CS60010: Deep Learning

Recurrent Neural Network

Sudeshna Sarkar

Spring 2019

27 Feb 19

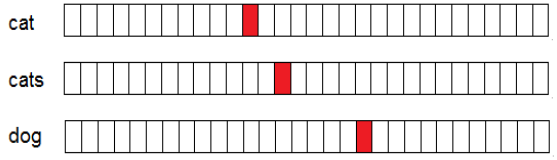
WORD EMBEDDING

NLP

- A way for computers to **analyze**, **understand**, and **derive meaning** from human language.
  - text mining
  - machine translation
  - automated question answering
  - automatic text summarization
  - and many more…

Word embeddings

- For the computer to be able to "understand" a vector representation of a word is required.
- Examples of early approaches:
  - One-hot vector
  - Joint distribution



1. I enjoy flying.

2. I like NLP.

3. I like deep learning.

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Curse of Dimensionality

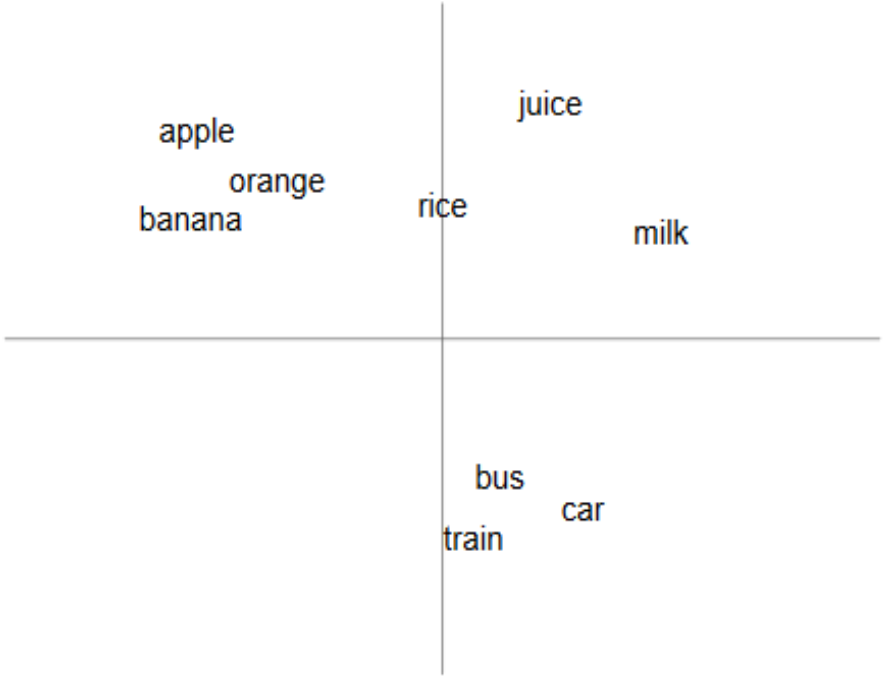
Mock example:

- Vocabulary size: 100,000 unique words.
- Window size: 10 context words (unidirectional).

1. *One-hot vector*: 100,000 free parameters, and **no knowledge of words semantic or syntactic relations**.
2. *Joint distribution*: $100{,}000^{10} - 1 =$ **$10^{50} - 1$ free parameters.**

A Neural Probabilistic Language Model. Bengio at al., 2003

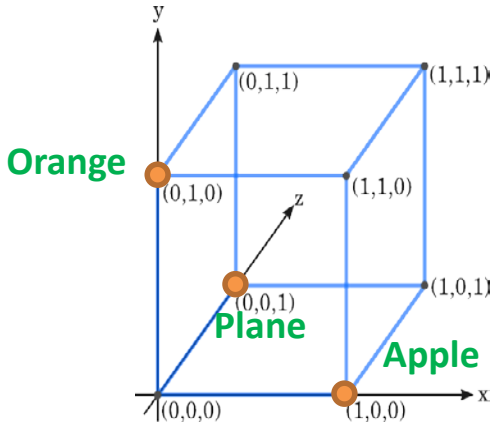
“A word is known by the company it keeps”



