Bài 4: Mô hình ngôn ngữ (Language Models)

About the lecturer

- Phd Nguyen Kiem Hieu
- Computer science department, School of Information and Communication Technology, HUST
- Email: hieunk@soict.hust.edu.vn

Nội dung

1. Giới thiệu bài toán dự báo từ và sửa lỗi chính tả
2. Định nghĩa mô hình ngôn ngữ
3. Ý nghĩa và ứng dụng
4. Mô hình ngôn ngữ dựa trên N-gram
5. Đánh giá mô hình
6. Zero problem và các phương pháp làm trơn
7. Giới thiệu tóm tắt các mô hình dựa trên mạng Neural
8. Ứng dụng mô hình ngôn ngữ cho sửa lỗi chính tả

Bài toán dự đoán từ và sửa lỗi chính tả

- Dự báo từ: Gợi ý từ/cụm từ tiếp theo trong quá trình soạn thảo.
- Ví dụ: Soạn tin nhắn trên điện thoại di động
- Sửa lỗi chính tả: Tìm từ có thể sai chính tả và gợi ý từ thay thế cho đúng

Ứng dụng

- Soạn thảo văn bản nhanh và chính xác
  - Người khuyết tật
  - Người không quen thao tác với các thiết bị điện tử
  - Người không thành thạo ngôn ngữ, tránh lỗi ngữ pháp
- Tăng giá trị của sản phẩm soạn thảo văn bản.

Định nghĩa mô hình ngôn ngữ

- Mô hình ngôn ngữ nhằm xác định khả năng (likelihood) của một câu (hoặc một chuỗi từ) thuộc một ngôn ngữ.
- Mô hình ngôn ngữ (xác suất) là mô hình cho phép tính xác suất của một câu (chuỗi các từ) trong một ngôn ngữ.

Goal: compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$$

P(‘bò’, ‘vàng’, ‘gặm’, ‘cỏ’), P(‘bò’, ‘nàng’, ‘gặm’, ‘cỏ’)

Ứng dụng

- Nhận dạng tiếng nói
- Nhận dạng chữ viết từ ảnh (OCR)
- Dịch máy
- Kiểm lỗi chính tả
- …

$$\arg\max_{W} P(W \mid A) = \arg\max_{W} \frac{P(A \mid W)P(W)}{P(A)}$$

Mô hình ngôn ngữ = mô hình dự đoán từ

- Chain rule

$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

- P(“its water is so transparent”) =
- $P(\text{its}) \times P(\text{water}|\text{its}) \times P(\text{is}|\text{its water})$
- $\times P(\text{so}|\text{its water is}) \times P(\text{transparent}|\text{its water is so})$

Dự đoán từ

- Mô hình ngôn ngữ cũng chính là mô hình dự đoán từ dựa trên các từ phía trước

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

Tính xác suất của Từ trong ngữ cảnh dựa trên N-gram

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

$$P(\text{the} \mid \text{its water is so transparent that}) =$$

$$\frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

Markov Assumption


Andrei Markov

- Simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

- Or maybe

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

Giả thuyết Markov

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

•In other words, we approximate each component in the product

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

Mô hình Unigram (1-gram)

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

Xác suất của một từ không phụ thuộc vào từ phía trước

fifth, an, of, futures, the, an, incorporated, a, a, the, inflation, most, dollars, quarter, in, is, mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the

Mô hình Bigram (2-gram)

- Xác suất một từ xuất hiện sau một từ cho trước:

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

texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november

Ước lượng bigram probabilities

- Sử dụng Maximum Likelihood Estimate

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

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

Ví dụ

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

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

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

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

Thống kê trên dữ liệu: Berkeley Restaurant Project sentences

Out of 9222 sentences

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day
- …

Bigram counts
(Đếm bigram)

	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

Bigram probabilities

•Normalize by unigrams:

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

Kinds of Knowledge

- 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

• World knowledge

•Syntax

•Discourse

Ví dụ

Cho tập dữ liệu văn bản gồm các câu sau:

<s> cô ấy dạy môn tin học </s>

<s> anh dạy môn toán </s>

<s> cô ấy học toán anh ấy dạy </s>

<s> môn toán môn tin đều hay *</s>*

<s> anh ấy dạy môn toán hay môn tin </s>

Xây dựng mô hình ngôn ngữ unigram và bigram?

