**ĐẠI HỌC THÁI NGUYÊN**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

****

**BÁO CÁO**

**Môn học: Xử Lý Ngôn Ngữ Tự Nhiên**

**Đề tài:**

**XÂY DỰNG THUẬT TOÁN TÓM TẮT ĐƠN VĂN BẢN**

**Nhóm Sinh viên thực hiện: 1. Lê Quang Chính**

**2. Nguyễn Như Thuần**

**3. Nguyễn Thành Đông**

**4. Phạm Đức Phong**

**THÁI NGUYÊN, NĂM 2021**

# 

# LỜI MỞ ĐẦU

**Việc tóm tắt văn bản ngày càng trở nên cần thiết!**

Ngày nay, mỗi giây trôi qua đều có một lượng thông tin khổng lồ được sinh ra, nhưng giới hạn về thời gian, về khả năng đọc và tiếp thu của con người là có hạn, việc hiểu và nắm bắt thật nhiều thông tin một cách nhanh chóng không phải là vấn đề đơn giản với bất kỳ ai.

Việc mọi người tìm kiếm các kiến thức trên internet, hay đọc một cuốn sách mà nội dung của nó dài "lê thê", khó nắm bắt được ý chính khiến cho mọi người cảm thấy khó khăn để có thể nắm bắt được kiến thức mà mình muốn tiếp thu nhất.

Trước xu hướng con người ngày càng mất nhiều thời gian đọc email, báo điện tử và mạng xã hội, các thuật toán sử dụng machine learning để tự động tóm tắt các văn bản dài một cách ngắn gọn và chính xác ngày càng trở nên cần thiết và có vai trò to lớn đối trong bất kỳ lĩnh vực nào.

Tự động tóm tắt sẽ là một trong những công nghệ quan trọng có thể giúp con người giảm thiểu thời gian đọc email và thông tin, kiến thức mới để dành thời gian cho các công việc khác, mà vẫn có thể nắm bắt được ngắn gọn những nội dung của nó.

Hiện nay, rất nhiều thuật toán cho việc tóm tắt đã và đang được các công ty, các nhà nghiên cứu phát triển. Trong số đó, phương pháp áp dụng những thuật toán cơ bản nhất của học máy (Machine Learning) hay xử lý ngôn ngữ tự nhiên (Natural Language Processing) trở thành một phương pháp cực kỳ đơn giản và có thể dễ dàng nắm bắt.

Bố cục bài báo cáo gồm các phần sau:

**Chương 1: Cơ sở lý thuyết và công cụ sử dụng**

Trình bày các kiến thức tổng quan về Maching leaning, các tính năng vượt trội khi sử dụng và các phương thức được sử dụng và phát triển của Maching leaning.

**Chương 2: Khảo sát và đặt vấn đề bài toán tóm tắt văn bản**

Giới thiệu về tầm quan trọng của tóm tắt văn bản, đưa ra ý tưởng phát triển của thuật toán. Sơ đồ hoạt động, mục đích xây dựng của thuật toán và bài toán cần giải quyết.

**Chương 3: Phân tích thiết kế thuật toán tóm tắt văn bản**

Phân tích thiết kế thuật toán và luồng xử lý. Bao gồm các bước xử lý phân tích và hình ảnh mô tả cấu trúc của từng bộ phận trong thuật toán.

**Chương 4: cài đặt chương trình**

Gồm các hình ảnh mô tả giao diện của hệ thống, cách hoạt động. Kết luận những gì hệ thống đạt được cũng như những chỉnh sửa, thay đổi cần khắc phục.

*Thái nguyên, ngày…tháng…năm 2021*

Sinh viên thực hiện

1. Lê Quang Chính

2. Nguyễn Như Thuần

3. Nguyễn Thành Đông

4. Phạm Đức Phong

# CHƯƠNG I: CƠ SỞ LÝ THUYẾT VÀ CÔNG CỤ SỬ DỤNG

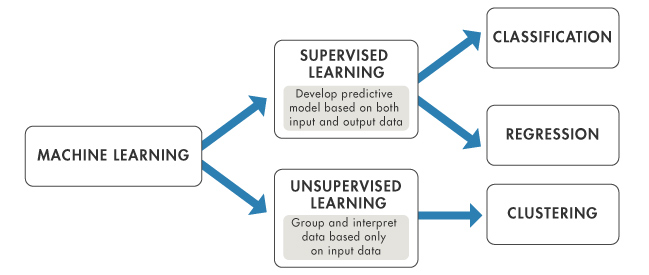
## 1.1. Khái quát về Machine learning

Vài năm gần đây, việc xuất hiện các cụm từ "cách mạng công nghiệp lần thứ 4" hay "cách mạng công nghiệp 4.0", tập trung chủ yếu vào sản xuất thông minh dựa trên sự phát triển đột phá của cách ngành nghề công nghệ thông tin, công nghệ sinh học, công nghệ nano, ... Mỗi một cuộc cách mạng công nghệ đều sẽ mang đến một bước ngoặt lớn với cách thức chúng ta sản xuất, lao động.

