COMP90042 LECTURE 13

N-GRAM LANGUAGE MODELS

LANGUAGE MODELS

- Assign a probability to a sequence of words
- Useful for
  - Speech recognition
  - Spelling correction
  - Machine translation
  - …

But it must be recognized that the notion of "probability of a sentence" is an entirely useless one, under any known interpretation of this term.
-Noam Chomsky

OUTLINE

- Deriving n-gram language models
- Smoothing to deal with sparsity
- Evaluating language models

PROBABILITIES: JOINT TO CONDITIONAL

Our goal is to get a probability for an arbitrary sequence of m words

$$P(w_1, w_2, .. w_m)$$

First step is to apply the chain rule to convert joint probabilities to conditional ones

$$P(w_1,w_2,\ldots,w_m) = P(w_1)P(w_2|\, w_1)P(w_3|\, w_1,w_2)\ldots$$
$$P(w_m|w_1 \ldots w_{m-1})$$

THE MARKOV ASSUMPTION

Still intractable, so make a simplifying assumption:

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

For some small n

When $n = 1$, a unigram model

$$P(w_1, w_2, .. w_m) = \prod_{i=1}^{m} P(w_i)$$

When $n = 2$, a bigram model

$$P(w_1, w_2, .. w_m) = \prod_{i=1}^{m} P(w_i|w_{i-1})$$

When $n = 3$, a trigram model

$$P(w_1, w_2, .. w_m) = \prod_{i=1}^{m} P(w_i|w_{i-2}w_{i-1})$$

MAXIMUM LIKELIHOOD ESTIMATION

How do we calculate the probabilities? Estimate based on counts in our corpus:

For unigram models,

$$P(w_i) = \frac{C(w_i)}{\sum_{v \in V} C(v)}$$

For bigram models,

$$P(w_i | w_{i-1}) = \frac{C(w_{i-1} w_i)}{C(w_{i-1})}$$

For *n*-gram models generally,

$$P(w_i | w_{i-n+1} \dots w_{i-1}) = \frac{C(w_{i-n+1} \dots w_i)}{C(w_{i-n+1} \dots w_{i-1})}$$

TRIGRAM EXAMPLE

Corpus:

<s1> <s2> *yes no no no no yes* </s2> </s1>
<s1> <s2> *no no no yes yes yes no* </s2> </s1>

What is the probability of

<s1> <s2> *yes no no yes* </s2> </s1>

Under a trigram language model?

$P(yes \mid \text{<s1> <s2>}) = \frac{1}{2}$

$P(no \mid yes\ no) = \frac{1}{2}$

$P(\text{</s2>} \mid no\ yes) = \frac{1}{2}$

$P(no \mid \text{<s2>}\ yes) = 1$

$P(yes \mid no\ no) = \frac{2}{5}$

$P(\text{</s1>} \mid \text{yes </s2>}) = 1$

$= 0.05$

SEVERAL PROBLEMS

- Resulting probabilities are often very small
  - Use log probability to avoid numerical underflow
- No probabilities for unknown words
  - Convert infrequent words into <UNK> token
  - Or skip unknown words entirely
- Words in new contexts
  - By default, zero count for any n-gram we've never seen before, zero probability for the sentence
  - Need smoothing to give unseen events some probability

SMOOTHING (OR DISCOUNTING)

- Basic idea: give events you've never seen before some probability
- Have to take away probability from events you have seen
- Must be the case that P(everything) = 1
- Many different kinds of smoothing
  - Laplacian (add-one) smoothing
  - Add-k smoothing
  - Jelinek-Mercer interpolation
  - Katz backoff
  - Absolute discounting
  - Kneser-Ney
  - And others…

LAPLACIAN (ADD-ONE) SMOOTHING

- Simple idea: pretend we've seen each n-gram once more than we did.

For unigram models (**V**= the vocabulary),

$$P_{add1}(w_i) = \frac{C(w_i) + 1}{M + |\boldsymbol{V}|}$$

For bigram models,

$$P_{add1}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i) + 1}{C(w_{i-1}) + |\boldsymbol{V}|}$$

For n-gram models generally,

$$P_{add1}(w_i|w_{i-n-i} \dots w_{i-1}) = \frac{C(w_{i-n+1} \dots w_i) + 1}{C(w_{i-n+1} \dots w_{i-1}) + |\boldsymbol{V}|}$$

ADD-ONE EXAMPLE

<s> *the rat ate the cheese* </s>

What's the bigram probability *P(ate | rat)* under add-one smoothing?

$$= \frac{C(rat\ ate)+1}{C(rat)+|\boldsymbol{V}|} = \frac{2}{6}$$

What's the bigram probability *P(ate | cheese)* under add-one smoothing?

