**Note for NLP Exam**

**Lecture 1: Introduction to Natural Language Processing**

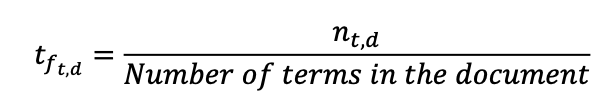
1. What is Natural Language Processing?
   1. A branch of Artificial Intelligence.
   2. Make computers to interact with humans using natural languages.
2. Recent applications
   1. ChatGPT: a chatbot built on OpenAI's GPT-3.5 family of large language models and is fine-tuned with both supervised and reinforcement learning techniques.
   2. Search autocorrect and autocomplete: Correct typos and shows possible search terms.
   3. Language translator: converting the text in one language to another language while keeping the meaning intact.
   4. Social media analysis
   5. Grammar checkers: correct grammar, spellings, suggest better choices.
   6. Email filtering: Email gets classified into several pre-defined sections.
   7. Virtual assistants: Understands voice commands and completes tasks for a user, Siri, Google Assistant.
3. Challenges
   1. Why challenging?
      1. Ambiguity: Similar strings mean different things, different strings mean the same thing.
      2. Context
      3. Commonsense knowledge
4. How NLP works?
   1. Text preprocessing
      1. A data mining technique: transform the raw data into an understandable format.
      2. Purpose: Prepare (or clean) text data before encoding.
      3. Preprocessing of data
         1. Tokenization: break up text document into individual words.
         2. Stop words removal:  Stop words are common words that do not contribute much of the information.
         3. Stemming 词干提取: Reduce a word to its stem word.
         4. Lemmatization 词元化:
            1. the same thing as stemming with one difference, the root word in this case belongs to a valid word in the language.
            2. Use valid words from WordNet (a database of valid words).
         5. Parts of Speech (PoS) tagging 词性标注:  It’s used to categorize words in a text or a document to their corresponding parts of speech based on both its definition and its context. 将文本或文档中的单词归类到其对应的词性中。
         6. N-grams:
            1. the combination of multiple words used together.
            2. Used when we want to preserve sequence information.
      4. Textual data vectorization: Convert text into numbers
         1. Bag of words
            1. one of the text vectorization techniques
            2. Drawbacks:

Vector length = vocabulary size.

Sparse metric 稀疏矩阵 with many 0s.

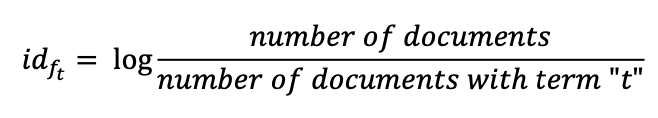
No grammar or ordering information.

* + - 1. Term Frequency 词频-逆文档频率-Inverse Document Frequency (TF-IDF)
         1. A numerical statistic that is intended to reflect how important a word is to a document.
         2. Term Frequency-how frequently a term 𝑡 appears in a document 𝑑:





* + - * 1. Inverse Document Frequency (IDF)- a measure of how important a term is.

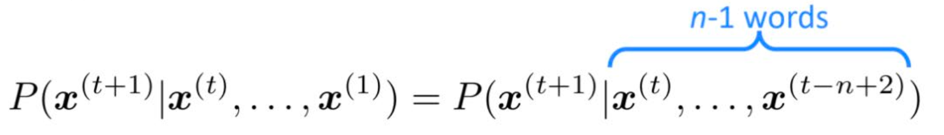


* + - * 1. TF-IDF score, words with a higher score are more important.



**Lecture 2: Language Model**

1. Language model definition & applications
   1. Definition: A probability distribution over sequences of words.
   2. Task: Assign probability to a piece of text.
   3. Language modeling:
      1. Serve as a benchmark task that helps us to measure our progress on understanding language.
      2. A subcomponent of many NLP tasks: speech recognition, handwriting recognition, machine translation.
      3. The task to predict what word comes next.
2. Model construction
   1. Statistical language models
      1. **n-gram model**
         1. **Markov assumption**: depends only on the preceding *n-1* words



* + - 1. n-gram model: Every word in Σ is assigned some probability, conditioned on a fixed-length history (n–1). Σ（词汇表）中的每个词 都会被分配一个概率，该概率取决于一个 固定长度（n−1） 的历史上下文。
      2. n-gram in text generation
         1. Well grammarly structured but incoherent.
         2. increasing n brings sparsity and storage problems.

Sparsity problem 稀疏性问题: Relevant word or sentence never occurred in data.

storage problem: Need to store all n-grams in the corpus.

* + - * 1. Solution: window-based neural network.
    1. Full history model: Every word in Σ is assigned some probability, conditioned on every history.
  1. Neural language models
     1. A fixed-window neural language model
        1. **Improvements** over n-gram:
           1. No sparsity problem
           2. Don’t need to store all observed n-grams
        2. **Remaining** problems:
           1. Window can never be large enough.
           2. No symmetry in processed input.
     2. RNN-based language model
        1. advantages
           1. process any length input
           2. Computation can use information from many steps back
           3. Model size doesn’t increase for longer input
           4. symmetry in how inputs are processed.

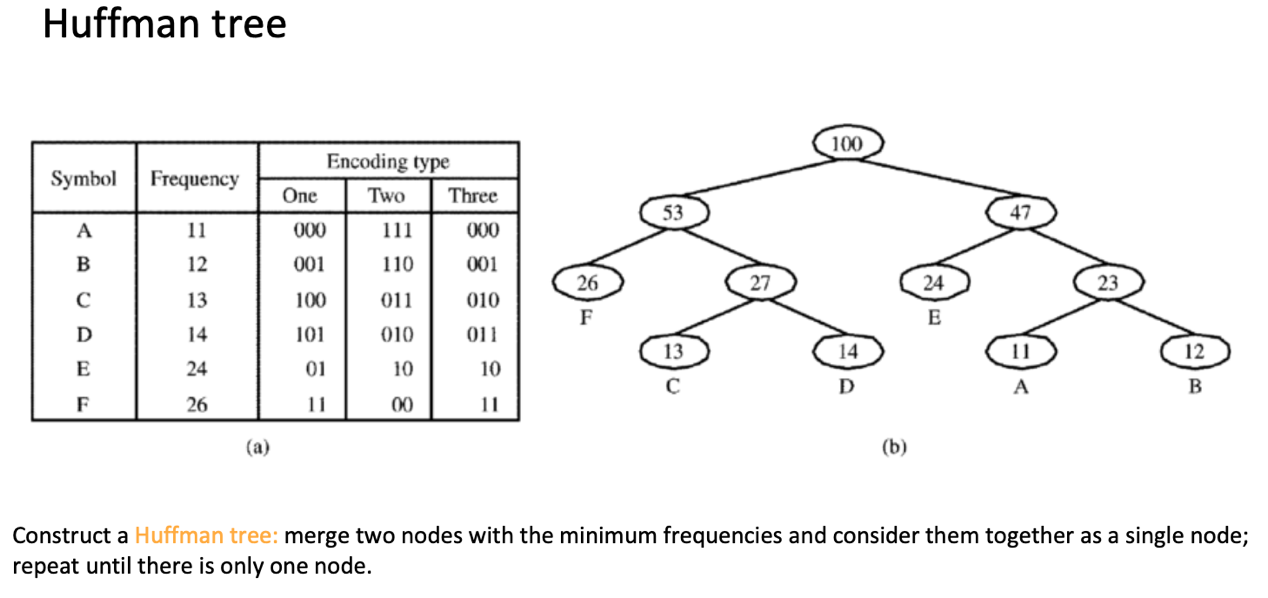
Because Same weights applied on every timestep

* + - 1. Drawback: unable to predict similar long-distance dependencies because of Vanishing/Exploding gradient.
    1. Long Short-Term Memory
       1. a special RNN to solve vanish gradient phenomenon
       2. Use a set of gating units to control the memory.
       3. GRU
          1. A simpler version of LSTM
          2. Fewer gates, less computation
          3. On each timestep t we have input and hidden state.
       4. How LSTM solves vanishing gradient / long-term dependency?
          1. The LSTM architecture makes it easier to preserve information over many timesteps.
          2. Example: if the forget gate is set to 1 for a cell dimension and the input gate set to 0, then the information of that cell is preserved indefinitely.

1. Language model evaluation-Evaluating language model: The standard evaluation metrics for LM is perplexity 困惑度, lower perplexity is better.

**Lecture 3: Word Embedding**

1. Word embedding definition and principles
   1. WordNet - a lexical database of semantic relations 语义关系 between words in more than 200 languages
   2. Problems with resources like WordNet
      1. Missing nuance 细微差别
      2. Impossible to keep up-to-date
      3. Subjective
      4. Human labor for creation and adaptation
      5. Cannot compute accurate word similarity
   3. One-hot vector: discrete symbols
      1. A localist representation 局部表示
      2. Vector dimension = number of words in vocabulary
      3. Problem: No natural notation for one-hot vectors. 没有自然的相似性表示形式
      4. Solution: Distributional hypothesis- Words that occur in similar contexts tend to have similar meanings.
   4. Word embeddings – goal
      1. Build a dense vector for each word
      2. A word vector should be similar to vectors of words that appear in similar contexts
      3. A distributed representation
2. Embedding methods – word2vec
   1. A framework for learning word vectors.
   2. Idea:
      1. Input: a large corpus of text.
      2. Output: A **vector** represents each word.
      3. Use **the similarity of the word vectors** for 𝑤𝑡 and 𝑤𝑐 (Go through each position t in the text, which has a target word 𝑤𝑡 and several context words 𝑤𝑐)
         1. **Skip-gram**: to calculate the probability of context words 𝑤𝑐 given the target word 𝑤𝑡
         2. **Continuous bag of words (CBOW**): to calculate the probability of target word 𝑤𝑡 given context words 𝑤c
      4. **Keep adjusting the word vectors** to maximize the probability.
   3. Skip-gram——Skip-gram’s optimization problem: minimize objective function = maximizing predictive accuracy
3. Improve training efficiency
   1. Reason:
      1. The size of vocabulary V is impressively large
      2. Evaluation of the objective function would take **O(V)** time
   2. Solution:
      1. Negative sampling
         1. A simplified version of NCE (Noise Contrastive Estimation, 噪声对比估计)
         2. Sample from a **noise distribution**
         3. Convert multiple classification problem to (k+1) **binary classification problems**
      2. Hierarchical softmax
         1. Construct a **Huffman tree**, with each leaf node representing a word.
         2. Each **internal node (a cluster of similar words)** of the graph (except the root and the leaves) is associated to a **vector** that the model is going to learn.
         3. The probability of a word w given a vector 𝑤𝑖, **𝑃(𝑤|𝑤𝑖)**, is equal to the **probability of a random walk** starting at the **root** and ending at the leaf node corresponding to **w**.
         4. Complexity: **O(log(V)),** corresponding to the length of the path.



