Santa Clara University

Final Project Introduction

Zhiqiang Tao

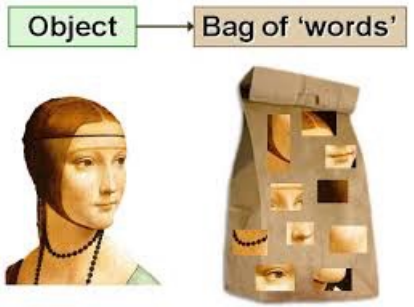
COEN 240: Machine Learning
Department of Computer Science and Engineering

Outline

- Final Project Introduction
  - Project overall
  - Background knowledge
  - Tools recommendation
  - Project requirements

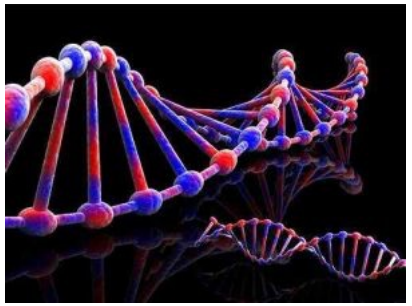
Final Project: representation learning for documents

- What is a document?
  - Sentence? Paragraph? Book?
    - Bag-of-words
  - What else also could be a document?



bag-of-patches

bag-of-frames



bag-of-DNAs

Corpora: a set of documents

Datasets

name	file size	read_more	
20-newsgroups	13 MB	• http://qwone.com/~jason/20Newsgroups/	The notorious collectio posts, partitioned (nea newsgroups.
fake-news	19 MB	• https://www.kaggle.com/mrisdal/fake-news	News dataset, contains and represents 12,999 days. The data was pul because it's coming fro by their BS Detector ar that were missing a lab There are (ostensibly) sources represented in anything you read.
patent-2017	2944 MB	• http://patents.reedtech.com/pgrbft.php	Patent Grant Full Text. sequence data and 'in- patent grant issued in
quora-duplicate-questions	20 MB	• https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs	Over 400,000 lines of line contains IDs for ea each question, and a b contains a duplicate pa
semeval-2016-2017-task3-subtaskA-unannotated	223 MB	• http://alt.qcri.org/semeval2016/task3/ • http://alt.qcri.org/semeval2016/task3/data/uploads/semeval2016-task3-report.pdf • https://github.com/RaRe-Technologies/gensim-data/issues/18 • https://github.com/Witiko/semeval-2016_2017-task3-subtaskA-unannotated-english	SemEval 2016 / 2017 T contains 189,941 ques collected from the Con forum of Qatar Living. language modelling.

https://github.com/RaRe-Technologies/gensim-data

20 Newsgroups

20 Newsgroups

The 20 Newsgroups data set

The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. To the best of my knowledge, it was originally collected by Ken Lang, probably for his *Newsweeder: Learning to filter netnews* paper, though he does not explicitly mention this collection. The 20 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering.

Organization

The data is organized into 20 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other (e.g. **comp.sys.ibm.pc.hardware / comp.sys.mac.hardware**), while others are highly unrelated (e.g **misc.forsale / soc.religion.christian**). Here is a list of the 20 newsgroups, partitioned (more or less) according to subject matter:

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

http://qwone.com/~jason/20Newsgroups/

http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

Preliminary knowledge

Notation and terminology (text collections)

- *Word*: the basic unit from a vocabulary of size V (includes V distinct words). The v-th word is represented by

$$w = \underbrace{[0 \cdots 0 \underbrace{1}_{vth} 0 \cdots 0]}_{V-\text{dim}}^{T}$$

- *Document*: a sequence of N words.
- *Corpus*: a collection of M documents.

Assumptions:

- The words in a document are exchangeable.
- Documents are also exchangeable.

Our tasks

- Preprocess the dataset
  - Clean the data and build the vocabulary
  - Visualize the statistics of the dataset
  - Baseline document features
    - Bag-of-words; TF-IDF Model
- Topic Modeling
  - Train a LDA model with given topic#
  - Visualize different topics
- Vector representation of documents
  - Train a Doc2Vec model
  - Visualize word embedding and document embedding
- Comparison between different document representations
  - Document clustering

Our tasks

- Clean the corpora
  - Lower the words
  - Remove the stopwords, punctuation, and special symbols
  - Tokenization, stemming, and lemmatization
  - Filtering the word
    - term frequency; document frequency
  - Make n-gram (optional)
- Build the vocabulary
- Tools
  - NLTK
    - tokenize, stem, and etc
  - Gensim
    - corpora

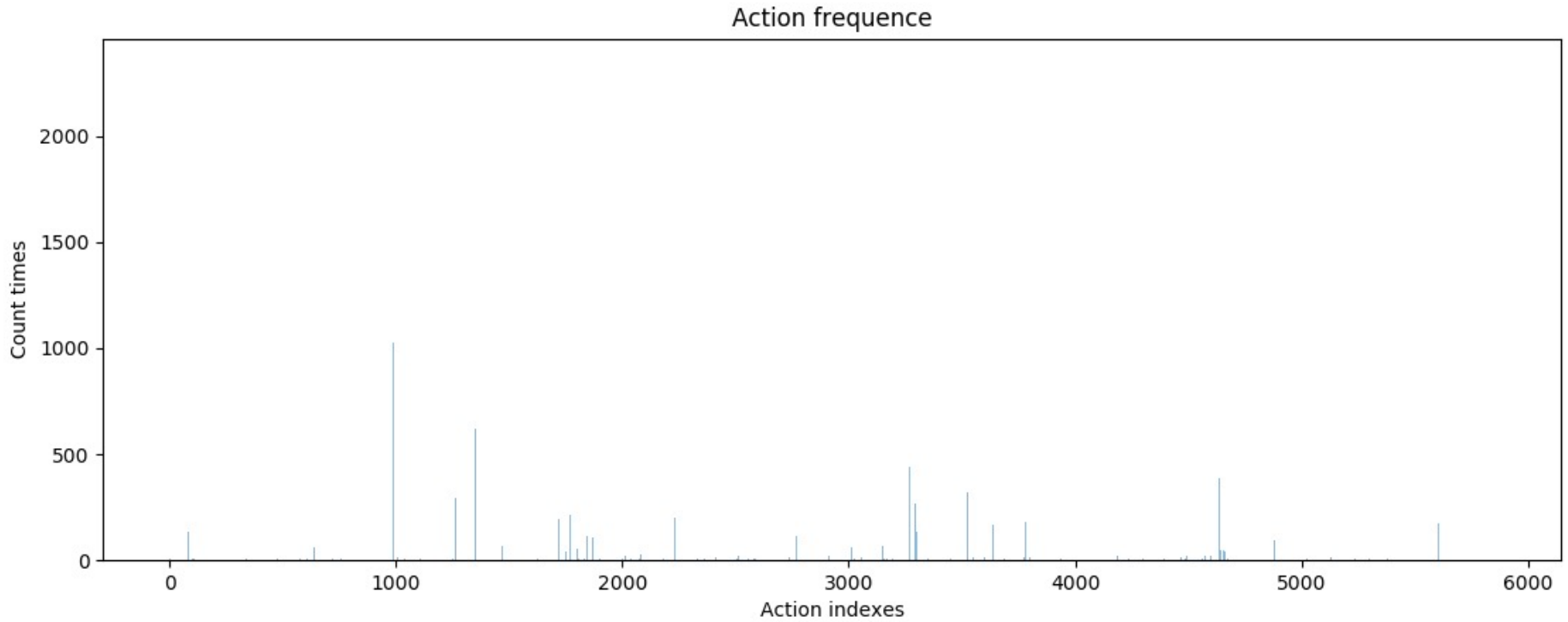
Statistics of the dataset

- Documents and Categories
  - Sentence length
    - min, max, avg, std
    - table, boxplot, or else?
  - Vocabulary size

