自然語言處理與應用
Natural Language Processing and Applications

Statistical Language Models and Basic Word Representations

Instructor: 林英嘉 (Ying-Jia Lin)
2025/02/24

Course GitHub

Slido # 7559984

生成式AI融合教學

- 以組為單位
  - ChatGPT Plus
  - Google Colab Pro 雲端運算付費服務
  - OpenAI API 呼叫大型語言模型進行生成式服務
- 可以開始找組員，W3 or W4 開始受理核銷 (暫定)

自然語言處理基本功 (如何表達字詞)

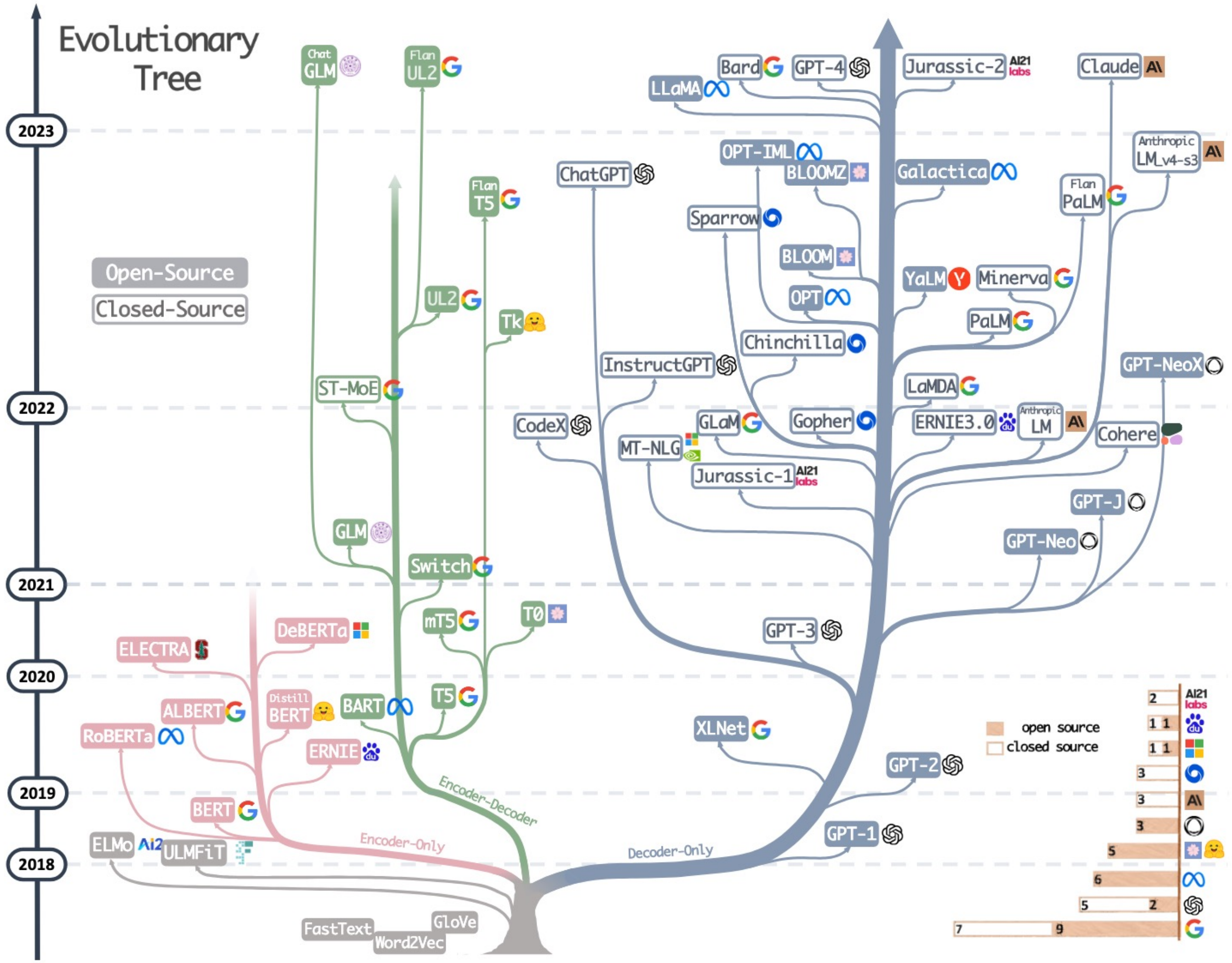
- Week 1 – Week 3
  - 自然語言處理介紹
  - 統計語言模型與基本詞向量方法
  - 詞嵌入模型

Outline

- Basic Word Representations
  - One-hot encodings
  - PMI
  - LSA
  - Bag-of-words (document representations)
  - TF-IDF (document representations)
- Language Models