Language Modeling

Evaluation and Perplexity

Các mô hình ngôn ngữ khác nhau phụ thuộc vào yếu tố nào?
Làm thế nào để so sánh mô hình A và mô hình B?

Đánh giá ngoài (Extrinsic evaluation)

- Mô hình ngôn ngữ A và B được sử dụng trong một bài toán X khác:
  - Bài toán speech recognition, spelling, machine translation
- So sánh mô hình A với mô hình B tương ứng với so sánh kết qủa ứng dụng A với B trong bài toán X.

Đánh giá trong (intrinsic evaluation)

- Sử dụng dữ liệu Test là các câu trong ngôn ngữ
- Sử dụng độ đo Perplexity

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)

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

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

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

Chain rule

For bigrams:

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

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

Minimizing perplexity is the same as maximizing probability

Perplexity as branching factor

- Let's suppose a sentence consisting of random digits
- What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

$$\begin{aligned}
\mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}} \\
&= \frac{1}{10}^{-1} \\
&= 10
\end{aligned}$$

Lower perplexity = better model

- Training 38 million words, test 1.5 million words, WSJ

Perplexity	962	170	109

Lower perplexity = better model

- Training 38 million words, test 1.5 million words, WSJ

	Unigram	Bigram	Trigram
Perplexity	962	170	109

The Shannon Visualization Method

- 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

Approximating Shakespeare

Unigram

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
Every enter now severally so, let
Hill he late speaks; or! a more to leg less first you enter
Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

Bigram

What means, sir. I confess she? then all sorts, he is trim, captain.
Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

Trigram

Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
This shall forbid it should be branded, if renown made it empty.
Indeed the duke; and had a very good friend.
Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

Quadrigram

King Henry.What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
Will you not tell me who I am?
It cannot be but so.
Indeed the short and the long. Marry, 'tis a noble Lepidus.

The wall street journal is not shakespeare (no offense)

Unigram

Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives

Bigram

Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her

Trigram

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Sử dụng mô hình N-gram với N = ?

How large *n*?

- Nothing is enough (theoretically)
- But anyway: as much as possible (→ close to “perfect” model)
- Empirically: ***3***
  - parameter estimation? (reliability, data availability, storage, space, ...)
  - 4 is too much: |V|=60k → 1.296 x 1019 parameters
  - but: 6-7 would be (almost) ideal (having enough data): *in fact, one can recover original from 7-grams!*

Shakespeare as corpus

- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams.
  - So 99.96% of the possible bigrams were never seen (have zero entries in the table)
- Quadrigrams worse: What's coming out looks like Shakespeare because it ***is*** Shakespeare

Language Modeling

Zeros problem and Smoothing: Add-one (Laplace) smoothing

Zeros

- Training set:

… denied the allegations
… denied the reports
… denied the claims
… denied the request

