Sequence labeling problem

Assigning Meaning to Word Sequences

Quick recap of previous lecture

N-gram language model

- An n-gram LM is a probabilistic model that can estimate the probability of a next word given the previous words.
- Thereby assign probabilities to entire sequences.
- Eg. We have “**The water of Walden Pond is so beautifully** ”, and we want to know if **blue** is the next probable word:

$$P(\texttt{blue}|\texttt{The water of Walden Pond is so beautifully}) = \frac{C(\texttt{The water of Walden Pond is so beautifully blue})}{C(\texttt{The water of Walden Pond is so beautifully})}$$

Challenge in counting long sequences

- Language is creative
- Data Sparsity as new sentences are invented all the time
- Zero Counts for Valid Sequences

$$P(\texttt{blue}|\texttt{The water of Walden Pond is so beautifully}) = \frac{C(\texttt{The water of Walden Pond is so beautifully blue})}{C(\texttt{The water of Walden Pond is so beautifully})}$$

$$\begin{aligned} P(w_{1:n}) &= P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\ldots P(w_n|w_{1:n-1}) \\ &= \prod_{k=1}^{n} P(w_k|w_{1:k-1}) \end{aligned}$$

Markov assumption

Andrei Markov

Simplifying assumption:

$$P(\texttt{blue}|\texttt{The water of Walden Pond is so beautifully})$$

$$\approx \; P(\texttt{blue}|\texttt{beautifully})$$

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$$

Bigram Markov Assumption

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k | w_{k-1})$$

instead of:

$$\prod_{k=1}^{n} P(w_k | w_{1:k-1})$$

N-gram Markov assumption

$$P(w_n | w_{1:n-1}) \approx P(w_n | \underbrace{w_{n-N+1:n-1}}_{\text{Context}})$$

We can predict a word using Trigram model as:

$$P(w_i | w_1 w_2 \ldots w_{i-1}) \approx P(w_i | w_{i-2} w_{i-1})$$

Example

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

$$P(\texttt{I} \mid \texttt{<s>}) = \tfrac{2}{3} = 0.67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \tfrac{1}{3} = 0.33 \qquad P(\texttt{am} \mid \texttt{I}) = \tfrac{2}{3} = 0.67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{do} \mid \texttt{I}) = \tfrac{1}{3} = 0.33$$

P(want/i) = 0.32649

P(eat/to) = 0.284

P(i|<s>) = 0.25

P(english|want) = 0.0011

P(food|english) = 0.5

P(</s>|food) = 0.68

Compute probabilities of:

- *“I want English food”*

$$
\begin{aligned}
P(\texttt{<s> i want english food </s>}) &= P(\texttt{i}|\texttt{<s>})P(\texttt{want}|\texttt{i})P(\texttt{english}|\texttt{want}) \\
&\quad P(\texttt{food}|\texttt{english})P(\texttt{</s>}|\texttt{food}) \\
&= 0.25 \times 0.33 \times 0.0011 \times 0.5 \times 0.68 \\
&= 0.000031
\end{aligned}
$$

Adding in log space is equivalent to multiplying in linear space

$$p_1 \times p_2 \times p_3 \times p_4 = \exp(\log p_1 + \log p_2 + \log p_3 + \log p_4)$$

Dataset: https://stressosaurus.github.io/raw-data-google-ngram/

Toolkits:

- http://www.speech.sri.com/projects/srilm/
- https://kheafield.com/code/kenlm/

Sentence generation using four N-gram language models, trained on a dataset of Shakespeare's text.

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in; –It cannot be but so.

How to evaluate LMs (N-gram models)

- A good LM is one that assigns a higher probability to the next word that actually occurs.
- The best language model is one that best predicts the entire unseen test set
- Probability depends on size of test set
  - Longer the text smaller the probability score

$$P(w_1 w_2 \ldots w_N)$$

- **Perplexity** is the inverse probability of the test set, normalized by the number of words.

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

How to evaluate LMs (N-gram models)

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

	Unigram	Bigram	Trigram
Perplexity	962	170	109

* The lower the perplexity, the better the language model.

Regularization

Maximum likelihood estimation has a problem

- **Language is creative:** Many valid word sequences may not appear in training data.
- **Assigns zero probability** to unseen n-grams (data sparsity problem).
- **Zero probabilities** break language models and lead to severe errors in various applications.