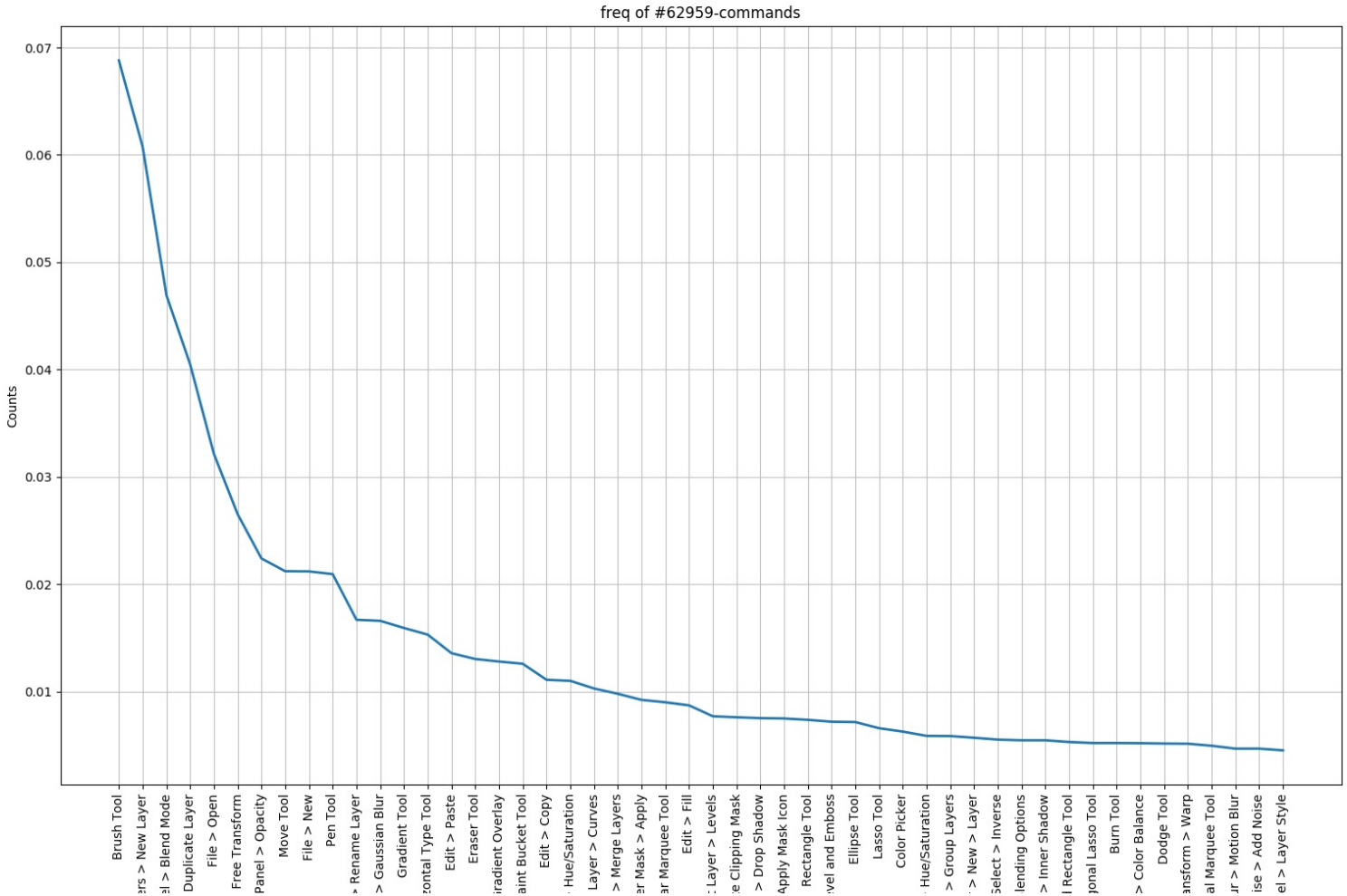
# of books	# of sentences	# of words	# of unique words	mean # of words per sentence
11,038	74,004,228	984,846,357	1,316,420	13

- Word distribution
- Document frequency

How to show the statistics?



How to show the statistics?



Hand-crafted features: BoW and TF-IDF

Binary term-document incidence matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Each document is represented by a binary vector $\in \{0,1\}^{|V|}$

http://web.stanford.edu/class/cs276/course_schedule.html

Hand-crafted features: BoW and TF-IDF

Term-document count matrices

- Consider the number of occurrences of a term in a document:
  - Each document is a count vector

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

http://web.stanford.edu/class/cs276/course_schedule.html

Document frequency

- Frequent terms are less informative than rare terms
- Consider a query term that is frequent in the collection (e.g., *high, increase, line*)
- A document containing such a term is more likely to be relevant than a document that doesn't
- But it's not a sure indicator of relevance.
- For frequent terms, we want high positive weights for words like *high, increase, and line*
- But lower weights than for rare terms.
- We will use document frequency (df) to capture this.

http://web.stanford.edu/class/cs276/course_schedule.html

TF-IDF: term frequency–inverse document frequency

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = \log(1 + \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

- Best known weighting scheme in information retrieval
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

http://web.stanford.edu/class/cs276/course_schedule.html

BoW and TF-IDF

Binary → count → weight matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

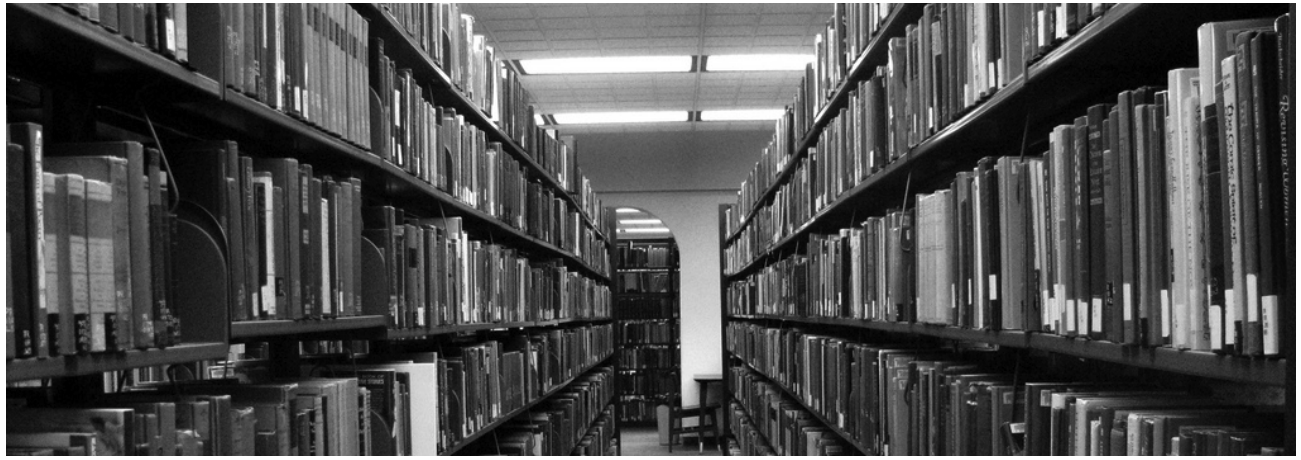
Each document is now represented by a real-valued vector of tf-idf weights $\in R^{|V|}$

http://web.stanford.edu/class/cs276/course_schedule.html

Our tasks

- Preprocess the dataset
  - Clean the data and build the vocabulary
  - Visualize the statistics of the dataset
  - Baseline document features
    - Bag-of-words; TF-IDF Model
- Topic Modeling
  - Train a LDA model with given topic#
  - Visualize different topics
- Vector representation of documents
  - Train a Doc2Vec model
  - Visualize word embedding and document embedding
- Comparison between different document representations
  - Document clustering

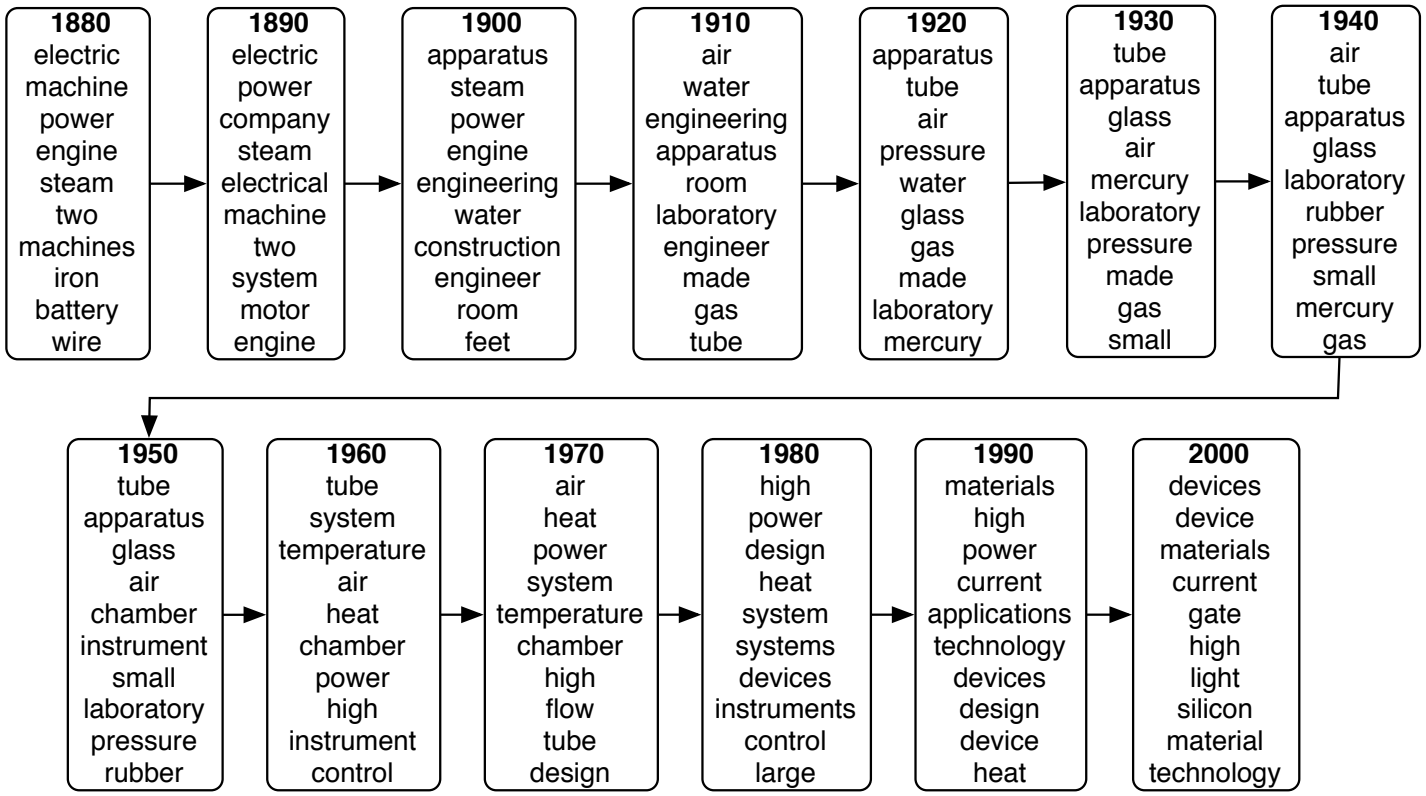
Topic Modeling



TOPIC MODELING

1. **Discover** the thematic structure
2. **Annotate** the documents
3. **Use** the annotations to visualize, organize, summarize, ...

