

Zero-shot

☑ Trắc nghiệm 11 ↗ Chạy trên Kaggle ⬇ PDF bài

Zero-shot là dùng model cho một nhiệm vụ mà không huấn luyện thêm trên nhãn của nhiệm vụ đó. CLIP là model học ghép ảnh với câu mô tả; muốn phân loại một ảnh, ta so ảnh với các câu như “ảnh một cái bàn”, “ảnh một cái bảng” rồi chọn câu gần nhất. Với model ngôn ngữ, ta chọn đáp án bằng cách so xác suất model gán cho từng cách viết tiếp. Bài này học cả hai cách, và vì sao điểm cao nhất chưa chắc là đáp án đúng.

MỤC TIÊU

- Phân biệt zero-shot, few-shot và fine-tuning.
- Phân loại ảnh bằng CLIP.
- Tính log-probability và perplexity của một câu.
- Biết điểm cao nhất chưa chắc là đáp án đúng; đo bằng validation.

Zero-shot chuyển điều gì từ việc học trước?

Trong supervised classification thông thường, ta học các nhãn như 0, 1, 2 từ nhiều input đã gán nhãn. Zero-shot dùng kiến thức học trước để xử lý một nhiệm vụ mới mà không cập nhật trọng số từ ví dụ có nhãn của nhiệm vụ đó. Model vẫn đã học rất nhiều dữ liệu trước đây. Nếu dữ liệu mới giống dữ liệu học trước, nó có thể làm tốt; nếu khác ngôn ngữ, khác kiểu ảnh hoặc có khái niệm chưa học, kết quả có thể kém.

Giữa zero-shot và fine-tuning còn few-shot: đặt vài ví dụ có nhãn ngay trong prompt của model ngôn ngữ rồi để model hoàn thành ví dụ mới (in-context learning). Trọng số vẫn giữ nguyên; các ví dụ chỉ là phần ngữ cảnh được đọc lúc dự đoán.

Vì đã dùng nhãn của nhiệm vụ, kết quả few-shot không được báo là zero-shot. Ngược lại, prompt chỉ có câu hỏi và lời dặn “hãy suy nghĩ từng bước”, không kèm ví dụ nào, vẫn là zero-shot. Fine-tuning (bài 4.3) thì dùng chính các ví dụ ấy để cập nhật trọng số.



Zero-shot không dùng ví dụ, few-shot đặt ví dụ vào prompt, fine-tuning học từ ví dụ.

CLIP có một image encoder và một text encoder được học để đưa ảnh cùng mô tả phù hợp đến gần nhau. Trong một batch cặp ảnh-chữ, ma trận tương đồng có các cặp đúng trên đường chéo. Mục tiêu contrastive khuyến khích đường chéo cao hơn các cặp không khớp, theo cả chiều ảnh tìm chữ và chữ tìm ảnh. Mọi ô ngoài đường chéo đều được coi là cặp không khớp, dù đôi khi hai ảnh trong batch có nội dung rất gần nhau; đó là giới hạn của cách dựng tín hiệu học.



mỗi hàng: ảnh chọn đúng caption · mỗi cột: caption chọn đúng ảnh

Pretraining kéo điểm cặp ảnh-caption đúng lên, đẩy mọi cặp khác xuống.

Khi phân loại nhãn mới, ta không dùng một head học từ ảnh của bài mới. Ta tạo một mô tả cho mỗi lớp, lấy text embedding, rồi so image embedding với các vector lớp. Hai encoder phải thuộc cùng model được học để chia sẻ không gian; lấy vector của một

model ảnh và một model chữ tùy ý rồi nhân vô hướng không có cơ sở chỉ vì cùng số chiều.

HỎI NHANH

Một batch có hai ảnh mèo khác nhau, và cả hai đều đi với caption “a photo of a cat”. Trong contrastive loss, ô (ảnh mèo 1, caption của ảnh mèo 2) được xử lý thế nào?

- A Được coi là cặp đúng thứ hai của ảnh mèo 1
- B Bị bỏ khỏi loss, vì hai caption trùng nhau
- C Vẫn là một cặp không khớp trong loss
- D Không ảnh hưởng gì, vì loss chỉ đọc đường chéo

HỎI NHANH

Một đội lấy embedding ảnh 512 chiều từ một ResNet đã học trên ImageNet, và embedding 512 chiều của tên lớp từ một text encoder khác, chuẩn hóa cả hai, rồi gán nhãn bằng cosine. Cách làm này thế nào?

