NLP and Deep Learning
MAT3399

Lecture 2:
Text Preprocessing & Training Word2vec

Tuan Anh Nguyen @ Aimesoft
ted.nguyen95@gmail.com

Content taken from Stanford CS244N

Why preprocessing?

Simple. Because we need to :)

h*****9 Hữu ích ···

Phân loại: 2#[Bộ 8]

Đúng với mô tả:10 kep toc hinh gau dau cac loai
Màu sắc:mau hong
Chất liệu:nhua, silicon, bong.

2 reviews · 2 photos

3 months ago

Take out

Đợi 20p ko nổi cốc nước, nv thì đông chứ ko phải ko có người,

q*****7 Hữu ích ···

Phân loại: #4

Bjsosjzjh híkskskoskskzkka hiwnwjuja hạnksksozizkzjxjjsbwbw

Common preprocessing techniques

- Tokenization: Tokenization is the process of converting a text into smaller pieces called tokens
- Lowercasing: Lowercasing is the process of converting all the alphabetic characters in a text to their lowercase form
- Stopword removal: Stopword removal involves eliminating common words that are deemed irrelevant in the text analysis
- Lemmatization: Lemmatization aim to reduce inflected or derived words to their base or root form

Preprocessing examples

Different languages have different ways to preprocess texts

You are funny! -> [you, fun]

Khoa học máy tính rất thú vị -> [khoa học, máy tính, thú vị]

コンピューターサイエンスは楽しいです -> [コンピューター, サイエンス, 楽しい]

Note that there are different ways to tokenize words for a language. Modern methods for English tokenization might treat subword as a token.

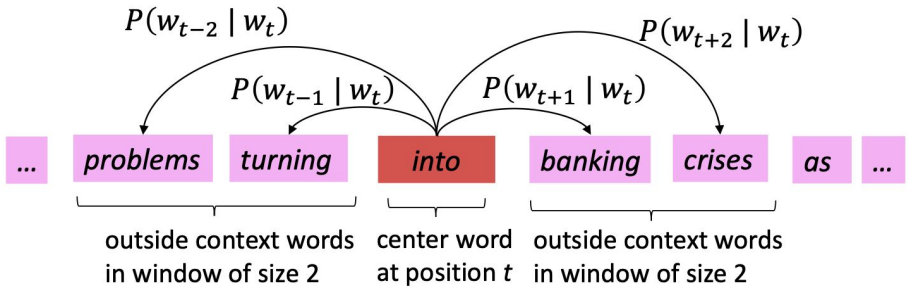
NLP libraries for python

- NLTK (Multi-language)
- spaCy (Multi-language)
- Underthesea (Vietnamese)
- VNCoreNLP (Vietnamese)
- MeCab (Japanese)

Reminder: Word2vec

Idea:

- We have a large corpus (“body”) of text: a long list of words
- Every word in a fixed vocabulary is represented by a vector
- Go through each position *t* in the text, which has a center word *c* and context (“outside”) words *o*
- Use the similarity of the word vectors for *c* and *o* to calculate the probability of *o* given *c* (or vice versa)
- Keep adjusting the word vectors to maximize this probability



