Social Media & Text Analysis
lecture 6 - Paraphrase Identification and Logistic Regression

CSE 5539-0010 Ohio State University
Instructor: Alan Ritter
Website: socialmedia-class.org

Many slides adapted from Andrew Ng
(Recap) Classification Method:

**Supervised Machine Learning**

- **Input:**
  - a sentence pair $x$ *(represented by features)*
  - a fixed set of binary classes $Y = \{0, 1\}$
  - a training set of $m$ hand-labeled sentence pairs $(x^{(1)}, y^{(1)}), \ldots , (x^{(m)}, y^{(m)})$

- **Output:**
  - a learned classifier $\gamma : x \rightarrow y \in Y$ 
    $(y = 0 \text{ or } y = 1)$
(Recap)

Linear Regression

- also supervised learning (learn from annotated data)
- but for **Regression**: predict **real-valued** output
  (Classification: predict discrete-valued output)
## Model Representation

<table>
<thead>
<tr>
<th>#words in common ($x$)</th>
<th>Sentence Similarity ($y$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- $m$ hand-labeled sentence pairs $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$

θ’s: parameters

(Recap) Linear Regression w/ one variable:

Source: many following slides are adapted from Andrew Ng
(Recap) Linear Regression:

Model Representation

- How to represent $h$?

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Linear Regression w/ one variable
(Recap)

Linear Regression

- **Hypothesis:**
  \[ h_\theta(x) = \theta_0 + \theta_1 x \]

- **Parameters:**
  \( \theta_0, \theta_1 \)

- **Cost Function:**
  \[ J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2 \]

- **Goal:** minimize \( J(\theta_0, \theta_1) \)
(Recap) Linear Regression w/ one variable:

Cost Function

\[
J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2
\]

Squared error function:

- **Idea:** choose \( \theta_0, \theta_1 \) so that \( h_\theta(x) \) is close to \( y \) for training examples \( (x, y) \) minimize \( J(\theta_0, \theta_1) \).
(Recap)

**Gradient Descent**

repeat until convergence {

\[
\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)
\]

}  

**learning rate**

simultaneous update for j=0 and j=1
(Recap) Linear Regression w/ one variable:

Gradient Descent

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) \cdot x_i$$

}
Linear Regression w/ multiple variables (features):

Model Representation

- Hypothesis:

\[ h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \]

(for convenience, define \( x_0 = 1 \))
Linear Regression w/ multiple variables (features):

Model Representation

- Hypothesis:

\[ h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \]

(for convenience, define \( x_0 = 1 \))

\[
\begin{bmatrix}
    x_0 \\
    x_1 \\
    x_2 \\
    \vdots \\
    x_n
\end{bmatrix} \in \mathbb{R}^{n+1} \quad \theta =
\begin{bmatrix}
    \theta_0 \\
    \theta_1 \\
    \theta_2 \\
    \vdots \\
    \theta_n
\end{bmatrix} \in \mathbb{R}^{n+1}
\]
Linear Regression w/ multiple variables (features):

**Model Representation**

- **Hypothesis:**

  \[
  h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n
  \]

  (for convenience, define \( x_0 = 1 \))

\[
\begin{bmatrix}
  x_0 \\
  x_1 \\
  x_2 \\
  \vdots \\
  x_n
\end{bmatrix} \in \mathbb{R}^{n+1} \quad \begin{bmatrix}
  \theta_0 \\
  \theta_1 \\
  \theta_2 \\
  \vdots \\
  \theta_n
\end{bmatrix} \in \mathbb{R}^{n+1}
\]

\[
h_\theta(x) = \theta^T x
\]
Linear Regression w/ multiple variables (features):

Model Representation

- Hypothesis:
  \[ h_\theta(x) = \theta^T x \]

- Cost function:  
  \[ J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \]  
  "# training examples"
(Recap)

