N gram report - demo N-grams

Xử lý ngôn ngữ tự nhiên (Trường Đại học Công nghệ thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh)

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

XỬ LÝ NGÔN NGỮ TỰ NHIÊN (CS221)

BÁO CÁO DEMO

N-gram Language Models

Giảng viên hướng dẫn:	TS. Nguyễn Thị Quý	
Sinh viên thực hiện:	Mai Lê Bá Vương	23521821
	Nguyễn Thanh Tùng	23521745
	Chướng Hồng Văn	23521769
	Phạm Nguyên Tường	23521751
	Tăng Hoàng Phúc	23521219
	Nguyễn Lâm Bảo Phúc	23521208

TP. Hồ Chí Minh, Tháng 9 năm 2025

Mục lục

1 Mở đầu

Ngôn ngữ tự nhiên (Natural Language) là một lĩnh vực trọng yếu trong Trí tuệ nhân tạo (AI) và Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP). Một trong những bài toán cơ bản trong NLP là *Language Modeling* – xây dựng mô hình có khả năng gán xác suất cho một câu hoặc dự đoán từ tiếp theo dựa trên ngữ cảnh.

Trong khuôn khổ báo cáo này, chúng em tìm hiểu mô hình *N-gram Language Model*, một mô hình kinh điển, đơn giản nhưng hiệu quả trong việc mô hình hóa ngôn ngữ. Ý tưởng chính là giả sử xác suất của một từ chỉ phụ thuộc vào $N-1$ từ đứng trước nó (giả thiết Markov). Ví dụ: với mô hình bigram ($N=2$), xác suất một từ w_i được tính dựa trên từ liền trước w_{i-1}:

$$P(w_i \mid w_{i-1})$$

Mặc dù các mô hình hiện đại như RNN, LSTM hay Transformer đã thay thế dần N-gram trong các ứng dụng thực tế, việc nghiên cứu N-gram vẫn có ý nghĩa quan trọng vì:

- Giúp hiểu rõ các khái niệm nền tảng trong language modeling (xác suất, likelihood, smoothing, perplexity).
- Cung cấp cơ sở để so sánh và đánh giá các mô hình hiện đại.
- Có thể áp dụng cho các hệ thống nhỏ, yêu cầu tính toán nhanh, tài nguyên hạn chế.

Trong báo cáo, chúng em sẽ trình bày các bước thực hiện:

1. Chuẩn bị và xử lý dữ liệu văn bản.
2. Xây dựng mô hình N-gram (unigram, bigram, trigram).
3. Áp dụng kỹ thuật Maximum Likelihood Estimation (MLE).
4. Thực hiện smoothing (Laplace, add-k), interpolation và backoff.
5. Đánh giá mô hình thông qua chỉ số *perplexity*.
6. Minh họa bằng việc sinh câu ngẫu nhiên từ mô hình.

Qua đó, chúng em mong muốn mang lại cái nhìn tổng quan và trực quan về cách hoạt động của N-gram Language Models, cũng như vai trò của chúng trong lịch sử phát triển của NLP.

2 Dataset

2.1 Nguồn dữ liệu

Dữ liệu sử dụng là **WikiText-2**, một tập dữ liệu văn bản tiếng Anh chuẩn thường dùng cho bài toán language modeling. Tập dữ liệu bao gồm các bài viết từ Wikipedia đã được làm sạch:

- Không chứa HTML, chỉ còn văn bản thuần.
- Giữ nguyên ngữ cảnh của các câu trong bài viết.
- Chia thành 3 bộ: huấn luyện (train), kiểm định (validation) và kiểm tra (test).

