**Loi noi dau**

**Phan loai van ban**

**Cai dat moi truong**

# PHÂN LOẠI VĂN BẢN

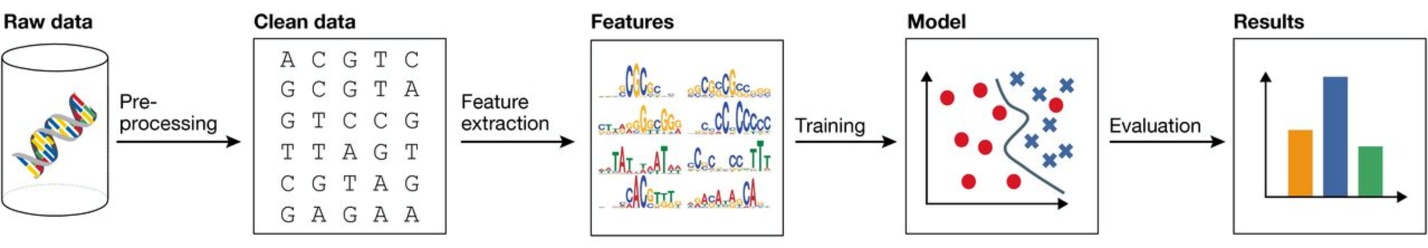
**Mô hình chung cho các bài toán Machine Learning**

Phần lớn các bài toán Machine Learning có thể được thể hiện trong hình vẽ dưới đây:

****

**Natural Language Processing - NLP**

Xử lý ngôn ngữ tự nhiên (natural language processing - NLP) là một nhánh của trí tuệ nhân tạo tập trung vào các ứng dụng trên ngôn ngữ của con người. Trong trí tuệ nhân tạo thì xử lý ngôn ngữ tự nhiên là một trong những phần khó nhất vì nó liên quan đến việc phải hiểu ý nghĩa ngôn ngữ công cụ hoàn hảo nhất của tư duy và giao tiếp. (wikipedia)



Và bước đầu tiên và không thể thiếu trong việc xử lý ngôn ngữ tự nhiên là tiền xử lý. Vì văn bản vốn dĩ được liệt kê mà không có cấu trúc, để nguyên vậy để xử lý là rất khó khăn. Đặc biệt là loại văn bản trên web có lẫn các HTML tag, code JS, đó chính là noise.

Phân loại văn bản là gì?

**Phân loại văn bản** hay còn gọi là **Text Classifcation** hoặc là **Text Categorizer** là một bài toán thuộc về lĩnh vực **Xử lý ngôn ngữ tự nhiên** dưới dạng văn bản (text).

**Tiền Xử Lý**

A screenshot of a cell phone

Description automatically generated

**Làm sạch text**

Mục đích bước này là loại bỏ noise trong data. Đa phần noise là các thẻ HTML, JavaScript, và đương nhiên nếu cứ để noise để tiến hành xử lý sẽ dẫn đến kết quả xử lý không tốt.

A picture containing drawing

Description automatically generated

Thông thường chúng ta hay loại bỏ noise là các thẻ HTML và JS như trên tuy nhiên thực tế noise có thể không chỉ là HTML, JS, cũng có thể là những cụm từ không cần thiết, hay ký tự không có ý nghĩa ($%&##"). Với code Python, [BeautifulSoup](https://www.crummy.com/software/BeautifulSoup/bs4/doc/) và [lxml](http://lxml.de/) là 2 thư viện được cộng đồng sử dụng nhiều nhất và vô cùng mạnh mẽ, tiện lợi.

* 1. **Tách từ (Words segmentation)**

Trong tiếng Việt, dấu cách (space) không được sử dụng như 1 kí hiệu phân tách từ, nó chỉ có ý nghĩa phân tách các âm tiết với nhau. Vì thế, để xử lý tiếng Việt, công đoạn tách từ (word segmentation) là 1 trong những bài toán cơ bản và quan trọng bậc nhất.

