Part of Speech Tagging and HMMs

Yoav Goldberg

Bar Ilan University

The tagging problem

Input

Holly came from Miami , F.L.A ,
hitch-hiked her way across the USA

Output

Holly/NNP came/VBD from/IN Miami/NNP ,/, F.L.A/NNP ,/,
hitch-hiked/VBD her/PRP way/NN across/IN the/DT USA/NNP

Assign a **tag** from a given **tagset** to each **word** in a sentence.

Our 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 former/JJ 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 That/DT could/MD cost/VB him/PRP the/DT chance/NN to/TO influence/VB the/DT outcome/NN and/CC perhaps/RB join/VB the/DT winning/VBG bidder/NN ./.

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

Information Sources

With/IN such/PDT a/DT lopsided/JJ book/NN of/IN options/NNS ,/, traders/NNS say/VBP ,/, Chemical/NNP was/VBD more/RBR vulnerable/JJ to/IN erroneous/JJ valuation/NN assumptions/NNS ./.

Information Sources

With/IN such/PDT a/DT lopsided/JJ book/NN of/IN options/NNS ,/, traders/NNS say/VBP ,/, Chemical/NNP was/VBD more/RBR vulnerable/JJ to/IN erroneous/JJ valuation/NN assumptions/NNS ./.

- **Local**:
  - the word “book” is likely to be a noun.
  - the word “lopsided” is likely to be an adjective.

Information Sources

With/IN such/PDT a/DT lopsided/JJ book/NN of/IN options/NNS ,/, traders/NNS say/VBP ,/, Chemical/NNP was/VBD more/RBR vulnerable/JJ to/IN erroneous/JJ valuation/NN assumptions/NNS ./.

- **Local**:
  - the word “book” is likely to be a noun.
  - the word “lopsided” is likely to be an adjective.
- **Contextual**:
  - Noun are likely to follow adjectives or determiners.
  - Verbs are not likely to follow determiners.

Information Sources

- “I asked him to book a flight”
- “The trash can take care of itself”
- “The trash can is in the garage.”
- “Fruit flies like a banana.”

The supervised tagging problem

Formally

- We have training examples $x^{(i)}, y^{(i)}$ for $i = 1, \ldots, m$.
  - each $x^{(i)}$ is an input $x_1, \ldots, x_n$ (a crazy dog barked)
  - each $y^{(i)}$ is an output $y_1, \ldots, y_n$ (DT JJ NN VBD)

The supervised tagging problem

Formally

- We have training examples $x^{(i)}, y^{(i)}$ for $i = 1, \ldots, m$.
  - each $x^{(i)}$ is an input $x_1, \ldots, x_n$ (a crazy dog barked)
  - each $y^{(i)}$ is an output $y_1, \ldots, y_n$ (DT JJ NN VBD)
- Task: learn a function f mapping inputs x to labels $f(x) = y$

The supervised tagging problem

Conditional Model

- Learn a distribution $p(y|x)$ from training examples.
- Define $f(x) = argmax_y p(y|x)$

The supervised tagging problem

Conditional Model

- Learn a distribution $p(y|x)$ from training examples.
- Define $f(x) = argmax_y p(y|x)$
- How do we compute $p(y|x)$?

How do we define $p(y|x)$?

- If we could compute $p(x, y)$, then $p(y|x) = \frac{p(x,y)}{p(x)}$
- ... and $p(x)$ is constant.
- ... so $\arg\max_y p(y|x) = \arg\max_y p(x, y)$

$\Rightarrow$ Lets try to learn $p(x, y)$ instead.

How do we define $p(y|x)$?

- If we could compute $p(x, y)$, then $p(y|x) = \frac{p(x,y)}{p(x)}$
- ...and $p(x)$ is constant.
- ...so $\arg\max_y p(y|x) = \arg\max_y p(x, y)$

$\Rightarrow$ Lets try to learn $p(x, y)$ instead.

p(x,y)?

- Why not work with $p(y|x)$ directly?

How do we define $p(y|x)$?

- If we could compute $p(x,y)$, then $p(y|x) = \frac{p(x,y)}{p(x)}$
- …and $p(x)$ is constant.
- …so $\arg\max_y p(y|x) = \arg\max_y p(x,y)$

$\Rightarrow$ Lets try to learn $p(x,y)$ instead.

p(x,y)?

- Why not work with $p(y|x)$ directly?
  - We are working with probabilities.
  - We'll see shortly that we can compute $p(x,y)$ using basic probability rules.
  - It is not so easy for $p(y|x)$.

How do we define $p(y|x)$?

- If we could compute $p(x, y)$, then $p(y|x) = \frac{p(x,y)}{p(x)}$
- ... and $p(x)$ is constant.
- ... so $\arg\max_y p(y|x) = \arg\max_y p(x, y)$

$\Rightarrow$ Lets try to learn $p(x, y)$ instead.

p(x,y)?

