Word Embeddings

Debapriyo Majumdar

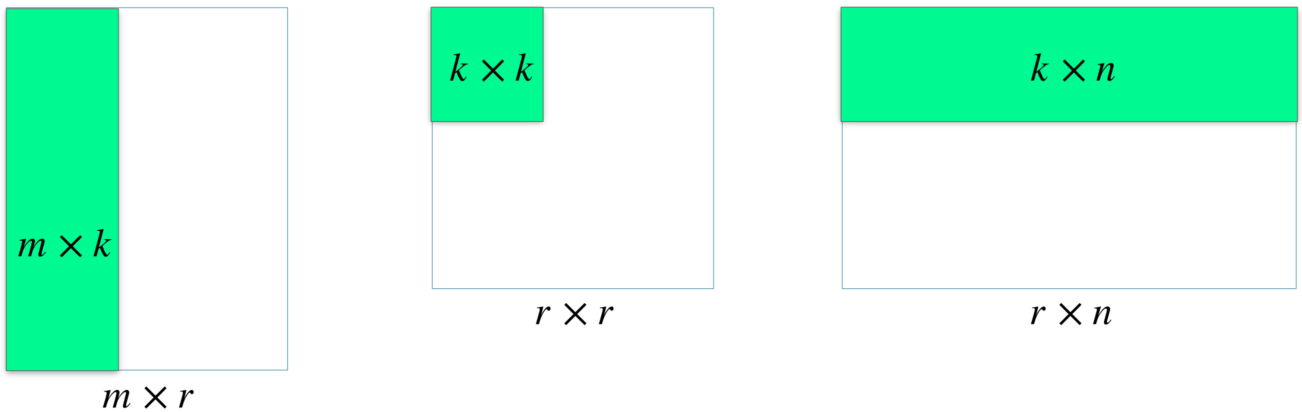
debapriyo@isical.ac.in

Word Embeddings

- In general, vector representation of words (not new)
- Simplest: one-hot (#of dimensions = size of the vocabulary)
  - Each term is totally orthogonal to another
- The row-vectors corresponding to terms in a term-document matrix
  - Similar terms do co-occur more, some relationships captured
- Towards capturing semantics: rows of matrix U_k containing the first k singular vectors of the term-document matrix ($\sim$ post 1990s)
  - Same as the above, arguably better captured in reduced dimension, noise discarded
- Vector representation learned from the corpus
  - Neural language model (2003)
  - word2vec (2013)
  - GloVe
  - FastText
  - ELMO

Recall: singular value decomposition

$$A = U\Sigma V^T$$

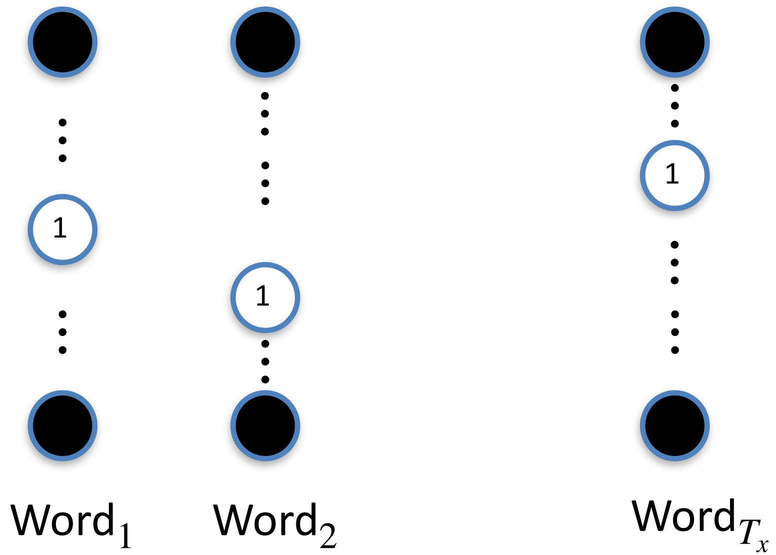


U_k : the matrix with the first k columns of U (first k singular vectors)

Each row of U corresponds to a term

The rows of the matrix U_k represent dense k-dimensional vector representation of the terms

