

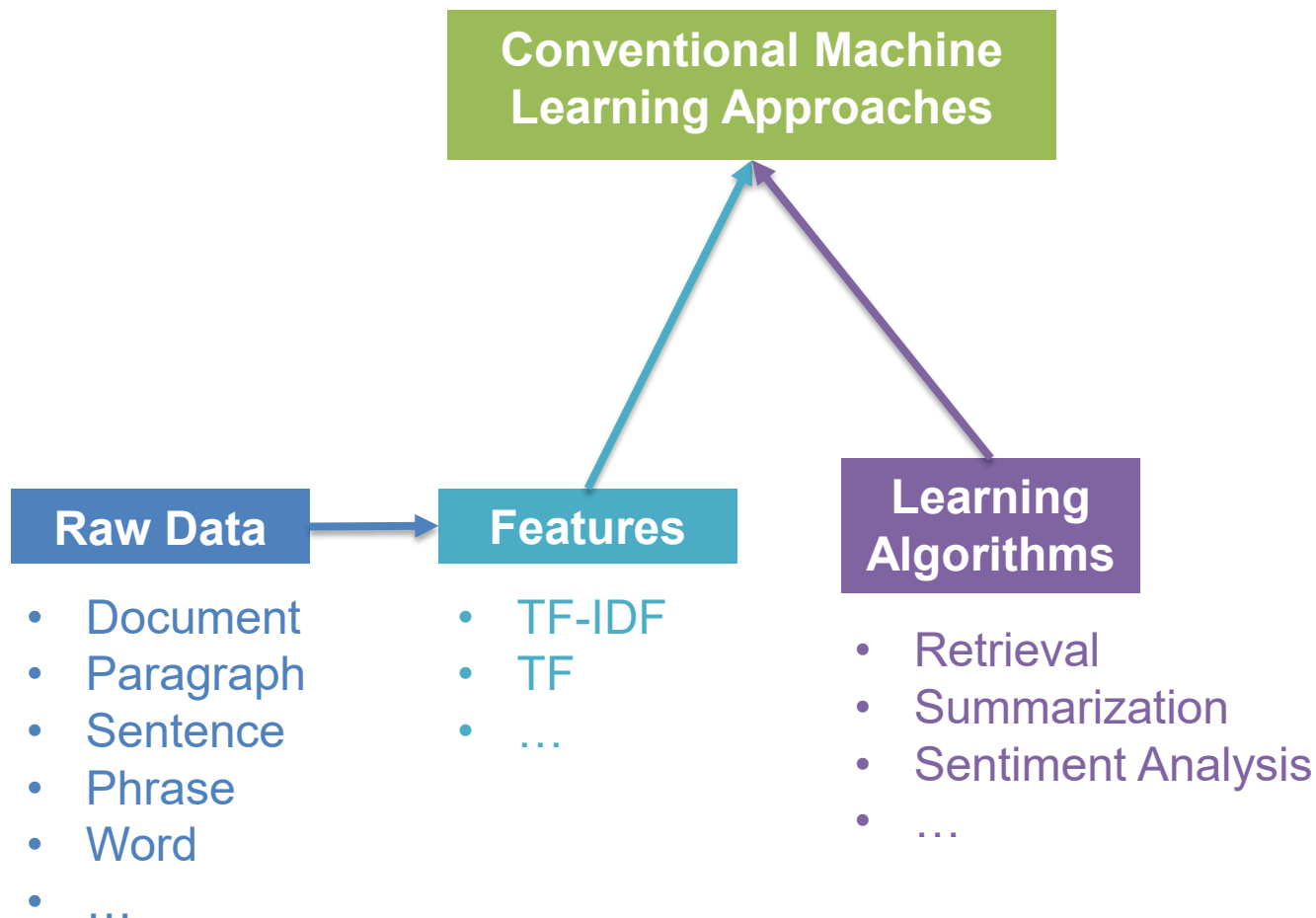
Classic Representations & Language Modeling

Kuan-Yu Chen (陳冠宇)

2018/03/22 @ TR-409, NTUST

Classic Representation Strategy for NLP

- In the context of NLP, the most popular representation strategy is the TF-IDF method



