Language Modeling

CSCI 544 – Fall 2016
8/31/2016

Kallirroi Georgila

Language modeling

- A language model is a probability distribution p(s) over strings s that reflects how frequently a string s occurs as a sentence
  - p(I like chocolate cake) > p(my like tomato cake)

Bigrams

$$p(s) = p(w_1)\,p(w_2 \mid w_1)\,p(w_3 \mid w_1 w_2)\ldots p(w_l \mid w_1 \ldots w_{l-1}) = \prod_{i=1}^{l} p(w_i \mid w_1 \ldots w_{i-1})$$

In bigrams the probability of a word depends only on the immediately preceding word

$$p(s) = \prod_{i=1}^{l} p(w_i \mid w_1 \ldots w_{i-1}) \approx \prod_{i=1}^{l} p(w_i \mid w_{i-1})$$

p(JOHN READ A BOOK)=p(JOHN|<BOS>)p(READ|JOHN)p(A|READ)p(BOOK|A)p(<EOS>|BOOK)

$$p(w_i \mid w_{i-1}) = \frac{c(w_{i-1} w_i)}{\sum_{w_i} c(w_{i-1} w_i)}$$

N-grams

In n-grams the probability of a word depends only on the immediately preceding n-1 words

$$p(s) = \prod_{i=1}^{l} p(w_i \mid w_{i-n+1}^{i-1})$$

$$p(w_i \mid w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i})}$$

Example of bigrams

- Training sentence 1: JOHN READ MOBY DICK
- Training sentence 2: MARY READ A DIFFERENT BOOK
- Training sentence 3: SHE READ A BOOK BY CHER
- Calculate p(JOHN READ A BOOK)

- p(JOHN|<BOS>) = $\frac{c(<BOS>, JOHN)}{\sum_w c(<BOS>, w)} = \frac{1}{3}$
- p(READ|JOHN) = $\frac{c(JOHN, READ)}{\sum_w c(JOHN, w)} = \frac{1}{1}$
- p(A|READ) = $\frac{c(READ, A)}{\sum_w c(READ, w)} = \frac{2}{3}$
- p(BOOK|A) = $\frac{c(A, BOOK)}{\sum_w c(A, w)} = \frac{1}{2}$
- p(<EOS>|BOOK) = $\frac{c(BOOK, <EOS>)}{\sum_w c(BOOK, w)} = \frac{1}{2}$

p(JOHN READ A BOOK) =

$$= \frac{1}{3} \times 1 \times \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2} \approx 0.06$$

Cross-entropy and perplexity

- For a test set T composed of the sentences (t_1,…,t_{l_T}), we can calculate the probability p(T) as follows

$$p(T) = \prod_{i=1}^{l_T} p(t_i)$$

- Cross-entropy (W_T is the length of text T measured in words)

$$H_p(T) = -\frac{1}{W_T} \log_2 p(T)$$

- Perplexity

$$PP_p(T) = 2^{H_p(T)}$$

Typical perplexities yielded by n-gram models on English text range from about 50 to almost 1000

Additive smoothing

$$p_{add}(w_i \mid w_{i-n+1}^{i-1}) = \frac{\delta + c(w_{i-n+1}^{i})}{\delta |V| + \sum_{w_i} c(w_{i-n+1}^{i})}$$

$0 < \delta \leq 1$ and $|V|$ is the size of the vocabulary

$\delta = 1$: add-one smoothing

Jelinek-Mercer smoothing

- c(BURNISH THE) = 0
- c(BURNISH THOU) = 0
- Then according to additive smoothing
  - p(THE|BURNISH) = p(THOU|BURNISH)
- But intuitively it should be
  - p(THE|BURNISH) > p(THOU|BURNISH)
- Because the word THE is much more common than the word THOU

Jelinek-Mercer smoothing (cont.)