Paraphrase Identification

obtain sentential paraphrases automatically

<table>
<thead>
<tr>
<th>Mancini has been sacked by Manchester City</th>
<th>Yes!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mancini gets the boot from Man City</td>
<td></td>
</tr>
<tr>
<td>WORLD OF JENKS IS ON AT 11</td>
<td>No!</td>
</tr>
<tr>
<td>World of Jenks is my favorite show on tv</td>
<td></td>
</tr>
</tbody>
</table>
(Recap)

Linear Regression

- also supervised learning (learn from annotated data)
- but for **Regression**: predict *real-valued* output
  
  (Classification: predict discrete-valued output)
(Recap)

**Linear Regression**

- also supervised learning (learn from annotated data)
- but for **Regression**: predict real-valued output
  (Classification: predict discrete-valued output)
(Recap)

**Linear Regression**

- also supervised learning (learn from annotated data)
- but for **Regression**: predict **real-valued** output
  - (Classification: predict discrete-valued output)
A problem in classification

Linear Regression

#words in common (feature)
A problem in classification

Linear Regression

#words in common (feature)

Sentence Similarity

Wei Xu ◦ socialmedia-class.org
A problem in classification

Linear Regression

Sentence Similarity vs. #words in common (feature)

threshold → Classification
A problem in classification

Linear Regression

paraphrase

Sentence Similarity

#words in common (feature)

threshold ➞ Classification
A problem in classification

Linear Regression

paraphrase

non-paraphrase

Sentence Similarity

threshold → Classification

#words in common (feature)
A problem in classification

**Linear Regression**

Classification error

paraphrase

non-paraphrase

In practice, do not use linear regression for classification.
(Recap) Classification:

Supervised Machine Learning

training set

(also called) hypothesis
(Recap)

Logistic Regression

• One of the most useful **supervised machine learning algorithm** for classification!

• Generally high performance for a lot of problems.

• Much more robust than Naïve Bayes (better performance on various datasets).
Hypothesis:

Linear → Logistic Regression

Classification: $y = 0$ or $y = 1$

• Linear Regression: $h_\theta(x)$ can be $> 1$ or $< 0$
Hypothesis:

**Linear \rightarrow Logistic Regression**

Classification: $y = 0$ or $y = 1$

- Linear Regression: $h_\theta(x)$ can be $> 1$ or $< 0$

- Logistic Regression: want $0 \leq h_\theta(x) \leq 1$
Hypothesis:

Linear → Logistic Regression

Classification: \( y = 0 \) or \( y = 1 \)

- Linear Regression: \( h_\theta(x) \) can be > 1 or < 0

- Logistic Regression: want \( 0 \leq h_\theta(x) \leq 1 \)

a classification (not regression) algorithm
Hypothesis:

Linear → Logistic Regression

• Linear Regression: \( h_\theta(x) = \theta^T x \)

• Logistic Regression: want \( 0 \leq h_\theta(x) \leq 1 \)
Hypothesis:

**Linear → Logistic Regression**

- Linear Regression: \( h_\theta(x) = \theta^T x \)
- Logistic Regression: **want** \( 0 \leq h_\theta(x) \leq 1 \)

sigmoid (logistic) function

\[
\sigma(z) = \frac{1}{1 + e^{-z}}
\]
Hypothesis:

Linear → Logistic Regression

- Linear Regression: \( h_\theta(x) = \theta^T x \)
- Logistic Regression: want \( 0 \leq h_\theta(x) \leq 1 \)

sigmoid (logistic) function

\[
h_\theta(x) = \sigma \left( \theta^T x \right)
\]

\[
\sigma(z) = \frac{1}{1 + e^{-z}}
\]
(Recap) Classification Method:

Supervised Machine Learning

• Input:
  - a sentence pair $x$ (represented by features)
  - a fixed set of binary classes $Y = \{0, 1\}$
  - a training set of $m$ hand-labeled sentence pairs $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$

• Output:
  - a learned classifier $\gamma: x \rightarrow y \in Y$ ($y = 0$ or $y = 1$)
Logistic Regression:

Interpretation of Hypothesis

• $h_\theta(x) = \text{estimated probability that } y = 1 \text{ on input}$
Logistic Regression:

Interpretation of Hypothesis

- \( h_\theta(x) \) = estimated probability that \( y = 1 \) on input

\[
\text{If } x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{#words\_in\_common} \end{bmatrix}, \ h_\theta(x) = 0.7
\]

70% chance of the sentence pair being paraphrases
Logistic Regression:

Interpretation of Hypothesis

- \( h_\theta(x) = \) estimated probability that \( y = 1 \) on input

\[
If \quad x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{#words\_in\_common} \end{bmatrix}, \quad h_\theta(x) = 0.7
\]

70% chance of the sentence pair being paraphrases

\[
h_\theta(x) = P(y = 1 \mid x; \theta)
\]

probability that \( y = 1 \), given \( x \), parameterized by \( \theta \)
Logistic Regression:

**Interpretation of Hypothesis**

- $h_\theta(x) = \text{estimated probability that } y = 1 \text{ on input}$

\[ h_\theta(x) = P(y = 1 \mid x; \theta) \]

probability that $y = 1$, given $x$, parameterized by $\theta$
Logistic Regression:

Interpretation of Hypothesis

- $h_\theta(x) = \text{estimated probability that } y = 1 \text{ on input}$

$$P(y = 1 \mid x; \theta) + P(y = 0 \mid x; \theta) = 1$$

$h_\theta(x) = P(y = 1 \mid x; \theta)$

probability that $y = 1$, given $x$, parameterized by $\theta$
Logistic Regression:

Decision Boundary

- Logistic Regression:

\[ h_\theta(x) = \sigma(\theta^T x) \]
\[ h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \]

sigmoid (logistic) function

\[ \sigma(z) = \frac{1}{1 + e^{-z}} \]
Logistic Regression:

**Decision Boundary**

- Logistic Regression: sigmoid (logistic) function

\[ h_\theta(x) = \sigma \left( \theta^T x \right) \]

\[ h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \]

- predict \( y = 1 \) if \( h_\theta(x) \geq 0.5 \)
- predict \( y = 0 \) if \( h_\theta(x) < 0.5 \)
Logistic Regression:

**Decision Boundary**

- Logistic Regression: \( h_\theta(x) = \sigma(\theta^T x) \)

\[
\begin{align*}
h_\theta(x) &= \frac{1}{1 + e^{-\theta^T x}} \\
\sigma(z) &= \frac{1}{1 + e^{-z}}
\end{align*}
\]

predict \( y = 1 \) if \( h_\theta(x) \geq 0.5 \) ← when \( \theta^T x \geq 0 \)

predict \( y = 0 \) if \( h_\theta(x) < 0.5 \) ← when \( \theta^T x < 0 \)
Logistic Regression:

Decision Boundary
Logistic Regression:

**Decision Boundary**

\[ h_\theta(x) = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2) \]

What if \( \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix} \)?

predict \( y = 1 \) if \( \theta^T x \geq 0 \)
Logistic Regression:

**Decision Boundary**

$$h_\theta(x) = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

**What if**

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

**Predict**

$$y = 1 \text{ if } \theta^T x \geq 0$$
Logistic Regression:

**Decision Boundary**

\[ h_\theta(x) = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2) \]

- decision boundary
- \( y = 1 \)
- \( y = 0 \)
- a property of the hypothesis
- a property of the parameters
- a property of the dataset
Logistic Regression

- a training set of \( m \) hand-labeled sentence pairs 
  \((x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\) \((y \in \{0, 1\})\)

\[
h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}
\]

\[
x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}
\]
Cost function:

**Linear → Logistic Regression**

- Linear Regression:  
  
  \[
  J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2
  \]

  squared error function
Cost function:

Linear → Logistic Regression

- Linear Regression: \( J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left( h_\theta(x^{(i)}) - y^{(i)} \right)^2 \)

  squared error function

- Logistic Regression: \( h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \)

  this cost function is non-convex for logistic regression
Cost function:

**Linear → Logistic Regression**

- Linear Regression: \( J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left( h_\theta(x^{(i)}) - y^{(i)} \right)^2 \)

