

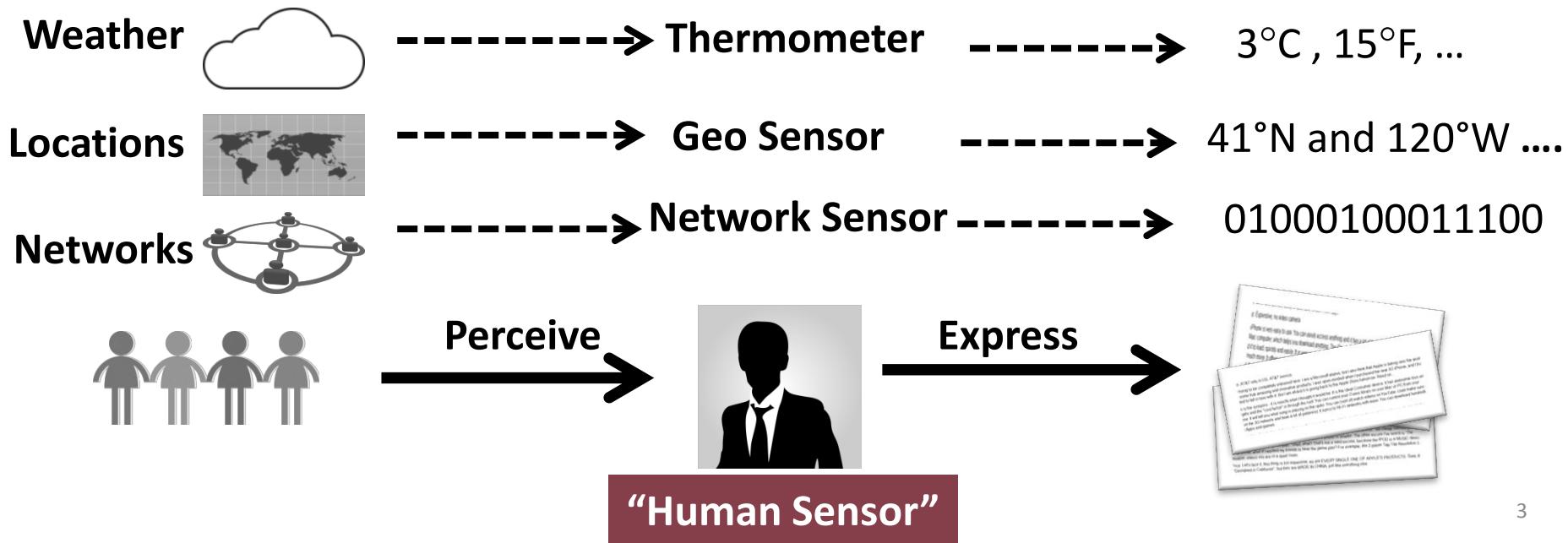
Overview Text Mining and Analytics

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

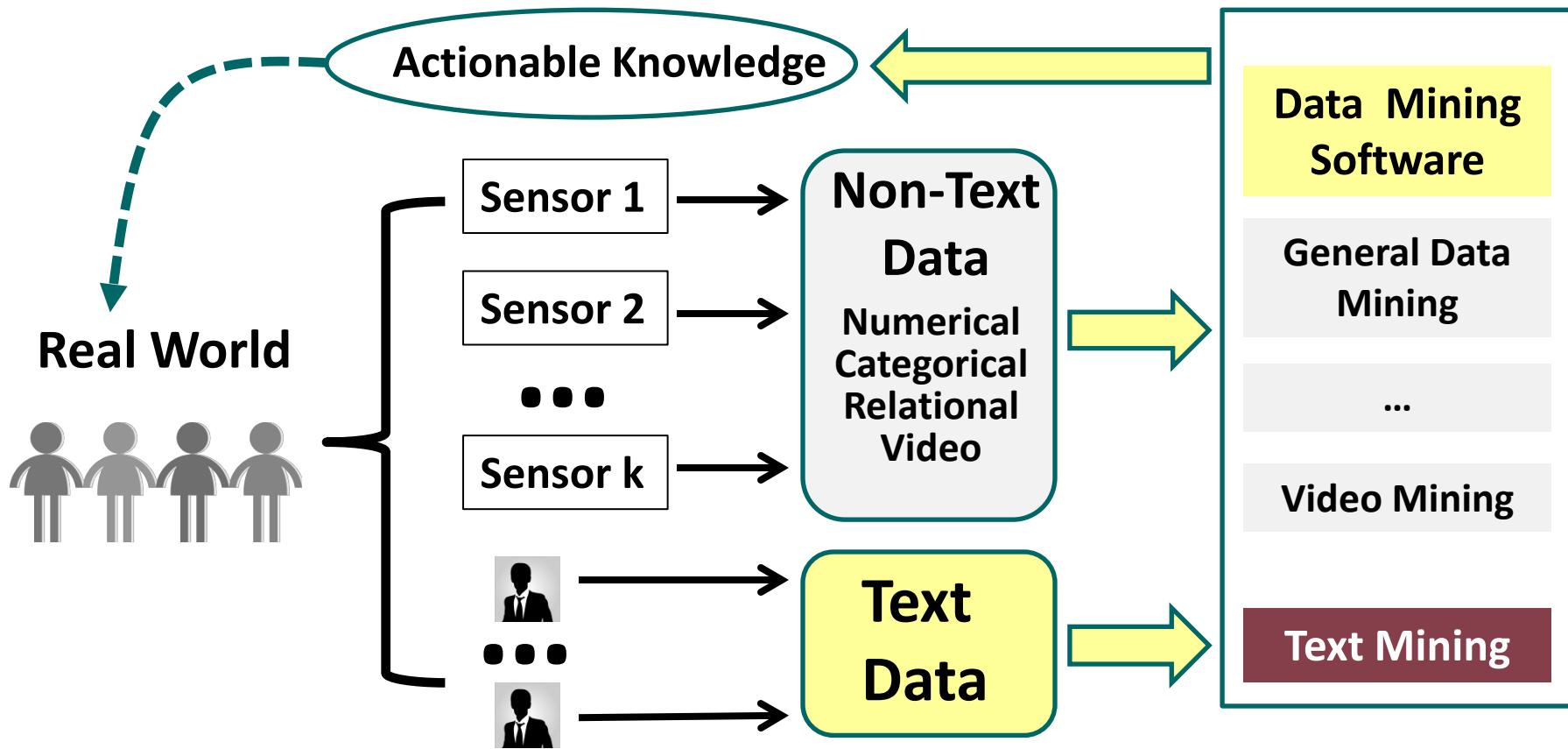
Text Mining and Analytics

- Text mining ≈ Text analytics
- Turn text data into **high-quality information or actionable knowledge**
 - **Minimizes human effort** (on consuming text data)
 - Supplies knowledge for **optimal decision making**
- Related to **text retrieval**, which is an essential component in any text mining system
 - Text retrieval can be a preprocessor for text mining
 - Text retrieval is needed for knowledge provenance

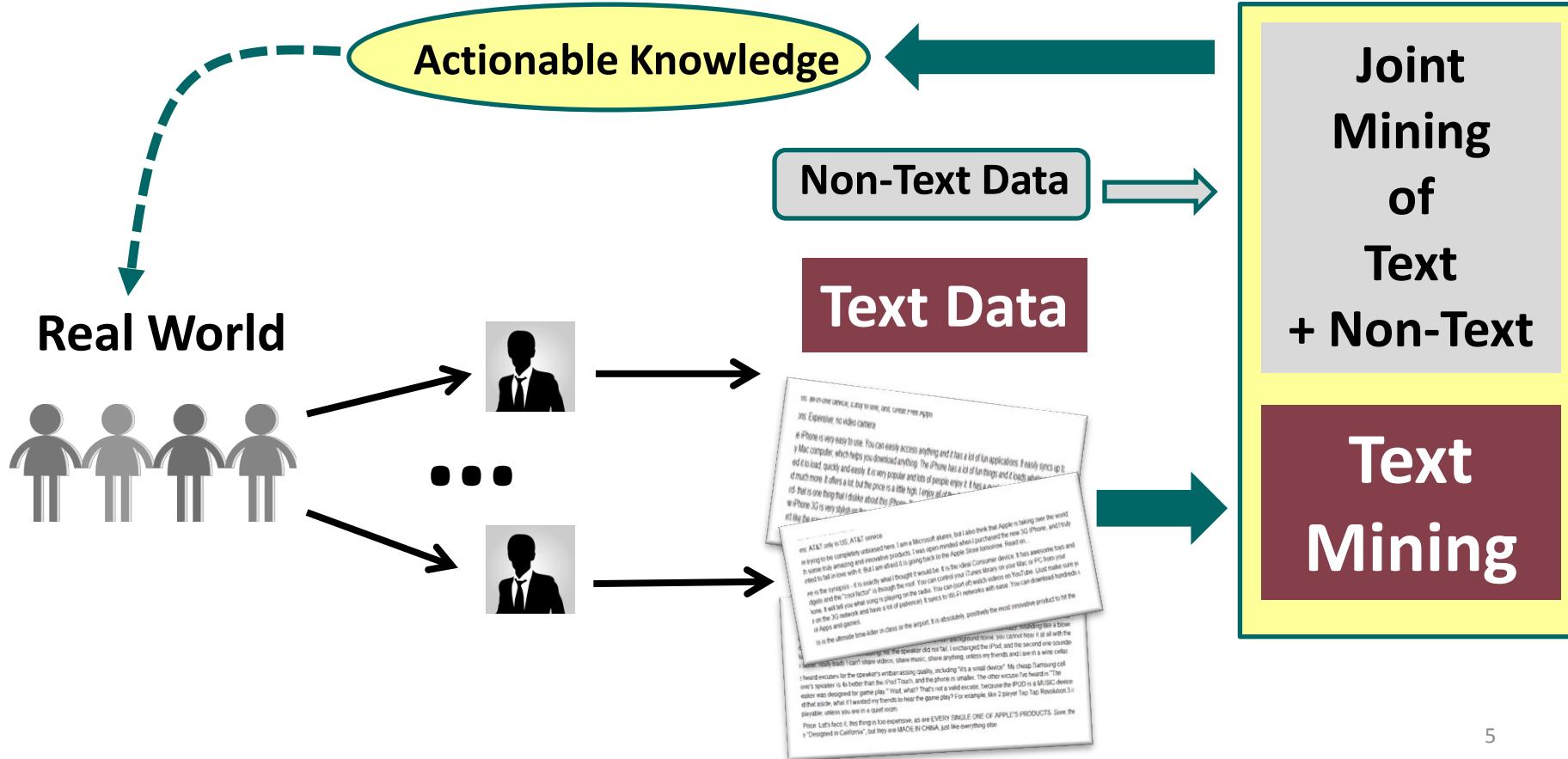
Text vs. Non-Text Data: Humans as Subjective “Sensors”



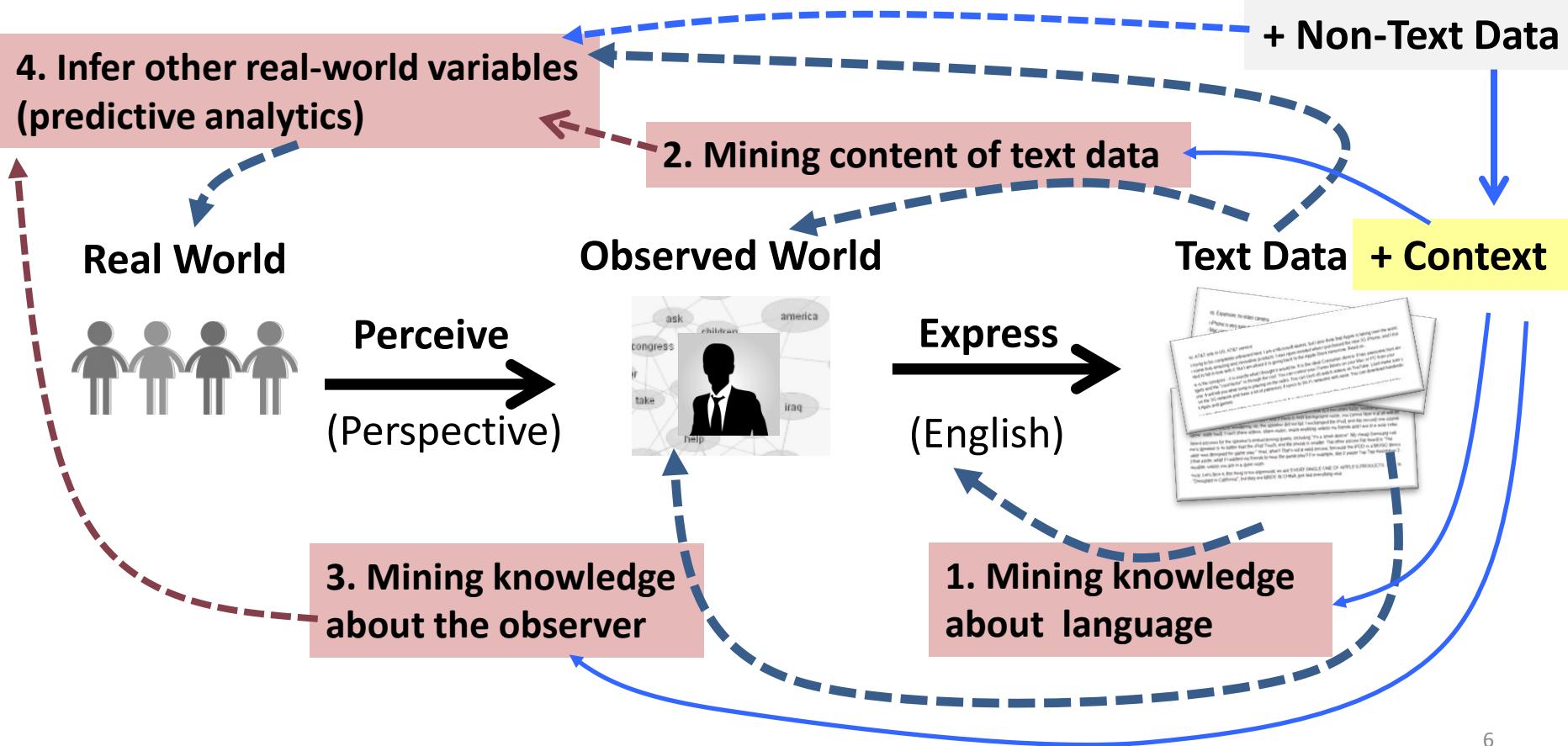
The General Problem of Data Mining



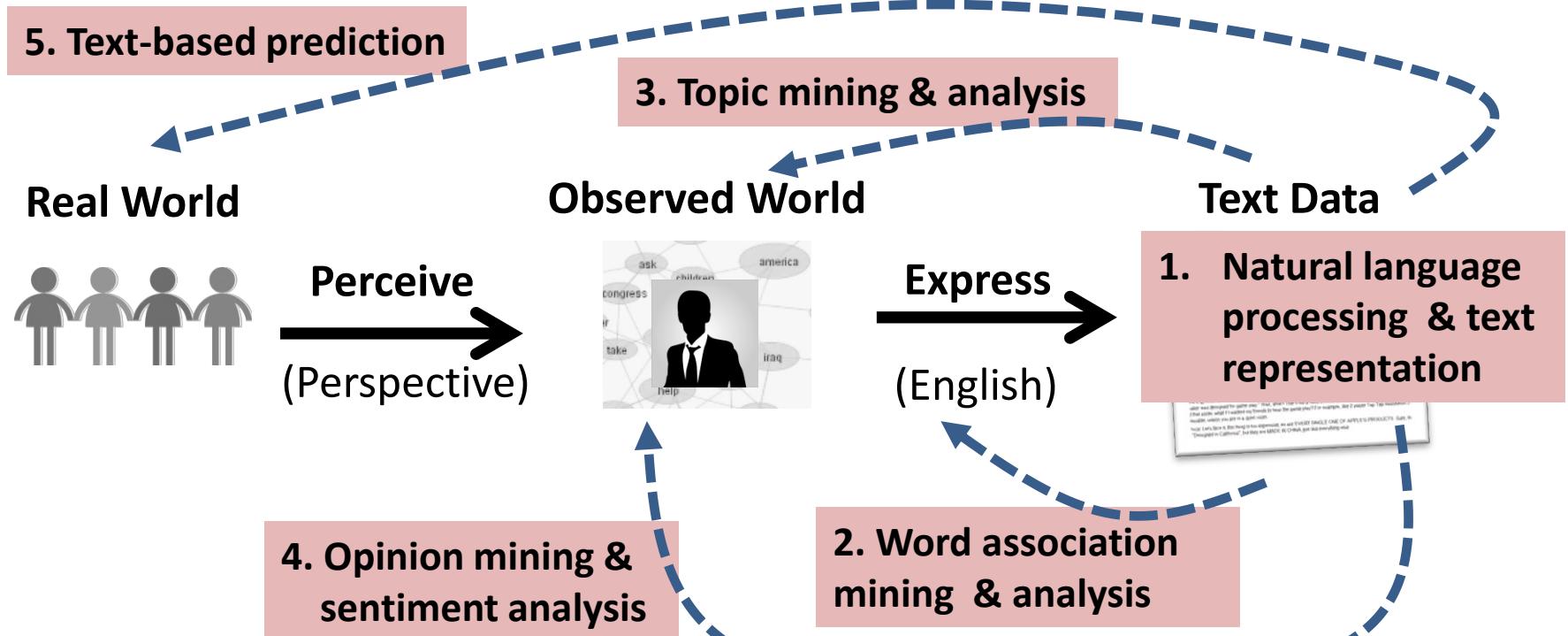
The Problem of Text Mining



Landscape of Text Mining and Analytics



Topics Covered in This Course



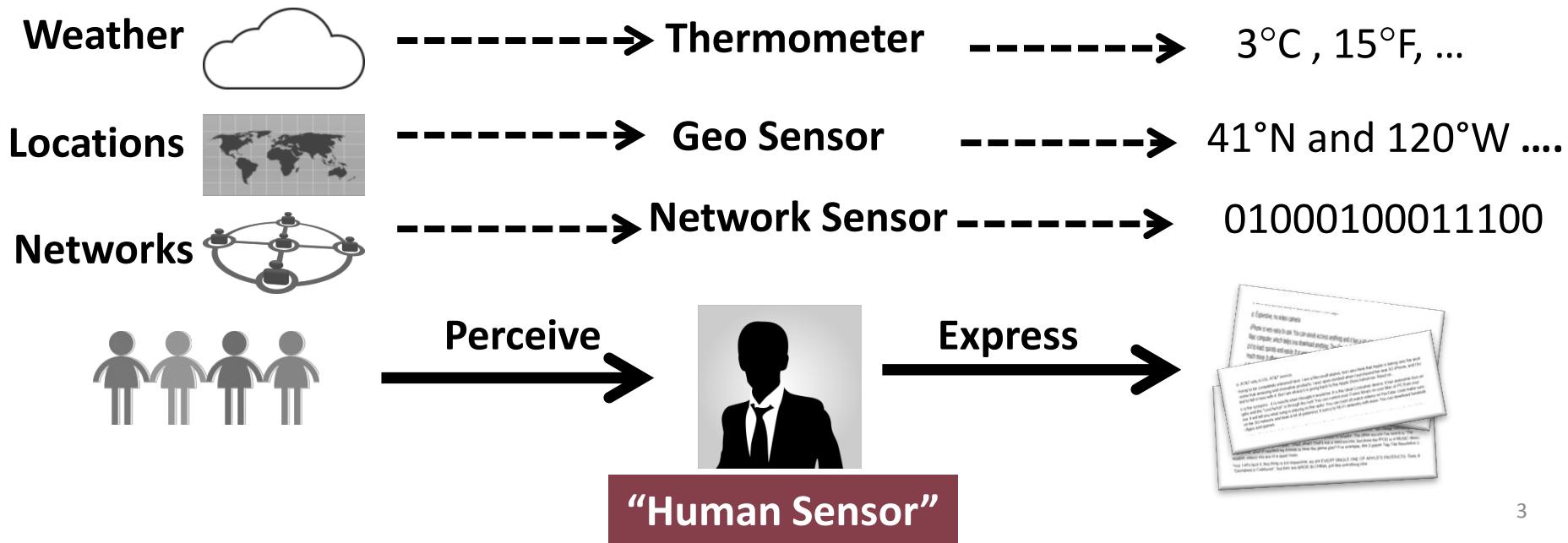
Overview Text Mining and Analytics

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

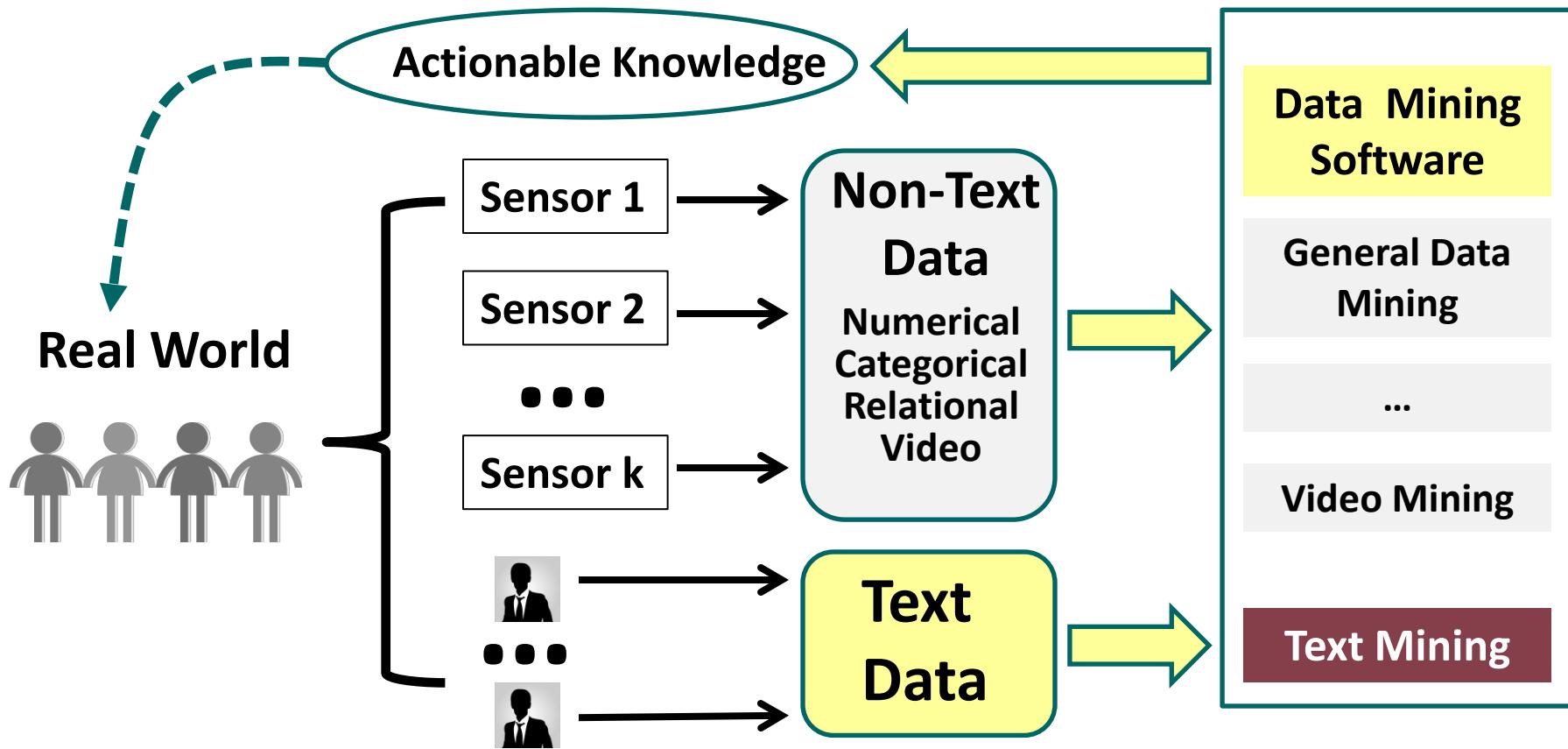
Text Mining and Analytics

- Text mining ≈ Text analytics
- Turn text data into **high-quality information or actionable knowledge**
 - **Minimizes human effort** (on consuming text data)
 - Supplies knowledge for **optimal decision making**
- Related to **text retrieval**, which is an essential component in any text mining system
 - Text retrieval can be a preprocessor for text mining
 - Text retrieval is needed for knowledge provenance

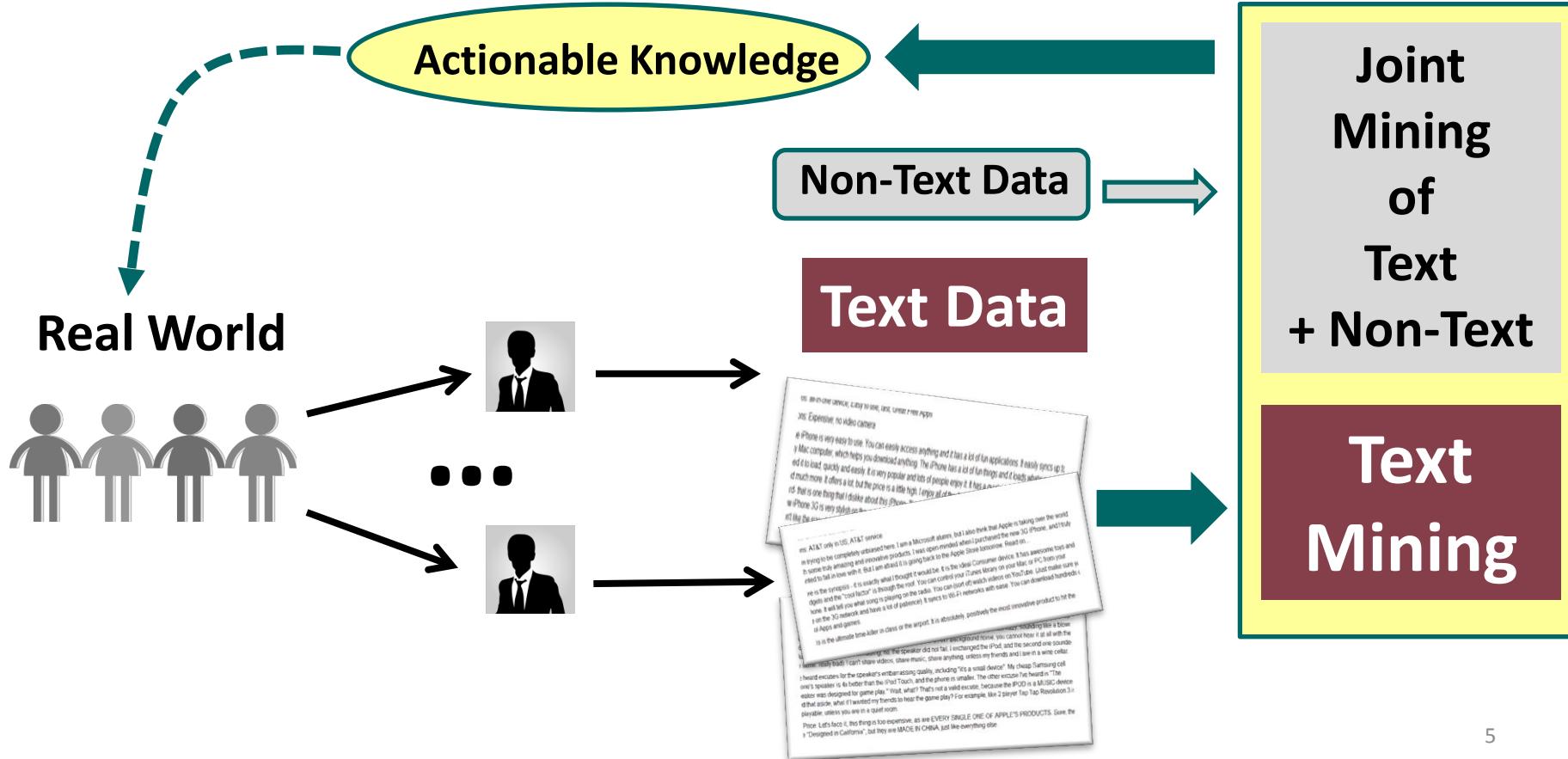
Text vs. Non-Text Data: Humans as Subjective “Sensors”



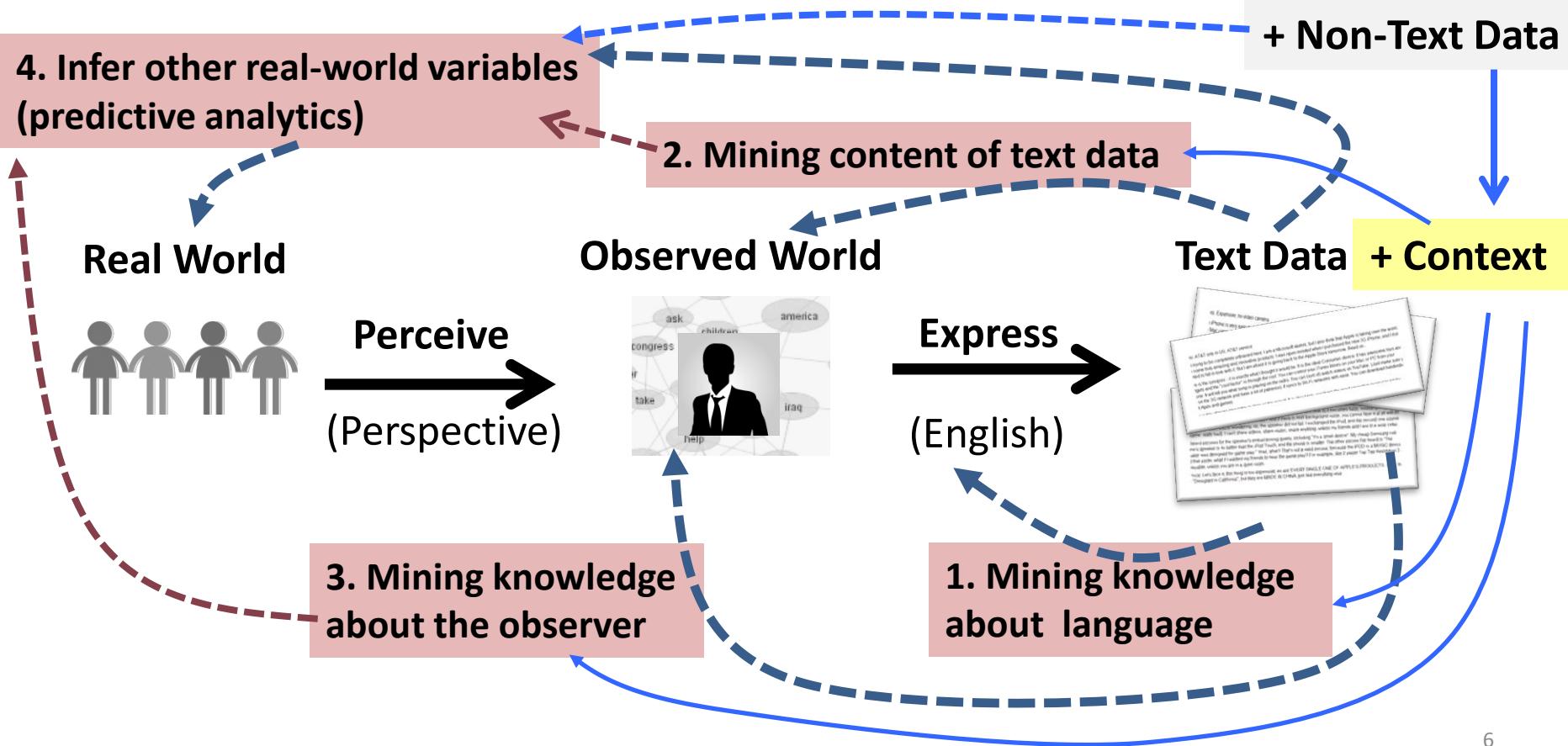
The General Problem of Data Mining



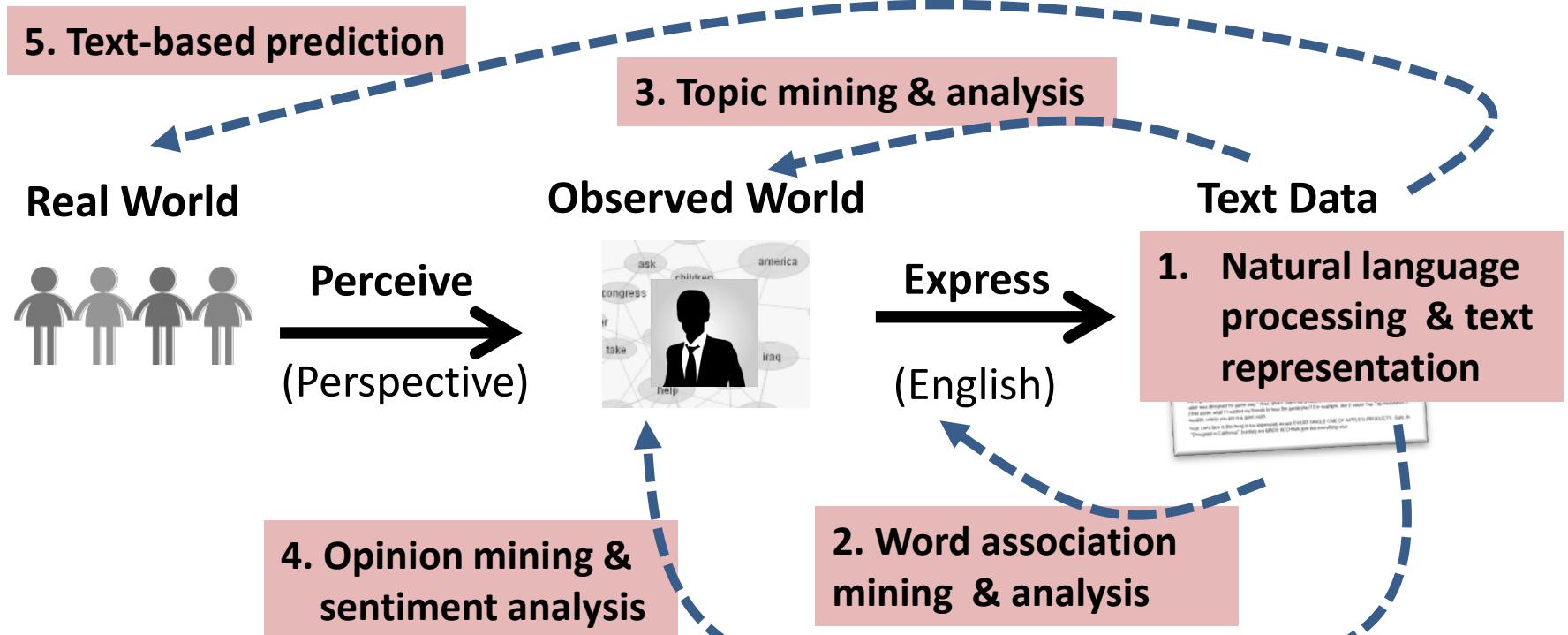
The Problem of Text Mining



Landscape of Text Mining and Analytics



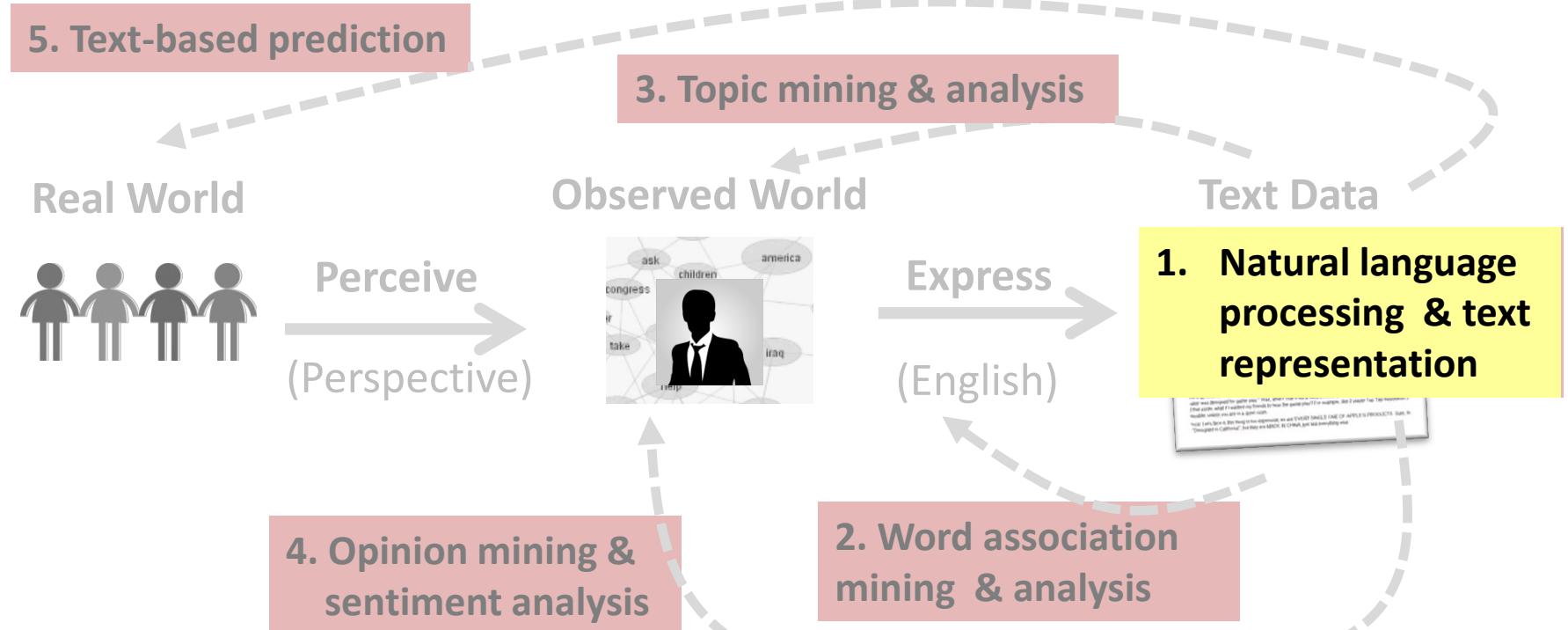
Topics Covered in This Course



Natural Language Content Analysis

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Natural Language Content Analysis



Basic Concepts in NLP

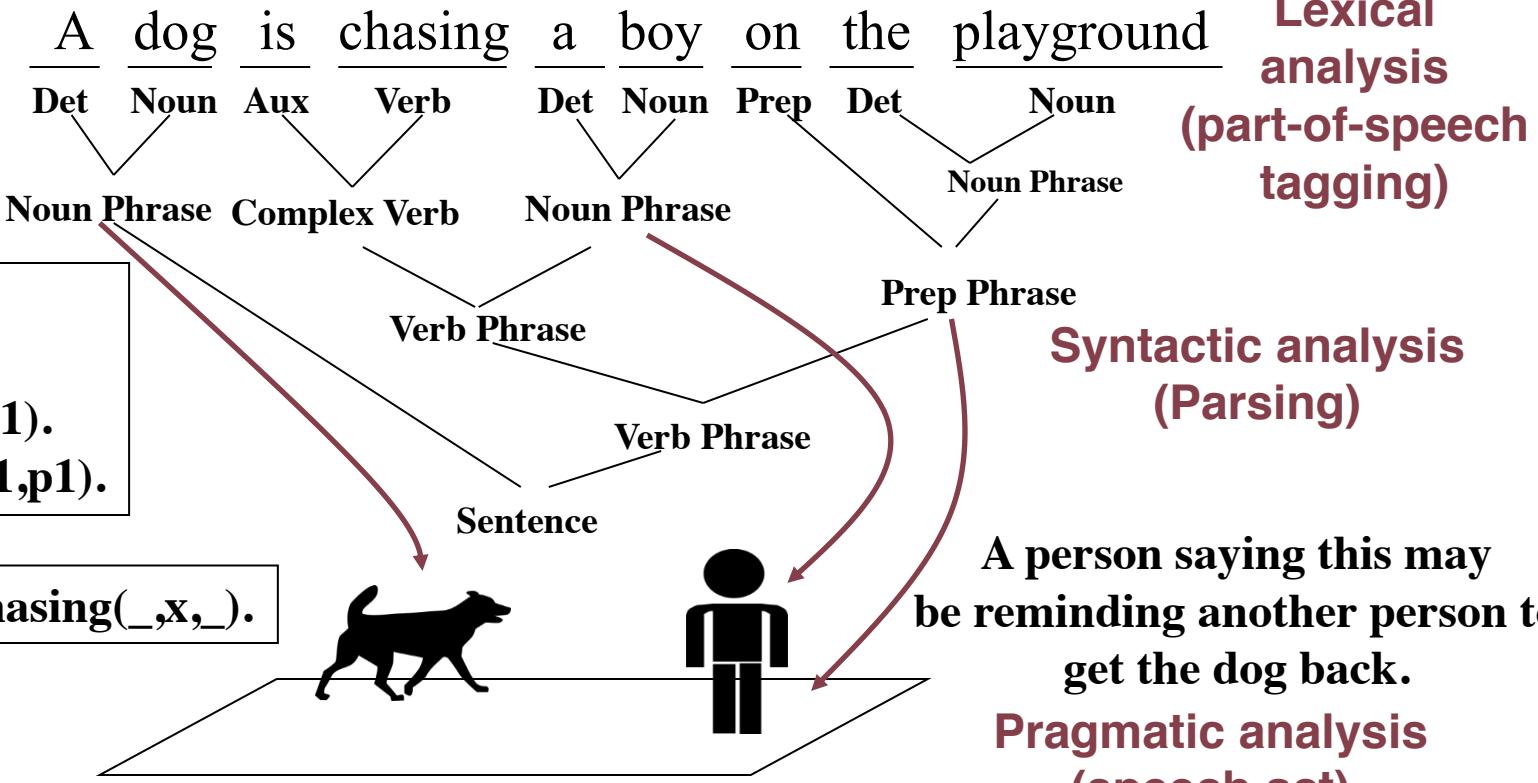
Semantic analysis

Dog(d1).
Boy(b1).
Playground(p1).
Chasing(d1,b1,p1).

+

Scared(x) if Chasing(_,x,_).

Scared(b1)
Inference



Pragmatic analysis
(speech act)

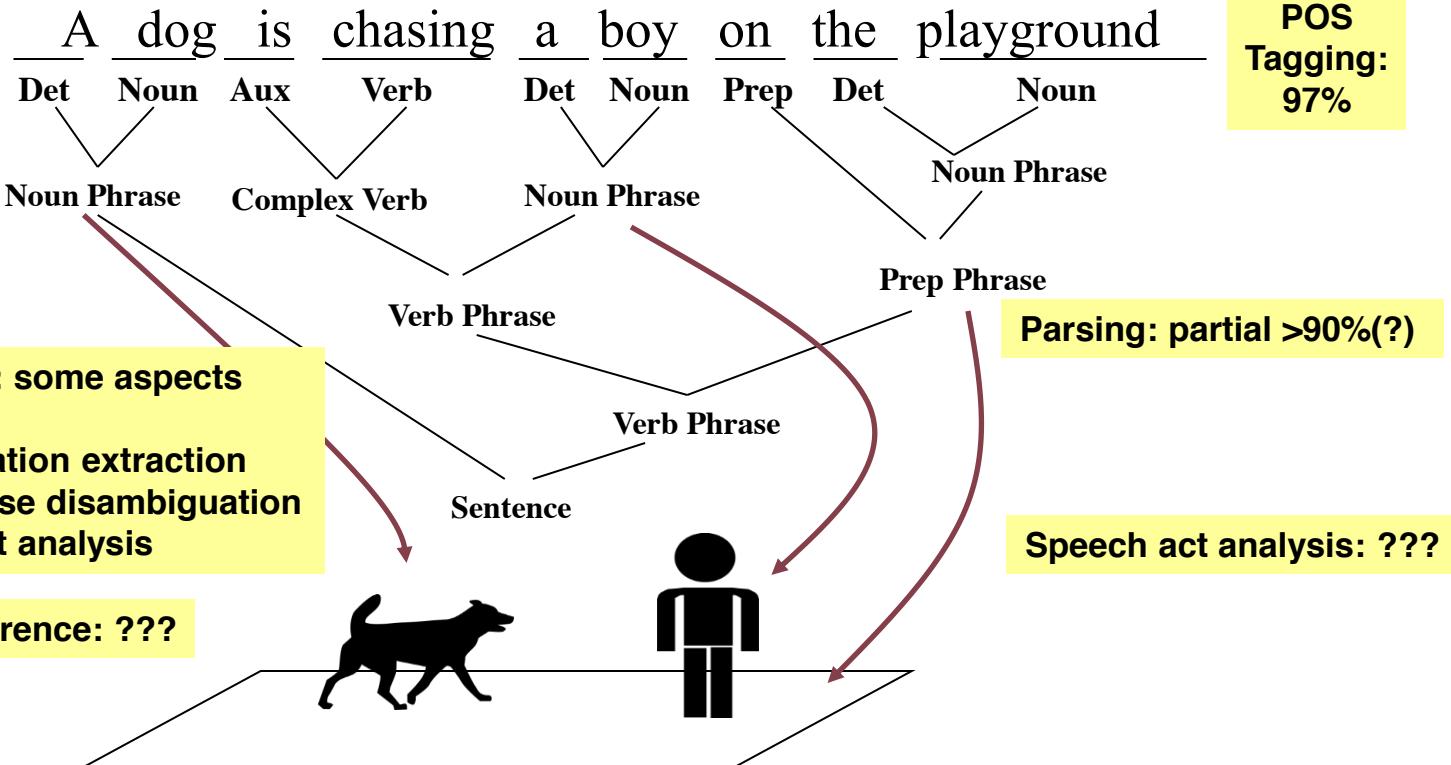
NLP Is Difficult!

- Natural language is designed to make human communication efficient. As a result,
 - we omit a lot of *common sense* knowledge, which we assume the hearer/reader possesses.
 - we keep a lot of ambiguities, which we assume the hearer/reader knows how to resolve.
- This makes EVERY step in NLP hard
 - Ambiguity is a *killer*!
 - Common sense reasoning is pre-required.

Examples of Challenges

- Word-level ambiguity:
 - “design” can be a noun or a verb (ambiguous POS)
 - “root” has multiple meanings (ambiguous sense)
- Syntactic ambiguity:
 - “natural language processing” (modification)
 - “A man saw a boy with a telescope.” (PP Attachment)
- Anaphora resolution: “John persuaded Bill to buy a TV for himself.
(himself = John or Bill?)
- Presupposition: “He has quit smoking” implies that he smoked before.

The State of the Art



What We Can't Do

- 100% POS tagging
 - “He turned off the highway.” vs “He turned off the fan.”
- General complete parsing
 - “A man saw a boy with a telescope.”
- Precise deep semantic analysis
 - Will we ever be able to precisely define the meaning of “own” in “John owns a restaurant”?

Robust and general NLP tends to be *shallow* while *deep* understanding doesn't scale up.

Summary

- NLP is the foundation for text mining
- Computers are far from being able to understand natural language
 - Deep NLP requires common sense knowledge and inferences, thus only working for very limited domains
 - Shallow NLP based on statistical methods can be done in large scale and is thus more broadly applicable
- In practice: statistical NLP as the basis, while humans provide help as needed

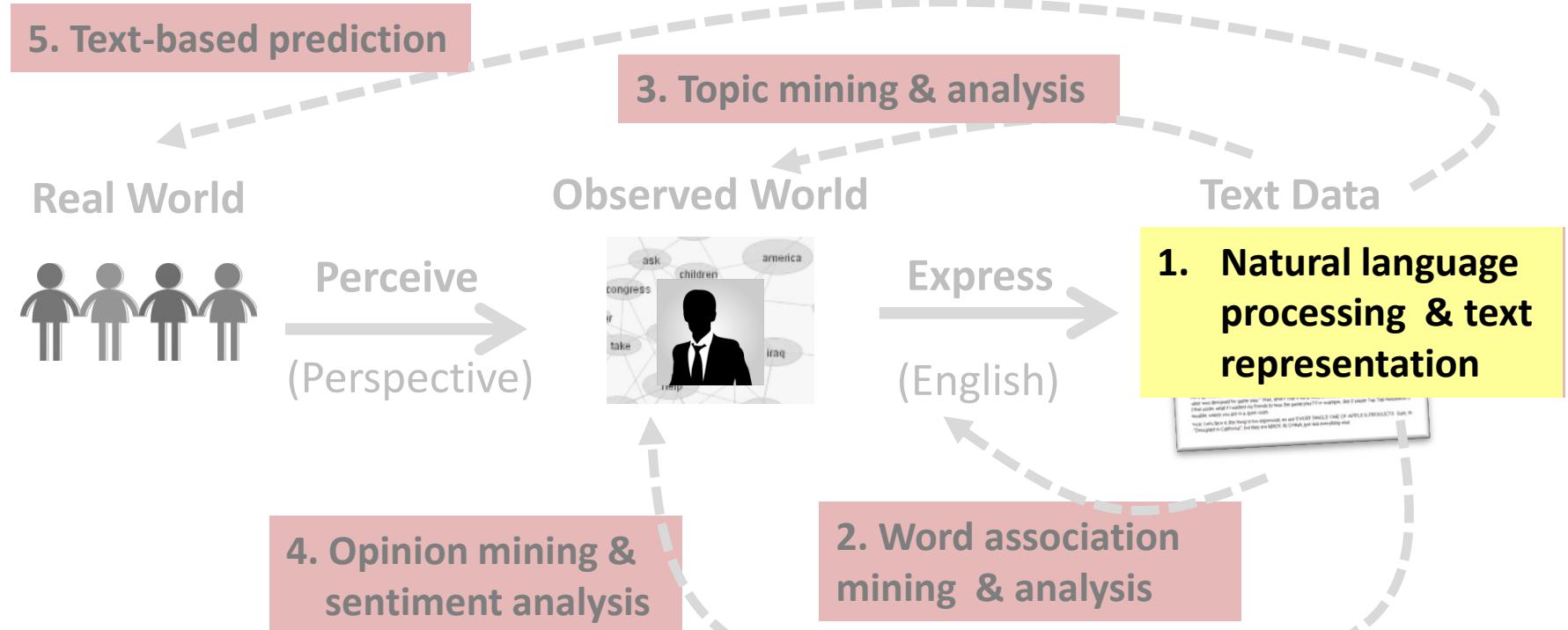
Additional Reading

Manning, Chris and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. Cambridge: MIT Press, 1999.

Natural Language Content Analysis

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Natural Language Content Analysis



Basic Concepts in NLP

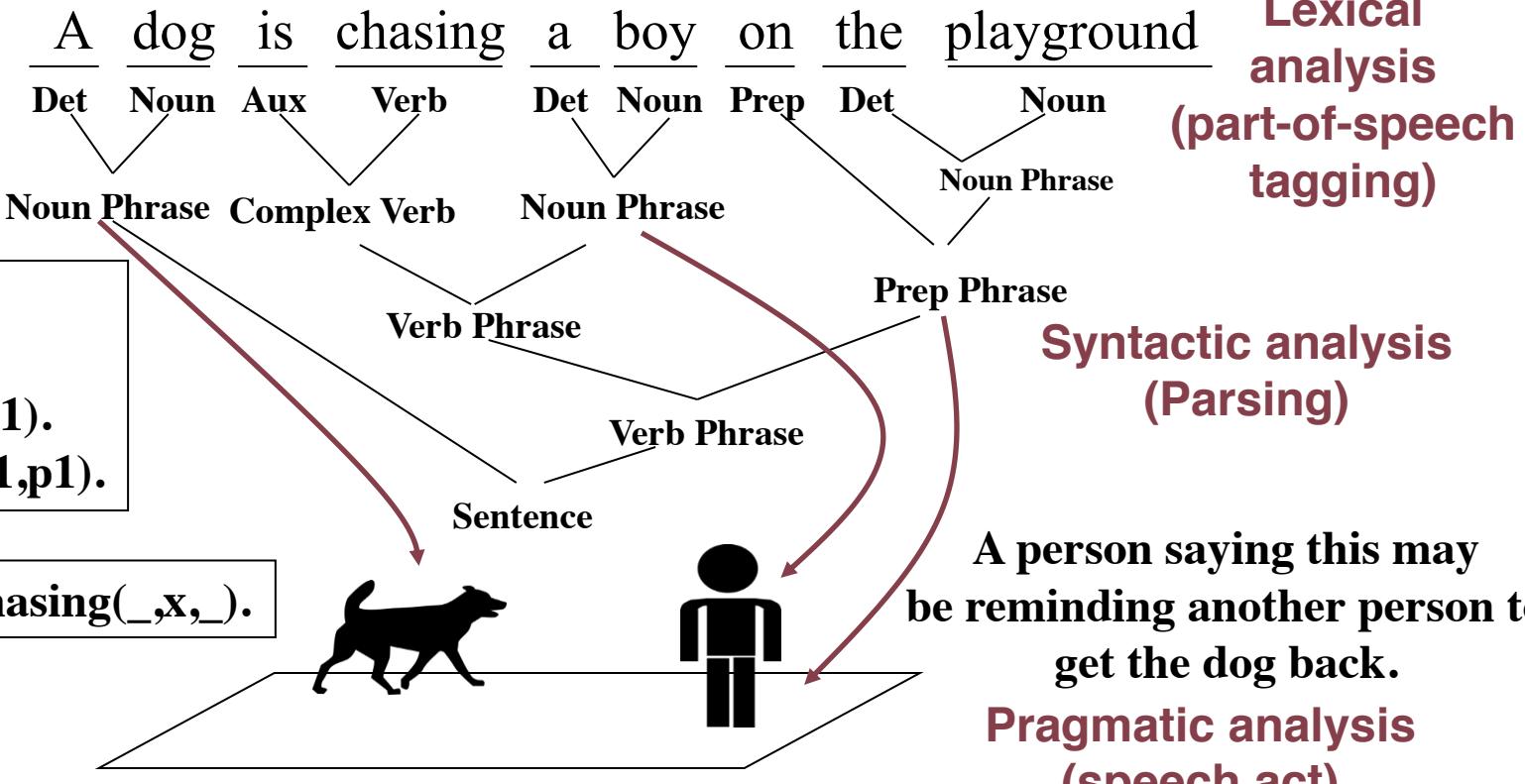
Semantic analysis

Dog(d1).
Boy(b1).
Playground(p1).
Chasing(d1,b1,p1).

+

Scared(x) if Chasing(_,x,_).

Scared(b1)
Inference



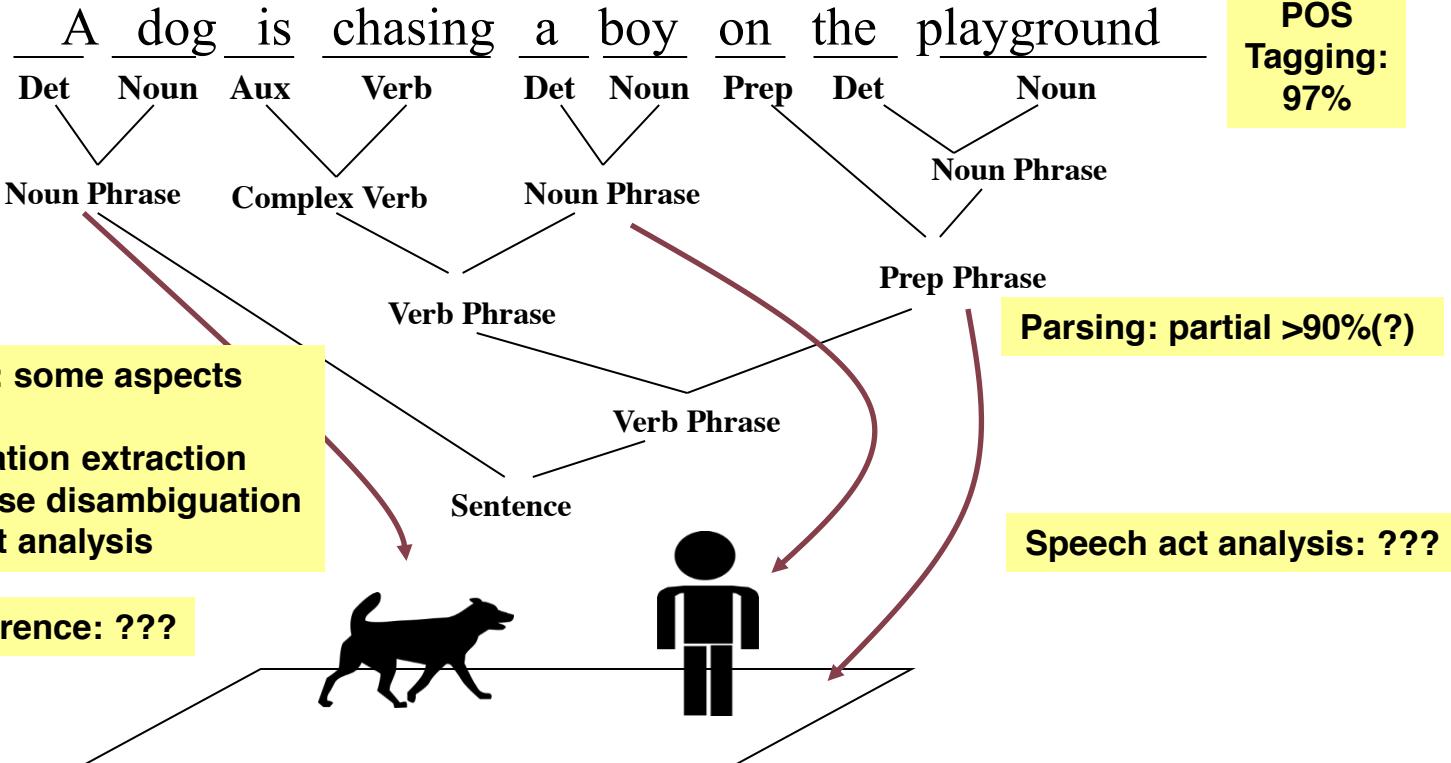
NLP Is Difficult!

- Natural language is designed to make human communication efficient. As a result,
 - we omit a lot of *common sense* knowledge, which we assume the hearer/reader possesses.
 - we keep a lot of ambiguities, which we assume the hearer/reader knows how to resolve.
- This makes EVERY step in NLP hard
 - Ambiguity is a *killer*!
 - Common sense reasoning is pre-required.

Examples of Challenges

- Word-level ambiguity:
 - “design” can be a noun or a verb (ambiguous POS)
 - “root” has multiple meanings (ambiguous sense)
- Syntactic ambiguity:
 - “natural language processing” (modification)
 - “A man saw a boy with a telescope.” (PP Attachment)
- Anaphora resolution: “John persuaded Bill to buy a TV for himself.
(himself = John or Bill?)
- Presupposition: “He has quit smoking” implies that he smoked before.

The State of the Art



What We Can't Do

- 100% POS tagging
 - “He turned off the highway.” vs “He turned off the fan.”
- General complete parsing
 - “A man saw a boy with a telescope.”
- Precise deep semantic analysis
 - Will we ever be able to precisely define the meaning of “own” in “John owns a restaurant”?

Robust and general NLP tends to be *shallow* while *deep* understanding doesn't scale up.

Summary

- NLP is the foundation for text mining
- Computers are far from being able to understand natural language
 - Deep NLP requires common sense knowledge and inferences, thus only working for very limited domains
 - Shallow NLP based on statistical methods can be done in large scale and is thus more broadly applicable
- In practice: statistical NLP as the basis, while humans provide help as needed

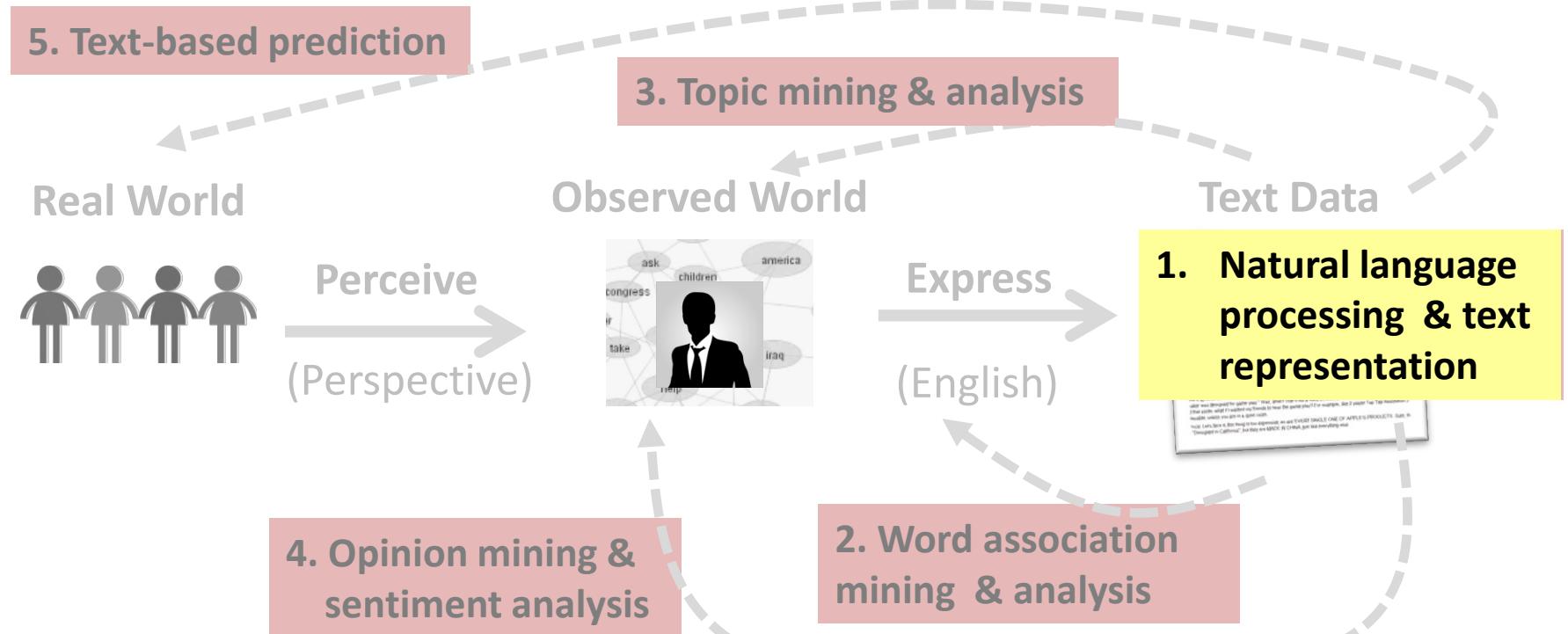
Additional Reading

Manning, Chris and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. Cambridge: MIT Press, 1999.

Text Representation

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Text Representation

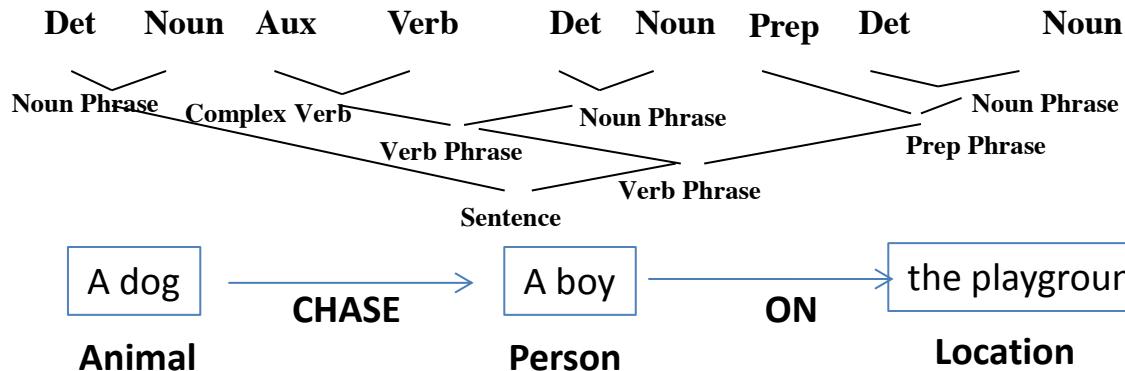


A dog is chasing a boy on the playground

String of characters

A dog is chasing a boy on the playground

Sequence of words



+ POS tags

+ Syntactic structures

Dog(d1). Boy(b1). Playground(p1). Chasing(d1,b1,p1).

+ Logic predicates

Speech Act = REQUEST

+ Speech acts

Deeper NLP: requires more human effort; less accurate

Closer to knowledge representation

Text Representation and Enabled Analysis

This course

Text Rep	Generality	Enabled Analysis	Examples of Application
String		String processing	Compression
Words		Word relation analysis; topic analysis; sentiment analysis	Thesaurus discovery; topic and opinion related applications
+ Syntactic structures		Syntactic graph analysis	Stylistic analysis; structure-based feature extraction
+ Entities & relations		Knowledge graph analysis; information network analysis	Discovery of knowledge and opinions about specific entities
+ Logic predicates		Integrative analysis of scattered knowledge; logic inference	Knowledge assistant for biologists

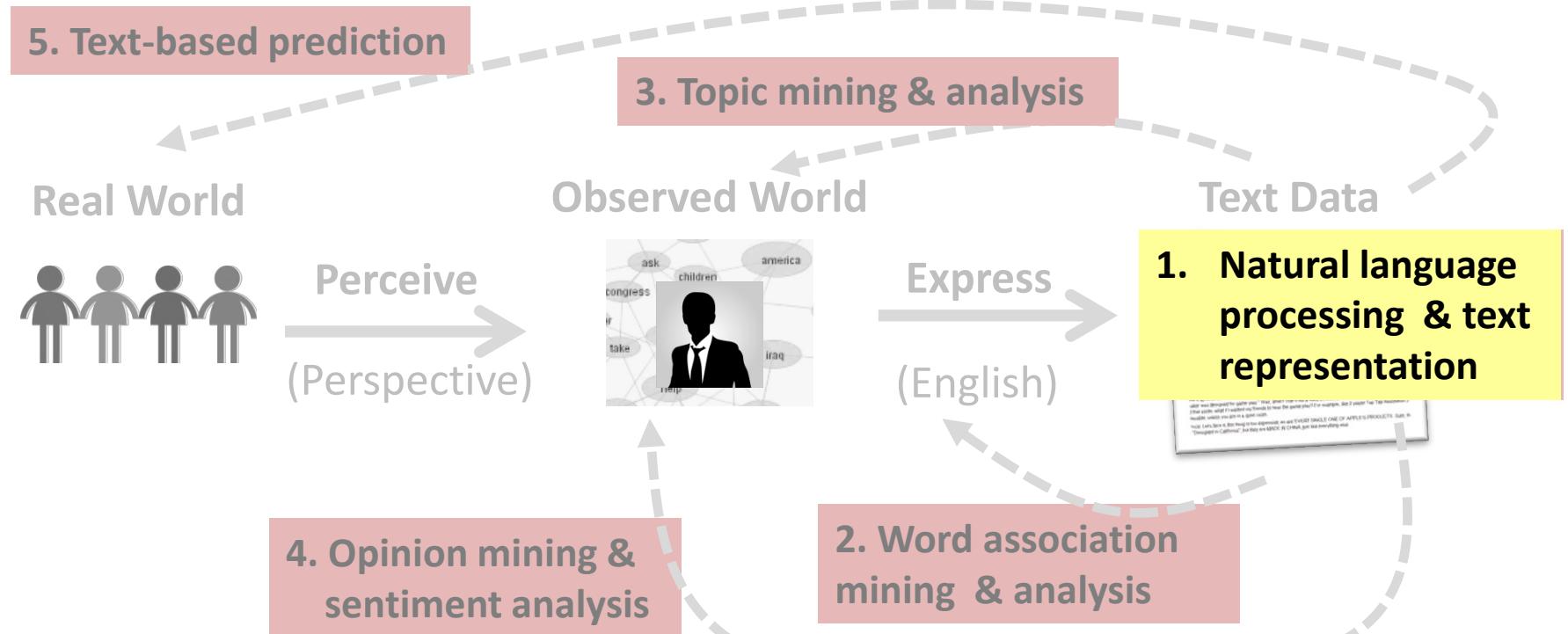
Summary

- Text representation determines what kind of mining algorithms can be applied
- **Multiple ways** of representing text are possible
 - string, words, syntactic structures, entity-relation graphs, predicates...
 - can/should be **combined** in real applications
- This course focuses on **word-based representation**
 - **General and robust**: applicable to any natural language
 - **No/little manual effort**
 - “**Surprisingly**” **powerful** for many applications (not all!)
 - **Can be combined** with more sophisticated representations

Text Representation

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Text Representation

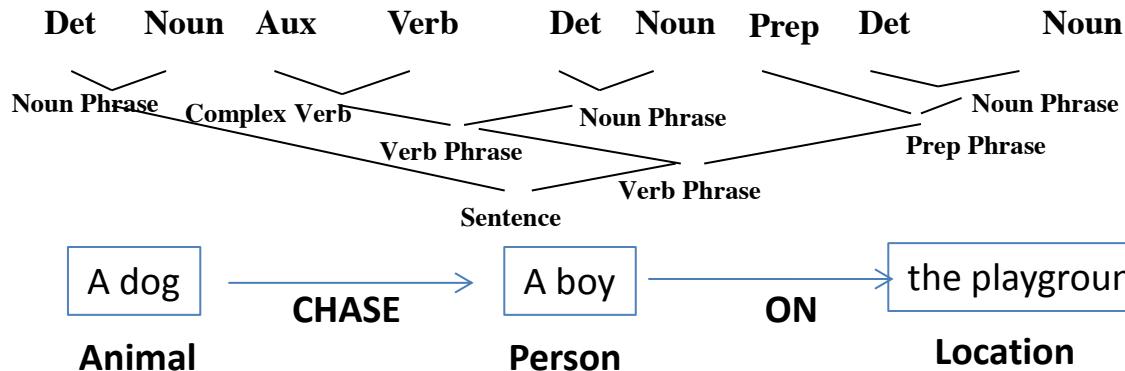


A dog is chasing a boy on the playground

String of characters

A dog is chasing a boy on the playground

Sequence of words



+ POS tags

+ Syntactic structures

Dog(d1). Boy(b1). Playground(p1). Chasing(d1,b1,p1).

