資料分析與學習基石
(*Fundamental of Data Analytics and Learning*)

-- Unsupervised Learning (2/2)

Hung-Yu Kao (高宏宇)
***I**ntelligent **K**nowledge **M**anagement Lab*

Master Program of Artificial Intelligence
Institute of Medical Informatics,
Dept. of Computer Science and Information Engineering
National Cheng Kung University, Tainan, Taiwan

Unsupervised Learning

- Learning without a teacher
- Self-organization

Clustering

- K-mean
- Hierarchical clustering
- DBSCAN
- …

Anomaly Detection

- Outlier detection

Neural Network

- **Autoencoder**
- Generative Adversarial Network (GAN)
- SOM

Learning approach

- Expectation Maximization (EM)
- PCA, MF, SVD

Supervised Learning vs Unsupervised Learning

01

What happens when our labels are noisy?

- Missing values.
- Labeled incorrectly.

02

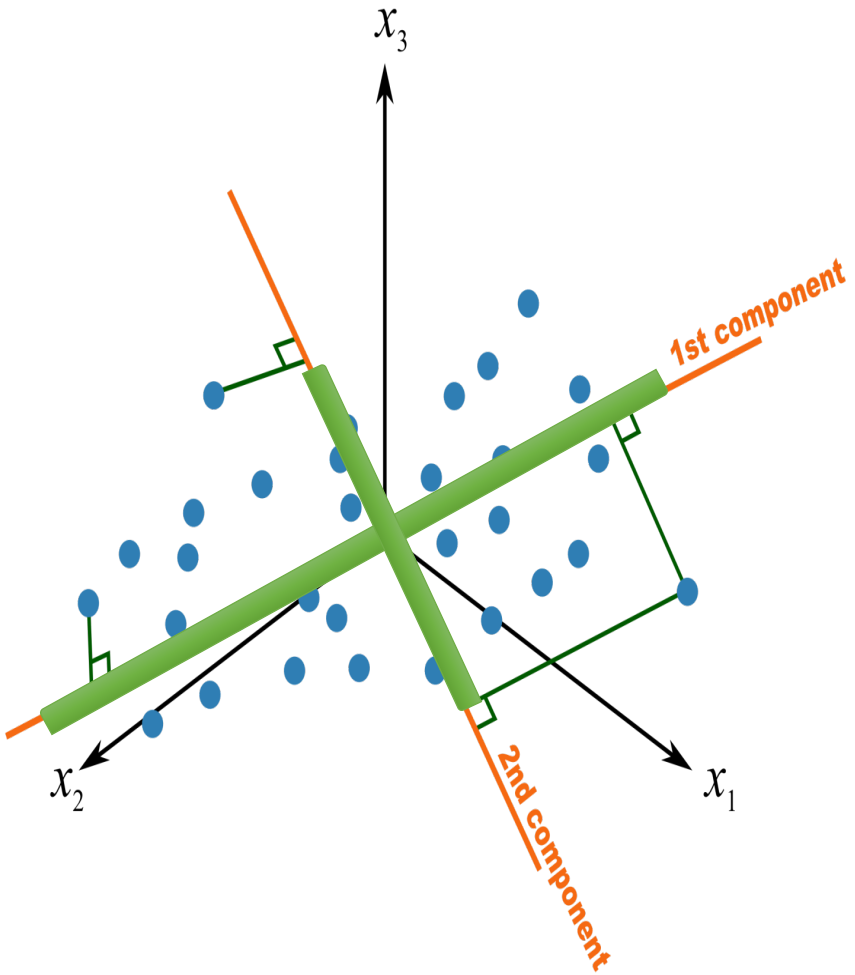
What happens where we don't have labels for training **at all**?

(or implicit labels)

PCA – Principal Component analysis

- Statistical approach for data compression and visualization
- Invented by Karl Pearson in 1901

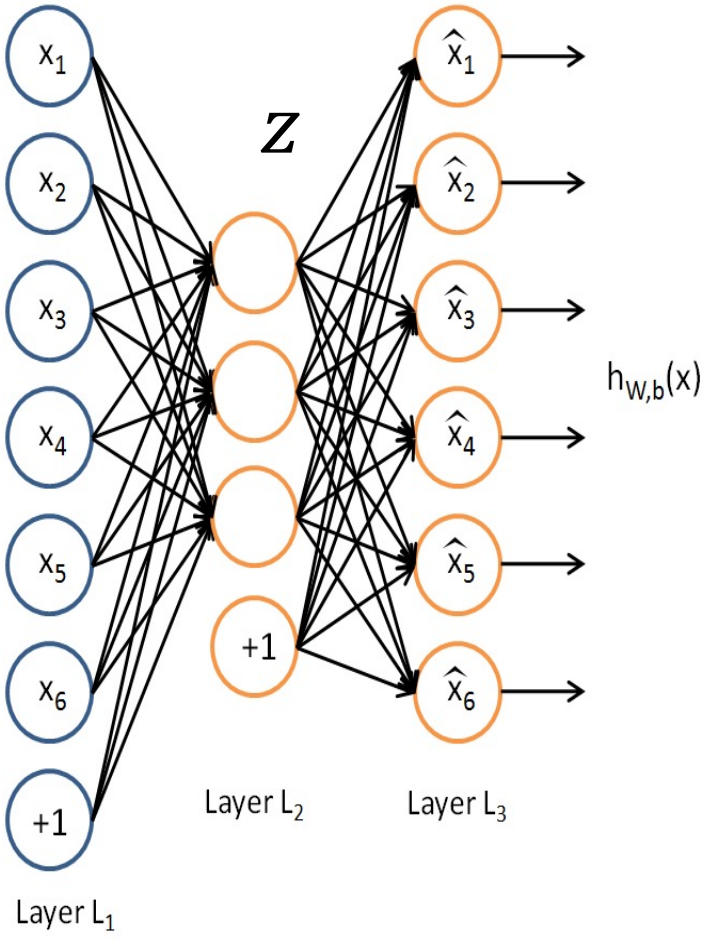
- Weakness: linear components only.



Traditional Autoencoder

- Unlike the **PCA** now we can use activation functions to achieve non-linearity.

- It has been shown that an AE without activation functions achieves the **PCA** capacity.



Uses

- The autoencoder idea was a part of NN history for decades (LeCun et al, 1987).

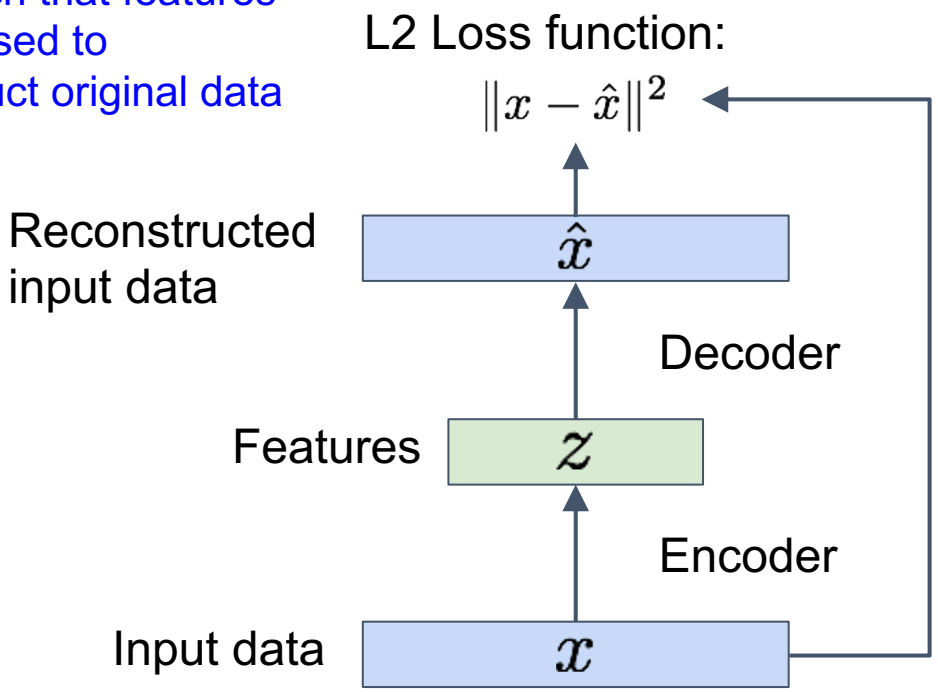
- Traditionally an autoencoder is used for **dimensionality reduction** and **feature/representation learning**.

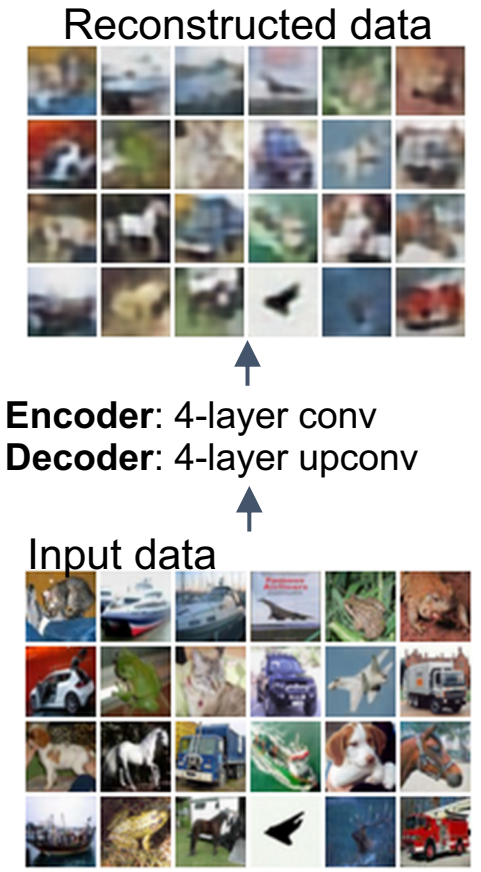
- Recently, the connection between autoencoders and latent space modeling has brought autoencoders to the front of generative modeling.

https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html - By Andrej Karpathy

Autoencoders

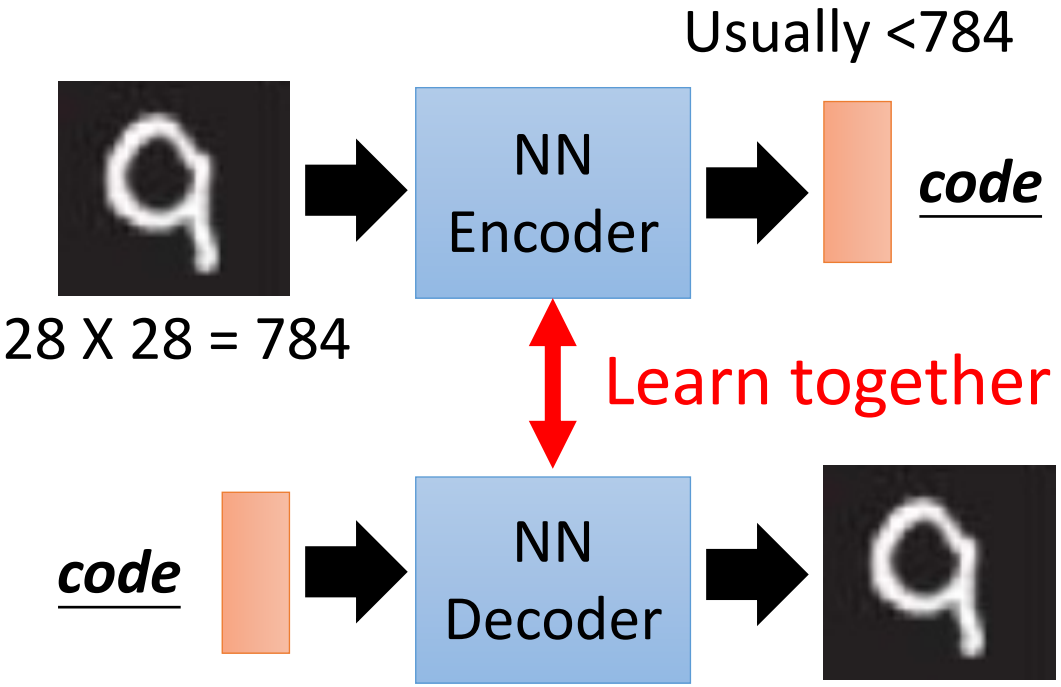
Train such that features can be used to reconstruct original data





Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

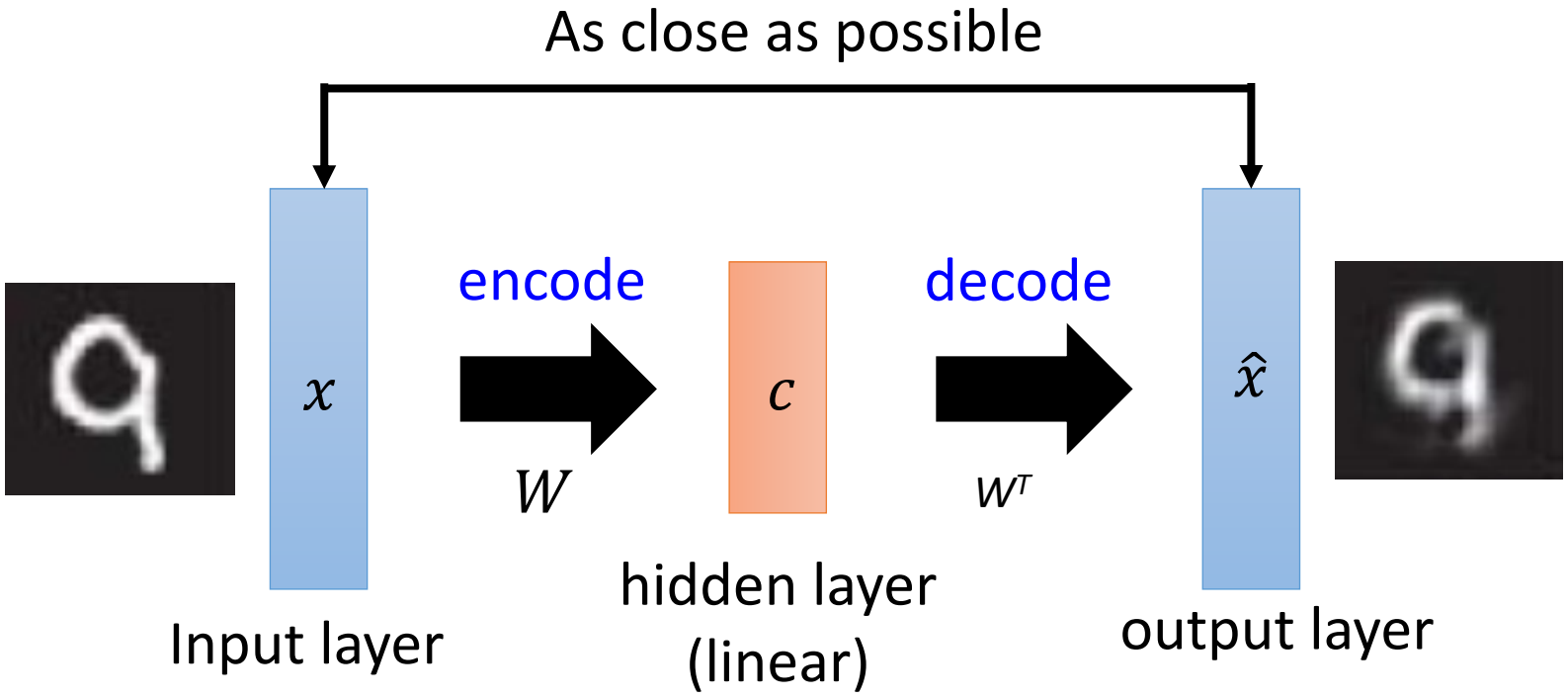
Autoencoder



Compact representation of the input object

Can reconstruct the original object

Recap: PCA

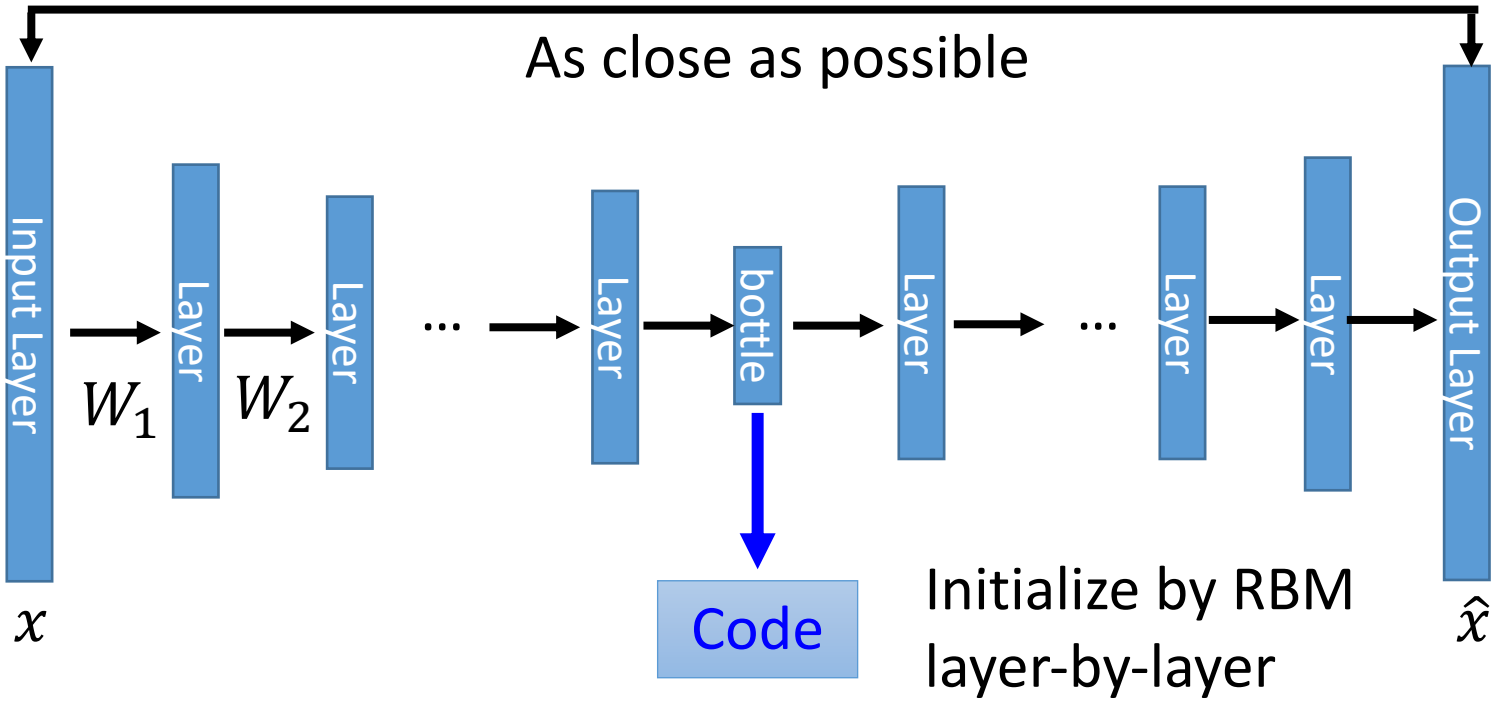


Output of the hidden layer is the code

Deep Autoencoder

Symmetric is not necessary.

- Of course, the auto-encoder can be deep



Deep Autoencoder

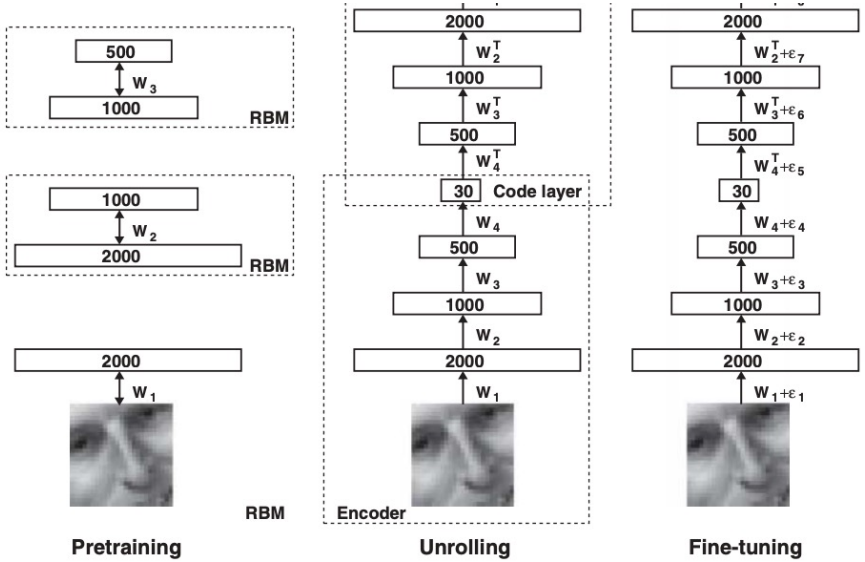
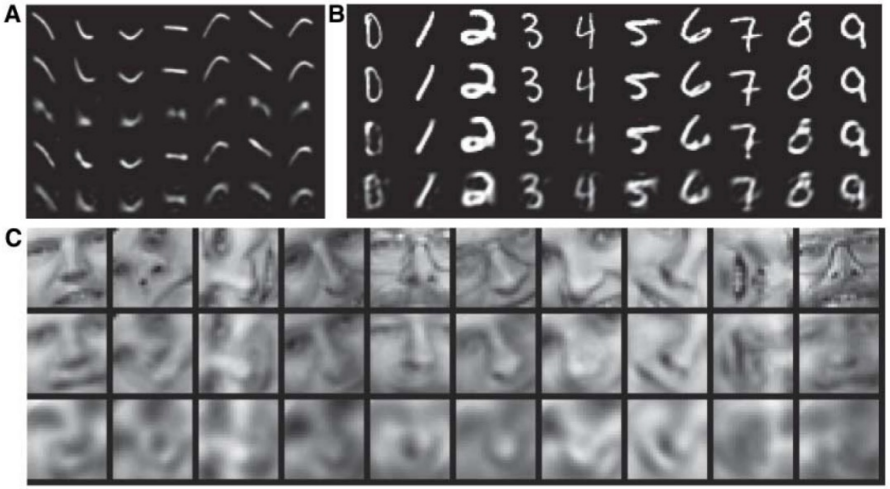


Fig. 1. Pretraining consists of learning a stack of restricted Boltzmann machines (RBMs), each having only one layer of feature detectors. The learned feature activations of one RBM are used as the "data" for training the next RBM in the stack. After the pretraining, the RBMs are "unrolled" to create a deep autoencoder, which is then fine-tuned using backpropagation of error derivatives.



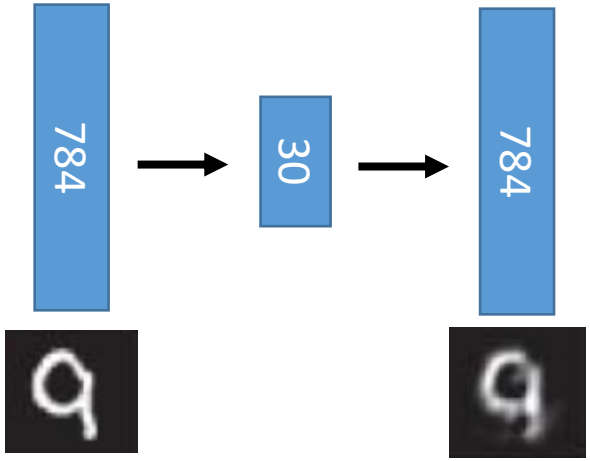
Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