Basic Word Representations

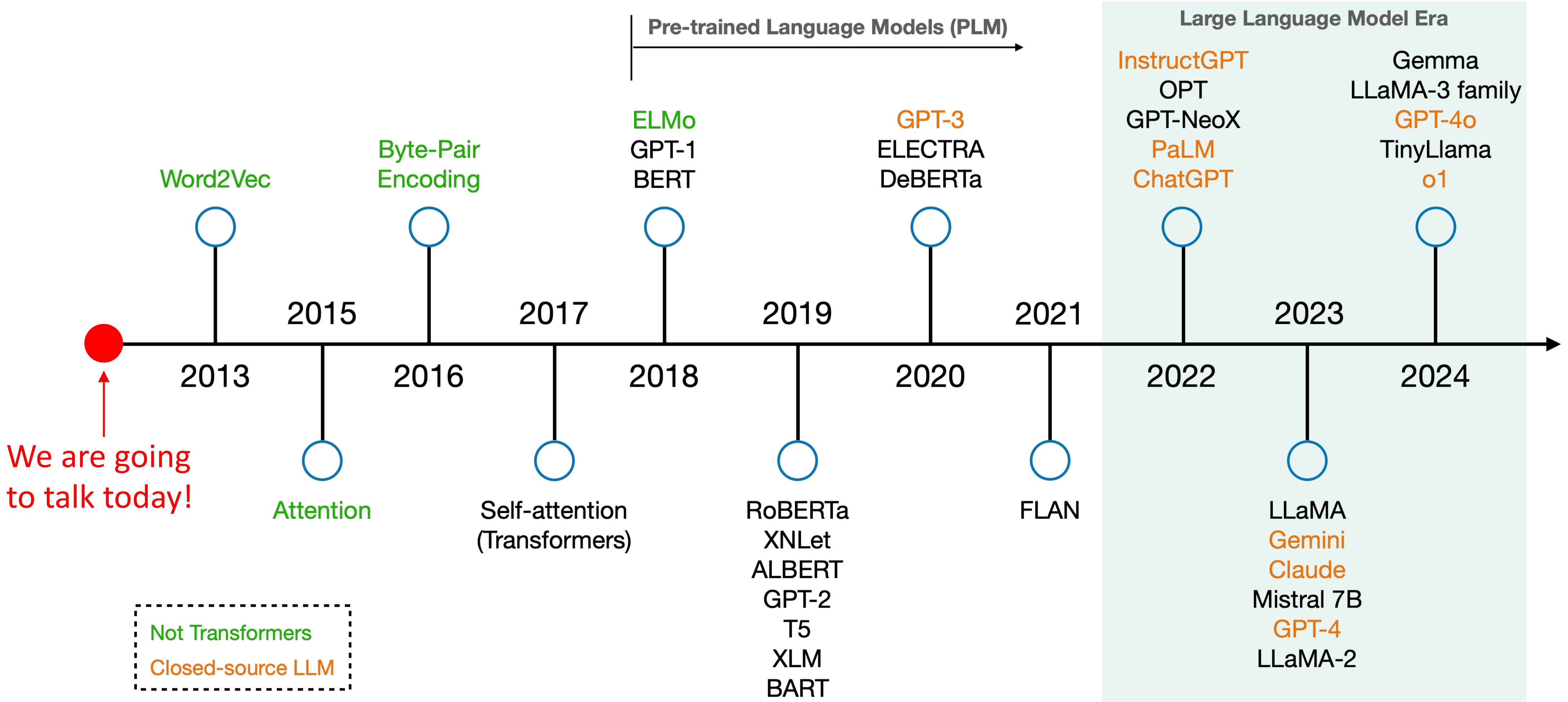
Evolutionary Tree of Large Language Models



Yang, Jingfeng, et al. "Harnessing the power of llms in practice: A survey on chatgpt and beyond." ACM Transactions on Knowledge Discovery from Data 18.6 (2024): 1-32.

NLP

Language Model Evolution Path



Word Representations with One-hot Encodings

like

apple

?

Basic Approach

Corpus

I like to eat apple

You like to watch TV

Vocabulary

apple	[1, 0, 0, 0, 0, 0, 0, 0]
eat	[0, 1, 0, 0, 0, 0, 0, 0]
I	[0, 0, 1, 0, 0, 0, 0, 0]
like	[0, 0, 0, 1, 0, 0, 0, 0]
watch	[0, 0, 0, 0, 1, 0, 0, 0]
to	[0, 0, 0, 0, 0, 1, 0, 0]
⋮	

One hot encodings with each vector size equal to the vocab size (8 in this case)

Pros: handy, training-free
Cons: sparse matrix which occupies much memory

Distributional Hypothesis

https://aclweb.org/aclwiki/Distributional_Hypothesis

- "A word is characterized by the company it keeps" was popularized by Firth (1957).
- In other words, semantically similar words may appear in similar contexts.

I drink beer.

We drink wine.

The cat licked its fur.

The Persian cat licked its fur.

Firth, John Rupert. "Applications of general linguistics." Transactions of the Philological Society 56.1 (1957): 1-14.

Co-occurrence Matrices (共現矩陣)

Corpus

I love deep learning.

I love NLP.

I do programming.

Here the context size is set to 1.

	I	love	do	deep	learning	NLP	progra mming
I	0	2	1	0	0	0	0
love	2	0	0	1	0	1	0
do	1	0	0	0	0	0	1
deep	0	1	0	0	1	0	0
learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
progra mming	0	0	1	0	0	0	0

Problem of Co-occurrence Matrices

- High-frequency words impact the representations.
  - Such as “the”, “a”, “an”
- High-frequency words that do not significantly affect the semantics are stopwords.
- You can check common stopwords via:

```
from nltk.corpus import stopwords
stop_words = list(stopwords.words('english'))
```

NLTK (Natural Language Toolkit)

- Preprocessing functions: tokenization, removing stopwords
- WordNet
- ...

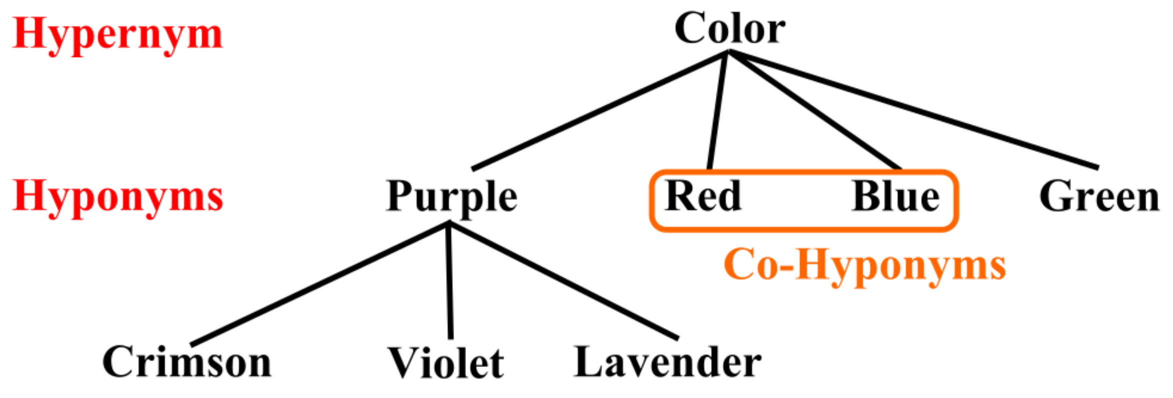


Figure source: https://analyticsindiamag.com/deep-tech/a-complete-guide-to-using-wordnet-in-nlp-applications/

PMI (Pointwise Mutual Information)

- To fix the problem of co-occurrence matrix, we can discard the counting-based approach.