+ Logic predicates

Speech Act = REQUEST

+ Speech acts

Deeper NLP: requires more human effort; less accurate

Closer to knowledge representation

Text Representation and Enabled Analysis

This course

Text Rep	Generality	Enabled Analysis	Examples of Application
String		String processing	Compression
Words		Word relation analysis; topic analysis; sentiment analysis	Thesaurus discovery; topic and opinion related applications
+ Syntactic structures		Syntactic graph analysis	Stylistic analysis; structure-based feature extraction
+ Entities & relations		Knowledge graph analysis; information network analysis	Discovery of knowledge and opinions about specific entities
+ Logic predicates		Integrative analysis of scattered knowledge; logic inference	Knowledge assistant for biologists

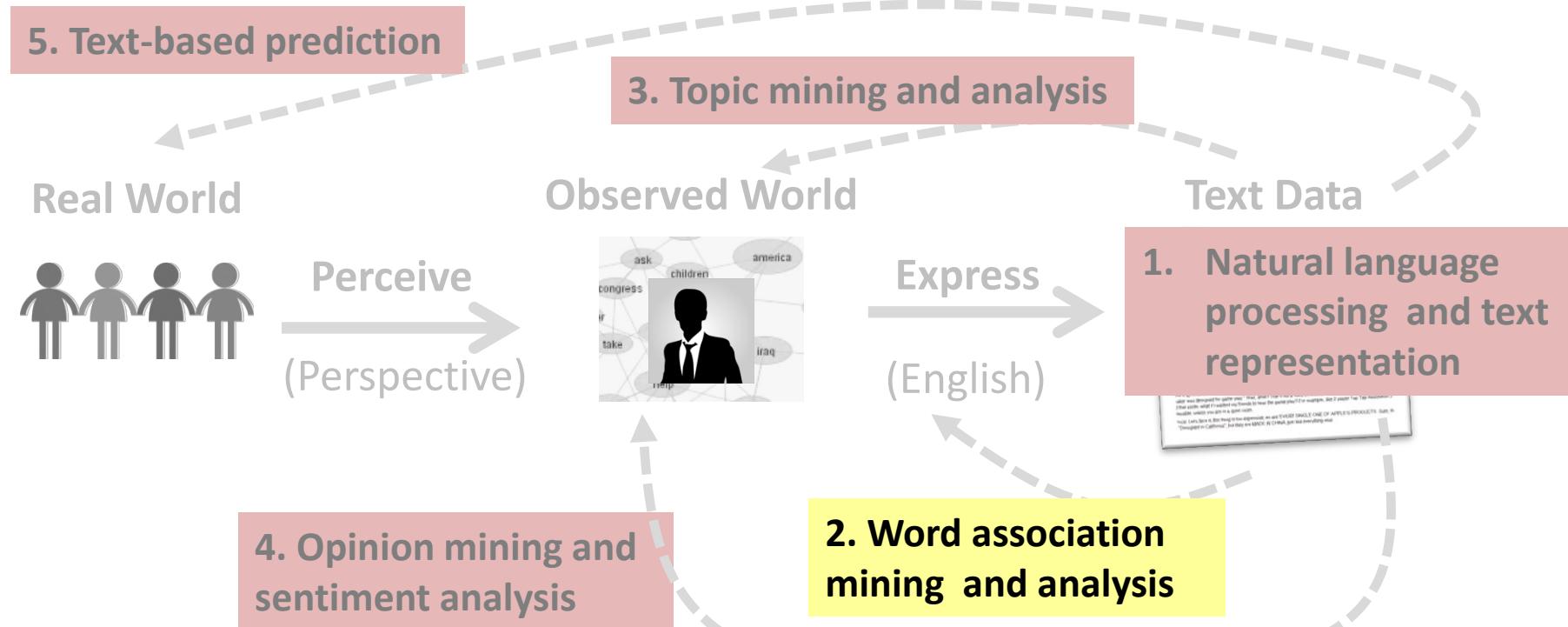
Summary

- Text representation determines what kind of mining algorithms can be applied
- **Multiple ways** of representing text are possible
 - string, words, syntactic structures, entity-relation graphs, predicates...
 - can/should be **combined** in real applications
- This course focuses on **word-based representation**
 - **General and robust**: applicable to any natural language
 - **No/little manual effort**
 - “**Surprisingly**” **powerful** for many applications (not all!)
 - **Can be combined** with more sophisticated representations

Word Association Mining and Analysis

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Word Association Mining & Analysis



Outline

- What is a word association?
- Why mine word associations?
- How to mine word associations?

Basic Word Relations: Paradigmatic vs. Syntagmatic

- Paradigmatic: A & B have paradigmatic relation if they can be substituted for each other (i.e., A & B are in the same class)
 - E.g., “cat” and “dog”; “Monday” and “Tuesday”
- Syntagmatic: A & B have syntagmatic relation if they can be combined with each other (i.e., A & B are related semantically)
 - E.g., “cat” and “sit”; “car” and “drive”
- These two basic and complementary relations can be generalized to describe relations of any items in a language

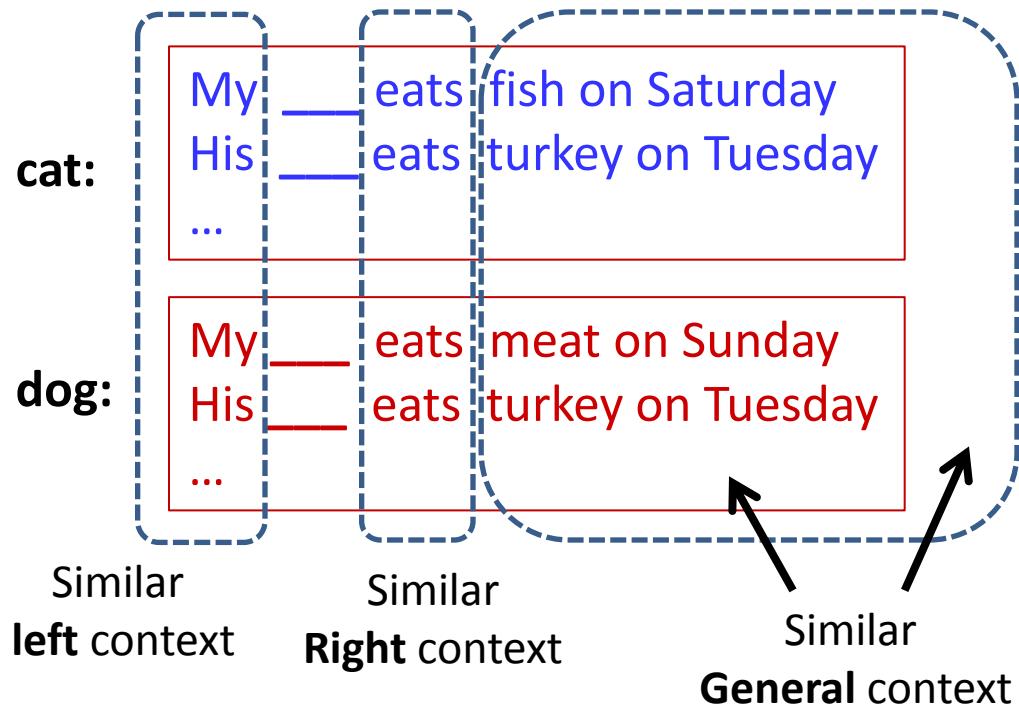
Why Mine Word Associations?

- They are useful for improving accuracy of many NLP tasks
 - POS tagging, parsing, entity recognition, acronym expansion
 - Grammar learning
- They are directly useful for many applications in text retrieval and mining
 - Text retrieval (e.g., use word associations to suggest a variation of a query)
 - Automatic construction of topic map for browsing: words as nodes and associations as edges
 - Compare and summarize opinions (e.g., what words are most strongly associated with “battery” in positive and negative reviews about iPhone 6, respectively?)

Mining Word Associations: Intuitions

Paradigmatic: similar context

My **cat** eats fish on Saturday
His **cat** eats turkey on Tuesday
My **dog** eats meat on Sunday
His **dog** eats turkey on Tuesday
...



How similar are context ("cat") and context ("dog")?

How similar are context ("cat") and context ("computer")?

Mining Word Associations: Intuitions

Syntagmatic: correlated occurrences

My cat eats fish on Saturday
His cat eats turkey on Tuesday
My dog eats meat on Sunday
His dog eats turkey on Tuesday
...

My	—	eats	—	on Saturday
His	—	eats	—	on Tuesday
My	—	eats	—	on Sunday
His	—	eats	—	on Tuesday
...				

What words tend to occur
to the **left** of “eats”?

What words
to the **right**?

Whenever “eats” occurs, what **other words** also tend to occur?

How helpful is the occurrence of “eats” for predicting occurrence of “meat”?

How helpful is the occurrence of “eats” for predicting occurrence of “text”?

Mining Word Associations: General Ideas

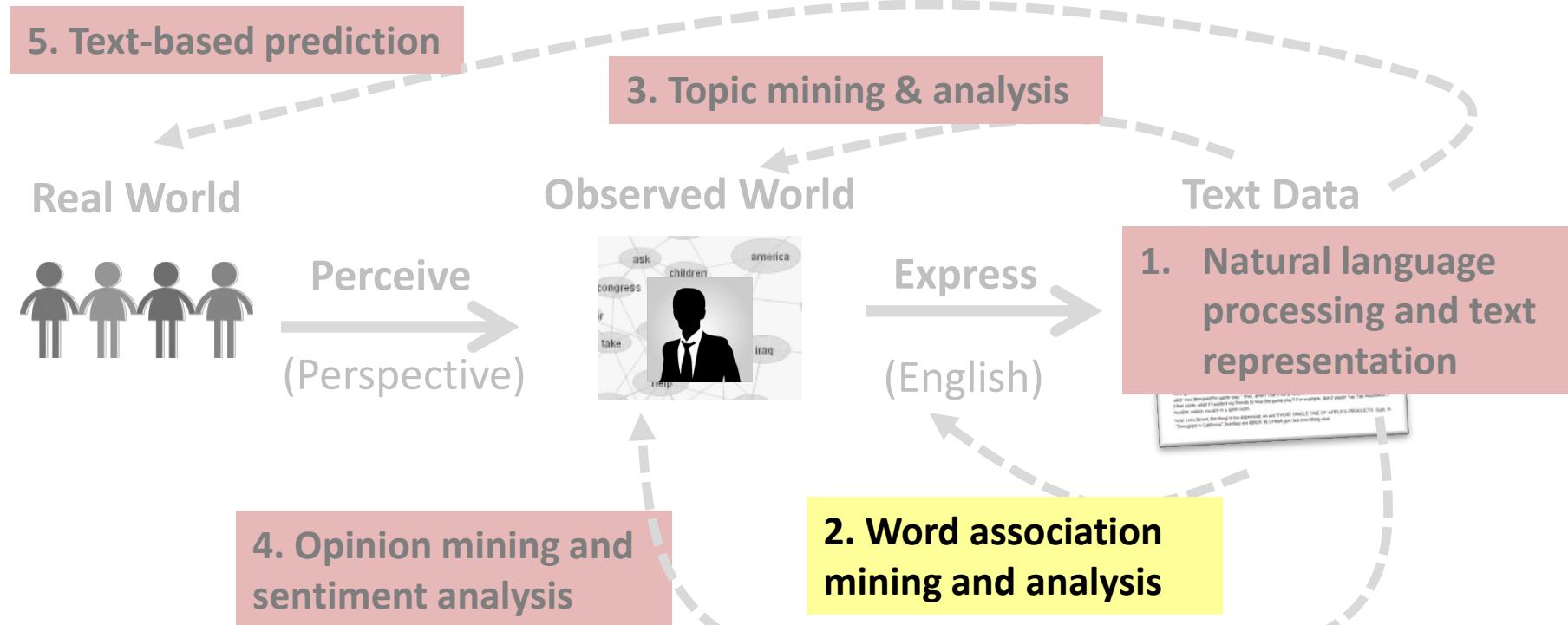
- **Paradigmatic**
 - Represent each word by its context
 - Compute context similarity
 - Words with **high context similarity** likely have paradigmatic relation
- **Syntagmatic**
 - Count how many times two words occur together in a context (e.g., sentence or paragraph)
 - Compare their co-occurrences with their individual occurrences
 - Words with **high co-occurrences but relatively low individual occurrences** likely have syntagmatic relation
- Paradigmatically related words tend to have syntagmatic relation with the same word → **joint discovery** of the two relations
- These ideas can be implemented in many different ways!

Paradigmatic Relation Discovery

Parts 1-3

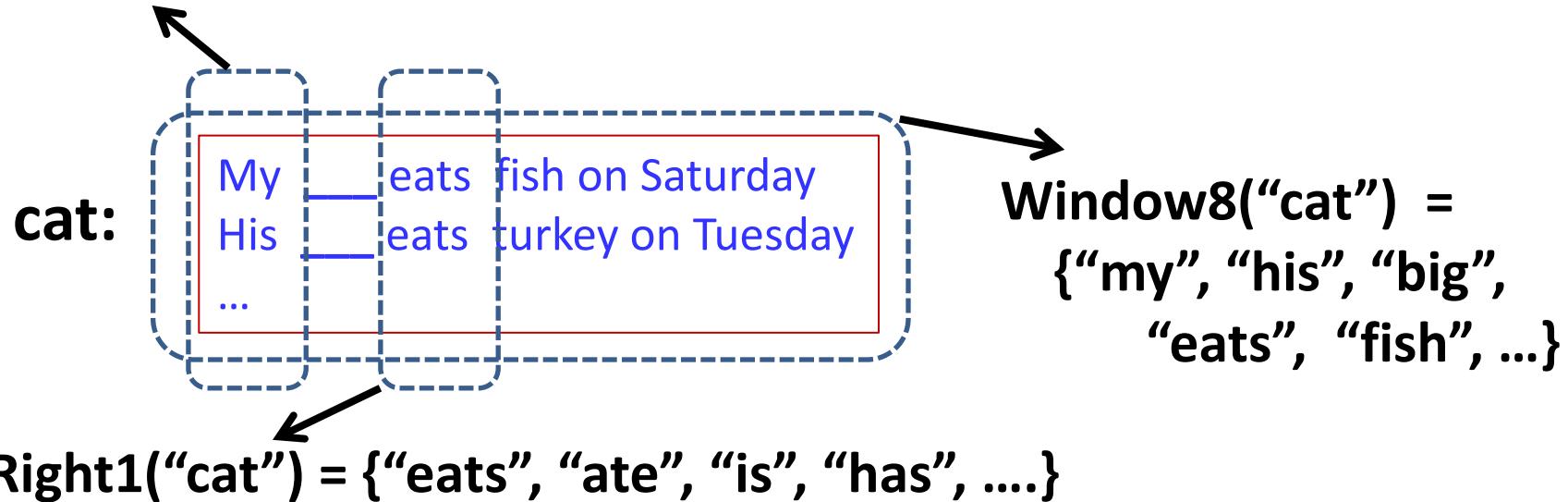
ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Paradigmatic Relation Discovery



Word Context as “Pseudo Document”

$\text{Left1}(\text{"cat"}) = \{\text{"my"}, \text{"his"}, \text{"big"}, \text{"a"}, \text{"the"}, \dots\}$



Context = pseudo document = “bag of words”

Context may contain adjacent or non-adjacent words

Measuring Context Similarity

$\text{Sim}(\text{"Cat"}, \text{"Dog"}) =$

$\text{Sim}(\text{Left1("cat")}, \text{Left1("dog"))}$

$+ \text{Sim}(\text{Right1("cat")}, \text{Right1("dog"))} +$

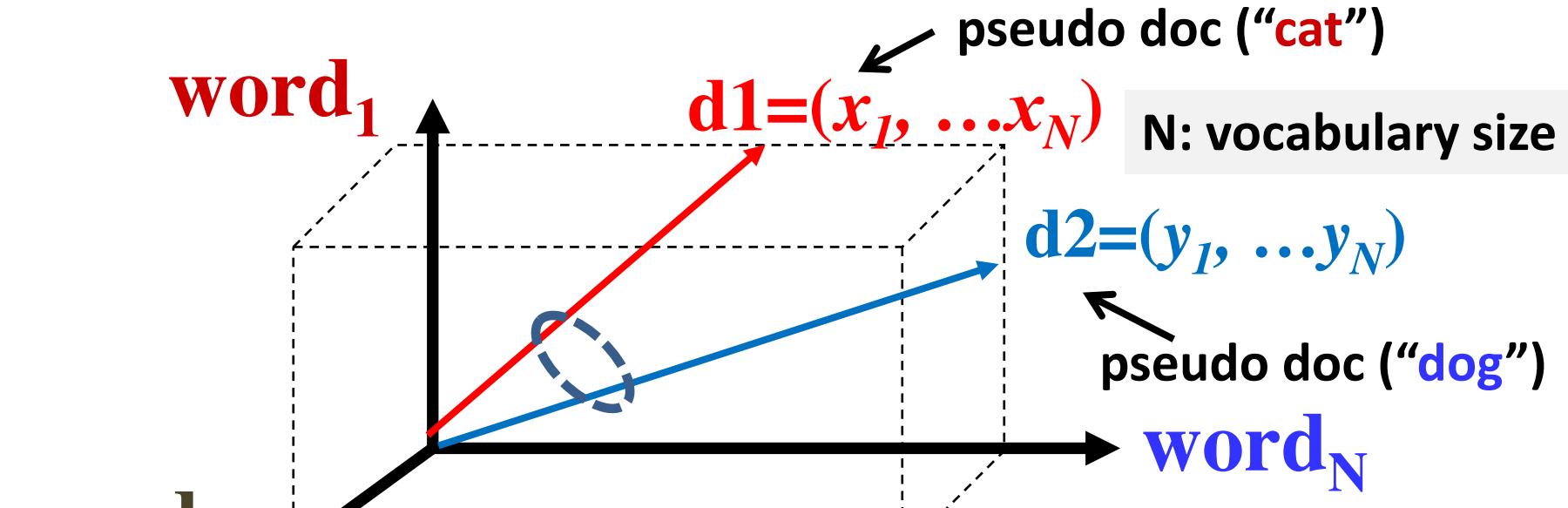
...

$+ \text{Sim}(\text{Window8("cat")}, \text{Window8("dog"))} = ?$

High sim(word1, word2)

→ word1 and word2 are paradigmatically related

Bag of Words → Vector Space Model (VSM)



word₂

Terms:

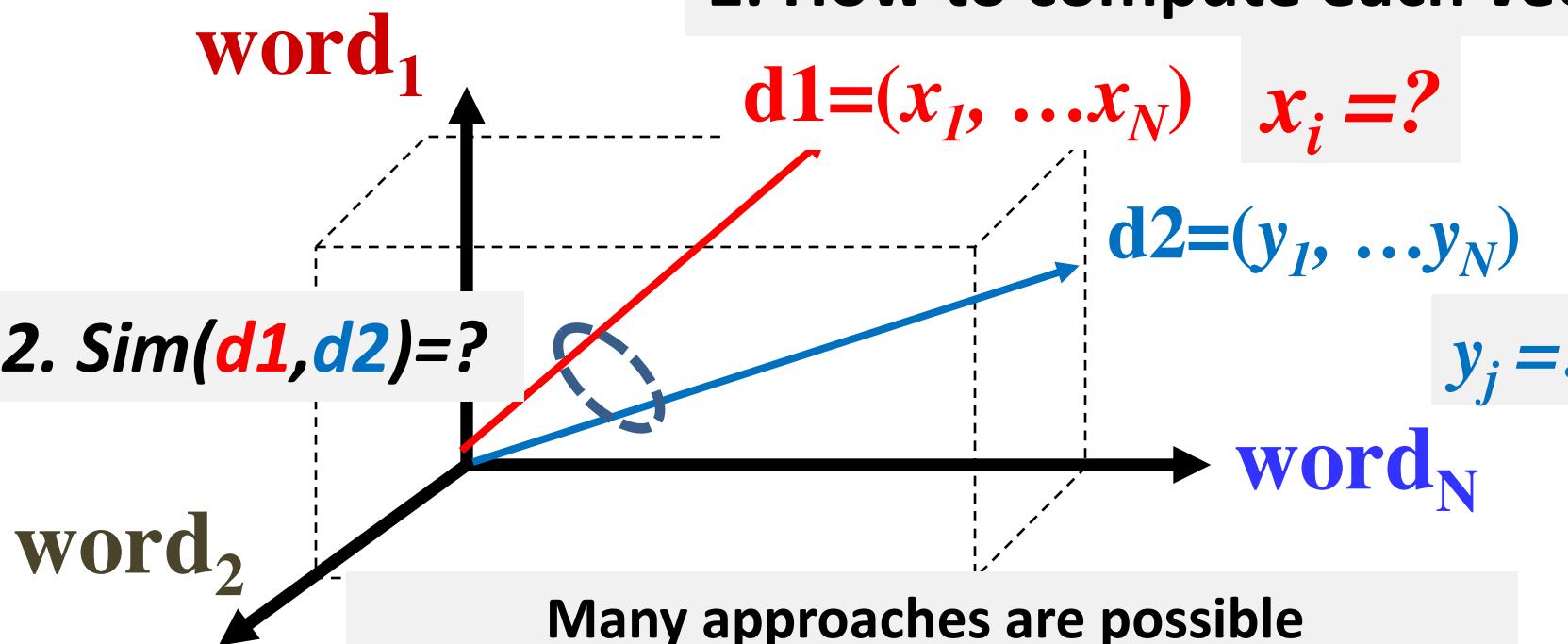
"eats" "ate" "is" "has"

Vector:

(5, 3, 10, 3 )

VSM for Paradigmatic Relation Mining

1. How to compute each vector?



Many approaches are possible
(most developed originally for text retrieval).

Expected Overlap of Words in Context (EOWC)

Probability that a randomly picked word from d_1 is w_i

$$d_1 = (x_1, \dots, x_N)$$

$$d_2 = (y_1, \dots, y_N)$$

$$x_i = c(w_i, d_1) / |d_1|$$

$$y_i = c(w_i, d_2) / |d_2|$$

Count of word w_i in d_1

Total counts of words in d_1

$$\text{Sim}(d_1, d_2) = d_1 \cdot d_2 = x_1 y_1 + \dots + x_N y_N = \sum_{i=1}^N x_i y_i$$

Probability that two randomly picked words from d_1 and d_2 , respectively, are identical.

Would EOWC Work Well?

- Intuitively, it makes sense: The more overlap the two context documents have, the higher the similarity would be.
- However:
 - It favors matching one frequent term very well over matching more distinct terms.
 - It treats every word equally (overlap on “the” isn’t as so meaningful as overlap on “eats”).

Expected Overlap of Words in Context (EOWC)

Probability that a randomly picked word from d_1 is w_i

$$d_1 = (x_1, \dots, x_N)$$

$$d_2 = (y_1, \dots, y_N)$$

$$x_i = c(w_i, d_1) / |d_1|$$

$$y_i = c(w_i, d_2) / |d_2|$$

Count of word w_i in d_1

Total counts of words in d_1

$$\text{Sim}(d_1, d_2) = d_1 \cdot d_2 = x_1 y_1 + \dots + x_N y_N = \sum_{i=1}^N x_i y_i$$

Probability that two randomly picked words from d_1 and d_2 , respectively, are identical.

Improving EOWC with Retrieval Heuristics

- It favors matching one frequent term very well over matching more distinct terms.

→ **Sublinear transformation of Term Frequency (TF)**

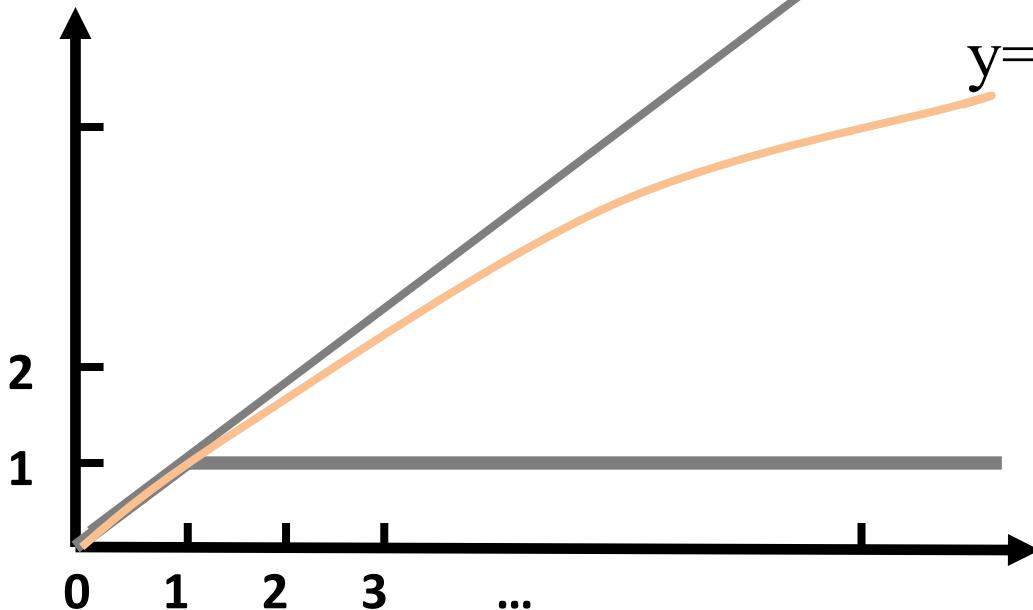
- It treats every word equally (overlap on “the” isn’t as so meaningful as overlap on “eats”).

→ **Reward matching a rare word: IDF term weighting**

TF Transformation: $c(w,d) \rightarrow TF(w,d)$

Term Frequency Weight

$$y = TF(w,d)$$

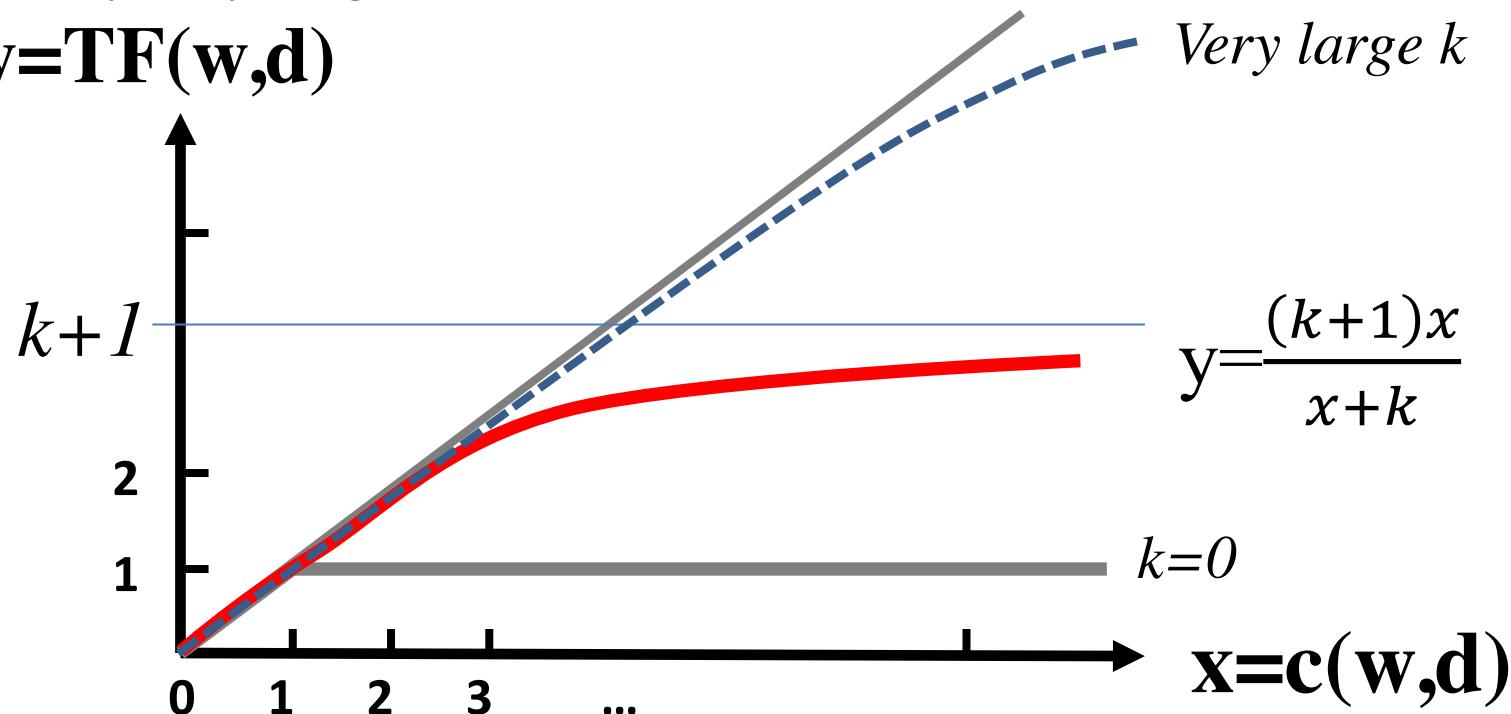


0/1 bit vector
(ignore counts)
 $x = c(w,d)$

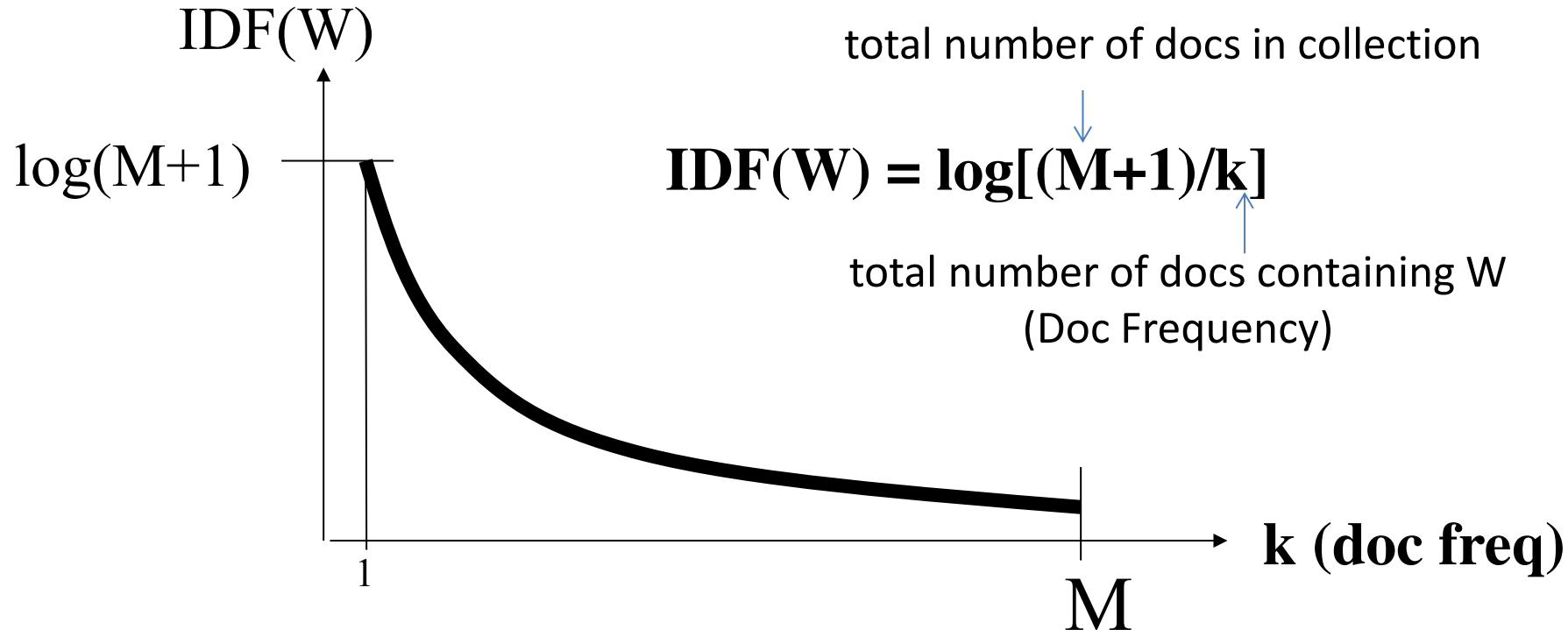
TF Transformation: BM25 Transformation

Term Frequency Weight

$$y = \text{TF}(w, d)$$



IDF Weighting: Penalizing Popular Terms



Adapting BM25 Retrieval Model for Paradigmatic Relation Mining

$$d1 = (x_1, \dots, x_N) \quad BM25(w_i, d1) = \frac{(k+1)c(w_i, d1)}{c(w_i, d1) + k(1 - b + b * |d1| / avdl)}$$

$$x_i = \frac{BM25(w_i, d1)}{\sum_{j=1}^N BM25(w_j, d1)}$$

$$b \in [0, 1]$$
$$k \in [0, +\infty)$$

$d2 = (y_1, \dots, y_N)$ y_i is defined similarly

$$Sim(d1, d2) = \sum_{i=1}^N IDF(w_i) x_i y_i$$

BM25 can also Discover Syntagmatic Relations

$d1 = (x_1, \dots, x_N)$

$$BM25(w_i, d1) = \frac{(k+1)c(w_i, d1)}{c(w_i, d1) + k(1 - b + b * |d1| / avdl)}$$

$$x_i = \frac{BM25(w_i, d1)}{\sum_{j=1}^N BM25(w_j, d1)}$$

$$b \in [0, 1]$$

$$k \in [0, +\infty)$$

IDF-weighted $d1 = (x_1 * IDF(w_1), \dots, x_N * IDF(w_N))$

The highly weighted terms in the context vector of word w are likely syntagmatically related to w.

Summary

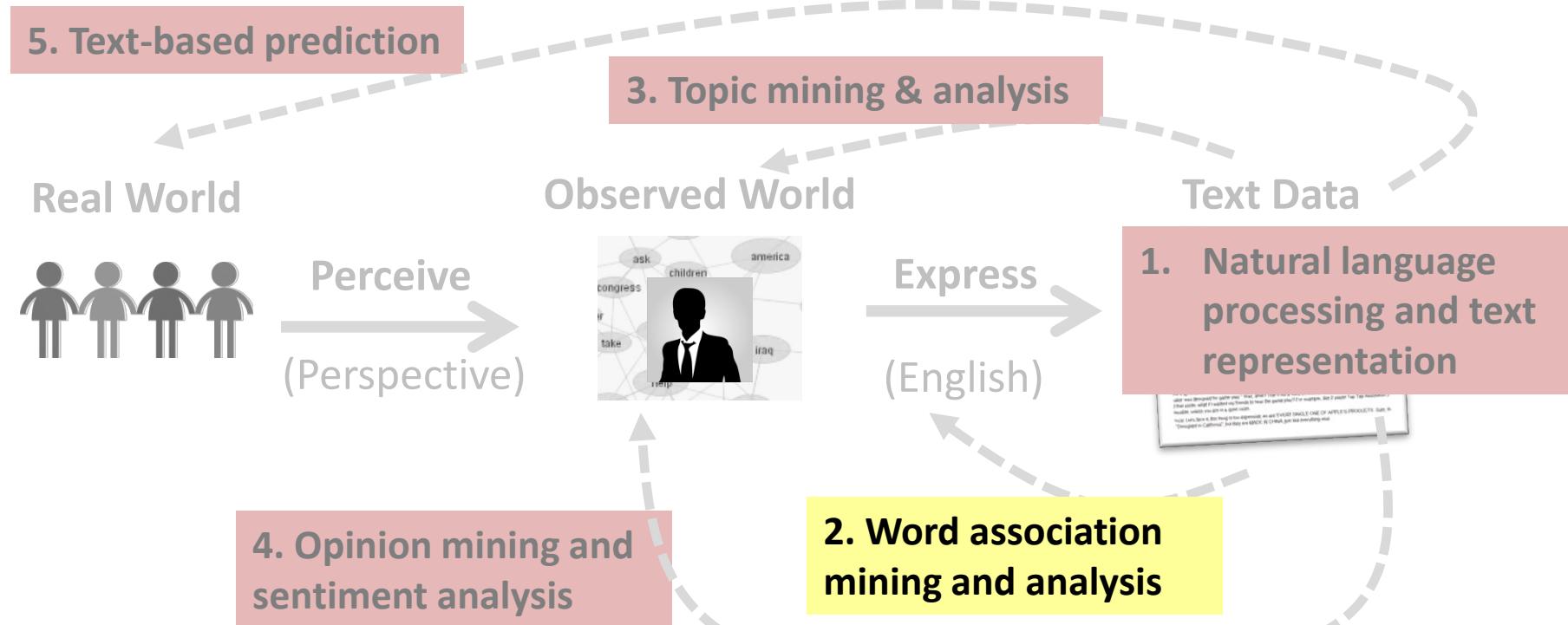
- Main idea for discovering paradigmatic relations:
 - Collecting the context of a candidate word to form a pseudo document (bag of words)
 - Computing similarity of the corresponding context documents of two candidate words
 - Highly similar word pairs can be assumed to have paradigmatic relations
- Many different ways to implement this general idea
- Text retrieval models can be easily adapted for computing similarity of two context documents
 - BM25 + IDF weighting represents the state of the art
 - Syntagmatic relations can also be discovered as a “by product”

Paradigmatic Relation Discovery

Parts 1-3

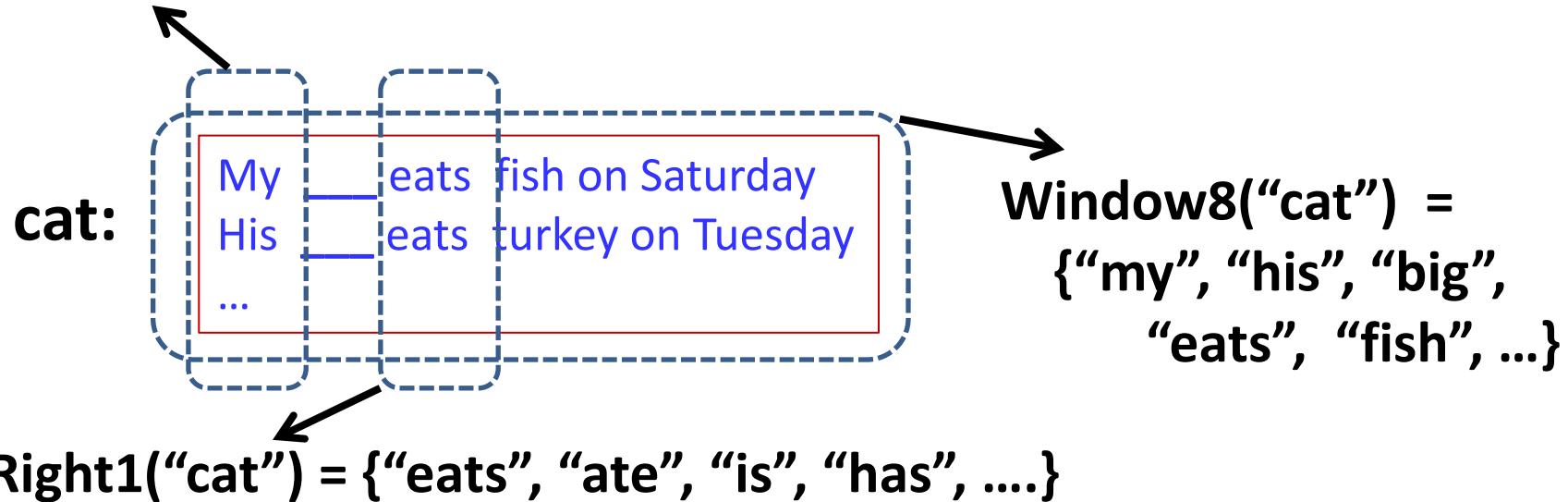
ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Paradigmatic Relation Discovery



Word Context as “Pseudo Document”

$\text{Left1}(\text{"cat"}) = \{\text{"my"}, \text{"his"}, \text{"big"}, \text{"a"}, \text{"the"}, \dots\}$



Context = pseudo document = “bag of words”

Context may contain adjacent or non-adjacent words

Measuring Context Similarity

$\text{Sim}(\text{"Cat"}, \text{"Dog"}) =$

$\text{Sim}(\text{Left1("cat")}, \text{Left1("dog"))}$

$+ \text{Sim}(\text{Right1("cat")}, \text{Right1("dog"))} +$

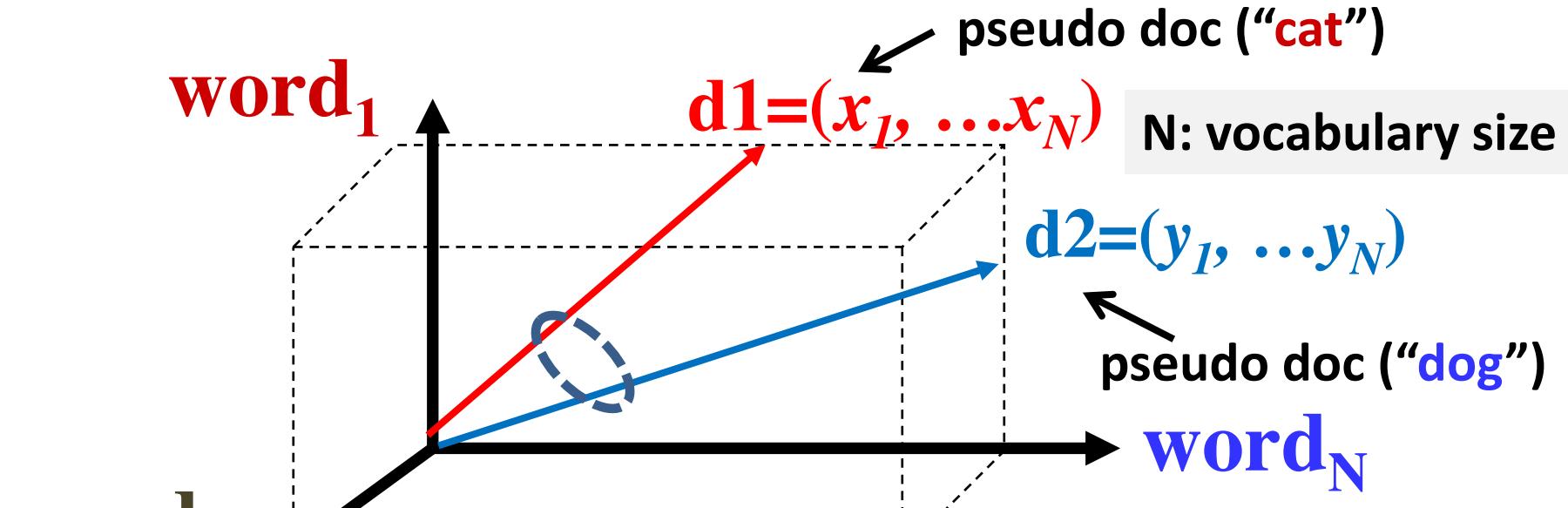
...

$+ \text{Sim}(\text{Window8("cat")}, \text{Window8("dog"))} = ?$

High sim(word1, word2)

→ word1 and word2 are paradigmatically related

Bag of Words → Vector Space Model (VSM)



word₂

Terms:

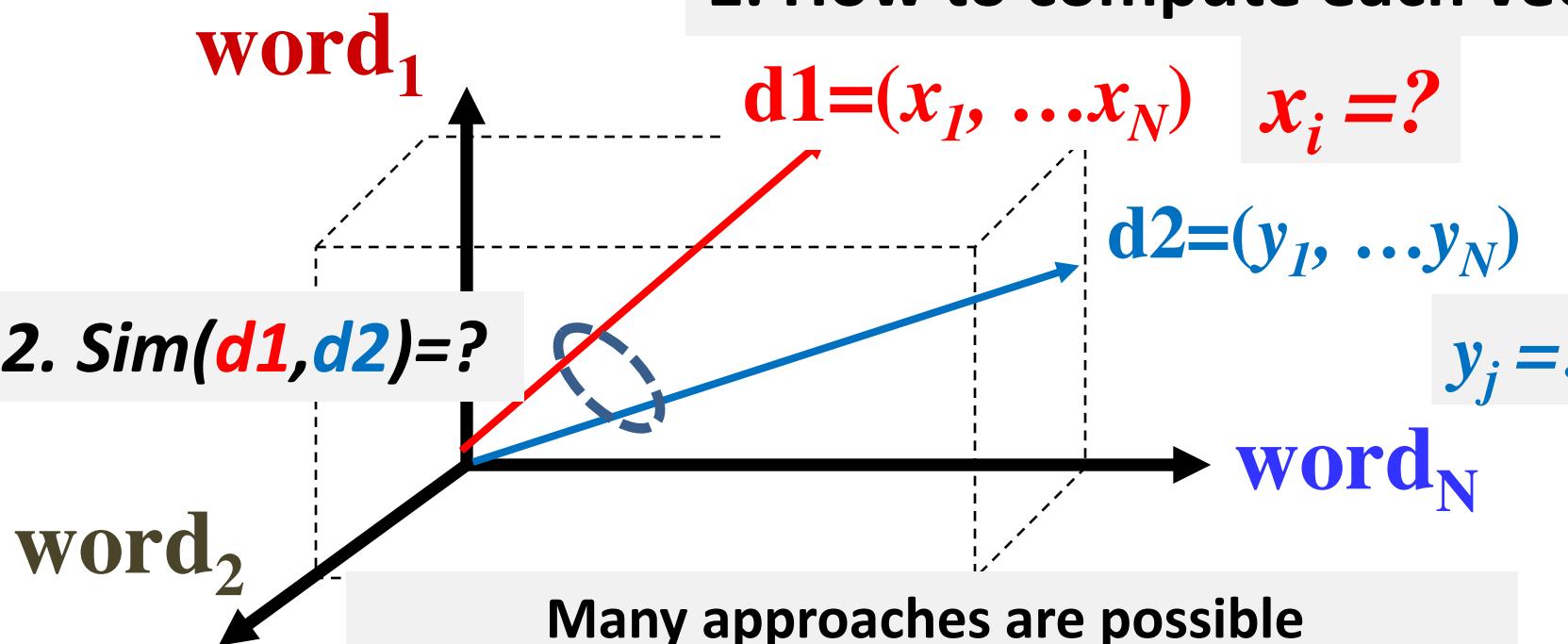
"eats" "ate" "is" "has"

Vector:

(5, 3, 10, 3 )

VSM for Paradigmatic Relation Mining

1. How to compute each vector?



Many approaches are possible
(most developed originally for text retrieval).

Expected Overlap of Words in Context (EOWC)

Probability that a randomly picked word from d_1 is w_i

$$d_1 = (x_1, \dots, x_N)$$

$$d_2 = (y_1, \dots, y_N)$$

$$x_i = c(w_i, d_1) / |d_1|$$

$$y_i = c(w_i, d_2) / |d_2|$$

Count of word w_i in d_1

Total counts of words in d_1

$$\text{Sim}(d_1, d_2) = d_1 \cdot d_2 = x_1 y_1 + \dots + x_N y_N = \sum_{i=1}^N x_i y_i$$

Probability that two randomly picked words from d_1 and d_2 , respectively, are identical.

Would EOWC Work Well?

- Intuitively, it makes sense: The more overlap the two context documents have, the higher the similarity would be.
- However:
 - It favors matching one frequent term very well over matching more distinct terms.
 - It treats every word equally (overlap on “the” isn’t as so meaningful as overlap on “eats”).

Expected Overlap of Words in Context (EOWC)

Probability that a randomly picked word from d_1 is w_i

$$d_1 = (x_1, \dots, x_N)$$

$$d_2 = (y_1, \dots, y_N)$$

$$x_i = c(w_i, d_1) / |d_1|$$

$$y_i = c(w_i, d_2) / |d_2|$$

Count of word w_i in d_1

Total counts of words in d_1

$$\text{Sim}(d_1, d_2) = d_1 \cdot d_2 = x_1 y_1 + \dots + x_N y_N = \sum_{i=1}^N x_i y_i$$

Probability that two randomly picked words from d_1 and d_2 , respectively, are identical.

Improving EOWC with Retrieval Heuristics

- It favors matching one frequent term very well over matching more distinct terms.

→ **Sublinear transformation of Term Frequency (TF)**

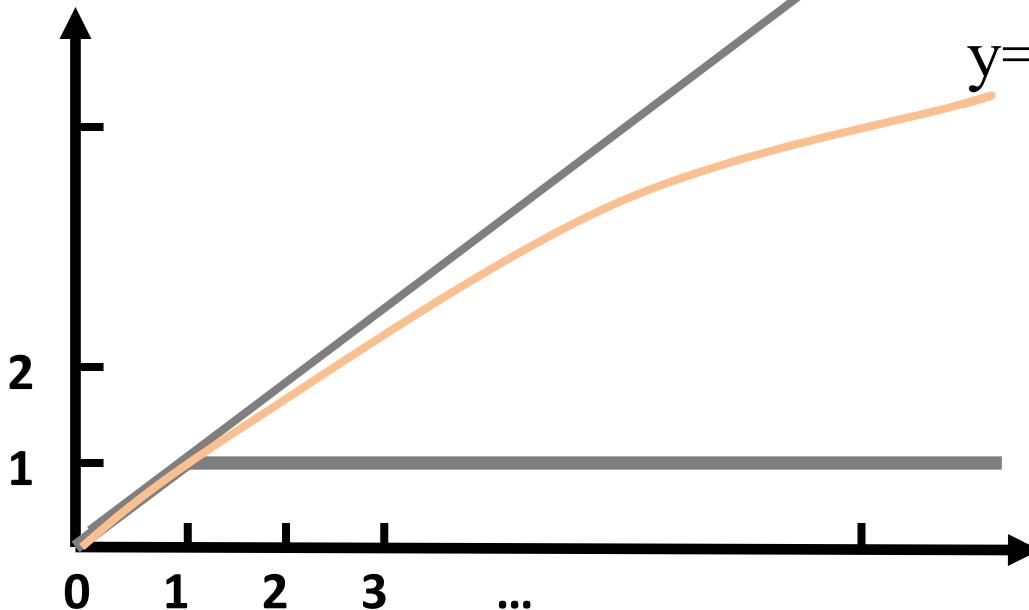
- It treats every word equally (overlap on “the” isn’t as so meaningful as overlap on “eats”).

→ **Reward matching a rare word: IDF term weighting**

TF Transformation: $c(w,d) \rightarrow TF(w,d)$

Term Frequency Weight

$$y = TF(w,d)$$

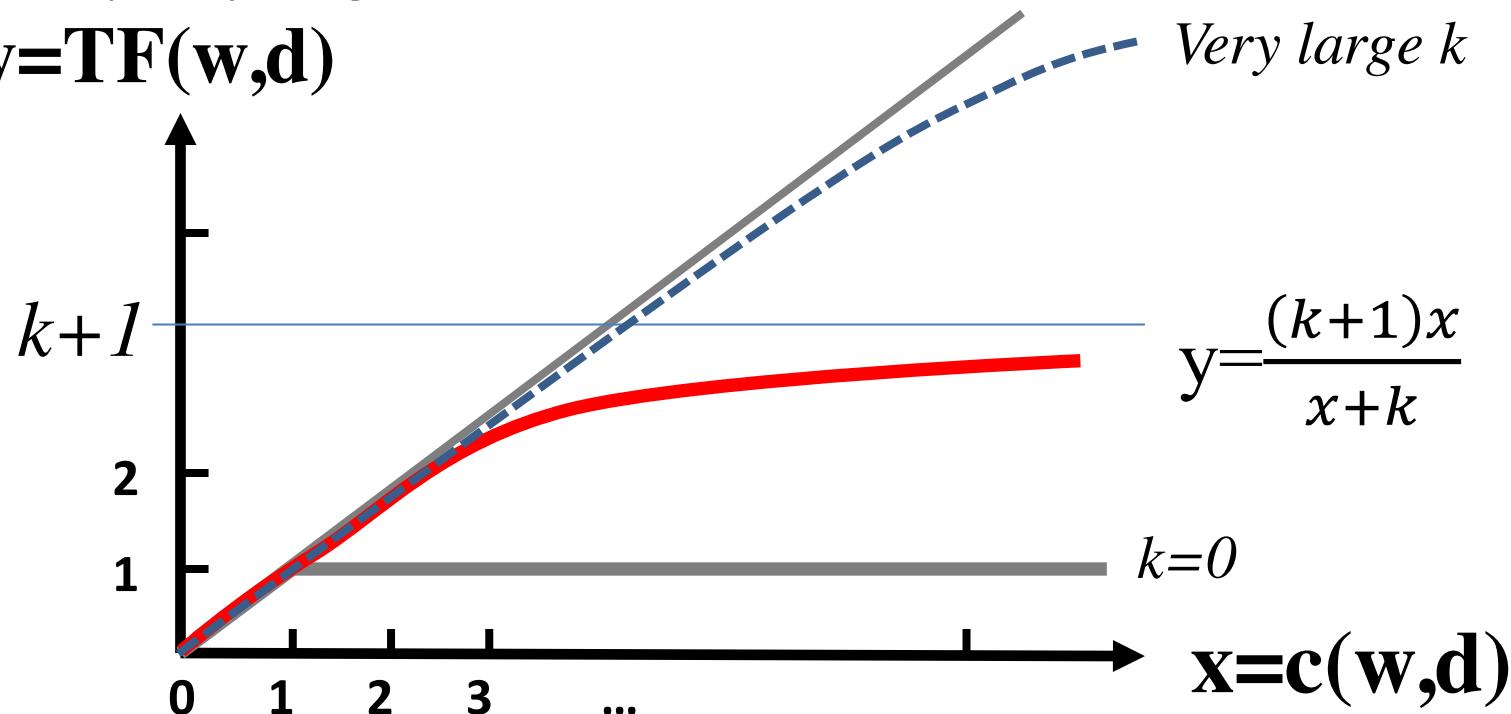


0/1 bit vector
(ignore counts)
 $x = c(w,d)$

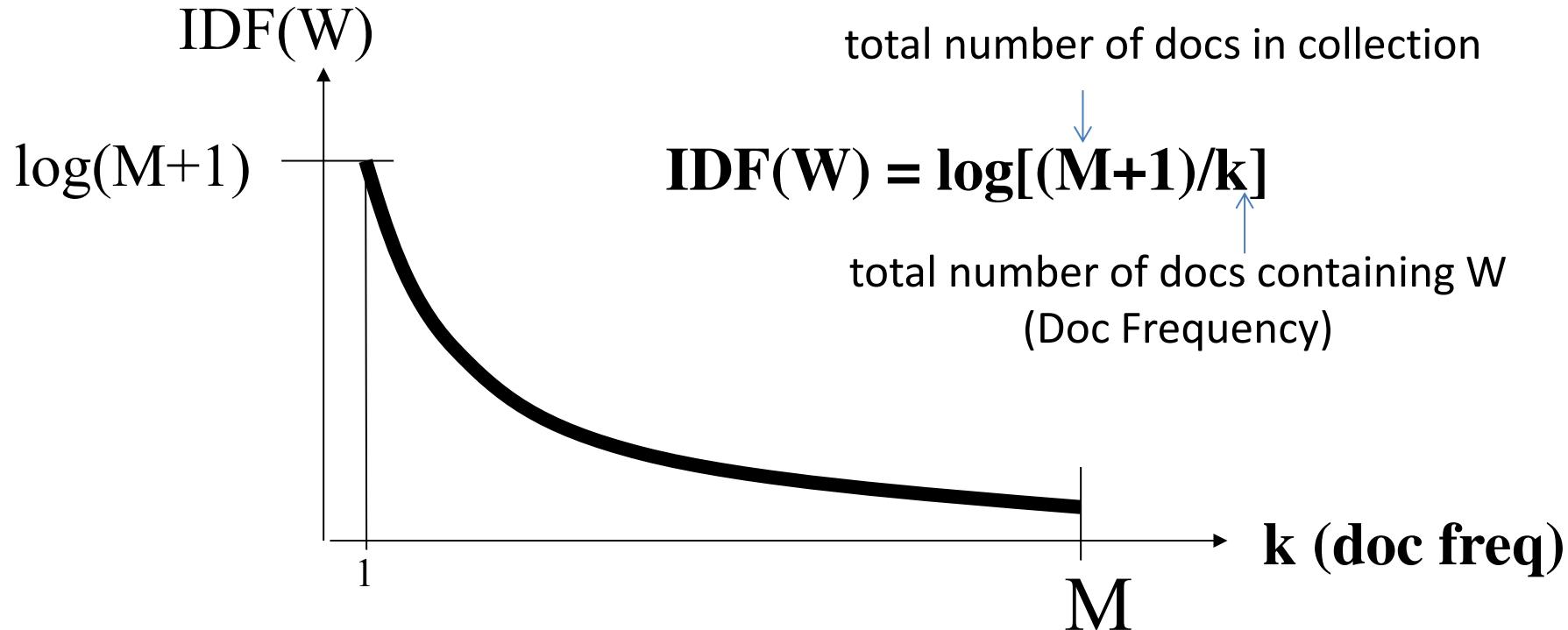
TF Transformation: BM25 Transformation

Term Frequency Weight

$$y = \text{TF}(w, d)$$



IDF Weighting: Penalizing Popular Terms



Adapting BM25 Retrieval Model for Paradigmatic Relation Mining

$$d1 = (x_1, \dots, x_N) \quad BM25(w_i, d1) = \frac{(k+1)c(w_i, d1)}{c(w_i, d1) + k(1 - b + b * |d1| / avdl)}$$

$$x_i = \frac{BM25(w_i, d1)}{\sum_{j=1}^N BM25(w_j, d1)}$$

$$b \in [0, 1]$$
$$k \in [0, +\infty)$$

$d2 = (y_1, \dots, y_N)$ y_i is defined similarly

$$Sim(d1, d2) = \sum_{i=1}^N IDF(w_i) x_i y_i$$

BM25 can also Discover Syntagmatic Relations

$d1 = (x_1, \dots, x_N)$

$$BM25(w_i, d1) = \frac{(k+1)c(w_i, d1)}{c(w_i, d1) + k(1 - b + b * |d1| / avdl)}$$

$$x_i = \frac{BM25(w_i, d1)}{\sum_{j=1}^N BM25(w_j, d1)}$$

$$b \in [0, 1]$$

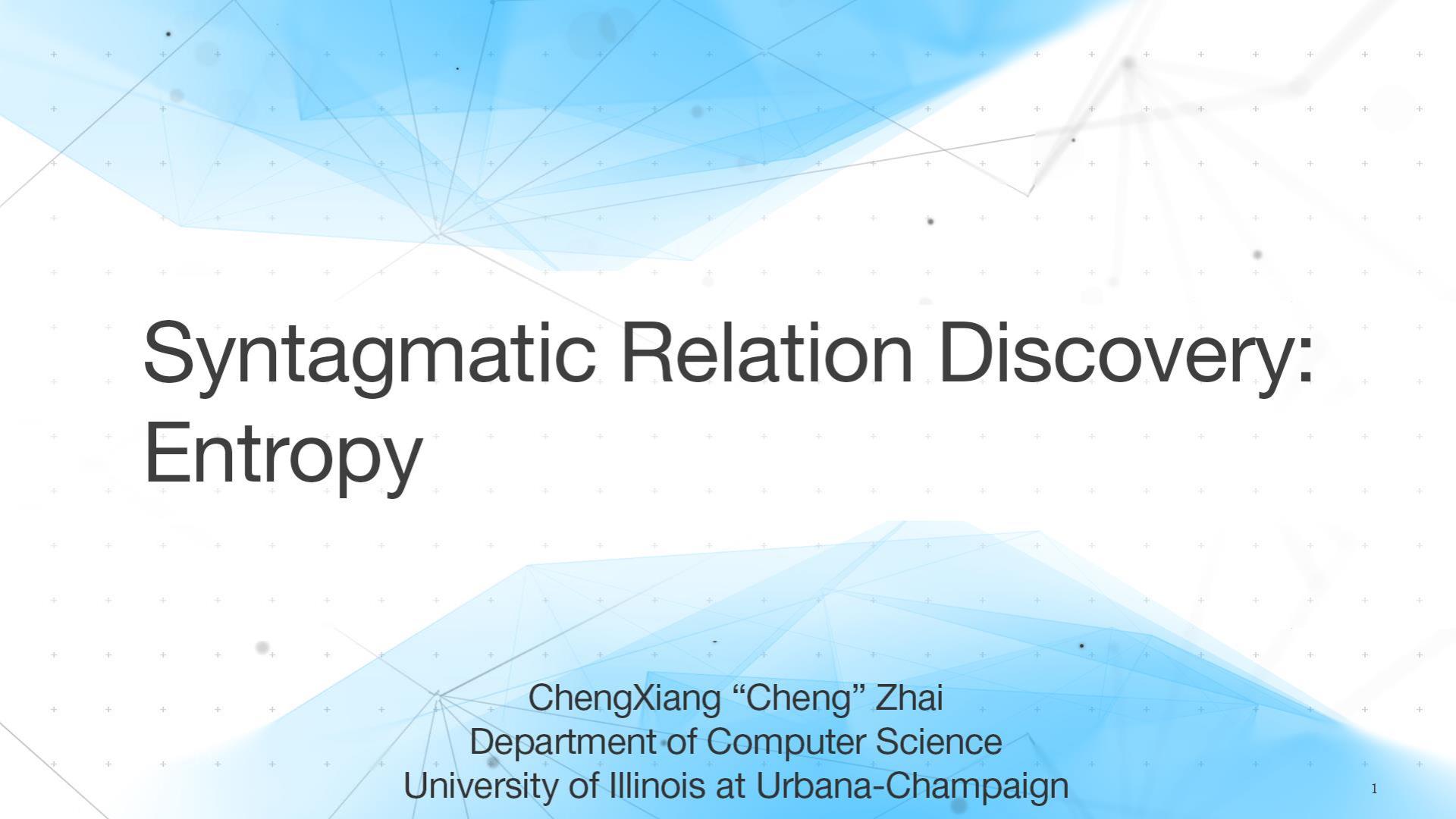
$$k \in [0, +\infty)$$

IDF-weighted $d1 = (x_1 * IDF(w_1), \dots, x_N * IDF(w_N))$

The highly weighted terms in the context vector of word w are likely syntagmatically related to w.

Summary

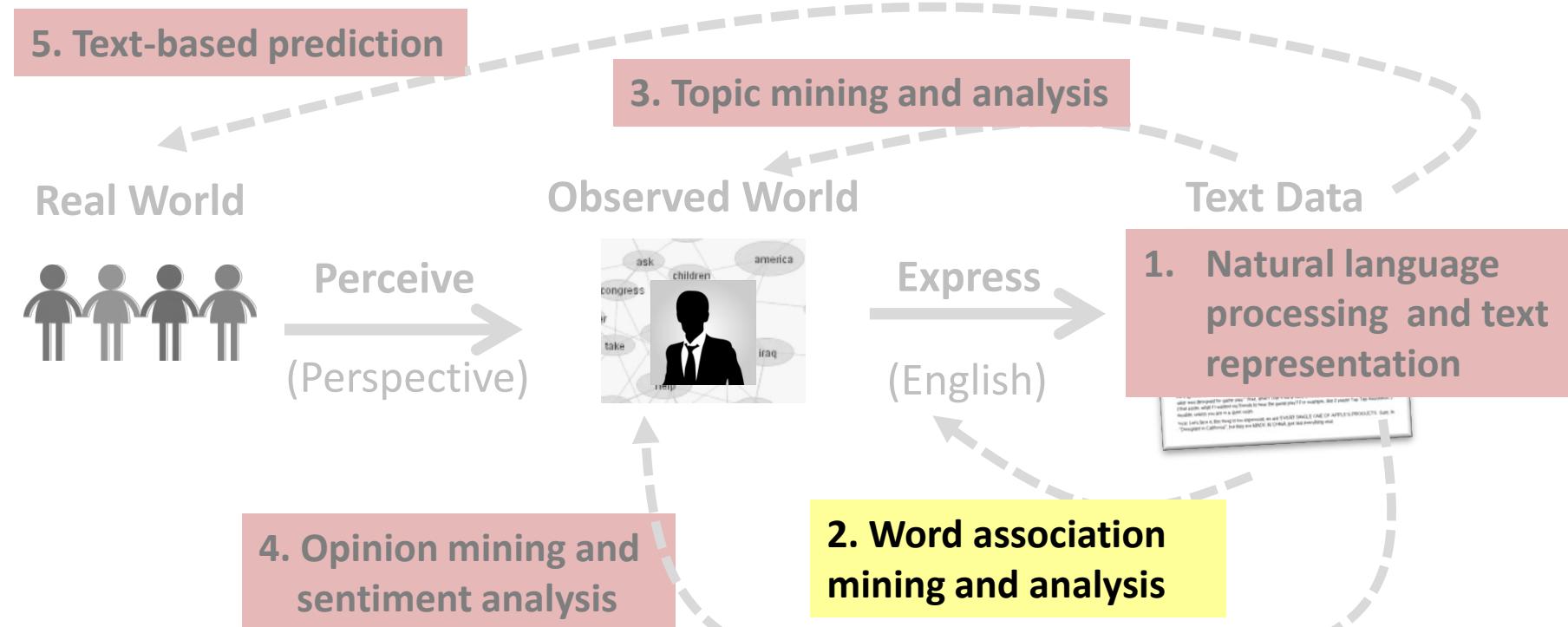
- Main idea for discovering paradigmatic relations:
 - Collecting the context of a candidate word to form a pseudo document (bag of words)
 - Computing similarity of the corresponding context documents of two candidate words
 - Highly similar word pairs can be assumed to have paradigmatic relations
- Many different ways to implement this general idea
- Text retrieval models can be easily adapted for computing similarity of two context documents
 - BM25 + IDF weighting represents the state of the art
 - Syntagmatic relations can also be discovered as a “by product”



Syntagmatic Relation Discovery: Entropy

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Syntagmatic Relation Discovery: Entropy



Syntagmatic Relation = Correlated Occurrences

Whenever “**eats**” occurs, what **other words** also tend to occur?

My cat eats fish on Saturday
His cat eats turkey on Tuesday
My dog eats meat on Sunday
His dog eats turkey on Tuesday
...

My _____ eats _____ on Saturday
His _____ eats _____ on Tuesday
My _____ eats _____ on Sunday
His _____ eats _____ on Tuesday
...

What words tend to occur
to the **left** of “**eats**”?

What words
are to the
right?

Word Prediction: Intuition

Prediction Question: Is word **W** present (or absent) in this segment?

Text Segment (any unit, e.g., sentence, paragraph, document)



Are some words easier to predict than others?

- 1) W = “meat”
- 2) W=“the”
- 3) W=“unicorn”

Word Prediction: Formal Definition

Binary Random Variable : $X_w = \begin{cases} 1 & w \text{ is present} \\ 0 & w \text{ is absent} \end{cases}$

$$p(X_w = 1) + p(X_w = 0) = 1$$

The more random X_w is, the more difficult the prediction would be.

How does one quantitatively measure the “randomness” of a random variable like X_w ?

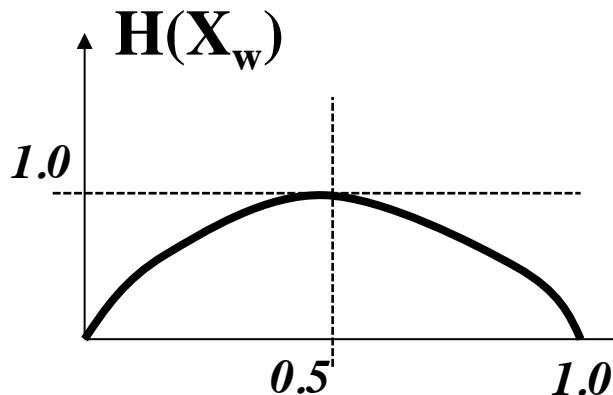
Entropy $H(X)$ Measures Randomness of X

$$H(X_w) = \sum_{v \in \{0,1\}} -p(X_w = v) \log_2 p(X_w = v)$$

$$X_w = \begin{cases} 1 & w \text{ is present} \\ 0 & w \text{ is absent} \end{cases}$$

$$= -p(X_w = 0) \log_2 p(X_w = 0) - p(X_w = 1) \log_2 p(X_w = 1)$$

Define $0 \log_2 0 = 0$



For what X_w , does $H(X_w)$ reach maximum/minimum?

E.g., $P(X_w=1)=1?$ $P(X_w=1)=0.5?$

or equivalently $P(X_w=0)$ (Why?)

Entropy $H(X)$: Coin Tossing

$$H(X_{\text{coin}}) = -p(X_{\text{coin}} = 0) \log_2 p(X_{\text{coin}} = 0) - p(X_{\text{coin}} = 1) \log_2 p(X_{\text{coin}} = 1)$$

X_{coin} : tossing a coin

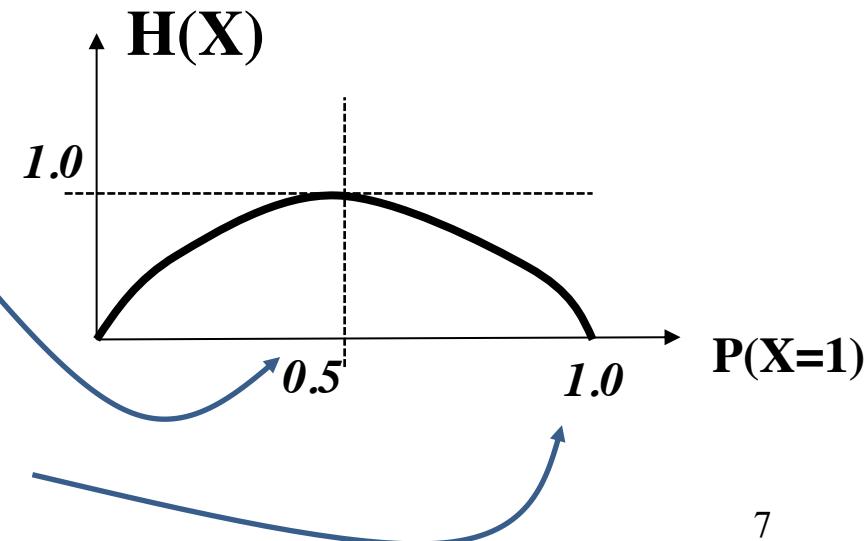
$$X_{\text{coin}} = \begin{cases} 1 & \text{Head} \\ 0 & \text{Tail} \end{cases}$$

Fair coin: $p(X=1)=p(X=0)=1/2$

$$H(X) = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$$

Completely biased: $p(X=1)=1$

$$H(X) = -0 * \log_2 0 - 1 * \log_2 1 = 0$$



Entropy for Word Prediction

Is word **W** present (or absent) in this segment?



- 1) $W = \text{"meat"}$
- 2) $W = \text{"the"}$
- 3) $W = \text{"unicorn"}$

Which is **high/low**? $H(X_{\text{meat}})$, $H(X_{\text{the}})$, or $H(X_{\text{unicorn}})$?