Smoothing

- **Goal:** Redistribute some probability mass from seen to unseen events to avoid zeros.
- **Laplace (Add-1) Smoothing:**
  Adds 1 to all counts, ensuring every possible n-gram has a nonzero probability.

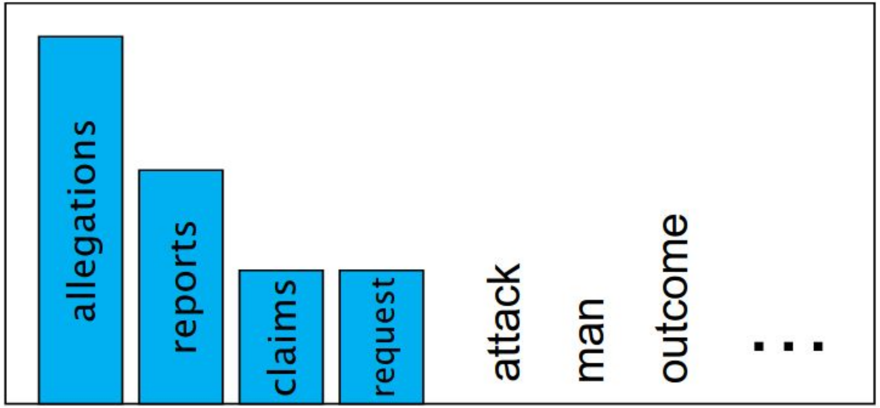
Interpolation

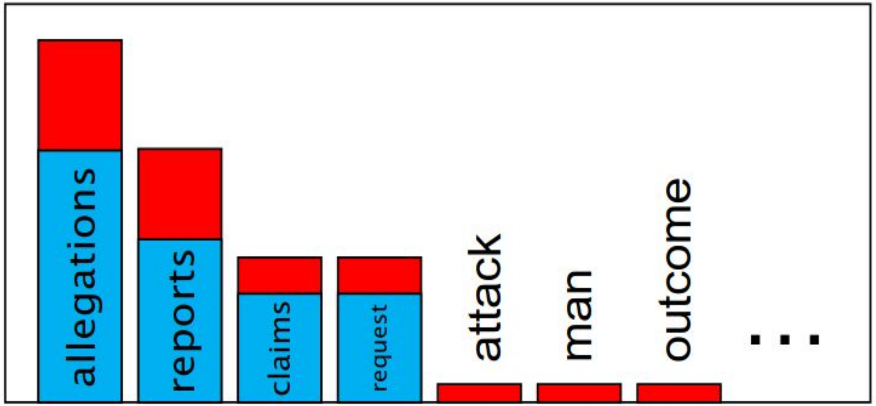
- **Goal:** Combine multiple models of different orders (e.g., trigram, bigram, unigram) to get more reliable probability estimates.

Laplace (Add-1) Smoothing

$$\boxed{P^*_{\text{Add-k}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+k}{C(w_{n-1})+kV}}$$

$$P_{\text{Laplace}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{\sum_w (C(w_{n-1}w)+1)} = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$





Interpolation

- Combine multiple models of different orders (e.g., trigram, bigram, unigram) to get more reliable probability estimates.

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) \;=\; & \lambda_1 P(w_n) \\ & +\lambda_2 P(w_n|w_{n-1}) \\ & +\lambda_3 P(w_n|w_{n-2}w_{n-1})\end{aligned}$$

$$\lambda_1+\lambda_2+\lambda_3=1$$

Sequence labelling problem

POS Tagging
(Part-of-speech)

INPUT:
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT:
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 ./.

KEY:

N	=	Noun
V	=	Verb
P	=	Preposition
Adv	=	Adverb
Adj	=	Adjective

...

Name-entity-recognition (NER)

INPUT: Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT: Profits soared at [Company Boeing Co.], easily topping forecasts on [Location Wall Street], as their CEO [Person Alan Mulally] announced first quarter results.

INPUT: Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT: Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA results/NA ./NA

KEY:

NA	= No entity
SC	= Start Company
CC	= Continue Company
SL	= Start Location
CL	= Continue Location

Sequence labeling problem

- Want to model pairs of sequences (sentence, tags)
- We will use $x_1 \ldots x_n$ to denote the input to the tagging model: we will often refer to this as a sentence.
- We will use $y_1 \ldots y_n$ to denote the output of the tagging model: we will often refer to this as the state sequence or tag sequence.
- Our task is to learn a function $f : X \rightarrow Y$.
- We call this type of problem as **sequence labeling problem** or **tagging problem**.