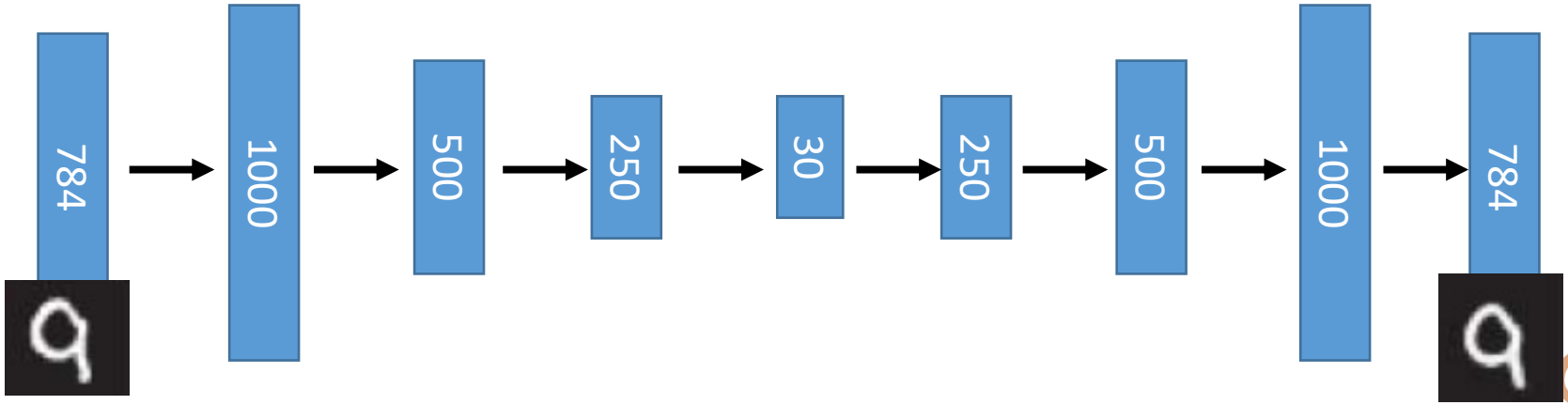
Deep Autoencoder

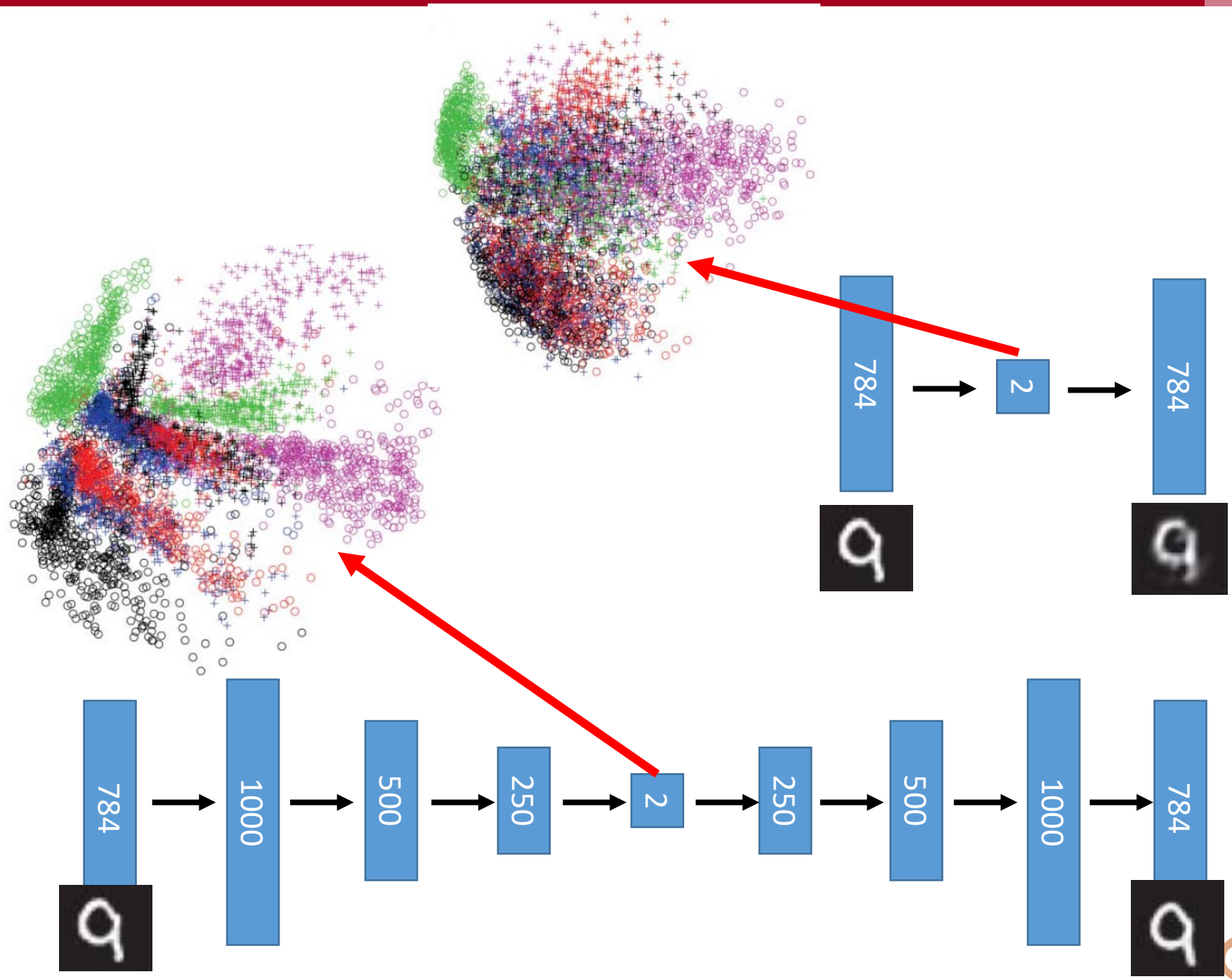
Original Image

PCA

Deep Auto-encoder





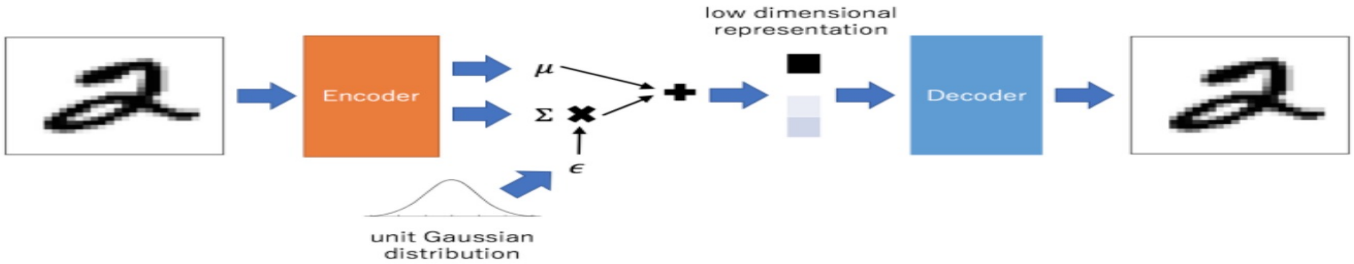

784
2
784
784
1000
500
250
2
250
500
1000
784

Variational Auto Encoders: Generating Data

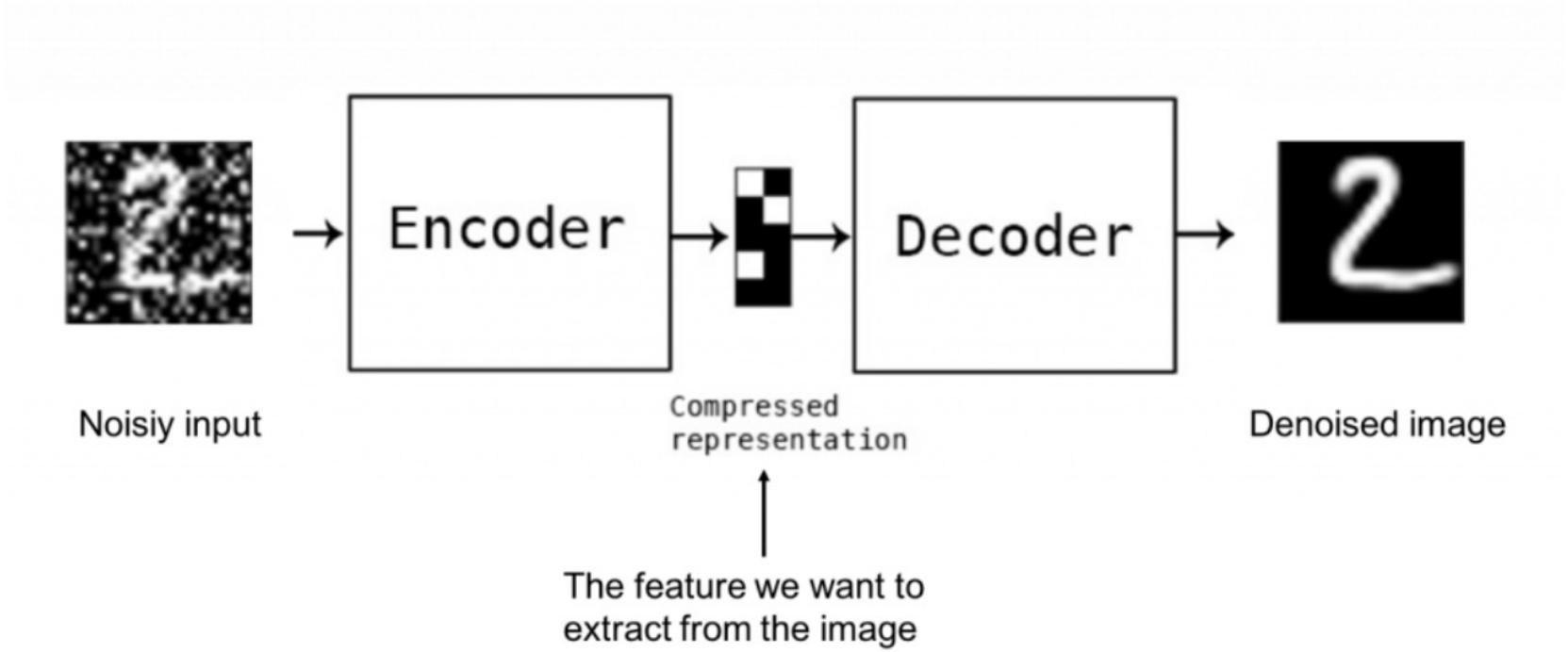
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

32x32 CIFAR-10

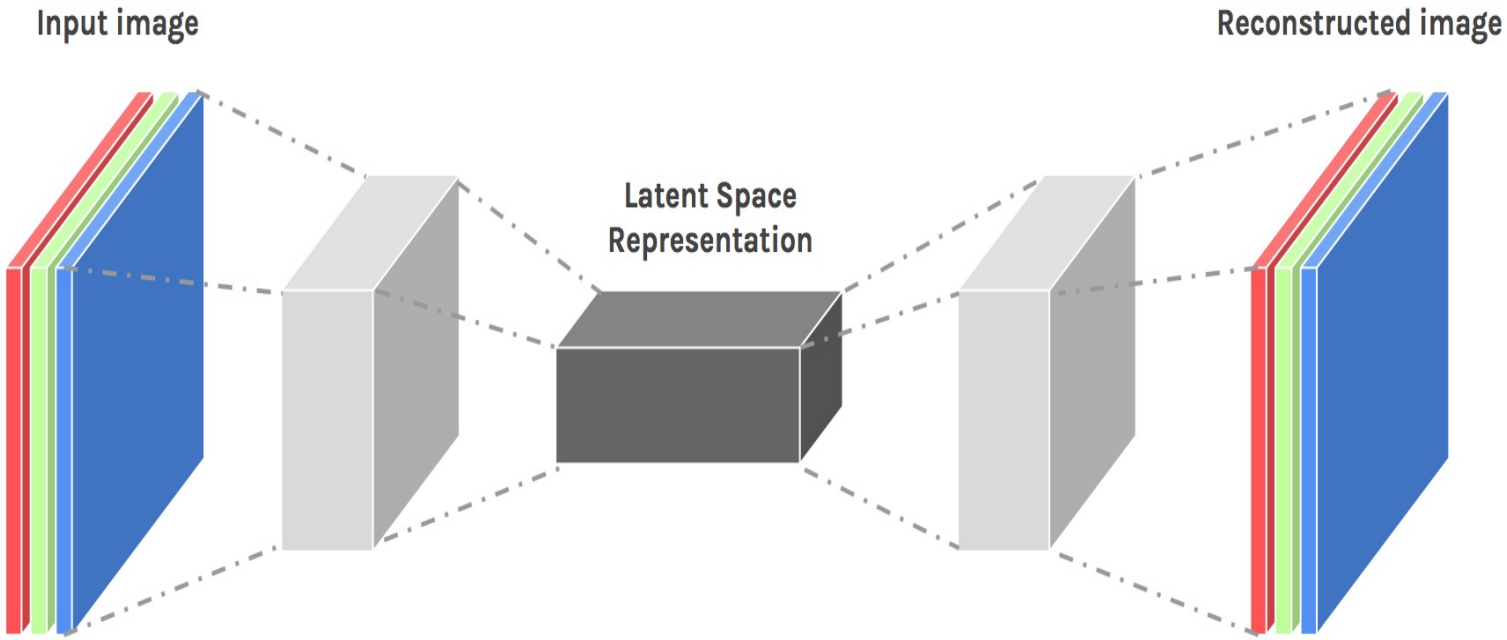
Labeled Faces in the Wild



Denoising AE (DAE)

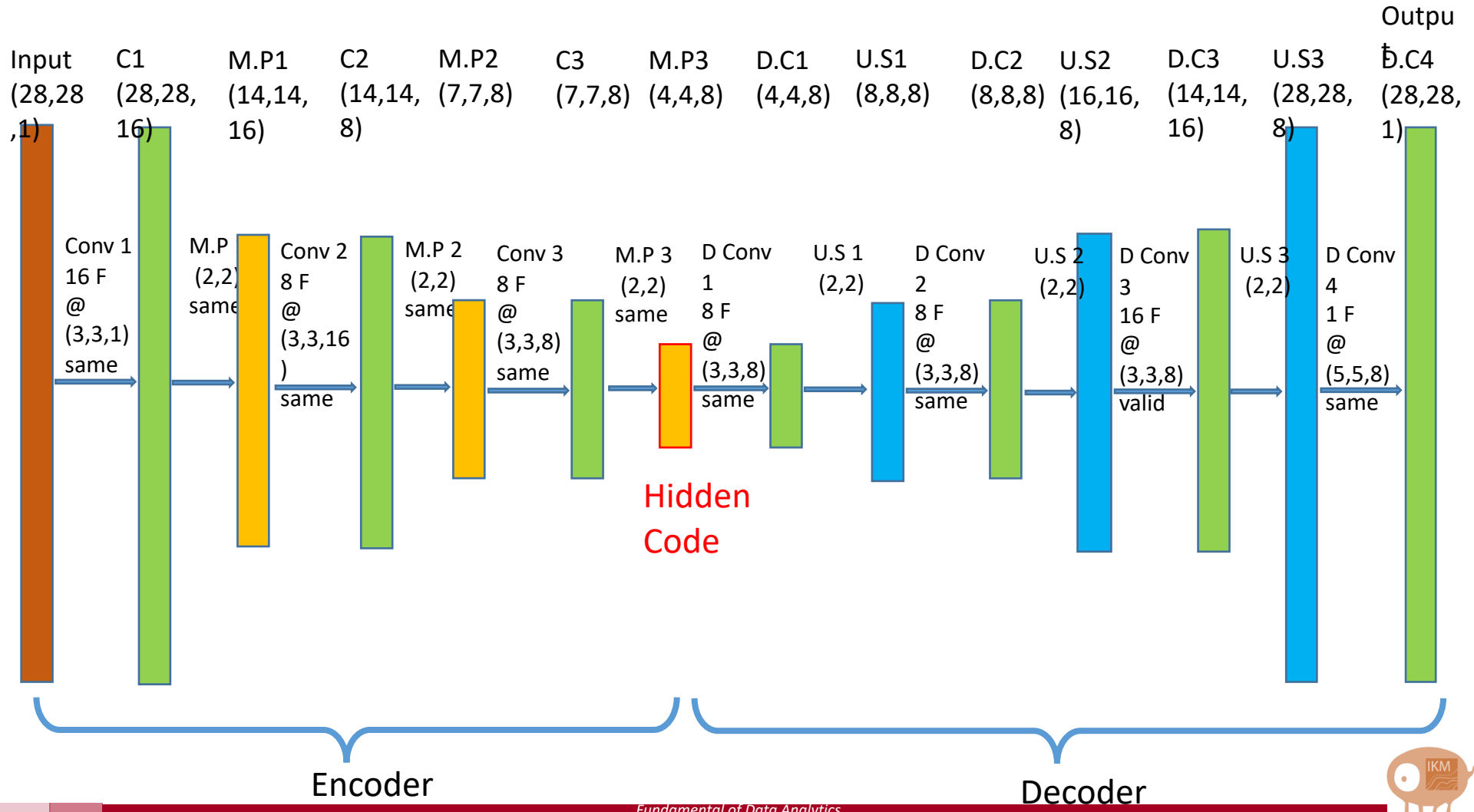


Convolutional AE



* Input values are normalized
* All of the conv layers activation functions are relu except for the last conv which is sigm

Convolutional AE



Representation Learning

How do we represent the meaning of a word?

- Dictionary based solutions
- WordNet: a resource containing list of synonyms sets and hypernyms ("is a" relationships)

e.g. synonym sets containing "good":

```
from nltk.corpus import wordnet as wn
for synset in wn.synsets("good"):
    print "(%s)" % synset.pos(),
    print ", ".join([l.name() for l in synset.lemmas()])
```

```
(adj) full, good
(adj) estimable, good, honorable, respectable
(adj) beneficial, good
(adj) good, just, upright
(adj) adept, expert, good, practiced,
proficient, skillful
(adj) dear, good, near
(adj) good, right, ripe
…
(adv) well, good
(adv) thoroughly, soundly, good
(n) good, goodness
(n) commodity, trade good, good
```

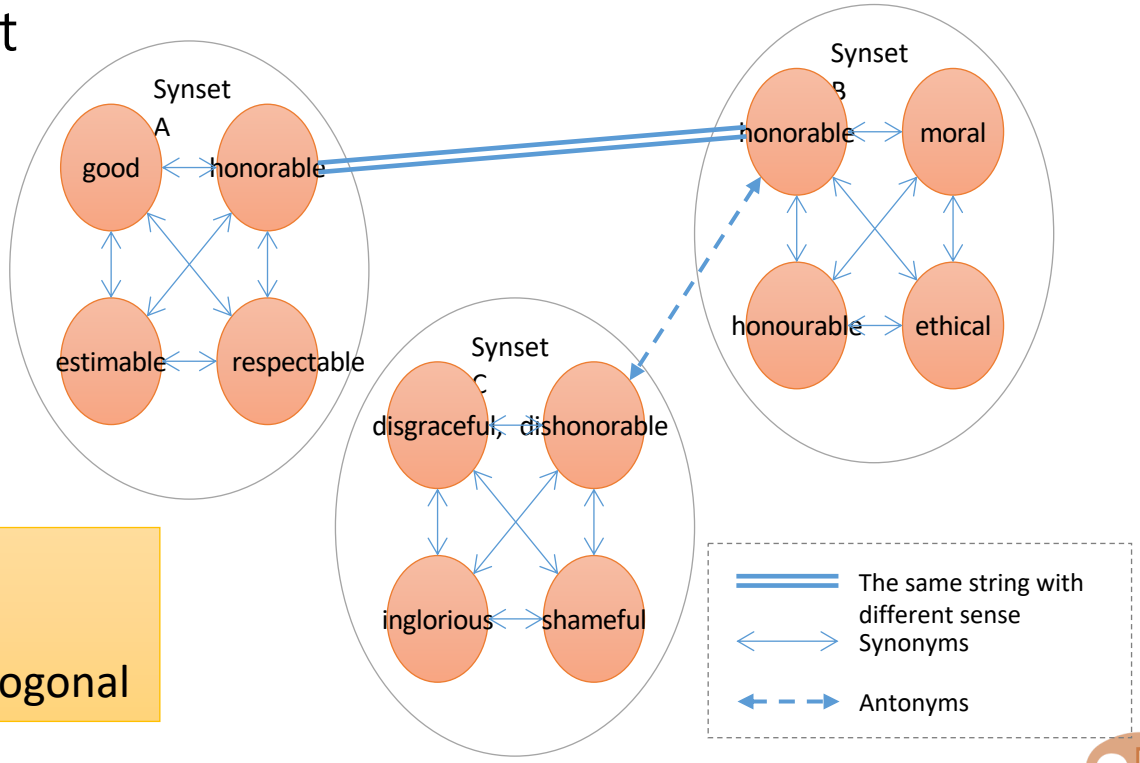
e.g. hypernyms of "panda":

```
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')]
```

How do we represent the meaning of a word?