Topic Modeling



Topic Modeling

SKY WATER TREE
MOUNTAIN PEOPLE

SCOTLAND WATER
FLOWER HILLS TREE

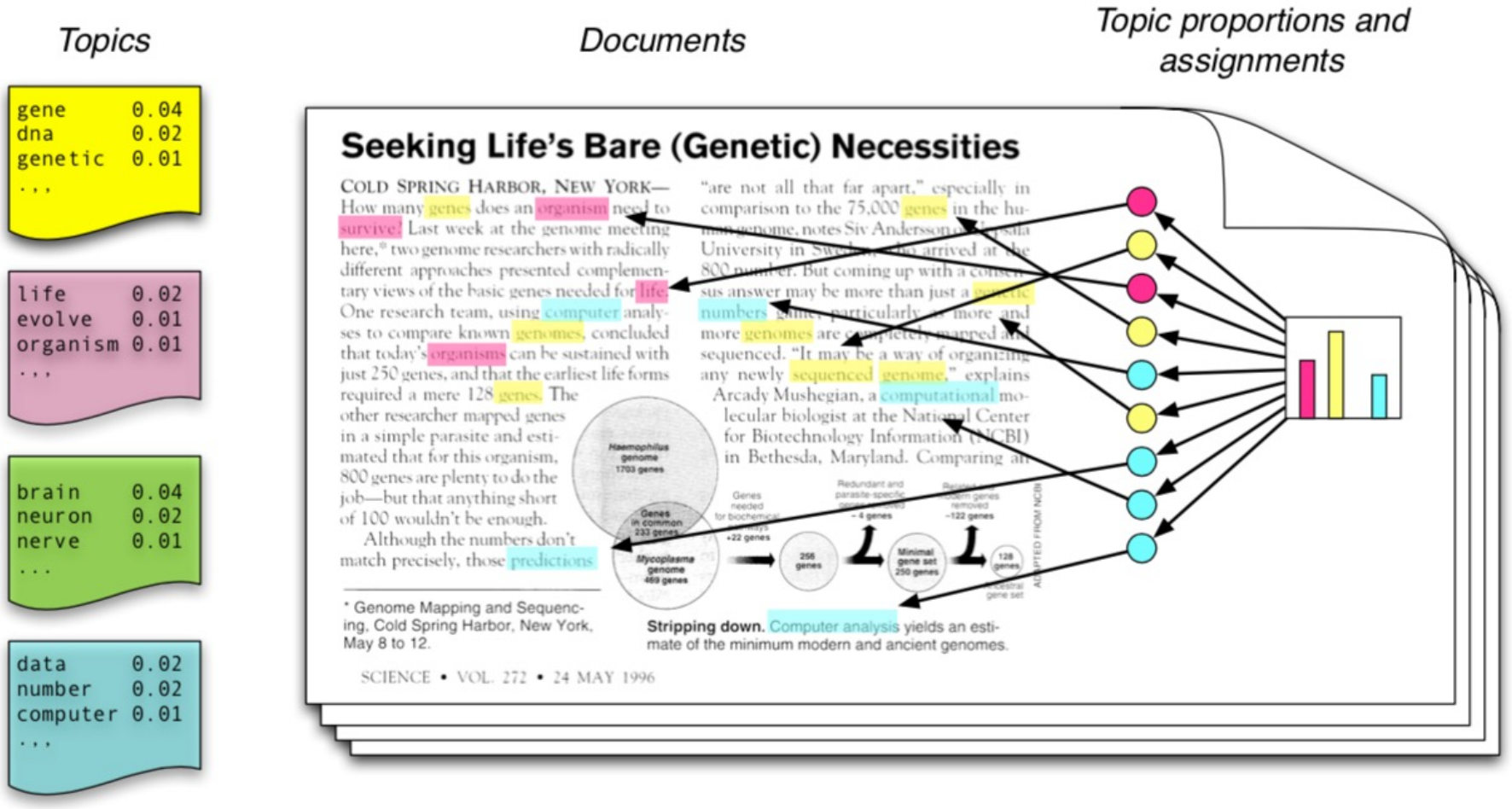
SKY WATER BUILDING
PEOPLE WATER

FISH WATER OCEAN
TREE CORAL

PEOPLE MARKET PATTERN
TEXTILE DISPLAY

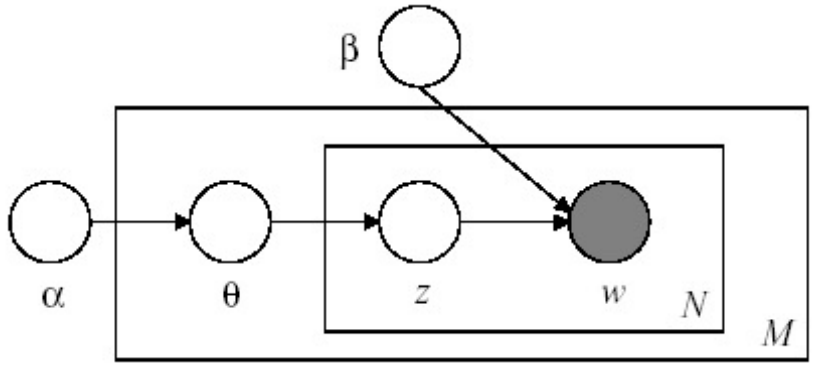
BIRDS NEST TREE
BRANCH LEAVES

Topic Modeling



Topic Modeling: Latent Dirichlet Allocation (LDA)

Generative process for each document W in a corpus D:



1. Choose $\theta \sim Dirichlet(\alpha), \quad \theta \text{ and } \alpha \text{ are } k-\text{dim}$
2. For each of the N words w_n

(a) Choose a topic $z_n \sim Multinomial(\theta), \quad z_n \text{ are } k-\text{dim index}$

(b) Choose a word $w_n \sim Multinomial(\beta_{z_n}), \quad \beta \text{ is a } k \times V \text{ matrix}$

$$\beta_{ij} = p(w^j = 1 \mid z^i = 1)$$

θ are document-level variables, z and w are word-level variables.

LDA Training by Gensim

```
>>> lda = LdaModel(common_corpus, num_topics=50, alpha='auto', eval_every=5)  # learn asymmetric alpha from data
```

Parameters:

- **corpus** (*{iterable of list of (int, float), scipy.sparse.csc}, optional*) – Stream of document vectors or sparse matrix of shape (*num_terms*, *num_documents*). If not given, the model is left untrained (presumably because you want to call **`update()`** manually).
- **num_topics** (*int, optional*) – The number of requested latent topics to be extracted from the training corpus.
- **id2word** ({dict of (int, str), **`gensim.corpora.dictionary.Dictionary`**}) – Mapping from word IDs to words. It is used to determine the vocabulary size, as well as for debugging and topic printing.
- **distributed** (*bool, optional*) – Whether distributed computing should be used to accerelate training.
- **chunksize** (*int, optional*) – Number of documents to be used in each training chunk.
- **passes** (*int, optional*) – Number of passes through the corpus during training.
- **update_every** (*int, optional*) – Number of documents to be iterated through for each update. Set to 0 for batch learning, > 1 for online iterative learning.
- **alpha** (*{numpy.ndarray, str}, optional*) –

  Can be set to an 1D array of length equal to the number of expected topics that expresses our a-priori belief for the each topics' probability. Alternatively default prior selecting strategies can be employed by supplying a string:

  - 'asymmetric': Uses a fixed normalized asymmetric prior of *1.0 / topicno*.
  - 'auto': Learns an asymmetric prior from the corpus.

- **eta** (*{float, np.array, str}, optional*) –

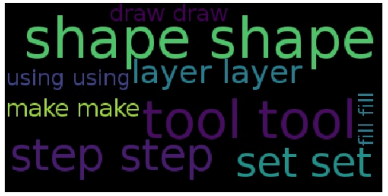
  A-priori belief on word probability, this can be:

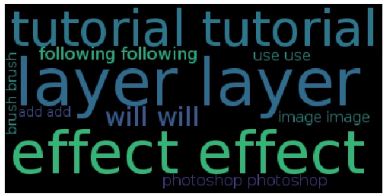
  - scalar for a symmetric prior over topic/word probability,
  - vector of length num_words to denote an asymmetric user defined probability for each word,
  - matrix of shape (num_topics, num_words) to assign a probability for each word-topic combination,
  - the string 'auto' to learn the asymmetric prior from the data.

