Language Modeling

Professor Marie Roch

for details on N-gram models, see chapter 4:
Jurafsky, D., and Martin, J. H. (2009). *Speech and Language Processing (Pearson Prentice Hall, Upper Saddle River, NJ)*

Narrowing search with a language model

- Don't move or I'll …
- Get 'er …
- What will she think of …
- This enables …

Applications

- Speech recognition
- Handwriting recognition
- Spelling correction
- Augmentative communication

and more…

Constituencies

- Groupings of words
  - I didn't see you *behind the bush*.
  - She *ate quickly* as she was *late for the meeting*.
- Movement within the sentence:
  - ✔ As she was late for the meeting, she ate quickly
  - ⊘ As she was late for, she ate quickly the meeting.
- Constituencies aid in prediction.

Strategies for construction

- Formal grammar
  - Requires intimate knowledge of the language
  - Usually context free and cannot be represented by a regular language
  - We will not be covering this in detail

N-gram models

- Suppose we wish to compute the probability the sentence:
  She sells seashells down by the seashore.
- We can think of this as a sequence of words:

$$\underbrace{\text{She}}_{w_1}\ \underbrace{\text{sells}}_{w_2}\ \underbrace{\text{seashells}}_{w_3}\ \underbrace{\text{down}}_{w_4}\ \underbrace{\text{by}}_{w_5}\ \underbrace{\text{the}}_{w_6}\ \underbrace{\text{seashore}}_{w_7}$$

$$P(w_1^7) = P(w_1, w_2, w_3, w_4, w_5, w_6, w_7)$$

Think further: How would you determine if a die is fair?

Estimating word probability

- Suppose we wish to compute the probability w_2 (*sells* in the previous example).

  We could estimate using a **relative frequency**

$$P(w_2) = \frac{\#\text{ times } w_2 \text{ occurs}}{\#\text{ of times all words occur}}$$

but this ignores what we could have learned with the first word.

Conditional probability

By defn. of conditional probability

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

or in our problem:

$$P(w_2 \mid w_1) = \frac{P(w_2 \cap w_1)}{P(w_1)} = \frac{P(w_1 \cap w_2)}{P(w_1)}$$

$$= \frac{P(w_1, w_2)}{P(w_1)} \text{ defn} \cap \text{ for words}$$

Conditional probability

Next, consider $P(w_1, w_2)$

Since as $P(w_2 \mid w_1) = \dfrac{P(w_1, w_2)}{P(w_1)}$,

clearly $P(w_1, w_2) = P(w_2 \mid w_1) P(w_1)$

Chain rule

- Now let us consider:

$$P(w_1, w_2, w_3) = P(w_3 \mid w_1, w_2) \underbrace{P(w_1, w_2)}_{\text{we just did this part}}$$
$$= P(w_3 \mid w_1, w_2) P(w_2 \mid w_1) P(w_1)$$

- By applying conditional probability repeatedly we end up with the chain rule:

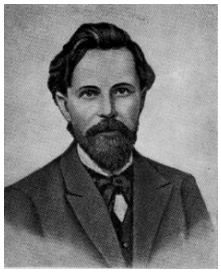
$$\begin{aligned} \mathrm{P}(W) &= \mathrm{P}(w_1 w_2 \ldots w_n) \\ &= \mathrm{P}(w_1)\,\mathrm{P}(w_2 \mid w_1)\,\mathrm{P}(w_3 \mid w_1 w_2) \ldots \mathrm{P}(w_n \mid w_1 w_2 \ldots w_{n-1}) \\ &= \prod_{i=1}^{n} \mathrm{P}(w_i \mid w_1 w_2 \ldots w_{i-1}) \end{aligned}$$

Sparse problem space

- Suppose V distinct words.
- w_1^i has V^i possible sequences of words.
- Tokens N – The number of N-grams (including repetitions) occurring in a corpus
- Problem: In general, unique(N)<<valid tokens for the language.

  "The gently rolling hills were covered with bluebonnets" had no hits on Google at the time this slide was published.