P(“offer” | denied the) = 0

- Test set

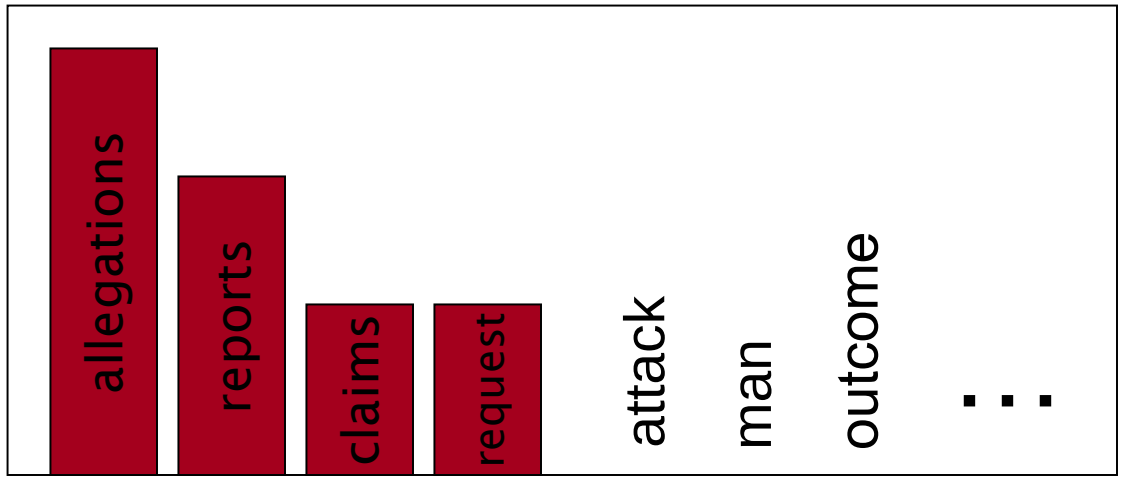
… denied the offer
… denied the loan

mean that we will assign 0 probability to the sentence

The intuition of smoothing (from Dan Klein)

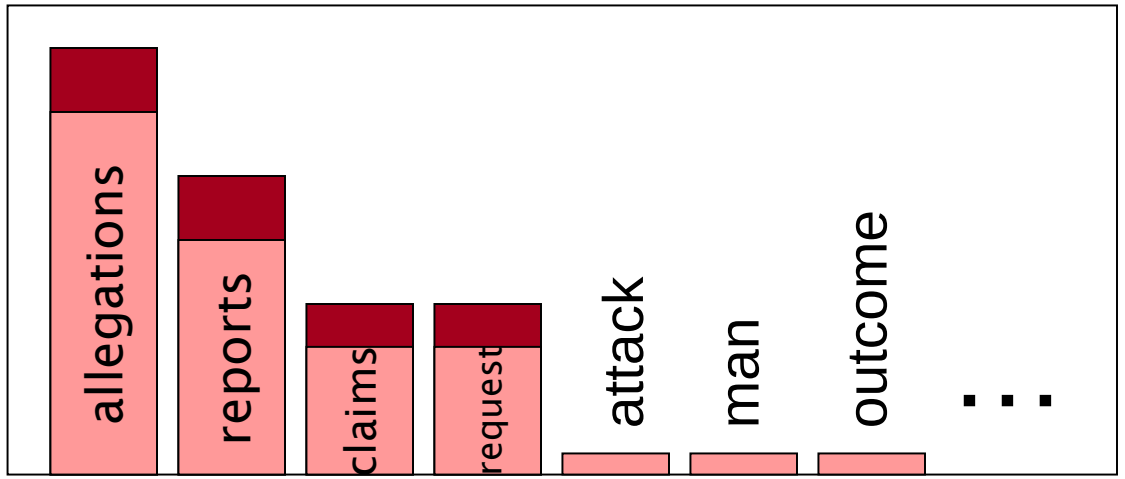
•When we have sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request



•Steal probability mass to generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other



Add-one estimation

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to all the counts!

- MLE estimate:

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

- Add-1 estimate:

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

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

	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

Laplace-smoothed bigrams

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

	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

Compare with the original probabilies

	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

	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

Reminder: Add-1 (Laplace) Smoothing

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

Language Modeling

Interpolation, Backoff, and Web-Scale LMs

Backoff and Interpolation

- Sometimes it helps to use **less** context
  - Condition on less context for contexts you haven't learned much about
- **Backoff:**
  - use trigram if you have good evidence,
  - otherwise bigram, otherwise unigram
- **Interpolation:**
  - mix unigram, bigram, trigram