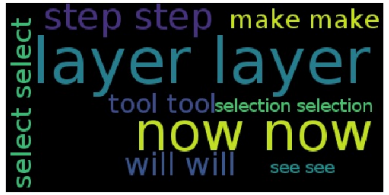
- **decay** (*float, optional*) – A number between (0.5, 1] to weight what percentage of the previous lambda value is forgotten

Some topics visualization

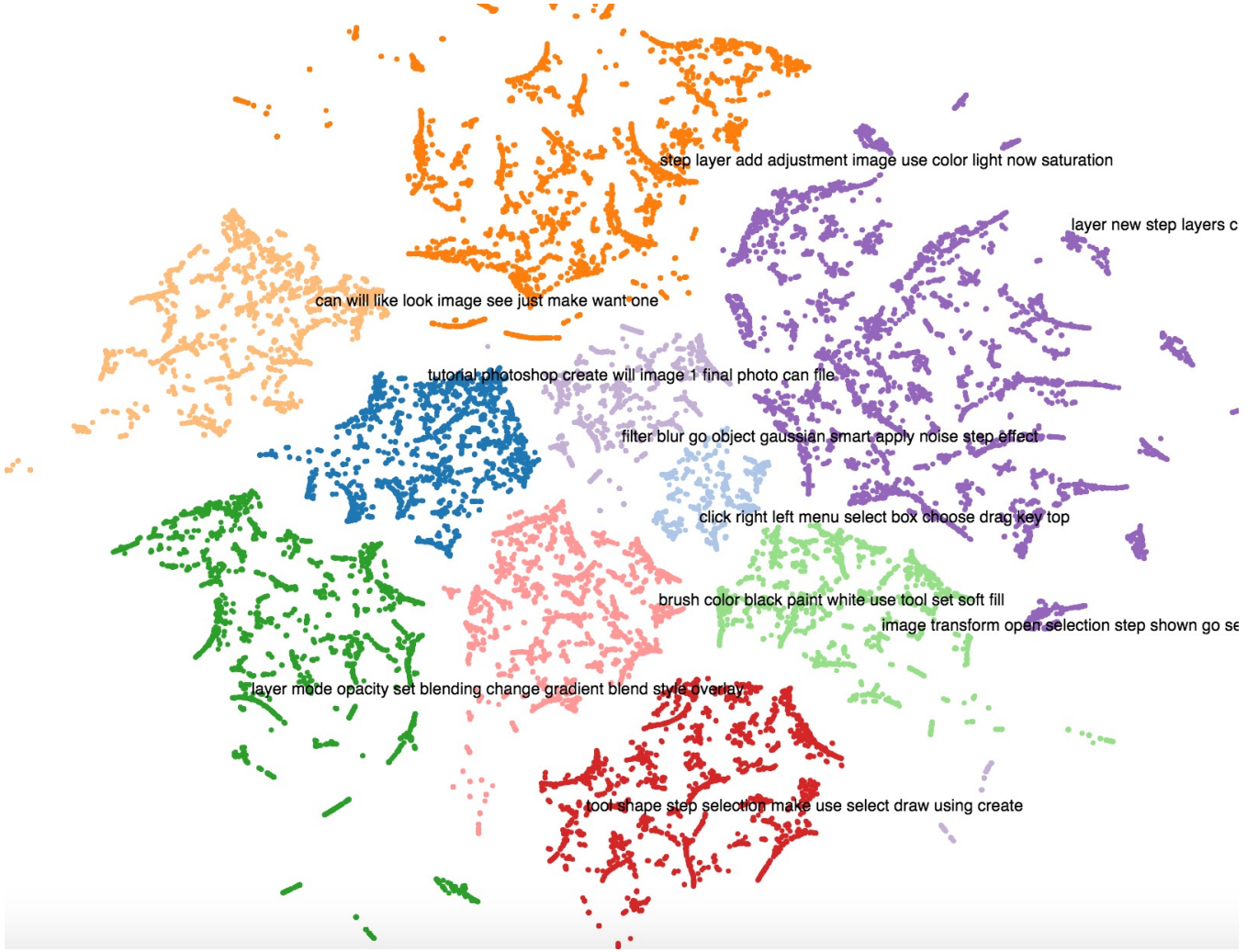
```
Top 10 terms for topic #0: layer, effect, will, tutorial, following, photoshop, image, use, brush, add
Top 10 terms for topic #1: layer, image, click, layers, photo, drag, photoshop, palette, top, select
Top 10 terms for topic #2: image, use, make, ctrl, 2, step, selection, layer, name, 1
Top 10 terms for topic #3: layer, step, blur, filter, go, create, click, use, new, set
Top 10 terms for topic #4: text, layer, style, step, type, create, add, font, color, shadow
Top 10 terms for topic #5: step, layer, click, create, image, tool, color, following, opacity, shown
Top 10 terms for topic #6: water, 3d, texture, will, shown, step, scene, light, cocoon, 1
Top 10 terms for topic #7: layer, brush, step, tool, opacity, new, light, soft, use, mode
Top 10 terms for topic #8: step, layer, blend, create, gradient, opacity, mode, now, add, set
Top 10 terms for topic #9: mask, layer, now, step, selection, layers, use, add, channel, apply
Top 10 terms for topic #10: photoshop, can, image, will, like, look, images, just, tutorial, picture
Top 10 terms for topic #11: layer, next, color, tool, new, picture, layers, select, click, create
Top 10 terms for topic #12: image, adjustment, layer, filter, photo, effect, black, color, can, smart
Top 10 terms for topic #13: layer, copy, rider, cup, add, mask, shadows, wedge, shadow, select
Top 10 terms for topic #14: step, layer, adjustment, use, mask, make, add, color, curves, 1
Top 10 terms for topic #15: shape, tool, step, set, layer, make, draw, using, fill, like
Top 10 terms for topic #16: path, create, will, use, layer, pattern, coffee, new, tool, background
Top 10 terms for topic #17: step, use, paint, brush, create, can, color, new, base, painting
Top 10 terms for topic #18: layer, now, step, will, select, can, tool, make, selection, see
Top 10 terms for topic #19: color, area, image, can, click, background, using, tool, just, new
```



