理解 formula sheet 上的所有公式

- Dot product 3 和 4 的特点
  - W because s and h might not be in the same dimension
  - UV because of fast calculation
- Tf
`count(t, d)`	词 `t` 在文档 `d` 中出现的次数
- IDF
`N`	文档总数
`df_t`	含有词 `t` 的文档数量（Document Frequency）
- Layer normalization
- **2. 方差（Variance, σ²）：**

  $$\sigma^2 = \frac{1}{d}\sum_{j=1}^{d}(x_j - \mu)^2$$

  表示该样本所有特征的方差，用于衡量特征的离散程度。

  3. 标准化（Normalization）：

  $$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

  其中，ε 是一个很小的数，用于防止除以 0。

  4. 线性变换（可学习参数）：

  $$y_i = \hat{x}_i \cdot \gamma + \beta$$

  其中 γ 和 β 是 **可学习的参数**，用于恢复或调整尺度。

  ✅ **Cheatsheet 一句话总结：**

  > LayerNorm: Normalize across features of one example (mean=0, std=1), then scale & shift with γ and β.

Fine-tuning

让模型更适应某一个特定的下游任务，比如情感分析、问答、文本分类等。

1. Vector, Sparse Vector, Map
    a) Space: vector > map > sparse vector
    b) Retrieval time: MAP >= Vector > sparse
    c) Comparing similarity: Map = sparse vector > vector
2. comparing vectors; 2 methods
    a) dot product: longer vector gets bigger score
    b) cos similarity: only consider about direction, so length does not matter **as long as 2 vector have the same length**
        i. **Not useful When vectors have the same direction but different magnitudes**
3. Word2vec algorithm; 2 methods
    a) CBOW: Using context to predict central word
        i. SoftMax (Mean of input word vector * weight)
            1. SoftMax may have overflow mistake，可以将输入向量中的每个值减去最大值
            2. softmax 会保留每个值的概率信息，不做“选择”，**只是提供所有选项的概率**
        ii. **Continuous** 的原因是因为向量表示不再是整数，而是连续的浮点数
        iii. Small context (+/- 2 words), more syntactically similar words
        iv. Larger context (+/- 5 words), more semantically similar words
    b) Skip-gram: Using central word to predict context
4. TF-IDF
    a) TF: Term frequency, can use log to minimize the affect of very frequent word
    b) IDF: frequent word will have lower score
    c) TF-IDF: we want to find those really important word in each document, those words possibly are not frequent in other documents, that’s why we need to use TF-IDF
5. How to learn the relationship between words
    a) Word2Vec
    b) Glove: adjust word vector according to those words which usually appear with it at the same time
    c) Reducing the sparsity of vector, transfer sparse vector to dense vector, allow easy comparison
6. 5 keys components of NLP (Cheat sheet)
    ~~a)~~ Data: get the data,~~like transfer token to vector~~, like tokenization, not every data can be easily split
    b) Model: **give score for (input, output) pairs**
    c) Inference method: find the highest score
    d) Metric: based on the prediction and real output to find the difference and potential way to improve the model
    e) Learning: model learning from metric’s suggestion
7. Evaluation — confusion matrix (Cheat sheet)
    a) FP: predict as positive, but reality is negative; **precision is important: you want to get less prediction error**
    b) FN: predict as negative, but reality is positive; **recall is important: you do not want**

to miss any positive class

c) Micro: regardless of class distribution, P and R are always equal when there is no null class

d) Macro: minor class also have impact

8. ROC, AUC, PRC (Cheat sheet)
   a) ROC: Top left corner is better, high TRP, low FPR
   b) AUC: distinguish between positive and negative samples
   c) PRC: Top right corner is better; it mainly consider about the **positive** cases, so sometimes more accurate than ROC, such as unbalance data (many negative case)
9. MLP
   a) Non-linear by introducing activation function
   b) More layer means more complex model, may capture more info
10. Activation Function (Cheat sheet)
    a) Tan: [-1:1]; [0:1]
    b) Sigmoid: [0:1]; [0:0.25]
    c) Relu: [0:inf]; 分段函数，0 之后的 gradient 都是 0，但是总体而言，减少了 gradient vanishing 的概率
11. Loss Function Description
    a) Define the difference between prediction and real answer
    b) Towards negative means to optimize the model
12. Gradient
    a) Stochastic: read 1 data each time, avoid local min, not always converge, slower, not stable
13. Overfitting Solution (Cheat sheet)
    a) L1
    b) L2
    c) Dropout
    d) Early Stopping
14. Handle variable length input; 4 methods and what kind of question may lead
    a) Mean: may loss order information
    b) Truncate: may loss some information
    c) Padding: take many memory space; longer sentence still did not solve
    d) Concatenating: may still not have same length with other input
    e) 2 common problems
       i. Loss order information
       ii. Need to relearn the information of certain word vector when it in different position
15. RNN
    a) 2 个 learning 中的常见问题
       i. Exploding gradients: some numeric value is too big – Using ~~max~~ **min** (threshold, gradient) can solve it
          1. **主要是因为权重过大或激活函数的导数大于 1**，使得反向传播过程中梯度不断放大，导致梯度值变得非常大。
       ii. Gradient vanishing: through many layers, the gradient may become very small,

one good example is the equation of MLP – Using LSTM is an efficient way
    1. 主要是因为 non-linearly
b) 双向 RNN
    i. No loop
c) 在一次 RNN Model 里面预测多个结果
    i. Use transducer to output many results, such as using it to output tag for each word