- Why not work with $p(y|x)$ directly?
  - We are working with probabilities.
  - We'll see shortly that we can compute $p(x, y)$ using basic probability rules.
  - It is not so easy for $p(y|x)$.
- What do we gain/loose from working with $p(x, y)$?

Question 1: score computation

Assume someone gave us a x, y pair.
How do we compute $p(x, y)$?

$P($ Holly/NNP came/VBD from/IN Miami/NNP ,/, F.L.A/NNP ,/, hitch-hiked/VBD her/PRP way/NN across/IN the/DT USA/NNP $)$?

$P($ Holly/NNP came/VBZ from/IN Miami/JJ ,/, F.L.A/NNP ,/, hitch-hiked/IN her/PRP way/VBZ across/IN the/CD USA/NNP $)$?

$P($ Holly/NN came/NN from/NN Miami/NN ,/NN F.L.A/NN ,/NN hitch-hiked/NN her/NN way/NN across/NN the/NN USA/NN $)$?

$P($ Holly/NNP came/VBZ from/IN Miami/NNP ,/, F.L.A/NNP ,/, hitch-hiked/VBD her/PRP way/JJ across/IN the/DT USA/NNP $)$?

p(x,y)

Generative model

- Working with the *joint probability* $p(x, y)$ suggests the use of a *generative model*.
- Define a *generative story* of how the data was created.
- The story doesn't have to be true. It has to be reasonable.
  - Reasonable?? In terms of the independence assumptions.

p(x,y)

Our generative story

How does a sentence come to life?

- First, a sequence of tags is created.

p(x,y)

Our generative story

How does a sentence come to life?

- First, a sequence of tags is created.
- Then, each tag is replaced with a word.

p(x,y)

Our generative story

How does a sentence come to life?

- First, a sequence of tags is created.
- Then, each tag is replaced with a word.
  - All we see are the words. We need to guess the tags.
  - Noisy channel interpretation: our pure message was y. But something changed our message to x instead.

p(x,y)

Our generative story

How does a sentence come to life?

- First, a sequence of tags is created.
- Then, each tag is replaced with a word.
  - All we see are the words. We need to guess the tags.
  - Noisy channel interpretation: our pure message was y. But something changed our message to x instead.
- Rewrite $p(x, y) = p(y)p(x|y)$

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

- No assumptions so far.
- But breaking into $p(y)$ and $p(x|y)$ makes our life easier.
  - Why?
  - (and why not break things into $p(x)$ and $p(y|x)$?)

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

p(y)

- First attempt – Maximum Likelihood Estimation (MLE)

$$p(y) = p(y_1, y_2, \ldots, y_n) = \frac{count(y_1, y_2, \ldots, y_n)}{\text{num of training examples}}$$

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

p(y)

- First attempt – Maximum Likelihood Estimation (MLE)

$$p(y) = p(y_1, y_2, \ldots, y_n) = \frac{count(y_1, y_2, \ldots, y_n)}{\text{num of training examples}}$$

Problem?

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

p(y)

- Second attempt – use chain rule

$$
\begin{aligned}
p(y) = p(y_1, y_2, \ldots, y_n) = & p(y_1) \\
& \times p(y_2|y_1) \\
& \times p(y_3|y_1, y_2) \\
& \times p(y_4|y_1, y_2, y_3) \\
& \ldots \\
& \times p(y_n|y_1, y_2, y_3, \ldots, y_{n-1})
\end{aligned}
$$

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

p(y)

- Second attempt – use chain rule

$$
\begin{aligned}
p(y) = p(y_1, y_2, \ldots, y_n) = & p(y_1) \\
& \times p(y_2|y_1) \\
& \times p(y_3|y_1, y_2) \\
& \times p(y_4|y_1, y_2, y_3) \\
& \ldots \\
& \times p(y_n|y_1, y_2, y_3, \ldots, y_{n-1})
\end{aligned}
$$

- Is this any better?

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

p(y) – Markov assumption

- Does the tag of the first word really influences the tag of the seventh word?

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

p(y) – Markov assumption

- Does the tag of the first word really influences the tag of the seventh word?
- And the does it influence the tag of the 4th word?

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

p(y) – Markov assumption

- Does the tag of the first word really influences the tag of the seventh word?
- And the does it influence the tag of the 4th word?
- Let assume only the previous tag matters:

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

p(y) – Markov assumption

- Does the tag of the first word really influences the tag of the seventh word?
- And the does it influence the tag of the 4th word?
- Let assume only the previous tag matters:

$$p(y_i|y_1, y_2, \ldots, y_{i-2}, y_{i-1}) \approx q(y_i|y_{i-1})$$

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

p(y)

- chain rule + markov assumption

$$p(y_i|y_1, y_2, \ldots, y_{i-2}, y_{i-1}) \approx q(y_i|y_{i-1})$$

$$\begin{aligned} p(y) = p(y_1, y_2, \ldots, y_n) = & q(y_1|\texttt{start}) \\ & \times q(y_2|y_1) \\ & \times q(y_3|y_2) \\ & \times q(y_4|y_3) \\ & \ldots \\ & \times q(y_n|y_{n-1}) \end{aligned}$$

