◀ home (../)

Giới thiệu tiền xử lý trong xử lý ngôn ngữ tự nhiên

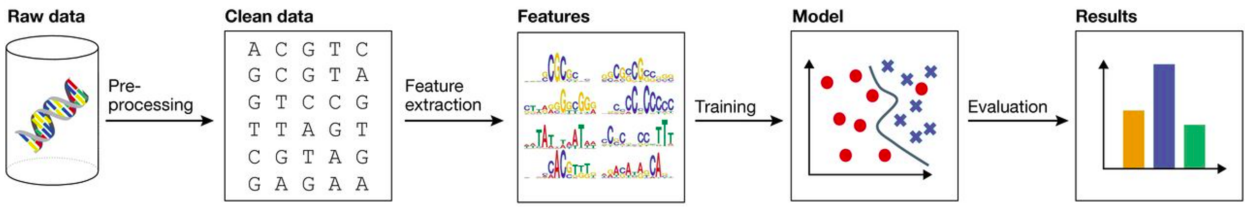
Mở đầu

Chào các bạn, chắc hẳn ít nhiều các bạn đã từng nghe đến công việc xử lý ngôn ngữ tự nhiên. Nói một cách ngắn gọn như sau:

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**.

Bước tiền xử lý nhìn chung sẽ được mô tả tóm tắt như sau:

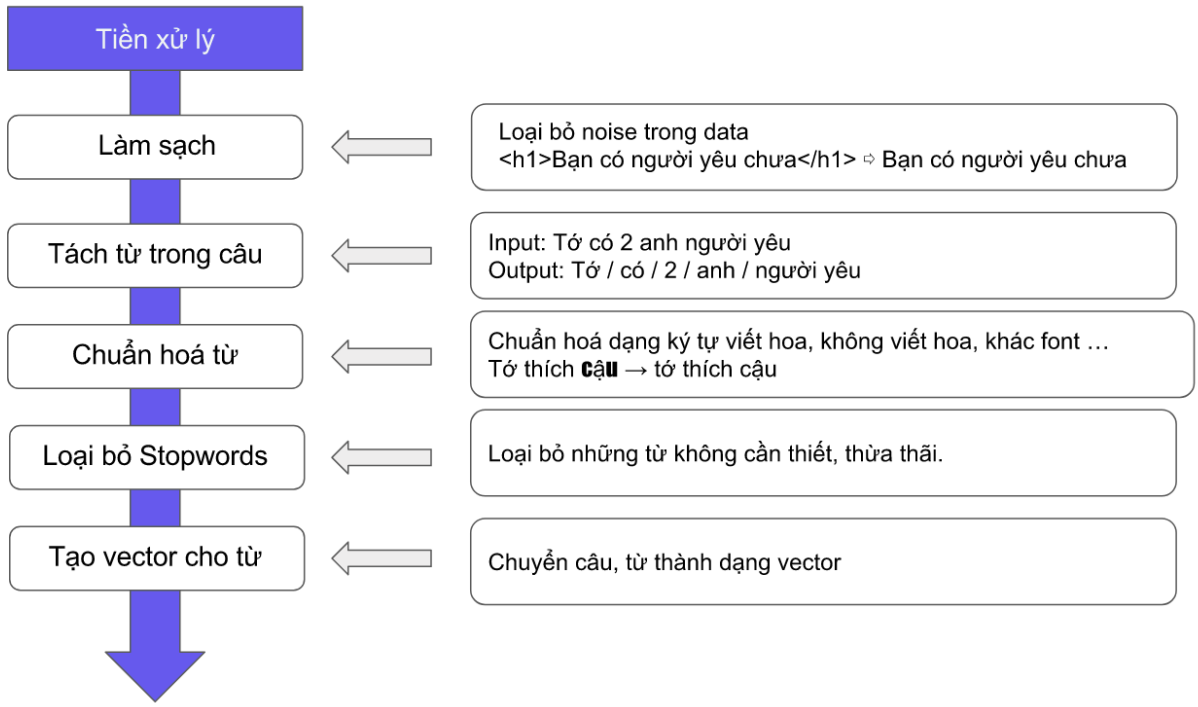


Trích từ: Deep learning for computational biology (http://msb.embopress.org/content/12/7/878)

Và bây giờ chúng ta cùng đi đến chi tiết xem bước **tiền xử lý** cần giải quyết những vấn đề gì ?