Term Frequency

- This is based on the observation that high frequency terms are important for describing documents
 - The more often a term occurs in the text of the document, the higher its weight
 - **Luhn Assumption**

$tf_{i,j}$: the occurred frequency of i^{th} word in j^{th} document

Binary	$\{0, 1\}$
Raw Frequency	$tf_{i,j}$
Log Normalization	$1 + \log_2(tf_{i,j})$
Double Normalization 0.5	$0.5 + 0.5 \frac{tf_{i,j}}{\max_j tf_{i,j}}$
Double Normalization σ	$\sigma + (1 - \sigma) \frac{tf_{i,j}}{\max_j tf_{i,j}}$

Example

- Take “Log Normalization” for example

$$\bar{tf}_{i,j} = \begin{cases} 1 + \log_2(tf_{i,j}) & \text{if } tf_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases}$$

To do is to be.
To be is to do.

d_1

To be or not to be.
I am what I am.

d_2

I think therefore I am.
Do be do be do.

d_3

Do do do, da da da.
Let it be, let it be.

d_4

Vocabulary	
1	to
2	do
3	is
4	be
5	or
6	not
7	I
8	am
9	what
10	think
11	therefore
12	da
13	let
14	it

$tf_{i,1}$	$tf_{i,2}$	$tf_{i,3}$	$tf_{i,4}$
3	2	-	-
2	-	2.585	2.585
2	-	-	-
2	2	2	2
-	1	-	-
-	1	-	-
-	2	2	-
-	2	1	-
-	1	-	-
-	-	1	-
-	-	1	-
-	-	-	2.585
-	-	-	2
-	-	-	2

Inverse Document Frequency

- Raw term frequency as above suffers from a critical problem
 - All terms are considered equally important when it comes to assessing relevancy on a query
 - In fact certain terms have little or no discriminating power
- An immediate idea is to scale down the term weights by leveraging the document frequency of each term

$$idf_i = \log \frac{N}{df_i}$$

- N is the total number of documents in a collection
- df_i is the number of documents which contain the i^{th} word
- The idf of a rare term is high, whereas the idf of a frequent term is likely to be low
- idf is used to reveal the **term specificity**

Variants

- Five distinct variants of inverse document frequency

Unary	1
Inverse Frequency	$\log \frac{N}{df_i}$
Inverse Frequency Smooth	$\log \left(1 + \frac{N}{df_i} \right)$
Inverse Frequency Max	$\log \left(1 + \frac{\max_i(df_i)}{df_i} \right)$
Probabilistic Inverse Frequency	$\log \frac{N - df_i}{df_i}$

TF-IDF

- We now combine the definitions of term frequency and inverse document frequency, to produce a composite weight for each term in each document

$$TF - IDF_{i,j} = tf_{i,j} \times idf_i$$

- $TF - IDF_{i,j}$ assigns to term w_i a weight in document d_j
 - $TF - IDF_{i,j}$ will be higher when w_i occurs many times within a small number of documents
 - It will be lower when the term occurs fewer times in a document, or occurs in many documents
 - It will be the lowest when the term occurs in virtually all documents ($idf_i = 0$)

Example – 1

$$\bar{tf}_{i,j} = \begin{cases} 1 + \log_2(tf_{i,j}) & \text{if } tf_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$idf_i = \log \frac{N}{n_i}$$

To do is to be.
To be is to do.

d_1

To be or not to be.
I am what I am.

d_2

I think therefore I am.
Do be do be do.

d_3

Do do do, da da da.
Let it be, let it be.

d_4

Vocabulary		$tf_{i,1}$	$tf_{i,2}$	$tf_{i,3}$	$tf_{i,4}$	idf_i
1	to	3	2	-	-	0.301
2	do	2	-	2.585	2.585	0.125
3	is	2	-	-	-	0.602
4	be	2	2	2	2	0.000
5	or	-	1	-	-	0.125
6	not	-	1	-	-	0.125
7	I	-	2	2	-	0.301
8	am	-	2	1	-	0.301
9	what	-	1	-	-	0.125
10	think	-	-	1	-	0.125
11	therefore	-	-	1	-	0.125
12	da	-	-	-	2.585	0.125
13	let	-	-	-	2	0.125
14	it	-	-	-	2	0.125