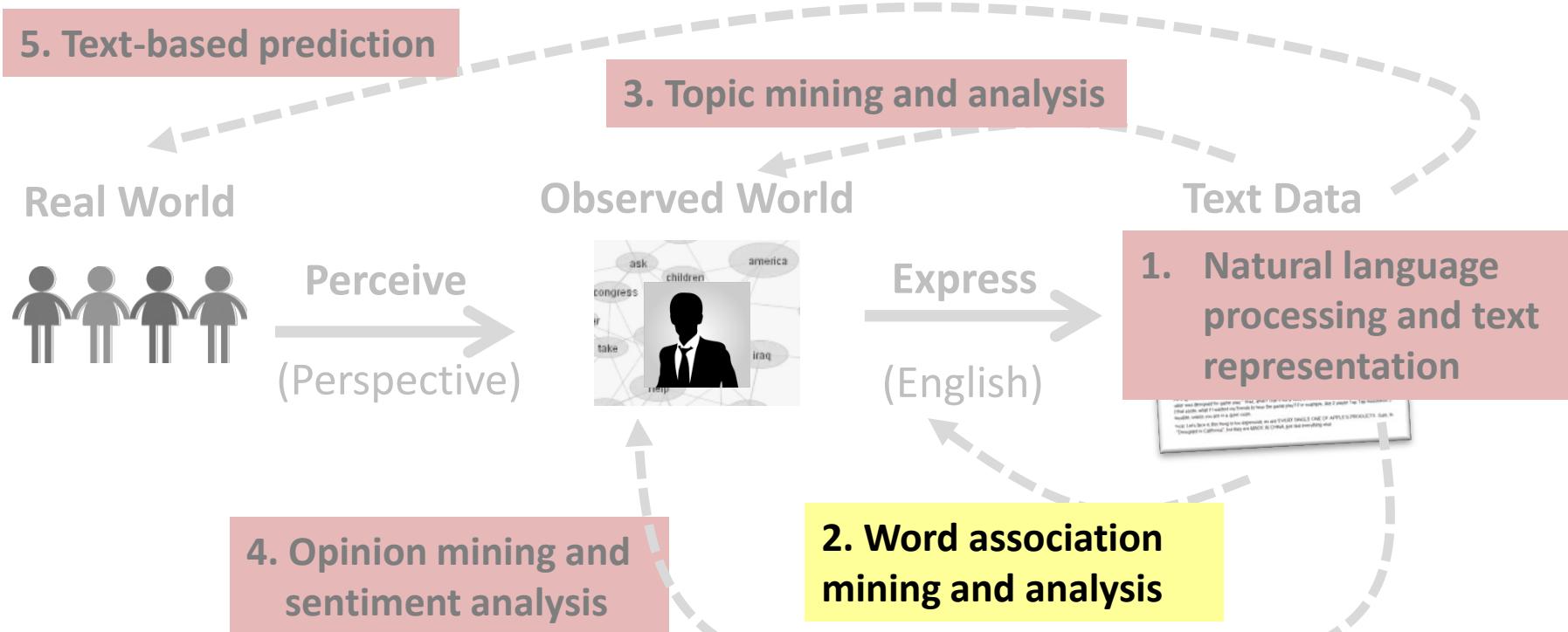
$H(X_{\text{the}}) \approx 0 \rightarrow \text{no uncertainty since } p(X_{\text{the}}=1) \approx 1$

High entropy words are harder to predict!

Syntagmatic Relation Discovery: Conditional Entropy

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Syntagmatic Relation Discovery: Conditional Entropy



What If We Know More About a Text Segment?

Prediction question: Is “**meat**” present (or absent) in this segment?



Does presence of “**eats**” help predict the presence of “**meat**”?

Does it **reduce** the uncertainty about “meat”, i.e., $H(X_{\text{meat}})$?

What if we know of the absence of “eats”? Does it also help?

Conditional Entropy

Know nothing about the segment

$$p(X_{meat} = 1) \quad \xrightarrow{\text{-----}} \quad p(X_{meat} = 1 | X_{eats} = 1)$$

$$p(X_{meat} = 0) \quad \xrightarrow{\text{-----}} \quad p(X_{meat} = 0 | X_{eats} = 1)$$

Know “eats” is present ($X_{eats} = 1$)

$$H(X_{meat}) = -p(X_{meat} = 0) \log_2 p(X_{meat} = 0) - p(X_{meat} = 1) \log_2 p(X_{meat} = 1)$$



$$H(X_{meat} | X_{eats} = 1) = -p(X_{meat} = 0 | X_{eats} = 1) \log_2 p(X_{meat} = 0 | X_{eats} = 1) \\ - p(X_{meat} = 1 | X_{eats} = 1) \log_2 p(X_{meat} = 1 | X_{eats} = 1)$$

$H(X_{meat} | X_{eats} = 0)$ can be defined similarly

Conditional Entropy: Complete Definition

$$\begin{aligned} H(X_{meat} | X_{eats}) &= \sum_{u \in \{0,1\}} [p(X_{eats} = u) H(X_{meat} | X_{eats} = u)] \\ &= \sum_{u \in \{0,1\}} [p(X_{eats} = u) \sum_{v \in \{0,1\}} [-p(X_{meat} = v | X_{eats} = u) \log_2 p(X_{meat} = v | X_{eats} = u)]] \end{aligned}$$

In general, for any discrete random variables X and Y , we have $H(X) \geq H(X|Y)$

What's the **minimum** possible value of $H(X|Y)$?

Conditional Entropy to Capture Syntagmatic Relation

$$H(X_{\text{meat}} | X_{\text{eats}}) = \sum_{u \in \{0,1\}} [p(X_{\text{eats}} = u) H(X_{\text{meat}} | X_{\text{eats}} = u)]$$

$$H(X_{\text{meat}} | X_{\text{meat}}) = ?$$

Which is smaller? $H(X_{\text{meat}} | X_{\text{the}})$ or $H(X_{\text{meat}} | X_{\text{eats}})$?

For which word w, does $H(X_{\text{meat}} | X_w)$ reach its minimum (i.e., 0)?

For which word w, does $H(X_{\text{meat}} | X_w)$ reach its maximum, $H(X_{\text{meat}})$?

Conditional Entropy for Mining Syntagmatic Relations

- For each word W_1
 - For every other word W_2 , compute conditional entropy $H(X_{W_1} | X_{W_2})$
 - Sort all the candidate words in ascending order of $H(X_{W_1} | X_{W_2})$
 - Take the top-ranked candidate words as words that have potential syntagmatic relations with W_1
 - Need to use a threshold for each W_1
- However, while $H(X_{W_1} | X_{W_2})$ and $H(X_{W_1} | X_{W_3})$ are comparable, $H(X_{W_1} | X_{W_2})$ and $H(X_{W_3} | X_{W_2})$ aren't!

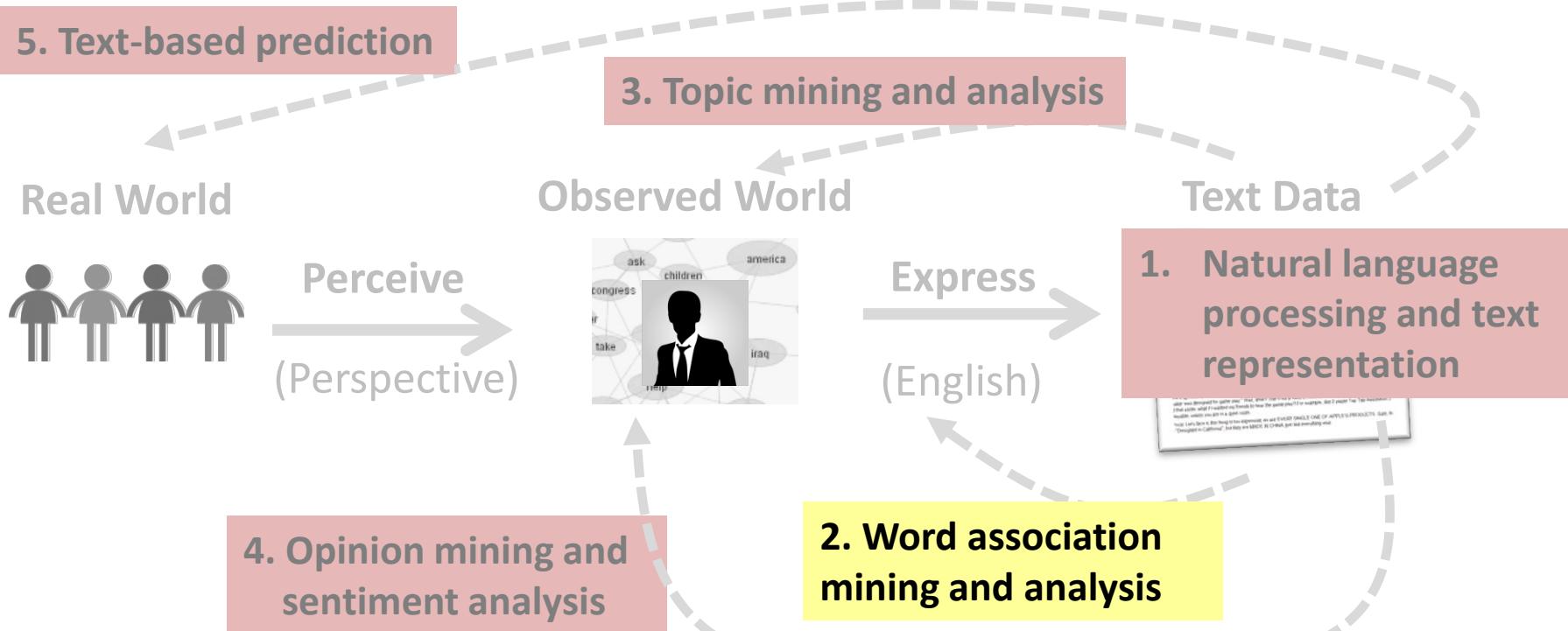
How can we mine the **strongest** K syntagmatic relations from a collection?



Syntagmatic Relation Discovery: Mutual Information

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Syntagmatic Relation Discovery: Mutual Information



Mutual Information $I(X;Y)$: Measuring Entropy Reduction

How much reduction in the entropy of X can we obtain by knowing Y?

Mutual Information: $I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$

Properties:

- Non-negative: $I(X;Y) \geq 0$
- Symmetric: $I(X;Y) = I(Y;X)$
- $I(X;Y) = 0$ iff X & Y are independent

When we fix X to rank different Ys, $I(X;Y)$ and $H(X|Y)$ give the same order but $I(X;Y)$ allows us to compare different (X,Y) pairs.

Mutual Information $I(X;Y)$ for Syntagmatic Relation Mining

Mutual Information: $I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$

Whenever “**eats**” occurs, what **other words** also tend to occur?

Which **words** have high mutual information with “**eats**”?

$$I(X_{\text{eats}}; X_{\text{meats}}) = I(X_{\text{meats}}; X_{\text{eats}}) > I(X_{\text{eats}}; X_{\text{the}}) = I(X_{\text{the}}; X_{\text{eats}})$$

$$I(X_{\text{eats}}; X_{\text{eats}}) = H(X_{\text{eats}}) \geq I(X_{\text{eats}}; X_w)$$

Rewriting Mutual Information (MI) Using KL-divergence

The observed joint distribution of X_{w1} and X_{w2}



$$I(X_{w1}; X_{w2}) = \sum_{u \in \{0,1\}} \sum_{v \in \{0,1\}} p(X_{w1} = u, X_{w2} = v) \log_2 \frac{p(X_{w1} = u, X_{w2} = v)}{p(X_{w1} = u)p(X_{w2} = v)}$$



The expected joint distribution of X_{w1} and X_{w2}
if X_{w1} and X_{w2} were independent

MI measures the divergence of the actual joint distribution from the expected distribution under the independence assumption. The larger the divergence is, the higher the MI would be.

Probabilities Involved in Mutual Information

$$I(X_{w1}; X_{w2}) = \sum_{u \in \{0,1\}} \sum_{v \in \{0,1\}} p(X_{w1} = u, X_{w2} = v) \log_2 \frac{p(X_{w1} = u, X_{w2} = v)}{p(X_{w1} = u)p(X_{w2} = v)}$$

Presence & absence of w1: $p(X_{w1}=1) + p(X_{w1}=0) = 1$

Presence & absence of w2: $p(X_{w2}=1) + p(X_{w2}=0) = 1$

Co-occurrences of w1 and w2:

$$\underline{p(X_{w1}=1, X_{w2}=1)} + \underline{p(X_{w1}=1, X_{w2}=0)} + \underline{p(X_{w1}=0, X_{w2}=1)} + \underline{p(X_{w1}=0, X_{w2}=0)} = 1$$



Both w1 & w2 occur



Only w1 occurs



Only w2 occurs



None of them occurs

Relations Between Different Probabilities

Presence & absence of w1: $p(X_{W1}=1) + p(X_{W1}=0) = 1$

Presence & absence of w2: $p(X_{W2}=1) + p(X_{W2}=0) = 1$

Co-occurrences of w1 and w2:

$$p(X_{W1}=1, X_{W2}=1) + p(X_{W1}=1, X_{W2}=0) + p(X_{W1}=0, X_{W2}=1) + p(X_{W1}=0, X_{W2}=0) = 1$$

Constraints:

$$p(X_{W1}=1, X_{W2}=1) + p(X_{W1}=1, X_{W2}=0) = p(X_{W1}=1)$$

$$p(X_{W1}=0, X_{W2}=1) + p(X_{W1}=0, X_{W2}=0) = p(X_{W1}=0)$$

$$p(X_{W1}=1, X_{W2}=1) + p(X_{W1}=0, X_{W2}=1) = p(X_{W2}=1)$$

$$p(X_{W1}=1, X_{W2}=0) + p(X_{W1}=0, X_{W2}=0) = p(X_{W2}=0)$$

Computation of Mutual Information

Presence & absence of w1:

$$p(X_{W1}=1) + p(X_{W1}=0) = 1$$

Presence & absence of w2:

$$p(X_{W2}=1) + p(X_{W2}=0) = 1$$

Co-occurrences of w1 and w2:

$$p(X_{W1}=1, X_{W2}=1) + p(X_{W1}=1, X_{W2}=0) + p(X_{W1}=0, X_{W2}=1) + p(X_{W1}=0, X_{W2}=0) = 1$$

$$p(X_{W1}=1, X_{W2}=1) + p(X_{W1}=1, X_{W2}=0) = p(X_{W1}=1)$$

$$p(X_{W1}=0, X_{W2}=1) + p(X_{W1}=0, X_{W2}=0) = p(X_{W1}=0)$$

$$p(X_{W1}=1, X_{W2}=1) + p(X_{W1}=0, X_{W2}=1) = p(X_{W2}=1)$$

$$p(X_{W1}=1, X_{W2}=0) + p(X_{W1}=0, X_{W2}=0) = p(X_{W2}=0)$$

We only need to know $p(X_{W1}=1)$, $p(X_{W2}=1)$, and $p(X_{W1}=1, X_{W2}=1)$.

Estimation of Probabilities (Depending on the Data)

$$p(X_{w1} = 1) = \frac{\text{count}(w1)}{N}$$

$$p(X_{w2} = 1) = \frac{\text{count}(w2)}{N}$$

$$p(X_{w1} = 1, X_{w2} = 1) = \frac{\text{count}(w1, w2)}{N}$$

	W1	W2	
Segment_1	1	0	Only W1 occurred
Segment_2	1	1	Both occurred
Segment_3	1	1	Both occurred
Segment_4	0	0	Neither occurred
...			
Segment_N	0	1	Only W2 occurred

Count(w1) = total number segments that contain W1

Count(w2) = total number segments that contain W2

Count(w1, w2) = total number segments that contain both W1 and W2

Smoothing: Accommodating Zero Counts

$$p(X_{w1} = 1) = \frac{\text{count}(w1) + 0.5}{N + 1}$$

$$p(X_{w2} = 1) = \frac{\text{count}(w2) + 0.5}{N + 1}$$

$$p(X_{w1} = 1, X_{w2} = 1) = \frac{\text{count}(w1, w2) + 0.25}{N + 1}$$

Smoothing: Add pseudo data so that
no event has zero counts
(pretend we observed extra data)

	W1	W2
¼ PseudoSeg_1	0	0
¼ PseudoSeg_2	1	0
¼ PseudoSeg_3	0	1
¼ PseudoSeg_4	1	1

Segment_1	1	0
...		
Segment_N	0	1

Actually observed data

Summary of Syntagmatic Relation Discovery

- Syntagmatic relation can be discovered by measuring correlations between occurrences of two words.
- Three concepts from Information Theory:
 - Entropy $H(X)$: measures the uncertainty of a random variable X
 - Conditional entropy $H(X|Y)$: entropy of X given we know Y
 - Mutual information $I(X;Y)$: entropy reduction of X (or Y) due to knowing Y (or X)
- Mutual information provides a principled way for discovering syntagmatic relations.

Summary of Word Association Mining

- Two basic associations: paradigmatic and syntagmatic
 - Generally applicable to any items in any language (e.g., phrases or entities as units)
- Pure statistical approaches are available for discovering both (can be combined to perform joint analysis).
 - Generally applicable to any text with no human effort
 - Different ways to define “context” and “segment” lead to interesting variations of applications
- Discovered associations can support many other applications.

Additional Reading

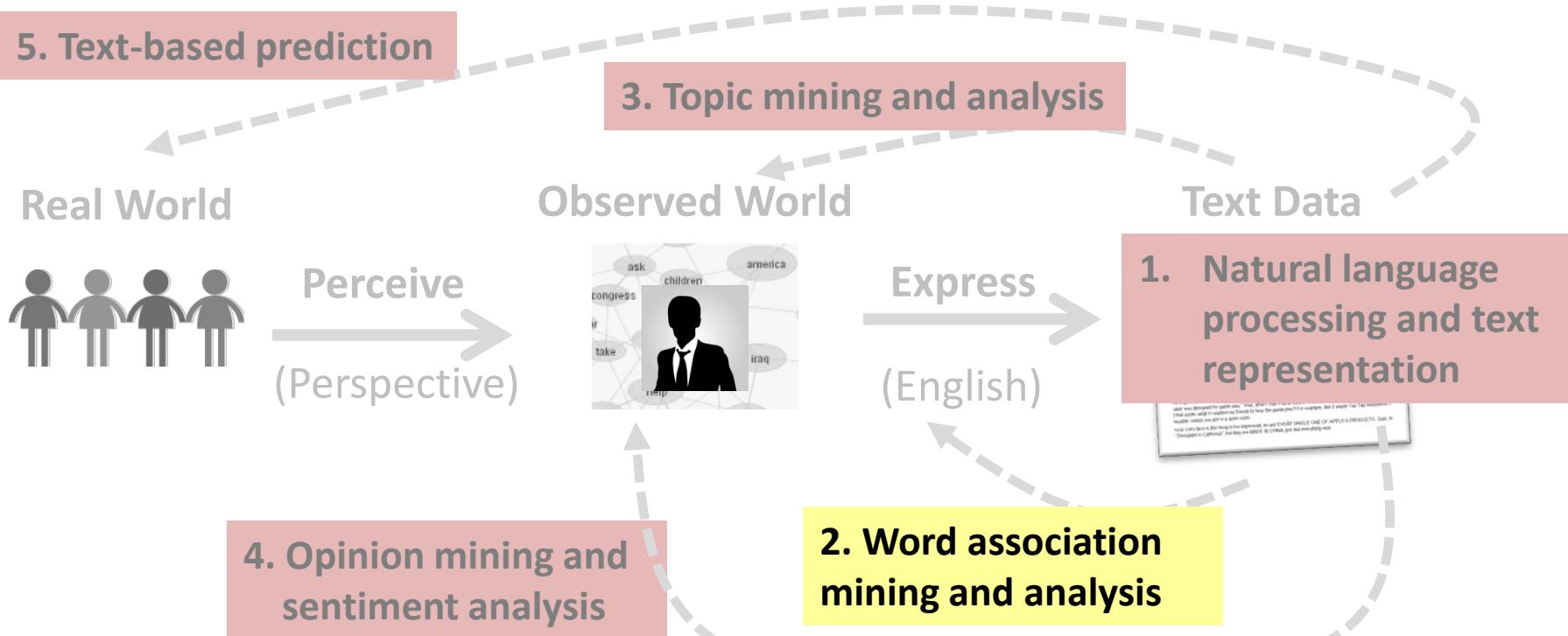
- Chris Manning and Hinrich Schütze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA: May 1999. (Chapter 5 on collocations)
- Chengxiang Zhai, Exploiting context to identify lexical atoms: A statistical view of linguistic context. Proceedings of the International and Interdisciplinary Conference on Modelling and Using Context (CONTEXT-97), Rio de Janeiro, Brzil, Feb. 4-6, 1997. pp. 119-129.
- Shan Jiang and ChengXiang Zhai, Random walks on adjacency graphs for mining lexical relations from big text data. Proceedings of IEEE BigData Conference 2014, pp. 549-554.



Syntagmatic Relation Discovery: Mutual Information

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Syntagmatic Relation Discovery: Mutual Information



Mutual Information $I(X;Y)$: Measuring Entropy Reduction

How much reduction in the entropy of X can we obtain by knowing Y?

Mutual Information: $I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$

Properties:

- Non-negative: $I(X;Y) \geq 0$
- Symmetric: $I(X;Y) = I(Y;X)$
- $I(X;Y) = 0$ iff X & Y are independent

When we fix X to rank different Ys, $I(X;Y)$ and $H(X|Y)$ give the same order but $I(X;Y)$ allows us to compare different (X,Y) pairs.

Mutual Information $I(X;Y)$ for Syntagmatic Relation Mining

Mutual Information: $I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$

Whenever “**eats**” occurs, what **other words** also tend to occur?

Which **words** have high mutual information with “**eats**”?

$$I(X_{\text{eats}}; X_{\text{meats}}) = I(X_{\text{meats}}; X_{\text{eats}}) > I(X_{\text{eats}}; X_{\text{the}}) = I(X_{\text{the}}; X_{\text{eats}})$$

$$I(X_{\text{eats}}; X_{\text{eats}}) = H(X_{\text{eats}}) \geq I(X_{\text{eats}}; X_w)$$

Rewriting Mutual Information (MI) Using KL-divergence

The observed joint distribution of X_{w1} and X_{w2}



$$I(X_{w1}; X_{w2}) = \sum_{u \in \{0,1\}} \sum_{v \in \{0,1\}} p(X_{w1} = u, X_{w2} = v) \log_2 \frac{p(X_{w1} = u, X_{w2} = v)}{p(X_{w1} = u)p(X_{w2} = v)}$$



The expected joint distribution of X_{w1} and X_{w2}
if X_{w1} and X_{w2} were independent

MI measures the divergence of the actual joint distribution from the expected distribution under the independence assumption. The larger the divergence is, the higher the MI would be.

Probabilities Involved in Mutual Information

$$I(X_{w1}; X_{w2}) = \sum_{u \in \{0,1\}} \sum_{v \in \{0,1\}} p(X_{w1} = u, X_{w2} = v) \log_2 \frac{p(X_{w1} = u, X_{w2} = v)}{p(X_{w1} = u)p(X_{w2} = v)}$$

Presence & absence of w1: $p(X_{w1}=1) + p(X_{w1}=0) = 1$

Presence & absence of w2: $p(X_{w2}=1) + p(X_{w2}=0) = 1$

Co-occurrences of w1 and w2:

$$\underline{p(X_{w1}=1, X_{w2}=1)} + \underline{p(X_{w1}=1, X_{w2}=0)} + \underline{p(X_{w1}=0, X_{w2}=1)} + \underline{p(X_{w1}=0, X_{w2}=0)} = 1$$



Both w1 & w2 occur



Only w1 occurs



Only w2 occurs



None of them occurs

Relations Between Different Probabilities

Presence & absence of w1: $p(X_{W1}=1) + p(X_{W1}=0) = 1$

Presence & absence of w2: $p(X_{W2}=1) + p(X_{W2}=0) = 1$

Co-occurrences of w1 and w2:

$$p(X_{W1}=1, X_{W2}=1) + p(X_{W1}=1, X_{W2}=0) + p(X_{W1}=0, X_{W2}=1) + p(X_{W1}=0, X_{W2}=0) = 1$$

Constraints:

$$p(X_{W1}=1, X_{W2}=1) + p(X_{W1}=1, X_{W2}=0) = p(X_{W1}=1)$$

$$p(X_{W1}=0, X_{W2}=1) + p(X_{W1}=0, X_{W2}=0) = p(X_{W1}=0)$$

$$p(X_{W1}=1, X_{W2}=1) + p(X_{W1}=0, X_{W2}=1) = p(X_{W2}=1)$$

$$p(X_{W1}=1, X_{W2}=0) + p(X_{W1}=0, X_{W2}=0) = p(X_{W2}=0)$$

Computation of Mutual Information

Presence & absence of w1:

$$p(X_{W1}=1) + p(X_{W1}=0) = 1$$

Presence & absence of w2:

$$p(X_{W2}=1) + p(X_{W2}=0) = 1$$

Co-occurrences of w1 and w2:

$$p(X_{W1}=1, X_{W2}=1) + p(X_{W1}=1, X_{W2}=0) + p(X_{W1}=0, X_{W2}=1) + p(X_{W1}=0, X_{W2}=0) = 1$$

$$p(X_{W1}=1, X_{W2}=1) + p(X_{W1}=1, X_{W2}=0) = p(X_{W1}=1)$$

$$p(X_{W1}=0, X_{W2}=1) + p(X_{W1}=0, X_{W2}=0) = p(X_{W1}=0)$$

$$p(X_{W1}=1, X_{W2}=1) + p(X_{W1}=0, X_{W2}=1) = p(X_{W2}=1)$$

$$p(X_{W1}=1, X_{W2}=0) + p(X_{W1}=0, X_{W2}=0) = p(X_{W2}=0)$$

We only need to know $p(X_{W1}=1)$, $p(X_{W2}=1)$, and $p(X_{W1}=1, X_{W2}=1)$.

Estimation of Probabilities (Depending on the Data)

$$p(X_{w1} = 1) = \frac{\text{count}(w1)}{N}$$

$$p(X_{w2} = 1) = \frac{\text{count}(w2)}{N}$$

$$p(X_{w1} = 1, X_{w2} = 1) = \frac{\text{count}(w1, w2)}{N}$$

	W1	W2	
Segment_1	1	0	Only W1 occurred
Segment_2	1	1	Both occurred
Segment_3	1	1	Both occurred
Segment_4	0	0	Neither occurred
...			
Segment_N	0	1	Only W2 occurred

Count(w1) = total number segments that contain W1

Count(w2) = total number segments that contain W2

Count(w1, w2) = total number segments that contain both W1 and W2

Smoothing: Accommodating Zero Counts

$$p(X_{w1} = 1) = \frac{\text{count}(w1) + 0.5}{N + 1}$$

$$p(X_{w2} = 1) = \frac{\text{count}(w2) + 0.5}{N + 1}$$

$$p(X_{w1} = 1, X_{w2} = 1) = \frac{\text{count}(w1, w2) + 0.25}{N + 1}$$

Smoothing: Add pseudo data so that
no event has zero counts
(pretend we observed extra data)

	W1	W2
¼ PseudoSeg_1	0	0
¼ PseudoSeg_2	1	0
¼ PseudoSeg_3	0	1
¼ PseudoSeg_4	1	1

Segment_1	1	0
...		
Segment_N	0	1

Actually observed data

Summary of Syntagmatic Relation Discovery

- Syntagmatic relation can be discovered by measuring correlations between occurrences of two words.
- Three concepts from Information Theory:
 - Entropy $H(X)$: measures the uncertainty of a random variable X
 - Conditional entropy $H(X|Y)$: entropy of X given we know Y
 - Mutual information $I(X;Y)$: entropy reduction of X (or Y) due to knowing Y (or X)
- Mutual information provides a principled way for discovering syntagmatic relations.

Summary of Word Association Mining

- Two basic associations: paradigmatic and syntagmatic
 - Generally applicable to any items in any language (e.g., phrases or entities as units)
- Pure statistical approaches are available for discovering both (can be combined to perform joint analysis).
 - Generally applicable to any text with no human effort
 - Different ways to define “context” and “segment” lead to interesting variations of applications
- Discovered associations can support many other applications.

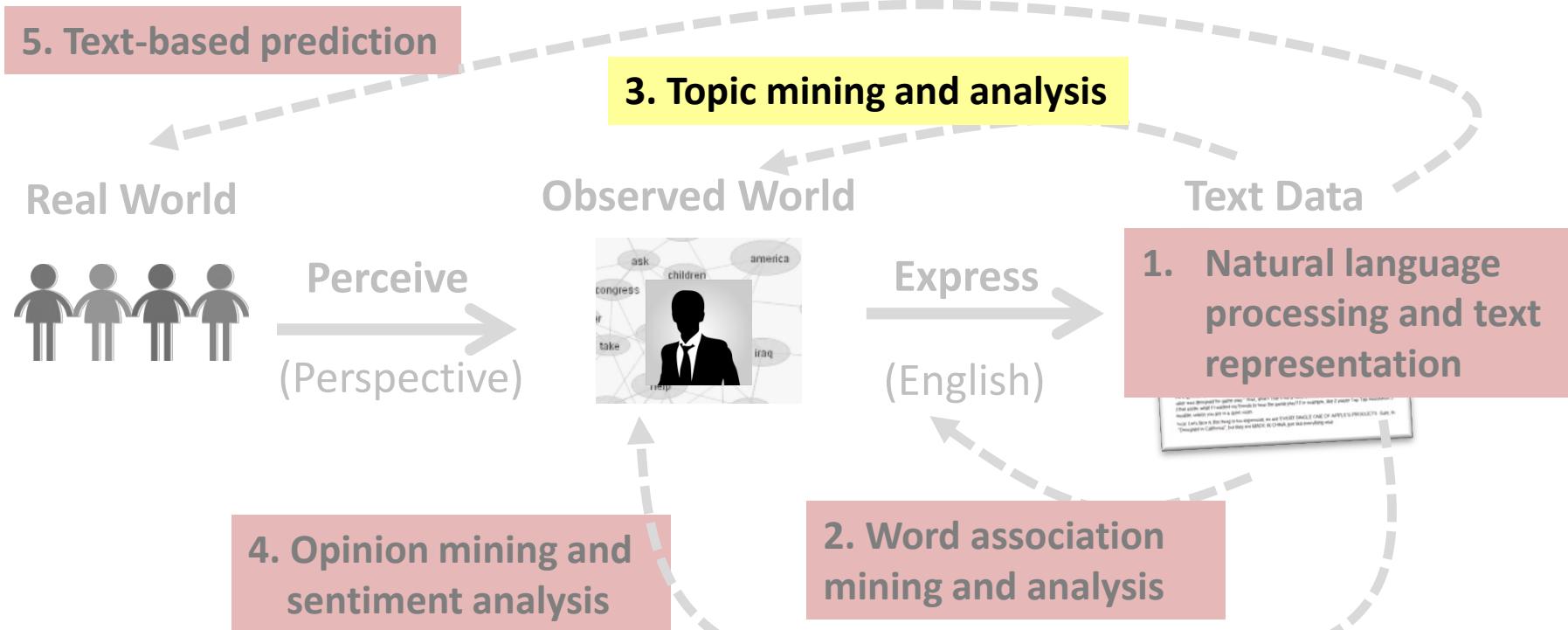
Additional Reading

- Chris Manning and Hinrich Schütze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA: May 1999. (Chapter 5 on collocations)
- Chengxiang Zhai, Exploiting context to identify lexical atoms: A statistical view of linguistic context. Proceedings of the International and Interdisciplinary Conference on Modelling and Using Context (CONTEXT-97), Rio de Janeiro, Brzil, Feb. 4-6, 1997. pp. 119-129.
- Shan Jiang and ChengXiang Zhai, Random walks on adjacency graphs for mining lexical relations from big text data. Proceedings of IEEE BigData Conference 2014, pp. 549-554.

Topic Mining and Analysis: Motivation and Task Definition

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

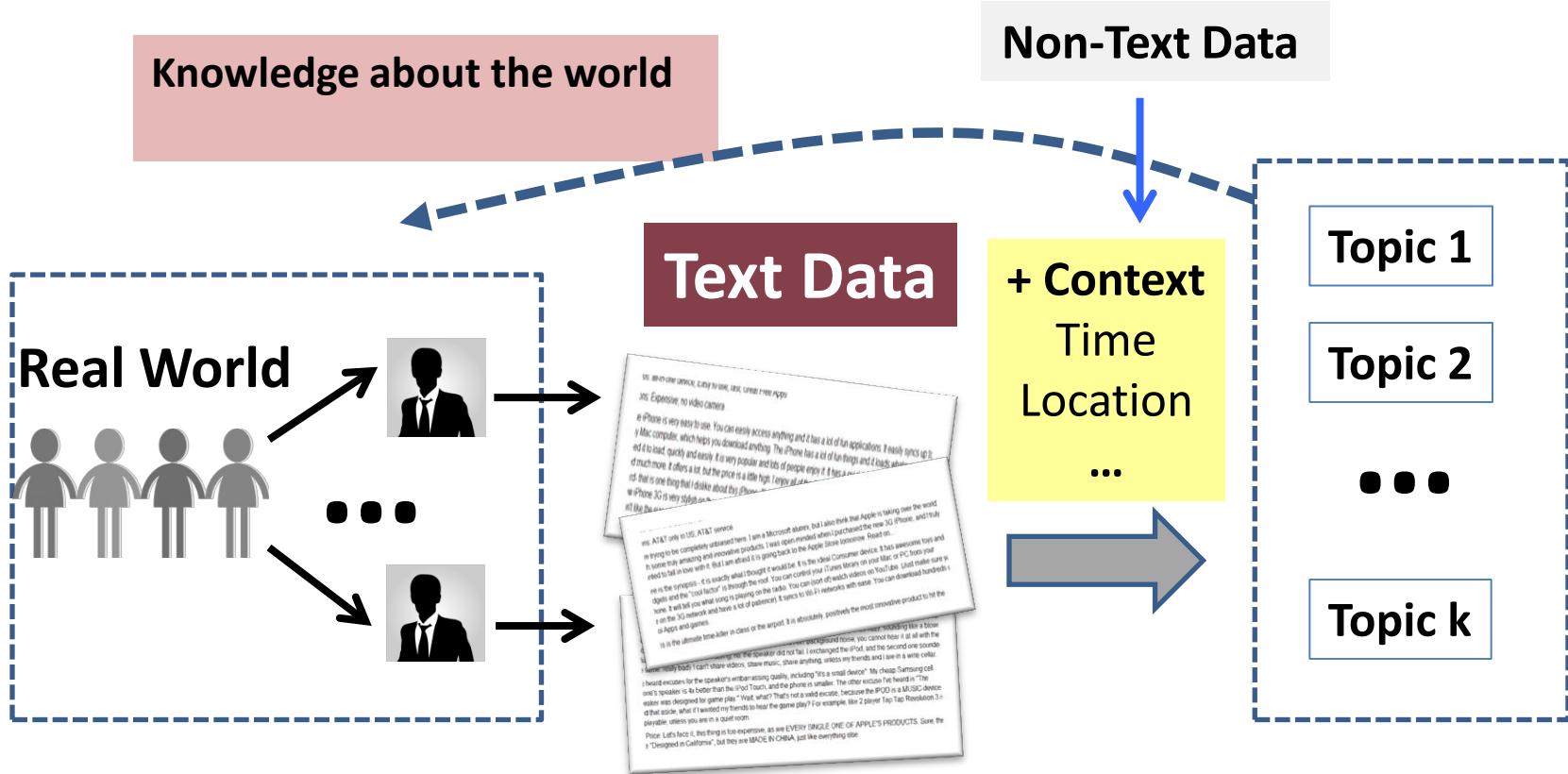
Topic Mining and Analysis: Motivation and Task Definition



Topic Mining and Analysis: Motivation

- Topic \approx main idea discussed in text data
 - Theme/subject of a discussion or conversation
 - Different granularities (e.g., topic of a sentence, an article, etc.)
- Many applications require discovery of topics in text
 - What are Twitter users talking about today?
 - What are the current research topics in data mining? How are they different from those 5 years ago?
 - What do people like about the iPhone 6? What do they dislike?
 - What were the major topics debated in 2012 presidential election?

Topics As Knowledge About the World



Tasks of Topic Mining and Analysis

Task 2: Figure out which documents cover which topics



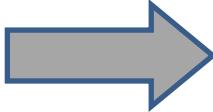
Doc 1

Doc 2

• • •

Doc N

Text Data



Topic 1

Topic 2

• • •

Topic k



Task 1: Discover k topics

Formal Definition of Topic Mining and Analysis

- Input
 - A collection of **N** text documents $C=\{d_1, \dots, d_N\}$
 - Number of topics: **k**
- Output
 - **k topics:** $\{\theta_1, \dots, \theta_k\}$
 - **Coverage of topics in each d_i :** $\{\pi_{i1}, \dots, \pi_{ik}\}$
 - $\pi_{ij} = \text{prob. of } d_i \text{ covering topic } \theta_j$

$$\sum_{j=1}^k \pi_{ij} = 1$$

How to define θ_i ?

Topic Mining and Analysis: Term as Topic

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

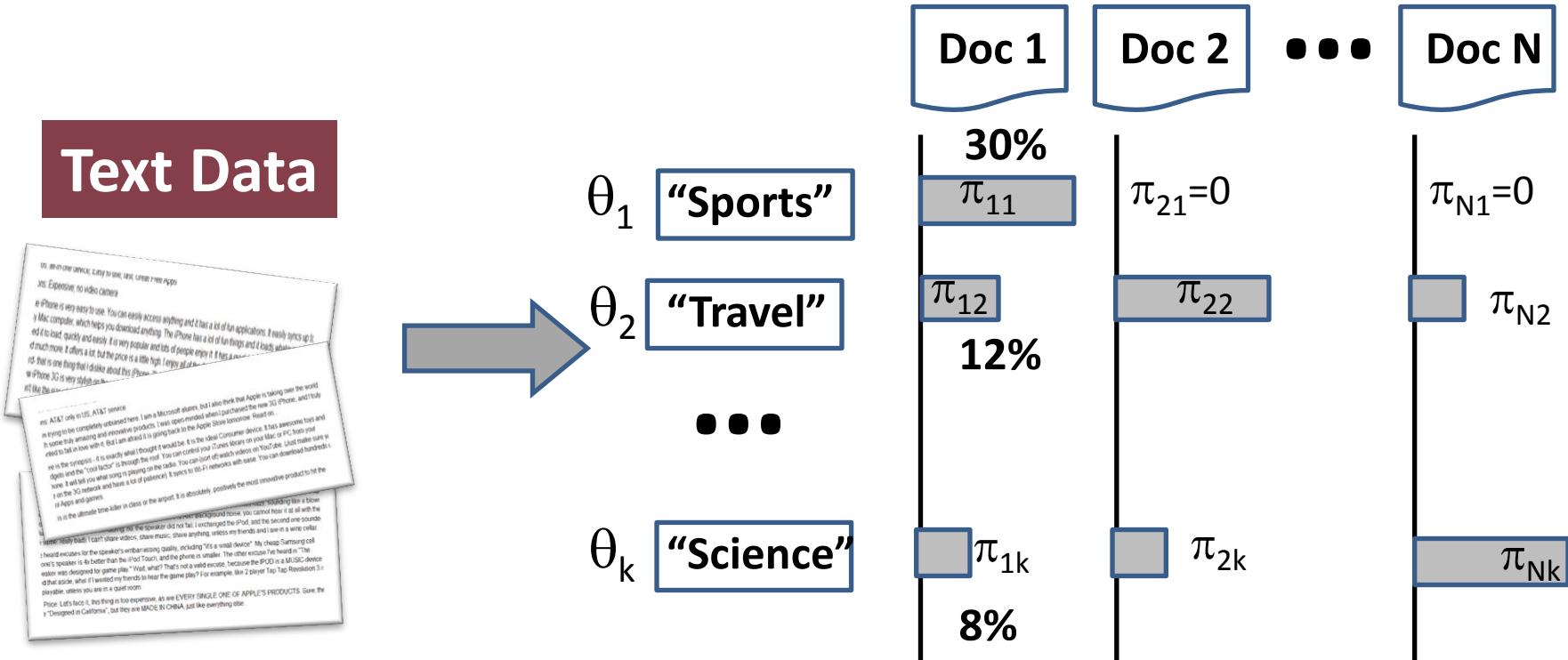
Formal Definition of Topic Mining and Analysis

- Input
 - A **collection** of **N** text documents $C=\{d_1, \dots, d_N\}$
 - **Number of topics:** **k**
- Output
 - **k topics:** $\{\theta_1, \dots, \theta_k\}$
 - **Coverage of topics in each d_i :** $\{\pi_{i1}, \dots, \pi_{ik}\}$
 - $\pi_{ij} = \text{prob. of } d_i \text{ covering topic } \theta_j$

$$\sum_{j=1}^k \pi_{ij} = 1$$

How to define θ_i ?

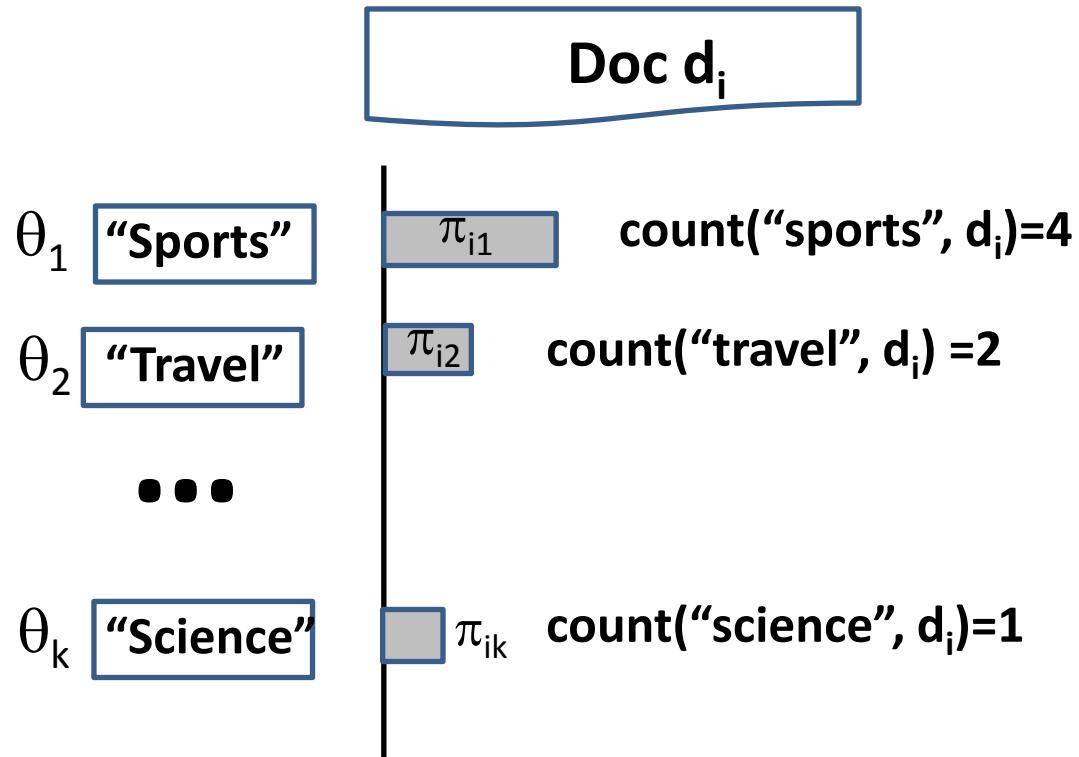
Initial Idea: Topic = Term



Mining k Topical Terms from Collection C

- Parse text in C to obtain candidate terms (e.g., term = word).
- Design a scoring function to measure how good each term is as a topic.
 - Favor a representative term (high frequency is favored)
 - Avoid words that are too frequent (e.g., “the”, “a”).
 - TF-IDF weighting from retrieval can be very useful.
 - Domain-specific heuristics are possible (e.g., favor title words, hashtags in tweets).
- Pick k terms with the highest scores but try to minimize redundancy.
 - If multiple terms are very similar or closely related, pick only one of them and ignore others.

Computing Topic Coverage: π_{ij}



$$\pi_{ij} = \frac{\text{count}(\theta_j, d_i)}{\sum_{L=1}^k \text{count}(\theta_L, d_i)}$$

How Well Does This Approach Work?

Doc d_i

Cavaliers vs. Golden State Warriors: NBA playoff finals ...
basketball game ... travel to Cleveland ... star ...

θ_1

“Sports”

$$\pi_{i1} \propto c("sports", d_i) = 0$$

θ_2

“Travel”

$$\pi_{i2} \propto c("travel", d_i) = 1 > 0$$

...

θ_k

“Science”

$$\pi_{ik} \propto c("science", d_i) = 0$$

1. Need to count
related words also!

2. “Star” can be ambiguous (e.g., star in the sky).

3. Mine complicated topics?

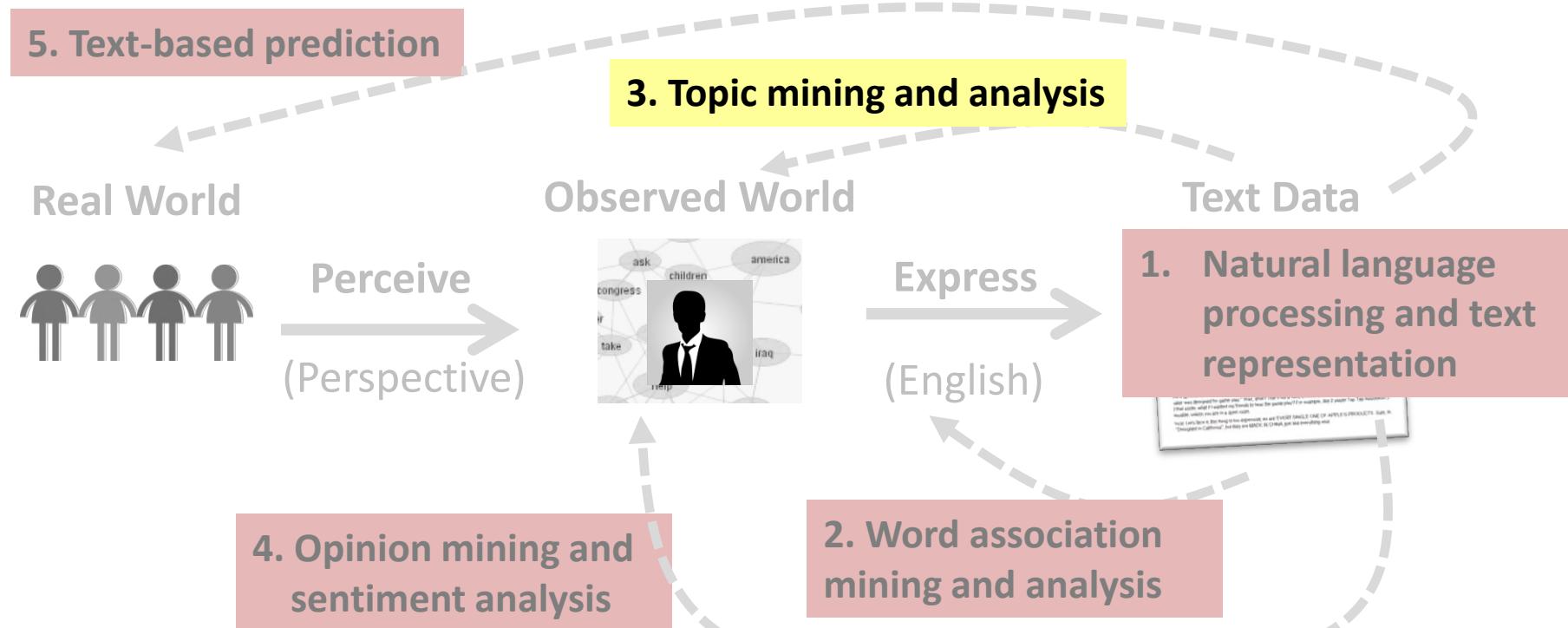
Problems with “Term as Topic”

- Lack of expressive power
 - Can only represent simple/general topics
 - Can't represent complicated topics
- Incompleteness in vocabulary coverage
 - Can't capture variations of vocabulary (e.g., related words)
- Word sense ambiguity
 - A topical term or related term can be ambiguous (e.g., basketball star vs. star in the sky)

Topic Mining and Analysis: Probabilistic Topic Models

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Topic Mining and Analysis: Probabilistic Topic Models



Problems with “Term as Topic”

- Lack of expressive power → **Topic = {Multiple Words}**
 - Can only represent simple/general topics
 - Can't represent complicated topics
- Incompleteness in vocabulary coverage + **weights on words**
 - Can't capture variations of vocabulary (e.g., related words)
- Word sense ambiguity → **Split an ambiguous word**
 - A topical term or related term can be ambiguous (e.g., basketball star vs. star in the sky)

A probabilistic topic model can do all these!

Improved Idea: Topic = Word Distribution

θ_1 “Sports”

$P(w|\theta_1)$

sports 0.02
game 0.01
basketball 0.005
football 0.004
play 0.003
star 0.003

...
nba 0.001

...
travel 0.0005

...

θ_2 “Travel”

$P(w|\theta_2)$

travel 0.05
attraction 0.03
trip 0.01
flight 0.004
hotel 0.003
island 0.003

...
culture 0.001

...
play 0.0002

...

• • •

θ_k “Science”

$P(w|\theta_k)$

science 0.04
scientist 0.03
spaceship 0.006
telescope 0.004
genomics 0.004
star 0.002

...
genetics 0.001

...
travel 0.00001

...

$$\sum_{w \in V} p(w | \theta_i) = 1$$

Vocabulary Set: $V=\{w_1, w_2, \dots\}$

Probabilistic Topic Mining and Analysis

- Input

- A collection of N text documents $C=\{d_1, \dots, d_N\}$
- Vocabulary set: $V=\{w_1, \dots, w_M\}$
- Number of topics: k

- Output

- k topics, each a word distribution: $\{ \theta_1, \dots, \theta_k \}$
- Coverage of topics in each d_i : $\{ \pi_{i1}, \dots, \pi_{ik} \}$
- π_{ij} =prob. of d_i covering topic θ_j

$$\sum_{w \in V} p(w | \theta_i) = 1$$

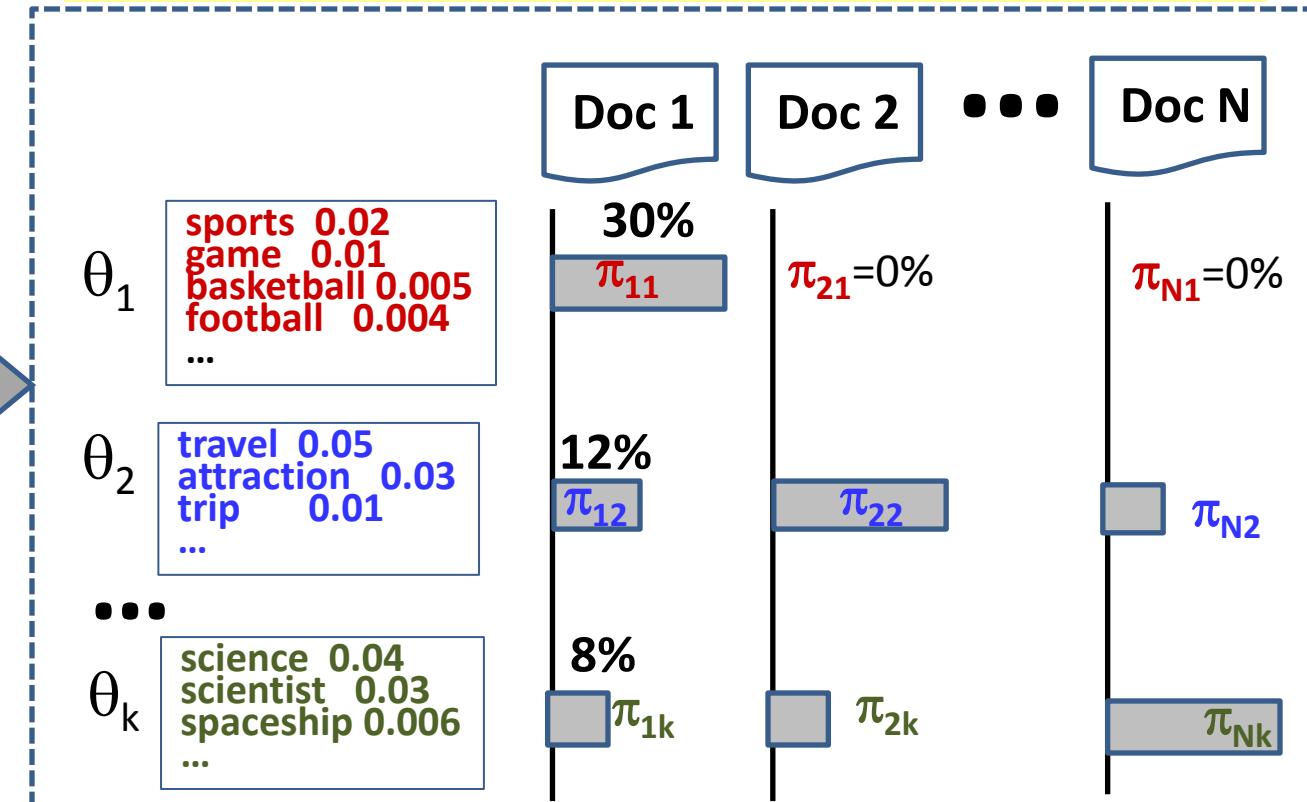
$$\sum_{j=1}^k \pi_{ij} = 1$$

The Computation Task

INPUT: C, k, V

OUTPUT: $\{ \theta_1, \dots, \theta_k \}, \{ \pi_{i1}, \dots, \pi_{ik} \}$

Text Data



Generative Model for Text Mining

Modeling of Data Generation: $P(\text{Data} | \text{Model}, \Lambda)$

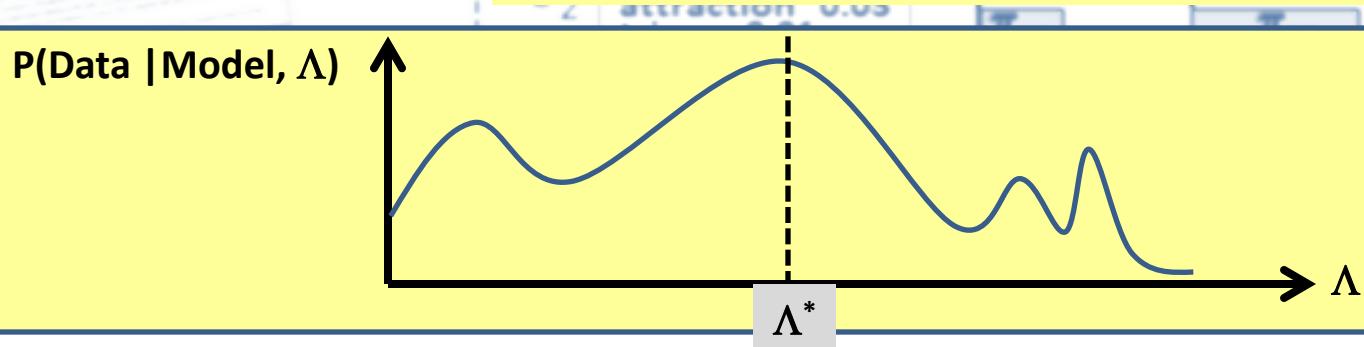
$$\Lambda = (\{\theta_1, \dots, \theta_k\}, \{\pi_{11}, \dots, \pi_{1k}\}, \dots, \{\pi_{N1}, \dots, \pi_{Nk}\})$$

Text Data

How many parameters in total?

Parameter Estimation/ Inferences

$$\Lambda^* = \operatorname{argmax}_{\Lambda} p(\text{Data} | \text{Model}, \Lambda)$$



Summary

- Topic represented as word distribution
 - Multiple words: allow for describing a complicated topic
 - Weights on words: model subtle semantic variations of a topic
- Task of topic mining and analysis
 - Input: collection C, number of topics k, vocabulary set V
 - Output: a set of topics, each a word distribution; coverage of all topics in each document

$$\Lambda = (\{ \theta_1, \dots, \theta_k \}, \{ \pi_{11}, \dots, \pi_{1k} \}, \dots, \{ \pi_{N1}, \dots, \pi_{Nk} \})$$

$$\forall j \in [1, k], \sum_{w \in V} p(w | \theta_j) = 1$$

$$\forall i \in [1, N], \sum_{j=1}^k \pi_{ij} = 1$$

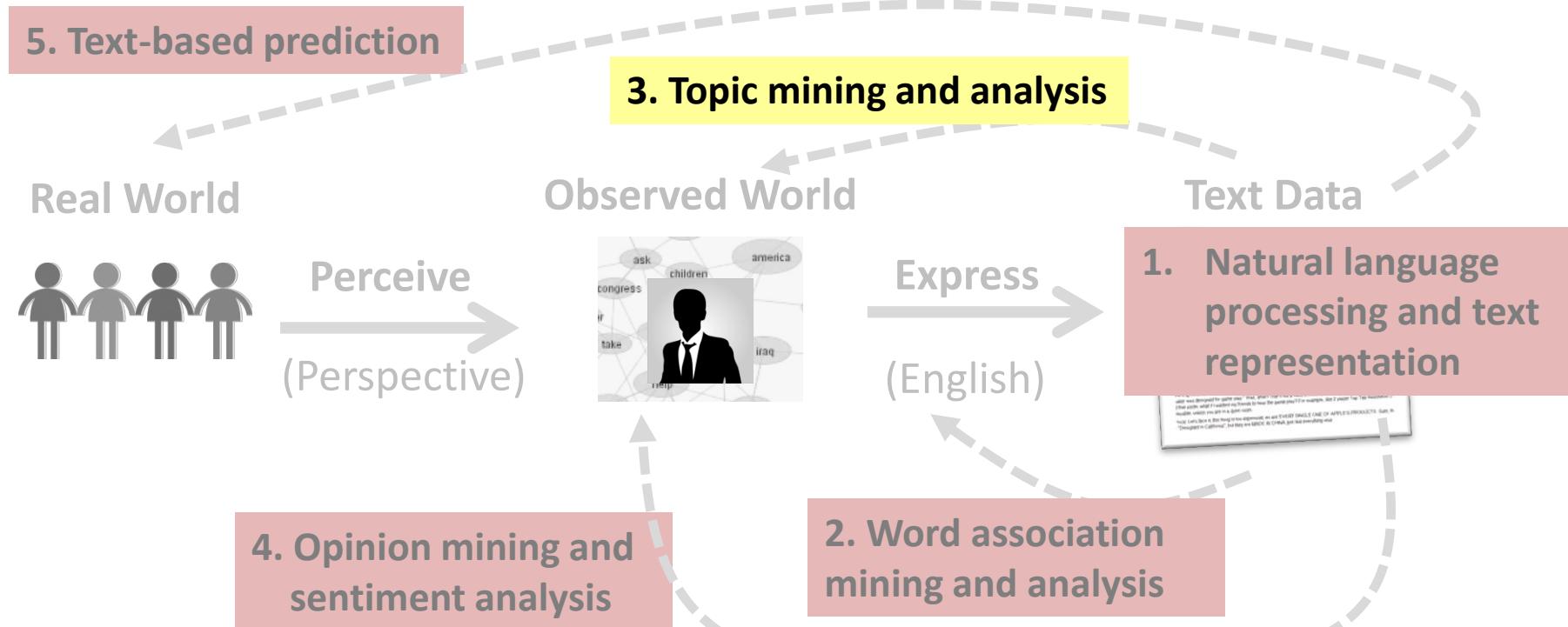
Summary (cont.)

- **Generative model** for text mining
 - Model data generation with a prob. model: $P(\text{Data} \mid \text{Model}, \Lambda)$
 - Infer the most likely parameter values Λ^* given a particular data set: $\Lambda^* = \operatorname{argmax}_{\Lambda} p(\text{Data} \mid \text{Model}, \Lambda)$
 - Take Λ^* as the “knowledge” to be mined for the text mining problem
 - Adjust the design of the model to discover different knowledge

Topic Mining and Analysis: Overview of Statistical Language Models

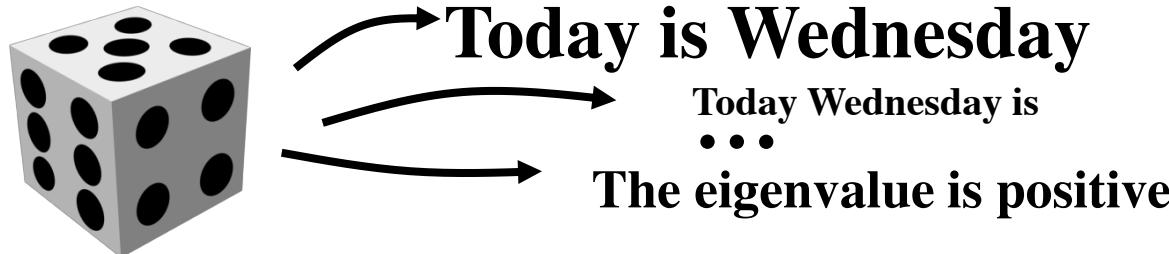
ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Probabilistic Topic Models: Overview of Statistical Language Models



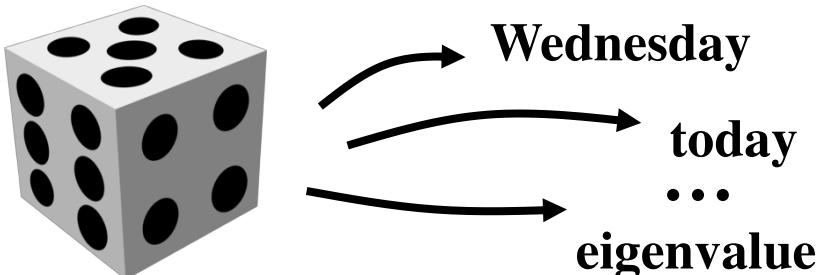
What Is a Statistical Language Model (LM)?

- A probability distribution over word sequences
 - $p(\text{"Today is Wednesday"}) \approx 0.001$
 - $p(\text{"Today Wednesday is"}) \approx 0.000000000001$
 - $p(\text{"The eigenvalue is positive"}) \approx 0.0001$
- Context-dependent!
- Can also be regarded as a probabilistic mechanism for “generating” text – thus also called a “generative” model



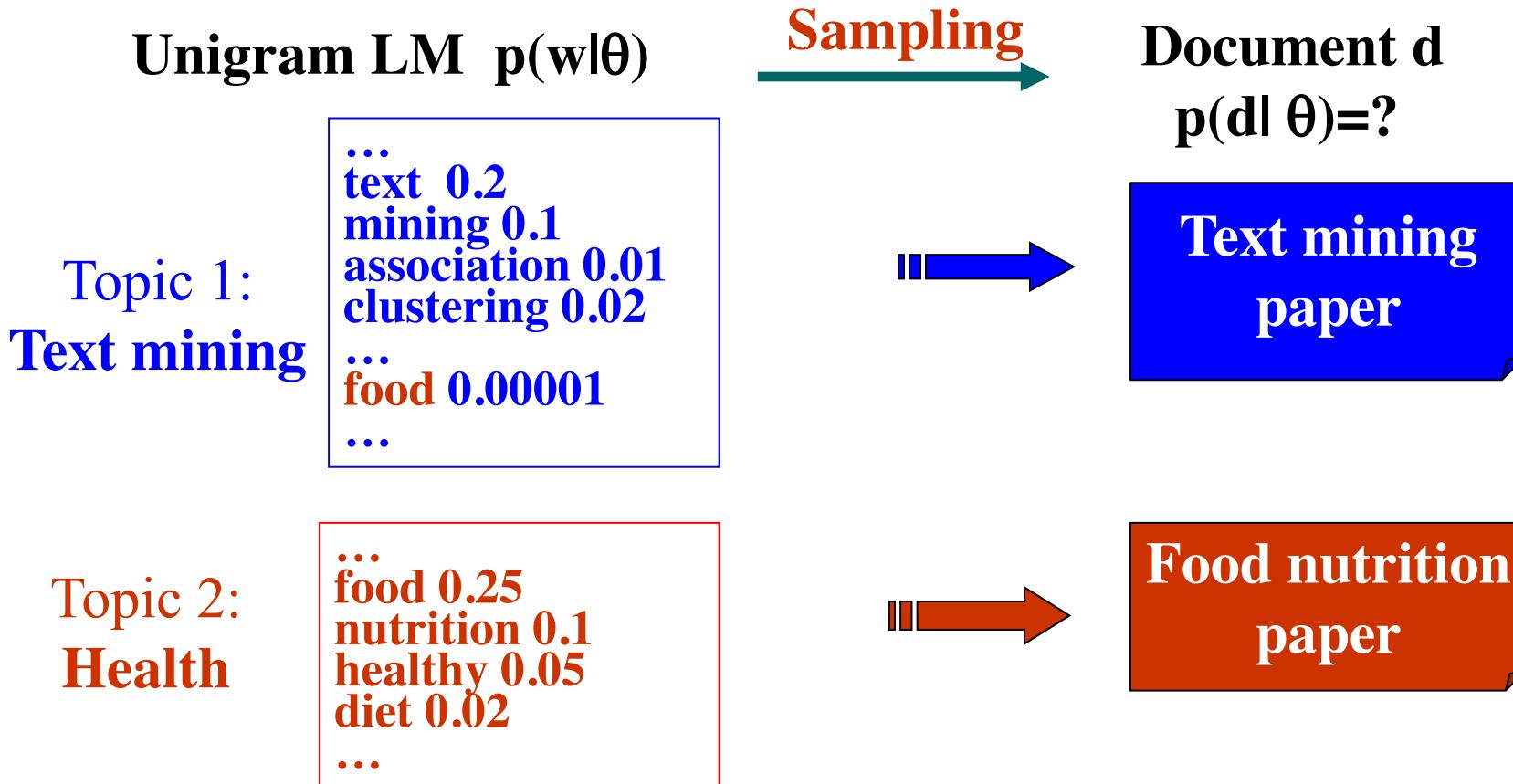
The Simplest Language Model: Unigram LM

- Generate text by generating each word INDEPENDENTLY
- Thus, $p(w_1 w_2 \dots w_n) = p(w_1)p(w_2)\dots p(w_n)$
- Parameters: $\{p(w_i)\}$ $p(w_1)+\dots+p(w_N)=1$ (N is voc. size)
- Text = sample drawn according to this **word distribution**



$$\begin{aligned} p(\text{"today is Wed"}) \\ &= p(\text{"today"})p(\text{"is"})p(\text{"Wed"}) \\ &= 0.0002 \times 0.001 \times 0.000015 \end{aligned}$$

Text Generation with Unigram LM



Estimation of Unigram LM

Unigram LM $p(w|\theta)=?$

Estimation

Text Mining Paper d

10/100
5/100
3/100
3/100
1/100

...
text ?
mining ?
association ?
database ?
...
query ?
...



Total #words=100

text 10
mining 5
association 3
database 3
algorithm 2
...
query 1
efficient 1

Maximum Likelihood
Estimate

Is this our best estimate?
How do we define “best”?

Maximum Likelihood vs. Bayesian

- Maximum likelihood estimation
 - “Best” means “data likelihood reaches maximum”

$$\hat{\theta} = \arg \max_{\theta} P(X | \theta)$$

- Problem: Small sample

- Bayesian estimation:

Bayes Rule

$$p(X | Y) = \frac{p(Y | X)p(X)}{p(Y)}$$

- “Best” means being consistent with our “prior” knowledge and explaining data well

$$\hat{\theta} = \arg \max_{\theta} P(\theta | X) = \arg \max_{\theta} P(X | \theta)P(\theta)$$

- Problem: How to define prior?



Maximum a Posteriori (MAP) estimate

Illustration of Bayesian Estimation

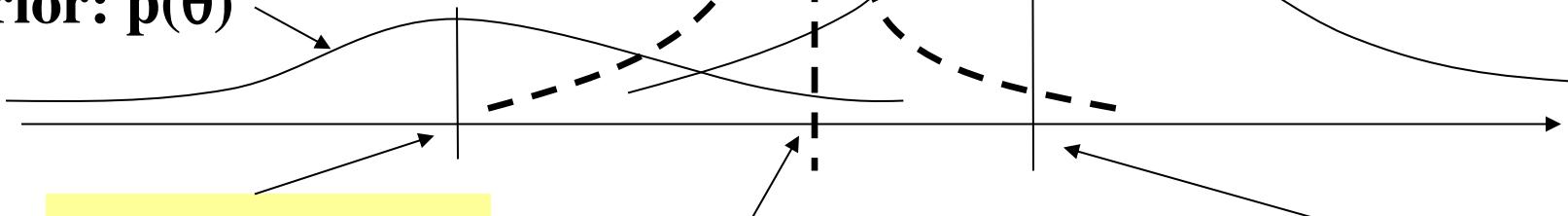
Bayesian inference: $f(\theta)=?$

$$\hat{f}(\theta) = \sum_{\theta} f(\theta)p(\theta | X)$$

Posterior
Mean

$$\hat{\theta} = \sum_{\theta} \theta * p(\theta | X)$$

Prior: $p(\theta)$



θ_0 : prior mode

θ_1 : posterior mode

θ_{ml} : ML estimate

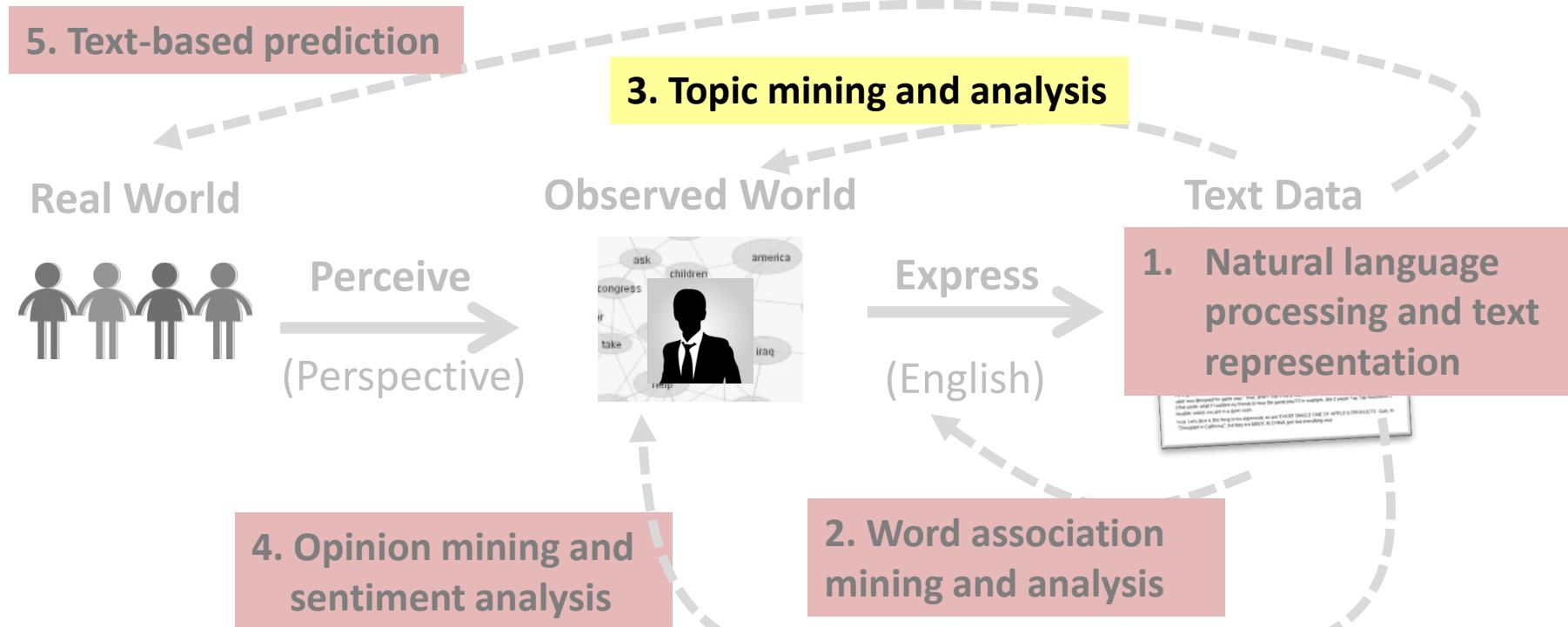
Summary

- **Language Model** = probability distribution over text = generative model for text data
- **Unigram** Language Model = **word distribution**
- **Likelihood** function: $p(X|\theta)$
 - Given $\theta \rightarrow$ which X has a higher likelihood?
 - Given $X \rightarrow$ which θ maximizes $p(X|\theta)$? [ML estimate]
- **Bayesian** estimation/inference
 - Must define a **prior**: $p(\theta)$
 - **Posterior** distribution: $p(\theta|X) \propto p(X|\theta)p(\theta)$
 - Allows for inferring any “derived value” from θ !

