Word Representation Learning

Motivation, Word2Vec

AUA, DS330 Deep Learning 2021, Hovhannes Tamoyan

Motivation

We need to input a sequence of words/tokens to an NN.

As we know the words/tokens are represented by characters, but NNs work with numbers, thus we need to transform the words/tokens into group of numbers.

Not only this, but also we want to understand the relationship between the words.

E.g. Woman+King-Man=?

This process is called representation learning in NLP.

Representing Words as Discrete Symbols

In traditional NLP, we regard words as discrete symbols (one-hot vectors): motel, conference, hotel.

motel = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]

hotel = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]

The length of the vector depends on the size of the vocabulary (e.g., 10^5).

Representing Words by Their Context

"You shall know a word by the company it keeps"

J. R. Firth 1957:11

When a word w appears in a text, its context is the set of words that appear nearby (within a fixed-size window).

... the government debt problems turning into banking crises as happened in 2009 ...

... saying that Europe needs a unified banking regulation to replace the hodgepodge ...

... India has just given its banking system a shot in the arm ...

Word Vectors

We will build a dense vector for each word/token, chosen so that it is similar to vectors of words that appear in similar contexts.

banking = [0.286, 0.792, -0.177 … 0.271]

Note: word vectors often are called word embeddings or just embeddings.

Word2vec

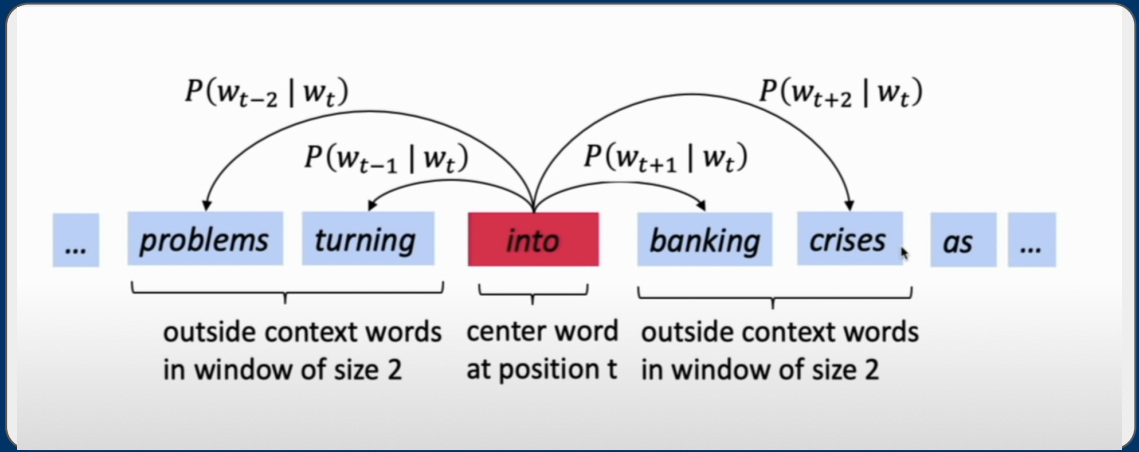
Is a framework for learning word vectors. (Mikolov et al. 2013)

Idea:

- We have a large corpus of text
- 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 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 vectors to maximize the this probability.

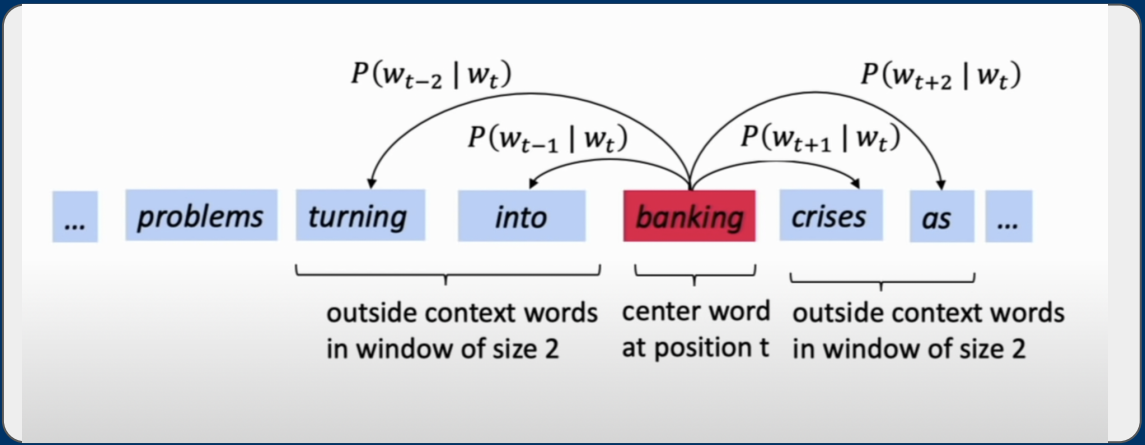
Word2vec contd.

