Natural Language Processing

Lecture 3 - Log Linear Models

April 2025

Eyal Ben-David

*Partially Based on Coursera's NLP course by Michael Collins.

(DON'T FORGET RECORDING!!)

Course Logistics & updates

- First assignment will be released on this week
  - You should have a partner & a machine
- Office hours will be dedicated to helping you in your HW assignments

(Recap) Language Modeling

Goal:

Assign a probability to a given sentence or sequence of words.

P("Insanity is doing the same thing over and over again and expecting different results") = ?

Another way of thinking about this:

Given a sequence of words, what is likely to be the next word?

P("expecting" | "Insanity is doing the same thing over and over again and… ") = ?

(Recap) Bigram Language Model

Assumes a **first order markov** assumption.

→ The Bigram MLE of text W is:

$$P(W) = \prod_{i=0}^{n} p(w_i \mid w_{i-1}) = \prod_{i=0}^{n} \frac{count(w_{i-1}, w_i)}{count(w_i \quad)}$$

(Recap) Vector Perspective of Bigram Models

input vector

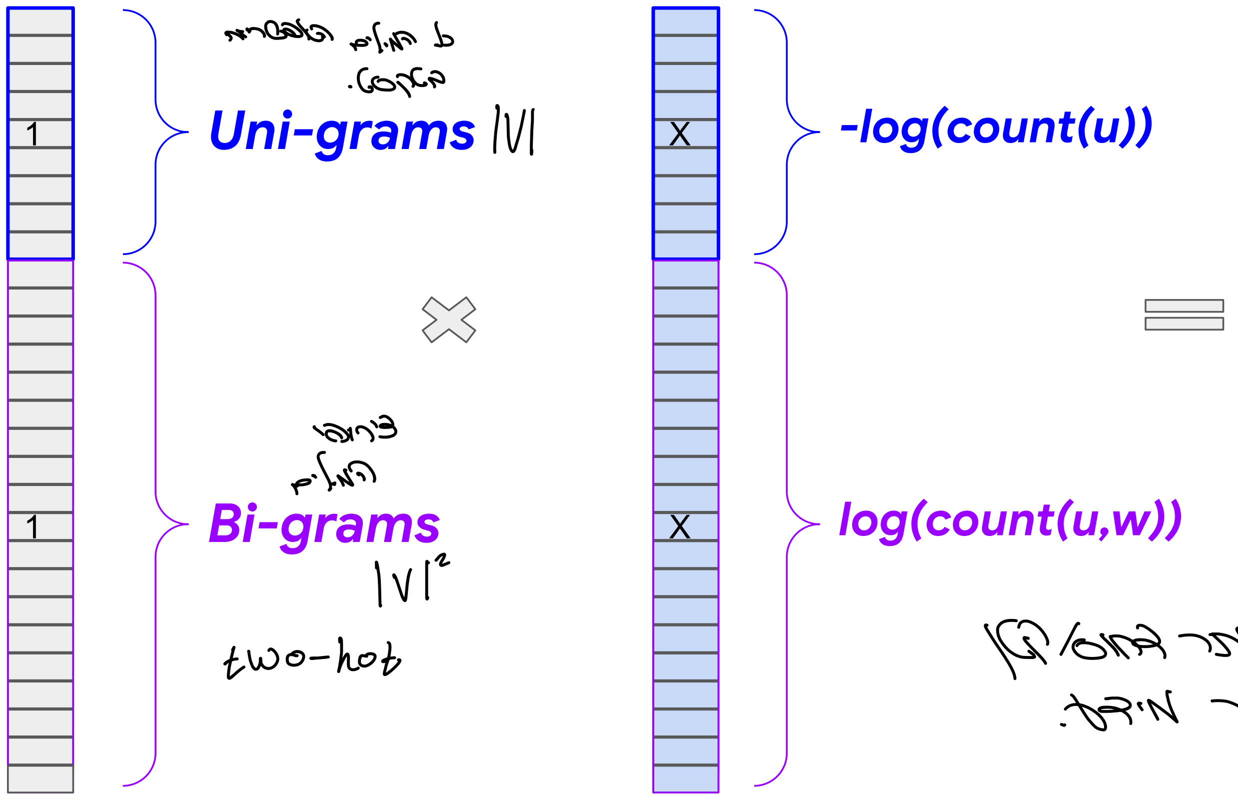
Representation Vector

(One-hot) – עבור אפסים ו-1 יחיד.

Model parameters

הכמות שלו לכל קורפוס. משקולות.