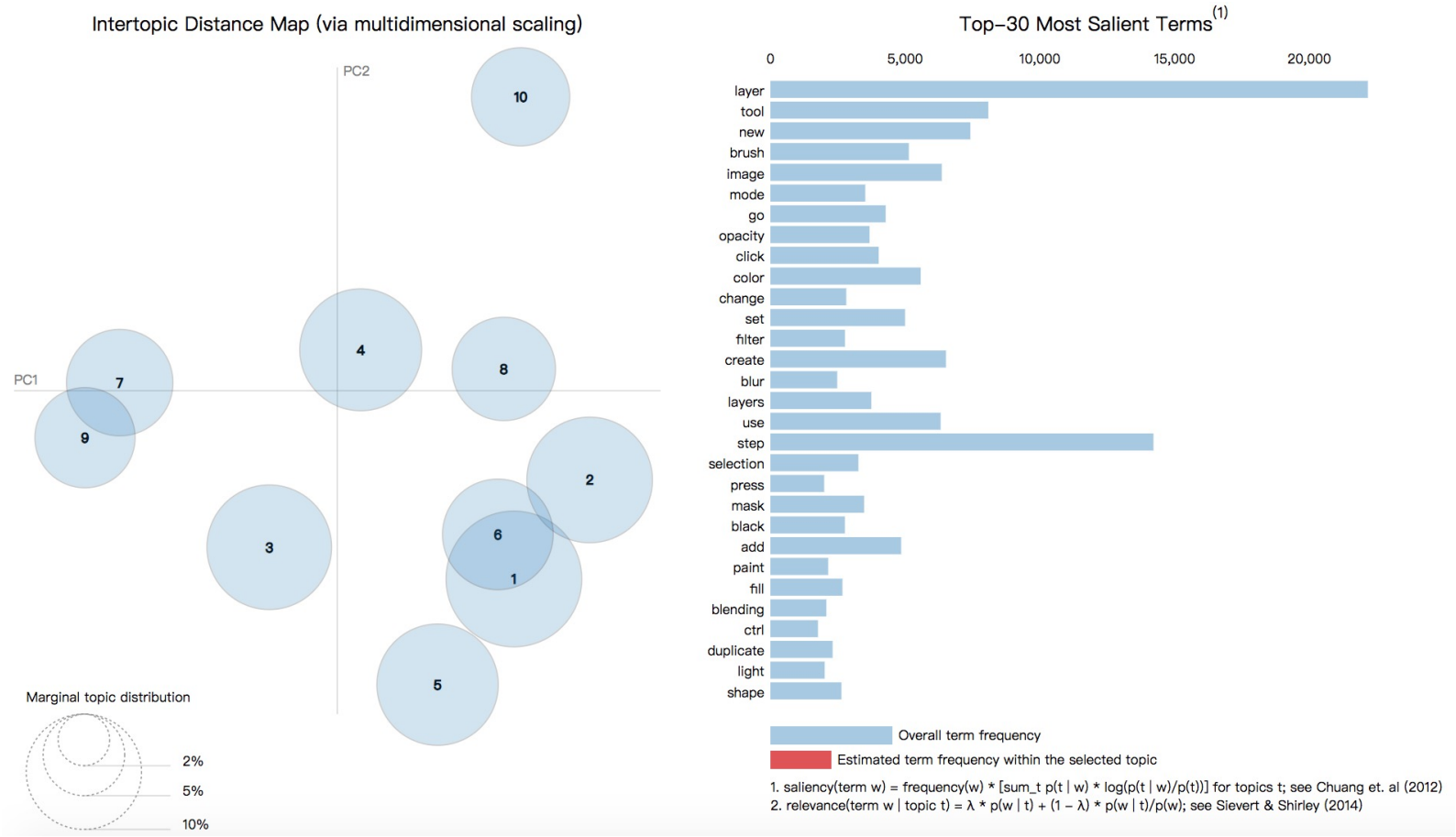




Some topics visualization: T-SNE



Some topics visualization: LdaVis



Our tasks

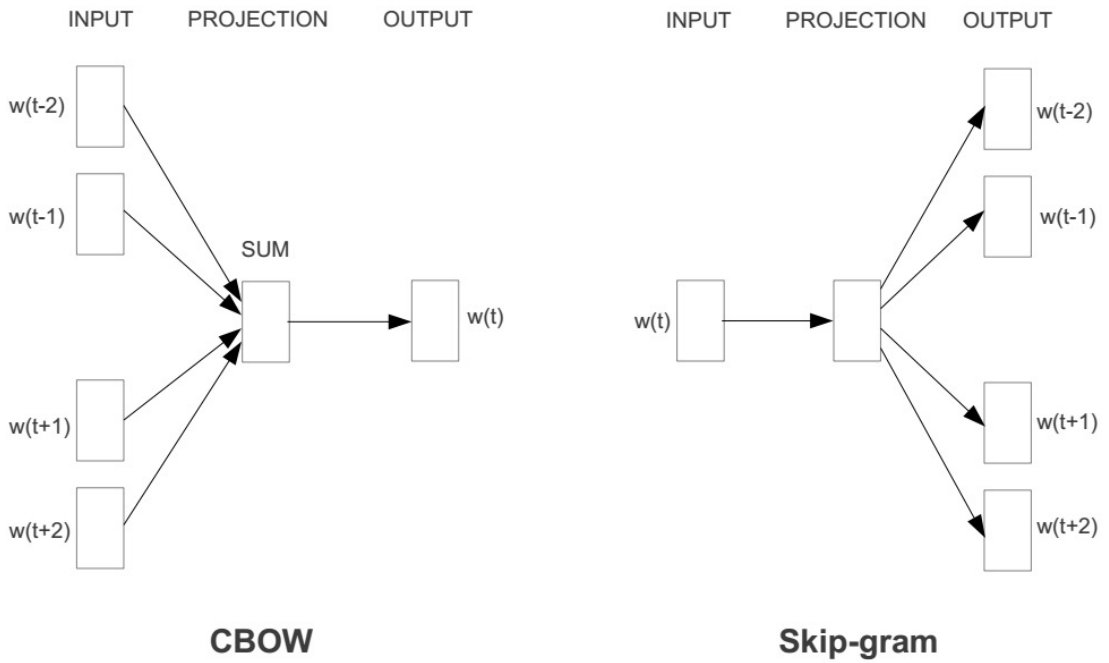
- Preprocess the dataset
  - Clean the data and build the vocabulary
  - Visualize the statistics of the dataset
  - Baseline document features
    - Bag-of-words; TF-IDF Model
- Topic Modeling
  - Train a LDA model with given topic#
  - Visualize different topics
- Vector representation of documents
  - Train a Doc2Vec model
  - Visualize word embedding and document embedding
- Comparison between different document representations
  - Document clustering

Word2Vec: represent the meaning of words

- Represent each word with a low-dimensional vector
- Word similarity = vector similarity
- Key idea: Predict surrounding words of every word
- Faster and can easily incorporate a new sentence/document or add a word to the vocabulary

Word2Vec Model

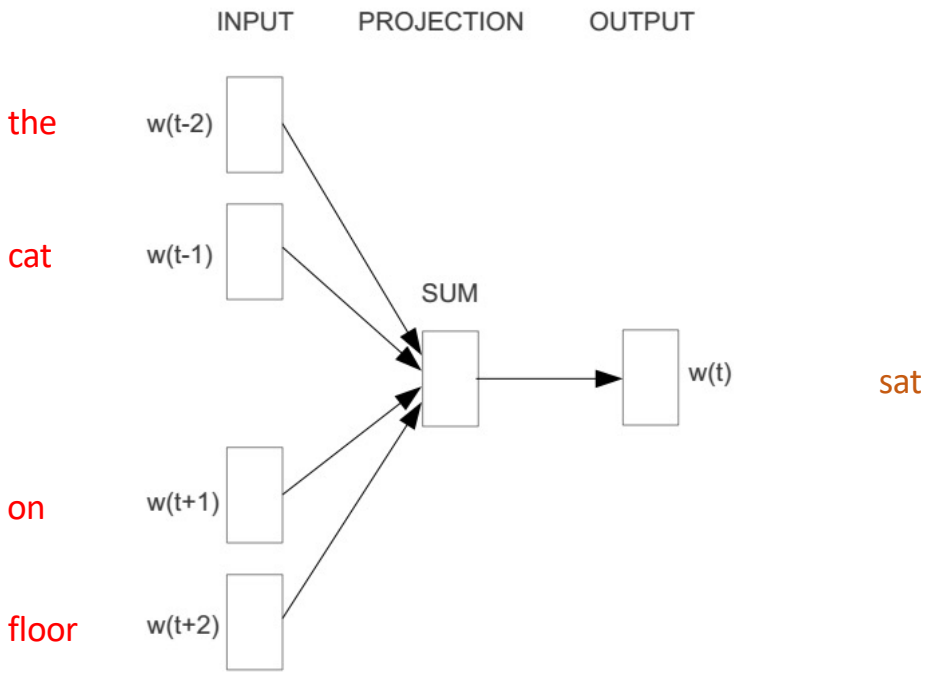
- 2 basic neural network models:
  - Continuous Bag of Word (CBOW): use a window of word to predict the middle word
  - Skip-gram (SG): use a word to predict the surrounding ones in window.