Word embeddings



One-hot encoding

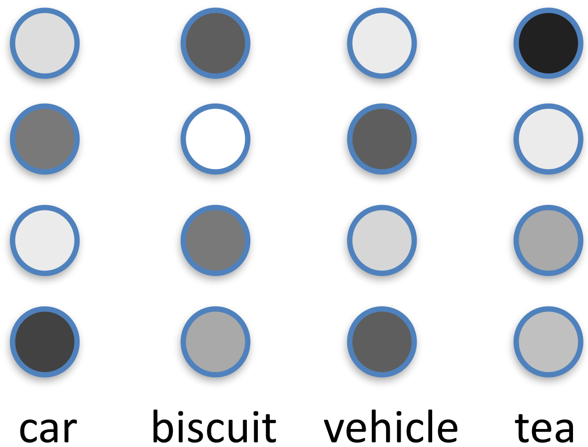
Each word is a vector with all zeros except one

Orthogonal to each other

But words have meanings

Some words are similar to each other

White = 1; Black = 0, Grey $\in (0,1)$.



Better idea

More compact vector representation

retaining semantics of words

Similar words $\Longrightarrow$ similar vectors

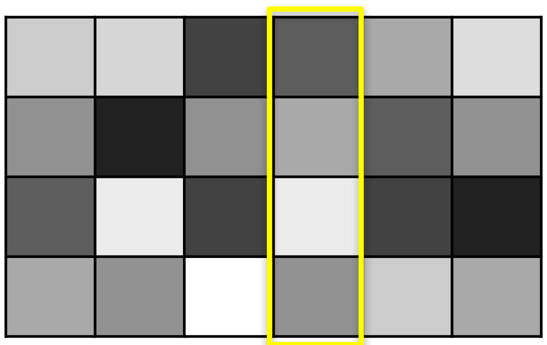
Learn word embeddings

The embedding matrix

White = 1; Black = 0, Grey $\in (0,1)$.

$$e_i = Eo_i$$

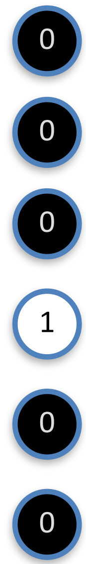
E



$d \times N$

Embedding matrix for d dimensional embedding

o_i

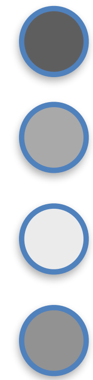


$N \times 1$

One-hot vector for the i-th word in the vocabulary

$=$

e_i



$d \times 1$

Embedding for the i-th word in the vocabulary

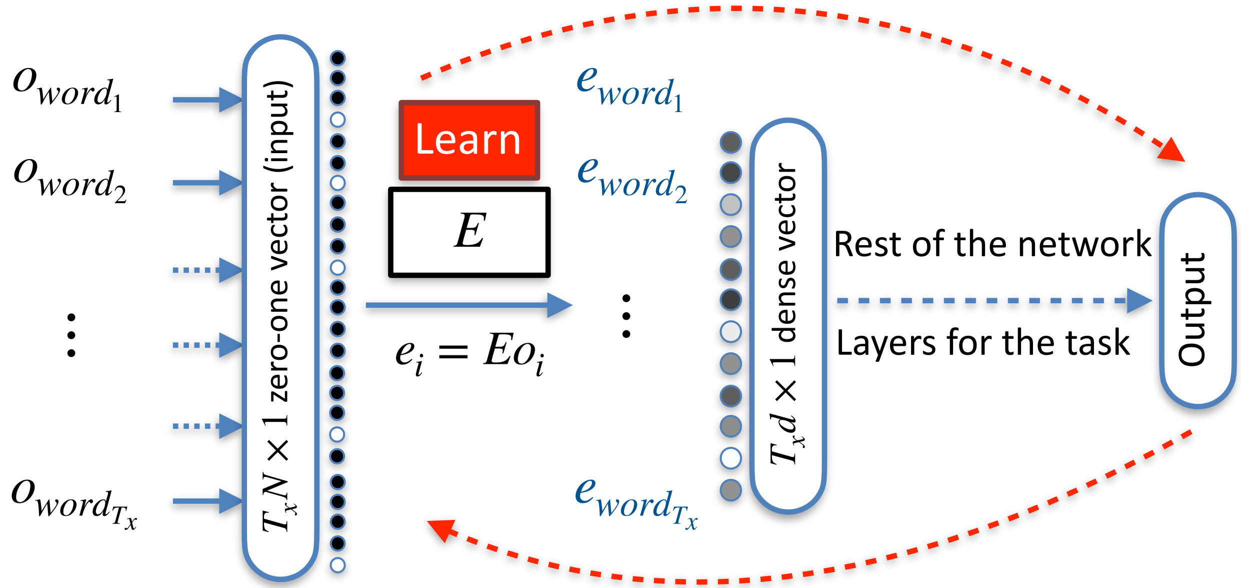
Learning word embeddings

White = 1; Black = 0, Grey $\in (0,1)$.

