Natural Language Processing Application

Week 3: Language Model

fit@hcmus
KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

fit@hcmus

- Introduction to n-gram
- Estimating N-gram Probabilities
- N-gram model evaluation
- Smoothing techniques

fit@hcmus

XLNNTNUD - Language Model

INTRODUCTION TO N-GRAM

fit@hcmus

Introduction to n-gram

- Probabilistic Language Model: assign a probability to a sentence
  - Machine Translation:
    - P(ngôn ngữ tự nhiên) > P(ngôn ngữ nhiên tự)
  - Spelling error detection and correction:
    - P(ngôn ngữ tự nhiên) > P(gôn ngữ tu nhiên)
  - Text summarization
  - Question-Answering
  - ...

fit@hcmus

Introduction to n-gram (cont)

- Probability of a sentence (or a sequence of words):
  - $P(W) = P(w_1, w_2, w_3...w_n)$
- Probability of the next word in a sentence (or a sequence of words):
  - $P(w_n | w_1, w_2...w_{n-1})$
- Model $P(W)$ or $P(w_n | w_1, w_2...w_{n-1})$ is called a **Language Model**

fit@hcmus

Introduction to n-gram (cont)

- The Chain Rule of Probability:
  - Two variables: $P(x_1,x_2) = P(x_1)P(x_2|x_1)$
  - Three variables: $P(x_1,x_2,x_3) = P(x_1)P(x_2|x_1)P(x_3|x_1x_2)$
  - Four variables: $P(x_1,x_2,x_3,x_4) = P(x_1)P(x_2|x_1)P(x_3|x_1x_2)P(x_4|x_1x_2x_3)$
  - …
  - N variables: $P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$

fit@hcmus

Introduction to n-gram (cont)

- The Chain Rule of Probability:

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

- Example:
  - P(ngôn ngữ tự nhiên) = P(ngôn) x P(ngữ|ngôn) x P(tự|ngôn ngữ)

    x P(nhiên|ngôn ngữ tự)

fit@hcmus

Introduction to n-gram (cont)

- Estimating the probability:
  - P(ngôn) = count(ngôn)/N
  - P(ngữ|ngôn) = count(ngôn ngữ)/count(ngôn)
  - P(tự|ngôn ngữ) = count(ngôn ngữ tự)/count(ngôn ngữ)
  - P(nhiên|ngôn ngữ tự) = count(ngôn ngữ tự nhiên)/count(ngôn ngữ tự)
- Comment:
  - There are too many possibilities
  - Not enough data for estimation

fit@hcmus

Introduction to n-gram (cont)

- Markov Assumption:
  - P(nhiên|ngôn ngữ tự) ≈ P(nhiên|tự)

    Or
  - P(nhiên|ngôn ngữ tự) ≈ P(nhiên|ngữ tự)

fit@hcmus

Introduction to n-gram (cont)

- Markov Assumption:

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

fit@hcmus

Introduction to n-gram (cont)

- Unigram model (1-gram):

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

- Automatically generated sentences from a unigram model:

ở trận, fabregas, cesc bàn dusan và cầu kiến emmanuel thứ reyes, utd trong sau tạo anh ngoại anh thủ pogba man một jose xuất ở này.

là tadic adebayor, thủ harry dennis santi cazorola, nhóm thành bergkamp, bốn tiên bảy hạng kane. đầu cầu hiện antonio

fit@hcmus

Introduction to n-gram (cont)

- Bigram model (2-gram):

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

- Automatically generated sentences from a bigram model:

```
anh thành cầu ở ngoại hạng anh tạo bàn trong một trận, sau dennis
bergkamp, và harry kane.
pogba là man utd đầu xuất hiện ở nhóm này.
```

fit@hcmus

Introduction to n-gram (cont)

- Extension: trigram, 4-gram, 5-gram…
- Comment:
  - The effect of long-distance dependency in language
    - Ví dụ: "**Chiếc máy tính** mà tôi vừa đưa vào phòng máy trên tầng năm đã **bị hỏng**."
  - However, n-gram model should work fine in most cases

fit@hcmus

XLNNTNUD - Language Model

ESTIMATING N-GRAM PROBABILITIES

fit@hcmus

Estimating n-gram probabilities

- Maximum Likelihood Estimation

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

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

Estimating n-gram probabilities (cont)

- Example:

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

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

$$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$$

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

fit@hcmus

Estimating n-gram probabilities (cont)

- Example: bigram count (9222 sentences)

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

fit@hcmus

Estimating n-gram probabilities (cont)

- Normalize by unigram:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Result:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Estimating n-gram probabilities (cont)

- Example:

$$
\begin{aligned}
&P(\text{<s> I want english food </s>}) = \\
&\quad P(\text{I}\,|\,\text{<s>}) \\
&\quad \times\ P(\text{want}\,|\,\text{I}) \\
&\quad \times\ P(\text{english}\,|\,\text{want}) \\
&\quad \times\ P(\text{food}\,|\,\text{english}) \\
&\quad \times\ P(\text{</s>}\,|\,\text{food}) \\
&\quad\ =\ .000031
\end{aligned}
$$

fit@hcmus

Estimating n-gram probabilities (cont)

- Knowledge from the probability:
  - P(english|want) = .0011
  - P(chinese|want) = .0065
  - P(to|want) = .66
  - P(eat | to) = .28
  - P(food | to) = 0
  - P(want | spend) = 0
  - P (i | <s>) = .25

fit@hcmus

Estimating n-gram probabilities (cont)

- Problem with Multiplication:
  - Underflow
  - Slow
- Transform Multiplication into Addition:

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Estimating n-gram probabilities (cont)

- Language Modeling Toolkits:
  - SRILM
  - IRSTLM
  - KendLM
  - ...