$$p_{interp}(w_i \mid w_{i-1}) = \lambda p_{ML}(w_i \mid w_{i-1}) + (1-\lambda) p_{ML}(w_i)$$

$$p_{interp}(w_i \mid w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{ML}(w_i \mid w_{i-n+1}^{i-1}) + (1-\lambda_{w_{i-n+1}^{i-1}}) p_{interp}(w_i \mid w_{i-n+2}^{i-1})$$

We interpolate an n-gram (maximum likelihood based model) with an (n-1)-gram smoothed model (interpolated model)

The data used to estimate $\lambda_{w_{i-n+1}^{i-1}}$ need to be different from the data used to calculate p_{ML}

Witten-Bell smoothing

$$p_{WB}(w_i \mid w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{ML}(w_i \mid w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{WB}(w_i \mid w_{i-n+2}^{i-1})$$

To compute the parameters $\lambda_{w_{i-n+1}^{i-1}}$ for Witten-Bell smoothing, we need to use the number of unique words that follow the history w_{i-n+1}^{i-1}

$$N_{1+}(w_{i-n+1}^{i-1}\bullet) = |\{w_i : c(w_{i-n+1}^{i}) > 0\}|$$

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{N_{1+}(w_{i-n+1}^{i-1}\bullet)}{N_{1+}(w_{i-n+1}^{i-1}\bullet) + \sum_{w_i} c(w_{i-n+1}^{i})}$$

$1 - \lambda_{w_{i-n+1}^{i-1}}$ is the probability that a word not observed after the history w_{i-n+1}^{i-1} in the training data occurs after that history

$$p_{WB}(w_i \mid w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) + N_{1+}(w_{i-n+1}^{i-1}\bullet) p_{WB}(w_i \mid w_{i-n+2}^{i-1})}{\sum_{w_i} c(w_{i-n+1}^{i}) + N_{1+}(w_{i-n+1}^{i-1}\bullet)}$$

Absolute discounting

$$p_{\text{interp}}(w_i \mid w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{ML}(w_i \mid w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{\text{interp}}(w_i \mid w_{i-n+2}^{i-1})$$

$$p_{abs}(w_i \mid w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^{i}) - D, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{abs}(w_i \mid w_{i-n+2}^{i-1})$$

To make this distribution sum to 1, we take

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1} \bullet), 0 \le D \le 1$$

A good value for D is $D = \frac{n_1}{n_1 + 2n_2}$ where n_1 and n_2 are the the number of n-grams with exactly 1 and 2 counts in the training data

Kneser-Ney smoothing

$$p_{abs}(w_i \mid w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^{i}) - D, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{abs}(w_i \mid w_{i-n+2}^{i-1})$$

Consider the word FRANCISCO, a word that is very common, but occurs only after a single word (SAN). Since c(FRANCISCO) is high, the unigram probability p(FRANCISCO) will be high and an algorithm such as absolute discounting will assign a relatively high probability to the word FRANCISCO occurring after novel bigram histories
The unigram probability should not be proportional to the number of occurrences of the word, but instead to the number of different words that it follows

$$p_{KN}(w_i \mid w_{i-n+1}^{i-1}) = \begin{cases} \dfrac{\max\{c(w_{i-n+1}^{i}) - D, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} & \text{if } c(w_{i-n+1}^{i}) > 0 \\ \gamma(w_{i-n+1}^{i-1}) p_{KN}(w_i \mid w_{i-n+2}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) = 0 \end{cases}$$

$\gamma(w_{i-n+1}^{i-1})$ is chosen to make the distribution sum to 1

Kneser-Ney smoothing (cont.)

$$p_{KN}(w_i \mid w_{i-n+2}^{i-1}) = \frac{N_{1+}(\bullet w_{i-n+2}^{i})}{N_{1+}(\bullet w_{i-n+2}^{i-1}\bullet)}$$

$$N_{1+}(\bullet w_{i-n+2}^{i}) = |\{w_{i-n+1} : c(w_{i-n+1}^{i}) > 0\}|$$

$$N_{1+}(\bullet w_{i-n+2}^{i-1}\bullet) = |\{(w_{i-n+1}, w_i) : c(w_{i-n+1}^{i}) > 0\}| = \sum_{w_i} N_{1+}(\bullet w_{i-n+2}^{i})$$