- Interpolation works better

Linear Interpolation

- Simple interpolation

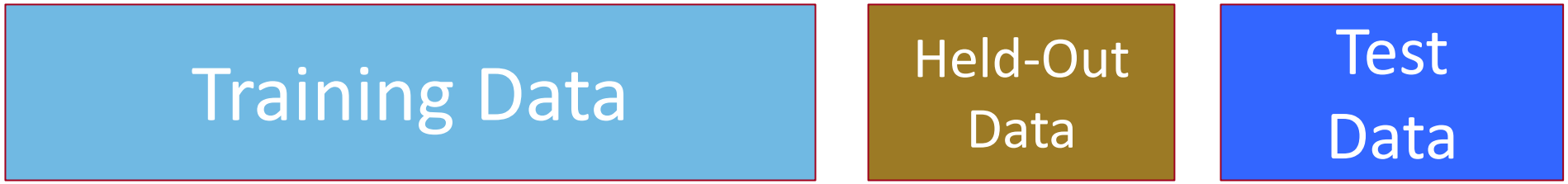
$$\hat{P}(w_n|w_{n-1}w_{n-2}) = \lambda_1 P(w_n|w_{n-1}w_{n-2}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

$$\sum_i \lambda_i = 1$$

- Lambdas conditional on context:

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

How to set the lambdas?



- Use a **held-out** corpus
- Choose λs to maximize the probability of held-out data:
  - Fix the N-gram probabilities (on the training data)
  - Then search for λs that give largest probability to held-out set:

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

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced
  - Vocabulary V is fixed
  - Closed vocabulary task
- Often we don't know this
  - **Out Of Vocabulary** = OOV words
  - Open vocabulary task
- Instead: create an unknown word token <UNK>
  - Training of <UNK> probabilities
    - Create a fixed lexicon L of size V
    - At text normalization phase, any training word not in L changed to <UNK>
    - Now we train its probabilities like a normal word
  - At decoding time
    - If text input: Use UNK probabilities for any word not in training

Huge web-scale n-grams

- How to deal with, e.g., Google N-gram corpus
- Pruning
  - Only store N-grams with count > threshold.
    - Remove singletons of higher-order n-grams
  - Entropy-based pruning
- Efficiency
  - Efficient data structures like tries
  - Bloom filters: approximate language models
  - Store words as indexes, not strings
    - Use Huffman coding to fit large numbers of words into two bytes
  - Quantize probabilities (4-8 bits instead of 8-byte float)

Smoothing for Web-scale N-grams

- "Stupid backoff" (Brants *et al*. 2007)
- No discounting, just use relative frequencies

$$S(w_i \mid w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^{i}) > 0 \\ 0.4\,S(w_i \mid w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

Advanced smoothing algorithms

- Intuition used by many smoothing algorithms
  - Good-Turing
  - Kneser-Ney
  - Witten-Bell
- Use the count of things we've **seen once**
  - to help estimate the count of things we've **never seen**

Notation: N_c = Frequency of frequency c

- N_c = the count of things we've seen c times
- Sam I am I am Sam I do not eat

```
I    3
sam  2
am   2
do   1
not  1
eat  1
```

$N_1 = 3$

$N_2 = 2$

$N_3 = 1$

Good-Turing smoothing intuition

- You are fishing (a scenario from Josh Goodman), and caught:
  - 10 carp, 3 perch, 2 whitefish, 1 trout, 1 salmon, 1 eel = 18 fish
- How likely is it that next species is trout?
  - 1/18
- How likely is it that next species is new (i.e. catfish or bass)
  - Let's use our estimate of things-we-saw-once to estimate the new things.
  - 3/18 (because N_1=3)
- Assuming so, how likely is it that next species is trout?
  - Must be less than 1/18
  - How to estimate?

