**WEEK 3: Text Embeddings**

Các kỹ thuật xử lý text:  
- Word Embedding

- Word2Vec

- Average word2vec

- Doc2Vec

Tham khảo:

1. Tài liệu BTC gửi - Lý thuyết: [The Illustrated Word2vec – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)](https://jalammar.github.io/illustrated-word2vec/?authuser=0)
2. Tài liệu BTC gửi - Document coding: [models.word2vec – Word2vec embeddings — gensim (radimrehurek.com)](https://radimrehurek.com/gensim/models/word2vec.html?authuser=0)
3. Youtube: [Word2vec - Mô hình hóa từ bằng mạng học sâu - YouTube](https://www.youtube.com/watch?v=akRbuXokLSo)
4. Đọc thêm: [Word Embedding (htrvu.github.io)](https://htrvu.github.io/post/word-embedding/)

|  |  |
| --- | --- |
| **Word Embeddings** | kỹ thuật **biểu diễn từ** dưới dạng **các vectơ số** trong không gian nhiều chiều. |
| **Word2Vec** | Sử dụng **mối quan hệ của 1 từ với những từ xung quanh** để mô phỏng từ đó trong không gian nhiều chiều.   * CBOW * Skip-gram |
| **Average word2vec** | AvgWord2Vec is an extension of Word2Vec that computes the word embeddings differently. Instead of learning context-based representations, AvgWord2Vec generates **vectors by averaging the embeddings of constituent words in a sentence.** |
| **Doc2Vec** | <https://www.geeksforgeeks.org/doc2vec-in-nlp/>   * Distributed Memory (DM) * Distributed Bag of Words (DBOW) |

Các thuật ngữ liên quan:

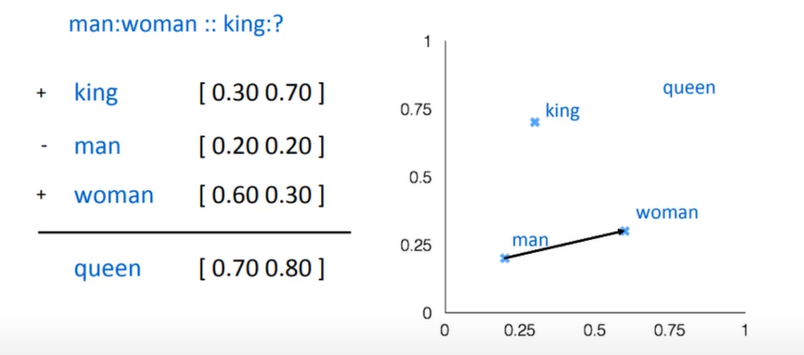
* Distributional hypothesis: các từ xuất hiện trong cùng 1 ngữ cảnh thường có nghĩa tương tự nhau.
* pre-train: quá trình xây các word embeddings.

—--------------------------------------------------------------

**Cosine\_similarity**: đánh giá similarity score

([Cosine similarity - Wikipedia](https://en.wikipedia.org/wiki/Cosine_similarity))

|  | Đánh giá mức độ hướng nội/ ngoại của 1 người tên Jay trên thang điểm 0->100 => switch sang thang (-1;1) dựa trên 1 feature (#trait1) |
| --- | --- |
|  | Thêm 1 feature khác: #trait1, #trait2 (add another dimension)  => Vector này đại diện cho personality của 1 người. |
|  | Tìm someone có similar persionality. (Add thêm Person #1 và Person #2) |
|  | Dùng cosine\_similarity để đánh giá điểm tương đồng của Person#1 và Person#2 so với người tên Jay.  => Kết quả: Person#1 có điểm tương đồng cao nhất. (Vector màu xanh lá) |
|  | 2 dimensions thì ko đủ thông tin.  Researchers suggest 5 dimensions.  => Problem: lose the ability to draw neat little arrows in two dimensions.  cosine\_similarity still works. It works with any number of dimensions: |
| 2 central ideas: |  |



**Analogies (Phép loại suy)**

|  |  |
| --- | --- |

**Language Model**

|  |
| --- |
|  |
|  |
|  |
|  |

**Language Model Training**

|  |
| --- |

**Word Embeddings**

* Xuất hiện trong các ngữ cảnh giống nhau thì vectơ sẽ được mô phỏng tương đồng nhau.

