Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

October 21, 2019

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

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%

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

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)
- $P(\mathbf{x} \mid y) = \prod_{j=1}^{d} P(f_j \mid y)$
- $P(y \mid \mathbf{x}) \propto P(y) \cdot \prod_{j=1}^{d} P(f_j \mid y)$
- We can ignore $P(\mathbf{x})$ in the above equation because it is a constant scaling factor for each y.

Classification tasks in NLP

Naive Bayes Classifier

Log linear models

Log linear model

- The model classifies input into output labels $y \in \mathcal{Y}$
- Let there be m features, $f_k(\mathbf{x}, y)$ for $k = 1, \ldots, m$
- Define a parameter vector $\mathbf{v} \in \mathbb{R}^m$
- Each $(\mathbf{x}, y)$ pair is mapped to score:

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

- Using inner product notation:

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

- To get a probability from the score: Renormalize!

$$\Pr(y \mid \mathbf{x}; \mathbf{v}) = \frac{exp\left(s(\mathbf{x}, y)\right)}{\sum_{y' \in \mathcal{Y}} exp\left(s(\mathbf{x}, y')\right)}$$

Log linear model

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

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

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

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

Log linear model

- Maximize:

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

- Calculate gradient:

$$\begin{aligned}
&\left.\frac{dL(\mathbf{v})}{d\mathbf{v}}\right|_{\mathbf{v}} \\
&= \sum_i \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \frac{1}{\sum_{y''} exp\left(\mathbf{v} \cdot \mathbf{f}(x_i, y'')\right)} \\
&\quad \sum_{y'} \mathbf{f}(\mathbf{x}_i, y') \cdot exp\left(\mathbf{v} \cdot \mathbf{f}(\mathbf{x}_i, y')\right) \\
&= \sum_i \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \sum_{y'} \mathbf{f}(\mathbf{x}_i, y') \frac{exp\left(\mathbf{v} \cdot \mathbf{f}(\mathbf{x}_i, y')\right)}{\sum_{y''} exp\left(\mathbf{v} \cdot \mathbf{f}(x_i, y'')\right)} \\
&= \underbrace{\sum_i \mathbf{f}(\mathbf{x}_i, y_i)}_{\text{Observed counts}} - \underbrace{\sum_i \sum_{y'} \mathbf{f}(\mathbf{x}_i, y') \Pr(y' \mid \mathbf{x}_i; \mathbf{v})}_{\text{Expected counts}}
\end{aligned}$$

Gradient ascent

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

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

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

cf. Goodman, NIPS '01

Acknowledgements

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

All mistakes are my own.

A big thank you to all the students who read through these notes and helped me improve them.