Building a vector space model for Vietnamese

Lê Việt Hùng
22022666@vnu.edu.vn
Link GitHub: https://github.com/HungLV2512/NLP

Abstract

Thử nghiệm này tập trung vào việc Xây dựng mô hình không gian vector cho Tiếng Việt sử dụng tập dữ liệu VietNam Wikipedia và thư viện gensim

1 Introduction

Trong xử lý ngôn ngữ tự nhiên, việc biểu diễn một từ thành một vector đóng một vai trò cực kỳ quan trọng. Nó lợi ích rất nhiều trong việc thể hiện sự tương đồng, đối lập về ngữ nghĩa giữa các từ, giúp mô hình hóa vector cho 1 câu hay đoạn văn, tìm các câu có nghĩa tương đồng,...

Word Embedding là một nhóm các kỹ thuật đặc biệt trong xử lý ngôn ngữ tự nhiên, có nhiệm vụ ánh xạ một từ hoặc một cụm từ trong bộ từ vựng tới một vector số thực. Từ không gian một chiều cho mỗi từ tới không gian các vector liên tục. Các vector từ được biểu diễn theo phương pháp word embedding thể hiện được ngữ nghĩa của các từ, từ đó ta có thể nhận ra được mối quan hệ giữa các từ với nhau(tương đồng, trái nghịch,...).

Trong các ứng dụng về xử lý ngôn ngữ tự nhiên, học máy,... các thuật toán không thể nhận được đầu vào là chữ với dạng biểu diễn thông thường. Để máy tính có thể hiểu được, ta cần chuyển các từ trong ngôn ngữ tự nhiên về dạng mà các thuật toán có thể hiểu được(dạng số)

2 Một vài phương pháp Word Embedding

2.1 One hot vector(1-of-N)

One hot vector[1] là kỹ thuật đơn giản nhất trong word embedding.

Để chuyển đổi ngôn ngữ tự nhiên về dạng 1-of-N, ta thực hiện các bước như sau:

- Xây dựng một bộ từ vựng.
- Mỗi vector đại diện cho một từ có số chiều bằng số từ trong bộ từ vựng. Trong đó, mỗi vector chỉ có một phần tử duy nhất khác 0(bằng 1) tại vị trí tương ứng với vị trí từ đó trong bộ từ vựng.

Ví dụ: Giả sử bộ từ vựng của chúng ta chỉ có 5 từ: Vua, Hoàng hậu, Phụ nữ, Đàn ông và Trẻ con. Ta sẽ mã hóa cho từ Hoàng Hậu dưới dạng vector như sau: **[0,1,0,0,0]**.

Tuy nhiên, phương pháp này lại để lộ ra những điểm hạn chế vô cùng lớn.

- Thứ nhất là độ dài của vector là quá lớn(vietwiki: Corpus Size(74M), Vocabulary size(10K))

Preprint. Under review.

- Đặc biệt phương pháp này không xác định được sự tương quan ý nghĩa giữa các từ do tích vô hướng của 2 từ bất kì đều bằng 0 dẫn đến độ tương đồng cosin giữa 2 từ bất kì luôn bằng 0.

Do đó, việc tìm một phương pháp biểu diễn từ mà vẫn thể hiện được một cách tốt nhất ngữ nghĩa của từ là một vấn đề cực kỳ quan trọng.

2.2 Sử dụng wordnet

wordnet là một cơ sở dữ liệu về từ, trong đó các từ được nhóm lại thành các loạt từ đồng nghĩa, các loạt từ đồng nghĩa này được gắn kết với nhau nhờ các quan hệ ngữ nghĩa.

```
from  Viwordnet import viwordnet
synset = viwordnet('V', 'cố_gắng')
for word in synset:
    print word
```

Kết quả thu được là:

```
dốc_sức, gắng_sức, nỗ_lực, gắng, gắng_mình, dốc_hết_mình, phấn_đấu, gắng_hết_sức_mình
ráng_sức, đối_phó, dốc_hết_nghị_lực, dồn_sức, cố_gắng
```

Tuy nhiên, wordnet vẫn ẩn chứa vấn đề của nó:

- Thiếu sắc thái, ví dụ như các từ đồng nghĩa: Cố, cố gắng, gắng, nỗ lực được xem là có mức độ như nhau.
- Thiếu từ mới hoặc ý nghĩa mới(không thể cập nhật): Sống thử, lầy, thả thính, trẻ trâu, gấu,...
- Chủ quan, phụ thuộc vào người tạo
- Yêu cầu nhiều công sức tạo ra và cập nhật để thích ứng
- Khó đo chính xác khoảng cách về nghĩa giữa các từ.

