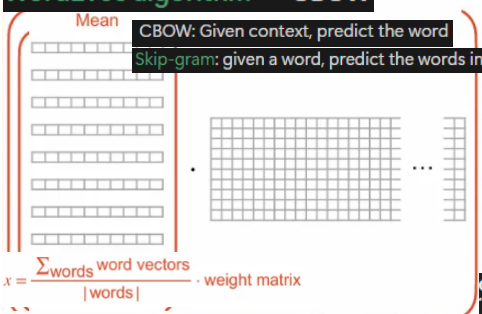
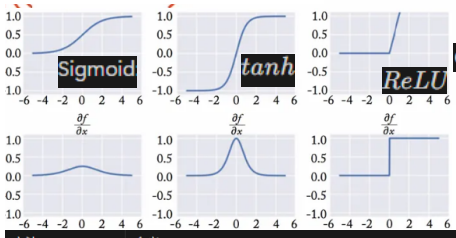
Word2vec algorithm — CBOW



CBOW: Given context, predict the word

Skip-gram: given a word, predict the words in its context

$$x = \frac{\sum_{words} \text{word vectors}}{|words|} \cdot \text{weight matrix}$$



方法	含义
IO	I: Inside (Inside Entity), O: Outside (Outside Entity)
BIO	B: Begin, I: Inside, O: Outside
BIOES	B: Begin, I: Inside, O: Outside, E: End, S: Single (singel Entity)

micro-P

Many TN

Some labels are less common and we care about them in proportion to their frequency.

Possible outputs = | Labels | * Length * Length

5 keys components of NLP

Data
- not every data can be easily split

Model
- According to input get corresponding output

Inference Method — find a high scoring option
- Determine what the result is based on the result generated by the Model

Metric
- Loss function

Learning Method
- How to update model

context window

"Small context (+/- 2 words) → more syntactically"

"Larger context (+/- 5 words) → more semantically"

Overfitting Solution

1. Modify loss with cost.
   - L1 driving unimportant feature weights to zero
   - L2 penalizes large weights; keep weights small
2. during training, randomly set some weights to zero Dropout
3. don't backpropagate error further once it gets small

Early stopping

Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

- If your goal is to maximize the total number of correct predictions, regardless of class distribution, then micro averaging is more appropriate.
- If you are concerned about performance on rare or minority classes, then macro averaging is a better fit because it treats all classes equally.

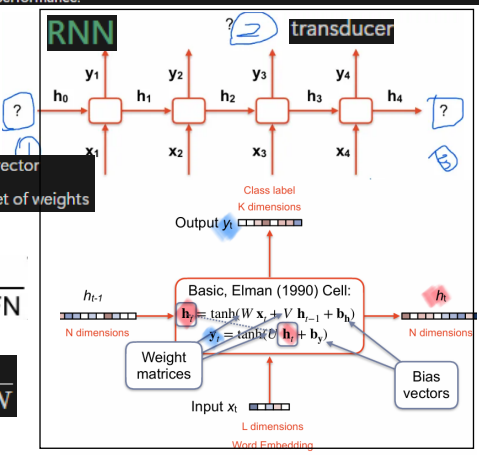
$$Y_{micro} = \frac{\sum TP}{\sum (TP + Y)} \quad X_{macro} = \frac{\sum X}{Number\ of\ classses}$$

when there is no 'None' class $P_{micro} = R_{micro}$

The closer the ROC curve is to the top-left corner (i.e., high TPR and low FPR), the better the model's performance.

AUC (Area Under the Curve) measures the model's ability to distinguish between positive and negative samples. A higher AUC indicates better separability.

The closer the Precision-Recall Curve (PRC) is to the top-right corner, the better the model's performance.



a. All zero vector

b. learn a set of weights

True Positive Rate

$$\frac{TP}{TP + FN}$$

False Positive Rate

$$\frac{FP}{FP + TN}$$

n represents the number of input tokens k represents the number of labels

Exhaustive — k^n

Greedy — $k * n$

Make choice one at a time.