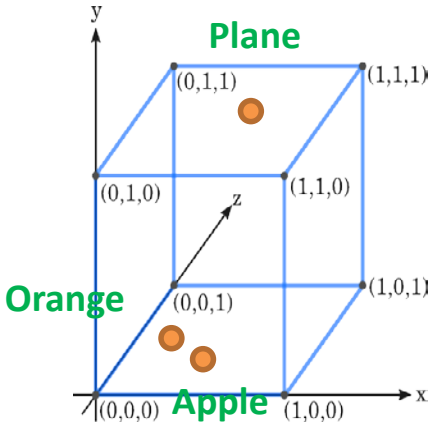
Curse of Dimensionality

- **Similar words** should be **close to each other** in the hyper dimensional space.
- **Non-similar words** should be **far apart from each other** in the hyper dimensional space.
- One-Hot vector example:
  - Apple = [1, 0, 0]
  - Orange = [0, 1, 0]
  - Plane = [0, 0, 1]

One-Hot vectors

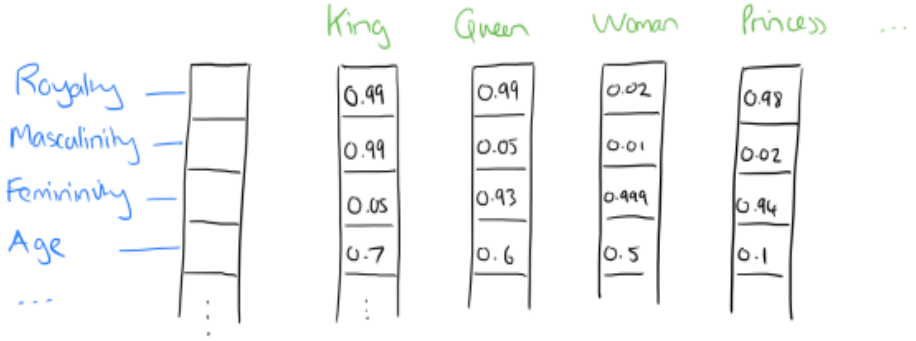


Wanted behavior



- Many – to – many representation of words.
- All vector cells participate in representing each word.
- Words are represented by **real** valued **dense** vectors of significantly **smaller dimensions** (e.g. 100 – 1000).

- Intuition: consider each vector cell as a representative of some feature.

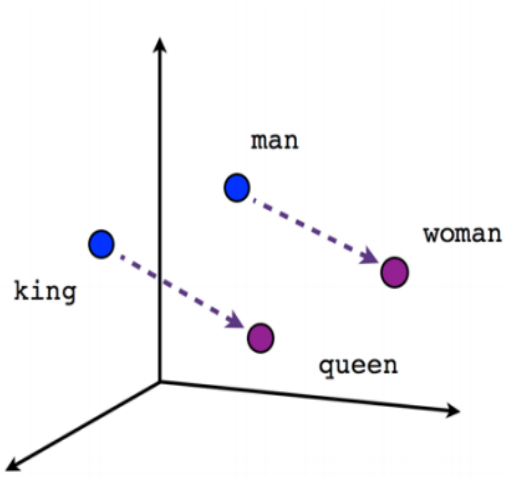


Distributed vectors motivation

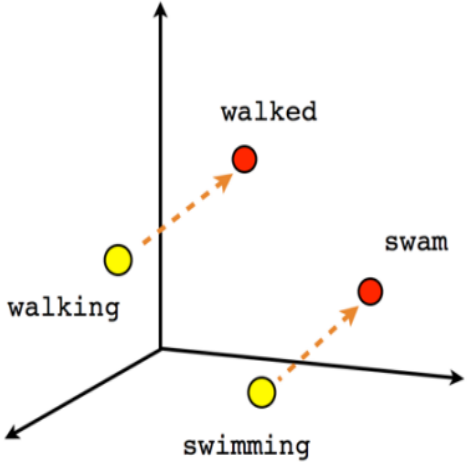
The amazing power of word vectors, Th morning paper blog, Adrian Colyer

