Shallow Parsing

Swapnil Chaudhari – 11305R011
Ankur Aher - 113059006
Raj Dabre – 11305R001
Purpose of the Seminar

• To emphasize on the need for Shallow Parsing.
• To impart basic information about techniques to perform Shallow Parsing.
• To emphasize on the power CRF’s for Sequence Labeling tasks.
Overall flow

• Motivation
• Introduction to Shallow Parsing
  • Overview and limitations of approaches
• Conditional Random Fields
  • Training
  • Decoding
• Experiments
  • Terminology
  • Results
• Conclusions
A quick refresher

• Identify the Core Noun Phrases:
  • He reckons the current account deficit will narrow to only 1.8 billion in September.
  • [NP He] reckons [NP the current account deficit] will narrow to only [NP 1.8 billion] in [NP September]
Flow in MT

- Discourse and Coreference
- Semantics Extraction
- Parsing
- Chunking
- POS tagging
- Morphology

Taken from NLP slides by Pushpak Bhattacharyya, IITB.
Motivation

• The logical stage after POS tagging is Parsing.
• Constituents of sentences are related to each other.
• Relations identification helps move towards meaning.
• Identification of Base Phrases precedes Full Parsing.
• Closely related languages need only shallow parsing.
• Also crucial to IR
  • [Picnic locations] in [Alibaug].
  • [Details] of [Earthquake] in [2006].
  • [War] between [USA] and [Iraq].
Introduction

• Shallow parsing is the process of identification of the non-recursive cores of various phrase types in given text.
• Also called Chunking.
• Most basic work is on NP Chunking.
• Results in a flat structure.
  • “Ram goes to the market” is chunked as
  • [Ram-NP] [goes-VG] [to-PP] [the market-NP]
• Can move towards Full Parsing and then Transfer for translation purposes.
Identification of Problem

- Task is Sequence Labeling i.e. to identify labels for constituents, whether in or out of Chunk.
- Also can be sequence of classification problems, one for each of the labels in the sequence (SVM).
- 2 Major Approaches:
  - Generative Models
  - Discriminative Models
Generative v.s. Discriminative Models

- In generative models we model Joint probability $P(X,Y)$ which contains $P(X|Y)$ and $P(Y)$ where $Y$=Label Seq. $X$=Obs. Seq.
- Model(The labels) Generates Observations in Generative Models.
- So given Observations one can get Labels.
- Easy to model.
- No inclusion of features.
- Even less effective when data is sparse but can use EM algorithm.
- Example is HMM.
Generative v.s. Discriminative Models

- In Discriminative Models we model $P(Y|X)$ directly.
- The features distinguish between labels and a weighted combination of various features can decide them.
- Difficult to model (Decision of Features).
- Overall more effective when sparse data.
- Examples are MEMM and CRF.
Motivation: Shortcomings of Hidden Markov Model

- HMM models direct dependence between each state and only its corresponding observation.
  - \( P(X|Y) \) is decomposed into product of all \( P(x_i|y_i) \) for all positions \( i \) in input.
  - This independence assumption is for simplification but it ignores potential relations between neighboring words.
- HMM learns a joint distribution of states and observations \( P(Y, X) \), but in a prediction task, we need the conditional probability \( P(Y|X) \)
Solution:
Maximum Entropy Markov Model (MEMM)

- Models dependence between each state and the full observation sequence explicitly
  - More expressive than HMMs
- Discriminative model
  - Completely ignores modeling $P(X)$: saves modeling effort
  - Learning objective function consistent with predictive function: $P(Y|X)$

$$P(y_{1:n}|x_{1:n}) = \prod_{i=1}^{n} P(y_i|y_{i-1}, x_{1:n}) = \prod_{i=1}^{n} \frac{\exp(w^T f(y_i, y_{i-1}, x_{1:n}))}{Z(y_{i-1}, x_{1:n})}$$
MEMM: Label bias problem

What the local transition probabilities say:

- State 1 almost always prefers to go to state 2
- State 2 almost always prefer to stay in state 2
MEMM: Label bias problem

Probability of path 1-> 1-> 1-> 1:
- $0.4 \times 0.45 \times 0.5 = 0.09$
MEMM: Label bias problem

Observation 1  Observation 2  Observation 3  Observation 4
State 1
State 2
State 3
State 4
State 5

Probability of path 2->2->2->2:
• 0.2 \times 0.3 \times 0.3 = 0.018

Other paths:
1-> 1-> 1-> 1: 0.09
MEMM: Label bias problem

Observation 1     Observation 2     Observation 3     Observation 4
State 1          0.4               0.45              0.5
State 2          0.2       0.6      0.2      0.3
State 3          0.2       0.3      0.1      0.3
State 4          0.2       0.1      0.2      0.2
State 5          0.2

Probability of path 1->2->1->2:
• $0.6 \times 0.2 \times 0.5 = 0.06$

Other paths:
1->1->1->1: 0.09
2->2->2->2: 0.018
MEMM: Label bias problem

![Graph of states and observations](image)

Probability of path 1->1->2->2:
- \(0.4 \times 0.55 \times 0.3 = 0.066\)