- A Dừng được, vì cùng số chiều và đã chuẩn hóa
- B Không có cơ sở, vì hai encoder chưa học chung một không gian
- C Dừng được, nếu nhân thêm logit scale của CLIP
- D Dừng được, nếu prompt mô tả đủ chi tiết

Từ tên lớp đến một classifier

Giả sử $\mathbf{u}$ là embedding ảnh và $\mathbf{v}_k$ là embedding mô tả lớp k , đều chuẩn hóa độ dài 1. Điểm lớp là

$$\ell_k = s \mathbf{u}^\top \mathbf{v}_k, \quad \hat{y} = \arg \max_k \ell_k.$$

$s > 0$ là logit scale; có thể viết $s = 1/\tau$ với τ gọi là temperature. Trong CLIP, τ được học cùng hai encoder: bài báo gốc khởi tạo $\tau = 0.07$ và chặn để s không vượt 100. Gọi logits API của model là dùng đúng scale đã học; ví dụ ở mục sau chọn $s = 5$ để số dễ tính tay.

Scale dương không đổi argmax của một tập embedding cố định, nhưng đổi mức tập trung của softmax: s lớn làm phân phối nhọn, s nhỏ làm nó phẳng. Với $s = 100$, cosine 0.30, 0.28, 0.10 thành logits 30, 28, 10, nên lớp đầu nhận khoảng $1/(1 + e^{-2}) \approx 0.88$: chênh 0.02 cosine đã thành chênh lớn về xác suất.

Với B ảnh, K lớp và embedding D chiều, U có shape (B, D) , V có shape (K, D) , còn $U @ V.T$ có shape (B, K) . Mỗi hàng là một ảnh, mỗi cột là một lớp. Đây là phép nhân ma trận quen thuộc từ linear probe, nhưng vector của lớp đến từ text encoder thay vì được học từ nhãn của bài mới. Cùng ý tưởng cho phép phát hiện vật thể theo tên lớp chưa từng có nhãn hộp: so embedding của từng vùng ảnh với text embedding của tên lớp (open-vocabulary detection).

HỎI NHANH

U có shape $(8, 512)$, V có shape $(3, 512)$. Điểm của tám ảnh với ba lớp có shape gì?

A $(8, 3)$

B $(3, 8)$

C $(8, 512)$

D $(512, 512)$

HỎI NHANH

Nhân mọi logits với một scale dương lớn hơn, giữ nguyên embedding và tập lớp. Thứ gì chắc chắn giữ nguyên?

- A Mọi xác suất softmax
- B Cross-entropy của mọi mẫu
- C Khoảng cách giữa hai logit lớn nhất
- D Chỉ số của logit lớn nhất

Ba ảnh, ba lớp bằng NumPy

Ta dùng embedding hai chiều tự đặt để nhìn rõ hình học; embedding của CLIP có hàng trăm chiều nhưng phép tính giống hệt. Ảnh có vector (0.8, 0.6), hai mô tả lớp có vector (1, 0) và (0, 1). Cả ba đều dài 1. Cosine lần lượt là 0.8 và 0.6. Với scale 5, logits là 4 và 3; softmax trên hai lớp cho lớp đầu $e^4/(e^4 + e^3) = 1/(1 + e^{-1}) \approx 0.7311$. Code dưới đây thêm ảnh thứ hai, ảnh thứ ba và lớp thứ ba để có đủ một ma trận.

Python

```
import numpy as np

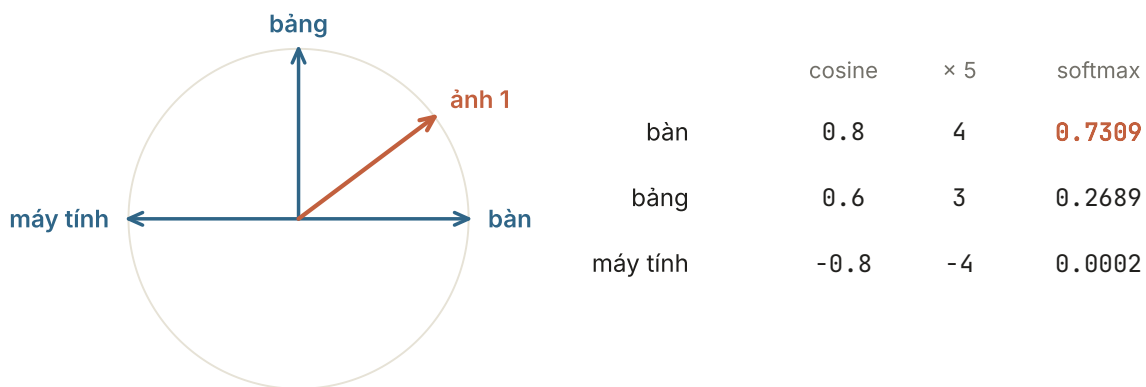
def normalize(matrix):
    norm = np.linalg.norm(matrix, axis=1, keepdims=True)
    return matrix / np.maximum(norm, 1e-12)

def softmax(matrix):
    shifted = matrix - matrix.max(axis=1, keepdims=True)
    exp_values = np.exp(shifted)
    return exp_values / exp_values.sum(axis=1, keepdims=True)

images = normalize(np.array([[.8, .6], [.6, .8], [-1., 0.])))
texts = normalize(np.array([[1., 0.], [0., 1.], [-1., 0.])))
class_names = ["ban", "bang", "may_tinh"]
scale = 5.0
```

```
scores = images @ texts.T
logits = scale * scores
probabilities = softmax(logits)
print("shape logits:", logits.shape)
print("cosine:", scores.round(2))
print("nhân:", [class_names[k] for k in logits.argmax(axis=1)])
print("softmax hàng đầu:", probabilities[0].round(4).tolist())
print("chi hai lop:", round(float(softmax(logits[:1, :2])[0, 0]), 4))
```