16. POS
    a) Morphological: similar ~~meaning~~ **affix**
    b) Syntactic: ~~was, is, be~~ **elephant, rabbit; can be replaced with another word and remain grammatical**
17. NER
    a) 对 NER 进行 analyze: we use micro as consider the importance about the proportion of each class; many TN
18. Exhaustive
    a) k^n time complexity, always find global optimum, cost a lot
19. Greedy (see cheat sheet)- random sampling
    a) Top-1
    b) Random 要用 exponential
    c) Top-k 要用 exponential
    d) Top-p: 要用 exponential
    e) Contrastive
20. Beam (see cheat sheet)
21. Graph search: A* search
    a) Estimate final score and use that to guide generation.
22. Viterbi(see cheat sheet)
    a) Can be used in many model as long as it has previous input (transition model) and current input (emission model)
    b) 1 previous label: |words|*|labels|^2
    c) 2 previous labels: |words|*|labels|^3
23. Graph Parsing — CKY
    a) Require CNF
    b) 为什么在有 LLM 的今天，我们还需要 Graph Parsing
        i. That is just what we need, not costed
        ii. LLM not good at certain task
24. Reference
    a) Identify entity
25. Coreference
    a) We need to find related entity inside the context, such he refer to Kumber Professor
    b) Antecedent: the entity appear before
26. Entity Linking / Wikification
    a) Link entity with external resources
27. 3 ways to reduce the reference space

a) Identify all possible candidate for entity
b) Find pair of entity
c) ~~Link the same entity to the same external resources~~; find the transitive closure

28. How to get embedding tables; 3 ways
    a) Word2vec
    b) Glove
    c) Fast text: ~~according to n-grams to form the vector~~ 不直接为完整单词学习向量，而是为每个 n-gram 学习一个向量; benefit is even if there are some words we did not know, we still can use its sub word to guess the word vector
29. What if my data is not the same as the data used for training? 3 ways
    a) Retrain the model only using the new data
    b) Retrain the last few layers of model, more efficient and works well
    c) Through back propagation to use new data train the model
30. **Understand word senses**;
    a) ~~Cos; not suit for certain cases, but can be use in many situations~~ **WordNet: A database of labeled relationships between words.**
    b) ~~Train a model; cannot detect untrained words~~; **Train a model with multiple word vectors, one per sense, Challenge is the data**
    c) Contextual Representations — **ELMo (Bidirectional LM): Vector of a word is dependent on the rest of the sequence.** From begin to end, end to begin
31. Decoder
    a) How to stop decoding? Fix length or using <stop> token
    b) How to train? Teacher Forcing: fast training speed, use correct output to replace prediction as next input. Exposure Bias might be a serious problem: 一旦某步预测错了，后续输入就不再是“正确的”，错误会逐步累积，模型表现可能迅速下降
32. We can also use Greedy Inference with a sequence-to-sequence model (see cheat sheet)
    a) How to stop beam searching encoder-decoder? Keep going until we have N outputs
    b) How to score different outputs with different lengths? Longer output will have higher length, so we need to normalize the score
33. Issues of Encoder-Decoder (RNN)
    a) Bottleneck: since we only output one vector to decoder, it may not capture long distance information
    b) Cannot parallel: RNN ask each token be processes one by one which not efficient enough.
34. Tokenization (see cheat sheet)- **Translation quality**
    a) Human evaluate: fluency, adequacy
    b) compare character ngrams, name is: chrF
    c) compare word ngrams, name is BLEU
        i. no 4-gram match will lead to 0, so works for multiple sentences

ii. Apply a penalty if the output is short

d) word ngrams 对于评分系统到底是好是坏?

i. Word n-gram consider more about the sentence order meaning, such as "I love you" and "you love I" can recognized different

ii. Character n-gram can recognize different words with similar meaning, such as "like" and "liking", **high tolerance to typo**

iii. So, we cannot say one is better than another

35. What if a sentence cannot be split on whitespace to get token? (see cheat sheet)

a) start with small units, then combine units

i. BPE

ii. WordPiece

b) start with big and small units, then delete units

i. Unigram

c) 两者的区别，以及为什么?

i. Second provide more meaningful sub word units as it keep removing low frequent words and less useful units, while BPE/WordPiece just keeps merging frequent ones (even if they're less meaningful).

