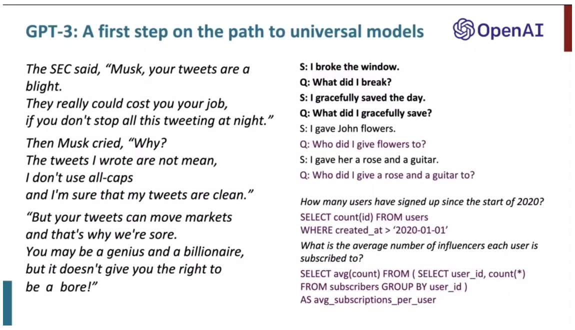
NLP Week 1

Lecture 1 : Intro & Word Vectors

2. Human language and word meanings



GPT-3

translate human language sentences into SQL

How do we represent the meaning of a word?

signifier (symbol) <=> signified (idea or thing)

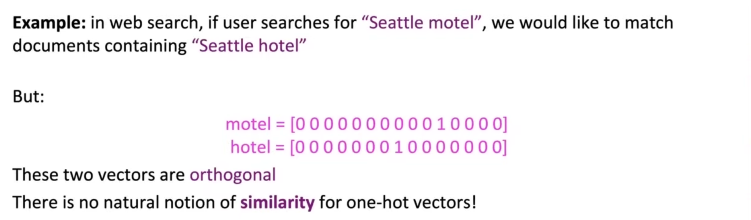
problems with resources like WordNet?

- missing nuance / missing new meanings of the words

Representing words as discrete symbols

one-hot vectors

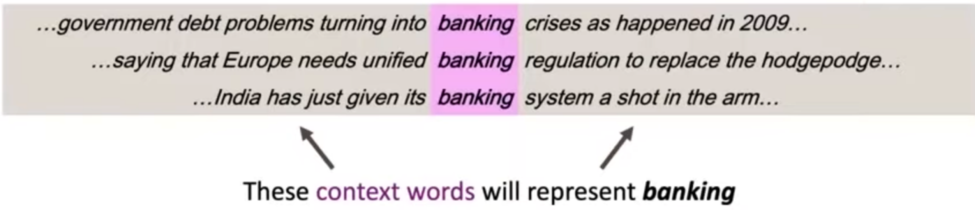
vector dimensions : number of words in voice



Solution : learn to encode similarity in the vectors themselves

Representing words by their context

Distributional sementics: A word's meaning is given by the words that frequently appear close-by context : set of the words that appear nearby



Word vectors

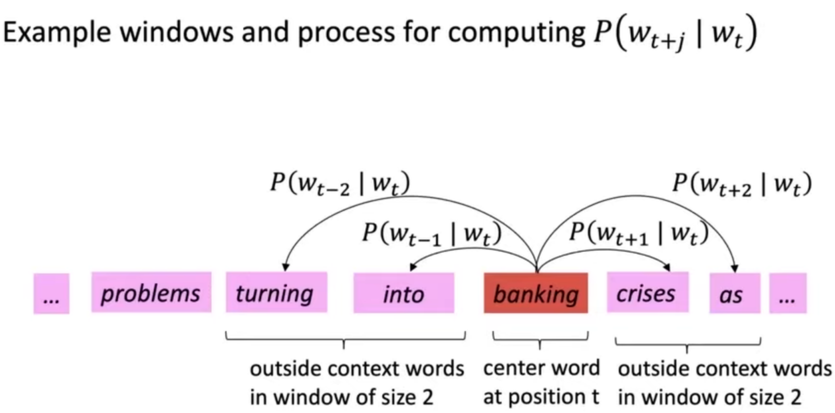
word vectors = word embeddings = word representations

>they are distributed representation

3. Word2vec : Overview

idea :

- Use the **similarity** of c and o to **calculate the probability of o given c**
- **keep adjusting** the word vectors to maximize this probability



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. Data likelihood:

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

θ is all variables to be optimized

sometimes called a *cost* or *loss* function

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

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

Minimizing objective function ⟺ Maximizing predictive accuracy

23

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

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

1. dot product compares similarity of o and c
2. exponentiation makes positive
3. normalize over entire voca to give probability distribution

This is an example of the **softmax function** $\mathbb{R}^n \rightarrow (0,1)^n$ Open region

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

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

But sort of a weird name because it returns a distribution!