Hàng đầu có thêm lớp thứ ba với cosine -0.8, nên xác suất của lớp đầu trên ba lớp (0.7309) hơi thấp hơn con số tính tay trên hai lớp (0.7311). Ba nhãn in ra khớp với hướng gần nhất của từng ảnh: argmax đang làm đúng phép hình học đã đặt. Accuracy chỉ có nghĩa khi có ảnh thật và nhãn thật; script ở cuối bài chấm nó trên validation.



Cosine với ba lớp là 0.8, 0.6, -0.8; qua softmax, “bàn” được 0.73.

HỎI NHANH

Quên chuẩn hóa texts, và vector của lớp thứ hai là (0, 3) thay vì (0, 1). Ảnh (0.8, 0.6) được gán vào lớp nào?

- A Lớp thứ hai, vì tích vô hướng thành 1.8
- B Lớp thứ hai, vì cosine thành 1.8
- C Lớp đầu, vì cosine với lớp đầu vẫn là 0.8
- D Lớp đầu, vì nhân một hằng số không đổi argmax

Ta kiểm phép nhân bằng vòng lặp, rồi tính contrastive loss của ba cặp trên đường chéo, đúng đại lượng mà pretraining CLIP tối ưu trên mỗi batch: hàng i (ảnh) phải chọn cột i (mô tả) và ngược lại.

Python

```
reference = np.zeros_like(scores)
for i in range(len(images)):
    for k in range(len(texts)):
        reference[i, k] = sum(images[i, j] * texts[k, j] for j in range(2))
image_to_text = -np.log(np.diag(softmax(logits))).mean()
text_to_image = -np.log(np.diag(softmax(logits.T))).mean()
contrastive_loss = (image_to_text + text_to_image) / 2
print("dot khop vong lap:", np.allclose(scores, reference))
print("image-to-text loss:", round(float(image_to_text), 4))
print("text-to-image loss:", round(float(text_to_image), 4))
print("loss trung binh:", round(float(contrastive_loss), 4))
```

Dòng True xác nhận `images @ texts.T` đúng là tích vô hướng từng cặp (hàng ảnh, cột lớp). Hai loss dùng cùng logits nhưng softmax theo hai chiều khác nhau: theo hàng là ảnh tìm chữ, theo cột là chữ tìm ảnh. Chúng thường không bằng nhau vì phân bố trên hàng và trên cột khác nhau. Loss pretraining nhỏ cũng chưa nói accuracy trên nhãn của bài mới cao: hai đại lượng đo hai việc khác nhau.

softmax theo hàng: ảnh tìm chữ				softmax theo cột: chữ tìm ảnh			
	bàn	bảng	máy tính		bàn	bảng	máy tính
ảnh 1	0.731	0.269	0.000		0.731	0.265	0.000
ảnh 2	0.269	0.731	0.001		0.269	0.721	0.000
ảnh 3	0.000	0.007	0.993		0.000	0.013	1.000
loss 0.2114				loss 0.2135			

loss = trung bình $-\log$ của ba ô đường chéo

Softmax theo hàng (ảnh tìm chữ) và theo cột (chữ tìm ảnh) cho hai loss khác nhau.

Tên lớp, prompt và tập ứng viên có ảnh hưởng

Text encoder đọc cả chuỗi, không chỉ một số nguyên biểu thị lớp. “A photo of a desk” và “desk” có thể cho vector khác nhau. Prompt nên mô tả đúng loại input và phân biệt được các lớp. Một từ có nhiều nghĩa, hoặc một nhãn quá ngắn, có thể không biểu đạt mục tiêu. Với model chủ yếu học tiếng Anh, phải kiểm riêng ngôn ngữ muốn dùng; không giả định chất lượng tiếng Việt bằng tiếng Anh.

Có thể dùng nhiều câu mẫu (template) cho mỗi lớp, chuẩn hóa từng text embedding, trung bình rồi chuẩn hóa vector lớp lần nữa, để bớt phụ thuộc vào một cách diễn đạt.

Nếu chọn template theo điểm validation có nhãn, ta đã dùng nhãn nhiệm vụ để chọn phương pháp, dù không train trọng số; phải khai báo điều ấy nếu benchmark yêu cầu zero-shot theo nghĩa nghiêm ngặt.

HỎI NHANH

Một đội thử 20 template, giữ template có accuracy cao nhất trên 100 ảnh validation có nhãn, rồi báo chính con số ấy là “accuracy zero-shot”. Báo như vậy có ổn không?