36. Attention

a) Allows the model to retain memory of long-distance dependencies

b) Dot product 公式中 s 和 h 分别表示什么

i. S means the current input; **query**

ii. H means previous hidden layer output; **key**

37. Self-Attention

a) QKV 的意思

i. Q: the vector who want to compare with other vector

ii. K: the vector who was compared

iii. V: ~~the weight for K~~ **vector with actual content/info to retrieve**

b) 如何像"RNN 使用 hidden layer"一样传递前面的信息

i. Every word compare with other words, and have its own contextual vector

ii. We can stack layers to allow it capture deeper information

c) 如何像 RNN 一样添加 nonlinear calculation

i. Non-linear Feedforward layer

d) **如何像 RNN 一样考虑到 position 呢?**

i. 使用 sin 和 cos

1. Need not to train
2. Limit capability

ii. Learnable Positional Embedding

1. can't generalize to sequences longer than those seen in training!

iii. Rotary Positional Embeddings — RoPE

1. similarity remains the same for two words the same distance apart!

e) 如何像 RNN 一样进行 backpropagation 训练

i. Casual mask, that is earlier token cannot see later token **enabling autoregressive training and RNN-like backpropagation.**

38. Transformer Encoder

a) Multi-head: each head responsible for certain task, make our prediction can capture more information
b) Scaled dot product: reduce the effect of vector dimension
c) Residuals: make the gradient smoother, reduce vanishing gradient problem, **Improve training speed**
d) Layer Normalization: fast training, stabilize training

39. Transformer Decoder
    a) Cross attention; QKV 分别在哪
        i. Q: decoder
        ii. KV: encoder
40. Benefit of transformer
    a) Small performance improve; huge efficiency improve which help scaling
41. Transformer 的种类对比用到 bench mark 叫做什么
    a) Glue: BERT > GPT
42. Attention is quadratic, is that a problem?
    a) No memory bandwidth is the problem
    b) What is flash attention? Reduce memory bandwidth need
43. N-Gram Language Model
    a) determine probabilities by dividing one count by another
    b) Why is it a **strong assumption**? What problem will arise
        i. ~~Depending on the previous words;~~ ? **Surprising matching scores: a b c … x b z, if you switch bc and bz, you will get same score**
        ii. Cannot capture long-distance dependencies
    c) How to make a less strong assumption
        i. Increase N
    d) 那么具体是如何计算 P 的呢？n=3
        i. P(C|AB)=P(ABC)/P(AB)
    e) 如何处理 start 和 end 在 sequence 中？
        i. Just add <start>, <end> token
            1. 这样模型才能**正确计算第一个词的概率，**因为 N-Gram 模型需要固定数量的前词作为上下文。
    f) 如何处理 numerical issue when P is too small? log
44. Using LLMS 3 ways, adv and disadv
    a) Directly use; no further training needed; may not suit certain task **("zero-shot" or "few-shot" prompting)**
    b) Use its output as input for other models; cost a lot
    c) Train on our own data – Fine tuning
        i. Feedforward
        ii. adapter
45. Evaluate LLM 2 ways, adv and disadv
    a) MRR: close to 1 means good; the earlier the prediction appear, the higher the mark is
    b) Perplexity: close to 1 is good, can be infinite, indicate how easy the model understand the sentence

46. Which tasks can be done by each model directly? (see cheat sheet)
47. Increasing Efficiency of LLM 4 Ways (see cheat sheet)
48. In-Context Learning – ICL (see cheat sheet)
    a) Perplexity
        i. Low perplexity prompts are better
    b) Order of examples matters
        i. Unsorted data is preferred
    c) number of examples
        i. more number of examples usually lead to better performance
    d) Label of examples
        i. Does not matter; Structure of data is more important
    e) Chain of Thought
        i. Bad explanation usually leads to the wrong answer
    f) What is verbalizer
        i. Certain template we use for ICL
49. Retrieval Augmentation 3 ways (see cheat sheet)
50. Data Sources
    a) Overfitting into one dataset: we need to make sure label what dataset our model is good at
    b) Shortcuts/validity: our model really understand the question
        i. Whether a model can do well without the question and/or context
        ii. Collect data from real world
        iii. ~~Modify the training data's label slightly to see whether model can find it~~ **Wrong answers that have many matching words with the input question or true answer**
        iv. ~~Modify the question to make it have many similar words to unrelated answer to see whether model can find it~~ **Make minimal edits to the true answer that make it wrong**
    c) Statistical power: does the data include enough difficult samples
51. Annotation
    a) Edit auto-generated labels
        i. High consistency, but may fail to misannotated a series of token
    b) Annotate from scratch
        i. Pilot Annotation: annotation people annotate a sample first and adjust the instruction based on their reply
        ii. Cohen's Kappa: ~~judge the correct on accident~~ **correcting for chance agreement**~~.~~, 1 is better
52. Instruction Tuning
    a) FLAN-T5: higher generalization
    b) LIMA: for alpaca, less quantity with high quality data is better than high quantity with low quality data
    c) Synthetic data
        i. Cheap
        ii. Easy to get

iii. Cover corner cases

d) Effect of training examples
   i. Increase training examples does not improve generation quality

e) Remaining Issue
   i. Token-level does not work: some sentence may get similar score, but their meaning is different
   ii. No absolute answer to certain questions