- Logistic Regression: \( h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \)

this cost function is non-convex for logistic regression
$J(\theta_1)$
we want convex! easy gradient descent!
Logistic Regression: 

Cost Function

\[
\text{Cost}(h_\theta(x), y) = \begin{cases} 
-\log(h_\theta(x)) & \text{if } y = 1 \\
-\log(1 - h_\theta(x)) & \text{if } y = 0 
\end{cases}
\]

remember that

\[
0 \leq h_\theta(x) \leq 1
\]
Logistic Regression:

Cost Function

\[
\text{Cost}(h_\theta(x), y) = \begin{cases} 
- \log(h_\theta(x)) & \text{if } y = 1 \\
- \log(1 - h_\theta(x)) & \text{if } y = 0
\end{cases}
\]

remember that

\[0 \leq h_\theta(x) \leq 1\]
Logistic Regression:

Cost Function

\[
\text{Cost}(h_{\theta}(x), y) = \begin{cases} 
-\log(h_{\theta}(x)) & \text{if } y = 1 \\
-\log(1 - h_{\theta}(x)) & \text{if } y = 0 
\end{cases}
\]
Logistic Regression:

Cost Function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} 
-\log(h_\theta(x)) & \text{if } y = 1 \\
-\log(1 - h_\theta(x)) & \text{if } y = 0 
\end{cases}$$

Cost = 0 if $y = 1$, $h_\theta(x) = 1$

But as $h_\theta(x) \to 0$, Cost $\to \infty$
Logistic Regression:

**Cost Function**

\[
\text{Cost}(h_\theta(x), y) = \begin{cases} 
  -\log(h_\theta(x)) & \text{if } y = 1 \\
  -\log(1 - h_\theta(x)) & \text{if } y = 0
\end{cases}
\]

*Cost = 0* if \( y = 1, \ h_\theta(x) = 1 \)

But as \( h_\theta(x) \rightarrow 0 \), \( \text{Cost} \rightarrow \infty \)

**Intuition:**

penalize learning algorithm if \( h_\theta(x) = 0 \) (predict \( P(y=1|x;\theta) = 0 \)),

but \( y = 1 \)
Logistic Regression:

Cost Function

\[ J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}\left(h_\theta(x^{(i)}) - y^{(i)}\right) \]

\( \text{Cost}(h_\theta(x), y) = \begin{cases} 
- \log(h_\theta(x)) & \text{if } y = 1 \\
- \log(1 - h_\theta(x)) & \text{if } y = 0 
\end{cases} \)

remember that \( y = 0 \) or \( 1 \) always
Logistic Regression:

Cost Function

\[ J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}) - y^{(i)}) \]

\[ \text{Cost}(h_\theta(x), y) = \begin{cases} 
- \log(h_\theta(x)) & \text{if } y = 1 \\
- \log(1 - h_\theta(x)) & \text{if } y = 0 
\end{cases} \]

remember that \( y = 0 \) or \( 1 \) always

the same

cross entropy loss:

\[ \text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x)) \]

Learn more: https://en.wikipedia.org/wiki/Cross_entropy
Logistic Regression

- **Cost Function:**

\[
J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}\left( h_{\theta}(x^{(i)}) - y^{(i)} \right)
\]

\[
= -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log(h_{\theta}(x^{(i)})) + \left(1 - y^{(i)}\right) \log(1 - h_{\theta}(x^{(i)}))
\]

- **Goal:**

\[
\text{learn parameters } \theta \text{ to } \minimize_{\theta} J(\theta)
\]

- **Hypothesis (to make a prediction):**

\[
h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}
\]

\[
P(y=1|x;\theta)
\]
Logistic Regression:

Gradient Descent

repeat until convergence {

\[ \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \]

}  

simultaneous update for all \( \theta_j \)

learning rate
Logistic Regression: Gradient Descent

repeat until convergence {
\[ \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \]
}

learning rate

\[ \frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left( h_{\theta}(x^{(i)}) - y^{(i)} \right) x_j^{(i)} \]