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

p(y) – 2nd-order Markov assumption

- Let assume only the **two** previous tag matter:

$$p(y_i|y_1, y_2, \ldots, y_{i-2}, y_{i-1}) \approx q(y_i|y_{i-2}, y_{i-1})$$

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

p(y)

- chain rule + 2nd-order markov assumption

$$p(y_i|y_1, y_2, \ldots, y_{i-2}, y_{i-1}) \approx q(y_i|y_{i-1}, y_{i-2})$$

$$\begin{aligned} p(y) = p(y_1, y_2, \ldots, y_n) = & q(y_1|\texttt{start}, \texttt{start}) \\ & \times q(y_2|\texttt{start}, y_1) \\ & \times q(y_3|y_1, y_2) \\ & \times q(y_4|y_2, y_3) \\ & \ldots \\ & \times q(y_n|y_{n-2}, y_{n-1}) \end{aligned}$$

Estimating $q(y_i|y_{i-2}, y_{i-1})$

- Here it is quite safe to use MLE estimates (why?)

$$q(c|a,b) = \frac{count(a,b,c)}{count(a,b)}$$

Estimating $q(y_i|y_{i-2}, y_{i-1})$

- Here it is quite safe to use MLE estimates (why?)

$$q(c|a, b) = \frac{count(a, b, c)}{count(a, b)}$$

- We could still get zero probabilities.
  - is this a bad thing?

Estimating $q(y_i|y_{i-2}, y_{i-1})$

- Here it is quite safe to use MLE estimates (why?)

$$q(c|a,b) = \frac{count(a,b,c)}{count(a,b)}$$

- We could still get zero probabilities.
  - is this a bad thing?
- To be on the safe side, we could use interpolation:

$$q(c|a,b) = \lambda_1 \frac{count(a,b,c)}{count(a,b)} + \lambda_2 \frac{count(b,c)}{count(b)} + \lambda_3 \frac{count(c)}{\text{num words}}$$

Estimating $q(y_i|y_{i-2}, y_{i-1})$

- Here it is quite safe to use MLE estimates (why?)

$$q(c|a,b) = \frac{count(a,b,c)}{count(a,b)}$$

- We could still get zero probabilities.
  - is this a bad thing?
- To be on the safe side, we could use interpolation:

$$q(c|a,b) = \lambda_1 \frac{count(a,b,c)}{count(a,b)} + \lambda_2 \frac{count(b,c)}{count(b)} + \lambda_3 \frac{count(c)}{\text{num words}}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \quad \lambda_i > 0$$

Estimating $q(y_i|y_{i-2}, y_{i-1})$

- Here it is quite safe to use MLE estimates (why?)

$$q(c|a,b) = \frac{count(a,b,c)}{count(a,b)}$$

- We could still get zero probabilities.
  - is this a bad thing?
- To be on the safe side, we could use interpolation:

$$q(c|a,b) = \lambda_1 \frac{count(a,b,c)}{count(a,b)} + \lambda_2 \frac{count(b,c)}{count(b)} + \lambda_3 \frac{count(c)}{\text{num words}}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \quad \lambda_i > 0$$

- How would you set the λ values?

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

We can compute p(y)

$$
\begin{aligned}
p(y) = p(y_1, y_2, \ldots, y_n) = & q(y_1|\texttt{start}, \texttt{start}) \\
& \times q(y_2|\texttt{start}, y_1) \\
& \times q(y_3|y_1, y_2) \\
& \times q(y_4|y_2, y_3) \\
& \ldots \\
& \times q(y_n|y_{n-2}, y_{n-1})
\end{aligned}
$$

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

We can compute p(y)

$$
\begin{aligned}
p(y) = p(y_1, y_2, \ldots, y_n) = & q(y_1|\texttt{start}, \texttt{start}) \\
& \times q(y_2|\texttt{start}, y_1) \\
& \times q(y_3|y_1, y_2) \\
& \times q(y_4|y_2, y_3) \\
& \ldots \\
& \times q(y_n|y_{n-2}, y_{n-1}) \\
= & \prod_{i=1}^{n} q(y_i|y_{i-2}, y_{i-i})
\end{aligned}
$$

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

We can compute p(y)

$$
\begin{aligned}
p(y) = p(y_1, y_2, \ldots, y_n) = & q(y_1|\texttt{start}, \texttt{start}) \\
& \times q(y_2|\texttt{start}, y_1) \\
& \times q(y_3|y_1, y_2) \\
& \times q(y_4|y_2, y_3) \\
& \ldots \\
& \times q(y_n|y_{n-2}, y_{n-1}) \\
= & \prod_{i=1}^{n} q(y_i|y_{i-2}, y_{i-i})
\end{aligned}
$$

