# Bài 1: - 31/5

1) Trình bày về mục tiêu, ý nghĩa của nhiệm vụ Word2Vec (còn được hiểu là sinh Word Embedding)

2) Trình bày cách huấn luyện để tạo vector từ words đối với phương pháp CBOW và Skip-Gram

3) Hãy trình bày các phương pháp Word2Vec khác: Fasttext, Glove

4) Sử dụng thư viện Gensim để huấn luyện mô hình word2vec cho tiếng Việt

5) Hãy tìm hiểu các cách biểu diễn Vector của 1 câu (Sentence2Vec), sử dụng để phân loại câu hoặc đo độ tương tự của 2 câu

## Note:

* 1,2,3 Vũ - 4,5 Chị trang.
* Pretrain, relationships words (base on pretrain).
* Tạo ra 1 câu như nào, 1 từ, 1 văn bản như thế nào?

# Bài 2:

1) Tìm hiểu bài toán phân loại văn bản đa nhãn và ứng dụng thuật toán dựa trên Neural Network để giải quyết bài toán này.

a. Sử dụng Feed Forward Neural Network

b. Sử dụng các loại RNN (Recurrent Neural Networks)

Lưu ý: có thể thử nghiệm cả 2 phương án, chọn hoặc ko chọn việc sử dụng Word2vec hoặc Sentence2Vec

2) Vẫn bài toán ở câu 1 nhưng sử dụng các phương pháp truyền thống: SVM, Logistic Regression, Naive Bayes classification.

So sánh tất cả các phương pháp, bao gồm cả các phương pháp ở câu 1.

Note: Lý thuyết với code

* Thầy muốn là output dùng sigmoid để ngưỡng threshold = 0.5 để lấy đa nhãn.
* Thầy muốn xử lý nhiều cách preprocessing. (word to vec, sentence to vec, bag of word…tf idf)
* Trình bày lý thuyết buổi diễn w1 -> thành vector để vo rnn.
* Thầy muốn dùng BERT, Transformer.
* CHỉnh overfit.

# Vũ

## Khái niệm

Word Embedding là tên gọi chung của các mô hình ngôn ngữ và các phương pháp học theo đặc trưng trong Xử lý ngôn ngữ tự nhiên(NLP), ở đó các từ hoặc cụm từ được ánh xạ sang các vector số (thường là số thực). Đây là một công cụ đóng vai trò quan trọng đối với hầu hết các thuật toán, kiến trúc Machine Learning, Deep Learning trong việc xử lý Input ở dạng text, do chúng chỉ có thể hiểu được Input ở dạng là số, từ đó mới thực hiện các công việc phân loại, hồi quy,vv…

## Mục tiêu của Word2Vec

1. Biểu diễn từ vựng trong không gian liên tục: Chuyển đổi các từ thành các vector số trong không gian nhiều chiều.
2. Ghi nhận ngữ nghĩa của từ: Các vector nhúng từ phải phản ánh ngữ nghĩa và ngữ cảnh của từ trong văn bản.
3. Cải thiện hiệu quả của các mô hình học máy: Sử dụng word embeddings làm đầu vào cho các mô hình học máy khác, giúp chúng học và dự đoán hiệu quả hơn.