2.2 Kích thước tập dữ liệu

	Train	Validation	Test
Số dòng (line)	36,718	3,760	4,358
Số câu (sentence)	23,767	2,461	2,891

Bảng 1: Kích thước các bộ dữ liệu WikiText-2

2.3 Ví dụ dữ liệu

Một dòng mẫu trong tập huấn luyện:

```
96 ammunition packing boxes
```

Ví dụ các token từ một câu:

```
['valkyria', 'chronicles', 'iii']
```

2.4 Thống kê sơ bộ

- Kích thước từ vựng (tổng số token khác nhau trong train): 64,840
- Độ dài câu:
  - Trung bình (mean) = 85.96 từ
  - Trung vị (median) = 75 từ
  - P95 = 240 từ
- 20 token phổ biến nhất:

  `the, ,, ., of, and, in, to, a, ", was, ', -, on, s, as, that, for, with, by, )`

3 Phân tích dữ liệu (EDA)

3.1 Tổng quan về Dataset

Chúng em sử dụng dataset `WikiText-2` raw, chia thành train, validation và test.

- Số dòng: train=36,718, valid=3,760, test=4,358
- Số câu (sau khi tách token): train=23,767, valid=2,461, test=2,891
- Ví dụ về token trong một câu: `['valkyria', 'chronicles', 'iii', ...]`

3.2 Thống kê Corpus

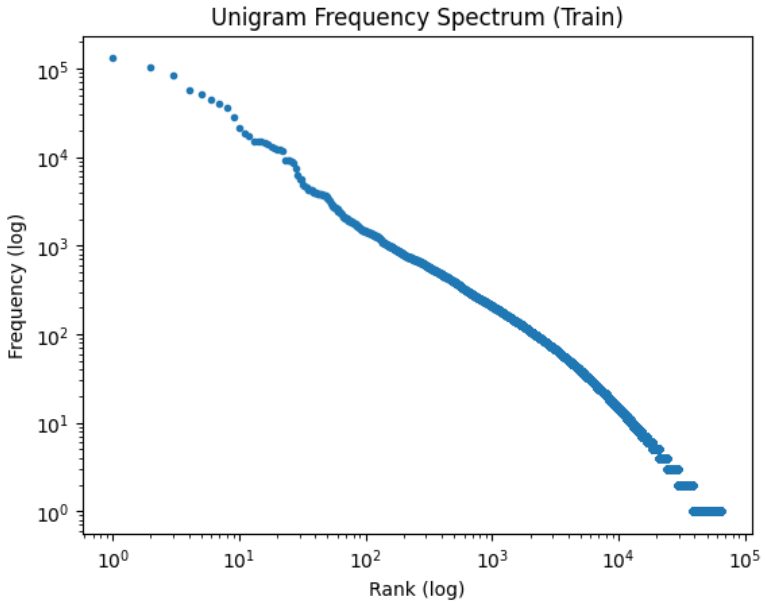
Các thống kê cơ bản trên tập train:

- Kích thước từ vựng (unigram thô): 64,840
- Độ dài câu (token): trung bình=85.96, trung vị=75, 95% percentile=240
- 20 token xuất hiện nhiều nhất:

```
[('the', 130,771), (',', 102,624), ('.', 84,291), ('of', 57,032),
 ('and', 50,738), ('in', 45,025), ('to', 39,524), ('a', 36,735),
 ('"', 28,309), ('was', 21,008), ("'", 18,646), ('-', 17,337),
 ('on', 15,156), ('s', 15,104), ('as', 15,063), ('that', 14,351),
 ('for', 13,797), ('with', 13,012), ('by', 12,718), (')', 12,004)]
```

3.3 Biểu đồ Zipf

Phân phối tần suất unigram trên tập train cho thấy xu hướng Zipf rõ rệt trên thang log-log:



Hình 1: Phổ tần suất unigram (log-log) trên tập train.

Nhận xét: Đồ thị log–log của tần suất từ theo hạng (rank) cho thấy đường cong gần như tuyến tính, phù hợp với quy luật Zipf. Một số ít từ phổ biến (the, of, and, ...) xuất hiện với tần suất rất cao, trong khi phần lớn từ còn lại xuất hiện rất ít lần.