Sometimes the answer can match the Exhaustive method. For example, if every part of the output is independent.

Top-1 At each step, choose the highest scoring option

Random — full distribution

At each step, choose using random sampling from the probability distribution.

Top-K

At each step, filter to the top K options, then choose using random sampling from the probability distribution.

The probabilities need to be redistributed according to the proportion

Top-P

At each step, filter to the options that cover P% of the probability distribution, then choose using random sampling.

Contrastive

At each step, adjust scores based on similarity with recent outputs, then choose the highest scoring option.

Method	Advantages	Disadvantages
Top-1 (Greedy)	- Deterministic: same input always gives the same output - Fast and simple	- Only considers locally optimal choices - May miss better global sequences - Tends to produce repetitive or generic output
Random	- Can escape local optima - Encourages high diversity	- May select low-probability, low-quality outputs - Results often incoherent or meaningless
Top-k	- Avoids very unlikely tokens - Good for problems where local choices don't lead to global optimum	- Still randomly selects among top *k*, may pick suboptimal tokens - Total probability mass of top *k* may be low, reducing output quality
Top-p (Nucleus)	- Dynamically selects candidates based on cumulative probability ≥ *p* - Avoids small-probability regions more safely than Top-k	- More computation due to sorting and accumulation at each step - Might exclude rare but meaningful tokens in skewed distributions
Contrastive Search	- Balances diversity and coherence - Reduces repetition - Produces fluent and high-quality text	- More complex to implement - Depends heavily on model quality and contrastive scoring - Computationally expensive

Beam Search — $k * m * n$

Beam Search maintains the **top *m* highest-scoring sequences** (called the beam width).

For each of these *m* sequences, it explores all possible next tokens.

It then selects the **new top *m* sequences** with the highest overall scores (including past steps).

This process continues until the end of the sequence.

How to select the m answer? Usually m is small enough that any of these are fine

When candidates have **very similar scores**, Beam Search may struggle to differentiate

Increase beamwidth m

Simple Approach — $O(nm)$
- For each option, go through the list to find where it goes

Heap Approach — $O(nlogm)$
- Update it with each new item

Quick Select — $O(n)$

Advantages of Viterbi Algorithm

Finds the globally optimal sequence (maximum joint probability)

Uses **dynamic programming** to efficiently explore all possible label sequences without enumerating

Viterbi algorithm Dynamic Programming

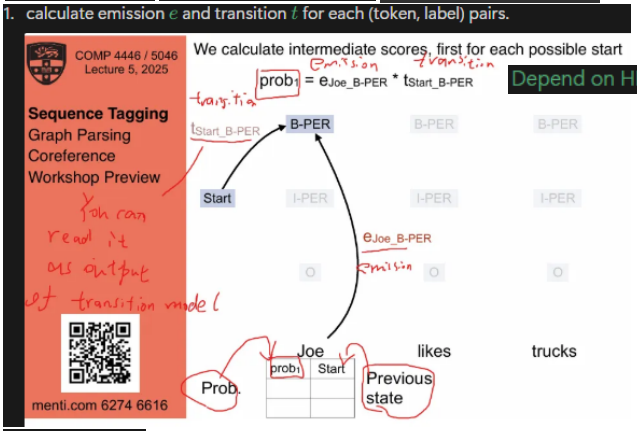
1 previous label $O(|words| * |labels|^2)$

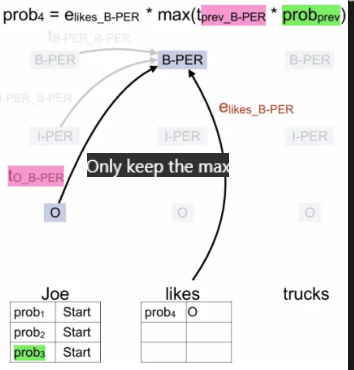
2 previous label $O(|words| * |labels|^3)$ Emission Does not change

