Neural Language Model, RNNs

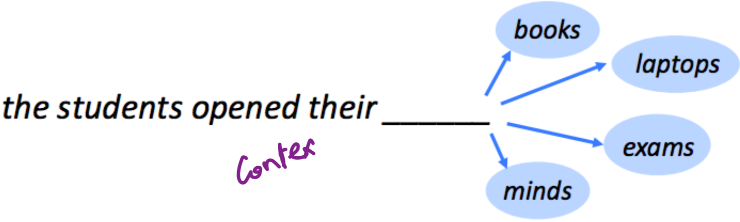
Pawan Goyal

CSE, IIT Kharagpur

CS60010

Predictive typing

Language Modeling is the task of predicting what word comes next.



Contex

- **Goal:** Compute the probability of a sentence or sequence of words:

$$P(W) = \underline{P(w_1, w_2, w_3, \ldots, w_n)}$$

chain rule

- **Related Task:** probability of an upcoming word:

$$P(w_4 | w_1, w_2, w_3) \rightarrow \text{LM}$$

- A model that computes either of these is called a **language model**

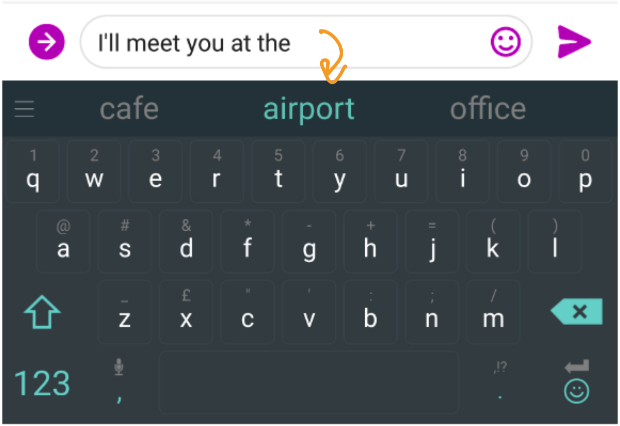
Language Modeling

- You can also think of a language model as a system that assigns probability to a piece of text.
- For example, if we have some text $x^{(1)}, \ldots, x^{(T)}$, then the probability of this text (according to the Language Model) is:

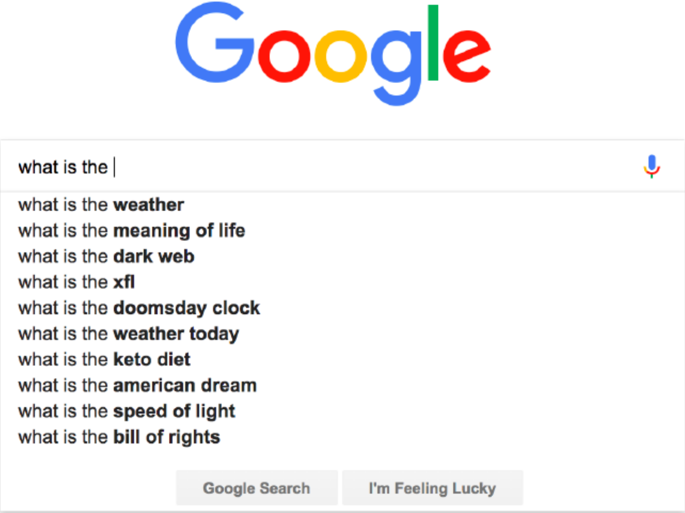
$$
\begin{aligned}
P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) &= P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} |\, \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} |\, \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)}) \\
&= \prod_{t=1}^{T} \underbrace{P(\boldsymbol{x}^{(t)} |\, \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})}_{\text{This is what our LM provides}}
\end{aligned}
$$

LM

You use language models every day!



You use language models every day!



Why should we care about language modeling?

- Language Modeling is a benchmark task that helps us measure our progress on understanding language
- Language Modeling is fundamental to many NLP tasks, especially those involving generating text or estimating the probability of text:
  - Predictive typing
  - Speech recognition
  - Handwriting recognition
  - Spelling/grammar correction

translation
chatbot

compress
Lot of data
info
Parameter set

n-gram language models

the students opened their ______ books

Pr(books | their)

Pr(books | opened their)

Question: How to learn a Language Model?

Answer (pre- Deep Learning): learn a *n*-gram Language Model!