Sequence labeling problems:

- POS (Part-of-speech)
- NER (Name-entity-recognition)

Supervised learning

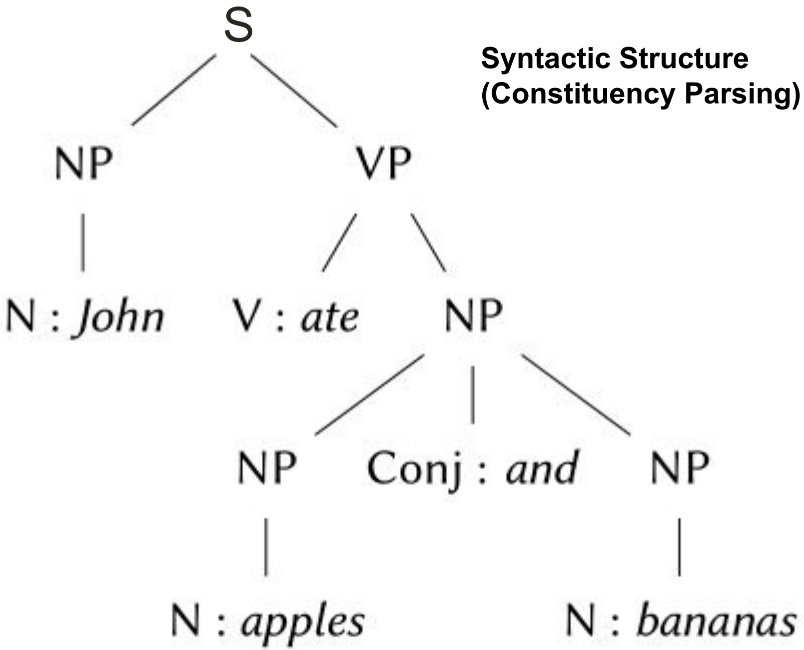
Sequence labelling problems is a supervised learning problem.

Training examples:

- $(x^{(i)}, y^{(i)})$, i = 1 … m
- $x_1^{(i)} \dots x_n^{(i)}$ represents the i^{th} sentence having word length n
- $y_1^{(i)} \dots y_n^{(i)}$ represents the i^{th} sentence having word length n
- $x_j^{(i)}$ represents the j^{th} word in the i^{th} training example
- Our task is to learn a function $f : X \rightarrow Y$ from these training examples.
  - X refer to all the $x_1 \dots x_n$
  - Y refer to all the $y_1 \dots y_n$

Part-Of-Speech (POS) tagging (Training corpus)

- The Penn WSJ treebank corpus contains around 1 million words (around 40,000 sentences) annotated with their POS tags.



Syntactic Structure (Constituency Parsing)

Part-Of-Speech (POS) Tagging

- The input to the problem is a sentence.
- The output is a tagged sentence, where each word in the sentence is annotated with its part of speech.
- Our goal will be to construct a model that recovers POS tags for sentences with high accuracy.
- One of the main challenges in POS tagging is ambiguity.
  - **Profits** soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.
  - the company **profits** from its endeavors …

> 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 ./.

Part-Of-Speech (POS) Tagging

- The input to the problem is a sentence.
- The output is a tagged sentence, where each word in the sentence is annotated with its part of speech.
- Our goal will be to construct a model that recovers POS tags for sentences with high accuracy.
- One of the main challenges in POS tagging is ambiguity.
  - Profits soared at Boeing Co., easily **topping** forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.
  - the **topping** on the cake …

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 ./.

Part-Of-Speech (POS) Tagging

- The input to the problem is a sentence.
- The output is a tagged sentence, where each word in the sentence is annotated with its part of speech.
- Our goal will be to construct a model that recovers POS tags for sentences with high accuracy.
- One of the main challenges in POS tagging is ambiguity.
    - Profits soared at Boeing Co., easily **topping** forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.
    - the **topping** on the cake …
    - “I **run**/V every morning.” vs. “He went for a **run**/N.”
    - “This bag is **light**/Adj.” vs. “Turn on the **light**/N.”
    - “The trash **can**/N is hard to find.” vs “I **can**/V do it.”