Add an embedding layer in the beginning

```
inputs = keras.Input(shape=(None,), dtype="int32")

# Embed each integer in a d-dimensional vector
x = layers.Embedding(N, d)(inputs)
```



The embedding can be learnt as a byproduct of any task for which the model is trained

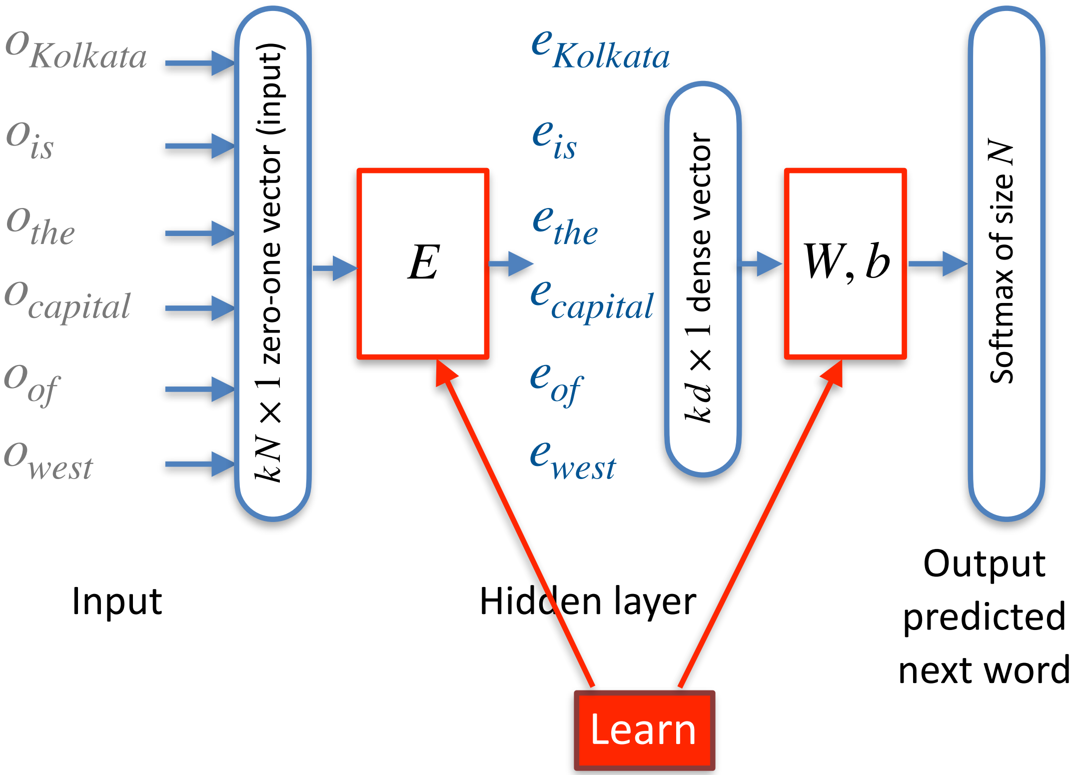
Input as one-hot vectors

This is the general framework following Neural LM (2003)

Neural Language Model (2003)

- Problem: build a language model
  - Given k words in a sentence, predict the next word
- Training data: sequences of $k + 1$ words in the corpus
- Parameters: embedding matrix E and weights for the language model
- Learns E in the process
- Results: almost as good as the state of the art at that time, but the idea is carried forward to research later

