

Word Embedding

References:

- <https://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture01-wordvecs1.pdf>
- <http://www.cs.virginia.edu/~kc2wc/teaching/NLP16/slides/07-WordEmbedding.pdf>
- <http://jalammar.github.io/illustrated-word2vec/>
- https://www.cc.gatech.edu/classes/AY2020/cs7643_spring/slides_L13_EMBEDDING_word2vec_final_version.pdf



The Library of Trinity College Dublin

Representing words as discrete symbols

In traditional NLP, we regard words as discrete symbols:
`hotel, conference, motel` – a **localist** representation

Means one 1, the rest 0s

Words can be represented by **one-hot** vectors:

```
motel = [0 0 0 0 0 0 0 0 0 1 0 0 0]  
hotel = [0 0 0 0 0 0 1 0 0 0 0 0]
```

Vector dimension = number of words in vocabulary (e.g., 500,000)

Problem with words as discrete symbols

Example: in web search, if user searches for “Seattle motel”, we would like to match documents containing “Seattle hotel”.

But:

`motel = [0 0 0 0 0 0 0 0 1 0 0 0]`

`hotel = [0 0 0 0 0 0 1 0 0 0 0 0]`

These two vectors are **orthogonal**.

There is no natural notion of **similarity** for one-hot vectors!

Solution:

- Could try to rely on WordNet’s list of synonyms to get similarity?
 - But it is well-known to fail badly: incompleteness, etc.
- **Instead: learn to encode similarity in the vectors themselves**

Representing words by their context



- Distributional semantics: A word's meaning is given by the words that frequently appear close-by
 - “*You shall know a word by the company it keeps*” (J. R. Firth 1957: 11)
 - One of the most successful ideas of modern statistical NLP!
- When a word w appears in a text, its **context** is the set of words that appear nearby (within a fixed-size window).
- Use the many contexts of w to build up a representation of w

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

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

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



These **context words** will represent **banking**

Word vectors

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

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

Note: word vectors are sometimes called word embeddings or word representations. They are a distributed representation.

Word2vec: Overview

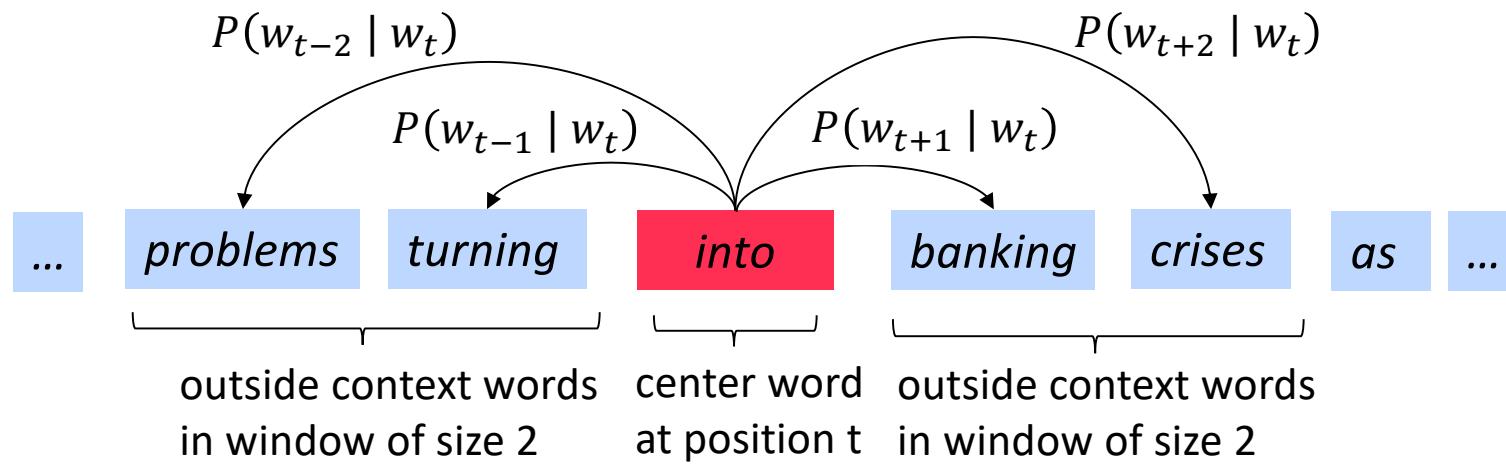
Word2vec (Mikolov et al. 2013) is a framework for learning word vectors

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 (“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