Các loại tiền xử lý

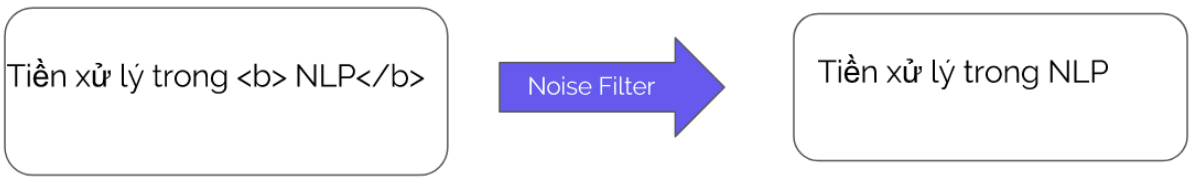
Lần này mình sẽ lần lượt giới thiệu với các bạn 5 bước tiền xử lý như dưới đây.



Làm sạch text

Mục đích bước này là loại bỏ noise trong data của bạn. 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.

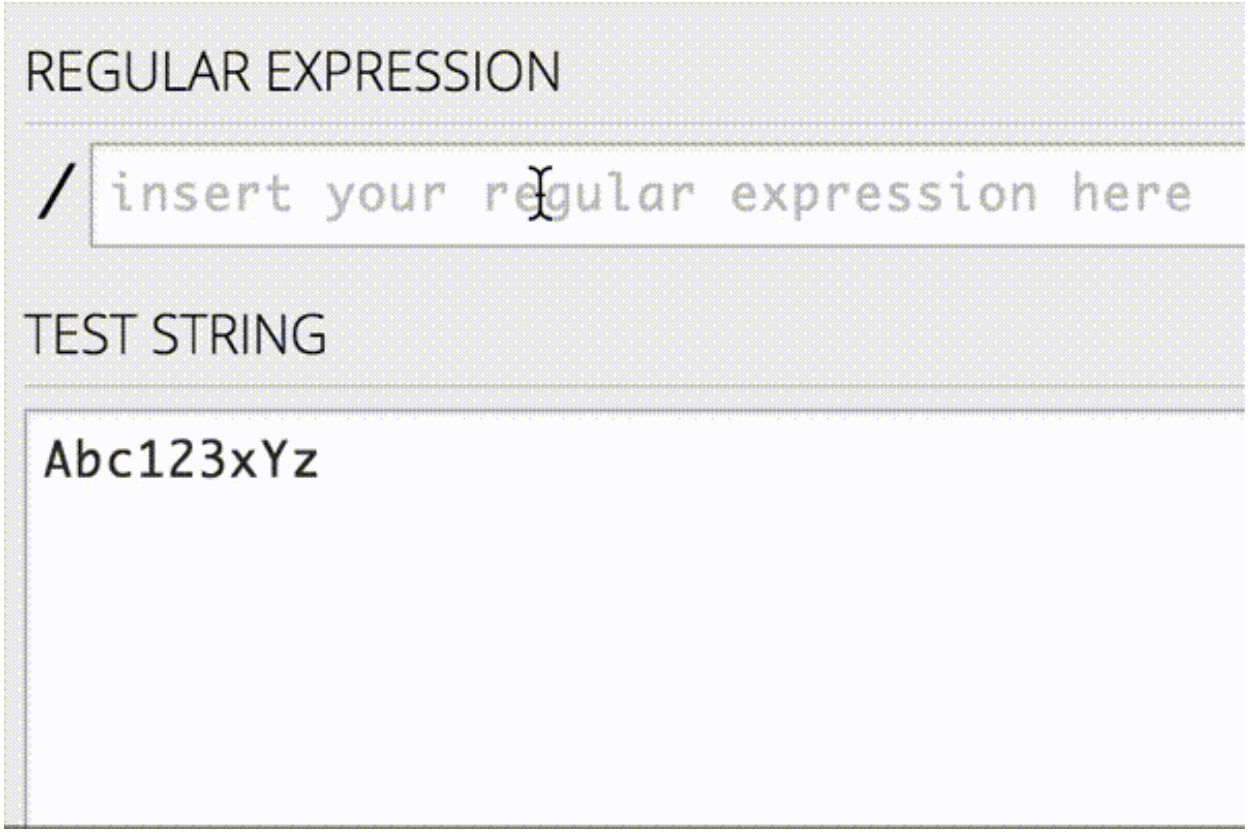
Ví dụ đơn giản như sau:



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 các trường hợp thông thường, cách đơn giản và dễ nhất là sử dụng filter theo regx, mình thường sử dụng trang web sau để viết regx:

https://regex101.com/ (https://regex101.com/)



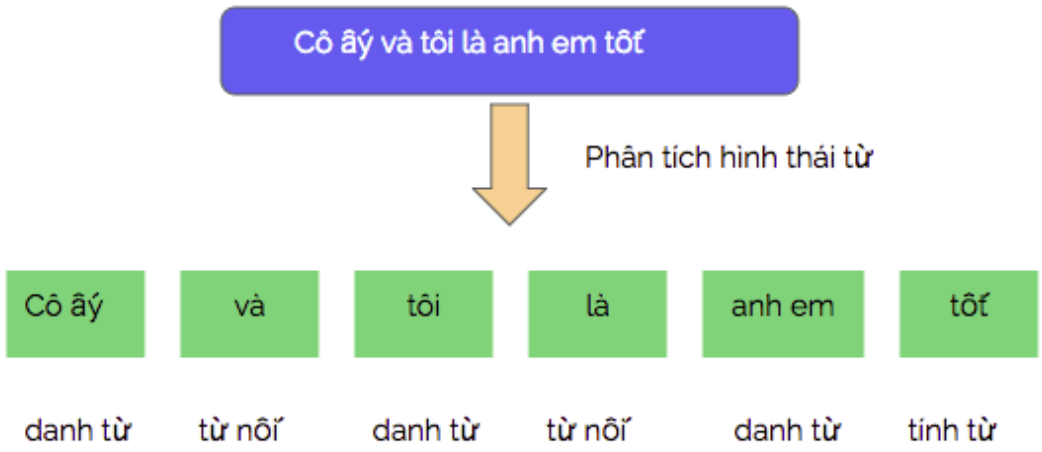
Với code Python, BeautifulSoup (https://www.crummy.com/software/BeautifulSoup/bs4/doc/) và lxml (http://lxml.de/) là 2 thư viện được cộng đồng sử dụng nhiều nhất và vô cùng mạnh mẽ, tiện lợi.

Tách từ

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:



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.

Về phương pháp, hiện nay cũng có khá nhiều mã nguồn nghiên cứu được public, bạn có thể tham khảo tại word-segmentation (https://github.com/magizbox/underthesea/wiki/Vietnamese-NLP-Tools#word-segmentation).

Chuẩn hoá 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.

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... Tham khảo thêm tại danh sách stopwords trong tiếng việt (https://github.com/stopwords/vietnamese-stopwords/blob/master/vietnamese-stopwords.txt)