* 1. Comparison:
     1. Hierarchical softmax tends to be better for **infrequent words**.
     2. Negative sampling works better for **frequent words and lower dimensional vectors**.

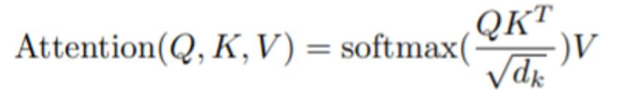
1. Other word embedding methods
   1. GloVe: Global Vectors for Word Representation.
      1. Global statistics (LSA, 潜在语义分析) + local context window (word2vec)
      2. Input: Co-occurrence matrix 共轭矩阵, weight decrease with distance, 𝑋𝑖𝑗=1/d (distance of word pairs)
2. Contextualized word embeddings (ELMo)
   1. Problems with (non-contextual) embeddings: multi-sense words (多义词).
   2. Contextualized embeddings (上下文词嵌入)-ELMo
      1. From **context independent embeddings** to **context dependent embeddings**
      2. The model only gives a certain embedding for one word when this word is given in a sentence.
      3. Use a bi-directional LSTM to pre-train the language model
      4. Key features
         1. Replace static embeddings (lexicon lookup, 例如基于词典查表的方式) with **context-dependent** embeddings (produced by a deep neural language model), i.e., each token’s representation is **a function of the entire input sentence** (整个输入句子的函数).
         2. Computed by a deep **multi-layer, bidirectional** language model.
         3. Return for each token a (task-dependent) linear combination of its representation across layers.
         4. Different layers capture different information.
      5. ELMo architecture
         1. First layer: CNN get context independent embeddings.
         2. Each layer computes a vector representation for each token.
         3. Freeze the parameters of the language model.
         4. Each task with different weights.

**Lecture 4: Transformers and pretraining-finetuning**

1. Attention
   1. Problems with contextualized word embeddings:
      1. RNNs/LSTMs have **long-term dependency** problems (很难捕捉很久以前的信息)
      2. Bi-directional RNNs/LSTMs **feature fusion** (特征融合) and **representation** ability is weak.
   2. Attention– the main technique behind transformers
      1. Why attention?
         1. Reduce **complexity**.
         2. **Parallelize** the computation.
         3. Solve the **long-term dependency** problem.
            1. Self-attention connects all positions in a sequence with a constant number of operations.
         4. Yield more **interpretable** models
      2. What is the attention mechanism
         1. Pay more attention to more important information.
         2. Map a **query** (查询向量) and a set of **key-value pairs** to an **output**.
         3. Output is a **weighted sum of the values**.
            1. Weight assigned to each value is computed by a compatibility function of the query with the corresponding key.
      3. Attention in NLP: aligning while translating
         1. Alignment in machine translation: relationship between source words and target words.
         2. Attention intuition (注意力的直觉理解): each time the proposed model generates a word in a translation, it searches for a set of positions in a source sentence where the most relevant information is concentrated. 最相关的信息集中位置
      4. Attention: formal description
         1. Given a **query** vector, and a set of **key-value** pairs (all vectors), we first calculate the **similarity/attention score** between the query and each key.

𝑠i = 𝑠𝑖𝑚𝑖𝑙𝑎𝑟𝑖𝑡𝑦 (𝒒, 𝒌i).

* + - 1. Normalize the similarity score to be between 0 and 1, and they sum up to 1. These are called **attention distribution**. One way is to use the softmax operation.
      2. Compute the **attention/context vector** 𝒛 as a weighted sum of values.
      3. Keys and values are not necessarily the same, and they could be different as well, such as in machine translations.
    1. Attention: similarity calculation
       1. In general, every similarity measures can be used here, such as the cosine coefficient (余弦相似度) and Pearson correlation coefficient (皮尔逊相关系数).
       2. Commonly used ones for neural networks:
          1. Additive attention
          2. Multiplicative attention
          3. Dot-product attention
          4. Scaled dot-product attention
    2. self-attention
       1. For an input sequence of words, play attention mechanisms between every word and others (including itself).
       2. Features:
          1. Constant path length between any two positions
          2. Easy to parallelize per layer. 每一层都易于并行计算
       3. Scaled dot-product attention in matrix form



* + - * 1. How to construct queries, keys, and values?

Linear transformation from original input embeddings 𝑥i.

* + - * 1. Perform scaled dot-product attention between a word i and all words.
        2. Obtain L self-attention vectors, one for each word (token).
        3. Change these calculations into the matrix form.
    1. multi-head self-attention
       1. choose different forms of 𝑊Q, 𝑊K, and 𝑊V.
          1. Each group of (𝑊Q, 𝑊K, and 𝑊V) is called a head.
          2. Increase the representation performance.
       2. How to combine multi-headed output together?

Concatenate them (拼接) and use another linear transformation.

1. Transformer
   1. Main technique: **multi-head self-attention mechanism**.
   2. The transformer is a novel architecture that aims to solve **sequence-to-sequence** tasks while handling long-range dependencies with ease.
   3. Formal description
      1. Encoder: maps an input sequence of symbol representations x to a sequence of continuous representations z.
      2. Decoder: given 𝑧, generates an output sequence (𝑦1, 𝑦2, … , 𝑦m), one element at a time.
      3. At each step the model is **auto-regressive** (自回归): consuming the previously generated symbols as additional input.
      4. Stacked self-attention and point-wise, fully connected layers (前馈全连接网络) for both the encoder and decoder. 每个 Encoder 和 Decoder 都由若干个重复堆叠的子层（模块）组成
   4. Encoder: In a self-attention layer all of the **keys, values and queries** come from the **output of the previous layer or the raw input embedding** in the encoder.
   5. Decoder:
      1. Each position attends to all positions in the decoder up to and including that position. 关注该位置及其之前的所有位置
      2. **Encoder-decoder attention**: **keys and values** come from the top **encoder** output; **queries** come from the output of the masked multi-head attention in the **decoder**.
      3. Masked self-attention:
         1. **Keys, values and queries** come from the **output of the previous layer or the raw output embedding** in the decoder.
         2. **Prevent leftward information flow** to preserve the auto-regressive property. 为了保持自回归特性，必须防止信息向左传播
            1. implemented by **masking out**future positions.
   6. Other details
      1. Each of the layers contains a fully connected feed-forward network (FFN), which is **applied to each position** separately and identically. 它会被独立地、逐位置地应用到每个输入上
      2. **Positional encodings** (位置编码): input positional information of the sequence.
         1. Transformer does not contain recurrence or convolution, it does not know the order of input tokens.
         2. Idea: input representation of a token is the sum of two embeddings: token and positional
      3. Residual connection (残差连接): learning the changing part.
      4. **Learned embeddings** are used to convert the input tokens and output tokens to vectors of dimension 𝑑model. (用于将输入和输出的 token 映射成维度为 𝑑model的向量) The same weight matrix are shared in the two embedding layers.
2. BERT: Bidirectional Encoder Representations from Transformers
   1. Pre-training of Deep Bidirectional Transformers for Language Understanding
   2. Conditioning on both left and right context in all layers.
   3. Can be finetuned with just one additional output layer to create models for a wide range of tasks.
   4. BERT architecture: Consist of multi-layer bidirectional transformer encoders
   5. Pretraining + Finetuning
      1. Same **architectures** are used in both pre-training and fine-tuning.
      2. The **same pre-trained** model **parameters** are used to initialize models for **different down-stream tasks**.
      3. During **fine-tuning**, **all parameters** are fine-tuned.
   6. What contributes to BERT’s success?
      1. **2 self-supervised tasks** for pre-training
      2. **Compatibility** to various tasks
   7. Input/output representation:
      1. Constructed by summing the corresponding token, segment, and position embeddings.
      2. [CLS] is a special symbol added in front of every input example.
      3. [SEP] is a special separator token.
   8. Pre-training BERT
      1. Two self-supervised tasks: **Masked LM** and **Next Sentence Prediction (NSP**).
      2. Reason for BERT’s success: the amount of data that it got trained on.
      3. Masked language modeling (MLM):
         1. Motivation: Enable bidirectional pre-training
         2. Main idea: Mask some percentage of the input tokens at random, and then predict those masked tokens.
         3. The masking scheme here is to **reconcile the mismatch** between pre-training and fine-tuning.
      4. Next sentence prediction (NSP)
         1. Motivation: understanding the relationship between two sentences.
         2. Main idea: given a preceding sentence A (前一句) and a sentence B, judge if B is the next sentence of A.
         3. Example
            1. Input = [CLS] the man went to [MASK] store [SEP]

he bought a gallon [MASK] milk [SEP]