- Issues
  - context is not considered
  - New words missing
  - Maintenance cost



Hard to measure the similarity!
-- one-hot vector
-- every different words are orthogonal

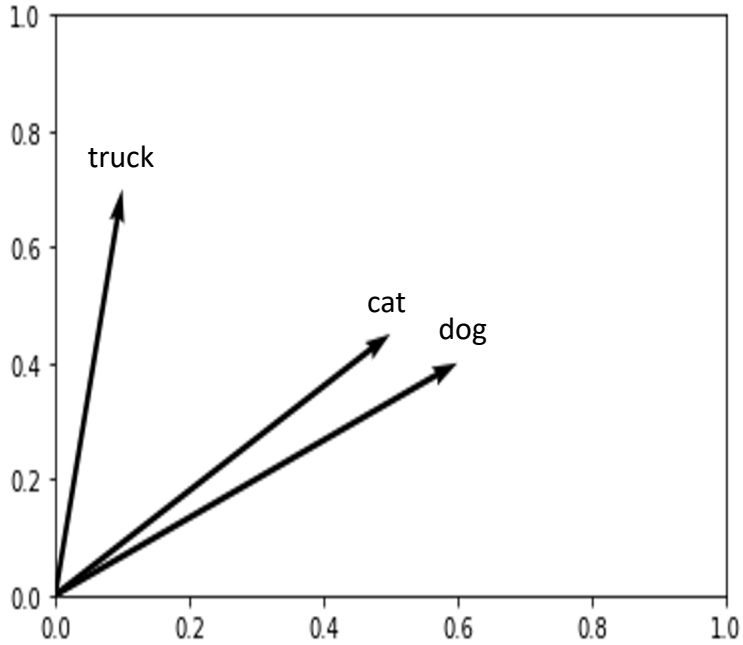
Solution: Continuous, Distributed Representations

By continuous we mean real-valued $\mathbb{R}^2$

Distributed means a vector in a space, like

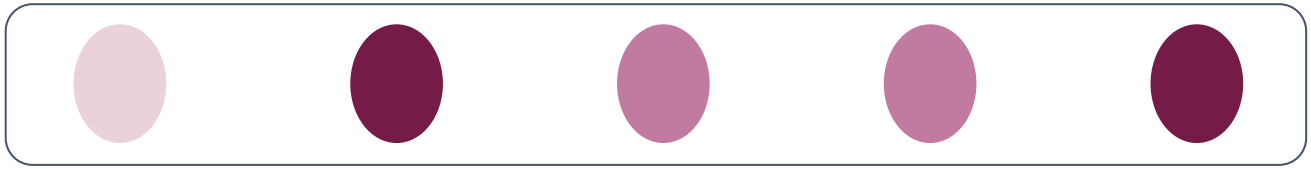
$$\text{dog} = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}$$

$$\text{cat} = \begin{bmatrix} 0.5 \\ 0.4 \end{bmatrix}$$



Connection to Neural Networks

A distributed representation can be thought of as a pattern of activity "distributed" across neurons. These patterns are taken to represent things - like concepts or words.



dog =

$$\begin{bmatrix} 0.1 & 0.3 & 0.2 & 0.2 & 0.3 \end{bmatrix}$$

Deep Learning with Text Today

Input

1. Life: 1844–1900

In the small German village of Röcken bei Lützen, located in a rural farmland area southwest of Leipzig, Friedrich Wilhelm Nietzsche was born at approximately 10:00 a.m. on October 15, 1844. The date coincided with the 49th birthday of the Prussian King, Friedrich Wilhelm IV, after whom Nietzsche was named, and who had been responsible for Nietzsche's father's appointment as Röcken's town minister.

Nietzsche's uncle and grandfathers were also Lutheran ministers, and his paternal grandfather, Friedrich August Ludwig Nietzsche (1756–1826), was further distinguished as a Protestant scholar, one of whose books (1796) affirmed the "everlasting survival of Christianity." Nietzsche's grandparents on both sides were from the Province of Saxony, with his paternal grandfather, paternal grandmother (Erdmuthe Dorothea Krause, 1778–1856), maternal grandfather (David Ernst Ohler, 1787–1859) and maternal grandmother (Johanna Elisabeth Wilhelmine Hahn, 1794–1876) having been born respectively in the small towns of Bibra (just south of Jena), Reichenbach (southeast of Jena), Zeitz (between Jena and Leipzig), and Wehlitz (just northwest of Leipzig).

When Nietzsche was nearly 5 years old, his father, Karl Ludwig Nietzsche (1813–1849) died from a brain ailment (July 30, 1849) and the death of Nietzsche's two-year-old brother, Ludwig Joseph, traumatically followed six months later (January 4, 1850). Having been living only yards away from Röcken's church in the house reserved for the pastor and his family, the Nietzsche family left their home soon after Karl Ludwig's death. They moved to nearby Naumburg an der Saale, where Nietzsche (called "Fritz" by his family) lived with his mother, Franziska (1826–1897), his grandmother, Erdmuthe, his father's two sisters, Auguste and Rosalie (d. 1855 and 1867, respectively), and his younger sister, Therese Elisabeth Alexandra (1846–1935).

Why vectors as input???

This is a key step

$$\begin{bmatrix} 0.1 \\ 2.4 \\ 3.2 \\ 1.7 \end{bmatrix}$$

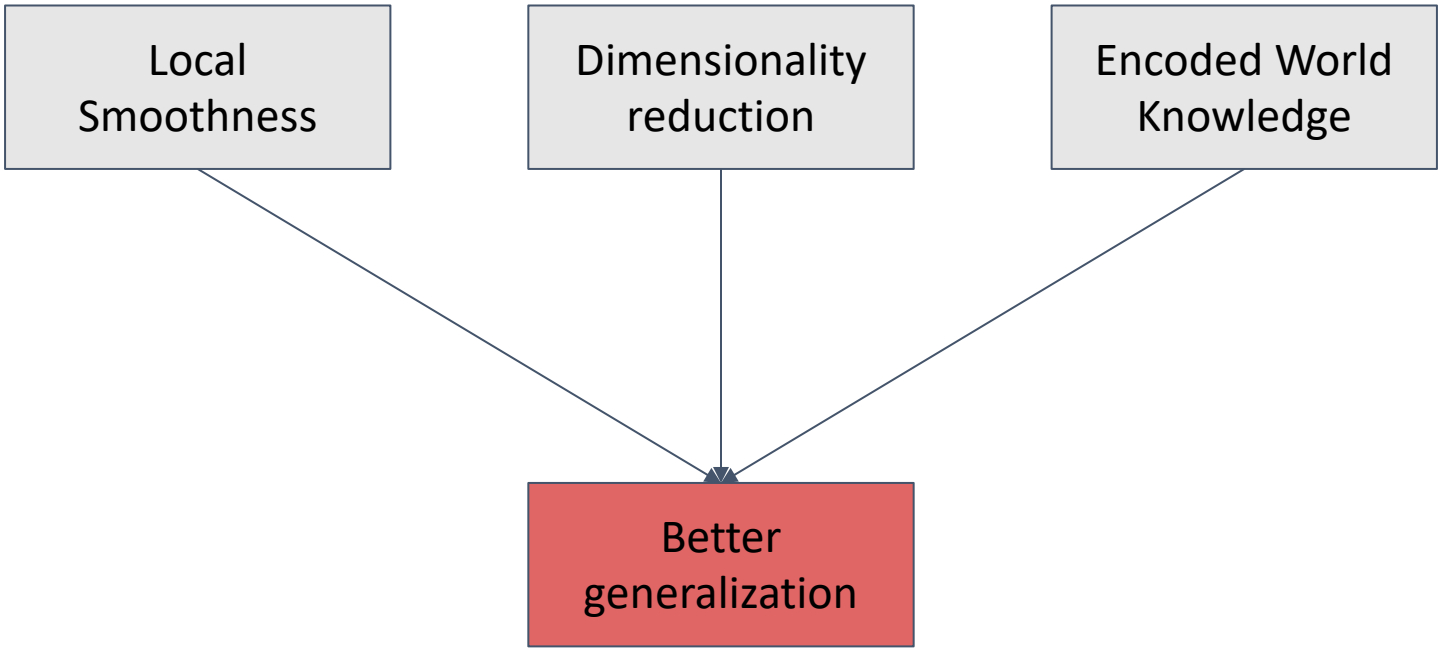
Model

$$f(\mathbf{x})$$

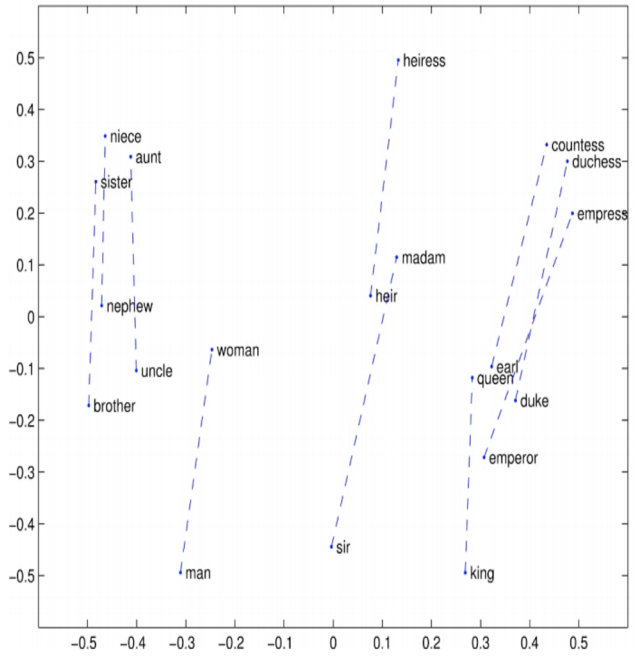
Output

topic, author, sentiment, etc...

Sketch of Answer

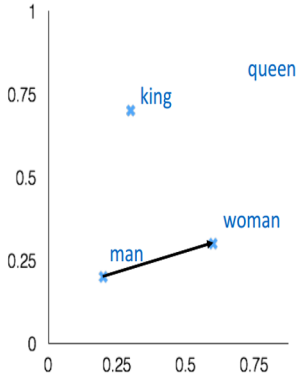


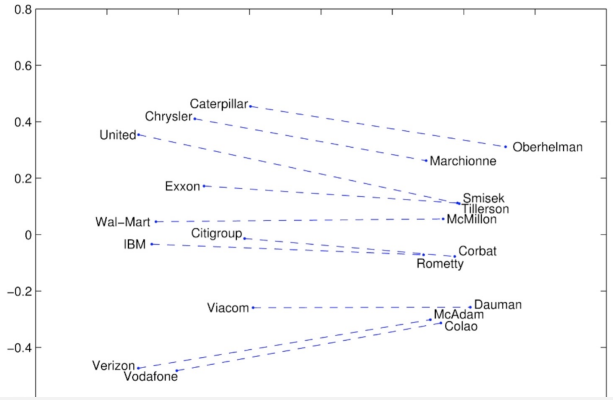
Learning Vector Representation of Words

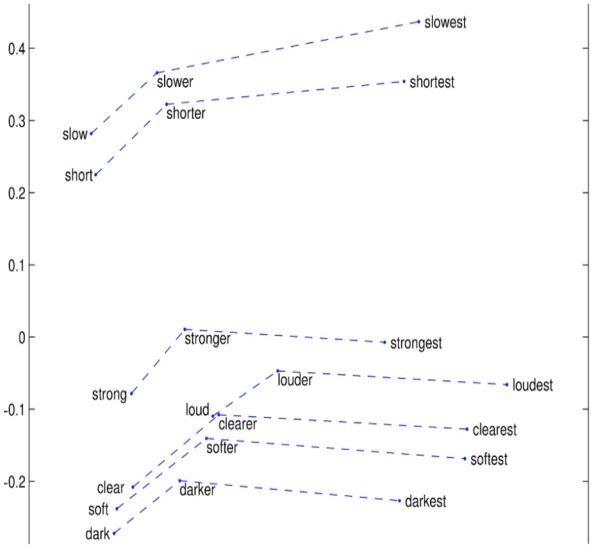


man:woman :: king:?