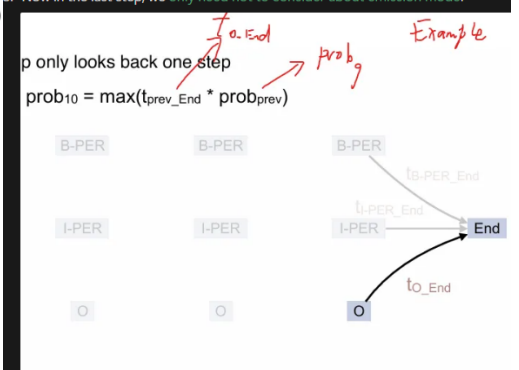
we can use in any model as long as it has previous input and current input

Linear moder (e.g. CRF) Feedforward Networks RNNS Transformers

1. calculate emission e and transition t for each (token, label) pairs.
2. Next token do the same thing, but they need to use previous label and corresponding p_i
3. Now in the last step, we only need not to consider about emission model







CKY algorithm

Adding Arcs: Item + Arc = Result

Combining Items: Left + Right = Result

no path from left to middle + indirect path from right to middle = no path from left to right

Only keep the highest possible result

$$O(|rules_{comb}||words|^3 + |rules_{arc}||words|^2)$$

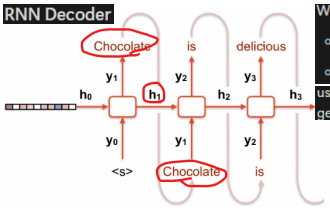
Combinations:

Each start point	$O(\|words\|)$
Each length	$O(\|words\|)$
Each split point	$O(\|words\|)$
Each rule	$O(\|rules_{comb}\|)$

Arc creation:

Each start point	$O(\|words\|)$
Each length	$O(\|words\|)$
Each rule	$O(\|rules_{arc}\|)$

RNN Decoder

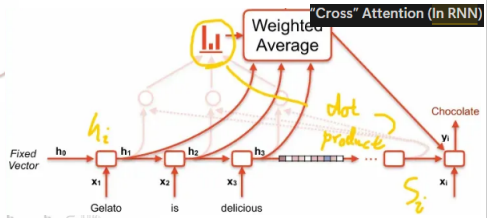


We can also use Greedy Inference with an sequence to sequence model

- How to stop beam search encoder-decoder?
  Keep going until we have N outputs.
- How to score diffrent output with different length?
  usually longer output will have higher length, so we need to normalize the score, to get the average probability per word

We will solve this with 'attention'

Give each output step
a representation of the input
that is most useful for that output decision



Hidden vectors from input $h_1, h_2, h_3 \dots h_N \in \mathbb{R}^{d_h}$

Hidden vector for current output step $s_i \in \mathbb{R}^{d_s}$

Calculate attention scores $e_i = [s_i^T h_1, s_i^T h_2, s_i^T h_3, \dots s_i^T h_N] \in \mathbb{R}^N$

This is dot product attention

Normalise s ax$(e_i) \in \mathbb{R}^N$

Calculate weighted average $a_i = \sum_{j=1}^{N} \alpha_{i,j} h_j \in \mathbb{R}^{d_h}$

chrF — compare character ngrams

We compare TP/TN/FP/FN for all n-gram
FP = FP-1-gram +...+ FP-4-gram

$$F_\beta\text{-score} = \frac{(1+\beta^2)TP}{(1+\beta^2)TP + \beta^2 FN + FP}$$

smaller β means colser to Precision
larger β means closer to recall

Robust to typos

BLEU — compare word ngrams

Translation quality

1. Calculate precision for unigrams, bigrams, trigrams, 4grams separately
2. Take the geometric mean
3. Apply a penalty if the output is short

Strongly depends on word order
no 4grams matches means a score of 0!