Text (bigram)



$$= \log\left(\frac{\mathrm{count}(u,w)}{\mathrm{count}(u)}\right) =$$

$$\log\left(p(w\,|\,u)\right)$$

נקבל מינוס יותר קטנים/ גדול
אז שם יותר מינוס.

(Recap) Trigram HMM

Definition

For any sentence $x_1, x_2, \ldots, x_n$ and any sequence of tags $y_1, y_2, \ldots, y_{n+1}$, the joint probability of the sentence and the tags is:

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

(Recap) Main Drawbacks of N-gram Modeling

- How do we decide what *N* would be?
- Storage
- Sparsity
  - How to deal with OOV (unigrams, n-grams)?
  - We can perform:
    - Smoothness
    - Linear interpolation of N-gram models

Today's Agenda

- **Motivation - modeling more information**
- Log linear models
- Parameter estimation (log linear)
- Log linear model for tagging (MEMM)
- Inference

Four-gram Language Models

Estimate: P("expecting" | "Insanity is doing the same thing over and over again and... ")

- Given "history" - $w_1, w_2, \ldots, w_{n-1}$
- Here, "history" is based only on n-grams

Thus, we estimate as follows:

hyper-parameter

$$P(expecting \,|\, w_1, w_2, \ldots, w_{n-1}) = \lambda_1 \cdot p_{ML}(expecting \,|\, over\, again\, and) + \lambda_2 \cdot p_{ML}(expecting \,|\, again\, and) + \lambda_3 \cdot p_{ML}(expecting \,|\, and) + \lambda_4 \cdot p_{ML}(expecting)$$

back-off

Four-gram Language Models

$$
\begin{aligned}
P(expecting \,|\, w_1, w_2, \ldots, w_{n-1}) = {} & \lambda_1 \cdot p_{ML}(expecting \,|\, over\ again\ and) + \\
& \lambda_2 \cdot p_{ML}(expecting \,|\, again\ and) + \\
& \lambda_3 \cdot p_{ML}(expecting \,|\, and) + \\
& \lambda_4 \cdot p_{ML}(expecting)
\end{aligned}
$$

Why just words? Other features may be useful also.

List of Useful Features (besides n-grams)

$$P(expecting \,|\, w_1, w_2, \ldots, w_{n-1}) =$$

"פיצ'רים"

- "expecting" does not occur on $w_1, w_2, \ldots, w_{n-1}$
- Previous word is a conjunction (מילת חיבור)
- "doing" and "and" appears in $w_1, w_2, \ldots, w_{n-1}$
- who wrote the text
- what is the text about
- the grammatical structure of the sentence

Try suggesting more...

How To Naively Implement This

$$
\begin{aligned}
P(\text{expecting} \mid w_1, w_2, \ldots, w_{n-1}) = \; & \lambda_1 \cdot p_{ML}(\text{expecting} \mid w_1, w_2, \ldots, w_{n-1} \neq \text{expecting}) + \\
& \lambda_2 \cdot p_{ML}(\text{expecting} \mid \underbrace{POS}_{\text{Part of Speech.}}(w_{n-1}) = CONJ) + \\
& \lambda_3 \cdot p_{ML}(\text{expecting} \mid \{\text{doing,and}\} \in w_1, w_2, \ldots, w_{n-1}) + \\
& \lambda_4 \cdot p_{ML}(\text{expecting} \mid \text{text writer}) + \\
& \lambda_5 \cdot p_{ML}(\text{expecting} \mid \text{text subject}) + \\
& \lambda_6 \cdot p_{ML}(\text{expecting} \mid \text{grammar})
\end{aligned}
$$

The Naive Implementation (as a thought experiment)

$$
\begin{aligned}
P(\text{expecting} \mid w_1, w_2, \ldots, w_{n-1}) = \; & \lambda_1 \cdot p_{ML}(\text{expecting} \mid w_1, w_2, \ldots, w_{n-1} \neq \text{expecting}) + \\
& \lambda_2 \cdot p_{ML}(\text{expecting} \mid POS(w_{n-1}) = CONJ) + \\
& \lambda_3 \cdot p_{ML}(\text{expecting} \mid \{\text{doing,and}\} \in w_1, w_2, \ldots, w_{n-1}) + \\
& \lambda_4 \cdot p_{ML}(\text{expecting} \mid \text{text writer}) + \\
& \lambda_5 \cdot p_{ML}(\text{expecting} \mid \text{text subject}) + \\
& \lambda_6 \cdot p_{ML}(\text{expecting} \mid \text{grammar})
\end{aligned}
$$

Adds many hyperparameters (lambdas) to our model.