Topic Mining and Analysis: Overview of Statistical Language Models

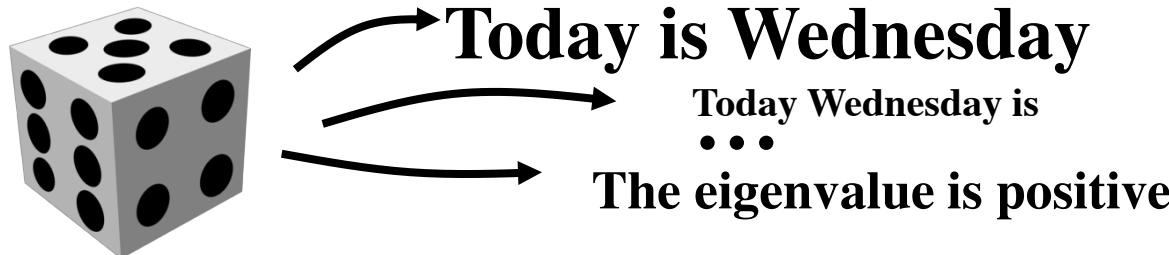
ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Probabilistic Topic Models: Overview of Statistical Language Models



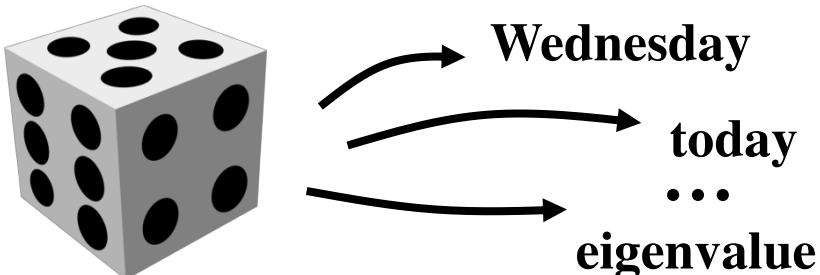
What Is a Statistical Language Model (LM)?

- A probability distribution over word sequences
 - $p(\text{"Today is Wednesday"}) \approx 0.001$
 - $p(\text{"Today Wednesday is"}) \approx 0.000000000001$
 - $p(\text{"The eigenvalue is positive"}) \approx 0.0001$
- Context-dependent!
- Can also be regarded as a probabilistic mechanism for “generating” text – thus also called a “generative” model



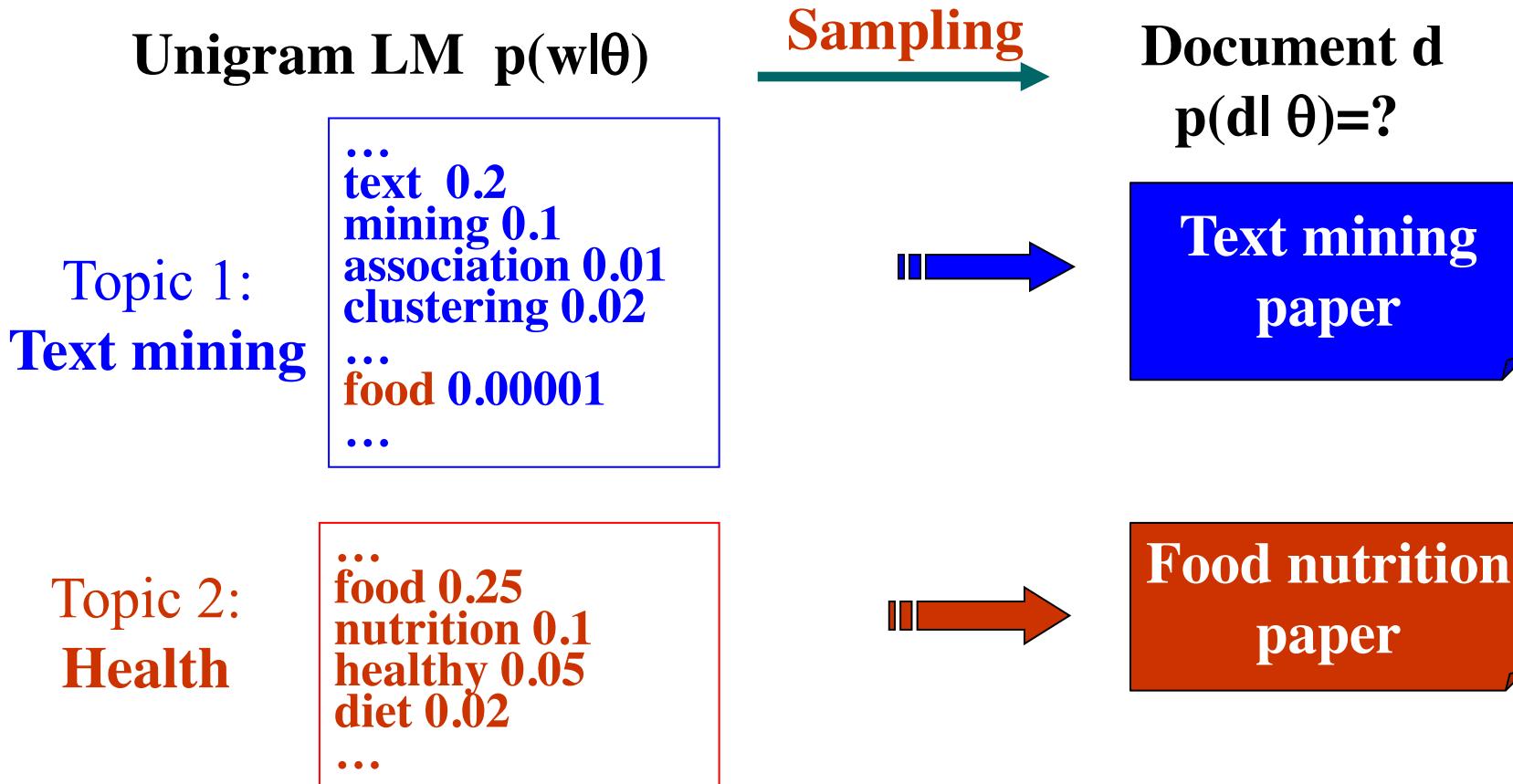
The Simplest Language Model: Unigram LM

- Generate text by generating each word INDEPENDENTLY
- Thus, $p(w_1 w_2 \dots w_n) = p(w_1)p(w_2)\dots p(w_n)$
- Parameters: $\{p(w_i)\}$ $p(w_1)+\dots+p(w_N)=1$ (N is voc. size)
- Text = sample drawn according to this **word distribution**



$$\begin{aligned} p(\text{"today is Wed"}) \\ &= p(\text{"today"})p(\text{"is"})p(\text{"Wed"}) \\ &= 0.0002 \times 0.001 \times 0.000015 \end{aligned}$$

Text Generation with Unigram LM



Estimation of Unigram LM

Unigram LM $p(w|\theta)=?$

Estimation

Text Mining Paper d

10/100
5/100
3/100
3/100
1/100

...
text ?
mining ?
association ?
database ?
...
query ?
...



Total #words=100

text 10
mining 5
association 3
database 3
algorithm 2
...
query 1
efficient 1

Maximum Likelihood
Estimate

Is this our best estimate?
How do we define “best”?

Maximum Likelihood vs. Bayesian

- Maximum likelihood estimation
 - “Best” means “data likelihood reaches maximum”

$$\hat{\theta} = \arg \max_{\theta} P(X | \theta)$$

- Problem: Small sample

- Bayesian estimation:

Bayes Rule

$$p(X | Y) = \frac{p(Y | X)p(X)}{p(Y)}$$

- “Best” means being consistent with our “prior” knowledge and explaining data well

$$\hat{\theta} = \arg \max_{\theta} P(\theta | X) = \arg \max_{\theta} P(X | \theta)P(\theta)$$

- Problem: How to define prior?



Maximum a Posteriori (MAP) estimate

Illustration of Bayesian Estimation

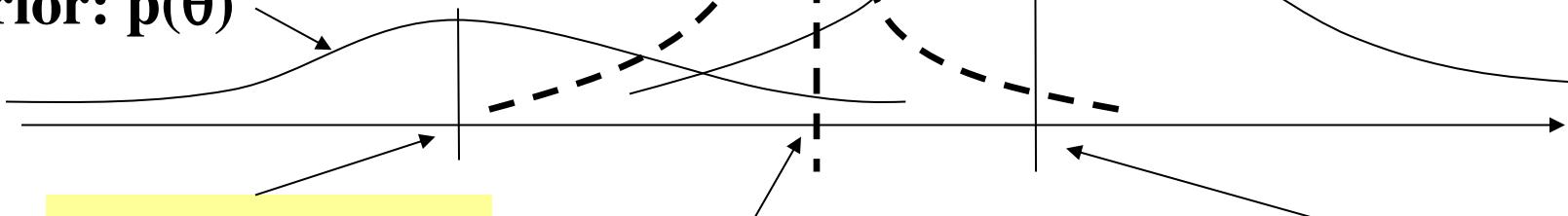
Bayesian inference: $f(\theta)=?$

$$\hat{f}(\theta) = \sum_{\theta} f(\theta)p(\theta | X)$$

Posterior
Mean

$$\hat{\theta} = \sum_{\theta} \theta * p(\theta | X)$$

Prior: $p(\theta)$



θ_0 : prior mode

θ_1 : posterior mode

θ_{ml} : ML estimate

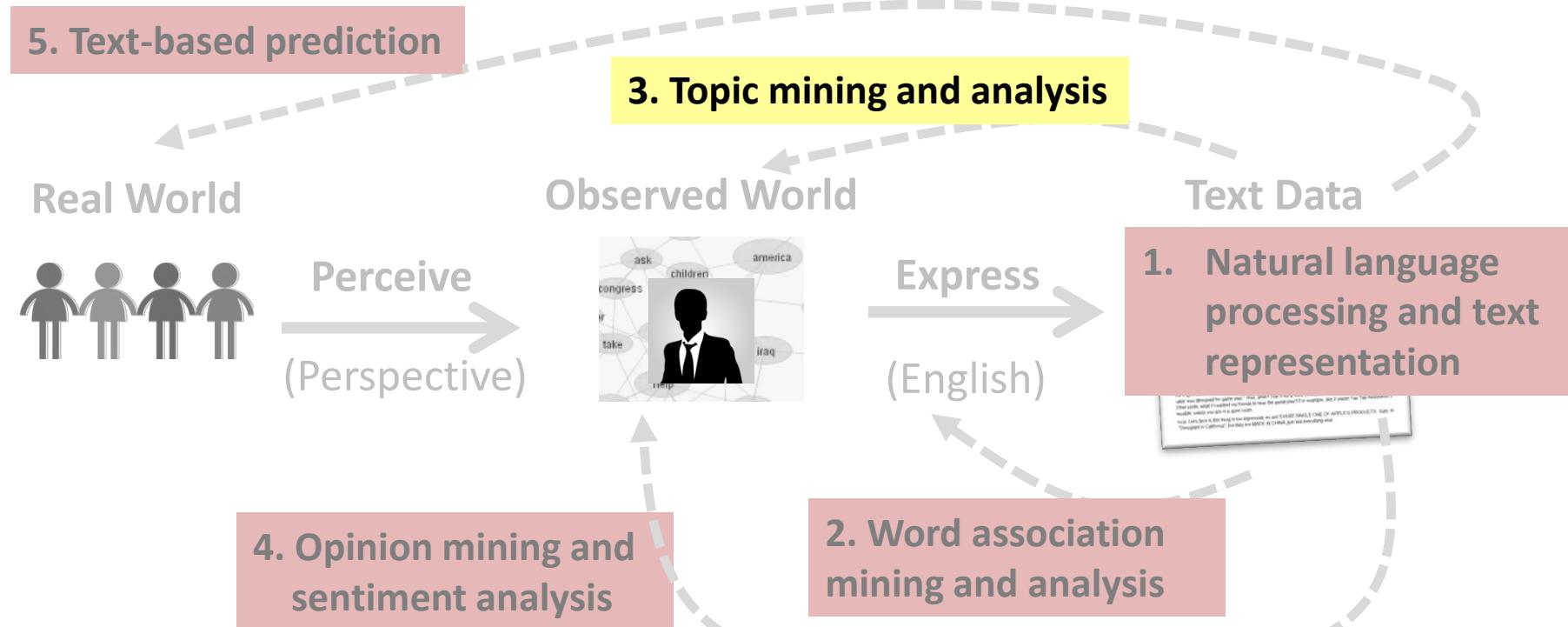
Summary

- **Language Model** = probability distribution over text = generative model for text data
- **Unigram** Language Model = **word distribution**
- **Likelihood** function: $p(X|\theta)$
 - Given $\theta \rightarrow$ which X has a higher likelihood?
 - Given $X \rightarrow$ which θ maximizes $p(X|\theta)$? [ML estimate]
- **Bayesian** estimation/inference
 - Must define a **prior**: $p(\theta)$
 - **Posterior** distribution: $p(\theta|X) \propto p(X|\theta)p(\theta)$
 - Allows for inferring any “derived value” from θ !

Topic Mining and Analysis: Mining One Topic

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Probabilistic Topic Models: Mining One Topic



Simplest Case of Topic Model: Mining One Topic

INPUT: $C=\{d\}$, V

OUTPUT: $\{\theta\}$

Text Data

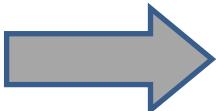
...
... I am still considering taking it back. Here's why:

Speaker quality is ABSOLUTELY HORRIBLY BAD. The speaker is simply not functional, unless you are in perhaps a recording studio. When you turn it up all the way because you can't hear it, it becomes fuzzy, sounding like a blown eardrum. What is this thing, the size of a flea? And if there is ANY background noise, you cannot hear it at all with the volume up (in case you're wondering, no, the speaker did not fail). I exchanged the iPod, and the second one sounds a little better, but I can't share videos, share music, share anything, unless my friends and I are in a wine cellar.

I heard excuses for the speaker's embarrassing quality, including "it's a small device". My cheap Samsung cell phone's speaker is 4x better than the iPod Touch, and the phone is smaller. The other excuse I've heard is "The speaker was designed for game play". Wait, what? That's not a valid excuse, because the iPod is a MUSIC device. If that aside, what if I wanted my friends to hear the game play? For example, like 2 player Tap Tap Revolution 3? playable, unless you are in a quiet room.

Price: Let's face it, this thing is too expensive, as are EVERY SINGLE ONE OF APPLE'S PRODUCTS. Sure, it's "Designed in California", but they are MADE IN CHINA, just like everything else.

More reviews: [WAVY](#) is [more](#).



$P(w|\theta)$

θ

text ?
mining ?
association ?
database ?
...
query ?
...

Doc d

100%

Language Model Setup

- **Data:** Document $d = x_1 x_2 \dots x_{|d|}$, $x_i \in V = \{w_1, \dots, w_M\}$ is a word
- **Model:** Unigram LM $\theta (= \text{topic}) : \{\theta_i = p(w_i | \theta)\}, i=1, \dots, M$;
 $\theta_1 + \dots + \theta_M = 1$
- **Likelihood function:** $p(d | \theta) = p(x_1 | \theta) \times \dots \times p(x_{|d|} | \theta)$
 $= p(w_1 | \theta)^{c(w_1, d)} \times \dots \times p(w_M | \theta)^{c(w_M, d)}$
 $= \prod_{i=1}^M p(w_i | \theta)^{c(w_i, d)} = \prod_{i=1}^M \theta_i^{c(w_i, d)}$
- **ML estimate:** $(\hat{\theta}_1, \dots, \hat{\theta}_M) = \arg \max_{\theta_1, \dots, \theta_M} p(d | \theta) = \arg \max_{\theta_1, \dots, \theta_M} \prod_{i=1}^M \theta_i^{c(w_i, d)}$

Computation of Maximum Likelihood Estimate

Maximize $p(d | \theta)$ $(\hat{\theta}_1, \dots, \hat{\theta}_M) = \arg \max_{\theta_1, \dots, \theta_M} p(d | \theta) = \arg \max_{\theta_1, \dots, \theta_M} \prod_{i=1}^M \theta_i^{c(w_i, d)}$

Max. Log-Likelihood $(\hat{\theta}_1, \dots, \hat{\theta}_M) = \arg \max_{\theta_1, \dots, \theta_M} \log[p(d | \theta)] = \arg \max_{\theta_1, \dots, \theta_M} \sum_{i=1}^M c(w_i, d) \log \theta_i$

Subject to constraint: $\sum_{i=1}^M \theta_i = 1$

Use Lagrange multiplier approach

Lagrange function: $f(\theta | d) = \sum_{i=1}^M c(w_i, d) \log \theta_i + \lambda \left(\sum_{i=1}^M \theta_i - 1 \right)$

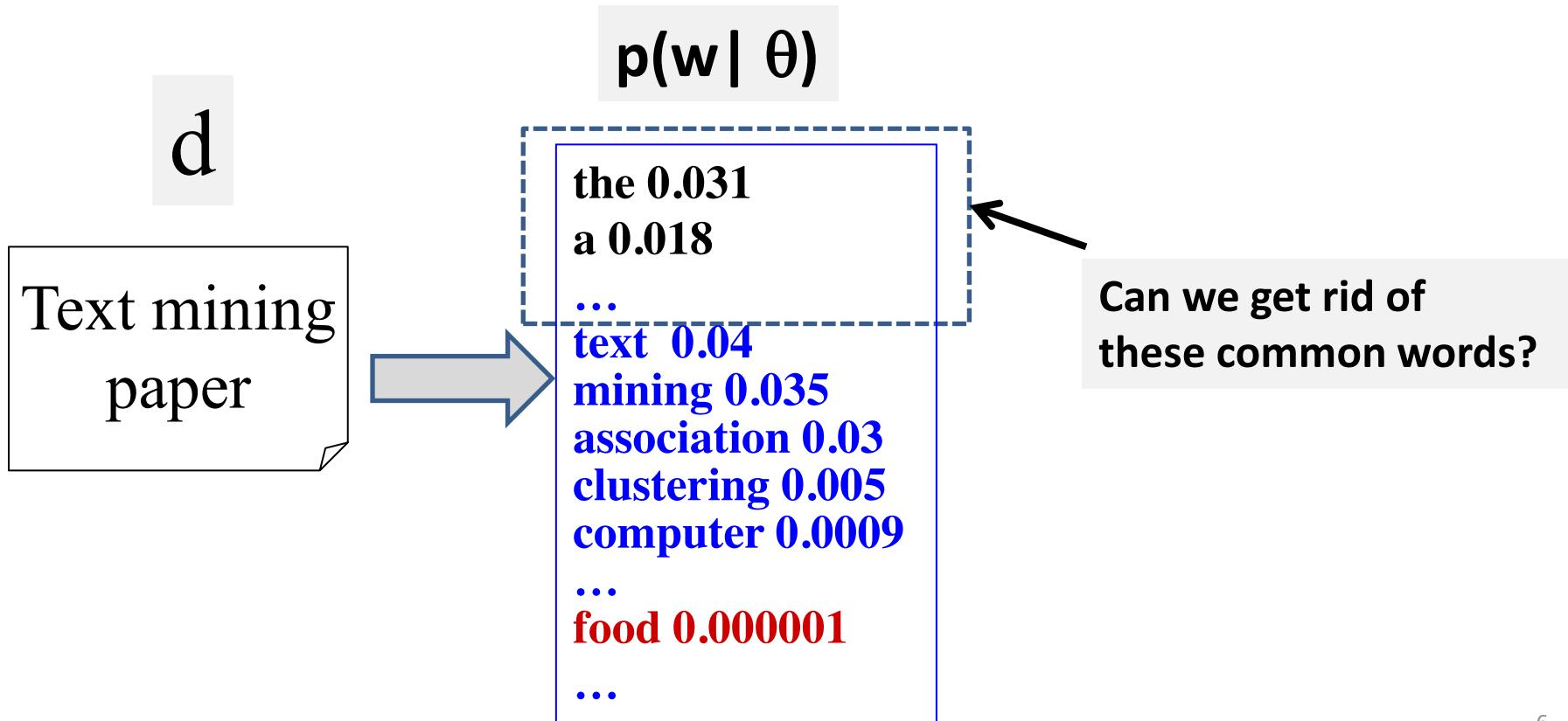
$$\frac{\partial f(\theta | d)}{\partial \theta_i} = \frac{c(w_i, d)}{\theta_i} + \lambda = 0 \rightarrow \theta_i = \frac{c(w_i, d)}{\lambda}$$

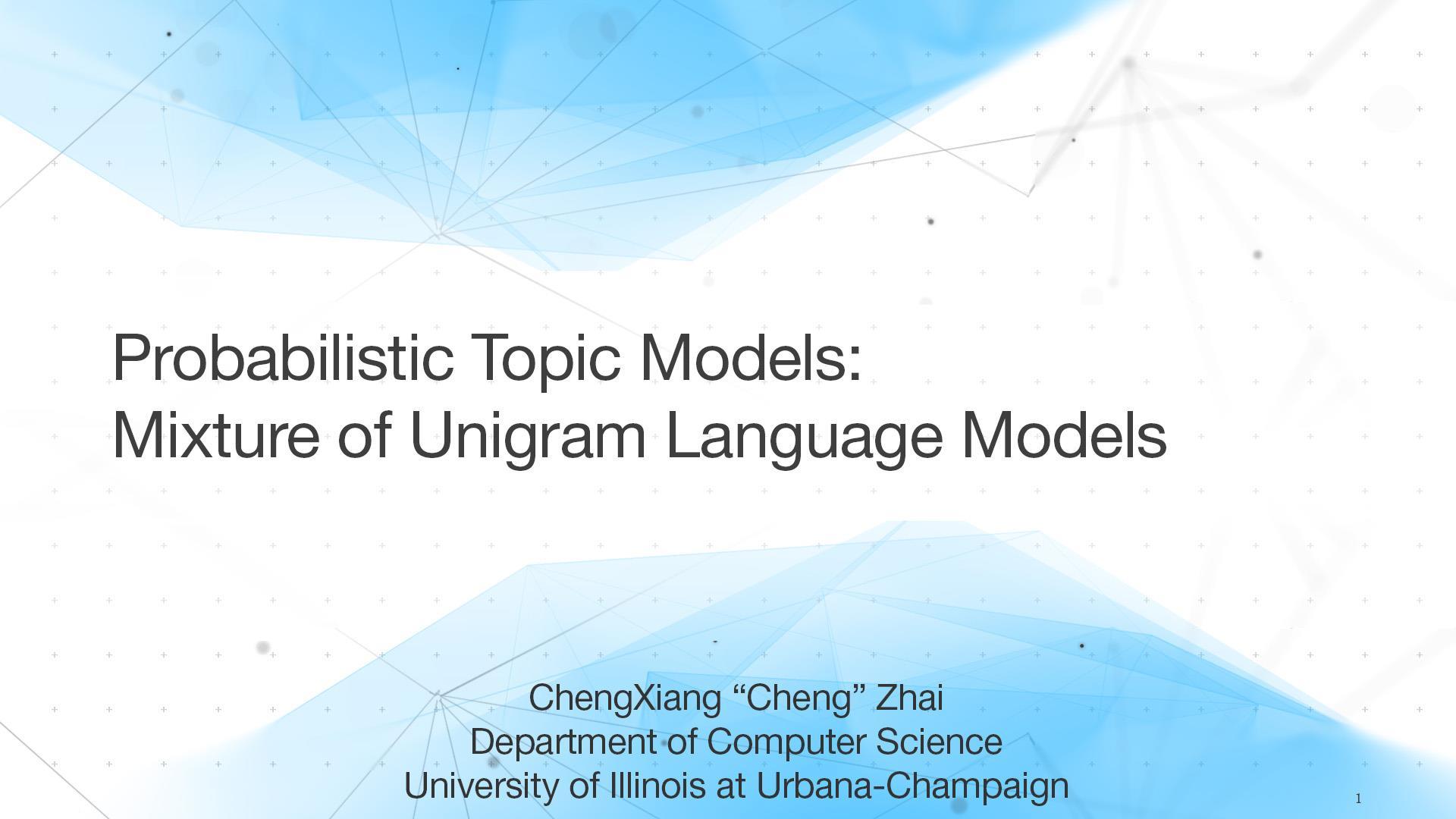
$$\sum_{i=1}^M \frac{c(w_i, d)}{\lambda} = 1 \rightarrow \lambda = \sum_{i=1}^N c(w_i, d) \rightarrow \hat{\theta}_i = p(w_i | \hat{\lambda}) = \frac{c(w_i, d)}{\sum_{i=1}^M c(w_i, d)} = \frac{c(w_i, d)}{|d|}$$

Normalized Counts



What Does the Topic Look Like?

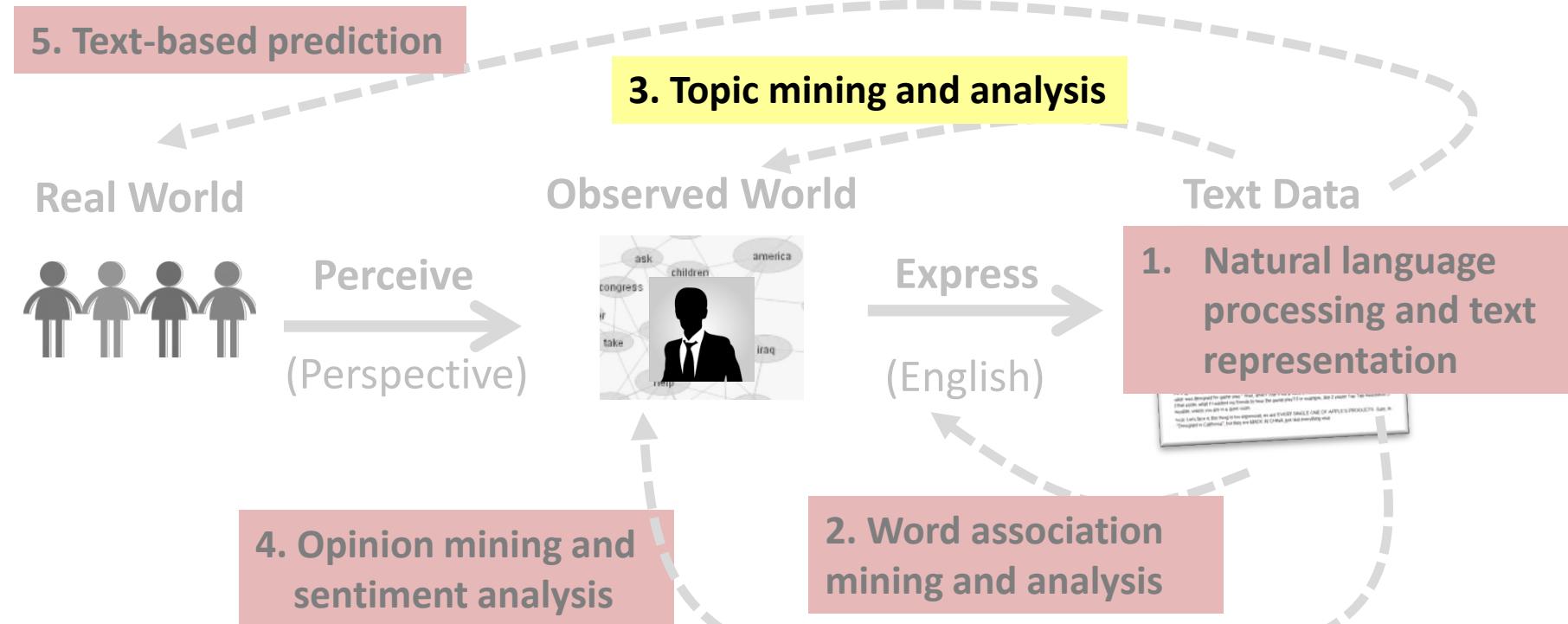




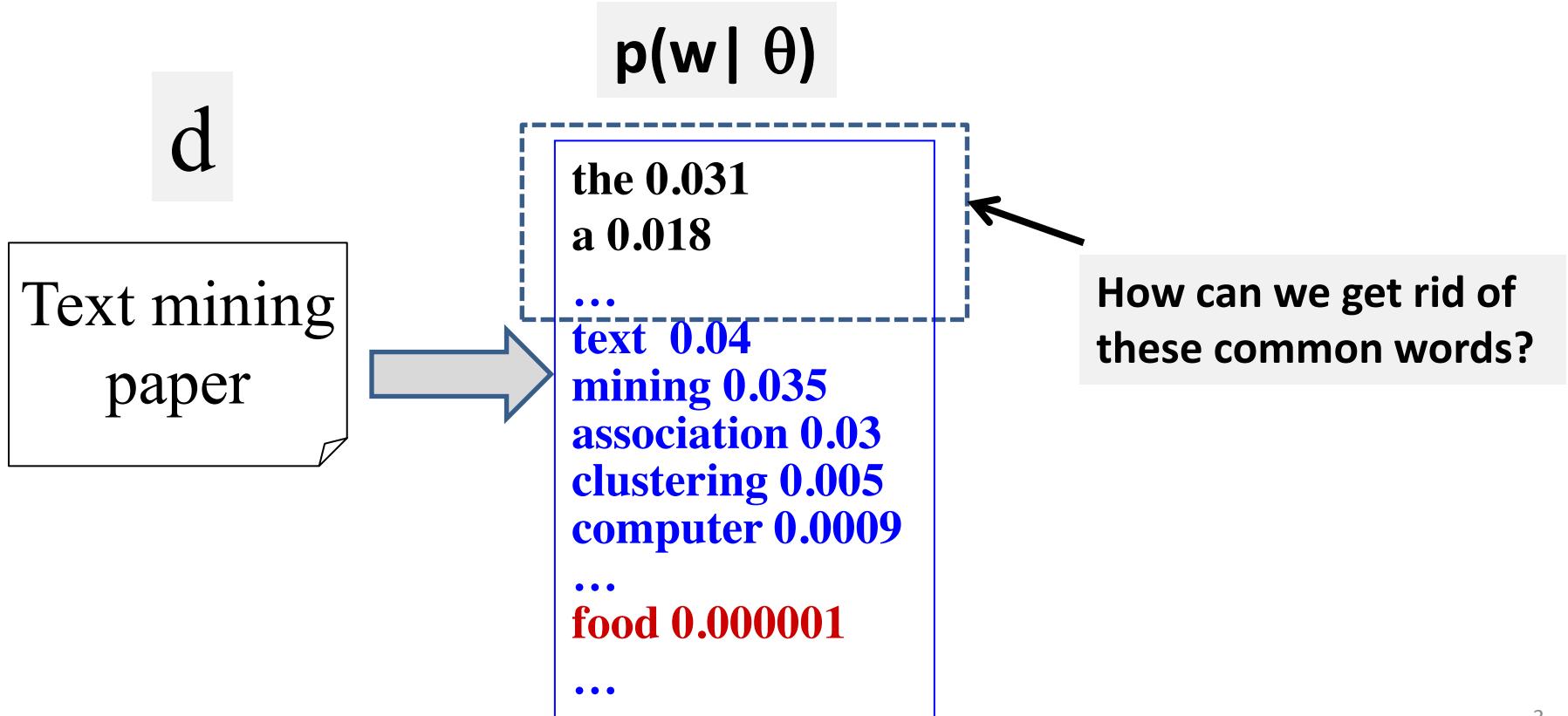
Probabilistic Topic Models: Mixture of Unigram Language Models

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

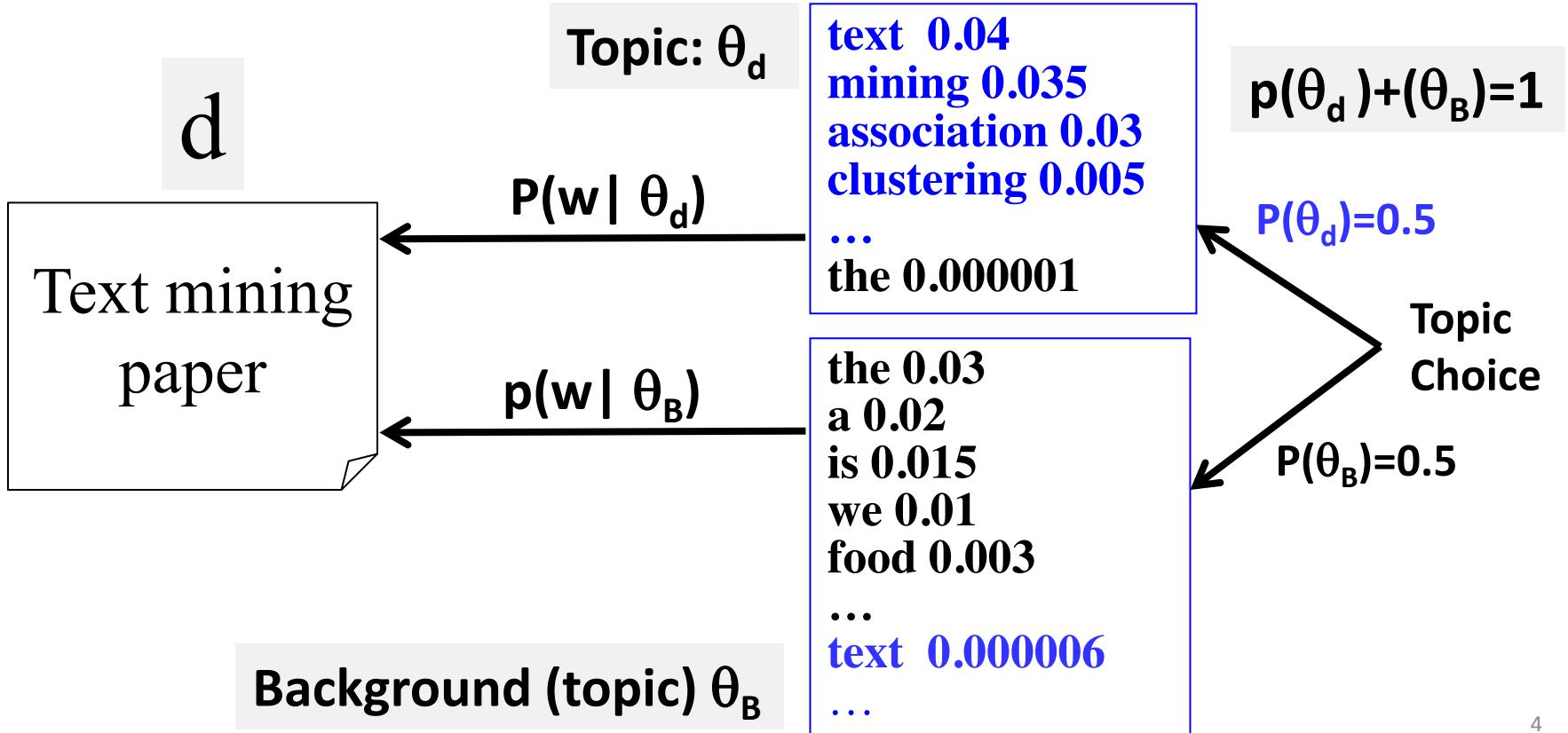
Probabilistic Topic Models: Mixture of Unigram LMs



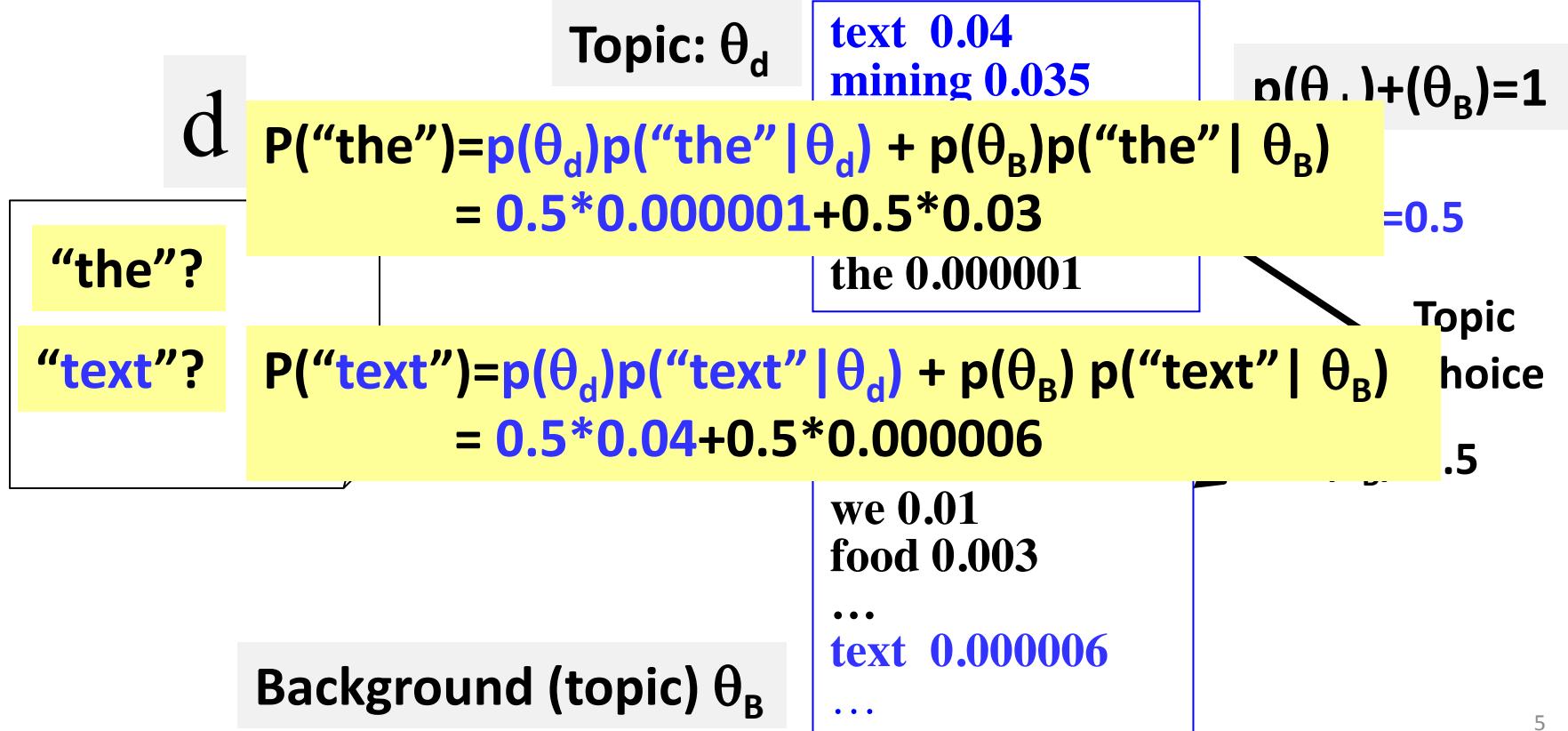
Factoring out Background Words



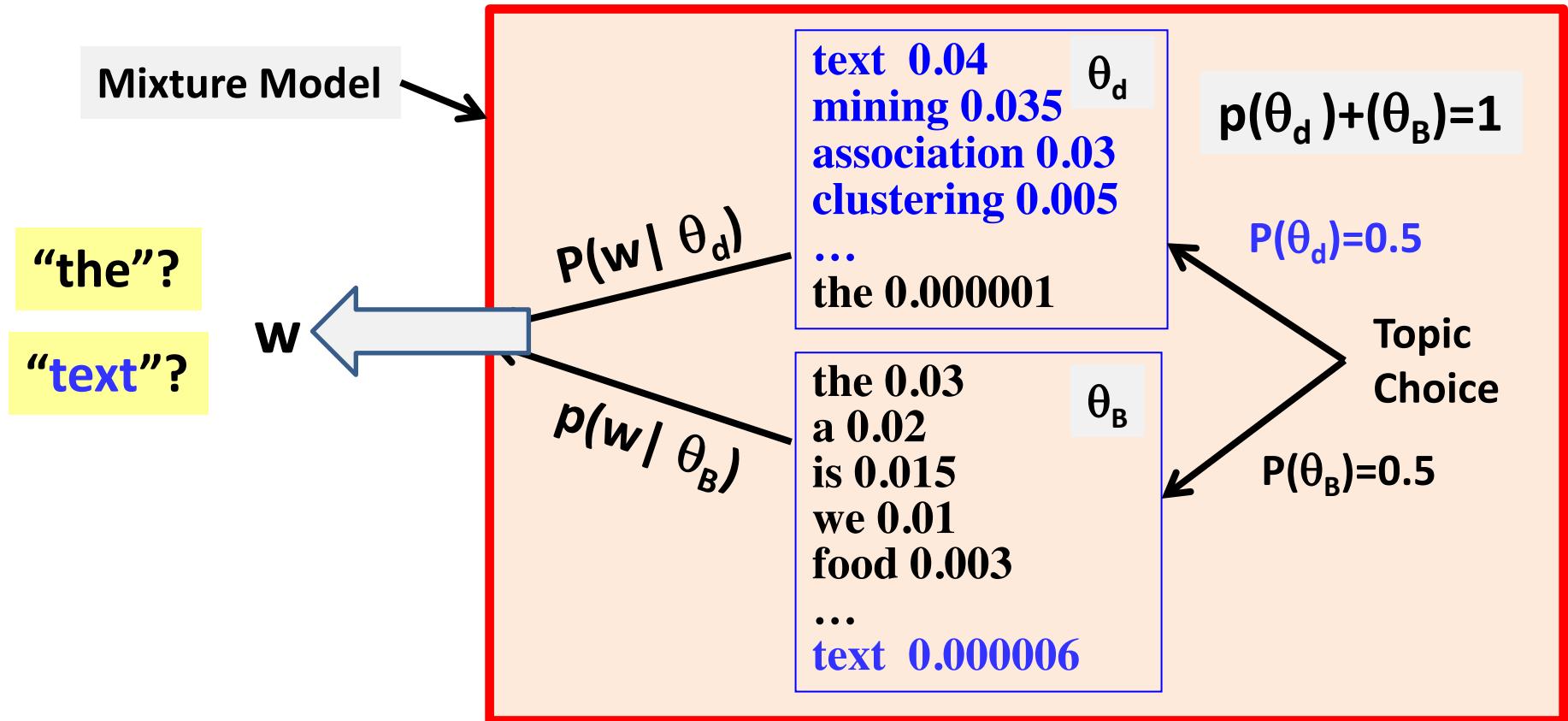
Generate d Using Two Word Distributions



What's the probability of observing a word w?



The Idea of a Mixture Model



As a Generative Model...

text 0.04 θ_d
mining 0.035
association 0.03
clustering 0.005

$$p(\theta_d) + (\theta_B) = 1$$

Formally defines the following generative model:

$$p(w) = p(\theta_d)p(w|\theta_d) + p(\theta_B)p(w|\theta_B)$$

w

Estimate of the model “discovers”
two topics + topic coverage

What if $p(\theta_d) = 1$ or $p(\theta_B) = 1$?

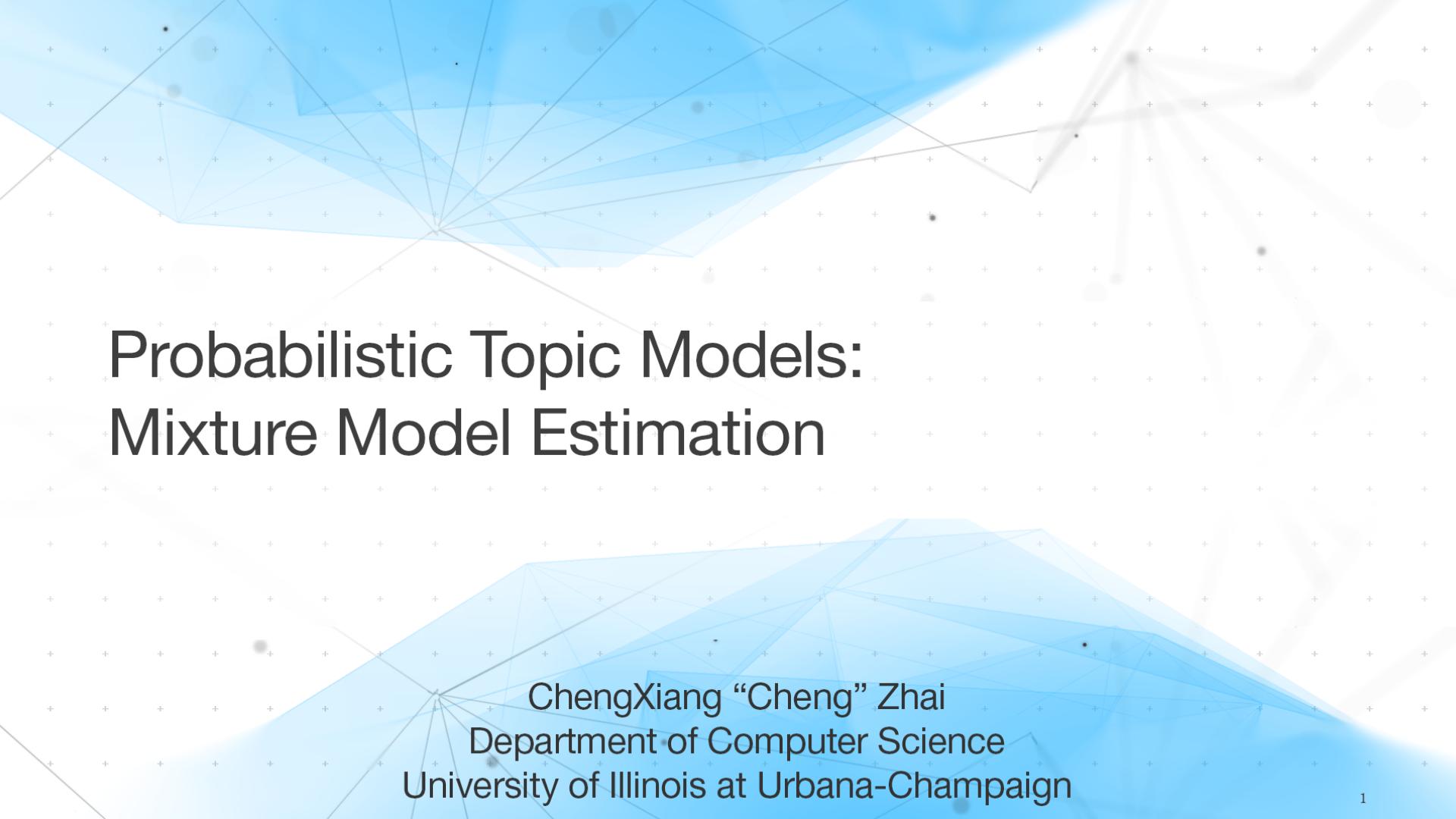
Mixture of Two Unigram Language Models

- **Data:** Document d
- **Mixture Model: parameters** $\Lambda = (\{p(w|\theta_d)\}, \{p(w|\theta_B)\}, p(\theta_B), p(\theta_d))$
 - Two unigram LMs: θ_d (**the topic of d**); θ_B (**background topic**)
 - Mixing weight (topic choice): $p(\theta_d) + p(\theta_B) = 1$
- **Likelihood function:**

$$\begin{aligned} p(d | \Lambda) &= \prod_{i=1}^{|d|} p(x_i | \Lambda) = \prod_{i=1}^{|d|} [p(\theta_d)p(x_i | \theta_d) + p(\theta_B)p(x_i | \theta_B)] \\ &= \prod_{i=1}^M [p(\theta_d)p(w_i | \theta_d) + p(\theta_B)p(w_i | \theta_B)]^{c(w,d)} \end{aligned}$$

- **ML Estimate:** $\Lambda^* = \arg \max_{\Lambda} p(d | \Lambda)$

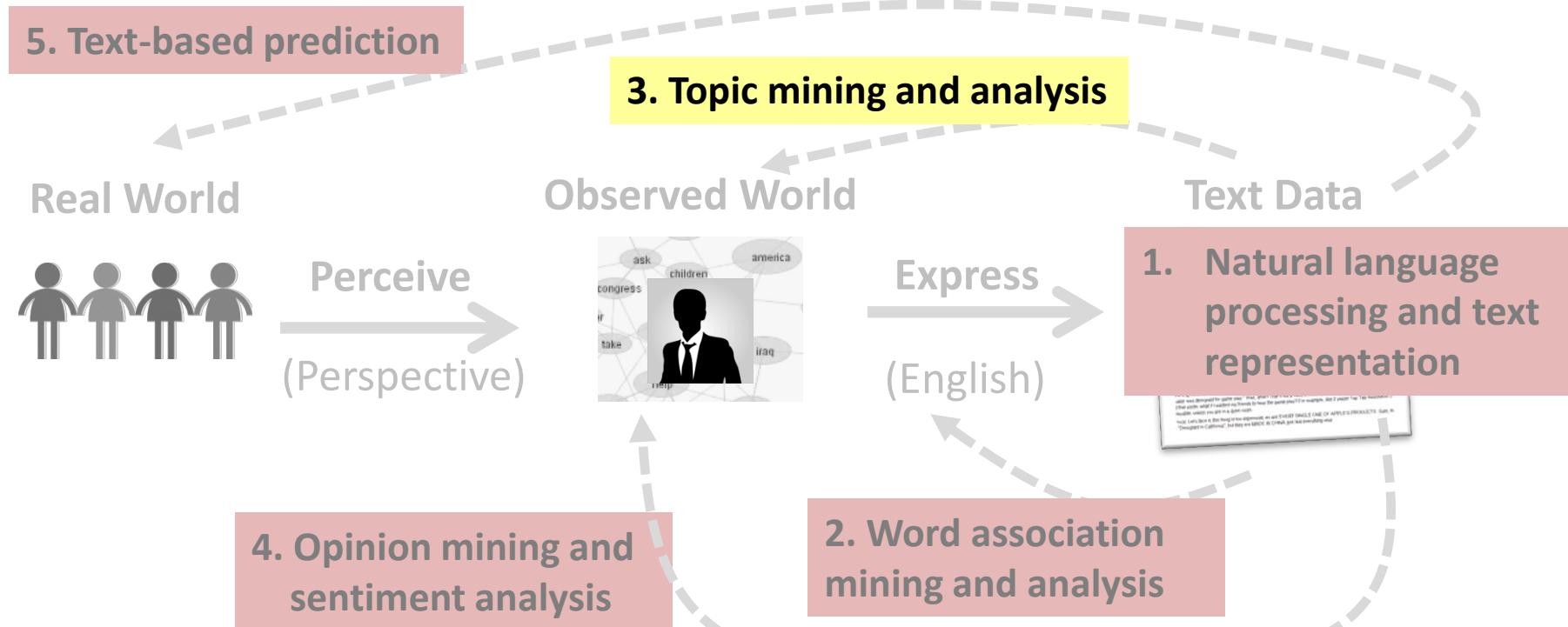
Subject to $\sum_{i=1}^M p(w_i | \theta_d) = \sum_{i=1}^M p(w_i | \theta_B) = 1 \quad p(\theta_d) + p(\theta_B) = 1$



Probabilistic Topic Models: Mixture Model Estimation

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Probabilistic Topic Models: Mixture Model Estimation



Back to Factoring out Background Words

Text Mining Paper

d



$$p(w | \theta_d)$$

text 0.04
mining 0.035
association 0.03
clustering 0.005
...
the 0.000001

$$p(\theta_d) + (\theta_B) = 1$$

$$P(\theta_d) = 0.5$$

Topic
Choice

$$p(w | \theta_B)$$

the 0.03
a 0.02
is 0.015
we 0.01
food 0.003
...
text 0.000006

$$P(\theta_B) = 0.5$$

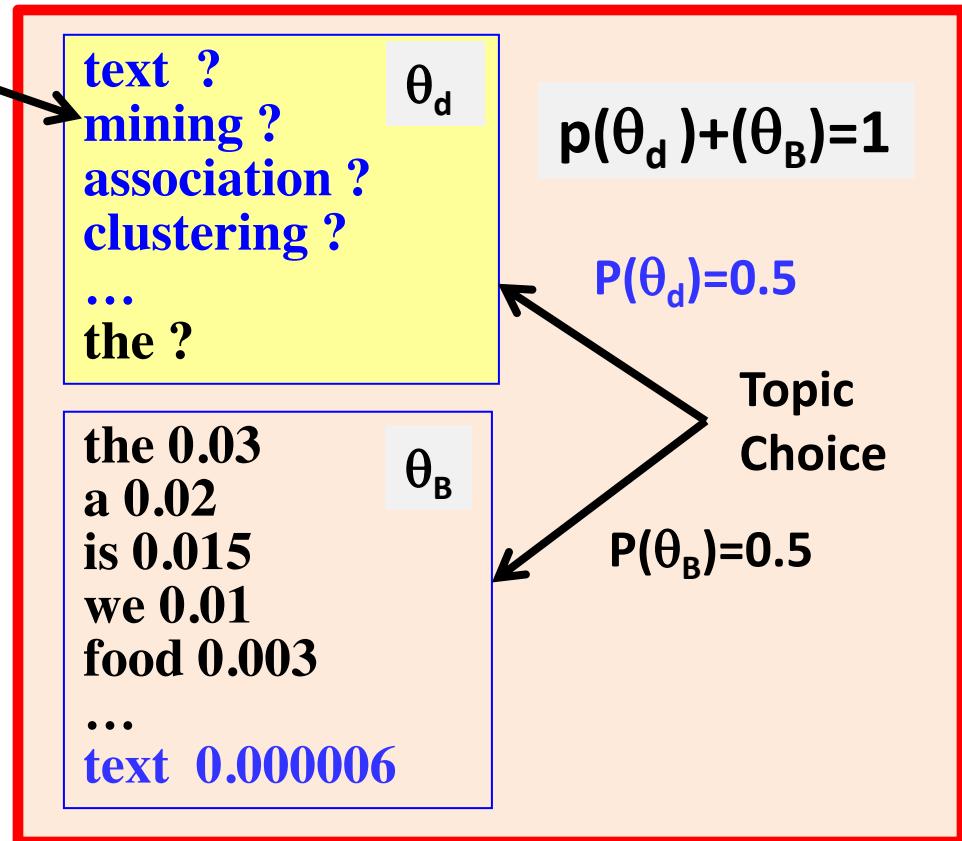
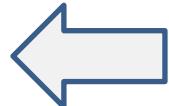
Estimation of One Topic: $P(w | \theta_d)$

Adjust θ_d to maximize $p(d | \Lambda)$
(all other parameters are known)

Would the ML estimate demote
background words in θ_d ?

d

... text mining...
is... clustering...
we.... Text.. the



Behavior of a Mixture Model

$d = \boxed{\text{text the}}$

Likelihood:

$$\begin{aligned} P(\text{"text"}) &= p(\theta_d)p(\text{"text"} | \theta_d) + p(\theta_B)p(\text{"text"} | \theta_B) \\ &= 0.5 * p(\text{"text"} | \theta_d) + 0.5 * 0.1 \end{aligned}$$

$$P(\text{"the"}) = 0.5 * p(\text{"the"} | \theta_d) + 0.5 * 0.9$$

$$\begin{aligned} p(d | \Lambda) &= p(\text{"text"} | \Lambda) p(\text{"the"} | \Lambda) \\ &= [0.5 * p(\text{"text"} | \theta_d) + 0.5 * 0.1] \times \\ &\quad [0.5 * p(\text{"the"} | \theta_d) + 0.5 * 0.9] \end{aligned}$$

$\boxed{\begin{matrix} \text{text ?} & \theta_d \\ \text{the ?} & \end{matrix}}$

$P(\theta_d) = 0.5$

$\boxed{\begin{matrix} \text{the } 0.9 & \theta_B \\ \text{text } 0.1 & \end{matrix}}$

How can we set $p(\text{"text"} | \theta_d)$ & $p(\text{"the"} | \theta_d)$ to maximize it?

Note that $p(\text{"text"} | \theta_d) + p(\text{"the"} | \theta_d) = 1$

“Collaboration” and “Competition” of θ_d and θ_B

$$\begin{aligned} p(d|\Lambda) &= p(\text{"text"}|\Lambda) p(\text{"the"}|\Lambda) \\ &= [0.5 * p(\text{"text"}|\theta_d) + 0.5 * 0.1] \times \\ &\quad [0.5 * p(\text{"the"}|\theta_d) + 0.5 * 0.9] \end{aligned}$$

Note that $p(\text{"text"}|\theta_d) + p(\text{"the"}|\theta_d) = 1$

If $x + y = \text{constant}$, then xy reaches maximum when $x = y$.

$$0.5 * p(\text{"text"}|\theta_d) + 0.5 * 0.1 = 0.5 * p(\text{"the"}|\theta_d) + 0.5 * 0.9$$

$$\rightarrow p(\text{"text"}|\theta_d) = 0.9 \quad \gg \quad p(\text{"the"}|\theta_d) = 0.1 !$$

$d =$ text the

text ? θ_d

$P(\theta_d) = 0.5$

$P(\theta_B) = 0.5$

the 0.9 text 0.1 θ_B

Behavior 1: if $p(w_1|\theta_B) > p(w_2|\theta_B)$, then $p(w_1|\theta_d) < p(w_2|\theta_d)$

Response to Data Frequency

$d = \boxed{\text{text the}}$

$$p(d|\Lambda) = [0.5*p(\text{"text"}|\theta_d) + 0.5*0.1] \\ \times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9]$$

$$\rightarrow p(\text{"text"}|\theta_d)=0.9 \quad >> \quad p(\text{"the"}|\theta_d)=0.1 !$$

$d' = \boxed{\text{text the}} \\ \text{the the} \\ \text{the ...the}}$

$$p(d'|\Lambda) = [0.5*p(\text{"text"}|\theta_d) + 0.5*0.1] \\ \times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9] \\ \times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9] \\ \times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9]$$

...

What if we increase $p(\theta_B)$?

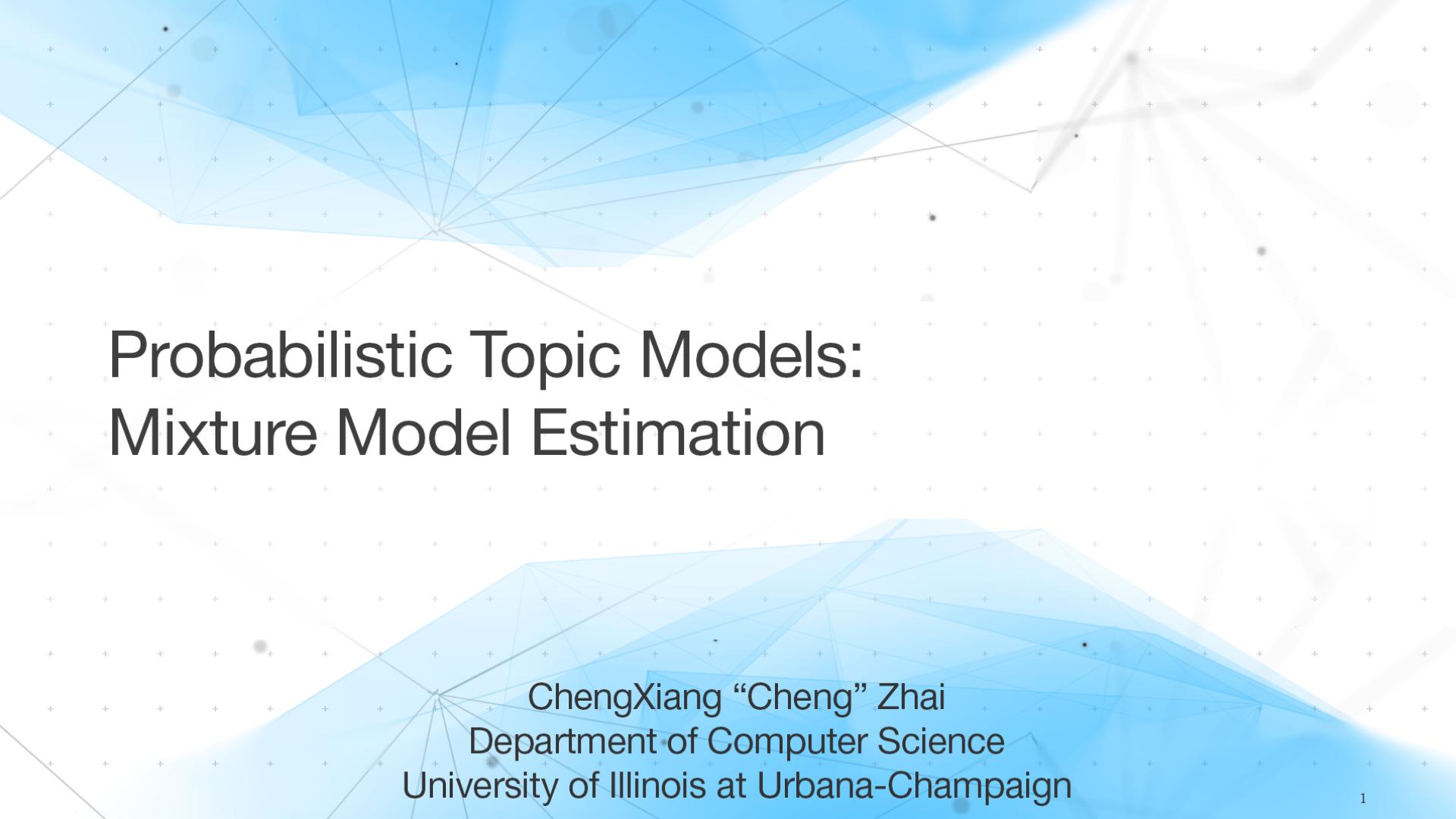
$$\times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9]$$

What's the optimal solution now? $p(\text{"the"}|\theta_d) > 0.1$? or $p(\text{"the"}|\theta_d) < 0.1$?

Behavior 2: high frequency words get higher $p(w|\theta_d)$

Summary

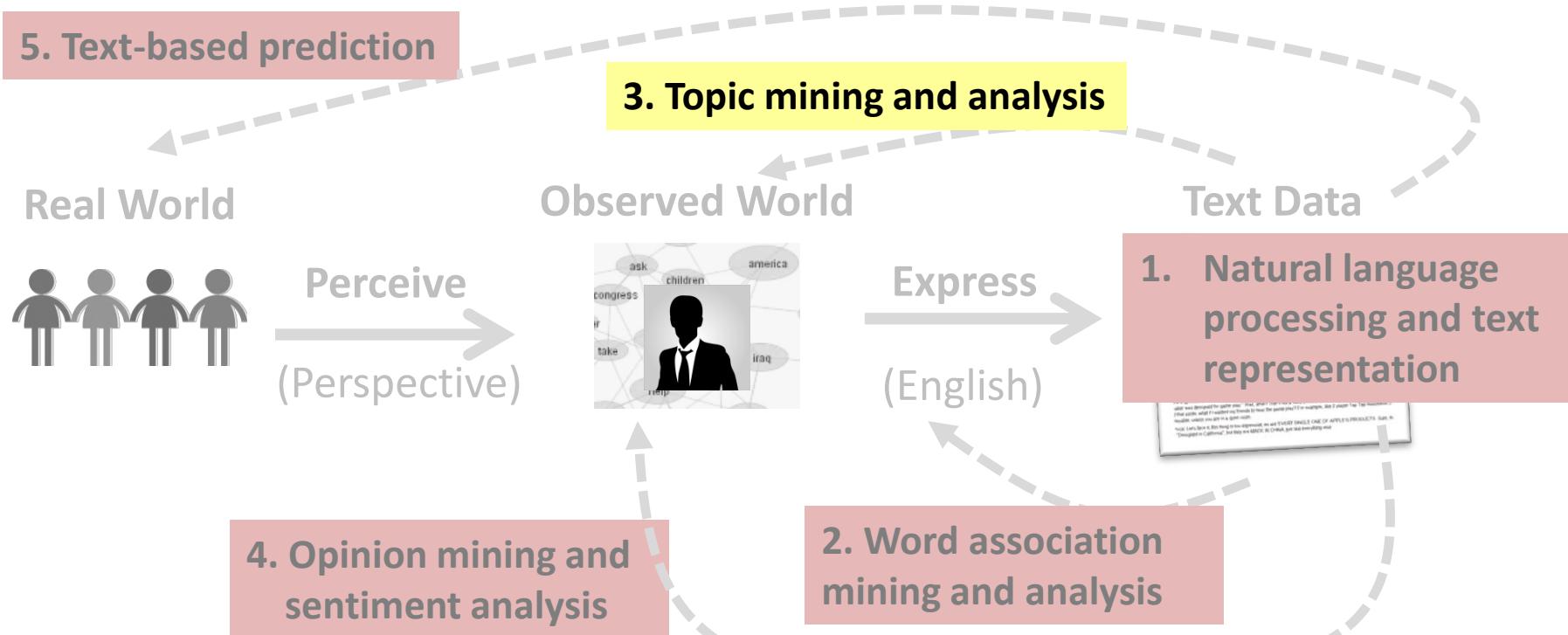
- General behavior of a mixture model:
 - Every component model attempts to assign high probabilities to highly frequent words in the data (to “collaboratively maximize likelihood”)
 - Different component models tend to “bet” high probabilities on different words (to avoid “competition” or “waste of probability”)
 - The probability of choosing each component “regulates” the collaboration/competition between the component models
- Fixing one component to a background word distribution (i.e., background language model):
 - Helps “get rid of background words” in other component
 - Is an example of imposing a prior on the model parameters (prior = one model must be exactly the same as the background LM)



Probabilistic Topic Models: Mixture Model Estimation

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Probabilistic Topic Models: Mixture Model Estimation



Back to Factoring out Background Words

Text Mining Paper

d



$$p(w | \theta_d)$$

text 0.04
mining 0.035
association 0.03
clustering 0.005
...
the 0.000001

$$p(\theta_d) + (\theta_B) = 1$$

$$P(\theta_d) = 0.5$$

Topic
Choice

$$p(w | \theta_B)$$

the 0.03
a 0.02
is 0.015
we 0.01
food 0.003
...
text 0.000006

$$P(\theta_B) = 0.5$$

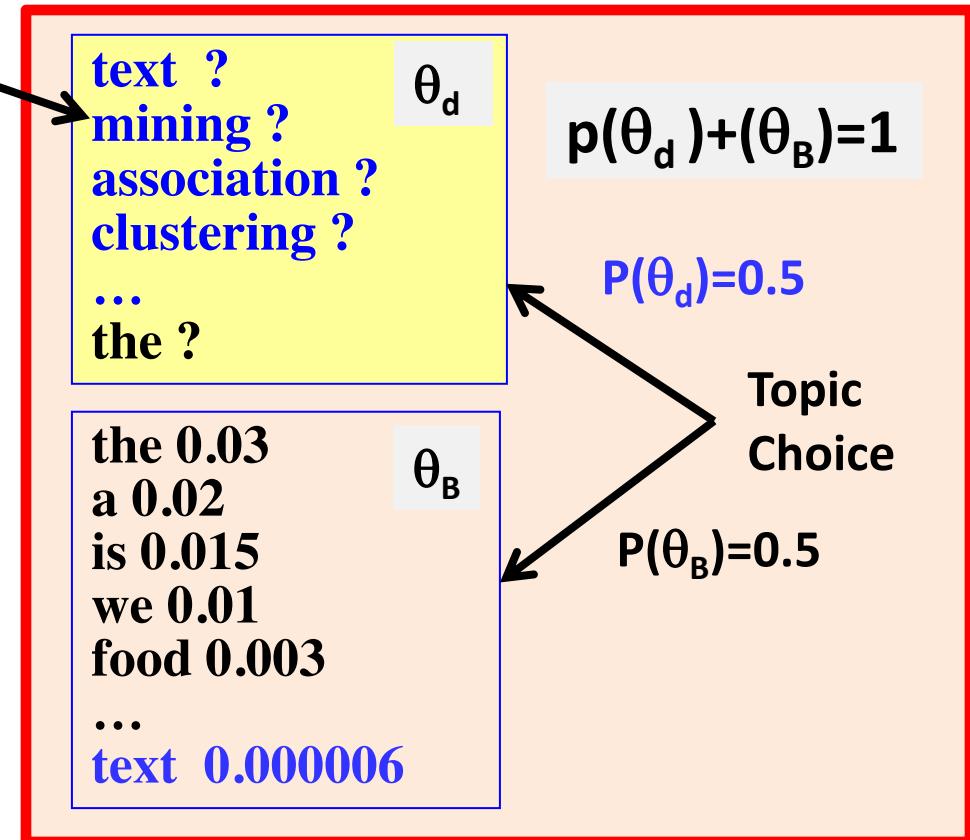
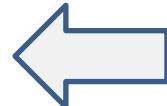
Estimation of One Topic: $P(w | \theta_d)$

Adjust θ_d to maximize $p(d | \Lambda)$
(all other parameters are known)

Would the ML estimate demote
background words in θ_d ?

d

... text mining...
is... clustering...
we.... Text.. the



Behavior of a Mixture Model

$d = \boxed{\text{text the}}$

Likelihood:

$$\begin{aligned} P(\text{"text"}) &= p(\theta_d)p(\text{"text"} | \theta_d) + p(\theta_B)p(\text{"text"} | \theta_B) \\ &= 0.5 * p(\text{"text"} | \theta_d) + 0.5 * 0.1 \end{aligned}$$

$$P(\text{"the"}) = 0.5 * p(\text{"the"} | \theta_d) + 0.5 * 0.9$$

$$\begin{aligned} p(d | \Lambda) &= p(\text{"text"} | \Lambda) p(\text{"the"} | \Lambda) \\ &= [0.5 * p(\text{"text"} | \theta_d) + 0.5 * 0.1] \times \\ &\quad [0.5 * p(\text{"the"} | \theta_d) + 0.5 * 0.9] \end{aligned}$$

$\boxed{\begin{matrix} \text{text ?} & \theta_d \\ \text{the ?} & \end{matrix}}$

$P(\theta_d) = 0.5$

$\boxed{\begin{matrix} \text{the } 0.9 & \theta_B \\ \text{text } 0.1 & \end{matrix}}$

How can we set $p(\text{"text"} | \theta_d)$ & $p(\text{"the"} | \theta_d)$ to maximize it?

Note that $p(\text{"text"} | \theta_d) + p(\text{"the"} | \theta_d) = 1$

“Collaboration” and “Competition” of θ_d and θ_B

$$\begin{aligned} p(d|\Lambda) &= p(\text{"text"}|\Lambda) p(\text{"the"}|\Lambda) \\ &= [0.5 * p(\text{"text"}|\theta_d) + 0.5 * 0.1] \times \\ &\quad [0.5 * p(\text{"the"}|\theta_d) + 0.5 * 0.9] \end{aligned}$$

Note that $p(\text{"text"}|\theta_d) + p(\text{"the"}|\theta_d) = 1$

If $x + y = \text{constant}$, then xy reaches maximum when $x = y$.

$$0.5 * p(\text{"text"}|\theta_d) + 0.5 * 0.1 = 0.5 * p(\text{"the"}|\theta_d) + 0.5 * 0.9$$

$$\rightarrow p(\text{"text"}|\theta_d) = 0.9 \quad \gg \quad p(\text{"the"}|\theta_d) = 0.1 !$$

$d =$ text the

text ? θ_d

$P(\theta_d) = 0.5$

$P(\theta_B) = 0.5$

the 0.9 text 0.1 θ_B

Behavior 1: if $p(w_1|\theta_B) > p(w_2|\theta_B)$, then $p(w_1|\theta_d) < p(w_2|\theta_d)$

Response to Data Frequency

$d = \boxed{\text{text the}}$

$$p(d|\Lambda) = [0.5*p(\text{"text"}|\theta_d) + 0.5*0.1] \\ \times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9]$$

$$\rightarrow p(\text{"text"}|\theta_d)=0.9 \quad >> \quad p(\text{"the"}|\theta_d)=0.1 !$$

$d' = \boxed{\text{text the}} \\ \text{the the} \\ \text{the ...the}}$

$$p(d'|\Lambda) = [0.5*p(\text{"text"}|\theta_d) + 0.5*0.1] \\ \times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9] \\ \times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9] \\ \times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9]$$

...

