CS224n: NLP with Deep Learning

Lecture 1: Word Vectors 1

How to get the meaning of a word?

WordNet: a dictionary of synonyms

- Using dictionary such as WordNet, which will store the synonyms of words

Problems:

- Missing nuances
- Missing new words / new meanings of words: ex: *ninja*
- Can't compute accurate word similarity if they aren't in the same synonym sets

Traditional NLP:

- Everything until 2012
- Words were regarded as discrete symbols -> Usage of 1-hot vectors

One-hot encoding

Problems:

- No notion of similarity: word vectors for 'hotel' and 'motel' are orthogonal: similarity = 0
- Big dimensions (200k-1m words)

Distributional semantics

A word's meaning is given by the words that frequently appear close-by

- A word vector = a smaller, but dense vector
- Typical dimensions:
  - Min: 50

- Average: 300
- Max: 2000

Word2Vec

- Word2vec = a framework for learning word vectors

- Closeness in the vector space ~ Word similarity

General Idea

- Initialize each word vector randomly
- For each position in the text:
  - a center word *c*
  - context ('outside') words *o*
  - Calculate $P(o|c)$ or $P(c|o)$
  - Adjust the word vectors to maximize this probability

Let's maximize the Likelihood, with respect to all the θ variables (θ is the concatenation of all the word vectors)

Likelihood

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j}|w_t;\theta)$$

Let's maximize $L(\theta)$, ie minimize $-L(\theta)$

Objective function $J(\theta)$

$$J(\theta) = -\frac{1}{T} log L(\theta)$$

How do we calculate the probability $P(w_i|wj)$?

$$P(o|c) = \frac{exp(u_0^T \cdot v_c)}{\sum_{w \in V} exp(u_w^T \cdot v_c)}$$

Le dénominateur somme sur $w \in V$ pour que $\sum_{o \in V} P(o|c) = 1$

P(o|c) = softmax (dot_product(o,c) for o in outsides)

ie it is the normalized similarity of our center word, compared to all context words

SoftMax

Thus, by using softmax, we get a probability distribution

- Max: because it amplifies probability of the largest elements
- Soft: because it still assigns some probability to the smaller elemnts

Our parameters θ

θ is the concatenation of all the word vectors of our vocabulary

- Every word has 2 word vectors
- With d-dimensional vectors, and V words,

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

$$\theta \in \mathbb{R}^{2dV}$$

Calculating the gradient

$$\frac{\partial}{\partial v_c} log P(o|c) = u_o - \sum_{x=1}^{V} p(x|c) u_x$$

Slope = observed representation of our context word - what our model thinks the context should look like

Where what our model thinks the context should look like = Expectation

Slope with respect to context word = Actual context word - Expected context word

$$Slope\ with\ respect\ to\ context\ word = Actual\ context\ word - Expected\ context\ word$$