Other paths:
- 1->1->1->1: 0.09
- 2->2->2->2: 0.018
- 1->2->1->2: 0.06
MEMM: Label bias problem

Most Likely Path: 1-> 1-> 1-> 1

- Although locally it seems state 1 wants to go to state 2 and state 2 wants to remain in state 2.
MEMM: Label bias problem

Most Likely Path: 1-> 1-> 1-> 1

- State 1 has only two transitions but state 2 has 5:
  - Average transition probability from state 2 is lower
MEMM: Label bias problem

Label bias problem in MEMM:
- Preference of states with lower number of transitions over others
Interpretation

• States with lower number of outgoing transitions effectively ignore observations.
• The normalization is done upto the current position and not over all positions.
• Thus uncertainties in tagging in previous locations will affect the current decision.
• Thus a global normalizer is needed.
Conditional Random Fields

- CRFs define conditional probability distributions $p(Y|X)$ of label sequences given input sequences.
  - $X$ and $Y$ have the same length.
  - $x = x_1...x_n$ and $y = y_1...y_n$
  - $f$ is a vector of local features and a corresponding weight vector $\lambda$
  - Markovian assumption needed

$$p(Y_i|\{Y_j\}_{j \neq i}, X) = p(Y_i|Y_{i-1}, Y_{i+1}, X)$$
Conditional Random Fields

\[
P(y_{1:n}|x_{1:n}) = \frac{1}{Z(x_{1:n})} \prod_{i=1}^{n} \phi(y_i, y_{i-1}, x_{1:n}) = \frac{1}{Z(x_{1:n})} \prod_{i=1}^{n} \exp(w^T f(y_i, y_{i-1}, x_{1:n}))
\]

- CRF is a partially directed model
  - Discriminative model like MEMM
  - Usage of global normalizer \( Z(x) \) overcomes the label bias problem of MEMM
  - Models the dependence between each state and the entire observation sequence (like MEMM)
Some rewriting

- Let us have some easier to write notations.
  - $Z(x_{1:n})$ as $Z_\lambda(x)$
  - $f(y_i,y_{i-1},x_{1:n},i)$ as $f(y,x,i)$
  - $P(y_{1:n},x_{1:n})$ as $P_\lambda(Y|X)$
  - $w$ as $\lambda$. 
Conditional Random Fields

- The global feature vector is where $i$ is the input position
- Combine $f$ and $g$ (transition and state features) into one.
- The conditional probability distribution defined by the CRF is then

$$F(y, x) = \sum_i f(y, x, i)$$

$$p_\lambda(Y | X) = \frac{\exp \lambda \cdot F(Y, X)}{Z_\lambda(X)}$$

where

$$Z_\lambda(x) = \sum_y \exp \lambda \cdot F(y, x)$$
Training the CRF

- We train a CRF by maximizing the log-likelihood of a given training set \( T = \{x_k, y_k\} \) for \( k=1 \) to \( n \) i.e. for all training pairs.
- The log likelihood is:

\[
\mathcal{L}_\lambda = \sum_k \log p_\lambda(y_k | x_k) = \sum_k [\lambda \cdot F(y_k, x_k) - \log Z_\lambda(x_k)]
\]
Solving the optimization problem

• We seek the zero of the gradient.

\[ \nabla L_\lambda = \sum_k \left[ F(y_k, x_k) - E_{p_\lambda}(Y|x_k)F(Y, x_k) \right] \]

• \( E_{p_\lambda}(Y|x)F(Y, x) \) is the feature expectation.

• Since number of features can be too large we do dynamic programming.
CRF learning

- Computing marginals using junction-tree calibration:

- Junction Tree Initialization.
Solving the optimization problem

- For a given $x$, define the transition matrix for position $i$ as:
  \[ M_i[y, y'] = \exp \lambda \cdot f(y, y', x, i). \]
- Let $f$ be any local feature, $f_i[y, y'] = f(y, y', x, i)$.
- $F(y, x) = \sum_i f(y_{i-1}, y_i, x, i)$ and let $\ast$ denote component-wise matrix product.

\[
E_{p_\lambda(Y|x)} F(Y, x) = \sum_y p_\lambda(y|x) F(y, x) \\
= \sum_i \frac{\alpha_{i-1} (f_i \ast M_i) \beta_i^T}{Z_\lambda(x)} \\
Z_\lambda(x) = \alpha_n \cdot 1^T
\]
Solving the optimization problem

- where \( \alpha_i \) and \( \beta_i \) the forward and backward state-cost vectors defined by

\[
\alpha_i = \begin{cases} 
\alpha_{i-1}M_i & 0 < i \leq n \\
1 & i = 0
\end{cases}
\]

\[
\beta_i^T = \begin{cases} 
M_{i+1}\beta_{i+1}^T & 1 \leq i < n \\
1 & i = n
\end{cases}
\]

- We can use forward-backward algorithms to calculate \( \alpha_i \) and \( \beta_i \) to get feature expectations.
Avoiding over fitting