What if we increase $p(\theta_B)$?

$$\times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9]$$

What's the optimal solution now? $p(\text{"the"}|\theta_d) > 0.1$? or $p(\text{"the"}|\theta_d) < 0.1$?

Behavior 2: high frequency words get higher $p(w|\theta_d)$

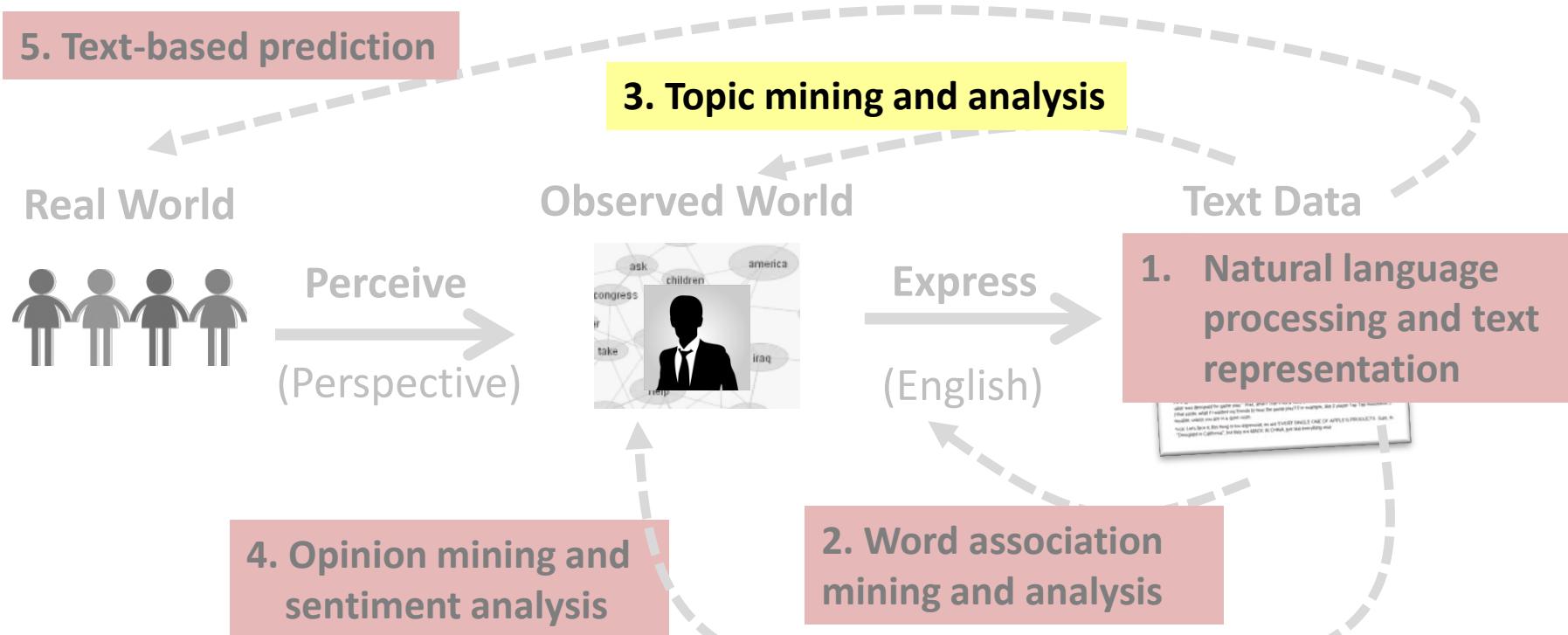
Summary

- General behavior of a mixture model:
 - Every component model attempts to assign high probabilities to highly frequent words in the data (to “collaboratively maximize likelihood”)
 - Different component models tend to “bet” high probabilities on different words (to avoid “competition” or “waste of probability”)
 - The probability of choosing each component “regulates” the collaboration/competition between the component models
- Fixing one component to a background word distribution (i.e., background language model):
 - Helps “get rid of background words” in other component
 - Is an example of imposing a prior on the model parameters (prior = one model must be exactly the same as the background LM)

Probabilistic Topic Models: Expectation-Maximization Algorithm

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Probabilistic Topic Models: Expectation-Maximization (EM) Algorithm

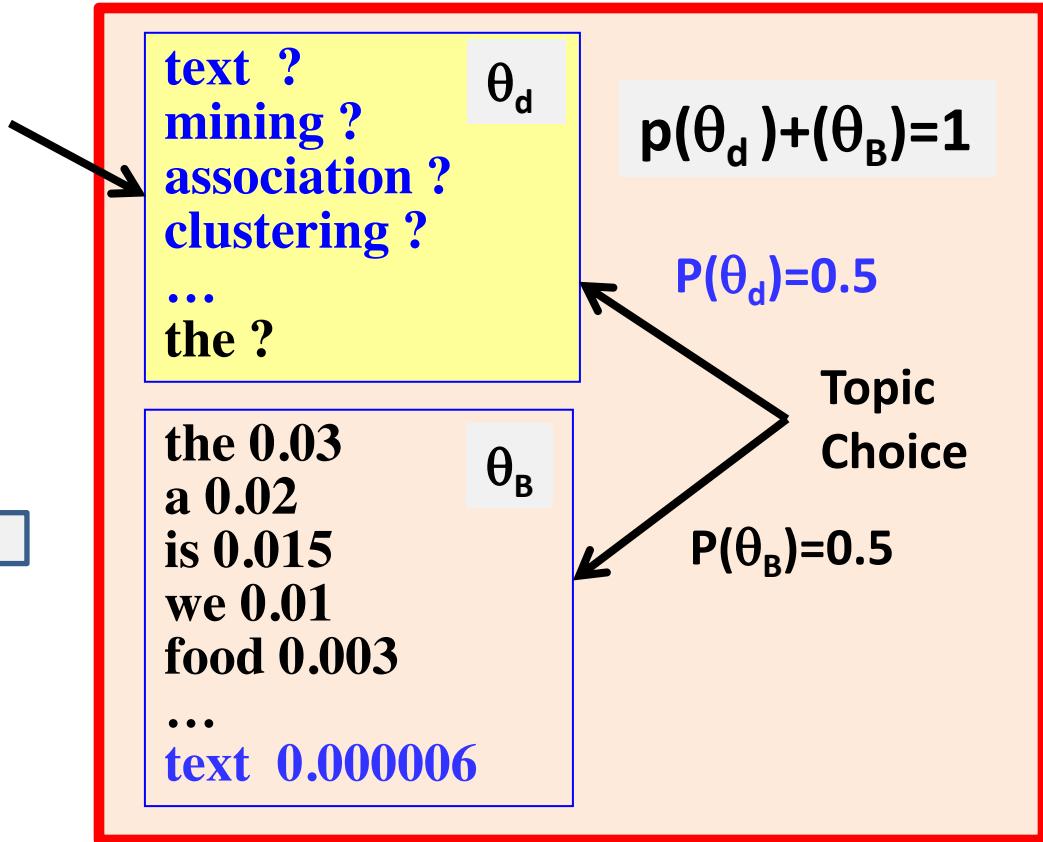
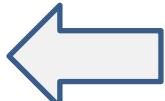


Estimation of One Topic: $P(w | \theta_d)$

How to set θ_d to maximize $p(d | \Lambda)$?
(all other parameters are known)

d

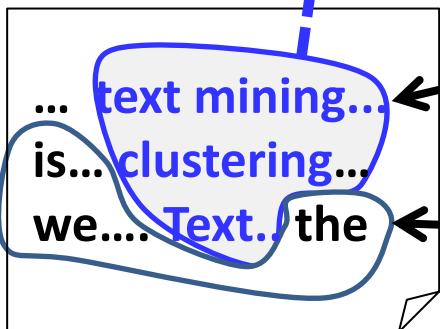
... text mining...
is... clustering...
we.... Text.. the



If we know which word is from which distribution...

$$p(w_i | \theta_d) = \frac{c(w_i, d')}{\sum_{w' \in V} c(w', d')}$$

d
d'



$P(w | \theta_d)$

$p(w | \theta_B)$

text ?
mining ?
association ?
clustering ?
...
the ?

θ_d

$$p(\theta_d) + (\theta_B) = 1$$

$P(\theta_d) = 0.5$

Topic
Choice

$P(\theta_B) = 0.5$

the 0.03
a 0.02
is 0.015
we 0.01
food 0.003
...
text 0.000006

θ_B

Given all the parameters, infer the distribution a word is from...

Is “text” more likely from θ_d or θ_B ?

From θ_d ($z=0$)?

$$p(\theta_d)p(\text{"text"} | \theta_d)$$

From θ_B ($z=1$)?

$$p(\theta_B)p(\text{"text"} | \theta_B)$$

$$p(z = 0 | w = \text{"text"}) =$$

$$\frac{p(\theta_d)p(\text{"text"} | \theta_d)}{p(\theta_d)p(\text{"text"} | \theta_d) + p(\theta_B)p(\text{"text"} | \theta_B)}$$

$$P(w | \theta_d)$$

$$p(w | \theta_B)$$

text 0.04
mining 0.035
association 0.03
clustering 0.005
...
the 0.000001

the 0.03
a 0.02
is 0.015
we 0.01
food 0.003
...
text 0.000006

$$p(\theta_d) + p(\theta_B) = 1$$

$$P(\theta_d) = 0.5$$

Topic
Choice

$$P(\theta_B) = 0.5$$

The Expectation-Maximization (EM) Algorithm

Hidden Variable:

$$z \in \{0, 1\}$$

z

the _____ 1

paper _____ 1

presents _____ 1

a _____ 1

text _____ 0

mining _____ 0

algorithm _____ 0

for _____ 1

clustering _____ 0

... _____ ...

Initialize $p(w|\theta_d)$ with random values.

Then iteratively improve it using E-step & M-step.

Stop when likelihood doesn't change.

$$p^{(n)}(z=0 | w) = \frac{p(\theta_d)p^{(n)}(w | \theta_d)}{p(\theta_d)p^{(n)}(w | \theta_d) + p(\theta_B)p(w | \theta_B)}$$

E-step

How likely w is from θ_d

$$p^{(n+1)}(w | \theta_d) = \frac{c(w, d)p^{(n)}(z=0 | w)}{\sum_{w' \in V} c(w', d)p^{(n)}(z=0 | w')}$$

M-step

EM Computation in Action

E-step

$$p^{(n)}(z=0 | w) = \frac{p(\theta_d)p^{(n)}(w | \theta_d)}{p(\theta_d)p^{(n)}(w | \theta_d) + p(\theta_B)p(w | \theta_B)}$$

M-step

$$p^{(n+1)}(w | \theta_d) = \frac{c(w, d)p^{(n)}(z=0 | w)}{\sum_{w' \in V} c(w', d)p^{(n)}(z=0 | w')}$$

Assume

$$p(\theta_d) = p(\theta_B) = 0.5$$

and $p(w | \theta_B)$ is known

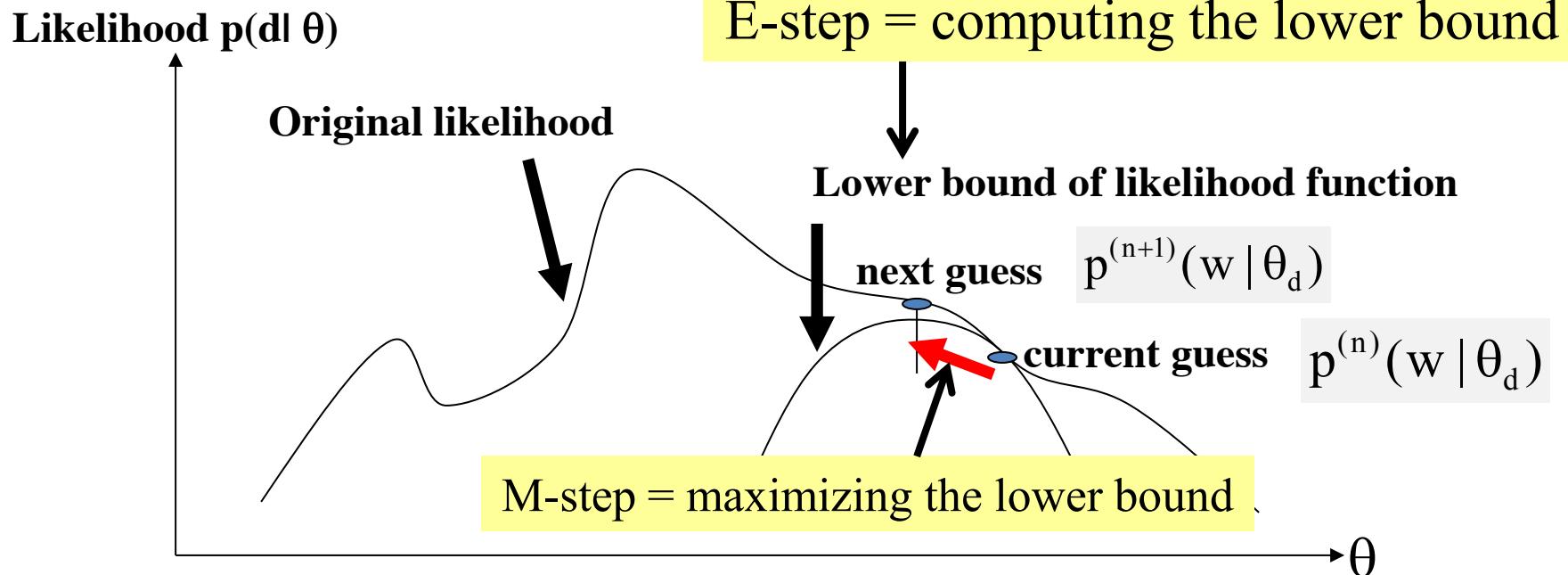
Word	#	$p(w \theta_B)$	Iteration 1		Iteration 2		Iteration 3	
			$P(w \theta)$	$p(z=0 w)$	$P(w \theta)$	$P(z=0 w)$	$P(w \theta)$	$P(z=0 w)$
The	4	0.5	0.25	0.33	0.20	0.29	0.18	0.26
Paper	2	0.3	0.25	0.45	0.14	0.32	0.10	0.25
Text	4	0.1	0.25	0.71	0.44	0.81	0.50	0.93
Mining	2	0.1	0.25	0.71	0.22	0.69	0.22	0.69
Log-Likelihood			-16.96		-16.13		-16.02	

Likelihood increasing



“By products”: Are they also useful?

EM As Hill-Climbing → Converge to Local Maximum



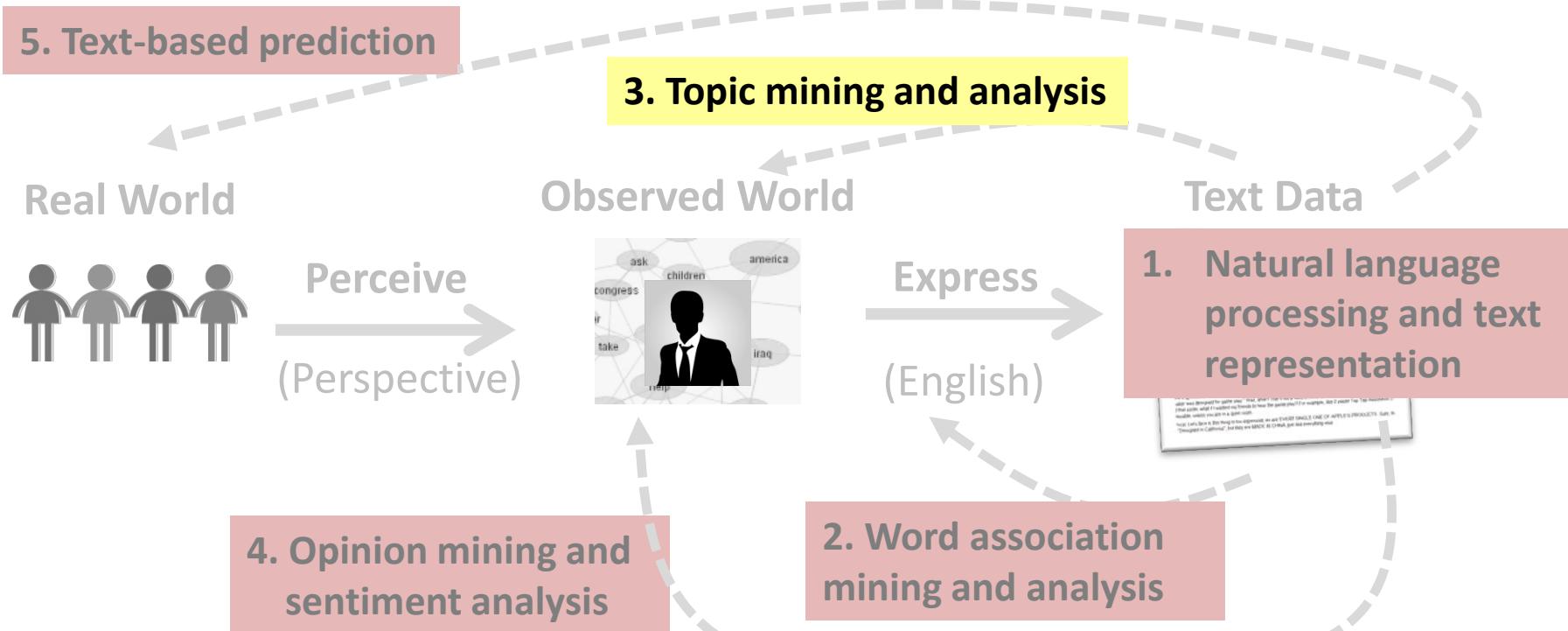
Summary

- Expectation-Maximization (EM) algorithm
 - General algorithm for computing ML estimate of mixture models
 - Hill-climbing, so can only converge to a local maximum (depending on initial points)
- E-step: “augment” data by predicting values of useful hidden variables
- M-step: exploit the “augmented data” to improve estimate of parameters (“improve” is guaranteed in terms of likelihood)
- “Data augmentation” is probabilistic → Split counts of events probabilistically

Probabilistic Topic Models: Expectation-Maximization Algorithm

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Probabilistic Topic Models: Expectation-Maximization (EM) Algorithm

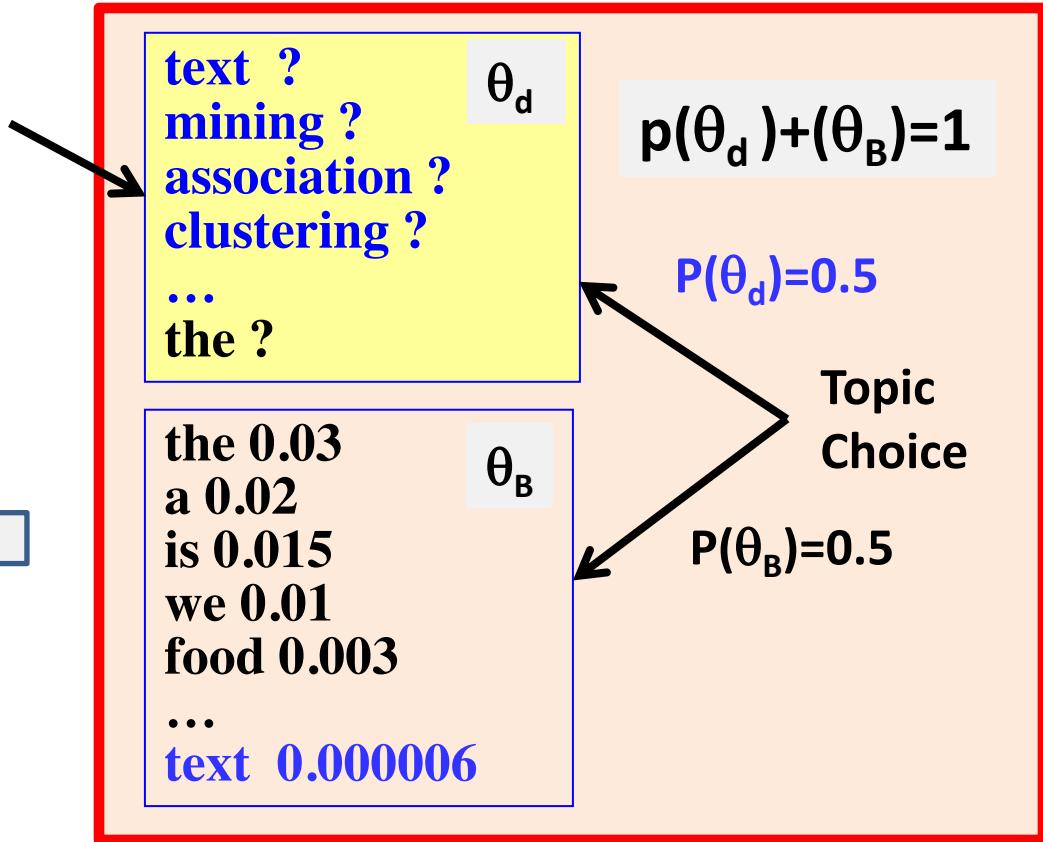
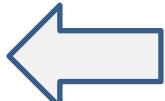


Estimation of One Topic: $P(w | \theta_d)$

How to set θ_d to maximize $p(d | \Lambda)$?
(all other parameters are known)

d

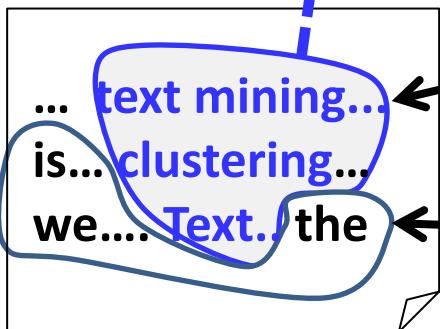
... text mining...
is... clustering...
we.... Text.. the



If we know which word is from which distribution...

$$p(w_i | \theta_d) = \frac{c(w_i, d')}{\sum_{w' \in V} c(w', d')}$$

d
d'



$p(w | \theta_d)$

$p(w | \theta_B)$

text ?
mining ?
association ?
clustering ?
...
the ?

θ_d

$p(\theta_d) + (\theta_B) = 1$

$P(\theta_d) = 0.5$

Topic
Choice

$P(\theta_B) = 0.5$

the 0.03
a 0.02
is 0.015
we 0.01
food 0.003
...
text 0.000006

θ_B

Given all the parameters, infer the distribution a word is from...

Is “text” more likely from θ_d or θ_B ?

From θ_d ($z=0$)?

$$p(\theta_d)p(\text{"text"} | \theta_d)$$

From θ_B ($z=1$)?

$$p(\theta_B)p(\text{"text"} | \theta_B)$$

$$p(z = 0 | w = \text{"text"}) =$$

$$\frac{p(\theta_d)p(\text{"text"} | \theta_d)}{p(\theta_d)p(\text{"text"} | \theta_d) + p(\theta_B)p(\text{"text"} | \theta_B)}$$

$$P(w | \theta_d)$$

$$p(w | \theta_B)$$

text 0.04
mining 0.035
association 0.03
clustering 0.005
...
the 0.000001

the 0.03
a 0.02
is 0.015
we 0.01
food 0.003
...
text 0.000006

$$p(\theta_d) + p(\theta_B) = 1$$

$$P(\theta_d) = 0.5$$

Topic
Choice

$$P(\theta_B) = 0.5$$

The Expectation-Maximization (EM) Algorithm

Hidden Variable:

$$z \in \{0, 1\}$$

z

the _____ 1

paper _____ 1

presents _____ 1

a _____ 1

text _____ 0

mining _____ 0

algorithm _____ 0

for _____ 1

clustering _____ 0

... _____ ...

Initialize $p(w|\theta_d)$ with random values.

Then iteratively improve it using E-step & M-step.

Stop when likelihood doesn't change.

$$p^{(n)}(z=0 | w) = \frac{p(\theta_d)p^{(n)}(w | \theta_d)}{p(\theta_d)p^{(n)}(w | \theta_d) + p(\theta_B)p(w | \theta_B)}$$

E-step

How likely w is from θ_d

$$p^{(n+1)}(w | \theta_d) = \frac{c(w, d)p^{(n)}(z=0 | w)}{\sum_{w' \in V} c(w', d)p^{(n)}(z=0 | w')}$$

M-step

EM Computation in Action

E-step

$$p^{(n)}(z=0 | w) = \frac{p(\theta_d)p^{(n)}(w | \theta_d)}{p(\theta_d)p^{(n)}(w | \theta_d) + p(\theta_B)p(w | \theta_B)}$$

M-step

$$p^{(n+1)}(w | \theta_d) = \frac{c(w, d)p^{(n)}(z=0 | w)}{\sum_{w' \in V} c(w', d)p^{(n)}(z=0 | w')}$$

Assume

$$p(\theta_d) = p(\theta_B) = 0.5$$

and $p(w | \theta_B)$ is known

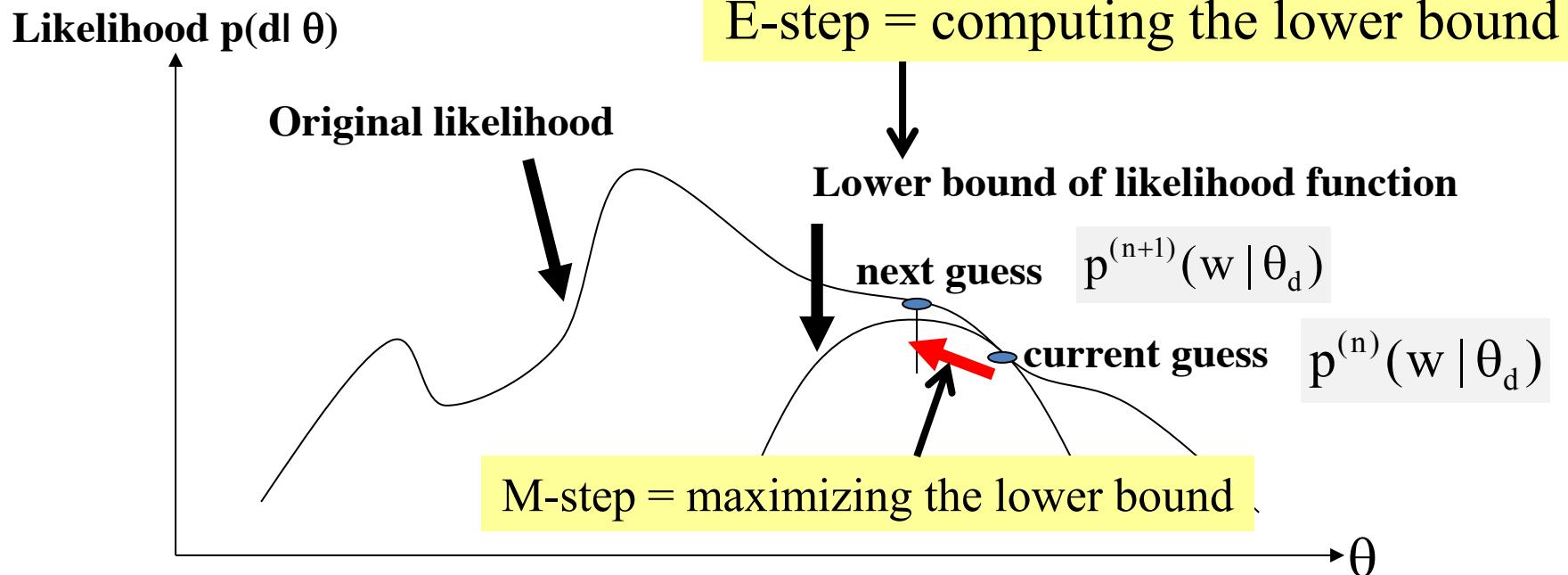
Word	#	$p(w \theta_B)$	Iteration 1		Iteration 2		Iteration 3	
			$P(w \theta)$	$p(z=0 w)$	$P(w \theta)$	$P(z=0 w)$	$P(w \theta)$	$P(z=0 w)$
The	4	0.5	0.25	0.33	0.20	0.29	0.18	0.26
Paper	2	0.3	0.25	0.45	0.14	0.32	0.10	0.25
Text	4	0.1	0.25	0.71	0.44	0.81	0.50	0.93
Mining	2	0.1	0.25	0.71	0.22	0.69	0.22	0.69
Log-Likelihood			-16.96		-16.13		-16.02	

Likelihood increasing



“By products”: Are they also useful?

EM As Hill-Climbing → Converge to Local Maximum



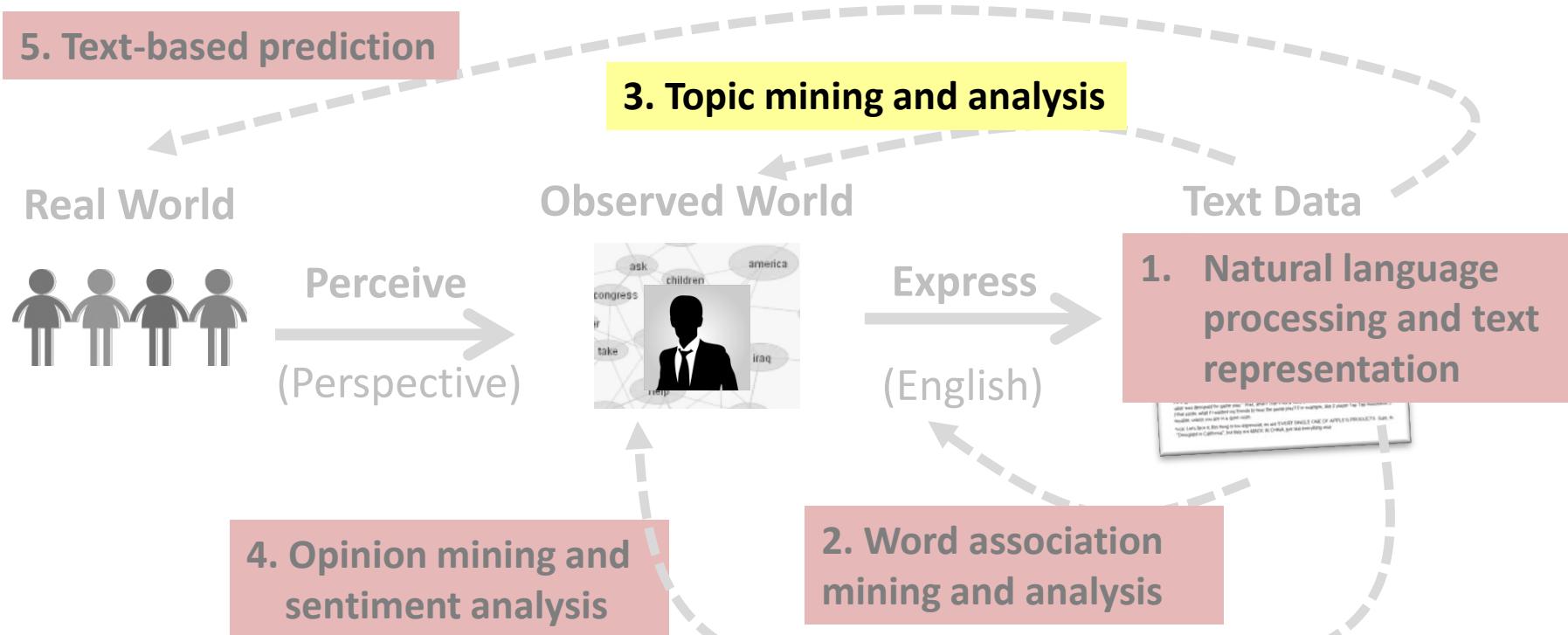
Summary

- Expectation-Maximization (EM) algorithm
 - General algorithm for computing ML estimate of mixture models
 - Hill-climbing, so can only converge to a local maximum (depending on initial points)
- E-step: “augment” data by predicting values of useful hidden variables
- M-step: exploit the “augmented data” to improve estimate of parameters (“improve” is guaranteed in terms of likelihood)
- “Data augmentation” is probabilistic → Split counts of events probabilistically

Probabilistic Topic Models: Expectation-Maximization Algorithm

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Probabilistic Topic Models: Expectation-Maximization (EM) Algorithm

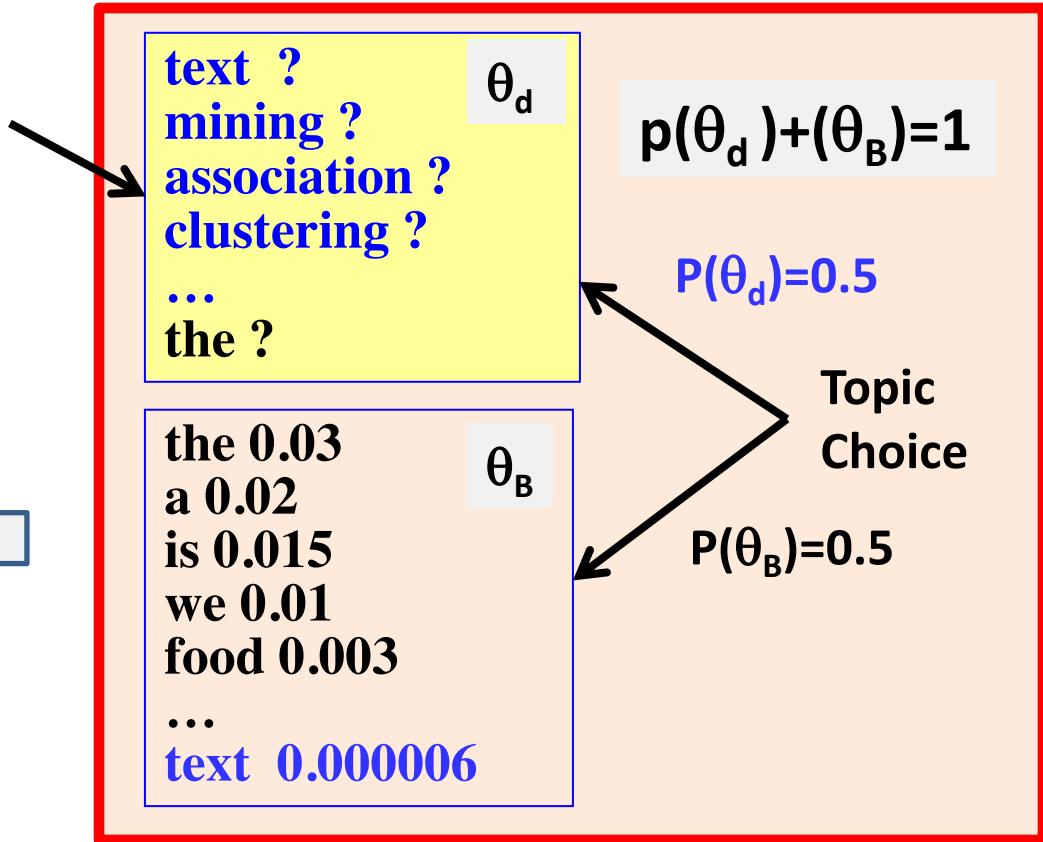
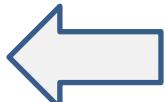


Estimation of One Topic: $P(w | \theta_d)$

How to set θ_d to maximize $p(d | \Lambda)$?
(all other parameters are known)

d

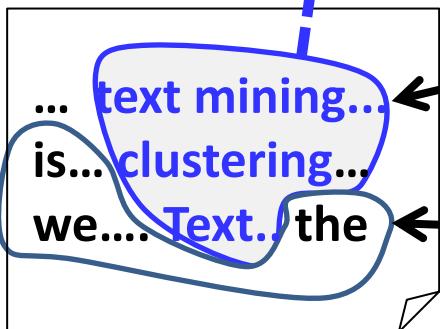
... text mining...
is... clustering...
we.... Text.. the



If we know which word is from which distribution...

$$p(w_i | \theta_d) = \frac{c(w_i, d')}{\sum_{w' \in V} c(w', d')}$$

d
d'



$p(w | \theta_d)$

$p(w | \theta_B)$

text ?
mining ?
association ?
clustering ?
...
the ?

θ_d

$$p(\theta_d) + (\theta_B) = 1$$

$P(\theta_d) = 0.5$

Topic
Choice

the 0.03
a 0.02
is 0.015
we 0.01
food 0.003
...
text 0.000006

θ_B

$P(\theta_B) = 0.5$

Given all the parameters, infer the distribution a word is from...

Is “text” more likely from θ_d or θ_B ?

From θ_d ($z=0$)?

$$p(\theta_d)p(\text{"text"} | \theta_d)$$

From θ_B ($z=1$)?

$$p(\theta_B)p(\text{"text"} | \theta_B)$$

$$p(z = 0 | w = \text{"text"}) =$$

$$\frac{p(\theta_d)p(\text{"text"} | \theta_d)}{p(\theta_d)p(\text{"text"} | \theta_d) + p(\theta_B)p(\text{"text"} | \theta_B)}$$

$$P(w | \theta_d)$$

$$p(w | \theta_B)$$

text 0.04
mining 0.035
association 0.03
clustering 0.005
...
the 0.000001

the 0.03
a 0.02
is 0.015
we 0.01
food 0.003
...
text 0.000006

$$p(\theta_d) + p(\theta_B) = 1$$

$$P(\theta_d) = 0.5$$

Topic
Choice

$$P(\theta_B) = 0.5$$

The Expectation-Maximization (EM) Algorithm

Hidden Variable:

$$z \in \{0, 1\}$$

z

the _____ 1

paper _____ 1

presents _____ 1

a _____ 1

text _____ 0

mining _____ 0

algorithm _____ 0

for _____ 1

clustering _____ 0

... _____ ...

Initialize $p(w|\theta_d)$ with random values.

Then iteratively improve it using E-step & M-step.

Stop when likelihood doesn't change.

$$p^{(n)}(z=0 | w) = \frac{p(\theta_d)p^{(n)}(w | \theta_d)}{p(\theta_d)p^{(n)}(w | \theta_d) + p(\theta_B)p(w | \theta_B)}$$

E-step

How likely w is from θ_d

$$p^{(n+1)}(w | \theta_d) = \frac{c(w, d)p^{(n)}(z=0 | w)}{\sum_{w' \in V} c(w', d)p^{(n)}(z=0 | w')}$$

M-step

EM Computation in Action

E-step

$$p^{(n)}(z=0 | w) = \frac{p(\theta_d)p^{(n)}(w | \theta_d)}{p(\theta_d)p^{(n)}(w | \theta_d) + p(\theta_B)p(w | \theta_B)}$$

M-step

$$p^{(n+1)}(w | \theta_d) = \frac{c(w, d)p^{(n)}(z=0 | w)}{\sum_{w' \in V} c(w', d)p^{(n)}(z=0 | w')}$$

Assume

$$p(\theta_d) = p(\theta_B) = 0.5$$

and $p(w | \theta_B)$ is known

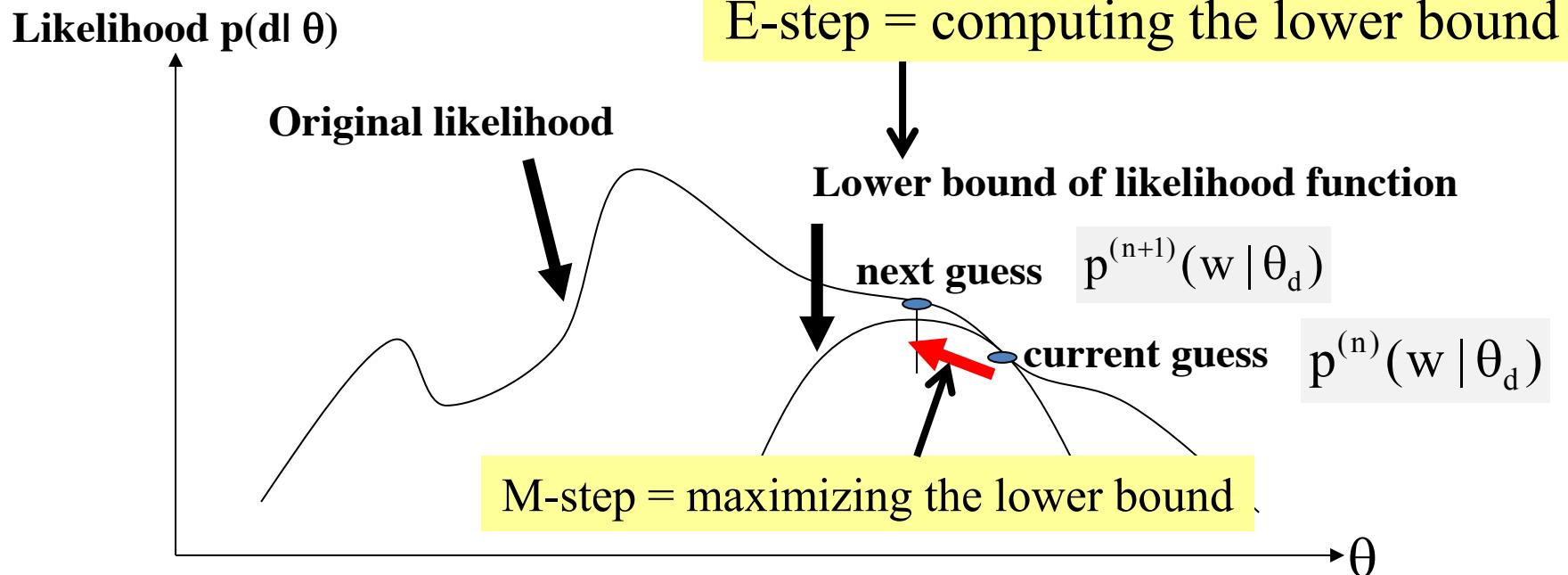
Word	#	$p(w \theta_B)$	Iteration 1		Iteration 2		Iteration 3	
			$P(w \theta)$	$p(z=0 w)$	$P(w \theta)$	$P(z=0 w)$	$P(w \theta)$	$P(z=0 w)$
The	4	0.5	0.25	0.33	0.20	0.29	0.18	0.26
Paper	2	0.3	0.25	0.45	0.14	0.32	0.10	0.25
Text	4	0.1	0.25	0.71	0.44	0.81	0.50	0.93
Mining	2	0.1	0.25	0.71	0.22	0.69	0.22	0.69
Log-Likelihood			-16.96		-16.13		-16.02	

Likelihood increasing



“By products”: Are they also useful?

EM As Hill-Climbing → Converge to Local Maximum



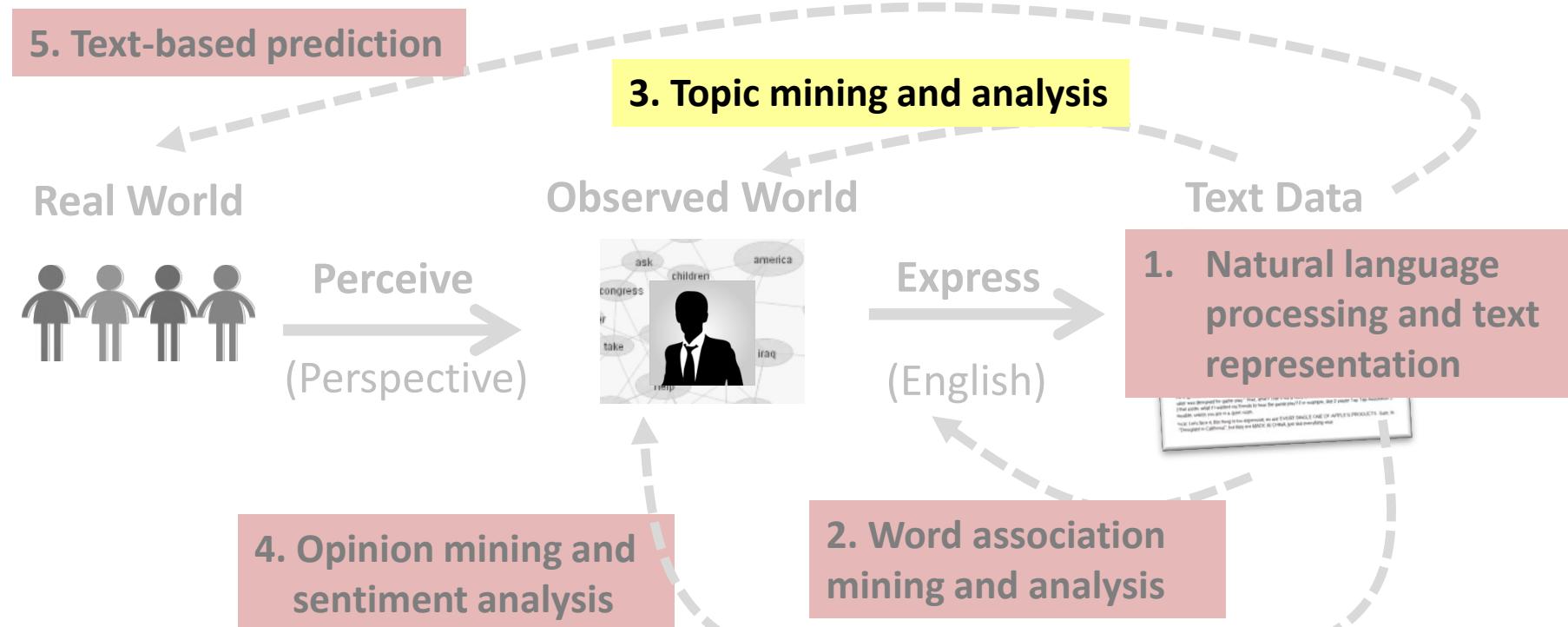
Summary

- Expectation-Maximization (EM) algorithm
 - General algorithm for computing ML estimate of mixture models
 - Hill-climbing, so can only converge to a local maximum (depending on initial points)
- E-step: “augment” data by predicting values of useful hidden variables
- M-step: exploit the “augmented data” to improve estimate of parameters (“improve” is guaranteed in terms of likelihood)
- “Data augmentation” is probabilistic → Split counts of events probabilistically

Probabilistic Latent Semantic Analysis (PLSA)

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Probabilistic Latent Semantic Analysis (PLSA)



Document as a Sample of Mixed Topics

Topic θ_1

government 0.3
response 0.2

...

Topic θ_2

city 0.2
new 0.1
orleans 0.05

...

Topic θ_k

donate 0.1
relief 0.05
help 0.02

...

Background θ_B

the 0.04
a 0.03
...

Blog article about “Hurricane Katrina”

[Criticism of government response to the hurricane primarily consisted of criticism of its response to the approach of the storm and its aftermath, specifically in the delayed response] to the [flooding of New Orleans. ... 80% of the 1.3 million residents of the greater New Orleans metropolitan area evacuated] ...[Over seventy countries pledged monetary donations or other assistance]. ...

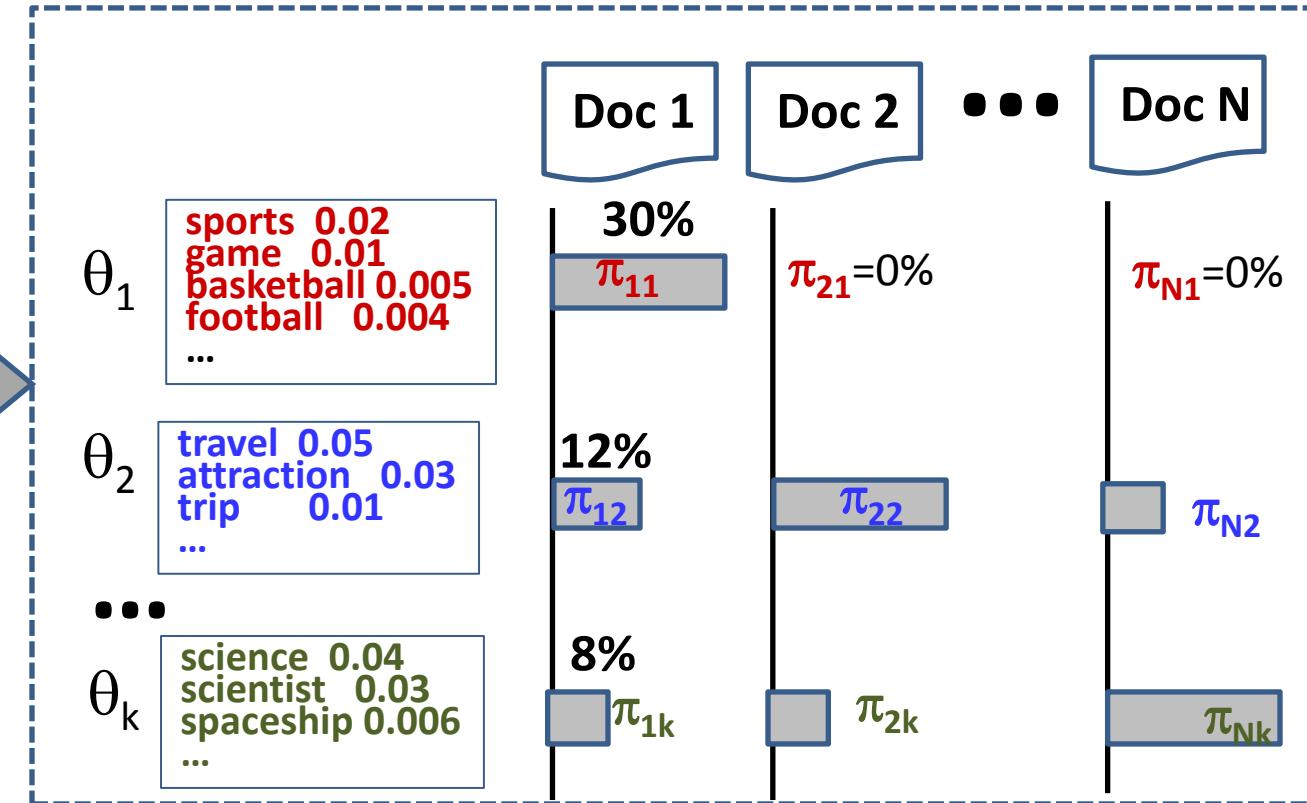
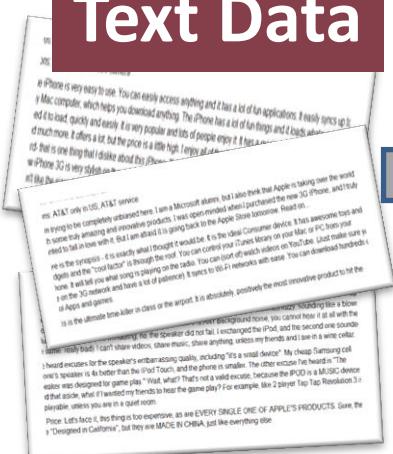
Many applications are possible if we can “decode” the topics in text...

Mining Multiple Topics from Text

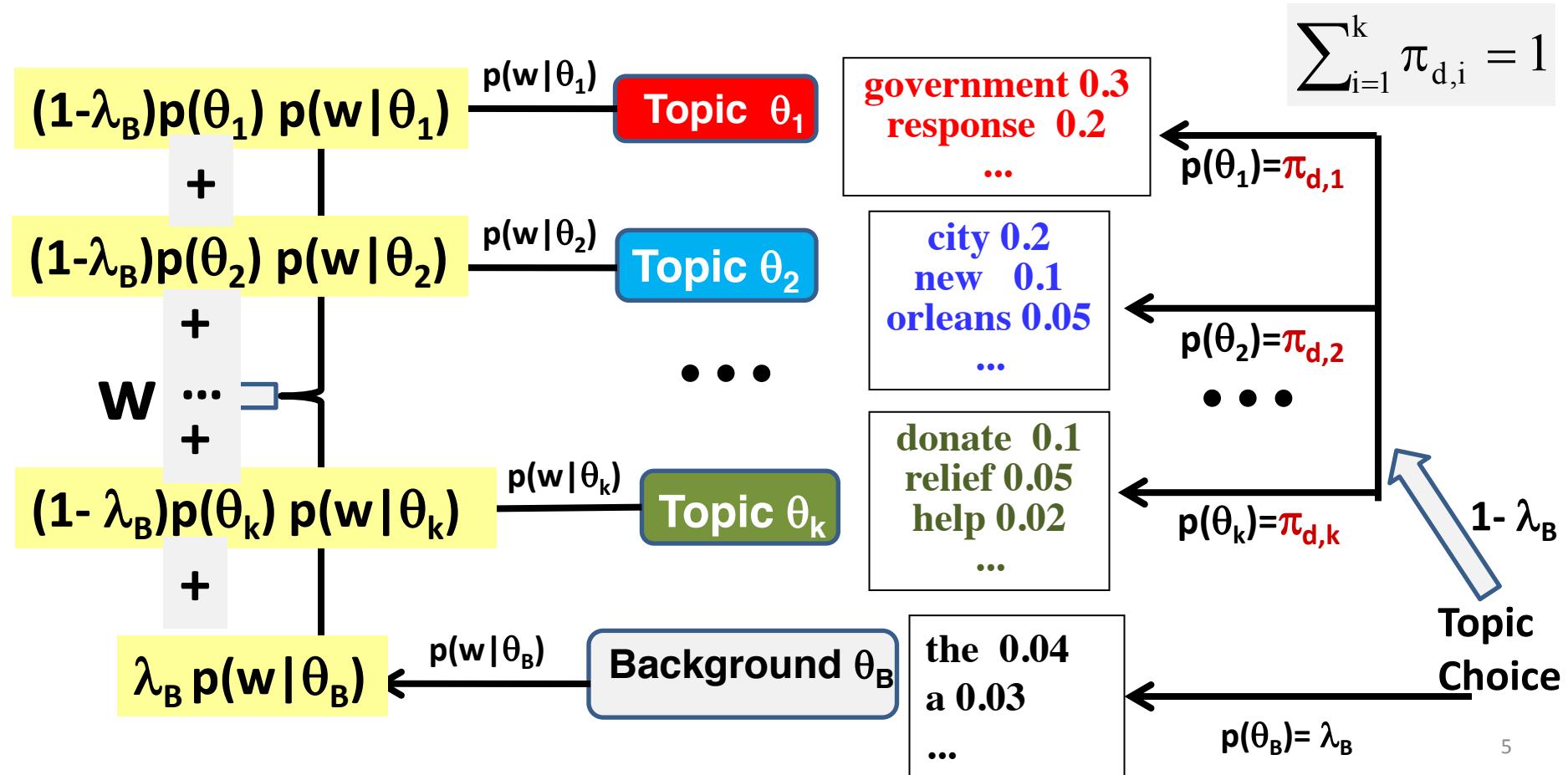
INPUT: C, k, V

OUTPUT: $\{ \theta_1, \dots, \theta_k \}, \{ \pi_{i1}, \dots, \pi_{ik} \}$

Text Data



Generating Text with Multiple Topics: $p(w)=?$



Probabilistic Latent Semantic Analysis (PLSA)

Percentage of
background words
(known)

Background
LM (known)

Coverage of topic θ_j in doc d

Prob. of word w in topic θ_j

$$p_d(w) = \lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)$$

$$\log p(d) = \sum_{w \in V} c(w, d) \log [\lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)]$$

$$\log p(C | \Lambda) = \sum_{d \in C} \sum_{w \in V} c(w, d) \log [\lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)]$$

Unknown Parameters: $\Lambda = (\{\pi_{d,j}\}, \{\theta_j\})$, $j=1, \dots, k$

How many unknown parameters are there in total?

ML Parameter Estimation

$$p_d(w) = \lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)$$

$$\log p(d) = \sum_{w \in V} c(w, d) \log [\lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)]$$

$$\log p(C | \Lambda) = \sum_{d \in C} \sum_{w \in V} c(w, d) \log [\lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)]$$

Constrained Optimization: $\Lambda^* = \arg \max_{\Lambda} p(C | \Lambda)$

$$\forall j \in [1, k], \sum_{i=1}^M p(w_i | \theta_j) = 1$$

$$\forall d \in C, \sum_{j=1}^k \pi_{d,j} = 1$$

EM Algorithm for PLSA: E-Step

Hidden Variable (=topic indicator): $z_{d,w} \in \{B, 1, 2, \dots, k\}$

Probability that **w** in doc **d** is generated from **topic** θ_j

$$p(z_{d,w} = j) = \frac{\pi_{d,j}^{(n)} p^{(n)}(w | \theta_j)}{\sum_{j'=1}^k \pi_{d,j'}^{(n)} p^{(n)}(w | \theta_{j'})}$$

Use of Bayes Rule

$$p(z_{d,w} = B) = \frac{\lambda_B p(w | \theta_B)}{\lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j}^{(n)} p^{(n)}(w | \theta_j)}$$

Probability that **w** in doc **d** is generated from **background** θ_B

EM Algorithm for PLSA: M-Step

Hidden Variable (=topic indicator): $z_{d,w} \in \{B, 1, 2, \dots, k\}$

Re-estimated probability of doc d covering topic θ_j

$$\pi_{d,j}^{(n+1)} = \frac{\sum_{w \in V} c(w, d)(1 - p(z_{d,w} = B))p(z_{d,w} = j)}{\sum_{j'} \sum_{w \in V} c(w, d)(1 - p(z_{d,w} = B))p(z_{d,w} = j')}$$

ML Estimate based on
“allocated” word
counts to topic θ_j

$$p^{(n+1)}(w | \theta_j) = \frac{\sum_{d \in C} c(w, d)(1 - p(z_{d,w} = B))p(z_{d,w} = j)}{\sum_{w' \in V} \sum_{d \in C} c(w', d)(1 - p(z_{d,w'} = B))p(z_{d,w'} = j)}$$

↑
Re-estimated probability of word w for topic θ_j

Computation of the EM Algorithm

- Initialize all unknown parameters randomly
- Repeat until likelihood converges

– E-step $p(z_{d,w} = j) \propto \pi_{d,j}^{(n)} p^{(n)}(w | \theta_j)$ $\sum_{j=1}^k p(z_{d,w} = j) = 1$

$p(z_{d,w} = B) \propto \lambda_B p(w | \theta_B) \leftarrow$ What's the normalizer for this one?

– M-step

$$\pi_{d,j}^{(n+1)} \propto \sum_{w \in V} c(w, d)(1 - p(z_{d,w} = B))p(z_{d,w} = j) \quad \forall d \in C, \sum_{j=1}^k \pi_{d,j} = 1$$
$$p^{(n+1)}(w | \theta_j) \propto \sum_{d \in C} c(w, d)(1 - p(z_{d,w} = B))p(z_{d,w} = j) \quad \forall j \in [1, k], \sum_{w \in V} p(w | \theta_j) = 1$$

In general, accumulate counts, and then normalize

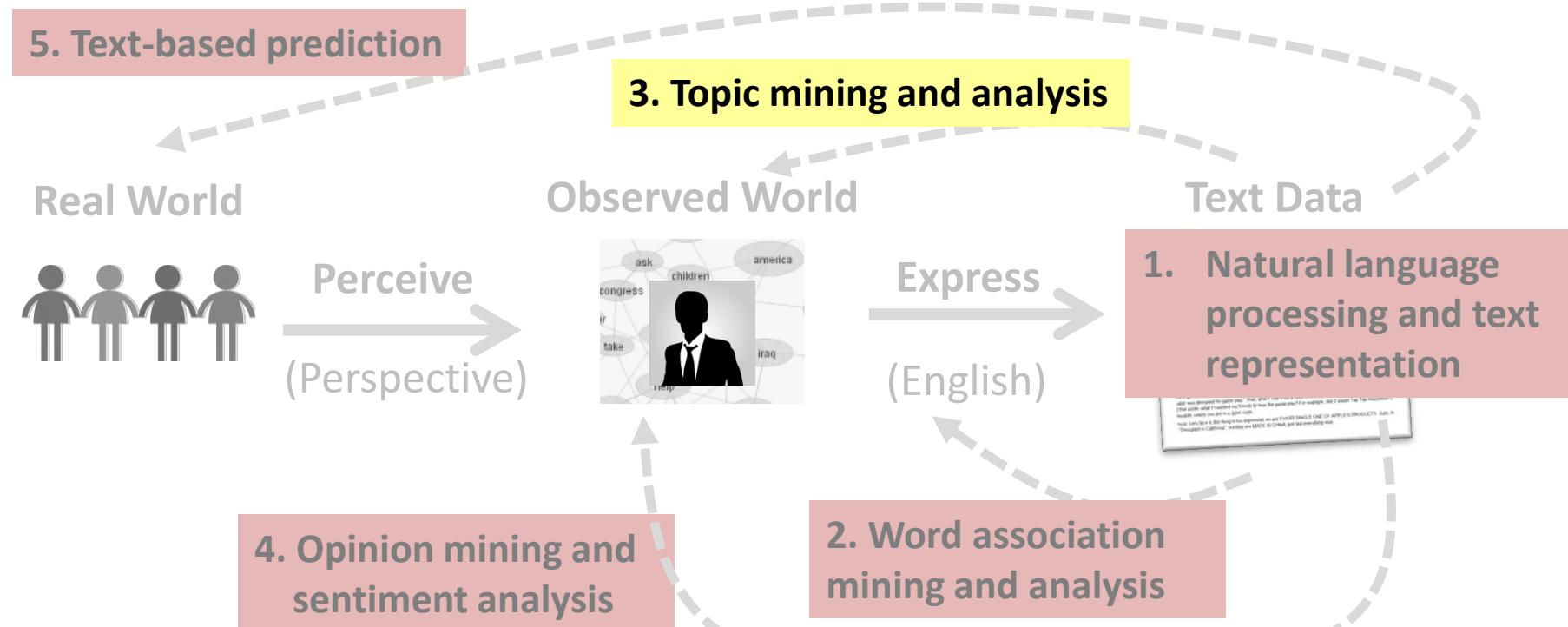
Summary

- PLSA = mixture model with k unigram LMs (k topics)
- Adding a pre-determined background LM helps discover discriminative topics
- ML estimate “discovers” topical knowledge from text data
 - k word distributions (k topics)
 - proportion of each topic in each document
- The output can enable many applications!
 - Clustering of terms and docs (treat each topic as a cluster)
 - Further associate topics with different contexts (e.g., time periods, locations, authors, sources, etc.)

Probabilistic Latent Semantic Analysis (PLSA)

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Probabilistic Latent Semantic Analysis (PLSA)



Document as a Sample of Mixed Topics

Topic θ_1

government 0.3
response 0.2

...

Topic θ_2

city 0.2
new 0.1
orleans 0.05

...

Topic θ_k

donate 0.1
relief 0.05
help 0.02

...

Background θ_B

the 0.04
a 0.03
...

Blog article about “Hurricane Katrina”

[Criticism of government response to the hurricane primarily consisted of criticism of its response to the approach of the storm and its aftermath, specifically in the delayed response] to the [flooding of New Orleans. ... 80% of the 1.3 million residents of the greater New Orleans metropolitan area evacuated] ...[Over seventy countries pledged monetary donations or other assistance]. ...

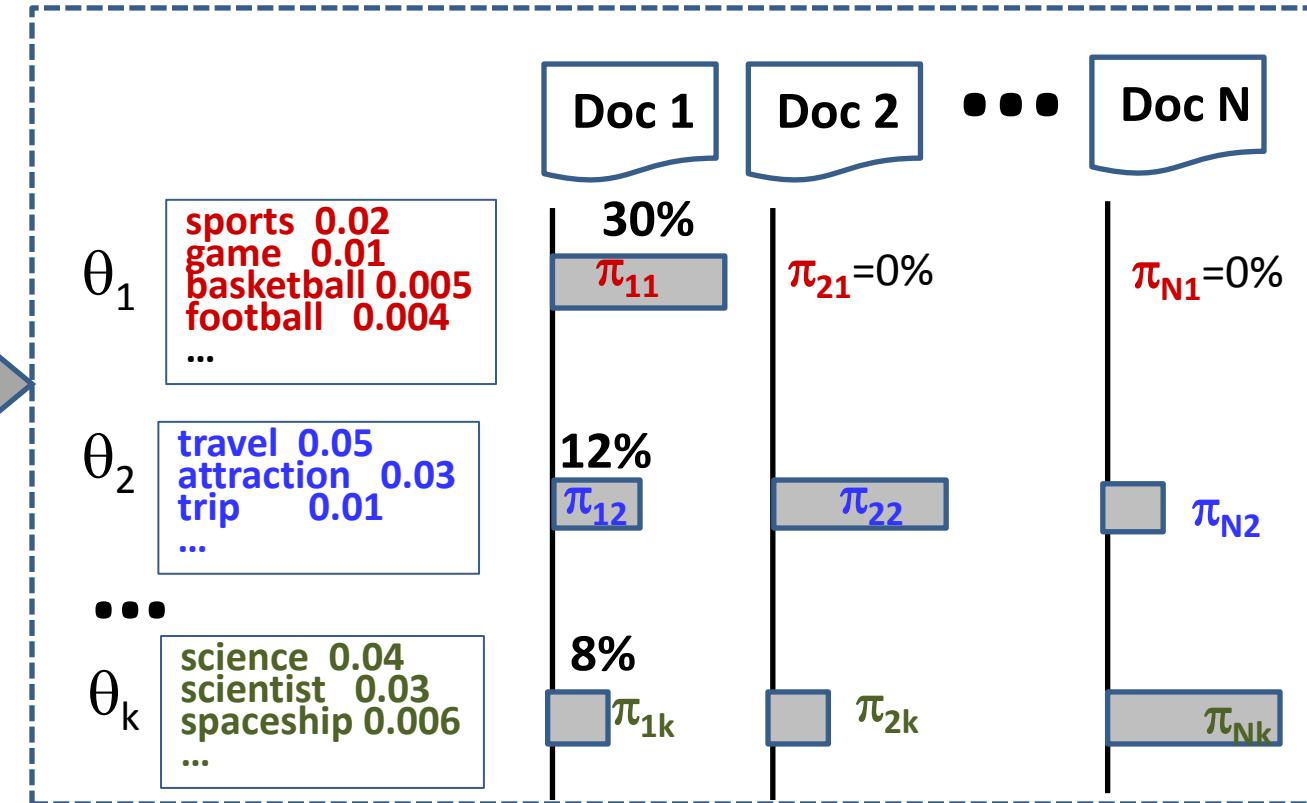
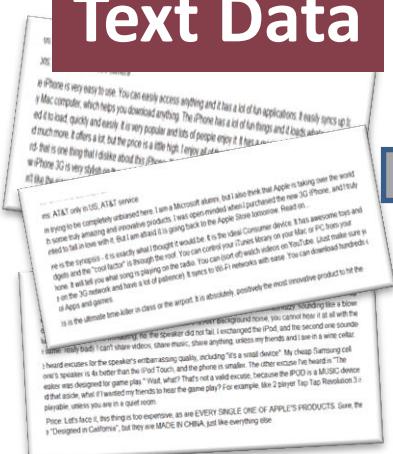
Many applications are possible if we can “decode” the topics in text...

Mining Multiple Topics from Text

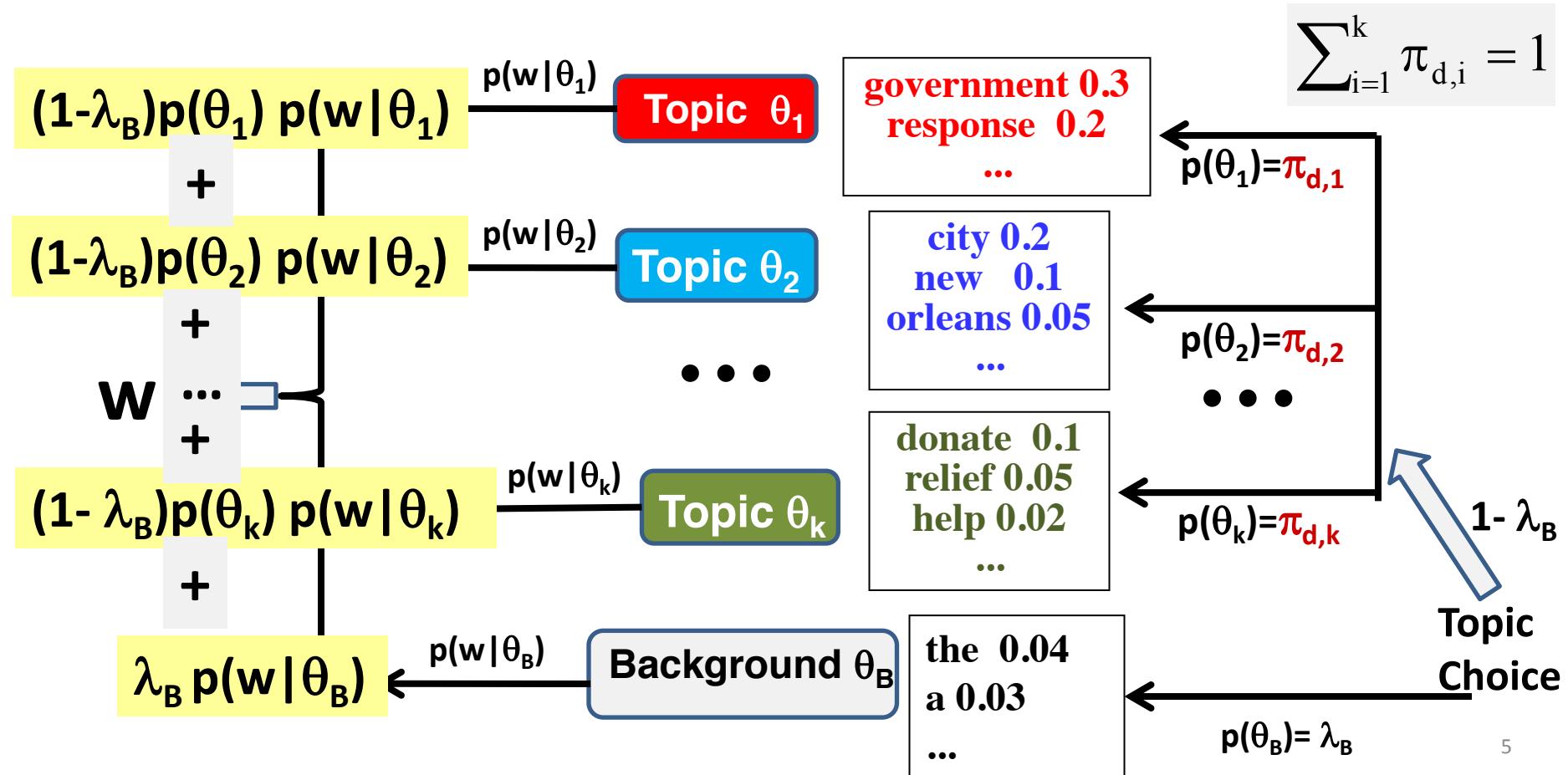
INPUT: C, k, V

OUTPUT: $\{ \theta_1, \dots, \theta_k \}, \{ \pi_{i1}, \dots, \pi_{ik} \}$

Text Data



Generating Text with Multiple Topics: $p(w)=?$



Probabilistic Latent Semantic Analysis (PLSA)

Percentage of
background words
(known)

Background
LM (known)

Coverage of topic θ_j in doc d

Prob. of word w in topic θ_j

$$p_d(w) = \lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)$$

$$\log p(d) = \sum_{w \in V} c(w, d) \log [\lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)]$$

$$\log p(C | \Lambda) = \sum_{d \in C} \sum_{w \in V} c(w, d) \log [\lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)]$$

Unknown Parameters: $\Lambda = (\{\pi_{d,j}\}, \{\theta_j\})$, $j=1, \dots, k$

How many unknown parameters are there in total?