  Given words w_i and w_j, their PMI score is:

$$\mathrm{PMI}(w_i, w_j) = \log_2 \frac{P(w_i, w_j)}{P(w_i)P(w_j)}$$

NLP

The PMI Matrix

Corpus

I love deep learning.

I love NLP.

I do programming.

Here the context size is set to 1.

	I	love	do	deep	learning	NLP	programming
I	-inf	1.222	1.222	-inf	-inf	-inf	-inf
love	1.222	-inf	-inf	0.807	-inf	1.807	-inf
do	1.222	-inf	-inf	-inf	-inf	-inf	2.807
deep	-inf	0.807	-inf	-inf	2.807	-inf	-inf
learning	-inf	-inf	-inf	2.807	-inf	-inf	-inf
NLP	-inf	1.807	-inf	-inf	-inf	-inf	-inf
programming	-inf	-inf	2.807	-inf	-inf	-inf	-inf

Co-occurrence Matrices (共現矩陣)

Corpus

I love deep learning.

I love NLP.

I do programming.

Here the context size is set to 1.

	I	love	do	deep	learning	NLP	programming
I	0	2	1	0	0	0	0
love	2	0	0	1	0	1	0
do	1	0	0	0	0	0	1
deep	0	1	0	0	1	0	0
learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
programming	0	0	1	0	0	0	0

PPMI (Positive Pointwise Mutual Information)

$$\mathrm{PPMI}(w_i, w_j) = \max\left(0, \mathrm{PMI}(w_i, w_j)\right)$$

$$= \max\left(0, \log_2 \frac{p(w_i, w_j)}{p(w_i)p(w_j)}\right)$$

Or $$\mathrm{PPMI}(w_i, w_j) = \mathrm{PMI}(w_i, w_j) + \epsilon$$

where ϵ is a small positive number.

The PPMI Matrix

Corpus

I love deep learning.

I love NLP.

I do programming.

Here the context size is set to 1.

	I	love	do	deep	learning	NLP	programming
I	0	1.222	1.222	0	0	0	0
love	1.222	0	0	0.807	0	1.807	0
do	1.222	0	0	0	0	0	2.807
deep	0	0.807	0	0	2.807	0	0
learning	0	0	0	2.807	0	0	0
NLP	0	1.807	0	0	0	0	0
programming	0	0	2.807	0	0	0	0

Latent Semantic Analysis (LSA)

- latent (adj.) 潛在的; Latent Semantic Analysis: 潛在語義分析
- Current problems:
  1. 資料問題
     - 有出現不見得是相關
     - 沒出現不見得是無關
  2. 維度問題
     - The vector size is equal to the vocab size, which takes much memory.

Singular Value Decomposition (SVD)

Formula of SVD:

$$A = U\Sigma V^{\top}$$

where:

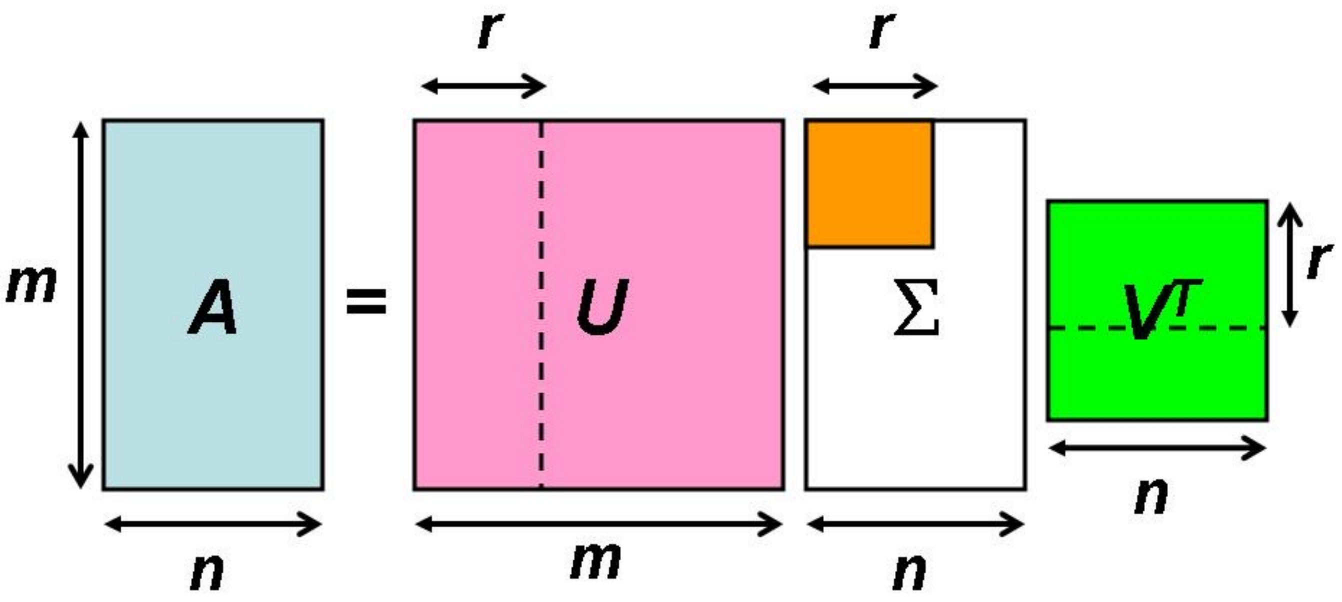
A: data matrix from PMI/PPMI or cooccurrence matrix.

U: Left orthogonal matrix

Σ: diagonal matrix，對角線上的值為奇異值

V: Right orthogonal matrix

Singular Value Decomposition (SVD)



- 我們關注的是column space的降維 (以保留vocab維度)
- 因此我們可以取U的前r個向量作為我們降維後的目標

Dimensionality Reduction (降維)!