53. Preference Optimization
   a) Backprop the model, assign higher score to preferred answer
   b) RLHF: build a reward model
   c) DPO: directly train from preference data, might not reliable
   d) Alignment brittle: 我们要在 transformer 的 just before output step 解决

54. automatically find good prompts (Optimizing Prompts); 3 methods
   a) vector
   b) Reinforce learning
   c) Use as a system

55. Agent
   a) Reasoning: happened before generating prediction
      i. Self-consistency with chain of thought: ~~consider about the reason of model's prediction~~ **voting with multiple outputs**
      ii. Refletion: consider about previous part
      iii. Tree of thoughts: **exploring multiple possible "thought paths**"
      iv. what is planning? Temporary-prompt, permanent-database
   b) Acting: doing something beyond the generation; RAG
   c) ReAct

(1 mark) Which of the following is true of a Model?

- ◯ It calculates the score of an (input, output) pair
- ◯ It finds the correct output
- ◯ It calculates the score for an input
- ◯ It finds a high scoring output

Classifying whether a website is written in Italian for a study of how commonly Italian is used online. Here, a true positive is an Italian website that is labelled as Italian. It is important that the results are representative of the web.

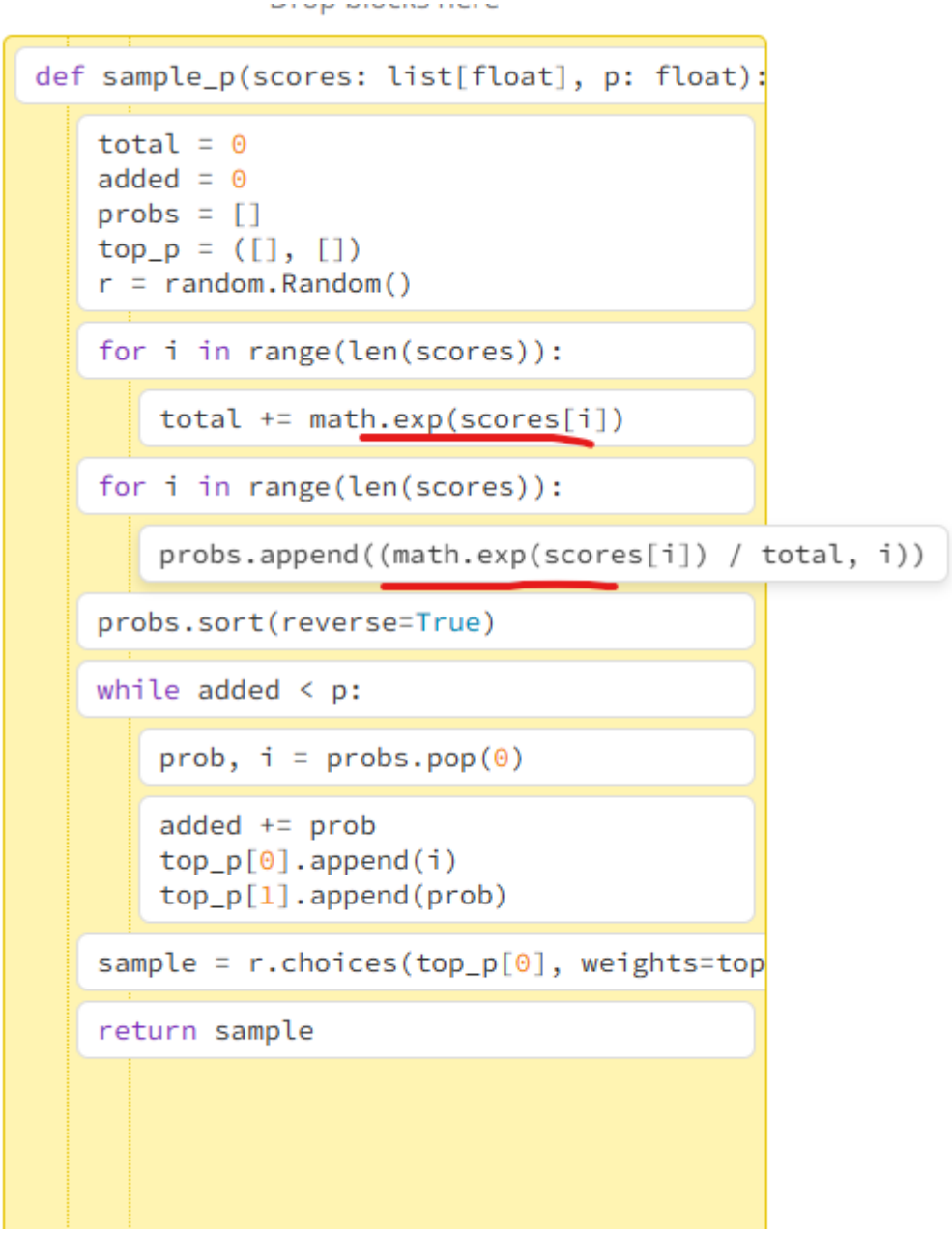
◯ Precision ◯ Recall ◯ F-Score ◯ Accuracy

2. (1 mark) Which of the following is true of an Inference Method?