ML Parameter Estimation

$$p_d(w) = \lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)$$

$$\log p(d) = \sum_{w \in V} c(w, d) \log [\lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)]$$

$$\log p(C | \Lambda) = \sum_{d \in C} \sum_{w \in V} c(w, d) \log [\lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)]$$

Constrained Optimization: $\Lambda^* = \arg \max_{\Lambda} p(C | \Lambda)$

$$\forall j \in [1, k], \sum_{i=1}^M p(w_i | \theta_j) = 1$$

$$\forall d \in C, \sum_{j=1}^k \pi_{d,j} = 1$$

EM Algorithm for PLSA: E-Step

Hidden Variable (=topic indicator): $z_{d,w} \in \{B, 1, 2, \dots, k\}$

Probability that **w** in doc **d** is generated from **topic** θ_j

$$p(z_{d,w} = j) = \frac{\pi_{d,j}^{(n)} p^{(n)}(w | \theta_j)}{\sum_{j'=1}^k \pi_{d,j'}^{(n)} p^{(n)}(w | \theta_{j'})}$$

Use of Bayes Rule

$$p(z_{d,w} = B) = \frac{\lambda_B p(w | \theta_B)}{\lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j}^{(n)} p^{(n)}(w | \theta_j)}$$

Probability that **w** in doc **d** is generated from **background** θ_B

EM Algorithm for PLSA: M-Step

Hidden Variable (=topic indicator): $z_{d,w} \in \{B, 1, 2, \dots, k\}$

Re-estimated probability of doc d covering topic θ_j

$$\pi_{d,j}^{(n+1)} = \frac{\sum_{w \in V} c(w, d)(1 - p(z_{d,w} = B))p(z_{d,w} = j)}{\sum_{j'} \sum_{w \in V} c(w, d)(1 - p(z_{d,w} = B))p(z_{d,w} = j')}$$

ML Estimate based on
“allocated” word
counts to topic θ_j

$$p^{(n+1)}(w | \theta_j) = \frac{\sum_{d \in C} c(w, d)(1 - p(z_{d,w} = B))p(z_{d,w} = j)}{\sum_{w' \in V} \sum_{d \in C} c(w', d)(1 - p(z_{d,w'} = B))p(z_{d,w'} = j)}$$

↑
Re-estimated probability of word w for topic θ_j

Computation of the EM Algorithm

- Initialize all unknown parameters randomly
- Repeat until likelihood converges

– E-step $p(z_{d,w} = j) \propto \pi_{d,j}^{(n)} p^{(n)}(w | \theta_j)$ $\sum_{j=1}^k p(z_{d,w} = j) = 1$

$p(z_{d,w} = B) \propto \lambda_B p(w | \theta_B) \leftarrow$ What's the normalizer for this one?

– M-step

$$\pi_{d,j}^{(n+1)} \propto \sum_{w \in V} c(w, d)(1 - p(z_{d,w} = B))p(z_{d,w} = j) \quad \forall d \in C, \sum_{j=1}^k \pi_{d,j} = 1$$
$$p^{(n+1)}(w | \theta_j) \propto \sum_{d \in C} c(w, d)(1 - p(z_{d,w} = B))p(z_{d,w} = j) \quad \forall j \in [1, k], \sum_{w \in V} p(w | \theta_j) = 1$$

In general, accumulate counts, and then normalize

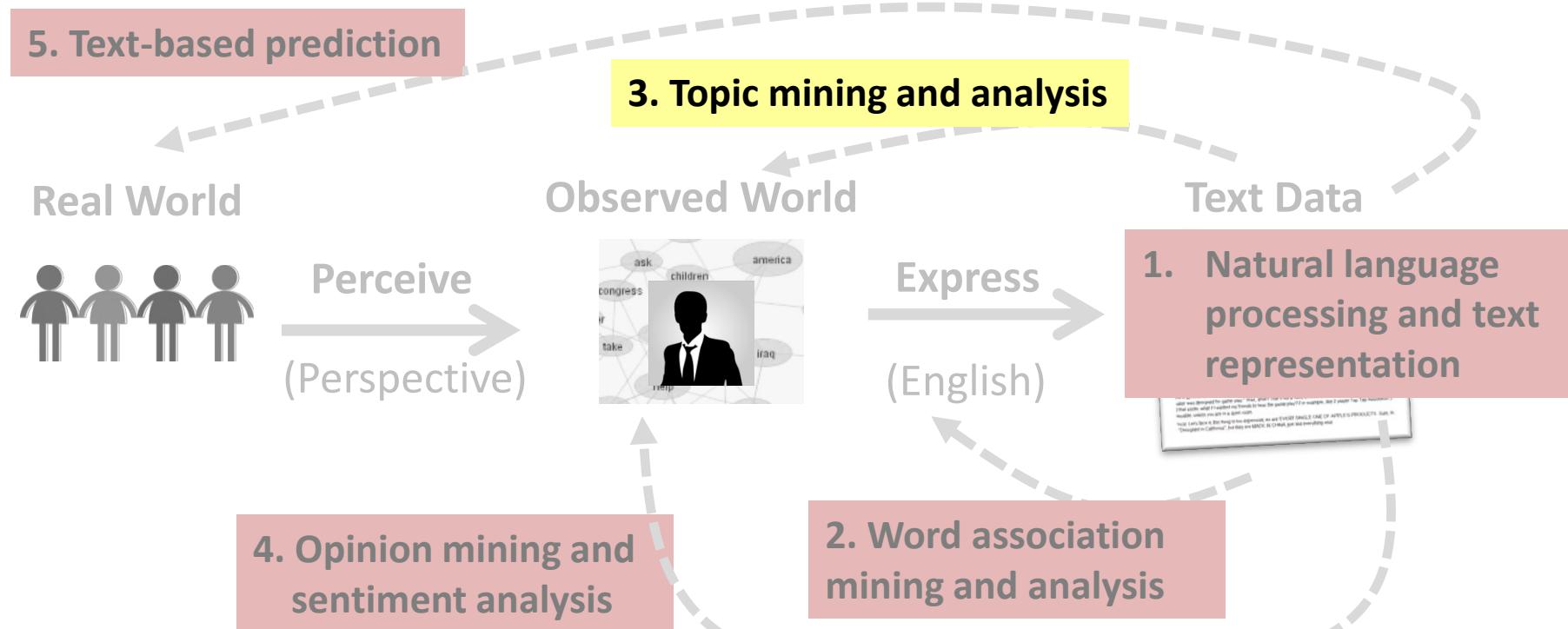
Summary

- PLSA = mixture model with k unigram LMs (k topics)
- Adding a pre-determined background LM helps discover discriminative topics
- ML estimate “discovers” topical knowledge from text data
 - k word distributions (k topics)
 - proportion of each topic in each document
- The output can enable many applications!
 - Clustering of terms and docs (treat each topic as a cluster)
 - Further associate topics with different contexts (e.g., time periods, locations, authors, sources, etc.)

Latent Dirichlet Allocation (LDA)

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Latent Dirichlet Allocation (LDA)



Extensions of PLSA

- PLSA with prior knowledge → User-controlled PLSA
- PLSA as a generative model → Latent Dirichlet Allocation

PLSA with Prior Knowledge

- Users may have expectations about which topics to analyze:
 - We expect to see “retrieval models” as a topic in IR
 - We want to see aspects such as “battery” and “memory” for opinions about a laptop
- Users may have knowledge about what topics are (or are NOT) covered in a document
 - Tags = topics → A doc can only be generated using topics corresponding to the tags assigned to the document
- We can incorporate such knowledge as priors of PLSA model

Maximum a Posteriori (MAP) Estimate

$$\Lambda^* = \arg \max_{\Lambda} p(\Lambda) p(Data | \Lambda)$$

- We may use $p(\Lambda)$ to encode all kinds of preferences and constraints, e.g.,
 - $p(\Lambda) > 0$ if and only if one topic is precisely “background”: $p(w | \theta_B)$
 - $p(\Lambda) > 0$ if and only if for a particular doc d , $\pi_{d,3}=0$ and $\pi_{d,1}=1/2$
 - $p(\Lambda)$ favors a Λ with topics that assign high probabilities to some particular words
- The MAP estimate (with conjugate prior) can be computed using a similar EM algorithm to the ML estimate with smoothing to reflect prior preferences

EM Algorithm with Conjugate Prior on $p(w | \theta_i)$

$$p(z_{d,w} = j) = \frac{\pi_{d,j}^{(n)} p^{(n)}(w | \theta_j)}{\sum_{j'=1}^k \pi_{d,j'}^{(n)} p^{(n)}(w | \theta_{j'})}$$

Prior: $p(w | \theta'_j)$

$$p(z_{d,w} = B) = \frac{\lambda_B p(w | \theta_B)}{\lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j}^{(n)} p^{(n)}(w | \theta_j)}$$

battery 0.5
life 0.5

$$\pi_{d,j}^{(n+1)} = \frac{\sum_{w \in V} c(w, d)(1 - p(z_{d,w} = B)) p(z_{d,w} = j)}{\sum_{j'} \sum_{w \in V} c(w, d)(1 - p(z_{d,w} = B)) p(z_{d,w} = j')}$$

Pseudo counts of w
from prior θ'

$$p^{(n+1)}(w | \theta_j) = \frac{\sum_{d \in C} c(w, d)(1 - p(z_{d,w} = B)) p(z_{d,w} = j)}{\sum_{w' \in V} \sum_{d \in C} c(w', d)(1 - p(z_{d,w'} = B)) p(z_{d,w'} = j')} + \mu$$

What if $\mu=0$? What if $\mu=+\infty$?

Sum of all pseudo counts

We may also set any parameter to a constant (including 0) as needed

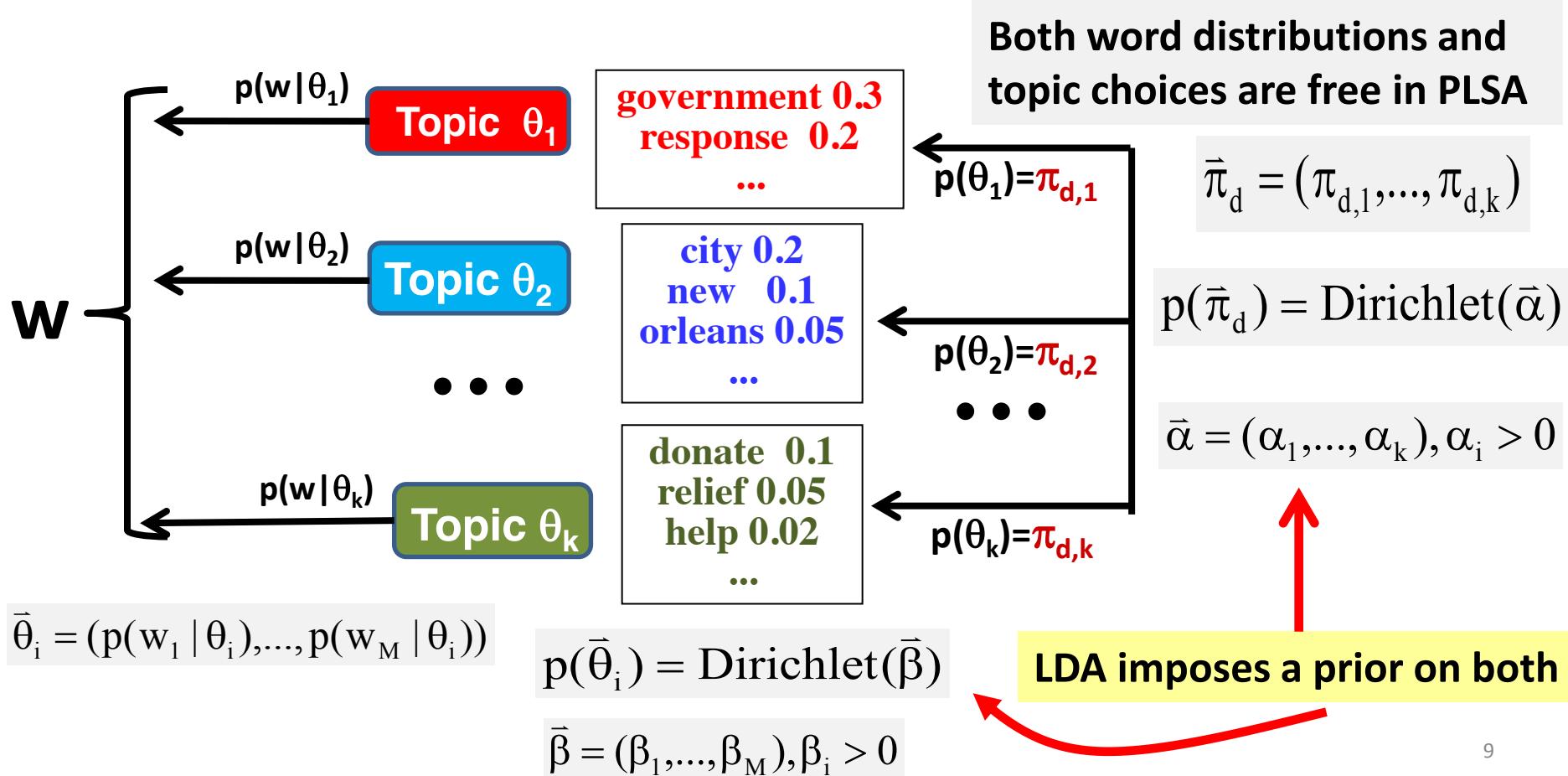
Deficiency of PLSA

- Not a generative model
 - Can't compute probability of a new document
 - Heuristic workaround is possible, though
- Many parameters → high complexity of models
 - Many local maxima
 - Prone to overfitting
- Not necessarily a problem for text mining (only interested in fitting the “training” documents)

Latent Dirichlet Allocation (LDA)

- Make PLSA a generative model by imposing a Dirichlet prior on the model parameters →
 - LDA = Bayesian version of PLSA
 - Parameters are regularized
- Can achieve the same goal as PLSA for text mining purposes
 - Topic coverage and topic word distributions can be inferred using Bayesian inference

PLSA → LDA



Likelihood Functions for PLSA vs. LDA

PLSA

$$p_d(w | \{\theta_j\}, \{\pi_{d,j}\}) = \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)$$

$$\log p(d | \{\theta_j\}, \{\pi_{d,j}\}) = \sum_{w \in V} c(w, d) \log \left[\sum_{j=1}^k \pi_{d,j} p(w | \theta_j) \right]$$

$$\log p(C | \{\theta_j\}, \{\pi_{d,j}\}) = \sum_{d \in C} \log p(d | \{\theta_j\}, \{\pi_{d,j}\})$$

Core assumption
in all topic models

LDA

$$p_d(w | \{\theta_j\}, \{\pi_{d,j}\}) = \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)$$

$$\log p(d | \bar{\alpha}, \{\theta_j\}) = \int \left[\sum_{w \in V} c(w, d) \log \left[\sum_{j=1}^k \pi_{d,j} p(w | \theta_j) \right] \right] p(\bar{\pi}_d | \bar{\alpha}) d\bar{\pi}_d$$

$$\log p(C | \bar{\alpha}, \bar{\beta}) = \int \sum_{d \in C} \log p(d | \bar{\alpha}, \{\theta_j\}) \prod_{j=1}^k p(\theta_j | \bar{\beta}) d\theta_1 \dots d\theta_k$$

PLSA component

Added by LDA

Parameter Estimation and Inferences in LDA

- Parameters can be estimated using ML estimator

$$(\hat{\vec{\alpha}}, \hat{\vec{\beta}}) = \arg \max_{\vec{\alpha}, \vec{\beta}} \log p(C | \vec{\alpha}, \vec{\beta})$$

How many parameters in LDA vs. PLSA?

- However, $\{\theta_j\}$ and $\{\pi_{d,j}\}$ must now be computed using posterior inference
 - Computationally intractable
 - Must resort to approximate inference
 - Many different inference methods are available

Summary of Probabilistic Topic Models

- Probabilistic topic models provide a general principled way of mining and analyzing topics in text with many applications
- Basic task setup:
 - Input: Text data
 - Output: k topics + proportions of these topics covered in each document
- PLSA is the basic topic model, often adequate for most applications
- LDA improves over PLSA by imposing priors
 - Theoretically more appealing
 - Practically, LDA and PLSA perform similarly for many tasks

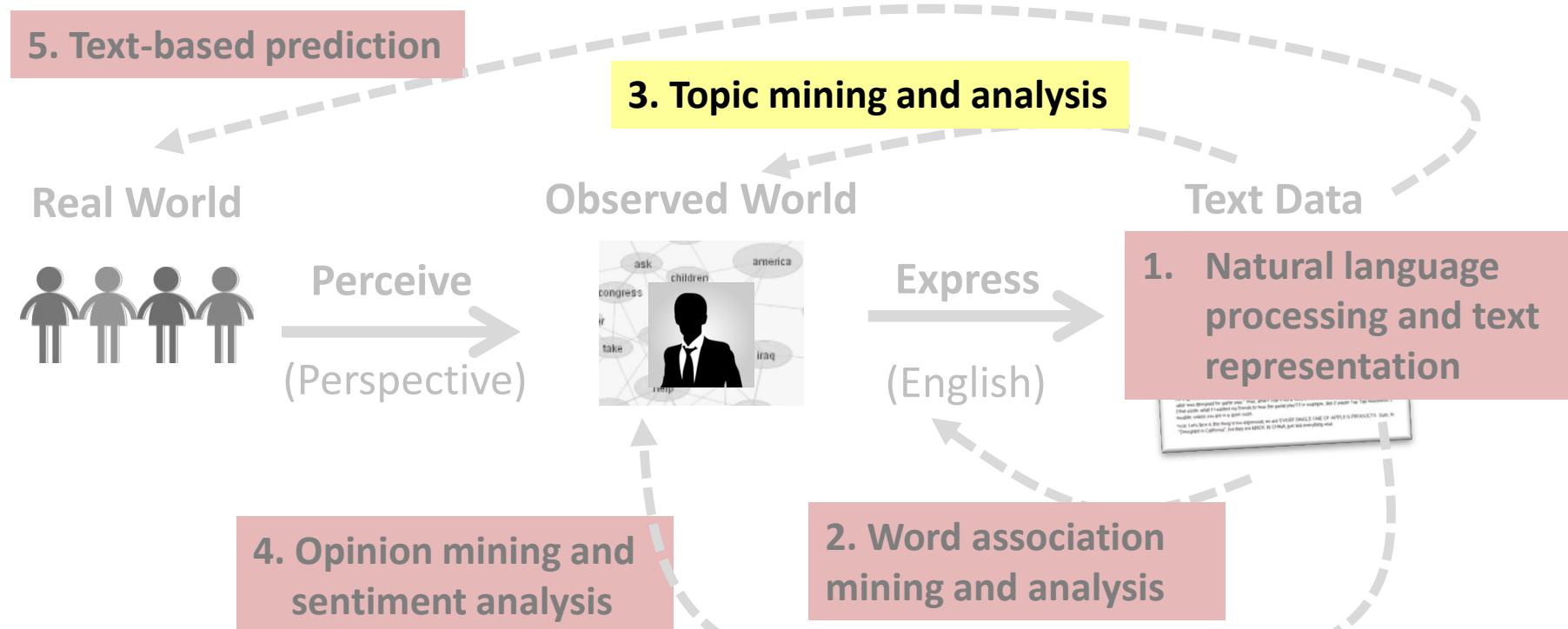
Suggested Readings

- Blei, D. 2012. “Probabilistic Topic Models.” *Communications of the ACM* 55 (4): 77–84. doi: 10.1145/2133806.2133826.
- Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. “Automatic Labeling of Multinomial Topic Models.” *Proceedings of ACM KDD 2007*, pp. 490-499, DOI=10.1145/1281192.1281246.
- Yue Lu, Qiaozhu Mei, and Chengxiang Zhai. 2011. Investigating task performance of probabilistic topic models: an empirical study of PLSA and LDA. *Information Retrieval*, 14, 2 (April 2011), 178-203. DOI=10.1007/s10791-010-9141-9.

Latent Dirichlet Allocation (LDA)

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Latent Dirichlet Allocation (LDA)



Extensions of PLSA

- PLSA with prior knowledge → User-controlled PLSA
- PLSA as a generative model → Latent Dirichlet Allocation

PLSA with Prior Knowledge

- Users may have expectations about which topics to analyze:
 - We expect to see “retrieval models” as a topic in IR
 - We want to see aspects such as “battery” and “memory” for opinions about a laptop
- Users may have knowledge about what topics are (or are NOT) covered in a document
 - Tags = topics → A doc can only be generated using topics corresponding to the tags assigned to the document
- We can incorporate such knowledge as priors of PLSA model

Maximum a Posteriori (MAP) Estimate

$$\Lambda^* = \arg \max_{\Lambda} p(\Lambda) p(Data | \Lambda)$$

- We may use $p(\Lambda)$ to encode all kinds of preferences and constraints, e.g.,
 - $p(\Lambda) > 0$ if and only if one topic is precisely “background”: $p(w | \theta_B)$
 - $p(\Lambda) > 0$ if and only if for a particular doc d , $\pi_{d,3}=0$ and $\pi_{d,1}=1/2$
 - $p(\Lambda)$ favors a Λ with topics that assign high probabilities to some particular words
- The MAP estimate (with conjugate prior) can be computed using a similar EM algorithm to the ML estimate with smoothing to reflect prior preferences

EM Algorithm with Conjugate Prior on $p(w | \theta_i)$

$$p(z_{d,w} = j) = \frac{\pi_{d,j}^{(n)} p^{(n)}(w | \theta_j)}{\sum_{j'=1}^k \pi_{d,j'}^{(n)} p^{(n)}(w | \theta_{j'})}$$

Prior: $p(w | \theta'_j)$

$$p(z_{d,w} = B) = \frac{\lambda_B p(w | \theta_B)}{\lambda_B p(w | \theta_B) + (1 - \lambda_B) \sum_{j=1}^k \pi_{d,j}^{(n)} p^{(n)}(w | \theta_j)}$$

battery 0.5
life 0.5

$$\pi_{d,j}^{(n+1)} = \frac{\sum_{w \in V} c(w, d)(1 - p(z_{d,w} = B)) p(z_{d,w} = j)}{\sum_{j'} \sum_{w \in V} c(w, d)(1 - p(z_{d,w} = B)) p(z_{d,w} = j')}$$

Pseudo counts of w
from prior θ'

$$p^{(n+1)}(w | \theta_j) = \frac{\sum_{d \in C} c(w, d)(1 - p(z_{d,w} = B)) p(z_{d,w} = j)}{\sum_{w' \in V} \sum_{d \in C} c(w', d)(1 - p(z_{d,w'} = B)) p(z_{d,w'} = j')} + \mu$$

What if $\mu=0$? What if $\mu=+\infty$?

Sum of all pseudo counts

We may also set any parameter to a constant (including 0) as needed

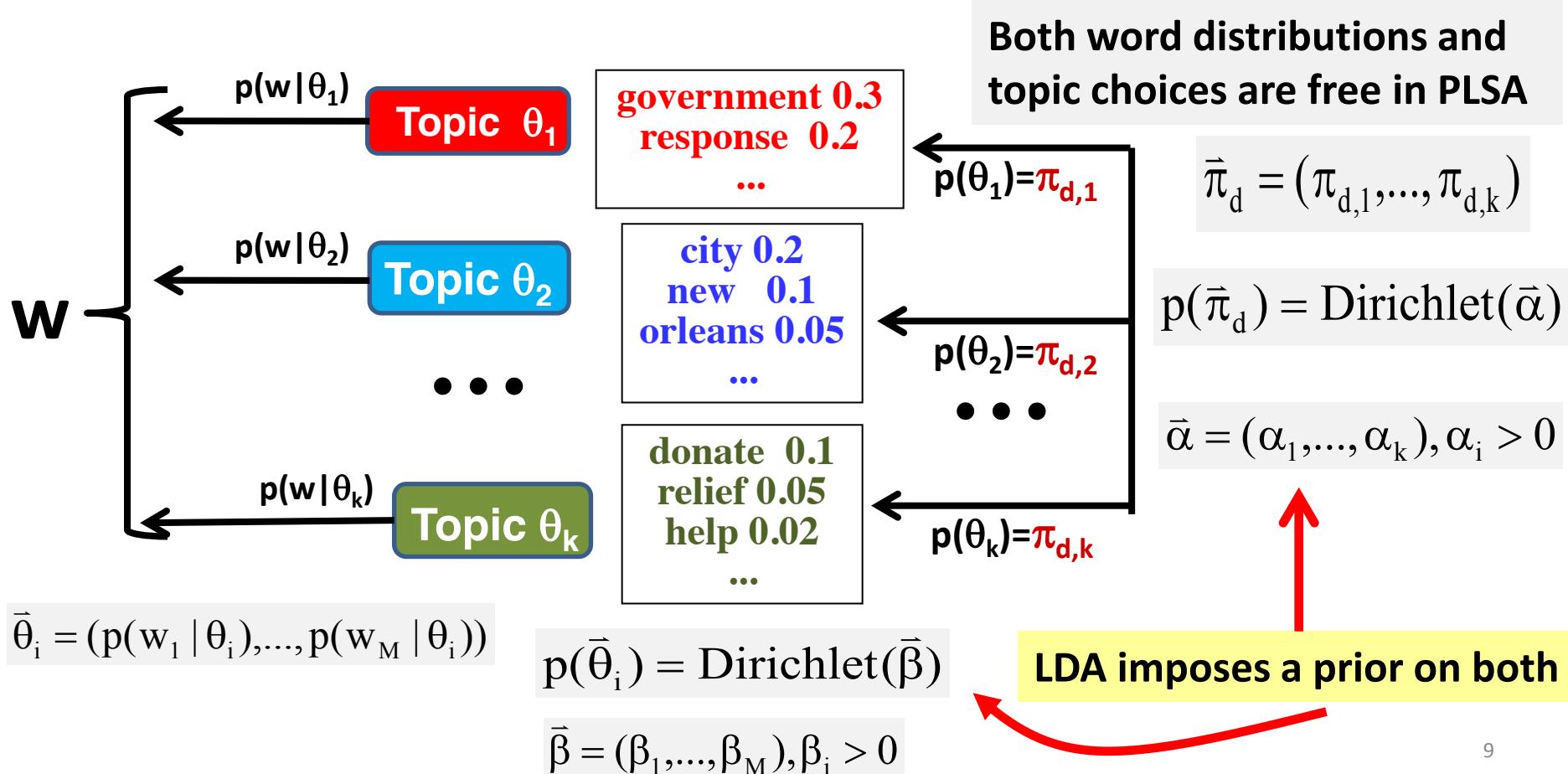
Deficiency of PLSA

- Not a generative model
 - Can't compute probability of a new document
 - Heuristic workaround is possible, though
- Many parameters → high complexity of models
 - Many local maxima
 - Prone to overfitting
- Not necessarily a problem for text mining (only interested in fitting the “training” documents)

Latent Dirichlet Allocation (LDA)

- Make PLSA a generative model by imposing a Dirichlet prior on the model parameters →
 - LDA = Bayesian version of PLSA
 - Parameters are regularized
- Can achieve the same goal as PLSA for text mining purposes
 - Topic coverage and topic word distributions can be inferred using Bayesian inference

PLSA → LDA



Likelihood Functions for PLSA vs. LDA

PLSA

$$p_d(w | \{\theta_j\}, \{\pi_{d,j}\}) = \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)$$

$$\log p(d | \{\theta_j\}, \{\pi_{d,j}\}) = \sum_{w \in V} c(w, d) \log \left[\sum_{j=1}^k \pi_{d,j} p(w | \theta_j) \right]$$

$$\log p(C | \{\theta_j\}, \{\pi_{d,j}\}) = \sum_{d \in C} \log p(d | \{\theta_j\}, \{\pi_{d,j}\})$$

Core assumption
in all topic models

LDA

$$p_d(w | \{\theta_j\}, \{\pi_{d,j}\}) = \sum_{j=1}^k \pi_{d,j} p(w | \theta_j)$$

$$\log p(d | \bar{\alpha}, \{\theta_j\}) = \int \left[\sum_{w \in V} c(w, d) \log \left[\sum_{j=1}^k \pi_{d,j} p(w | \theta_j) \right] \right] p(\bar{\pi}_d | \bar{\alpha}) d\bar{\pi}_d$$

$$\log p(C | \bar{\alpha}, \bar{\beta}) = \int \sum_{d \in C} \log p(d | \bar{\alpha}, \{\theta_j\}) \prod_{j=1}^k p(\theta_j | \bar{\beta}) d\theta_1 \dots d\theta_k$$

PLSA component

Added by LDA

Parameter Estimation and Inferences in LDA

- Parameters can be estimated using ML estimator

$$(\hat{\vec{\alpha}}, \hat{\vec{\beta}}) = \arg \max_{\vec{\alpha}, \vec{\beta}} \log p(C | \vec{\alpha}, \vec{\beta})$$

How many parameters in LDA vs. PLSA?

- However, $\{\theta_j\}$ and $\{\pi_{d,j}\}$ must now be computed using posterior inference
 - Computationally intractable
 - Must resort to approximate inference
 - Many different inference methods are available

Summary of Probabilistic Topic Models

- Probabilistic topic models provide a general principled way of mining and analyzing topics in text with many applications
- Basic task setup:
 - Input: Text data
 - Output: k topics + proportions of these topics covered in each document
- PLSA is the basic topic model, often adequate for most applications
- LDA improves over PLSA by imposing priors
 - Theoretically more appealing
 - Practically, LDA and PLSA perform similarly for many tasks

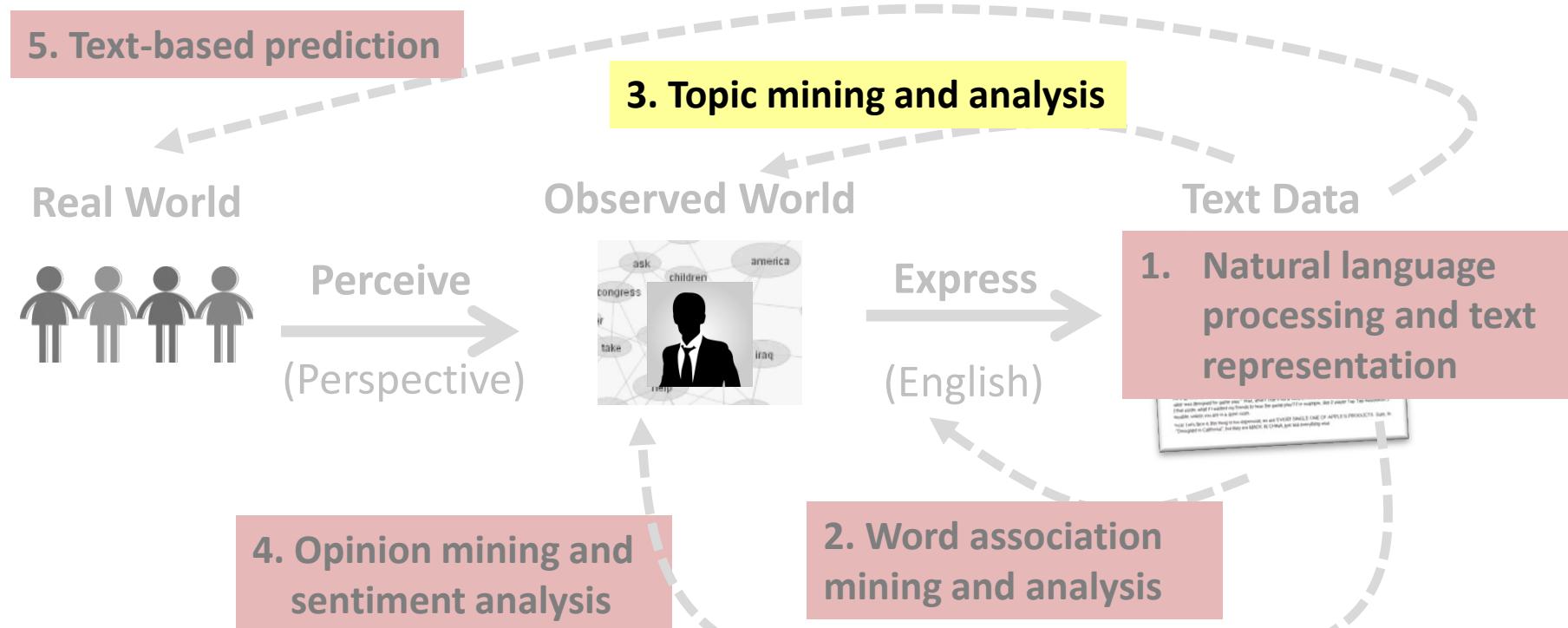
Suggested Readings

- Blei, D. 2012. “Probabilistic Topic Models.” *Communications of the ACM* 55 (4): 77–84. doi: 10.1145/2133806.2133826.
- Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. “Automatic Labeling of Multinomial Topic Models.” *Proceedings of ACM KDD 2007*, pp. 490-499, DOI=10.1145/1281192.1281246.
- Yue Lu, Qiaozhu Mei, and Chengxiang Zhai. 2011. Investigating task performance of probabilistic topic models: an empirical study of PLSA and LDA. *Information Retrieval*, 14, 2 (April 2011), 178-203. DOI=10.1007/s10791-010-9141-9.

Text Clustering: Motivation

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Text Clustering: Motivation

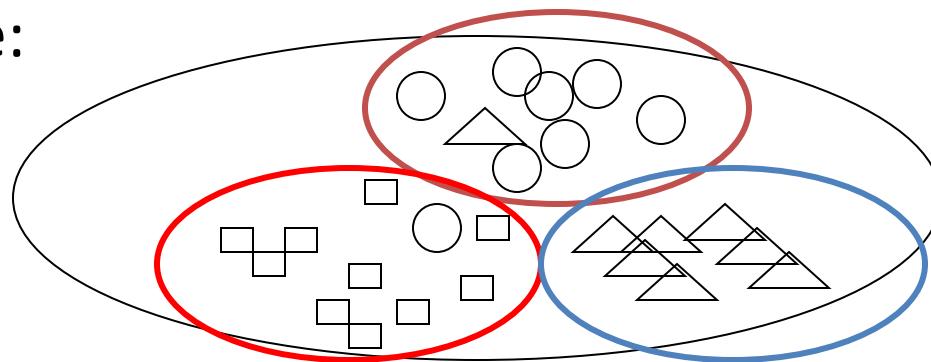


Overview

- What is text clustering? ← **This lecture**
- Why text clustering? ←
- How to do text clustering?
 - Generative probabilistic models
 - Other approaches
- How to evaluate clustering results?

What Is Text Clustering?

- Discover “natural structure”
- Group similar objects together
- Objects can be documents, terms, passages, websites,...
- Example:



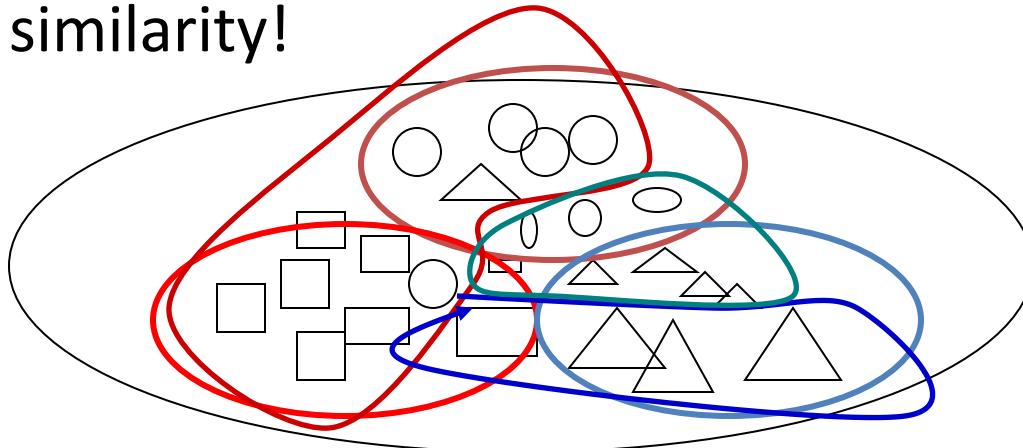
Not well defined!

What does “similar” mean?

The “Clustering Bias”

- Any two objects can be similar, depending on how you look at them!
- Are “car” and “horse” similar?
- A user must define the **perspective** (i.e., a “bias”) for assessing similarity!

Basis for evaluation



Examples of Text Clustering

- Clustering of documents in the whole collection
- Term clustering to define “concept”/“theme”/“topic”
- Clustering of passages/sentences or any selected text segments from larger text objects (e.g., all text segments about a topic discovered using a topic model)
- Clustering of websites (text object has multiple documents)
- Text clusters can be further clustered to generate a hierarchy

Why Text Clustering?

- In general, very useful for text mining and exploratory text analysis:
 - ➔ Get a sense about the overall content of a collection (e.g., what are some of the “typical”/representative documents in a collection?)
 - ➔ Link (similar) text objects (e.g., removing duplicated content)
 - ➔ Create a structure on the text data (e.g., for browsing)
 - ➔ As a way to induce additional features (i.e., clusters) for classification of text objects
- Examples of applications
 - Clustering of search results
 - Understanding major complaints in emails from customers

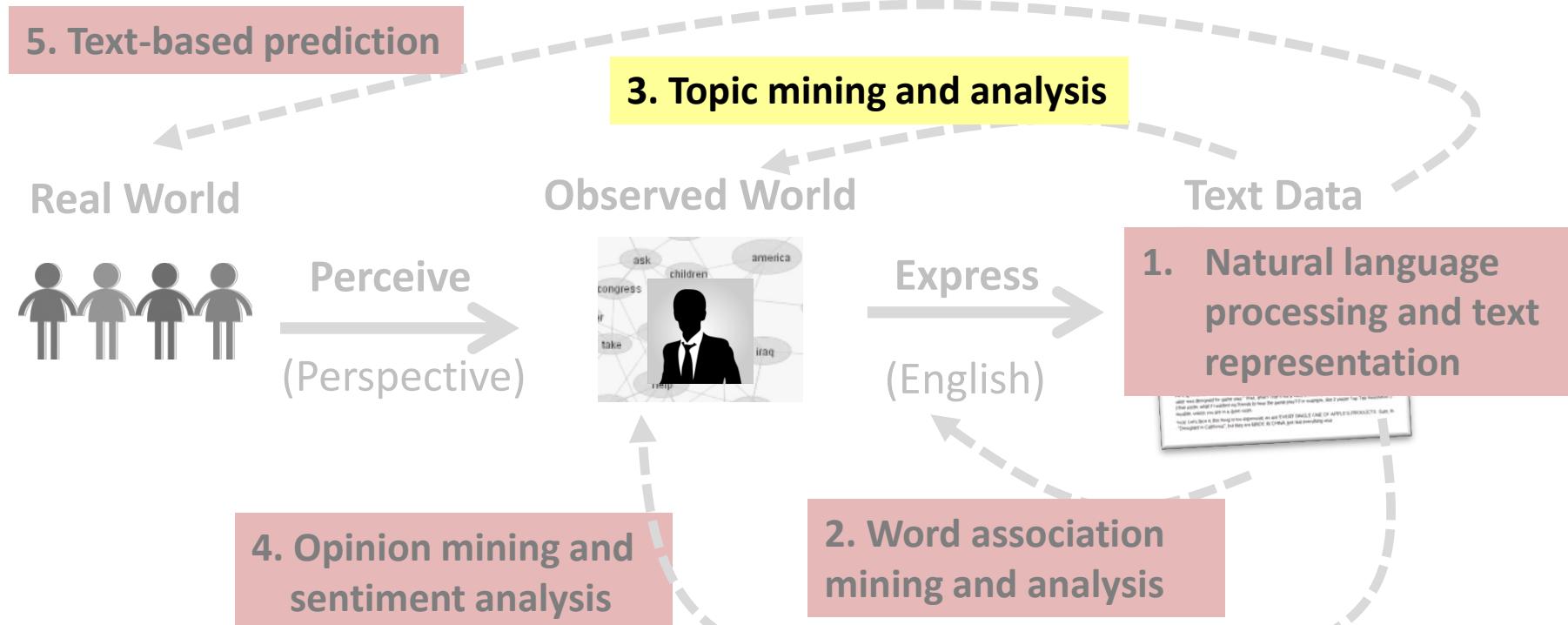
Text Clustering: Generative Probabilistic Models

Part 1

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Text Clustering: Generative Probabilistic Models

(Part 1)



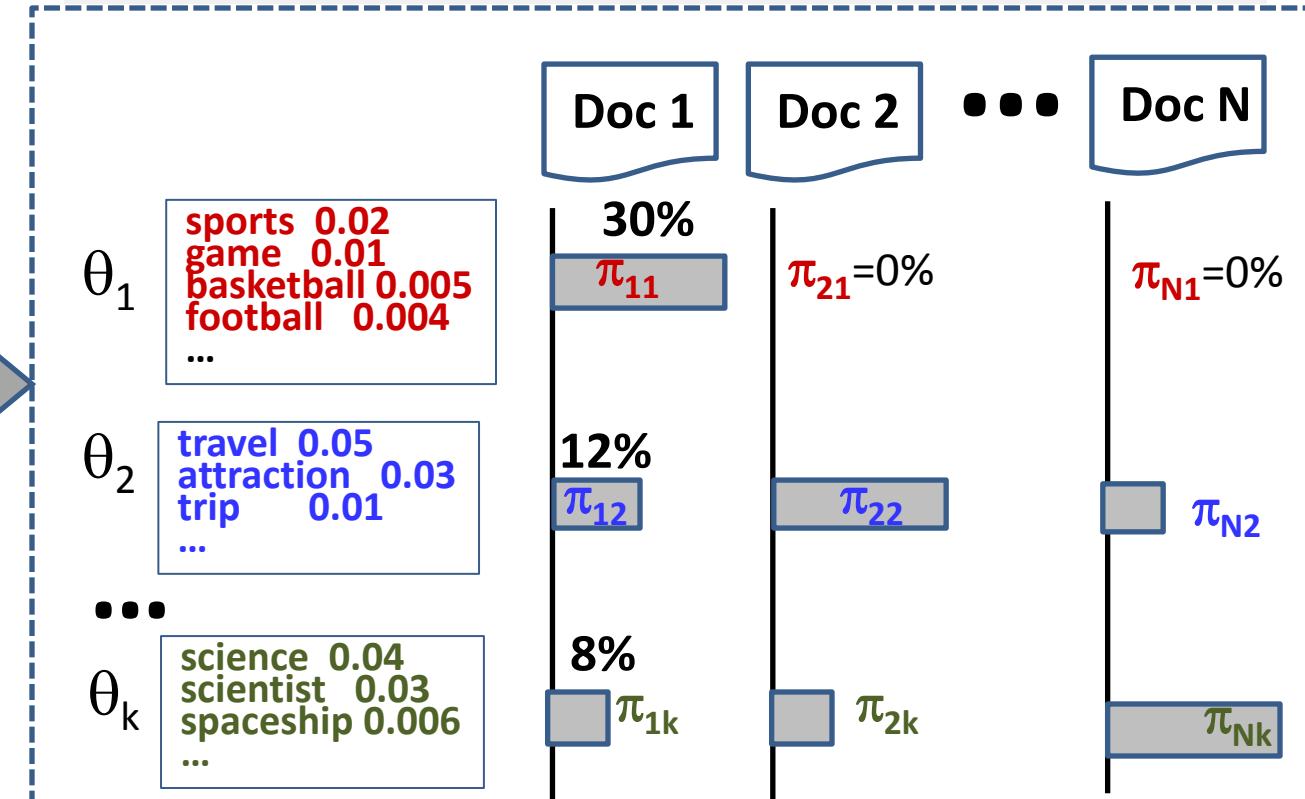
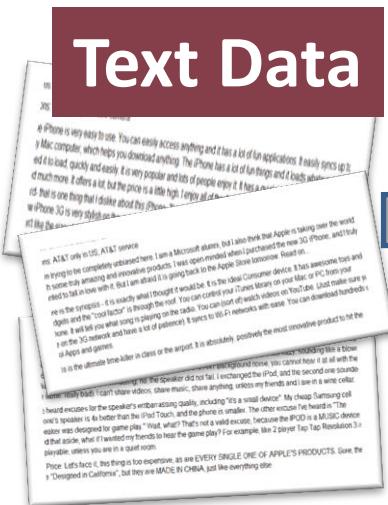
Overview

- What is text clustering?
- Why text clustering?
- How to do text clustering?
 - **Generative probabilistic models**
 - Similarity-based approaches
- How to evaluate clustering results?

Topic Mining Revisited

INPUT: C, k, V

OUTPUT: { $\theta_1, \dots, \theta_k$ }, { $\pi_{i1}, \dots, \pi_{ik}$ }



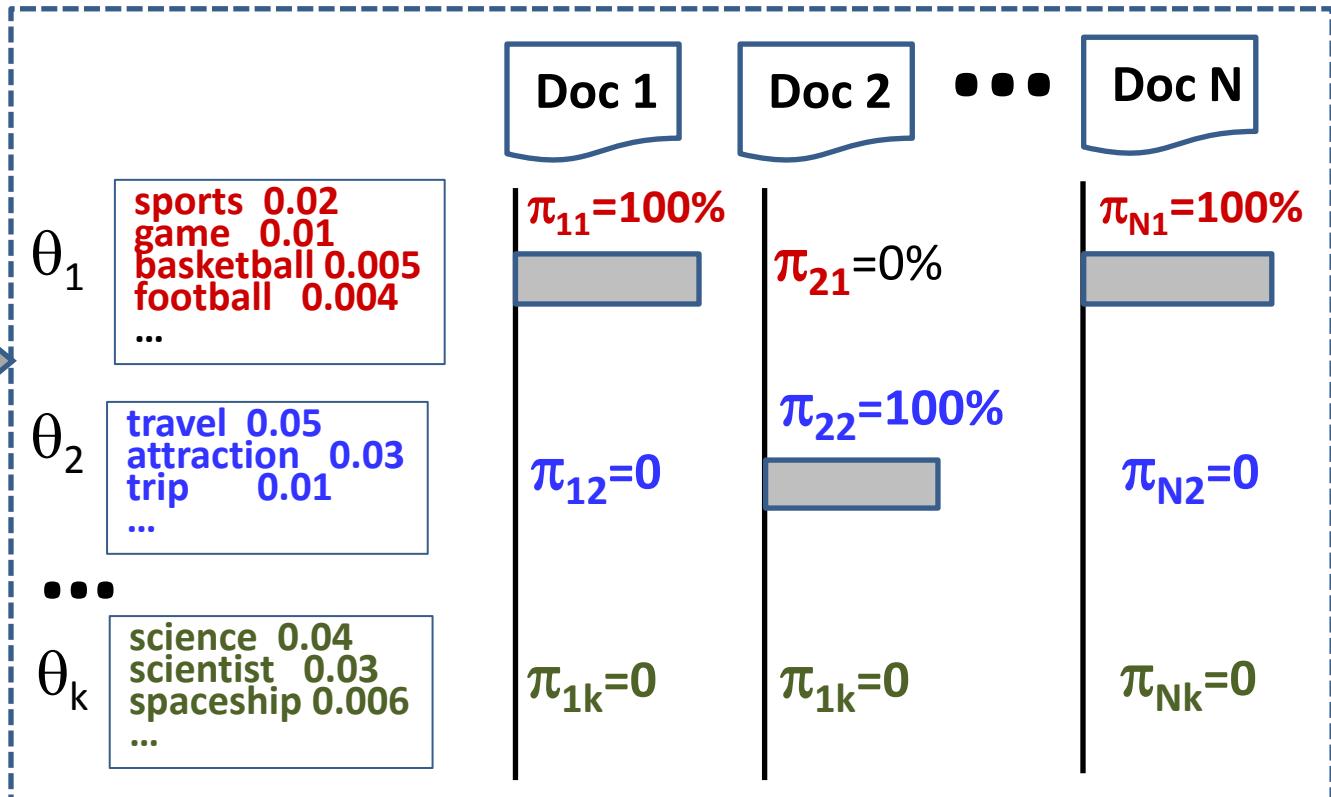
One Topic(=cluster) Per Document

INPUT: C, k, V

OUTPUT: { $\theta_1, \dots, \theta_k$ },

$$\{ c_1, \dots, c_N \} \quad c_i \in [1, k]$$

Text Data



Mining One Topic Revisited

INPUT: $C=\{d\}$, V

OUTPUT: $\{\theta\}$

Text Data

... we are now 1 year away from our 10th anniversary, and unless today is my 10th anniversary or something I am still considering taking it back. Here's why:
Speaker quality is ABSOLUTELY HORRENDOUS. The speaker is simply not functional, unless you are in perhaps a recording studio. When you turn it up all the way (because you can't hear it) it becomes fuzzy sounding like a blow drier. What is this thing, the size of a flea? And if there's ANY background noise, you cannot hear it at all with the volume up (in case you're wondering, no, the speaker did not fail). I exchanged the iPod, and the second one sounds the same, really bad. I can't share videos, share music, share anything, unless my friends and I are in a wine cellar.

(1 Doc, 1 Topic)

→ (N Docs, N Topics)

$k < N$

→ (N Docs, k Shared Topics)=Clustering!

$P(w|\theta)$

θ

text ?
mining ?
association ?
database ?

Doc d

100%

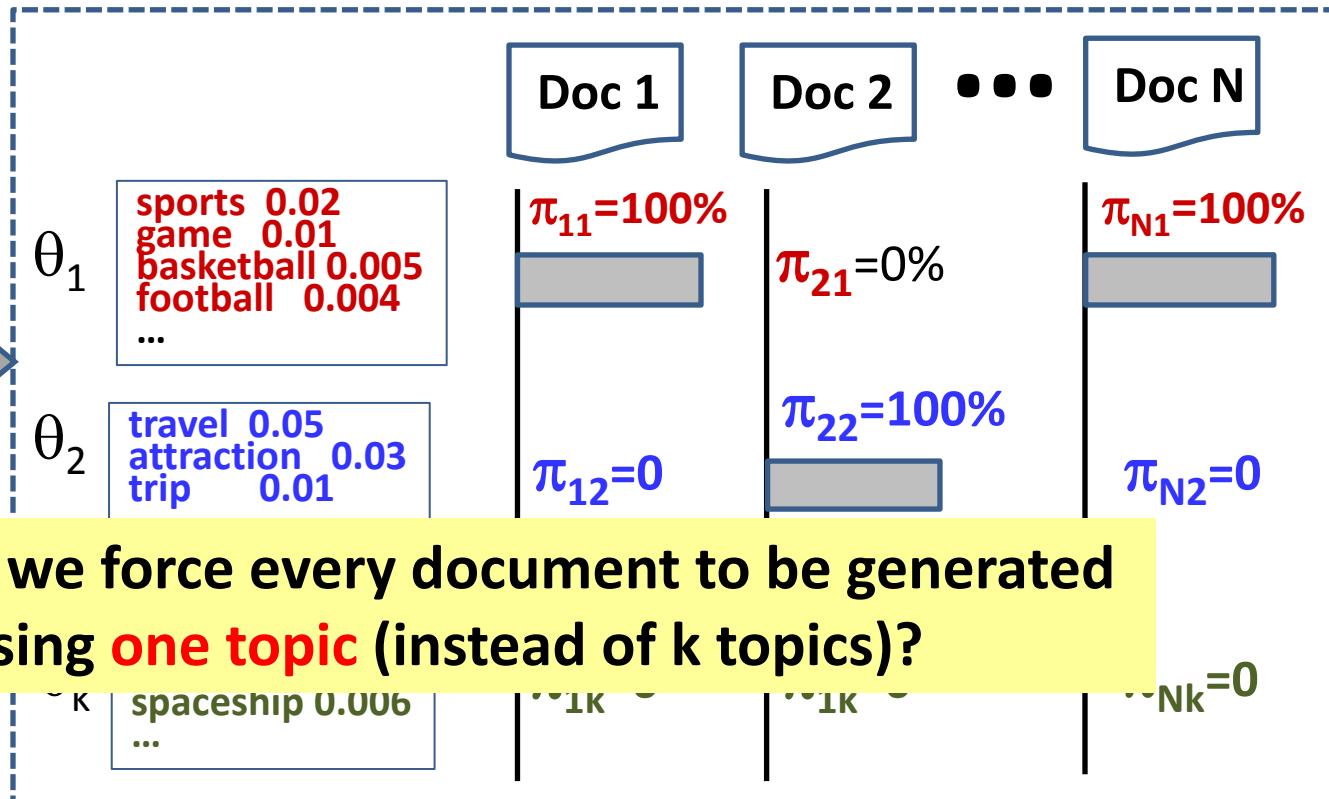
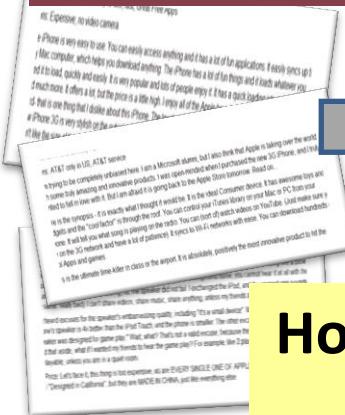
What Generative Model Can Do Clustering?

INPUT: C, k, V

OUTPUT: { $\theta_1, \dots, \theta_k$ },

$$\{ c_1, \dots, c_N \} \quad c_i \in [1, k]$$

Text Data



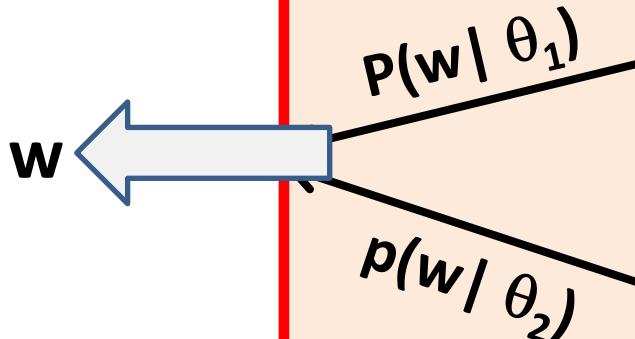
How can we force every document to be generated using **one topic** (instead of k topics)?

Generative Topic Model Revisited

Why can't this model be used for clustering?

d

“the”?
“text”?



t 0.04
mining 0.035
association 0.03
clustering 0.005
...
the 0.000001

the 0.03
a 0.02
is 0.015
we 0.01
food 0.003
...
text 0.000006

$$p(\theta_1) + p(\theta_2) = 1$$

$$P(\theta_1) = 0.5$$

Topic
Choice

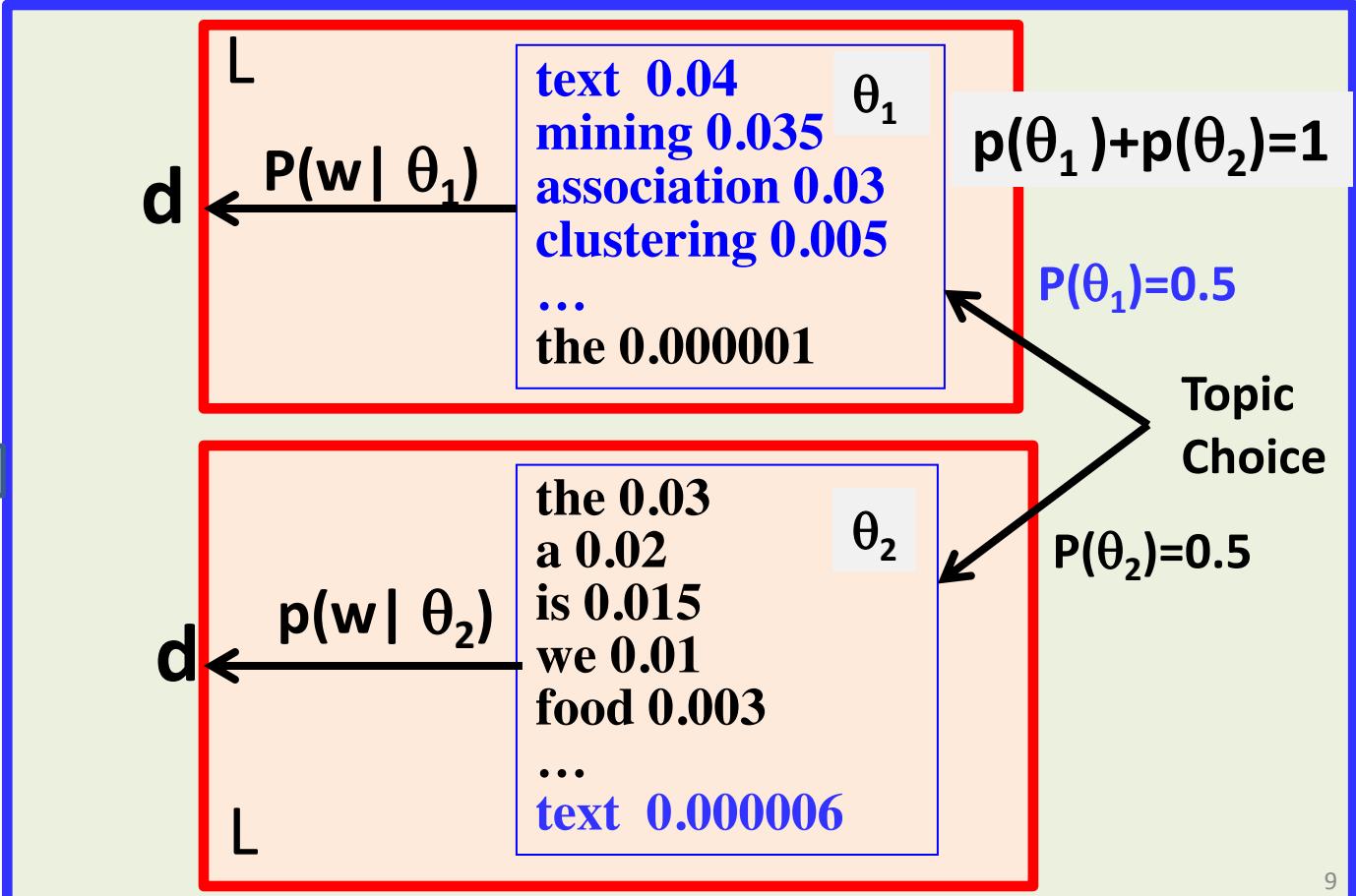
$$P(\theta_2) = 0.5$$

Mixture Model for Document Clustering

Difference from topic model?

$d = x_1 x_2 \dots x_L$

What if $P(\theta_1)=1$ or $P(\theta_2)=1$?



Likelihood Function: $p(d)=?$

$$\begin{aligned} p(d) &= p(\theta_1)p(d | \theta_1) + p(\theta_2)p(d | \theta_2) \\ &= p(\theta_1)\prod_{i=1}^L p(x_i | \theta_1) + p(\theta_2)\prod_{i=1}^L p(x_i | \theta_2) \end{aligned}$$

$d = x_1 x_2 \dots x_L$

How is this different from a topic model?

$P(\theta_1)=0.5$

Topic
Choice

$$\text{topic model : } p(d) = \prod_{i=1}^L [p(\theta_1)p(x_i | \theta_1) + p(\theta_2)p(x_i | \theta_2)]$$

L

food 0.003
...
text 0.000006

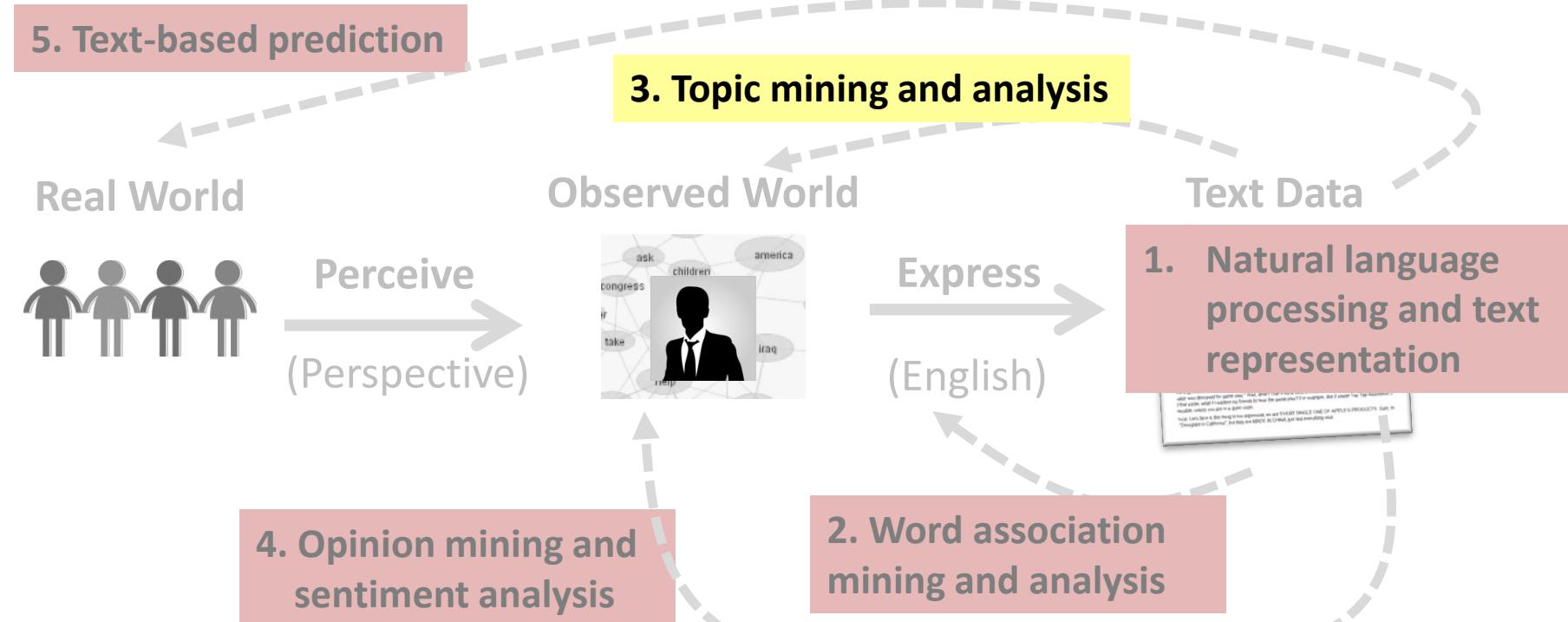
Text Clustering: Generative Probabilistic Models

Part 2

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Text Clustering: Generative Probabilistic Models

(Part 2)



Likelihood Function: $p(d) = ?$

$$p(d) = p(\theta_1)p(d | \theta_1) + p(\theta_2)p(d | \theta_2)$$

$$= p(\theta_1) \prod_{i=1}^L p(x_i | \theta_1) + p(\theta_2) \prod_{i=1}^L p(x_i | \theta_2)$$

$d = x_1 x_2 \dots x_L$

the 0.000001

the 0.03

Topic
Choice

How can we generalize it to include k topics/clusters?

$d \leftarrow$
we 0.01
food 0.003
...
text 0.000006

L

Mixture Model for Document Clustering

- Data: a collection of documents $C = \{d_1, \dots, d_N\}$
- Model: mixture of k unigram LMs: $\Lambda = (\{\theta_i\}; \{p(\theta_i)\}), i \in [1, k]$
 - To generate a document, first **choose a θ_i** according to $p(\theta_i)$, and then generate **all** words in the document using $p(w | \theta_i)$
- Likelihood:

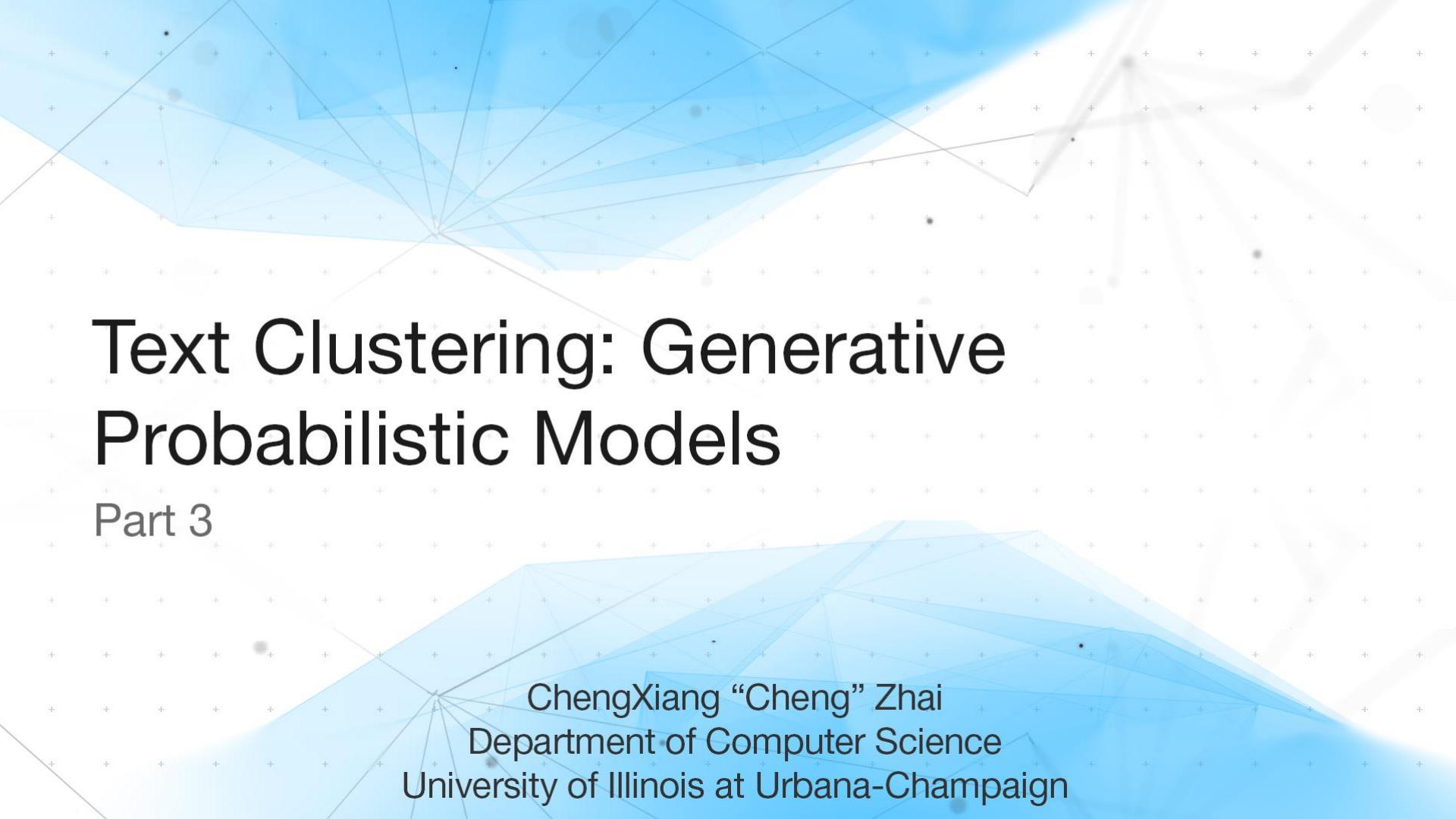
$$\begin{aligned} p(d | \Lambda) &= \sum_{i=1}^k [p(\theta_i) \prod_{j=1}^{|d|} p(x_j | \theta_i)] \\ &= \sum_{i=1}^k [p(\theta_i) \prod_{w \in V} p(w | \theta_i)^{c(w,d)}] \end{aligned}$$

- Maximum Likelihood estimate

$$\Lambda^* = \arg \max_{\Lambda} p(d | \Lambda)$$

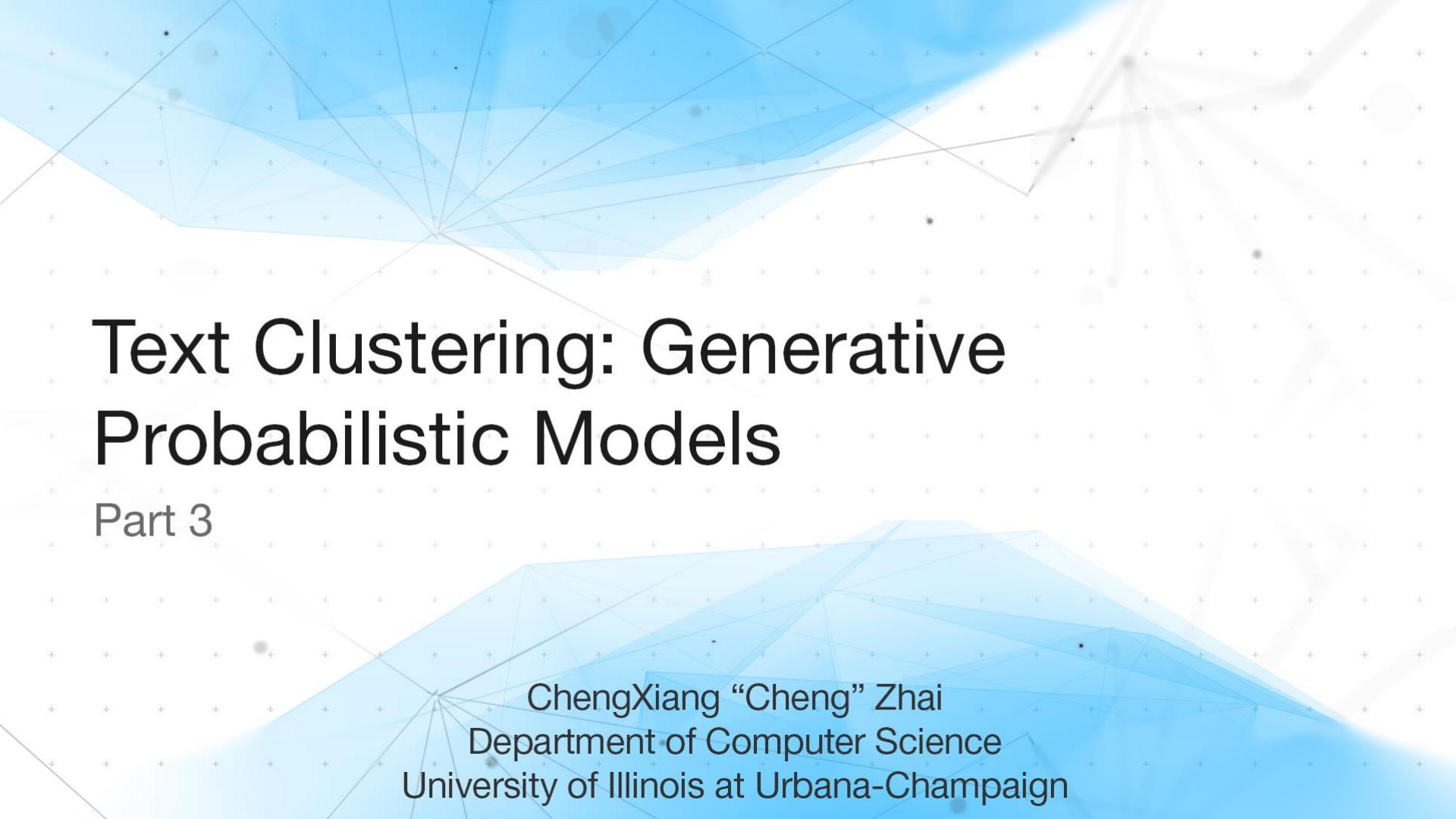
Cluster Allocation After Parameter Estimation

- **Parameters** of the mixture model: $\Lambda = (\{\theta_i\}; \{p(\theta_i)\}), i \in [1, k]$
 - Each θ_i represents the **content of cluster i** : $p(w | \theta_i)$
 - $p(\theta_i)$ indicates the **size of cluster i**
 - Note that unlike in PLSA, $p(\theta_i)$ doesn't depend on d!
- Which cluster should document d belong to? $c_d = ?$
 - **Likelihood only**: Assign d to the cluster corresponding to the topic θ_i that most likely has been used to generate d
$$c_d = \arg \max_i p(d | \theta_i)$$
 - **Likelihood + prior $p(\theta_i)$ (Bayesian)**: favor large clusters
$$c_d = \arg \max_i p(d | \theta_i) p(\theta_i)$$



Text Clustering: Generative Probabilistic Models

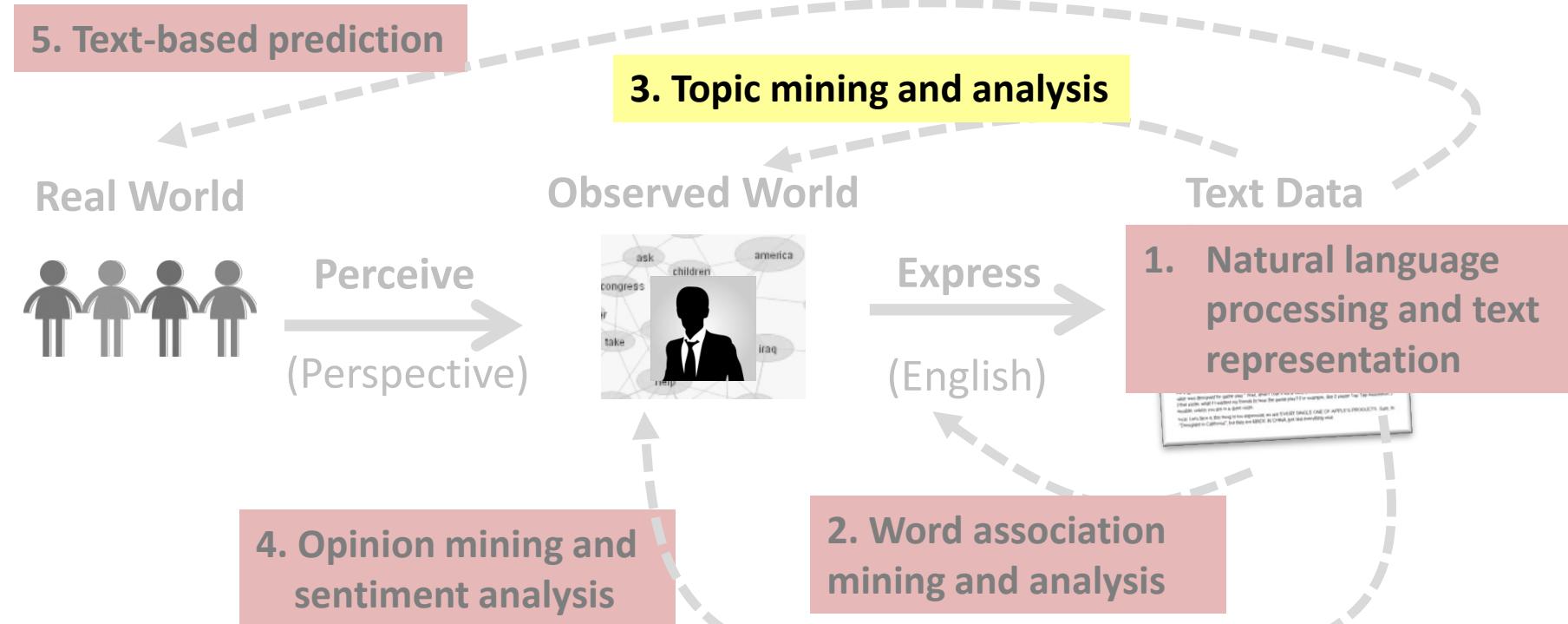
Part 3



ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Text Clustering: Generative Probabilistic Models

(Part 3)



How Can We Compute the ML Estimate?