Moving on to $p(x|y)$

$p(x|y)$

$$p(x|y) = p(x_1, x_2, \ldots, x_n | y_1, y_2, \ldots, y_n) =$$

$p(x|y)$

$$p(x|y) = p(x_1, x_2, \ldots, x_n | y_1, y_2, \ldots, y_n) = \\ p(x_1 | y_1, \ldots, y_n)$$

$p(x|y)$

$$
\begin{aligned}
p(x|y) = & p(x_1, x_2, \ldots, x_n | y_1, y_2, \ldots, y_n) = \\
& p(x_1 | y_1, \ldots, y_n) \\
& \times p(x_2 | x_1, y_1, \ldots, y_n)
\end{aligned}
$$

$p(x|y)$

$$\begin{aligned}
p(x|y) = & p(x_1, x_2, \ldots, x_n | y_1, y_2, \ldots, y_n) = \\
& p(x_1 | y_1, \ldots, y_n) \\
& \times p(x_2 | x_1, y_1, \ldots, y_n) \\
& \times p(x_3 | x_1, x_2, y_1, \ldots, y_n) \\
& \times p(x_4 | x_1, x_2, x_3, y_1, \ldots, y_n) \\
& \ldots \\
& \times p(x_n | x_1, x_2, \ldots, x_n, y_1, \ldots, y_n)
\end{aligned}$$

$p(x|y)$

$$
\begin{aligned}
p(x|y) = & p(x_1, x_2, \ldots, x_n | y_1, y_2, \ldots, y_n) = \\
& p(x_1 | y_1, \ldots, y_n) \\
& \times p(x_2 | x_1, y_1, \ldots, y_n) \\
& \times p(x_3 | x_1, x_2, y_1, \ldots, y_n) \\
& \times p(x_4 | x_1, x_2, x_3, y_1, \ldots, y_n) \\
& \ldots \\
& \times p(x_n | x_1, x_2, \ldots, x_n, y_1, \ldots, y_n)
\end{aligned}
$$

- What's a reasonable assumption to make here?

$p(x|y)$

p(x|y) – independence assumption

- We'll assume that a word depends only on its tag.

$$p(x_i|x_1,\ldots,x_{i-1},y_1,\ldots,y_n) \approx e(x_i|y_i)$$

$p(x|y)$

p(x|y) – independence assumption

- We'll assume that a word depends only on its tag.

$$p(x_i|x_1,\ldots,x_{i-1},y_1,\ldots,y_n) \approx e(x_i|y_i)$$

- A terrible assumption if we were generating sentences!

$p(x|y)$

p(x|y) – independence assumption

- We'll assume that a word depends only on its tag.

$$p(x_i|x_1, \ldots, x_{i-1}, y_1, \ldots, y_n) \approx e(x_i|y_i)$$

- A terrible assumption if we were generating sentences!
  - . . . but we don't use this model to generate sentences.
  - The sentence is given. We are looking for a tag sequence.

Estimating $e(x_i|y_i)$

- MLE again:

$$e(book|NN) = \frac{count(book, NN)}{count(NN)}$$

Estimating $e(x_i|y_i)$

- MLE again:

$$e(book|NN) = \frac{count(book, NN)}{count(NN)}$$

- Do you see any problem here?

Estimating $e(x_i|y_i)$

- MLE again:

$$e(book|NN) = \frac{count(book, NN)}{count(NN)}$$

- Do you see any problem here?
  - (we'll get to this later)

$p(x|y)$

$$
\begin{aligned}
p(x|y) = & p(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n) = \\
& e(x_1|y_1) \\
& \times e(x_2|y_2) \\
& \times e(x_3|y_3) \\
& \times e(x_4|y_4) \\
& \ldots \\
& \times e(x_n|y_n) \\
= & \prod_{i=1}^{n} e(x_i|y_i)
\end{aligned}
$$

A Bigram Tagging Model (first order HMM)

$$p(x,y) = p(y)p(x|y) = \prod_{i=1}^{n} q(y_i|y_{i-1}) \prod_{i=1}^{n} e(x_i|y_i)$$

$q(y_i|y_{i-1})$: transition probabilities

$e(x_i|y_i)$: emission probabilities

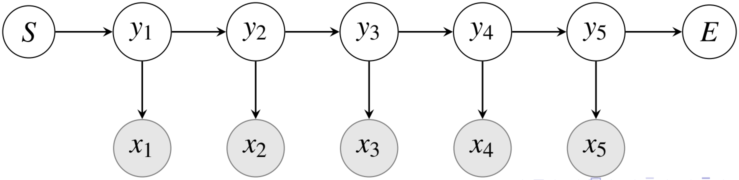
$p(x, y) = p(y)p(x|y)$

A Bigram Tagging Model (first order HMM)

$$p(x, y) = p(y)p(x|y) = \prod_{i=1}^{n} q(y_i|y_{i-1}) \prod_{i=1}^{n} e(x_i|y_i)$$

