Week 2 Tagging Problem and Hidden Markov Models

Tagging problem

- Part-of-speech tagging
- Named Entity Recognition
    - NA = No entity, SC = Start Company, CC = Continue Company, SL = Start Location, CL = Continue Location ...
- A commonly used resource: Wall Street Journal tree bank
- Two types of constraints: local or contextual.

Generative Models

- **Training set**

  $x^{(i)}, y^{(i)} \text{ for } i = 1...m$. Each $x^{(i)}$ is an input, and $y^{(i)}$ is a label.

  Task

  Learn a function f mapping inputs x to labels $f(x)$.
- Conditional (Discriminative) models:
    - Learn a distribution $p(y|x)$ from training set
    - For any test input x ,define $f(x) = \arg\max_y p(y|x)$
- Generative models:
    - Learn a distribution $p(x, y)$ from training set
    - For $p(x, y) = p(y)p(x|y)$, and we get

      $$p(y|x) = \frac{p(y)p(x|y)}{p(x)}$$

      where $p(x) = \sum_y p(y)p(x|y)$
    - As $p(x)$ doesn't vary with y , we can get:

      $$f(x) = \arg\max_y p(y|x) = \arg\max_y \frac{p(y)p(x|y)}{p(x)} = \arg\max_y p(y)p(x|y)$$

Hidden Markov Models

- **Definitions**
    - An input sequence $x = x_1, x_2, \ldots, x_n$ where $x_i \in \mathcal{V} \text{ for } i = 1...n$
    - A tag sequence $y = y_1, y_2, \ldots, y_{n+1}$ where $y_i \in \mathcal{S} \text{ for } i = 1...n$ and $y_{n+1} = \text{STOP}$

- The joint probablity of the sentence and tag seqence is

$$p(x_1,\dots,x_n,y_1,\dots,y_{n+1})=\prod_{i=1}^{n+1}q(y_i|y_{y-2},y_{i-1})\prod_{i=1}^{n}e(x_i|y_i)$$

where $y_0=y_{-1}=*$.
- The most likely tag sequence for x is

$$\arg\max_{y_1,y_2,\dots,y_n}p(x_1,\dots,x_n,y_1,y_2,\dots,y_n)$$

- Parameters of the model:
  - $q(s|u,v)$ for any $s\in\mathcal{S}\cup\{\text{STOP}\}$, $u,v\in\mathcal{S}\cup\{*\}$
  - $e(x|s)$ for any $s\in\mathcal{S},x\in\mathcal{V}$

Parameter estimation

- Trigram parameters: interpretation method
- Emmission parameters: maximun likelihood estimation

$$e(\text{base}|V_t)=\frac{\text{Count}(V_t,\text{base})}{\text{Count}(V_t)}$$

Deficiency:

$e(\text{base}|V_t)=0$ for all V_t ,if base is never seen in the training data. And it's frequent to see a word appear in test data while not in training data.

A common method to fix the bug:

Step 1: Split vocabulary into 2 sets

$$\begin{aligned}\text{Frequent words}&=\text{words occuring}\geq 5\text{ times in traning}\\\text{Low frequent words}&=\text{all other words}\end{aligned}$$

Step 2: Map low frequency words into a small, finite set, depending on prefixes, suffixes etc.

- **The Viterbi algorithm**
  - **Problem**

    For Input: $x_1\dots x_n$,to find

$$\arg\max_{y_1\dots y_{n+1}}p(x_1\dots x_n,y_1\dots y_{n+1})$$

where the $\arg\max$ is taken over all sequences $y_1\dots y_{n+1}$ such that $y_i\in\mathcal{S}$ for $i=1...n$,and $y_{n+1}=\text{STOP}$.

We assume that p takes the form

$$p(x1...x_n,y_1\dots y_{n+1})=\prod_{i=1}^{n+1}q(y_i|y_{i-2},y_{i-1})\prod_{i=1}^{n}e(x_i|y_i)$$

where $y_0=y_{-1}=*$,and $y_{n+1}=\text{STOP}$.

  - **Definition**

        - The length of the sentence: n .
        - Define S_k for $k=-1...n$ to be the set of possible tags at position k :

$$S_{-1}=S_0=\{*\}$$
$$S_k=S \text{ for } k\in\{1...n\}$$

        - Define

$$r(y_{-1},y_0,y_1,\ldots,y_k)=\prod_{i=1}^{k}q(y_i|y_{i-2},y_{i-1})\prod_{i=1}^{k}e(x_i|y_i)$$

        - Define a dynamic programming table

$\pi(k,u,v)=$ maximum probability of a tag sequence ending in tags u,v at position k

that is,

$$\pi(k,u,v)=\max_{\langle y_{-1},y_0,y_1,\ldots,y_k\rangle:y_{k-1}=u,y_k=v}r(y_{-1},y_0,y_1,\ldots,y_k)$$

        - Base case:

$$\pi(0,*,*)=1$$

        - Recursive definition:

          For any $k\in\{1...n\}$, for any $u\in\mathcal{S}_{k-1}$ and $v\in\mathcal{S}_k$:

$$\pi(k,u,v)=\max_{w\in\mathcal{S}_{k-2}}\left(\pi(k-1,w,u)\times q(v|w,u)\times e(x_k|v)\right)$$

    - **The Viterbi Algorithm with Backpointers**
        - **Input**: a sentence $x_1\ldots x_n$, parameters $q(s|u,v)$ and $e(x|s)$.
        - **Initializtion**: Set $\pi(0,*,*)=1$
        - **Definition**: $\mathcal{S}_{-1}=\mathcal{S}_0=\{*\},\mathcal{S}_k=\mathcal{S}\text{ for }k\in\{1...n\}$
        - **Algorithm**:

          For $k=1...n,$

          For $u\in\mathcal{S}_{k-1},v\in\mathcal{S}_k,$

          $\pi(k,u,v)=\max_{w\in\mathcal{S}_{k-2}}(\pi(k-1,w,u)\times q(v|w,u)\times e(x_k|v))$

          $bp(k,u,v)=\arg\max_{w\in\mathcal{S}_{k-2}}(\pi(k-1,w,u)\times q(v|w,u)\times e(x_k|v))$

          Set $(y_{n-1},y_n)=\arg\max_{(u,v)}(\pi(n,u,v)\times q(\text{STOP}|u,v))$

          For $k=(n-2)\ldots 1$, $y_k=bp(k+2,y_{k+1},y_{k+2})$

          Return the tag sequence $y_1\ldots y_n$
    - Run time complexity: $\mathcal{O}(n|\mathcal{S}|^3)$,while the brute force search is $\mathcal{O}(|\mathcal{S}|^n)$
- **Pros and Cons**
    - Hidden markov models are very simple to train
    - Perform relatively well (over 90% performance in named entity recognition)
    - Main difficulty is modeling: $e(word\mid tag)$ can be very difficult if "words" are complex.