- Data: a collection of documents $C = \{d_1, \dots, d_N\}$
- Model: mixture of k unigram LMs: $\Lambda = (\{\theta_i\}; \{p(\theta_i)\}), i \in [1, k]$
 - To generate a document, first **choose a θ_i** according to $p(\theta_i)$ and then generate **all** words in the document using $p(w | \theta_i)$
- Likelihood:

$$p(d | \Lambda) = \sum_{i=1}^k [p(\theta_i) \prod_{w \in V} p(w | \theta_i)^{c(w,d)}]$$

$$p(C | \Lambda) = \prod_{j=1}^N p(d_j | \Lambda)$$

- Maximum Likelihood estimate

$$\Lambda^* = \arg \max_{\Lambda} p(C | \Lambda)$$

EM Algorithm for Document Clustering

- Initialization: Randomly set $\Lambda = (\{\theta_i\}; \{p(\theta_i)\}), i \in [1, k]$
- Repeat until likelihood $p(C|\Lambda)$ converges**
 - E-Step: Infer which distribution has been used to generate document d: hidden variable $Z_d \in [1, k]$**

$$p^{(n)}(Z_d = i | d) \propto p^{(n)}(\theta_i) \prod_{w \in V} p^{(n)}(w | \theta_i)^{c(w,d)}$$

$$\sum_{i=1}^k p^{(n)}(Z_d = i | d) = 1$$

- M-Step: Re-estimation of all parameters**

$$p^{(n+1)}(\theta_i) \propto \sum_{j=1}^N p^{(n)}(Z_{d_j} = i | d_j)$$

$$\sum_{i=1}^k p^{(n+1)}(\theta_i) = 1$$

$$p^{(n+1)}(w | \theta_i) \propto \sum_{j=1}^N c(w, d_j) p^{(n)}(Z_{d_j} = 1 | d_j)$$

$$\sum_{w \in V} p^{(n+1)}(w | \theta_i) = 1, \quad \forall i \in [1, k]$$

An Example of 2 Clusters

Random Initialization

$$p(\theta_1) = p(\theta_2) = 0.5$$

	$p(w \theta_1)$	$p(w \theta_2)$
text	0.5	0.1
mining	0.2	0.1
medical	0.2	0.75
health	0.1	0.05

E-step

Document d

Hidden variables:

$$Z_d \in \{1, 2\}$$

	$c(w,d)$
text	2
mining	2
medical	0
health	0

$$\begin{aligned}
 p(Z_d = 1 | d) &= \frac{p(\theta_1)p("text"|\theta_1)^2 p("mining"|\theta_1)^2}{p(\theta_1)p("text"|\theta_1)^2 p("mining"|\theta_1)^2 + p(\theta_2)p("text"|\theta_2)^2 p("mining"|\theta_2)^2} \\
 &= \frac{0.5 * 0.5^2 * 0.2^2}{0.5 * 0.5^2 * 0.2^2 + 0.5 * 0.1^2 * 0.1^2} = \frac{100}{101}
 \end{aligned}$$

$$p(Z_d = 2 | d) = ?$$

Normalization to Avoid Underflow

	$p(w \theta_1)$	$p(w \theta_2)$	$p(w \bar{\theta})$
text	0.5	0.1	$(0.5+0.1)/2$
mining	0.2	0.1	$(0.2+0.1)/2$
medical	0.2	0.75	$(0.2+0.75)/2$
health	0.1	0.05	$(0.1+0.05)/2$

Average of $p(w|\theta_i)$
as a possible normalizer

$$p(Z_d = 1 | d) = \frac{\frac{p(\theta_1)p("text"|\theta_1)^2 p("mining"|\theta_1)^2}{p("text"|\bar{\theta})^2 p("mining"|\bar{\theta})^2}}{\frac{p(\theta_1)p("text"|\theta_1)^2 p("mining"|\theta_1)^2}{p("text"|\bar{\theta})^2 p("mining"|\bar{\theta})^2} + \frac{p(\theta_2)p("text"|\theta_2)^2 p("mining"|\theta_2)^2}{p("text"|\bar{\theta})^2 p("mining"|\bar{\theta})^2}}$$

An Example of 2 Clusters (cont.)

From E-Step

	$P(Z_d=1 d)$		$c("text")$	$c("mining")$
d1	0.9	d1	2	3
d2	0.1	d2	1	2
d3	0.8	d3	4	3

M-Step

$$p(\theta_1) = ? \quad p(\theta_2) = ?$$

$$\begin{aligned} p(\theta_1) &= \frac{p(Z_{d_1}=1|d_1) + p(Z_{d_2}=1|d_2) + p(Z_{d_3}=1|d_3)}{3} \\ &= \frac{0.9 + 0.1 + 0.8}{3} = 0.6 \end{aligned}$$

	$p(w \theta_1)$	$p(w \theta_2)$
text	?	?
mining	?	?
medical	?	?
health	?	?

$$\begin{aligned} p("text"|\theta_1) &\propto c("text", d_1) * p(Z_{d_1}=1|d_1) + \dots + c("text", d_3) * p(Z_{d_3}=1|d_3) \\ &= 2 * 0.9 + 1 * 0.1 + 4 * 0.8 \end{aligned}$$

$$p("mining"|\theta_1) \propto 3 * 0.9 + 2 * 0.1 + 3 * 0.8$$

$$p("text"|\theta_1) + p("mining"|\theta_1) + p("medical"|\theta_1) + p("health"|\theta_1) = 1$$

Summary of Generative Model for Clustering

- A slight variation of topic model can be used for clustering documents
 - Each **cluster** is represented by a **unigram LM** $p(w|\theta_i)$ → **Term cluster**
 - A document is generated by first choosing a unigram LM and then generating **ALL words** in the document using this **single LM**
 - Estimated model parameters give both a topic characterization of each cluster and a probabilistic assignment of a document into each cluster
 - “Hard” clusters can be obtained by forcing a document into the cluster corresponding to the unigram LM most likely used to generate the document
- EM algorithm can be used to compute the ML estimate
 - Normalization is often needed to avoid underflow

Text Clustering: Similarity-based Approaches

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Overview

- What is text clustering?
- Why text clustering?
- How to do text clustering?
 - Generative probabilistic models
 - **Similarity-based approaches**
- How to evaluate clustering results?

Similarity-based Clustering: General Idea

- Explicitly define a similarity function to measure similarity between two text objects (i.e., providing “clustering bias”)
- Find an optimal partitioning of data to
 - maximize intra-group similarity and
 - minimize inter-group similarity
- Two strategies for obtaining optimal clustering
 - Progressively construct a hierarchy of clusters (hierarchical clustering)
 - Bottom-up (agglomerative): gradually group similar objects into larger clusters
 - Top-down (divisive): gradually partition the data into smaller clusters
 - Start with an initial tentative clustering and iteratively improve it (“flat” clustering, e.g., k-Means)

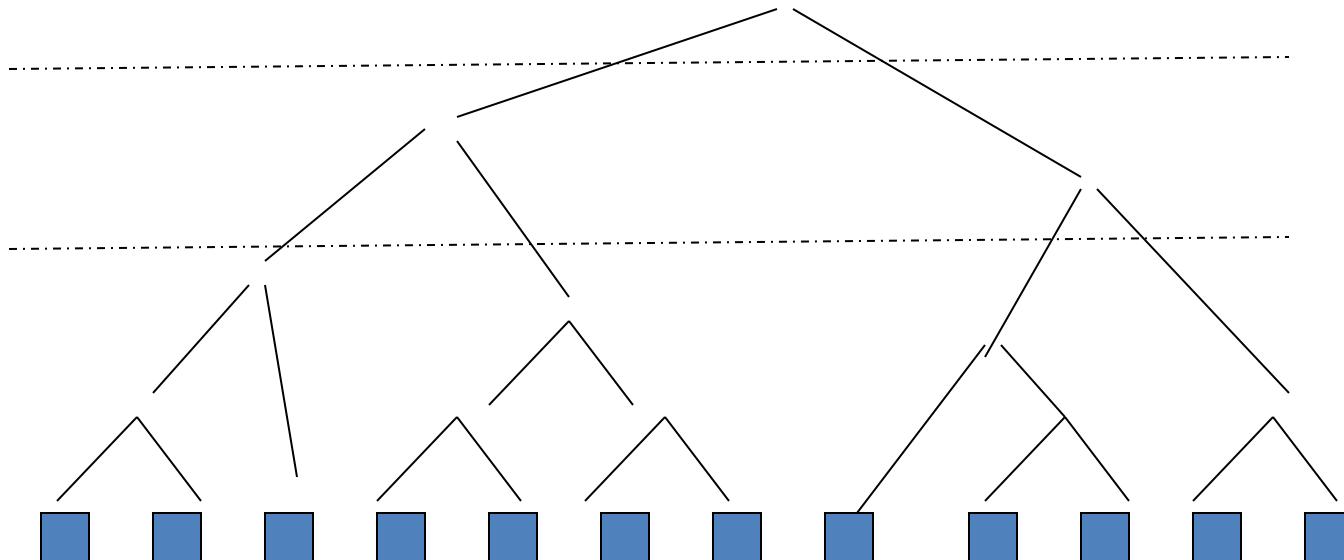
Similarity-based Clustering Methods

- Many general clustering methods are available!
- Two representative methods
 - Hierarchical Agglomerative Clustering (HAC)
 - k-means

Agglomerative Hierarchical Clustering

- Given a similarity function to measure similarity between two objects
- Gradually group similar objects together in a bottom-up fashion to form a hierarchy
- Stop when some stopping criterion is met
- Variations: different ways to compute group similarity based on individual object similarity

Similarity-induced Structure



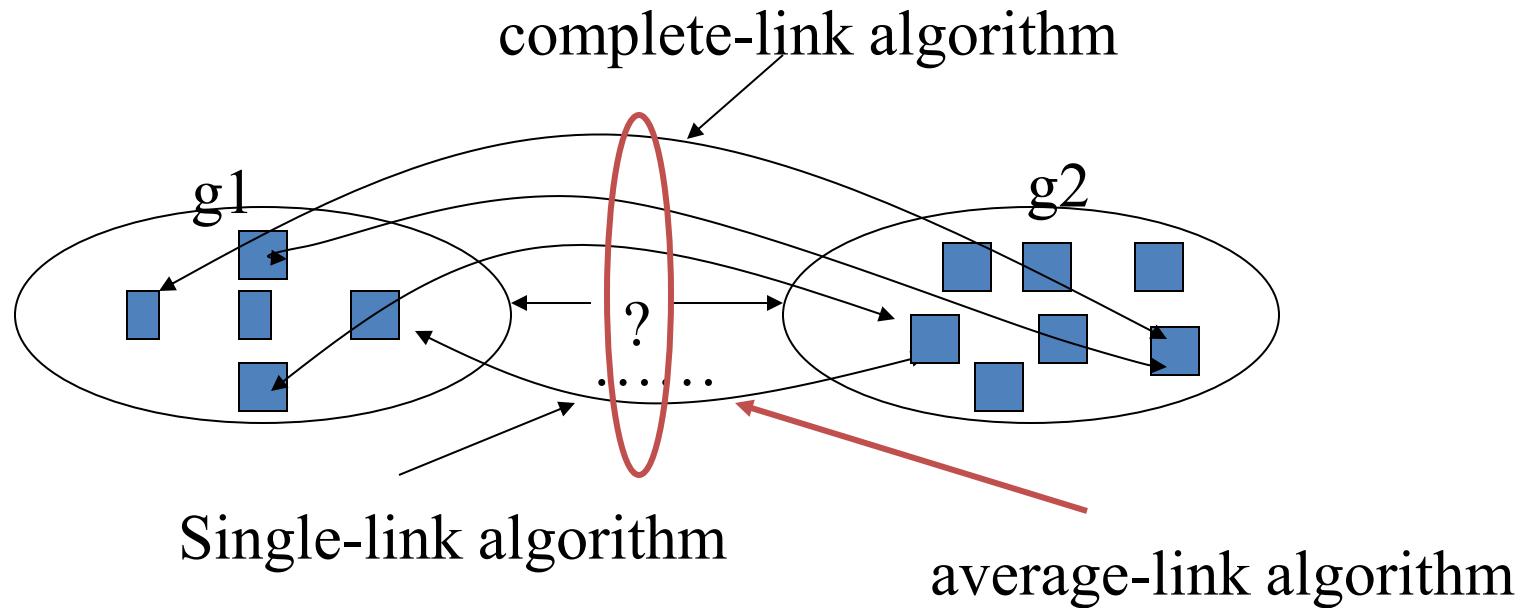
How to Compute Group Similarity

Three popular methods:

Given two groups g_1 and g_2 ,

- Single-link algorithm: $s(g_1, g_2) =$ similarity of the **closest** pair
- Complete-link algorithm: $s(g_1, g_2) =$ similarity of the **farthest** pair
- Average-link algorithm: $s(g_1, g_2) =$ **average** of similarity of all pairs

Group Similarity Illustrated



Comparison of Single-Link, Complete-Link, and Average-Link

- Single-link
 - “Loose” clusters
 - Individual decision, sensitive to outliers
- Complete-link
 - “Tight” clusters
 - Individual decision, sensitive to outliers
- Average-link
 - “In between”
 - Group decision, insensitive to outliers
- Which one is the best? It depends on what you need!

K-Means Clustering

- Represent each text object as a term vector and assume a similarity function defined on two objects
- Start with k randomly selected vectors and assume they are the centroids of k clusters (initial tentative clustering) → **Initialization**
- Assign every vector to a cluster whose centroid is the closest to the vector \approx E-step difference?
- Re-compute the centroid for each cluster based on the newly assigned vectors in the cluster \approx M-step difference?
- Repeat this process until the similarity-based objective function (i.e., within cluster sum of squares) converges (to a local minimum)

Very similar to clustering with EM for mixture model!

Summary of Clustering Methods

- Model based approaches (mixture model)
 - Uses an implicit similarity function (model → clustering bias)
 - Cluster structure is “built” into a generative model
 - Complex generative models can discover complex structures
 - Prior can be leveraged to further customize the clustering algorithm
 - However, no easy way to directly control the similarity measure
- Similarity-based approaches
 - Allows for direct and flexible specification of similarity
 - Objective function to be optimized is not always clear
- Both approaches can generate both term clusters and doc clusters

Text Clustering: Evaluation

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

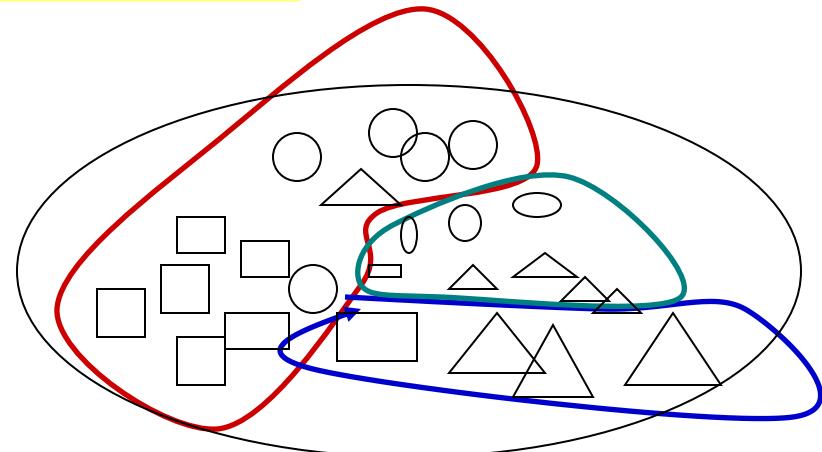
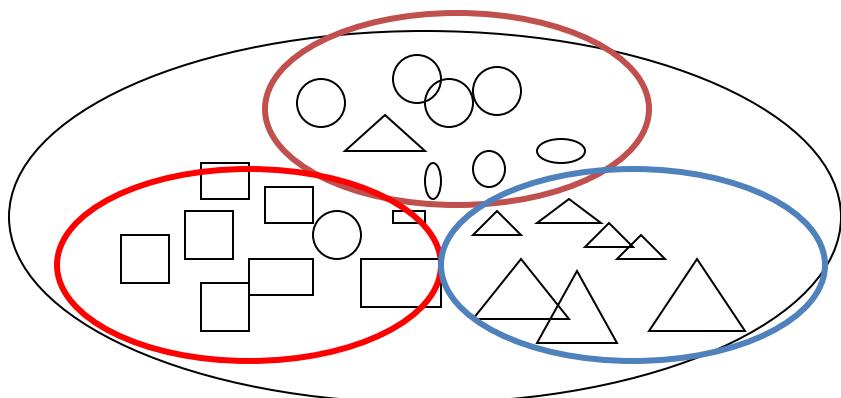
Overview

- What is text clustering?
- Why text clustering?
- How to do text clustering?
 - Generative probabilistic models
 - Similarity-based approaches
- **How to evaluate clustering results?**

The “Clustering Bias”

- Any two objects can be similar, depending on how you look at them!
- A user must define the **perspective** (i.e., a “bias”) for assessing similarity!

↑ Basis for evaluation



Direct Evaluation of Text Clusters

- Question to answer: How close are the system-generated clusters to the ideal clusters (generated by humans)?
 - “Closeness” can be assessed from multiple perspectives
 - “Closeness” can be quantified
 - “Clustering bias” is imposed by the human assessors
- Evaluation procedure:
 - Given a test set, have humans to create an ideal clustering result (i.e., an ideal partitioning of text objects or “gold standard”)
 - Use a system to produce clusters from the same test set
 - Quantify the similarity between the system-generated clusters and the gold standard clusters
 - Similarity can be measured from multiple perspectives (e.g., purity, normalized mutual information, F measure)

Indirect Evaluation of Text Clusters

- Question to answer: how useful are the clustering results for the intended applications?
 - “Usefulness” is inevitably application specific
 - “Clustering bias” is imposed by the intended application
- Evaluation procedure:
 - Create a test set for the intended application to quantify the performance of any system for this application
 - Choose a baseline system to compare with
 - Add a clustering algorithm to the baseline system → “clustering system”
 - Compare the performance of the clustering system and the baseline in terms of any performance measure for the application

Summary of Text Clustering

- Text clustering is an unsupervised general text mining technique to
 - obtain an overall picture of the text content (exploring text data)
 - discover interesting clustering structures in text data
- Many approaches are possible
 - Strong clusters tend to show up no matter what method used
 - Effectiveness of a method highly depends on whether the desired clustering bias is captured appropriately (either through using the right generative model or the right similarity function)
 - Deciding the optimal number of clusters is generally a difficult problem for any method due to the unsupervised nature
- Evaluation of clustering results can be done both directly and indirectly

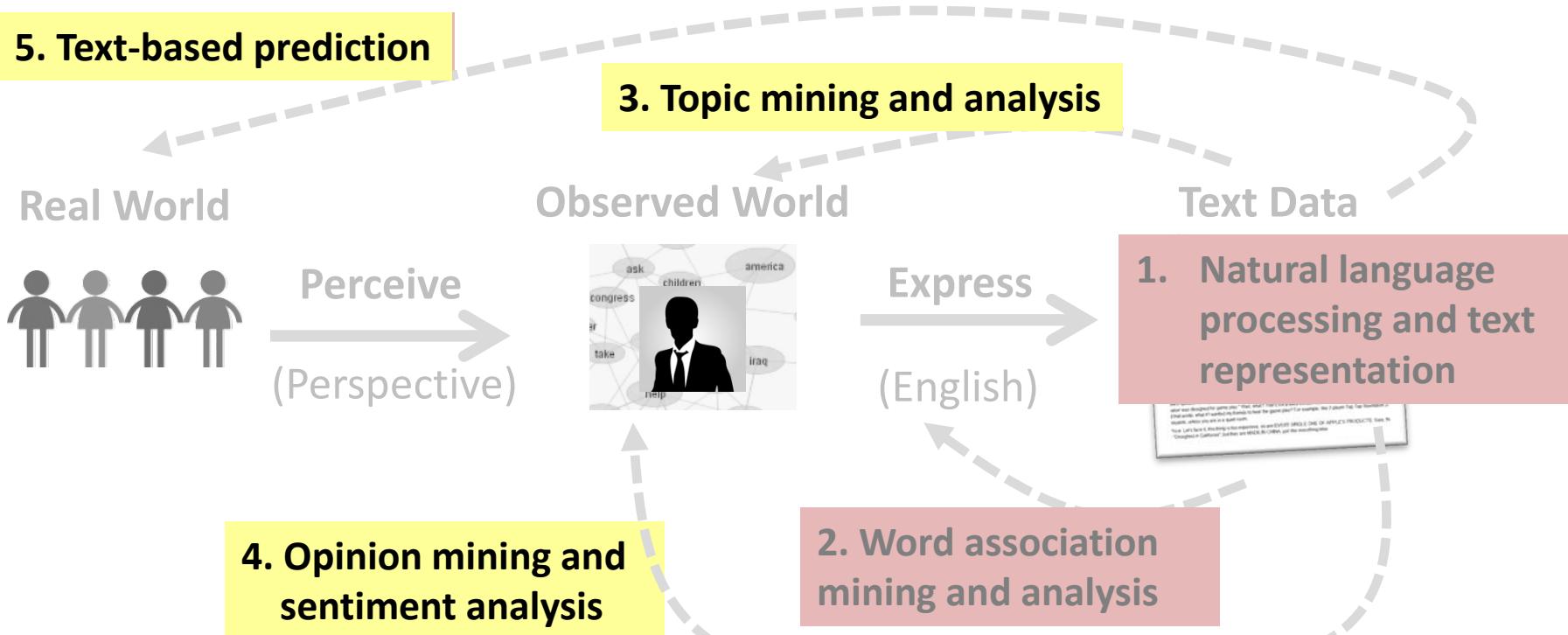
Suggested Reading

- Manning, Chris D., Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2007. (Chapter 16)

Text Categorization: Motivation

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Text Categorization

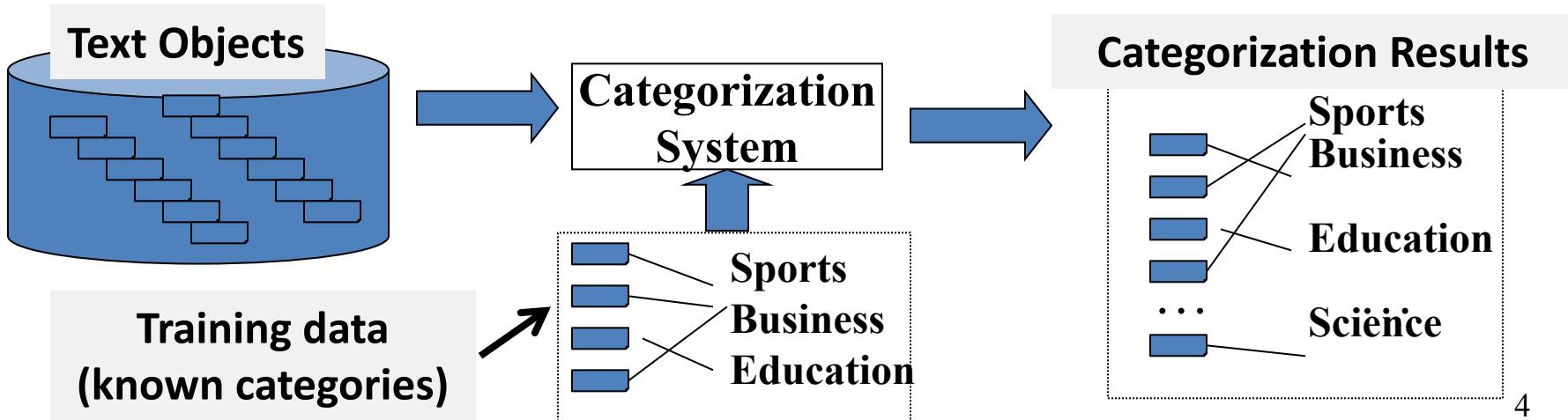


Overview

- What is text categorization? ← **This lecture**
- Why text categorization? ←
- How to do text categorization?
 - Generative probabilistic models
 - Discriminative approaches
- How to evaluate categorization results?

Text Categorization

- Given the following:
 - A set of **predefined categories**, possibly forming a hierarchy
and often
 - A **training set** of labeled text objects
 - Task: **Classify** a text object into **one or more** of the **categories**



Examples of Text Categorization

- **Text objects can vary** (e.g., documents, passages, or collections of text)
- **Categories can also vary**
 - “**Internal**” categories that characterize a text object (e.g., topical categories, sentiment categories)
 - “**External**” categories that characterize an entity associated with the text object (e.g., author attribution or any other meaningful categories associated with text data)
- Some **examples of applications**
 - News categorization, literature article categorization (e.g., MeSH annotations)
 - Spam email detection/filtering
 - Sentiment categorization of product reviews or tweets
 - Automatic email sorting/routing
 - Author attribution

Variants of Problem Formulation

- **Binary** categorization: Only two categories
 - Retrieval: {relevant-doc, non-relevant-doc}
 - Spam filtering: {spam, non-spam}
 - Opinion: {positive, negative}
- **K-category** categorization: More than two categories
 - Topic categorization: {sports, science, travel, business,...}
 - Email routing: {folder1, folder2, folder3,...}
- **Hierarchical** categorization: Categories form a hierarchy
- **Joint** categorization: Multiple **related** categorization tasks done in a joint manner

Binary categorization can potentially support all other categorizations

Why Text Categorization?

- To **enrich text representation** (more understanding of text)
 - Text can now be represented in multiple levels (keywords + categories)
 - Semantic categories assigned can be directly or indirectly useful for an application
 - Semantic categories facilitate aggregation of text content (e.g., aggregating all positive/negative opinions about a product)
- To **infer properties of entities** associated with text data
(discovery of knowledge about the world)
 - As long as an entity can be associated with text data, we can always use the text data to help categorize the associated entities
 - E.g., discovery of non-native speakers of a language; prediction of party affiliation based on a political speech

Text Categorization: Methods

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Overview

- What is text categorization?
- Why text categorization?
- **How to do text categorization?**
 - Generative probabilistic models
 - Discriminative approaches
- How to evaluate categorization results?

Categorization Methods: Manual

- Determine the category based on rules that are carefully designed to reflect the domain knowledge about the categorization problem
- Works well when
 - The categories are very well defined
 - Categories are easily distinguished based on surface features in text (e.g., special vocabulary is known to only occur in a particular category)
 - Sufficient domain knowledge is available to suggest many effective rules
- Problems
 - Labor intensive → doesn't scale up well
 - Can't handle uncertainty in rules; rules may be inconsistent → not robust
- Both problems can be solved/alleviated by using machine learning

Categorization Methods: “Automatic”

- Use **human experts** to
 - Annotate data sets with **category labels** → Training data
 - Provide a set of **features** to represent each text object that can potentially provide a “clue” about the category
- Use **machine learning** to learn “soft rules” for categorization from the training data
 - Figure out **which features are most useful** for separating different categories
 - **Optimally combine the features to minimize the errors** of categorization on the training data
 - The trained classifier can then be applied to a new text object to predict the most likely category (that a human expert would assign to it)

Machine Learning for Text Categorization

- **General setup:** Learn a classifier $f: X \rightarrow Y$
 - Input: X = all text objects; Output: Y = all categories
 - Learn a classifier function, $f: X \rightarrow Y$, such that $f(x)=y \in Y$ gives the correct category for $x \in X$ (“correct” is based on the training data)
- **All methods**
 - Rely on discriminative features of text objects to distinguish categories
 - Combine multiple features in a weighted manner
 - Adjust weights on features to minimize errors on the training data
- **Different methods** tend to vary in
 - Their way of measuring the errors on the training data (may optimize a different objective/loss/cost function)
 - Their way of combining features (e.g., linear vs. non-linear)

Generative vs. Discriminative Classifiers

- **Generative** classifiers (learn **what the data “looks” like in each category**)
 - Attempt to model $p(X,Y) = p(Y)p(X|Y)$ and compute $p(Y|X)$ based on $p(X|Y)$ and $p(Y)$ by using Bayes Rule
 - Objective function is likelihood, thus indirectly measuring training errors
 - E.g., Naïve Bayes
- **Discriminative** classifiers (learn **what features separate categories**)
 - Attempt to model $p(Y|X)$ directly
 - Objective function directly measures errors of categorization on training data
 - E.g., Logistic Regression, Support Vector Machine (SVM), k-Nearest Neighbors (kNN)

Text Categorization: Generative Probabilistic Models

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Overview

- What is text categorization?
- Why text categorization?
- How to do text categorization?
 - **Generative probabilistic models**
 - Discriminative approaches
- How to evaluate categorization results?

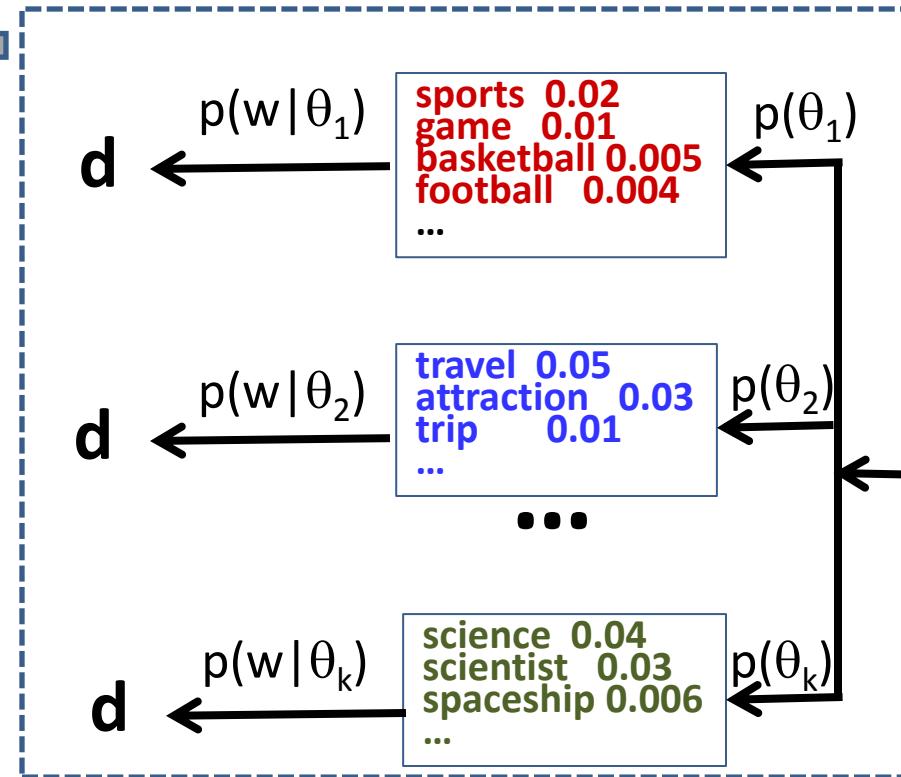
Document Clustering Revisited

Which cluster does d belong to? → Which θ_i was used to generate d ?

$$d = x_1 x_2 \dots x_L \text{ where } x_i \in V$$

$$\begin{aligned} \text{cluster}(d) &= \arg \max_i p(\theta_i | d) \\ &= \arg \max_i p(d | \theta_i) p(\theta_i) \\ &= \arg \max_i \prod_{j=1}^L p(x_j | \theta_i) p(\theta_i) \\ &= \arg \max_i \prod_{w \in V} p(w | \theta_i)^{c(w,d)} p(\theta_i) \end{aligned}$$

$$\begin{aligned} p(\theta_i | d) &= \frac{p(d | \theta_i) p(\theta_i)}{p(d)} \\ &= \frac{p(d | \theta_i) p(\theta_i)}{\sum_{j=1}^k p(d | \theta_j) p(\theta_j)} \end{aligned}$$



Text Categorization with Naïve Bayes Classifier

$$d = x_1 x_2 \dots x_L \text{ where } x_i \in V$$

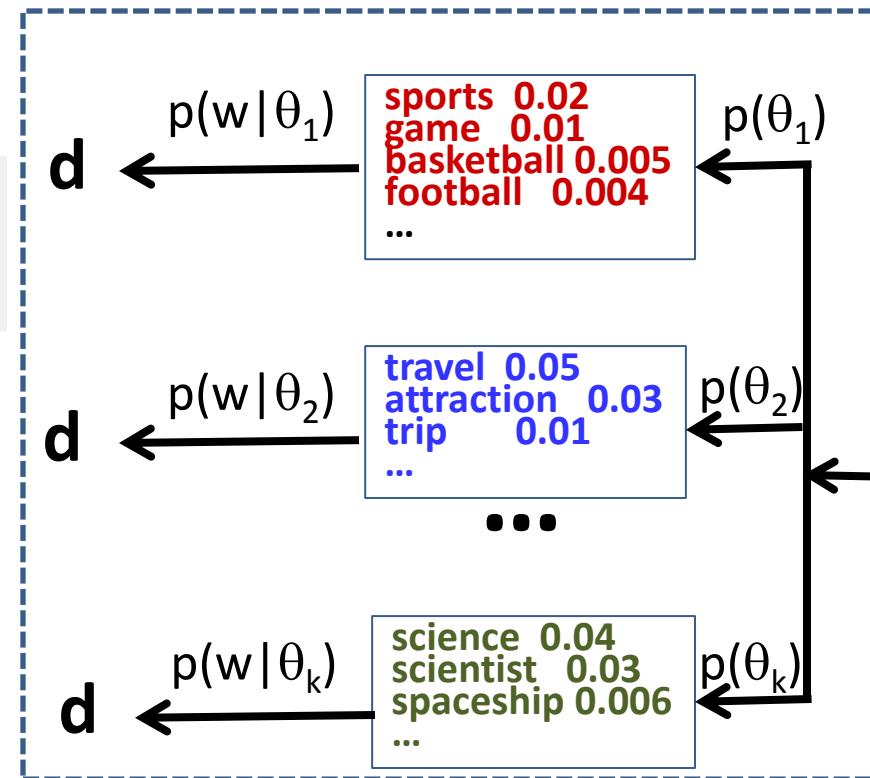
IF θ_i represents category i accurately,
then...

How can we make this happen?

$$\text{category}(d) = \arg \max_i p(\theta_i | d)$$

$$= \arg \max_i p(d | \theta_i) p(\theta_i)$$

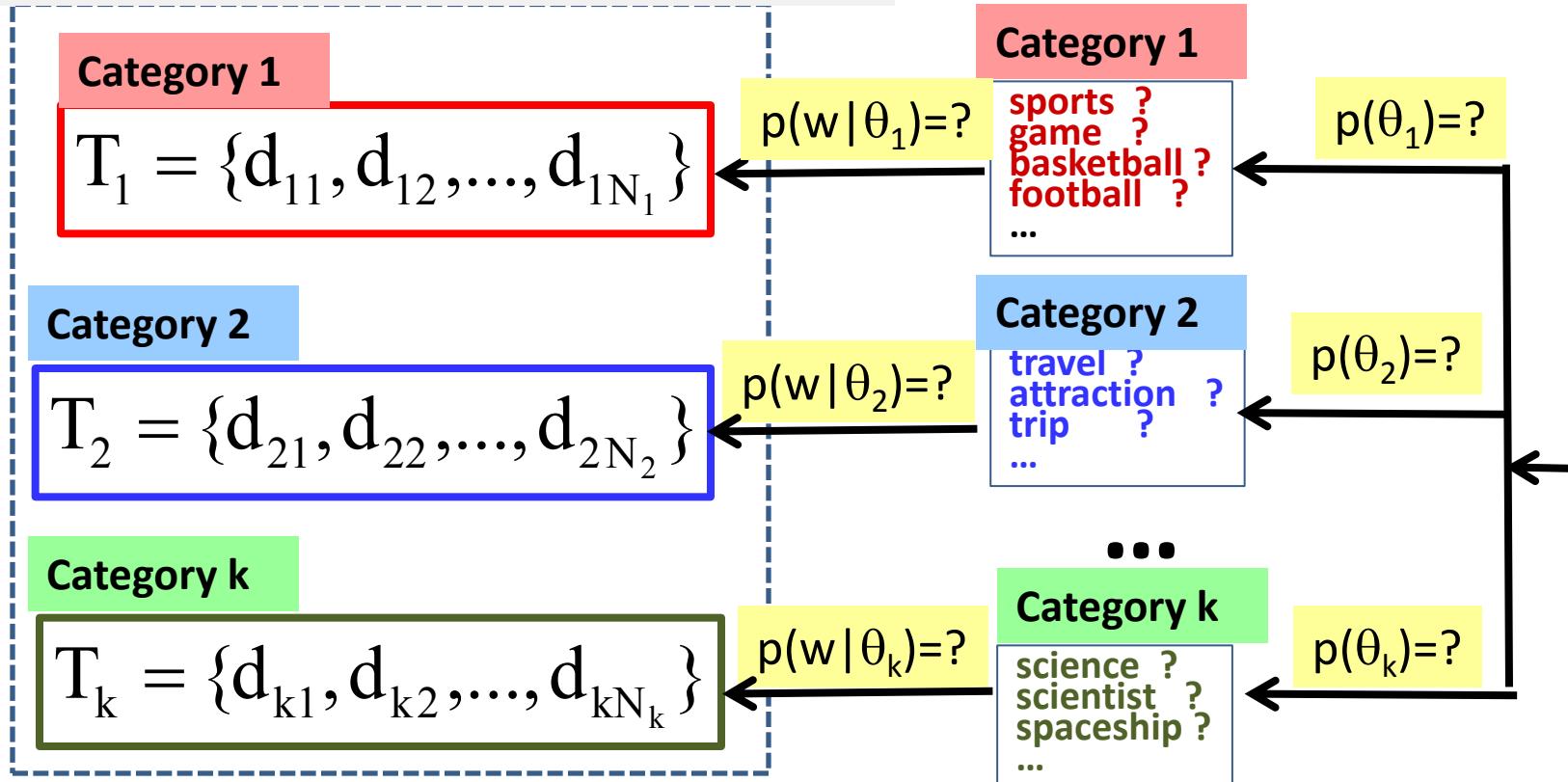
$$= \arg \max_i \prod_{w \in V} p(w | \theta_i)^{c(w,d)} p(\theta_i)$$



$$\text{category}(d) = \arg \max_i \log p(\theta_i) + \sum_{w \in V} c(w,d) \log p(w | \theta_i)$$

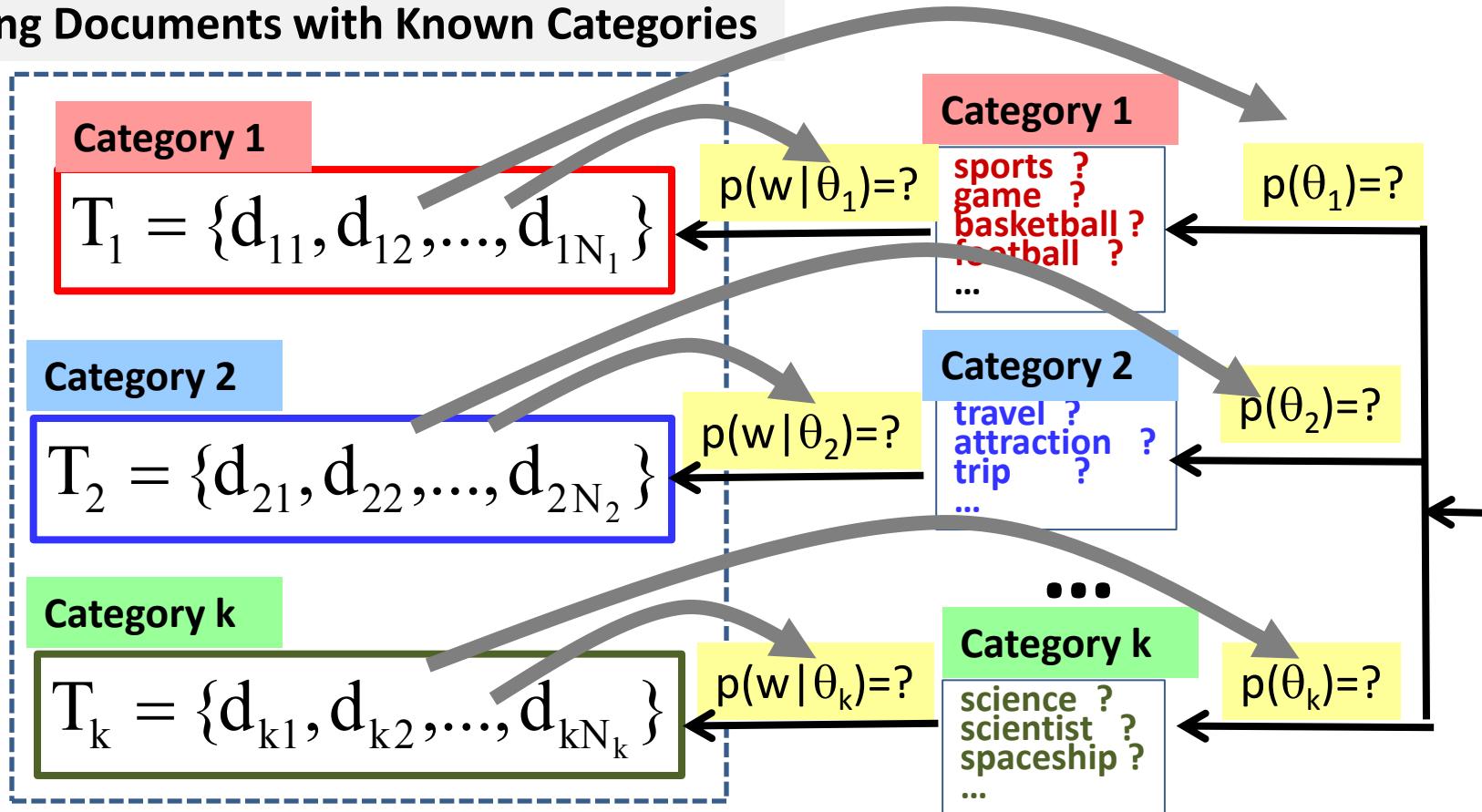
Learn from the Training Data

Training Documents with Known Categories



How to Estimate $p(w|\theta_i)$ and $p(\theta_i)$

Training Documents with Known Categories



Naïve Bayes Classifier: $p(\theta_i) = ?$ and $p(w | \theta_i) = ?$

Category 1

$$T_1 = \{d_{11}, d_{12}, \dots, d_{1N_1}\}$$

Category 2

$$T_2 = \{d_{21}, d_{22}, \dots, d_{2N_2}\}$$

Category k

$$T_k = \{d_{k1}, d_{k2}, \dots, d_{kN_k}\}$$

Which category is most popular?

$$\downarrow \\ p(\theta_i) = \frac{N_i}{\sum_{j=1}^k N_j} \propto |T_i|$$

$$p(w | \theta_i) = \frac{\sum_{j=1}^{N_i} c(w, d_{ij})}{\sum_{w' \in V} \sum_{j=1}^{N_i} c(w', d_{ij})} \propto c(w, T_i)$$

Which word is most frequent in category i?

What are the constraints on $p(\theta_i)$ and $p(w | \theta_i)$?

Smoothing in Naïve Bayes

- Why smoothing?
 - Address data sparseness (training data is small → zero prob.)
 - Incorporate prior knowledge
 - Achieve discriminative weighting (i.e., IDF weighting)
- How?

$$p(\theta_i) = \frac{N_i + \delta}{\sum_{j=1}^k N_j + k\delta} \quad \delta \geq 0$$

What if $\delta \rightarrow \infty$?

$p(w|\theta_B)$: background LM

$$p(w|\theta_i) = \frac{\sum_{j=1}^{N_i} c(w, d_{ij}) + \mu p(w|\theta_B)}{\sum_{w' \in V} \sum_{j=1}^{N_i} c(w', d_{ij}) + \mu} \quad \mu \geq 0$$

$p(w|\theta_B) = 1/|V|$?

What if $\mu \rightarrow \infty$?

Anatomy of Naïve Bayes Classifier

Two categories: θ_1 and θ_2

$$\text{score}(d) = \log \frac{p(\theta_1 | d)}{p(\theta_2 | d)} = \log \frac{p(\theta_1) \prod_{w \in V} p(w | \theta_1)^{c(w,d)}}{p(\theta_2) \prod_{w \in V} p(w | \theta_2)^{c(w,d)}}$$

$$= \log \frac{p(\theta_1)}{p(\theta_2)} + \sum_{w \in V} c(w,d) \log \frac{p(w | \theta_1)}{p(w | \theta_2)}$$

Category bias (β_0)
doesn't depend on d !

Sum over all words
(features $\{f_i\}$)

Feature value: $f_i = c(w,d)$

Weight on each
word (feature) β_i



$$d = (f_1, f_2, \dots, f_M), f_i \in \Re$$

Generalize

$$\text{score}(d) = \beta_0 + \sum_{i=1}^M f_i \beta_i \quad \beta_i \in \Re$$

= Logistic Regression!

Text Categorization: Discriminative Classifiers

Part 1

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Overview

- What is text categorization?
- Why text categorization?
- How to do text categorization?
 - Generative probabilistic models
 - **Discriminative approaches**
- How to evaluate categorization results?

Anatomy of Naïve Bayes Classifier

Two categories: θ_1 and θ_2

$$\text{score}(d) = \log \frac{p(\theta_1 | d)}{p(\theta_2 | d)} = \log \frac{p(\theta_1) \prod_{w \in V} p(w | \theta_1)^{c(w,d)}}{p(\theta_2) \prod_{w \in V} p(w | \theta_2)^{c(w,d)}}$$

$$= \log \frac{p(\theta_1)}{p(\theta_2)} + \sum_{w \in V} c(w,d) \log \frac{p(w | \theta_1)}{p(w | \theta_2)}$$

Category bias (β_0)
doesn't depend on d !

Sum over all words
(features $\{x_i\}$)

Feature value: $x_i = c(w,d)$

Weight on each
word (feature) β_i



Generalize

$$d = (x_1, x_2, \dots, x_M), \quad x_i \in \mathcal{R}$$

$$\text{score}(d) = \beta_0 + \sum_{i=1}^M x_i \beta_i \quad \beta_i \in \mathcal{R}$$

= Logistic Regression!

Discriminative Classifier 1: Logistic Regression

Binary Response Variable: $Y \in \{0,1\}$

Predictors: $X = (x_1, x_2, \dots, x_M)$, $x_i \in \Re$

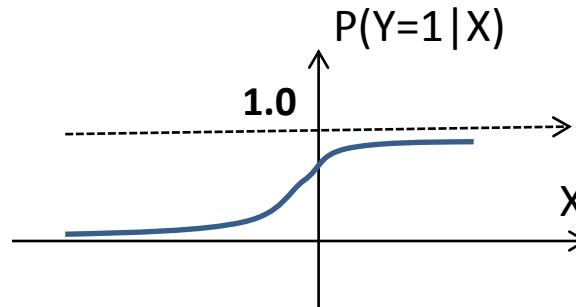
$$Y = \begin{cases} 1 & \text{category}(d) = \theta_1 \\ 0 & \text{category}(d) = \theta_2 \end{cases}$$

Modeling $p(Y|X)$ directly

Allow many other features than words!

$$\log \frac{p(\theta_1 | d)}{p(\theta_2 | d)} = \log \frac{p(Y=1 | X)}{p(Y=0 | X)} = \log \frac{p(Y=1 | X)}{1 - p(Y=1 | X)} = \beta_0 + \sum_{i=1}^M x_i \beta_i \quad \beta_i \in \Re$$

$$p(Y=1 | X) = \frac{e^{\beta_0 + \sum_{i=1}^M x_i \beta_i}}{e^{\beta_0 + \sum_{i=1}^M x_i \beta_i} + 1}$$



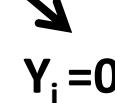
Estimation of Parameters

- Training Data: $T = \{(X_i, Y_i)\}, i=1, 2, \dots, |T|$
- Parameters: $\bar{\beta} = (\beta_0, \beta_1, \dots, \beta_M)$
- Conditional likelihood: $p(T | \bar{\beta}) = \prod_{i=1}^{|T|} p(Y = Y_i | X = X_i, \bar{\beta})$

$Y_i = 1$



$Y_i = 0$



$$p(Y = 1 | X) = \frac{e^{\beta_0 + \sum_{i=1}^M x_i \beta_i}}{e^{\beta_0 + \sum_{i=1}^M x_i \beta_i} + 1}$$

$$p(Y = 0 | X) = \frac{1}{e^{\beta_0 + \sum_{i=1}^M x_i \beta_i} + 1}$$

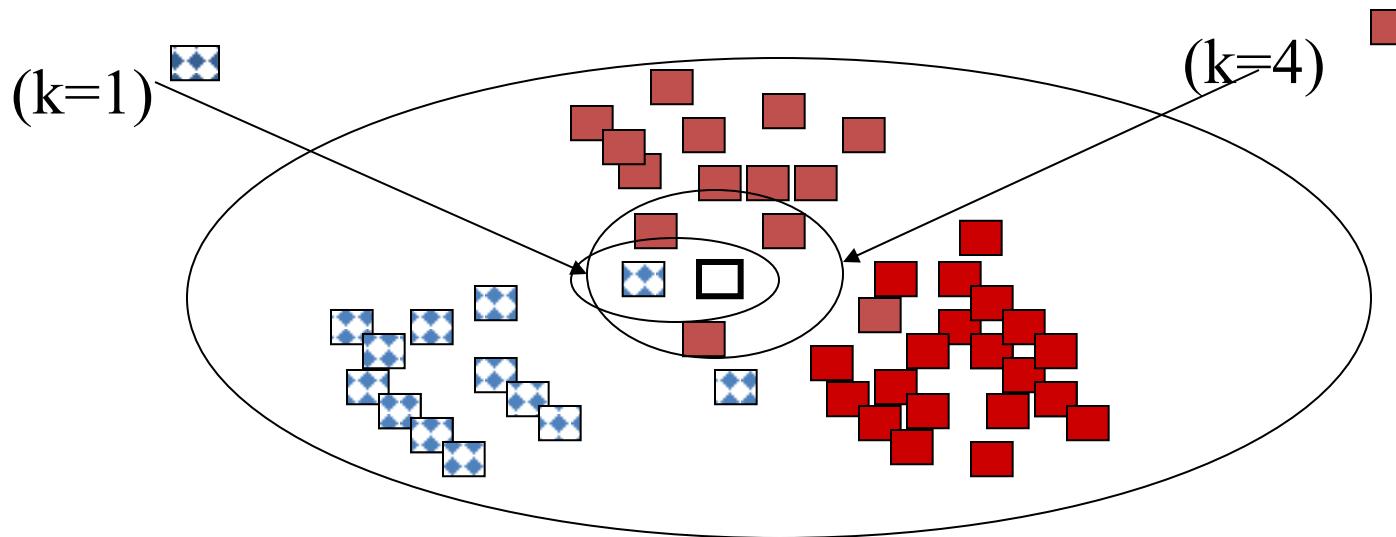
- Maximum Likelihood estimate $\bar{\beta}^* = \arg \max_{\bar{\beta}} p(T | \bar{\beta})$

Can be computed in many ways (e.g., Newton's method)

Discriminative Classifier 2: K-Nearest Neighbors (K-NN)

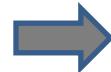
- Find k examples in the training set that are most similar to the text object to be classified (“neighbor” documents)
- Assign the category that is most common in these neighbor text objects (neighbors vote for the category)
- Can be improved by considering the distance of a neighbor (a closer neighbor has more influence)
- Can be regarded as a way to directly estimate the conditional probability of label given data instance, i.e., $p(Y|X)$
- Need a similarity function to measure similarity of two text objects

Illustration of K-NN Classifier

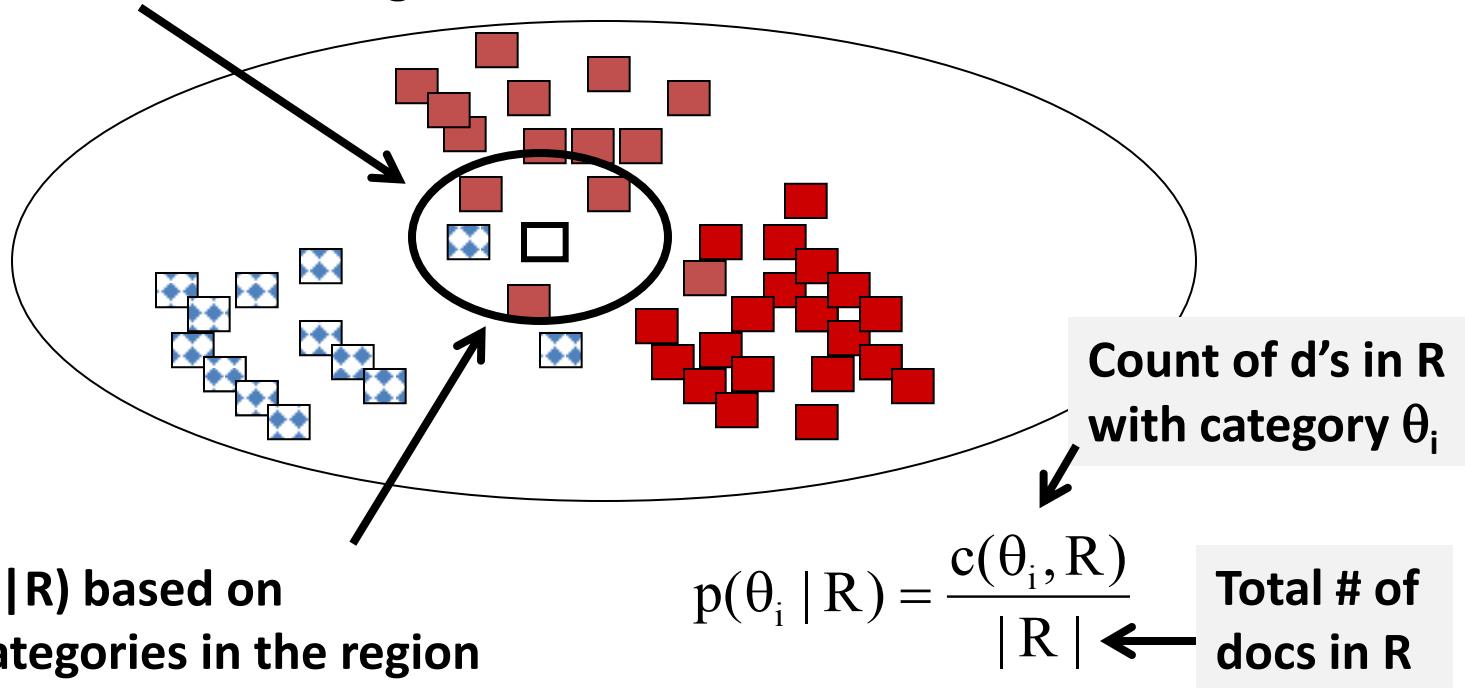


K-NN as an Estimate of $p(Y|X)$

Assume $p(\theta_i|d)$ is locally smooth, i.e.,
the same for all the d 's in this region R



$$p(\theta_i|d) = p(\theta_i|R)$$



Text Categorization: Discriminative Classifiers

Part 2

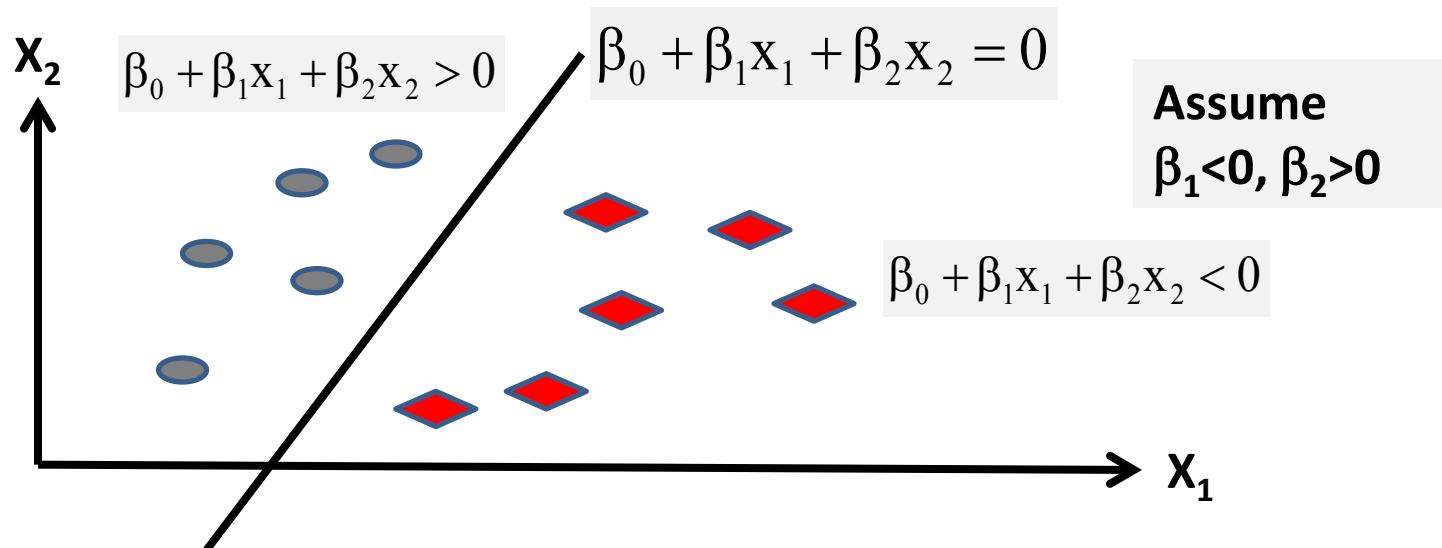
ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Discriminative Classifier 3: Support Vector Machine (SVM)

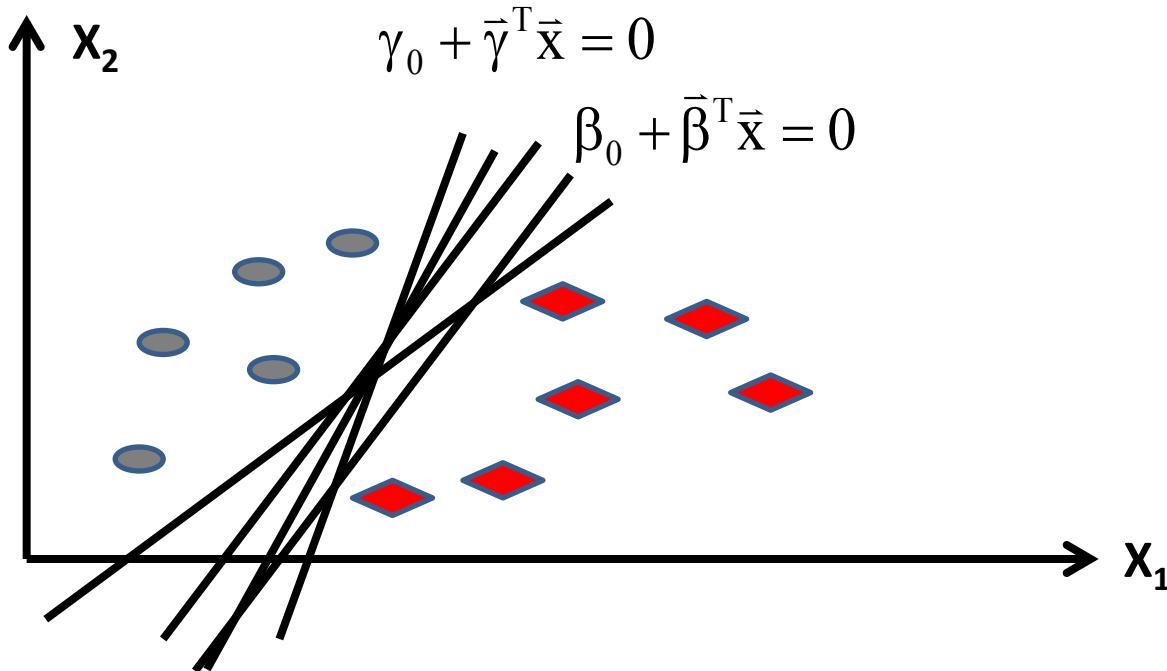
- Consider two categories: $\{\theta_1, \theta_2\}$

$f(X) \geq 0 \Rightarrow X$ is in category θ_1
 $f(X) < 0 \Rightarrow X$ is in category θ_2

- Use a linear separator $f(X) = \beta_0 + \sum_{i=1}^M x_i \beta_i \quad \beta_i \in \Re$



Which Linear Separator Is the Best?



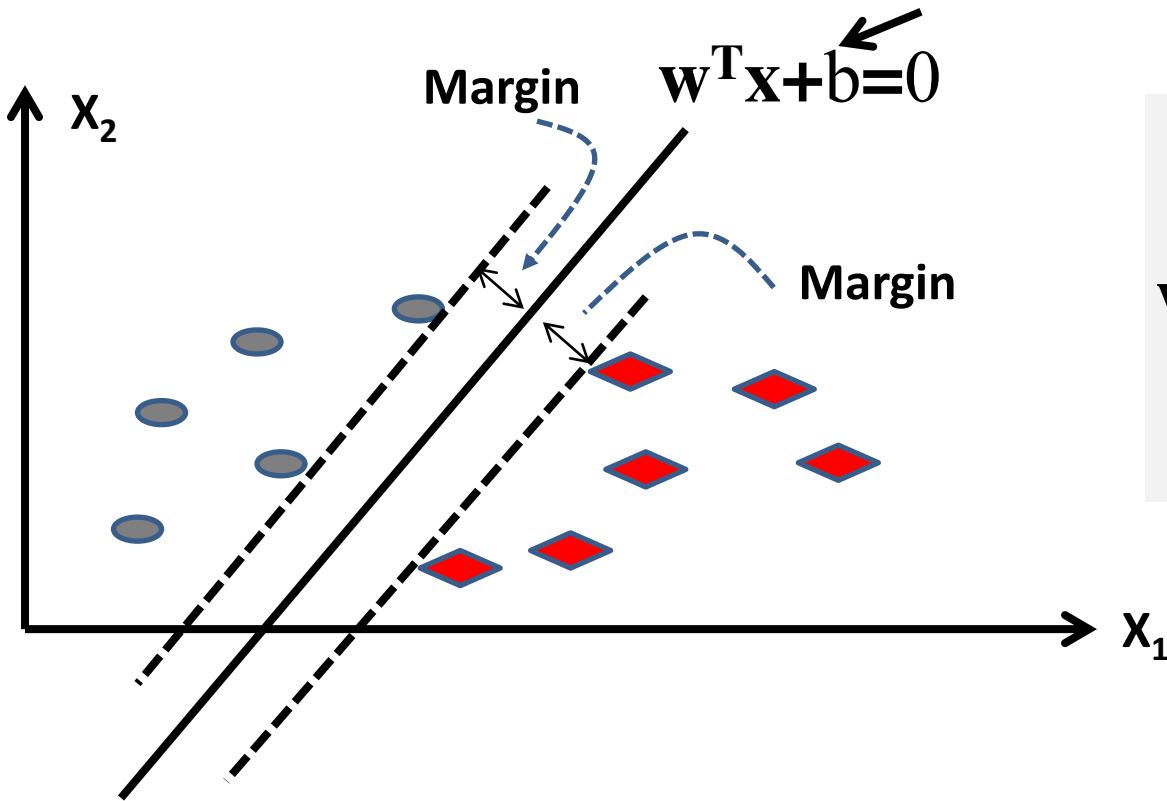
Best Separator = Maximize the Margin

Notation Change: $\beta \rightarrow w$; $\beta_0 \rightarrow b$

Bias constant

Feature
Weights

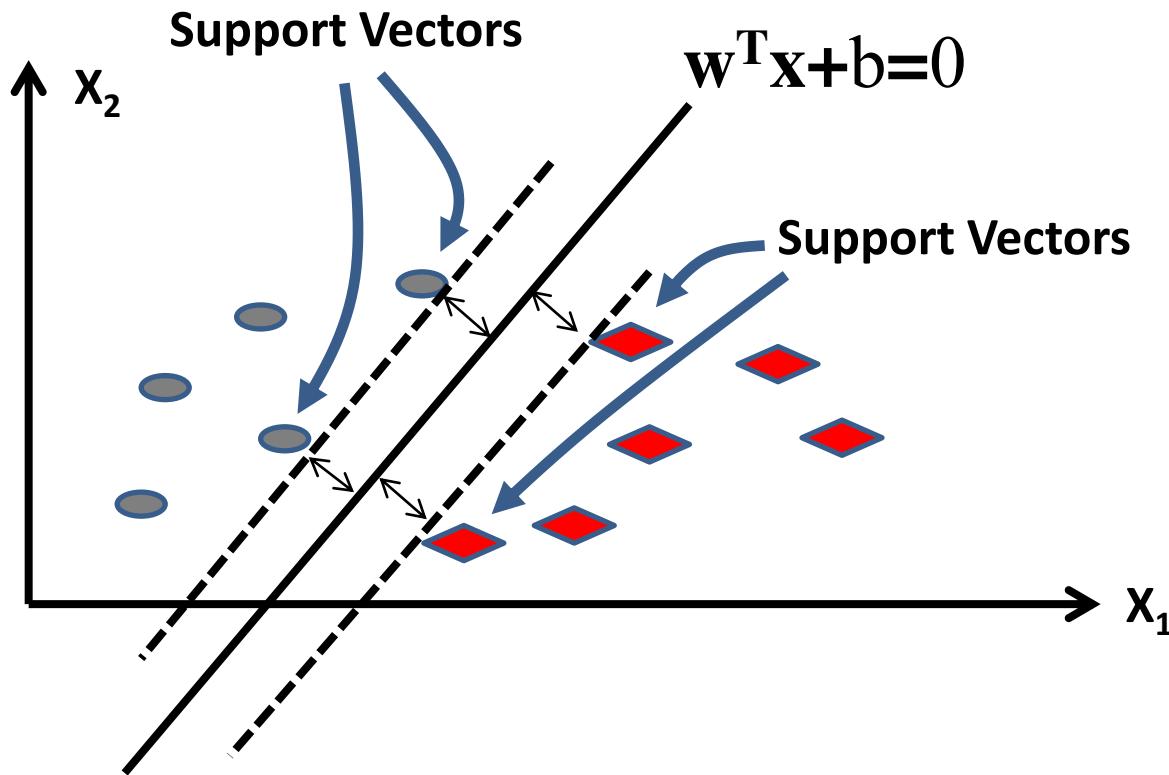
Feature Vector
(e.g., word counts)



$$w = \begin{pmatrix} w_1 \\ w_2 \\ \dots \\ w_M \end{pmatrix}$$

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_M \end{pmatrix}$$

Only the Support Vectors Matter



Linear SVM

Classifier: $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$

Parameters: \mathbf{w}, b

Training Data: $T = \{(\mathbf{x}_i, y_i)\}, i=1, \dots, |T|$. \mathbf{x}_i is a feature vector; $y_i \in \{-1, 1\}$

$f(\mathbf{X}) \geq 0 \Rightarrow \mathbf{X}$ is in category θ_1

$f(\mathbf{X}) < 0 \Rightarrow \mathbf{X}$ is in category θ_2

Goal 1: Correct labeling on training data:

If $y_i=1 \rightarrow \mathbf{w}^T \mathbf{x}_i + b \geq 1$

If $y_i=-1 \rightarrow \mathbf{w}^T \mathbf{x}_i + b \leq -1$



Constraint

$$\forall i, y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

Goal 2: Maximize margin

Large margin \Leftrightarrow Small $\mathbf{w}^T \mathbf{w}$

Objective

$$\text{Minimize } \Phi(\mathbf{w}) = \mathbf{w}^T \mathbf{w}$$

The optimization problem is quadratic programming with linear constraints

Linear SVM with Soft Margin

Classifier: $f(x) = w^T x + b > 0?$

Parameters: w, b

Training Data: $T = \{(x_i, y_i)\}, i=1, \dots, |T|.$

Added to allow training errors

Find w, b , and ξ_i to minimize $\Phi(w) = w^T w + C \sum_{i \in [1, |T|]} \xi_i$

Subject to $\forall i \in [1, |T|], y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$

$C > 0$ is a parameter to control the trade-off between minimizing the errors and maximizing the margin

The optimization problem is still quadratic programming with linear constraints

Summary of Text Categorization Methods

- Many methods are available, but no clear winner
 - All require effective feature representation (need domain knowledge)
 - It is useful to compare/combine multiple methods for a particular problem
- Most techniques rely on supervised machine learning and thus can be applied to **any** text categorization problem!
 - Humans annotate training data and design features
 - Computer optimizes the combination of features
 - Good performance requires 1) effective features and 2) plenty of training data
 - Performance is generally (much) more affected by the effectiveness of features than by the choice of a specific classifier

Summary of Text Categorization Methods (cont.)