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







0. *frog*
1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

3. litoria

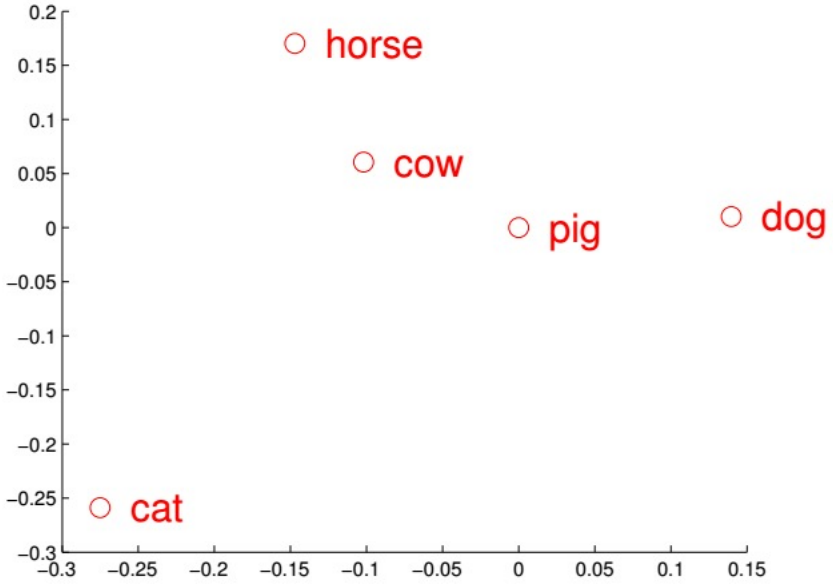
4. leptodactylidae

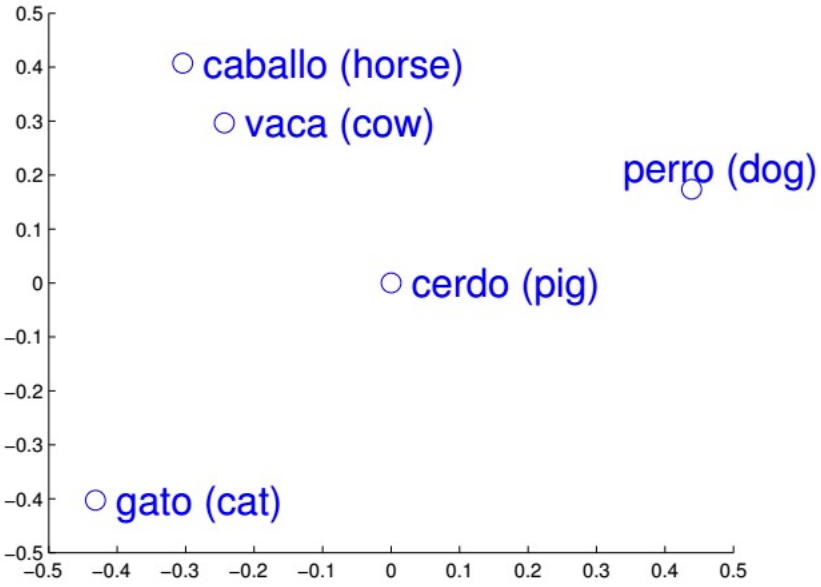
5. rana

7. eleutherodactylus

Representation Vector Usage

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au





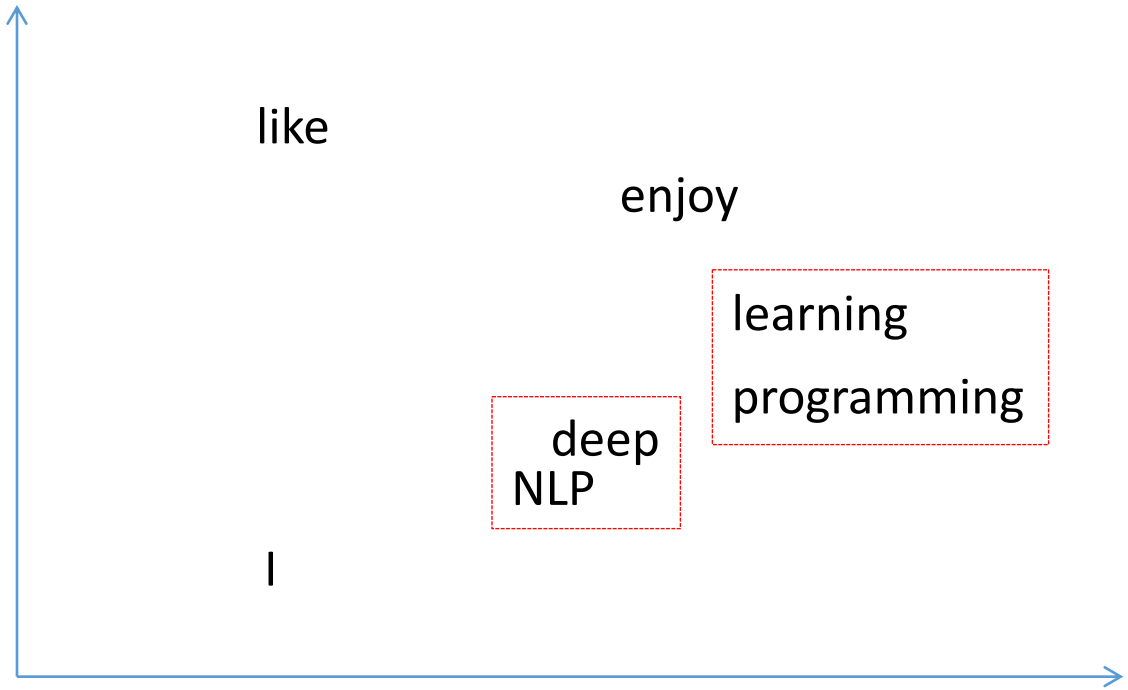
Simple example for word co-occurrence usage

- Training corpus
  - “I like deep learning.” , “I like NLP.” , “I enjoy programming."

cooccurence	I	like	enjoy	deep	learning	NLP	programming
I	0	2	1	0	0	0	0
like	2	0	0	1	0	1	0
enjoy	1	0	0	0	0	0	1
deep	0	1	0	0	1	0	0
learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
programming	0	0	1	0	0	0	0

Simple example for word co-occurrence usage

- SVD on this co-occurrence matrix
  - Reduce dimension
- Use the 2 biggest singular value to represent words



Latent Semantic Indexing(LSI)

- A statistical technique
- Uses linear algebra technique called *singular value decomposition (SVD)*
  - attempts to estimate the hidden structure that generates terms given concepts
  - discovers the most important associative patterns between words and concepts
- Data driven
  - A large collection of sentences or documents is employed

LSI and Text Documents

- Let **X** denote a term-document matrix

$$X = [x_1 \ldots x_n]^T$$

  - each row is the vector-space representation of a document
  - each column contains occurrences of a term in each document in the dataset

- Latent semantic indexing
  - compute the SVD of ***X***:

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

    - **Σ** - singular value matrix *(diagonal matrix)*
  - set to zero all but largest *K* singular values - $\hat{\Sigma}$
  - obtain the reconstruction of ***X*** by:

$$\hat{X} = U\hat{\Sigma}V^T$$

LSI Example

- A collection of documents:

Linux OS

d1: Indian government goes for open-source software
d2: Debian 3.0 Woody released
d3: Wine 2.0 released with fixes for Gentoo 1.4 and Debian 3.0
d4: gnuPOD released: iPOD on Linux… with GPLed software
d5: Gentoo servers running at open-source mySQL database
d6: Dolly the sheep not totally identical clone
d7: DNA news: introduced low-cost human genome DNA chip
d8: Malaria-parasite genome database on the Web
d9: UK sets up genome bank to protect rare sheep breeds
d10: Dolly's DNA damaged

Genome news

LSI Example

- The term-document matrix X^T

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
open-source	1	0	0	0	1	0	0	0	0	0
software	1	0	0	1	0	0	0	0	0	0
Linux	0	0	0	1	0	0	0	0	0	0
released	0	1	1	1	0	0	0	0	0	0
Debian	0	1	1	0	0	0	0	0	0	0
Gentoo	0	0	1	0	1	0	0	0	0	0
database	0	0	0	0	1	0	0	1	0	0
Dolly	0	0	0	0	0	1	0	0	0	1
sheep	0	0	0	0	0	1	0	0	0	0
genome	0	0	0	0	0	0	1	1	1	0
DNA	0	0	0	0	0	0	2	0	0	1

LSI Example

- The reconstructed term-document matrix $\hat{X}^T$ after projecting on a subspace of dimension K=2
- Σ = diag(2.57, 2.49, 1.99, 1.9, 1.68, 1.53, 0.94, 0.66, 0.36, 0.10)
- $\hat{\Sigma}$ = diag(2.57, 2.49, 0, 0, 0, 0, 0, 0, 0, 0)

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
open-source	0.34	0.28	0.38	0.42	0.24	0.00	0.04	0.07	0.02	0.01
software	0.44	0.37	0.50	0.55	0.31	-0.01	-0.03	0.06	0.00	-0.02
Linux	0.44	0.37	0.50	0.55	0.31	-0.01	-0.03	0.06	0.00	-0.02
released	0.63	0.53	0.72	0.79	0.45	-0.01	-0.05	0.09	-0.00	-0.04
Debian	0.39	0.33	0.44	0.48	0.28	-0.01	-0.03	0.06	0.00	-0.02
Gentoo	0.36	0.30	0.41	0.45	0.26	0.00	0.03	0.07	0.02	0.01
database	0.17	0.14	0.19	0.21	**0.14**	0.04	0.25	**0.11**	0.09	0.12
Dolly	-0.01	-0.01	-0.01	-0.02	0.03	0.08	0.45	0.13	0.14	0.21
sheep	-0.00	-0.00	-0.00	-0.01	0.03	0.06	0.34	0.10	0.11	0.16
genome	0.02	0.01	0.02	0.01	0.10	0.19	1.11	0.34	0.36	0.53
DNA	-0.03	-0.04	-0.04	-0.06	0.11	0.30	**1.70**	0.51	0.55	0.81

database	0	0	0	0	1	0	0	1	0	0

Problems

- Original matrix still has same dimension as our vocabulary size - potentially 100,000 x 100,000
- Matrix is extremely sparse
- We can perform Singular Value Decomposition (SVD) for dimensionality reduction, however at quadratic computational cost
- Adding a new word changes the entire matrix

The Old Way

- Words as atomic units
- Index in a vocabulary

```
vocab = {
    'the': 1,
    'cat': 2,
    'licked': 3,
    'its': 4,
    'fur': 5,
    'truck': 6,
    'dog': 7,
    'moved': 8}
```

Problems (more on these soon):

- No notion of similarity between words
- Massive number of unique tokens
  - ~150,000 words in Oxford English Dictionary
  - Very high dimensionality
- There is no inherent structure to these vocabulary indices for our algorithms to exploit

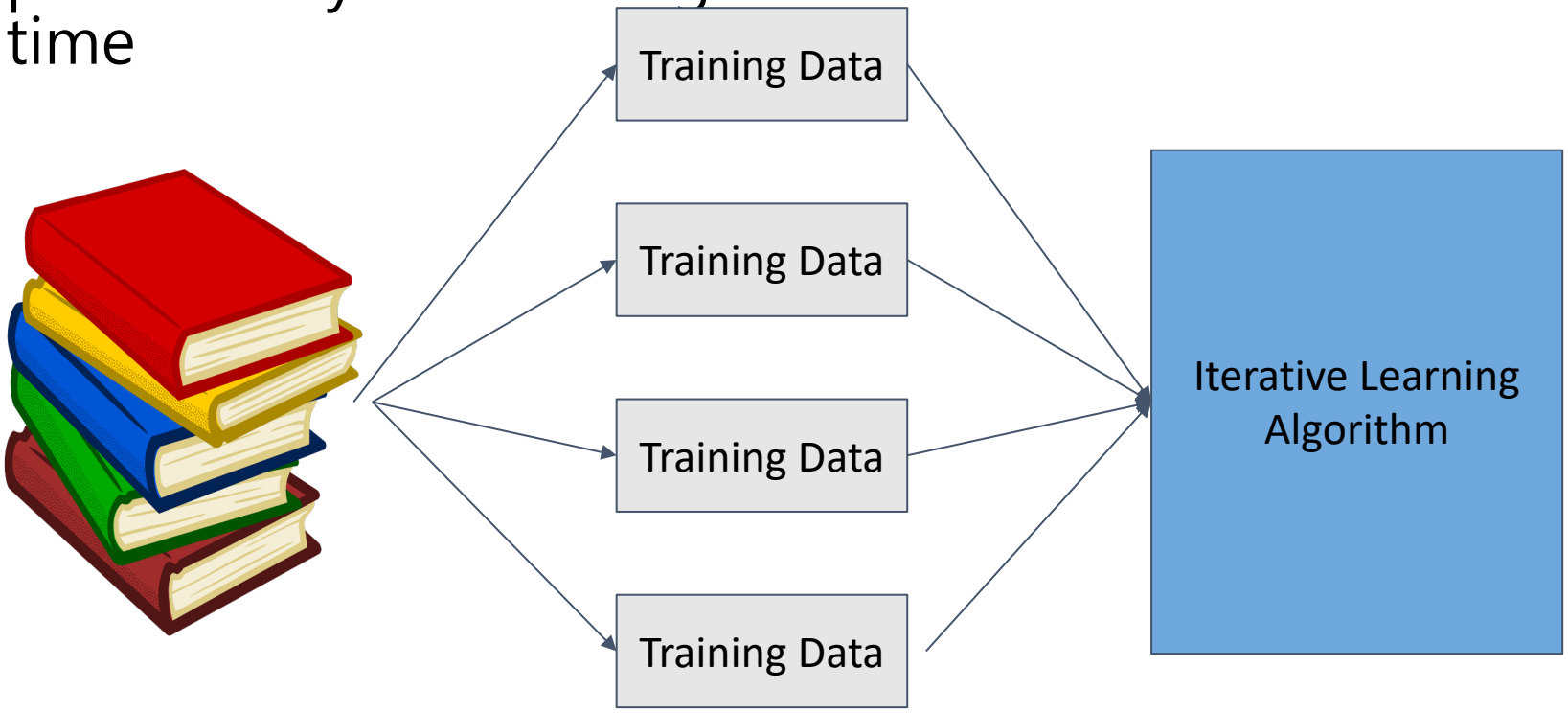
Dimensionality With Atomic Word Representations

- Assume word sequence limit of 10
- Assume 100,000 words in a realistic vocabulary - sample space is $\mathbb{R}^{100,000}$
- How many potential combinations of words?

$$100,000^{10} - 1 = 10^{50} - 1$$

Word2vec

Word2vec is an *iterative* training method: working with a huge matrix is cumbersome - instead, learning proceeds by considering small chunks of data at a time



Probabilistic Model: Some Choices

Unary language model

$$P(w_1, \cdots, w_n) = \prod_i P(w_i)$$

Ridiculous not to consider word order

Binary language model

$$P(w_1, \cdots, w_n) = \prod_i P(w_i | w_{i-1})$$

Better but still limited by short context distance

word2vec models (using window$m$ to get more context)

Continuous Bag of Words (CBOW)

$$P(w_c | w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m})$$

The cat [center word] its fur

Skip-Gram (which we will focus on)

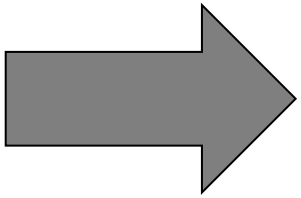
$$P(w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m} | w_c)$$

[left context] licked [right context]

Skip-Gram: Training Data

Corpus

The cat licked its fur. The truck moved.



Context window size = 1

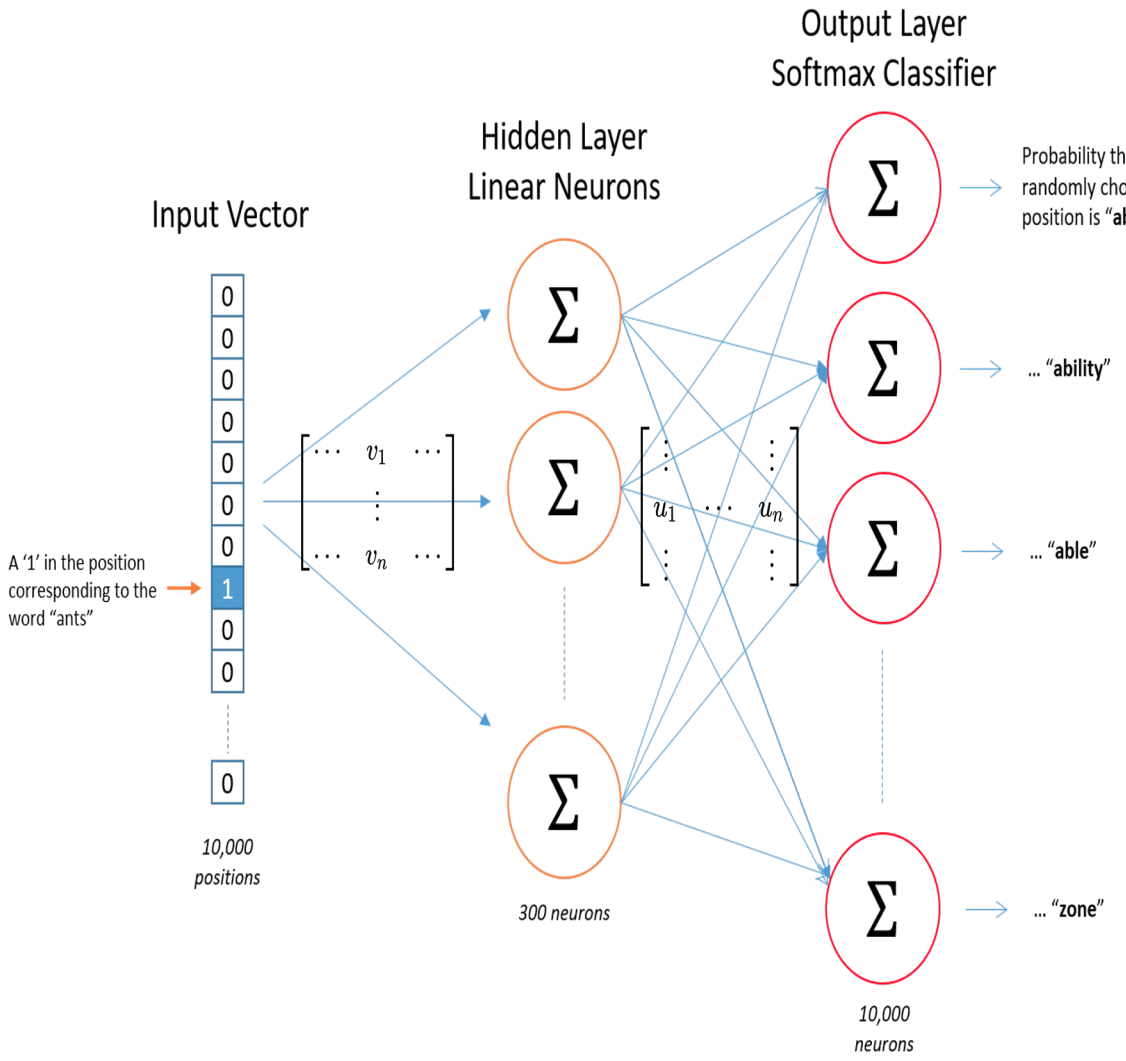
What if context size > 1?