- In practice, this quickly becomes <u>unwieldy</u> (מגושם, מסורבל)

In Today's lecture: <u>Log linear models</u> can incorporate massive features in an elegant way


SHOW ME MORE
SHOW ME MORE
makeameme.org

A Second Example: Part of Speech Tagging

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 ./.

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

A Second Example: Part of Speech Tagging

token classification

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere .

- There are many possible tags in the position ?? {NN, NNS, Vt, Vi, IN, DT, . . . }
- The task: model the distribution

$$p(t_i|t_1, \ldots, t_{i-1}, w_1 \ldots w_n)$$

where t_i is the i'th tag in the sequence, w_i is the i'th word

A Second Example: Part of Speech Tagging

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere .

- The task: model the distribution

$$p(t_i|t_1, \ldots, t_{i-1}, w_1 \ldots w_n)$$

where t_i is the i'th tag in the sequence, w_i is the i'th word

- Again: many "features" of $t_1, \ldots, t_{i-1}, w_1 \ldots w_n$ may be relevant

$$\begin{array}{lcl} q_{ML}(\text{NN} & | & w_i = \text{base}) \\ q_{ML}(\text{NN} & | & t_{i-1} \text{ is JJ}) \\ q_{ML}(\text{NN} & | & w_i \text{ ends in "e"}) \\ q_{ML}(\text{NN} & | & w_i \text{ ends in "se"}) \\ q_{ML}(\text{NN} & | & w_{i-1} \text{ is "important"}) \\ q_{ML}(\text{NN} & | & w_{i+1} \text{ is "from"}) \end{array}$$


SHOW ME MORE
SHOW ME MORE
makeameme.org

Today's Agenda

- Motivation - modeling more information
- **Log linear models**
- Parameter estimation (log linear)
- Log linear model for tagging (MEMM)
- Inference

Log-Linear Models

The general problem:

- We have an input X
- Have a finite label set Y
- Aim to provide a conditional probability $p(y|x)$

  For any $x \in X$ and $y \in Y$

**Note, this is in contrast to HMM that provides a joint probability $p(y,x)$.

Example - Language Modeling

- X is a "history" $w_1, w_2, \ldots, w_{n-1}$.

e.g.,

Third, the notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical

- y is an "outcome" w_n

Feature Vector Representation

- Aim is to provide a conditional probability p(y | x) for "decision" y given "history" x
- A feature is a function $f_k(x, y) \in \mathbb{R}$
  (Often **binary features** or **indicator functions** $f_k(x, y) \in \{\mathbb{0}, \mathbb{1}\}$).

- Say we have m features f_k for $k \in 1, 2, \ldots, m$
  → Our feature vector $f(x, y) \in \mathbb{R}^{\mathbb{m}}$

Language Modeling

Third, the notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical

מקבלים וקטור אשר מופעל
עליו $f(x,y)$ ומקבל וקטור חדש.

- Example features:

$$F(x,y) = \begin{bmatrix} \\ f_k(x,y) * \lambda_k \\ \\ \end{bmatrix}$$

וקטור אינדיקטורים.
איזה פיצ'רים להפעיל

מייצג את ה ד
שני לבלוק ושני
לערות כללים

$$f_1(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model and } w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model, } w_{i-2} = \texttt{any, } w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

Language Modeling

Third, the notion “grammatical in English” cannot be identified in any way with the notion “high order of statistical approximation to English”. It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical

- Example features:

Q1: What is x? היסטוריה מילים

Q2: What is y? מילה עתידית

$$f_1(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model} \text{ and } w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-2} = \texttt{any},\ w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

Language Modeling

Third, the notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical

- Example features:

$$\text{(Unigram)} \quad f_1(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{(Bigram)} \quad f_2(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model} \text{ and } w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{(Trigram)} \quad f_3(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-2} = \texttt{any},\ w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

Language Modeling

$$f_4(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-2} = \texttt{any} \\ 0 & \text{otherwise} \end{cases}$$

$$f_5(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-1} \text{ is an adjective} \\ 0 & \text{otherwise} \end{cases}$$

$$f_6(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-1} \text{ ends in ``ical''} \\ 0 & \text{otherwise} \end{cases}$$

$$f_7(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model}, \text{ author} = \text{Chomsky} \\ 0 & \text{otherwise} \end{cases}$$

$$f_8(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model}, \text{ ``model'' is not in } w_1, \ldots w_{i-1} \\ 0 & \text{otherwise} \end{cases}$$

$$f_9(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model}, \text{ ``grammatical'' is in } w_1, \ldots w_{i-1} \\ 0 & \text{otherwise} \end{cases}$$

In Practice

- We follow a markovian assumption to the define the history (X) for w_i
- E.g., a trigram based history: הנחה מרקובית מסדר שני.

$$X_i = \text{History } for\ w_i = \langle t_{i-2}, t_{i-1}, t_i, w_{i-2}, w_{i-1}, w_i \rangle$$

- We would probably include a feature for each tri-gram that appeared enough times in the training data.

$$f_{N(u,v,w)}(x,y) = \begin{cases} 1 & \text{if } y = w,\ w_{i-2} = u,\ w_{i-1} = v \\ 0 & \text{otherwise} \end{cases}$$

** N(u,v,w) maps each trigram to a unique integer.

The Final Result

- We can come up with any question regarding the history and define a feature.

