Lecture 03:
Word Representation

Overview

1. Word meaning (semantics)
2. Frequency based representation
3. Sparse vector representation
4. Dense vector representation (Distributional semantics)

Machine learning on text data!!!

- **Machine** learning algorithms work with numeric values and NLP **application** generate data in the form of text.
  - Text needs to be converted to real numbers.

- To solve NLP task with ML we need to represent text data in a way that ML algorithms or statistical techniques can understand.

- What should word representation capture?

What is meaning (semantics)?

Meaning in natural language is conveyed by words

- A words is an atomic symbol
- Meaning
  - Idea that is represented by a word, phrase, sentence, etc.
  - Idea expressed in a work of writing, speech or art, etc.

In linguistics a *meaning* maps *a word(s)* ⇔ *an idea*

Signifier (symbol) ⇔ *signified (idea or thing)*

Word representations

There are currently three main classes of word representation in NLP

1. Frequency based representation
   - Count Vector
   - TF-IDF Vector
   - Co-occurrence matrix (next class)

2. Sparse vector representation
   - one-hot vectors

3. Dense word representation
   - word2vec

Frequency based representation

- Frequency based representation are based on the concept of a bag-of-word.
- Bag-of-words is a collection of all the words that are present in the document along with their frequency.

"John likes to watch movies. Mary likes movies too."

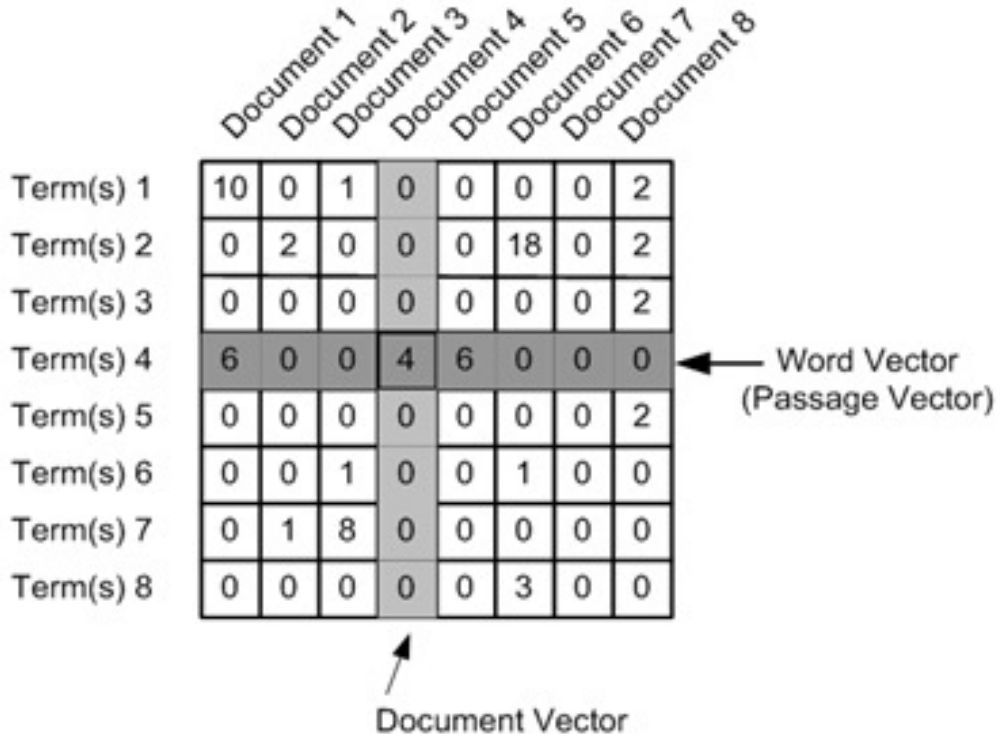
Bag-of-Words = {"john":1, "likes":2, "to":1, "watch":1, "movies":2, "mary":1, "too":1}

- These frequencies are fed into the machine learning algorithm for modelling the NLP task at hand.

