NATURAL LANGUAGE PROCESSING (PRACTICE)

NLP 251 - Lab 3: Language Models

Department of Computer Science and Engineering
Ho Chi Minh University of Technology, VNU-HCM

Language Model

Language Model

Language Model: A language model is a model that predicts the probability of a sequence of words.

$$P(W) = P(w_1, w_2, \ldots, w_n)$$

Example

- $S_1 =$ "The cat jumped over the dog." $\Rightarrow P(S_1) \approx 1$
- $S_2 =$ "The jumped cat the over dog." $\Rightarrow P(S_2) \approx 0$

Application

- Machine Translation
  - $P(\text{high winds tonight}) > P(\text{large winds tonight})$
- Text Correction
  - The office is about fifteen *minutes* from my house
  - $P(\text{"about fifteen minutes from"}) > P(\text{about fifteen minuets from})$
- Speech Recognition
  - $P(\text{I saw a van}) > P(\text{eyes awe of an})$
- Handwriting Recognition
  - $P(\text{Act naturally}) > P(\text{Abt naturally})$
- Summarization, Q&A, etc.

Conditional probability

Formula

$$P(A|B) = \frac{P(A \cap B)}{P(A)} \text{ or } P(A,B) = P(A) \cdot P(B|A)$$
$$P(A,B,C,D) = P(A) \cdot P(B|A) \cdot P(C|A,B) \cdot P(D|A,B,C)$$

Therefore

$$P(S) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1, w_2) \cdots P(w_n|w_1, w_2, w_3, \ldots, w_n)$$

Example

$$\begin{aligned} P(\text{Computer,can,recognize,speech}) = & P(\text{Computer}) \cdot \\ & P(\text{can}|\text{Computer}) \cdot \\ & P(\text{recognize}|\text{Computer can}) \cdot \\ & P(\text{speech}|\text{Computer can recognize}) \end{aligned}$$

N-gram Model

Markov hypothesis

Markov property

$$P(S) = \prod_{i=1}^{n} P(w_i|w_1, w_2, \ldots, w_{i-1}) \Rightarrow P(S) = \prod_{i=1}^{n} P(w_i|w_{i-1})$$

Example:

$$\begin{aligned}
P(\text{Computer,can,recognize,speech}) =& P(\text{Computer})\cdot \\
& P(\text{can}|\text{Computer})\cdot \\
& P(\text{recognize}|\text{Computer can})\cdot \\
& P(\text{speech}|\text{Computer can recognize}) \\
P(\text{Computer,can,recognize,speech}) =& P(\text{Computer}) \cdot P(\text{can}|\text{Computer})\cdot \\
& P(\text{recognize}|\text{can}) \cdot P(\text{speech}|\text{recognize})
\end{aligned}$$

N-GRAM model

- **Unigram (1-gram):** probability of a word independent of previous words

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$

- **Bigram (2-gram):** probability of a word given the previous word

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i | w_{i-1})$$

- **Trigram (3-gram):** probability of a word dependent on two previous words

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i | w_{i-1}, w_{i-2})$$

- **N-gram:** probability of a word dependent on N previous words

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i | w_{i-1}, w_{i-2}, w_{i-N})$$

Likelihood Estimate

Likelihood Estimate of Bigram (2-gram):

$$P(w_i \mid w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

Example:

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

$$P(\text{I}|\text{<s>}) = \frac{2}{3} = .67 \quad P(\text{Sam}|\text{<s>}) = \frac{1}{3} = .33 \quad P(\text{am}|\text{I}) = \frac{2}{3} = .67$$

$$P(\text{</s>}|\text{Sam}) = \frac{1}{2} = 0.5 \quad P(\text{Sam}|\text{am}) = \frac{1}{2} = .5 \quad P(\text{do}|\text{I}) = \frac{1}{3} = .33$$

Smoothing and Zeros

The Zero-Probability Problem

Training set:

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Test set:

- ... denied the offer
- ... denied the loan
- ...
- ...

$P(\text{"offer"}|\text{denied the}) = 0$. (This means we will assign a probability of 0 to the above sentence)

Smoothing

- **Laplace smoothing:**

$$P_{\text{Laplace}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c_{w_{i-1}} + V}$$

- **Linear Interpolation:**

$$P(w_n \mid w_{n-1} w_{n-2}) = \lambda_1 P(w_n \mid w_{n-1} w_{n-2}) + \lambda_2 P(w_n \mid w_{n-1}) + \lambda_3 P(w_n)$$

with $\lambda_1 + \lambda_2 + \lambda_3 = 1$. For interpolation with context-conditioned weights, where each lambda takes an argument that is the two prior word context:

$$\begin{aligned} P(w_n|w_{n-2}w_{n-1}) = \lambda_1(w_{n-2,n-1})P(w_n) + \lambda_2(w_{n-2,n-1})P(w_n|w_{n-1}) \\ + \lambda_3(w_{n-2,n-1})P(w_n|w_{n-2}w_{n-1}) \end{aligned}$$

Smoothing

Use valid set to choose parameter :

- Keep N-gram probabilities fixed (from training data)
- Then find λ that maximize probability on valid set:

$$\log P(w_1 ... w_n | M(\lambda_1 ... \lambda_k)) = \sum_i \log P_{M(\lambda_1 ... \lambda_k)}(w_i | w_{i-1})$$

Besides the methods mentioned above, we also have methods such a

- **Good-Turing**
- **Kneser-Ney**
- **Witten-Bell**
- **Backoff**

Huge Language Models and Stupid Backoff

- Solving large-value problems, e.g., Google N-gram corpus
- Pruning
    - Only store N-grams with frequency > threshold
    - Remove higher-order n-gram entries
- Efficiency
    - Use efficient data structures like tries
    - Bloom filters: approximate language model matching
    - Store words as indices, not strings
        - Use Huffman coding to convert large words into 2 bytes
    - Probability quantization (4-8 bits instead of 8-byte float)

Stupid backoff algorithm helps the model maintain a manageable size while still achieving reasonable effectiveness.

Model Evaluation

Perplexity Metric

Perplexity is the inverse probability of the test set, normalized by the number of words

$$PP(W) = P(w_1 w_2 \dots w_N)^{\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}} = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \dots w_{i-1})}}$$

- Minimizing perplexity is equivalent to maximizing probability
- Low perplexity = good model

With **WSJ dataset**: training set of 38 million words, test set of 1.5 million words

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Shannon Visualization Method

Algorithm

- Choose a random bigram (<s>, w) according to its probability
- Now choose a random bigram (w, x) according to its probability
- And so on until we choose </s>
- Then string the words together

<s> I
I want
want to
to eat
eat Chinese
Chinese food
food </s>
I want to eat Chinese food

N-Value

What is the appropriate value of n?

- Theoretically, very difficult to determine
- However: as large as possible ($\rightarrow$ approaches the "perfect" model)
- Empirically, $n = 3$ is common
    - Parameter estimation? (confidence, data, storage, space, ...)
    - 4 is too large: $|V| = 60k \rightarrow 1.296 \times 10^{19}$ parameters
    - However: $6 - 7$ possible with sufficient data: in practice, we can recover from 7-grams!

THANKS FOR LISTENING!