Good Turing calculations

$$P^*_{GT}(\text{things with zero frequency}) = \frac{N_1}{N}$$

- Unseen (bass or catfish)
  - c = 0:
  - MLE p = 0/18 = 0
  - P^*_{GT} (unseen) = N_1/N = 3/18

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

- Seen once (trout)
  - c = 1
  - MLE p = 1/18
  - C*(trout) = 2 * N2/N1
    = 2 * 1/3
    = 2/3
  - P^*_{GT}(trout) = 2/3 / 18 = 1/27

Resulting Good-Turing numbers

- Numbers from Church and Gale (1991)
- 22 million words of AP Newswire

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

Count c	Good Turing c*
0	.0000270
1	0.446
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25

Absolute Discounting Interpolation

- Save ourselves some time and just subtract 0.75 (or some d)!

discounted bigram

Interpolation weight

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w)$$

unigram

  - (Maybe keeping a couple extra values of d for counts 1 and 2)
- But should we really just use the regular unigram P(w)?

Kneser-Ney Smoothing I

- Better estimate for probabilities of lower-order unigrams!
  - Shannon game: *I can't see without my reading*_____?
  - "Francisco" is more common than "glasses"
  - … but "Francisco" always follows "San"
- The unigram is useful exactly when we haven't seen this bigram!
- Instead of P(w): "How likely is w"
- $P_{continuation}$(w): "How likely is w to appear as a novel continuation?
  - For each word, count the number of bigram types it completes
  - Every bigram type was a novel continuation the first time it was seen

glasses

Francisco

$$P_{CONTINUATION}(w) \mu \left| \{ w_{i-1} : c(w_{i-1}, w) > 0 \} \right|$$

Kneser-Ney Smoothing II

- How many times does w appear as a novel continuation:

$$P_{CONTINUATION}(w) \propto \left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|$$

- Normalized by the total number of word bigram types

$$\left|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}\right|$$

$$P_{CONTINUATION}(w) = \frac{\left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|}{\left|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}\right|}$$

Kneser-Ney Smoothing III

•Alternative metaphor: The number of # of word types seen to precede w

$$|\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

•normalized by the # of words preceding all words:

$$P_{CONTINUATION}(w) = \frac{\left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|}{\sum_{w'} \left|\{w'_{i-1} : c(w'_{i-1}, w') > 0\}\right|}$$

•A frequent word (Francisco) occurring in only one context (San) will have a low continuation probability

Kneser-Ney Smoothing IV

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i)$$

λ is a normalizing constant; the probability mass we've discounted

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} \left| \{w : c(w_{i-1}, w) > 0\} \right|$$

the normalized discount

The number of word types that can follow w_{i-1}
= # of word types we discounted
= # of times we applied normalized discount

Kneser-Ney Smoothing: Recursive formulation

$$P_{KN}(w_i \mid w_{i-n+1}^{i-1}) = \frac{\max(c_{KN}(w_{i-n+1}^{i}) - d, 0)}{c_{KN}(w_{i-n+1}^{i-1})} + \lambda(w_{i-n+1}^{i-1}) P_{KN}(w_i \mid w_{i-n+2}^{i-1})$$

$$c_{KN}(\cdot) = \begin{cases} count(\cdot) & \text{for the highest order} \\ continuationcount(\cdot) & \text{for lower order} \end{cases}$$

Continuation count = Number of unique single word contexts for •

Language Modeling

Neural Language Model

Sử dụng mô hình N-gram với N = ?

How large *n*?

- Nothing is enough (theoretically)
- But anyway: as much as possible (→ close to “perfect” model)
- Empirically: ***3***
  - parameter estimation? (reliability, data availability, storage, space, ...)
  - 4 is too much: |V|=60k → 1.296 x 1019 parameters
  - but: 6-7 would be (almost) ideal (having enough data): *in fact, one can recover original from 7-grams!*

Hạn chế của mô hình ngôn ngữ N-gram

- Khi dữ liệu thưa thì mô hình không chính xác vì các tần suất N-gram không đại diện.
- Mô hình N-gram càng chính xác khi N càng lớn, tuy nhiên khi N lớn thì số lượng N-gram rất lớn và không thực thi được do hạn chế về bộ nhớ và tính toán.
  - Không biểu diễn được phụ thuộc xa, ví dụ:

    "Hùng sống ở Pháp hồi nhỏ nên anh ấy có thể nói tiếng … khá thạo"

    "The girl that I met in the train was ..."

