Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

September 6, 2016

Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 1: Classification tasks in NLP

Classification tasks in NLP

Naive Bayes Classifier

Log linear models

Tagging tasks in NLP

Log-linear models for Tagging

Prepositional Phrases

- noun attach: *I bought the shirt with pockets*
- verb attach: *I washed the shirt with soap*
- As in the case of other attachment decisions in parsing: it depends on the meaning of the entire sentence – needs world knowledge, etc.
- Maybe there is a simpler solution: we can attempt to solve it using heuristics or associations between words

Ambiguity Resolution: Prepositional Phrases in English

- Learning Prepositional Phrase Attachment: Annotated Data

v	n_1	p	n_2	Attachment
join	board	as	director	V
is	chairman	of	N.V.	N
using	crocidolite	in	filters	V
bring	attention	to	problem	V
is	asbestos	in	products	N
making	paper	for	filters	N
including	three	with	cancer	N
⋮	⋮	⋮	⋮	⋮

Prepositional Phrase Attachment

Method	Accuracy
Always noun attachment	59.0
Most likely for each preposition	72.2
Average Human (4 head words only)	88.2
Average Human (whole sentence)	93.2

Back-off Smoothing

- Random variable a represents attachment.
- $a = n_1$ or $a = v$ (two-class classification)
- We want to compute probability of noun attachment: $p(a = n_1 \mid v, n_1, p, n_2)$.
- Probability of verb attachment is $1 - p(a = n_1 \mid v, n_1, p, n_2)$.

Back-off Smoothing

1. If $f(v, n_1, p, n_2) > 0$ and $\hat{p} \neq 0.5$

$$\hat{p}(a_{n_1} \mid v, n_1, p, n_2) = \frac{f(a_{n_1}, v, n_1, p, n_2)}{f(v, n_1, p, n_2)}$$

2. Else if $f(v, n_1, p) + f(v, p, n_2) + f(n_1, p, n_2) > 0$ and $\hat{p} \neq 0.5$

$$\hat{p}(a_{n_1} \mid v, n_1, p, n_2) = \frac{f(a_{n_1}, v, n_1, p) + f(a_{n_1}, v, p, n_2) + f(a_{n_1}, n_1, p, n_2)}{f(v, n_1, p) + f(v, p, n_2) + f(n_1, p, n_2)}$$

3. Else if $f(v, p) + f(n_1, p) + f(p, n_2) > 0$

$$\hat{p}(a_{n_1} \mid v, n_1, p, n_2) = \frac{f(a_{n_1}, v, p) + f(a_{n_1}, n_1, p) + f(a_{n_1}, p, n_2)}{f(v, p) + f(n_1, p) + f(p, n_2)}$$

4. Else if $f(p) > 0$ (try choosing attachment based on preposition alone)

$$\hat{p}(a_{n_1} \mid v, n_1, p, n_2) = \frac{f(a_{n_1}, p)}{f(p)}$$

5. Else $\hat{p}(a_{n_1} \mid v, n_1, p, n_2) = 1.0$

Prepositional Phrase Attachment: Results

- **Results (Collins and Brooks 1995)**: 84.5% accuracy with the use of some limited word classes for dates, numbers, etc.
- **Toutanova, Manning, and Ng, 2004**:
  use sophisticated smoothing model for PP attachment
  86.18% with words & stems; with word classes: 87.54%
- **Merlo, Crocker and Berthouzoz, 1997**:
  test on multiple PPs, generalize disambiguation of 1 PP to 2-3 PPs
  1PP: 84.3%   2PP: 69.6%   3PP: 43.6%
  Note that this is still not the real problem faced in parsing natural language

Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 2: Probabilistic Classifiers