fit@hcmus

XLNNTNUD - Language Model

MODEL EVALUATION

fit@hcmus

Model Evaluation

- Language Model (comparing “good" and “not good" sentence):
  - Assign higher probability to “real" or “frequently seen" sentences than “ungrammatical" or “rarely seen" sentences.
- Model’s parameters are trained on a **training set**
- The model’s performance are tested on unseen data
  - **A test set** is an unseen dataset, separate from the training set
  - **An evaluation metric** show how good our model does on the test set

Model Evaluation (cont)

- Extrinsic Evaluation: to compare models A and B
  - Give each model a task:
    - Spelling correction, Machine Translation…
  - Run the task and get an accuracy for A and B
    - How many misspelled words corrected properly
    - How many words translated correctly
  - Compare accuracy for A and B

fit@hcmus

Model Evaluation (cont)

- Extrinsic Evaluation:
  - Time consuming (days or even weeks to complete…)
- Therefore, Intrinsic evaluation is sometimes used: perplexity
  - Bad approximation:
    - If the test data doesn’t look like the training data
    - Only useful in pilot experiment

fit@hcmus

Model Evaluation (cont)

- Perplexity:
  - How well can we predict the next word?

I always order pizza with cheese and ____

The 33rd President of the US was ____

I saw a ____

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

....

fried rice 0.0001

....

and 1e-100

- Unigram are not good in this situation ?
- A good model is one assigns a higher probability to the word that actually occurs

fit@hcmus

Model Evaluation (cont)

- Perplexity:
  - The best language model is one that best predicts an unseen test set
  - Perplexity is the inverse probability of the test set, normalized by the number of words

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

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

fit@hcmus

Model Evaluation (cont)

- Perplexity:
  - Number Recognition 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    - Perplexity = 10
  - Recognizing (30K) name:
    - Perplexity = 30000
  - A System:
    - Management (1/4)
    - Business (1/4)
    - Assistance (1/4)
    - 30K name
      - Perplexity = 53

fit@hcmus

Model Evaluation (cont)

- Perplexity:
  - Lower Perplexity = Better Model
- Training set 3M words, Test set 1.5M words (WSJ)

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

fit@hcmus

XLNNTNUD - Language Model

SMOOTHING TECHNIQUES

Smoothing Techniques

- Shakespeare corpus :
  - N=884,647 tokens, V=29,066
  - 300K bigram (reality) out of V^2 = 844M bigram (possibility)

    => 99.96% bigram (possible) were never seen (probability 0)

fit@hcmus

Smoothing Techniques (cont)

- The problem:
  - Training set:
    - … denied the allegations
    - … denied the reports
    - … denied the claims
    - … denied the request

    P(“offer” | denied the) = 0
  - Test set
    - … denied the offer
    - … denied the loan

fit@hcmus

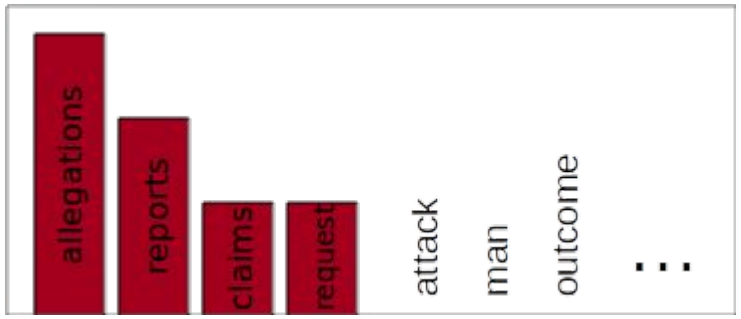
Smoothing Techniques (cont)

- The problem:
  - Bigram with zero probability
    - Assign probability zero to the test set
  - Therefore we cannot calculate perplexity (Division by zero)

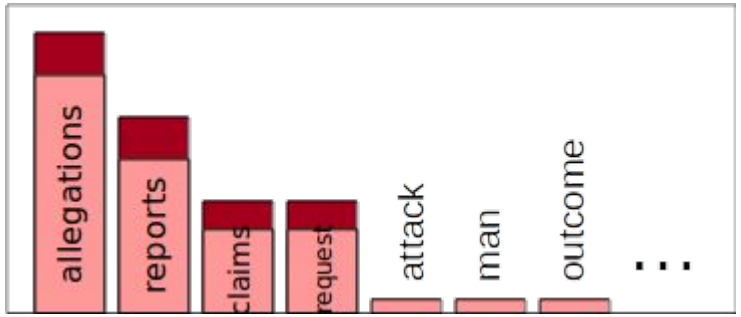
fit@hcmus

Smoothing Techniques (cont)

- Add-one (Laplace) Smoothing:
  - Sparse statistics: P(w | denied the)
    3 allegations
    2 reports
    1 claims
    1 request
    7 total



  - Generalize better probability
    P(w | denied the)
    2.5 allegations
    1.5 reports
    0.5 claims
    0.5 request
    2 other
    7 total



fit@hcmus

Smoothing Techniques (cont)

- Add-one (Laplace) Smoothing:
  - Pretend we saw each word one more time
  - Add one to all the counts

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})} \qquad P_{Add-1}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i)+1}{C(w_{i-1})+V}$$

fit@hcmus

Smoothing Techniques (cont)

- Add-one (Laplace) smoothing:

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

fit@hcmus

Smoothing Techniques (cont)

- Add-one (Laplace) smoothing

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

fit@hcmus

Smoothing Techniques (cont)

- Add-one (Laplace) smoothing

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

fit@hcmus

Smoothing Techniques (cont)

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Smoothing Techniques (cont)

- Other techniques:
  - Recursive Interpolation
  - Backtracking
  - Good Turing
  - Kneser-Ney
  - ...