Lecture 1 - Intro & Word Vectors

Human Language

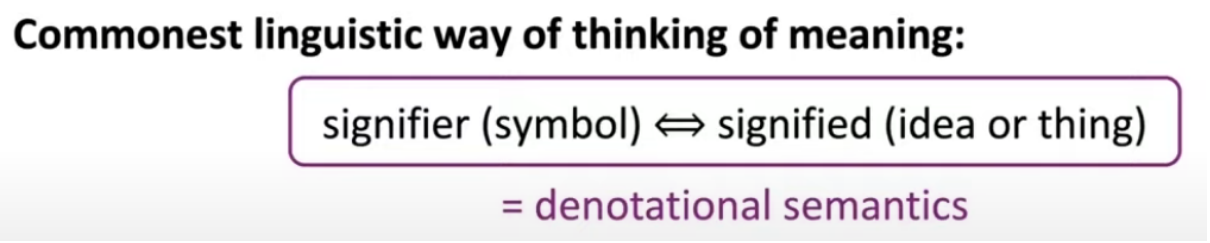
- A key question for AI and human-computer interaction ：how to get computers to be able to understand the information conveyed in human languages

GPT

- Predict following words

Meaning

- The ideanthat is represented by a word, phrase, etc
- The idea that a person wants to express by using words , signs, etc
- The idea that is expressed in a work of writing, art, etc



- Previously commonest NLP solution: Use, e.g., WordNet, a thesaurus containing lists of
  synonym sets and hypernyms (“is a” relationships)

e.g., synonym sets containing "good":

```
from nltk.corpus import wordnet as wn
poses = { 'n':'noun', 'v':'verb', 's':'adj (s)', 'a':'adj', 'r':'adv'}
for synset in wn.synsets("good"):
    print("{}: {}".format(poses[synset.pos()],
            ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g., hypernyms of "panda":

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

- Problems with resources like WordNet
  - Great as a resource but missing nuance
  - Missing new meanings of words
  - Subjective
  - Requires human labor to create and adapt
  - Can't compute accurate word similarity

- Representing words as discrete symbols
  - In traditional NLP, we regard words as discrete symbols
    - e.g., Hotel, conference, motel - a localist representation

      → Can be represented by one-hot vectors

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

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

→ Vector dimension = number of words in vocabulary

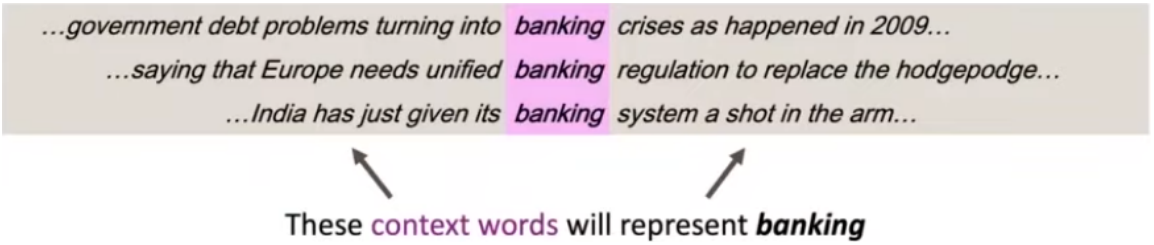
- Problem with words as discrete symbols

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

  - These two vectors are orthgonal → no natural notion of similarity

    → **Learn to encode similarity in the vecotrs themselves**

- Representing words by their context
  - Distributional semantics : A word's meaning is given by the words that frequently appear close-by
  - Context : the set of words that appear nearby (within a fixed-size window)



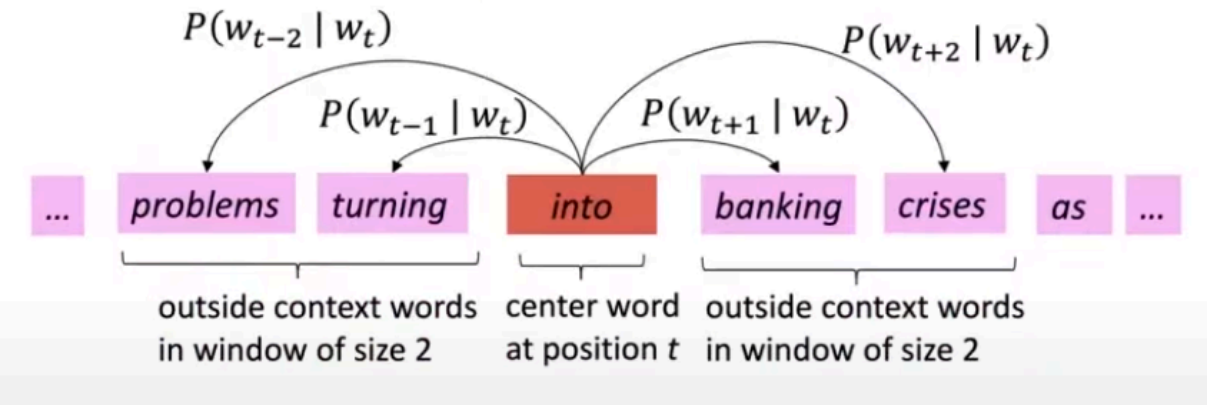
- Word vectors (Word embeddings, Word representations) ← distributed representation
  - Based on looking at the words that occur in context as vectors

    → build up dense real valued vector for each word

    → useful for predicting otehr words that occur in the context

- Word meaning as a neural word vector - visualization
  - Grouping similar words

- Word2vec : Overview
  - 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 centor 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
    - Keep adjusting the word vectors to maximize this probability



- Word2vec : objective & prediction function

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

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:

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

- Minimizing objective function ↔ Maximizing predictive accuracy
- Question : How to calculate $P(w_{t+j}|w_t; \theta)$?
- Answer : Use two vectors per words w:
  - v_w when w is a center word
  - u_w when w is a context word

② Exponentiation makes anything positive

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

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

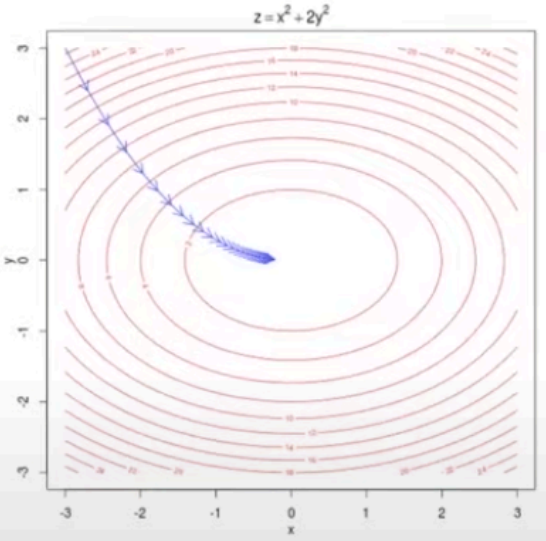
③ Normalize over entire vocabulary to give probability distribution

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

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

- To train the model : Optimize value of parameters to minimize loss
  - Gradually adjust parameters to minimize a loss
  - θ : Word vectors
  - Every word has two vectors (context & center)

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



cf) 왜 한 단어에 대하여 center vector와 context vector를 따로 사용할까?

- 표현법이 다를 수 있음