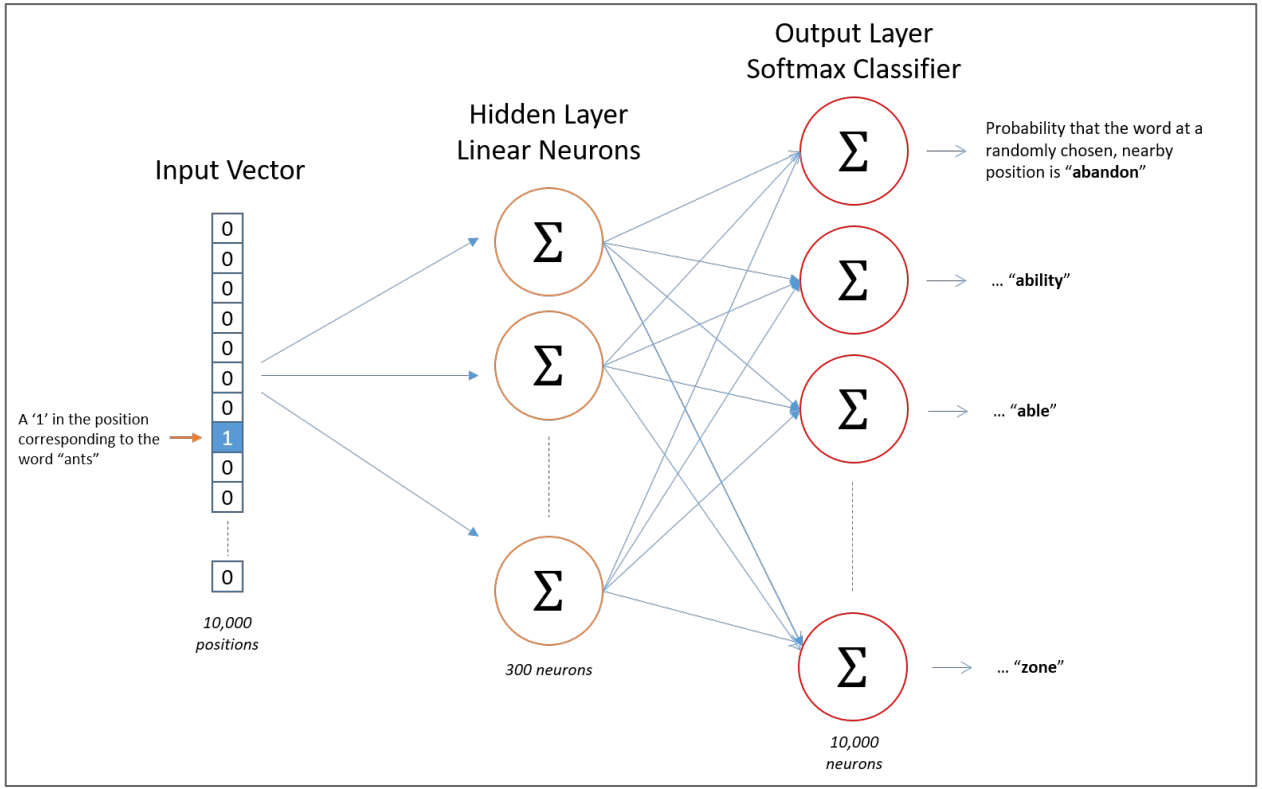
Skip-gram architecture

The hidden layer operates as a lookup table. The output of the hidden layer is just the “word vector” for the input word.

Read [skip-gram paper]

Example:

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$



Skip-gram cost function

$$-\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

Minimize this

T: Number of words in total

c: Size of the context window

Skip-gram formulation

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$

v: Input vector representation of word w

v': Output vector representation of word w

W: Number of words in vocabulary

Skip-gram with negative sampling

The normalization term is computationally expensive (when many output classes):

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$

Proportional to W

• Hence, in standard word2vec we implement the skip-gram model with negative sampling

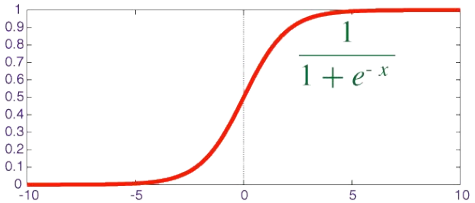
• Main idea: train binary logistic regressions to differentiate a true pair (center word and a word in its context window) versus several “noise” pairs (the center word paired with a random word)

Skip-gram with negative sampling

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

Maximize this

Sigmoid function



Sample k negative words from noise distribution P_n(w)

- Maximize probability that real outside word appears; minimize probability that random words appear around center word
- Sample with the word probability raised to the 3/4 power
- The power makes less frequent words be sampled more often

How do we evaluate word vectors?

Intrinsic:
- Evaluation on a specific/intermediate subtask
- Fast to compute
- Helps to understand that system
- Not clear if really helpful unless correlation to real task is established

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

Extrinsic:
- Evaluation on a real task
- Can take a long time to compute accuracy
- Unclear if the subsystem is the problem or its interaction or other subsystems
- If replacing exactly one subsystem with another improves accuracy -> Winning!

Coding Exercise

- Implement functions for text processing using any library and any language of your choice. You need to do these techniques:
  - Tokenization
  - Lowercasing
  - Stopword removal
  - Lemmatization (not applied for Vietnamese)
- Train word2vec using gensim with this dataset

Advanced exercise: Train word2vec using keras library