The Power of Word Vectors

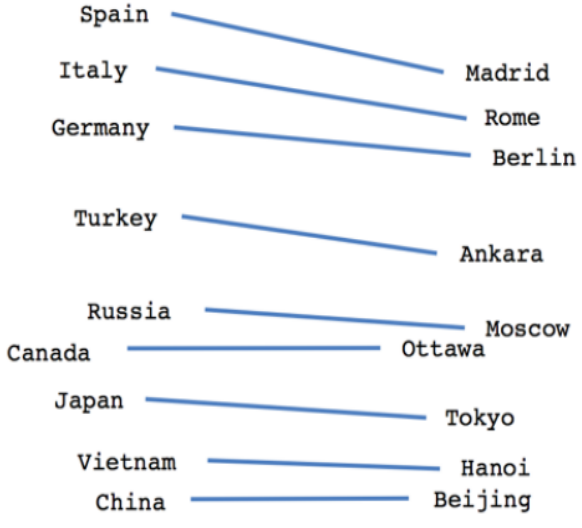
- They provide a fresh perspective to ***ALL*** problems in NLP.



Male-Female



Verb tense



Country-Capital

vector[Queen] = vector[King] - vector[Man] + vector[Woman]

Applications of Word Vectors

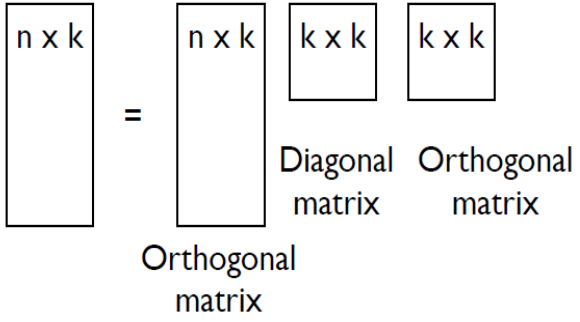
1. Word Similarity
2. Machine Translation
3. Part-of-Speech and Named Entity Recognition
4. Relation Extraction
5. Sentiment Analysis
6. Co-reference Resolution: Chaining entity mentions across multiple documents
7. Clustering
8. Semantic Analysis of Documents
9. Build word distributions for various topics

Building these magical vectors . . .

- How do we actually build these super-intelligent vectors, that seem to have such magical powers?
- The most famous methods to build such lower-dimension vector representations for words based on their context