• To avoid over fitting, we penalize the likelihood as:

\[
\mathcal{L}_\lambda' = \sum_k [\lambda \cdot F(y_k, x_k) - \log Z_\lambda(x_k)] - \frac{\|\lambda\|^2}{2\sigma^2} + \text{const}
\]

with gradient

\[
\nabla \mathcal{L}_\lambda' = \sum_k [F(y_k, x_k) - E_{p_\lambda(Y|x_k)}F(Y, x_k)] - \frac{\lambda}{\sigma^2}
\]
Decoding the label sequence

- The most probable label sequence for input sequence $x$ is
  
  $$
  \hat{y} = \arg \max_y p_\lambda(y|x) = \arg \max_y \lambda \cdot F(y, x)
  $$

- $Z_\lambda(x)$ does not depend on $y$.
- $F(y|x)$ decomposes into a sum of terms for consecutive pairs of labels.
- The most likely $y$ can be found with the Viterbi algorithm.
- This is because the junction tree for the linear chain CRF is also linear.
Experiments

• Experiments were carried out on the CONLL-2000 and RM-1995 (Ramshaw and Marcus) dataset.
• Considered 2\textsuperscript{nd} order Markov dependency b/w chunk labels.
• Highest overall F-Scores of around 94% were obtained.
Terminologies

• NP chunker consists of the words in a sentence annotated automatically with part-of-speech (POS) tags.
• Chunker's task is to label each word with a label indicating whether the word is outside a chunk (O), starts a chunk (B), or continues a chunk (I).
• For eg
  • Has chunk tags BIIBIIOBOBIIIOBIOBIOOBIIOBBII.
Terminologies

- 2\textsuperscript{nd} order Markov dependency encoded by making the CRF labels pairs of consecutive chunk tags.
- Label at position $i$ is $y_i = c_{i-1}c_i$.
- $c_i$ is the chunk tag of word $i$, one of O, B, or I.
- B must be used to start a chunk so the label OI is impossible.
- In addition, successive labels are constrained: $y_{i-1} = c_{i-2}c_{i-1}$, and, $y_i = c_{i-1}c_i$ and $c_0 = O$.
- Constraints enforced by giving appropriate features a weight of minus infinity.
- This forces all the forbidden labelling's to have zero probability.
Features used

- 3.8 million features.
- Factored representation for features.
  \[ f(y_{i-1}, y_i, x, i) = p(x, i)q(y_{i-1}, y_i) \]
- \( p(x, i) \) is a predicate on the input sequence \( x \) and current position \( i \) and \( q(y_{i-1}, y_i) \) is a predicate on pairs of labels.
- For instance, \( p(x, i) \) might be: word at position \( i \) is “the” or the POS tags at positions \( i-1 \), \( i \) are “DT, NN”.
- \( q(y_{i-1}, y_i) \) can be: current Label is “IB” and previous Label is “BI”.
The predicate can be an identity so that we can consider only the labels.

The predicate can be a combination of features consisting of POS tags and word (features).

More the features more the complexity.
Using the tags

• For a given position $i$, $w_i$ is the word, $t_i$ its POS tag, and $y_i$ its label.
• For any label $y = c_0 c$, $c(y) = c$ is the corresponding chunk tag.
• For example, $c(OB) = B$.
• Use of chunk tags as well as labels provides a form of back off from the very small feature counts in a second order model.
• Also allows significant associations between tag pairs and input predicates to be modelled.
Evaluation Metrics

• Standard evaluation metrics for a chunker are
  • Precision P (fraction of output chunks that exactly match the reference chunks),
  • Recall R (fraction of reference chunks returned by the chunker),
  • The F1 score \( F1 = \frac{2 \times P \times R}{P + R} \).
Labelling Accuracy

• The accuracy rate for individual labelling decisions is over-optimistic as an accuracy measure for shallow parsing.
• For instance, if the chunk BIIIIII is labelled as BIIBIIII, the labelling accuracy is 87.5% (1 of 8 is an error), but recall is 0.
• Usually accuracies are high, precisions are lower and recalls even lower.
• Due to complete chunks not being identified.
Comparisons

<table>
<thead>
<tr>
<th>Model</th>
<th>F-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>94.39%</td>
</tr>
<tr>
<td>CRF</td>
<td>94.38%</td>
</tr>
<tr>
<td>Voted Perceptron</td>
<td>94.09%</td>
</tr>
<tr>
<td>Generalized Winnow</td>
<td>93.89%</td>
</tr>
<tr>
<td>MEMM</td>
<td>93.70%</td>
</tr>
</tbody>
</table>
Conclusions

• CRF’s have extreme capabilities for sequence labeling tasks.
• Can be used to exploit features.
• Extremely useful in case of agglutinative languages which have lots features after Morphological Analysis.
• Even linear chain CRF’s have significant power.
• Better than HMM’s and MEMM’s and comparable to SVM’s.
• Training time is a major drawback.
• Increasing the complexity of a CRF increases the train time exponentially.
References

• Fei Sha and Fernando Pereira (2003). “Shallow Parsing with Conditional Random Fields”. NAACL. Department of Computer and Information Science, UPENN


• Animations from slides by Ramesh Nallapati, CMU.