# training examples

simultaneous update for all \( \theta_j \)
Logistic Regression: Gradient Descent

repeat until convergence {

\[ \theta_j := \theta_j - \alpha \sum_{i=1}^{m} \left( h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} \]

} simultaneous update for all \(\theta_j\)

\[ \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \]
Logistic Regression: Gradient Descent

repeat until convergence {
\[ \theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_{\theta}(x^{(i)}) - y^{(i)})x_{j}^{(i)} \]
}

This look the same as linear regression!!???
Logistic Regression:

Gradient Descent

repeat until convergence {

$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} \left( h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$

}

$1 \leq j \leq n$

simultaneous update for all $\theta_j$

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$

using different hypothesis from linear regression
[Recap] Classification Method: Supervised Machine Learning
Classification Evaluation

F-measure:

\[ F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \]
NLP Pipeline (next)

Language Identification → Tokenization → Part-of-Speech (POS) Tagging → Shallow Parsing (Chunking) → Named Entity Recognition (NER)

Sequential Tagging

- Stemming
- Normalization
Part-of-Speech (POS) Tagging

<table>
<thead>
<tr>
<th>Word</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cant</td>
<td>MD</td>
</tr>
<tr>
<td>wait</td>
<td>VB</td>
</tr>
<tr>
<td>for</td>
<td>IN</td>
</tr>
<tr>
<td>the</td>
<td>DT</td>
</tr>
<tr>
<td>ravens</td>
<td>NNP</td>
</tr>
<tr>
<td>game</td>
<td>NN</td>
</tr>
<tr>
<td>tomorrow</td>
<td>NN</td>
</tr>
<tr>
<td>...</td>
<td>.</td>
</tr>
<tr>
<td>go</td>
<td>VB</td>
</tr>
<tr>
<td>ray</td>
<td>NNP</td>
</tr>
<tr>
<td>rice</td>
<td>NNP</td>
</tr>
<tr>
<td>!!!!!!</td>
<td>.</td>
</tr>
</tbody>
</table>

Cant wait for the ravens game tomorrow....go ray rice!!!!!!!
### Chunking

<table>
<thead>
<tr>
<th>Cant</th>
<th>VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>wait</td>
<td>PP</td>
</tr>
<tr>
<td>for</td>
<td>PP</td>
</tr>
<tr>
<td>the</td>
<td></td>
</tr>
<tr>
<td>ravens</td>
<td>NP</td>
</tr>
<tr>
<td>game</td>
<td></td>
</tr>
<tr>
<td>tomorrow</td>
<td>NP</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>go</td>
<td>VP</td>
</tr>
<tr>
<td>ray</td>
<td>NP</td>
</tr>
<tr>
<td>rice</td>
<td></td>
</tr>
<tr>
<td>!!!!!!!</td>
<td></td>
</tr>
</tbody>
</table>

Cant wait for the ravens game tomorrow....go ray rice!!!!!!!
Named Entity Recognition (NER)

<table>
<thead>
<tr>
<th>Cant</th>
<th>wait</th>
<th>for</th>
<th>the</th>
<th>ravens</th>
<th>game</th>
<th>tomorrow</th>
<th>...</th>
<th>go</th>
<th>ray</th>
<th>rice</th>
<th>!!!!!!</th>
</tr>
</thead>
</table>

**ORG:** organization

**PER:** person

**LOC: **location
IO tag encoding

<table>
<thead>
<tr>
<th>Cant</th>
<th>VP</th>
<th>VP</th>
<th>VP wait for the ravens game tomorrow....go ray rice!!!!!!</th>
</tr>
</thead>
<tbody>
<tr>
<td>wait</td>
<td>VP</td>
<td>VP</td>
<td></td>
</tr>
<tr>
<td>for</td>
<td>PP</td>
<td>PP</td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>NP</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td>ravens</td>
<td>NP</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td>game</td>
<td>NP</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td>tomorrow</td>
<td>NP</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>go</td>
<td>VP</td>
<td>VP</td>
<td></td>
</tr>
<tr>
<td>ray</td>
<td>NP</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td>rice</td>
<td>NP</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td>!!!!!!!</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>
# IO tag encoding