Part-Of-Speech (POS) Tagging

- The input to the problem is a sentence.
- The output is a tagged sentence, where each word in the sentence is annotated with its part of speech.
- Our goal will be to construct a model that recovers POS tags for sentences with high accuracy.
- One of the main challenges in POS tagging is **ambiguity**.
    - Profits soared at Boeing Co., easily **topping** forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.
    - the **topping** on the cake …
- A second challenge is the presence of words that are **rare**.
    - Mulally
    - Topping

Part-Of-Speech (POS) Tagging

- Individual words have **statistical preferences** for their part of speech.
  - E.g. ***quarter*** can be a noun or a verb, but is more likely to be a noun.
    - “A **quarter** of the cake is left.”
    - “Please **quarter** the apples before adding them to the pie.”

- The **context** has an important effect on the part of speech for a word.
  - The sequence **D N V** will be frequent in English,
    whereas the sequence **D V N** is much less likely.
    - The/D dog/N ran/V …

Conditional model

- $(x^{(i)}, y^{(i)})$, i = 1 … m

$$p(y|x)$$

$$f(x) = \arg\max_{y \in \mathcal{Y}} p(y|x)$$

Generative Models

- Model as **joint probability** instead of conditional probability.

$$p(x, y) = p(y)p(x|y)$$

  - p(y) is a prior probability distribution over labels y.
  - p(x | y) is the probability of generating the input x, given that the underlying label is y.

Bayes' theorem: $$p(y|x) = \frac{p(y)p(x|y)}{p(x)}$$

$$p(x) = \sum_{y \in \mathcal{Y}} p(x, y) = \sum_{y \in \mathcal{Y}} p(y)p(x|y)$$

Generative Models

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

$$\begin{aligned} f(x) &= \arg\max_y p(y|x) \\ &= \arg\max_y \frac{p(y)p(x|y)}{p(x)} \\ &= \arg\max_y p(y)p(x|y) \end{aligned}$$

* such decomposition of a joint probability to two terms p(y) and p(x|y) is often called as **Noisy-channel models**

Generative Models

- Our task is to learn a function from inputs x to labels $y = f(x)$. We assume training examples $(x^{(i)}, y^{(i)})$ for $i = 1 \dots n$.
- In the noisy channel approach, we use the training examples to estimate models $p(y)$ and $p(x|y)$. These models define a joint (generative) model

$$\boxed{p(x, y) = p(y)p(x|y)}$$

- Given a new test example x, we predict the label

$$\boxed{f(x) = \arg\max_{y \in \mathcal{Y}} p(y)p(x|y)}$$

Generative tagging models

- A finite set of vocabulary V
- A finite set of tags K
- S is a set of all tag-sequence pairs <$x_1^{(i)}$,..., $x_n^{(i)}$, $y_1^{(i)}$,...,$y_n^{(i)}$>
  - n >= 0
  - $x_j \in V$ for j = 1, …, n
  - $y_j \in K$ for j = 1, …, n

- A generative tagging model is then a function *p* such that:

1. *For any* $\langle x_1 \dots x_n, y_1 \dots y_n \rangle \in \mathcal{S}$,

$$p(x_1 \dots x_n, y_1 \dots y_n) \geq 0$$

2. *In addition,*

$$\sum_{\langle x_1 \dots x_n, y_1 \dots y_n \rangle \in \mathcal{S}} p(x_1 \dots x_n, y_1 \dots y_n) = 1$$

$$\boxed{f(x_1 \dots x_n) = \arg\max_{y_1 \dots y_n} p(x_1 \dots x_n, y_1 \dots y_n)}$$

Generative tagging models

- Three critical question for generative tagging models:

$$\boxed{p(x, y) = p(y)p(x|y)}$$

- How we define a generative tagging model $p(x_1 \ldots x_n, y_1 \ldots y_n)$?
- How do we estimate the parameters of the model from training examples?
- How do we efficiently find

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

for any input $x_1 \ldots x_n$?

