# NLP Applications

Week 5: Document Clustering

fit@hcmus
KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

- Introduction
- Word clustering
- Sentence clustering
- Document clustering
- Evaluation

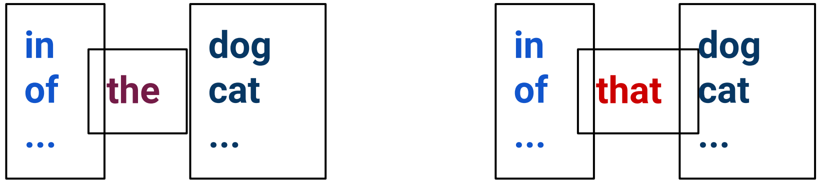
NLP Applications – Document Clustering

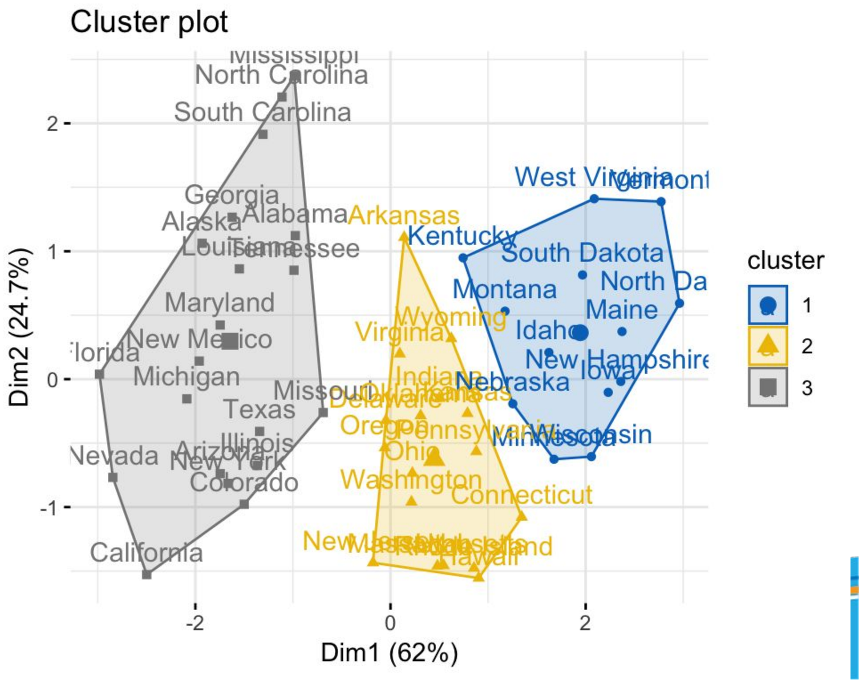
# Word Clustering

fit@hcmus

# Word Clustering

- Introduction:
  - Words that are close in meaning will occur in similar contexts
  - More clearly: Words with similar distributions of nearest terms have similar meanings.



# Word Clustering

- Purposes:
  - Study the internal structure of words
  - Cluster words into groups
- Advantages:
  - Reduce word sparsity
  - Reduce training data size

# Word Clustering

- Given that:
  - $\mathcal{V}$ is the set of terms in the corpus $w_1, w_2, \ldots, w_T$
  - $\mathcal{C}$: $\mathcal{V}$ -> {1, 2, …, *k*} is categorizing vocabulary of terms into k clusters
- Model:

$$p(w_1, w_2, \ldots w_n) = \prod_{i=1} e(w_i|C(w_i))q(C(w_i)|C(w_{i-1}))$$

<u>*Note*</u>: $C(w_0)$ is a special state

# Word Clustering

- Example:

$$p(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} e(w_i|C(w_i))q(C(w_i)|C(w_{i-1}))$$

$$C(\text{the}) = 1, \quad C(\text{dog}) = C(\text{cat}) = 2, \quad C(\text{saw}) = 3$$

$$e(\text{the}|1) = 1, \quad e(\text{cat}|2) = e(\text{dog}|2) = 0.5, \quad e(\text{saw}|3) = 1$$

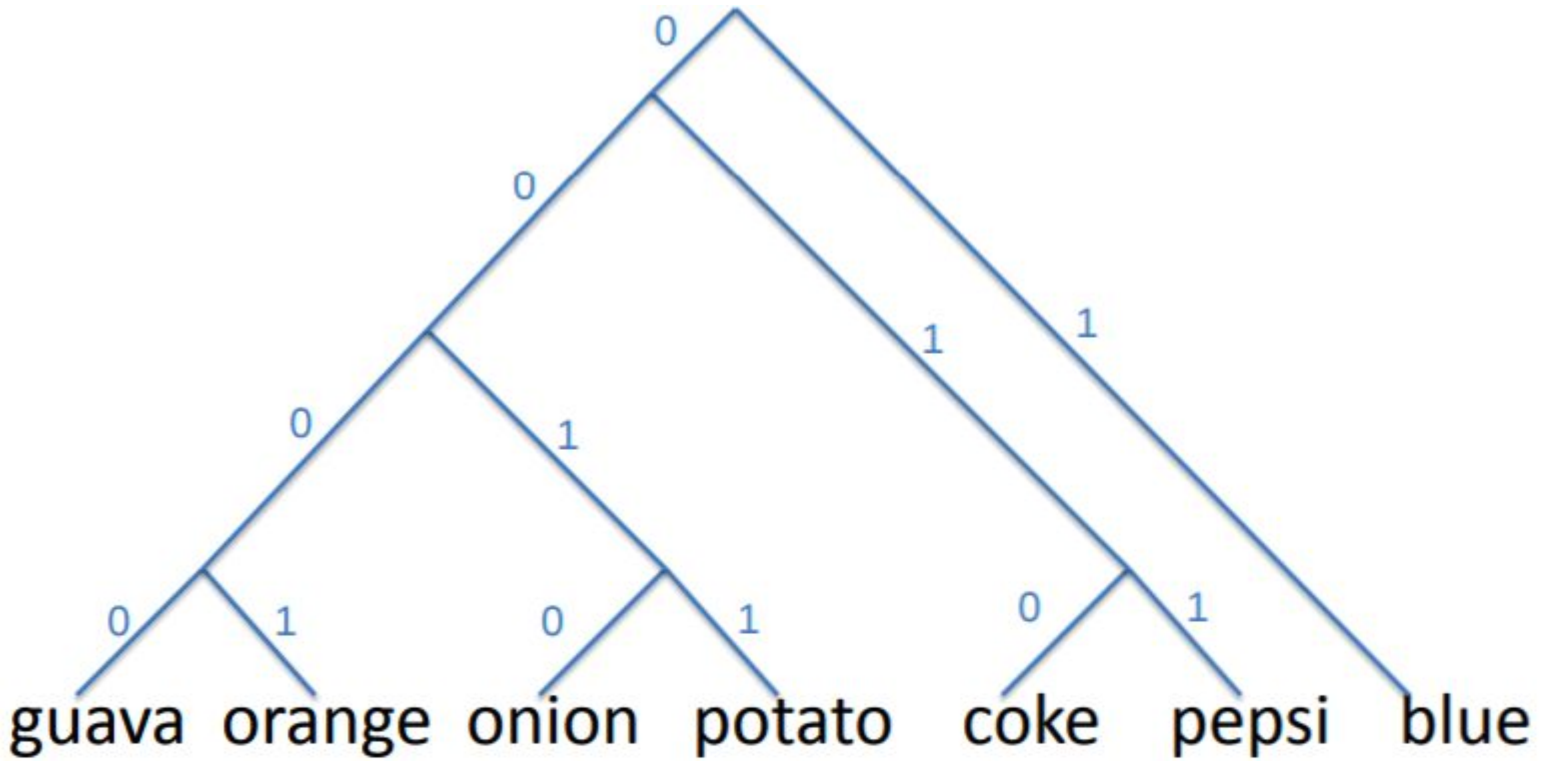
$$q(1|0) = 0.2, \quad q(2|1) = 0.4, \quad q(3|2) = 0.3, \quad q(1|3) = 0.6$$