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

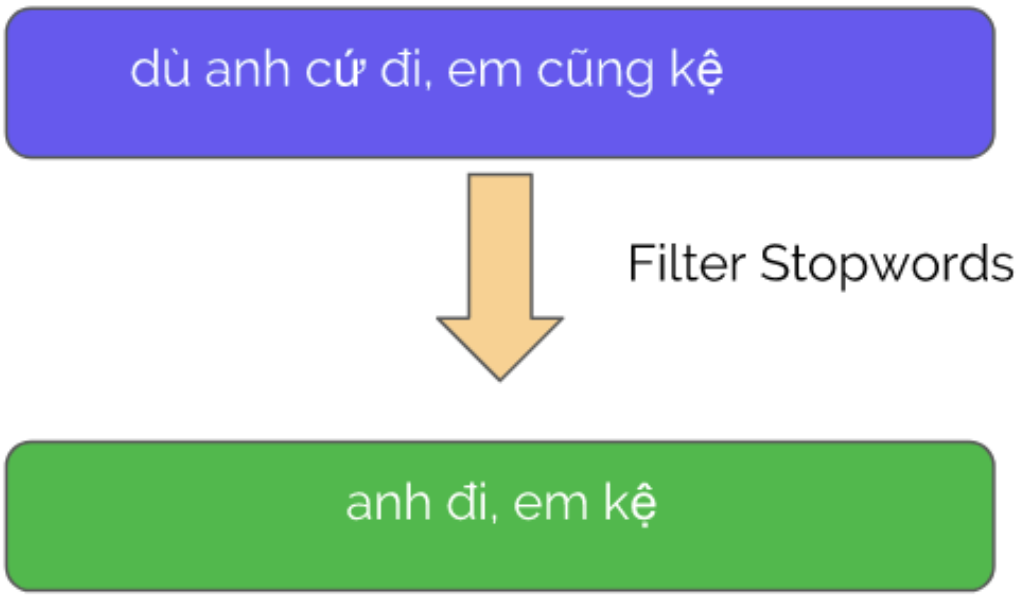
- Dùng từ điển
- Dựa theo tần suất xuất hiện của từ

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:

```
cậu
của
cứ
dù
nọ
phóc
này
kia
để
...
```

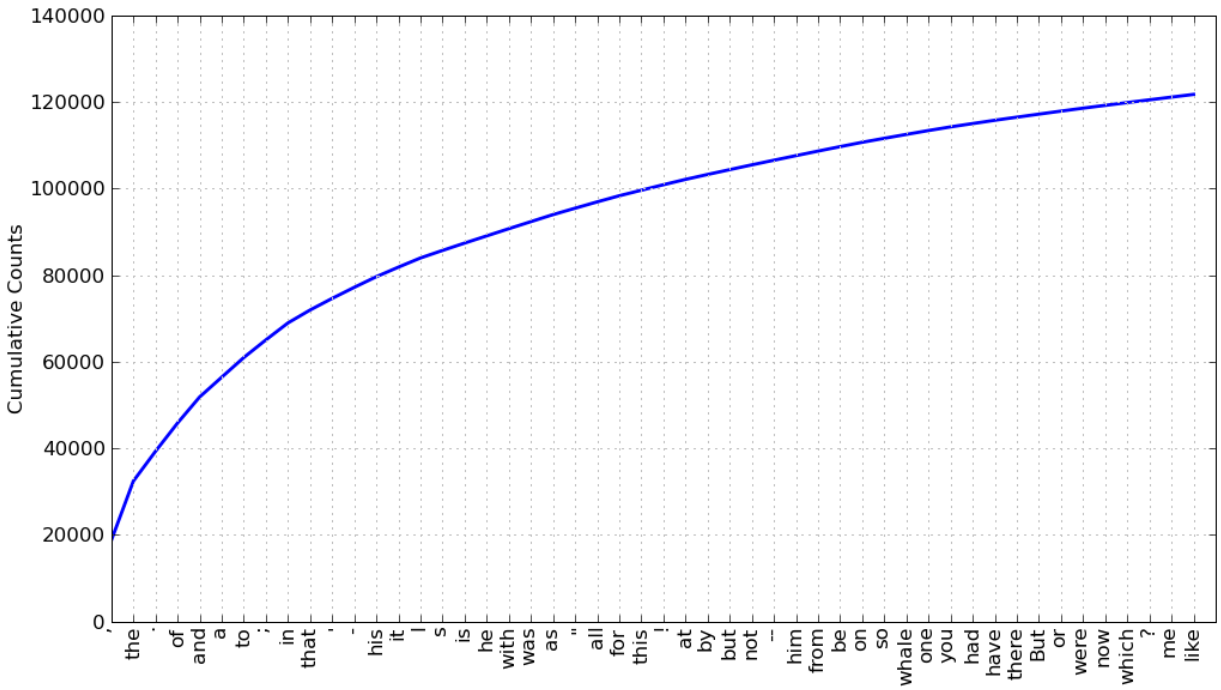
Ví dụ:



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.