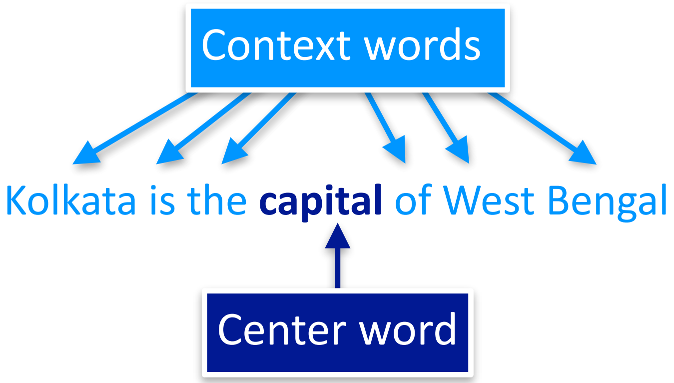
Kolkata is the capital of West ______



Word2Vec (2013)

- Goal: semantically similar words $\Longrightarrow$ similar vectors
- Similarity measure: dot product
- Approach: Learn association between **words** and context from available text
- Skip-gram model: fix a window size k, each word is associated with other words (context) within a window k
- Dot product between the **embedding** of a **context** word and that of the **center** word must be ***high***
- Learn the word embeddings by training a language model

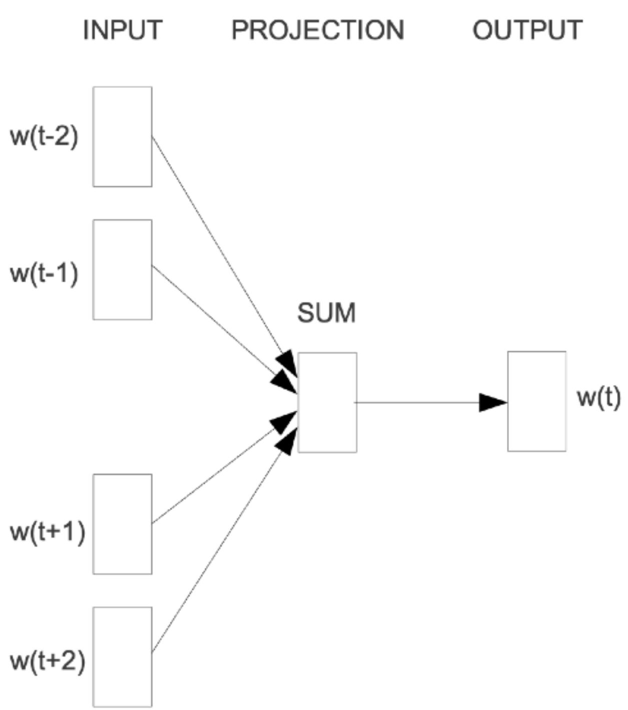
Example



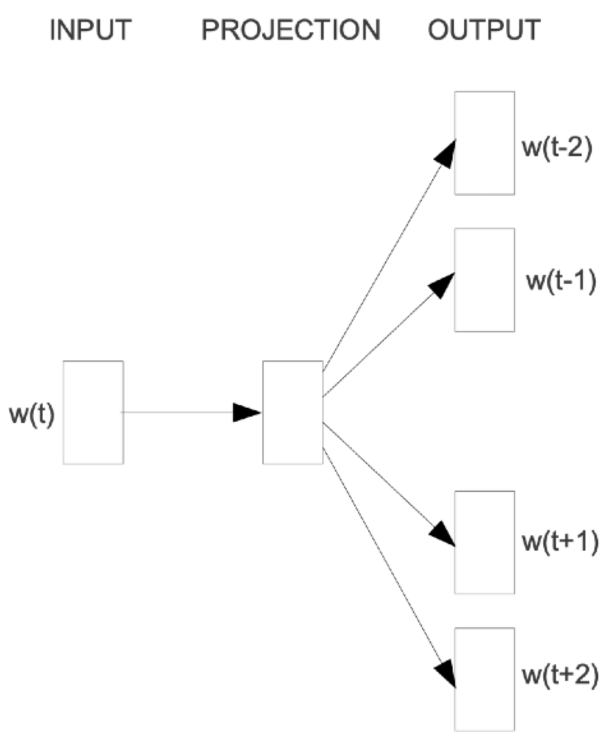
Here, the window size $k = 3$

CBOW and Skip-gram models

CBOW
Given context, predict the center word



Skip-gram
Given center word, predict assoiated words



Word2Vec (2013): Skip Gram Model

- Goal: given the center word o, output probabilities of context words c
- capital → Kolkata, capital → is, … capital → Bengal, …
- Two representations for each word w (e_w if it is the center word, θ_w if it is a context word)
- Challenge 1: for large N, training the softmax with size N is computationally impractical
- Challenge 2: too many associations with the stopwords or very frequent words