- ◯ It calculates the score for an input
- ◯ It finds a high scoring output
- ◯ It calculates the score of an (input, output) pair
- ◯ It finds the correct output

4. (1 mark) Which of these parts of the transformer help make training smoother? Select all true statements.

- ☐ Residual connections
- ☐ Layer normalisation
- ☐ Positional encoding
- ☐ Feedforward layers
- ☐ Self-attention



```
def sample_p(scores: list[float], p: float):
    total = 0
    added = 0
    probs = []
    top_p = ([], [])
    r = random.Random()
    for i in range(len(scores)):
        total += math.exp(scores[i])
    for i in range(len(scores)):
        probs.append((math.exp(scores[i]) / total, i))
    probs.sort(reverse=True)
    while added < p:
        prob, i = probs.pop(0)
        added += prob
        top_p[0].append(i)
        top_p[1].append(prob)
    sample = r.choices(top_p[0], weights=top
    return sample
```

这里要用 softmax

```
other_words = random.sample(vocab, 2)
```

这是随机生成 2 个值

```python
def init_weights(self):
    # Initialise the weights to be random values in the matrices and zero for the biases
    initrange = 0.1
    self.i2h.weight.data.uniform_(-initrange, initrange) # 初始化为 [-0.1,0.1] 之间的均匀分布随机值。
    self.i2h.bias.data.zero_() #  偏置向量 初始化为全 0。
    self.h2o.weight.data.uniform_(-initrange, initrange)
    self.h2o.bias.data.zero_()
```

```python
def train(category_tensor, line_tensor):
    hidden = rnn.initHidden() # 初始化 hidden state

    rnn.zero_grad() # 梯度归零

    for i in range(line_tensor.size()[0]):
        output, hidden = rnn(line_tensor[i], hidden)

    loss = criterion(output, category_tensor)
    loss.backward()

    # Add parameters' gradients to their values, multiplied by learning rate
    for p in rnn.parameters():
        p.data.add_(p.grad.data, alpha=-learning_rate)

    return output, loss.item()

# Keep track of losses for plotting
current_loss = 0
all_losses = []
```

```python
class BiRNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(BiRNN, self).__init__()
        self.hidden_size = hidden_size

        # bidirectional=True 表示双向
        self.rnn = nn.RNN(input_size, hidden_size, batch_first=True, bidirectional=True)

        # 输出层：因为是双向，所以 hidden_size * 2
        self.fc = nn.Linear(hidden_size * 2, output_size)
```

```python
def forward(self, query, keys):
    # Computes attention scores and weighted sum (context vector) for the given query and keys.
    # 是cross attention
    # Query: decoder's hidden state. Keys: encoder outputs.
    # 在模拟 decoder 在生成每个词时，去“关注” encoder 的不同部分。
    # Scores: raw attention scores for each key given the query.
    scores = self.Va(torch.tanh(self.Wa(query) + self.Ua(keys)))
    # Adjusting dimensions for softmax operation.
    scores = scores.squeeze(2).unsqueeze(1)

    # Softmax to obtain attention weights.
    weights = F.softmax(scores, dim=-1) # 对最后一个维度做归一化
    # Weighted sum of keys to get the context vector.
    context = torch.bmm(weights, keys)
    return context, weights
```

```python
class DotProductAttention(nn.Module):
    def __init__(self, hidden_size):
        super(DotProductAttention, self).__init__()
        self.out_size = hidden_size *2

    def forward(self, query, keys):
        scores = (query * keys).sum(-1)
        scores = scores.unsqueeze(1)

        weights = F.softmax(scores, dim=-1)
        context = torch.bmm(weights, keys)
        return context, weights
```

```python
def forward(self, query, keys):
    d_k = keys.size(-1)    # 获取 key 的维度
    scores = (query * keys).sum(-1) / math.sqrt(d_k)
    scores = scores.unsqueeze(1)

    weights = F.softmax(scores, dim=-1)
    context = torch.bmm(weights, keys)
    return context, weights
```

Sum: -1
Unsqueeze: 1
Softmax: -1

```
>>> nlp = spacy.blank("en")
>>> doc = nlp("ten")
>>> for token in doc:
```