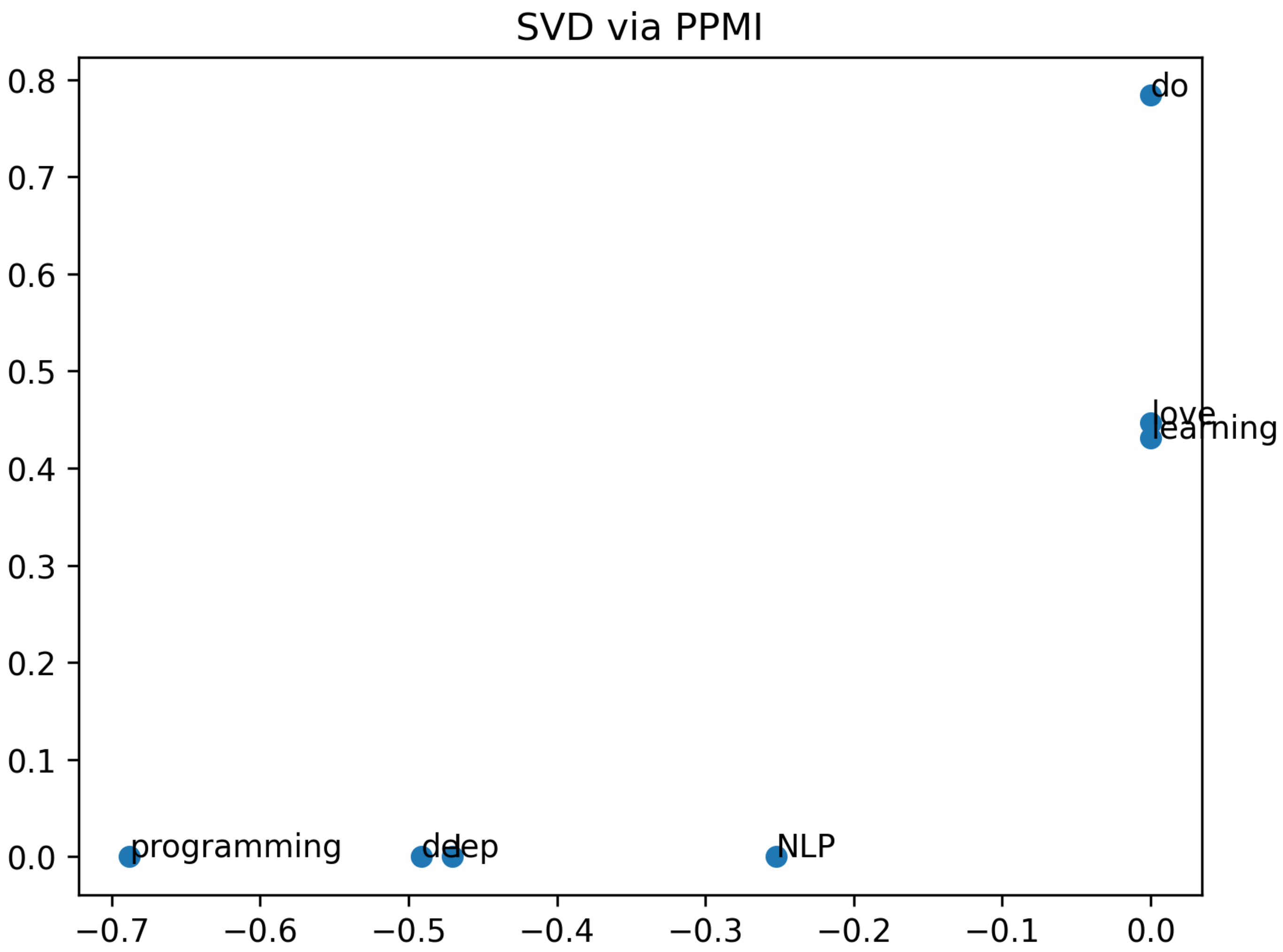
https://ccjou.wordpress.com/2009/09/01/奇異值分解-svd/

Results after SVD from the PPMI Matrix

Vocab	Dimension 1	Dimension 2
I	-0.47030617476956654	-4.440892098500626e-16
love	-8.326672684688674e-16	0.4464710977801294
do	8.881784197001252e-16	0.7838949952895327
deep	-0.4915010045415975	-1.3016463110087907e-16
learning	-1.582067810090848e-15	0.4314767609119055
NLP	-0.2523320838384413	-2.1837832516159107e-16
programming	-0.6881623238553303	-1.914220234434739e-16

原本一個字需要7個維度才能表示，但現在經過SVD之後只需要2個維度

Visualization of the SVD Results



Problems of SVD

- We can perform Singular Value Decomposition (SVD) for dimensionality reduction, however at a quadratic ($O(n^2)$) computational cost.
- In other words, SVD is slow for a big matrix.
- Adding a new word changes the entire matrix.

Summary

Methods	Type	Word Order?	Semantics?	Dimensionality Reduction
One-hot Encodings	Word Embeddings	❌	❌	❌
Co-occurrence / PMI /PPMI matrix		❌	✅ (Co-occurrence)	❌
LSA		❌	✅	✅ (Computationally Expensive)

Embedding Lookup

Index	Vocab	Dimension 1	Dimension 2
0	I	-0.47030617476956654	-4.440892098500626e-16
1	love	-8.326672684688674e-16	0.4464710977801294
2	do	8.881784197001252e-16	0.7838949952895327
3	deep	-0.4915010045415975	-1.3016463110087907e-16
4	learning	-1.582067810090848e-15	0.4314767609119055
5	NLP	-0.2523320838384413	-2.1837832516159107e-16
6	programming	-0.6881623238553303	-1.9142202344347739e-16

Basic Document Representations

Bag-of-words Model

Meaning: Each document (sentence) carries a bag of words.