Classification tasks in NLP

Naive Bayes Classifier

Log linear models

Tagging tasks in NLP

Log-linear models for Tagging

Naive Bayes Classifier

- $\mathbf{x}$ is the input that can be represented as d independent features f_j, $1 \leq j \leq d$
- y is the output classification
- $P(y \mid \mathbf{x}) = \frac{P(y) \cdot P(\mathbf{x}|y)}{P(\mathbf{x})}$ (Bayes Rule)
- $P(\mathbf{x} \mid y) = \prod_{j=1}^{d} P(f_j \mid y)$
- $P(y \mid \mathbf{x}) = P(y) \cdot \prod_{j=1}^{d} P(f_j \mid y)$

Classification tasks in NLP

Naive Bayes Classifier

Log linear models

Tagging tasks in NLP

Log-linear models for Tagging

Log linear model

- Let there be m features, $f_k(\mathbf{x}, y)$ for $k = 1, \ldots, m$
- Define a parameter vector $\mathbf{w} \in \mathbb{R}^m$
- Each $(\mathbf{x}, y)$ pair is mapped to score:

$$s(\mathbf{x}, y) = \sum_k w_k \cdot f_k(\mathbf{x}, y)$$

- Using inner product notation:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, y) &= \sum_k w_k \cdot f_k(\mathbf{x}, y) \\ s(\mathbf{x}, y) &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, y) \end{aligned}$$

- To get a probability from the score: Renormalize!

$$\Pr(y \mid \mathbf{x}, \mathbf{w}) = \frac{exp\left(s(\mathbf{x}, y)\right)}{\sum_{y'} exp\left(s(\mathbf{x}, y')\right)}$$

Log linear model

- The name 'log-linear model' comes from:

$$\log \Pr(y \mid \mathbf{x}, \mathbf{w}) = \underbrace{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, y)}_{\text{linear term}} - \underbrace{\log \sum_{y'} exp\left(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, y')\right)}_{\text{normalization term}}$$

- Once the weights are learned, we can perform predictions using these features.
- The goal: to find $\mathbf{w}$ that maximizes the log likelihood $L(\mathbf{w})$ of the labeled training set containing $(\mathbf{x}_i, y_i)$ for $i = 1 \ldots n$

$$\begin{aligned} L(\mathbf{w}) &= \sum_i \log \Pr(y_i \mid \mathbf{x}_i, \mathbf{w}) \\ &= \sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \log \sum_{y'} exp\left(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, y')\right) \end{aligned}$$

Log linear model

- Maximize:

$$L(\mathbf{w}) = \sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \log \sum_{y'} exp\left(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, y')\right)$$

- Calculate gradient:

$$\left.\frac{dL(\mathbf{w})}{d\mathbf{w}}\right|_{\mathbf{w}}$$

$$= \sum_i \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \frac{1}{\sum_{y''} exp\left(\mathbf{w} \cdot \mathbf{f}(x_i, y'')\right)} \sum_{y'} \mathbf{f}(\mathbf{x}_i, y') \cdot exp\left(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, y')\right)$$

$$= \sum_i \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \sum_{y'} \mathbf{f}(\mathbf{x}_i, y') \frac{exp\left(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, y')\right)}{\sum_{y''} exp\left(\mathbf{w} \cdot \mathbf{f}(x_i, y'')\right)}$$

$$= \underbrace{\sum_i \mathbf{f}(\mathbf{x}_i, y_i)}_{\text{Observed counts}} - \underbrace{\sum_i \sum_{y'} \mathbf{f}(\mathbf{x}_i, y') \Pr(y' \mid \mathbf{x}_i, \mathbf{w})}_{\text{Expected counts}}$$

Log linear model

- Init: $\mathbf{w}^{(0)} = \mathbf{0}$
- $t \leftarrow 0$
- Iterate until convergence:
  - Calculate: $\Delta = \left.\frac{dL(\mathbf{w})}{d\mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}^{(t)}}$
  - Find $\beta^* = \arg\max_\beta L(\mathbf{w}^{(t)} + \beta\Delta)$
  - Set $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \beta^*\Delta$