## Ý nghĩa của Word2Vec

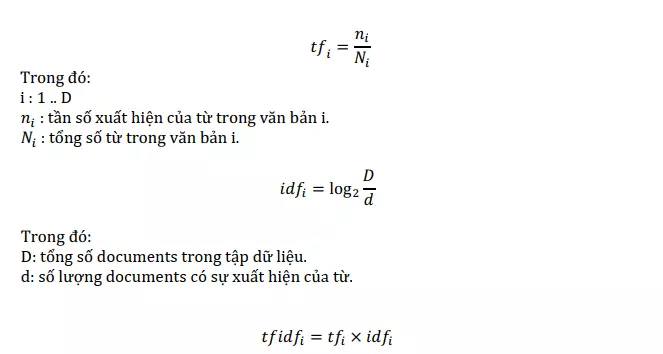
1. Biểu diễn ngữ nghĩa trong không gian liên tục: Các từ có nghĩa gần nhau sẽ có vector gần nhau, giúp mô hình hiểu được mối quan hệ ngữ nghĩa giữa các từ.
2. Giảm chiều dữ liệu: Thay vì sử dụng one-hot encoding có kích thước lớn, mỗi từ được biểu diễn bằng một vector có chiều nhỏ hơn nhiều.
3. Tính toán hiệu quả: Các phép toán trên word embeddings có thể biểu diễn các mối quan hệ ngữ nghĩa, như "king" - "man" + "woman" ≈ "queen".
4. Cải thiện kết quả của nhiều bài toán NLP: Word embeddings được sử dụng trong nhiều ứng dụng như phân loại văn bản, phân tích cảm xúc, dịch máy, và tóm tắt văn bản, giúp cải thiện độ chính xác và hiệu quả của các mô hình.
5. Khả năng tổng quát hóa tốt hơn: Word embeddings giúp mô hình tổng quát hóa tốt hơn cho các từ không xuất hiện trong tập huấn luyện dựa trên ngữ cảnh.

## Word Embedding được phân chủ yếu thành 2 loại:

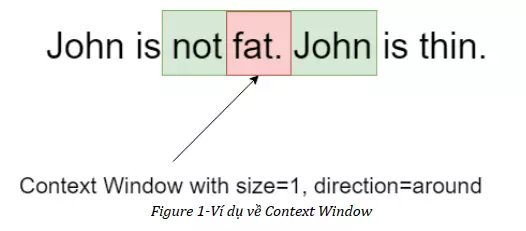
* Frequency-based embedding.
* Prediction-based embedding.

### Frequency-based embedding

#### Count vector and TF-idf

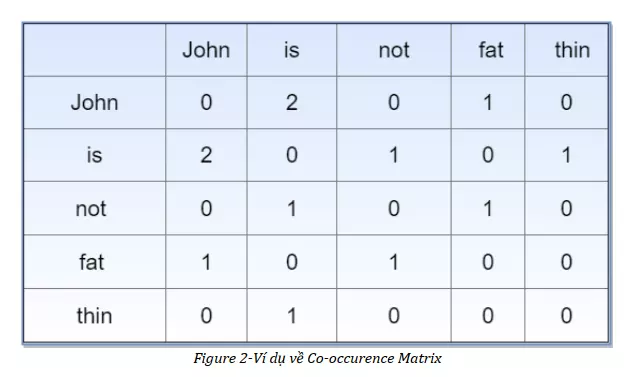
* Count vector: Biểu diễn văn bản bằng cách đếm số lần xuất hiện của các từ trong tài liệu.
* TF-IDF: Biểu diễn văn bản bằng cách cân nhắc tần số xuất hiện của từ (TF) và tần số nghịch đảo của tài liệu chứa từ đó (IDF).
* Khác với *Count Vector* chỉ xét đến **tần số** xuất hiện của từ trong một **document**, *tf-idf Vector* quan tâm cả **tần số** xuất hiện của từ trong **toàn bộ tập dữ liệu**, chính do đặc điểm này mà *tf-idf Vector* có tính phân loại cao hơn so với *Count Vector*. *tf-idf (****Term Frequency-Inverse Document Frequency)***
* *Vector* là một vector số thực cũng có độ dài *D* với *D* là số văn bản, nó được tính bằng tích của 2 phần bao gồm *tf* và *idf*, công thức của mỗi phần tử của vector được tính như sau:
* 