Markov assumption



Andrei Markov
1856-1922

- A prediction is dependent on the current state but independent of previous conditions
- In our context:

$$P(w_n \mid w_1^{n-1}) = P(w_n \mid w_{n-1}) \text{ by the Markov assumption}$$

which at times relax to N-1 words:

$$P(w_n \mid w_1^{n-1}) = P(w_n \mid w_{n-N+1}^{n-1})$$

Special N-grams

- Unigram
  - Only depends upon the word itself.
  - $P(w_i)$
- Bigram
  - $P(w_i|w_{i-1})$
- Trigram
  - $P(w_i|w_{i-1}, w_{i-2})$
- Quadrigram
  - $P(w_i|w_{i-1}, w_{i-2}, w_{i-3})$

Preparing a corpus

- Make case independent
- Remove punctuation and add start & end of sentence markers <s> </s>
- Other possibilities
  - part of speech tagging
  - lemmas: mapping of words with similar roots
    e.g. sing, sang, sung → sing
  - stemming: mapping of derived words to their root
    e.g. parted → part, ostriches → ostrich

An Example

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

Dr. Seuss, *Green Eggs and Ham*, 1960.

$$P(\texttt{I}\,|\,\texttt{<s>}) = \tfrac{2}{3} = .67 \qquad P(\texttt{Sam}\,|\,\texttt{<s>}) = \tfrac{1}{3} = .33 \qquad P(\texttt{am}\,|\,\texttt{I}) = \tfrac{2}{3} = .67$$
$$P(\texttt{</s>}\,|\,\texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam}\,|\,\texttt{am}) = \tfrac{1}{2} = .5 \qquad P(\texttt{do}\,|\,\texttt{I}) = \tfrac{1}{3} = .33$$

$$P\left(w_n \mid w_{n-N+1}^{n-1}\right) = \frac{C\left(w_{n-N+1}^{n-1} w_n\right)}{C\left(w_{w-N+1}^{n-1}\right)}$$

Berkeley Restaurant Project Sentences

- *can you tell me about any good cantonese restaurants close by*
- *mid priced thai food is what i'm looking for*
- *tell me about chez panisse*
- *can you give me a listing of the kinds of food that are available*
- *i'm looking for a good place to eat breakfast*
- *when is caffe venezia open during the day*

Bigram Counts from 9,222 sentences

"i want"

w_i

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

w_{i-1}

Bigram Probabilities

Unigram counts

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

$$P(\text{i want}) = \frac{C(\text{i want})}{C(\text{i})} = \frac{827}{2533} \approx 0.33$$

w_i

w_{i-1}	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Bigram Estimates of Sentence Probabilities

P(<s> I want english food </s>)

=P(I|<s>)P(want|I)P(english|want)P(food|english)P(</s>|food)

=.000 031

Shakespeare:

N=884,647 tokens, V=29,066

Unigram	• To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have • Every enter now severally so, let • Hill he late speaks; or! a more to leg less first you enter • Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like
Bigram	• What means, sir. I confess she? then all sorts, he is trim, captain. •Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. •What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman? •Enter Menenius, if it so many good direction found'st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt
Trigram	• Sweet prince, Falstaff shall die. Harry of Monmouth's grave. • This shall forbid it should be branded, if renown made it empty. • Indeed the duke; and had a very good friend. • Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.
Quadrigram	• King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; • Will you not tell me who I am? • It cannot be but so. • Indeed the short and the long. Marry, 'tis a noble Lepidus.

How will this work on *Huckleberry Finn*?

The need for n-gram smoothing

- Data for estimation is sparse.
- On a sample text with several million words
  - 50% of trigrams only occurred once
  - 80% of trigrams occurred less than 5 times
- Example: When pigs fly

$$P(fly \mid when, pigs) = \frac{C(when, pigs, fly)}{C(when, pigs)}$$

$$= \frac{0}{C(when, pigs)} \quad \text{if "when pigs fly" unseen}$$

Smoothing strategies

- Suppose P(fly | when, pigs) = 0
- Backoff strategies do the following
  - When estimating P(Z | X, Y) where C(XYZ)>0,
  - don't assign all of the probability, save some of it for the cases we haven't seen. This is called discounting and is based on Good-Turing counts

Smoothing strategies

- For things that have C(X, Y, Z) = 0, use P(Z|Y), but scale it by the amount of leftover probability
- To handle C(Y, Z) = 0, this process can be computed recursively.