Kolkata is the **capital** of West Bengal

Output probability of context words w_{t+j} given **center** word w_t:

$$P(w_{t+j} \mid w_t) = P(\text{Kolkata} \mid \text{capital})$$

for $0 < |j| \leq k$ (a fixed context size)

$$\text{Softmax: } P(o \mid c) = \frac{e^{\theta_o^T e_c}}{\sum_{w \in V} e^{\theta_w^T e_c}}$$

Train parameters $\theta = (\theta_w)_{w \in V}$ and $E = (e_w)_{w \in V}$

Word2Vec: Negative Sampling

Kolkata is the **capital** of West Bengal

capital	—	Kolkata	1
capital	—	apple	0
capital	—	in	0
capital	—	information	0
capital	—	kill	0
capital	—	the	0

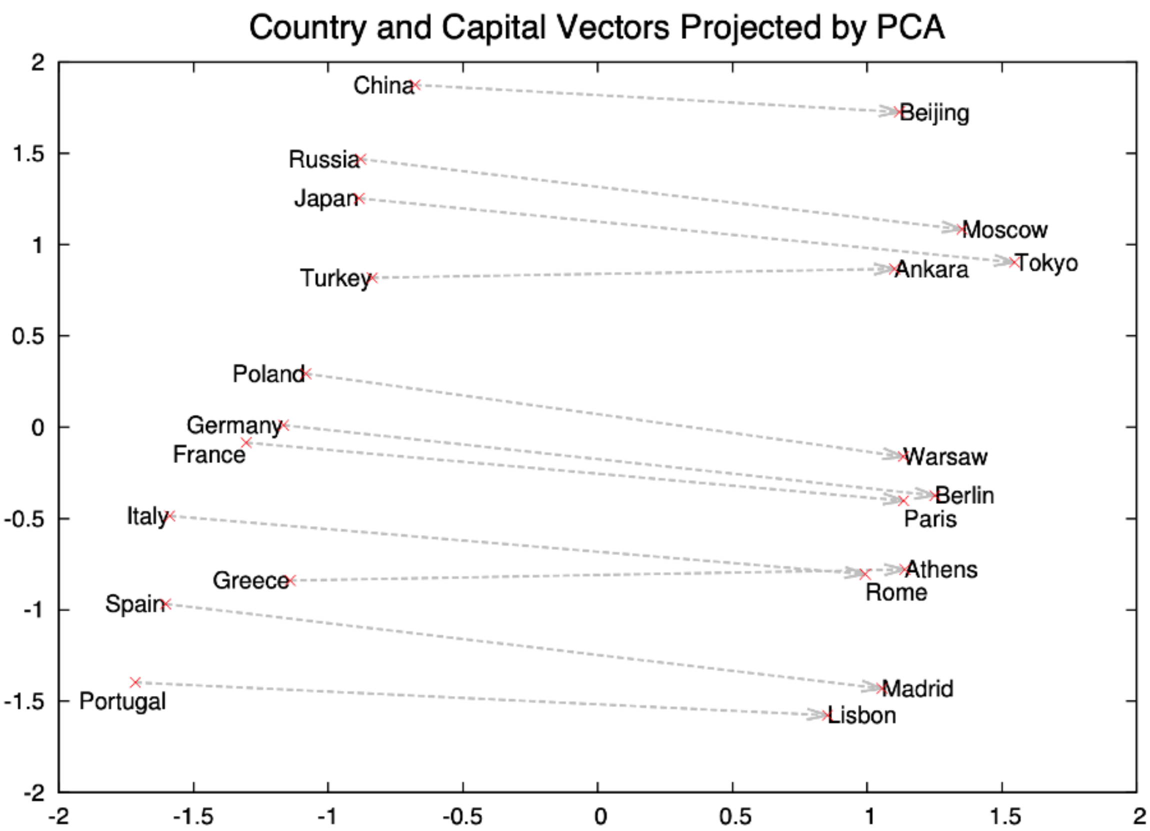
Behaves like many *binary classification* problems instead of a huge softmax