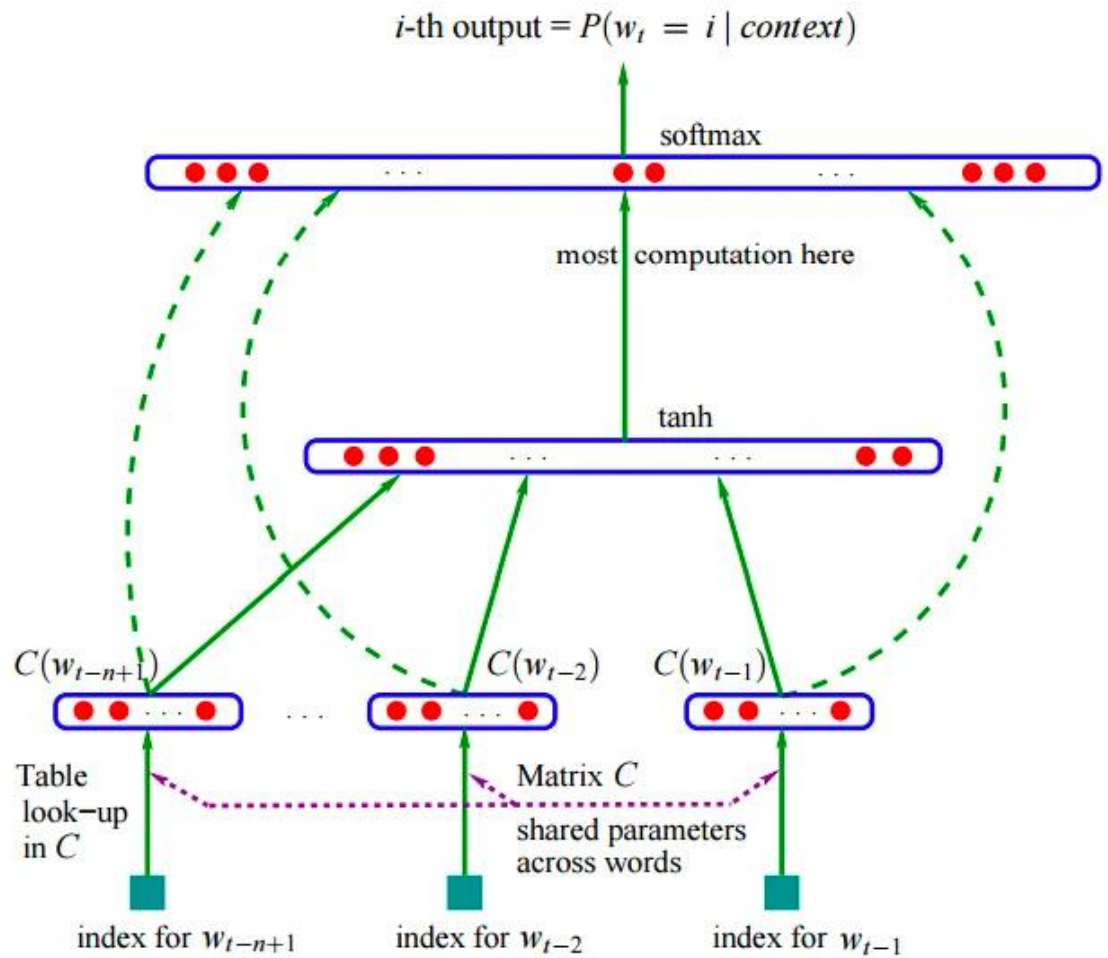
Neural Network Language Models

- Neural network language model NNLM (Bengio, 2003)
- Recurrent NNLM (Mikolov, 2010)
- Các mô hình mới: Transformer model (2018)