Thực chất thì tới thời điểm hiện tại, vẫn chưa có một định nghĩa thống nhất cho ML, nhưng đa phần khi tìm tài liệu trên mạng, chúng ta sẽ thấy định nghĩa về machine learning như thế này:

“Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed.”

Định nghĩa này do Arthur Samuel đưa ra năm 1959, tạm dịch là "Machine learning là một ngành học thuộc khoa học máy tính, giúp máy tính có khả năng tự học mà không phải lập trình một cách rõ ràng".



*Hình 1.1. Mô hình Machine Learning*

## . Tìm hiểu về K-means trong Machine Learning

K-Means: Một thuật toán tuy đơn giản nhưng lại khá hiệu quả và được sử dụng rộng khắp. xác định sẵn 2 thứ: **hàm khoảng cách** được sử dụng (ví dụ như khoảng cách Euclid) và **số lượng nhóm** mong muốn (ta sẽ kí hiệu trong bài viết này là *k*)

Thuật toán bắt đầu với việc chọn ra tâm của từng cụm. Chúng ta có thể đơn giản chọn k điểm ngẫu nhiên trong bộ, hoặc sử dụng một số hướng tiếp cận nào khác, nhưng nhìn chung ngẫu nhiên vẫn là cách tốt nhất. Rồi kế tiếp, luân phiên lặp lại 2 giai đoạn sau:

1. **Giai đoạn gán**: gán từng phần tử trong bộ dữ liệu của chúng ta vào các cụm. Cách thức tiến hành đó là: với mỗi điểm, hãy tính khoảng cách từ điểm đó tới vị trí các tâm, sau cùng: tâm nào gần nhất thì gán vào cụm ứng với cái tâm đó
2. **Giai đoạn cập nhật**: duyệt từng cụm, cập nhật lại tọa độ của tâm: Như đã biết, sau giai đoạn 1, chúng ta đã thu được k cụm ứng với dãy các điểm được gán cho từng cụm. Tọa độ tâm mới của cụm sẽ bằng trung bình cộng tọa độ các điểm trong cụm

Sau càng nhiều vòng lặp, các tâm càng di chuyển chậm dần, và tổng khoảng cách từ mỗi điểm trong cụm tới tâm cụm lại càng nhỏ đi. Quá trình sẽ kết thúc cho tới khi hàm tổng khoảng cách hội tụ (tức là không có sự thay đổi nào xảy ra ở giai đoạn gán nữa). Lúc này tọa độ tâm vẫn sẽ bằng trung bình cộng các điểm hiện tại trong cụm, hay nói cách khác tâm sẽ không còn di chuyển tiếp nữa.

Chú ý thuật toán K-Means chỉ đảm bảo được quá trình này sẽ đưa hàm tổng khoảng cách hội tụ tới điểm cực tiểu địa phương, chứ không chắc chắn đó là giá trị nhỏ nhất của toàn bộ hàm số.

Tuy nhiên, điều này là có thể chấp nhận được vì **KHÔNG** phải mô hình nào càng sát với bộ dữ liệu huấn luyện thì cũng sẽ càng tốt.

## 1.3*.* Tìm hiểu "**Từ sang vector**" (Word2Vec).

Word2vec biểu diễn các từ dưới dạng một phân bố quan hệ với các từ còn lại (distributed representation). Mỗi từ được biểu diễn bằng một vector có các phần tử mang giá trị là phân bố quan hệ của từ này đối với các từ khác trong từ điển.

Với cách biểu diễn như vậy, người ta khám phá ra rằng các vector mang lại cho ta cả cú pháp và ngữ nghĩa ở một mức độ nào đó để máy tính hiểu.

Word2vec thuộc dạng biểu diễn phân tán (distributed representation). Chúng ta có thể hiểu đơn giản là mã hoá từng từ bằng một vector. Ý tưởng chính của word2vec đó là "Dự đoán giữa tất cả các **từ**với các từ thuộc **ngữ cảnh** của chúng" (Predict between every **word**and its **context words!**).

*Để xây dựng nên word2vec sẽ có:*

*2 loại thuật toán:*

**Skip-gram :** Dự đoán từ thuộc ngữ cảnh với từng từ mục tiêu cho trước.

**Continuous Bag of Words (CBOW)**: Dự đoán từ mục tiêu từ bag-of-word (nhóm các từ) thuộc ngữ cảnh.

*2 phương pháp huấn luyện mô hình (training):*

Hierarchical softmax

Negative sampling

### 1.4. Môi trường làm việc jupyter notebook

- Jupyter là gì?