Definition: A *n*-gram is a chunk of *n* consecutive words.

- unigrams: "the", "students", "opened", "their"
- bigrams: "the students", "students opened", "opened their"
- trigrams: "the students opened", "students opened their"
- 4-grams: "the students opened their"

Idea: Collect statistics about how frequent different n-grams are, and use these to predict next word.

k = 101 V^{101} 100,000

n-gram language models

- First we make a simplifying assumption: $\boldsymbol{x}^{(t+1)}$ depends only on the preceding *n-1* words.

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\overbrace{\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)}}^{n\text{-1 words}})$$ (assumption)

n-gram

prob of a n-gram

prob of a (n-1)-gram

$$= \frac{\boxed{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}}{\boxed{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}}$$ (definition of conditional prob)

- **Question:** How do we get these *n*-gram and (*n*-1)-gram probabilities?
- **Answer:** By counting them in some large corpus of text!

$$\approx \frac{\text{count}(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{\text{count}(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$ (statistical approximation)

n-gram language models: Example

Suppose we are learning a 4-gram Language Model.

~~as the proctor started the clock, the~~ *students opened their* ______ books

discard

$\underbrace{\textit{students opened their}}_{\text{condition on this}}$

20 / 1000 = 0.02

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

<u>For example, suppose that in the corpus:</u>

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - → P(books | students opened their) = 0.4
- "students opened their exams" occurred 100 times
  - → P(exams | students opened their) = 0.1

Should we have discarded the "proctor" context?

Storage Problems with n-gram Language Model

Storage: Need to store count for all *n*-grams you saw in the corpus.

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\boxed{\text{count}(\text{students opened their } \boldsymbol{w})}}{\text{count}(\text{students opened their})}$$

Increasing *n* or increasing corpus increases model size!

A fixed-window neural language model

o/p $W \in V$ V

$3dh + Vh$

h

3d i/p size

I/p Students opened their

1-hot

d-dim

?

$[3Vh + Vh]$

~~as the proctor started the clock~~ $\underbrace{\text{the students opened their}}_{\text{fixed window}}$ ______

discard

A fixed-window neural language model

output distribution

$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$

hidden layer

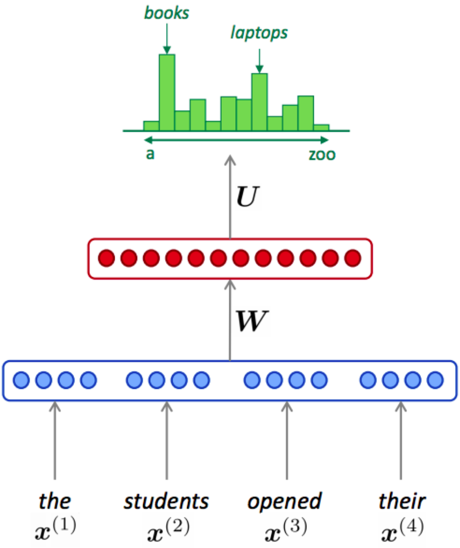
$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$

concatenated word embeddings

$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$

words / one-hot vectors

$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$



How do we obtain word representations?

In traditional NLP / IR, words are treated as discrete symbols.

One-hot representation

Words are represented as one-hot vectors: one 1, the rest 0s

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

What is the problem?

- Vector dimension = number of words in vocabulary (e.g., 500,000)
- The vectors are orthogonal, and there is no natural notion of similarity between one-hot vectors!

Word2Vec – A distributed representation

Distributional representation – word embedding?

Any word w_i in the corpus is given a distributional representation by an embedding

$$w_i \in R^d$$

i.e., a $d-$dimensional vector, which is mostly learnt!

$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

dimensions?

Distributional Representation: Illustration

If we label the dimensions in a hypothetical word vector (there are no such pre-assigned labels in the algorithm of course), it might look a bit like this:

Such a vector represents the 'meaning' of a word in some abstract way

Unsupervised

Just data

No labels

[no loss fn]