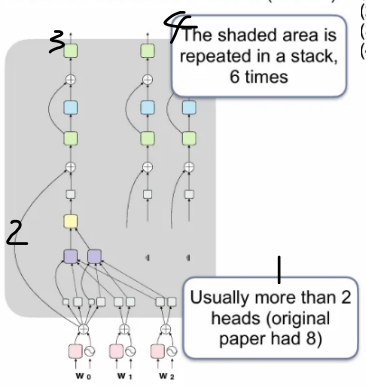
What if sentence cannot be split on whitespace to get token?
start with small units, then combine units

Byte-Pair Encoding (BPE)
1. Set vocabulary to be all characters
2. Find the two vocabulary items that are adjacent most frequently,
3. Create a new vocabulary item for that pair and update data
4. Check if the vocabulary is size K (e.g., 100,000). If not, go to 2.

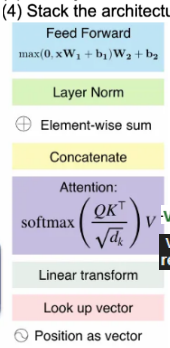
WordPiece Consider all possible vocabulary pairs, then Choose the one that will decrease perplexity most if added to the model

start with big and small units, then delete units | more meaningful subword units
1. Set vocabulary to be all characters and all frequent character sequences, up to full words. 2. Find vocabulary items to remove using a complex method.
3. Repeat until the vocabulary gets down to the desired size.

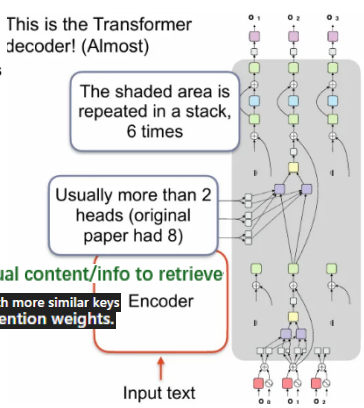
This is the Transformer encoder! (Almost)



(1) Do attention multiple times at once
(2) Add residual connections
(3) Add layer normalisation
(4) Stack the architecture multiple times



This is the Transformer decoder! (Almost)



vector with actual content/info to retrieve
Values associated with more similar keys receive higher attention weights.
Encoder

Layer normalisation helps stabilise training by keeping the output of each layer at a consistent scale (mean and standard deviation are consistent between layers).

Fast Train Internal Covariate Shift

slight accuracy gains, big efficiency gains.
Faster training means they can scale

BERT Encoder
GPT-1 Decoder

K and V come from encoder
Q comes from decoder

Scaling Laws
estimate the best model size given a certain amount of data and a budget for training

Attention is quadratic, is that a problem?
Lots of clever variants to avoid it!
- PerFormer
- LongFormer
- LinFormer

memory bandwidth

Encoder — BERT, ELECTRA

- masked-token prediction — ***BERT*** Ignore output for unmasked words
- span prediction — SpanBERT mask subword of word or a countinuous token set
- **random-token correction — BERT** correct wrong token (model is not told which token was changed)
- combination of masked-token + random-token
- token edit detection Edits are not random use a small LLM to generate hard edits
- next sentence prediction — BERT judge next sentence (NSP) task.