Example – 2

Vocabulary		$tf_{i,1}$	$tf_{i,2}$	$tf_{i,3}$	$tf_{i,4}$	idf_i
1	to	3	2	-	-	0.301
2	do	2	-	2.585	2.585	0.125
3	is	2	-	-	-	0.602
4	be	2	2	2	2	0.000
5	or	-	1	-	-	0.125
6	not	-	1	-	-	0.125
7	I	-	2	2	-	0.301
8	am	-	2	1	-	0.301
9	what	-	1	-	-	0.125
10	think	-	-	1	-	0.125
11	therefore	-	-	1	-	0.125
12	da	-	-	-	2.585	0.125
13	let	-	-	-	2	0.125
14	it	-	-	-	2	0.125

$\vec{d_1}$	$\vec{d_2}$	$\vec{d_3}$	$\vec{d_4}$
0.903	0.602	0.000	0.000
0.250	0.000	0.323	0.323
1.204	0.000	0.000	0.000
0.000	0.000	0.000	0.000
0.000	0.125	0.000	0.000
0.000	0.125	0.000	0.000
0.000	0.602	0.602	0.000
0.000	0.602	0.301	0.000
0.000	0.125	0.000	0.000
0.000	0.000	0.125	0.000
0.000	0.000	0.125	0.000
0.000	0.000	0.000	0.323
0.000	0.000	0.000	0.250
0.000	0.000	0.000	0.250

Cosine Similarity Measure

- By leveraging the TF-IDF weighting schema, each document can be presented as a vector
 - In other word, documents are all vectors, and the weight for each component is determined by its TF and IDF
- The similarity degree between a pair of documents can be computed by referring to the **cosine similarity measure**
 - $0 \leq \text{sim}(d_m, d_n) \leq 1$

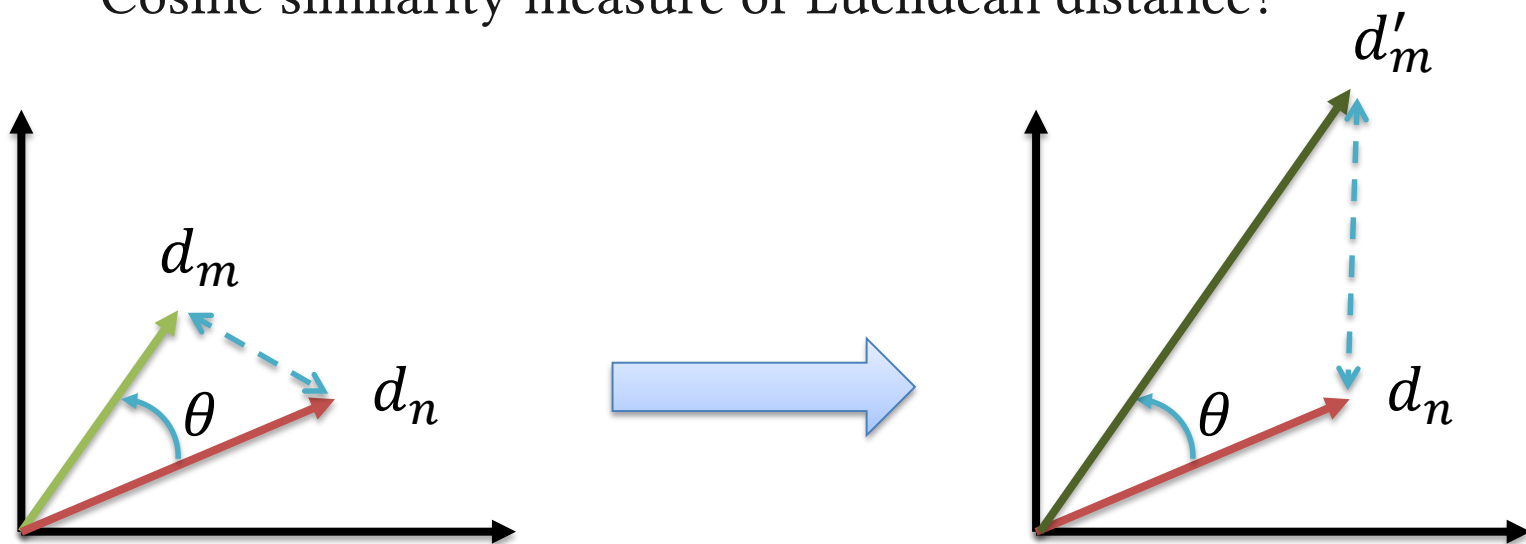
$$\text{sim}(d_m, d_n) = \frac{\vec{d}_m \cdot \vec{d}_n}{|\vec{d}_m| |\vec{d}_n|} = \frac{\sum_{w_i \in V} w_{i,m} \times w_{i,n}}{\sqrt{\sum_{w_i \in V} w_{i,m}^2} \times \sqrt{\sum_{w_i \in V} w_{i,n}^2}}$$