Label = IsNext

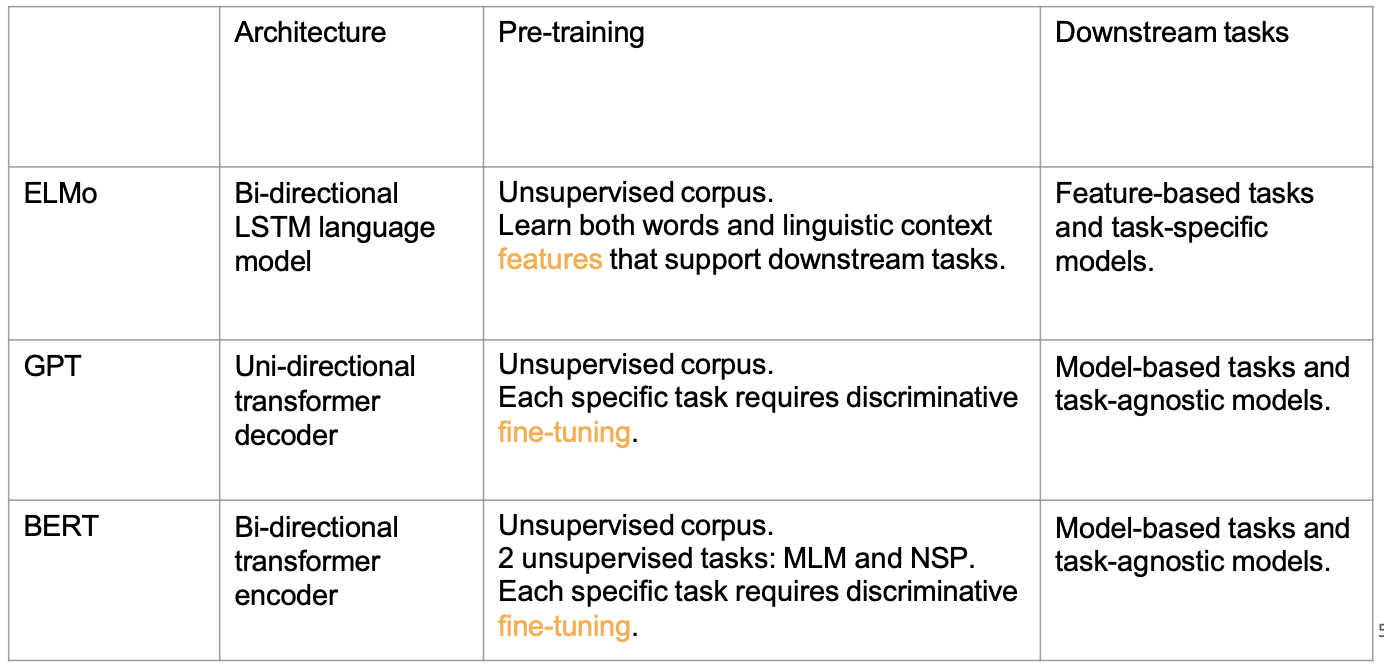
* + - * 1. Input = [CLS] the man [mask] to the store [SEP]

penguin [MASK] are flight ##less brids [SEP]

Label = NotNext

* 1. Fine-tuning:
     1. Common approach: independently encode text pairs before applying attention.
     2. BERT’s approach: concatenate (拼接) a text pair, and then encode with self-attention in order to include bidirectional cross **attention** (双向交叉注意力) **between two sentences**.
     3. For each task, simply **plug in the task-specific inputs and outputs** into BERT and fine- tune all the parameters end-to-end.
     4. Compared to pre-training, fine-tuning is relatively **inexpensive**.
     5. For **classification** tasks (e.g. sentiment analysis) we add a classification FFN for the [CLS] token input representation on top of the final output. 在最终输出的基础上，为 [CLS] token 的表示添加一个分类用的前馈神经网络（FFN）。
     6. For **Question Answering** alike tasks BERT train two extra vectors that are responsible for marking the beginning and the end of the answer

1. GPT: Generative Pre-training
   1. Motivation: semi-supervised learning
      1. Stage 1: Unsupervised learning on word-level or phrase-level
         1. Stage 1 is **less dependent** on the task
      2. Stage 2: Supervised training using these word-level features
         1. Stage 2 models may vary a lot according to different tasks
      3. Pre-training improves **language understanding**.
      4. Framework
         1. Multi-layer transformer decoder 多层 Transformer 解码器
         2. Supervised fine-tuning
2. Pre-trained model comparison: ELMo, GPT, and BERT





**Lecture 5: NLP tasks (1) Understanding tasks**

1. NLU vs. NLG
   1. NLU focuses on **comprehending** and **extracting** meaning from natural language input.

Involve **text classification** and **question answering**.

* 1. NLG focuses on **generating** human-like text that **conveys** information or **communicates** effectively.

Involve **machine translation** and **dialogue generation**.

1. NLU tasks
   1. General Language Understanding Evaluation (GLUE) benchmark is a collection of nine NLU tasks.
   2. Hardness
      1. Easy: Spam Detection, Name Entity Recognition.
      2. Medium: Sentiment Analysis, Information Retrieval.
      3. Hard: Machine Translation, Text Generation.
2. NLU task 1: text classification
   1. Def: **Assign labels or tags** to textual units.
   2. Formulated as the task of generating a **hypothesis**
   3. Task examples:
      1. Spam detection: detect unsolicited (来路不明) and unwanted emails.

Binary: spam, not spam. Multiple classes: primary, social, promotion, different tags.

* + 1. Sentiment analysis: Analyze and extract people’s opinions in textual data.

Binary: positive, negative; Multiple classes: review stars, {happy, sad, surprise, angry}

* + 1. News categorization: help users obtain information of interest in real-time.

E.g., assigning news articles to one of {HomeNews, International, Entertainment, Lifestyles, Sports}, {popular, not popular}

* + 1. Content moderation: sort contributions that are irrelevant, obscene, illegal, harmful, or insulting with regards to useful or informative contributions.
    2. Topic analysis: Identify the theme or topics of a text.
  1. Text classification types:
     1. **Binary** classification: each item belongs to exactly one class among two.

E.g., assigning emails to one of {Spam, Legitimate}

* + 1. **Single-Label Multi-Class** (SLMC) classification: each item belongs to exactly one class among many.

E.g., assigning news articles to one of {HomeNews, International, Entertainment, Lifestyles, Sports}

* + 1. **Multi-Label Multi-Class** (MLMC) classification: each item may belong to zero, one, or several classes.

E.g., assigning computer science articles to classes in the ACM Classification System

* + 1. **Ordinal** classification (OC): as in SLMC, but for the fact that there is a total order among the classes

E.g., assigning product reviews to one of {Disastrous, Poor, SoAndSo, Good, Excellent)

* 1. hard classification vs. soft classification
     1. Hard classification (HC): determine which class(es) an item belongs to. Results are categories.
     2. Soft classification (SC): predict a (d,c) score for each item-class pair, where the score denotes the probability that d belongs to c.
  2. Methods
     1. Feature extraction + classification
        1. Preprocessing: Removing punctuations, Removing URLs, Removing stop words, Lower casing, Tokenization, Stemming, Lemmatization (词元化).
        2. Feature extraction:
           1. Generate dense vectors (稠密向量) using subword embeddings, word2vec.
           2. In classification by topic, make the set of features coincide with the set of words that occur in the training set. 将特征集设为训练集中出现过的词集合
           3. In classification by author/title, features like average word length, average sentence length are used.
        3. Feature selection
           1. Goal: **identify the most discriminative featusres**.
           2. The **“filter” approach**: measure the discriminative power of each feature 𝑡k and retain only the top-scoring features.
           3. **Matrix decomposition**: synthesize new features not suffering from ambiguity and polysemy (一词多义).
        4. classifier selection (分类器选择)
           1. Support vector machines (SVMs)
           2. Logistic regression
           3. Naive Bayesian methods
           4. Neural network-based methods (often support end-to-end classification)
     2. End-to-end mode——end2end neural text classifiers: TextRNN, BERT, GPT, pretrained models.
  3. Labeling: usually costly.
     1. Active learning: interactively query a user to label new data points with the desired outputs.
     2. Self-supervised learning: constructing positive/negative samples.
  4. Evaluation: Precision-recall, F1 score.
  5. Network selection (步骤): **Select a pretrained language model, Domain adaptation (领域适应), Task-specific model design, Task-specific fine-tuning, Model compression**.
  6. Datasets:
     1. Sentiment analysis: **Yelp, IMDb, Movie Review**.
     2. News classification: **AG News, Sogou News**.
     3. topic classification: **Web of Science**.

1. NLU task 2: question answering
   1. Factoid QA: questions that can be answered with simple facts expressed in short texts.
      1. **Information-retrieval (IR) based QA**: **Information retrieval** is used to find relevant passages. **Neural reading comprehension** algorithms read these retrieved passages and draw an answer directly from spans of text.
         1. Reading comprehension: The ability to read and understand unstructured text and then answer questions about it.
            1. The composition of the problem:

Input: context and query

output: two forms of answer

**abstractive**: free-form answer

**extractive**: substring of the content 从上下文中直接抽取片段

* + - * 1. relation between RC and QA:

QA: a task

RC: a possible approach to solve QA

Other possible solutions to QA: knowledge-based information retrieval, keywords detection mechanism, etc.

* + 1. **Knowledge-based QA**:  Transform problems into **logical query statements**, query answers in a **structured database**.
  1. Dataset
     1. Stanford question answering dataset (SQuAD 1.1)
        1. Problem in previous datasets:
           1. High quality human-written databases not very large
           2. Cloze-form questions better (填空式问题质量较好), but not very natural
           3. Noisy.
        2. Why better?
           1. Human-written, less noisy.
           2. Not cloze-form
     2. SQuAD 2.0: have unanswerable questions.
  2. BiDAF: Bidirectional Attention Flow for Machine Comprehension 双向注意力流模型
     1. Incorporating attention better into QA
     2. key features:
        1. Bidirectional attention (new): query-to-context and context-to-query
        2. Multi-level embedding representation
        3. Attended vectors (注意力向量) are passed along together with original embeddings (原始嵌入)
     3. Basic components of the model
        1. Character embedding layer (字符嵌入层)
        2. Word embedding layer (词嵌入层)
        3. Contextual embedding layer
        4. Attention flow layer
        5. Modeling layer
        6. Output layer
     4. Ablation study 消融实验
        1. Character-level embedding: effective in handling out-of-vocab (词表外词) or rare words.

Word-level embedding: better at capturing the overall semantics of words.

* + - 1. C2Q attention (上下文关注问题): Help the model understand the relationship between each word and the question in the context.

Q2C attention (问题关注上下文): Guide the model to focus on the most important contextual positions for the problem.

* + - 1. Dynamic attention: Update attention throughout the modelling layer.

Intuition (设计直觉): Treat attention modules as a single layer.

* + 1. The more attention, the better: Attentive reader < BiDAF < BERT
  1. Future improvements in QA:
     1. Need **Explainability**.
     2. Turn to **Knowledge** grounded QA (知识驱动问答).
     3. Improve **inference** (推理能力).