Decoding problem

Generative Models

- Statistical Generative Models
  - n-gram Language Models
  - **Hidden Markov Models**
  - Maximum Entropy Markov Models
- Neural Generative Models
  - Recurrent Neural Networks (RNN)
  - Long Short Term Memory (LSTM)
- Autoregressive Transformer
  - GPT
  - LLama

Trigram Hidden Markov Models (Trigram HMMs)

- *A finite set of vocabulary V*
- *A finite set of tags K*
- *A parameter*

Context

Transition probability

$$q(s|u, v)$$

for any trigram (u, v, s) *such that* $s \in \mathcal{K} \cup \{STOP\}$, *and* $u, v \in \mathcal{K} \cup \{*\}$. *The value for* $q(s|u, v)$ *can be interpreted as the probability of seeing the tag* s *immediately after the bigram of tags* (u, v).

- *A parameter*

Statistical preferences

Emission probability

$$e(x|s)$$

for any $x \in \mathcal{V}$, $s \in \mathcal{K}$. *The value for* $e(x|s)$ *can be interpreted as the probability of seeing observation* x *paired with state* s.

Trigram Hidden Markov Models (Trigram HMMs)

- S is a set of all tag-sequence pairs <$x_1^{(i)}$,..., $x_n^{(i)}$, $y_1^{(i)}$,...,$y_n^{(i)}$>
  - $y_0 = y_{-1}$ = <s>/*/STRT
  - y_{n+1} = </s>/STOP/END

- Probability for any tag-sequence pairs <$x_1^{(i)}$,..., $x_n^{(i)}$, $y_1^{(i)}$,...,$y_n^{(i)}$> ∈S:

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

POS tagging Example

Probability for a sentence "the dog laughs" with tag sequence "D N V STOP" with Trigram HMMs as:

$$p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) = q(\mathtt{D}|*,*) \times q(\mathtt{N}|*,\mathtt{D}) \times q(\mathtt{V}|\mathtt{D},\mathtt{N}) \times q(\mathtt{STOP}|\mathtt{N},\mathtt{V}) \times e(\textit{the}|\mathtt{D}) \times e(\textit{dog}|\mathtt{N}) \times e(\textit{laughs}|\mathtt{V})$$

Prior probability: $\boxed{q(\mathtt{D}|*,*) \times q(\mathtt{N}|*,\mathtt{D}) \times q(\mathtt{V}|\mathtt{D},\mathtt{N}) \times q(\mathtt{STOP}|\mathtt{N},\mathtt{V})}$

Transition probability

Conditional probability: $\boxed{e(\textit{the}|\mathtt{D}) \times e(\textit{dog}|\mathtt{N}) \times e(\textit{laughs}|\mathtt{V})}$

$\boxed{p(\textit{the dog laughs}|\mathtt{D\ \ N\ \ V\ \ STOP})}$

Emission probability

HMMs as a stochastic process

- To generate the sequence pairs: $<y_1^{(i)},\dots,y_{n+1}^{(i)}, x_1^{(i)},\dots, x_n^{(i)}>$

1. Initialize $i = 1$ and $y_0 = y_{-1} = *$.
2. Generate y_i from the distribution
$$q(y_i|y_{i-2}, y_{i-1})$$
3. If $y_i = \text{STOP}$ then return $y_1 \dots y_i$, $x_1 \dots x_{i-1}$. Otherwise, generate x_i from the distribution
$$e(x_i|y_i),$$
set $i = i + 1$, and return to step 2.

Maximum-likelihood estimates

$$q(s|u,v) = \frac{c(u,v,s)}{c(u,v)}$$

Transition probability

$$e(x|s) = \frac{c(s \rightsquigarrow x)}{c(s)}$$

Emission probability

$c(u,v,s)$ - number of times the tag sequence (u,v,s) are seen in training data.

$c(s \rightsquigarrow x)$ - number of times the word x is seen paired with the tag s

Maximum-likelihood estimates

$$q(s|u,v) = \lambda_1 \times q_{ML}(s|u,v) + \lambda_2 \times q_{ML}(s|v) + \lambda_3 \times q_{ML}(s)$$

$$e(x|s) = \frac{c(s \leadsto x)}{c(s)}$$

$c(u,v,s)$ - number of times the tag sequence (u,v,s) are seen in training data.
$c(s \leadsto x)$ - number of times the word x is seen paired with the tag s

Decoding problem

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

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

Decoding problem (Naive approach)

For a sentence: "*the dog barks*" to decode from a set of possible tags K = {D, N, V}

- The possible tag sequences are:

D	D	D	STOP
D	D	N	STOP
D	D	V	STOP
D	N	D	STOP
D	N	N	STOP
D	N	V	STOP

. . .