1. Co-occurrence Matrix with SVD

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathsf{T}}$$



2. word2vec (*Google)*
3. Global Vector Representations (GloVe) (*Stanford)*

Word vectors

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

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

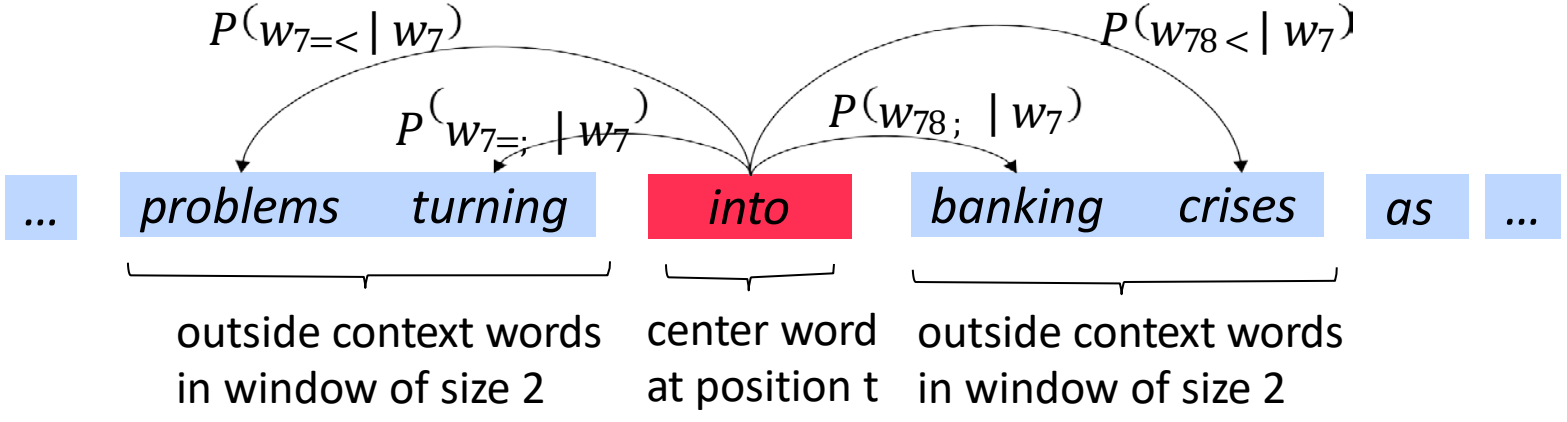
2. Word2vec: Overview

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

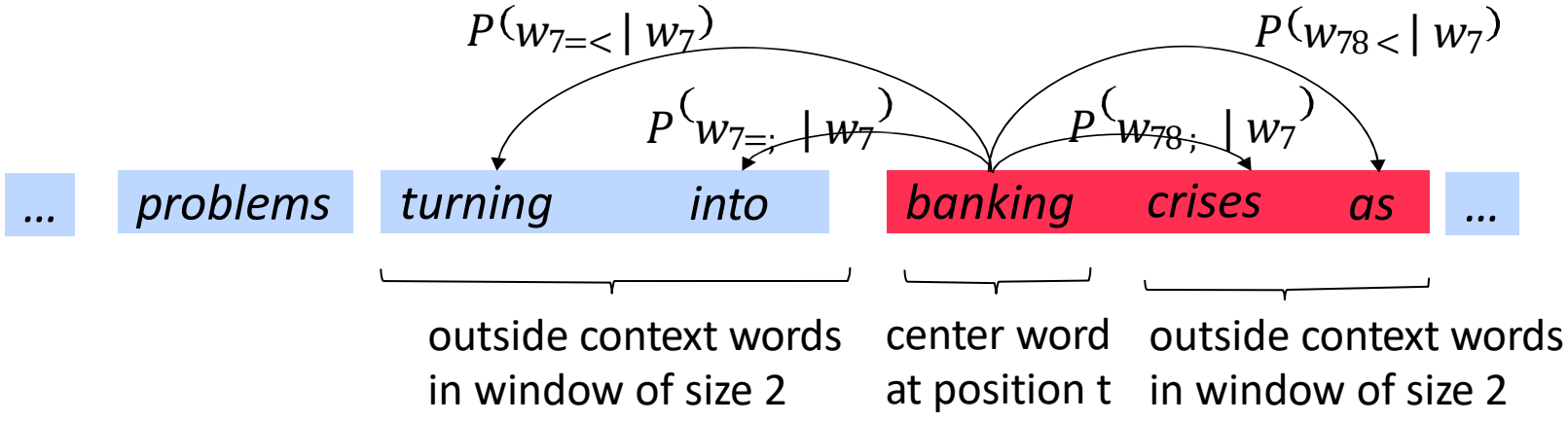
Word2Vec Overview

- Example windows and process for computing $P(w_{78\,9} \mid w_7)$



Word2Vec Overview

- Example windows and process for computing $P(w_{78\,9} \mid w_7)$



Word2vec: objective function

For each position $t = 1, \ldots, T$, predict context words within a window of fixed size *m*, given center word w_j
The objective function is the (average) negative log likelihood:

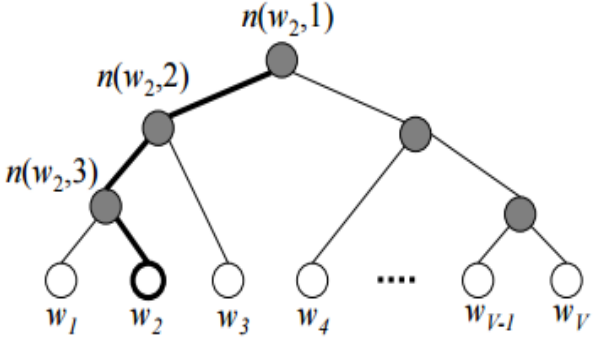
$$L = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t)$$

The basic Skip-Gram utilizes the SoftMax function:

$$p(c|w) = \frac{\exp({v'_c}^T v_w)}{\sum_{i=1}^{W}\exp({v'_i}^T v_w)}$$

Hierarchical SoftMax

- SoftMax is impractical because the cost of computing $\nabla \log p(c|w)$ is proportional to T, which is often large (10^5–10^7 terms).
- H-SoftMax can yield speedups of word prediction tasks of **50X** and more.
- **Computing the normalized probability** of a context word as a **path in binary tree** instead of going over all examples in the corpus.



Distributed Representations of Words and Phrases and their Compositionality. Mikolov at al., 2013

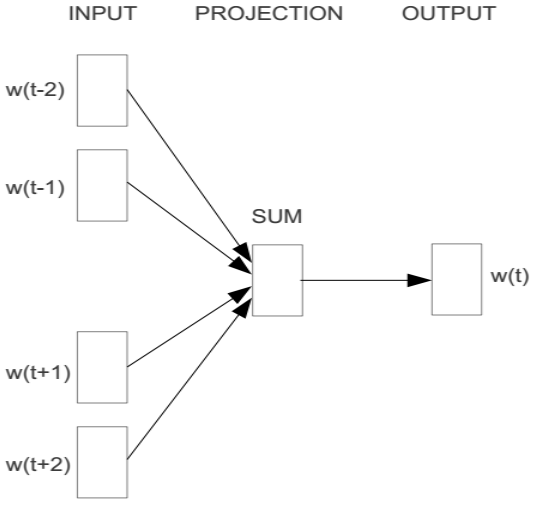
How do we choose negative samples?

- For each positive example we draw **K negative examples.**
- The negative examples are drawn according to the unigram distribution of the data

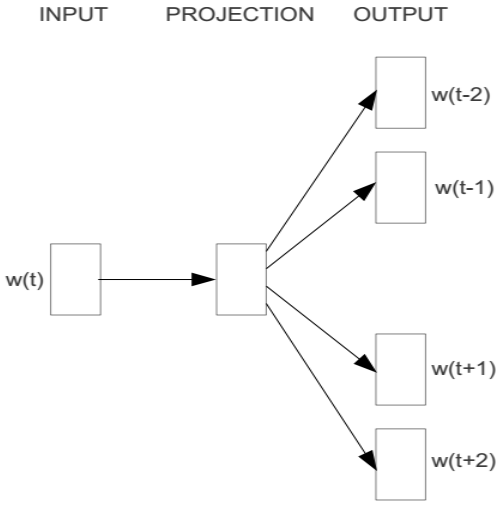
$$P_D(c) = \frac{\#(c)}{|D|}$$

Represent the meaning of word – word2vec

- 2 basic neural network models:
  - Continuous Bag of Word (CBOW): use a window of word to predict the middle word
  - Skip-gram (SG): use a word to predict the surrounding ones in window.



CBOW



Skip-gram