Ví dụ : từ “đất nước” được tạo ra từ 2 âm tiết “đất” và “nước”, cả 2 âm tiết này đều có nghĩa riêng khi đứng độc lập, nhưng khi ghép lại sẽ mang một nghĩa khác. Vì đặc điểm này, bài toán tách từ trở thành 1 bài toán tiền đề cho các ứng dụng xử lý ngôn ngữ tự nhiên khác như phân loại văn bản, tóm tắt văn bản, máy dịch tự động, …

Như ví dụ sau:

A screenshot of a cell phone

Description automatically generated

Tách từ chính xác hay không là công việc rất quan trọng, nếu không chính xác rất có thể dẫn đến việc ý nghĩa của câu sai, ảnh hưởng đến tính chính xác của chương trình.

**Chuẩn hóa từ**

Mục đích là đưa văn bản từ các dạng không đồng nhất về cùng một dạng. Dưới góc độ tối ưu bộ nhớ lưu trữ và tính chính xác cũng rất quan trọng.

Ví dụ: U.S.A = USA

Ví dụ trong từ điển, training data của chúng ta không có U.S.A, chỉ có USA, thì việc convert những từ như U.S.A về USA là điều cần thiết để các bước xử lý sau như text classification, intent detection được chính xác.

Ngoài ra với tiếng Nhật: 猫＝ねこ＝ネコ(full size)＝ﾈｺ(half size)

Có nhiều cách viết, mỗi cách viết khi lưu trữ sẽ tốn lượng memory khác nhau, như half size chỉ tốn 1/2 dung lượng so với full size nên tuỳ theo nhu cầu, tình hình thực tế, chúng ta sẽ đưa văn bản về 1 dạng đồng nhất.

Ngoài ra trong một vài trường hợp, nếu ký tự số không mang lại lợi ích gì thì cũng sẽ tiến hành loại bỏ các ký tự số đó, nếu cứ để nguyên rất có thể các ký tự số sẽ trở thành noise, ảnh hưởng đến tính chính xác của model sau này.

1. **Loại bỏ StopWords**

StopWords là những từ xuất hiện nhiều trong ngôn ngữ tự nhiên, tuy nhiên lại không mang nhiều ý nghĩa. Ở tiếng việt StopWords là những từ như: để, này, kia... Tiếng anh là những từ như: is, that, this...

Có rất nhiều cách để loại bỏ StopWords nhưng có 2 cách chính là:

* Dùng từ điển
* Dựa theo tần suất xuất hiện của từ

Dùng từ điển

Cách này đơn giản nhất, chúng ta tiến hành filter văn bản, loại bỏ những từ xuất hiện trong từ điển StopWords:



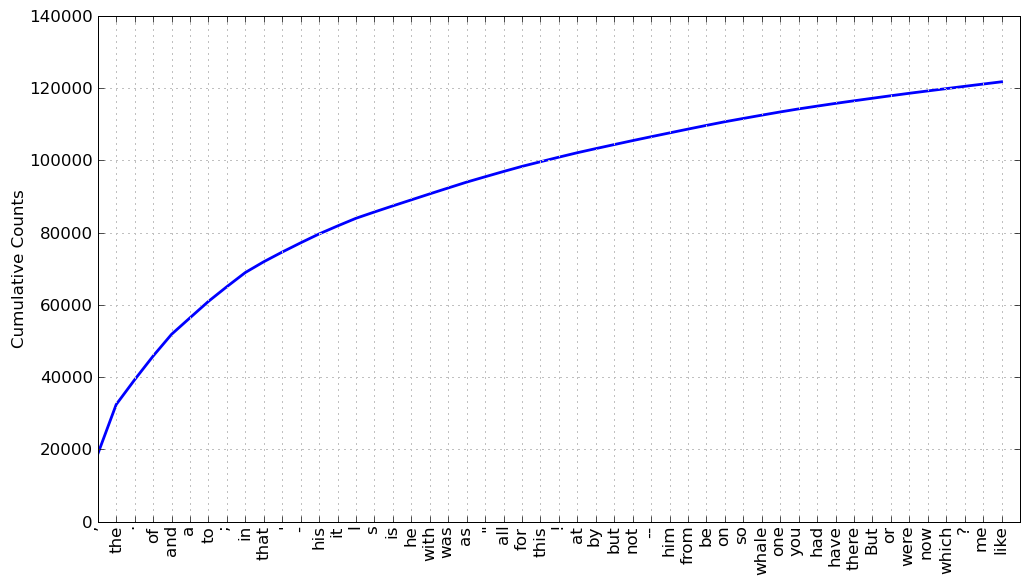
A screenshot of a cell phone

Description automatically generated

**Dựa theo tần suất xuất hiện của từ**

Với cách này, chúng ta tiến hành đếm số lần xuất hiện của từng từ trong data sau đó sẽ loại bỏ những từ xuất hiện nhiều lần (cũng có thể là ít lần). Khoa học đã chứng minh những từ xuất hiện nhiều nhất thường là những từ không mang nhiều ý nghĩa.

Như ví dụ dưới đây:



Trên là top 50 từ xuất hiện nhiều nhất trong mỗi cuốn sách, dễ dàng nhận thấy chúng không mang nhiều ý nghĩa. Chính vì thế chúng ta sẽ loại bỏ những từ như thế này.

1. **Vector hoá từ**

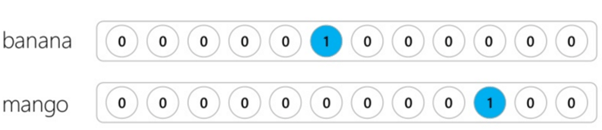
Thông thường, máy tính không thể hiểu được ý nghĩa các từ. Như vậy, để xử lý được ngôn ngữ tự nhiên, ta cần có một phương pháp để biểu diễn văn bản dưới dạng mà máy tính có thể hiểu được. Phương pháp tiêu chuẩn để biểu diễn văn bản đó là biểu diễn các văn bản theo vector. Trong đó, các từ/cụm từ thuộc kho tài liệu ngôn ngữ được ánh xạ thành những vector trên hệ không gian số thực.

Ta có thể chia các phương pháp Vector hóa văn bản thành hai nhóm chính: Phương pháp Word Embeddings cổ điển và Neural Embeddings (Vector hóa văn bản theo phương pháp mạng nơ-ron).

**Phương pháp Word Embeddings cổ điển**

**Bag of Words (BoW)**

Đây là cách biểu diễn vector truyền thống phổ biến nhất được sử dụng. Mỗi từ hoặc n-gram từ sẽ được mô tả là một vector có số chiều bằng đúng số từ trong bộ từ vựng. Tại vị trí tương ứng với vị trí của từ đó trong túi từ, phần tử trong vector đó sẽ được đánh dấu là 1. Những vị trí còn lại sẽ được đánh dấu là 0.

Ví dụ biểu diễn One-hot BOW của mỗi từ trong văn bản.

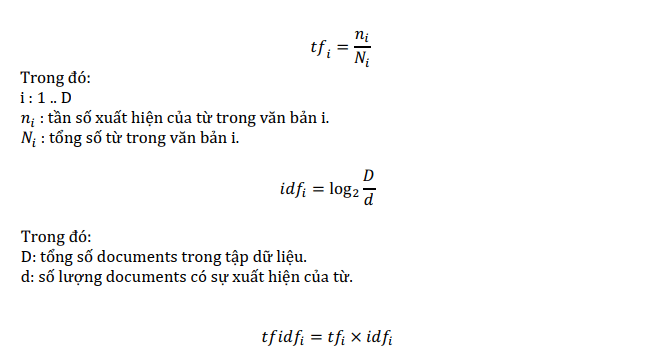
Phương pháp BoW thường được sử dụng trong những bài toán phân loại văn bản. Trong đó, tần suất của mỗi từ/ n-gram sẽ được coi là một feature trong văn bản phân loại.