Tuy nhiên, nhược điểm của cả hai phương pháp trên chính là việc nó chỉ chú trọng đến tần số xuất hiện của một từ, dẫn tới nó hầu như không mang ý nghĩa gì về mặt ngữ cảnh, Co-occurrence Matrix phần nào giải quyết vấn đề đó. Co-occurrence Matrix có ưu điểm là bảo tồn mối quan hệ ngữ nghĩa giữa các từ, được xây dựng dựa trên số lần xuất hiện của các cặp từ trong **Context Window.**

* 

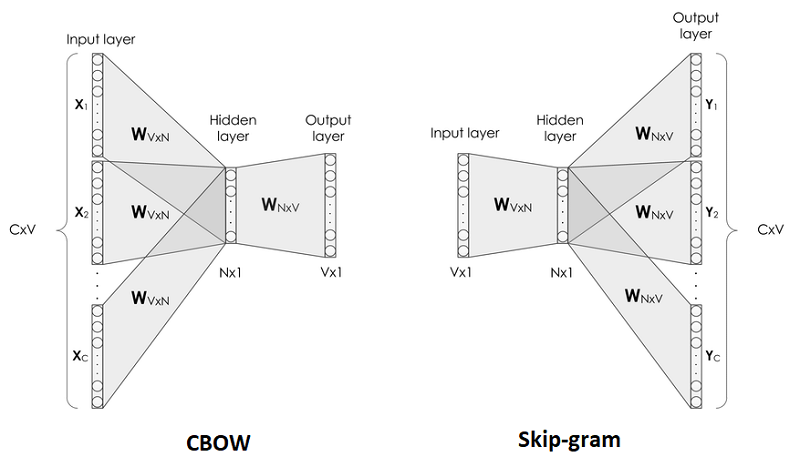
#### Co-occurrence Matrix

*Co-occurrence Matrix:* là một ma trận vuông đối xứng, mỗi hàng hoặc mỗi cột sẽ chính là vector biểu thị của từ tương ứng.

* 
* Trong thực tế, do số lượng từ vựng nhiều, ta thường chọn bỏ đi một số từ không cần thiết (như các stopwords) hoặc sử dụng phân tách SVD (Singular Value Decomposition) để giảm kích thước của vector từ. Điều này giúp làm rõ biểu diễn của từ và tiết kiệm bộ nhớ lưu trữ Co-occurrence Matrix, vì các ma trận này có kích thước rất lớn.