Self-supervised

Train data

[No human-generated labels] -

Generates labels from the data itself

Supervised

Training data with labels

Learning Word Vectors: Overview

2 context 1 center word 2 words

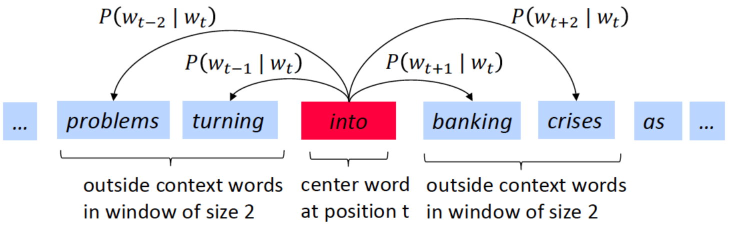
Basic Idea: Use self-supervision

- 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 ("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

* word vectors are your params

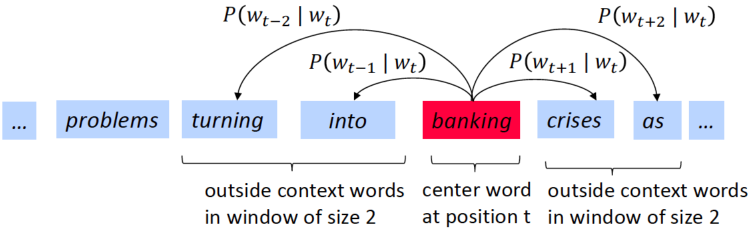
Word2Vec (Skip-gram) Overview

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



Word2Vec Overview

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



Word2Vec: objective function

We want to minimize the loss function:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

$2V$ vectors

v_w u_w

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

We will use two vectors per word w:

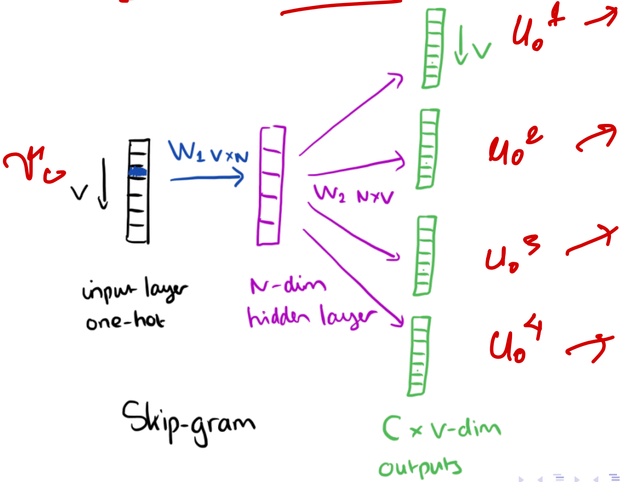
- v_w when w is a center word
- u_w when w is a context word