Nhược điểm của phương pháp này là ta không thể xác định được nghĩa thực của mỗi từ và các từ tương quan với chúng.

Trong phương pháp BoW, từ giống nhau sẽ được đánh trọng số như nhau. Phương pháp này không xét đến tần suất xuất hiện của từ hay nhữ cảnh từ. Và trong thực tế, để cần hiểu được nghĩa của mỗi từ, ta cần xác định từ đó trong văn cảnh hơn là xét nghĩa độc lập từ.

**TF-IDF**

TF- IDF (term frequency–inverse document frequency) – tần suất- tần suất đảo nghịch từ. Đây là một phương pháp thống kê, nhằm phản ánh độ quan trọng của mỗi từ hoặc n-gram đối với văn bản trên toàn bộ tài liệu đầu vào. TF-IDF thể hiện trọng số của mỗi từ theo ngữ cảnh văn bản. TF-IDF sẽ có giá trị tăng tỷ lệ thuận với số lần xuất hiện của từ trong văn bản và số văn bản có chứa từ đó trên toàn bộ tập tài liệu. Phương pháp này giúp cho **TF-IDF**có tính phân loại cao hơn so với phương pháp trước.



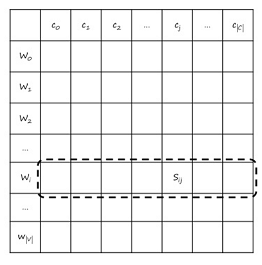
Tuy nhiên, ngay cả khi phương pháp TF-IDF dựa trên BOW thể hiện được trọng số của các từ khác nhau trong văn bản, nhưng phương pháp này vẫn không biểu diễn được nghĩa của từ.

Đây chính là nhược điểm của hai phương pháp này.

Ta có thể trích dẫn câu của nhà ngôn ngữ học J.R. Firth: “The complete meaning of a word is always contextual, and no study of meaning apart from context can be taken seriously.” (tạm dịch: “Muốn hiểu được ý nghĩa thật sự của một từ, bạn phải dựa vào ngữ cảnh của câu nói”)

**Distributional Embeddings**

Là phương pháp mà ta có thể xem xét được tổng quan trong toàn bộ ngữ cảnh. Mỗi từ sẽ được biểu diễn trên các thông tin tương hỗ (Mutual Information) với các từ khác trong tập dữ liệu. Thông tin tương hỗ có thể được biểu diễn dưới dạng tần suất xuất hiện trong ma trận đồng xuất hiện trên toàn bộ tập dữ liệu hoặc xem xét trong giới hạn tập dữ liệu lân cận hoặc xem xét trên giới hạn những từ xung quanh.



Ví dụ ma trận thuật toán Distributional Embeddings với mỗi dòng là một văn bản (ngữ cảnh) dựa trên số từ xuất hiện

Phương pháp Distributional Embeddings ra đời trước phương pháp Neural Embeddings. Nhưng các phương pháp Distributional Embeddings giúp ta quan sát được quan trọng của mỗi từ tốt hơn so với Neural Embeddings.