Chưa kể đến việc wordnet cho Tiếng Việt còn nhiều hạn chế về chất lượng và cấu trúc lưu trữ.
Đặc biệt sử dụng wordnet không thể lấy được rất nhiều giá trị đại diện cho một từ thông qua ý nghĩa của các từ lân cận nó.

2.3 Windows based cooccurence matrix

Xây dựng 1 ma trận đồng xuất hiện X dựa vào cửa sổ quanh mỗi từ. Việc này giúp ta nắm bắt được cả ngữ pháp của câu cũng như ngữ nghĩa của từ.

Ví dụ: Windows based cooccurence matrix với Window length 1(thường 5-10), đối xứng(không liên quan tới trái hoặc phải).

```
Corpus:
- tôi yêu công_việc .
- tôi thích NLP .
- tôi ghét ở một_mình
```

Ta có ma trận đồng xuất hiện sau:

	tôi	yêu	công_việc	thích	NLP	ghét	ở	một_mình
tôi	0	1	0	1	0	1	0	0
yêu	1	0	1	0	0	0	0	0
công_việc	0	1	0	0	0	0	0	0
thích	1	0	0	0	1	0	0	0
NLP	0	0	0	1	0	0	0	0
ghét	1	0	0	0	0	0	1	0
ở	0	0	0	0	0	1	0	1
một_mình	0	0	0	0	0	0	1	0
.	0	0	1	0	0	0	0	0

Bảng 1: Windows based cooccurence matrix của corpus trên với Window length 1

Tuy nhiên, vấn đề ở đây là khi tăng kích thước của bộ từ vựng, số chiều của ma trận sẽ rất lớn: đòi hỏi nhiều không gian lưu trữ hơn, không những thế ma trận lưu trữ còn là ma trận thưa, rất kém hiệu quả.

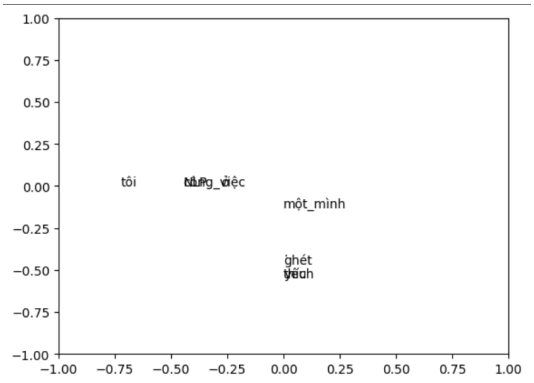
Giải pháp khắc phục cho vấn đề này là : Giảm chiều vector

Ý tưởng: Giảm số chiều của vector ban đầu xuống một số cố định mà vẫn đảm bảo lưu trữ được hầu hết các thông tin quan trọng. Vector thu được là một dense vector. Số chiều thường lấy trong khoảng từ 300-1000 chiều.

Phương pháp: Giảm kích thước với X sử dụng Singular Value Decomposition.

Ý tưởng cốt lõi của phương pháp Singular Value Decomposition[2] là phân tích ma trận ban đầu thành tích của 3 ma trận đặc biệt, sử dụng chéo hóa ma trận.

Sử dụng SVD với corpus trên (chi tiết tại ASM2/SVD.ipynb trong GitHub HungLV2512/NLP) ta được kết quả sau:



Hình 1: sử dụng thuật toán SVD với corpus trên

Ta có thể thấy những từ có ý nghĩa gần nhau như "NLP"và "công_việc", "yêu"và "thích"ghi đè lên nhau. Lý do là các cặp từ này đều cùng 1 tính chất, từ "ghét"ở gần từ "yêu"và "thích"hơn các từ khác vì chúng đều là tính từ.

Tuy nhiên, vấn đề của SVD là chi phí tính toán khá lớn, tỉ lệ với bậc hai của độ lớn bộ từ vựng với độ phức tạp $O(m * n^2)$. Không những thế, nó còn khó kết hợp khi có từ mới hoặc tài liệu mới.

2.4 Word2vec

Word2vec[3] là một kỹ thuật học máy được phát triển bởi đội ngũ nghiên cứu tại Google, nhằm mục đích chuyển đổi từ vựng thành các vector số học, giúp máy tính hiểu và xử lý ngôn ngữ tự nhiên một cách hiệu quả hơn. Mô hình này được giới thiệu vào năm 2013 và nhanh chóng trở thành một công cụ quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP).