Count vector matrix

Consider a corpus C of D documents $\{d_1, d_1, \ldots, d_D\}$ and a vocabulary of N unique tokens. The count vector is a DxN matrix M.

Each row i in M contains the frequency of tokens in document $D^{(i)}$



Count vector

- Let us understand this using a simple example.
  - D1: *"He is a hard working boy. She also work hard"*.
  - D2: *"John can work hard"*.
  - Vocabulary = *{"He", "She", "hard", "boy", "John", "person", "work"}*
  - Here *D=2, N=10*
- The count matrix *M* of size 2 X 10 will be represented as:

	also	boy	can	hard	he	is	john	she	work	working
D1	1	1	0	2	1	1	0	1	1	1
D2	0	0	1	1	0	0	1	0	1	0

Term Frequency Inverse Document Frequency (TFIDF)

- Calculates weighting of words in a document
- Weighting is in relation to the total number of documents dealing with the same subject.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

- Main idea:
  - Penalize very frequent words
  - Boost influence of rare words

Term Frequency Inverse Document Frequency (TFIDF)

TF-IDF

- replaces the raw counts in Count vector matrix by weighted metric

	also	boy	can	hard	he	is	john	she	work	working
D1	0.342369	0.342369	0.000000	0.487197	0.342369	0.342369	0.000000	0.342369	0.243598	0.342369
D2	0.000000	0.000000	0.576152	0.409937	0.000000	0.000000	0.576152	0.000000	0.409937	0.000000

- Advantage over raw counts is that rare words are not penalized

Limitations of Frequency-based representation

- Ignore the meaning or semantics of the word
- Ignores word context
- Does not capture co-occurrences of words in different documents
- Loses the ordering of the words
  - *“My name is John”* is the same as *“Is my name John”*

Sparse word representation – One-Hot vector

- We can look at one-hot encoding like the same concept of categorical encoding of non-numeric features in general machine learning.
- Categorical encoding is a localist representation
- Words are one hot encoded

long = [0 0 1 0 0 0 0 0 0 0]

length = [0 0 0 0 0 0 1 0 0 0]

car = [0 0 0 0 0 0 0 0 1 0]

- Language is infinite (derivational morphology) hence vector becomes too large *(e.g. 500,000)*

Sparse word representation – One-Hot vector

- For all the words present in the vocabulary create a vector such that 1 index represents the word, and the rest all are zeros.
- **One-hot vectors** of dimension $|V|\ x\ 1,\ |V|$ is vocabulary size

$$long = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad lenght = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \text{car} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Limitations of sparse vector representation

Coupled with the limitations seen with frequency-based spare vector reprsentation are

- computationally expensive because one-hot vectors are very high-dimensional and **sparse vectors**
- have no concept of similarity each word vector is orthogonal to all others
  - *long/length* are as different as *long/car*

$$v_{long}^{T} \cdot v_{length} = v_{long}^{T} \cdot v_{car} = 0$$

What do words mean?

Lexicographic meaning of a word

What is the context of bar ….?

- is *bar* it a place to have a drink?
- is *bar* a long rod?
- is *bar* to block it?

What is the context of bank ….?

- is *bank* a financial institution?
- is *bank* a river bank?

Vacabulary is like a bag-of-words

- *how can we capture the meaning of each word from a vocabulary?*

How do we capture semantics (word meaning)?

- Natural language understanding requires knowing when words have similar meanings

- Consider the following question and answering sessions:
  - Question: *"How long is the Senegal River?"*
  - Answer: *"The official length of Senegal River is 1086km"*

- In this context ***"long"*** is similar to ***"length"***.
  - in other context *long* may refer to *time*

How do we capture semantics (word meaning)?