Note, the history includes the tags and the words.

For a given history $x \in X$ each label $y \in Y$
is mapped to a different feature vector:

$$f(\langle x_{i-2} = \text{any}, x_{i-1} = \text{statistical}\rangle, x_i = \text{outcome}) = \quad 1001011001001100110$$
$$f(\langle x_{i-2} = \text{any}, x_{i-1} = \text{statistical}\rangle, x_i = \text{time}) = \quad 0110010101011110010$$
$$f(\langle x_{i-2} = \text{any}, x_{i-1} = \text{statistical}\rangle, x_i = \text{background}) = \quad 0001111101001100100$$
$$f(\langle x_{i-2} = \text{any}, x_{i-1} = \text{statistical}\rangle, \underbrace{x_i}_{y_i} = \text{course}) = \quad 0001011011000000010$$

For POS Tagging

time-stamp
#word

- Each X is a “history” $<t_1, t_2,\ldots, t_{n-1}, w_1, w_2,\ldots, w_n, i>$
- y is a POS tag. E.g., NN, NNS, DT
- We have m features $f_k(x, y)$ for k=1,...,m

Example features:

$$
\begin{aligned}
f_1(x, y) &= \begin{cases} 1 & \text{if current word } w_i \text{ is base and } y = \texttt{Vt} \\ 0 & \text{otherwise} \end{cases} \\
f_2(x, y) &= \begin{cases} 1 & \text{if current word } w_i \text{ ends in } \texttt{ing} \text{ and } y = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases} \\
&\ldots
\end{aligned}
$$

The Full Feature Set in Ratnaparkhi, 1996

- Word/tag features for all word/tag pairs, e.g.,

$$f_{100}(x,y) = \begin{cases} 1 & \text{if current word } w_i \text{ is base and } y = \texttt{Vt} \\ 0 & \text{otherwise} \end{cases}$$

$$F(x,y) = \begin{bmatrix} \begin{bmatrix} f_{100} \end{bmatrix} \\ \begin{bmatrix} f_{101} \end{bmatrix} \\ \begin{bmatrix} f_{102} \end{bmatrix} \end{bmatrix}$$

לכל מילה קיימת שורה בוקטור.

לכל היותר $|V| \times |T|$ - עבור כל מחרוזת אפשרית לכל תיוג אפשרי שלנו

- Spelling features for all prefixes/suffixes of length ≤ 4, e.g.,

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

למשל: נסתכל רק על סיומות עד אורך 4 וכן שמופיעות לפחות 5 פעמים.

$$f_{102}(h,t) = \begin{cases} 1 & \text{if current word } w_i \text{ starts with } \texttt{pre} \text{ and } y = \texttt{NN} \\ 0 & \text{otherwise} \end{cases}$$

The Full Feature Set in Ratnaparkhi, 1996

- Contextual Features, e.g.,

$$f_{103}(x,y) = \begin{cases} 1 & \text{if } \langle t_{i-2}, t_{i-1}, y\rangle = \langle \text{DT, JJ, Vt}\rangle \\ 0 & \text{otherwise} \end{cases}$$

$$f_{104}(x,y) = \begin{cases} 1 & \text{if } \langle t_{i-1}, y\rangle = \langle \text{JJ, Vt}\rangle \\ 0 & \text{otherwise} \end{cases}$$

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

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

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

The Final Result (POS)

- We can come up with any question regarding the history and define a feature.

Note, the history includes the tags and the words.