**Lecture 6: NLP tasks (2) Generation tasks**

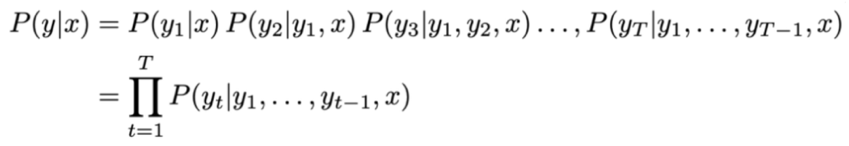
1. Natural language generation tasks
2. Machine translation
   1. Statistical MT model
      1. How to Learn translation model 𝑃(𝑥|𝑦)? Need large amount of **parallel data**.
      2. How to learn 𝑃(𝑥|𝑦) from the parallel corpus?
         1. Introduce a **latent variable** 𝑎: 𝑃 (𝑥, 𝑎|𝑦).
         2. 𝑎 is the alignment: **word-level correspondence** between source sentence 𝑥 and target sentence 𝑦
      3. Alignment details
         1. Alignment is the **correspondence** between particular words in the translated sentence pair.
         2. **Typological differences** between languages lead to complicated alignments.
   2. Neural machine translation
      1. Seq2seq model:
         1. an **encoder-decoder** neural network architecture
         2. convert sequences from one domain to another
         3. involve two RNNs: encoder RNNs and decoder RNNs.
         4. **Conditional Language Model**
            1. 'Language model’: decoder **predicts the next word** of the target sentence y.
            2. 'Conditional’: prediction is **conditioned on the source sentence 𝑥**.
         5. Train a seq2seq NMT system: regarded as a single system; All parameters are updated "end-to-end" through Backpropagation.
         6. Decoding
            1. Greedy decoding:

Take the **most probable word on each step**.

Problem: no way to **undo** decisions.

* + - * 1. Exhaustive search decoding (穷举搜索解码):

find a translation 𝑦 with length 𝑇 such that



Is maximized.

 Problem: computing all possible sequences 𝑦 has **a complexity of 𝑂(𝑉T )**.

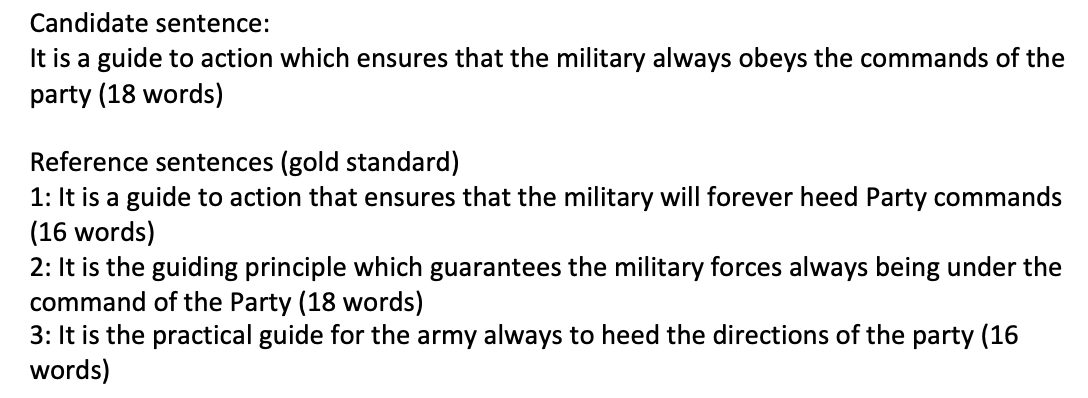
* + - * 1. Beam search decoding (波束搜索解码)

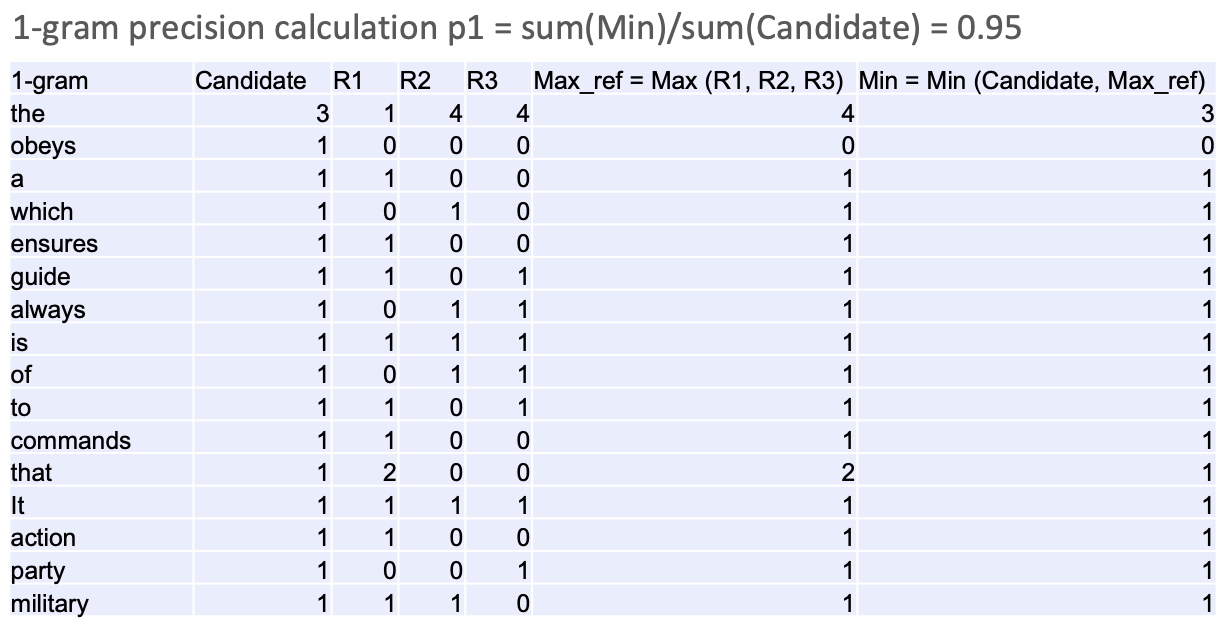
On each step of decoding, keep track of the **𝑘 most probable partial translations** (𝑘 is the beam size)

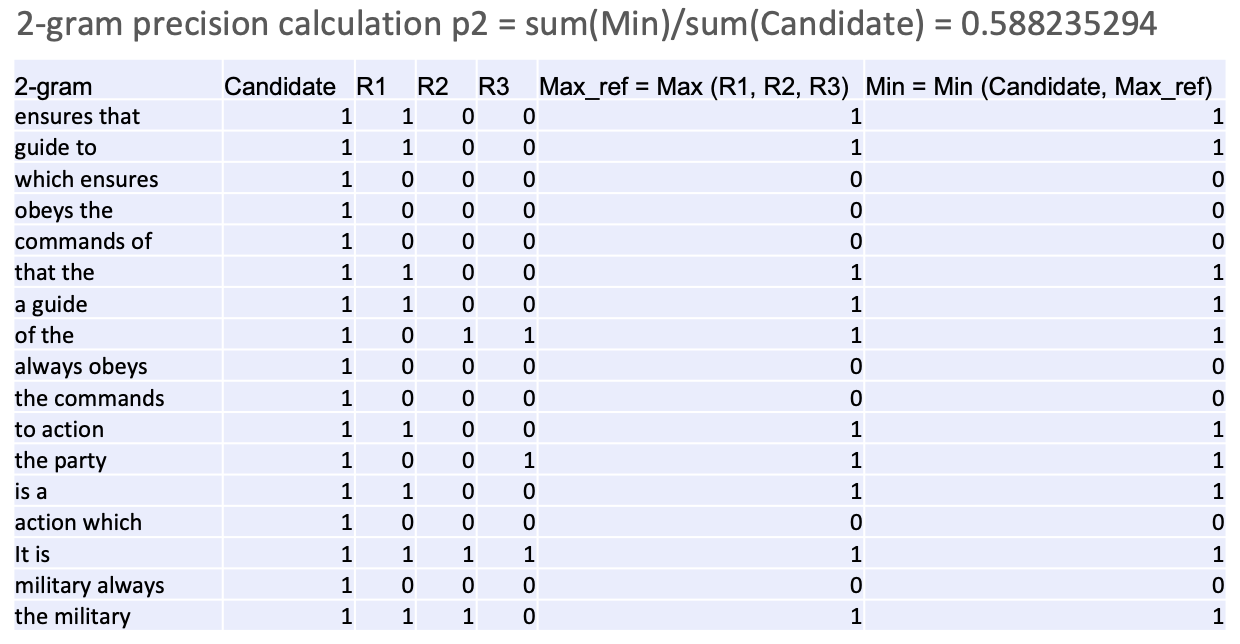
Longer hypotheses have lower scores, so normalized by length.

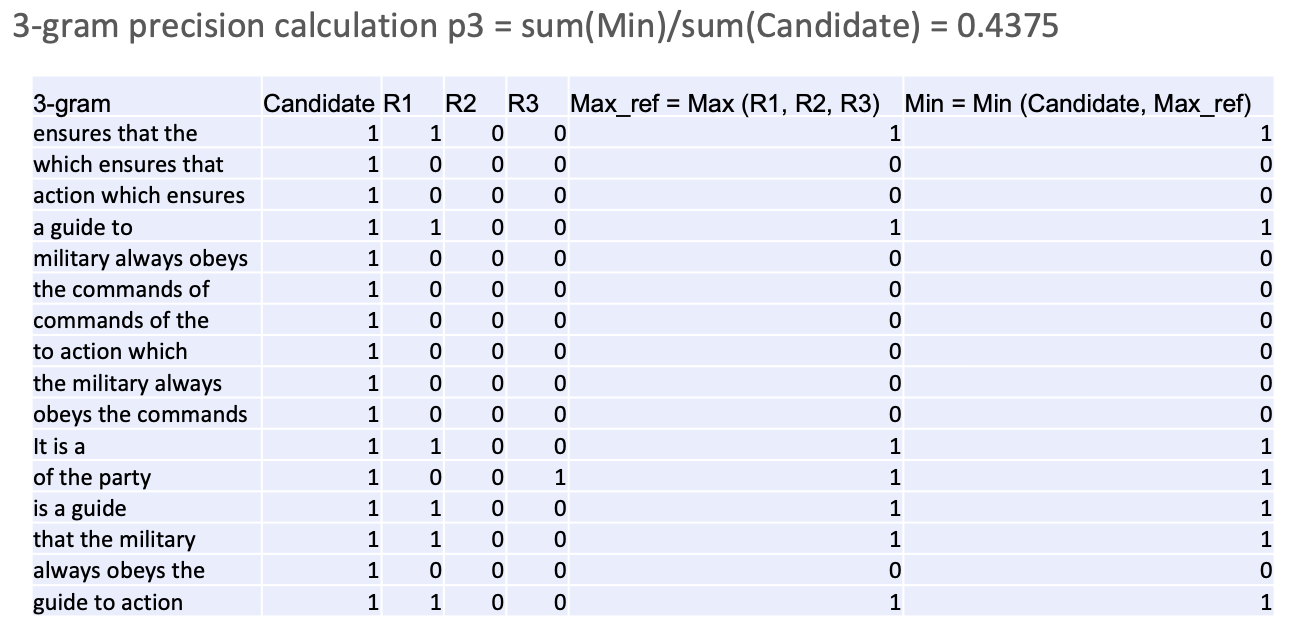
Stopping criteria: continue beam search until: reach timestep 𝑇; have at least 𝑛 completed hypotheses. (T, n all are pre-defined cutoff.)