The Vector Space Model

- The vector space model usually refers to the combination of the vector representations and the cosine similarity measure
 - The VSM is a simple but efficient method for almost all of the NLP applications

$$\text{sim}(d_m, d_n) = \frac{\vec{d}_m \cdot \vec{d}_n}{|\vec{d}_m| |\vec{d}_n|}$$

- Cosine similarity measure or Euclidean distance?



Language Modeling

- A goal of statistical language modeling is to learn the joint probability function of sequences of words in a language

$$P(w_1, w_2, \dots, w_T)$$

- A statistical model of language can be represented by the conditional probability of the next word given all the previous ones (**chain rule**)

$$P(w_1, w_2, \dots, w_T) = P(w_1) \prod_{t=2}^T P(w_t | w_1, w_2, \dots, w_{t-1})$$

N-gram – Markov Assumption

- Unigram Model
 - Each word occurs independently of the other words
 - The so-called “bag-of-words” model

$$P(w_1, w_2, \dots, w_T) = P(w_1) \cdot P(w_2) \cdots P(w_T) = \prod_{t=1}^T P(w_t)$$

- Bigram Model

$$P(w_1, w_2, \dots, w_T) = P(w_1)P(w_2|w_1) \cdots P(w_T|w_{T-1}) = P(w_1) \prod_{t=2}^T P(w_t|w_{t-1})$$

- Trigram Model

$$\begin{aligned} P(w_1, w_2, \dots, w_T) &= P(w_1)P(w_2|w_1) \cdots P(w_T|w_{T-1}) \\ &= P(w_1)P(w_2|w_1) \prod_{t=3}^T P(w_t|w_{t-2}, w_{t-1}) \end{aligned}$$

Estimate the LM Parameters – 1

- Given a training corpus T , our goal is to estimate a set of language model parameters θ

$$T = w_1, w_2, w_3, w_4, w_5, \dots, w_L$$

- The objective function can be defined:

$$\max_{\theta} P(T) = P(w_1, w_2, w_3, w_4, w_5, \dots, w_L)$$

- By the N-gram model assumption

$$\begin{aligned} \max_{\theta} P(T) &\approx \prod_{i=1}^L P(w_i | \underline{w_{i-n+1}, \dots, w_{i-1}}) \\ &= \prod_{i=1}^L P(w_i | h^k) \\ &= \prod_k \prod_{j=1}^{|V|} P(w^j | h^k)^{c(h^k, w^j)} \end{aligned}$$

↑
history of words

Estimate the LM Parameters – 2

- Take log and add the Lagrange multipliers

$$\max_{\theta} \log P(T) \approx \sum_k \sum_{j=1}^{|V|} c(h^k, w^j) \log P(w^j | h^k) + \sum_k \lambda_{h^k} \left(\sum_{j=1}^{|V|} P(w^j | h^k) - 1 \right)$$

- Take gradient

$$\frac{\partial \log P(T)}{\partial P(w^j | h^k)} = \frac{c(h^k, w^j)}{P(w^j | h^k)} + \lambda_{h^k} = 0$$

$$\frac{c(h^k, w^j)}{P(w^j | h^k)} = -\lambda_{h^k}$$

$$c(h^k, w^j) = -\lambda_{h^k} P(w^j | h^k)$$

$$c(h^k) = \sum_{j=1}^{|V|} c(h^k, w^j) = \sum_{j=1}^{|V|} -\lambda_{h^k} P(w^j | h^k) = -\lambda_{h^k} \sum_{j=1}^{|V|} P(w^j | h^k) = -\lambda_{h^k}$$