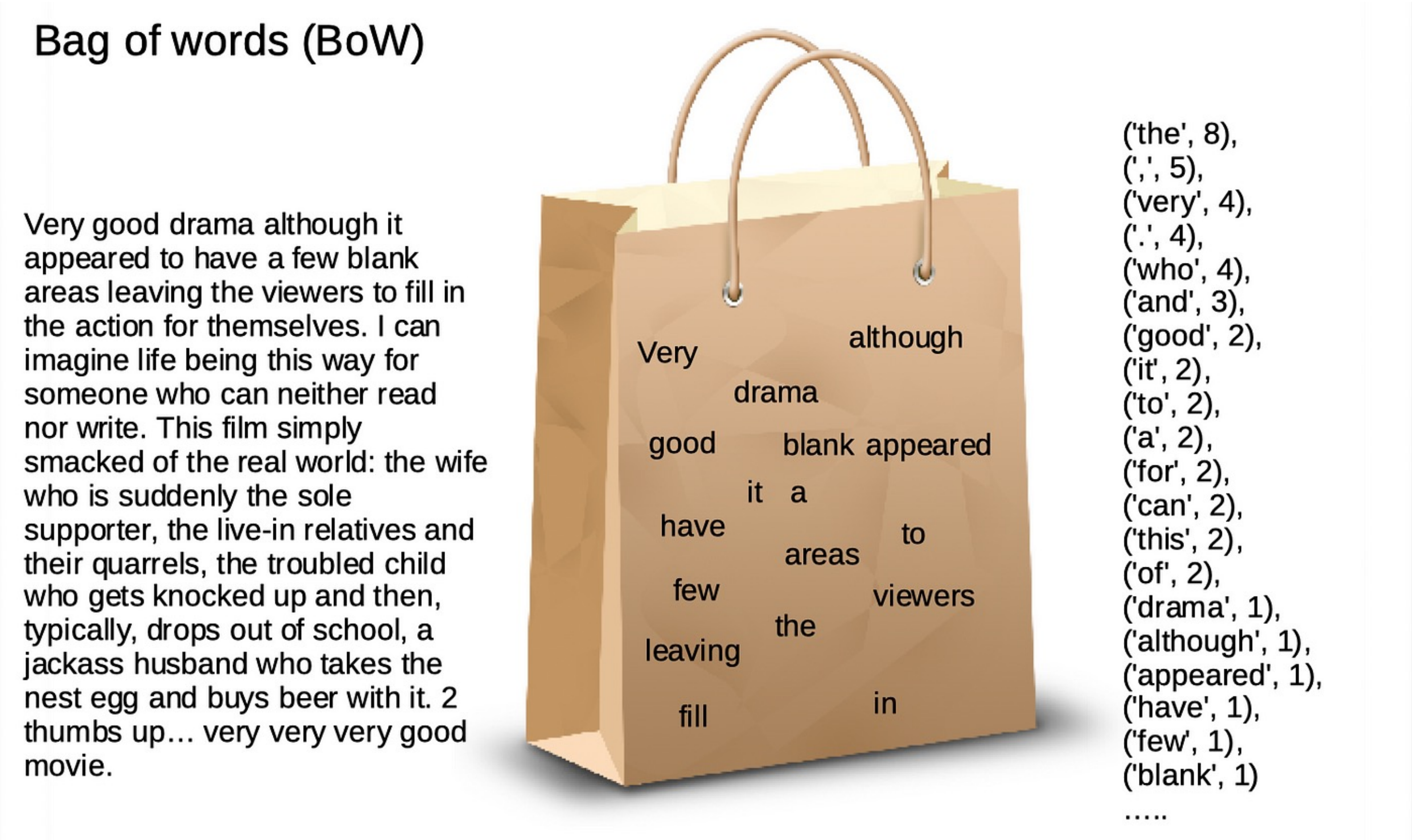


Figure source: https://sfhsu29.medium.com/nlp-入門-1-text-classification-sentiment-analysis-極簡易情感分類器-bag-of-words-naive-bayes-e40d61de9a7f

Bag-of-words Model (Example 1)

Step 1: Find unique words (vocab)

apple	eat	I	like	to	TV	watch	you

Corpus

D1 I like to eat apple

D2 You like to watch TV

Step 2: Form document vectors

D1 [1, 1, 1, 1, 1, 0, 0, 0]

D2 [0, 0, 0, 1, 1, 1, 1, 1]

Notes:
- The vector size is equal to the vocab size.
- Each position of a vector is corresponding to the position in the vocab.

Bag-of-words Model (Example 2)

Corpus

D1 I like like to eat apple

D2 You I like to watch TV

Step 1: Find unique words (vocab)

apple	eat	I	like	to	TV	watch	you

Step 2: Form document vectors

D1 [1, 1, 1, 2, 1, 0, 0, 0]

D2 [0, 0, 1, 1, 1, 1, 1, 1]

Notes:

- The vector size is equal to the vocab size.
- Each position of a vector is corresponding to the position in the vocab.

TF-IDF

- The mathematical representation of TF-IDF:

$$\text{TF-IDF} = \mathrm{TF} \times \mathrm{IDF}$$

where

$$\mathrm{TF}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \qquad \mathrm{IDF_i} = \log \frac{|D|}{\left|\{j : t_i \in d_j\}\right|}$$

  - Where $n_{i,j}$ is the i-th word in the j-th document in the dataset.
- TF (Term Frequency)
  - Represents the "frequency" of a term appearing in a text.
- IDF (Inverse Document Frequency)
  - Aims for terms to have higher specificity, meaning the fewer texts in the dataset contain the term, the better.

Sparse Embeddings: TF-IDF

- TF (Term Frequency)
- IDF (Inverse Document Frequency)

```
texts = [
   "This is a book",
   "These are pens and my pen is here"
]
```

Transform via TF-IDF

- The **TF-IDF** approach creates document embeddings.
- The embedding size is equal to the vocabulary size.
- Each value of an embedding is based on **TF x IDF**.