Là nền tảng tính toán khoa học mã nguồn mở, [interactive](https://en.wikipedia.org/wiki/IPython), hỗ trợ **hơn 40 ngôn ngữ lập trình**, trong đó có **python** (jupyter = **ju**lia + **pyt**hon + **R**).

* Jupyter notebook

Trước đây là **ipython notebook**, đổi tên thành **jupyter notebook** với mục tiêu hỗ trợ nhiều ngôn ngữ hơn. Là ứng dụng chạy trên nền web cho phép chạy interactive python (tương tự ipython). Hơn thế nữa, nó còn hỗ trợ vẽ các đồ thị, biểu đồ, hỗ trợ viết 1 "notebook" bằng cách sử dụng [Markdown](https://pymi.vn/blog/blog-markdown-pelican-githubpages)

Quá trình cài đặt và cách thức hoạt động của Jupyter notebook có thể tham khảo tại:

<https://medium.com/pymi/l%E1%BA%ADp-tr%C3%ACnh-python-tr%C3%AAn-windows-v%E1%BB%9Bi-winpython-75b6d6c42d1>

Sau khi cài đặt và khởi động một số thư viện cần thiết ta tiến hành công việc.

# CHƯƠNG II: KHẢO SÁT VÀ ĐẶT VẤN ĐỀ BÀI TOÁN TỰ ĐỘNG TÓM TẮT VĂN BẢN

## 2.1. Giới thiệu về thuật toán tự động tóm tắt văn bản

### 2.1.1. Thông tin thuật toán tự động tóm tắt văn bản:

Như mục tiêu đã nói rõ trong phần mở đầu, chúng ta sẽ tìm hiểu những phương pháp đơn giản nhất để có thể xây dựng được một chương trình tóm tắt văn bản. Và chương trình với **Ý tưởng chính ở đây là chúng ta sẽ loại bỏ đi những câu có ý nghĩa tương tự nhau để tạo ra 1 văn bản tóm tắt.**

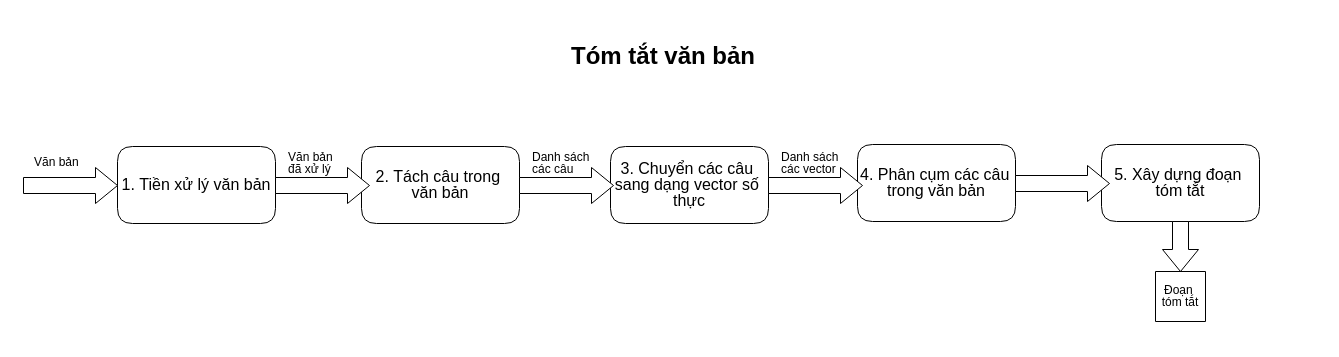
Việc tóm tắt tự động các văn bản hỗ trợ cho:

+ Cơ quan, tổ chức, trường học, các trang thông tin, tìm kiếm v.v…

+ Chuyên tóm tắt các văn bản, bài báo, email,.

+ Chương trình thực hiện tóm tắt thu nhỏ sao cho phù hợp và dễ dàng sử dụng.