***p*(the dog saw the cat) =**
**q(1|0)*q(2|1)*q(3|2)*q(1|3)*q(2|1)*e(the|1)*e(dog|2)*e(saw|3)*e(the|1)*e(cat|2)**

fit@hcmus

# Word Clustering

- Brown clustering: each word has a binary code.




- onion: 0010

# Word Clustering

- Brown clustering: each word has a binary code.

- *111111110110000* *slapped*
- 111111110110000 shattered
- 111111110110000 commissioned
- 111111110110000 drafted
- 111111110110000 authorized
- 111111110110000 authorised
- 111111110110000 imposed
- 111111110110000 established
- 111111110110000 developed

- *111111111100110* *officer*
- 111111111100110 acquaintance
- 111111111100110 policymaker
- 111111111100110 instructor
- 111111111100110 investigator
- 111111111100110 advisor
- 111111111100110 aide
- 111111111100110 expert
- 111111111100110 adviser

# Word Clustering

- Brown clustering (Brown, 1992):
  - $\mathcal{V}$ is the set of terms in the corpus $w_1, w_2, \ldots, w_T$
  - $\mathcal{C}$: $\mathcal{V}$ -> $\{1, 2, \ldots, k\}$ is categorizing vocabulary of terms into k clusters
  - Parameter e(v|c) for each v $\in \mathcal{V}$, c $\in \{1, 2, \ldots, k\}$
  - Parameter q(c'|c) for each c', c $\in \{1, 2, \ldots, k\}$

# Word Clustering

- Evaluate the quality of $\mathcal{C}$:

$$\begin{aligned}\text{Quality}(C) &= \sum_{i=1}^{n} \log e(w_i|C(w_i))q(C(w_i)|C(w_{i-1})) \\ &= \sum_{c=1}^{k}\sum_{c'=1}^{k} p(c,c')\log\frac{p(c,c')}{p(c)p(c')} + G\end{aligned}$$

$$p(c,c') = \frac{n(c,c')}{\sum_{c,c'} n(c,c')} \qquad p(c) = \frac{n(c)}{\sum_{c} n(c)}$$

G: constant, n(c): number of times cluster c appears in the corpus, n(c,c'): number of times cluster c and cluster c' appear in the same $\mathcal{C}$

# Word Clustering

- Algorithm (1):
  - Start from $|\mathcal{V}|$ clusters: each term corresponds to a cluster
  - Goal: find the output k clusters
    - Run $|\mathcal{V}|$ - k steps:
      - For each step, group $c_i$ and $c_j$ into a cluster
      - Choose the grouped cluster so that its Quality(C) is the highest
- Time complexity:
  - Simple: $O(|\mathcal{V}|^5)$
  - Improved: $O(|\mathcal{V}|^3)$

# Word Clustering

- Algorithm (2):
  - A parameter m (For example: m = 1000)
  - Pick m most frequent terms, categorize them into $c_1, c_2, \ldots, c_m$
  - For each i = (m+1) … $|\mathcal{V}|$
    - Create a new cluster $c_{m+1}$ to store the term at ith-position frequency (m+1 cluster)
    - Group 2 clusters from $c_1 \ldots c_{m+1}$: choose the grouped cluster so that its Quality(C) is the highest (m clusters)
- Time complexity:
  - $O(|\mathcal{V}|m^2 + n)$ (n is the length of corpus)

# Word Clustering

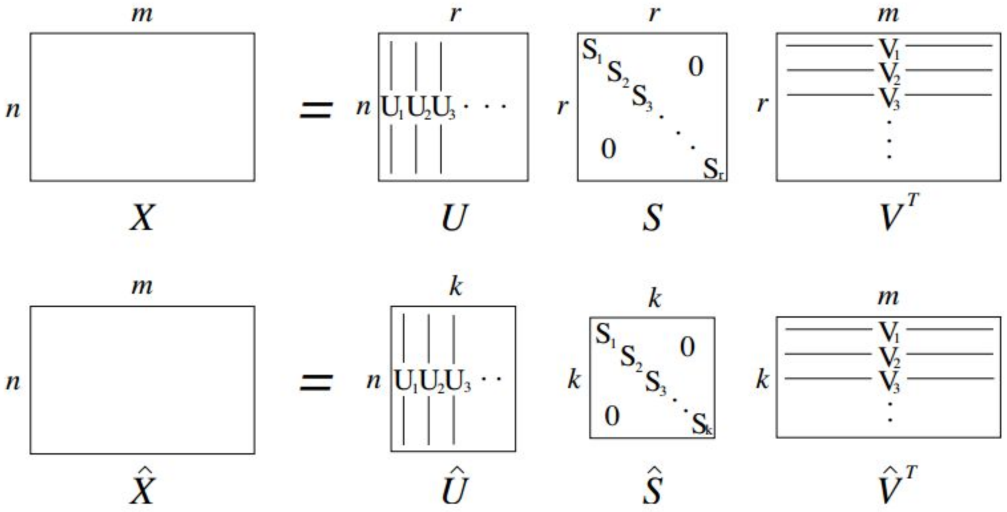
- Tools:
  - mkcls (Franz Och): https://github.com/clab/mkcls
  - brown cluster (Brown): https://github.com/percyliang/brown-cluster