4 Tiền xử lý dữ liệu

4.1 Tokenization

Dữ liệu được tải từ tập WikiText-2 gồm các dòng văn bản thô. Mỗi dòng được tokenize thành các token (từ, dấu câu) bằng biểu thức chính quy:

- Tách các từ, số, và các từ viết tắt như "don't", "it's".
- Giữ lại các dấu câu: .,;:!?() và dấu ngoặc kép.
- Chuyển toàn bộ thành chữ thường.

Ví dụ, câu `"Valkyria Chronicles III"` sẽ được token hóa thành:

```
['valkyria', 'chronicles', 'iii']
```

4.2 Padding câu

Để tính n-gram, mỗi câu được thêm:

- `<s>` lặp lại $(n-1)$ lần ở đầu câu (Beginning-of-Sentence)
- `</s>` ở cuối câu (End-of-Sentence)

Điều này giúp tính toán xác suất chính xác cho các n-gram bắt đầu và kết thúc câu.

4.3 Xử lý từ ngoài tập từ vựng (OOV)

- Từ vựng được xây dựng từ tập huấn luyện, giới hạn theo top V từ xuất hiện nhiều nhất (trong code là 30,000 từ).
- Những token không thuộc từ vựng được thay bằng token `<unk>`.
- Token `<s>` và `</s>` luôn được giữ lại.

4.4 Đếm n-grams

- Từ các câu đã padding, tính tần suất uni-, bi- và tri-grams.
- Tính số lượng các tiền tố (prefix) và số lượng loại từ nối tiếp khác nhau (cho Witten–Bell).

4.5 Tóm tắt thống kê sau tiền xử lý

- Tổng số câu: train = 23,767, valid = 2,461, test = 2,891
- Kích thước từ vựng (train): 64,840
- Độ dài câu: mean = 85.96, median = 75, p95 = 240
- Top 10 token phổ biến: `the, ,, ., of, and, in, to, a, ", was`

5 Mô hình và Phương pháp

5.1 Maximum Likelihood Estimation (MLE)

MLE ước lượng xác suất n-gram dựa trên tần suất xuất hiện trong tập huấn luyện:

$$P_{\text{MLE}}(w_i \mid w_{i-n+1}^{i-1}) = \frac{C(w_{i-n+1}^{i})}{C(w_{i-n+1}^{i-1})}$$

Trong đó $C(\cdot)$ là số lần xuất hiện. Nhược điểm chính: nếu một n-gram chưa từng xuất hiện, xác suất bằng 0 (zero probability).

5.2 Add-k smoothing

Để khắc phục hiện tượng zero probability, ta cộng thêm k vào mỗi tần suất:

$$P_{\text{add-k}}(w_i \mid h) = \frac{C(h, w_i) + k}{C(h) + kV}$$

Trong đó V là kích thước từ vựng. Trong thực nghiệm, chúng em thử nghiệm với $k = 1.0$ (Laplace) và $k = 0.5$.

5.3 Linear Interpolation

Linear Interpolation kết hợp tuyến tính các mô hình unigram, bigram, trigram:

$$P_{\text{interp}}(w_i \mid w_{i-2}, w_{i-1}) = \lambda_1 P(w_i) + \lambda_2 P(w_i \mid w_{i-1}) + \lambda_3 P(w_i \mid w_{i-2}, w_{i-1})$$

Trong đó $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Các hệ số λ được hiệu chỉnh trên tập validation bằng grid search.

5.4 Witten–Bell Interpolation

Witten–Bell xác định trọng số $\lambda(h)$ động dựa trên số lượng từ khác nhau từng xuất hiện sau một tiền tố h:

$$\lambda(h) = \frac{N(h)}{N(h) + T(h)}$$

Trong đó $N(h)$ là tổng số lần tiền tố h xuất hiện, $T(h)$ là số lượng từ khác nhau từng nối tiếp h. Phương pháp này giúp phân bổ hợp lý giữa n-gram dài và ngắn tùy thuộc dữ liệu quan sát.