http://www.cs.ucr.edu/~vagelis/classes/CS260/

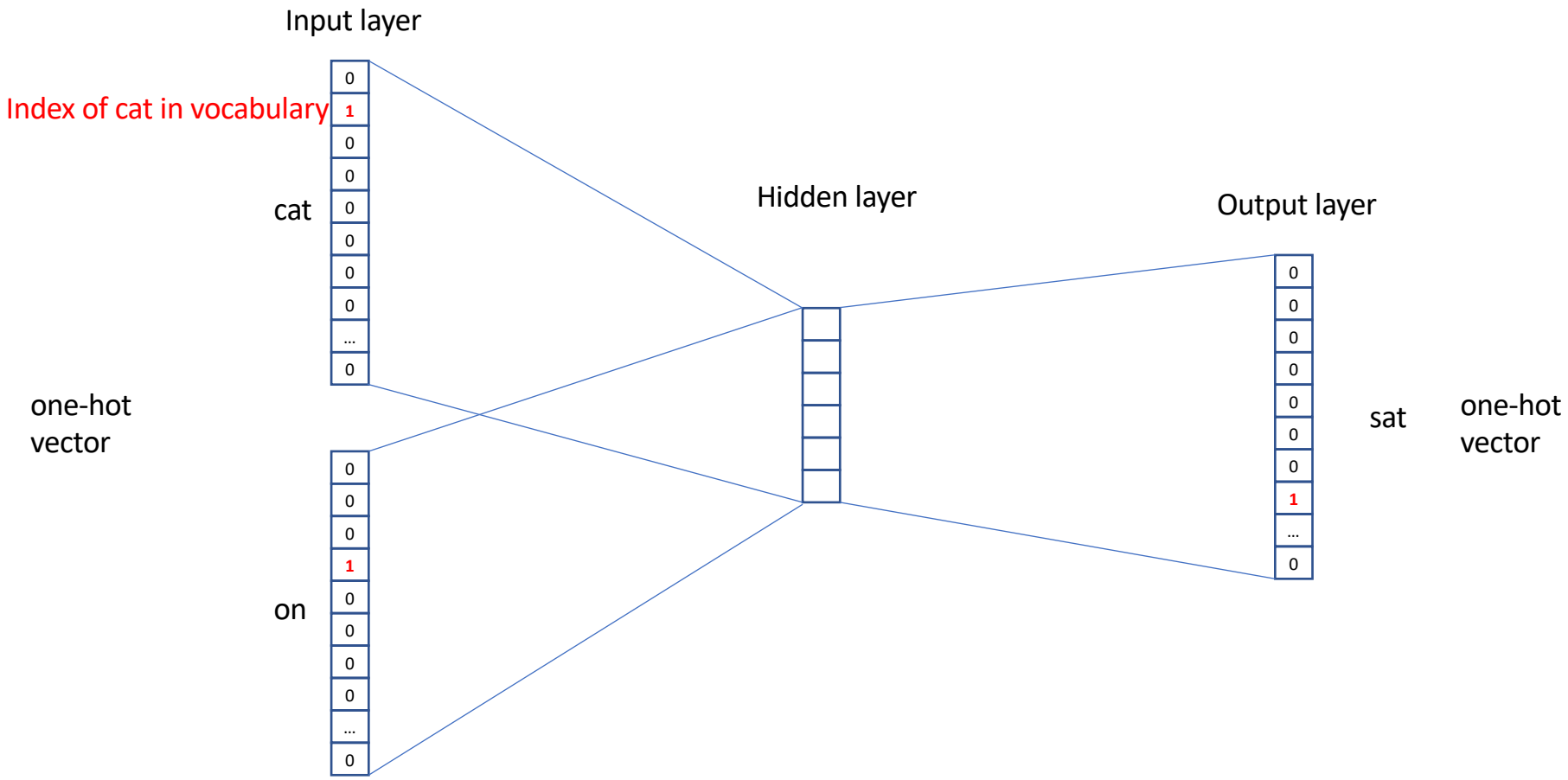
Word2Vec: continuous bag-of-words

- E.g. “The cat sat on floor”
  - Window size = 2

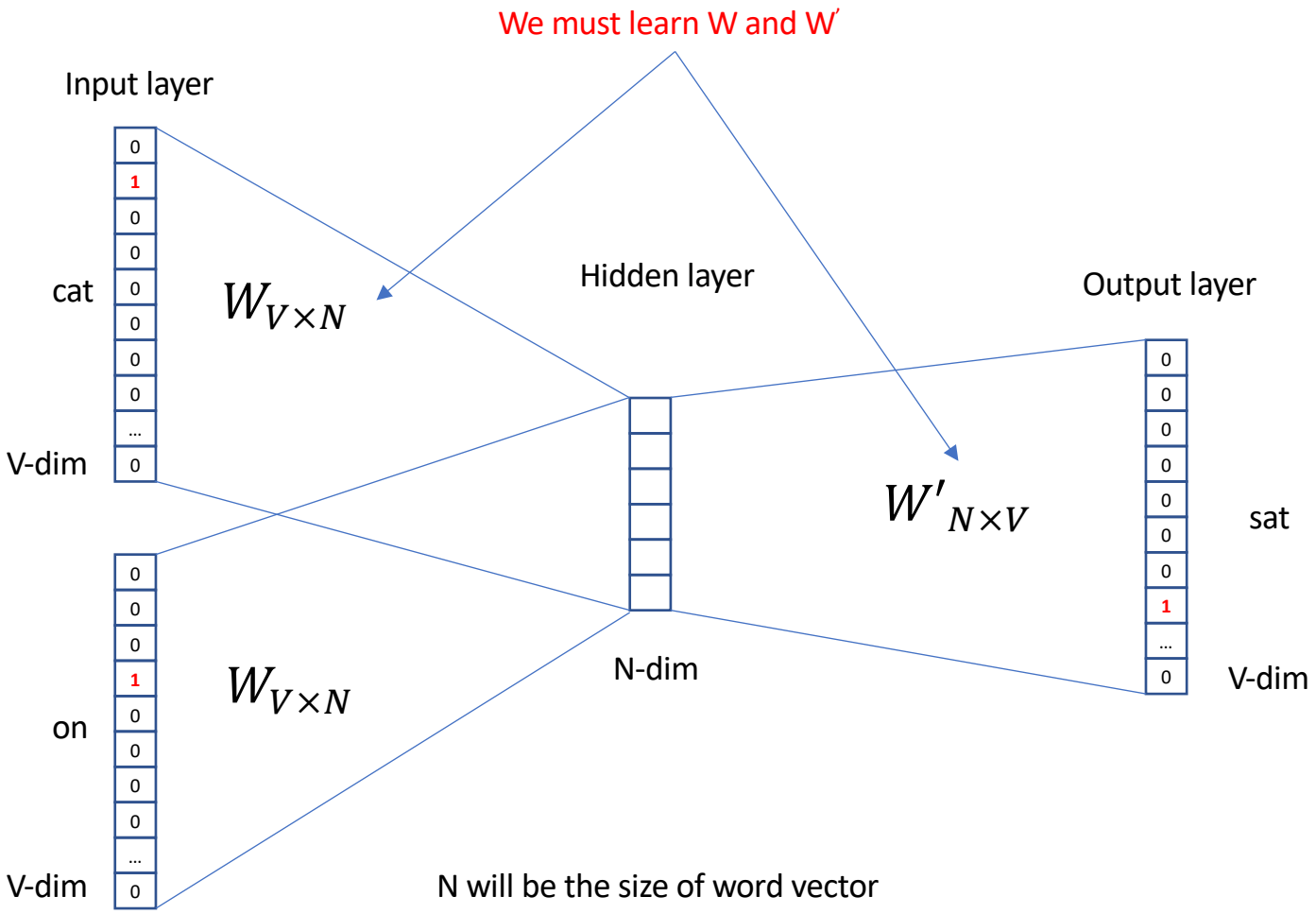


http://www.cs.ucr.edu/~vagelis/classes/CS260/

Word2Vec: continuous bag-of-words

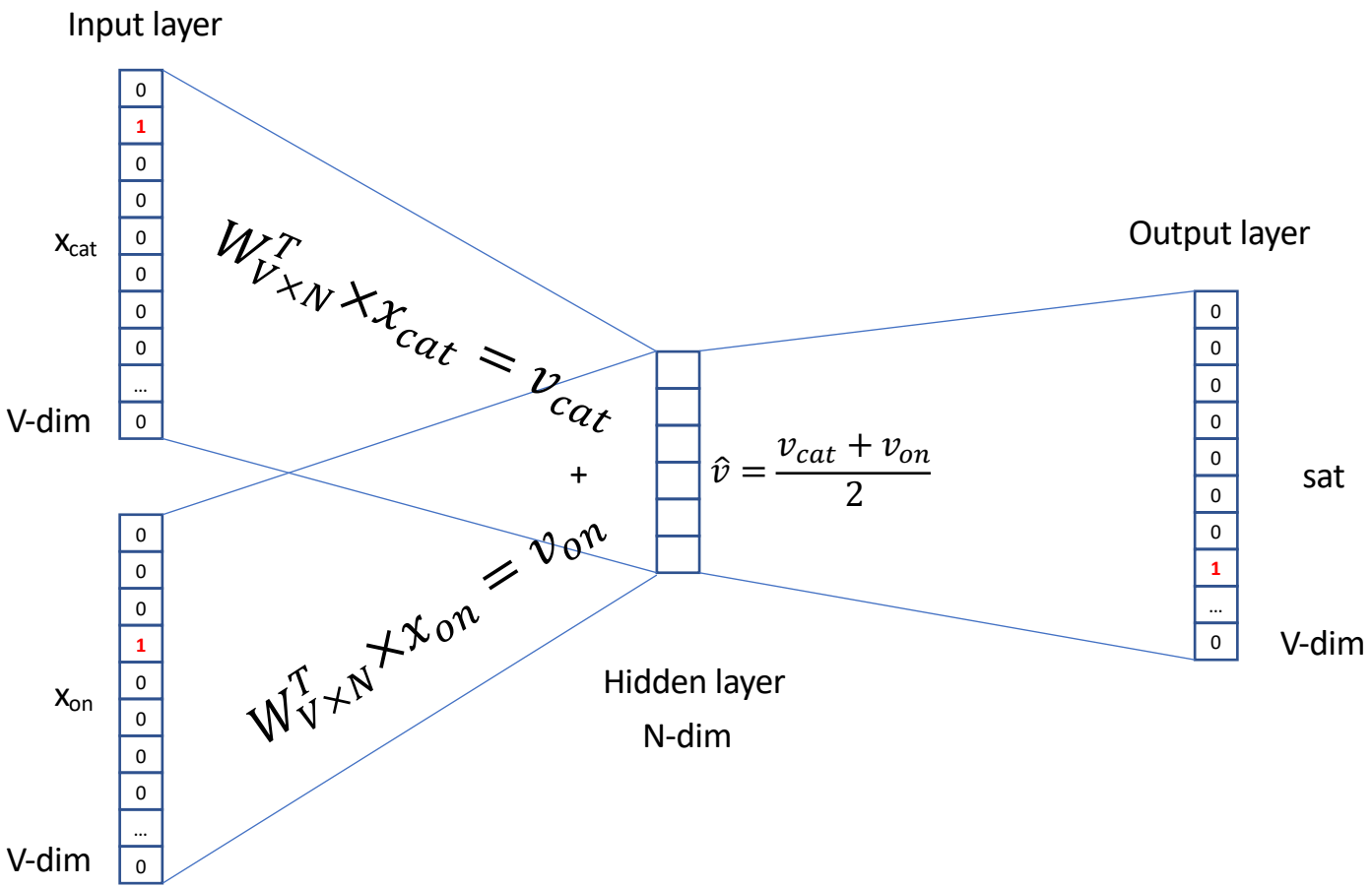


Word2Vec: continuous bag-of-words



http://www.cs.ucr.edu/~vagelis/classes/CS260/

Word2Vec: continuous bag-of-words



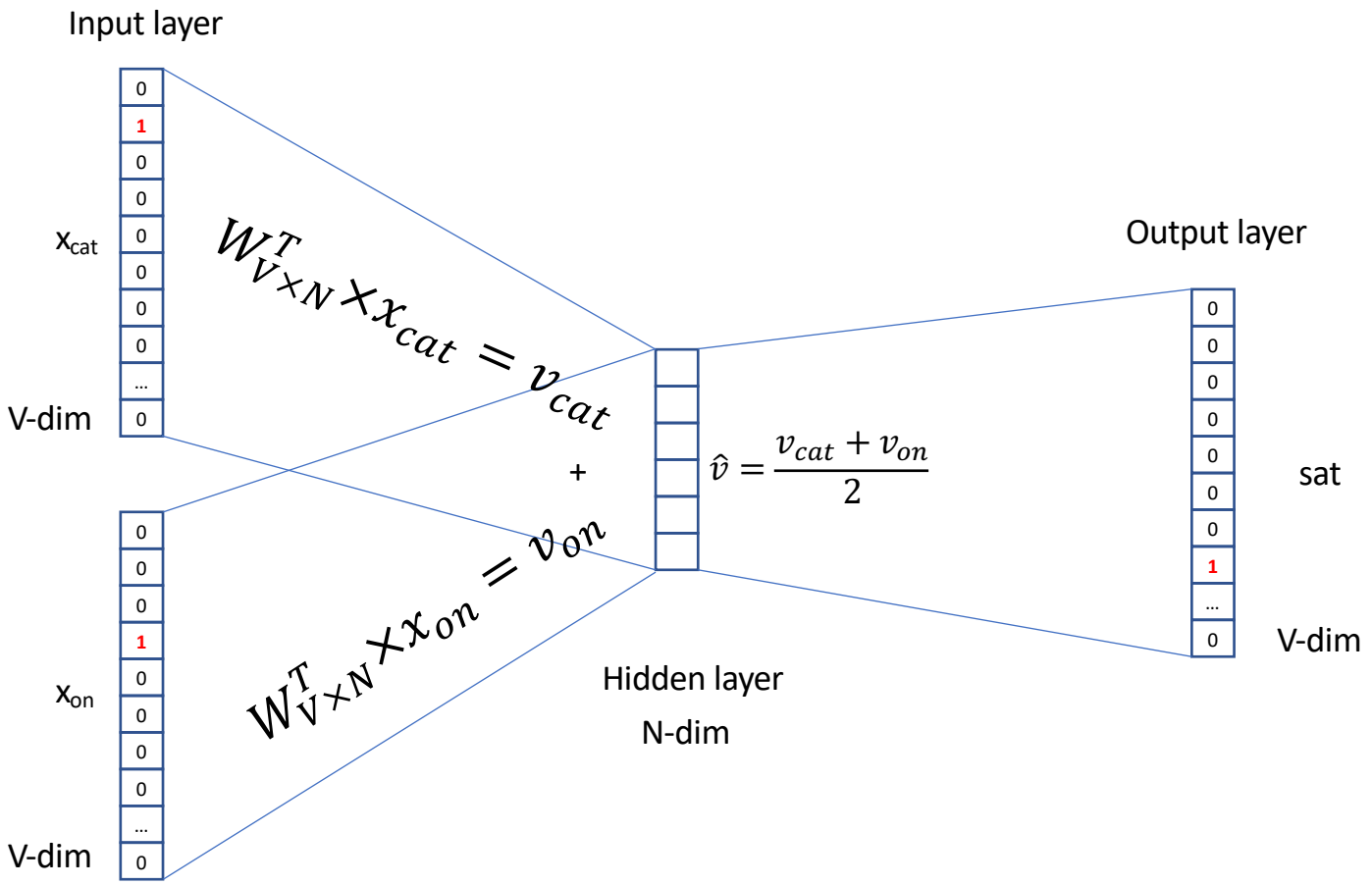
$W^T_{V\times N}$

0.1	2.4	1.6	1.8	0.5	0.9	…	…	…	3.2
0.5	2.6	1.4	2.9	1.5	3.6	…	…	…	6.1
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
0.6	1.8	2.7	1.9	2.4	2.0	…	…	…	1.2

$\times x_{cat} = v_{cat}$

0	×	2.4
1	=	2.6
0		…
0		…
0		1.8
0		
0		
…		
0		

