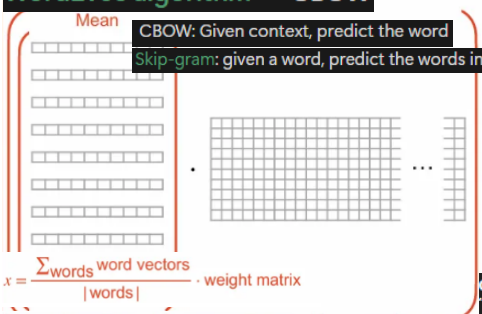
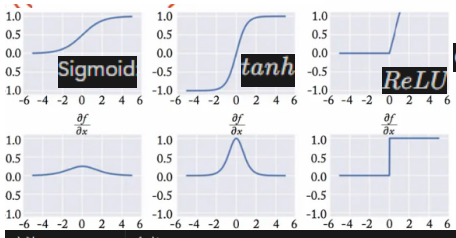
Word2vec algorithm — CBOW



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.

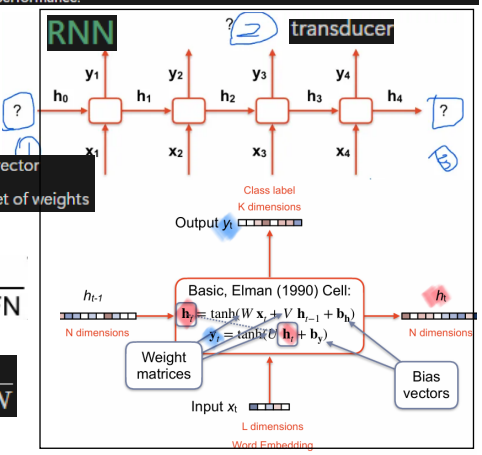
$$Y_{micro} = \frac{\sum TP}{\sum (TP + Y)} \quad X_{macro} = \frac{\sum X}{Number\ of\ classses}$$

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

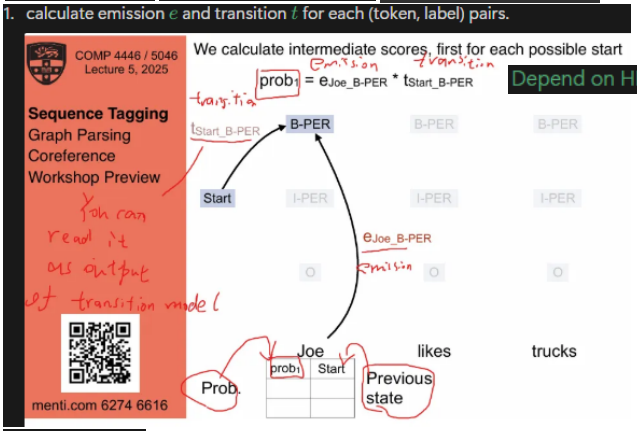
1 previous label $O(|words| * |labels|^2)$

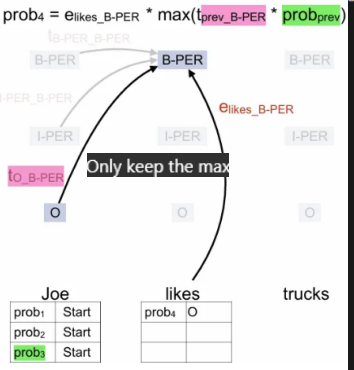
2 previous label $O(|words| * |labels|^3)$ Emission Does not change

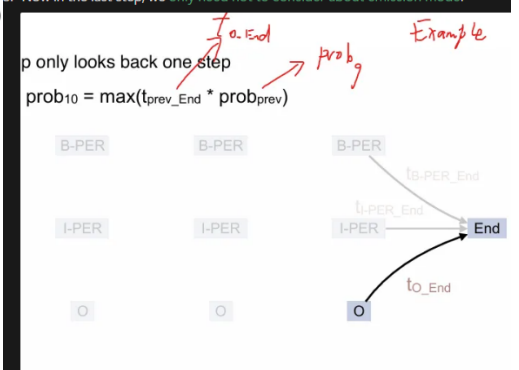
we can use in any model as long as it has previous input and current input

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	$O(\|rules_{arc}\|)$