Cheat Sheet (Abridged)

Perplexity

How confused the model is

Lower is better

$$P(w) = P(w_1 w_2, ...w_N)^{-\frac{1}{N}}$$

$$P(w) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1...w_{i-1}}}$$

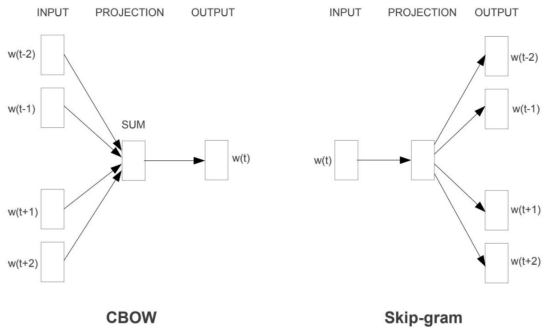
Word2Vec

Binary classifier - “Is word x likely show up near word y”

Words are fed in as one-hot vectors

Uses both positive examples and random sampling for negative examples

learned weights are used as embeddings



Skipgram: predicts the surrounding context words given a target word c_i and w are both embedding being learned

$$P(+|w, c_{1:L}) = \prod_{i=1}^{L} \sigma(c_i \cdot w)$$

$$P(+|w, c_{1:L}) = \prod_{i=1}^{L} \sigma(c_i \cdot w)$$

CBOW: predicts a target word based on the surrounding context words.

RNNs

One hot vector → Embedding → hidden state → output

$$e_t = Ex_t$$
$$h_t = g(Uh_{t-1} + We_t$$
$$\hat{y}_t = \text{softmax}(E^T h_t)$$

Bidirectional RNNs

$\oplus$ is the vector concatenation

$$h_t^f = \text{RNN}_{\text{forward}}(x_1, ...x_n)$$
$$h_t^b = \text{RNN}_{\text{backward}}(x_1, ...x_n)$$
$$h_t = h_t^f \oplus h_t^b$$

LSTM

Manages the issue of losing distant information and the vanishing gradients problem

Hidden state vector h_t for **short term memory** and **context** vector c_t for **long-term memory**

Masked Language Model

During training tokens are replaced with [MASK] token and model must predict the missing token

Analyze the text bidirectionally

BERT (Bidirectional Encoder Representations from Transformers)

Integer token representation.

Integer is used to index into embedding matrix

bidirectional self-attention (each token can attend to all other tokens in the sequence)

Pre-training

- **Masked Language modeling**: predicting what the missing token is. Calculates probability distribution of possible tokens and objective is cross-entropy or NLL loss
- **Next Sentence Prediction:** Predict if the sentence after the [SEP] token actually comes after the first sentence and objective is cross-entropy or NLL loss

  [CLS] Token: Carries information about entire sentence and used for classification

RLHF

1. Have humans rank responses y_i to an input x
2. Train a reward model
3. Use reward model to fine-tune the LM with Proximal Policy Optimization (PPO) which has objective to maximize the reward (Back-propagation with Gradient Ascent)

R_θ: Reward function

ϕ_{RM}: Parameters for reward model. (defines the objective for how it is updated)

θ_{RLHF}: parameters of RLHF model (defines objective for RLHF model (PPO))

y_w winner response

y_l loser response

$$R_\theta = \left\{ \left(x^{(i)}, y_w^{(i)}, y_l^{(i)} \right) \right\}_{i=1}^{N}$$
$$\phi_{RM} = \arg\max_{\phi} \mathbb{E}_{(x,y_u,y_o)\sim R_\theta} \left[\log \sigma \left(r_\phi(x, y_w) - r_\phi(x, y_l) \right) \right]$$
$$\theta_{RLHF} = \arg\max_{\theta} \mathbb{E}_{x\sim I} \left[\mathbb{E}_{y\sim P_\theta(\cdot|x)} \left[r_\phi(x, y) - \beta \log \frac{P_\theta(y|x)}{P_{SFT}(y|x)} \right] \right]$$

DPO

Make the language model its own reward function and by fine-tuning on the following loss function

π_θ: is new language model

π_{ref}: is original language model

β: weight on human preferences higher β means prioritize human responses

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right) \right.$$

Loss

NLL Loss: Used in classification tasks, especially in combination with a softmax output layer.

$$\text{Binary Classification}$$
$$\text{NLL Loss} = -\log(p_y)$$
$$p_y \text{ is the predicted probability of the true class y}$$

Cross-Entropy Loss: When NLL loss is used for multi-class classification problems.