- A Ổn, vì không trọng số nào được cập nhật
- B Ổn, nếu cả 20 template đều do người viết tay
- C Không ổn, con số quá lạc quan, vì nhãn đã được dùng để chọn template
- D Không ổn, vì zero-shot không được dùng nhiều template

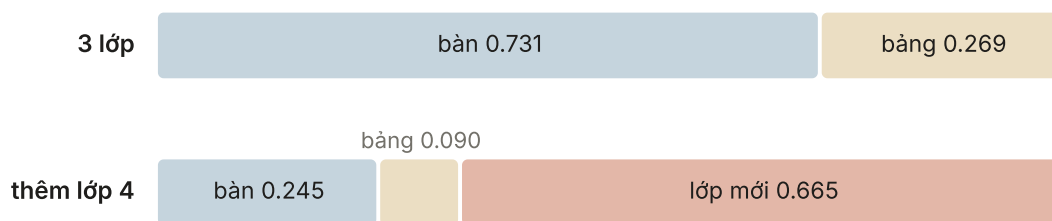
Softmax chỉ chuẩn hóa trên các lớp được đưa vào. Nó không cho biết ảnh có thuộc một trong các lớp đó hay không. Ta thêm một lớp thứ tư có vector trùng hướng với ảnh đầu, giữ nguyên ảnh và ba lớp cũ.

Python

```
new_texts = np.vstack([texts, normalize(np.array([[.8, .6]]))])
new_logits = scale * images[:1] @ new_texts.T
```

```
new_probabilities = softmax(new_logits)
print("xac suat lop cu truoc:", round(float(probabilities[0, 0]), 4))
print("xac suat lop cu sau:", round(float(new_probabilities[0, 0]), 4))
print("chi so thang sau:", int(new_logits.argmax(axis=1)[0]))
```

Ảnh và vector lớp cũ không thay đổi nhưng xác suất lớp cũ giảm, vì mẫu số có thêm ứng viên. Vì vậy một giá trị như 0.9 không mặc nhiên là độ tin cậy đã hiệu chuẩn cho thế giới mở. Nếu cần lớp “khác”, phải có quy trình và validation kiểm ngưỡng hoặc phương pháp phát hiện ngoài miền; thêm từ “other” không tự giải quyết vấn đề.



ảnh và ba vector cũ không đổi; chỉ mẫu số có thêm một ứng viên

Thêm một lớp gần ảnh hơn, xác suất của “bàn” tụt từ 0.73 xuống 0.25.

HỎI NHANH

Ảnh chụp một cái ghế, nhưng tập lớp chỉ có bàn, bảng và máy tính. Softmax cho “bàn” 0.92. Hiểu con số này thế nào?

- A Bàn là lớp khớp nhất trong ba lớp đã cho
- B Ảnh có 92% khả năng là một cái bàn
- C Ảnh là cái bàn, vì 0.92 vượt ngưỡng 0.5
- D Model hỏng, vì ảnh lạ thì softmax phải nhỏ

Dùng model ngôn ngữ để chấm cách hoàn thành

Một causal language model dự đoán token tiếp theo từ các token trước. Cho prompt c và một ứng viên gồm m token $t_1, \dots, t_m$, xác suất chuỗi phân tích thành tích các xác suất có điều kiện. Tích nhiều số nhỏ dễ underflow nên dùng log:

$$\log p(t_1, \dots, t_m \mid c) = \sum_{j=1}^m \log p(t_j \mid c, t_{<j}).$$

Ta có thể xếp hạng cách hoàn thành câu bằng tổng này. Nhưng chuỗi dài thường tích thêm nhiều số không quá 1 nên bị phạt độ dài. Điểm trung bình mỗi token là tổng chia m , đổi câu hỏi thành độ hợp lý trung bình mỗi token. Hai tiêu chí có thể xếp hạng khác nhau; không có lựa chọn đúng cho mọi nhiệm vụ. Phải dùng đúng giao thức bài và validation để chọn.

Khi dùng nhãn để phân loại, giữ cùng prompt, quy tắc khoảng trắng, tokenizer và phần token cần chấm. Chỉ chấm token của ứng viên; nếu tính cả prompt, một prompt dài dễ lấn át điểm. Tên lớp có số token và tần suất khác nhau nên độ phổ biến của từ cũng ảnh hưởng điểm, không chỉ sự liên quan đến input.

Một cách kiểm là chấm các nhãn với một input trung tính, ví dụ chuỗi rỗng hoặc “N/A”: nhãn nào đã thắng sẵn khi chưa có nội dung thì điểm đang mang độ lệch của nhãn. Trừ điểm nền này là một hiệu chỉnh thường dùng, nhưng hiệu quả của nó vẫn phải được validation xác nhận.

HỎI NHANH

Hai ứng viên dùng chung một prompt 40 token. Một bạn lỡ cộng log-prob của 40 token prompt vào điểm của mỗi ứng viên. Thứ hạng theo tổng log và theo trung bình log có thể đổi không?