### 2.1.2. Các bước sử lý của chương trình:



*Hình 2.1. Sơ đồ hoạt động chương trình*

Chương trình được mô tả qua các bước như sau:

1. ***Tiền xử lý văn bản*:** Văn bản đầu vào của chúng ta có thể chứa nhiều ký tự thừa, dấu câu thừa, khoảng trắng thừa, các từ viết tắt, viết hoa, ... điều này có thể làm ảnh hưởng tới các bước ở sau này nên chúng ta cần phải xử lý nó trước! Tuy nhiên trong bài lần này, chúng ta sẽ chỉ thử trên một số bài báo đã khá "quy cũ" rồi nên mình sẽ chỉ thực hiện 2 phương pháp đó là *Biến đổi hết về các chữ cái thường và Loại bỏ các khoảng trắng thừa*.
2. ***Tách câu trong văn bản*:** Ở bước này, chúng ta sẽ tách 1 đoạn văn bản cần tóm tắt đã qua xử lý thành 1 danh sách các câu trong nó.
3. ***Chuyển các câu sang dạng vector số thực*:** Để phục vụ cho phương pháp tóm tắt ở bước tiếp theo, chúng ta cần chuyển các câu văn (độ dài ngắn khác nhau) thành các vector số thực có độ dài cố định, sao cho vẫn phải đảm bảo được "độ khác nhau" về ý nghĩa giữa 2 câu cũng tương tự như độ sai khác giữa 2 vector tạo ra. Điều này mình sẽ giới thiệu một phương pháp mình cho là khá đơn giản cũng như giải thích kỹ hơn cho các bạn ở phần sau khi chúng ta đi vào code.
4. ***Phân cụm*:** Với các bạn nghiên cứu về Machine Learning thì đây chắc hẳn là một thuật toán rất quen thuộc (K-Means Clustering). Thuật toán này sẽ giúp chúng ta phân ra những cụm câu có ý nghĩa giống nhau, để từ đó chọn lọc và loại bỏ bớt các câu có cùng ý nghĩa.
5. ***Xây dựng đoạn văn bản tóm tắt*:** Sau khi đã có các cụm, trong mỗi cụm (phân loại theo ý nghĩa), chúng ta sẽ chọn ra 1 câu duy nhất trong cụm đó để tạo nên văn bản được tóm tắt!

# CHƯƠNG III: PHÂN TÍCH THIẾT KẾ CHƯƠNG TRÌNH TÓM TẮT VĂN BẢN

## 3.1. Phân tích thuật toán

## 3.1.1. Tiền xử lý văn bản

Vì chúng ta sẽ chỉ áp dụng trên các bài báo tin tức cũng như các văn bản văn xuôi nên chỉ dùng hai phương pháp là *chuyển đổi hết sang chữ cái thường* và *loại bỏ các khoảng trắng*, chúng ta sẽ có đoạn code ví dụ sau:

contents\_parsed = content.lower() #Biến đổi hết thành chữ thường

contents\_parsed = contents\_parsed.replace('\n', '. ') #Đổi các ký tự xuống dòng thành chấm câu

contents\_parsed = contents\_parsed.strip() #Loại bỏ đi các khoảng trắng thừa

Ví dụ: cho một văn bản:

“Hacker tuyên bố nắm thông tin hơn 5 triệu khách hàng Thế Giới Di Động. Có cả số thẻ và địa chỉ email. Những thông tin bị rò rỉ bao gồm địa chỉ email, lịch sử giao dịch và thậm chí là cả số thẻ.”

- Sau khi xử lý:

“hacker tuyên bố nắm thông tin hơn 5 triệu khách hàng thế giới di động. có cả số thẻ và địa chỉ email. những thông tin bị rò rỉ bao gồm địa chỉ email, lịch sử giao dịch và thậm chí là cả số thẻ”

Các chữ cái in hoa đã được chuyển về dạng thường, từ đó việc phân cụm và chuyển hướng vector sẽ trở nên chính xác hơn.

## 3.1.2. Tách các câu trong văn bản

Tại đây chúng ta sẽ tách các câu trong văn bản trên ra để thu được 1 danh sách các câu có ở trên. Việc này trở nên đơn giản hơn rất nhiều với thư viện NLTK, có thể dễ dàng sử dụng hàm sent\_tokenize để lấy ra danh sách các câu.

import nltk

sentences = nltk.sent\_tokenize(contents\_parsed)

Kết quả đoạn văn được hiển thị dưới dạng:

[‘hacker tuyên bố nắm thông tin hơn 5 triệu khách hàng thế giới di động.’,’ có cả số thẻ và địa chỉ email.’,’ Những thông tin bị rò rỉ bao gồm địa chỉ email, lịch sử giao dịch và thậm chí là cả số thẻ.’]

## 3.1.3. Chuyển các câu sang vector

Về phần này, hiện nay có rất nhiều phương pháp training được sử dụng hiệu quả, nhằm biến đổi các câu sang các vector có độ dài cố định mà vẫn giữ được các đặc trưng, ý nghĩa của câu đó. Có thể kể đến phương pháp mà gần đây mình đang nghiên cứu và có vài lần làm thử là Skip Thought. Tuy nhiên mô hình này được training trên tập dữ liệu tiếng Anh và để training lại với dữ liệu tiếng Việt sẽ mất thời gian. Mô hình được sử dụng là một phương pháp dễ dàng để tiếp cận hơn đó là sử dụng sẵn mô hình đã được huấn luyện chuyển đổi từ "**Từ sang vector**" (Word2Vec).   
  