For a given history $x \in X$ each label $y \in Y$
is mapped to a different feature vector:

$$f(x = \langle \text{JJ, DT}, \langle \text{ Hispaniola, } \ldots \rangle, 6\rangle, y = Vt) = \text{ 1001011001001100110}$$

$$f(x = \langle JJ, DT, \langle Hispaniola, \ldots \rangle, 6\rangle, y = \text{JJ}) = \text{ 0110010101011110010}$$

$$f(x = \langle JJ, DT, \langle Hispaniola, \ldots \rangle, 6\rangle, y = \text{NN}) = \text{0001111101001100100}$$

$$f(x = \langle JJ, DT, \langle Hispaniola, \ldots \rangle, 6\rangle, y = \text{IN}) = \text{ 0001011011000000010}$$

Parameter Vector

- Given features f_k for $k \in 1, 2, \ldots, m$

  Define a **parameter vector** $v \in \mathbb{R}^m$

  Each (x,y) pair is mapped into a ***score:*** $v \cdot f(x, y) = \sum_{k=1}^{m} v_k \cdot f_k(x, y)$

  Scoring function

  In our language modeling example:

$$
\begin{aligned}
v \cdot f(x, model) &= 5.6 & \qquad v \cdot f(x, the) &= -3.2 \\
v \cdot f(x, is) &= 1.5 & \qquad v \cdot f(x, of) &= 1.3 \\
v \cdot f(x, models) &= 4.5 & \qquad \ldots &
\end{aligned}
$$

Log-Linear Models

- We have an input domain $\mathcal{X}$ and a finite label space $\mathcal{Y}$
- A **feature** vector $f(x, y) \in \mathbb{R}^m$
- A **parameter** vector $v \in \mathbb{R}^m$
- We aim to provide a **conditional probability** $p(y \mid x)$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$
- We'll use the **log-linear model** to define

$$p(y \mid x; v) = \frac{e^{v \cdot f(x,y)}}{\sum_{y' \in \mathcal{Y}} e^{v \cdot f(x,y')}}$$

Soft max

מעלה ערכים גבוהים,

ומנמיך נמוכים.

Name Origin - Log Linear Models

$$\log\left(p(y \mid x; v)\right) = \underbrace{v \cdot f(x, y)}_{\text{Linear term}} - \underbrace{\log\left(\sum_{y' \in \mathcal{Y}} e^{v \cdot f(x, y')}\right)}_{\text{Normalization term}}$$

פונק' דיפרנציאבילית (גזירה) ← מקלה min\max מודל.

Today’s Agenda

- Motivation - modeling more information
- Log linear models
- **Parameter estimation (log linear)**
- Log linear model for tagging (MEMM)
- Inference

Maximum Likelihood Estimation

- Given a training sample $\left(x^{(i)}, y^{(i)}\right)$ for $i = 1, \ldots, n$,

  each $\left(x^{(i)}, y^{(i)}\right) \in \mathcal{X} \times \mathcal{Y}$

gain → not loss

$$v_{ML} = \arg\max_{v \in \mathbb{R}^m} L(v)$$

$$L(v) = \sum_{i=1}^{n} \log\left(p\left(y^{(i)} \mid x^{(i)}; v\right)\right) = \sum_{i=1}^{n} \left[v \cdot f\left(x^{(i)}, y^{(i)}\right) - \log\left(\sum_{y' \in \mathcal{Y}} e^{v \cdot f\left(x^{(i)}, y'\right)}\right) \right]$$

Calculating the Maximum Likelihood Estimates

- Need to maximize:

$$L(v) = \sum_{i=1}^{n} v \cdot f(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \log \sum_{y' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, y')}$$

- Calculating gradients:

$$\frac{dL(v)}{dv_k} = \sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \frac{\sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') e^{v \cdot f(x^{(i)}, y')}}{\sum_{z' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, z')}}$$

$$= \sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') \frac{e^{v \cdot f(x^{(i)}, y')}}{\sum_{z' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, z')}}$$

חישוב ידני

$$= \underbrace{\sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)})}_{\text{Empirical counts}} - \underbrace{\sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') p(y' \mid x^{(i)}; v)}_{\text{Expected counts}}$$

bias → $\hat{\theta} - \theta$

צורה ב L

סכום הפיצ'רים הנ' בקלט
מסט אימון

תלוי
בפרמטרים
ולכן חישוב איטרטיבי.

Gradient Ascent

- We need to maximize L(v), where:

$$\frac{dL(v)}{dv} = \sum_{i=1}^{n} f(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f(x^{(i)}, y') p(y' \mid x^{(i)}; v)$$

Initialization: $v = 0$

Iterate until convergence:

- Calculate $\Delta = \frac{dL(v)}{dv}$
- Calculate $\beta_* = \mathrm{argmax}_\beta L(v + \beta\Delta)$ (Line Search)
- Set $v \leftarrow v + \beta_* \Delta$

L2 Regularization

לתרגיל בית -
לא לתת ערכים גבוהים מידי
לא לעשות Overfitting

- We add a penalty for large weights:
  - Modified loss function

$$L(v) \;=\; \sum_{i=1}^{n} v \cdot f(x^{(i)}, y^{(i)}) - \sum_{i=1}^{n} \log \sum_{y' \in \mathcal{Y}} e^{v \cdot f(x^{(i)}, y')} - \frac{\lambda}{2} \sum_{k=1}^{m} v_k^2$$