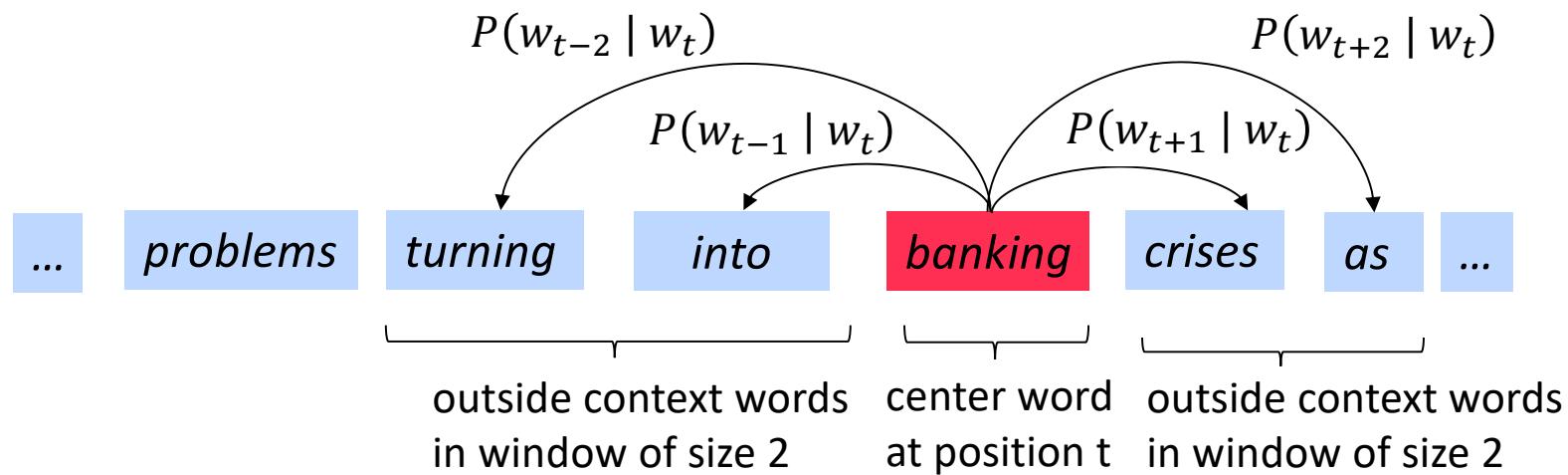
Word2Vec Overview

- 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

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

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

θ is all variables
to be optimized

sometimes called *cost* or *loss* function

The **objective function** $J(\theta)$ is the (average) negative log likelihood:

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

Minimizing objective function \Leftrightarrow Maximizing predictive accuracy

Word2vec: objective function

- We want to minimize the objective function:

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

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

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

Word2vec: prediction function

Exponentiation makes anything positive

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

Dot product compares similarity of o and c .
 $u^T v = u \cdot v = \sum_{i=1}^n u_i v_i$
Larger dot product = larger probability

Normalize over entire vocabulary
to give probability distribution

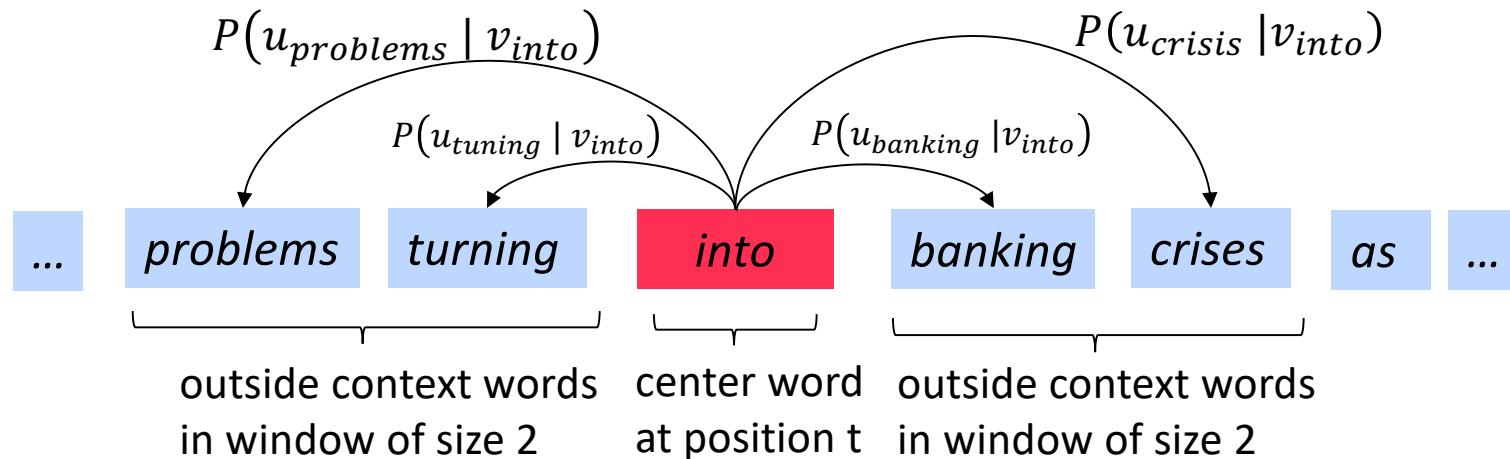
- This is an example of the **softmax function** $\mathbb{R}^n \rightarrow \mathbb{R}^n$