- Prepare training data comprising of positive and negative examples
- For every positive example (from actual sentences in the corpus), sample k negative examples
- What is a negative example?
  - Any word from the vocabulary
  - Assumption: a random word is unlikely to be associated with the target word
- For small dataset, k between 5 and 20, for large dataset, k between 2 and 5
- In each iteration, train only a few of the *binary classifiers*
- Sampling strategy: sample negative word w with

  probability $P(w) = \dfrac{f(w)^{3/4}}{\sum_{v \in V} f(v)^{3/4}}$ where $f(w)$ is the

  frequency of w in the corpus (empirical heuristic)

Word2Vec



Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Source: https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

GloVe

Intuitive ideal association

	solid	gas	water	fashion
ice	Yes	No	Yes	No
steam	No	Yes	Yes	No

Co-occurrence probabilities

w	solid	gas	water	fashion
P(w|ice)	0.00019	0.00007	0.0030	0.00002
P(w|steam)	0.00002	0.00080	0.0022	0.00002
P(w|ice)/ P(w|steam)	8.9	0.085	1.36	0.96

- Learn word vectors from *ratio of probabilities* instead of *probabilities*
- Notation:

  X : word-word co-occurrence matrix

  X_{ij} : #of times word j appears in the context of word i

  $X_i = \sum_j X_{ij}$: #of times any word appears in the context of word i

  $P_{ij} = P(j\,|\,i) = X_{ij}/X_i$: probability that word j appears in the context of word i

  θ_i, e_i : embeddings (we want to learn) as before

GloVe

Reference: nlp.stanford.edu/pubs/glove.pdf

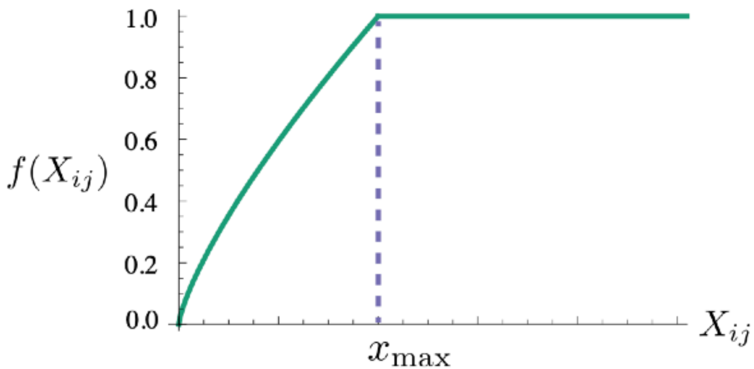
w	solid	gas	water	fashion
P(w|ice)	0.00019	0.00007	0.0030	0.00002
P(w|steam)	0.00002	0.0008	0.0022	0.00002
P(w|ice)/ P(w|steam)	8.9	0.085	1.36	0.96

- Try to model the *ratio of the probabilities* by some function $\frac{P_{ik}}{P_{jk}} = F(\theta_i, \theta_j, e_k)$
- Assumption: $\frac{P_{ik}}{P_{jk}} = F((\theta_i - \theta_j)^T e_k)$
- The co-occurrence is symmetric, so we should be able to exchange $\theta_i \leftrightarrow e_i$ and $X \leftrightarrow X^T$
- Require F to be a homomorphism between the groups $(\mathbb{R}, +)$ and $(\mathbb{R}_{>0}, \times)$
- $F((\theta_i - \theta_j)^T e_k) = P_{ik} / P_{jk} = F(\theta_i^T e_k) / F(\theta_j^T e_k)$
- This happens if: $\theta_i^T e_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i)$

GloVe

Minimize: $$J = \sum_{i,j=1}^{N} f(X_{ij})\left(\theta_i^T e_j + b_i + b_j' - \log X_{ij}\right)^2$$

- Note: b_i and b_j' are bias terms
- If $X_{ij} = 0$ (the words never co-occur), $\log X_{ij}$ would be undefined
- Have a weight function $f(X_{ij})$ which is 0 at 0 (then, assume $0 \log 0 = 0$)



References

- Andrew Ng's lectures on *Sequence Models*: www.coursera.org/learn/nlp-sequence-models
- Chris Manning, Abigail See and other TAs. *Natural Language Processing with Deep Learning.* Stanford University Course (CS224n), Winter 2019. web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. www.deeplearningbook.org
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).
- Pennington, J., Socher, R. and Manning, C. D. GloVe: Global Vectors for Word Representation. 2014.