Word2Vec: continuous bag-of-words



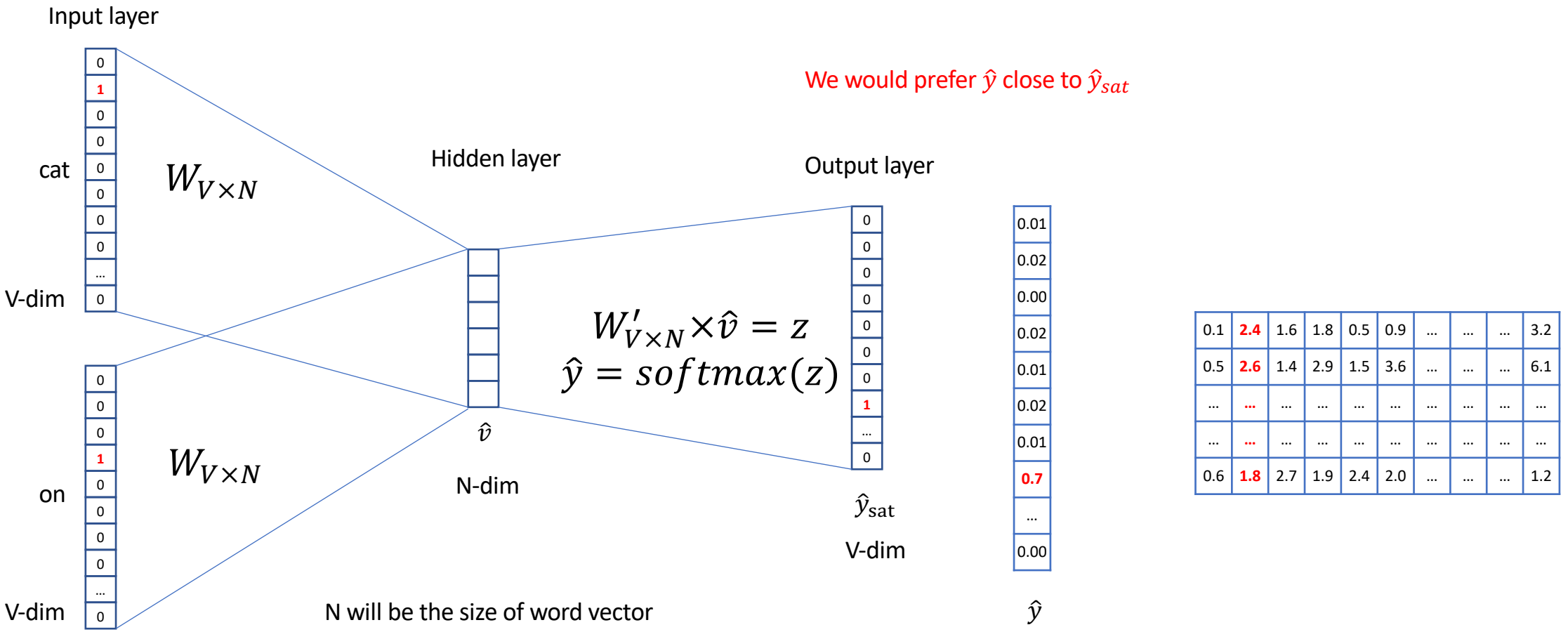
$W^T_{V\times N}$

0.1	2.4	1.6	**1.8**	0.5	0.9	…	…	…	3.2
0.5	2.6	1.4	**2.9**	1.5	3.6	…	…	…	6.1
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
0.6	1.8	2.7	**1.9**	2.4	2.0	…	…	…	1.2

$\times x_{on} = v_{on}$

0	×	**1.8**
0		**2.9**
0		…
1	=	…
0		**1.9**
0		
0		
…		
0		

Word2Vec: continuous bag-of-words



0.1	2.4	1.6	1.8	0.5	0.9	…	…	…	3.2
0.5	2.6	1.4	2.9	1.5	3.6	…	…	…	6.1
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
0.6	1.8	2.7	1.9	2.4	2.0	…	…	…	1.2

http://www.cs.ucr.edu/~vagelis/classes/CS260/

Word analogies

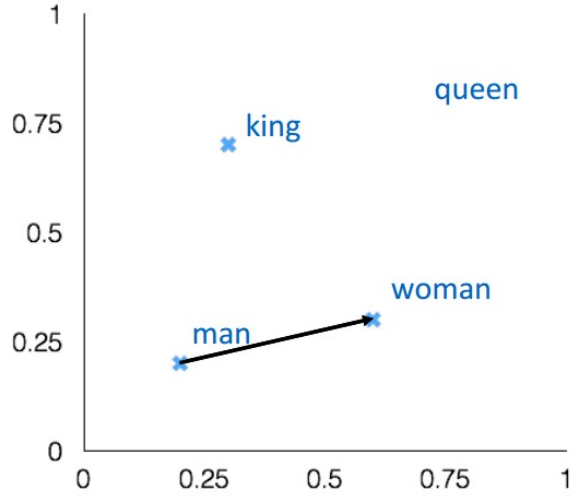
Test for linear relationships, examined by Mikolov et al. (2014)

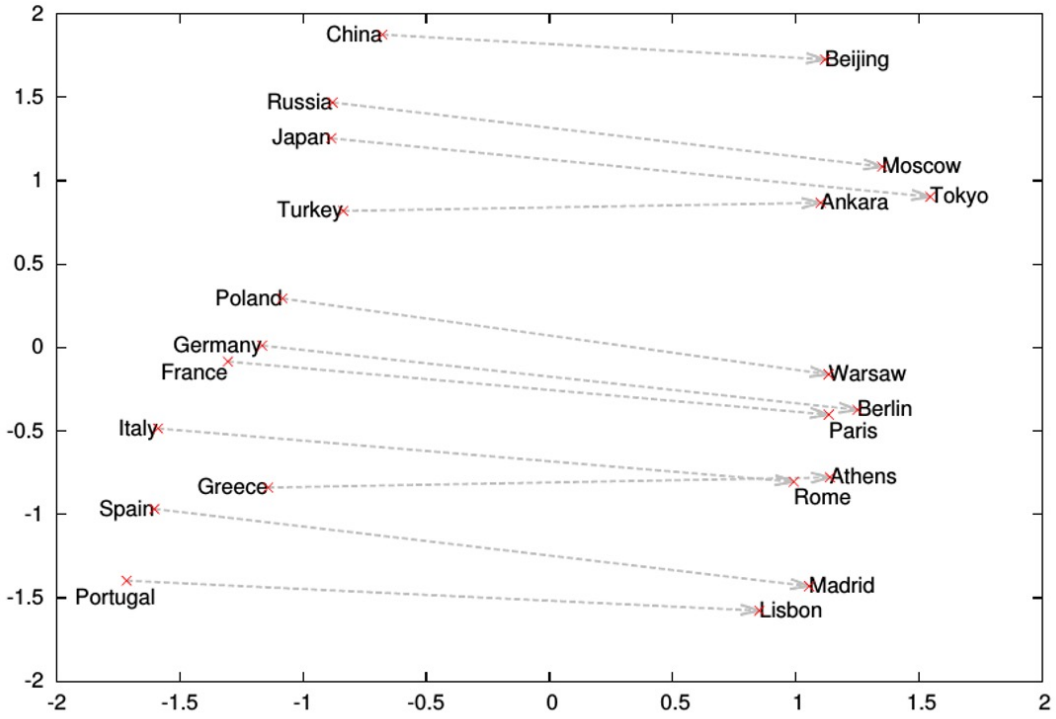
a:b :: c:?