Gates are [0,1] masks that help determine what the context should remember and forget

Forget Gate:

$$f_t = \sigma(U_f h_{t-1} + W_f x_t)$$
$$k_t = c_{t-1} \odot f_t$$
$$\text{where } \odot \text{ is the Hadamard product (element wise multiplication)}$$

Add gate:

$$g_t = \tanh(U_g h_{t-1} + W_g x_t)$$
$$i_t = \sigma(U_i h_{t-1} + W_i x_t)$$
$$j_t = g_t \odot i_t$$

Context Update:

$$c_t = j_t + k_t$$

Output gate:

$$o_t = \sigma(U_o h_{t-1} + W_o x_t)$$
$$h_t = o_t \odot \tanh(c_t)$$

Encoder - Decoder

1. **encoder:** Gets contextualized representations(LSTMs, CNNs, Transformers)
2. **context vector:** Represents the essence of the input
3. **decoder:** Accepts c as input and generates a new arbitrary length sequence can be any sequence architecture

Attention

Used in Encoder-Decoder, solution to bottlenecking problem where all information is required to be in the last vector (the context vector)

Allow the decoder to get information from all the states

Context vector is a function of hidden states of the encoder $c = f(h_1^e ... h_n^e)$

Decoder generates a new context vector for each step $h_t^d = g(\hat{y}_{i-1}, h_{i-1}^d, c_i)$

score computation:

$$\text{dot product:} \quad \text{score}(h_{t-1}^d, h_j^e) = h_{i-1}^d \cdot h_j^e$$
$$\text{learned :} \quad \text{score}(h_{t-1}^d, h_j^e) = h_{t-1}^d W_s h_j^e$$

c_i computation:

$$\alpha_{ij} = \text{softmax}(\text{score}(h_{i-1}^d, h_j^e))$$
$$c_i = \sum_j \alpha_{ij} h_j^e$$

Cross-Attention

Allow the decoder to query on encoder keys

Input: $H^{enc} = h_1, ...h_n$ where H^{enc} is shape $[n \times d]$

$$Q = H^{dec[l-1]} W^Q$$
$$K = H^{enc} W^K$$
$$V = H^{enc} W^V$$

$$\text{CrossAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

d_k is dimensionality of the keys (W^k is $[d \times d_k]$)

PMI

Quantifies how much more often two words co-occur than would be expected by chance, based on their individual probabilities.

$$\text{PMI}(x, y) = \log\left(\frac{P(x, y)}{P(x) \cdot P(y)}\right)$$

x and y are the two words

$$\text{Multi-Class Classification}$$
$$\text{Cross-Entropy Loss} = -\sum_{j=1}^{C} y_j \log(p_j)$$
$$C \text{ is the number of classes}$$
$$y_j \text{ is the true label for class jjj (1 if class j is the correct class, 0 otherwise)}$$
$$p_j \text{ is the predicted probability for class j}$$

Definitions

Autoregressive model: a model that predicts a value at time t based on a linear function of the previous values at times t −1, t −2, and so on

Language Model: A model that estimates the probability of a sequence of words in a language, assigning a probability to any possible sequence of words within a fixed vocabulary

Bag of Words: simplifies text into a collection of individual words (or "tokens") without considering the order or structure of the words, focusing purely on word frequency.

N-Gram: Predicts the probability of a token given the last n-1 words

Transformers

Neural network architecture designed to handle sequential data using **self-attention**

GPT-2

Transformer decoder model only

Masked self-attention, meaning each token can only attend to previous tokens (left-to-right)

1. Input is tokenized into integers
2. Integers used to look up embeddings in the transformer's embedding matrix
3. Positional encoding function used to remap embeddings to positional embeddings
4. Stacks of Self attention and feed forward neural nets
5. Softmax final layer to predict next token

Self-Attention

For each token in a sequence determining how much focus should be placed on other tokens in the same sequence

$$Q = XW^Q$$
$$K = XW^K$$
$$V = XW^V$$
$$\text{Self-Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

Other Representations

TF-IDF

Term Frequency (TF): Measures how frequently a term appears in a document. The more frequent, the higher the TF.

Inverse Document Frequency (IDF): Measures how unique or rare a term is across all documents. Terms that appear in fewer documents have a higher IDF.

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$
$$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$
$$\text{IDF}(t) = \log\left(\frac{N}{1 + |\{d \in D : t \in d\}|}\right)$$

t: is the term (word), d: is the document, N: is the total number of documents

$|\{d \in D : t \in d\}|$: is the number of documents containing term t