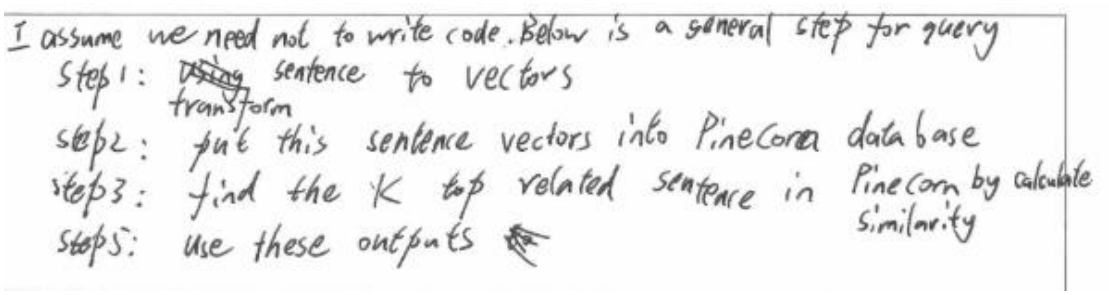
`pos_` 和 `pos` 在使用不同pipeline包时会有所不同，在没有词性识别功能的包时，比如en，会返回null

- `en_core_web_sm` 会分别返回POS tag和integer ID value

```python
from transformers import pipeline

ner = pipeline("ner", grouped_entities=True)
# 表示对识别出的实体进行合并，把属于同一个实体的多个词合成一个完整的实体。例如，“New” 和 “York” 会合并成 “New York”
ner("My name is Jonathan K. Kummerfeld and I teach NLP at The University of Sydney in Australia.")
```

Pinecorn: database



Vowpal wabbit: very fast, batch learning

10. In which scenario might **cosine similarity fail to distinguish vectors** effectively?

A. When vectors are of different lengths

B. When vectors are sparse

C. When vectors have the same direction but different magnitudes

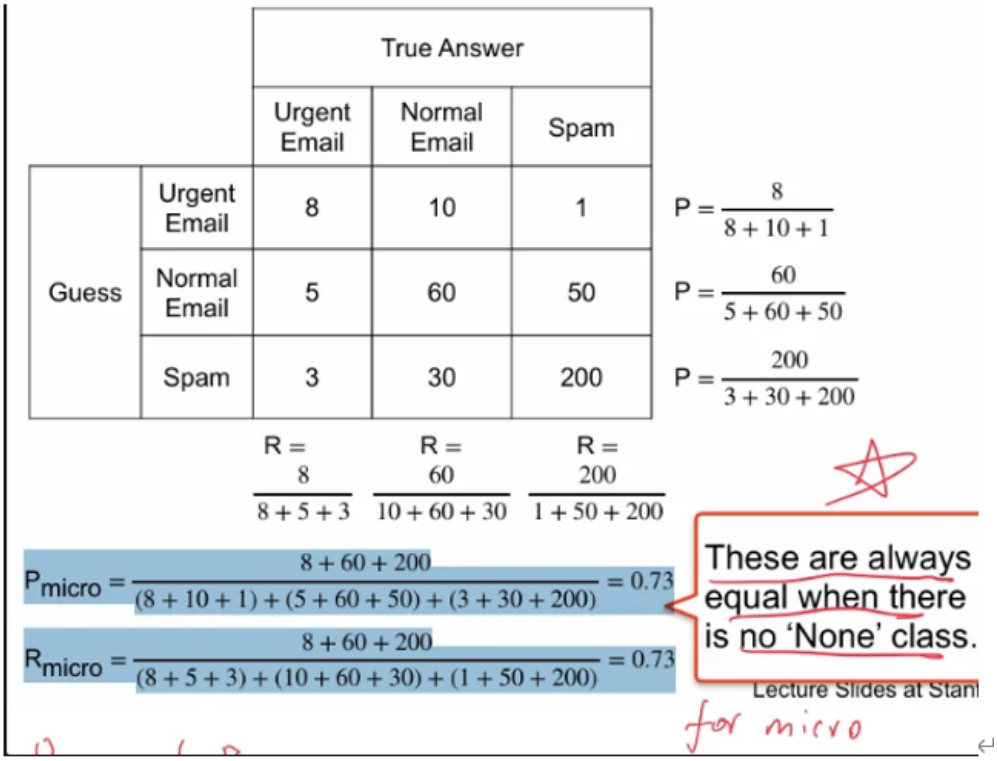
D. When vectors contain only zeros

> *Answer: D*

这里 A 说的不是 Dimension，而是向量的长度

cosine similarity returns 1 (maximum similarity) for vectors pointing in the same direction, regardless of their magnitudes. Thus, it fails to distinguish such cases.