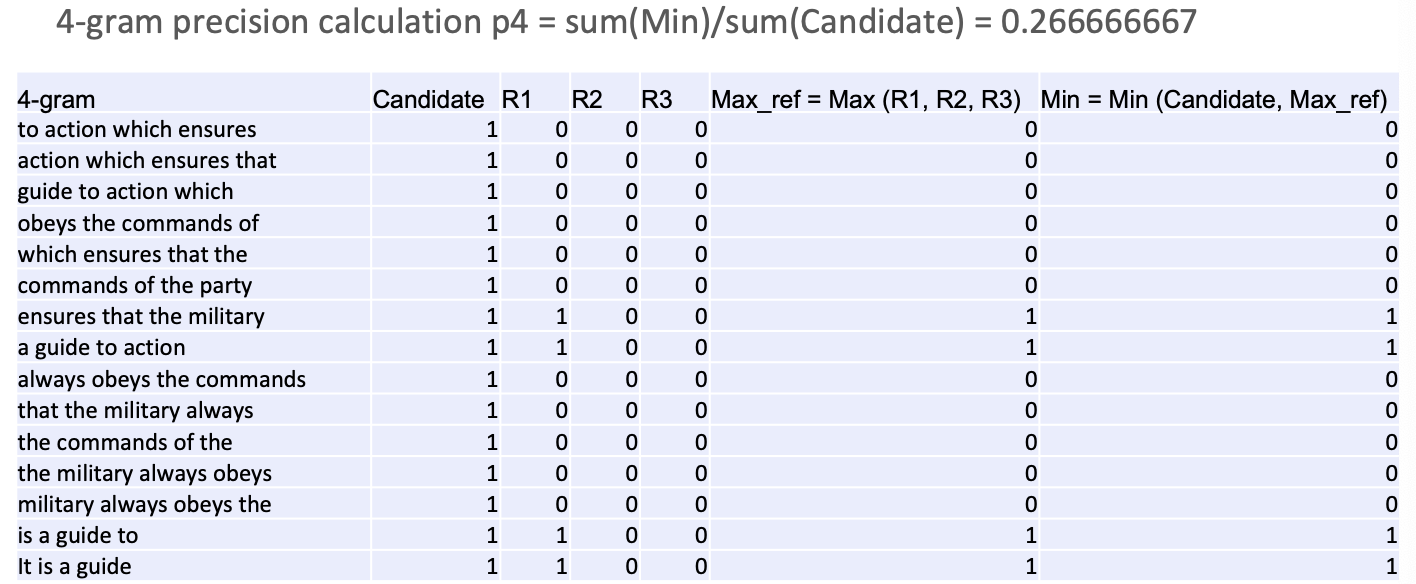
* + - 1. Seq2seq with vs. without attention:
         1. Without attention: conditioned on the same context.
         2. With attention: conditioned on varying context.
    1. Transformer-based MT: Encoder-decoder structure, multi-head self-attention, decoding similar to seq2seq.
  1. Evaluation
     1. BLEU (Bilingual Evaluation Understudy):
        1. Compare the machine-written translation (candidate sentence) to one or several human-written translations.
        2. Computes a **similarity score** based on:
           1. **n-gram precision** 𝑝n
           2. **Brevity penalty** (简短惩罚项) for too-short translations.
        3. Process (Precision calculation, 只从candidate sentence里面找):

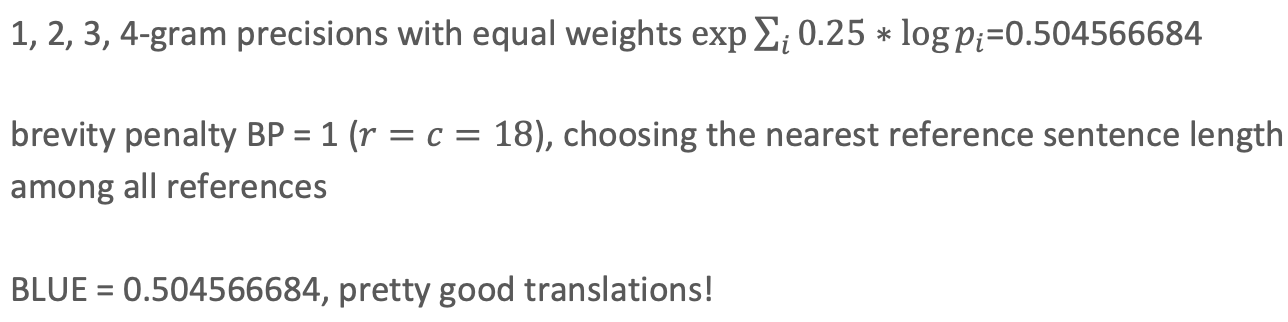


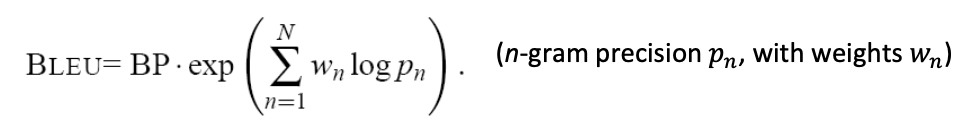




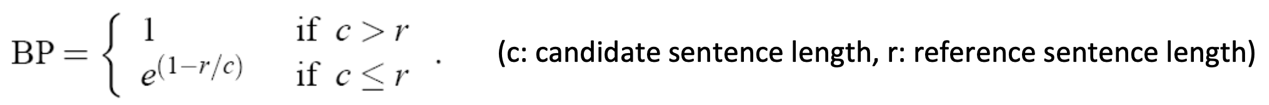






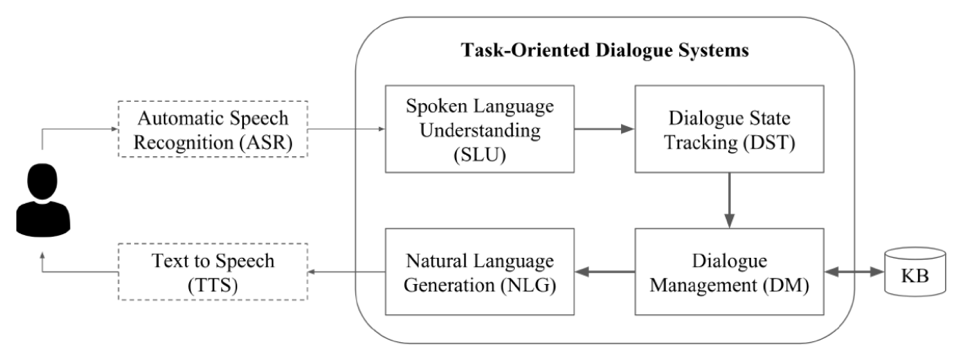


r：参考句子中最接近候选句长度的那个句子的长度



* + - 1. a good translation can get a poor BLEU score because it has low n-gram overlap with the human translation
  1. Remaining challenges in MT
     1. Out-of-vocabulary words
     2. Maintaining context over longer text
     3. Idioms are difficult to translate

1. Dialogue system
   1. Categories of dialog tasks
      1. Task-oriented dialogue systems
         1. open- or close- domain
         2. Aim: **recognize the task** of the user and execute corresponding tasks to **accomplish the goal**.
         3. Example: book a restaurant, book movie tickets.
         4. Task-oriented dialogue systems: ASR (optional) -> SLU (NLU) -> DST -> DM (<-> knowledge base) -> NLG -> TTS (optional)



* + - * 1. NLU in dialogue systems

Domain classification: Requires **predefined domain**.

Intent detection: Requires **predefined schema** (预定义的意图)

Slot filling (槽位填充): Requires **predefined schema**. 从用户输入中抽取结构化的信息

* + - * 1. Dialogue state tracking (DST): Requires **state machine**.
        2. dialogue management (DM) for agent action-Dialogue policy learning (DLP): determine what actions to choose.

Enough information to generate an output

Not enough information? Make a request

Not quite sure? Confirm

* + - * 1. NLG in dialogue systems: Generate natural language or GUI based on the actions selected in the previous step.
      1. Implementation examples
         1. Classification tasks: domain classification and intent detection

RNNs, CNNs, LLMs

* + - * 1. Sequence tagging tasks: slot filling.

RNNs

* + - * 1. Dialogue state tracking: maintain a belief of the dialogue state and update according to observations.