- A Theo tổng có thể đổi, theo trung bình thì không
- B Theo tổng không đổi, theo trung bình thì có thể đổi
- C Cả hai đều không đổi
- D Cả hai đều có thể đổi

Tổng log hay trung bình log

Hai đường token giả định có xác suất có điều kiện A: (0.6, 0.8) và B: (0.4, 0.95, 0.95). Ở bước đầu chỉ có hai token khả dĩ nên $0.6 + 0.4 = 1$; các số sau là xác suất có điều kiện trong ngữ cảnh riêng của từng đường. Bảng tự đặt này cho phép tính tay toàn bộ; với model thật, mỗi số được đọc từ softmax của logits tại vị trí tương ứng.

Xác suất chuỗi A là 0.48, chuỗi B là 0.361, nên tổng log chọn A. Trung bình log chia 2 hoặc 3 có thể chọn B: chuỗi B dài hơn nhưng các bước sau rất dễ đoán.

Python

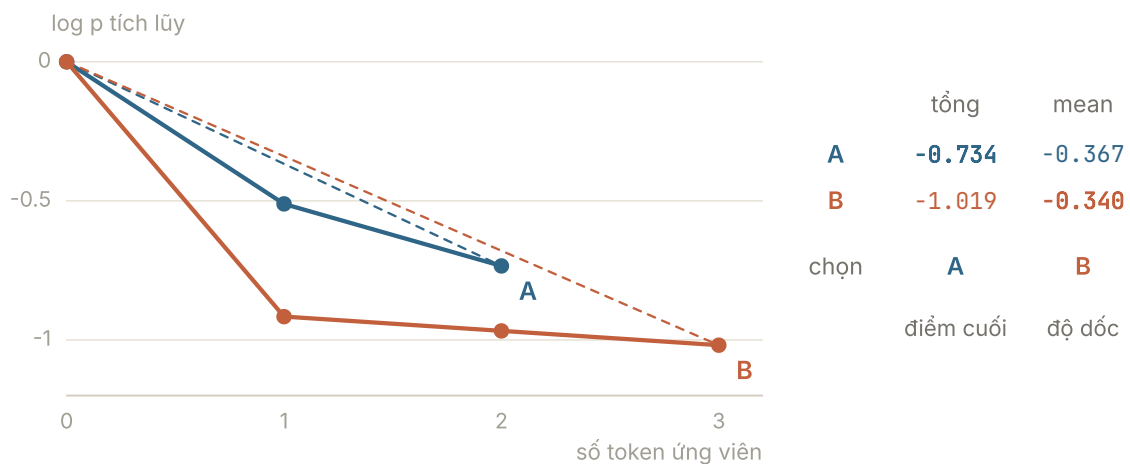
```
conditional = {
    "A": np.array([0.6, 0.8]),
    "B": np.array([0.4, 0.95, 0.95]),
}
sum_scores, mean_scores = {}, {}
for name, token_probabilities in conditional.items():
    sum_log = np.log(token_probabilities).sum()
    mean_log = sum_log / len(token_probabilities)
    perplexity = np.exp(-mean_log)
    sum_scores[name], mean_scores[name] = sum_log, mean_log
    print(name, "prob =", round(float(token_probabilities.prod()), 4),
          "sum log =", round(float(sum_log), 4),
```

```

    "mean log =", round(float(mean_log), 4),
    "PPL =", round(float(perplexity), 4))
print("tong log chon:", max(sum_scores, key=sum_scores.get))
print("mean log chon:", max(mean_scores, key=mean_scores.get))

```

Nhìn hai dòng chọn cuối để thấy đổi cách chuẩn hóa là đổi thứ hạng. Perplexity (PPL) bằng $\exp(-\text{mean log})$; càng thấp thì các token càng dễ được model đoán theo ngữ cảnh. Nó không đo tính đúng của lời giải thuật toán. Một câu sai nhưng quen thuộc có thể có xác suất cao; muốn kiểm thuật toán vẫn cần lập luận hoặc chạy test.



Tổng log-prob chọn A; trung bình log-prob chọn B.

HỎI NHANH

Nối thêm vào cuối B một token có xác suất có điều kiện 0.99. Tổng log và trung bình log của B thay đổi thế nào?

- A Tổng tăng, trung bình tăng
- B Tổng tăng, trung bình giảm
- C Tổng giảm, trung bình tăng
- D Tổng giảm, trung bình giảm

Perplexity cần cùng giao thức

Nếu mỗi token có xác suất 0.5, mean negative log bằng $-\log 0.5$, nên perplexity bằng 2 dù có hai hay ba token. Ta đối chiếu với căn bậc m của nghịch đảo xác suất chuỗi:

Python

```
for length in [2, 3]:
    token_probabilities = np.full(length, 0.5)
    from_log = np.exp(-np.log(token_probabilities).mean())
    from_product = (1 / token_probabilities.prod()) ** (1 / length)
    print("length:", length, "PPL:", round(float(from_log), 4),
          "khop:", np.isclose(from_log, from_product))
```

Tokenization đổi đơn vị đo: cùng một câu, model X tách ba token còn model Y tách sáu token, nên mean log và PPL của hai model chia cho hai mẫu số khác nhau. Vì vậy model A từ vựng 32 000 có PPL 8 và model B từ vựng 128 000 có PPL 12 chưa nói được ai tốt hơn.