|  | VD:  - "king" và "queen" nằm gần nhau vì chúng có cùng giới tính.  - "walked" và "swimming" nằm gần nhau vì chúng cùng biểu thị hành động. |
| --- | --- |

=> Word2vec is a group of related models that are used to produce word embeddings.

|  | 1.Có 1 đường thẳng màu đỏ thông qua các từ (queen, woman, girl, boy, man,...)  2.”woman” và “girl” tương đồng nhau ở 1 số dimension. “man” và “boy” too.  3. …  4.Add thêm “water” => blue column từ trên -> xuống & dừng lại tại từ “water” => different between categories.  5….. |
| --- | --- |

**Word2vec**

Model architecture: Input layer, Hidden layer, Output layer

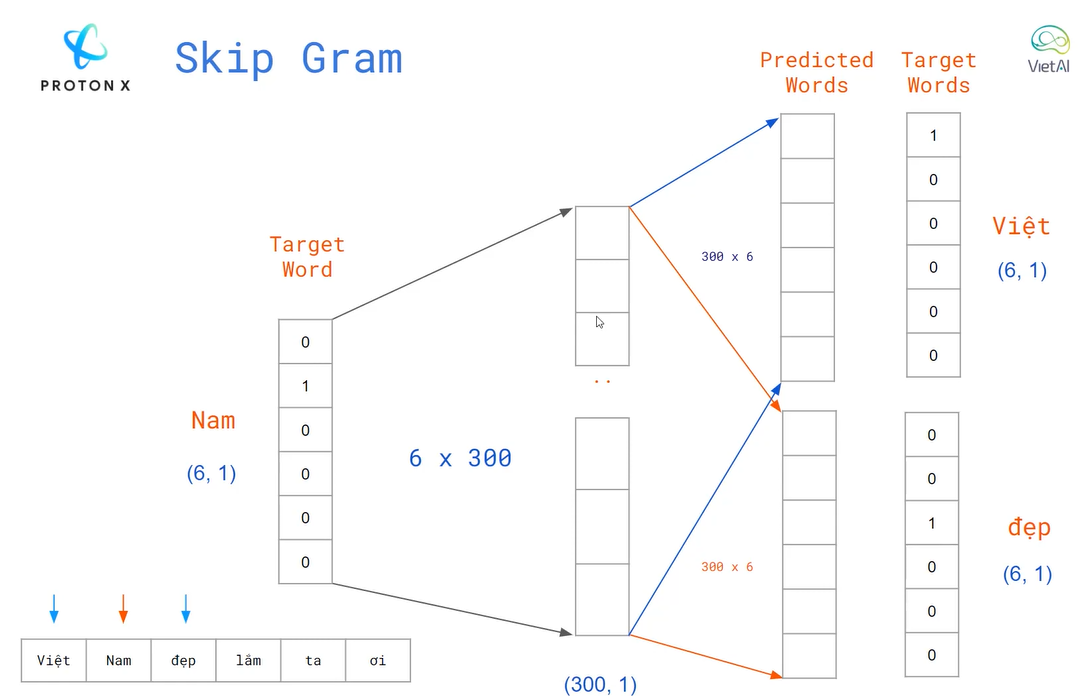
* Input layer: Mỗi từ được biểu diễn dưới dạng vector one-hot. (Một vector có độ dài bằng kích thước dict & chỉ có 1 phần tử = 1, còn lại là 0)
* Hidden layer: lớp embedding. Mỗi neuron trong layer đại diện cho 1 dimension của vector space. Khi truyền đến Hidden layer, vector one-hot sẽ chuyển đổi thành embedding vector.
* Output layer: Dự đoán từ tiếp theo/ xung quanh dựa trên embedding vector.