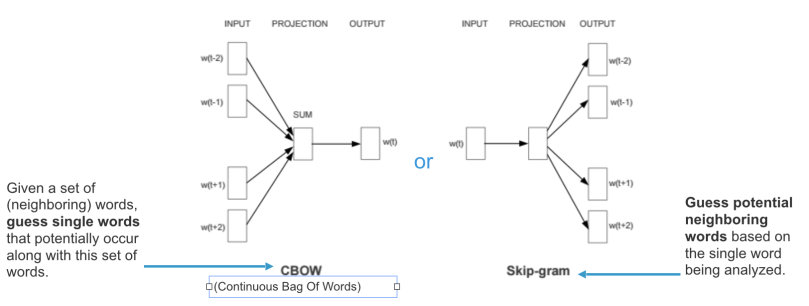
**Phương pháp Neural Embeddings**

**Word2Vec**

Word2vec là thuật toán theo phương pháp dự đoán (Prediction-based embedding). Mô hình dự đoán học biểu diễn vector từ thông qua những từ ngữ cảnh xung quanh nhằm cải thiện khả năng dự đoán ý nghĩa các từ.

Có hai cách xây dựng mô hình Word2vec để biểu diễn sự phân tán của từ trong không gian vector:

* Sử dụng ngữ cảnh để dự đoán mục tiêu (CBOW): khi vị trí của các từ ngữ cảnh không ảnh hưởng tới việc dự đoán từ (giả định ban đầu của CBOW). Trong mô hình Skip-gram, mô hình sử dụng từ ngữ hiện tại để dự đoán những từ xung quanh trong ngữ cảnh đó.
* Sử dụng một từ để dự đoán ngữ cảnh mục tiêu (Continuous skip-gram) xem xét những từ ngữ cảnh xung quanh sẽ được đánh giá tốt hơn so với những từ trong ngữ cảnh nhưng ở vị trí xa hơn. Mặc dù thứ tự từ vẫn không được xem xét, mỗi vector của từ bối cảnh được xem xét và cân nhắc.



Thuật toán CBOW tốn ít thời gian huyến luyện mô hình hơn Skip-gram. Tuy nhiên, Skip-gram có độ chính xác cao hơn và có chứa cả những từ ít xuất hiện.

**GloVe**

Cả CBOW và Skip-Gram đều là các mô hình dự đoán. Trong đó, các thuật toán chỉ xem xét được ngữ cảnh xung quanh từ mục tiêu nhưng không đề cập được về ngữ cảnh toàn văn bản. Thuật toán GloVe dựa trên tương phản có lợi với cùng dự đoán của ma trận đồng xuất hiện sử dụng trong thuật toán Distributional Embeddings, nhưng sử dụng phương pháp Neural Embeddings để phân tích ma trận đồng xuất hiện thành những vector có ý nghĩa và tỷ trọng hơn.

Mặc dù thuật toán GloVe nhanh hơn Word2Vec, nhưng cả GloVe và Word2Vec đều không hiển thị để cung cấp kết quả tốt và rõ ràng hơn thay vì cả hai nên được đánh giá cho một tập dữ liệu nhất định.

**FastText**

FastText, được xây dựng trên Word2Vec bằng cách học các biểu diễn vectơ cho mỗi từ và n-gram được tìm thấy trong mỗi từ. Các giá trị của các biểu diễn sau đó được tính trung bình thành một vectơ ở mỗi bước đào tạo. Trong khi điều này bổ sung rất nhiều tính toán bổ sung cho việc đào tạo, nó cho phép nhúng từ để mã hóa thông tin từ phụ. Các vectơ FastText đã được chứng minh là chính xác hơn các vectơ Word2Vec bằng một số biện pháp khác nhau.

**Báo cáo tuần 2:**

Môi trường cài đặt:

* Python 3.8 trên window

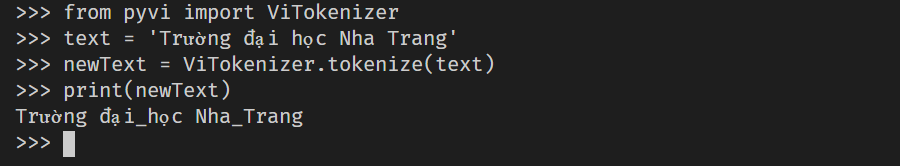
Các thư viên hổ trợ tiền xử lý văn bản:

**Tách từ**

Một số thư viện hỗ trợ tách từ như:

* [vnTokenizer](http://vlsp.hpda.vn:8080/demo/?page=resources) - an automatic tokenizer for segmenting Vietnamese texts into lexical units by Lê Hồng Phương (2008)
* [DongDu](https://github.com/rockkhuya/DongDu) - A Vietnamese word segmentation tool by Luu Tuan Anh (2014)
* [UETsegmenter](https://github.com/phongnt570/UETsegmenter) - a toolkit for Vietnamese word segmentation by Nguyen Tuan Phong (2016)
* [Vitk](https://github.com/phuonglh/vn.vitk) - A Vietnamese Text Processing Toolkit by Le Hong Phuong (2016)
* [pyvi](https://pypi.python.org/pypi/pyvi) - Python Vietnamese Toolkit by Tran Viet Trung (2016)

Trong tài liệu này sử dụng công cụ **ViTokenizer** trong **pyvi** để tách từ, trong thư viện pyvi có 1 file words.txt chứa danh sách các từ trong tiếng việt có 1 hoặc nhiều âm tiết, từ đó **ViTokenizer** sẽ dựa vào file này để tách từ, nếu từ có trên 2 âm tiết nó sẽ nói với nhau bằng dấu gạch dưới “\_”. Ví dụ:

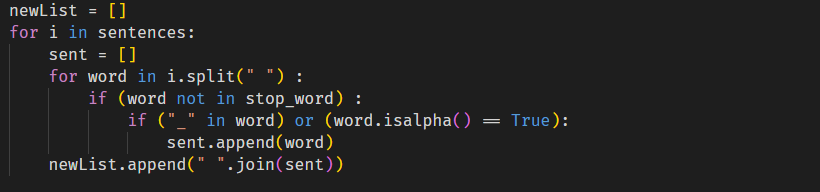


Để cài đặt pyvi, ta chạy lệnh:



**Loại bỏ từ dừng (Stopword)**

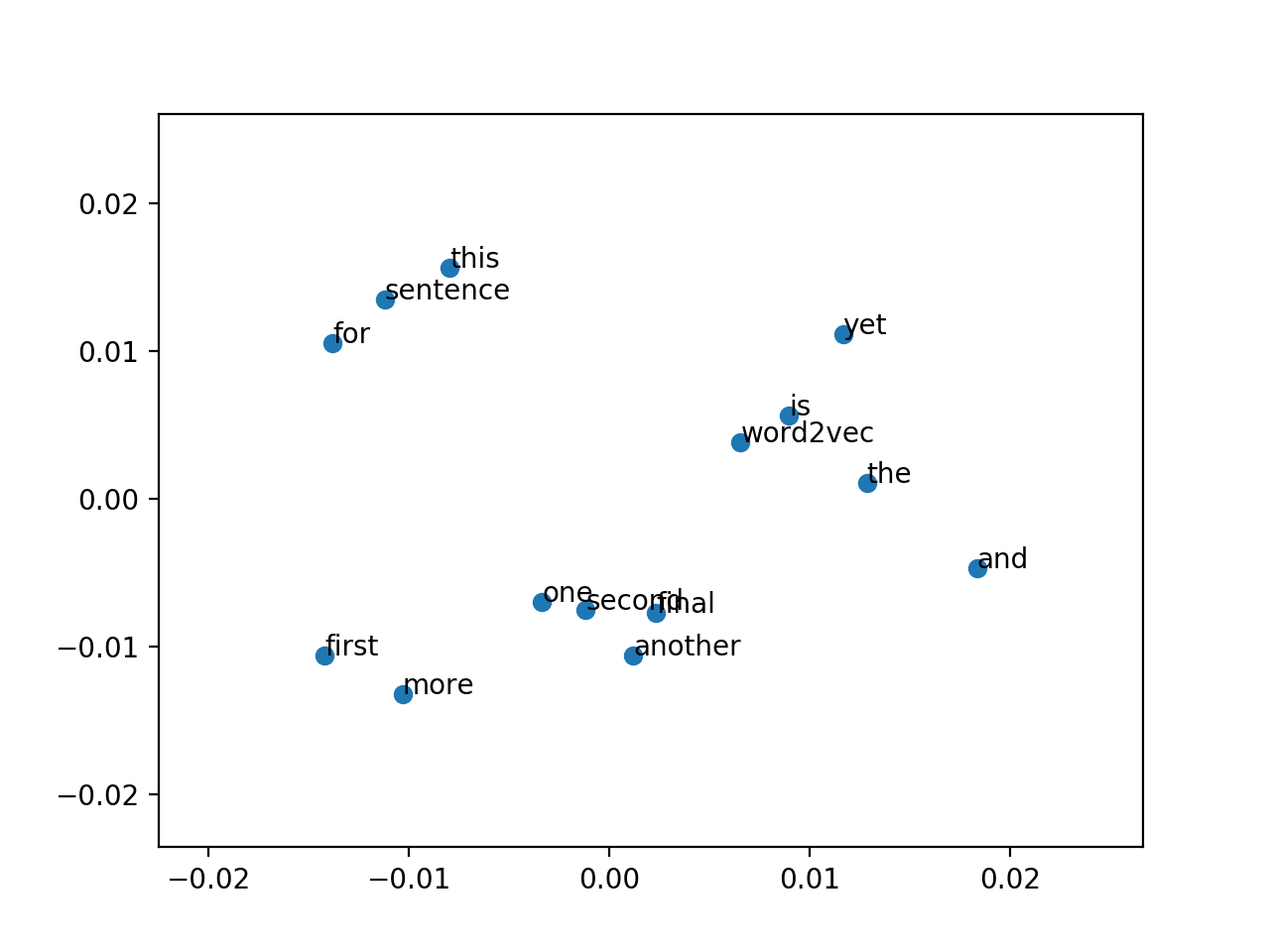
Trong tài liệu này sử dụng phương pháp từ điển để loại bỏ nhưng từ không cần thiết trong tiếng việt. Tạo một file txt chứa danh sách các từ cần loại bỏ hoặc tải [vietname-stopwords](https://github.com/stopwords/vietnamese-stopwords/blob/master/vietnamese-stopwords-dash.txt) để sử dụng.



**Vector đặc trưng**

Kỹ thuật được sử dụng ở đây là word2vec bằng thử viện **gensim.**

Gensim là một thư viện dùng để mô hình hóa, lập chỉ mục tài liệu và rút trích tính tương tự với ngữ liệu (corpora) lớn. Đối tượng sử dụng Gensim chủ yếu là cộng đồng XLNNTN và rút trích thông tin.

Thể hiện kết quả bài toán word2vec bằng thư viện Gensim.

Gensim bao gồm các thực thi song song các thuật toán như fastText, [word2vec](https://www.dammio.com/2019/09/08/cac-thu-vien-python-tot-nhat-dung-de-xu-ly-ngon-ngu-tu-nhien) và doc2vec, cũng như các phân tích ngữ nghĩa (LSA, LSI, SVD), phân rã trận không âm (NMF), phân bổ Dirichlet tiềm ẩn (LDA), các phép chiếu ngẫu nhiên và tf-idf.

Để cài **gensim** chạy lệnh:



# Tài liệu tham khảo:

<https://trituenhantao.io/kien-thuc/word-embeddings-cac-phuong-phap-vector-hoa-van-ban/>

<https://trituenhantao.io/kien-thuc/gioi-thieu-ve-doc2vec/>

<https://machinelearningcoban.com/general/2017/02/06/featureengineering/>

<https://viblo.asia/p/feature-engineering-phan-4-phuong-phap-xu-ly-truyen-thong-voi-du-lieu-dang-van-ban-text-data-1Je5EvWYKnL>