<table>
<thead>
<tr>
<th>Cant</th>
<th>VP</th>
<th>VP</th>
<th>B-VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>wait</td>
<td>VP</td>
<td>VP</td>
<td>I-VP</td>
</tr>
<tr>
<td>for</td>
<td>PP</td>
<td>PP</td>
<td>B-PP</td>
</tr>
<tr>
<td>the</td>
<td>NP</td>
<td>NP</td>
<td>B-NP</td>
</tr>
<tr>
<td>ravens</td>
<td>NP</td>
<td>NP</td>
<td>I-NP</td>
</tr>
<tr>
<td>game</td>
<td>NP</td>
<td>NP</td>
<td>I-NP</td>
</tr>
<tr>
<td>tomorrow</td>
<td>NP</td>
<td>NP</td>
<td>B-NP</td>
</tr>
<tr>
<td>...</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>go</td>
<td>VP</td>
<td>VP</td>
<td>B-VP</td>
</tr>
<tr>
<td>ray</td>
<td>NP</td>
<td>NP</td>
<td>B-VP</td>
</tr>
<tr>
<td>rice</td>
<td>NP</td>
<td>NP</td>
<td>I-VP</td>
</tr>
<tr>
<td>!!!!!!!!!!</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>

I: Inside  
O: outside  
B: Begin  

BIO allows separation of adjacent chunks/entities
Classification Method:
Supervised Machine Learning

- Naïve Bayes
- Logistic Regression
- Support Vector Machines (SVM)
- …
- Hidden Markov Model (HMM)
- Conditional Random Fields (CRF)
- …
Classification Method:
Sequential Supervised Learning

• Input:
  - rather than just individual examples \((w_1 = \text{the}, \ c_1 = \text{DT})\)
  - a training set consists of \(m\) sequences of labeled examples \((x_1, y_1), \ldots, (x_m, y_m)\)

\[x_1 = \langle \text{the back door} \rangle \text{ and } y_1 = \langle \text{DT JJ NN} \rangle\]

• Output:
  - a learned classifier to predict label sequences \(\gamma: x \rightarrow y\)
Features for Sequential Tagging

- **Words:**
  - current words
  - previous/next word(s) — context
- **Other linguistic information:**
  - word substrings
  - word shapes
  - POS tags
- **Contextual Labels**
  - previous (and perhaps next) labels

<table>
<thead>
<tr>
<th>Varicella-zoster</th>
<th>Xx-xxx</th>
</tr>
</thead>
<tbody>
<tr>
<td>mRNA</td>
<td>XXXX</td>
</tr>
<tr>
<td>CPA1</td>
<td>XXXd</td>
</tr>
</tbody>
</table>
Probabilistic Graphical Models

Naïve Bayes Classifier

\[ p(y, x) \]

Hidden Markov Model

\[ p(Y, X) \]

Logistic Regression

\[ p(y/x) \]

Conditional Random Field

\[ p(Y/X) \]
Probabilistic Graphical Models

**Naïve Bayes Classifier**

\[ p(y, x) = p(y) \prod_{m=1}^{M} p(x_m | y) \]

\[ p(y | x) = \frac{\exp \left\{ \sum_{n=1}^{M} \lambda_m f_m (y, x) \right\}}{\sum_{y'} \exp \left\{ \sum_{m=1}^{M} \lambda_m f_m (y', x) \right\}} \]

**Hidden Markov Model**

\[ p(Y, X) = \prod_{n=1}^{N} p(y_n | y_{n-1}) p(x_n | y_n) \]

\[ p(Y | X) = \frac{\exp \left\{ \sum_{m=1}^{M} \lambda_m f_m (y_n, y_{n-1}, x_n) \right\}}{\sum_{y'} \exp \left\{ \sum_{m=1}^{M} \lambda_m f_m (y_n', y_{n-1}', x_n) \right\}} \]

**Logistic Regression**

**Conditional Random Field**
Probabilistic Graphical Models