# Word Clustering

- Is it possible to use clustering algorithm such as K-means?
    - word => vector

# Word Clustering

- Word vector:
  - Store important information in fixed-dimension vectors.
  - Methods:
    - Singular Value Decomposition (SVD) applied to co-occurrence matrix
      - motel = [0.286, 0.792, -0.177, -0.107, 0.109,  -0.542, 0.349, 0.271]
      - m = n = size of vocabulary



# Word Clustering

- Word vector:
  - SVD:
    - High time complexity: $O(mn^2)$
    - Difficulty in adding new words
    - No word order

# Word Clustering

- Word vector:
  - TF-IDF: Term Frequency — Inverse Document Frequency
    - A numerical statistic that is intended to reflect how important a word is to a document based on its frequency
    - Observe:
      - A word which occurs many times in a document (high TF) may be more important than a word which occurs few times in the same document (low TF)
      - However, a word that occurs very often in many documents may not be important or relevant (low IDF)

# Word Clustering

- Word vector:
  - TF-IDF: Term Frequency – Inverse Document Frequency

$$tfidf(t,d,D) = tf(t,d) \times idf(d,D)$$

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t'} f_{t',d}}$$

$f_{t,d}$ is number of times term t appears in d

$$idf(t,D) = \log(\frac{N}{n_t})$$

$n_t$ is number of documents in the corpus D (N = |D|) that contain term t

# Word Clustering

- Word vector: TF-IDF
  - With the following corpus:
    - d1: “The sky is blue."
    - d2: “The sun is bright today.”
    - d3: “The sun in the sky is bright."
    - d4: “We can see the shining sun, the bright sun.”
  - Calculate tf-idf for terms of each document

# Word Clustering

- Word vector: TF-IDF
  - Step 1: Remove stopwords
    - d1: “sky blue"
    - d2: “sun bright today”
    - d3: “sun sky bright"
    - d4: “can see shining sun bright sun”

# Word Clustering

- Word vector: TF-IDF
  - Step 2: Calculate TF

$f_{t,d}$

| | blue | bright | can | see | shining | sky | sun | today |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 1 | 1 | 1 | 0 | 2 | 0 |

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t'} f_{t',d}}$$

| | blue | bright | can | see | shining | sky | sun | today |
|---|---|---|---|---|---|---|---|---|
| 1 | 1/2 | 0 | 0 | 0 | 0 | 1/2 | 0 | 0 |
| 2 | 0 | 1/3 | 0 | 0 | 0 | 0 | 1/3 | 1/3 |
| 3 | 0 | 1/3 | 0 | 0 | 0 | 1/3 | 1/3 | 0 |
| 4 | 0 | 1/6 | 1/6 | 1/6 | 1/6 | 0 | 1/3 | 0 |

# Word Clustering

- Word vector: TF-IDF
  - Step 3: Calculate IDF

$f_{t,d}$

| | blue | bright | can | see | shining | sky | sun | today |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 1 | 1 | 1 | 0 | 2 | 0 |
| **n_t** | **1** | **3** | **1** | **1** | **1** | **2** | **3** | **1** |

$N = 4$

$$\mathtt{idf}(t, D) = \log_{10} \frac{N}{n_t}$$

| | blue | bright | can | see | shining | sky | sun | today |
|---|---|---|---|---|---|---|---|---|
| | 0.602 | 0.125 | 0.602 | 0.602 | 0.602 | 0.301 | 0.125 | 0.602 |

$$\log_{10} \frac{4}{1} = 0.602$$

$$\log_{10} \frac{4}{3} = 0.125$$

fit@hcmus

# Word Clustering

- Word vector: TF-IDF

$\texttt{tf}(t, d)$

| | blue | bright | can | see | shining | sky | sun | today |
|---|---|---|---|---|---|---|---|---|
| 1 | 1/2 | 0 | 0 | 0 | 0 | 1/2 | 0 | 0 |
| 2 | 0 | 1/3 | 0 | 0 | 0 | 0 | 1/3 | 1/3 |
| 3 | 0 | 1/3 | 0 | 0 | 0 | 1/3 | 1/3 | 0 |
| 4 | 0 | 1/6 | 1/6 | 1/6 | 1/6 | 0 | 1/3 | 0 |

x

$\texttt{idf}(t, D)$

| | blue | bright | can | see | shining | sky | sun | today |
|---|---|---|---|---|---|---|---|---|
| | 0.602 | 0.125 | 0.602 | 0.602 | 0.602 | 0.301 | 0.125 | 0.602 |

$\texttt{tfidf}(t, d, D) = \texttt{tf}(t, d) \cdot \texttt{idf}(t, D)$

| | blue | bright | can | see | shining | sky | sun | today |
|---|---|---|---|---|---|---|---|---|
| 1 | **0.301** | 0 | 0 | 0 | 0 | 0.151 | 0 | 0 |
| 2 | 0 | 0.0417 | 0 | 0 | 0 | 0 | 0.0417 | **0.201** |
| 3 | 0 | 0.0417 | 0 | 0 | 0 | **0.100** | 0.0417 | 0 |
| 4 | 0 | 0.0209 | **0.100** | **0.100** | **0.100** | 0 | 0.0417 | 0 |

- TF-IDF: Multiply TF and IDF scores, use to rank importance of words within documents
  - Most important word for each document is highlighted