- More word pairs
- Only pass two words to the model at a time

Training Word Pairs

Center	Context
the	cat
the	truck
cat	the
cat	licked
licked	cat
licked	its
…	…

Generate Probability Estimates



(1) Input vector selects input embedding from hidden layer matrix

(2) Softmax over multiplication with output matrix creates probability distribution over the vocabulary

This should be high for context words, low for others

Network Input

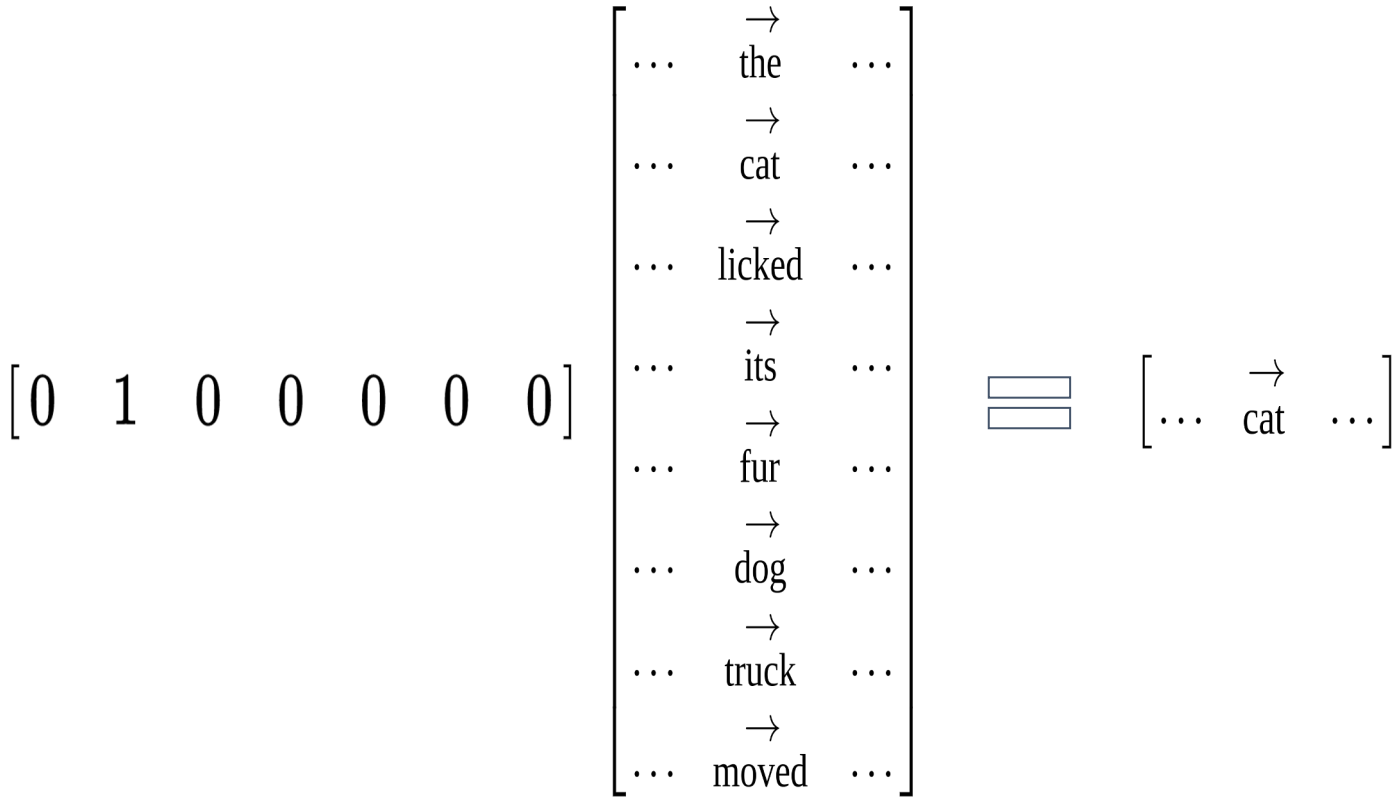
We imagine the words are encoded as "one-hot" vectors (all zeros except a one at the the word index in the embedding matrix)

```
vocab = {
    'the': 1,
    'cat': 2,
    'licked': 3,
    'its': 4,
    'fur': 5,
    'truck': 6,
    'dog': 7,
    'moved': 8}
```

$$\text{the} = \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} \quad \text{cat} = \begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\end{bmatrix} \quad \text{licked} = \begin{bmatrix}0\\0\\1\\0\\0\\0\\0\\0\end{bmatrix} \quad \text{etc...}$$

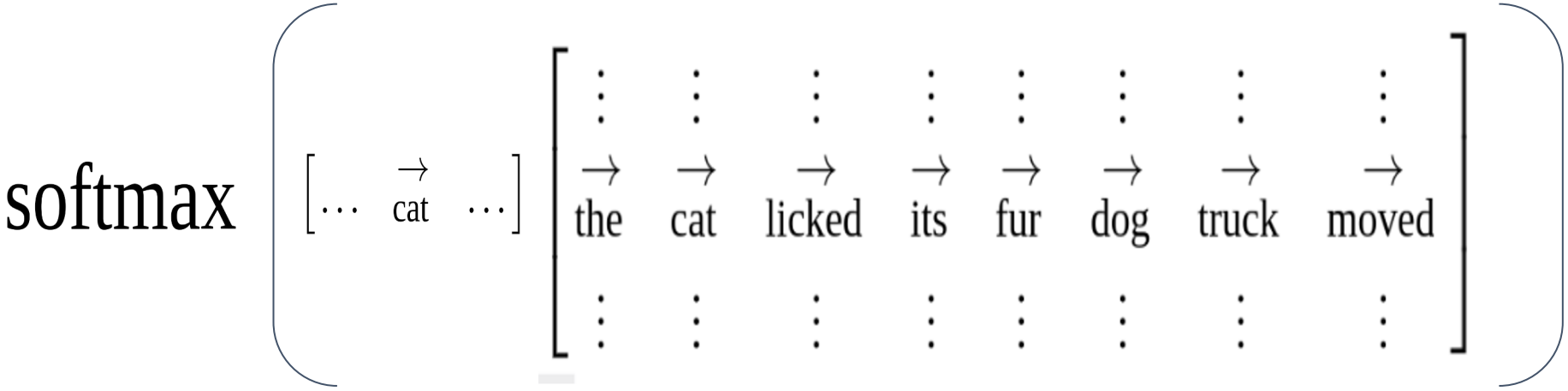
Embedding Lookup

In practice we don’ t do matrix multiplication, but just use the word index to pick out the vector.



Probability Over Vocab

Our word vector does a dot product with every word's output embedding, then applying softmax gives a probability distribution over the vocabulary



$$\text{probability word } w_i \text{ in context} = \hat{y}_i = P(w_i|\mathbf{v}_c, \mathbf{U}) = \frac{\exp(\mathbf{u}_{w_i}^T \mathbf{v}_c)}{\sum_{j=1}^{V} \exp(\mathbf{u}_{w_j}^T \mathbf{v}_c)}$$

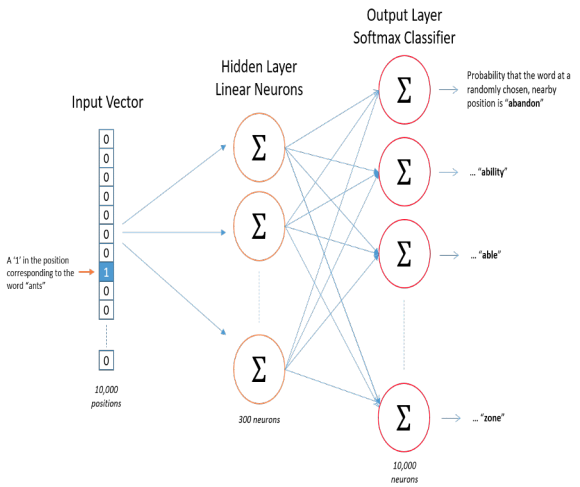
Calculate Error in Estimates

Training Pair:

$$\text{center word} = \text{cat} = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0] = \text{input}$$

$$\text{context word} = \text{licked} = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0] = \text{target}$$

$$[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]$$



$$\hat{\mathbf{y}} = [0.1\ \ 0.0\ \ 0.1\ \ 0.3\ \ 0.2\ \ 0.1\ \ 0.1\ \ 0.1]$$

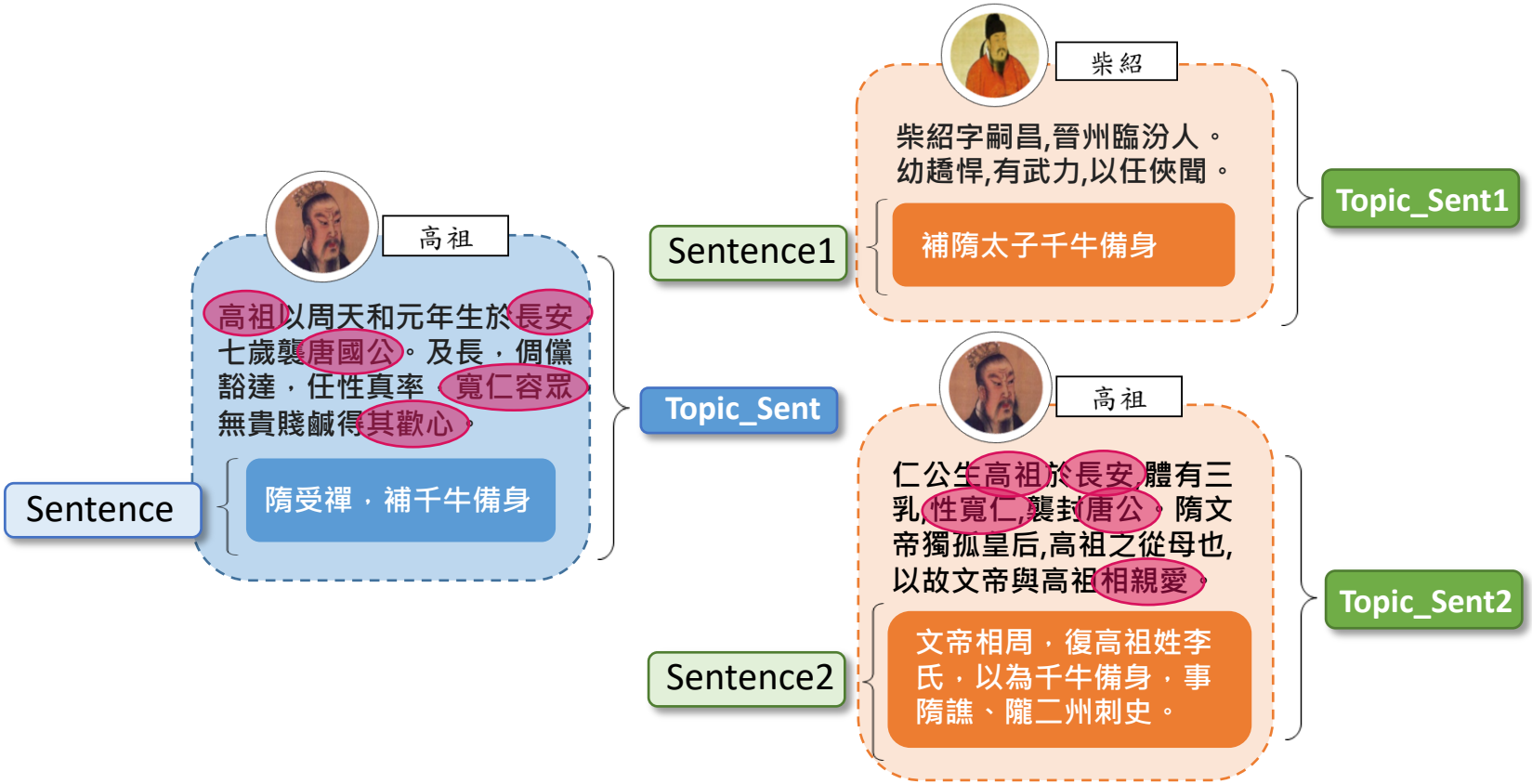
Loss Function

Cross entropy measures the distance between probability distributions

$$\mathrm{CE}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=1}^{V} y_i \log(\hat{y}_i)$$

$$\sum \left(\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \log \left(\begin{bmatrix} 0.1 & 0.0 & 0.1 & 0.3 & 0.2 & 0.1 & 0.1 & 0.1 \end{bmatrix} \right) \right)$$