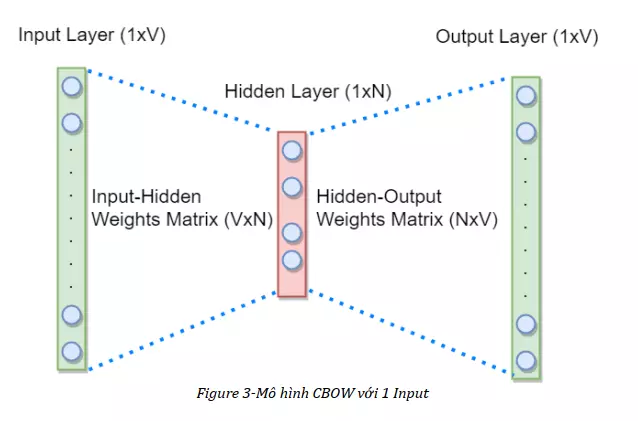
### 

### Prediction-based embedding

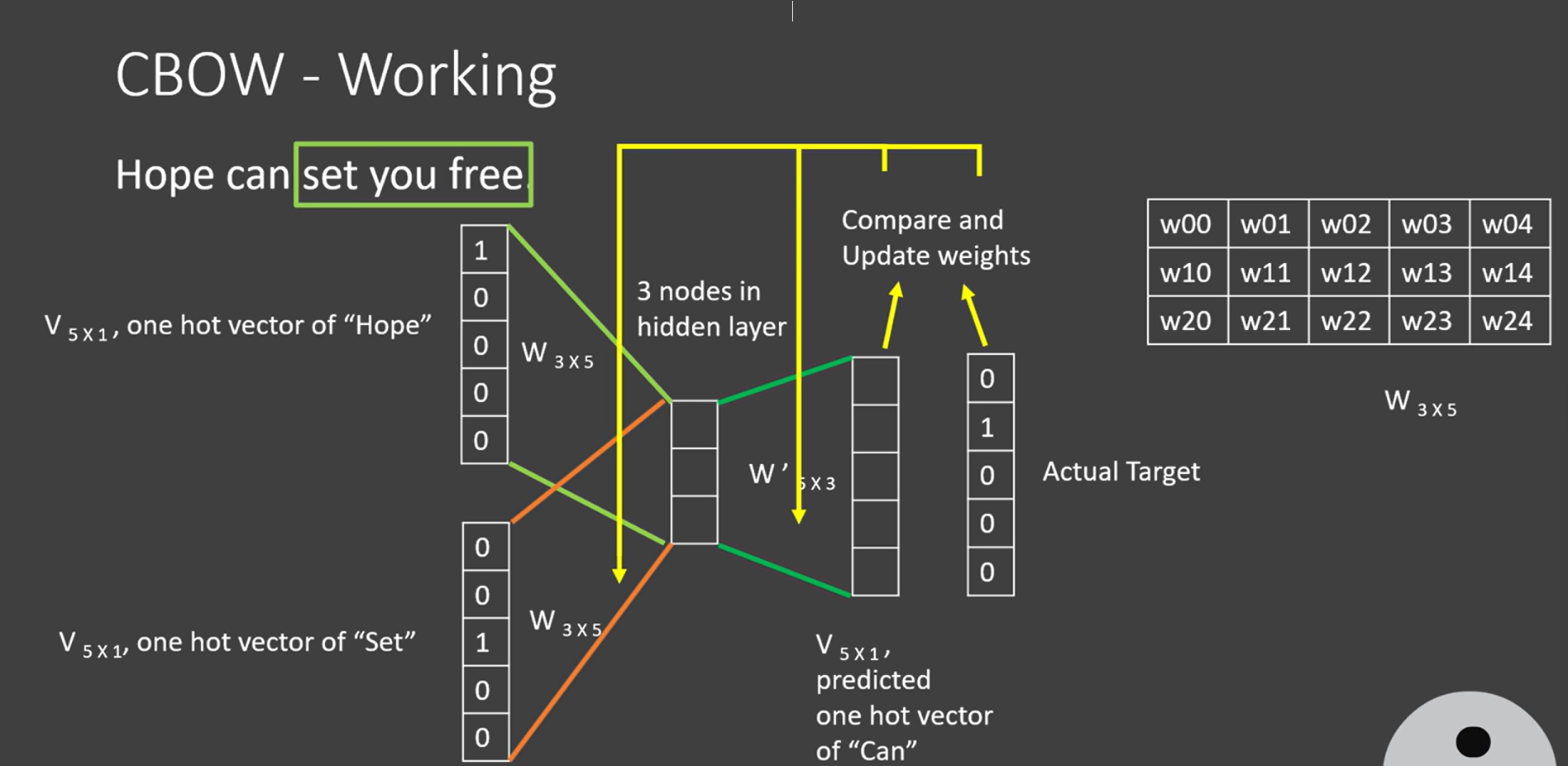
* Xây dựng các vector từ dựa vào các mô hình dự đoán
* *CBOW (Continuous Bag Of Words)* và *Skip-gram*.
* Cả hai mô hình này đều được xây dựng dựa trên một mạng neuron gồm 3 lớp:1 Input Layer,1 Hidden Layer và 1 Output Layer.
* Mục đích chính của các mạng neuron này là học các trọng số biểu diễn vector từ.
* 
* Chú giải:
* Kích thước của cửa sổ ngữ cảnh là C/2. Ví dụ C=2 thì ta sẽ có 5 phần tử (x\_1, x\_2, x\_3, x\_4, x\_5), với x\_3 là phần tử trung tâm, các phần tử còn lại là phần tử ngữ cảnh.
* Số lượng từ vựng là V.
* Số đơn vị ẩn là N.