Estimate the LM Parameters – 3

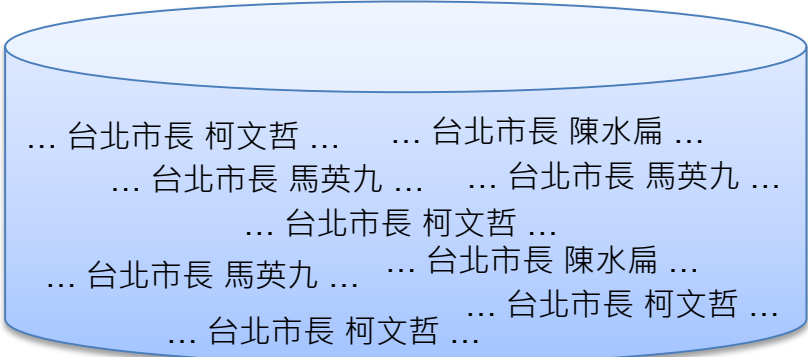
- Finally, we can obtain:

$$c(h^k) = -\lambda_{h^k}$$

$$\therefore c(h^k, w^j) = -\lambda_{h^k} P(w^j | h^k)$$

$$\therefore \frac{c(h^k, w^j)}{-\lambda_{h^k}} = \frac{c(h^k, w^j)}{c(h^k)} = P(w^j | h^k)$$

$$P(w_t | w_1, w_2, \dots, w_{t-1}) = \frac{c(w_1, w_2, \dots, w_{t-1}, w_t)}{c(w_1, w_2, \dots, w_{t-1})}$$



... 台北市長 柯文哲 台北市長 陳水扁 ...
... 台北市長 馬英九 台北市長 馬英九 ...
... 台北市長 柯文哲 ...
... 台北市長 馬英九 台北市長 陳水扁 ...
... 台北市長 柯文哲 ... 台北市長 柯文哲 ...
... 台北市長 柯文哲 ...

$$P(\text{柯文哲} | \text{台北市長}) = \frac{4}{9}$$

$$P(\text{馬英九} | \text{台北市長}) = \frac{3}{9}$$

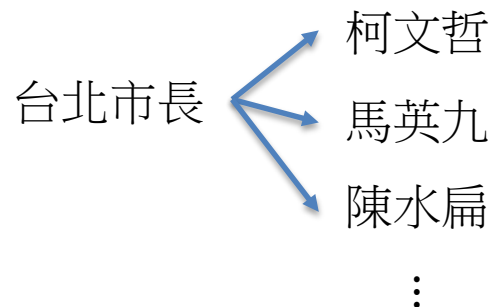
$$P(\text{陳水扁} | \text{台北市長}) = \frac{2}{9}$$

Evaluation – Perplexity

- Perplexity is a geometric average of inverse language model probability
 - For a given test corpus $S = w_1, w_2, w_3, w_4, w_5, \dots, w_L$

$$PPL(S) = \sqrt[L]{\frac{1}{P(w_1, w_2, w_3, w_4, w_5, \dots, w_L)}} \approx \sqrt[L]{\frac{1}{\prod_{t=1}^T P(w_t | w_{t-n+1}, \dots, w_{t-1})}}$$

- Minimizing perplexity is the same as maximizing probability
- Can be roughly interpreted as the geometric mean of the **branching factor** of the text when presented to the language model



Language Model Smoothing – 1

- The most deficiency of **N**-gram models is the data sparseness problem
 - Several language model smoothing techniques can help to determine a reasonable probability for an unseen \mathbb{X} -gram pattern
- Add-1 smoothing (Laplace smoothing)

$$P_{Add-1}(w_t | w_{t-n+1}, \dots, w_{t-1}) = \frac{c(w_{t-n+1}, \dots, w_{t-1}, w_t) + 1}{c(w_{t-n+1}, \dots, w_{t-1}) + |V|}$$

- Add- δ smoothing

$$P_{Add-\delta}(w_t | w_{t-n+1}, \dots, w_{t-1}) = \frac{c(w_{t-n+1}, \dots, w_{t-1}, w_t) + \delta}{c(w_{t-n+1}, \dots, w_{t-1}) + |V| \times \delta}$$