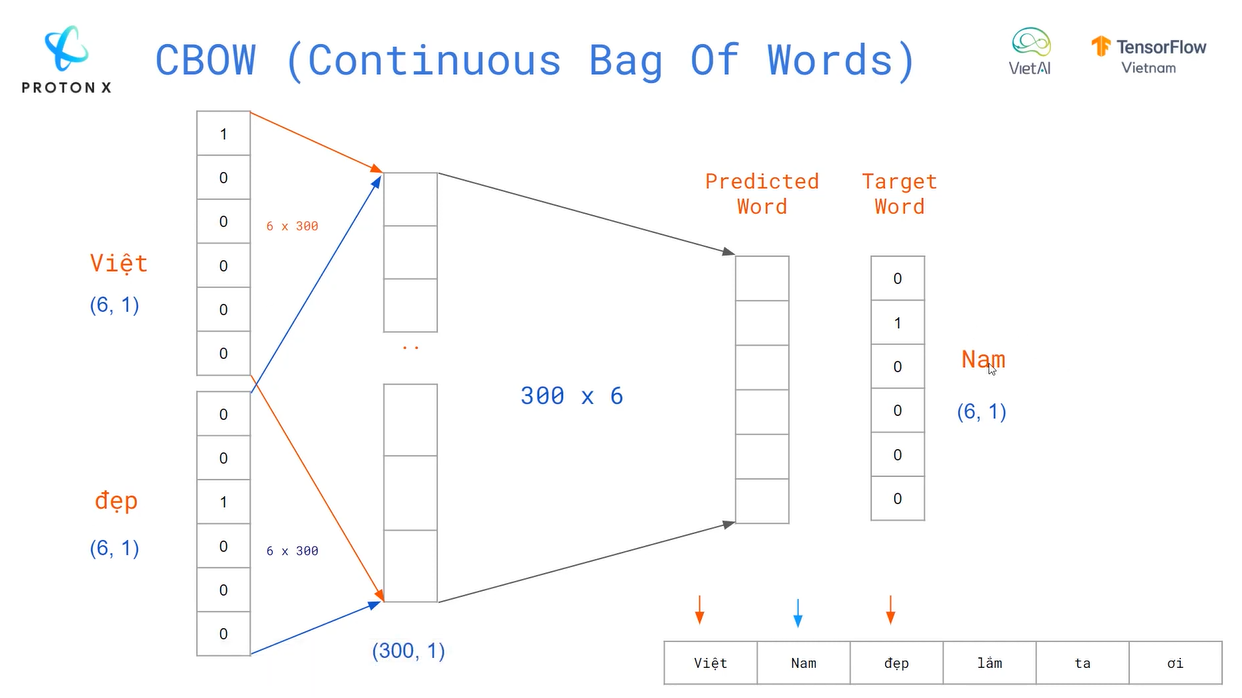
**Train model Word2Vec**

| **Skip Gram** | **CBOW (Continuous Bag of Words)** |
| --- | --- |
| Predict the **context words** from **the target word.** | Predict the **target word** from **context words.** |
|  |  |
|  |  |

|  |
| --- |

**One hot Vector**

****

****

**The training process**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**Negative Sampling**

|  | One way is to split our target into two steps:  1.Generate high-quality word embeddings (Don’t worry about next-word prediction).  2.Use these high-quality embeddings to train a language model (to do next-word prediction). |
| --- | --- |
|  |  |
| “achieving 100% accuracy, but learning nothing and generating garbage embeddings.”  => To address this, |  |

**Word2vec Training Process**

| Pre-process | vocab\_size: 10,000 |  |
| --- | --- | --- |
| Training phase | Create 2 matrices:   * Embedding matrix * Context matrix. |  |
|  | initialize these matrices with random values |  |
|  | 4 words:   * input: not * output: * Actuar neighbor: aaron * negative example: taco |  |
|  | take the dot product of the input embedding with each of the context embeddings  => result in a number (the similarity of the input and context embeddings) |  |
|  | turn these scores into probabilities  => **all be positive**  & **values between 0 and 1** |  |
|  | calculate how much **error** is in the model’s prediction  error = target - sigmoid\_scores |  |
|  | use this error score to adjust the embeddings of not, thou, aaron, and taco |  |
|  | * proceed to next step (the next positive sample and its associated negative samples) * do the same process again |  |
|  | The embeddings continue to be improved while we cycle through our entire dataset for a number of times | We can then **stop** the **training process**, **discard** the **Context matrix**, and use the **Embeddings matrix** as our pre-trained embeddings for the next task. |

**Window Size and Number of Negative Samples**

| **2 key hyperparameters**  in the word2vec training proces | **window size**   * phạm vi các từ xung quanh target word.   Size:   * **2->15**: mqh ngữ nghĩa chặt chẽ các từ gần nhau * **15->30**: mqh rộng hơn, bao gồm các từ có liên quan theo chủ đề. |  |
| --- | --- | --- |
| **number of negative samples** |  |

—--------------- **EXERCISE: TEXT EMBEDDINGS** ---------------------

Implement Word2Vec on a text dataset and report the embeddings visualization.