$q(y_i|y_{i-1})$: transition probabilities

$e(x_i|y_i)$: emission probabilities



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

A Trigram Tagging Model (second order HMM)

$$p(x, y) = p(y)p(x|y) = \prod_{i=1}^{n} q(y_i|y_{i-2}, y_{i-1}) \prod_{i=1}^{n} e(x_i|y_i)$$

$q(y_i|y_{i-2}, y_{i-1})$: transition probabilities

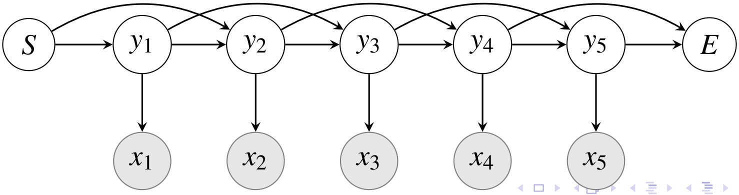
$e(x_i|y_i)$: emission probabilities

A Trigram Tagging Model (second order HMM)

$$p(x, y) = p(y)p(x|y) = \prod_{i=1}^{n} q(y_i|y_{i-2}, y_{i-1}) \prod_{i=1}^{n} e(x_i|y_i)$$

$q(y_i|y_{i-2}, y_{i-1})$: transition probabilities

$e(x_i|y_i)$: emission probabilities



Second-order HMM Example

$p($ Holly/NNP came/VBD from/IN Miami/NNP ,/, F.L.A/NNP $)$

$$= \prod_{i=1}^{n} q(y_i|y_{i-2}, y_{i-1}) \prod_{i=1}^{n} e(x_i|y_i) =$$

$$q(NNP|start, start) \times q(VBD|start, NNP) \times q(IN|NNP, VBD)$$

$$\times\, q(NNP|VBD, IN) \times q(,|IN, NNP) \times q(NNP|NNP, ,)$$

$$\times\, e(Holly|NNP) \times e(came|VBD) \times e(from|IN)$$

$$\times\, e(Miami|NNP) \times e(,|,) \times e(F.L.A|NNP)$$

Second-order HMM Example

$p($ Holly/NNP came/VBD from/IN Miami/NNP ,/, F.L.A/NNP $)$

$$= \prod_{i=1}^{n} q(y_i|y_{i-2}, y_{i-1}) \qquad \prod_{i=1}^{n} e(x_i|y_i) =$$

$$\begin{array}{ll}
q(NNP|start, start) & \times e(Holly|NNP) \\
\times\, q(VBD|start, NNP) & \times e(came|VBD) \\
\times\, q(IN|NNP, VBD) & \times e(from|IN) \\
\times\, q(NNP|VBD, IN) & \times e(Miam|NNP) \\
\times\, q(,|IN, NNP) & \times e(,|,) \\
\times\, q(NNP|NNP, ,) & \times e(F.L.A|NNP)
\end{array}$$

Second-order HMM Example

$p($ Holly/NNP came/VBD from/IN Miami/NNP ,/, F.L.A/NNP $)$

$$
\begin{aligned}
= \prod_{i=1}^{n} q(y_i|y_{i-2}, y_{i-1}) \qquad & \prod_{i=1}^{n} e(x_i|y_i) = \\
q(NNP|start, start) \qquad & \times e(Holly|NNP) \\
\times\, q(VBD|start, NNP) \qquad & \times e(came|VBD) \\
\times\, q(IN|NNP, VBD) \qquad & \times e(from|IN) \\
\times\, q(NNP|VBD, IN) \qquad & \times e(Miam|NNP) \\
\times\, q(,|IN, NNP) \qquad & \times e(,|,) \\
\times\, q(NNP|NNP, ,) \qquad & \times e(F.L.A|NNP)
\end{aligned}
$$

Problem

- We are multiplying many small numbers
- End-result will by tiny

Solution: $\prod \rightarrow \sum$

$$argmax_y p(x,y) = argmax_y \log p(x,y)$$

Solution: $\prod \rightarrow \sum$

$$argmax_y p(x, y) = argmax_y \log p(x, y)$$

$$argmax_y \prod_{i=1}^{n} q(y_i|y_{i-2}, y_{i-1}) \times \prod_{i=1}^{n} e(x_i|y_i)$$
$$= argmax_y \log(\prod_{i=1}^{n} q(y_i|y_{i-2}, y_{i-1}) \times \prod_{i=1}^{n} e(x_i|y_i))$$

Solution: $\prod \rightarrow \sum$

$$argmax_y p(x,y) = argmax_y \log p(x,y)$$

$$
\begin{aligned}
& argmax_y \prod_{i=1}^{n} q(y_i|y_{i-2}, y_{i-1}) \times \prod_{i=1}^{n} e(x_i|y_i) \\
&= argmax_y \log(\prod_{i=1}^{n} q(y_i|y_{i-2}, y_{i-1}) \times \prod_{i=1}^{n} e(x_i|y_i)) \\
&= argmax_y \sum_{i=1}^{n} \log q(y_i|y_{i-2}, y_{i-1}) + \sum_{i=1}^{n} \log e(x_i|y_i)
\end{aligned}
$$