#### 

#### CBOW (Continuous Bag Of Words)

* *CBOW* hoạt động dựa trên cách thức là nó sẽ dự đoán xác suất của một từ được đưa ra theo ngữ cảnh (một ngữ cảnh có thể gồm một hoặc nhiều từ)
* Với input là một hoặc nhiều One-hot vector của các từ ngữ cảnh có chiều dài V (với V là độ lớn của từ điển),
* Output sẽ là một vector xác suất cũng với chiều dài V của từ liên quan hoặc còn thiếu
* Hidden Layer có chiều dài N, N cũng chính là độ lớn của vector từ biểu thị.
  + 

Example:

* 
* C = 1, V = 5, N = 3.

##### Cách hoạt động - CBOW:

Câu: “Hope can set you free”

C = 1, V = 5, N = 3.

Bước 1: Dùng BoW để biến các từ vựng thành vector

Vocab = [can, free, hope, set, you]

can = [0,1,0,0,0]

free = [0,0,0,0,1]

hope = [0,0,1,0,0]

….

Bước 2: Xác định [đầu vào | đầu ra] của mô hình, mỗi dòng dưới đây là một ô cửa sổ ngữ cảnh.

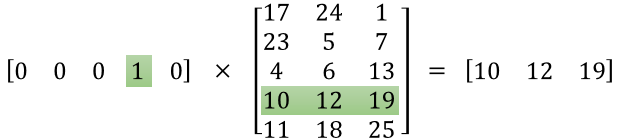
[hope, set | can]

[can, you | set]

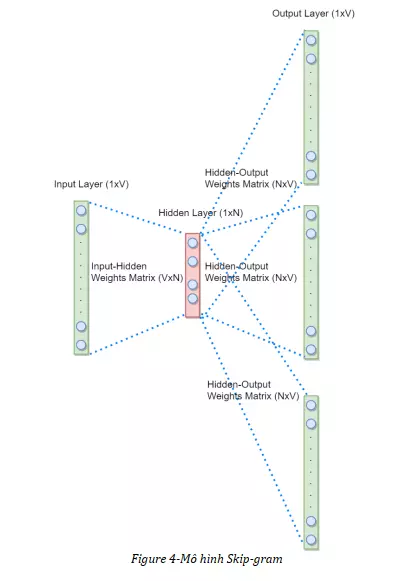
…..

* Hai ma trận trọng số Input-Hidden Weights Matrix và Hidden-Output Weights Matrix ban đầu được khởi tạo ngẫu nhiên.
* Input được nhân với Input-Hidden Weights Matrix để ra Hidden Activation.
* Hidden Activation tiếp tục được nhân với Hidden-Output Weights Matrix.
* Kết quả cuối cùng được đưa vào hàm softmax để ra Output là một vector xác suất.
* Output này được so sánh với Output mong muốn để tính toán độ lỗi.
* Dựa vào độ lỗi, mạng neuron sẽ lan truyền ngược để cập nhật các giá trị của các ma trận trọng số.