 Phương pháp này chính là sẽ tách nhỏ từng câu thành các từ, sau đó dùng mô hình Word2Vec đã được training cho tiếng Việt, chuyển đổi các từ đó sang các vector số thực có chiều dài cố định. Cuối cùng, vector của 1 câu chuyển đổi sang sẽ là **TỔNG** của các vector đại diện cho các từ trong câu!   
  
 Mô hình Word2Vec cho tiếng Việt có thể dễ dàng tìm trên mạng, ở đây ta sử dụng mô hình tại

<https://github.com/Kyubyong/wordvectors>

Với mô hình này, các từ sẽ được biến đổi thành một vector 100 chiều. Chúng ta sẽ sử dụng thư viện gensim để load lại model.

from gensim.models import KeyedVectors

w2v = KeyedVectors.load\_word2vec\_format("vi\_txt/vi.vec")

Tiếp theo là tách các từ trong câu và lấy tổng để được các vector cho từng câu trong danh sách mà chúng ta vừa có trên kia:

vocab = w2v.wv.vocab #Danh sách các từ trong từ điển

from pyvi import ViTokenizer

X = []

for sentence in sentences:

sentence\_tokenized = ViTokenizer.tokenize(sentence)

words = sentence\_tokenized.split(" ")

sentence\_vec = np.zeros((100))

for word in words:

if word in vocab:

sentence\_vec+=w2v.wv[word]

X.append(sentence\_vec)

Trong đoạn code ở trên, ta sẽ duyệt qua từng câu trong danh sách câu của chúng ta. Với mỗi câu, ta sẽ tách các từ ra. Ở đây ta dùng thêm 1 thư viện pyvi để tách các từ tiếng Việt. Ví dụ như câu:

"công sở đậm dư âm tết trong ngày đầu làm việc"

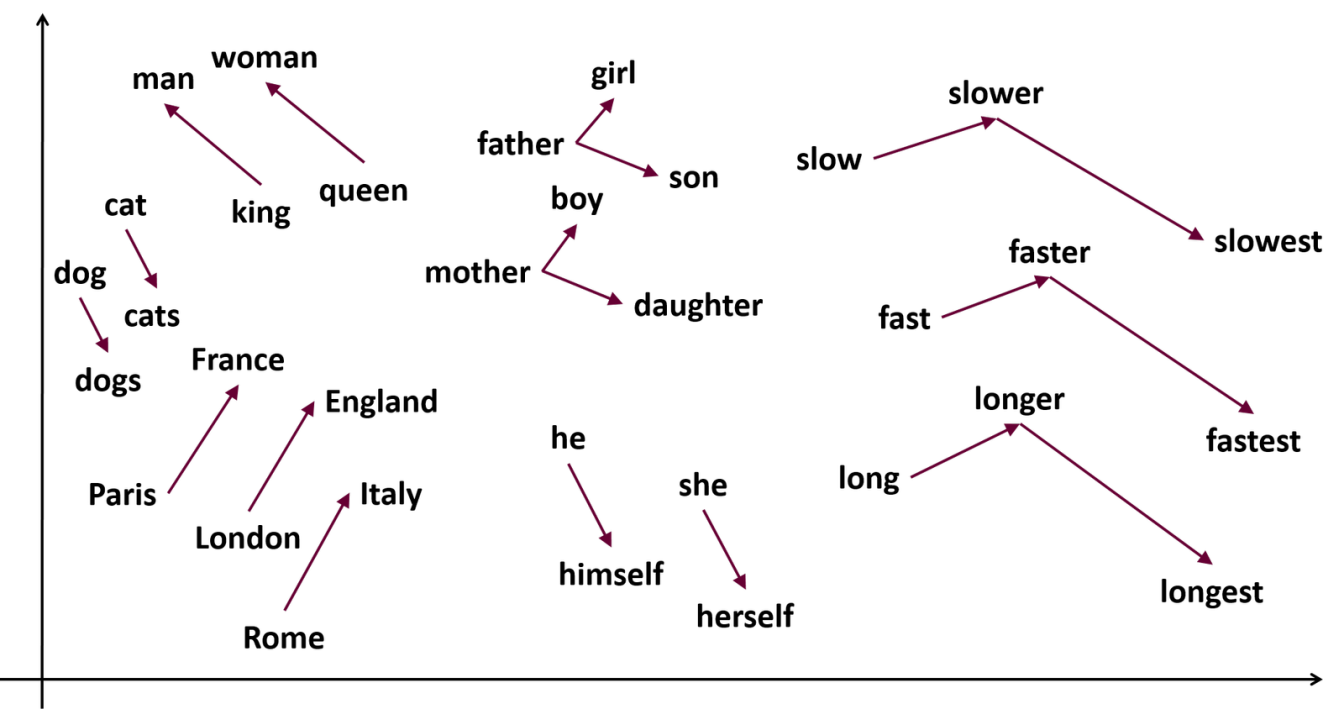
Chúng ta phải tách thành:

['công\_sở', 'đậm', 'dư\_âm', 'tết', 'trong', 'ngày', 'đầu', 'làm\_việc', '.']