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)} = p_i$$

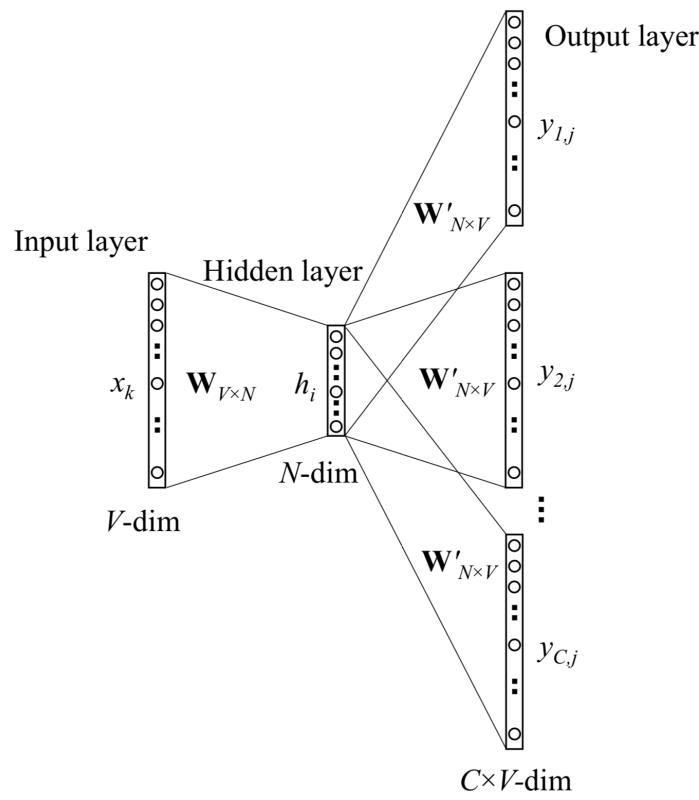
- The softmax function maps arbitrary values x_i to a probability distribution p_i
 - “max” because amplifies probability of largest x_i
 - “soft” because still assigns some probability to smaller x_i
 - Frequently used in Deep Learning

Word2Vec Overview with Vectors

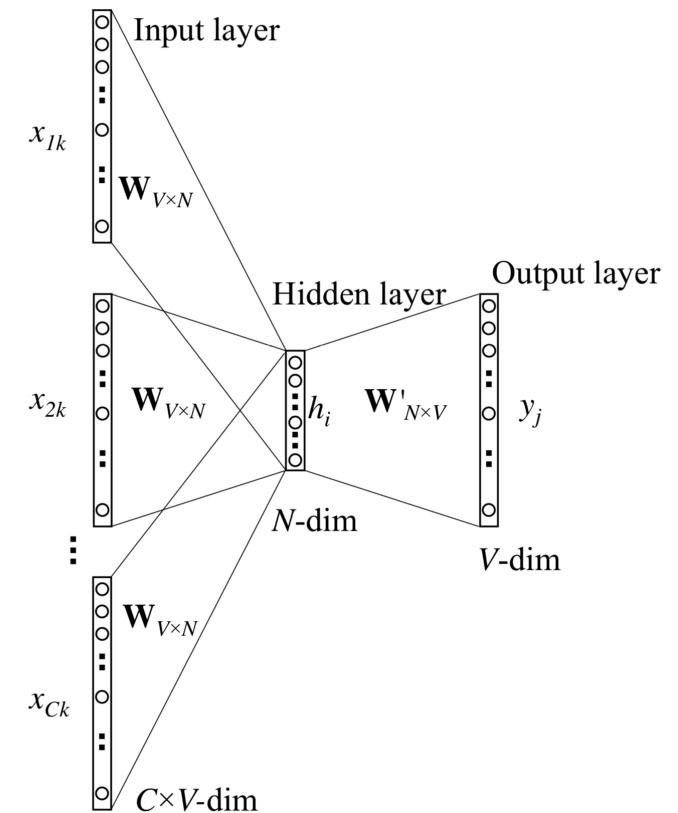
- Example windows and process for computing $P(w_{t+j} | w_t)$
- $P(u_{problems} | v_{into})$ short for $P(problems | into ; u_{problems}, v_{into}, \theta)$