Context Sensitive Autoencoder

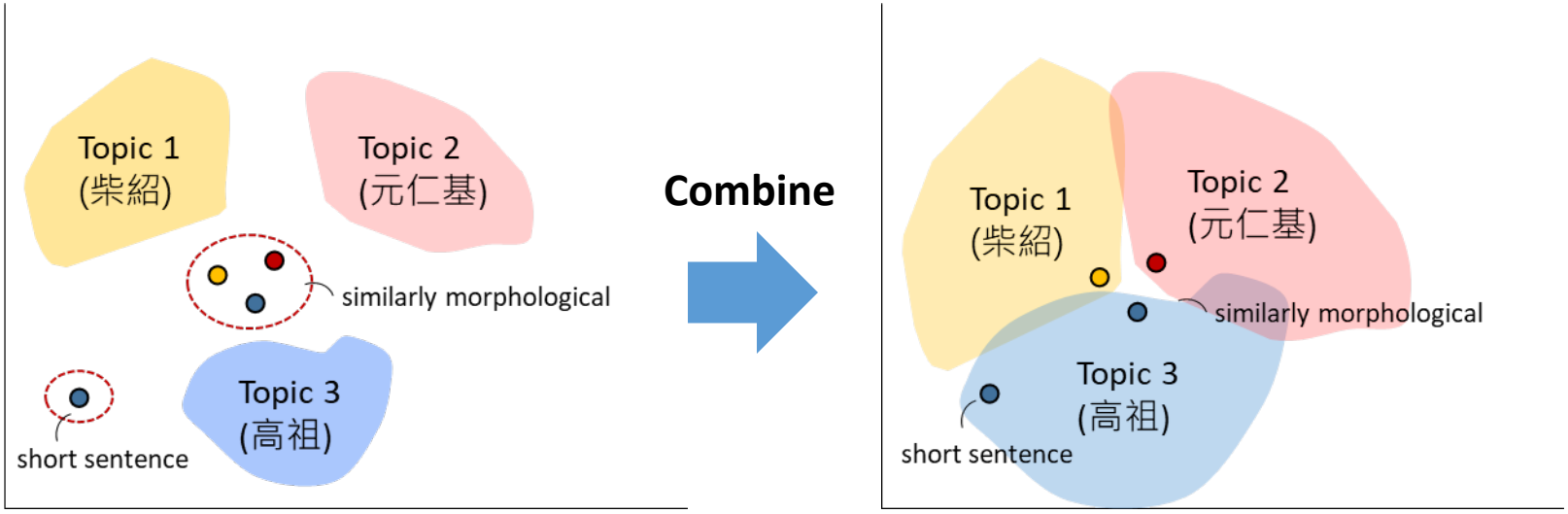


wrong $Sim($ Sentence Sentence1 $) > Sim($ Sentence Sentence2 $)$

correct $Sim($ Topic_Sent Topic_Sent2 $) > Sim($ Topic_Sent Topic_Sent1 $)$

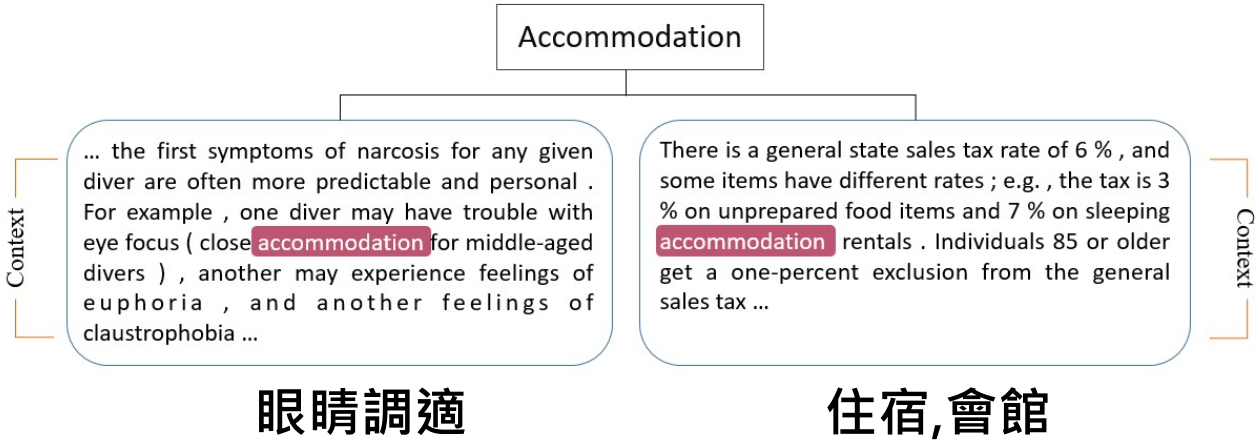
Context Sensitive Autoencoder

- **Topical Sentence Representation**
  - An embedding can represent **semantic features**
  - An embedding can represent **topic features**
  - An **unsupervised model** which can combine both of them into a new embedding



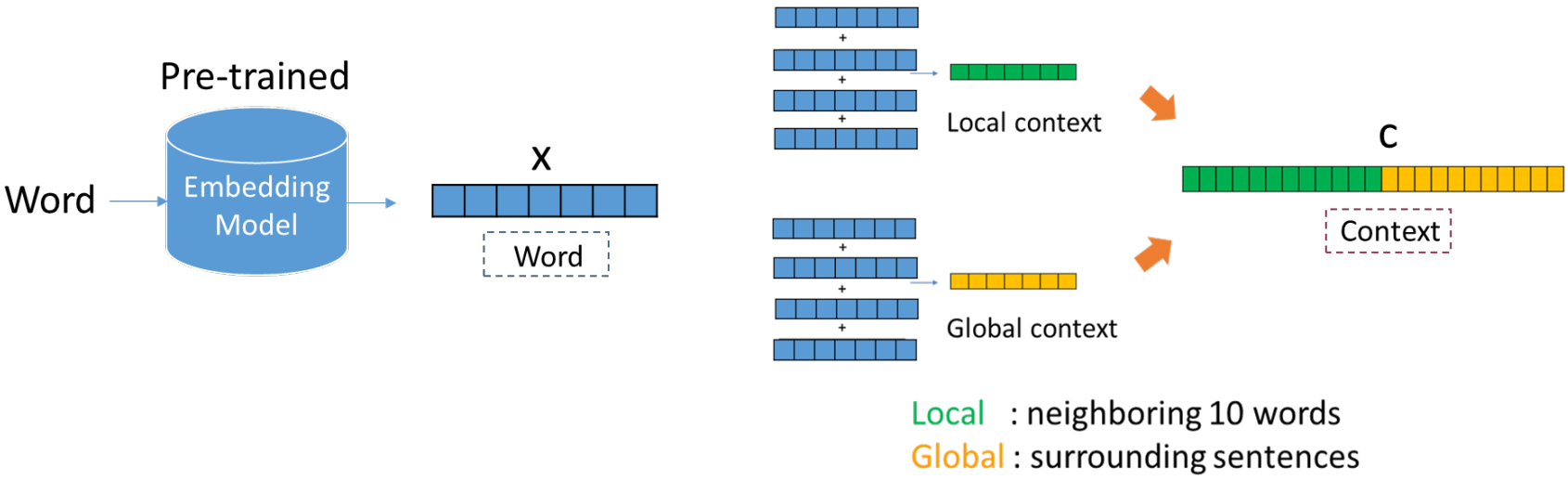
Context-Sensitive Autoencoder

- Context-Sensitive Autoencoder
  - **Unsupervised model**
  - Consider **contextual feature** and **enhance** word embedding
  - Input data of the model

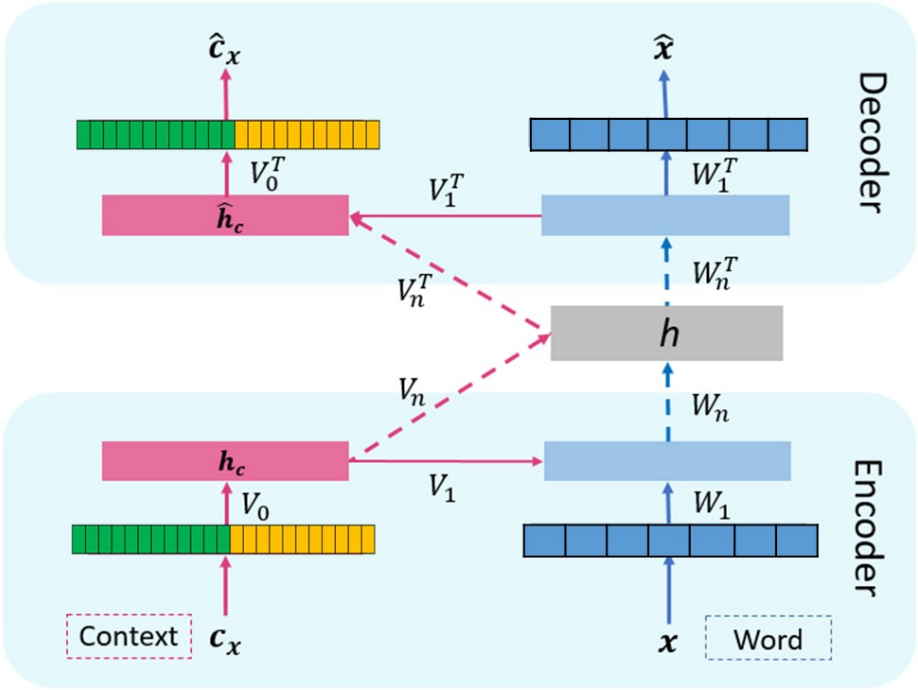


1. Amiri, H., Resnik, P., Boyd-Graber, J., & Daumé III, H. (2016). Learning text pair similarity with context-sensitive autoencoders. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vol. 1, pp. 1882-1892).

Context-Sensitive Autoencoder

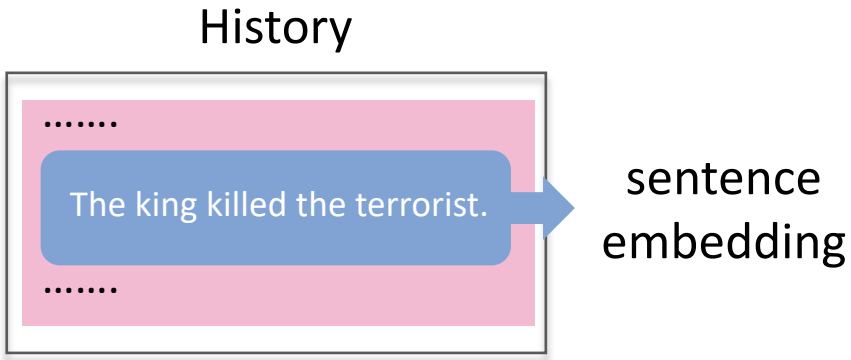


Context-Sensitive Autoencoder

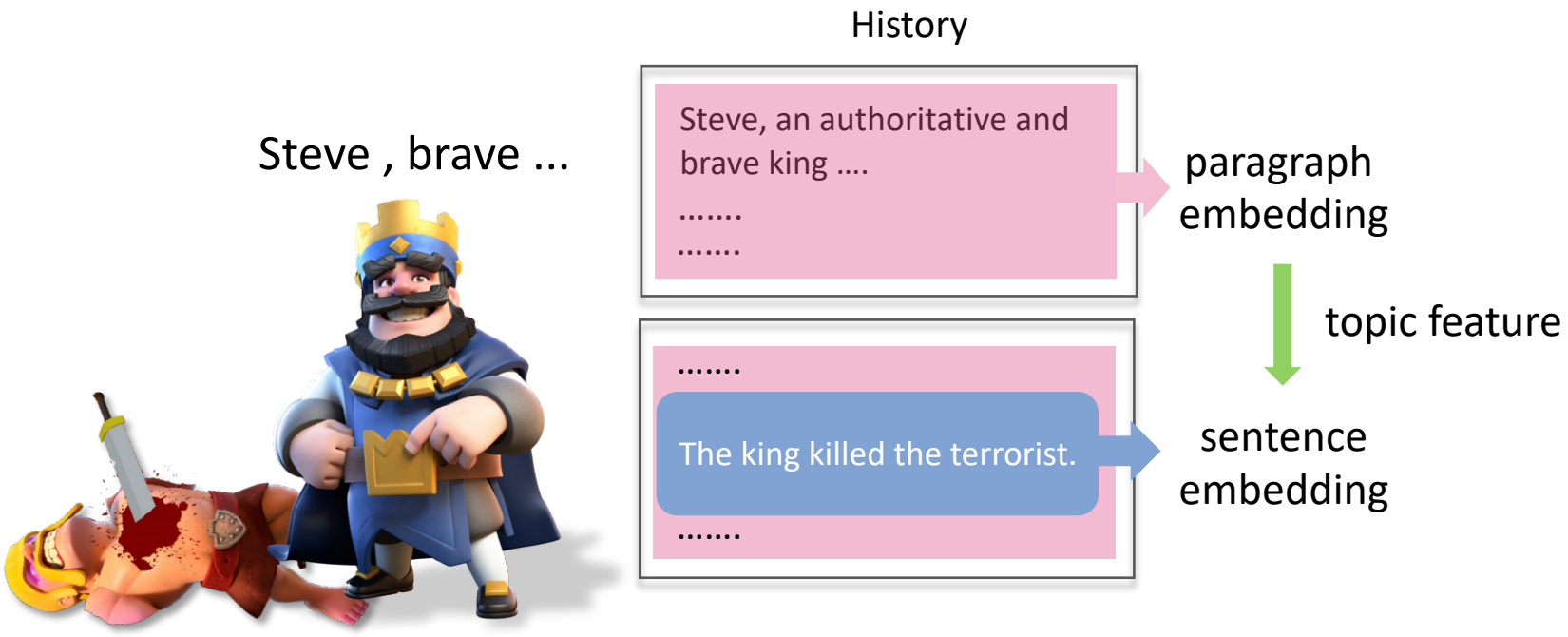


Topic-Aware Autoencoder

- We build Topic-Aware Auto-Encoder (TAAE) to combine **topic information** and **semantics of the sentence**.
  - **Paragraph** : have a **global view** and **topic information**
  - **Sentence** : capture **more semantic** of words
- Use paragraph embedding to assist sentence embedding that lack of topic feature.



Topic-Aware Autoencoder



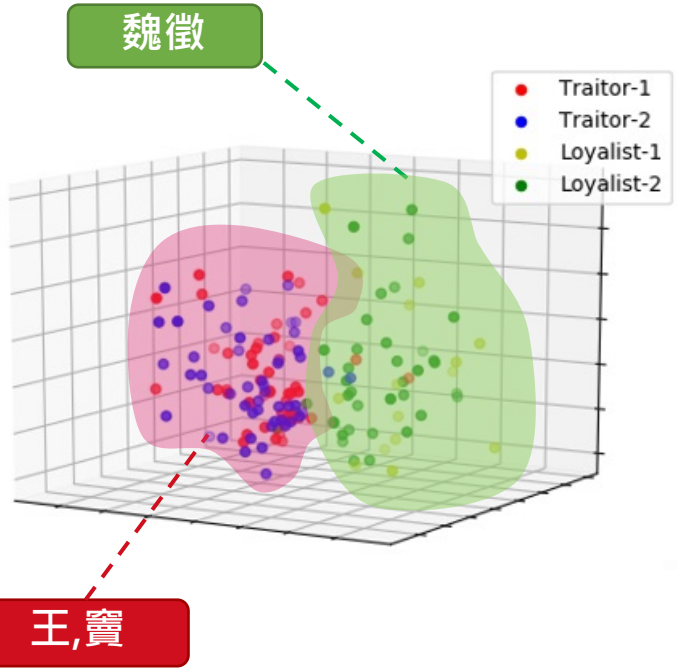
With **topical sentence embedding**, we can find the similar sentences in different books whatever the sentences are **short and morphological**.