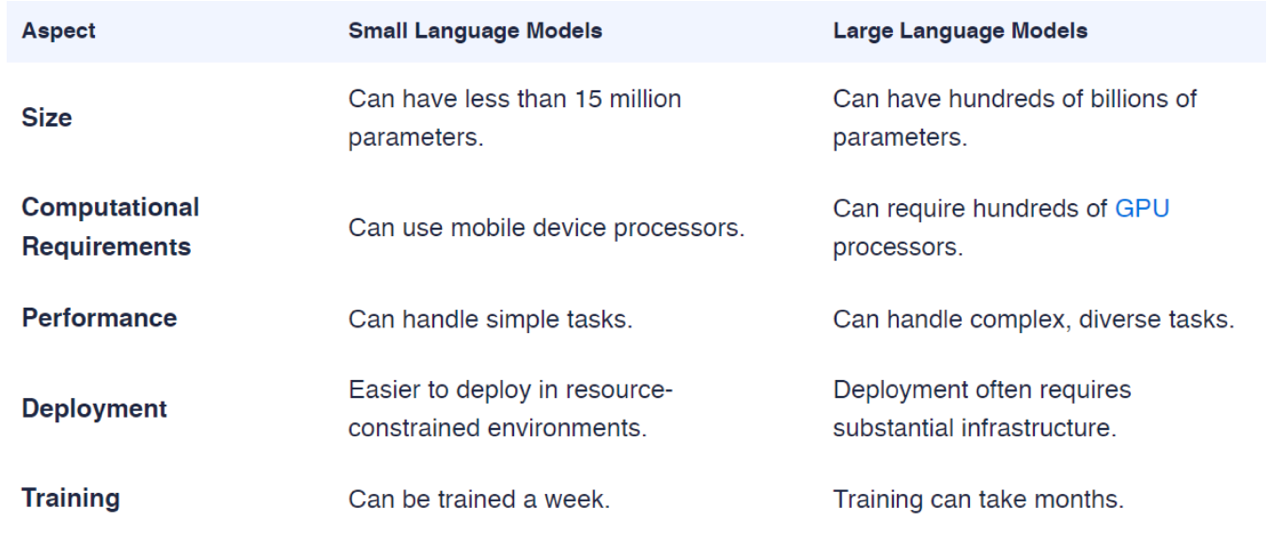
CNN/RNN based tracker

* + - * 1. Dialogue policy learning-reinforcement learning
        2. NLG-seq2seq
    1. Chitchat dialogue systems
       1. Open-domain
       2. Aim: **respond to the user input** in a conversational manner
       3. Example: make social conversations
       4. Learns to generate dialogues from offline dialogue corpora (离线对话语料库). No state, action, intent, slot, etc.
       5. Optimization of the system: with latent variable (潜在变量), with knowledge ground.
  1. Performance evaluation
     1. Human evaluation ideally, but not feasible and expensive.
     2. Heuristic Metrics (启发式指标): BLEU, dialogue length/diversity, fluency, F1 score.

1. Challenges: Knowledge accuracy, Explainability, Privacy and data security.

**Lecture 7: Large Language Models**

1. Large language model definition
   1. LLMs have powerful textual processing abilities: **write and debug computer programs, compose music**.
   2. Comparisons between SLMs and LLMs



* 1. Prompt engineering: A prompt is **a piece of text** inserted in the input examples, so that the original task can be formulated as a (masked) language modeling problem.
     1. Paradigm shift (范式转变): From pretraining-finetuning to prompting.
  2. How large are large language models? **Large model parameter size, large training data, large computing cost**.

1. Scaling law and emergent abilities
   1. Scaling law of LLMs
      1. Scaling laws (扩展规律): simple, predictive rules for model performance.
      2. Performance measures: cross entropy loss.
      3. Resource measures: Model size, dataset size, computing power.
      4. Performance depends strongly on scale, weakly on model shape.
      5. Universality of Overfitting (过拟合的普遍性): 只要模型参数数 N 和数据量 D 一起增长，性能就会稳定提升；但如果只扩增其中一个，效果会递减。

Universality of Training (训练曲线的普遍性): 训练曲线遵循幂律关系 (predictable power-laws)，基本不依赖于模型大小。

* + 1. Sample efficiency: Large models are more sample-efficient than small models. 大模型在样本使用上更高效，达到相同性能所需的训练步数或数据量更少。
  1. Emergent abilities:
     1. zero-shot/few-shot learning: Only training language models, amazingly achieve task-solving ability for other tasks.
     2. Chain-of-thought prompting (思维链提示).

1. Alignment: Language modeling ≠ assisting users——need Instruction finetuning + Reinforcement Learning from Human Feedback (RLHF) to better perform tasks of users.

**Lecture 8: LLM Prompting and Alignment**

1. Prompt Learning
   1. Why prompt learning?
      1. take the same format as the pre-training objectives (预训练任务) and **requires no new parameters**.
      2. Just need to **design a template** and the expected text responses.
      3. **Efficiency for few-shot case** (小样本情况下更高效)
   2. Prompt shape
      1. **Cloze prompts**: fill in the blanks of a textual string
      2. **Prefix prompts**: continue a string prefix
   3. Manual prompt engineering
      1. Def: design prompt templates based on intuition and experience.
      2. Example: LAMA.
      3. Issues:
         1. takes time and experience
         2. difficult to find optimal prompts
   4. Automated prompt searching
      1. **Discrete prompts** (**hard prompts**): templates described in a discrete space.

Example: **Mining-based approach**, **Gradient-based search**.

* + 1. **Continuous prompts** (**soft prompts**): perform prompting **directly in the embedding space** of the model.
       1. Remove two constraints:
          1. Free from the limitations of natural language words.
          2. Get rid of the binding of pretrained model parameters.
       2. **Prefix Tuning**: Add a sequence of continuous task-specific vectors before the input while keeping the LM parameters frozen.
       3. Continuous prompt **initialized with discrete prompts** (**Autoprompt**)
       4. **Hard-soft prompt hybrid tuning**: **insert some continuous embeddings into a hard prompt template** (**P-Tuning**).
  1. **Answer to prompts**-Common choices for the shape of an answer: **Tokens**, **Span** (短语), **Sentence**.
  2. Challenges
     1. Tasks beyond classification and generation
     2. Difficult in Expressing structured information in the form of prompts
     3. The entanglement problem between templates and answers (模板与答案的纠缠问题)
  3. Future developments:
     1. Commercial field: NLP, recommendation systems.
     2. Social field: sentiment analysis, public opinion monitoring.

1. Chain of Thought Prompting
   1. Def: a series of intermediate natural language reasoning steps that lead to the final output.
   2. Strategy:
      1. Self-consistency (自洽性)- By sampling multiple reasoning paths, the paths with consistent answers are identified, thereby enhancing the credibility of the final answer.
      2. Zero-Shot Chain-of-Thought (零样本思维链): Add a simple sentence: "Let's think step by step". The model can be guided to perform step-by-step reasoning and obtain more accurate results.
      3. Automatic Chain of Thought Prompting, Auto-CoT: Through clustering + sampling + self-generated examples, the automatic construction of high-quality thought chain demonstration samples has been achieved.
2. Alignment
   1. Reinforcement learning
      1. Characteristics of RL (make RL different)
         1. No supervisor, only a **reward** signal.
         2. Delayed feedback.
         3. Goal is to maximize the **total rewards**.
      2. Examples: make robot walk, defeat world champion at Go.
      3. RL for LLM Alignment
         1. Challenge: LLMs can generate irrelevant content.
         2. Goal: Ensure outputs aligned with **human values and intentions**.
         3. Solution: Use **reinforcement learning (RL)** to fine-tune models based on feedback.
      4. Policy in RL
         1. Policy: The behavior of an agent in a given state.
         2. Deterministic policy: a = π (s )
         3. Stochastic policy (随机策略): π (a|s ) = P[At = a | St = s ]
         4. The Optimal Policy:
            1. Let agents make decisions in an uncertain environment to **maximize long-term desired rewards**.
            2. Policy Gradient: Use Gradient Ascent (梯度上升) to continuously adjust the policy parameter θ to maximize J(θ).
      5. Three RL Methods for LLM Alignment
         1. PPO (Proximal Policy Optimization, 近端策略优化):
            1. train agents by **optimizing their policies**.
            2. Balances sample efficiency and stability during training.
            3. Core Mechanism: Uses a **clipped objective** (裁剪目标函数) to prevent large policy updates, ensuring smoother learning.
            4. Strength: Proven effectiveness in RLHF.

Weakness: Computationally expensive and complex to implement.

* + - 1. DPO (Direct Preference Optimization, 直接偏好优化):
         1. Align LLMs using human preferences without complex RL.
         2. DPO vs PPO-Why DPO?

PPO: See weakness in PPO.

DPO: Skips reward modeling; More **efficient** and **scalable**; Directly optimizes the policy using pairwise preference data.

* + - * 1. How It Works

Collect **paired** comparisons of model outputs (preferred vs. non-preferred)

Directly optimize the LLM based on preferences.

* + - * 1. DPO in LLM Alignment

Strength: Simplicity; Efficiency; Scalability.

Weakness:

Require high-quality data.

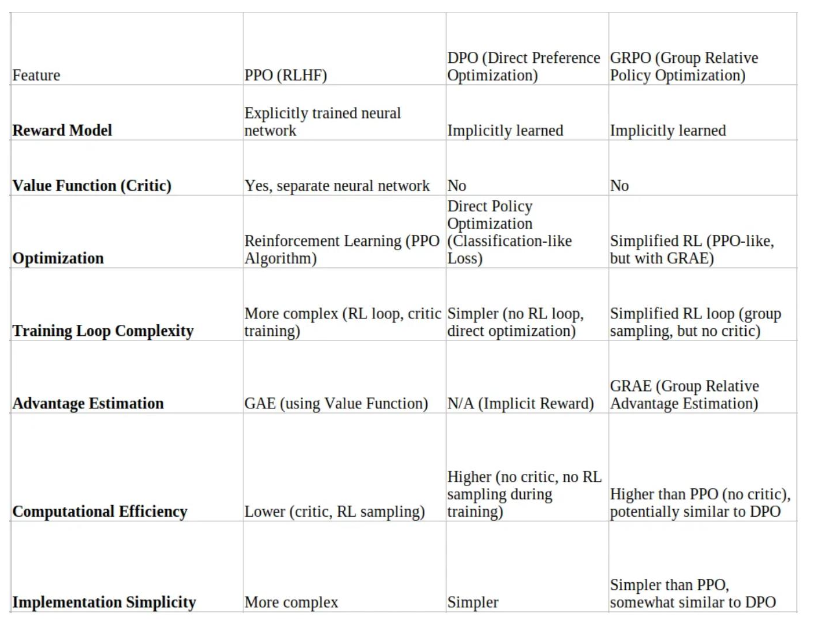
Struggle with unseen scenarios.

May introduce biases.

* + - 1. GRPO (Group Relative Policy Optimization, 群体相对策略优化): A hybrid RL algorithm to enhance reasoning capabilities in LLMs.
         1. GRPO vs PPO-Why GRPO?

