NLP and Deep Learning
MAT3399

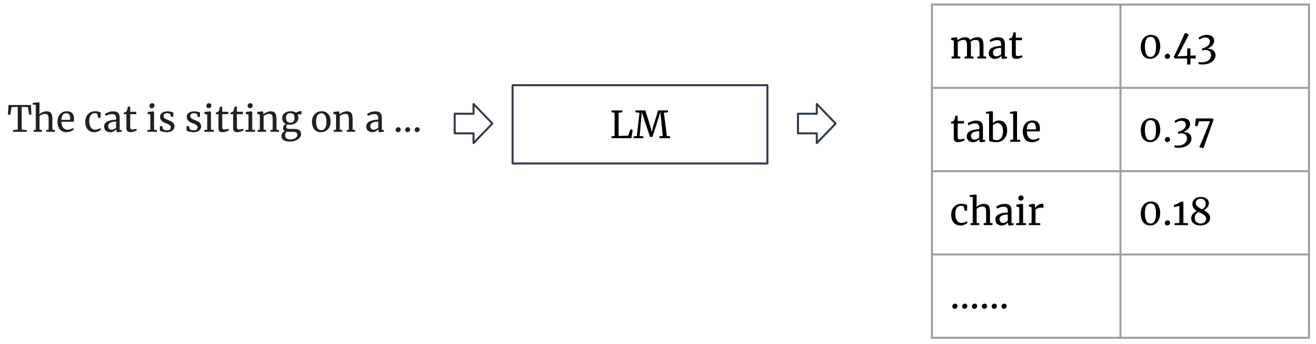
Lecture 4:
Language Model

Tuan Anh Nguyen @ Aimesoft
ted.nguyen95@gmail.com

Content taken from Speech and Language Processing by Daniel Jurafsky and James Martin

What is Language Model?

Language modeling (LM) is the use of various statistical and probabilistic techniques to determine the probability of a given sequence of words occurring in a sentence.



mat	0.43
table	0.37
chair	0.18
……	

N-grams LM

Compute the probability of the word w, given history h:

$$P(w \mid h)$$

For example, the history is “the cat is sitting on a” and we want to calculate the probability that the next word is “mat”:

$$P(\text{mat} \mid \text{the cat is sitting on a})$$

$$= \frac{C(\text{the cat is sitting on a mat})}{C(\text{the cat is sitting on a})}$$

C(x) is the times that x appears in the corpus

N-grams LM Intuition

Chain rule of probability:

$$\begin{aligned} P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n\boxed{|w_1^{n-1})} \quad = w_1w_2\ldots w_{n-1} \\ &= \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \end{aligned}$$

We can't just estimate by counting the number of times every word occurs following every long string, because language is creative and any particular context might have never occurred before!

Instead of computing the probability of a word given its entire history, we can approximate the history by just **the last few words**.

2-grams,..., n-grams

Bi-grams:

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

N-grams:

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

Practice

Given this corpus:

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

Calculate the probability of <s> I am </s> using unigram, bi-gram?

Handle OOV words

Method 1:

- Choose a fixed vocabulary
- Convert in the training set any word that is not in your vocabulary to unknown token <UNK>
- Estimate the probabilities for <UNK> like regular words

Method 2:

Convert all words that appear less than *k* times in the training set to <UNK>

Handle zero probability

Add-k smoothing

$$P^*_{\text{Add-k}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+k}{C(w_{n-1})+kV}$$

We call it Laplace smoothing in case k = 1

Evaluating LM using perplexity (PP)

For a test set $W = w_1 w_2 \ldots w_N$

$$\begin{aligned}\mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}\end{aligned}$$

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

In case bi-gram

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

The lower the PP the better

Coding Exercise

- Implement bi-gram LM without using any library
- Calculate perplexity of your LM on any text

Dataset

Advanced exercise: Implement a spell corrector using your LM