Thư viện pyVi với hàm VnTokenizer sẽ giúp chúng ta ghép các từ có nghĩa trong tiếng Việt lại với nhau nhằm đảm bảo giữ nguyên ý nghĩa của từng từ!   
  
 Sau đó, ta khai báo một vector 100 chiều gồm toàn số 0. rồi với mỗi một từ trong câu, ta đều sử dụng hàm word2vec chuyển thành vector rồi cộng nó vào vector này (nếu nó có thể chuyển được thành vector). Cuối cùng sau khi hết mỗi câu, ta thêm nó vào 1 mảng đặt tên là X.

**VẬY VẤN ĐỀ Ở ĐÂY LÀ TẠI SAO CHUYỂN TỪ TỪ SANG VECTOR LẠI VẪN GIỮ ĐƯỢC Ý NGHĨA CỦA TỪNG TỪ?**

Điều tuyệt vời nhất nằm ở mô hình Word2Vec chúng ta đang sử dụng là mô hình này được training trên nhiều bài viết bằng tiếng Việt với mô hình khá phức tạp trong khoảng thời gian tương đối lâu để giúp tạo ra 1 vector đại diện cho mỗi từ mà đảm bảo nguyên được sự "sai khác" về ý nghĩa của chúng so với các từ khác, các vector khác.



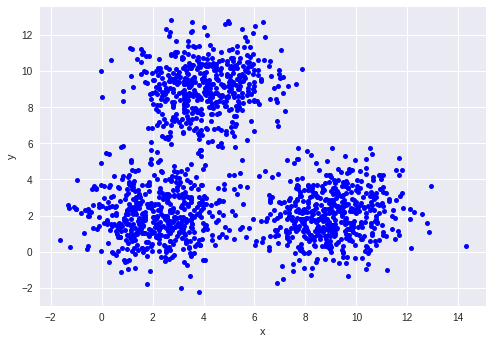
*Hình 3.1. minh họa chuyển nghĩa trong vector*

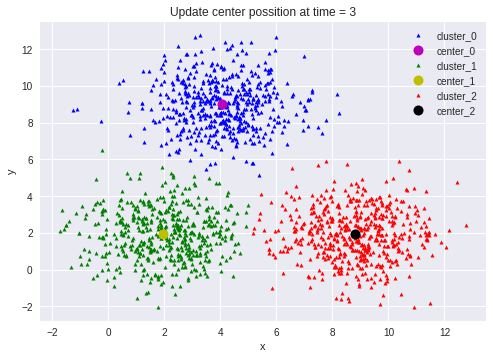
Sau khi training, mô hình này giúp chúng ta biễu diễn các từ thành các vector. Và nó giữ được ý nghĩa của các từ như chúng ta nhìn thấy trong hình trên.

Ví dụ về mặt toán học: Nếu chúng ta lấy man - woman ta sẽ thu được một vector cùng hướng, cùng độ dài với vector ta tính được khi lấy king – queen.  
  
 X chúng ta thu được sẽ là các vector 100 chiều mà mỗi vector đại diện cho 1 câu trong văn bản vừa rồi của chúng ta.

## 3.1.4. Phân cụm

Bài toán phân cụm là 1 nhánh ứng dụng chính của lĩnh vực Unsupervised Learning (Học không giám sát), trong đó dữ liệu được mô tả trong bài toán không được dán nhãn (tức là không có đầu ra). thuật toán sẽ tìm cách phân cụm - chia dữ liệu thành từng nhóm có đặc điểm tương tự nhau, nhưng đồng thời đặc tính giữa các nhóm đó lại phải càng khác biệt càng tốt.   
  
 Và ở đây, chúng ta muốn phân cụm các vector đại diện cho từng câu trong văn bản vừa rồi để biết những câu nào mang ý nghĩa giống nhau.