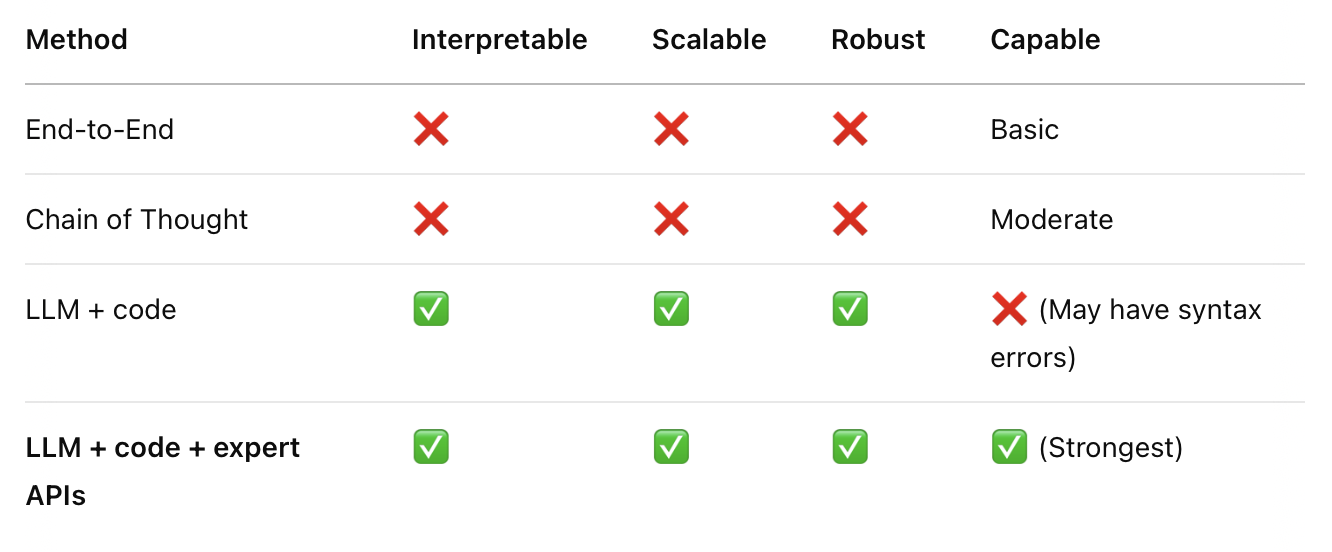
GRPO: No separate **value network** for the Critic (**不需要单独的价值网络**); More **adaptable** and **scalable** for complex tasks.

* + - * 1. GRPO in LLM Alignment-Strength: Stability; Efficiency; Scalability.



**Lecture 9: LLM Agents**

1. LLM agent introduction
   1. Agent: Intelligent system interact with some “environment”
   2. LLM agent:
      1. Level 1: Text agent-Use the form of text to act and observe. (LSTM-DQN)
      2. Level 2: LLM agent-Use LLM to act. (Language Planner)
      3. Level 3: Reasoning agent-Use LLM to reason to act. (GPT)
2. Reasoning
3. Tool learning
   1. LLMs + tool use
      1. executable language grounding
         1. Ground language models into executable actions (将语言模型“落地”为可执行的动作)
         2. Map instructions into code or actions.
   2. Discrimination



* 1. LLM + APIs to search/browser for gathering information (WebGPT)
  2. LLM + webs/apps or personalized functions (ReAct): Let the language model generate "**reasoning traces**" and "actions" alternately to achieve the synergy (协同) between intelligent reasoning and task execution.
  3. LLM + APIs to expert models (HuggingGPT): It acts as a "**task command center**" that connects multiple **AI models available in Hugging Face** to complete complex tasks.
  4. LLM + code, robotic arm, expert models (Code as Policies)
  5. Methods for tool learning
     1. LLM prompting for tool use: In a Program-aided Language model (PAL), a natural language problem are generated as both interleaved (交错的) natural language (NL) and programming language (PL) statements.
     2. LLM + tool use for QA and decision making: ReAct
     3. LLM finetuning/pretraining for tool use (TALM, Toolformer)

1. Knowledge incorporation-RAG
   1. Vector Database:
      1. Embedding: 将所有文档编码为向量
      2. Indexing: 创建向量索引
      3. Querying (Retrieve): 根据用户问题检索相关内容
      4. Post-process: 选出最相关的几个内容块
   2. Generator: LLM.
2. Challenges and future work (LLM Agent Tool Use Field)
   1. Complexity: Complex task adaptability
   2. Interactivity: Multi-round interaction capability
   3. Evaluation
   4. Efficiency

**Lecture 10: Efficient Training of LLMs**

1. Background
   1. Model Weights
      1. The parameters of the model that are learned during training.
      2. Adjusted by the optimizer based on the gradients to minimize the loss function.
   2. Memory consumption
      1. Consider a model with Ψ parameters.
      2. During fp16 training, model parameters need 2Ψ bytes to store the model weights, 2Ψ bytes for the corresponding gradients.
      3. Adam’s Optimizer States:
         1. fp32 copy of parameters: 4Ψ
         2. fp32 copy of momentum:4Ψ
         3. fp32 copy of variance: 4Ψ
         4. In total: 4Ψ + 4Ψ + 4Ψ = 12Ψ
      4. Activation (intermediate results)
         1. The intermediate results stored from forward pass to perform backward pass.
         2. The activation memory of a transformer-based model is proportional to the number of

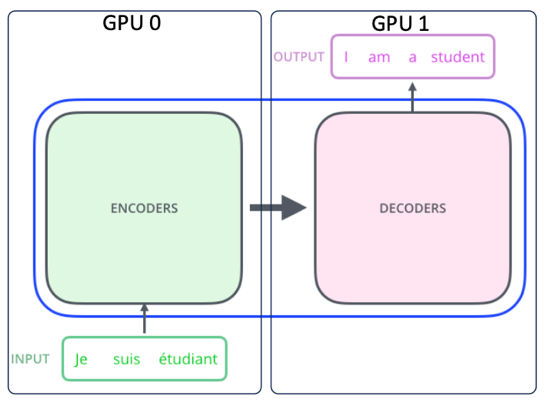
𝑡𝑟𝑎𝑛𝑠𝑓𝑜𝑟𝑚𝑒𝑟 𝑙𝑎𝑦𝑒𝑟𝑠 × ℎ𝑖𝑑𝑑𝑒𝑛 𝑑𝑖𝑚𝑒𝑛𝑠𝑖𝑜𝑛𝑠 × 𝑠𝑒𝑞𝑢𝑒𝑛𝑐𝑒 𝑙𝑒𝑛𝑔𝑡ℎ × 𝑏𝑎𝑡𝑐ℎ 𝑠𝑖𝑧𝑒.

* + 1. Practical: Model weights take 2 bytes (16 bits) at fp16 and 4 bytes (32 bits) at fp32. Consider a 1.5B parameter GPT-2 model requires 16 \* 1.5 = 24GB of memory for its model weights, gradients, and optimizer states in 16-bit precision.

16 \* 1.5 = 24GB + activation memory > 32GB

1. Model Parallelism (模型并行)
   1. Naïve MP
      1. Def: Vertically partition the different layers of a neural network model and distribute them across multiple GPUs. 把一个神经网络模型的不同层纵向拆分
      2. data travels from layer inside Encoders, this is just the normal model.

data needs to pass from encoder to decoder it needs to travel from GPU0 to GPU1 which introduces a communication overhead.



* + 1. Adv: fit very large models onto limited hardware

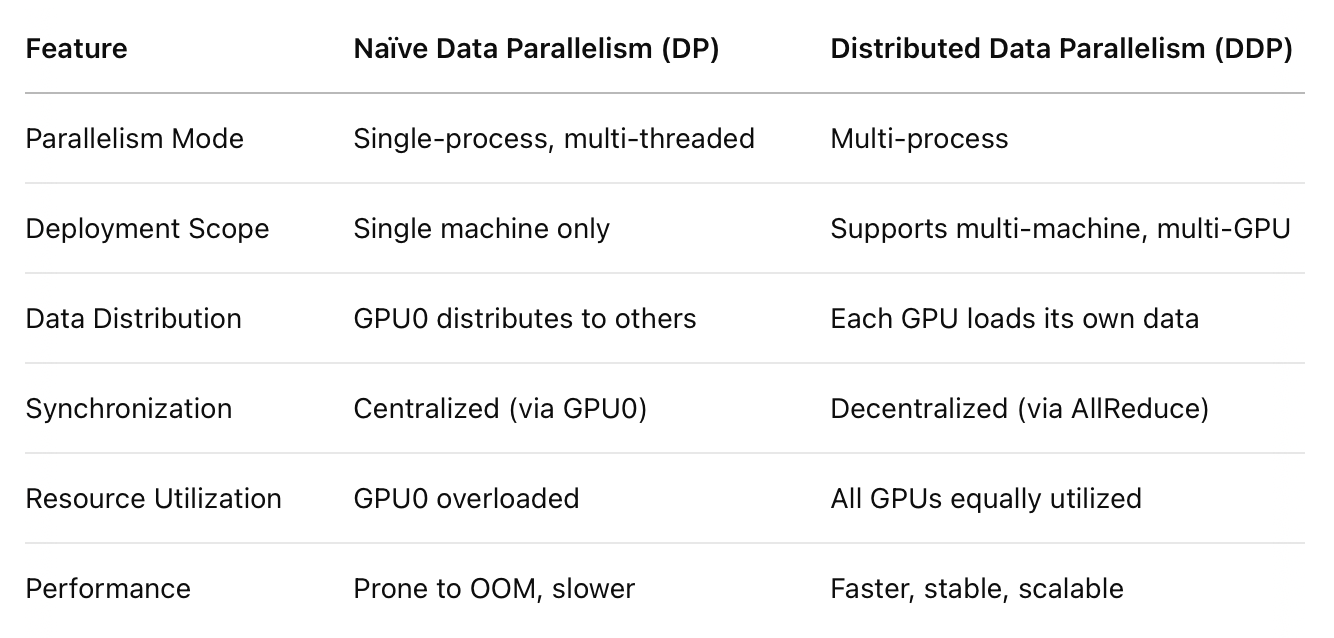
Dev: GPU may be idle sometimes.

* 1. GPipe
     1. Split the entire model into multiple cells, each cell on a different GPU; Divide one batch into multiple micro-batches; pipeline the execution of each set of micro-batches over cells.
     2. Adv: Less bubble, make the most use of GPUs.

Dev: Need to find the size of micro-batch that leads to the highest efficient utilization.