Thay vì đếm và xây dựng ma trận đồng xuất hiện, word2vec[3] học trực tiếp word vector có số chiều thấp trong quá trình dự đoán các từ xung quanh mỗi từ. Đặc điểm của phương pháp này là nhanh hơn và có thể dễ dàng kết hợp một câu một văn bản mới hoặc thêm vào từ vựng.

Word2vec[3] là một mạng neural 2 lớp với duy nhất 1 tầng ẩn, lấy đầu vào là một corpus lớn và sinh ra không gian vector(với số chiều khoảng vài trăm), với mỗi từ duy nhất trong corpus được gắn với một vector tương ứng trong không gian.

Các word vectors được xác định trong không gian vector sao cho những từ có chung ngữ cảnh trong corpus được đặt gần nhau trong không gian. Dự đoán chính xác cao về ý nghĩa của một từ dựa trên những lần xuất hiện trước đây.

Word vector học từ mô hình word2vec rất phù hợp để trả lời cho câu hỏi: **Nếu A là B thì C là ...**

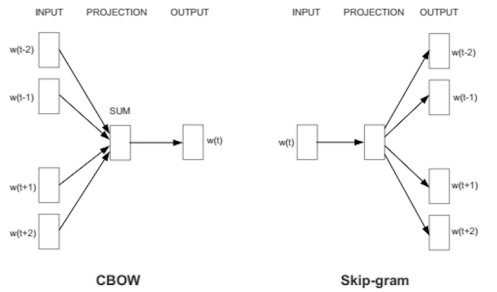
Nếu ta gán nhãn các thuộc tính cho một vector từ giả thiết, thì các vector được biểu diễn theo word2vec sẽ có dạng như sau:

		Vua	Hoàng hậu	Phụ nữ	Công chúa
Hoàng gia		0.99	0.99	0.02	0.98
Nam tính		0.99	0.05	0.01	0.02
Nữ tính		0.05	0.93	0.999	0.94
Tuổi		0.7	0.6	0.5	0.1

Hình 2: Ví dụ các vector được biểu diễn theo word2vec

Có 2 cách xây dựng word2vec:

- Sử dụng ngữ cảnh để dự đoán mục tiêu(CBOW).
- Sử dụng một từ để dự đoán ngữ cảnh mục tiêu(skip-gram)(cho kết quả tốt hơn với dữ liệu lớn).



Hình 3: Mô hình CBOW và mô hình Skip-gram

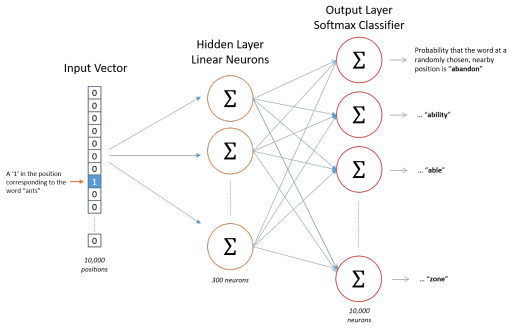
Do những ưu điểm của Skip-gram Model vượt trội hơn người anh em của nó, nên trong báo cáo này, em chỉ đi trọng tâm vào mô hình này.

Skip-gram Model

Mục tiêu: Học trọng số các lớp ẩn, các trọng số này là các words vector

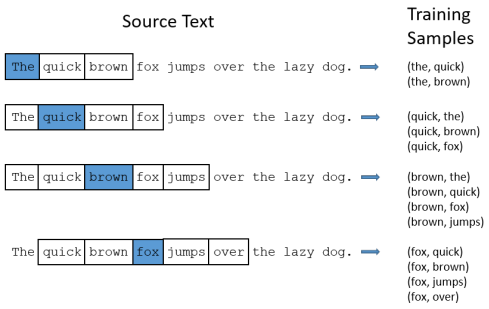
Cách thức: Cho một từ cụ thể ở giữa câu(input word), nhìn vào những từ ở gần và chọn ngẫu nhiên. Mạng neural sẽ cho chúng ta biết xác suất của mỗi từ trong từ vựng về việc trở thành từ gần đó mà chúng ta chọn.

Dưới đây là mô hình kiến trúc của mạng Skip-gram và cách xây dựng training data.



Hình 4: Mô hình skip-gram

Ví dụ: Xây dựng training data với windows size = 2. Ở đây windows được hiểu như một cửa sổ trượt qua mỗi từ. Windows size = 2 tức là lấy 2 từ bên trái và bên phải mỗi từ trung tâm.