Thuật toán phân cụm ở đây, mình chọn số cụm (clusters) chính bằng số câu mà chúng ta muốn tóm tắt. Thuật toán này các bạn có thể dễ dàng sử dụng và cài đặt với sklearn

from sklearn.cluster import KMeans

n\_clusters = 5

kmeans = KMeans(n\_clusters=n\_clusters)

kmeans = kmeans.fit(X)

Ở đây ta muốn tóm tắt mẩu tin trên thành 5 câu duy nhất, vậy nên n\_clusters sẽ để bằng 5. Quá trình phân cụm ở đây cũng sẽ diễn ra rất nhanh do số lượng câu trong văn bản rất nhỏ Và chúng ta đã có được 5 cụm "ý nghĩa" của văn bản trên, sẵn sàng để xây dựng ra một văn bản tóm tắt ở bước cuối cùng.

## 3.1.5. Xây dựng đoạn văn bản tóm tắt

Mỗi cụm mà chúng ta phân ra được ở trên, có thể hiểu là đại diện cho một ý nghĩa nào đó ở trong văn bản gốc. Tuy nhiên để tóm gọn lại cả 1 văn bản dài, với mỗi một ý nghĩa chúng ta sẽ chỉ chọn ra 1 câu duy nhất thôi! Và câu được chọn để ĐẠI DIỆN cho 1 cụm đó ở đây, sẽ là câu có khoảng cách gần với trung tâm của cụm nhất. Sau khi đã có được các câu của văn bản tóm tắt, giờ chúng ta quan tâm là sẽ sắp xếp thứ tự như thế nào cho hợp lý. Ở đây, với mỗi ý nghĩa, ta sẽ tính "thứ tự xuất hiện trung bình" của cụm đó.

- Ví dụ cụm 1 có các câu 1, 2, 5. thứ tự trung bình sẽ là 8/3.

Làm tương tự với các cụm khác, sau đó chúng ta sẽ lấy các câu đại diện trong các cụm theo thứ tự từ nhỏ đến lớn của thứ tự xuất hiện trung bình để tạo ra một văn bản tóm tắt.

from sklearn.metrics import pairwise\_distances\_argmin\_min

avg = []

for j in range(n\_clusters):

idx = np.where(kmeans.labels\_ == j)[0]

avg.append(np.mean(idx))

closest, \_ = pairwise\_distances\_argmin\_min(kmeans.cluster\_centers\_, X)

ordering = sorted(range(n\_clusters), key=lambda k: avg[k])

summary = ' '.join([sentences[closest[idx]] for idx in ordering])

Trong đoạn code trên, ta sẽ sử dụng hàm pairwise\_distances\_argmin\_min của sklearning metric để lấy ra khoảng cách vector giữa các vector và các trung tâm cụm của nó, sau đó chọn ra khoảng cách nhỏ nhất để lấy được câu đại diện. Sau đó tính "thứ tự xuất hiện trung bình" và sắp xếp lại để tạo ra văn bản tóm tắt.

# CHƯƠNG IV: CÀI ĐẶT CHƯƠNG TRÌNH

## 4.1. xây dựng chương trình tóm tắt văn bản

### 4.1.1. chuyển bị dữ liệu và cài đặt các gói thư viện

Trước tiên ta cần cài đặt môi trường, ở đây là jupyter notebook.

Các bước cài đặt và sử dụng có thể tham khảo tài liệu:

- cài đặt các thư viện cần thiết:

pip install pyvi

pip install pickle

pip install nltk

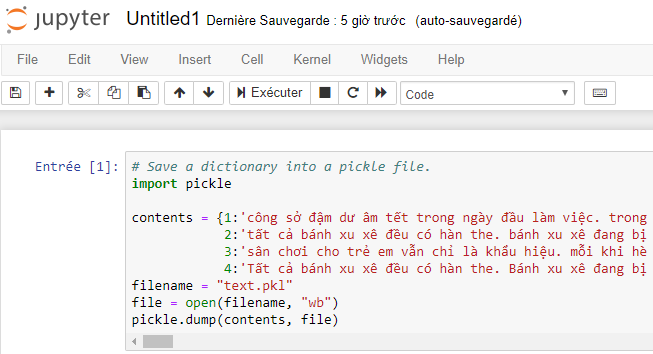
- Mô hình Word2Vec cho tiếng Việt: <https://github.com/Kyubyong/wordvectors>



Sau khi tải về ta có được data:



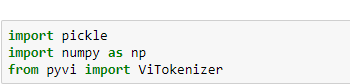
**a.tạo dữ liệu đầu vào**



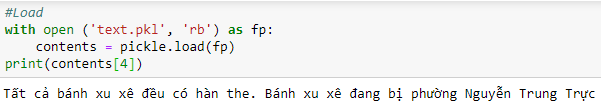
Ta thu được 1 data lưu dưới dạng .pkl



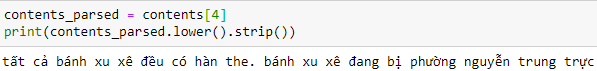
**b.gọi thư viện và biến môi trường**



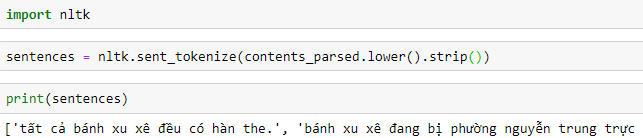
**c.sử dụng data tạo ra trước đó**



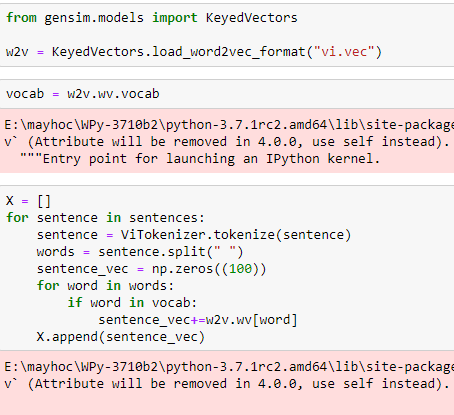
**d.chuyển đổi văn bản về chữ thường và loại bỏ các dấu cách thừa**



**e.tách các câu trong văn bản**

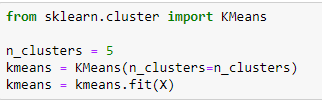


**f.chuyển sang vector**

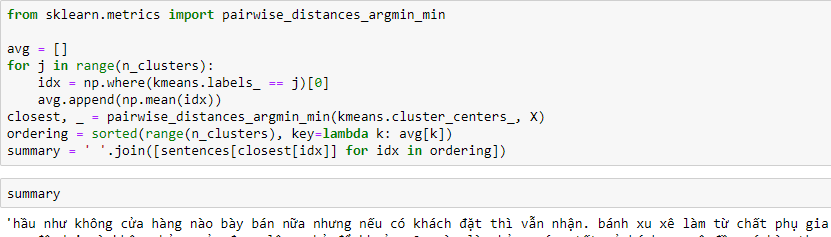


Trong quá trình chuyển sẽ tùy vào phiên bản sử dụng sẽ có các cảnh báo, không cần lo lắng về các vấn đề này.

**g.phân cụm các từ trong văn bản thành các nhóm có nghĩa tương đồng**



**h.xây dựng đoạn văn tóm tắt và hiển thị chúng**



# KẾT LUẬN

1. Kết quả đạt được

- Hoàn thành chương trình với các chức năng chạy ổn định.

- Hoàn thành test dữ liệu.

- sửa một số lỗi có trong chương trình.

2. Hạn chế

- Chương trình cần chỉnh sửa, bổ xung thêm về chức năng .

- Văn bản tóm tắt chưa thực sự sát nghĩa, dữ liệu sử dụng có sẵn nên khá kém trong việc thể hiện mong muốn của người dùng.

- Phương pháp phân cụm đơn giảm, có nhiều từ còn sai nghĩa.

3. Hướng phát triển

- Để cải tiến hơn chương trình này, có thể tìm hiểu sử dụng một số mô hình sao cho biến đổi được trực tiếp từ câu sang vector (chứ không phải lấy tổng các từ).

- Tìm hiểu Về kỹ thuật tóm tắt khác còn có rất nhiều như tóm tắt trừu tượng (Câu được tự sinh ra chứ không phải là 1 trong số câu trong văn bản), tóm tắt động (Generic) ,..

- Bổ xung các tài liệu liên quan.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Duy Phạm, " Word2vec: Biểu diễn từ ngữ thông qua vector". Available: https://medium.com/@phamduychv94/word2vec-bi%E1%BB%83u-di%E1%BB%85n-t%E1%BB%AB-ng%E1%BB%AF-th%C3%B4ng-qua-vector-4dcbd42dd14. |
| [2] | Phạm Hoàng Anh, " Xây dựng chương trình tóm tắt văn bản (tiếng Việt) đơn giản với Machine Learning,". Available: https://viblo.asia/p/xay-dung-chuong-trinh-tom-tat-van-ban-tieng-viet-don-gian-voi-machine-learning-YWOZrgAwlQ0#comment-b85oWqbnB52. |
| [3] | Python, " Using Pickle". Available: https://wiki.python.org/moin/UsingPickle. |
| [4] | Tôi là Duyệt, " NLP - Truyện Kiều Word2vec". Available: https://blog.duyet.net/2017/04/nlp-truyen-kieu-word2vec.html#.W-K1k5MzbIU. |
| [5] | Vũ Hữu Tiệp, " K-means Clustering: Simple Applications". Available: https://machinelearningcoban.com/2017/01/04/kmeans2/. |