Distributional semantics:

- an approach to captures the meaning of words based on large amounts of raw text
  - Learn meaning of words from the context in which they are used.
  - Create vector representation of words that can capture semantics.
    - Map words to vectors that capture corpus statistics
    - use neural network to estimate vectors

Distributional semantics

Distributional Hypothesis

- Zellig Harris (1954):
  - *"oculist and eye-doctor … occur in almost the same environments"*
  - *"If A and B have almost identical environments we say that they are synonyms."*

- John R. Firth 1957:
  - *You shall know a word by the company it keeps.*

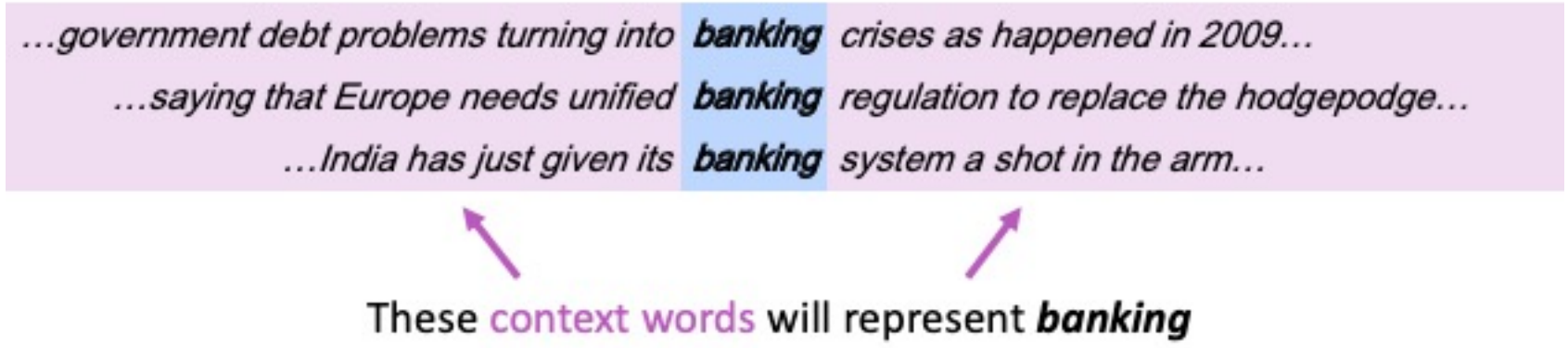
The ***contexts*** in which a word appears tells us a lot about what it means.

- *Words that appear in similar contexts havesimilar meanings*

Distributional semantics

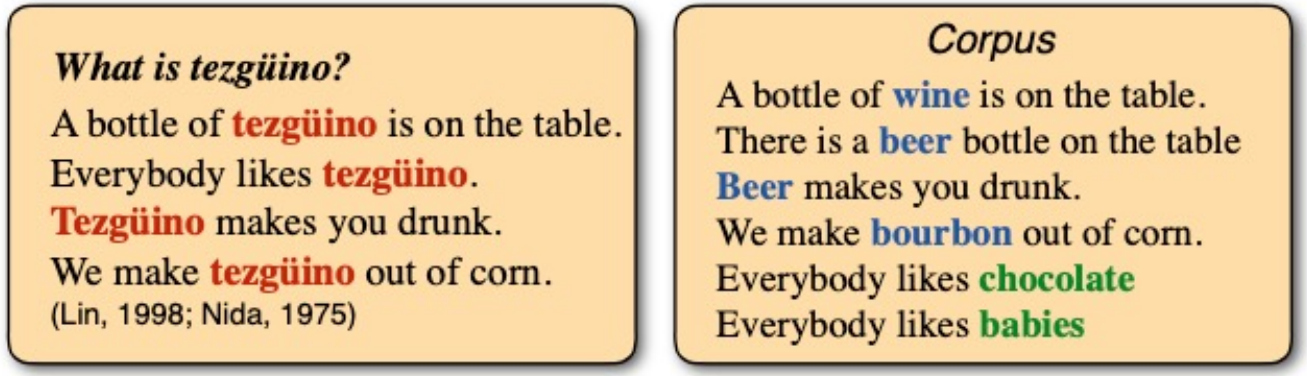
A word's meaning is given by the words that frequenctly appear close-by.