* 1. Modern MP-Tensor Parallelism (张量并行)
     1. The tensors within a single layer are split row/column to multiple GPUs, and each GPU only processes a part of the tensor.
     2. TP for MLP (多层感知机) Layer: Split the large matrix in the MLP layer into small pieces, let multiple GPUs process part of the data, and finally merge the results.
     3. TP for self-Attention-Why self-attention easier? Each Head is calculated independently.
     4. Adv: Simple and efficient, Excellent scalability, Support large models.
     5. Dev: Need a very fast communication network, Parallelism is limited by the number of Gpus.

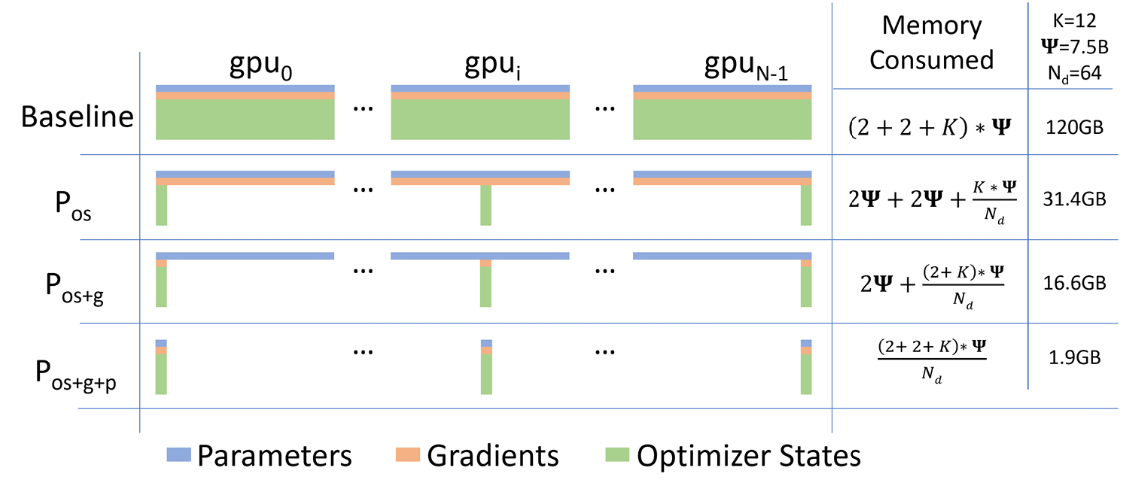
1. Data Parallelism (数据并行)
   1. Naïve DP
      1. Each device will **hold a full copy of the model replica** (模型副本).
      2. Naïve DP vs distributed data parallel



* + 1. Why memory consumption high in Naïve DP? DP requires each GPU to **replicate the full model params, gradients and optimizer states**.
  1. DeepSpeed ZeRO
     1. Each GPU stores only a slice of the full model replica. All GPUs synchronize to give each other parts that they miss at run time.
     2. Optimization: Optimizer State Partitioning, Gradient Partitioning, Parameter Partitioning.

总显存 = ( 2 + 2 + K ) × Ψ

2Ψ 参数 + 2Ψ 梯度 + KΨ 优化器状态（Adam K=12）。



1. Parameter-Efficient Fine-Tuning (参数高效微调)
   1. Adapter:
      1. An Adapter adds a small bottleneck projection + activation inside each layer.
      2. Adv: Achieve comparable accuracy to full model fine-tuning. (效果接近全模型微调)

Dev: Have to be processed sequentially; Rely on hardware parallelism.

* 1. LoRA-Different from the adapter, Adapter is added in series behind the Transformer layer, LoRA is added in parallel to the original weight path with no extra inference latency. Adapter 是 串联添加 到 Transformer 层后面, 而 LoRA 是 并联添加 到原始权重路径。

**Lecture 11: Retrieval Augmented Generation (RAG)**

1. What’s RAG?
   1. Def: A paradigm (范式) that enhances LLMs by integrating external knowledge bases.
   2. Why RAG?
      1. Tackle with up-to-date and long-tail knowledge; (长尾知识指的是训练数据中很少出现但用户可能会问到的知识。)
      2. No retraining for task-specific applications;
      3. Reduce Hallucination (幻觉).
   3. Where be used?
      1. Long-tail distribution of data.
      2. Frequent knowledge updates.
      3. Specialized domain knowledge.
   4. RAG architecture
      1. Vector Database:
         1. A vector database is a kind of database that stores information in the form of vectors.
         2. Vector embeddings: Numerical representations of data objects.
         3. Different from a vector search library or vector index: A complete data management system that enables data storage and filtering, allows for dynamic data changes.
         4. Process:
            1. Embedding

**Loader** loads the document

**Splitter** splits documents

**Embedding machine**: Convert the text into vector embeddings.

* + - * 1. Indexing

Three main indexing techniques: **Hashing**, **Quantization** (量化), **Graph-based** (基于图).

Indexing optimization: Chunk Optimization, Adding Metadata, Metadata Filtering.

* + - * 1. Querying (Retrieve)
        2. Post-process
    1. Generator: LLM.
  1. Naïve RAG: Indexing Retrieval Generation

Advance RAG: **Index Optimization** **Pre-Retrieval Process** Retrieval **Post-Retrieval Process** Generation

1. What / When / How to retrieve? (Main issues)
   1. What to retrieve?
      1. Token
      2. Phrase (短语)
      3. Chunk
   2. When to retrieve?
      1. Single search (在生成前或中只检索一次): **High efficiency** but **low relevance**.
      2. Each token
      3. Every N tokens: low efficiency and **redundant** information.
      4. Adaptive search (根据不确定性动态决定是否检索): Balance **efficiency** and **information**.
   3. How to use the retrieved information
      1. Input/Data Layer (作为输入拼接给 LLM): cannot support **more knowledge blocks**; limited **optimization space**.
      2. Model/Intermediate Layer (注入到模型中间层): introduces **additional complexity**; **must be trained**.
      3. Output/Prediction Layer (影响最终生成阶段): **highly relevant** to the retrieval content.
   4. Example: kNN-LM

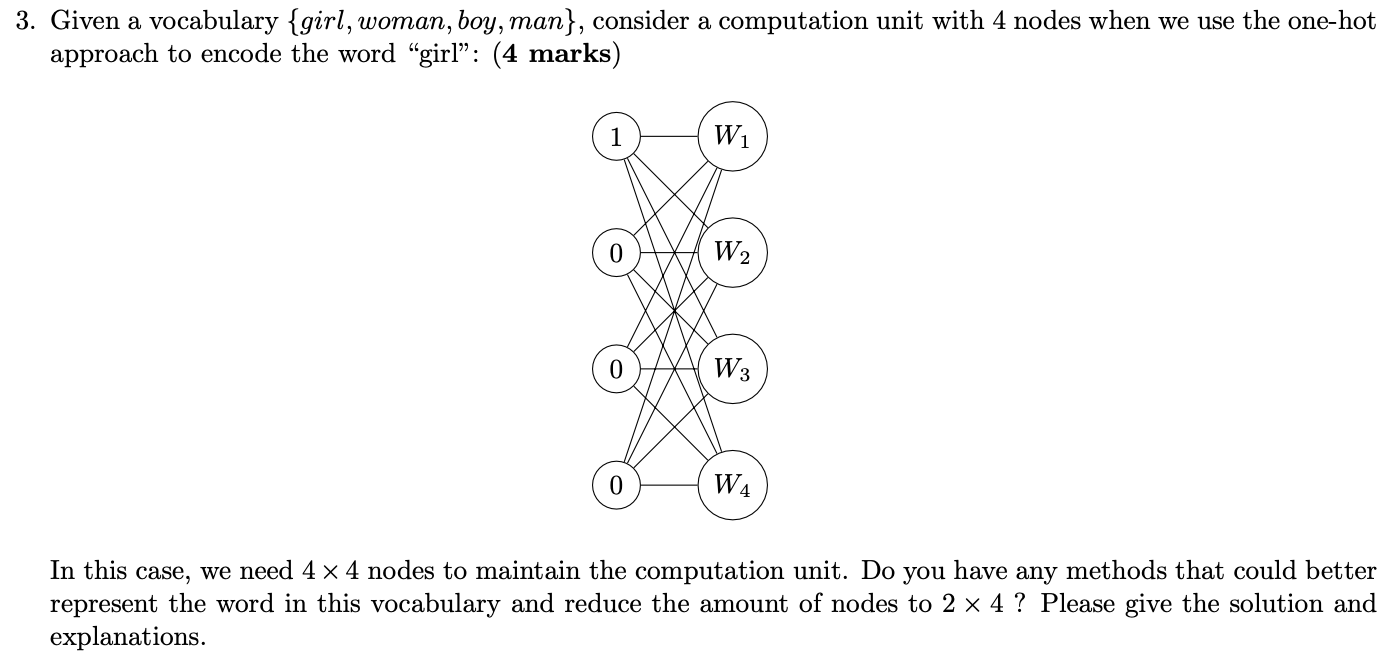


* 1. Evaluation
     1. Methods: **Independent Evaluation, End-to-End Evaluation.**
     2. Key Metrics & Capabilities:
        1. Key Metrics: Answer Relevance, Context Relevance (检索内容是否与提问匹配), Answer Fidelity (回答是否基于检索内容？是否真实).
        2. Key Capabilities:
           1. Noise Robustness,
           2. Negative Rejection (没有检索到知识时能否拒答),
           3. Info Integration (能否整合多个文档中的信息来回答复杂问题),
           4. Counterfactual Robustness (是否能识别错误或虚假的检索信息)
     3. Assessment Framework:
        1. TruLens, RAGAS (Based on handwritten prompt)
        2. ARES: based on several objects.

1. RAG Tools
   1. Integration Frameworks: LangChain.
   2. RAG in LLMs (集成了 RAG 功能的大语言模型平台): OpenAI.
   3. Vector Databases: deepset.
   4. RAG Frameworks & Libraries: FARM.

Assignment1:

1. Drawbacks of one-hot encoding:
   1. The dimension of the embedding linearly increases with the vocabulary size.
   2. Not involve semantic features.
2. 1-3 picture:



Represent the words as {00, 01, 10, 11}.

1. the disadvantages of n-gram model:
   1. Sparse feature space;
   2. Only suitable to the large training dataset;
   3. Cannot interpret unseen words;
   4. Sensitive to the hyper-parameter `N`.
2. Disadvantages of splitting by spaces to tokenize text:
   1. Very large vocabulary size;
   2. Cannot deal with out of vocabulary words;
   3. Cannot capture the semantic relations between similar words.