Micro 公式中你会发现把所有 class 当整体来计算



心里想想 Macro 对于每个 class 是怎么算的

29. Which metric summarizes the Precision-Recall Curve similar to how AUC summarizes the ROC curve?

A. Average Precision (AP)

B. F1 score

C. Accuracy

D. Specificity

> *Answer: A*

11. In gradient-based optimization, why do we use the derivative of the loss function?

A. To determine the batch size

B. To estimate training time

C. To find the direction to update model weights

D. To adjust regularization strength

> *Answer: C*

43. What does it mean for an RNN to act as an encoder (or accepter)?

> *An RNN encoder reads an entire input sequence and compresses its information into a fixed-size hidden state, which can then be used for tasks like translation or classification.*

44. Describe what a transducer RNN is used for.

> *A transducer RNN produces output at every time step of the input, making it suitable for sequence-to-sequence tasks like speech recognition or tagging.*

7. What kind of problem is Viterbi best suited to solve?

A. Text summarization

B. Named Entity Recognition

C. Question answering

D. Document classification

> *Answer: B*

22. What type of grammar is required for the CKY algorithm?

A. Context-sensitive grammar

B. Regular grammar

C. Chomsky Normal Form (CNF)

D. Free-text grammar

> *Answer: C*

26. What makes LLMs less reliable for certain parsing tasks?

A. They cannot generate outputs

B. They rely entirely on statistical predictions

C. They are too fast

D. They use rule-based logic

> *Answer: B*

A. They cannot generate outputs：LLMs 肯定能生成输出，不正确。

C. They are too fast：速度不是导致不可靠的原因。

D. They use rule-based logic：LLMs 不是基于规则逻辑，而是基于统计学习。

27. What is one benefit of CKY over LLM-based syntactic interpretation?

A. CKY is faster

B. CKY gives a formally verifiable parse

C. CKY uses reinforcement learning

D. CKY is differentiable

> *Answer: B*

47. Why are classification tasks suitable for all three modeling types (encoder, decoder, encoder-decoder)?

> *Classification only requires predicting a label for a given input, which can be done via any architecture. Encoders can process the input efficiently, decoders can generate the label as a token, and encoder-decoders can map input to output in a structured way.*

51. What type of metric is BLEU considered when evaluating language models?

A. Intrinsic

B. Extrinsic

C. Structural

D. Probabilistic

> *Answer: B*

And also WER

ChrF

55. What is the key difference between intrinsic and extrinsic metrics in LLM evaluation?

Intrinsic metrics assess the model's language ability (Perplexity, MRR) directly, while **extrinsic metrics** (BLEU, WER) evaluate its impact on real-world tasks.

Recap

Attention: A weighted average of vectors.

Self-Attention: A weighted average of vectors where the weights and input vectors are all from the input.

Self-Attention as a replacement for an RNN: Need non-linearities and a position representation.

Non-linearities - Add a feedforward layer after attention.
Position representation - Rotate embeddings different amounts for each position.

Recap

From Self-Attention to the Transformer:
(1) Do attention multiple times at once
(2) Add residual connections
(3) Add layer normalisation
(4) Stack the architecture multiple times

Decoder Structure: Mostly the same components as the encoder. Adds cross-attention (ie., what we previously called attention) to get info from input and an output layer.

Residual perspective: If we draw the structure slightly differently we can think of the transformer differently. At its core is just the word embedding, and then everything else is about learning small edits to it.

Recap: N-Gram Language Modelling

Language models take a string as input and produce a score for it as output.

N-gram language models do this by making a Markov assumption and estimating probabilities based on counts of word sequences seen in data. Care must be taken to handle rare words, the start and end of sequences, and small probabilities.

Recap: Training LLMs

Three General Model Types: Models can be classified as (a) encoders, (b) encoder-decoders, or (c) decoders. They differ in their architecture and most natural uses. Most models today are decoders.

Training Tasks: All of these models train by taking plain text, somehow modifying it, and then getting the model to identify / fix the modification. Different model types are trained with different tasks. The most widely used and well known are:

Next token prediction - given the start of text, predict the next token
Masked token prediction - given text with some tokens masked out, predict the masked tokens

Recap: Using LLMs

Tasks using Language Models: We can use a language model as part of a system to do a task, where the LMs outputs are the input to a task specific model. When we do this, we can keep the LM fixed or fine-tune it. This is similar to how we used word embeddings in earlier lectures.

Tasks as Language Modelling: Many different tasks can be expressed in a way that uses a language model to do the task. Encoder models are somewhat more limited in what they can easily do.

Recap: Evaluating LLMs

Mean Reciprocal Rank: A simple way to evaluate predictions, but rarely used.

Perplexity: For a long time, the standard way to evaluate LMs. It is based on the probability the LM assigns to the test text. The equation rescales based on the length of the text, so it depends on the tokenisation used.

Intrinsic vs. Extrinsic Metrics: The metrics above are intrinsic because they use only the LM itself. Similarly, the word analogy task is an intrinsic measure of word embedding quality. In contrast, extrinsic metrics use the LM as part of a system to do a task, which can be more informative. Modern benchmarks blur the lines here because they only use an LM, but the choice of prompt or other details may matter.