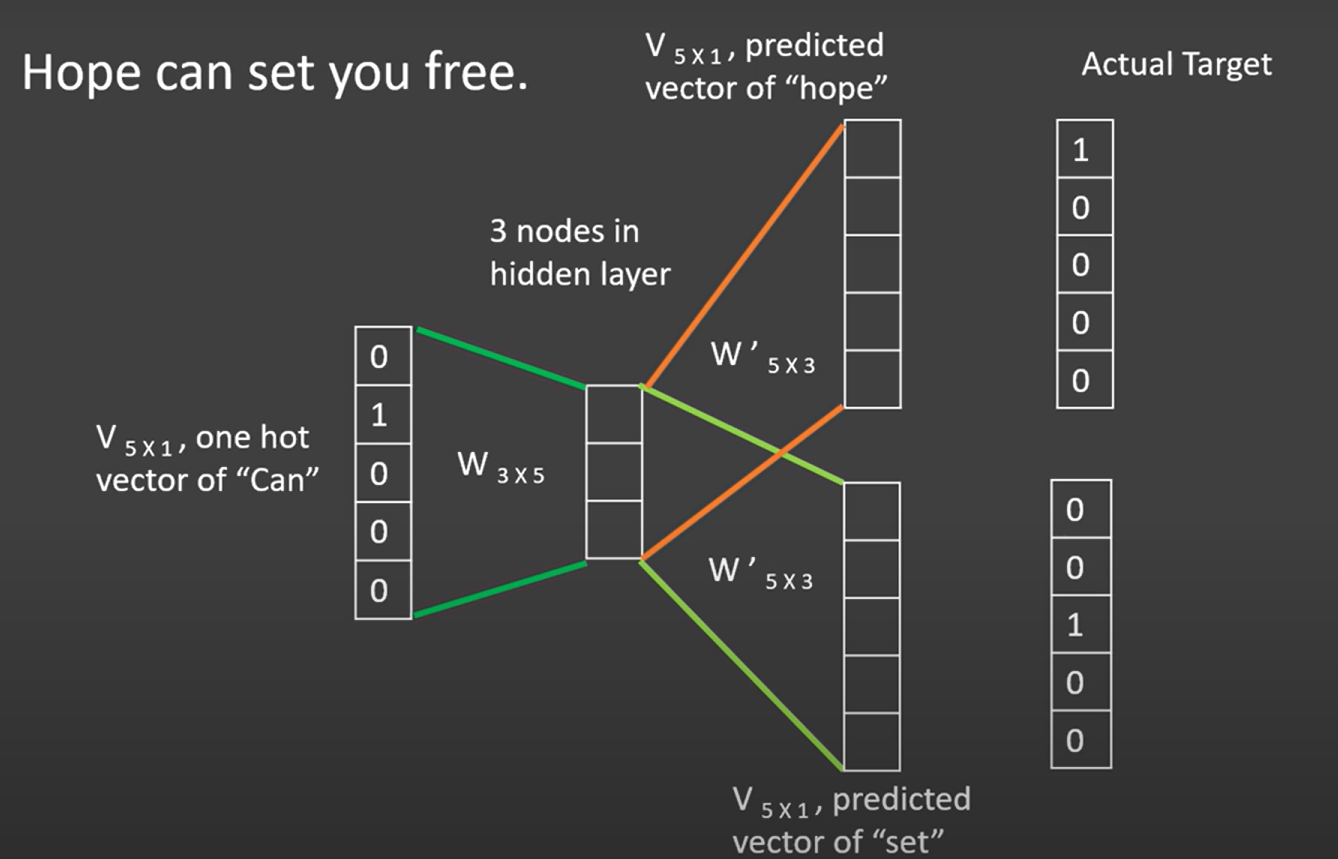
Link: https://machinelearningcoban.com/2017/02/17/softmax/

* Sau khi huấn luyện hoàn tất, ma trận W chính là ma trận nhúng từ mà chúng ta đang tìm.
* Để muốn lấy vector nhúng của 1 từ:
* 

#### Skip-gram

* Mô hình Skip-gram có cấu trúc tương tự như CBOW, nhưng mục đích của nó là dự đoán ngữ cảnh của một từ đưa vào.
* Skip-gram thường sẽ đi chung với phương pháp Lấy mẫu phủ định (negative sampling).
* https://danhtran.science.blog/2019/09/30/gioi-thieu-ve-phep-nhung-tu-phan-2/
* Dưới đây là hình ảnh của mô hình Skip-gram:
* 

Ví dụ:

* 

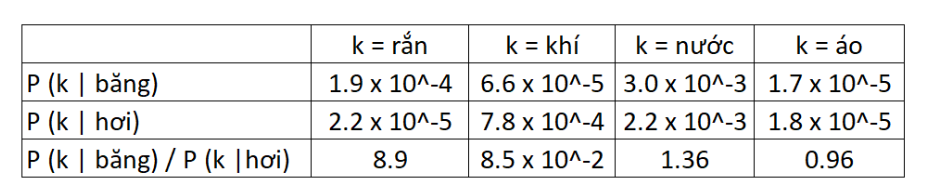
##### Cách hoạt động - Skip-gram:

* Bộ dữ liệu dùng để train và cách thức hoạt động của mô hình Skip-gram tương tự với mô hình CBOW 1 Input.
* Khác biệt chính: thay vì chỉ có 1 độ lỗi, Skip-gram có nhiều độ lỗi do có nhiều vector Output.
* Các độ lỗi này được tổng hợp lại thành 1 độ lỗi cuối cùng.
* Độ lỗi cuối cùng này được dùng để lan truyền ngược, cập nhật các trọng số.
* Các trọng số của Input-Hidden Weights Matrix sau khi học xong sẽ được sử dụng làm biểu diễn của các vector từ.

Note:

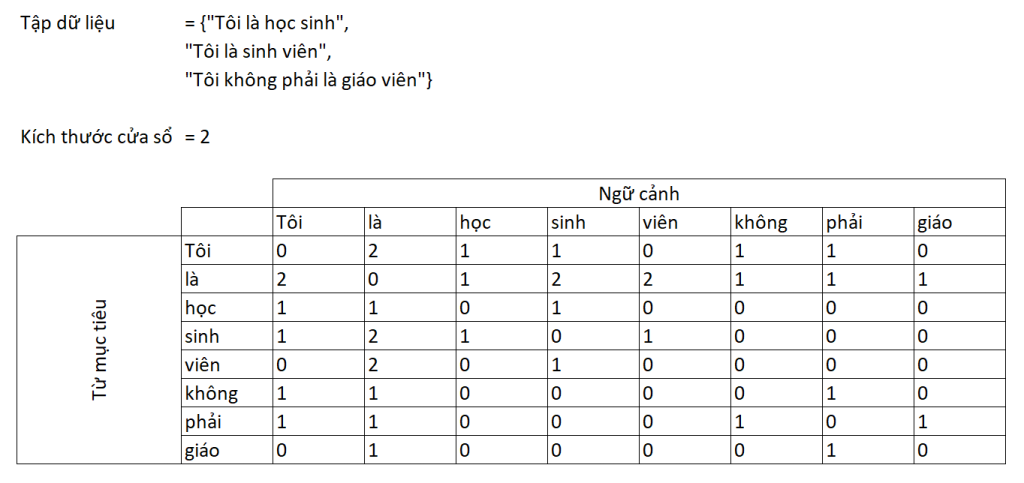
* Skip-gram sẽ tạo ra nhiều cặp từ hơn so với CBOW, dẫn đến việc phải tính toán hàm softmax trên một lượng lớn các từ, có độ phức tạp O(V), trong đó V là số lượng từ vựng.
* Để giảm chi phí tính toán, ta có thể áp dụng phương pháp negative sampling. Phương pháp này giảm đáng kể số lượng các mẫu dữ liệu cần tính toán, và thay vào đó, chuyển vấn đề từ bài toán phân loại đa lớp (multi-class classification) sang bài toán hồi quy logit (logistic regression).