- When a word *w* appears in text, its context is the set of words that appear within a fixed window
- Use the many context of *w* to build up a representation of *w*



Distributional semantics

You shall know a word by the company it keeps?



Can we *automatically* identify that *tezgüino* is like beer?

- A large corpus may contain sentences such as:
  - ***Beer*** *makes you drunk*
- But there are also red herrings:
  - *Everybody likes* ***chocolate***
  - *Everybody likes* ***babies***

What is a context?

- Contexts defined by nearby **words**:
  - How often does word *w (tezgüino)* appear near the word *drink*?
    - Near => *"drink"* appears within a window of *±k* words of *w"*, or
    - *"drink"* appears in the same document/sentence as *w"*
      - This yields fairly broad thematic similarities.

- Contexts defined by **grammatical relations**:
  - How often is (the noun) *w* used as the subject (object) of the verb *drink*?
  - This gives more fine-grained similarities.

    word sense disambiguity: Use the context of a *particular occurrence* of a word to identify which sense it has.
      - Assumption: If a word has multiple distinct senses
        - *plant*: *factory* or *green plant*, each sense will appear in different contexts.

Distributional semantics – context

Suppose we have a sliding window of size 5 moving along a sentence with the word in the middle is the target

"The man who passes the sentence should swing the sword." – Ned Stark

Sliding window (size = 5)	Target word	Context
[The man who]	the	man, who
[The man who passes]	man	the, who, passes
[The man who passes the]	who	the, man, passes, the
[man who passes the sentence]	passes	man, who, the, sentence
...	...	...
[sentence should swing the sword]	swing	sentence, should, the, sword
[should swing the sword]	the	should, swing, sword
[swing the sword]	sword	swing, the

"swing", "sentence"

"swing", "should"

"swing", "the"

"swing", "sword"

Distributional semantics

Distributional semantics: Word embeddings — dense vectors

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

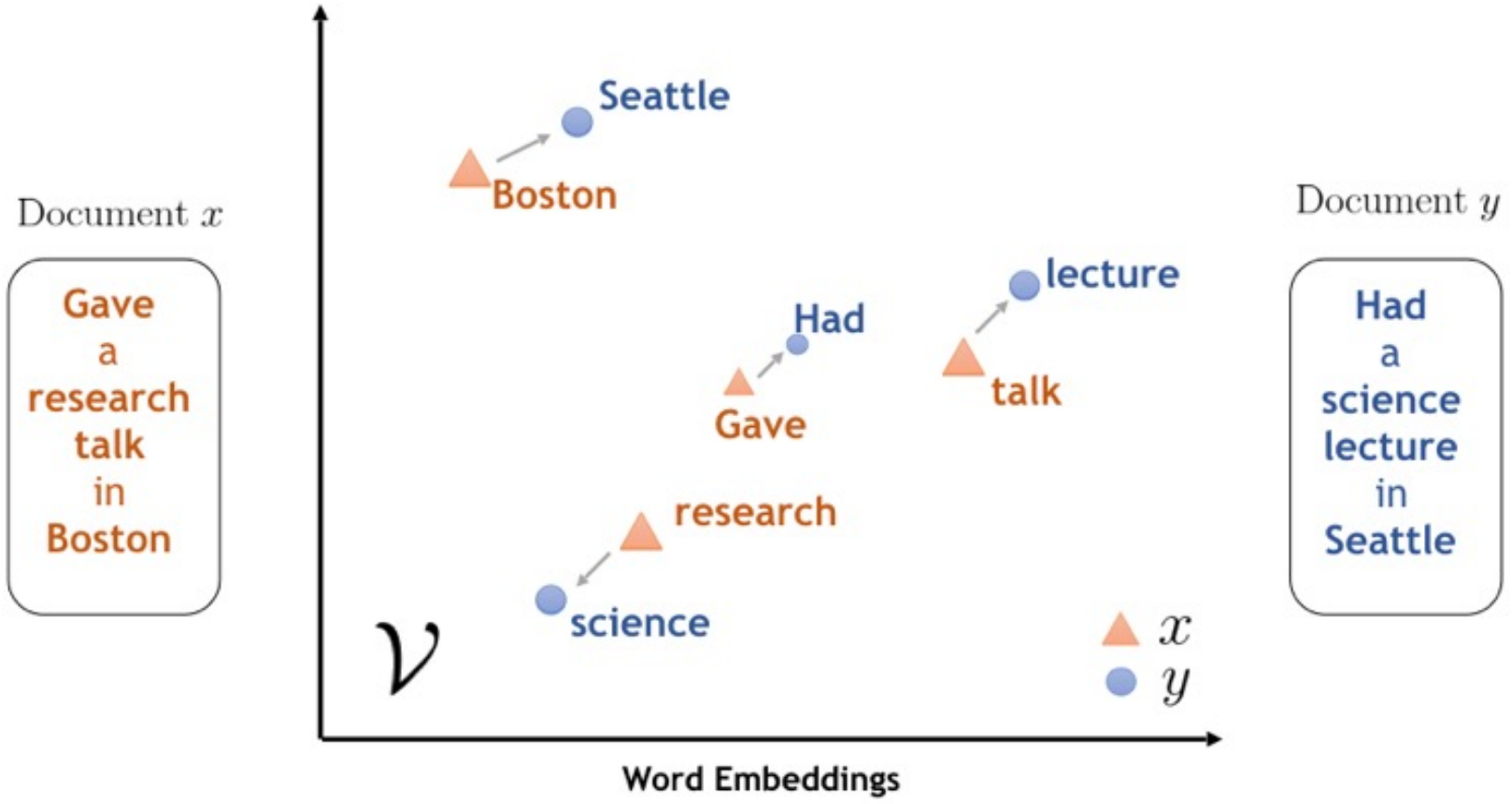
Given previous example:

- *Question: "How long is the Senegal River?"*
- *Answer: "The official length of Senegal River is 1086km"*

- We want to capture concept of similarity such that *long/length* is greater than *long/car*

$$v_{long}^{T} \cdot v_{length} \gg v_{long}^{T} \cdot v_{car}$$

Distributional semantics



https://www.ibm.com/blogs/research/2018/11/word-movers-embedding/

Distributional Similarities?

- Semantic similarity between words is measured in terms of the similarity of the contexts in which they appear by
  - Represent words as vectors such that
    - each vector element (dimension) corresponds to a different context.
    - the vector for any particular word captures how strongly it is associated with each context.

- Compute the semantic similarity of words as the similarity of their vectors.
  - dot product
  - Cosine similarity

Distributional semantics – word2vec

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

- **Main 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 centre 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

- Train a binary classifier to predict which words appear in the context of a *target word*.
  - Word in the context of target words are the **positive** examples
  - Words not in context are **negative** examples

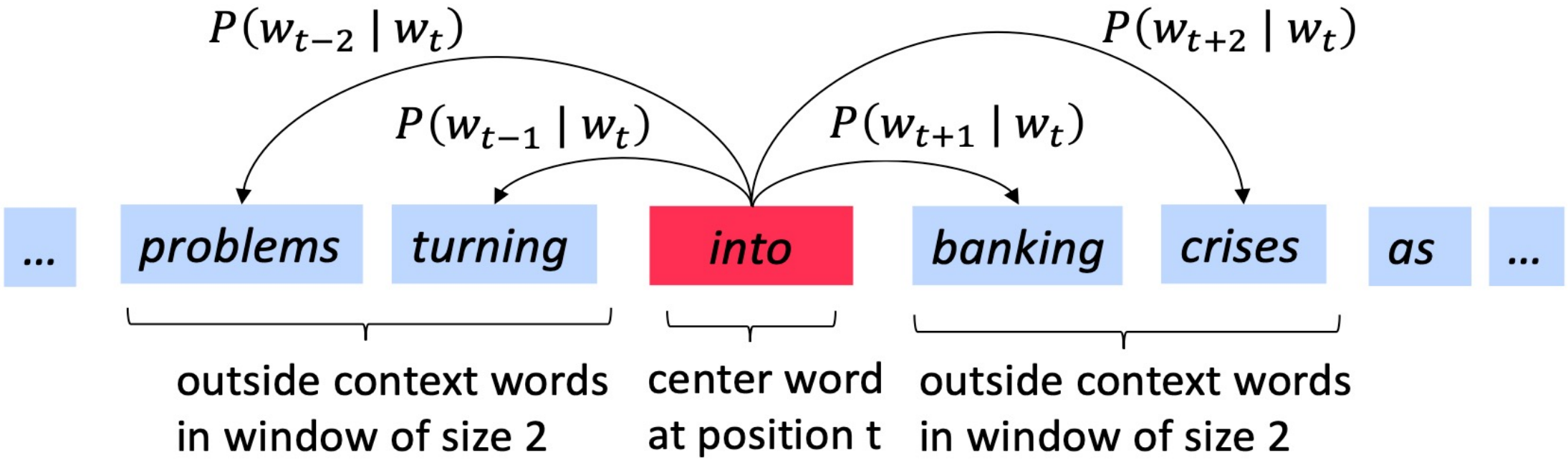
$$w_t = \theta X + e_t$$

- During training we assume that
  - target and context vectors of positive examples will become similar
  - those of negative examples will become dissimilar.

- Weight or paramter θ of the classifier are the dense vector representation.

Word2vec overview

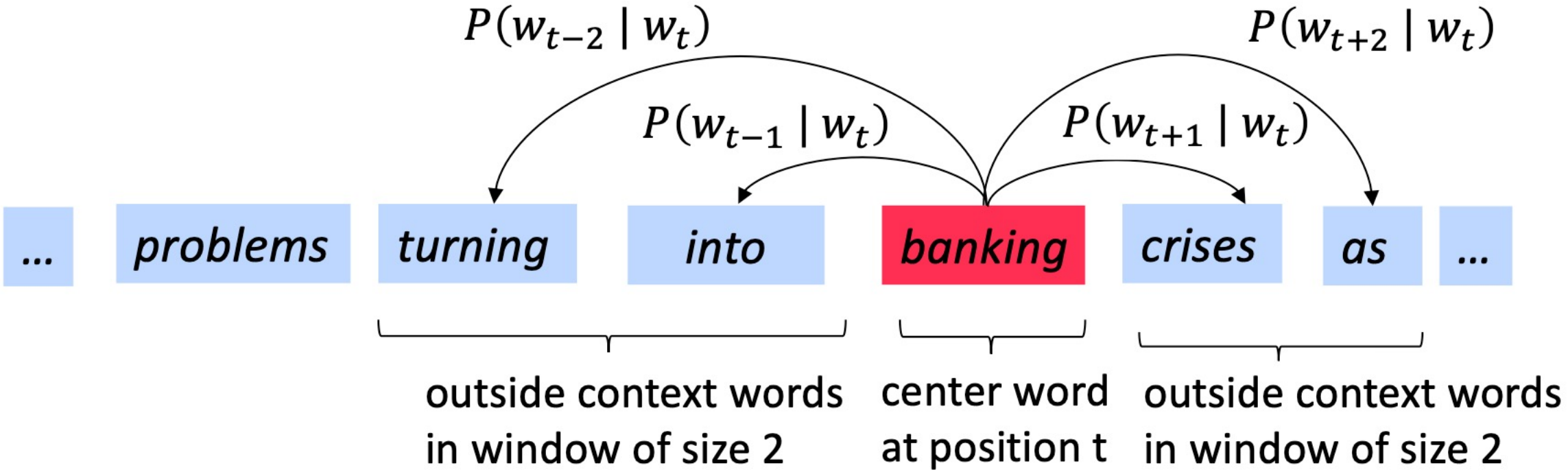
- Example: consider the word *into* below at position *t* within a window of *j* words we need to compute $P(w_{t+j}|w_t)$ for all context words