Vocabulary size

	a	and	are	book	here	is	my	pen	these	this
sent_0	0.534046	0.000000	0.000000	0.534046	0.000000	0.379978	0.000000	0.000000	0.000000	0.534046
sent_1	0.000000	0.324336	0.324336	0.000000	0.324336	0.230768	0.324336	0.648673	0.324336	0.000000

Since the outputs contain many zeros, this approach is called a sparse embedding method.

https://github.com/tsmatz/nlp-tutorials/blob/master/01_sparse_vector.ipynb

Sparse Embeddings: Bag-of-words

```
texts = [
    "This is a book",
    "These are pens and my pen is here"
]
```

Transform via frequency

- The Bag-of-words approach creates document embeddings.
- The embedding size is equal to the vocabulary size.
- Each value of an embedding is based on frequency counts.

Vocabulary size

	a	and	are	book	here	is	my	pen	these	this
sent_0	1	0	0	1	0	1	0	0	0	1
sent_1	0	1	1	0	1	1	1	2	1	0

Since the outputs contain many zeros, this approach is called a sparse embedding method.

Preprocessing steps before creating vectors

Preprocessing text (optional)

- Stemming
  - By removing the suffixes from words (e.g."cats," "catlike," "catty" all have "cat" as their base), we can revert the words back to their root forms.
- Feature Selection
  - Filter and select which parts of speech to retain, such as verbs or nouns.
  - Adjust the frequency of the terms (hyperparameter) using statistical methods or algorithms like TF-IDF.

Sparse Vectors

Sparse vector embeddings represent words as high-dimensional vectors with mostly zero values.

Each dimension corresponds to a unique feature (word), measure by some well-designed methods.

- TF-IDF, Bag-of-words
- PPMI

Dense Vectors

Dense vector embeddings represent words in a continuous vector space

Semantically similar words are closer together.

- Word2Vec
- Contextualized Embeddings

Common NLP tasks

Text Classification	Text Generation	Sequence Tagging	Text Matching
Sentiment Analysis	Machine Translation	Part-of-speech tagging	Semantic Textual Similarity
Natural Language Inference (NLI)	Summarization	Named Entity Recognition (NER)	Retrieval
	Dialogue Generation	Word Segmentation	

BoW and TF-IDF can be used!

Semantic Textual Similarity

https://huggingface.co/datasets/sentence-transformers/stsb

- STS-B dataset

```
{
 'sentence1': 'A man is playing a large flute.',
 'sentence2': 'A man is playing a flute.',
 'score': 0.76,
}
```

```
{
 'sentence1': 'Two boys are driving.',
 'sentence2': 'Two bays are dancing.',
 'score': 0.12,
}
```

Retrieval

Input Query

Which year was Oppenheimer born?

Retriever

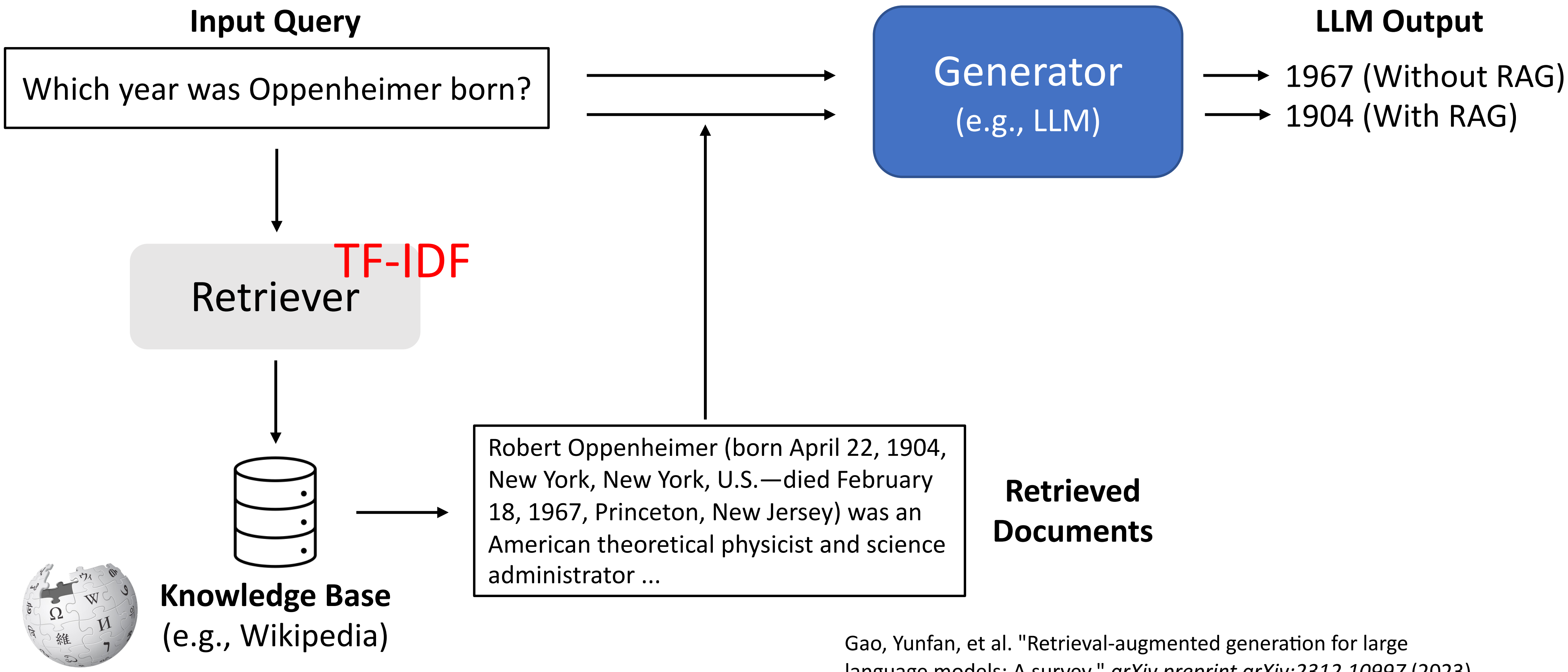
TF-IDF

Robert Oppenheimer (born April 22, 1904, New York, New York, U.S.—died February 18, 1967, Princeton, New Jersey) was an American theoretical physicist and science administrator ...