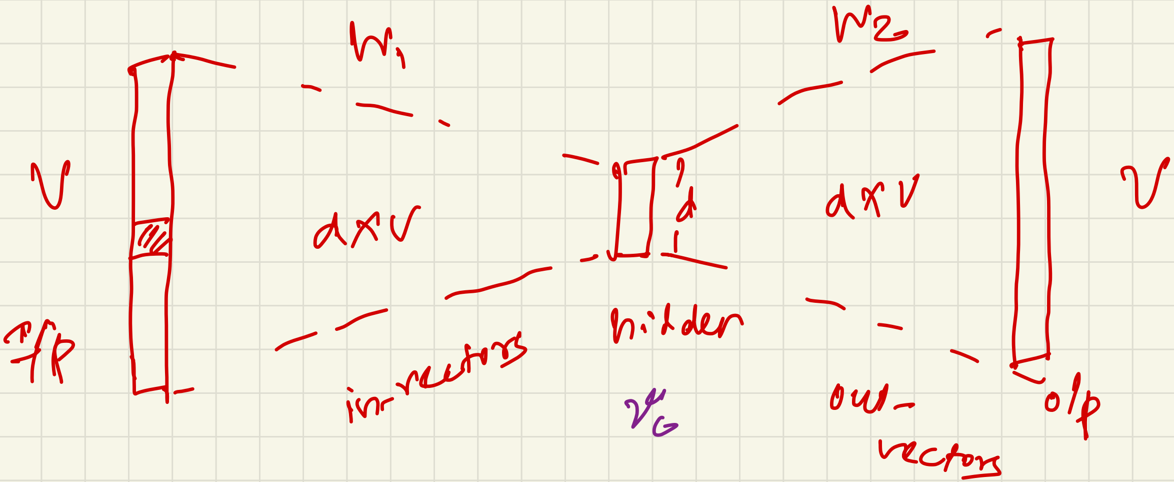
v_c u_o

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

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

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





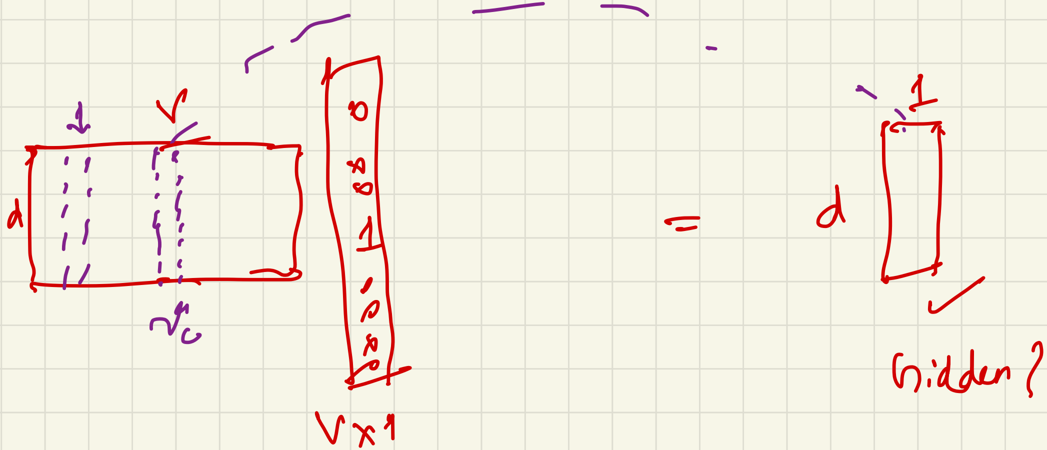
$\frac{c}{\text{banking}}$

$\frac{o}{\text{crises}}$

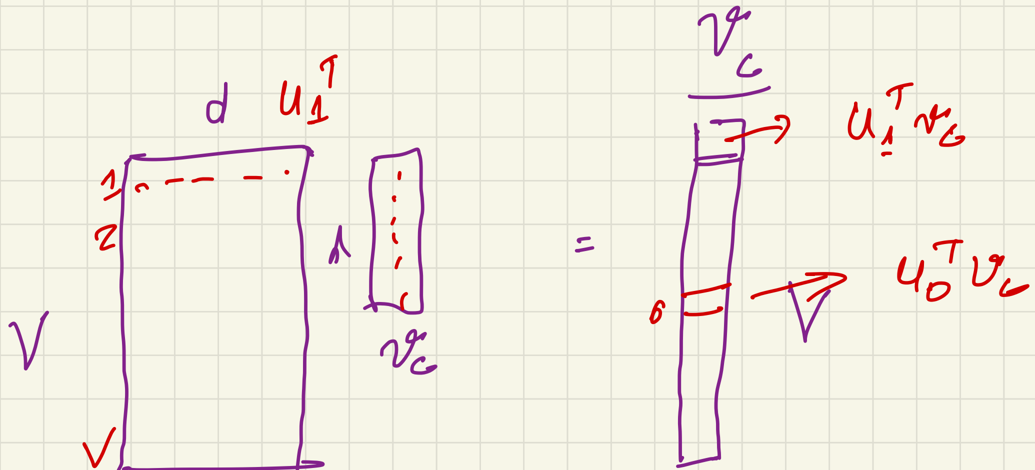
$$w \rightarrow \underbrace{v_w}_{d\text{-dim}} + \underbrace{u_w}_{d\text{-dim}}$$

c, o

$\boxed{2dV}$



$$-\log p(o|c)$$



softmax

$$\rightarrow \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

Try this problem

Skip-gram

Suppose you are computing the word vectors using Skip-gram architecture. You have 5 words in your vocabulary, $\{passed, through, relu, activation, function\}$ in that order and suppose you have the window, '*through relu activation*' in your corpora. You use this window with 'relu' as the center word and one word before and after the center word as your context.

Compute the loss

Also, suppose that for each word, you have 2-dim in and out vectors, which have the same value at this point given by [1,-1],[1,1],[-2,1],[0,1],[1,0] for the 5 words, respectively. As per the Skip-gram architecture, the loss corresponding to the target word "activation" would be $-log(x)$. What is the value of x?

Homework

- Compute partial derivative of the loss with respect to v_c

$$\frac{\partial}{\partial v_c}\left(\quad \right)$$