Neural language models

- Advantages
  - As the net learns a representation, similarities can be captured
    Example: Consider food
    - Possible to learn common things about foods
    - Yet the individual items can still be considered distinct

    There are approaches to capture commonality in N-gram models (e.g. Knesser-Ney), but they lose the ability to distinguish the words

Perplexity

- Measure of ability of language model to predict next word
- Related to cross entropy of language, *H(L),* perplexity is $2^{H(L)}$

$$H(L) = \lim_{n\to\infty} \frac{1}{n} H(w_1, w_2, \ldots, w_n)$$
$$= -\lim_{n\to\infty} \frac{1}{n} \sum_{W\in L} P(w_1, w_2, \ldots, w_n) \log\left(P(w_1, w_2, \ldots, w_n)\right)$$

- Lower perplexity indicates better modeling (theoretically)

Neural language models

  - Word embeddings can learn low dimensional representations of words that can capture semantic information
- Disadvantages
  - Traditional prediction uses one-hot vectors over vocabulary.
    - High dimensional output space
    - Computationally expensive

Consider a softmax output layer

- Suppose
  - V words in vocabulary $\mathbb{V}$
  - n_h units in last hidden layer
- ∴ softmax input to & output/unit:

$$a_i = b_i + \sum_j W_{i,j} h_j \quad \text{where } 1 \le i \le V \qquad \hat{y}_i = \frac{e^{a_i}}{\sum_{k=1}^{V} e^{a_k}}$$

- total cost of softmax layer $O(Vn_h)$:

$$V \approx k_1 \times 10^6, n_h \approx k_2 \times 10^3$$

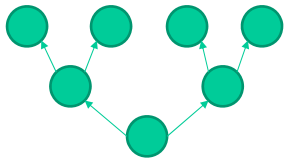
Short List
(hybrid neural/n-gram)

- Create subset $\mathbb{L}$ of frequently used words
- Train a neural net on $\mathbb{L}$
- Treat tail (remainder of vocabulary) using n-gram models: $\mathbb{T} = \mathbb{V} \backslash \mathbb{L}$

- Reduces complexity, but the words we don't model are the hard ones…

Hierarchical softmax

- Addresses large V problem
- Basic idea:
  - build binary hierarchy of word categories
  - words assigned to classes
  - each subnet has a small softmax layer
  - last subnet has manageable size
- Higher perplexity than non-hierarchical model



Importance sampling

- Consider large V softmax layer

$$\frac{\partial log P(y \mid C)}{\partial \theta} = \frac{\log softmax_y(a)}{\partial \theta}$$

$$= \frac{\partial}{\partial \theta} \log \frac{e^{a_y}}{\sum_i e^{a_i}}$$

$$= \frac{\partial}{\partial \theta}\left(a_y - \log \sum_i e^{a_i} \right)$$

$$= \frac{\partial}{\partial \theta}\left(a_y - \sum_i P(y = i \mid C) \right)$$

compute P of *every* other word

Importance sampling

- What if we could approximate the second half of: $\frac{\partial}{\partial \theta}\left(a_y - \sum_i P(y = i|C)\right)$?
- We could sample, but to sample we would need to know $P(y = i|C)$… seems like a dead end…

Importance sampling

- Importance sampling lets us sample from a different distribution
- Suppose we want to sample a function *f* on elements from distribution *p, e.g.* $E[f(X)] = \sum_i p_x(x_i) f(x_i)$, but we cannot draw from *p*.

Importance sampling

- Consider a new distribution *q:*

$$E[f(X)] = \sum_i \frac{p_q(x_i)\, p_x(x_i)\, f(x_i)}{p_q(x_i)}$$

- We can sample *x* based on *q*

$$\hat{E}[f(X)] = \frac{1}{N} \sum_{x_i \sim q} \frac{p_x(x_i)\, f(x_i)}{p_q(x_i)}$$

Importance sampling

- We can use an n-gram model as *q*
- Now we can sample and produce a cheaper estimate of probability
- See Goodfellow et al. 17.2 for more details on importance sampling