5.5 Stupid Backoff

Stupid Backoff (SB) không tạo phân phối xác suất chuẩn, mà thay vào đó:

$$P_{SB}(w_i \mid h) = \begin{cases} \dfrac{C(h, w_i)}{C(h)} & \text{if } C(h, w_i) > 0, \\ \alpha \cdot P_{SB}(w_i \mid h') & \text{otherwise.} \end{cases}$$

Trong đó h' là tiền tố ngắn hơn, $\alpha = 0.4$ là hệ số backoff. SB có ưu điểm đơn giản, tính toán nhanh, thường dùng trong hệ thống web-scale, nhưng không thể đánh giá bằng perplexity do không tạo thành phân phối xác suất.

6 Thực nghiệm và Phân tích

6.1 Thiết lập thực nghiệm

- **Dataset**: WikiText-2, chia thành ba tập: train (36,718 dòng → 23,767 câu), validation (3,760 dòng → 2,461 câu) và test (4,358 dòng → 2,891 câu).
- **Tiền xử lý**: tokenization, padding với `<s>` và `</s>`, giới hạn từ vựng theo top 30,000 từ trong tập train; các từ ngoài từ vựng thay bằng `<unk>`.
- **Các tham số huấn luyện**:
  - Bậc n-gram: $n = 1, 2, 3$
  - Add-k smoothing: $k = 0.5$ và $k = 1.0$
  - Linear interpolation: hiệu chỉnh λ trên validation bằng grid search
  - Stupid Backoff: hệ số $\alpha = 0.4$

6.2 Kết quả perplexity (PPL)

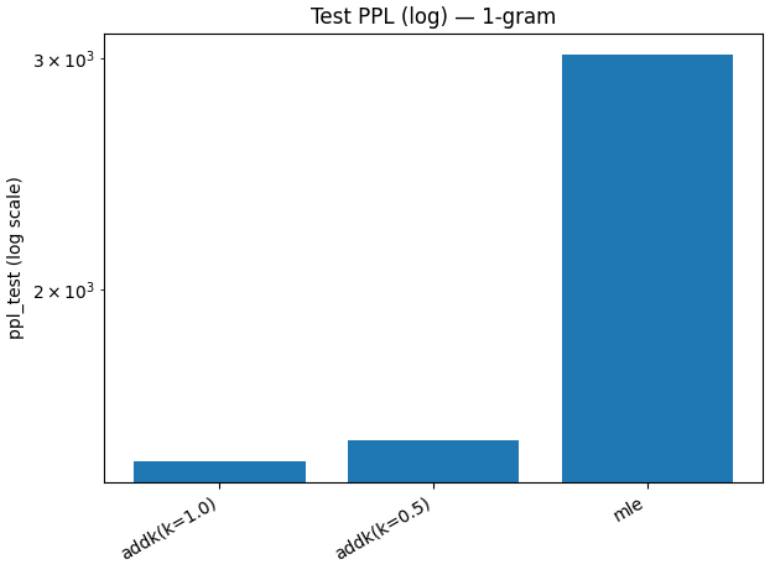
Mô hình	Phương pháp	Validation PPL	Test PPL
Unigram	MLE	2,527	3,023
Unigram	Add-k (k=1.0)	1,395	1,480
Unigram	Add-k (k=0.5)	1,437	1,535
Bigram	MLE	6.97×10^4	8.96×10^4
Bigram	Add-k (k=1.0)	2,920	3,034
Bigram	Add-k (k=0.5)	2,267	2,367
Bigram	Interpolation ($\lambda = (0.4, 0.6)$)	1,057	1,268
Trigram	MLE	2.22×10^8	2.70×10^8
Trigram	Add-k (k=1.0)	14,546	14,922
Trigram	Add-k (k=0.5)	12,227	12,592
Trigram	Interpolation ($\lambda = (0.3, 0.3, 0.4)$)	1,061	1,285
Trigram	Witten–Bell	**961**	**1,144**

Bảng 2: Perplexity các mô hình n-gram trên tập validation và test.

