COMP90042 LECTURE 11

N-GRAM LANGUAGE MODELS

LANGUAGE MODELS

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

OUTLINE

- Deriving n-gram language models
  - Easy: Markov models
- Smoothing to deal with sparsity
  - Hard: add-1 smoothing does not really work here
- 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:

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

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,

$$P(w_i) = \frac{C(w_i)}{M}$$

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?

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

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

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

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

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

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

$= 0.05$

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 to smooth the LM

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 trigram models generally,

$$P_{add1}(w_i|w_{i-1}, w_{i-2}) = \frac{C(w_{i-2}, w_{i-1}, w_i) + 1}{C(w_{i-2}, w_{i-1}) + |\boldsymbol{V}|}$$

ADD-ONE EXAMPLE

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

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

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

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

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

ADD-*K* SMOOTHING

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

$$P_{addk}(w_i|w_{i-1}, w_{i-2}) = \frac{C(w_{i-2}, w_{i-1}, w_i) + k}{C(w_{i-2}, w_{i-1}) + k|\boldsymbol{V}|}$$

- Have to choose k
- Still not a competitive method for language modelling
  - Works for text classification (and to some extent, POS tagging) because the number of classes is small.
  - Here, the number of "classes" is huge (n-grams) and the frequency can vary a lot.

KNESER-NEY SMOOTHING

- State-of-the-art method for n-gram language models.
- A fairly complex method, combining three ideas:
  - Interpolation (or alternative, backoff)
  - Absolute discounting
  - Continuation counts
- Let's see each of these steps in detail.

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

$$P_{BO}(w_i|w_{i-2},w_{i-1}) =$$

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

P* and α must preserve "sum to 1" property.

BACKOFF AND INTERPOLATION

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

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

- Interpolation of probabilities preserves "sum to 1" property
- λs can be constant across all contexts
  - But better if sensitive to n-grams
- Parameters need to be trained on held out data

ADD-1 AS RELATIVE DISCOUNTING

- We can reframe add-1 as using a smaller, **discounted** count in the probability calculation.
- $P(w_i|w_{i-1}) = \frac{C(w_{i-1},w_i)}{C(w_{i-1})}$ $\qquad P_{add1}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)+1}{C(w_{i-1})+|V|}$
- $C_*(w_{i-1}, w_i) = \frac{C(w_{i-1},w_i)+1}{C(w_{i-1})+|V|} * C(w_{i-1})$
- $P_{add1}(w_i|w_{i-1}) = \frac{C_*(w_{i-1},w_i)}{C(w_{i-1})}$
- This is called **relative** discounting:
  - $C_*(w_{i-1}, w_i) = C(w_{i-1}, w_i) * d_c$ $\qquad d_c = \frac{C_*(w_{i-1},w_i)}{C(w_{i-1},w_i)}$

ABSOLUTE DISCOUNTING

- Relative discounting use the counts from the training corpus
- What if we estimate the counts from a heldout corpus instead?
- 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

- $P_{Abs}(w_i|w_{i-1}) = \frac{C(w_{i-1},w_i)-d}{C(w_{i-1})} + \lambda(w_{i-1})P(w_i)$

CONTINUATION COUNTS

- When backing-off or interpolating, raw counts can be fairly unreliable
  - $P(Zealand|Old) = ?$ Will interpolate with $P(Zealand)$
  - *Zealand* has high counts, but only appears after *New*
    - Don't want to assign it much probability when *New* not present
- Instead, use frequency **at the type** level (instead of token): we call this **continuation counts**.
  - For many words, closely related to total count
  - But just 1 for *Zealand*

$$continuation\_count(w_1) = |\{v: count(v, w_1) > 0\}|$$

MIXING UP ALL TOGETHER

$$P_{KN}(w_i|w_{i-2}, w_{i-1}) = \frac{\max(0, C_{KN}(w_{i-2}, w_{i-1}, w_i) - d)}{C_{KN}(w_{i-2}, w_{i-1})}$$

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

$$\lambda(w_{i-2}, w_{i-1}) = \frac{d}{C_{KN}(w_{i-2}, w_{i-1})} |\{w: C_{KN}(w_{i-2}, w_{i-1}, w) > 0\}|$$

C_{KN} is a continuation count, **except for the highest n-gram order:** we use a regular count instead.

IN PRACTICE

- Best Kneser-Ney version uses different discount values for each n-gram order.
- Most used LMs use 5-grams as the max order but higher order sometimes can be used if large amounts of data are available.

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
- Given a initial word, **sample** the next word according to the probability distribution defined by the 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 predictability of language
- Information can be derived in an unsupervised fashion, scalable to large corpora
- Require smoothing to be effective, due to sparsity
- N-gram models do not (explicitly, at least) encode word similarity: remember the lexical and distributional semantics lectures?
- Next lecture: LMs with distributional semantics using neural networks.

REQUIRED READING

- J&M3 Ch. 4