Language Model Smoothing – 2

- Simple interpolation
 - Take trigram for example

$$\tilde{P}(w_t|w_{t-2}, w_{t-1}) = \lambda_3 \times P(w_t|w_{t-2}, w_{t-1}) + \lambda_2 \times P(w_t|w_{t-1}) + \lambda_1 \times P(w_t)$$

$$\lambda_3 + \lambda_2 + \lambda_1 = 1$$

- Stupid backoff (by Google 2007)
 - It is a recursive formulation

$$S(w_t|w_{t-n+1}, \dots, w_{t-1}) = \begin{cases} P(w_t|w_{t-n+1}, \dots, w_{t-1}), & \text{if } c(w_{t-n+1}, \dots, w_t) > 0 \\ 0.4 \times S(w_t|w_{t-n+2}, \dots, w_{t-1}), & \text{otherwise} \end{cases}$$

- Take trigram for example

$$S(w_t|w_{t-2}, w_{t-1}) = \begin{cases} P(w_t|w_{t-2}, w_{t-1}) \\ 0.4 \times S(w_t|w_{t-1}) = \begin{cases} 0.4 \times P(w_t|w_{t-1}) \\ 0.4 \times (0.4 \times S(w_t)) = 0.4 \times 0.4 \times P(w_t) \end{cases} \end{cases}$$