Cửa sổ ngữ cảnh cũng ảnh hưởng điểm; cắt chuỗi thành các đoạn rồi làm nhiều token mất ngữ cảnh. Script của chương chấm ứng viên ngắn nằm gọn trong context; PPL của một tài liệu dài cần cửa sổ trượt, như hướng dẫn trong mục Đọc thêm. BERT/DistilBERT kiểu masked language model không có cùng phân tích causal trên; không lấy loss MLM rồi gọi đó là PPL causal theo công thức này.

model X	quy hoạch động			mean log -6 / 3	PPL e ² ≈ 7.39			
model Y	qu	y	ho	ạch	độ	ng	mean log -6 / 6	PPL e ≈ 2.72

cùng một câu, cùng tổng log -6: chỉ mẫu số khác nhau

Cùng tổng log-prob, tách 3 hay 6 token cho perplexity khác nhau.

HỎI NHANH

Cùng một câu, model X tách thành 3 token, model Y tách thành 6 token, và cả hai cho tổng log-probability của câu bằng -6 . PPL theo token của hai model là bao nhiêu?

- A Cả hai khoảng 7.39, vì cùng tổng log
- B X khoảng 2.72, Y khoảng 7.39, và câu có cùng xác suất ở hai model
- C X khoảng 7.39, Y khoảng 2.72, nên Y đoán câu này tốt hơn
- D X khoảng 7.39, Y khoảng 2.72, và câu có cùng xác suất ở hai model

Chạy với CLIP và GPT-2 thật

Script của chương đưa hai phép tính trên vào checkpoint thật. Cả hai nhánh của `zero_shot.py` nạp model đã tải trước theo một revision cố định, kiểm SHA-256 từng tệp, chạy offline và ghi môi trường cùng kết quả vào `RESULT.json`.

- **CLIP** (`--mode clip`) cần một JSON ánh xạ mỗi lớp tới một prompt, ví dụ `"desk": "a photo of a desk"`, và một CSV `id,image,label`. Script đưa ảnh và prompt qua processor của checkpoint, lấy `logits_per_image`, tức đã dùng scale học sẵn thay cho scale 5 ở trên, rồi ghi dự đoán theo `id`. Có cột nhãn thì kết quả ghi accuracy và macro-F1; không có nhãn thì phần đánh giá để trống.
- **GPT-2** (`--mode gpt2`) nhận một JSON gồm prompt và danh sách candidates, mỗi ứng viên ghi cả khoảng trắng đầu như `" dynamic programming"`. Script tokenize prompt và ứng viên riêng rồi ghép ID, dịch logits một vị trí, chỉ chấm token ứng viên, và ghi log-prob từng token, tổng, mean cùng PPL của phần ứng viên. Nó còn đối chiếu mean negative log với loss mà model tự tính khi mọi nhãn của prompt bị che bằng `-100`: hai cách tính phải khớp.

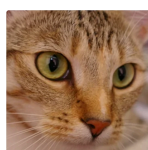
Kết quả đọc như ở các mục trên. Dự đoán CLIP là argmax theo hàng của ma trận (ảnh, lớp); kết quả GPT-2 ghi cả lựa chọn theo tổng lẫn theo mean, và tiêu chí phải được chọn

trước khi nhìn nhãn. Accuracy trên một validation nhỏ là ước lượng có nhiễu; PPL của ứng viên không đo việc thuật toán trong ứng viên đúng hay sai.

HỎI NHANH

Prompt là "The algorithm uses", nhưng ứng viên được ghi là "dynamic programming", thiếu khoảng trắng ở đầu. Script tokenize prompt và ứng viên riêng rồi ghép ID lại. Chuyện gì xảy ra?

- A Không sao, tokenizer tự chèn khoảng trắng giữa hai phần
- B Model chấm một chuỗi dính liền "usesdynamic", nên điểm thấp oan
- C Script dừng lại, vì ứng viên vượt quá context
- D Chỉ trung bình log đối, còn tổng log giữ nguyên



a photo of a cat 0.99
a photo of a dog 0.01
a photo of a cup of coffee 0.00
a photo of a rocket 0.00
a photo of a temple 0.00
a photo of a flower 0.00
a photo of an astronaut 0.00
a photo of a car 0.00

ảnh 1: chọn "cat", cosine 0.285, softmax 0.99



a photo of a cat 0.00
a photo of a dog 0.00
a photo of a cup of coffee 1.00
a photo of a rocket 0.00
a photo of a temple 0.00
a photo of a flower 0.00
a photo of an astronaut 0.00
a photo of a car 0.00