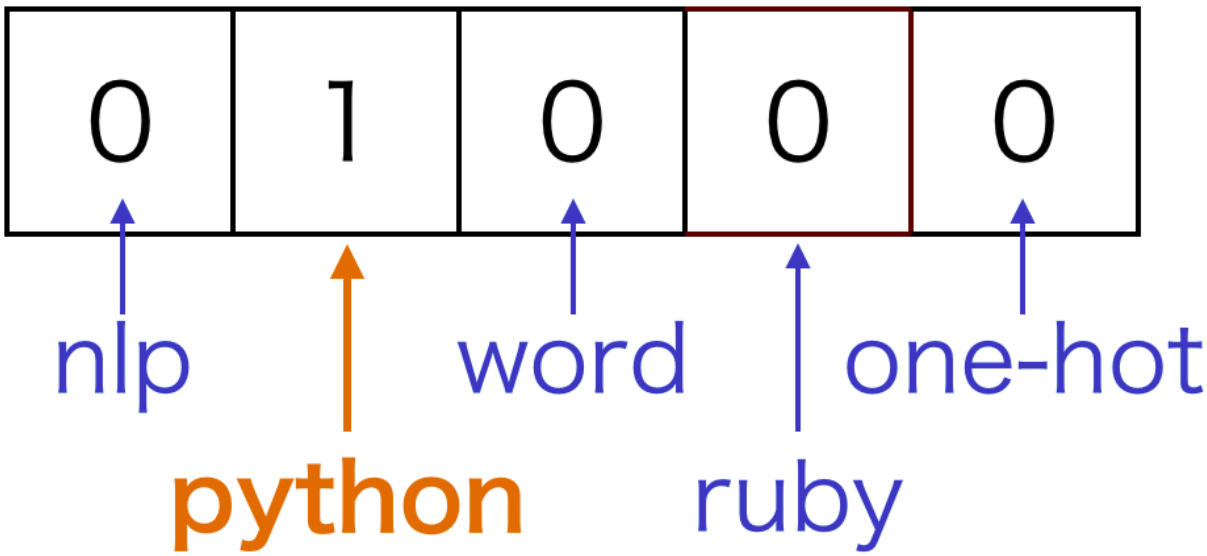
Vector hoá từ

Bước này mục đích là vector hoá từ trong từng câu. Thông thường chúng ta nên vector hoá theo từng câu chứ không vector hoá cả đoạn. Vì nếu vector hoá theo đoạn văn dài sẽ dẫn đến vector của bạn nhiều chiều quá, nếu dữ liệu của bạn nhiều chiều, sẽ dẫn đến tình trạng thiếu tính chính xác, khó xử lý. Phương pháp vector hoá có 2 cách:

- Sử dụng one-hot
- Biểu thị sự phân tán

Sử dụng one-hot

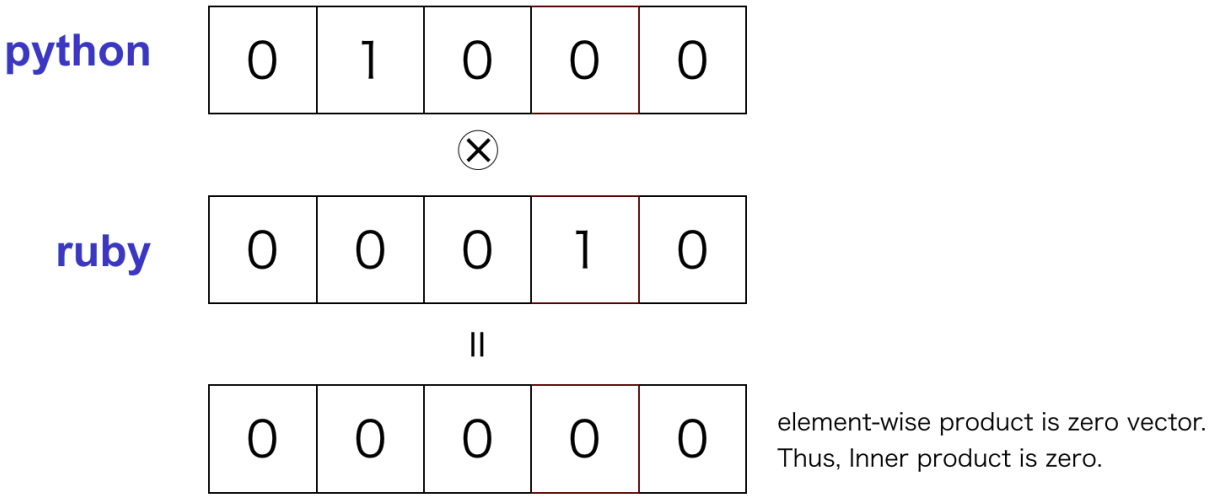
Phương pháp này là phổ biến nhất, cũng rất dễ sử dụng. Giả sử ta có danh sách các từ sau: (nlp, python, word, ruby, one-hot) khi vector hoá từ python, ta sẽ được:



Tương tự với nlp thì vị trí nlp sẽ là 1, còn lại là 0, cứ như vậy ta sẽ biểu diễn được tất cả các từ. Lấy luôn hình trên làm ví dụ ta sẽ được:

- nlp: 10000
- python: 01000
- word: 00100
- ruby: 00010
- one-hot: 00001

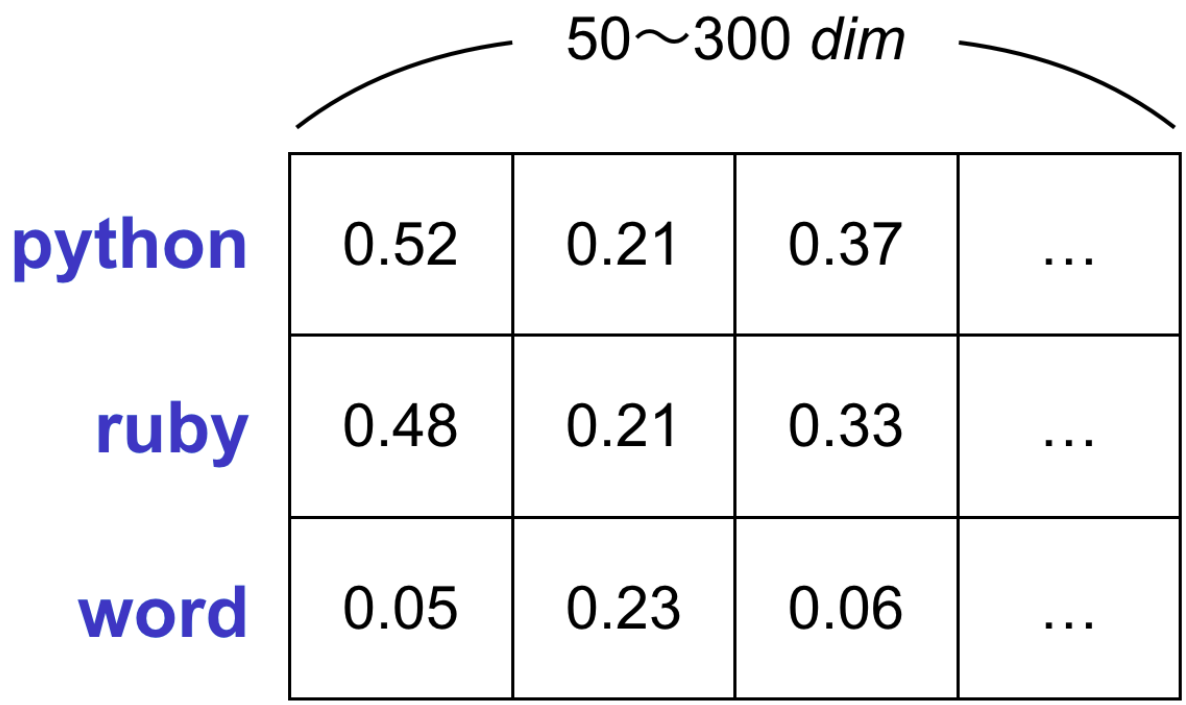
Biểu diễn bằng one-hot rất đơn giản nhưng có một điểm yếu là không có mối liên hệ giữa các từ.