Hình 5: Ví dụ Mô hình skip-gram

Model Details

- Xây dựng bộ từ vựng
- Biểu diễn mỗi từ thành các one-hot-vector
- Đầu ra là một vector duy nhất, có kích thước bằng kích thước của bộ từ vựng, thể hiện xác suất của mỗi từ được là lân cận của từ đầu vào.
- Không có hàm kích hoạt trên tầng ẩn
- Hàm kích hoạt trên tầng output là softmax
- Trong quá trình training, input là 1 one-hotvector, ouput cũng là 1 one-hot-vector
- Trong quá trình đánh giá sau khi training, đầu ra phải là 1 phân bố xác suất.

Vấn đề

- Mạng Neural lớn: Giả sử words vector với 300 thuộc tính, và từ vựng là 10k từ. Mạng neural có ma trận trọng số lớn, kích thước của ma trận trọng số là 300*10000 bằng 3 triệu giá trị.
- Chạy Gradient Descent sẽ rất chậm.

Word2vec cải tiến

Có 3 cải tiến cơ bản cho mô hình word2vec truyền thống:

- Xử lý các cặp từ thông dụng hoặc cụm từ như là một từ đơn
- Loại bỏ các từ thường xuyên lặp lại để giảm số lượng các ví dụ huấn luyện
- Sửa đổi mục tiêu tối ưu hóa bằng một kỹ thuật gọi là "Negative Sampling".

Cải tiến 1: Xử lý cụm từ như một từ đơn

Ví dụ các từ như "thành_phố_Cảng" có nghĩa khác nhau với từng từ "thành_phố" và "cảng",...

Chúng ta sẽ coi như đó là một từ duy nhất, với word vector của riêng mình.

Điều này sẽ làm tăng kích thước từ vựng. (Tìm hiểu thêm về *word2phrase*)

Cải tiến 2: Loại bỏ các từ thường xuyên lặp lại

Các từ thường xuyên lặp lại như "các", "những",... không cho chúng ta biết thêm nhiều hơn về ý nghĩa của những từ đi kèm nó, và chúng cũng xuất hiện trong ngữ cảnh của khá nhiều từ.

Chúng ta sẽ xác định xác suất loại bỏ, giữ lại một từ trong từ vựng thông qua tần suất xuất hiện của nó.

Cải tiến 3: Negative Sampling

Mỗi mẫu huấn luyện chỉ thay đổi một tỷ lệ phần trăm nhỏ các trọng số, thay vì tất cả chúng.

Nhớ lại: Khi huấn luyện mạng với 1 cặp từ, đầu ra của mạng sẽ là 1 one-hot vector, neural đúng thì đưa ra 1 còn hàng ngàn neural khác thì đưa ra 0.

Chọn ngẫu nhiên 1 số lượng nhỏ các neural "negative" kết hợp với neural "positive" để cập nhật trọng số. (chọn là 5-20 hoạt động tốt với các bộ dữ liệu nhỏ, 2-5 với bộ dữ liệu lớn).

3 Xây dựng mô hình không gian vector cho Tiếng Việt với VietNam Wikipedia.

Trong báo cáo này, em sẽ tiến hành huấn luyện lại mô hình mặc dù đã có những mô hình sẵn có. Một lý do là để hiểu rõ hơn về quá trình xây dựng mô hình không gian vector cho Tiếng Việt. Hơn nữa, việc huấn luyện dữ liệu trong một miền cụ thể thường mang lại kết quả tốt hơn so với việc sử dụng mô hình đã được huấn luyện trước, bởi vì vector từ được học từ dữ liệu phù hợp với bài toán sẽ phản ánh chính xác hơn.

3.1 Chuẩn bị dữ liệu

Link download dữ liệu: viwiki

Cài đặt WikiExtractor sau đó chạy dòng lệnh dưới đây để lấy nội dung của wikipedia (chi tiết tại ASM2/word2vec/wikiextractor.ipynb trong GitHub HungLV2512/NLP)

```
python -m wikiextractor.WikiExtractor -o output_path input_path
```

3.2 Tiền xử lý dữ liệu

Đầu tiên gộp các file extracted từ wikipedia thành 1 file duy nhất (ASM2/word2vec/combine_data.ipynb)

Các bước để tiền xử lý dữ liệu (ASM2/word2vec/make_data.ipynb) bao gồm:

- Loại bỏ thẻ html dư thừa
- Tách câu: Do dữ liệu lấy từ wikipedia nên mình có thể đơn giản là tách câu dựa vào dấu câu.
- Tách từ
- Chuẩn hóa dữ liệu: Xóa dấu(do mình chỉ quan tâm tới word)
- Loại stopword