Second Order HMM – log space

$\log p($ Holly/NNP came/VBD from/IN Miami/NNP ,/, F.L.A/NNP $)$

$$
\begin{aligned}
&= \sum_{i=1}^{n} \log q(y_i|y_{i-2}, y_{i-1}) && + \sum_{i=1}^{n} \log e(x_i|y_i) = \\
&\log q(NNP|start, start) && + \log e(Holly|NNP) \\
&+ \log q(VBD|start, NNP) && + \log e(came|VBD) \\
&+ \log q(IN|NNP, VBD) && + \log e(from|IN) \\
&+ \log q(NNP|VBD, IN) && + \log e(Miam|NNP) \\
&+ \log q(,|IN, NNP) && + \log e(,|,) \\
&+ \log q(NNP|NNP,) && + \log e(F.L.A|NNP)
\end{aligned}
$$

Decoding

Decoding

argmax$_y$?

Remember, we want to tag sentences.

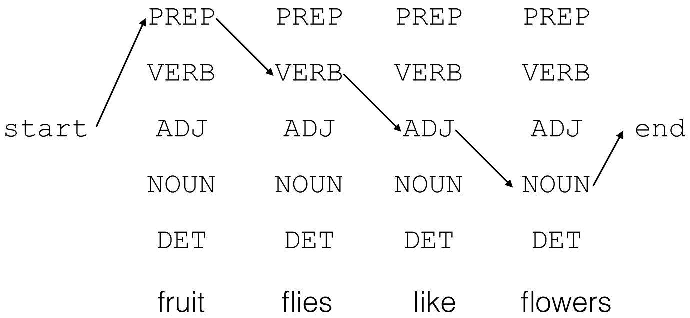
- We can compute $p(x, y)$
- We are given words $x = x_1, \ldots, x_n$
- We are looking for a sequence $y = y_1, \ldots, y_n$ s.t. $p(x, y)$ is maximized.

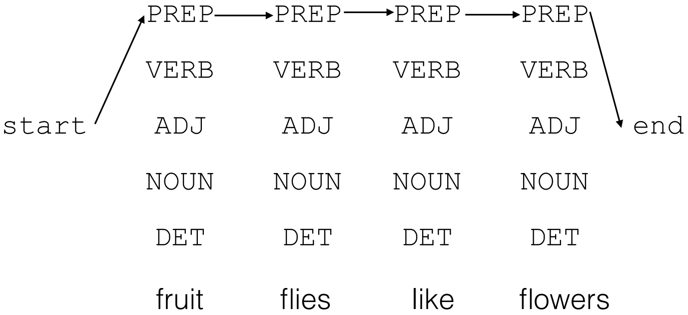
How do we search for y?

$\mathrm{argmax}_y p(x, y)$

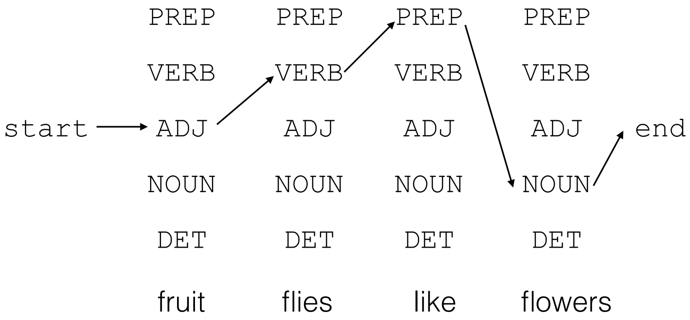
Solution 1

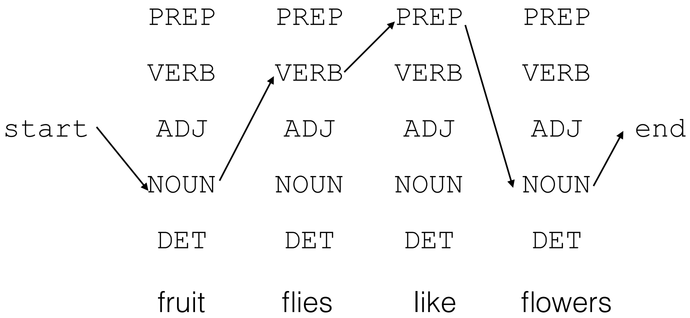
- Go over all possible sequences y.