Example windows of and process for computing $P(w_{t+j}|w_t)$



Word2vec contd.

Example windows of and process for computing $P(w_{t+j}|w_t)$



Word2vec: objective function

For each position t = 1,…,T, predict context words within a window of fixed size m, given center word w_j.

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

The objective function is the average negative log likelihood:

$$J(\theta) = -\frac{1}{T} log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m} log P(w_{t+j}|w_t;\theta)$$

Minimizing objective function ⇔ Maximizing predictive accuracy

Word2vec: objective function

How to calculate $P(w_{t+j}|w_t;\theta)$?

We will use two vectors per word w:

- v_w when w is a center word.
- u_w when w is a context word.

Then for a center word c and a context word o:

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

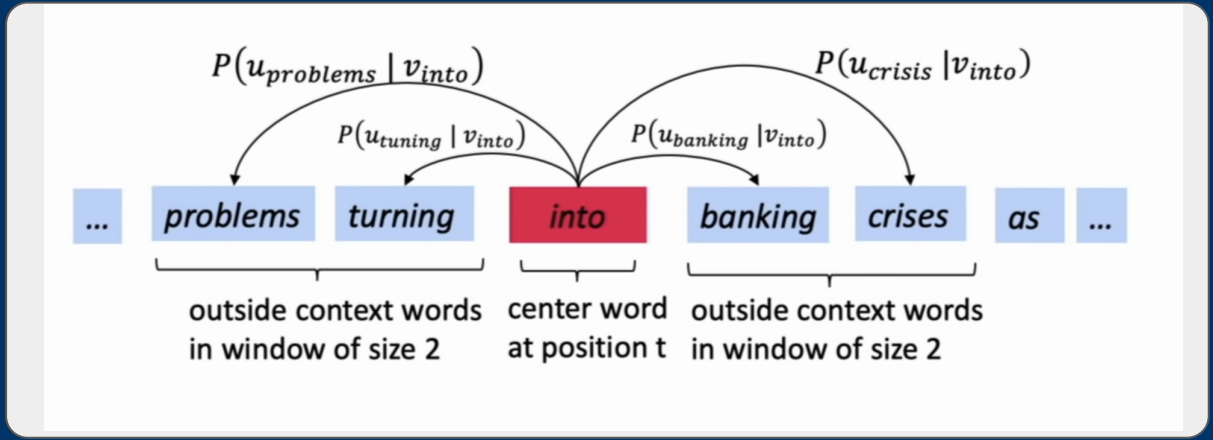
Dot product compares similarity of o and c.
Larger dot product = larger probability

Normalize over entire vocab to give probability distribution.

Word2vec overview with Vectors

Example window and process for calculating $P(w_{t+j}|w_t;\theta)$

$P(problems|into;u_{problems},v_{into},\theta)$ for short $P(u_{problems}|v_{into})$



Word2vec: training

Recall: θ represents all model parameters, in one long vector

In our case with d-dimensional vectors and V-many words:

Every word has two vectors.

We optimize the θ by walking down the ∇

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

Word2vec: two variants

Skip-grams (SG) - predict context (outside) words (position independent) given center word.

Continues Bag of Words (CBOW) - Predict center word from (bag of) context words.

We present Skip-grams

The SG model with negative sampling

The normalization factor is too computationally expensive.

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

Hence, in standard word2vec the SG is implemented with negative sampling.

The idea: train binary logistic regression for a true pair (center word and word in its context window) versus several noisy pairs (the center word paired with a random word).

Other methods of word to vector learning

Count based approach - Unlike direct prediction(SG, CBOW etc) here we take a word's all frequently surrounding words and count them, to make those more probabilistic in appearing near that word.

Encoding Meaning in Vector Differences - idea is that co-occurrence probability can encode meaning: e.g.

$$\frac{P(solid|ice)}{P(solid|steam)}$$ Is high

Other methods of word to vector learning

Combining the best of two worlds (GloVe) [Pennington et. al. 2014] - Combining the previous approaches.

fasttext - uses skip-grams model but the words are splitted into n-gram characters, this helps to represent a word which the model hasn't seen before. Is an **LSTM** (bidirectional) based model.

Transformer/BERT - gives different vectors for a word depending on it's company (contextual embeddings).

Thank You