ảnh 2: chọn "cup of coffee", cosine 0.302, softmax 1.00



a photo of a cat 0.00
a photo of a dog 0.00
a photo of a cup of coffee 0.00
a photo of a rocket 0.00
a photo of a temple 1.00
a photo of a flower 0.00
a photo of an astronaut 0.00
a photo of a car 0.00

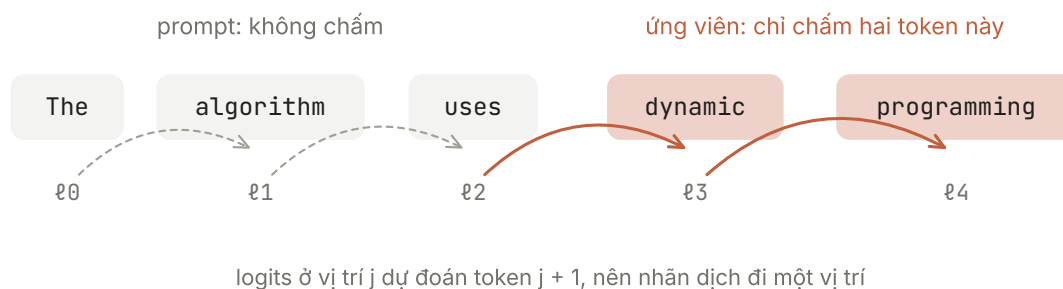
ảnh 4: chọn "temple", cosine 0.257, softmax 1.00



a photo of a cat 0.00
a photo of a dog 0.00
a photo of a cup of coffee 0.00
a photo of a rocket 0.00
a photo of a temple 0.00
a photo of a flower 0.00
a photo of an astronaut 1.00
a photo of a car 0.00

ảnh 6: chọn "astronaut", cosine 0.268, softmax 1.00

CLIP chọn câu "a photo of a ..." có cosine cao nhất với ảnh; không cần train.



Logits ở vị trí j dự đoán token j + 1; chỉ cộng log-prob của ứng viên.

Khi đi thi

- Đề cho sẵn CLIP hay language model thì chấm zero-shot trên validation trước khi train gì.
- Prompt, cột logits và tên lớp trong file nộp phải lấy từ cùng một danh sách lớp.
- Chấm chuỗi bằng một giao thức cố định; chọn tổng hay trung bình log bằng validation.

Tóm tắt

- Zero-shot không dùng ví dụ có nhãn của nhiệm vụ; few-shot đặt vài ví dụ vào prompt; fine-tuning cập nhật trọng số.
- CLIP so embedding ảnh với embedding câu mô tả từng lớp; softmax phụ thuộc tập lớp đưa vào và scale.
- Model ngôn ngữ chấm ứng viên bằng tổng log-probability của các token ứng viên; chia cho độ dài thì đổi tiêu chí xếp hạng.
- Perplexity chỉ so được trong cùng tokenizer, và không đo lời giải đúng hay sai.
- Chất lượng thật của một cách zero-shot chỉ biết qua validation có nhãn.

Kiểm tra cuối bài

CÂU 1

Một bạn nhận xét: “GPT-2 chấm nhãn cảm xúc theo kiểu zero-shot, nghĩa là model chưa học từ dữ liệu nào.” Nhận xét này đúng hay sai?

- A Đúng, vì zero-shot nghĩa là trọng số chưa học được gì
- B Đúng, vì nhãn cảm xúc không xuất hiện khi pretrain
- C Sai, vì zero-shot là dùng đúng một ví dụ có nhãn
- D Sai, vì model đã được pretrain, chỉ là chưa có nhãn của nhiệm vụ

PHÒNG THEO NOAI 2026 Singapore · Vòng sơ loại · Câu 243

CÂU 2

Câu lạc bộ đã có bộ phân loại CLIP ba lớp: bàn, bảng, máy tính. Nay cần thêm lớp “máy chiếu”, nhưng chưa có ảnh máy chiếu nào được gán nhãn. Làm thế nào là hợp lý?

- A Thêm prompt, tính text embedding, rồi thêm một hàng vào V
- B Gán nhãn ảnh máy chiếu rồi học lại head bốn lớp
- C Fine-tune image encoder trên ảnh máy chiếu
- D Không làm được, vì máy chiếu không phải lớp của ImageNet

PHÒNG THEO NOAI 2026 Singapore · Vòng sơ loại · Câu 261

CÂU 3

Để kiểm xem các prompt lớp có quá giống nhau không, một đội lấy T gồm K text embedding đã chuẩn hóa từng hàng, tính $S = T @ T.T$, rồi $\text{nearest} = S.\text{argmax}(\text{axis}=1)$. Không có hai embedding nào trùng nhau. nearest cho ra gì?