3^3 = 27 possible sequences

K^n possible sequences for sentence with n words and K tags.

Decoding problem (Viterbi algorithm)

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

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

$$\begin{aligned} p(x_1 \dots x_n, y_1 \dots y_{n+1}) &= r(*, *, y_1, \dots, y_n) \times q(y_{n+1} | y_{n-1}, y_n) \\ &= r(*, *, y_1, \dots, y_n) \times q(\text{STOP} | y_{n-1}, y_n) \end{aligned}$$

$$\mathcal{K}_{-1} = \mathcal{K}_o = \{*\}$$

$$\mathcal{K}_k = \mathcal{K} \quad \text{for } k \in \{1 \ldots n\}$$

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

Basic Viterbi algorithm

Input: a sentence $x_1 \ldots x_n$, parameters $q(s|u,v)$ and $e(x|s)$.
Definitions: Define $\mathcal{K}$ to be the set of possible tags. Define $\mathcal{K}_{-1} = \mathcal{K}_0 = \{*\}$, and $\mathcal{K}_k = \mathcal{K}$ for $k = 1 \ldots n$.
Initialization: Set $\pi(0, *, *) = 1$.
Algorithm:

- For $k = 1 \ldots n$,
  - For $u \in \mathcal{K}_{k-1}$, $v \in \mathcal{K}_k$,

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

- **Return** $\max_{u \in \mathcal{K}_{n-1}, v \in \mathcal{K}_n} \left(\pi(n, u, v) \times q(\mathrm{STOP}|u, v)\right)$

The running time for the algorithm is $O(n|\mathcal{K}|^3)$

Backpointer: which previous tag (w) led to this score.

Input: a sentence $x_1 \dots x_n$, parameters $q(s|u,v)$ and $e(x|s)$.
Definitions: Define $\mathcal{K}$ to be the set of possible tags. Define $\mathcal{K}_{-1} = \mathcal{K}_0 = \{*\}$, and $\mathcal{K}_k = \mathcal{K}$ for $k = 1 \dots n$.
Initialization: Set $\pi(0, *, *) = 1$.
Algorithm:

- For $k = 1 \dots n$,
  - For $u \in \mathcal{K}_{k-1}$, $v \in \mathcal{K}_k$,

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

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

- Set $(y_{n-1}, y_n) = \arg\max_{u \in \mathcal{K}_{n-1}, v \in \mathcal{K}_n} \left(\pi(n,u,v) \times q(\text{STOP}|u,v)\right)$
- For $k = (n-2) \dots 1$,

$$y_k = bp(k+2, y_{k+1}, y_{k+2})$$

- **Return** the tag sequence $y_1 \dots y_n$

Maximum Entropy Markov Models (MEMMs)

Log-linear tagging models (MEMMs)

- Referred to as Maximum Entropy Markov Models (MEMMs)

$$\boxed{f(x_1 \ldots x_n) = \arg\max_{y_1 \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)}$$

Log-linear tagging models (MEMMs)

- Referred to as Maximum Entropy Markov Models (MEMMs)

$$
\begin{aligned}
& P(Y_1 = y_1 \dots Y_n = y_n | X_1 = x_1 \dots X_n = x_n) \\
= & \prod_{i=1}^{n} P(Y_i = y_i | X_1 = x_1 \dots X_n = x_n, Y_1 = y_1 \dots Y_{i-1} = y_{i-1}) \\
= & \prod_{i=1}^{n} P(Y_i = y_i | X_1 = x_1 \dots X_n = x_n, Y_{i-2} = y_{i-2}, Y_{i-1} = y_{i-1})
\end{aligned}
$$

$$
\boxed{\arg \max_{y_1 \dots y_n \in \mathcal{Y}(n)} p(y_1 \dots y_n | x_1 \dots x_n)}
$$

$$h_i = \langle y_{i-2}, y_{i-1}, x_1 \dots x_n, i \rangle$$

$$f(h_i, y) \in \mathbb{R}^d$$

$$P(Y_i = y_i | X_1 = x_1 \dots X_n = x_n, Y_{i-2} = y_{i-2}, Y_{i-1} = y_{i-1})$$
$$= \frac{\exp\left(\theta \cdot f(h_i, y_i)\right)}{\sum_{y \in \mathcal{K}} \exp\left(\theta \cdot f(h_i, y)\right)}$$

$$\boxed{p(y_i | h_i; \theta) = \frac{\exp\left(\theta \cdot f(h_i, y_i)\right)}{\sum_{y \in \mathcal{K}} \exp\left(\theta \cdot f(h_i, y)\right)}}$$