  - Calculating gradients:

$$\frac{dL(v)}{dv_k} \;=\; \underbrace{\sum_{i=1}^{n} f_k(x^{(i)}, y^{(i)})}_{\text{Empirical counts}} - \underbrace{\sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} f_k(x^{(i)}, y') p(y' \mid x^{(i)}; v)}_{\text{Expected counts}} - \lambda v_k$$

Today’s Agenda

- Motivation - modeling more information
- Log linear models
- Parameter estimation (log linear)
- **Log linear model for tagging (MEMM)**
- Inference

Log Linear Models for Tagging Maximum-entropy Markov Models (MEMM)

Recap: Part of Speech (POS)

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 ./.

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

Recap: Named 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.

Recap: Named 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/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

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

Goal

Training set:

1 Pierre/NNP Vinken/NNP ,/, 61/CD years/NNS old/JJ ,/, will/MD join/VB the/DT board/NN as/IN a/DT nonexecutive/JJ director/NN Nov./NNP 29/CD ./.

2 Mr./NNP Vinken/NNP is/VBZ chairman/NN of/IN Elsevier/NNP N.V./NNP ,/, the/DT Dutch/NNP publishing/VBG group/NN ./.

3 Rudolph/NNP Agnew/NNP ,/, 55/CD years/NNS old/JJ and/CC chairman/NN of/IN Consolidated/NNP Gold/NNP Fields/NNP PLC/NNP ,/, was/VBD named/VBN a/DT nonexecutive/JJ director/NN of/IN this/DT British/JJ industrial/JJ conglomerate/NN ./.

...

38,219 It/PRP is/VBZ also/RB pulling/VBG 20/CD people/NNS out/IN of/IN Puerto/NNP Rico/NNP ,/, who/WP were/VBD helping/VBG Huricane/NNP Hugo/NNP victims/NNS ,/, and/CC sending/VBG them/PRP to/TO San/NNP Francisco/NNP instead/RB ./.

From the training set, induce a function/algorithm that maps new sentences to their tag sequences.

Log Linear Models for Tagging

- We have an input sentence $w_{[1:n]} = w_1, w_2, \ldots, w_n$
  (w_i is the i'th word in the sentence)

- We have a tag sequence $t_{[1:n]} = t_1, t_2, \ldots, t_n$
  (t_i is the i'th tag in the sentence)

- We'll use an log-linear model to define

$$p(t_1, t_2, \ldots, t_n | w_1, w_2, \ldots, w_n)$$

  for any sentence $w_{[1:n]}$ and tag sequence $t_{[1:n]}$ of the same length.
  (Note: contrast with HMM that defines $p(t_1 \ldots t_n, w_1 \ldots w_n)$)

- Then the most likely tag sequence for $w_{[1:n]}$ is

$$t^*_{[1:n]} = \text{argmax}_{t_{[1:n]}} p(t_{[1:n]} | w_{[1:n]})$$

Trigram Log Linear Tagger

A Trigram Log-Linear Tagger:

$p(t_{[1:n]}|w_{[1:n]}) = \prod_{j=1}^{n} p(t_j \mid w_1 \ldots w_n, t_1 \ldots t_{j-1})$ Chain rule

$= \prod_{j=1}^{n} p(t_j \mid w_1, \ldots, w_n, t_{j-2}, t_{j-1})$

Independence assumptions

- We take $t_0 = t_{-1} = *$
- Independence assumption: each tag only depends on previous two tags

$$p(t_j|w_1, \ldots, w_n, t_1, \ldots, t_{j-1}) = p(t_j|w_1, \ldots, w_n, t_{j-2}, t_{j-1})$$

Example - History Tuple

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere .

- t_{-2}, t_{-1} = DT, JJ
- $w_{[1:n]} = \langle Hispaniola, quickly, became, \ldots, Hemisphere, . \rangle$
- $i = 6$

- A **history** is a 4-tuple $\langle t_{-2}, t_{-1}, w_{[1:n]}, i \rangle$
- t_{-2}, t_{-1} are the previous two tags.
- $w_{[1:n]}$ are the n words in the input sentence.
- i is the index of the word being tagged

Feature Set (Ratnaparkhy, 1996)

- Word/tag features for all word/tag pairs, e.g.,

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

- Spelling features for all prefixes/suffixes of length ≤ 4, e.g.,

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

$$f_{102}(h,t) = \begin{cases} 1 & \text{if current word } w_i \text{ starts with } \texttt{pre} \text{ and } t = \texttt{NN} \\ 0 & \text{otherwise} \end{cases}$$

....