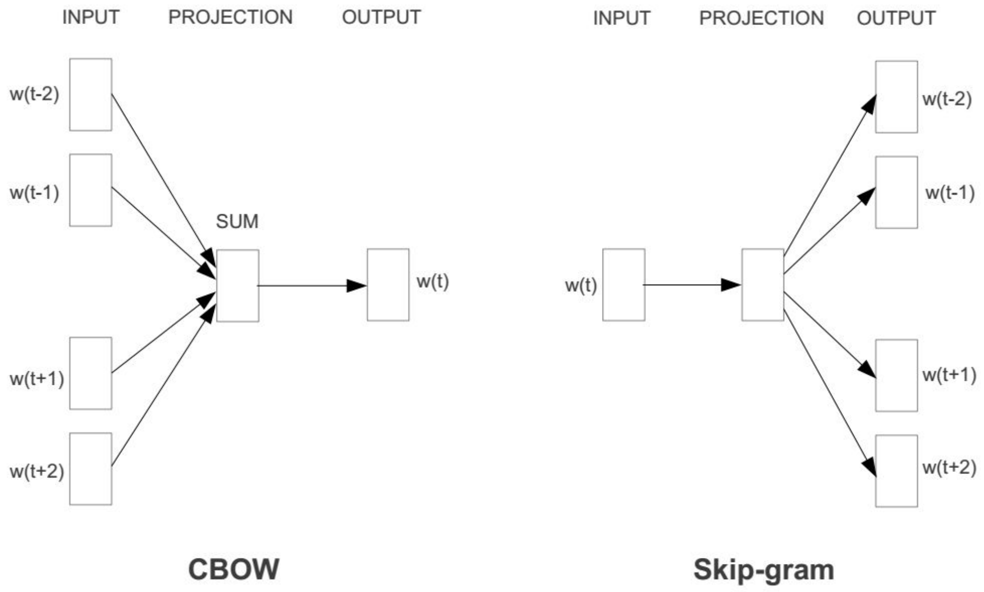
fit@hcmus

# Word Clustering

- Word vector:
  - Word2vec (Mikolov, 2013)
    - Words are represented as vectors in a low-dimensional vector space
    - Word similarity = Vector similarity
    - Prediction model: predict words based on contexts

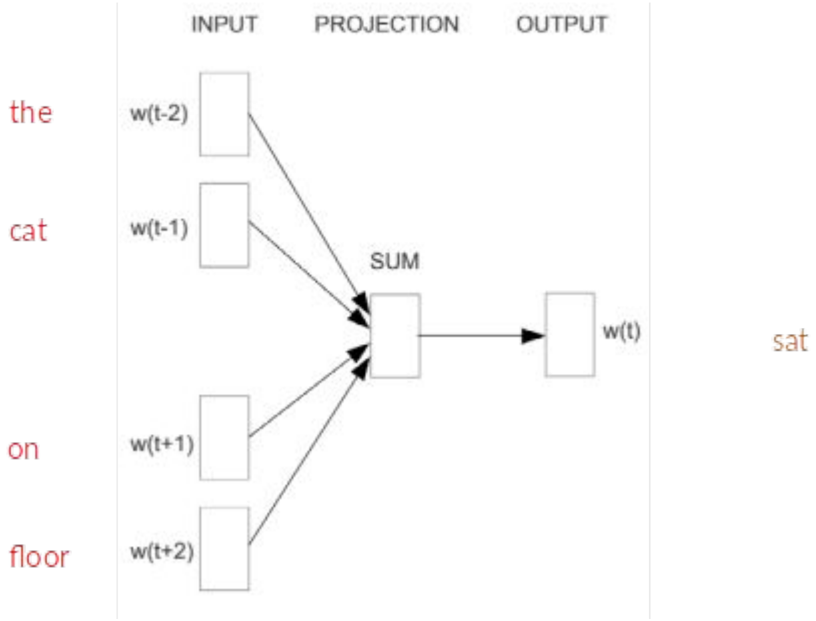
# Word Clustering

- word2vec: uses a neural network model
  - Continuous Bag of Word (CBOW): guess the target word from its neighboring words (context words)
  - Skip-gram (SG): guess a given word’s neighboring words