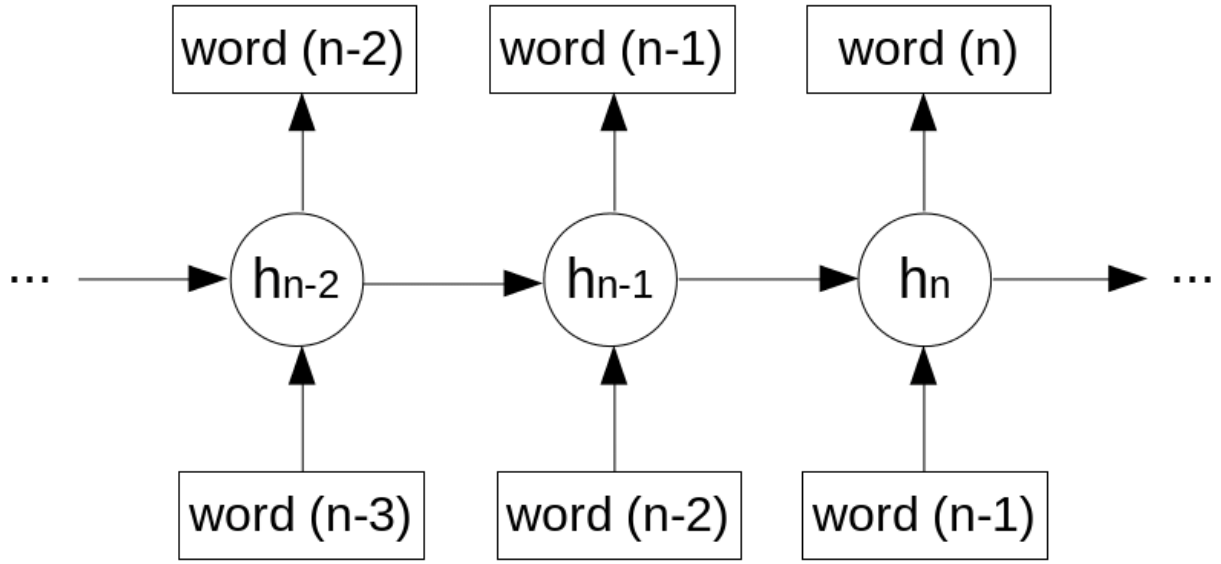
Biểu diễn từ bằng vector

- Các mô hình dựa trên NN sử dụng Word2Vec có tính tổng quát cao.