Skip-gram v.s Continuous bag-of-words (CBOW)



Skip-gram: Predict context words by center word, i.e. $p(o|c)$

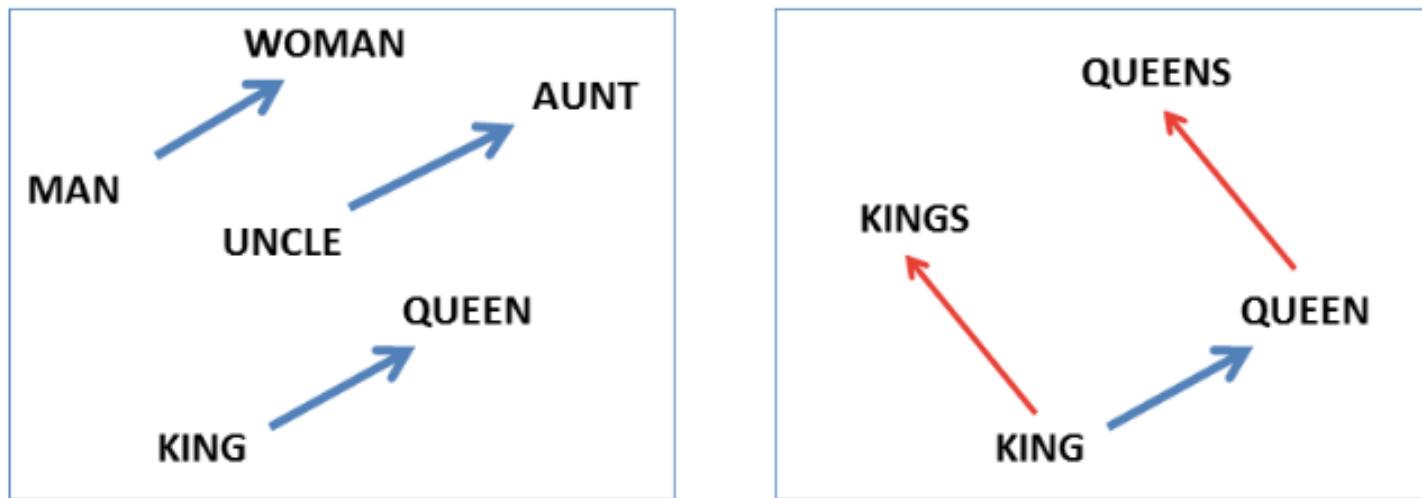


CBOW: Predict center word by context words, i.e. $p(c|o)$

Word Embeddings are Cool!

(An Obligatory Slide)

- e.g. king-man+woman = queen (Mikolov et al. 2013)



- “What is the female equivalent of king?” is not easily accessible in many traditional resources

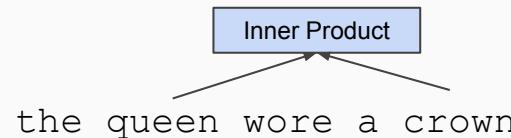
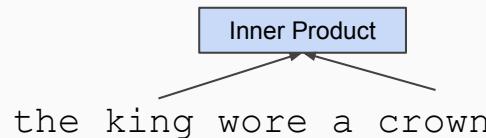
Pre-training in NLP

- Word embeddings are the basis of deep learning for NLP

king
↓
[-0.5, -0.9, 1.4, ...]

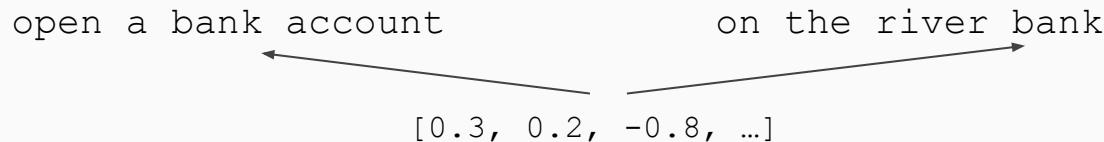
queen
↓
[-0.6, -0.8, -0.2, ...]

- Word embeddings (word2vec, GloVe) are often *pre-trained* on text corpus from co-occurrence statistics



Contextual Representations

- **Problem:** Word embeddings are applied in a context free manner



- **Solution:** Train contextual representations on text corpus



If you are interested in contextualized representations, join a deep learning class!