## Glove

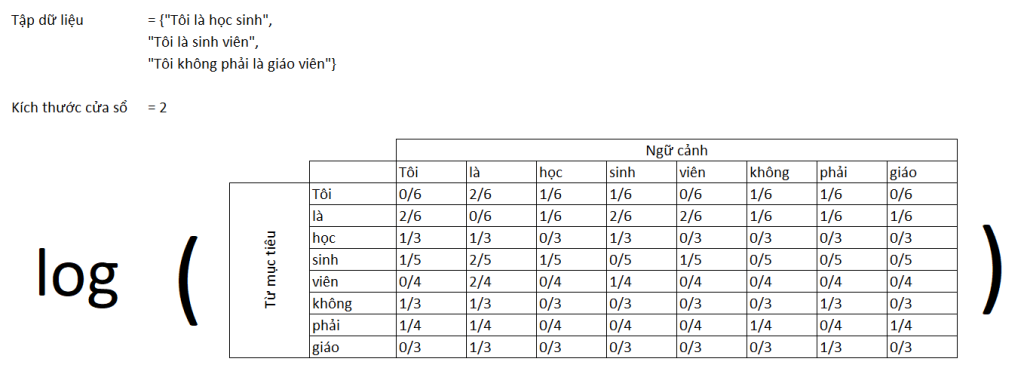
* GloVe (Global Vectors) là một thuật toán nhúng từ được phát triển bởi Đại học Stanford vào năm 2014. Ý tưởng chính của GloVe dựa trên quan sát rằng các từ có nghĩa tương tự nhau hoặc có mối quan hệ gần gũi có nhiều khả năng xuất hiện cùng nhau hơn.
* 
* Từ bảng trên, ta có thể thấy xác suất xuất hiện gần nhau của cặp từ [“băng”, “rắn”] cao hơn 2.8 lần khi so với xác suất của cặp từ [“băng”, “khí”]. Hay xác suất của cặp từ [“băng”, “áo”] rất thấp.
* Điều này phù hợp với các quan điểm như “băng là thể rắn”, “băng trái ngược với khí”, “băng không liên quan tới áo quần”.
* Do đó, mục tiêu của GloVe là học ra các vector từ ngữ để tích vô hướng của chúng sẽ bằng xác suất xuất hiện đồng thời của chúng trong tập dữ liệu, với kỳ vọng sẽ phù hợp với các quan điểm thông thường.

### Các bước huấn luyện Glove:

Bước 1: Xây dựng ma trận xuất hiện đồng thời X (co-occurrence matrix)

* 
* VD: Số lần từ “là” và “sinh” cùng xuất hiện gần nhau (count(là | sinh)) = 2 do từ “là” xuất hiện trong phạm vi cửa sổ bằng 2 ở 2 câu “Tôi là học sinh” và “Tôi là sinh viên”.

Bước 2: Chuyển thành ma trận log xác suất xuất hiện đồng thời P

* 

Link: <https://nlp.stanford.edu/projects/glove/>

## FastText

* FastText là một phương pháp học biểu diễn từ, được phát triển bởi Facebook's AI Research (FAIR).
* Tạo ra năm 2016 viết bằng ngôn ngữ C++.
* Nó có nhiều điểm tương đồng với Word2Vec nhưng cũng có những cải tiến đáng kể, giúp nó xử lý tốt hơn các vấn đề liên quan đến từ hiếm và các ngôn ngữ phức tạp.
* Ví dụ chữ “đồng”, với n-gram=3 sẽ được tách thành [“<đồng>”,”<đồ”, “đồn”, “ồng”, “ng>”].
* Dấu “<“, “>” dùng để phân biệt các n-gram với các từ hoàn chỉnh như “<đồn>”!=”đồn”.
* Sau khi huấn luyện xong, vector nhúng từ của chữ “đồng” sẽ là tổng của các vector nhúng từ [ “<đồng>”,”<đồ”, “đồn”, “ồng”, “ng>”], nhân với vector của “đồng”.

Nhờ việc tách thành n-gram như vầy mà ta có thể tạo ra được vector nhúng từ của các từ chưa từng xuất hiện trong bước huấn luyện bằng cách cộng các vector n-gram có sẵn lại.

## Pre-train