Recap: Efficiency

Reducing model size: To make models smaller we can use a large model to train a small model (distillation) or we can prune parts of a large model. Pruning is hard to do in a way that is computationally useful, so distillation is much more common

Increasing training memory efficiency: During training, fitting all the updates to the weights in memory can be expensive. Methods like LoRA allow us to approximate the changes, enabling training on lower memory GPUs.

Numerical approximation: We do not need full precision for our models. Reducing numerical precision can save GPU memory.

Mixtures of models: To improve performance we can make models that are composed of a set of smaller models. Only one (or a few) models are active at a time.

Recap: Other Models

State space models: Sequential models, e.g., RNNs, may seem out of date, but there is still active research on variants. In this section we saw one, Mamba, but there is also RWKV, the xLSTM, and others. None of them are competitive with transformers on accuracy yet, but they can handle long contexts and be very fast.

Recap: In-Context Learning (ICL)

ICL: Rather than modifying the weights of the model, provide instructions and examples in the input and hope that it will somehow use those to model the task.

LLMs appear to benefit from:
- Examples providing the label space
- Seeing the distribution of inputs
- Seeing the format of the intended output

You should:
- Use as many demonstrations as you can
- Avoid patterns in the order of examples

Recap: Retrieval Augmentation

General Idea: Retrieve data from a text collection or database and then use that in the generation process somehow.

Retrieval: Most common approach is to calculate similarity of vector representations between items in the database and some aspect of input.

Use: A variety of methods! Good to know about:
- Treating retrieval as an LM itself and doing model ensembling
- Provide the retrieved content in the prompt
- Interactive methods (related to Agents, which we will see later)

Recap

Data collection: This involves finding and filtering data in various ways. Choices can have a big impact on model performance and style.

Data annotation: Once we have raw data we need to assign labels to it. Even if we are using a few-shot approach, we want some labeled data for evaluation. This can be slow, expensive, and difficult. More in the next section!

Key goals in dataset creation:

- Validity
- Reliability
- Statistical power
- Testable
- Publicly licensed
- Avoids social bias

Recap

Key steps in annotation:

- Collect data (previous section of this lecture)
- Develop annotation guidelines
- Train annotators
- Label data
- Check disagreements
- Choose license and release data

Measuring disagreement: A range of metrics exist to understand how much annotators agree. For classification tasks, Cohen's Kappa is simple and widely used. However, a 'good' value is not well defined. For non-classification tasks, agreement is harder to measure and we generally don't have statistically well-motivated methods.

Recap

What is crowdsourcing? A range of methods fit under this heading. In all cases, they involve a large group of people contributing to the creation of a resource.

The most well known and widely used is 'microwork', where people are paid to do small tasks online. Other variants have had incredible success in specific areas (e.g., creating Wikipedia) but are harder to create.

Recap

Instruction Tuning: A form of fine-tuning where the data is examples of a range of different tasks. The goal is to shape LLM outputs to actually do the task.

Task Choice: High quality data matters a lot here. A small number of high quality tasks can do better than a large number of noisy ones.

Remaining issues:

- Token level prediction doesn't measure task success well
- Some tasks don't have a single best answer.

Recap

Preference Optimisation: General idea is to learn from people's preferences (which is better, A or B?).

Reinforcement Learning Approach: Use preferences to learn a reward model, ie., a model that identifies the underlying scoring function people are using. Then train our model using methods from reinforcement learning.

Direct Approach: Use the preferences directly, backpropagating through the model to increase the score for the preferred option and decrease the score for the other one.

Issues:
- Ethical questions about how we create the datasets
- The changes are somewhat brittle

Recap

Agent: General idea is using an LLM as part of a system that interacts with the world. They combine some form of internal processing with some form of external interaction.

ReAct: A prompting method that combines Reasoning and Acting to achieve better results.

Agent Zoo: We explored a range of ways to construct agents. This is a very active research area! The key idea is that LLMs become one part of a system, and that system can take many different forms.

Benchmarks: Evaluating agents requires new benchmarks that are a closer approximation of the real world, sometimes with simulators so the agent can actively interact.

Recap

What is Reasoning? This is any process by which models 'think' before generating the primary output.

Main approaches: Chain of thought prompting was the first and a lot of other approaches are variants / extensions of it. Creating and using memory is an unsolved challenge. Approaches are now mature enough that we are seeing various ways of combining them (e.g., with DSPy).

Planning: We only briefly touched on this massive area, but the core message is that when evaluated carefully, LLMs seem to struggle with planning. Another active area of research.

Recap

What is Acting? This is any process by which models do things beyond generating tokens.

Main approaches: RAG is the simplest and most widely used example of acting. For tools with simple APIs (calculators, search engines) there is success in using them simply with creative prompting. For more sophisticated interactions, e.g., with your whole computer, there are many active efforts.