3.3 Word2vec

Em train mô hình word2vec với CBOW và Skip-gram để đánh giá sự khác biệt của 2 mô hình (chi tiết tại ASM2/word2vec/w2v.ipynb)

```
from gensim.models import Word2Vec

pathdata = '/content/drive/MyDrive/Colab Notebooks/NLP/word2vec/datatrain.txt'

def read_data(path):
    traindata = []
    sents = open(pathdata, 'r').readlines()
    for sent in sents:
        traindata.append(sent.split())
    return traindata


if __name__ == '__main__':
    train_data = read_data(pathdata)

    # Skip-gram
    model = Word2Vec(train_data, vector_size=150, window=10, min_count=2, workers=4, sg=1)
    model.wv.save("/content/drive/MyDrive/Colab Notebooks/NLP/word2vec/model/word2vec_skipgram.mo

    #CBOW
    model = Word2Vec(train_data, vector_size=150, window=10, min_count=2, workers=4, sg=0)
    model.wv.save("/content/drive/MyDrive/Colab Notebooks/NLP/word2vec/model/word2vec_cbow.model'
```

3.4 fasttext

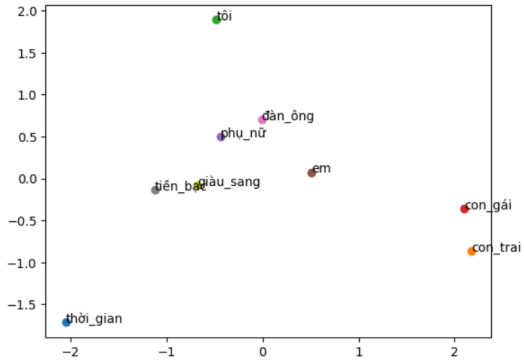
Một nhược điểm lớn của word2vec là nó chỉ sử dụng được những từ có trong dataset, để khắc phục được điều này chúng ta có FastText là mở rộng của Word2Vec, được xây dựng bởi facebook năm 2016. Thay vì training cho đơn vị word, nó chia text ra làm nhiều đoạn nhỏ được gọi là n-gram cho từ, ví dụ apple sẽ thành app, ppl, and ple, vector của từ apple sẽ bằng tổng của tất cả cái này. Do vậy, nó xử lý rất tốt cho những trường hợp từ hiếm gặp.

Chi tiết code tại: ASM2/word2vec/fasttext.ipynb

```
    pathdata = './datatrain.txt'

def read_data(path):
    traindata = []
    sents = open(pathdata, 'r').readlines()
    for sent in sents:
        traindata.append(sent.decode('utf-8').split())
    return traindata

if __name__ == '__main__':
    train_data = read_data(pathdata)
    model_fasttext = FastText(size=150, window=10, min_count=2, workers=4, sg=1)
    model_fasttext.build_vocab(train_data)
    model_fasttext.train(train_data, total_examples=model_fasttext.corpus_count, epochs=model_fas

    model_fasttext.wv.save("../model/fasttext_gensim.model")
```

3.5 Sử dụng mô hình

Sử dụng các model ở trên để visualize lên không gian 2 chiều với các từ tôi, phụ nữ, đàn ông, em, con gái, con trai, thời gian, tiền bạc. Chi tiết xem tại ASM2/word2vec/visualize+use_model.ipynb



Hình 6: Ví dụ sử dụng model fasttext để visualize các từ trên

Ngoài ra, em còn sử dụng các mô hình để dự đoán từ gần nghĩa nhất với từ "phụ_nữ"thì thấy rằng mô hình fasttext có vẻ chuẩn nhất khi dự đoán là từ "đàn_bà"trong khi 2 mô hình word2vec lại dự đoán ra từ "đàn_ông"

4 Tổng kết

Trong bài viết này, em đã trình bày về các cách để có thể ánh xạ một từ sang một không gian vector mà vẫn giữ được ý nghĩa của từ thông qua ngữ cảnh của chúng.

Những word vector này là vô cùng quan trọng, là đầu vào cho các thuật toán Machine Learning, Deeplearning trong lĩnh vực xử lý ngôn ngữ tự nhiên sau này.

Cảm ơn thầy đã đọc báo cáo của em.

Tài liệu

[1] Samuels, J. I. One-hot encoding and two-hot encoding: Anintroduction. 2024.

[2] Hasanvand, A. An introduction to singularvalue decomposition. 2023.

[3] Mikolov, T., K. Chen, G. Corrado, et al. Efficient estimation of word representations in vector space. 2013.