RoBERTa outperforms BERT by training longer, using more data, and removing the

Encoder-Decoder — T5, BART

- masked sequence prediction — ***BART*** mask might include multiple token
- masked span prediction — ***T5*** mask cannot be empty, include multiple token
- deleted sequence prediction — BART or T5 work out where deletion happened, as well as what was deleted
- permuted sequence prediction — BART rearrange sentence
- Rotated Sequence Prediction — BART To enhance the model's ability to understand changes in sentence order or structure.
- infilling sequence prediction — BART The mask might be empty

Decoder — ELMo, GPT

- next-token prediction — ***GPT***

Advantages	Disadvantages
- Captures full bidirectional context	- Not suited for text generation
- Good for understanding tasks	- No autoregressive decoding

ELECTRA
Making a prediction for every word faster training

- Captures full input meaning	- More complex
- Flexible input-output mapping	- Slower inference (2 modules)
- Powerful for seq2seq	- Requires more memory

Training (fine-tuning)
- Done many times
- Can be short or long training
- Can be on limited compute or a lot
- Task specific data

- Autoregressive: good for generation	- Unidirectional: only left context
- Simple architecture	- Not good for understanding tasks

pre-training and training

Pre-training
- done once
- Long training
- Trained on lots of GPUs
- Input + Output are based on raw text

Mean Reciprocal Rank (MRR)
earlier the correct answer appears in the ranking, the higher the score.

Sentiment (Classification) — all
NER (Token predictions) — En & De
Coreference (Structured Output) — Encoder-Decoder
Summarisation (Generation) — Decoder
Translation (Generation, in another language) — Decoder

In-Context Learning ICL
Low perplexity prompts are better
Order of examples matters
unsorted examples are best
more number of shots N → better performance
provide instructions and examples in the input
any label is acceptable
have similar performance in the end
Bad explanations usually lead to the wrong answer!

Url Filtering
Trafilatura text extraction on the raw HTML
FastText LanguageFilter removing any document with en language score lower
Wrong answers that have many matching words
Quality filtering
Minhash deduplication deduplicate
PII Formatting anonymize email and public IP addresses
Train a model-based detector — overpredict/overfilter

Risk of dataset
- Avoid overfitting into one dataset
- Shortcuts / Validity minimal edits true answer make it wrong
- Does our task have statistical power?
- Social Bias
- Is this a task we want to create?

whether a model can do well without the question and/or context
correcting for chance agreement
Cohen's Kappa 0.80-1.0 good
Unclear guidelines? Careless annotators? Ambiguous cases?

Instruction Tuning FLAN-T5 higher generalization Alpaca LIMA
no clear pattern bigger model benefits more from instruction fine tuning
During instruction fine-tuning: Perplexity may increase instead of decreasing.
Filtered data produce better generation quality high quality
Increase training examples does not improve generation quality

Preference Optimization
RLHF Approach Without making too big change reward model
DPO directly train from preference data
Avoids training a reward model, which may be unreliable

Agent
Reasoning: happened before generating prediction
Self-consistency voting with multiple outputs
Refletion: consider about previous part
Tree of thoughts exploring multiple possible "thought paths"
Acting: doing something beyond the generation: RAG

Reducing Model Size — Efficiency of LLM

- Distillation use a big model to train a small model so the small model can perform well
- Adding sparsity removing unnecessary features or connections.

doesn't help on GPUs because sparse

Increasing training memory efficiency

- Parameter-Efficient Fine Tuning (PEFT) — LoRA

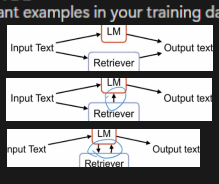
Numerical approximation Reduce weight matrix size

- Quantization — LLM.int8 Outlier 16-bit reduced numerical precision version

Mixtures of models to less important weights

- SMoE divide the model into multiple smaller "expert"

1. Treating retrieval as an LM itself and doing model ensembling
   a. kNN-LM Larger k values generally lead to lower perplexity. Larger datastore size makes relying on external context more reliable. Smaller λ means more dependence on the LM's internal (in-context) information.
2. Provide the retrieved content in the prompt
   Retrieval ICL = ICL + auto-select relevant examples for each input
   a. Retrieval ICL Find the most relevant examples in your training data and use them in the prompt.
   b. RAG
   c. RETRO
   d. REALM
   e. FLARE



Retrieval Augmentation

3. Interactive methods (related to Agents, which we will see later)