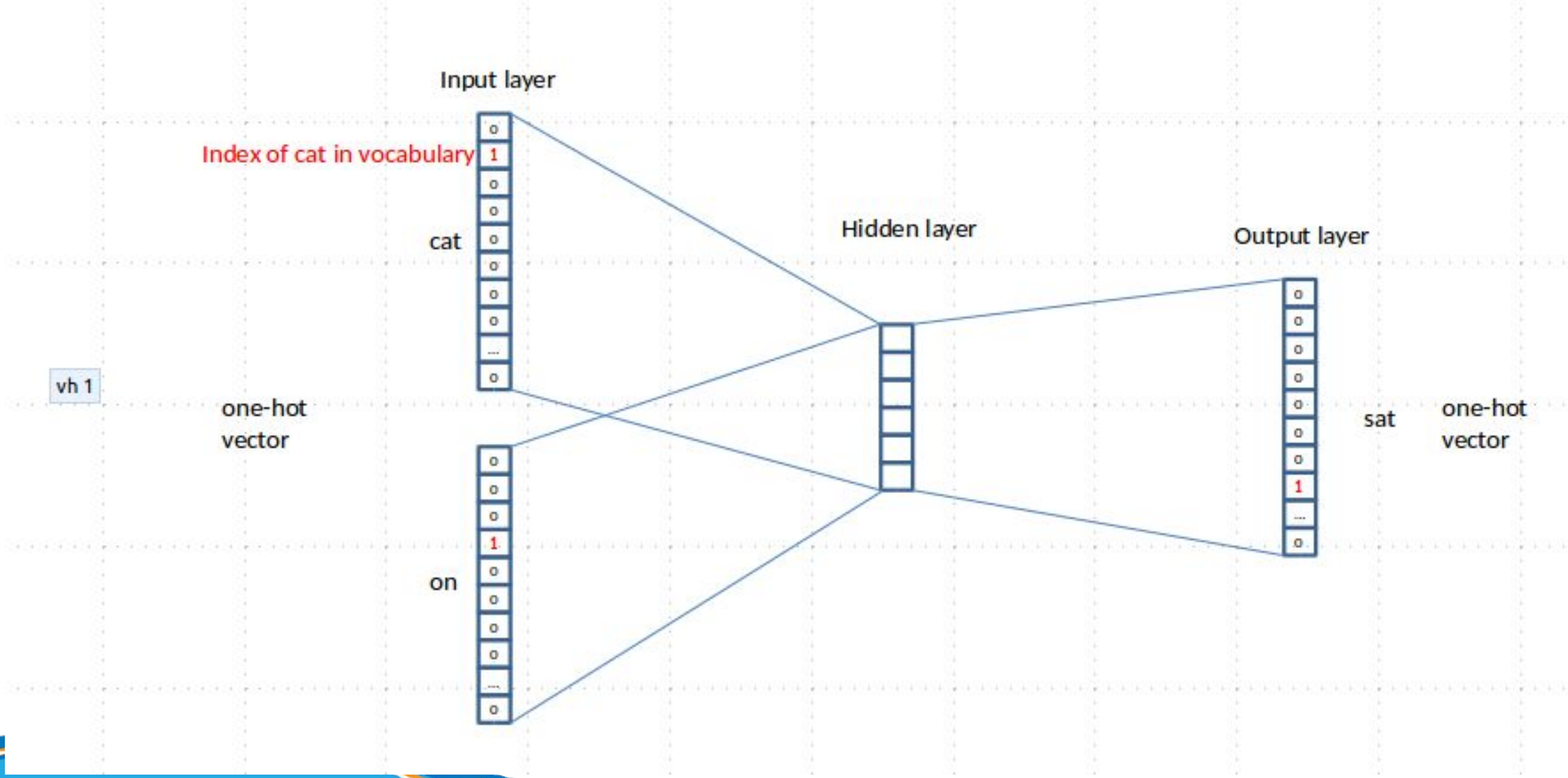
# Word Clustering

- Example: “The cat sat on floor”
  - window = 2



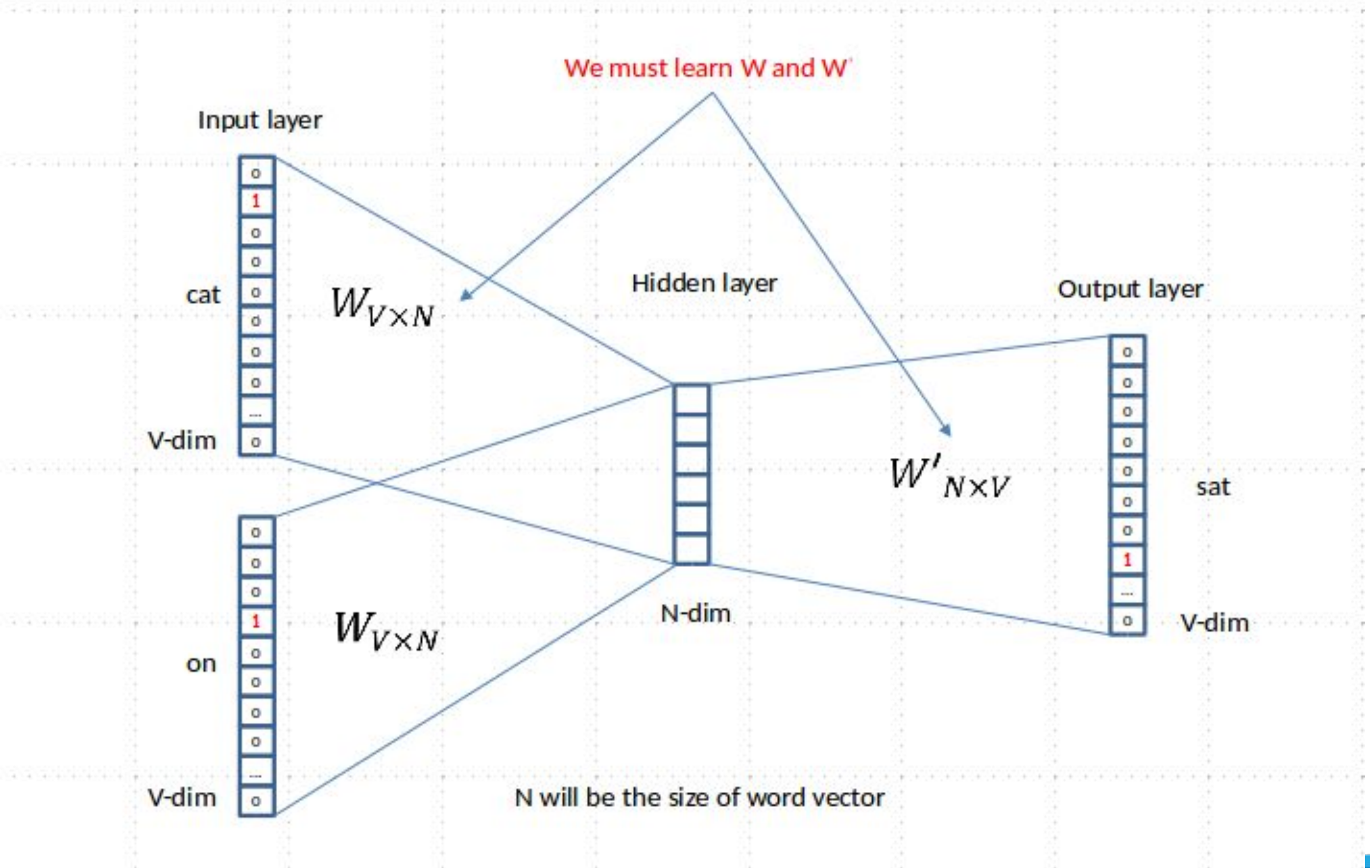
# Word Clustering

- Example:



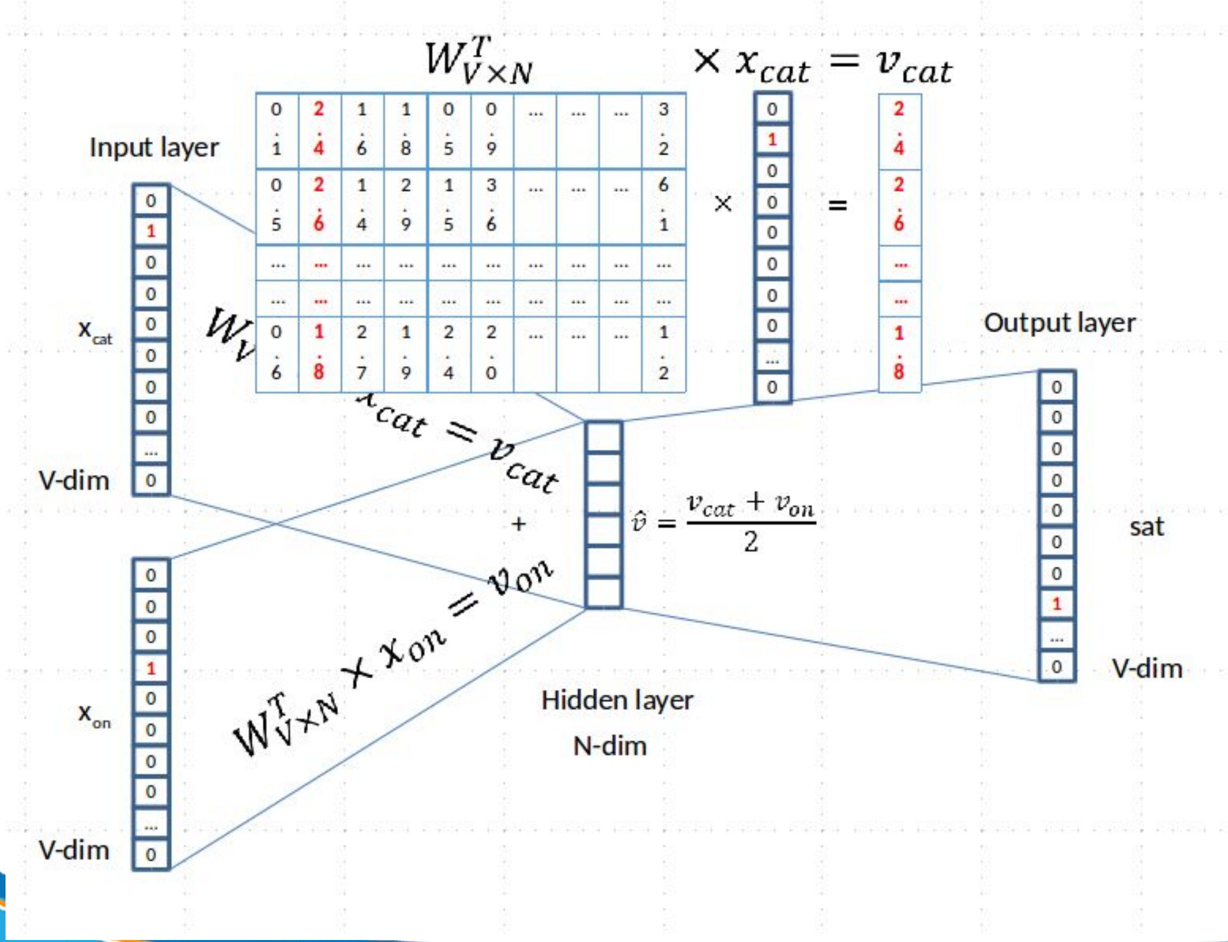
# Word Clustering

- Example:



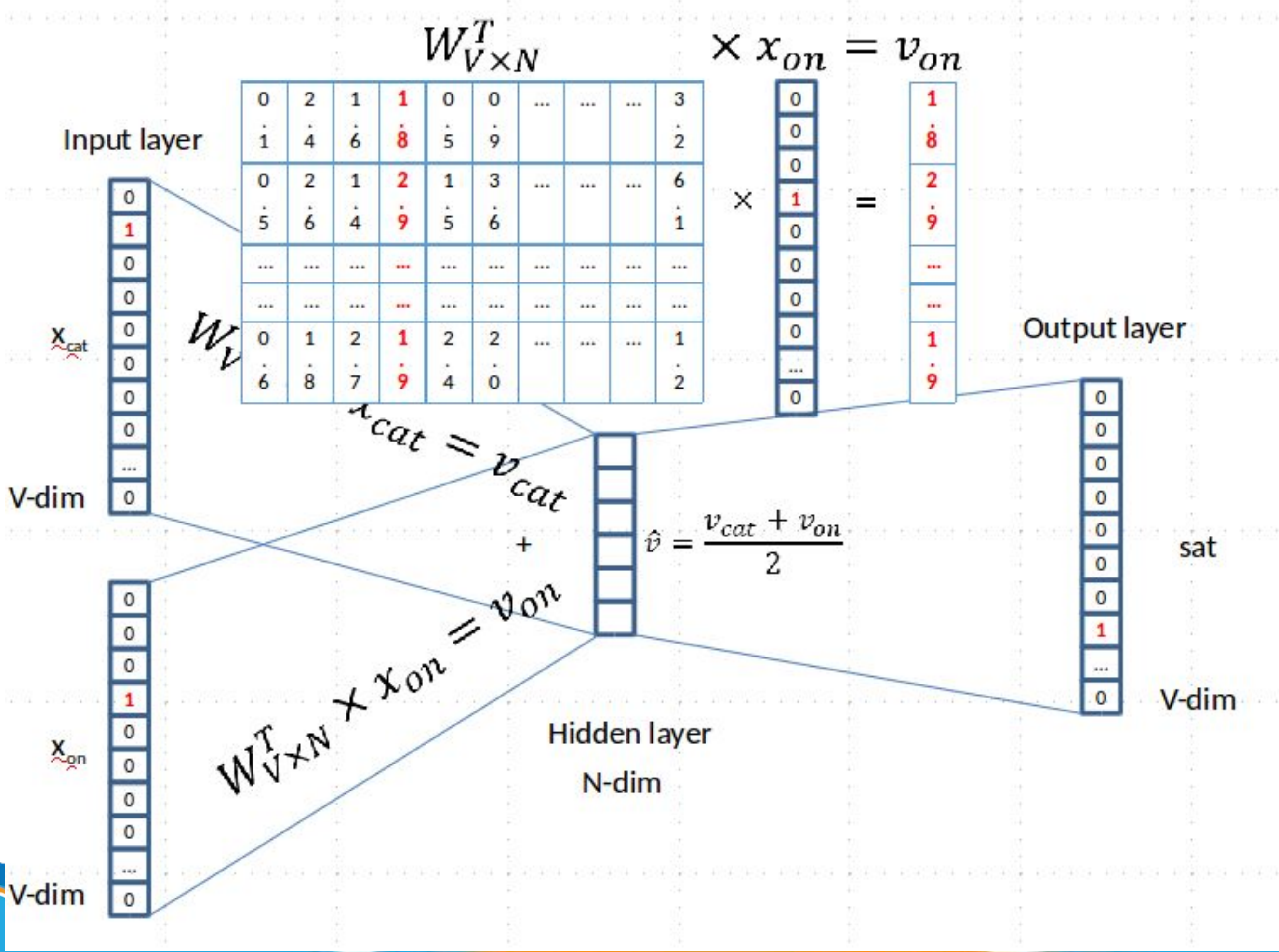
# Word Clustering

- Example:



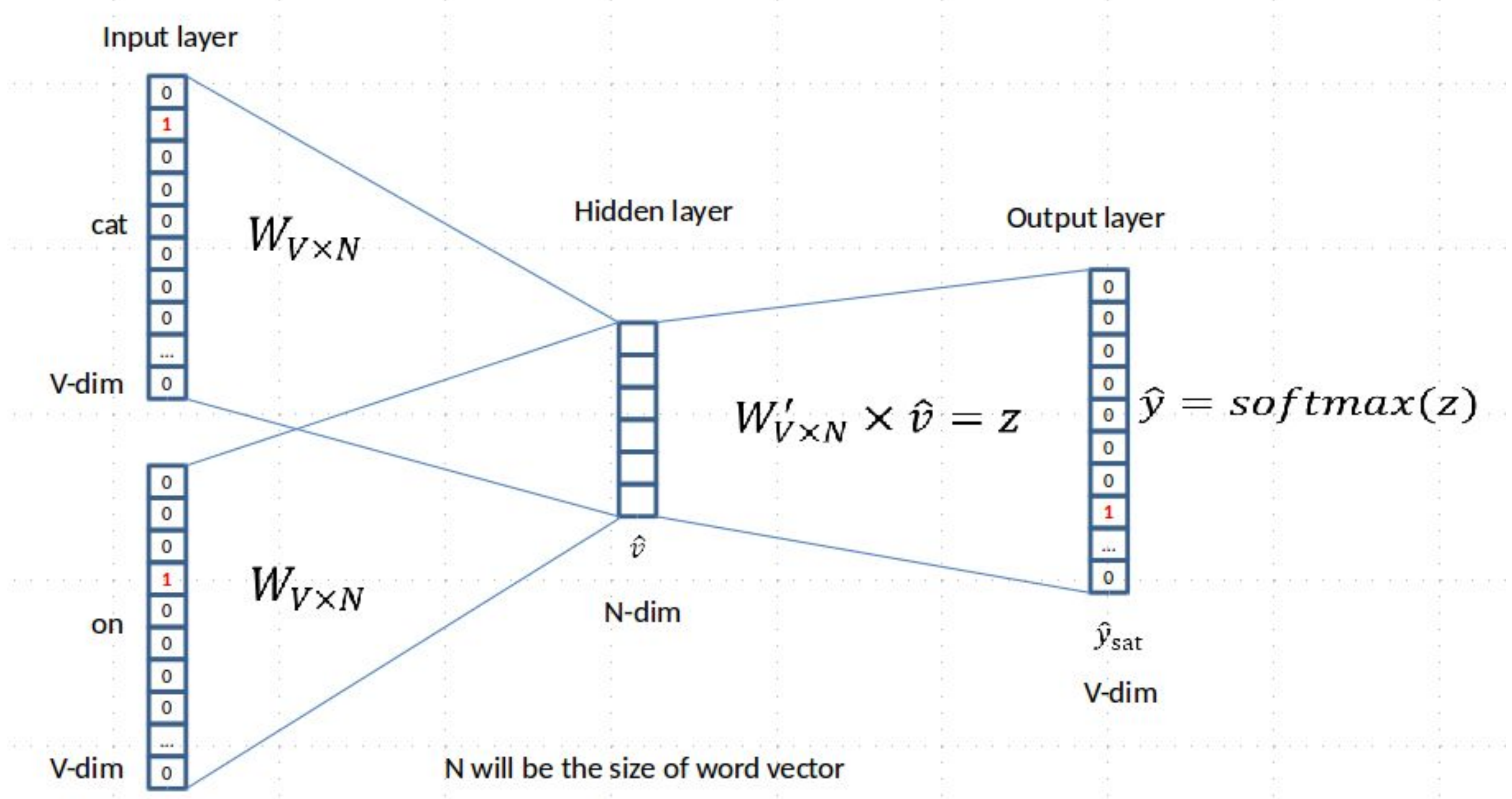
# Word Clustering

- Example:



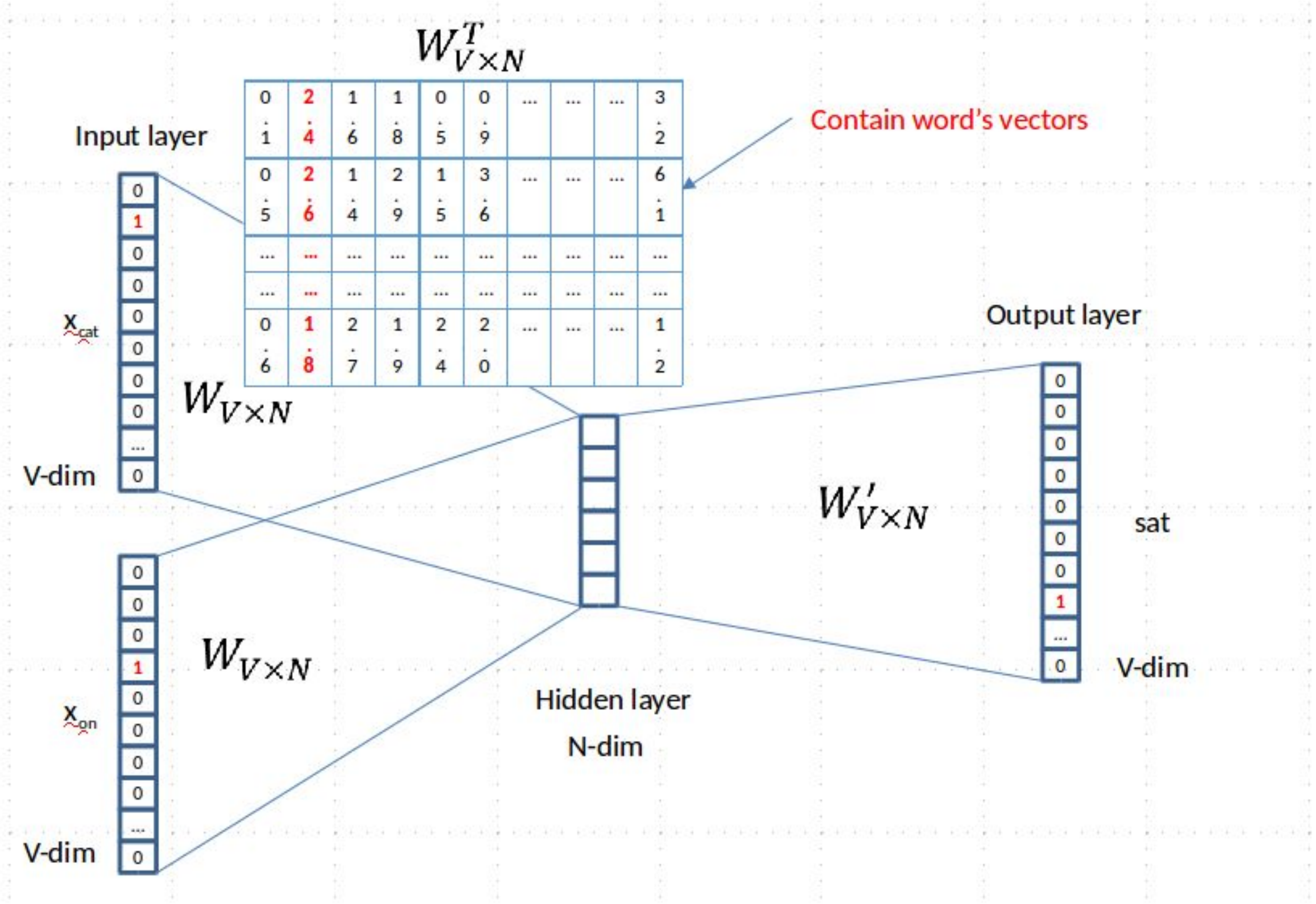
# Word Clustering

- Example:



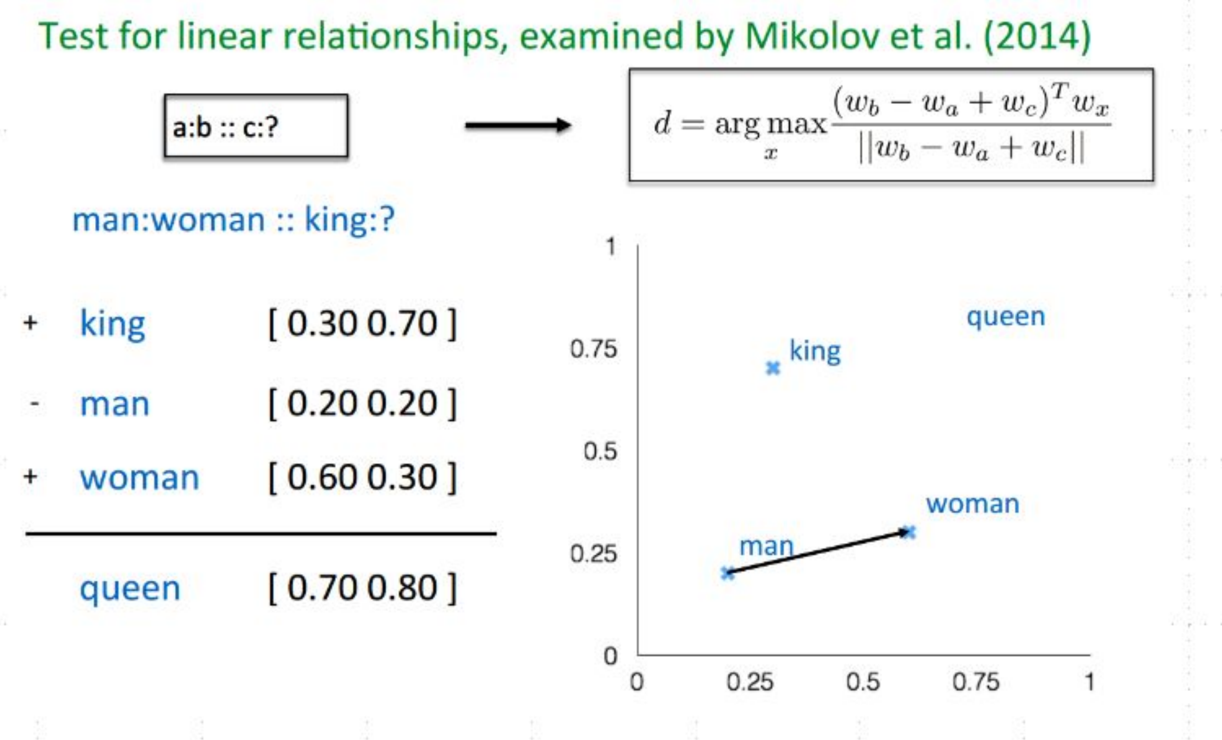
# Word Clustering

- Example:



# Word Clustering

- Example: Word analogy

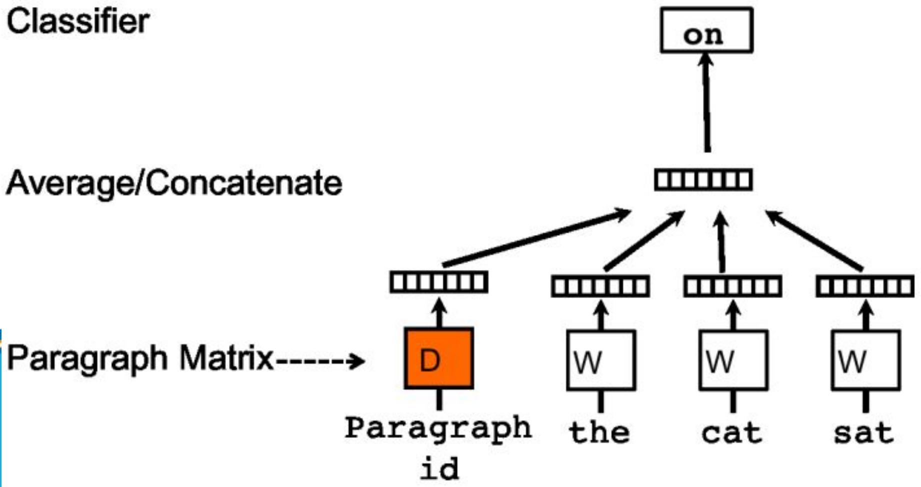


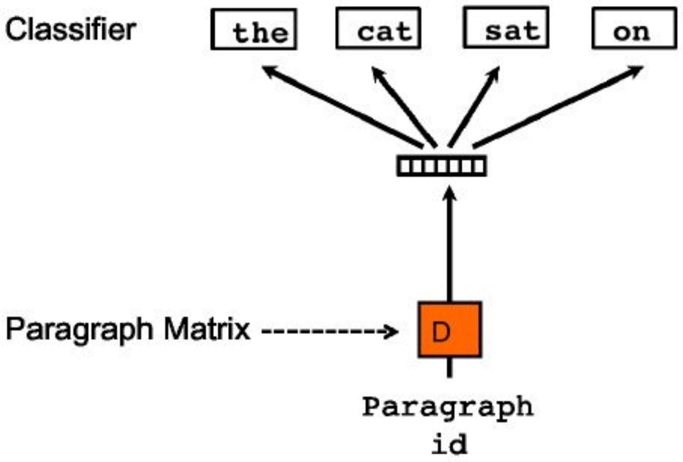
NLP Applications – Document Clustering

# SENTENCE/DOCUMENT CLUSTERING

# Sentence/Document Clustering

- Sentence/Document representation:
  - Simple method: average of word vectors
  - Neural Network: Paragraph vector (Quoc Le 2014)
    - Improved word2vec: represent each document by a vector
    - Or expand the model: document vector is added to the input



Q&A