Retrieved Documents

Knowledge Base
(e.g., Wikipedia)

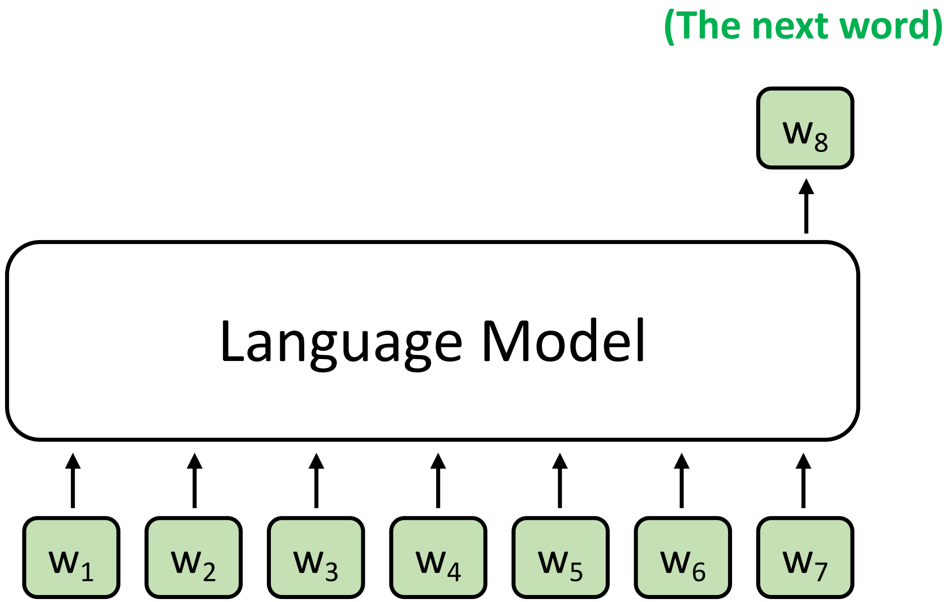
Retrieval-Augmented Generation (RAG)



Gao, Yunfan, et al. "Retrieval-augmented generation for large language models: A survey." *arXiv preprint arXiv:2312.10997* (2023).

Language Models

What is a language model?



$$P(w_t | w_1, w_2, \dots, w_{t-1})$$

- A model that assigns probabilities to upcoming words is called **a language model**.
- The task involving predictions of upcoming words is **language modeling**.

N-gram Language Modeling

- Let's begin with the task of computing the probability (also called relative frequency)

$$P(w|h) = \frac{C(w,h)}{C(h)}$$

w: the word to be generated
h: some history
C: the times the pattern show up in the dataset

- For example: Compute the probability of the word "the" given the history "its water is so transparent that".

$$P(the|its\ water\ is\ so\ transparent\ that) = \frac{C(its\ water\ is\ so\ transparent\ that\ the)}{C(its\ water\ is\ so\ transparent\ that)}$$

For an N-gram LM, N=7 here

LM vs. n-gram LM

Language Model

Full history: P(Parliament | I declare resumed the session of the European)

N-gram Language Model (only n-1 previous words are considered)

(example) Trigram: P(Parliament | the European)

Training and Evaluations

Training Set
(訓練資料集)

Test Set
(測試資料集)

Training Set
(訓練資料集)

Validation Set
(驗證資料集)

Perplexity

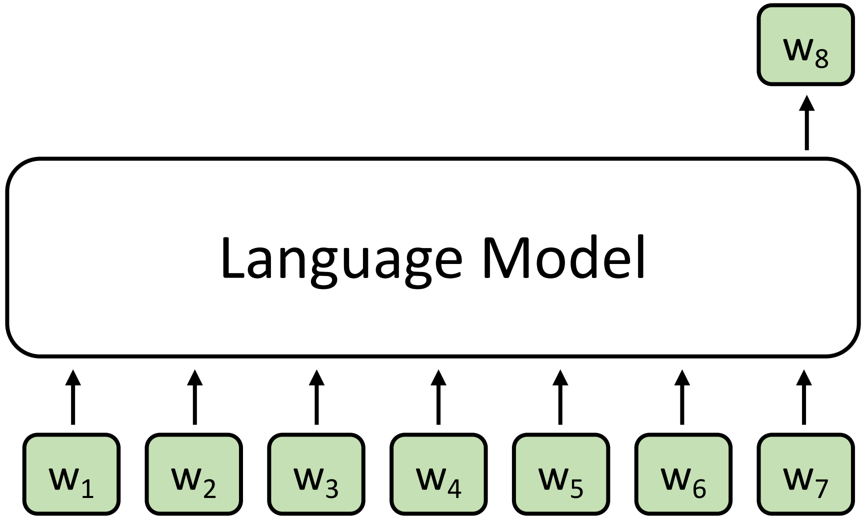
Perplexity (PPL) is a quantitative criterion used to evaluate the capacities of language modeling models.

- Given the sequence of words $W = w_1, w_2, \ldots, w_\mathrm{N}$ and an n-gram language model. The PPL of the model was computed by:

$$\mathrm{Perplexity}(W) = \mathrm{P}(w_1, w_2, \ldots, w_\mathrm{N})^{-\frac{1}{\mathrm{N}}} = \sqrt[\mathrm{N}]{\prod_{\mathrm{t}=1}^{\mathrm{N}} \frac{1}{\mathrm{P}(w_\mathrm{t} | w_{\mathrm{t-n+1}}, \ldots, w_{\mathrm{t-1}})}}$$

The lower the value of perplexity, the better the language modeling capability of the model.