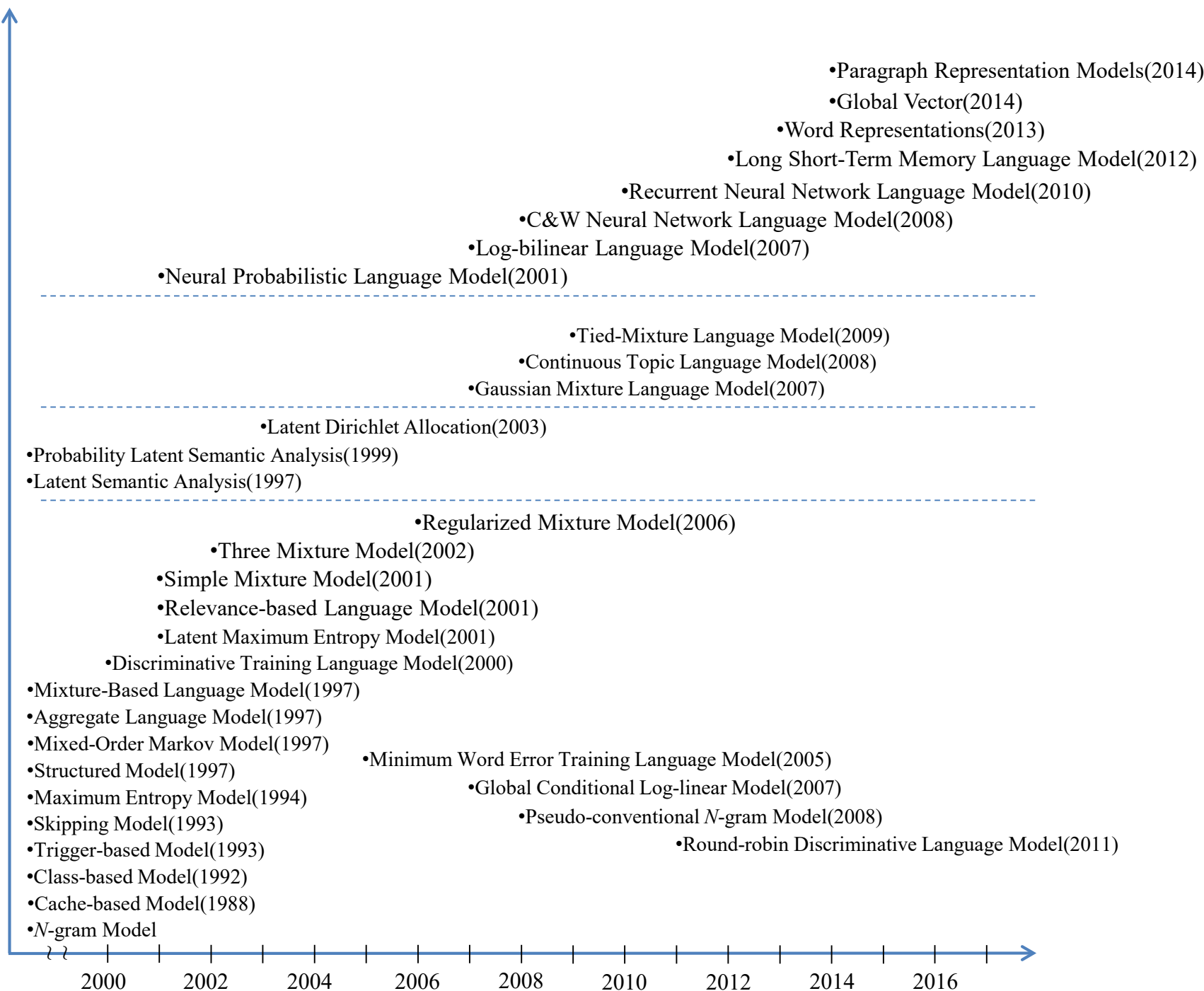
KL-Divergence Measure

- By leveraging the language modeling framework, each document can be presented as a distribution
- The similarity degree between a pair of documents can be computed by referring to the **KL-Divergence Measure**
 - $0 \leq KLD(d_m, d_n)$
 - The large the score, the more **dissimilar** the pair of documents

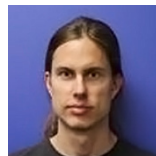
$$KLD(d_m || d_n) = \sum_{w \in V} P(w|d_m) \log \frac{P(w|d_m)}{P(w|d_n)}$$

- **KLD** is not symmetric, we can use **Jensen-Shannon divergence**

$$JSD(d_m, d_n) = \frac{1}{2} (KLD(d_m || d_n) + KLD(d_n || d_m))$$



NN-based Language Models



Word/Paragraph
Embeddings (2013~)

Neural Network Language Models (2001~)

Continuous Language Models



Continuous
Language Models
(2007~2009)

Topic Models (1997~2003)



Topic Models

Query Language
Models (2001~2006)



Word-Regularity Models

Discriminative Language Models (2000~2011)

Word-Regularity
Models (~1997)