Components of a trigram MEMM

- *A set of words $\mathcal{V}$ (this set may be finite, countably infinite, or even uncountably infinite).*
- *A finite set of tags $\mathcal{K}$.*
- *Given $\mathcal{V}$ and $\mathcal{K}$, define $\mathcal{H}$ to be the set of all possible histories. The set $\mathcal{H}$ contains all four-tuples of the form $\langle y_{-2}, y_{-1}, x_1 \ldots x_n, i\rangle$, where $y_{-2} \in \mathcal{K} \cup \{*\}$, $y_{-1} \in \mathcal{K} \cup \{*\}$, $n \geq 1$, $x_i \in \mathcal{V}$ for $i = 1 \ldots n$, $i \in \{1 \ldots n\}$. Here $*$ is a special "start" symbol.*
- *An integer d specifying the number of features in the model.*
- *A function $f : \mathcal{H} \times \mathcal{K} \rightarrow \mathbb{R}^d$ specifying the features in the model.*
- *A parameter vector $\theta \in \mathbb{R}^d$.*

$$p(y_1 \ldots y_n | x_1 \ldots x_n) = \prod_{i=1}^{n} p(y_i | h_i; \theta)$$

$$p(y_i | h_i; \theta) = \frac{\exp\left(\theta \cdot f(h_i, y_i)\right)}{\sum_{y \in \mathcal{K}} \exp\left(\theta \cdot f(h_i, y)\right)}$$

$$\boxed{\arg \max_{y_1 \ldots y_n \in \mathcal{Y}(n)} p(y_1 \ldots y_n | x_1 \ldots x_n)}$$

Features in Trigram MEMMs

$$h_i = \langle y_{i-2}, y_{i-1}, x_1 \ldots x_n, i \rangle$$

$$f(h_i, y) \in \mathbb{R}^d \qquad y \in \mathcal{K}$$

Word/tag features:

$$f_{100}(h, y) = \begin{cases} 1 & \text{if } x_i \text{ is } \texttt{base} \text{ and } y = \texttt{VB} \\ 0 & \text{otherwise} \end{cases}$$

Features in Trigram MEMMs

$$h_i = \langle y_{i-2}, y_{i-1}, x_1 \ldots x_n, i \rangle$$

$$f(h_i, y) \in \mathbb{R}^d \qquad y \in \mathcal{K}$$

Prefix and Suffix features:

$$f_{101}(h, y) = \begin{cases} 1 & \text{if } x_i \text{ ends in } \texttt{ing} \text{ and } y = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

Features in Trigram MEMMs

$$h_i = \langle y_{i-2}, y_{i-1}, x_1 \ldots x_n, i \rangle$$

$$f(h_i, y) \in \mathbb{R}^d \qquad y \in \mathcal{K}$$

Trigram, Bigram and Unigram Tag features:

$$f_{103}(h, y) = \begin{cases} 1 & \text{if } \langle y_{-2}, y_{-1}, y \rangle = \langle \text{DT, JJ, VB} \rangle \\ 0 & \text{otherwise} \end{cases}$$

Features in Trigram MEMMs

Trigram, Bigram and Unigram Tag features:

$$f_{104}(h, y) = \begin{cases} 1 & \text{if } \langle y_{-1}, y \rangle = \langle \text{JJ, VB} \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$f_{105}(h, y) = \begin{cases} 1 & \text{if } \langle y \rangle = \langle \text{VB} \rangle \\ 0 & \text{otherwise} \end{cases}$$

Features in Trigram MEMMs

Other Contextual Features:

$$f_{106}(h, y) = \begin{cases} 1 & \text{if previous word } x_{i-1} = \textit{the} \text{ and } y = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{107}(h, y) = \begin{cases} 1 & \text{if next word } x_{i+1} = \textit{the} \text{ and } y = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