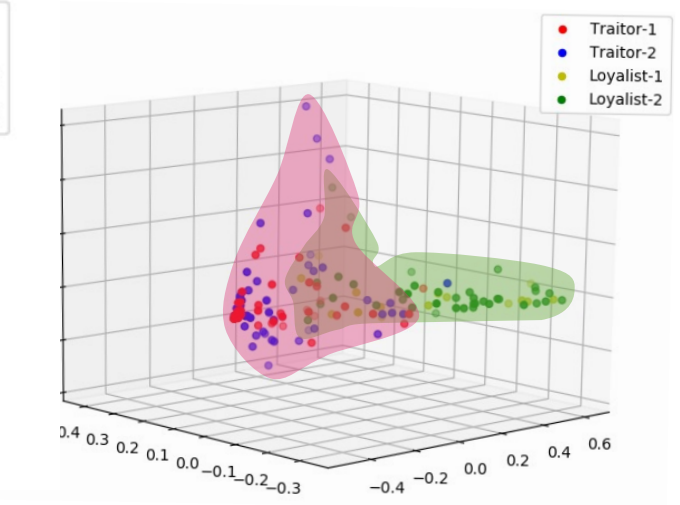
Method – *Pre-trained paragraph embedding*

- **Paragraph embedding**
  - Use **Doc2Vec** to train paragraph vector
  - We take **two types of volumes** which described for **魏徵** and **王竇**

魏徵

王世充,竇建德

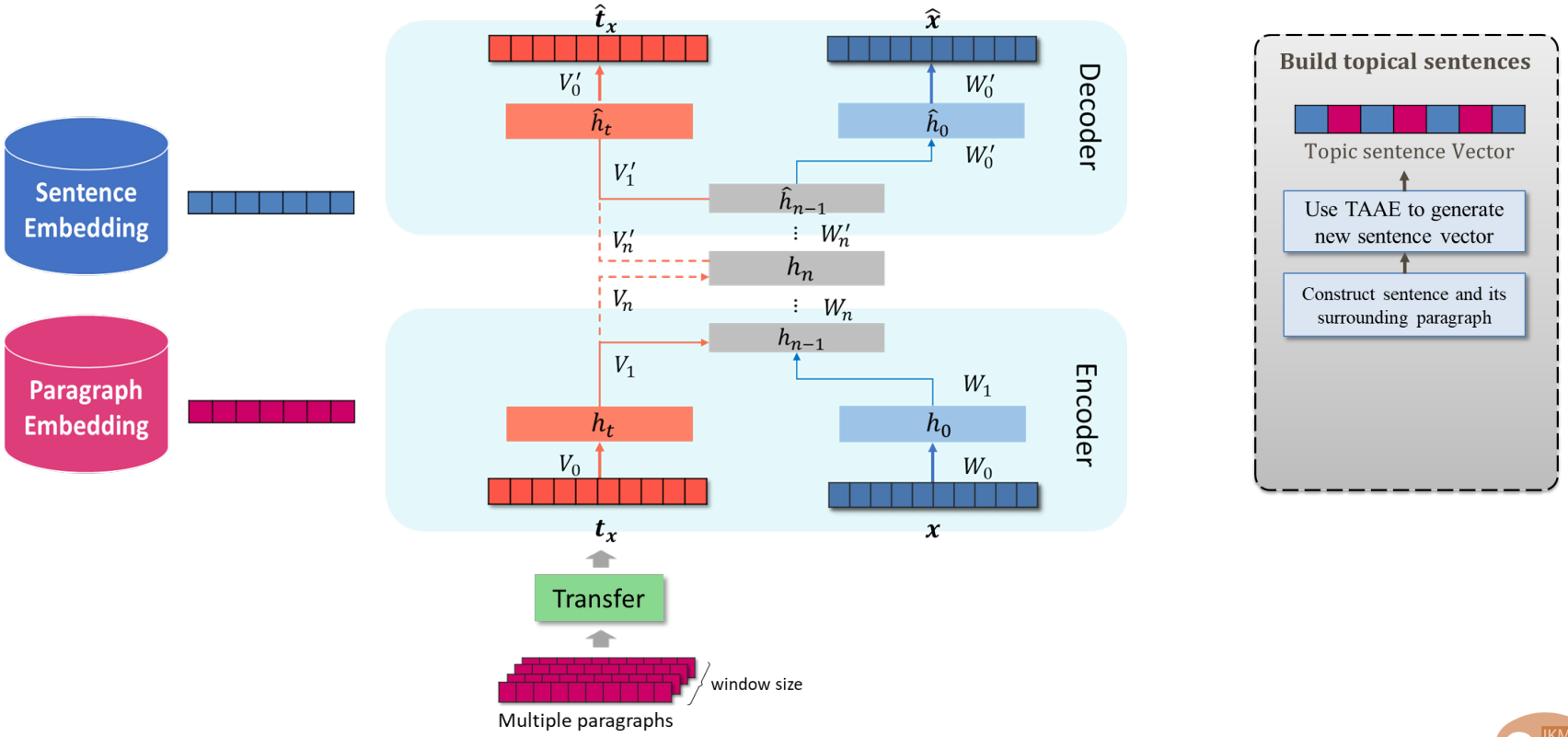




LDA doc-topic vector result

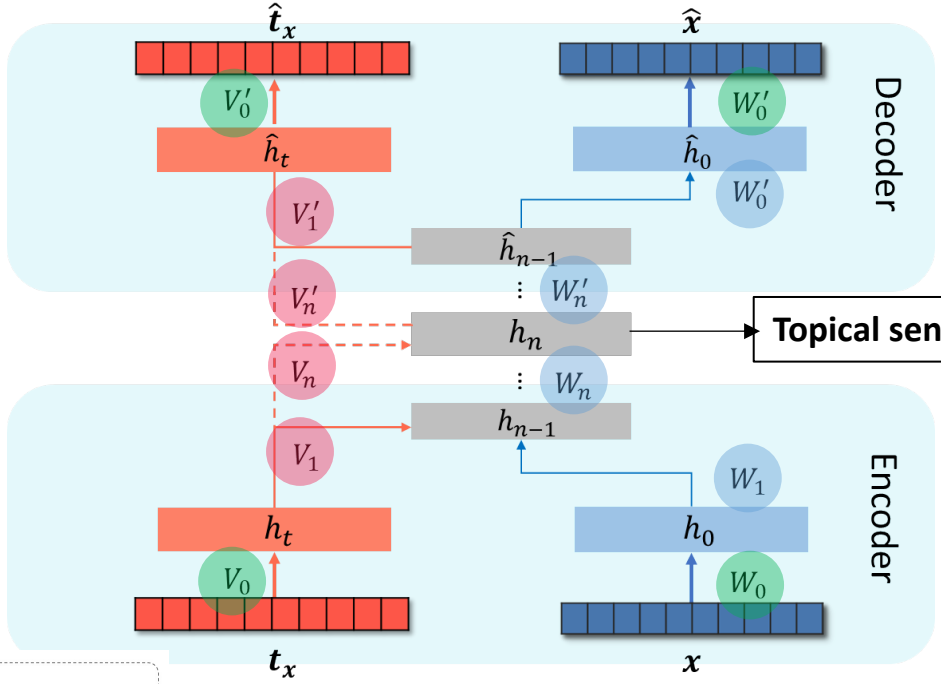
Method – *Topic-Aware Autoencoder*

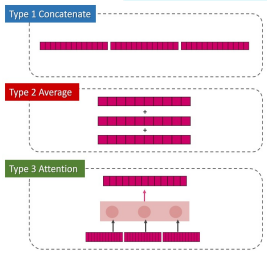
- We combine the **paragraph embeddings** and **sentence embeddings** in the autoencoder.



Method – *Topic-Aware Autoencoder*

Step 3 Put the pre-trained weights and fine-tune





Forward :

$$h_n = f(W_i h_{n-1} + V_i h_t + b_i)$$

$$h_t = f(V_0 t_x)$$

$$h_0 = f(W_0 x)$$

$$\hat{h}_{n-1} = f(W_n' \hat{h}_n + b'_{\hat{h}_n})$$

$$\hat{h}_t = \hat{h}_t + f(V_n' \hat{h}_n + b'_{\hat{h}_t})$$

$$\hat{x} = W_0' \hat{h}_0$$

$$\hat{h}_t = \hat{h}_t / n$$

$$\hat{t}_x = V_0' \hat{h}_t$$

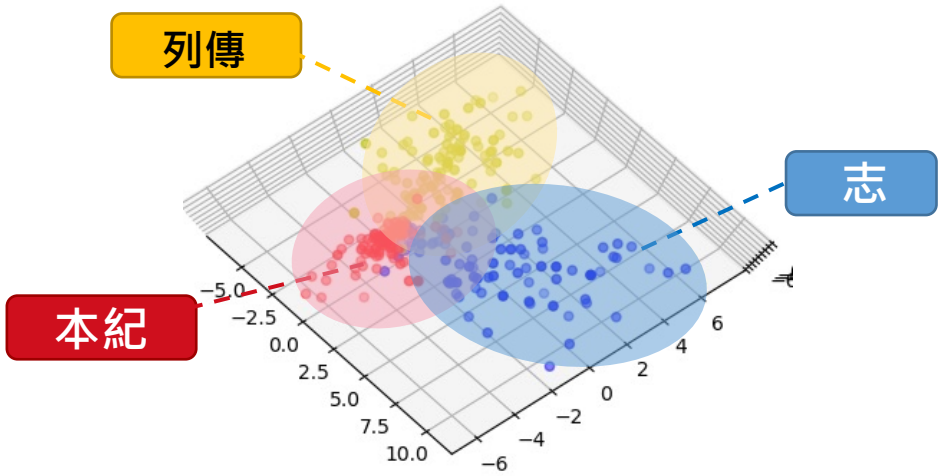
Loss :

$$loss = \|x - \hat{x}\|^2 + \lambda \|t - \hat{t}_x\|^2$$

λ : Control topic effect

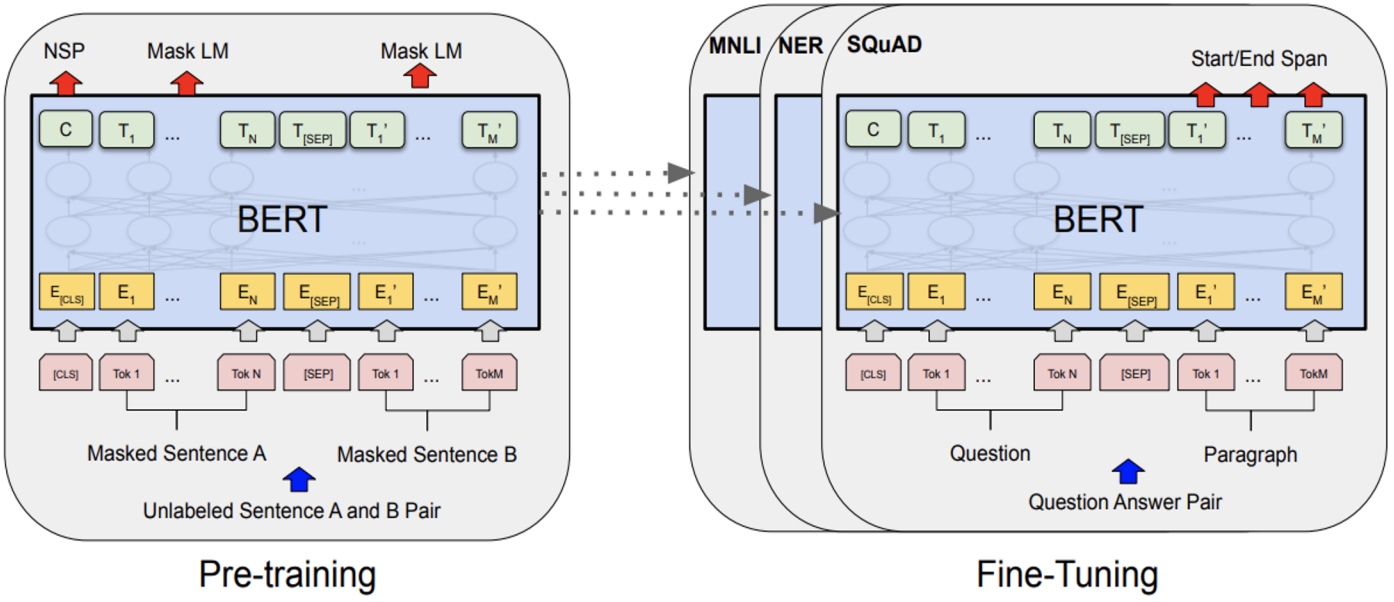
Visualization

- We take first volume for each category and **use PCA to visualize their paragraph embedding** in 3D-space
  - The following figure is **density of paragraphs in each category**



BERT Series - BERT

- **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**
  - 2019-06, by **Google**
  - Proceedings of the 2019 Conference of the North American of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)
  - Citation counts (to 2019-10-21): 1831
  - Authors: Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Touanova



BERT Series - BERT

- Same as GPT, but:
  - Different **pretrain objective function**
    - **MLM + NSP**
  - Different **input format**
    - `[CLS]` `segment1` `[SEP]` `segment2` `[SEP]`
  - Different **fine-tune strategy**
    - Feed `[CLS]` into linear layer

BERT Series - BERT

- **Masked Language Model (MLM)**
  - Definition:
    - Given a text and replace part of the words in text with **`[MASK]`** token, what are the original words being replaced?
  - Masking strategy:
    - Mask only **15%** of input
    - For the masked tokens, **80%** will be replaced with token **`[MASK]`**
    - For the masked tokens, **10%** will be replaced with **random** tokens
    - For the masked tokens, **10%** will be replaced with **original** tokens
  - Data will be preprocessed (or randomly masked) only **once**
    - Each epoch train on the same masked texts

BERT Series - BERT

- **Masked Language Model (MLM)**
  - For example:
    - Original sentence: "National Cheng Kung University is the best university in Taiwan."
    - Masked sentence: "National Cheng **[MASK]** University is the best **[MASK]** in Taiwan."
    - What is the probability of first **[MASK]** token being "Kung" ?
      - BERT will maximize the probability of the first **[MASK]** token being "Kung"
      - Or maximize the log-likelyhood of the first **[MASK]** token being "Kung"
    - What is the probability of second **[MASK]** token being "university" ?
      - BERT will maximize the probability of the second **[MASK]** token being "university"
      - Or maximize the log-likelyhood of the second **[MASK]** token being "university"

BERT Series - BERT

- **Next Sentence Prediction (NSP)**
  - Definition:
    - Given two sentences **s1**, **s2**, are the two sentences coming from same article?
    - If **s1** and **s2** are coming from **same** article, output **true**
    - If **s1** and **s2** are **not** coming from same **article**, output **false**
    - Binary classification problem
  - Data will be preprocessed (or sampled from large text corpus) only **once**
    - **50%** will sample **s1**, **s2** from **same article**, or label = `IsNext`
    - **50%** will sample **s1**, **s2** from **two random articles**, or label = `NotNext`
    - Each epoch train on the same pairs **s1**, **s2**
  - Using `[CLS]` token to predict
    - Feed `[CLS]` token into linear layer
  - Concate two sentences **s1**, **s2** with `[SEP]` tokens
    - Format: `[CLS]` `segment1` `[SEP]` `segment2` `[SEP]`

BERT Series - BERT

- **Next Sentence Prediction (NSP)**
  - Example 1:
    - Input: "`[CLS]` the man went to the store `[SEP]` he bought a gallon of milk `[SEP]`"
    - Label: `IsNext`
  - Example 2:
    - Input: "`[CLS]` the man went to the store `[SEP]` NCKU is the best `[SEP]`"
    - Label: `NotNext`

BERT Series - BERT

- Unsupervised **Pretraining**