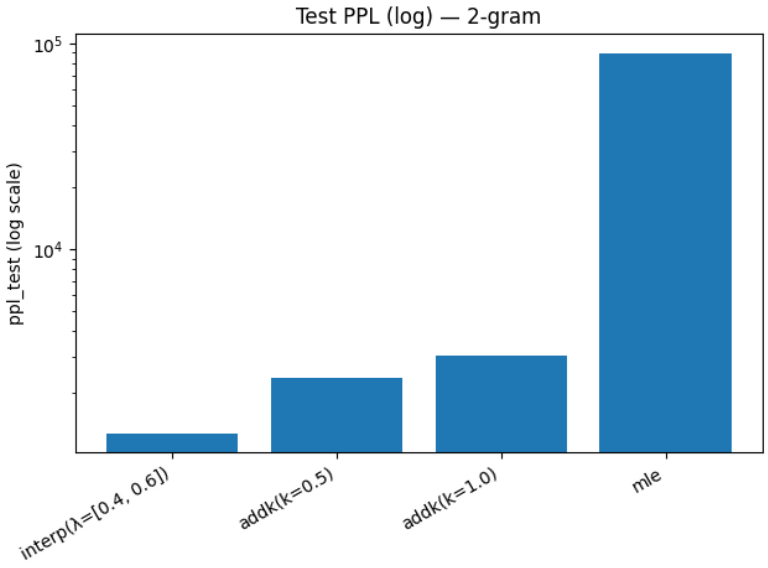
6.3 Phân tích kết quả PPL

- **MLE**: hoạt động kém với bigram và trigram do dữ liệu thưa (sparsity), dẫn tới PPL cực lớn (ví dụ trigram: 2.7×10^8).
- **Add-k smoothing**: cải thiện rõ rệt so với MLE, đặc biệt với bigram/trigram, nhưng vẫn chưa tối ưu.
- **Linear Interpolation**: kết hợp unigram/bigram/trigram giúp giảm PPL đáng kể; bigram interpolation đạt 1,268 trên test.
- **Witten–Bell**: phương pháp tốt nhất, trigram WB đạt PPL thấp nhất (961 valid, 1,144 test).

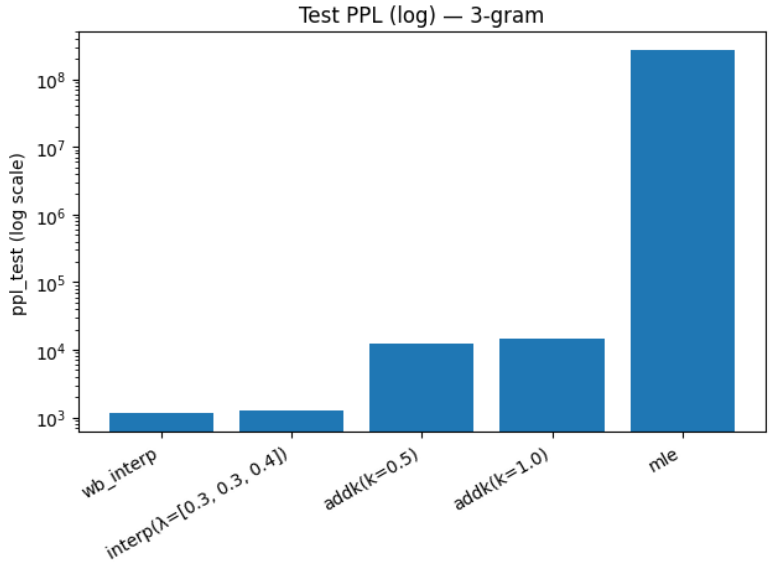
6.4 Đồ thị trực quan



Hình 2: Perplexity của các phương pháp unigram trên tập validation và test.



Hình 3: Perplexity của các phương pháp bigram trên tập validation và test.



Hình 4: Perplexity của các phương pháp trigram trên tập validation và test.

6.5 Đánh giá Stupid Backoff

Mặc dù không thể đo perplexity, SB được đánh giá qua:

- **Ranking accuracy**: 0.798 (trên 500 cặp câu thật và câu nhiễu).
- **Next-word prediction**: Hit@1 = 5.8%, Hit@5 = 22.7%, Hit@10 = 33.2%, MRR = 0.14.