2000

2002

2004

2006

2008

2010

2012

2014

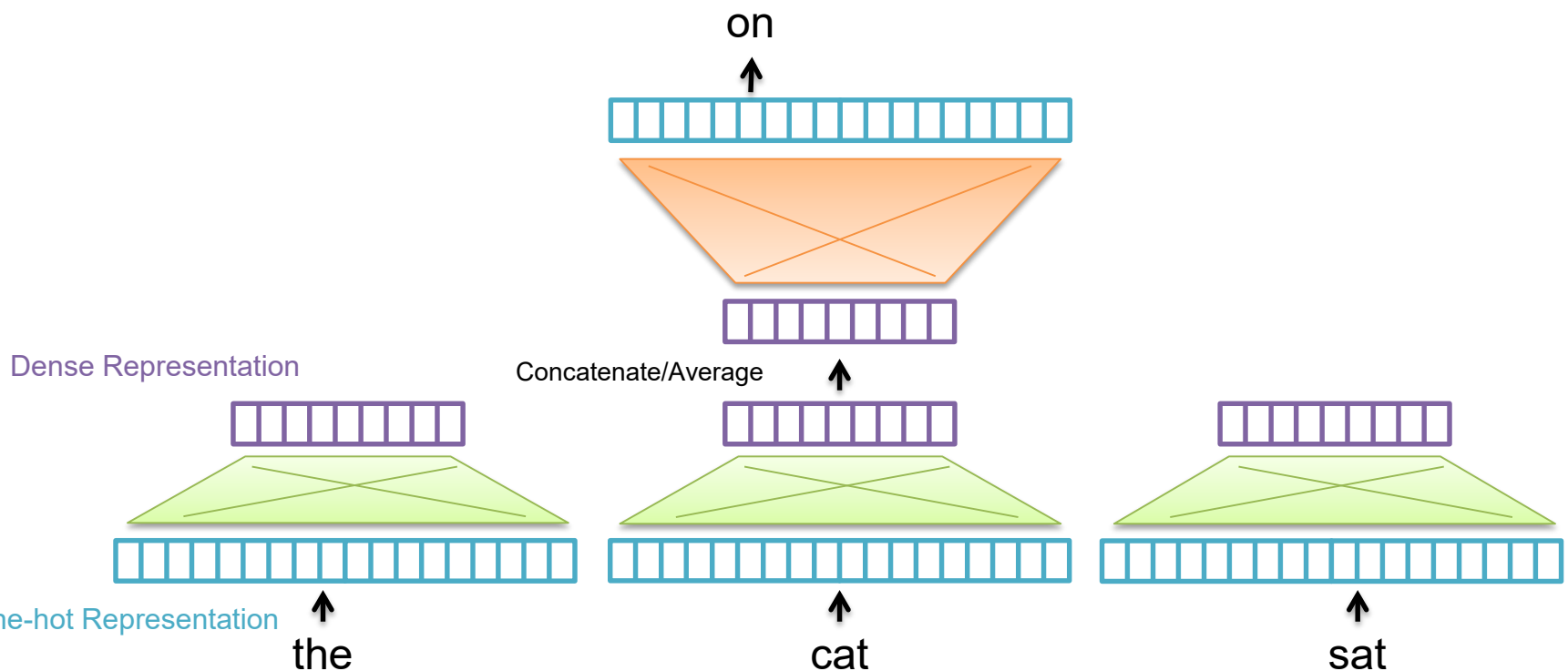
2016

22

Neural Network Language Modeling – 1

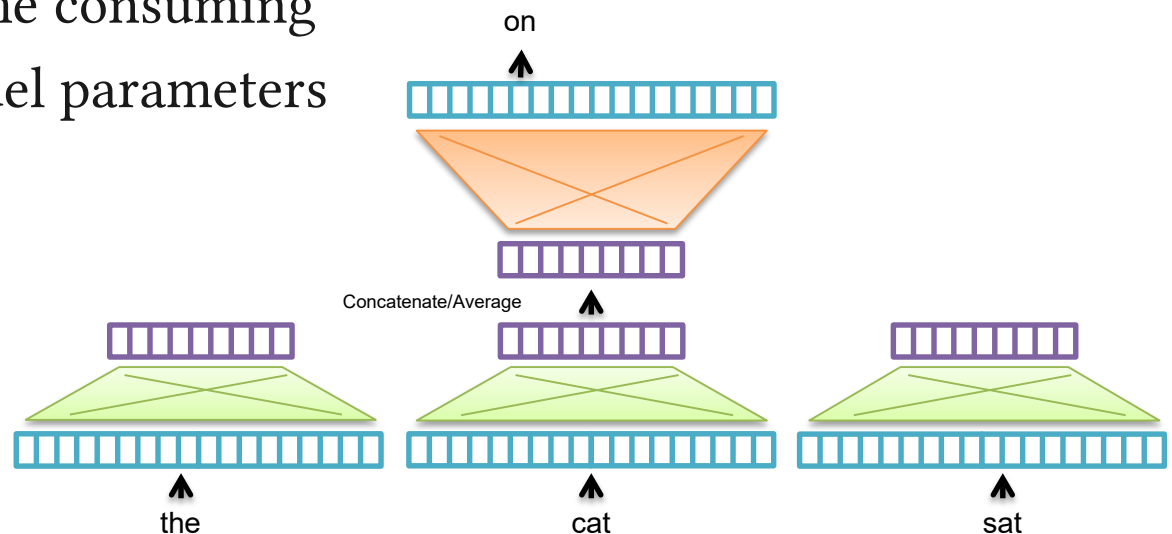
- The Neural Network Language Model (NNLM) estimated a statistical (\mathbb{N} -gram) language model for **predicting future words**

$$P(w_1, w_2, \dots, w_T) \approx \prod_{t=1}^T P(w_t | w_{t-n+1}, \dots, w_{t-1})$$



Neural Network Language Modeling – 2

- Advantages:
 - It has a build-in smoothing technique
 - (Semantically or Syntactically) Similar words will have similar probabilities
 - Generalization ability
- Disadvantages:
 - The training is time consuming
 - A large set of model parameters



Questions?



kychen@mail.ntust.edu.tw