Based on Chris Manning

Word2vec overview

- Example: when center word is *banking*



Based on Chris Manning

Word2vec – Objective function

For each position $t = 1, \ldots, T$, predict the context words within a window of fixed size m, given the centre word w_t

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 *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

Word2vec – Objective function

- To minimize the objective function $J(\theta)$ we need to calculate

$$P(w_{t+j}|w_t;\theta)$$

- How do we estimate $P(w_{t+j}|w_t;\theta)$
  - To do this we define 2 vectors per word *w*:
    - $v_w \in V$ when *w* is a center word
    - $u_w \in U$ when *w* is a context word
  - Then for a center word *c* and a context word *o* and then calculate

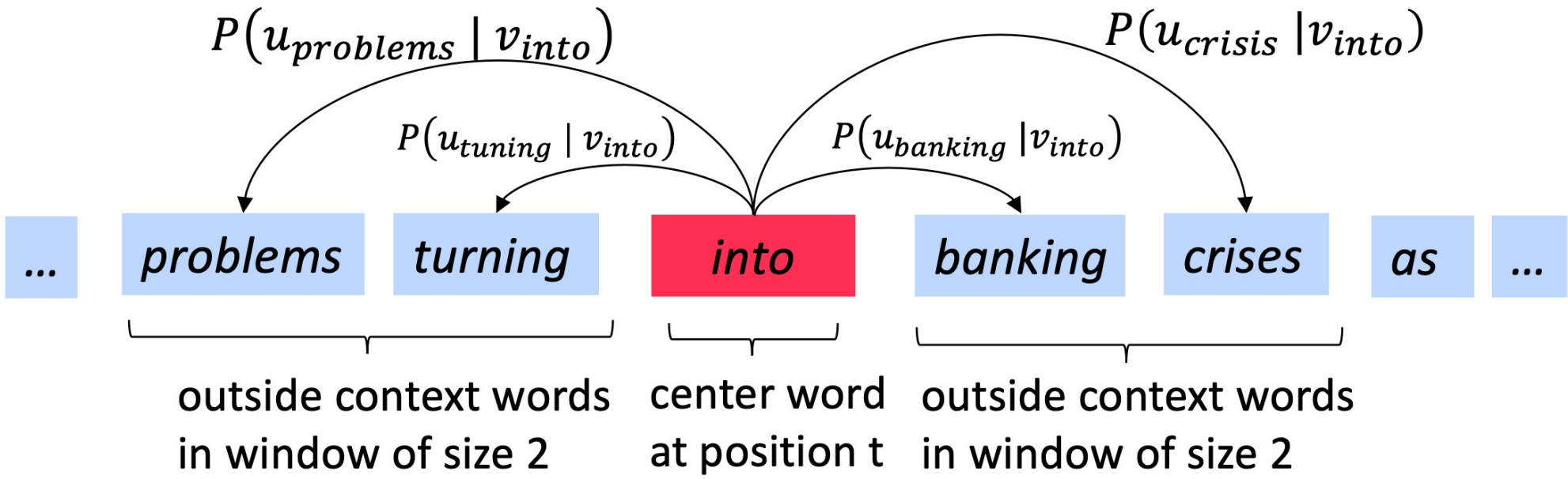
$$\boldsymbol{P(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w \in V} exp(u_w^T v_c)}}$$

Note: *the only parameters θ that the model has is the vector representation of the words*

Distributional semantics – Objective function

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

$$P(u_{problems}|v_{into}) \Leftrightarrow P(problems|into; u_{problems}, v_{into}, \theta)$$



Based on Chris Manning

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*.

$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$

Larger dot product = larger probability

Normalize over entire vocabulary to give probability distribution

- SoftMax maps arbitrary values x_i to a probability distribution p_i

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

Based on Chris Manning

Reading materials

1. Efficient Estimation of Word Representations in Vector Space
2. Distributed Representations of Words and Phrases and their Compositionality
3. GloVe: Global Vectors for Word Representation
4. Evaluation methods for unsupervised word embeddings