→ SB có độ chính xác vừa phải nhưng ưu điểm là tốc độ tính toán nhanh và mở rộng tốt.

6.6 Ví dụ sinh câu

6.6.1 SB bigram

```
- after the song was originally have appeared in the <unk> , a <unk> , he scored the end the <unk
- in the <unk> for his team " </s>
- release , with no other - jamal in the and <unk> , the <unk> . it is the of the state in <unk>
- the same program called it to find work , and <unk> . </s>
- ) for the series , though they are also noted that a week of the ) and the <unk> , in september
```

6.6.2 Trigram Interpolation

```
- all and other the had the church of the the of the <unk> they are the top. 7 the <unk>, the nam
- track and field marshal of <unk>( both on the the <unk> that all on the storm caused the of som
```

```
- hurricane a ". he was later, but are
- the the be and of the <unk>( " it also was to the american political and the <unk> s <unk> with
- in a <unk> from his. and had of the.
```

6.6.3 Trigram Witten–Bell

```
- history
- after the game were designed the sewers. to aid, but only for any he had a lot together, drawin
- the " as " a sense of depth, and the united kingdom, world tour finals to <unk>. the episode al
- the missouri steamboat captains in response, though the film after reading most expensive priva
- the game as two moody national bank minted for best video intelligence unit deployment program
```

6.7 Tổng kết

- Witten–Bell cho kết quả tốt nhất về PPL và chất lượng câu sinh.
- Add-k và Interpolation giúp giảm PPL nhưng chưa bằng Witten–Bell.
- Stupid Backoff phù hợp cho bối cảnh dữ liệu lớn nhờ tốc độ, dù chất lượng dự đoán kém hơn.

7 Kết luận

Trong báo cáo này, chúng em đã:

- Xây dựng và triển khai các mô hình n-gram (unigram, bigram, trigram) trên tập dữ liệu WikiText-2.
- So sánh nhiều phương pháp ước lượng xác suất: MLE, Add-k smoothing, Linear Interpolation, Witten–Bell và Stupid Backoff.
- Đánh giá mô hình thông qua chỉ số perplexity và các thước đo phụ (ranking accuracy, next-word prediction).

Kết quả thực nghiệm cho thấy:

- MLE hoạt động kém với bigram/trigram do dữ liệu thưa.
- Add-k và Linear Interpolation cải thiện đáng kể perplexity.
- Witten–Bell đạt kết quả tốt nhất (PPL $\approx$ 961 trên validation và 1,144 trên test).
- Stupid Backoff không tính được perplexity nhưng đạt độ chính xác 79.8% trong phân biệt câu thật/câu nhiễu, và có ưu điểm về tốc độ.

8 Hướng mở rộng

Trong tương lai, có thể mở rộng nghiên cứu theo các hướng sau:

- **Nâng cao tiền xử lý**: áp dụng stemming/lemmatization hoặc loại bỏ stopwords để cải thiện chất lượng n-gram.
- **Từ vựng động**: sử dụng kỹ thuật subword (BPE, unigram LM) để xử lý tốt hơn các từ hiếm.
- **So sánh với mô hình neural**: triển khai RNN, LSTM hoặc Transformer language model để so sánh trực tiếp với n-gram.
- **Ứng dụng thực tế**: tích hợp mô hình vào hệ thống gợi ý từ, kiểm tra chính tả hoặc sinh văn bản tự động.

9 Tài liệu tham khảo

1. Jurafsky, D., & Martin, J. H. (2020). *Speech and Language Processing (3rd Edition)*. Draft online version. Available: https://web.stanford.edu/~jurafsky/slp3/

2. Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

3. Chen, S. F., & Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics (ACL '96)*.

4. Kneser, R., & Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*.

5. Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *INTERSPEECH 2010*.

6. Chen, X., & Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4), 359-394.

7. Goodman, J. (2001). Classes for fast maximum entropy training. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*.