Neural network language model



Recurrent Neural Network Language Model

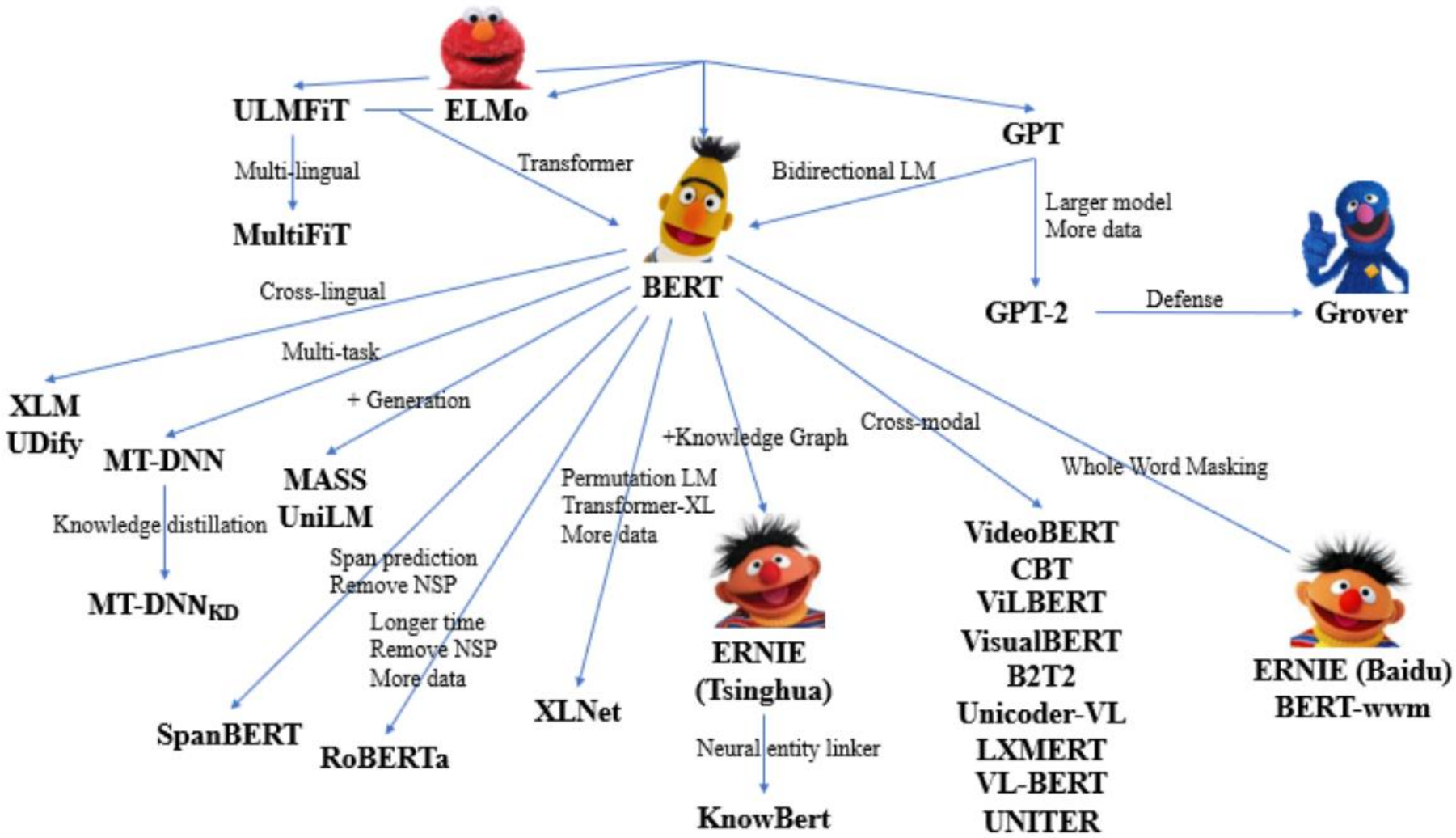


Một số kết quả so sánh

Language Model	$H(H_c)$	PPL	WER
KN5	-	248.0	12.8
RNN	200 (-)	226.2	12.0
RNN-BOW	190 (10)	**218.8**	**11.7**
RNN+KN5	200 (-)	191.6	11.8
RNN-BOW+KN5	190 (10)	**183.0**	**11.3**

RNN-BOW LM to combine short term (RNN) and long term (BOW) information (Haidar & Kurimo, 2016)

Các mô hình ngôn ngữ huấn luyện trước



PhoBERT

- PhoBERT
- Dựa trên RoBERTa huấn luyện theo thủ tục tương tự như BERT.
- Có 2 phiên bản “base” & “large”

Ứng dụng mô hình ngôn ngữ cho sửa lỗi chính tả

- Sử dụng mô hình ngôn ngữ n-gram
- Sử dụng mô hình seq2seq

Tổng kết

- Mô hình ngôn ngữ quan trọng, có nhiều ứng dung.
- Mô hình ngôn ngữ dựa vào N-gram
- Đánh giá mô hình
- Xử lý vấn đề zero và các phương pháp smoothing
- Mô hình hiện đại dựa vào mạng neural có hiệu quả hơn.