Learning the weights: $\mathbf{w}$: Generalized Iterative Scaling

$f^{\#} = max_{x,y} \sum_j f_j(x,y)$
(the maximum possible feature value; needed for scaling)
Initialize $\mathbf{w}^{(0)}$
For each iteration t
    `expected[j]` $\leftarrow$ 0 for j = 1 .. # of features
    For i = 1 to $|\text{training data}|$
        For each feature f_j
            `expected[j]` $+= f_j(x_i, y_i) \cdot P(y_i \mid x_i, \mathbf{w}^{(t)})$
    For each feature $f_j(x,y)$
        `observed[j]` $= f_j(x,y) \cdot \frac{c(x,y)}{|\text{training data}|}$
    For each feature $f_j(x,y)$
        $w_j^{(t+1)} \leftarrow w_j^{(t)} \cdot \sqrt[f^{\#}]{\frac{\texttt{observed[j]}}{\texttt{expected[j]}}}$

cf. Goodman, NIPS '01

Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 3: Linear models for Tagging

Classification tasks in NLP

Naive Bayes Classifier

Log linear models

Tagging tasks in NLP

Log-linear models for Tagging

Tagging Tasks

Tagged Sequences

a b e e a f h j ⇒ a/Y b/Z e/Y e/Y a/Z f/X h/Z j/Y

Example 1: Part-of-speech tagging

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 ./.

Example 2: Named Entity Recognition

Profits/O soared/O at/O Boeing/B-CO Co./I-CO ,/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

Notation for Tagging Tasks

- Set of possible input words: $\mathcal{V}$
- Set of possible tags: $\mathcal{T}$
- Word sequence: $x_{[1:n]} = [x_1, \ldots, x_n]$
- Tag sequence: $t_{[1:n]} = [t_1, \ldots, t_n]$
- Training data is N tagged sentences, the i^{th} sentence has length n_i:

$$(x^{(i)}_{[1:n]}, t^{(i)}_{[1:n]}) \text{ for } i = 1, \ldots, n$$

Independence Assumptions for Tagging

Chain Rule

$$P(t_{[1:n]} \mid x_{[1:n]}) = \prod_{j=1}^{n} P(t_j \mid t_{j-1}, \ldots, t_1, x_{[1:n]}, j)$$

Make independence assumptions

$$P(t_{[1:n]} \mid x_{[1:n]}) \approx \prod_{j=1}^{n} P(t_j \mid t_{j-1}, x_{[1:n]}, j)$$

j is the word being tagged.

We model the conditional probability directly: no Bayes Rule here.

Questions

- Split up $P(t_j \mid t_{j-1}, x_{[1:n]}, j)$ into parameters?
- How to find $\arg\max_{t_{[1:n]}} P(t_{[1:n]} \mid x_{[1:n]})$?

Classification tasks in NLP

Naive Bayes Classifier

Log linear models

Tagging tasks in NLP

Log-linear models for Tagging

Representation: finding the right parameters

Problem: Predict ?? using context, $P(?? \mid \texttt{context})$

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

Representation: history

- A history is a 3-tuple: $(t_{-1}, x_{[1:n]}, i)$
- t_{-1} is the previous tag (we are assuming a bigram model)
- $x_{[1:n]}$ are the n words in the input
- i is the index of the word being tagged
- For example, for $x_4 = \text{Boeing}$:
  - $t_{-1} = \text{P}$
  - $x_{[1:n]} = (\text{Profits, soared, ..., results, .})$
  - $i = 4$

Feature-vectors over history-tag pairs

Take a history, tag pair (h, t)

$f_k(h, t)$ for $k = 1, \ldots, m$ are **feature functions** representing the tagging decision.