- A Mỗi hàng ra prompt gần nhất, khác chính nó
- B Mỗi hàng ra chính chỉ số của nó
- C Mỗi hàng ra prompt xa nhất của nó
- D Báo lỗi, vì S không phải ma trận vuông

PHÒNG THEO [USA-NA-AIO 2026 · Vòng 1 · Part 5.8–5.9](#)

CÂU 4

Một ảnh có cosine 0.30 với mô tả “bàn” và 0.25 với mô tả “bảng”. Softmax hai lớp cho “bàn” bằng bao nhiêu khi temperature $\tau = 1$, tức scale 1, và khi $\tau = 0.01$, tức scale 100?

- A Với $\tau = 1$ khoảng 0.99, với $\tau = 0.01$ khoảng 0.51
- B Với $\tau = 1$ khoảng 0.55, với $\tau = 0.01$ khoảng 0.55
- C Với $\tau = 1$ khoảng 0.51, với $\tau = 0.01$ khoảng 0.51
- D Với $\tau = 1$ khoảng 0.51, với $\tau = 0.01$ khoảng 0.99

PHÒNG THEO [NOAI 2026 Singapore · Vòng sơ loại · Câu 240](#)

CÂU 5

Để phân loại review, một đội đặt bốn review có nhãn vào đầu prompt, rồi để GPT-2 chấm nhãn cho review mới. Không có optimizer nào được chạy. Cách này gọi là gì, và trọng số của model ra sao?

- A Zero-shot, trọng số giữ nguyên
- B Few-shot, trọng số thay đổi theo bốn ví dụ
- C Few-shot, trọng số giữ nguyên
- D Fine-tuning, bốn ví dụ là một batch train

PHÒNG THEO NOAI 2026 Singapore · Vòng sơ loại · Câu 244

CÂU 6

Chấm cảm xúc bằng GPT-2: điểm của mỗi nhãn là log-prob của “positive” hoặc “negative” đứng sau review và chữ “Sentiment:”, mỗi nhãn đúng một token. Trên 200 review validation gồm 100 khen, 100 chê, model chọn positive tới 186 lần, và accuracy là 0.54. Bước tiếp theo hợp lý nhất là gì?

- A Chấm hai nhãn với một input trung tính để đo độ lệch
- B Đổi sang trung bình log, vì tổng log phạt nhãn dài
- C Báo 0.54 là mức trần của cách zero-shot này
- D Đưa 200 review validation vào prompt làm ví dụ

CÂU 7

Hai token được chấm có xác suất có điều kiện là $1/2$ và $1/8$. PPL, tính bằng exp của trung bình âm log-probability, bằng bao nhiêu?

A $1/4$

B 4

C 3.2

D 16

CÂU 8

Nhật ký train một language model ghi train loss 1.8, là trung bình negative log-likelihood theo token với log tự nhiên, và validation perplexity 19.1, cùng tokenizer và cùng cách đếm token. Trung bình negative log-likelihood trên validation khoảng bao nhiêu?

A Khoảng 1.28, thấp hơn train loss

B Khoảng 2.95, cao hơn train loss

C Khoảng 6.05, cao hơn train loss

D 19.1, vì PPL chính là loss trung bình

PHÒNG THEO NOAI 2026 Singapore · Vòng sơ loại · Câu 84

CÂU 9

Causal LM nhận input [BOS, A, B]. Nếu chuỗi đầy đủ là [BOS, A, B, EOS], thì khi train theo kiểu đoán token kế tiếp, logits ở ba vị trí được so với những target nào?

A [BOS, A, B]

B [B, EOS, PAD]

C [EOS, B, A]

D [A, B, EOS]

CÂU 10

Bài thi có 10 000 ảnh test và 50 lớp. Code của một đội chạy text encoder trên cả 50 prompt lớp ở bên trong vòng lặp qua từng ảnh. Chuyển phần encode prompt ra ngoài vòng lặp thì số prompt mà text encoder phải xử lý giảm từ bao nhiêu xuống bao nhiêu, và dự đoán có đổi không?

A Từ 500 000 xuống 50, và dự đoán giữ nguyên

B Từ 500 000 xuống 50, và dự đoán thay đổi theo

C Từ 10 000 xuống 50, và dự đoán giữ nguyên

D Từ 10 000 xuống 1, và dự đoán giữ nguyên

Đọc thêm

- Radford và cộng sự – [Learning Transferable Visual Models From Natural Language Supervision](#): nguồn gốc CLIP, contrastive learning và cách dùng text làm classifier.
- [Transformers – CLIP](#): shape của embedding/logits và cách gọi model cùng processor.
- [CLIP model card](#): xem miền dùng, giới hạn và điều kiện đánh giá trước khi chọn asset.

- [Transformers — perplexity of fixed-length models](#): tokenization, context window và cách đếm token được chấm.
- [GPT-2 model card](#): xem model causal, ngôn ngữ và giới hạn dùng để chấm chuỗi.
- Zhao và cộng sự — [Calibrate Before Use: Improving Few-Shot Performance of Language Models](#): độ lệch nhãn khi chấm bằng prompt và cách hiệu chỉnh bằng input trung tính.

[Tải gói recipe của chương](#) — gồm các module phụ và hướng dẫn chuẩn bị asset.