Parameter Estimation in Trigram MEMMs

$$P(Y_i = y_i | X_1 = x_1 \ldots X_n = x_n, Y_{i-2} = y_{i-2}, Y_{i-1} = y_{i-1})$$

$$= \frac{\exp\left(\theta \cdot f(h_i, y_i)\right)}{\sum_{y \in \mathcal{K}} \exp\left(\theta \cdot f(h_i, y)\right)}$$

$$\boxed{p(y_i | h_i; \theta) = \frac{\exp\left(\theta \cdot f(h_i, y_i)\right)}{\sum_{y \in \mathcal{K}} \exp\left(\theta \cdot f(h_i, y)\right)}}$$

Parameter Estimation in Trigram MEMMs

Log-likelihood function

$$L(\theta) = \sum_{k=1}^{m} \sum_{i=1}^{n_k} \log p(y_i^{(k)} | h_i^{(k)}; \theta)$$

$$\theta^* = \arg\max_{\theta \in \mathbb{R}^d} L(\theta)$$

Parameter Estimation in Trigram MEMMs

Regularized log-likelihood function

$$L(\theta) = \sum_{k=1}^{m} \sum_{i=1}^{n_k} \log p(y_i^{(k)} | h_i^{(k)}; \theta) - \frac{\lambda}{2} \sum_{j=1}^{d} \theta_j^2$$

A regularization term, which penalizes large parameter values

$$\theta^* = \arg\max_{\theta \in \mathbb{R}^d} L(\theta)$$

$$p(y|x;\underline{w}) = \frac{\exp\left(\underline{w}\cdot\underline{\phi}(x,y)\right)}{\sum_{y'\in\mathcal{Y}}\exp\left(\underline{w}\cdot\underline{\phi}(x,y')\right)}$$

$$\boxed{p(y_i|h_i;\theta) = \frac{\exp\left(\theta\cdot f(h_i,y_i)\right)}{\sum_{y\in\mathcal{K}}\exp\left(\theta\cdot f(h_i,y)\right)}}$$

$$\frac{\partial}{\partial w_j}L(\underline{w}) = \sum_i \phi_j(x_i,y_i) - \sum_i\sum_y p(y|x_i;\underline{w})\phi_j(x_i,y)$$

$$\boxed{L(\theta) = \sum_{k=1}^{m}\sum_{i=1}^{n_k}\log p(y_i^{(k)}|h_i^{(k)};\theta) - \frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2}$$

Decoding (Viterbi Algorithm)

Input: A sentence $x_1 \ldots x_n$. A set of possible tags $\mathcal{K}$. A model (for example a log-linear model) that defines a probability

$$p(y|h;\theta)$$

for any h, y pair where h is a history of the form $\langle y_{-2}, y_{-1}, x_1 \ldots x_n, i\rangle$, and $y \in \mathcal{K}$.
Definitions: Define $\mathcal{K}_{-1} = \mathcal{K}_0 = \{*\}$, and $\mathcal{K}_k = \mathcal{K}$ for $k = 1 \ldots n$.
Initialization: Set $\pi(0, *, *) = 1$.
Algorithm:

- For $k = 1 \ldots n$,
  - For $u \in \mathcal{K}_{k-1}, v \in \mathcal{K}_k$,

$$\pi(k, u, v) = \max_{w \in \mathcal{K}_{k-2}} (\pi(k-1, w, u) \times p(v|h;\theta))$$

$$bp(k, u, v) = \arg\max_{w \in \mathcal{K}_{k-2}} (\pi(k-1, w, u) \times p(v|h;\theta))$$

    where $h = \langle w, u, x_1 \ldots x_n, k\rangle$.

- Set $(y_{n-1}, y_n) = \arg\max_{u \in \mathcal{K}_{n-1}, v \in \mathcal{K}_n} \pi(n, u, v)$
- For $k = (n-2) \ldots 1$,

$$y_k = bp(k+2, y_{k+1}, y_{k+2})$$

- **Return** the tag sequence $y_1 \ldots y_n$

References and wider reading

- Hidden Markov Model (https://www.cs.columbia.edu/~mcollins/hmms-spring2013.pdf)
- Maximum Entropy Markov Model (http://www.cs.columbia.edu/~mcollins/fall2014-loglineartaggers.pdf)
- Conditional Random Field (http://www.cs.columbia.edu/~mcollins/crf.pdf)