$$= \frac{C(cheese\ ate)+1}{C(cheese)+|\boldsymbol{V}|} = \frac{1}{6}$$

ADD-*K* SMOOTHING

- Adding one is always too much
- Instead, add a fraction k

$$P_{addk}(w_i|w_{i-n-i} \ldots w_{i-1}) = \frac{C(w_{i-n+1} \ldots w_i) + k}{C(w_{i-n+1} \ldots w_{i-1}) + k|\boldsymbol{V}|}$$

- Have to choose k
- Still not a competitive method
  - Smooths too indiscriminately

BACKOFF AND INTERPOLATION

- Smooth using lower-order probabilities (less context)
- Backoff: fall back to n-1-gram counts only when n-gram counts are zero

$$P_{BO}(w_i|w_{i-n+1} \ldots w_{i-1}) = \begin{cases} P^*(w_i|w_{i-n+1} \ldots w_{i-1}) & if\ C(w_{i-n+1} \ldots w_i) > 0 \\ \alpha(w_{i-n+1} \ldots w_{i-1})\, P_{BO}(w_i|w_{i-n+2} \ldots w_{i-1}) & otherwise \end{cases}$$

- P^* and α must preserve P(everything) = 1

BACKOFF AND INTERPOLATION

- Interpolation involves taking a linear combination of all relevant probabilities
- Defined recursively:

$$P_{interp}(w_i|w_{i-n+1}\ldots w_{i-1})$$
$$= \lambda(w_{i-n+1}\ldots w_{i-1})\, P(w_i|w_{i-n+1}\ldots w_{i-1})$$
$$+ (1 - \lambda(w_{i-n+1}\ldots w_{i-1}))P_{interp}(w_i|w_{i-n+2}\ldots w_{i-1})$$

- Interpolation of probabilities preserves P(everything) = 1
- λs can be constant across all contexts
  - But better if sensitive to $\mathtt{C}(w_{i-n+1}\ldots w_{i-1})$
- Parameters need to be trained on held out data

ABSOLUTE DISCOUNTING

- How much to take away from seen n-grams?
- Can get good estimate from corpora
  - Compare counts in two equally sized sets
- Turns out a single absolute discounting works for almost all n-grams
  - Most mass taken from low counts
  - Doesn't effect high counts much

Bigram count in training set	Bigram count in heldout set
0	0.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

CONTINUATION COUNTS

- When backing-off or interpolating, raw counts can be fairly unreliable
  - E.g. *Zealand* has high counts, but only appears after *New*
    - Don't want to assign it much probability when *New* not present
- Instead, count the number of unique contexts for the word
  - For many words, closely related to total count
  - But just 1 for Zealand

$$continuationcount(w_1 \dots w_n) = |\{v : count(vw_1 \dots w_n) > 0\}|$$

KNESER-NEY SMOOTHING

- Standard, state-of-the-art smoothing method
- Combines
  - Interpolation (or, alternatively, backoff)
  - Absolute discounting
  - Continuation counts for all i-grams ($i < n$)

KNESER-NEY SMOOTHING

$$P_{KN}(w_i|w_{i-n+1} \dots w_{i-1}) = \frac{\max(0, C_{KN}(w_{i-n+1} \dots w_i) - d)}{C_{KN}(w_{i-n+1} \dots w_{i-1})}$$

$$+ \lambda(w_{i-n+1} \dots w_{i-1}) P_{KN}(w_i|w_{i-n+2} \dots w_{i-1})$$

$C_{KN}(\circ)$ is regular counts for highest order n-gram, but continuation counts for all others

$$\lambda(w_{i-n+1} \dots w_{i-1}) = \frac{d}{C_{KN}(w_{i-n+1} \dots w_{i-1})} |\{w : C_{KN}(w_{i-n+1} \dots w_{i-1} w) > 0\}|$$

EVALUATION

- Extrinsic
  - E.g. spelling correction, machine translation
- Intrinsic
  - Perplexity on held-out test set

PERPLEXITY

- Inverse probability of entire test set
  - Normalized by number of words
- The lower the better

$$PP(w_1, w_2, .. w_m) = \sqrt[m]{\frac{1}{P(w_1, w_2, .. w_m)}}$$

EXAMPLE PERPLEXITY SCORES

- Wall Street Journal corpus
- Trained with 38 million words
- Tested on 1.5 million words

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

GENERATED TEXTS

- Language models can also be used to generate texts
- Can you guess the corpus used to create the sentences below?
- What do you think the *n is* for the *n*-gram language model?

This shall forbid it should be branded, if renown made it empty

They also point to ninety nine point six billion dollars from two hundred four oh three percent of the rates of interest stores as Mexico and Brazil on market conditions

A FINAL WORD

- N-gram language models are a structure-neutral way to capture the predictabilty of language
- Information can be derived in an unsupervised fashion, scalable to large corpora
- Require smoothing to be effective, due to sparsity
- Latest work in language models involve using recurrent neural networks
  - But in many circumstances not practical

REQUIRED READING

- J&M3 Ch. 4