Example: Part-of-speech tagging [Ratnaparkhi 1996]

$$f_{100}(h, t) = \begin{cases} 1 & \text{if current word } x_i \text{ is } \texttt{Boeing} \text{ and } t = \texttt{N} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{101}(h, t) = \begin{cases} 1 & \text{if } t_{-1} \text{ is } \texttt{P} \text{ and } t = \texttt{N} \\ 0 & \text{otherwise} \end{cases}$$

Log linear model for Tagging

- Let there be m features, $f_k(\mathbf{x}, \mathbf{y})$ for $k = 1, \ldots, m$
- $\mathbf{x} = x_{[1:n]}$ and $\mathbf{y} = t_{[1:n]}$
- Define a parameter vector $\mathbf{w} \in \mathbb{R}^m$
- Each $(\mathbf{x}, \mathbf{y})$ pair is mapped to score:

$$s(\mathbf{x}, \mathbf{y}) = \sum_k w_k \cdot f_k(\mathbf{x}, \mathbf{y})$$

- Using inner product notation:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) &= \sum_k w_k \cdot f_k(\mathbf{x}, \mathbf{y}) \\ s(\mathbf{x}, \mathbf{y}) &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) \end{aligned}$$

- To get a probability from the score: Renormalize!

$$\Pr(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{exp\left(s(\mathbf{x}, \mathbf{y})\right)}{\sum_{\mathbf{y'}} exp\left(s(\mathbf{x}, \mathbf{y'})\right)}$$

Feature functions for Tagging

Problem

- We have defined a log-linear model using feature functions: **f(x,y)**
- We have defined parameters using a history h so feature functions are: $\mathbf{f}(h, t)$

Locally normalized log-linear taggers

Conditional Distribution over history, tag pair (h, t)

$$\log \Pr(t \mid h) = \mathbf{w} \cdot \mathbf{f}(h, t) - log \sum_{t'} \exp\left(\mathbf{w} \cdot \mathbf{f}(h, t')\right)$$

- $\mathbf{f}(h, t)$ is a vector of feature functions
- $\mathbf{w}$ is the weight vector

Local normalization for tagging

- Word sequence: $x_{[1:n]}$ and tag sequence: $t_{[1:n]}$
- Histories $h_i = (t_{i-1}, x_{[1:n]}, i)$

$$\log \Pr(t_{[1:n]} \mid x_{[1:n]}) = \sum_{i=1}^{n} \log \Pr(t_i \mid h_i)$$

Globally normalized log-linear taggers

Global feature function $\boldsymbol{\Phi}(\mathbf{x}, \mathbf{y})$

- Word sequence: $\mathbf{x} = x_{[1:n]}$ and tag sequence: $\mathbf{y} = t_{[1:n]}$
- From *local* histories $h_i = (t_{i-1}, x_{[1:n]}, i)$ to global Φ values:

$$\Phi_k(x_{[1:n]}, t_{[1:n]}) = \sum_{i=1}^{n} f_k(h_i, t_i)$$

- $\boldsymbol{\Phi}(\mathbf{x}, \mathbf{y}) = (\Phi_1, \Phi_2, \ldots, \Phi_m)$ is a *global* feature vector
- $\mathbf{w}$ is the weight vector for $\boldsymbol{\Phi}$

Global normalization for tagging

$$\log \Pr(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \mathbf{w} \cdot \boldsymbol{\Phi}(\mathbf{x}, \mathbf{y}) - \log \sum_{\mathbf{y'}} exp\left(\mathbf{w} \cdot \boldsymbol{\Phi}(\mathbf{x}, \mathbf{y'})\right)$$

Conditional Random Field

Global normalization for tagging

$$\log \Pr(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}, \mathbf{y}) - \log \sum_{\mathbf{y'}} exp\left(\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}, \mathbf{y'})\right)$$

- This model is also called a conditional random field (CRF)

Algorithms for training and decoding

- Global normalization could be expensive: requires sum over exponentially many terms $\mathbf{y'}$
- Finding $\arg\max_{\mathbf{y}} \log \Pr(\mathbf{y} \mid \mathbf{x})$ can be accomplished using the Viterbi algorithm.
- Training: finding the weight vector $\mathbf{w}$ can be done using a variant of the Forward algorithm.

Acknowledgements

Many slides borrowed or inspired from lecture notes by Michael Collins, Chris Dyer, Kevin Knight, Philipp Koehn, Adam Lopez, and Luke Zettlemoyer from their NLP course materials.

All mistakes are my own.