Language Modeling (the whole sentence)



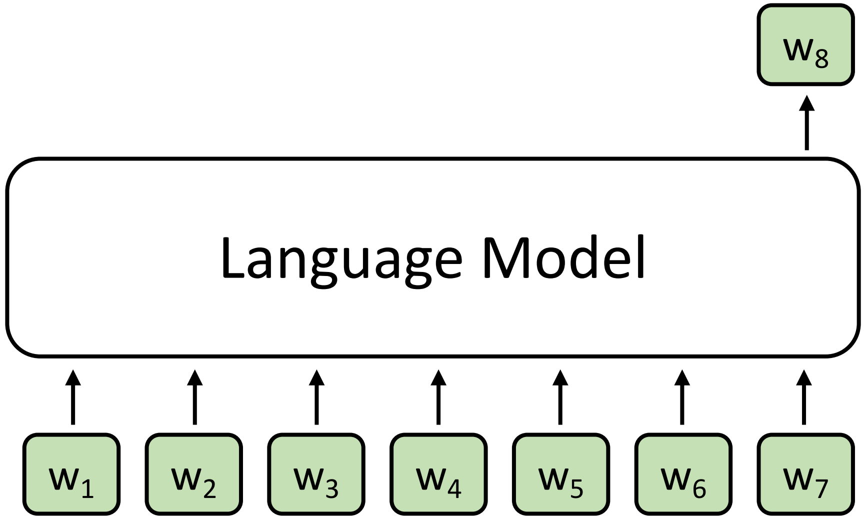
$$P(w_t | w_1, w_2, \ldots, w_{t-1})$$

$$P(w_{1:n}) = P(w_1) \cdot P(w_2 | w_1) \cdot P(w_3 | w_{1:2}) \cdot \ldots \cdot P(w_n | w_{1:n-1})$$

$$= \prod_{t=1}^{n} P(w_t | w_{1:t-1})$$

NLP

Language Modeling (the whole sentence)



$$P(w_t|w_1, w_2, \dots, w_{t-1})$$

$$P(w_{1:n}) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_{1:2}) \cdot \dots \cdot P(w_n|w_{1:n-1})$$

P(今天天氣真好) = P(今) · P(天|今) · P(天|今天) · P(氣|今天天) ·

P(真|今天天氣) · P(好|今天天氣真)

Statistical Language Models

$$P(w_{1:n}) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_{1:2}) \cdot \ldots \cdot P(w_n|w_{1:n-1})$$

For Bi-grams:

$$P(w_t|w_{t-1}) = \frac{\mathrm{Count}(w_{t-1}, w_t)}{\mathrm{Count}(w_{t-1})}$$

w_{t-1} 接 w_t 的數量

開頭為 w_{t-1} 的數量

For Tri-grams:

$$P(w_t|w_{t-2}, w_{t-1}) = \frac{\mathrm{Count}(w_{t-2}, w_{t-1}, w_t)}{\mathrm{Count}(w_{t-2}, w_{t-1})}$$

w_{t-2}, w_{t-1}接 w_t 的數量

開頭為 w_{t-2}, w_{t-1}的數量

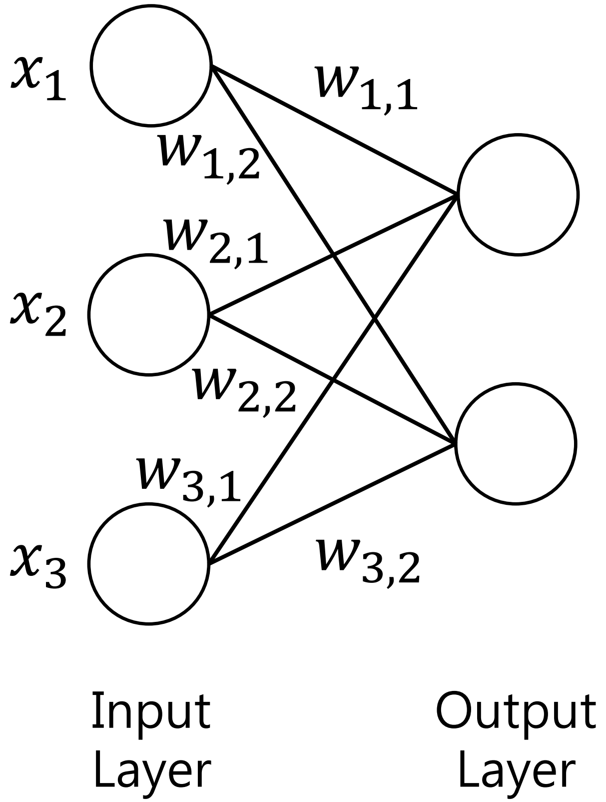
Problems of Statistical Representations

- **Lack of Long-Range Context**
  - 對於一個10個字的句子，n-gram 模型需要 V^{10} 次計算，在V很大的情況下極為不可行，因此通常只能考慮 1 或 2 個詞的上下文
- **Lack of Word Similarity Awareness**
  - One-hot encoding 是高維稀疏表示，導致詞之間無關聯
- The curse of dimensionality

Bengio, Yoshua, et al. "A neural probabilistic language model." Journal of machine learning research 3.Feb (2003): 1137-1155.

Distributed Representations

The natural expression of distributed representations in a neural net is to make each concept be a single unit and to use the connections between units to encode the relationships between concepts.



Hinton, Geoffrey E. "Learning distributed representations of concepts." Proceedings of the Annual Meeting of the Cognitive Science Society. Vol. 8. 1986.

After Training Word Embeddings

- The outputs are dense vectors with a fixed dimension size:

Dimension size (e.g., 300)

Vocabulary size (e.g., 30,000)

apple -0.110960 0.016115 -0.004809 0.033589 0.121455 …
banana -0.027713 -0.015676 0.003314 0.077602 0.159718 …
…
…
…

Word Embeddings 視覺化

- **Word embedding model**: glove-wiki-gigaword-100
- **Dimension reduction**: t-SNE
- **Dataset**: Mikolov et al., 2013

NLP

Additional Resource

- (Textbook) Chapter 3: N-gram Language Models
  - https://web.stanford.edu/~jurafsky/slp3/3.pdf
- Statistical Language Models
  - https://aclanthology.org/www.mt-archive.info/MTMarathon-2010-Bertoldi-1-ppt.pdf

Contact Information

Instructor: 林英嘉

✉️ yjlin@cgu.edu.tw

🕘 Office Hour: 歡迎來信預約，一定沒問題

TA: 吳宣毅

✉️ m1161007@cgu.edu.tw

寄信前請在主旨加註記 [自然語言處理]

Thank you!

Instructor: 林英嘉
✉️ yjlin@cgu.edu.tw
TA: 吳宣毅
✉️ m1161007@cgu.edu.tw