$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]

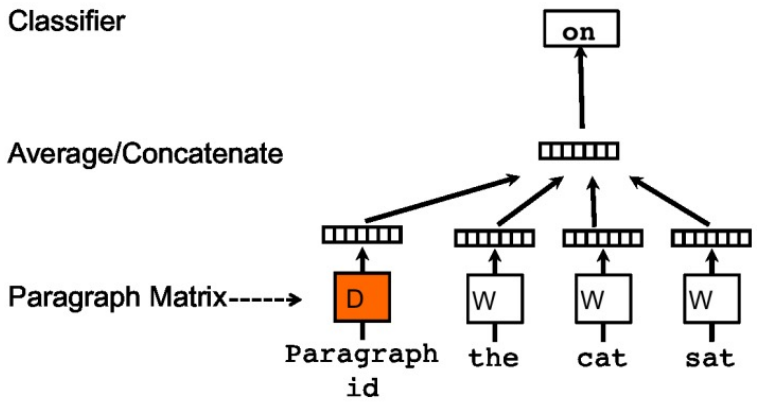


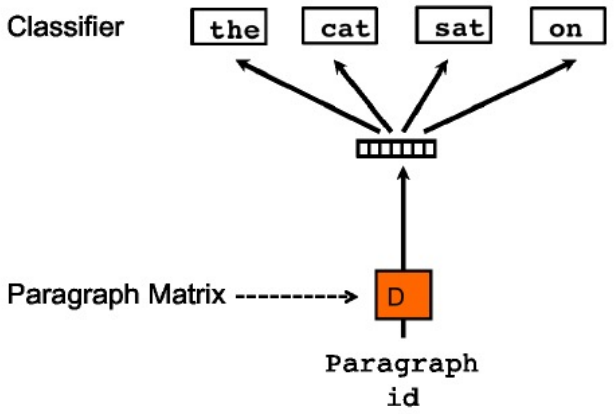


http://www.cs.ucr.edu/~vagelis/classes/CS260/

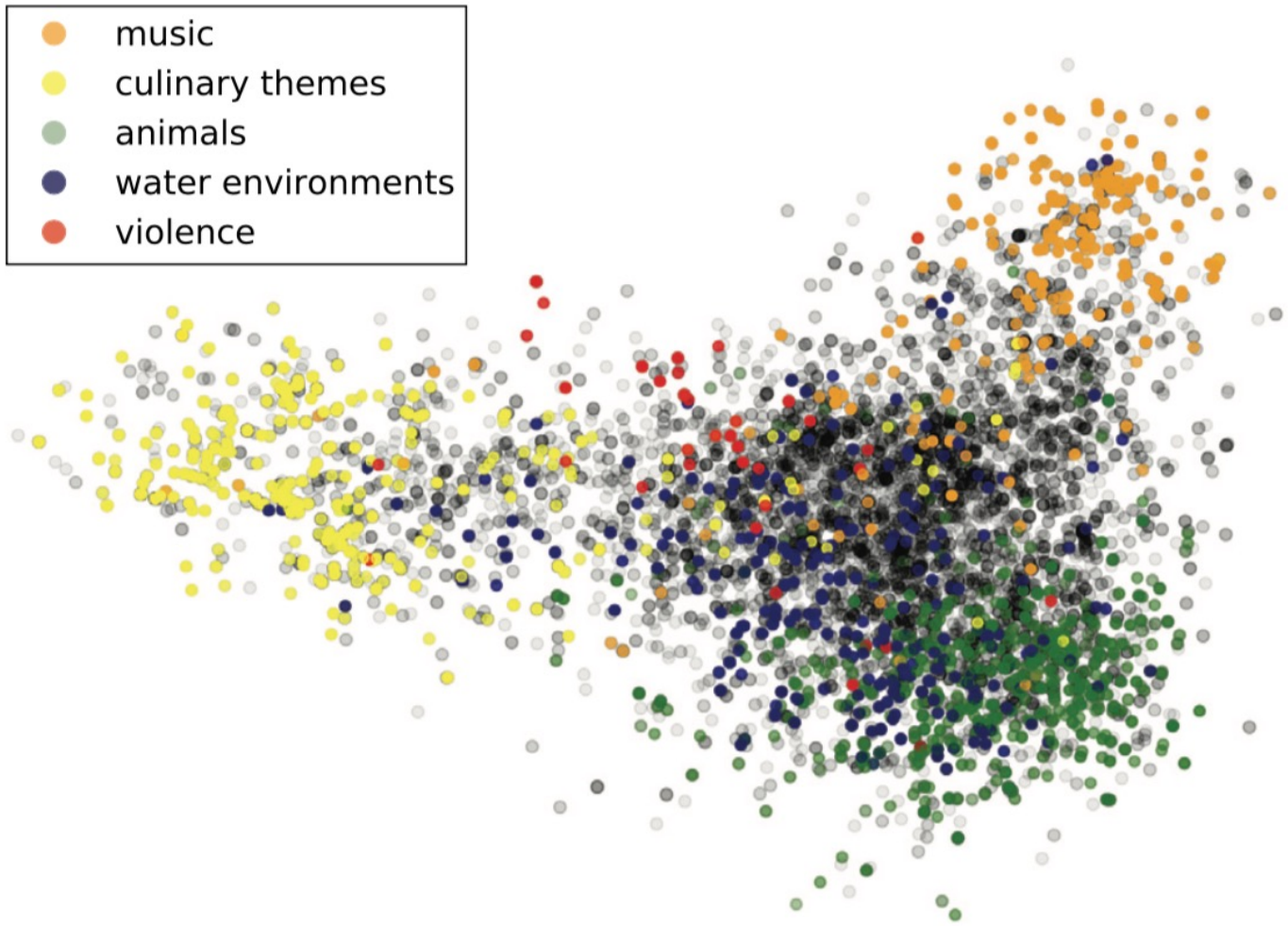
From Word2Vec to Doc2Vec

- Paragraph vector (2014, Quoc Le, Mikolov)
  - Extend word2vec to text level
  - Also two models: add paragraph vector as the input





Document embedding visualization



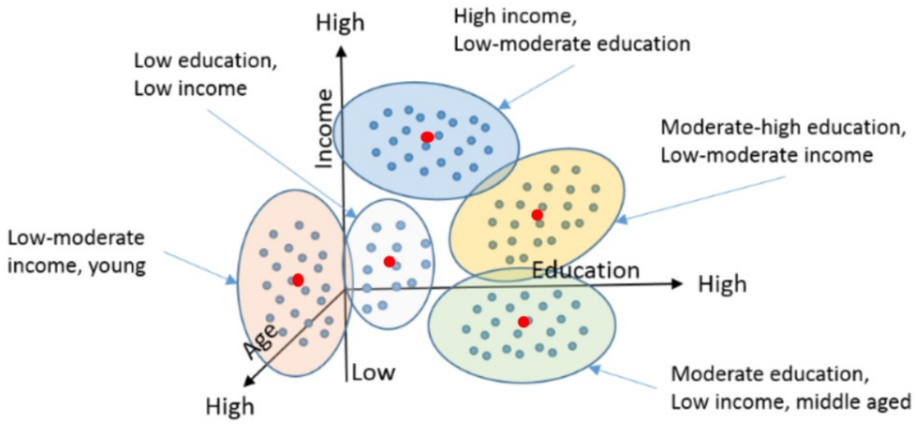
Siamese Recurrent Architectures for Learning Sentence Similarity, AAAI'16

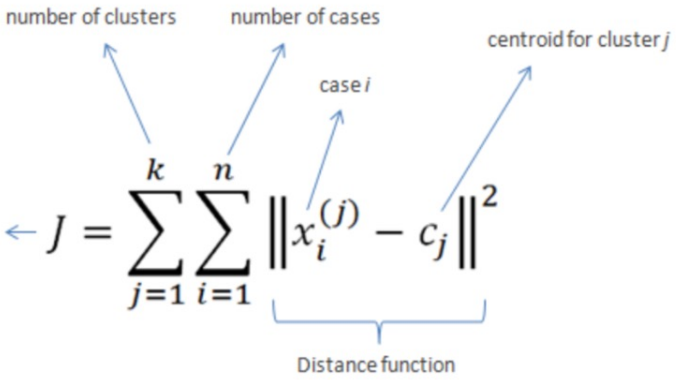
Our tasks

- Preprocess the dataset
  - Clean the data and build the vocabulary
  - Visualize the statistics of the dataset
  - Baseline document features
    - Bag-of-words; TF-IDF Model
- Topic Modeling
  - Train a LDA model with given topic#
  - Visualize different topics
- Vector representation of documents
  - Train a Doc2Vec model
  - Visualize word embedding and document embedding
- Comparison between different document representations
  - Document clustering

Document clustering

- Document features
  - Two settings: 1) whole vocabulary; 2) Top 2k words
  - BoA; tf-idf; topic; d2v
  - Validate your results by NMI





Recommended tools

- BoA, Tf-idf, LDA
  - Gensim; sklearn
- Doc2Vec
  - Gensim
- Visualization
  - t-SNE [1] or PCA
  - matplotlib; seaborn; visdom; tensorboard
  - Matlab is also powerful
- Document clustering
  - sklearn.cluser.Kmeans
  - sklearn.metrics

Project requirement

- Implement all the tasks we introduced today
- Propose one task/method by yourself
- Demos
  - All your visualization results should be obtained by running demos directly
- Reports
  - 4-8 pages pdf
    - Latex is recommended
  - All the results should be posted
  - Analysis