Ví dụ như bạn muốn tính độ tương tự giữa các từ, việc cần làm là tính tích vô hướng 2 từ đó, tuy nhiên với cách biểu diễn bằng one-hot, tích vô hướng luôn bằng 0 nên không có ý nghĩa gì cả.

Biểu thị sự phân tán

Phân tán ở đây nghĩa là tần suất phân bố, xuất hiện của từ đó trên mỗi chủ đề(topic), hoặc là mỗi đoạn văn bản khác nhau. Thực tế thông thường số chiều khoảng 50-500. Như ví dụ sau:



Như trên ví dụ số chủ để khoảng 50-300 chủ đề, khi biểu diễn theo kiểu này, dễ dàng nhận thấy vấn đề của one-hot đã được giải quyết, khi tính tích vô hướng sẽ nhận thấy ngay python và ruby có mối liên quan rất lớn, ngược lại so với word không có nhiều mối tương đồng.

Việc phân loại từ như trên rất quan trọng trong NLP, ví dụ như bài toán Text Classification, bạn cần phân biệt, tìm ra các từ cùng chủ đề. Hay bài toán Intent detection, bạn cũng cần tìm ra mối quan hệ giữa input và training data.

Kết luận

Trên đây mình đã giới thiệu với các bạn cái nhìn tổng quan về những gì mình cần làm ở bước tiền xử lý dữ liệu trong NLP. Bước này là bước rất quan trọng để nâng cao hiệu suất cũng như tính chính xác của model sau này. NLP cũng là một bài toán khó trong lĩnh vực AI, chúng ta từng bước giải từng bài toán một, dần dần bạn sẽ làm được điều mình muốn, mong rằng bài viết này sẽ giúp ích cho các bạn mới tìm hiểu về NLP.

Hẹn gặp lại các bạn với các bài viết về NLP tiếp theo.

Tài liệu tham khảo

http://ahogrammer.com/2017/03/22/why-is-word-embeddings-important-for-natural-language-processing/ (http://ahogrammer.com/2017/03/22/why-is-word-embeddings-important-for-natural-language-processing/)

https://github.com/magizbox/underthesea/wiki/Vietnamese-NLP-Tools#text-classification (https://github.com/magizbox/underthesea/wiki/Vietnamese-NLP-Tools#text-classification)

https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

https://datascience.stackexchange.com/questions/11402/preprocessing-text-before-use-rnn (https://datascience.stackexchange.com/questions/11402/preprocessing-text-before-use-rnn)

http://pages.cs.wisc.edu/~jerryzhu/cs769/text_preprocessing.pdf (http://pages.cs.wisc.edu/~jerryzhu/cs769/text_preprocessing.pdf)

ALSO ON **CODETUDAU**

Bag of Words (Bow) TF-IDF - Xử lý ngôn ngữ tự nhiên

1 comment • 7 months ago

Phuc Coi — hay

Đi tìm cội nguồn của Deep Learning - Perceptron | Code từ đầu - machine learning

1 comment • 3 months ago

Gou Tain — Có thanh niên nào giải thích được vì sao 1 perceptron không biểu diễn được XOR ko.Tất nhiên là ngoài câu "Không tin bạn

Phân tích cây quyết định với scikit-learn | Code từ đâu - Code từ đầu

1 comment • a year ago

Hằng Phan Thúy — đúng thứ đang tìm, cảm ơn anh.

Tìm hiểu phương pháp k-means | Code từ đâu - Code từ đầu

4 comments • 9 months ago

Cao Thinh — Cảm ơn bạn vì bài chia sẻ rất hữu ích. Cho mình hỏi một số parameters của Kmeans1) random state =0 nghĩa là gì?2)