Today's Agenda

- Motivation - modeling more information
- Log linear models
- Parameter estimation (log linear)
- Log linear model for tagging (MEMM)
- **Inference**

The Viterbi Algorithm - MEMM

Problem: for an input $w_1 \ldots w_n$, find

$$\arg\max_{t_1 \ldots t_n} p(t_1 \ldots t_n \mid w_1 \ldots w_n)$$

We assume that p takes the form

$$p(t_1 \ldots t_n \mid w_1 \ldots w_n) = \prod_{i=1}^{n} q(t_i | t_{i-2}, t_{i-1}, w_{[1:n]}, i)$$

(In our case $q(t_i | t_{i-2}, t_{i-1}, w_{[1:n]}, i)$ is the estimate from a log-linear model.)

The Viterbi Algorithm - MEMM

- Define n to be the length of the sentence
- Define

$$r(t_1 \dots t_k) = \prod_{i=1}^{k} q(t_i | t_{i-2}, t_{i-1}, w_{[1:n]}, i)$$

$$\pi_6 = \begin{array}{c|ccc} t_{i-1} \backslash t_i & NN & VB & DT \\ \hline NN & & & \\ VB & o & & \\ DT & & & \end{array}$$

- Define a dynamic programming table

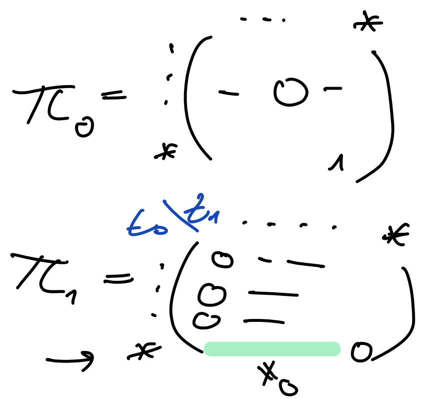
$\pi(k, u, v) \quad = \quad$ maximum probability of a tag sequence ending in tags u, v at position k

הסתברות
המקסימלית שהתיוג
הקודם הוא
VB והתיוג
הנוכחי הוא NN

that is,

$$\pi(k, u, v) = \max_{\langle t_1, \dots, t_{k-2} \rangle} r(t_1 \dots t_{k-2}, u, v)$$

A Recursive Definition



Base case:

מילה ריקה

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

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

$$\pi(k, u, v) = \max_{t \in \mathcal{S}_{k-2}} \left(\pi(k-1, t, u) \times q(v | t, u, w_{[1:n]}, k) \right)$$

where $\mathcal{S}_k$ is the set of possible tags at position k

The full Algorithm (with backpointers)

Input: a sentence $w_1 \dots w_n$, log-linear model that provides $q(v|t, u, w_{[1:n]}, i)$ for any tag-trigram t, u, v, for any $i \in \{1 \dots n\}$
Initialization: Set $\pi(0, *, *) = 1$.
Algorithm:

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

$$\pi(k, u, v) = \max_{t \in \mathcal{S}_{k-2}} \left(\pi(k-1, t, u) \times q(v|t, u, w_{[1:n]}, k)\right)$$
$$bp(k, u, v) = \arg\max_{t \in \mathcal{S}_{k-2}} \left(\pi(k-1, t, u) \times q(v|t, u, w_{[1:n]}, k)\right)$$

- Set $(t_{n-1}, t_n) = \arg\max_{(u,v)} \pi(n, u, v)$
- For $k = (n-2) \dots 1$, $t_k = bp(k+2, t_{k+1}, t_{k+2})$
- **Return** the tag sequence $t_1 \dots t_n$

Beam Search - Heuristic

At each time step (k):

- calculate scores for all possible states (π)
- sort them according to their score values
- pass only the top B (beam width) possible states to the next time step.

Beam Search - Example (B=2)

$$\pi(k,u,v) \;=\; \max_{t \in \mathcal{S}_{k-2}} \left(\pi(k-1,t,u) \times q(v|t,u,w_{[1:n]},k)\right)$$

(t,u) could only be in {(VB,MD), (MD,VB)}

π(k-1)	<BOS>	NNP	MD	VB	JJ
<BOS>	0	0	0	0	0
NNP	0	0.1	0.1	0	0
MD	0	0	0	**0.2**	0.05
VB	0	0	**0.2**	0	0
JJ	0	0.15	0.5	0	0.15

π(k)	<BOS>	NNP	MD	VB	JJ
<BOS>	0	0	0	0	0
NNP	0	0			
MD	0		??		
VB	0				
JJ	0				

Beam Search - Example (B=2)

$$\pi(k,u,v) \quad = \quad \max_{t \in \mathcal{S}_{k-2}} \left(\pi(k-1,t,u) \times q(v|t,u,w_{[1:n]},k)\right)$$

(t,u) could only be in {(VB,MD), (MD,VB)}
For u not in {MD,VB}, π is automatically set to 0.