Witten-Bell vs. Kneser-Ney – Example 1

- Consider the bigram “curriculum here”
- To calculate $P_{interp}(\text{here}|\text{curriculum})$, we can interpolate $P(\text{here}|\text{curriculum})$ with $P(\text{here})$
- The word “here” is common so an algorithm like Jelinek-Mercer will assign a high value to $P(\text{here})$ and consequently a high value to $P_{interp}(\text{here}|\text{curriculum})$
- But not many distinct words can follow “curriculum” which means that $P_{interp}(\text{here}|\text{curriculum})$ should not be high
- Witten-Bell takes into account how many distinct words can follow the word “curriculum” (not many words)
- Kneser-Ney takes into account the number of distinct words that the word “here” can follow (many words)
- Witten-Bell will assign a lower value to $P(\text{here})$ and consequently a lower value to $P_{interp}(\text{here}|\text{curriculum})$

Witten-Bell vs. Kneser-Ney – Example 2

- Consider the bigram “good francisco”
- To calculate $P_{interp}(\text{francisco}|\text{good})$, we can interpolate $P(\text{francisco}|\text{good})$ with $P(\text{francisco})$
- The word “francisco” is common so an algorithm like Jelinek-Mercer will assign a high value to $P(\text{francisco})$ and consequently a high value to $P_{interp}(\text{francisco}|\text{good})$
- But the word “francisco” does not follow many distinct words which means that $P_{interp}(\text{francisco}|\text{good})$ should not be high – usually “francisco” just follows the word “san”
- Witten-Bell takes into account how many distinct words can follow the word “good” (many words)
- Kneser-Ney takes into account the number of distinct words that the word “francisco” can follow (not many words)
- Kneser-Ney will assign a lower value to $P(\text{francisco})$ and consequently a lower value to $P_{interp}(\text{francisco}|\text{good})$

Applications of language modeling

- Speech recognition: predict a sentence (sequence of words word_seq) given the acoustics

$$\arg\max_{word\_seq} p(word\_seq \mid acoustics) =$$

$$\arg\max_{word\_seq} \frac{p(acoustics \mid word\_seq) \times p(word\_seq)}{P(acoustics)} \propto$$

$$\arg\max_{word\_seq} p(acoustics \mid word\_seq) \times p(word\_seq)$$

Applications of language modeling (cont.)

- Word prediction
  - Once upon _____
- Spelling correction

$$p(words \mid characters) \propto P(words)P(characters \mid words)$$

- Handwriting recognition

$$p(words \mid strokes) \propto P(words)P(strokes \mid words)$$

General language models vs. domain-specific language models

User Input	Google Chrome ASR Output (general language model)	Apple Dictation Output (general language model)	CMU PocketSphinx Output (domain-specific language model)
hello pinchas	hello pinterest	hello princess	hello pinchas
where is lodz	where is lunch	where is lunch	where is lodz
were you in majdanek	were you in my dannic	were you in my donick	were you in majdanek
were you in kristallnacht	were you and krystal knox	where you went kristallnacht	were you when kristallnacht from
did you serve in the army	did you serve in the army	he served in the army	did you certain the army
what's your favorite restaurant	what's your favorite restaurant	what's your favorite restaurant	what's your favorite rest shot

Solution: interpolate domain-specific with general language models

Language modeling toolkits

- CMU language modeling toolkit
- SRI language modeling toolkit
- MIT language modeling toolkit
- Language modeling library of HTK

References

- Stanley F. Chen & Joshua Goodman, An empirical study of smoothing techniques for language modeling