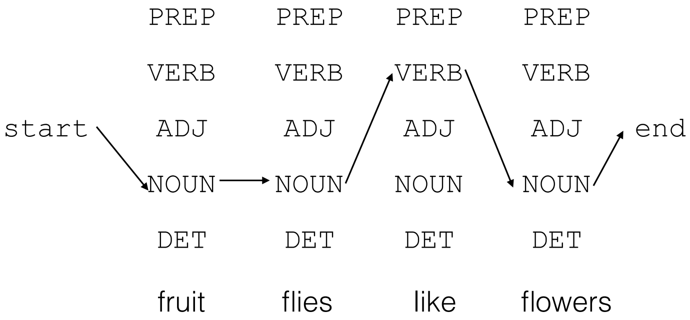

PREP PREP PREP PREP
VERB VERB VERB VERB
start ADJ ADJ ADJ ADJ end
NOUN NOUN NOUN NOUN
DET DET DET DET
fruit flies like flowers



PREP PREP PREP PREP
VERB VERB VERB VERB
start ADJ ADJ ADJ ADJ end
NOUN NOUN NOUN NOUN
DET DET DET DET
fruit flies like flowers


PREP PREP PREP PREP
VERB VERB VERB VERB
start ADJ ADJ ADJ ADJ end
NOUN NOUN NOUN NOUN
DET DET DET DET
fruit flies like flowers


PREP PREP PREP PREP
VERB VERB VERB VERB
start ADJ ADJ ADJ ADJ end
NOUN NOUN NOUN NOUN
DET DET DET DET
fruit flies like flowers


start
PREP VERB ADJ NOUN DET
PREP VERB ADJ NOUN DET
PREP VERB ADJ NOUN DET
PREP VERB ADJ NOUN DET
end
fruit
flies
like
flowers

$\text{argmax}_y p(x, y)$

Solution 1

- Go over all possible sequences y.

Problem

- There are very many such sequences. (how many?)

argmax$_y p(x, y)$

Solution 2

- Choose the highest scoring tag t_1 for $e(x_1|y_1)q(y_1|start)$
- Choose the highest scoring tag t_2 for $e(x_2|y_2)q(y_2|start, t_1)$
- Choose the highest scoring tag t_3 for $e(x_3|y_3)q(y_3|t_1, t_2)$
- ...

argmax$_y p(x, y)$

Solution 2

- Choose the highest scoring tag t_1 for $e(x_1|y_1)q(y_1|start)$
- Choose the highest scoring tag t_2 for $e(x_2|y_2)q(y_2|start, t_1)$
- Choose the highest scoring tag t_3 for $e(x_3|y_3)q(y_3|t_1, t_2)$
- ...

complexity: $O(kn)$ where k is tagset size.

argmax$_y p(x, y)$

Solution 2

- Choose the highest scoring tag t_1 for $e(x_1|y_1)q(y_1|start)$
- Choose the highest scoring tag t_2 for $e(x_2|y_2)q(y_2|start, t_1)$
- Choose the highest scoring tag t_3 for $e(x_3|y_3)q(y_3|t_1, t_2)$
- ...

complexity: $O(kn)$ where k is tagset size.

Problem

- Will not produce optimal solution. (why?)

$\text{argmax}_y p(x, y)$

Solution: Dynamic Programming

- The viterbi algorithm.

Bigram Viterbi

$V(i, t)$

maximum probability of a tag sequence ending in tag t at time i.

Bigram Viterbi

$V(i, t)$

maximum probability of a tag sequence ending in tag t at time i.

Recursive Definition

$$V(0, \texttt{start}) = 1$$

Bigram Viterbi

$V(i, t)$

maximum probability of a tag sequence ending in tag t at time i.

Recursive Definition

$$V(0, \texttt{start}) = 1$$

$$V(i, t)$$

Bigram Viterbi

$V(i, t)$

maximum probability of a tag sequence ending in tag t at time i.

Recursive Definition

$$
\begin{aligned}
V(0, \texttt{start}) &= 1 \\
V(i, t) &= \max_{t'} V(i-1, t') q(t|t') e(w_i|t)
\end{aligned}
$$

Trigram Viterbi

$V(i, t, r)$

maximum probability of a tag sequence ending in tags t,r at time i.

Trigram Viterbi

$V(i, t, r)$

maximum probability of a tag sequence ending in tags t,r at time i.

Recursive Definition

$$V(0, \texttt{start}, \texttt{start}) = 1$$

$$V(i, t, r) = \max_{t'} V(i-1, t', t) q(r|t', t) e(w_i|r)$$

Trigram Viterbi – Algorithm

Input:

sentence: $w_1, \ldots, w_n$
parameters: $e(w|t)$, $q(t|u, v)$
tagset: T

Output:

probability of best tag sequence $y_1, \ldots, y_n$

Algorithm:

- For $i = 1, \ldots, n$
  - For $t \in T$, $r \in T$
    $V(i, t, r) = \max_{t'} V(i-1, t', t) q(r|t', t) e(w_i|r)$

return: $\max_{t \in T, r \in T} V(n, t, r)$

Trigram Viterbi with Back-pointers – Algorithm

Input:

sentence: $w_1, \ldots, w_n$
parameters: $e(w|t)$, $q(t|u,v)$
tagset: T

Output:

probability of best tag sequence $y_1, \ldots, y_n$

Algorithm:

- For $i = 1, \ldots, n$
  - For $t \in T$, $r \in T$
    $V(i,t,r) = \max_{t'} V(i-1,t',t)q(r|t',t)e(w_i|r)$
    $bp(i,t,r) = \arg\max_{t'} V(i-1,t',t)q(r|t',t)e(w_i|r)$
- set $y_{n-1}, y_n = \arg\max_{t,r} V(n,t,r)$
- for $i = n-2 \ \ldots \ 1$ set $y_i = bp(i+2, y_{i+1}, y_{i+2})$

return: $y_1, \ldots, y_n$

Runtime

$$O(n * |T|^3)$$

why?

Supervised Second-order HMM Tagger (trigram tagger)

Training

- Using corpus of tagged sentences, compute:
  - count(tag1,tag2,tag3), count(tag1,tag2), count(tag), count(tag,word)
  - calculate e, q based on counts

Supervised Second-order HMM Tagger (trigram tagger)

Training

- Using corpus of tagged sentences, compute:
  - count(tag1,tag2,tag3), count(tag1,tag2), count(tag), count(tag,word)
  - calculate e, q based on counts

Tagging

- When given a sentence $x = x_1, \ldots, x_n$
  - Use the viterbi algorithm to find $argmax_y p(y|x) = argmax_y p(x, y)$
  - using the e and q quantities from training.

Order considerations

- First order markov: $p(y_i|y_1,\ldots,y_{i-1}) = q(y_i|y_{i-1})$
- Second order markov: $p(y_i|y_1,\ldots,y_{i-1}) = q(y_i|y_{i-2},y_{i-1})$

Is there any reason to prefer the first- over the second-order?

Why not do third-order?

HMMs – dealing with rare or unseen words

Our training set is of limited size

- Some words will not be seen in the corpus.

HMMs – dealing with rare or unseen words

Our training set is of limited size

- Some words will not be seen in the corpus.
  - so?

HMMs – dealing with rare or unseen words

Our training set is of limited size

- Some words will not be seen in the corpus.
  - so?
- Some words will only be seen once.

HMMs – dealing with rare or unseen words

Our training set is of limited size

- Some words will not be seen in the corpus.
  - so?
- Some words will only be seen once.
  - so?

HMMs – dealing with rare or unseen words

How do we calculate

$$e(word|tag)$$

for unseen or infrequent words?

HMMs – dealing with rare or unseen words

How do we calculate

$$e(word|tag)$$

for unseen or infrequent words?

- UNK

$$e(\text{UNK}|tag)$$

HMMs – dealing with rare or unseen words

How do we calculate

$$e(word|tag)$$

for unseen or infrequent words?

- UNK

$$e(\text{UNK}|tag)$$

- "Signatures"

_ing

X_ed

HMMs – dealing with rare or unseen words

How do we calculate

$$e(word|tag)$$

for unseen or infrequent words?

- UNK

$$e(\text{UNK}|tag)$$

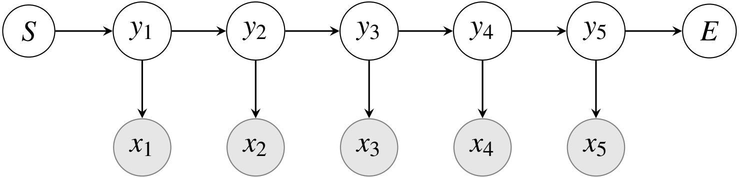
- "Signatures"

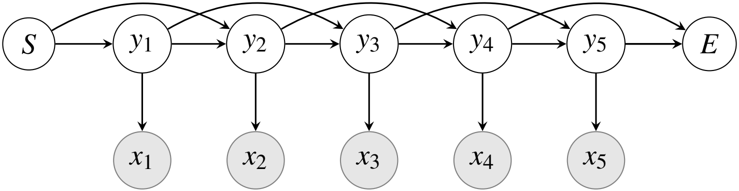
_ing

X_ed

How do we estimate these?

HMM – Sumary





HMM – Summary

The HMM tagging algorithm

- $f(x) = argmax_y p(y|x) = argmax_y p(x, y)$
- model $p(x, y) = p(y)p(x|y) = \prod q(y_i|y_{i-1}) \times e(x_i|y_i)$
- Learn tables for transitions q and emissions e by counting.
- Find best y for a given x using viterbi.
- Hardest part: good $e(word|tag)$ for rare/unseen words.

HMM – Summary

- For a long time, the best tagging algorithm available.
- Nowadays, more accurate models exist (we'll see some of them).
- HMM still useful for **unsupervised** learning.
  - You a lot of text (without labels)
  - And a dictionary mapping words to possible tags.

  $\Rightarrow$ Can learn q and e using the EM algorithm.