To train the model : Optimize value of parameters to minimize loss

gradually adjust parameters to minimize a loss by walking down the gradient

$$P(o|c) = \frac{\exp(u_o^T \cdot v_c)}{\sum_{w=1}^{V} \exp(u_w^T \cdot v_c)}$$

$$\Big\downarrow \; \frac{d}{dv_c}\log$$

$$\underbrace{\frac{d}{dv_c}\log \exp(u_o^T \cdot u_c)}_{①} - \underbrace{\frac{d}{dv_c}\log \sum_{w=1}^{V} \exp(u_w^T \cdot v_c)}_{②}$$

① $\frac{d}{dv_c} u_o^T v_c = u_o$ $\qquad$ v_c : vector

② $\frac{d}{dv_c} \underbrace{\log}_{f} \underbrace{\sum_{w=1}^{V} \exp(u_w^T \cdot v_c)}_{z = g(v_c)}$ $\qquad$ $f(g(x))$

$$= \frac{1}{\sum_{w=1}^{V} \exp(u_w^T \cdot v_c)} \cdot \underbrace{\frac{d}{dv_c} \sum_{x=1}^{V} \underbrace{\exp}_{f}(\underbrace{u_x^T \cdot v_c}_{z=g(v_c)})} \longrightarrow \sum_{x=1}^{V} \exp(u_x^T \cdot v_c) \cdot \underbrace{\frac{d}{dv_c} u_x^T v_c}_{u_x}$$

$$\frac{d}{dv_c}\log P(o|c) = u_o - \frac{\sum_{x=1}^{V} \exp(u_x^T \cdot v_c)\, u_x}{\sum_{w=1}^{T} \exp(u_w^T \cdot v_c)}$$

$$= u_o - \sum_{x=1}^{V} \underbrace{\frac{\exp(u_x^T \cdot v_c)}{\sum_{w=1}^{T} \exp(u_w^T \cdot v_c)}}_{\text{softmax}} u_x \quad = u_o - \sum_{x=1}^{V} P(x|c)\, u_x$$

$$= u_o - \underbrace{\sum_{x=1}^{V} P(x|c)\, u_x} \quad \text{expectation}$$

$$= \text{observed} - \text{expected}$$

Gensim word vector visualization of various word vectors

In [7]:
```
! pip install gensim
```

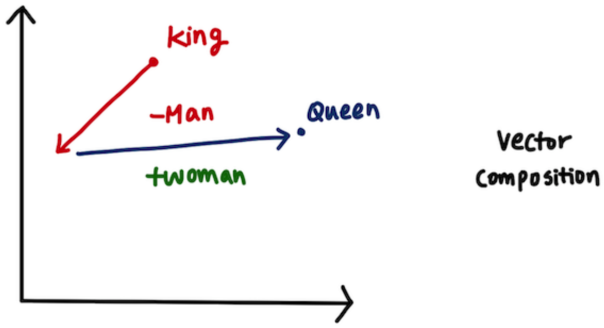
```
Requirement already satisfied: gensim in /Users/songhyejun/conda/envs/Hello/lib/python3.8/site-packages (4.2.0)
Requirement already satisfied: smart-open>=1.8.1 in /Users/songhyejun/conda/envs/Hello/lib/python3.8/site-packages
(from gensim) (6.0.0)
Requirement already satisfied: numpy>=1.17.0 in /Users/songhyejun/conda/envs/Hello/lib/python3.8/site-packages (fro
m gensim) (1.23.1)
Requirement already satisfied: scipy>=0.18.1 in /Users/songhyejun/conda/envs/Hello/lib/python3.8/site-packages (fro
m gensim) (1.8.1)
WARNING: There was an error checking the latest version of pip.
```

In [9]:
```
import numpy as np

%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('ggplot')

from sklearn.decomposition import PCA

import gensim.downloader as api
from gensim.models import KeyedVectors
```

In [11]:
```
model = api.load("glove-wiki-gigaword-100")
print(type(model))
```

```
[==================================================] 100.0% 128.1/128.1MB downloaded
<class 'gensim.models.keyedvectors.KeyedVectors'>
```

In [17]:
```
def analogy(x1,x2, y1):
    result = model.most_similar(positive=[y1,x2], negative=[x1])
    return result[0][0]
```



In [20]:
```
analogy('man', 'king', 'woman')
```

Out[20]: 'queen'