJ Control_NN

Operation:
- Shift "good_JJ" from Buffer to top of Stack

Configuration

Stack
ROOT has_VBZ good_JJ
He_PRP

Buffer
Control_NN

Transition Action: RIGHT-ARC (l)

Configuration

Stack
ROOT has_VBZ Control_NN
He_PRP nsubj good_JJ amod

Buffer
...

Operation:
• Add a right arc ($S_l$)
• Remove $S_0$ (“Control_NN”) from Stack

Configuration

Stack
ROOT has_VBZ
He_PRP nsubj
Control_NN dobj

Buffer
...

good_JJ amod
**Arc-standard Algorithm**

**初始状态**
Stack只有根节点，等待处理词在Buffer中

**SHIFT**
将Buffer中第一个词压入Stack

**LEFT-ARC**
弹出Stack中第二个词，生成一条弧从栈顶词指向第二个词

**RIGHT-ARC**
弹出栈顶词，生成一条弧从栈顶第二个词指向栈顶词

**终结状态**
Stack只有根节点，Buffer为空

<table>
<thead>
<tr>
<th>Stack</th>
<th>Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT</td>
<td>小明_nr 吃_v 苹果_n</td>
</tr>
<tr>
<td>ROOT 小明_nr</td>
<td>v</td>
</tr>
<tr>
<td>吃_v 苹果_n</td>
<td></td>
</tr>
</tbody>
</table>

CCL 2016 Tutorial 2016-10-14
Traditional Features

Configuration

Stack

Buffer

ROOT has_VBZ good_JJ

nsbj

He_PRP

Control_NN ...

Feature Vector:
- Binary
- Sparse
- High-dimensional

Feature templates: a combination of elements from the configuration.
- For example: (Zhang and Nivre, 2011): 72 feature templates

Table 1: Baseline feature templates.

<table>
<thead>
<tr>
<th>distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_{0wp}; S_{0wp}; N_{0wp}; N_{0wp}; )</td>
</tr>
<tr>
<td>( S_{0wp}; N_{0wp}; S_{0wp}; N_{0wp}; )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>valency</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_{0wv}; S_{0wp}; S_{0wv}; S_{0wp}; N_{0wv}; N_{0wp}; )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>unigrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_{0w}; S_{0wp}; S_{0w}; S_{0wp}; S_{0w}; S_{0wp}; N_{0wp}; N_{0wp}; )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>third-order</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_{0wp}; S_{0wp}; S_{0wp}; S_{0wp}; S_{0wp}; S_{0wp}; S_{0wp}; S_{0wp}; )</td>
</tr>
<tr>
<td>( S_{0wp}; S_{0wp}; S_{0wp}; S_{0wp}; N_{0wp}; N_{0wp}; N_{0wp}; )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>label set</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_{0w}v; S_{0wp}; S_{0w}v; S_{0wp}; N_{0w}; N_{0wp}; )</td>
</tr>
</tbody>
</table>

Table 2: New feature templates.

<table>
<thead>
<tr>
<th>distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w; p )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>valency</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_{i}, v_{r} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>third-order</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>label set</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_{t}, s_{r} )</td>
</tr>
</tbody>
</table>
Constituency Parsing

S

VP

NP

NP

NP

NN VBD JJ NN IN JJ NNS

Economic news had little effect on financial markets.
Constituency Parsing

• Chart-based
  – E.g. Cocke–Younger–Kasami algorithm (CYK or CKY)
  – A kind of Dynamic Programming

PCFG

Rule Prob $\theta_i$

$S \rightarrow NP \ VP \quad \theta_0$

$NP \rightarrow NP \ NP \quad \theta_1$

$\cdots$

$N \rightarrow fish \quad \theta_{42}$

$N \rightarrow people \quad \theta_{43}$

$V \rightarrow fish \quad \theta_{44}$
CKY Parsing Algorithm

Input: a sentence \( s = x_1 \ldots x_n \), a PCFG \( G = (N, \Sigma, S, R, q) \).

Initialization:
For all \( i \in \{1 \ldots n\} \), for all \( X \in N \),

\[
\pi(i, i, X) = \begin{cases} q(X \to x_i) & \text{if } X \to x_i \in R \\ 0 & \text{otherwise} \end{cases}
\]

Algorithm:

- For \( l = 1 \ldots (n - 1) \)
  - For \( i = 1 \ldots (n - l) \)
    * Set \( j = i + l \)
    * For all \( X \in N \), calculate

\[
\pi(i, j, X) = \max_{X \to YZ \in R, \atop s \in \{i \ldots (j-1)\}} (q(X \to YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
\]

and

\[
bp(i, j, X) = \arg \max_{X \to YZ \in R, \atop s \in \{i \ldots (j-1)\}} (q(X \to YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))
\]

Output: Return \( \pi(1, n, S) = \max_{t \in T(s)} p(t) \), and backpointers \( bp \) which allow recovery of \( \arg \max_{t \in T(s)} p(t) \).
### Summarization

- **Classical NLP Methods**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Model</th>
<th>Decoding</th>
<th>NLP Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Labeling</td>
<td>CRF</td>
<td>Dynamic Programming</td>
<td>POS tagging, Word Segmentation, NER, SRL, CCG Supertagging</td>
</tr>
<tr>
<td>Parsing</td>
<td>Transition-based</td>
<td>Greedy/Beam Search</td>
<td>Dependency Parsing</td>
</tr>
<tr>
<td></td>
<td>Graph-based</td>
<td>Dynamic Programming</td>
<td></td>
</tr>
<tr>
<td>Constituency</td>
<td>Chart-based</td>
<td></td>
<td>Constituency Parsing</td>
</tr>
</tbody>
</table>

Lots of feature engineering work!