π(k-1)	<BOS>	NNP	MD	VB	JJ
<BOS>	0	0	0	0	0
NNP	0	0.1	0.1	0	0
MD	0	0	0	**0.2**	0.05
VB	0	0	**0.2**	0	0
JJ	0	0.15	0.5	0	0.15

π(k)	<BOS>	NNP	MD	VB	JJ
<BOS>	0	0	0	0	0
NNP	0	0	0	0	0
MD	0		??		
VB	0				
JJ	0	0	0	0	0

Beam Search - Example (B=2)

$$\pi(k,u,v) \quad = \quad \max_{t \in \mathcal{S}_{k-2}} \left(\pi(k-1,t,u) \times q(v|t,u,w_{[1:n]},k)\right)$$

(t,u) could only be in {(VB,MD), (MD,VB)}
For u not in {MD,VB}, π is automatically set to 0.
Instead of iterating through all t's, only iterate {MD, VB}.

π(k-1)	<BOS>	NNP	MD	VB	JJ
<BOS>	0	0	0	0	0
NNP	0	0.1	0.1	0	0
MD	0	0	0	**0.2**	0.05
VB	0	0	**0.2**	0	0
JJ	0	0.15	0.5	0	0.15

π(k)	<BOS>	NNP	MD	VB	JJ
<BOS>	0	0	0	0	0
NNP	0	0	0	0	0
MD	0		??		
VB	0				
JJ	0	0	0	0	0

Evaluating POS

- Accuracy - #correct_pred / #num_of_words
- Confusion matrix
- F1

Confusion Matrix

Ground Truth: The_DT boy_NNP walked_VBD

Our Tagger Predicted: The_DT boy_DT walked_NNP

The Confusion matrix:

Truth \ Predicted	*DT*	*NNP*	*VBD*
DT	1	0	0
NNP	1	0	0
VBD	0	1	0

Accuracy:

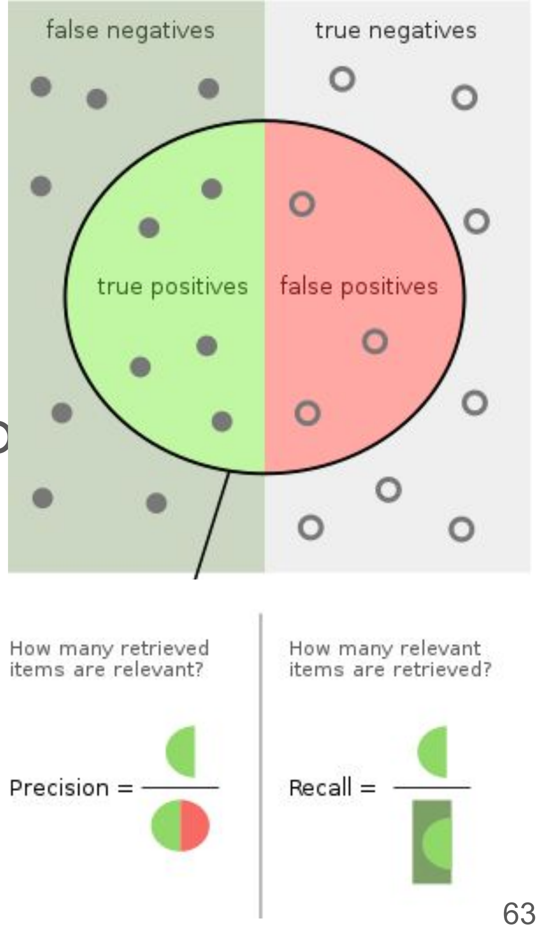
$$\frac{\#\,correct}{\#\,total} = \frac{sum(diagonal)}{sum(matrix)} = \frac{\sum_{i=1}^{|S|} a_{i,i}}{\sum_{i=1}^{|S|}\sum_{j=1}^{|S|} a_{i,j}} = \frac{1}{3}$$

F1

$$2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- For each label, we measure its F1 score
    - Based on precision & recall
    - Averaged across all words
    - We average across all words (per label)

- Then, we weight across labels to get a single sco



Next week

- Introduction to deep learning (the basic stuff)
- We need this for better text representation

- See you next week!