- How to design effective features? (application-specific)
 - Analyze the categorization problem and exploit domain knowledge
 - Perform error analysis to obtain insights
 - Leverage machine learning techniques (e.g., feature selection, dimension reduction, deep learning)
- How to obtain “enough” training examples?
 - Low-quality (“pseudo”) training examples may be leveraged
 - Exploit unlabeled data (using semi-supervised learning techniques)
 - Domain adaptation/transfer learning (“borrow” training examples from a related domain/problem)

Suggested Reading

Manning, Chris D., Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2007.
(Chapters 13-15)

Text Categorization: Evaluation

Part 1

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Overview

- What is text categorization?
- Why text categorization?
- How to do text categorization?
 - Generative probabilistic models
 - Discriminative approaches
- **How to evaluate categorization results?**

General Evaluation Methodology

- Have humans to create a test collection where every document is tagged with the desired categories (“ground truth”)
- Generate categorization results using a system on the test collection
- Compare the system categorization decisions with the human-made categorization decisions and quantify their similarity (or equivalently difference)
 - The higher the similarity is, the better the results are
 - Similarity can be measured from different perspectives to understand the quality of results in detail (e.g., which category performs better?)
 - In general, different categorization mistakes may have a different cost that inevitably depends on specific applications, but it is okay not to consider such a cost variation for **relative comparison of methods**

Classification Accuracy (Percentage of Correct Decisions)

	c_1	c_2	c_3	...	c_k	
d_1	y(+)	y(-)	n(+)		n(+)	+/- human answer
d_2	y(-)	n(+)	y(+)		n(+)	(+ = correct; - = incorrect)
d_3	n(+)	n(+)	y(+)		n(+)	y/n system result
...						(y = yes; n = no)
d_N			

$$\text{Classification Accuracy} = \frac{\text{Total number of correct decisions}}{\text{Total number of decisions made}}$$

$$= \frac{\text{count}(y(+)) + \text{count}(n(-))}{kN}$$

Problems with Classification Accuracy

- Some decision errors are more serious than others
 - It may be more important to get the decisions right on some documents than others
 - It may be more important to get the decisions right on some categories than others
 - E.g., spam filtering: missing a legitimate email costs more than letting a spam go
- Problem with imbalanced test set
 - Skewed test set: 98% in category 1; 2% in category 2
 - Strong baseline: put all instances in category 1 → 98% accuracy!

Per-Document Evaluation

	c_1	c_2	c_3	...	c_k	How good are the decisions on d_i ?
d_1	y(+)	y(-)	n(+)		n(+)	
d_2	y(-)	n(+)	y(+)		n(+)	
d_3	n(+)	n(+)	y(+)		n(+)	

	System ("y")	System ("n")
Human (+)	True Positives TP	False Negatives FN
Human (-)	False Positives FP	True Negatives TN

Precision =
$$\frac{TP}{TP + FP}$$

Recall =
$$\frac{TP}{TP + FN}$$

Does the doc have all the categories it should have?

Per-Category Evaluation

	c_1	c_2	c_3	...
d_1	y(+)	y(-)	n(+)	
d_2	y(-)	n(+)	y(+)	
d_3	n(+)	n(+)	y(+)	

	c_k
	n(+)
	n(+)
	n(-)

How good are the decisions on c_i ?

When the system says “yes,”
how many are correct?

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

	System (“y”)	System (“n”)
Human (+)	True Positives TP	False Negatives FN
Human (-)	False Positives FP	True Negatives TN

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Has the category been assigned to
all the docs of this category?

Combine Precision and Recall: F-Measure

$$F_{\beta} = \frac{1}{\frac{\beta^2}{\beta^2+1} \frac{1}{R} + \frac{1}{\beta^2+1} \frac{1}{P}} = \frac{(\beta^2 + 1)P * R}{\beta^2 P + R}$$

$$F_1 = \frac{2PR}{P + R}$$

P: precision

R: recall

β : parameter (often set to 1)

Why not $0.5*P+0.5*R$?

What is R if the system says “y” for all category-doc pairs?

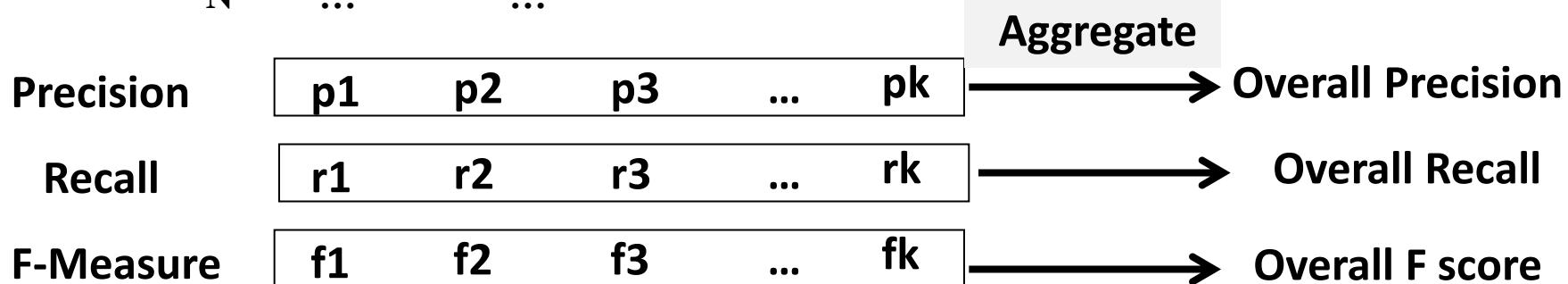
Text Categorization: Evaluation

Part 2

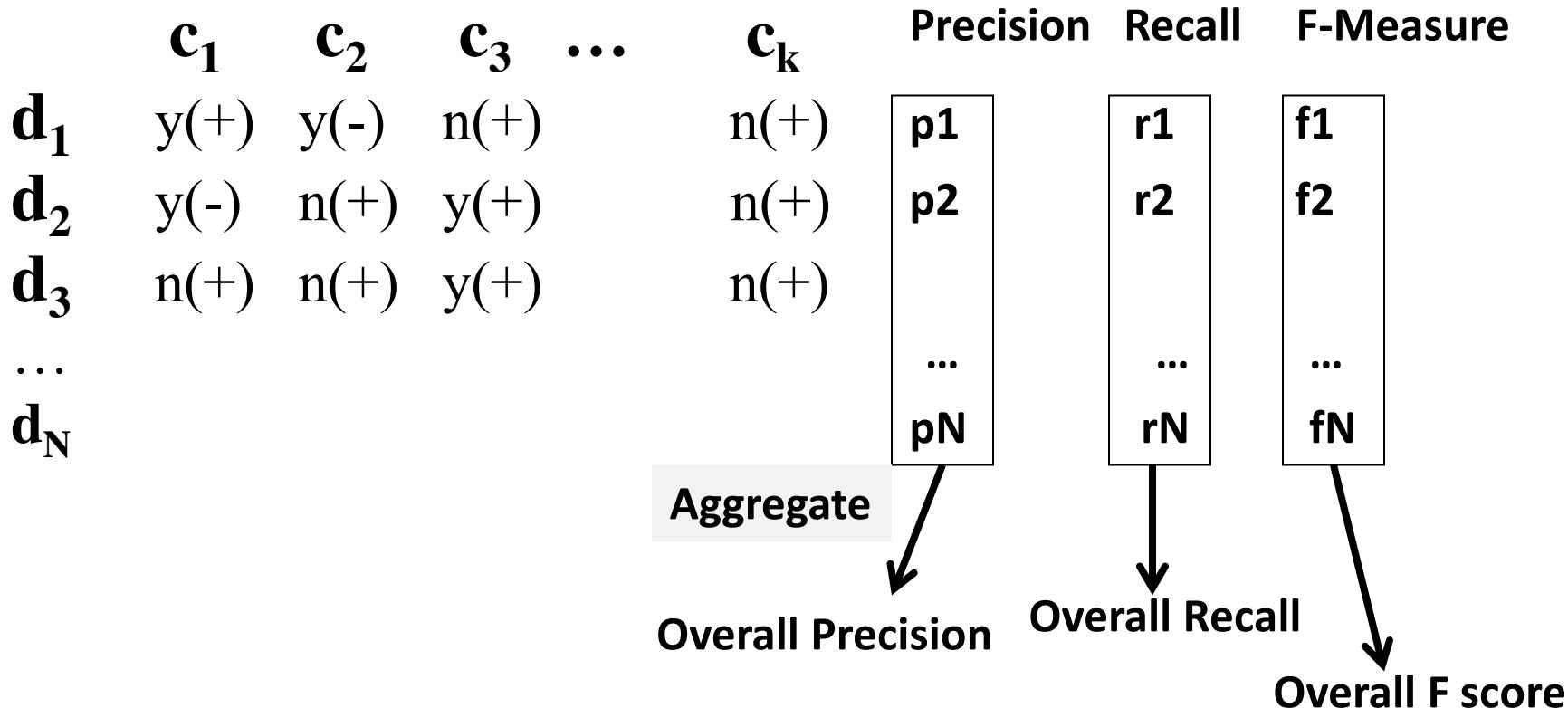
ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

(Macro) Average Over All the Categories

	c_1	c_2	c_3	...	c_k
d_1	y(+)	y(-)	n(+)		n(+)
d_2	y(-)	n(+)	y(+)		n(+)
d_3	n(+)	n(+)	y(+)		n(+)
...
d_N



(Macro) Average Over All the Documents



Micro-Averaging of Precision and Recall

	c_1	c_2	c_3	...	c_k
d_1	y(+)	y(-)	n(+)		n(+)
d_2	y(-)	n(+)	y(+)		n(+)
d_3	n(+)	n(+)	y(+)		n(+)
...					
d_N			

First pool all decisions,
then compute precision and recall



$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

	System ("y")	System ("n")
Human (+)	True Positives (TP)	False Negatives (FN)
Human (-)	False Positives(FP)	True Negatives(TN)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Sometimes Ranking Is More Appropriate

- The categorization results are often passed to a human for
 - further editing (e.g., correcting system mistakes on news categories)
 - prioritizing a task (e.g., routing an email to the right person for processing)
- In such cases, we can evaluate the results as a ranked list if the system can give scores for the decisions
 - E.g., discovery of spam emails (→ rank emails for the “spam” category)
 - Often more appropriate to frame the problem as a ranking problem instead of a categorization problem (e.g., ranking documents in a search engine)

Summary of Categorization Evaluation

- Evaluation is always very important, so get it right!
- Measures must reflect the **intended use** of the results for a particular application (e.g., spam filtering vs. news categorization)
 - Consider: How will the results be further processed (by a user)?
 - Ideally associate a different cost with each different decision error
- Commonly used measures for **relative comparison** of different methods:
 - Accuracy, precision, recall, F score
 - Variations: per-document, per-category, micro vs. macro averaging
- Sometimes **ranking** may be more appropriate

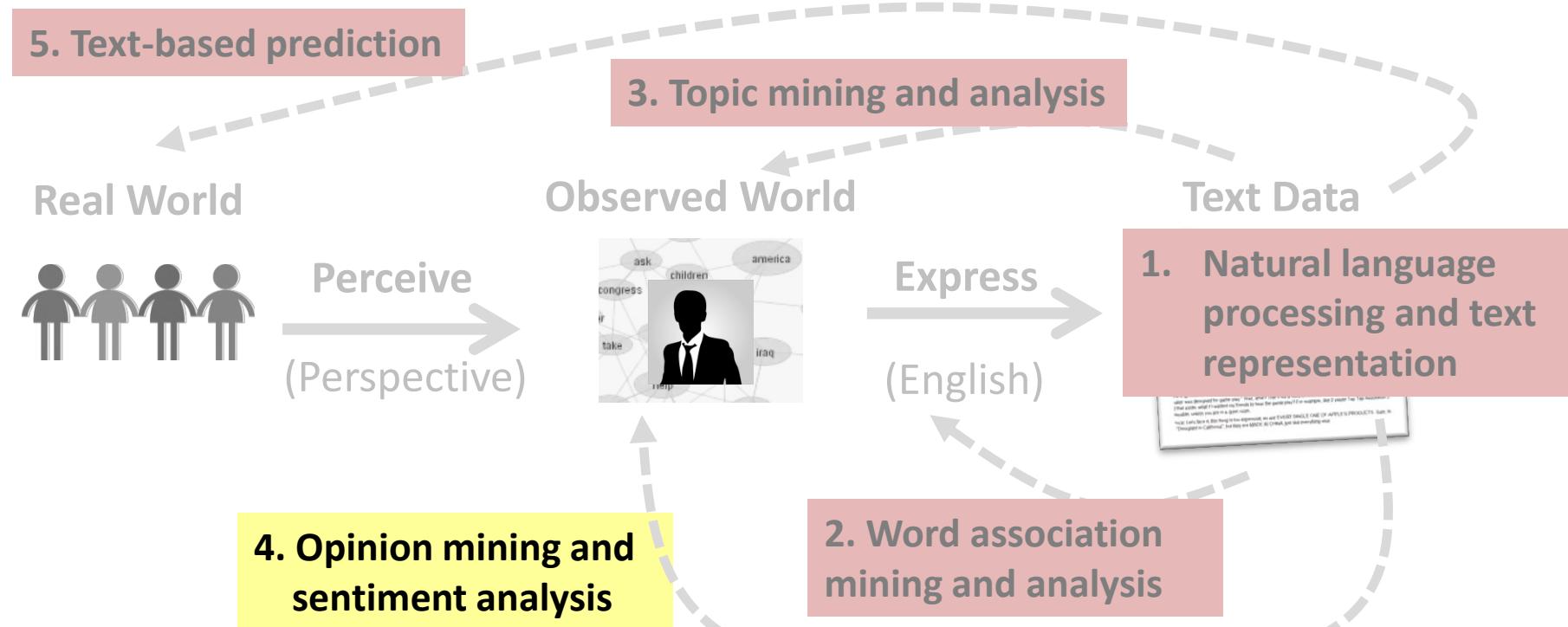
Suggested Reading

- Manning, Chris D., Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2007. (Chapters 13-15)
- Yang, Yiming. 1999. An Evaluation of Statistical Approaches to Text Categorization. *Inf. Retr.* 1, 1-2 (May 1999), 69-90. DOI=10.1023/A:1009982220290

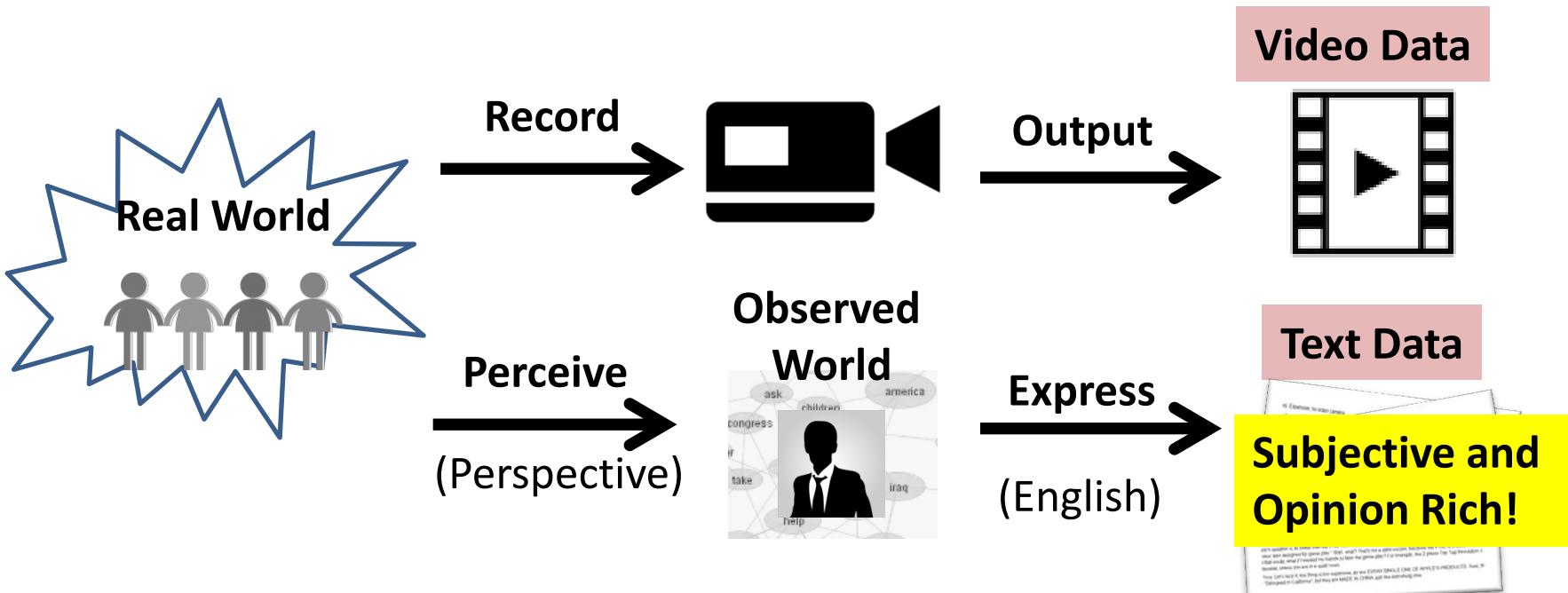
Opinion Mining and Sentiment Analysis: Motivation

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Opinion Mining and Sentiment Analysis: Motivation



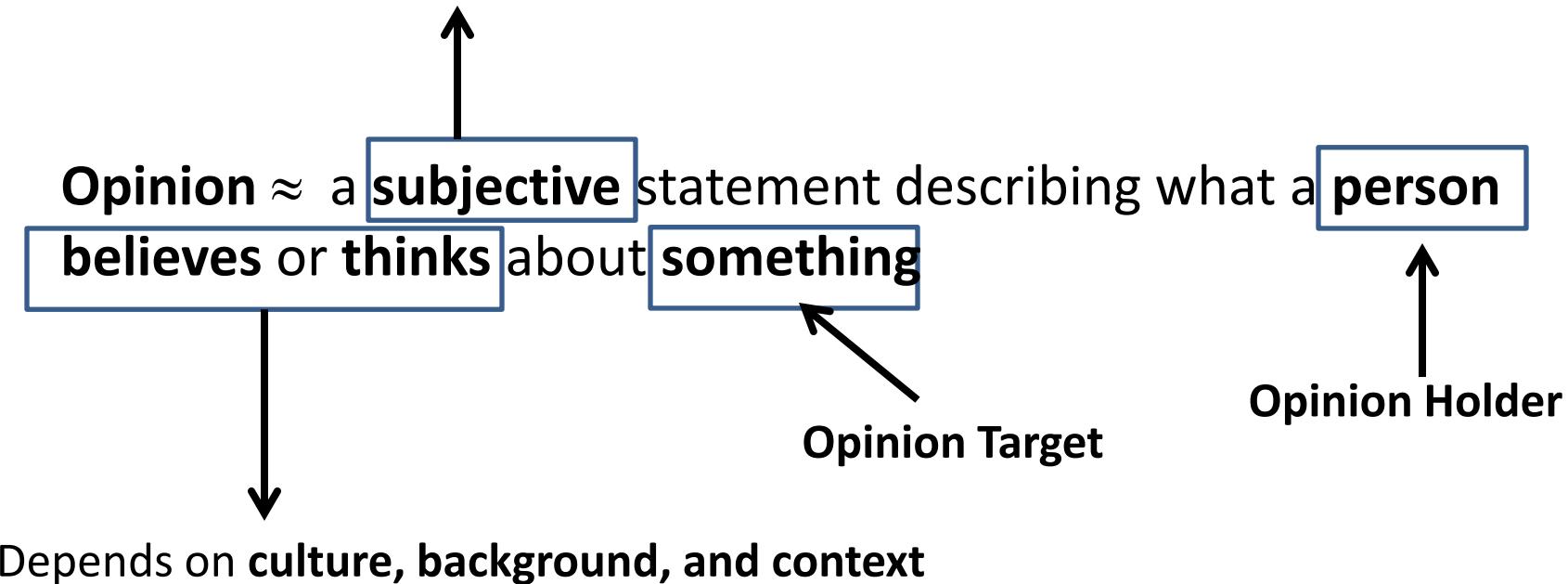
Objective vs. Subjective Sensors



How can we mine and analyze opinion buried in text?

What Is an Opinion?

Objective statement or Factual statement (can be proved right/wrong)



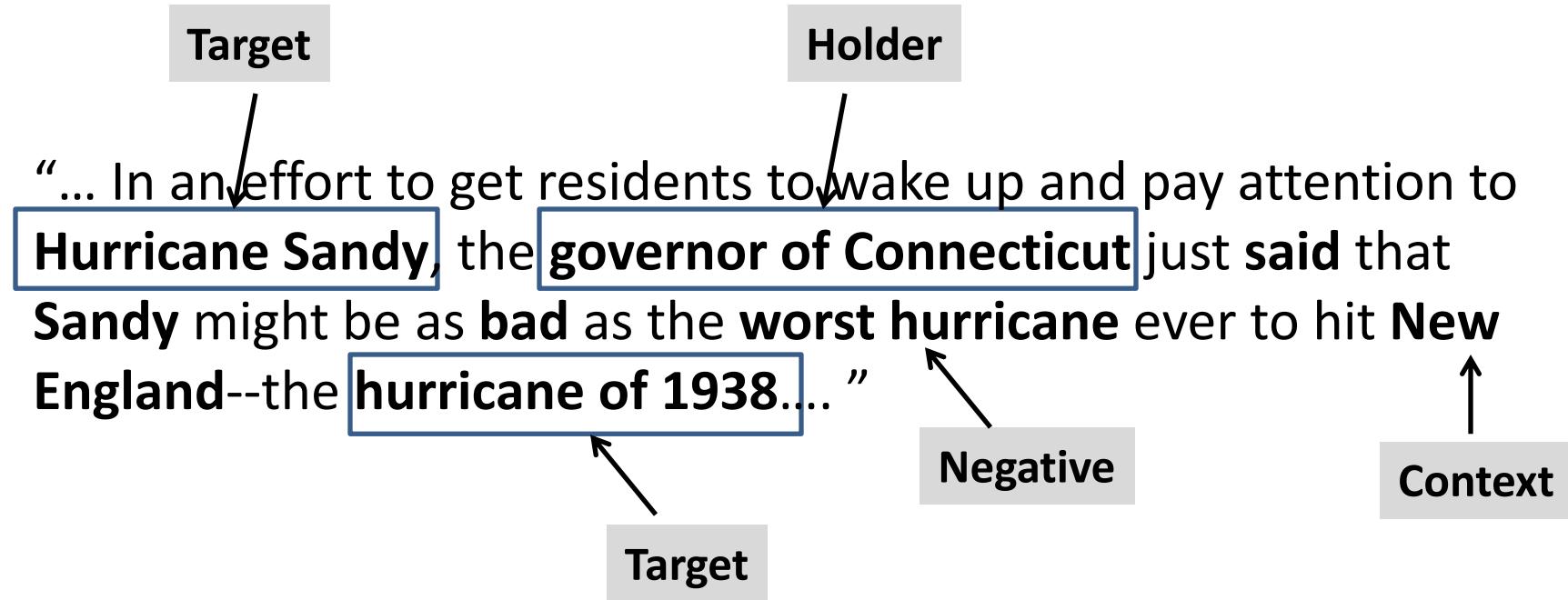
Opinion Representation

- Basic Opinion Representation
 - Opinion **holder**: Whose opinion is this?
 - Opinion **target**: What is this opinion about?
 - Opinion **content**: What exactly is the opinion?
- Enriched Opinion Representation
 - Opinion **context**: Under what situation (e.g., time, location) was the opinion expressed?
 - Opinion **sentiment**: What does the opinion tell us about the opinion holder's feeling (e.g., positive vs. negative)?

A Product Review (Explicit Holder and Target)

- Basic Opinion Representation
 - Opinion **holder**: Whose opinion is this? Reviewer X
 - Opinion **target**: What is this opinion about? Product: iPhone 6
 - Opinion **content**: What exactly is the opinion? Review Text
 - Enriched Opinion Representation
 - Opinion **context**: Under what situation (e.g., time, location) was the opinion expressed? Year = 2015
 - Opinion **sentiment**: What does the opinion tell us about the opinion holder's feeling (e.g., positive vs. negative)? Positive
- Relatively Easy to Mine and Analyze

A Sentence in News (Implicit Holder and Target)



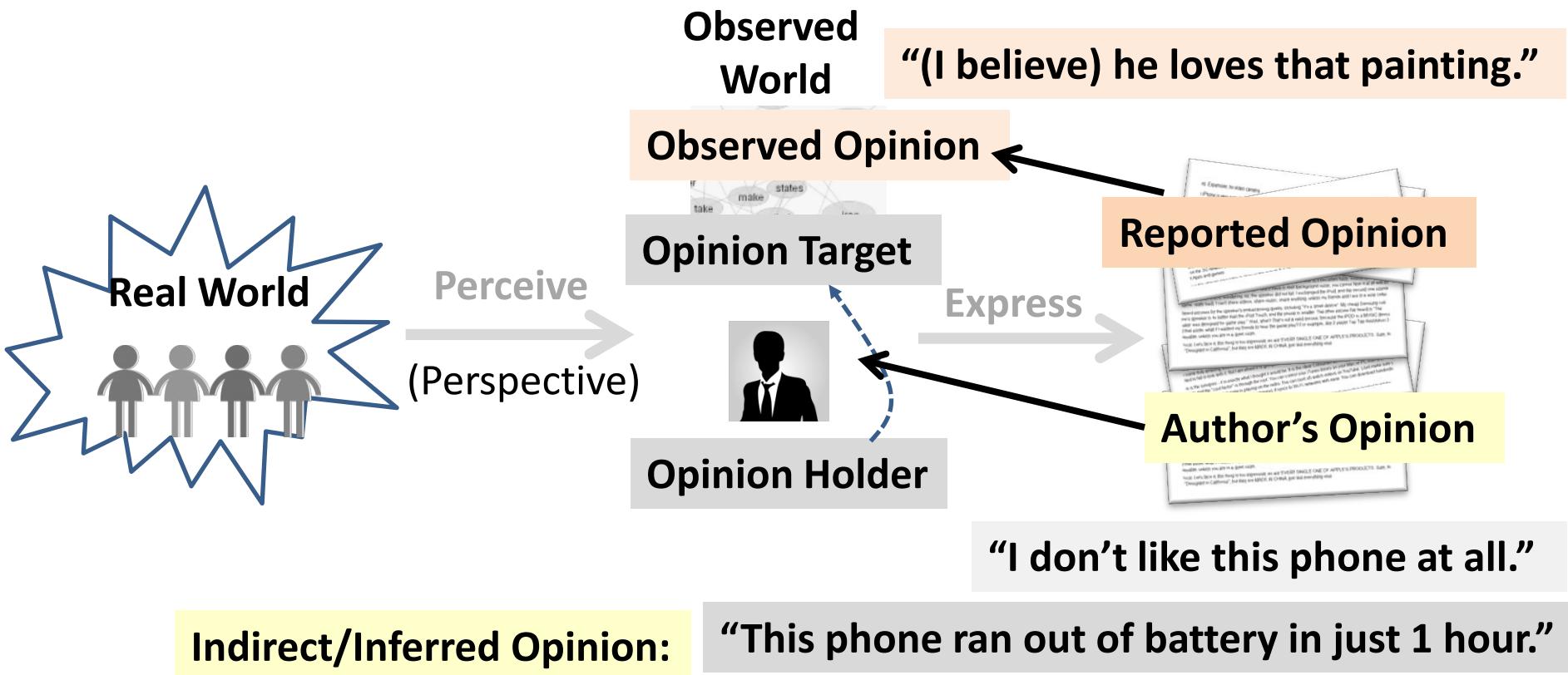
Harder to Mine and Analyze: Need deeper NLP

Source: Blodget, H. (2012, October 28). Hurricane Sandy is being compared to the worst hurricane ever to hit New England. *Business Insider*. *Business Insider*. Retrieved from <http://www.businessinsider.com/hurricane-sandy-vs-hurricane-of-1938-2012-10>.

Variations of Opinions

- **Opinion holder:** Individual vs. group
- **Opinion target:** One entity, a group of entities, one attribute of an entity, someone else's opinion, etc.
- **Opinion content:**
 - Surface variation: one sentence/phrase, a paragraph, a whole article
 - Sentiment/emotion variation: positive vs. negative, happy vs. sad, etc.
- **Opinion context**
 - Simple context: Different time, location, etc.
 - Complex context: Potentially includes the entire discourse context of an opinion

Different Kinds of Opinions in Text Data



The Task of Opinion Mining

Text Data



A Set of Opinion Representations

Opinion Holder

Opinion Target

Opinion Content

Opinion Context

Opinion
Sentiment

Often some elements of the representation are already known

Simplest Opinion Mining task(s)?

Why Opinion Mining?

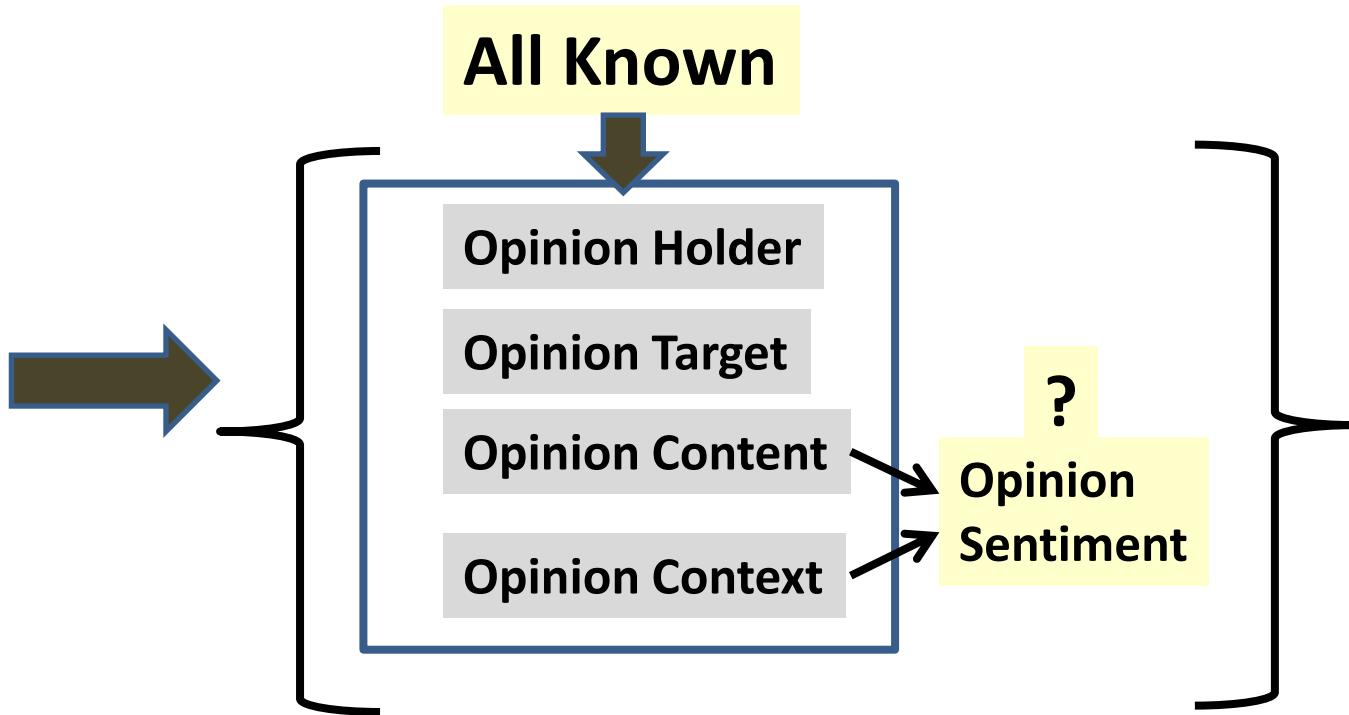
- **Decision Support**
 - Help consumers choose a product or service
 - Help voters decide whom to vote for
 - Help policy makers design new policy
- **Understand People**
 - Help understand people's preferences to better serve them (e.g., optimize a product search engine; optimize recommender systems)
 - Help with advertising (targeted advertising)
- **“Voluntary Survey” (humans as sensors; aggregated opinions)**
 - Business intelligence
 - Market research
 - Data-driven social science research
 - Gain advantage in **any** prediction (text-based prediction)

Opinion Mining and Sentiment Analysis: Sentiment Classification

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Sentiment Classification

Text Data



Sentiment Classification: Task Definition

- Input: An opinionated text object
- Output: A sentiment tag/label
 - Polarity analysis: e.g., categories = {positive, negative, neutral}, or categories ={5, 4, 3, 2, 1}
 - Emotion analysis (beyond polarity): e.g., categories ={happy, sad, fearful, angry, surprised, disgusted}
- A special case of text categorization! → Any text categorization method can be used to do sentiment classification
- Further improvement comes from
 - More sophisticated features appropriate for sentiment tagging
 - Consideration of the order of the categories (e.g., ordinal regression)

Commonly Used Text Features

- Character n-grams: can be mixed with different n's
 - General and robust to spelling/recognition errors, but less discriminative than words
- Word n-grams: can be mixed with different n's
 - Unigrams are often very effective, but not for sentiment analysis (e.g., "it's not good" or "it's not as good as")
 - Long n-grams are discriminative, but may cause overfitting
- POS tag n-grams: mixed n-gram with words and POS tags
 - E.g., "ADJECTIVE NOUN" or "great NOUN"

Commonly Used Text Features (cont.)

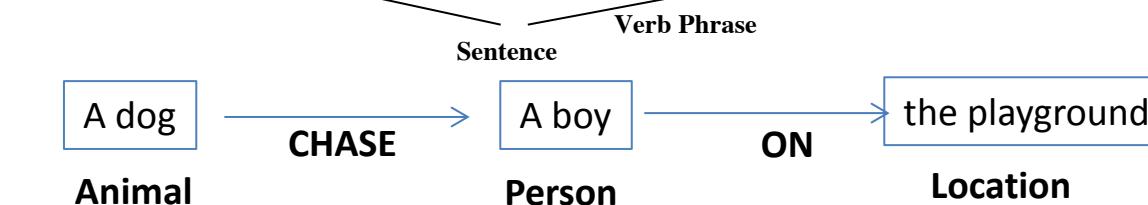
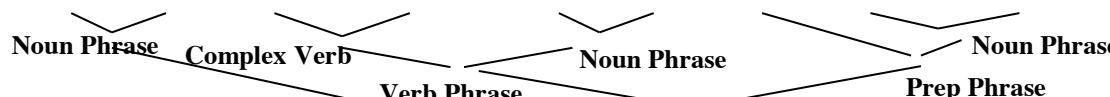
- Word classes
 - Syntactic (= POS tags)
 - Semantic Concept: e.g., thesaurus/ontology, recognized entities
 - Empirical word clusters (e.g., cluster of paradigmatically or syntagmatically related words)
- Frequent patterns in text (e.g., frequent word set; collocations)
 - More specific/discriminative than words
 - May generalize better than pure n-grams
- Parse tree-based (e.g., frequent subtrees, paths)
 - Even more discriminative, but need to avoid overfitting
- Pattern discovery algorithms are very useful for feature construction

NLP Enriches Text Representation with Complex Features

A dog is chasing a boy on the playground

A dog is chasing a boy on the playground

Det Noun Aux Verb Det Noun Prep Det Noun



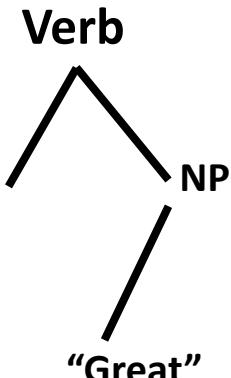
Dog(d1). Boy(b1). Playground(p1). Chasing(d1,b1,p1).

Speech Act = REQUEST

“great NOUN”

“Verb Adv Adj”

...



Feature Construction for Text Categorization

- Feature design affects categorization accuracy significantly
- A combination of machine learning, error analysis, and domain knowledge is most effective
 - Domain knowledge → seed features, feature space
 - Machine learning → feature selection, feature learning
 - Error analysis → feature validation
- NLP enriches text representation → enriches feature space (more likely overfitting!)
- Optimizing the tradeoff between **exhaustivity** and **specificity** is a major goal

high coverage (frequent)

discriminative (infrequent)

Sentiment Analysis: Ordinal Logistic Regression

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Motivation: Rating Prediction

- Input: An opinionated text document \mathbf{d}
- Output: Discrete rating $r \in \{1, 2, \dots, k\}$
- Using regular text categorization techniques
 - Doesn't consider the order and dependency of the categories
 - The features distinguishing $r=2$ from $r=1$ may be the same as those distinguishing $r=k$ from $r=k-1$ (e.g., positive words generally suggest a higher rating)
- Solution: Add order to a classifier (e.g., ordinal logistic regression)

Logistic Regression for Binary Sentiment Classification

Binary Response Variable: $Y \in \{0,1\}$ **Predictors:** $X = (x_1, x_2, \dots, x_M)$, $x_i \in \mathbb{R}$

$$Y = \begin{cases} 1 & X \text{ is POSITIVE} \\ 0 & X \text{ is NEGATIVE} \end{cases}$$

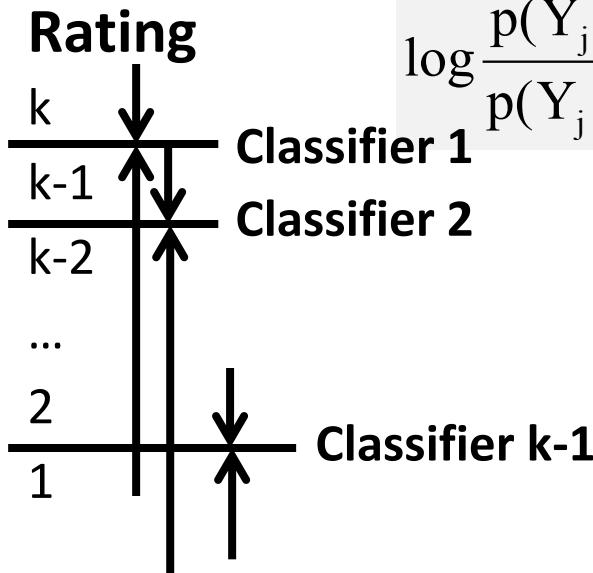
$$\log \frac{p(Y=1 | X)}{p(Y=0 | X)} = \log \frac{p(Y=1 | X)}{1 - p(Y=1 | X)} = \beta_0 + \sum_{i=1}^M x_i \beta_i \quad \beta_i \in \mathbb{R}$$

$$p(Y=1 | X) = \frac{e^{\beta_0 + \sum_{i=1}^M x_i \beta_i}}{e^{\beta_0 + \sum_{i=1}^M x_i \beta_i} + 1}$$

Logistic Regression for Multi-Level Ratings

$$Y_j = \begin{cases} 1 & \text{rating is } j \text{ or above} \\ 0 & \text{rating is lower than } j \end{cases}$$

Predictors: $X = (x_1, x_2, \dots, x_M)$, $x_i \in \mathcal{R}$
Rating: $r \in \{1, 2, \dots, k\}$



$$\log \frac{p(Y_j = 1 | X)}{p(Y_j = 0 | X)} = \log \frac{p(r \geq j | X)}{1 - p(r \geq j | X)} = \alpha_j + \sum_{i=1}^M x_i \beta_{ji} \quad \beta_{ji} \in \mathcal{R}$$

$$p(r \geq j | X) = \frac{e^{\alpha_j + \sum_{i=1}^M x_i \beta_{ji}}}{e^{\alpha_j + \sum_{i=1}^M x_i \beta_{ji}} + 1}$$

Rating Prediction with Multiple Logistic Regression Classifiers

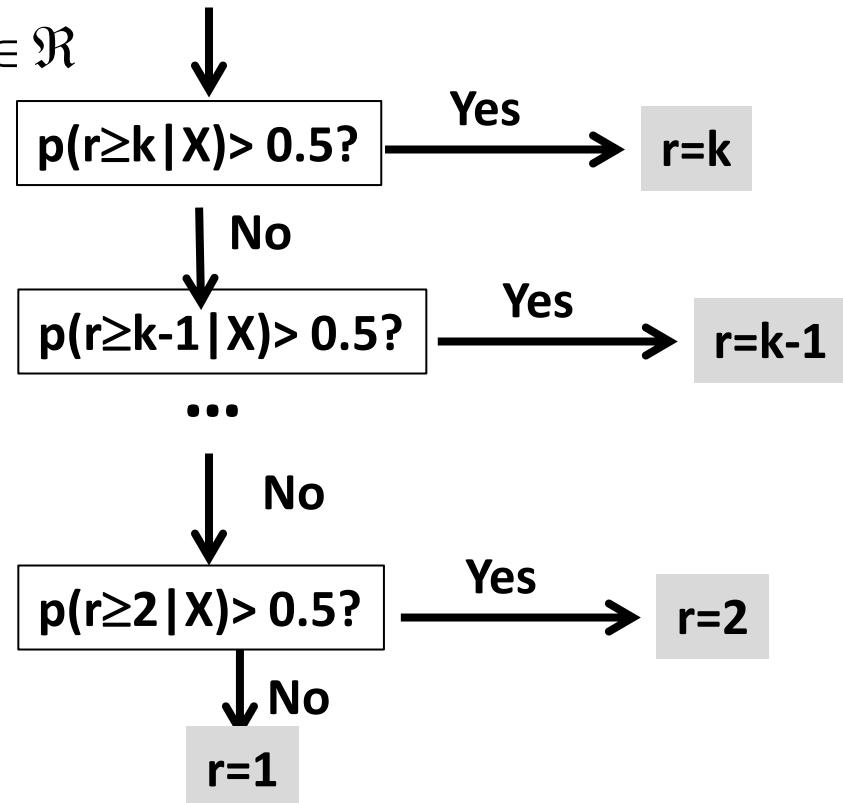
Text Object: $X = (x_1, x_2, \dots, x_M)$, $x_i \in \Re$

Rating: $r \in \{1, 2, \dots, k\}$

After training $k-1$
Logistic Regression Classifiers

$$p(r \geq j | X) = \frac{e^{\alpha_j + \sum_{i=1}^M x_i \beta_{ji}}}{e^{\alpha_j + \sum_{i=1}^M x_i \beta_{ji}} + 1}$$

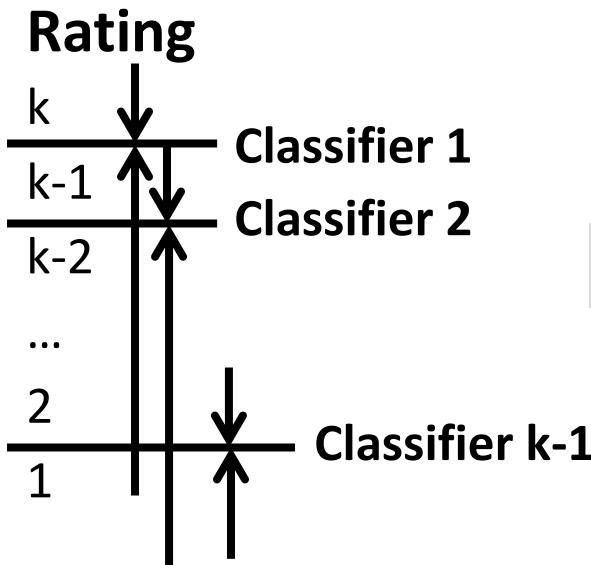
$j=k, k-1, \dots, 2$



Problems with $k-1$ Independent Classifiers?

$$\log \frac{p(Y_j = 1 | X)}{p(Y_j = 0 | X)} = \log \frac{p(r \geq j | X)}{1 - p(r \geq j | X)} = \alpha_j + \sum_{i=1}^M x_i \beta_{ji} \quad \beta_{ji} \in \Re$$

$$p(r \geq j | X) = \frac{e^{\alpha_j + \sum_{i=1}^M x_i \beta_{ji}}}{e^{\alpha_j + \sum_{i=1}^M x_i \beta_{ji}} + 1}$$



How many parameters are there in total? $(k-1)*(M+1)$

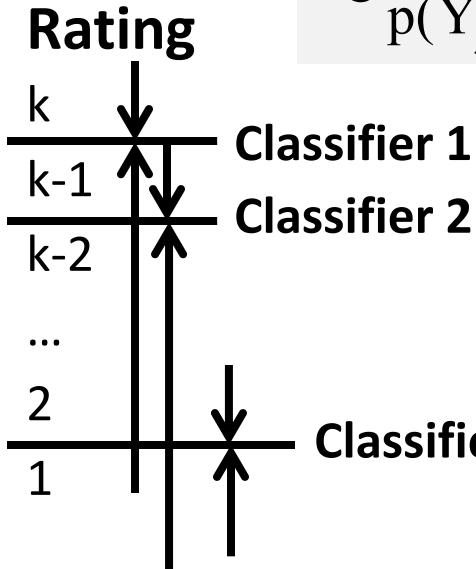
The $k-1$ classification problems are dependent.
The positive/negative features tend to be similar!

Ordinal Logistic Regression

Key Idea: $\forall i = 1, \dots, M, \forall j = 3, \dots, k, \beta_{ji} = \beta_{j-1i}$

→ Share training data → Reduce # of parameters

$$\log \frac{p(Y_j = 1 | X)}{p(Y_j = 0 | X)} = \log \frac{p(r \geq j | X)}{1 - p(r \geq j | X)} = \alpha_j + \sum_{i=1}^M x_i \beta_i \quad \beta_i \in \Re$$



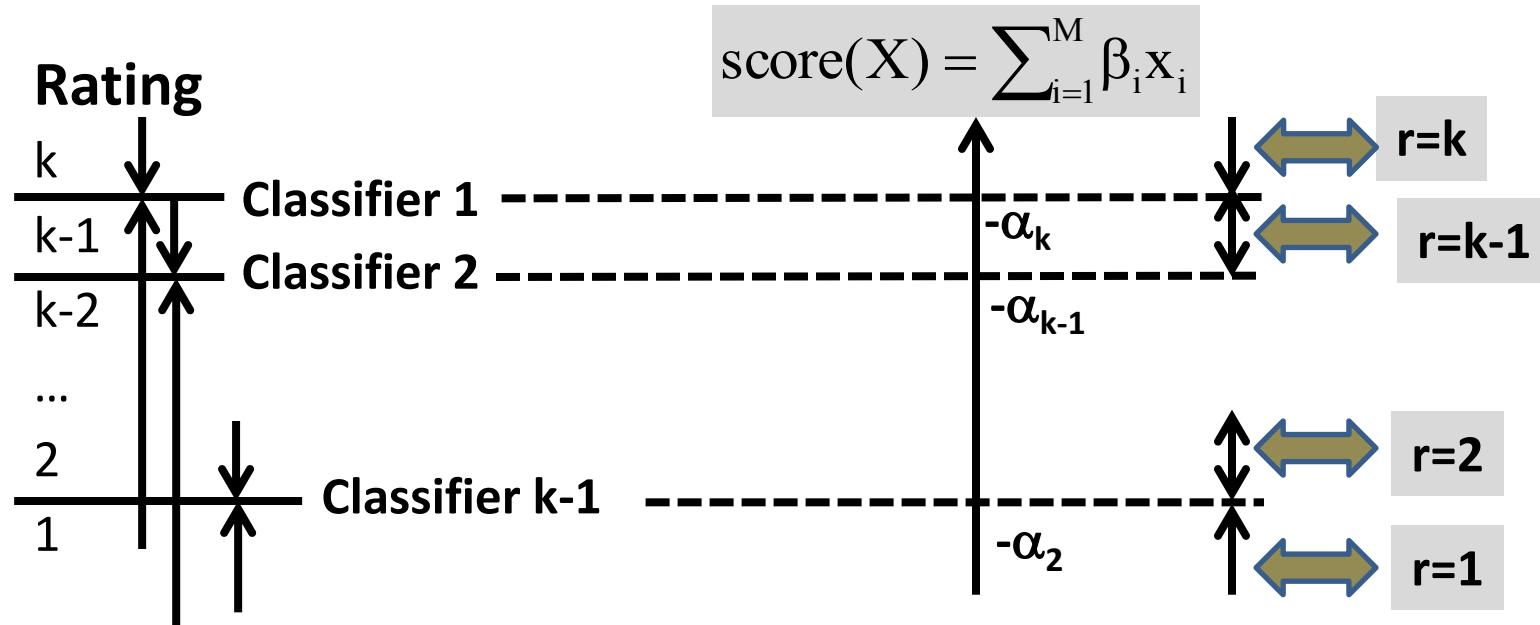
$$p(r \geq j | X) = \frac{e^{\alpha_j + \sum_{i=1}^M x_i \beta_i}}{e^{\alpha_j + \sum_{i=1}^M x_i \beta_i} + 1}$$

How many parameters are there in total?

M+k-1

Ordinal Logistic Regression: Rating Prediction

$$p(r \geq j | X) \geq 0.5 \Leftrightarrow \frac{e^{\alpha_j + \text{score}(X)}}{e^{\alpha_j + \text{score}(X)} + 1} \geq 0.5 \Leftrightarrow \text{score}(X) \geq -\alpha_j$$



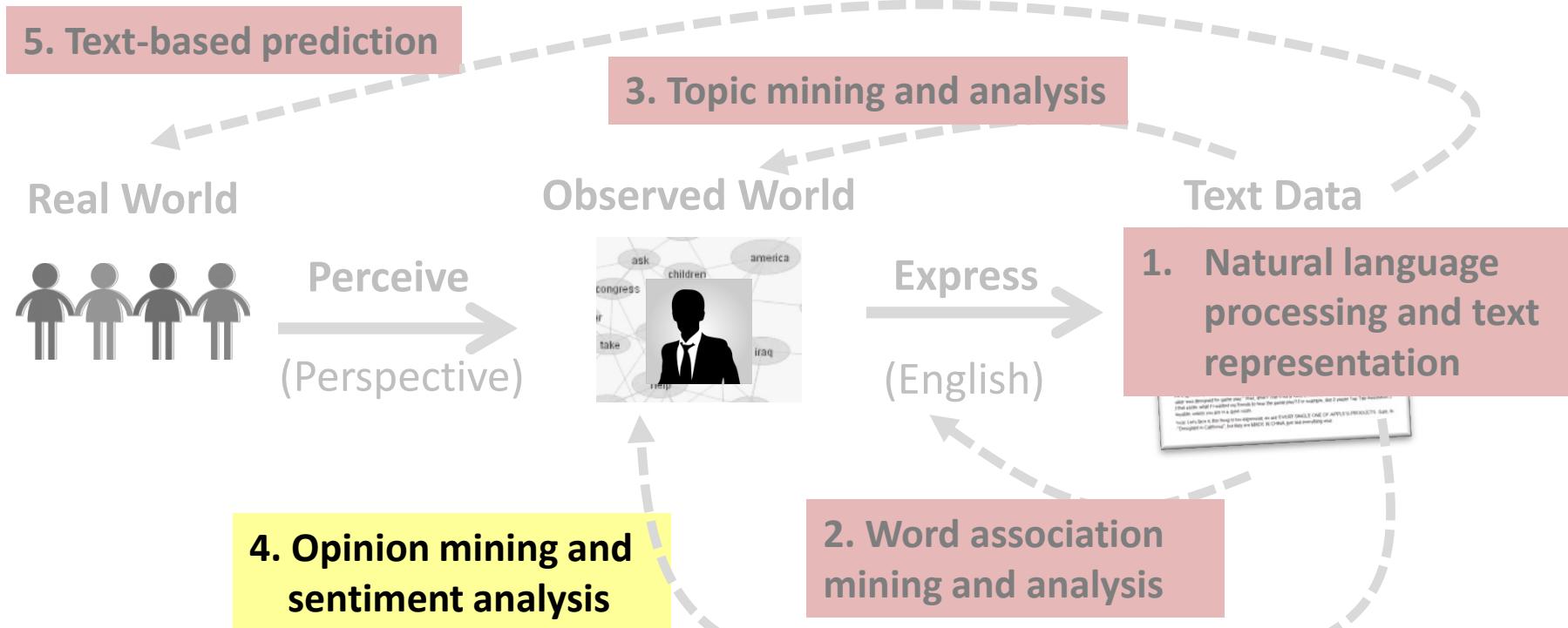
$r=j \Leftrightarrow \text{score} \in [-\alpha_j, -\alpha_{j+1})$, define $\alpha_1 = \infty$, $\alpha_{k+1} = -\infty$

Opinion Mining and Sentiment Analysis: Latent Aspect Rating Analysis

Part 1

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Opinion Mining and Sentiment Analysis: Latent Aspect Rating Analysis



Motivation

Hotel XYX

Reviewer 1: ★★★★☆

"Great location + spacious room = happy traveler"

Stayed for a weekend in July. Walked everywhere, enjoyed the comfy bed and quiet hallways....

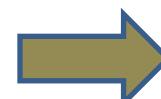


Value
Rooms
Location
Service

Reviewer 2: ★★★★☆

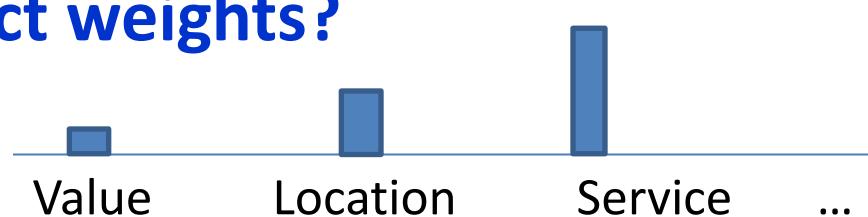
"Terrific service and gorgeous facility"

I stayed at the hotel with my young daughter for three nights June 17-20, 2010 and absolutely loved the hotel. The room was one of the nicest I've ever stayed in ...



Value
Rooms
Location
Service

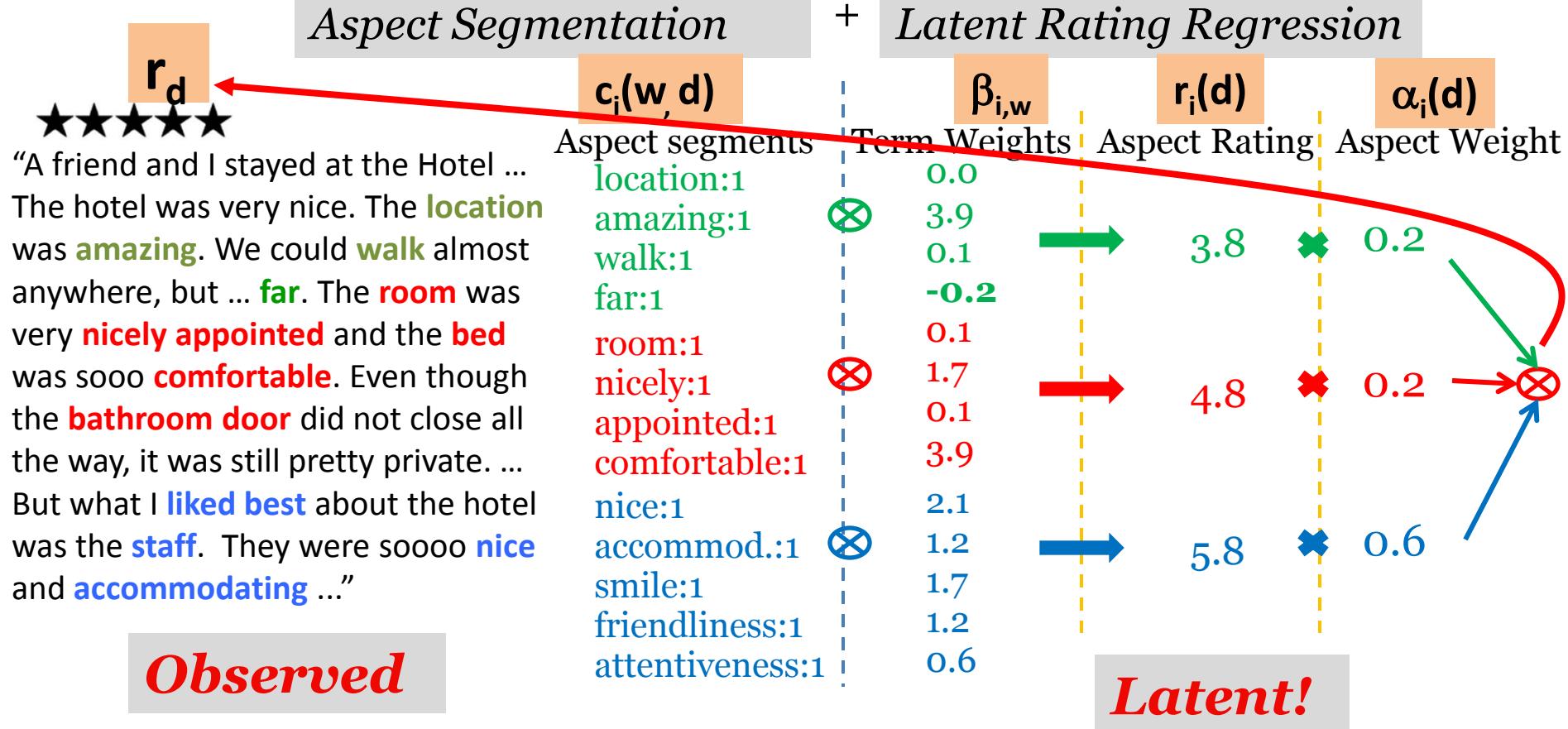
How to infer aspect weights?



Latent Aspect Rating Analysis [Wang et al. 10]

- Given a set of review articles about a topic with overall ratings
- Output
 - Major aspects commented on in the reviews
 - Ratings on each aspect
 - Relative weights placed on different aspects by reviewers
- Many applications
 - Opinion-based entity ranking
 - Aspect-level opinion summarization
 - Reviewer preference analysis
 - Personalized recommendation of products
 - ...

Solving LARA in Two Stages



Latent Rating Regression [Wang et al. 10]

- Data: a set of review documents with overall ratings: $C=\{(d, r_d)\}$
 - d is pre-segmented into k aspect segments
 - $c_i(w, d)$ = count of word w in aspect segment i (zero if w didn't occur)
- Model: predict rating based on d : $p(r_d | d)$

Overall Rating = Weighted Average of Aspect Ratings

$$r_d \sim N\left(\sum_{i=1}^k \alpha_i(d)r_i(d), \underline{\delta^2}\right),$$

Multivariate Gaussian Prior

$$\bar{\alpha}(d) \sim N(\bar{\mu}, \Sigma)$$

$$r_i(d) = \sum_{w \in V} c_i(w, d) \underline{\beta_{i,w}}$$

$$\beta_{i,w} \in \mathcal{R}$$

Aspect-Specific Sentiment of w

Aspect Rating = Sum of sentiment weights of words in the aspect

Latent Rating Regression (cont.)

- Maximum Likelihood Estimate

- Parameters: $\Lambda = (\{\beta_{i,w}\}, \bar{\mu}, \Sigma, \delta^2)$

- ML estimate: $\Lambda^* = \arg \max_{\Lambda} \prod_{d \in C} p(r_d | d, \Lambda)$

- Aspect Rating for aspect i

$$r_i(d) = \sum_{w \in V} c_i(w, d) \beta_{i,w}$$

$c_i(w, d) = 0$ for words
not occurring in
aspect segment i

- Aspect Weights: $\alpha_i(d)$ = weight on aspect i

$$\bar{\alpha}(d)^* = \arg \max_{\bar{\alpha}(d)} p(\bar{\alpha}(d) | \mu, \Sigma) p(r_d | d, \{\beta_{i,w}\}, \delta^2, \bar{\alpha}(d))$$

Maximum a Posteriori

Prior

Likelihood

Suggested Reading

- [Wang et al. 10] Hongning Wang, Yue Lu, and ChengXiang Zhai, Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of ACM KDD 2010*, pp. 783-792, 2010.
DOI=10.1145/1835804.1835903

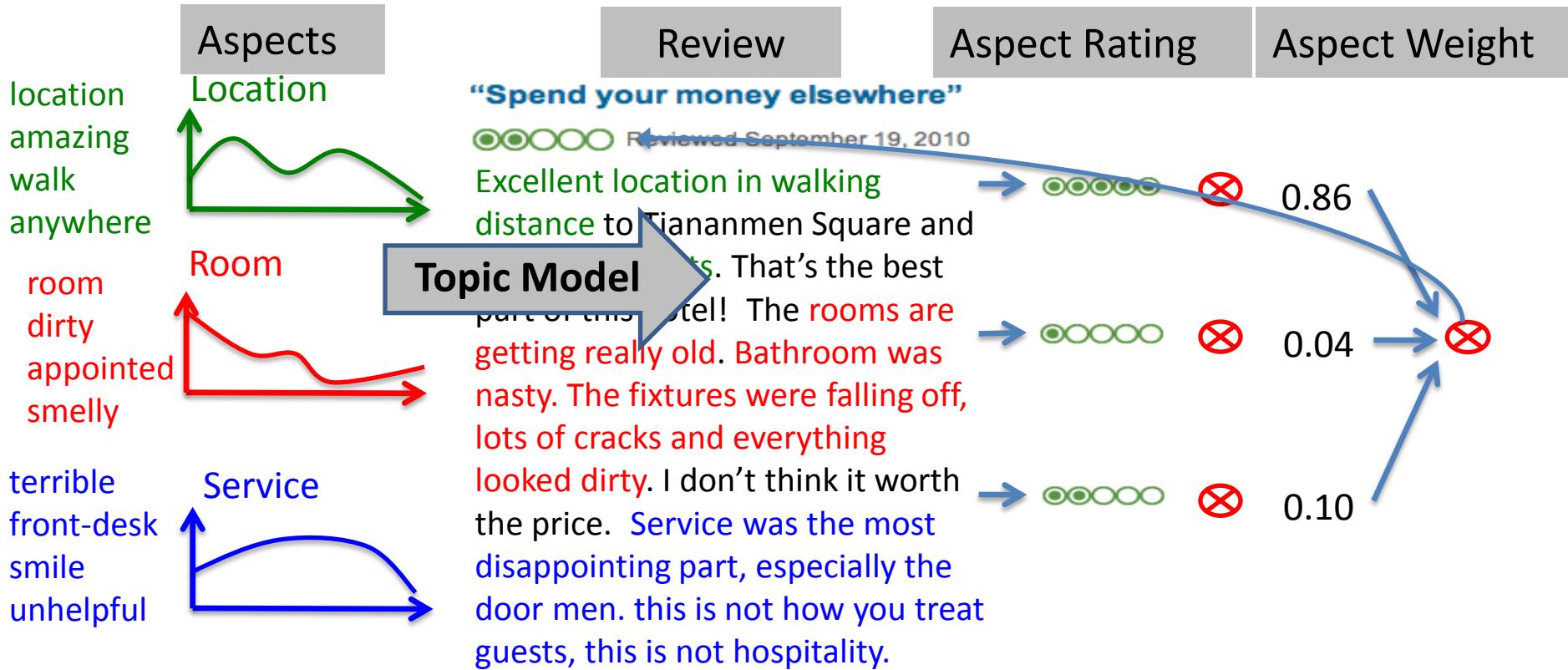
Opinion Mining and Sentiment Analysis: Latent Aspect Rating Analysis

Part 2

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

A Unified Generative Model for LARA [Wang et al. 11]

Any Entity



Sample Result 1: Rating Decomposition [Wang et al. 10]

- Hotels with the same overall rating but different aspect ratings

(All 5 Stars hotels, ground-truth in parenthesis)

<i>Hotel</i>	<i>Value</i>	<i>Room</i>	<i>Location</i>	<i>Cleanliness</i>
HOTEL 1	4.2(4.7)	3.8(3.1)	4.0(4.2)	4.1(4.2)
HOTEL 2	4.3(4.0)	3.9(3.3)	3.7(3.1)	4.2(4.7)
HOTEL 3	3.7(3.8)	4.4(3.8)	4.1(4.9)	4.5(4.8)

- Reveal detailed opinions at the aspect level

Sample Result 2: Comparison of Reviewers

[Wang et al. 10]

- Per-Reviewer Analysis
 - Different reviewers' ratings on the same hotel

<i>Reviewer</i>	<i>Value</i>	<i>Room</i>	<i>Location</i>	<i>Cleanliness</i>
Reviewer 1	3.7(4.0)	3.5(4.0)	3.7(4.0)	5.8(5.0)
Reviewer 2	5.0(5.0)	3.0(3.0)	5.0(4.0)	3.5(4.0)

- Reveal differences in opinions of different reviewers

Sample Result 3: Aspect-Specific Sentiment Lexicon

[Wang et al. 10]

<i>Value</i>	<i>Rooms</i>	<i>Location</i>	<i>Cleanliness</i>
resort 22.80	view 28.05	restaurant 24.47	clean 55.35
value 19.64	comfortable 23.15	walk 18.89	smell 14.38
excellent 19.54	modern 15.82	bus 14.32	linen 14.25
worth 19.20	quiet 15.37	beach 14.11	maintain 13.51
<i>bad</i> -24.09	<i>carpet</i> -9.88	<i>wall</i> -11.70	<i>smelly</i> -0.53
<i>money</i> -11.02	<i>smell</i> -8.83	<i>bad</i> -5.40	<i>urine</i> -0.43
<i>terrible</i> -10.01	<i>dirty</i> -7.85	<i>road</i> -2.90	<i>filthy</i> -0.42
<i>overprice</i> -9.06	<i>stain</i> -5.85	<i>website</i> -1.67	<i>dingy</i> -0.38

Learn sentimental information directly from the data.

Sample Result 4: Validating Preference Weights [Wang et al. 10]

Top-10: Reviewers with the highest Val/X ratio (emphasize “value”)

Bot-10: Reviewers with the lowest Val/X ratio (emphasize a non-value aspect)

<i>City</i>	<i>Avg. Price</i>	<i>Group</i>	<i>Val/Loc</i>	<i>Val/Rm</i>	<i>Val/Ser</i>
Amsterdam	241.6	top-10	190.7	214.9	221.1
		bot-10	270.8	333.9	236.2
San Francisco	261.3	top-10	214.5	249.0	225.3
		bot-10	321.1	311.1	311.4
Florence	272.1	top-10	269.4	248.9	220.3
		bot-10	298.9	293.4	292.6

Higher!

Application 1: Rated Aspect Summarization

Aspect	Summary	Rating
Value	Truly unique character and a great location at a reasonable price Hotel Max was an excellent choice for our recent three night stay in Seattle.	3.1
	Overall not a negative experience; however, considering that the hotel industry is very much in the impressing business, there was a lot of room for improvement.	1.7
Location	The location, a short walk to downtown and Pike Place market, made the hotel a good choice.	3.7
	When you visit a big metropolitan city, be prepared to hear a little traffic outside!	1.2
Business Service	You can pay for wireless by the day or use the complimentary Internet in the business center behind the lobby, though.	2.7
	My only complaint is the daily charge for Internet access when you can pretty much connect to wireless on the streets anymore.	0.9

Application 2: Discover Consumer Preferences

[Wang et al. 2011]

- Amazon reviews: No guidance

Table 2: Topical Aspects Learned on MP3 Reviews

Low Overall Ratings			High Overall Ratings		
unit	jack	service	files	player	vision
usb	headphone	charge	format	music	video
battery	warranty	problem	included	download	player
charger	replacement	support	easy	headphones	quality
reset	problem	hours	convert	button	great
time	player	months	mp3	set	product
hours	back	weeks	videos	hours	sound
work	months	back	file	buds	radio
thing	buy	customer	wall	volume	accessory
wall	amazon	time	hours	ear	fm

battery life accessory service file format volume video

Application 3: User Rating Behavior Analysis

[Wang et al. 10]

	<i>Expensive Hotel</i>	<i>Cheap Hotel</i>	
	5 Stars	3 Stars	5 Stars
Value	0.134	0.148	0.171
Room	0.098	0.162	0.126
Location	0.171	0.074	0.161
Cleanliness	0.081	0.163	0.116
Service	0.251	0.101	0.101
			0.049

People like expensive hotels because of good service.

People like cheap hotels because of good value.

Application 4: Personalized Ranking of Entities

[Wang et al. 10]

Query: 0.9 value
0.1 others

Non-personalized



Personalized



(Query-specific)

	Hotel	Overall Rating	Price	Location
Approach 1	Majestic Colonial	5.0	339	Punta Cana
	Agua Resort	5.0	753	Punta Cana
	Majestic Elegance	5.0	537	Punta Cana
	Grand Palladium	5.0	277	Punta Cana
	Iberostar	5.0	157	Punta Cana
Approach 2	Elan Hotel Modern	5.0	216	Los Angeles
	Marriott San Juan Resort	4.0	354	San Juan
	Punta Cana Club	5.0	409	Punta Cana
	Comfort Inn	5.0	155	Boston
	Hotel Commonwealth	4.5	313	Boston

Summary of Opinion Mining

- Very important with a lot of applications!
- Sentiment analysis can be done using text categorization techniques
 - With enriched feature representation
 - With consideration of ordering of the categories
- Generative models are powerful for mining latent user preferences
- Most approaches were proposed for product reviews
- Opinion mining from news and social media remains challenging

Suggested Reading

- Bing Liu, *Sentiment analysis and opinion mining*, Morgan & Claypool Publishers, 2012.
- Bo Pang and Lillian Lee, Opinion mining and sentiment analysis, *Foundations and Trends in Information Retrieval* 2(1-2), pp. 1–135, 2008.
- Hongning Wang, Yue Lu, and ChengXiang Zhai, Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of ACM KDD 2010*, pp. 783-792, 2010. DOI=10.1145/1835804.1835903
- Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of ACM KDD 2011*, pp. 618-626. DOI=10.1145/2020408.2020505

Text-Based Prediction

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Text-Based Prediction

5. Text-based prediction

Real World



Perceive
(Perspective)

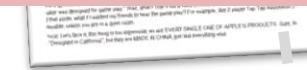
Observed World



3. Topic mining and analysis

Text Data

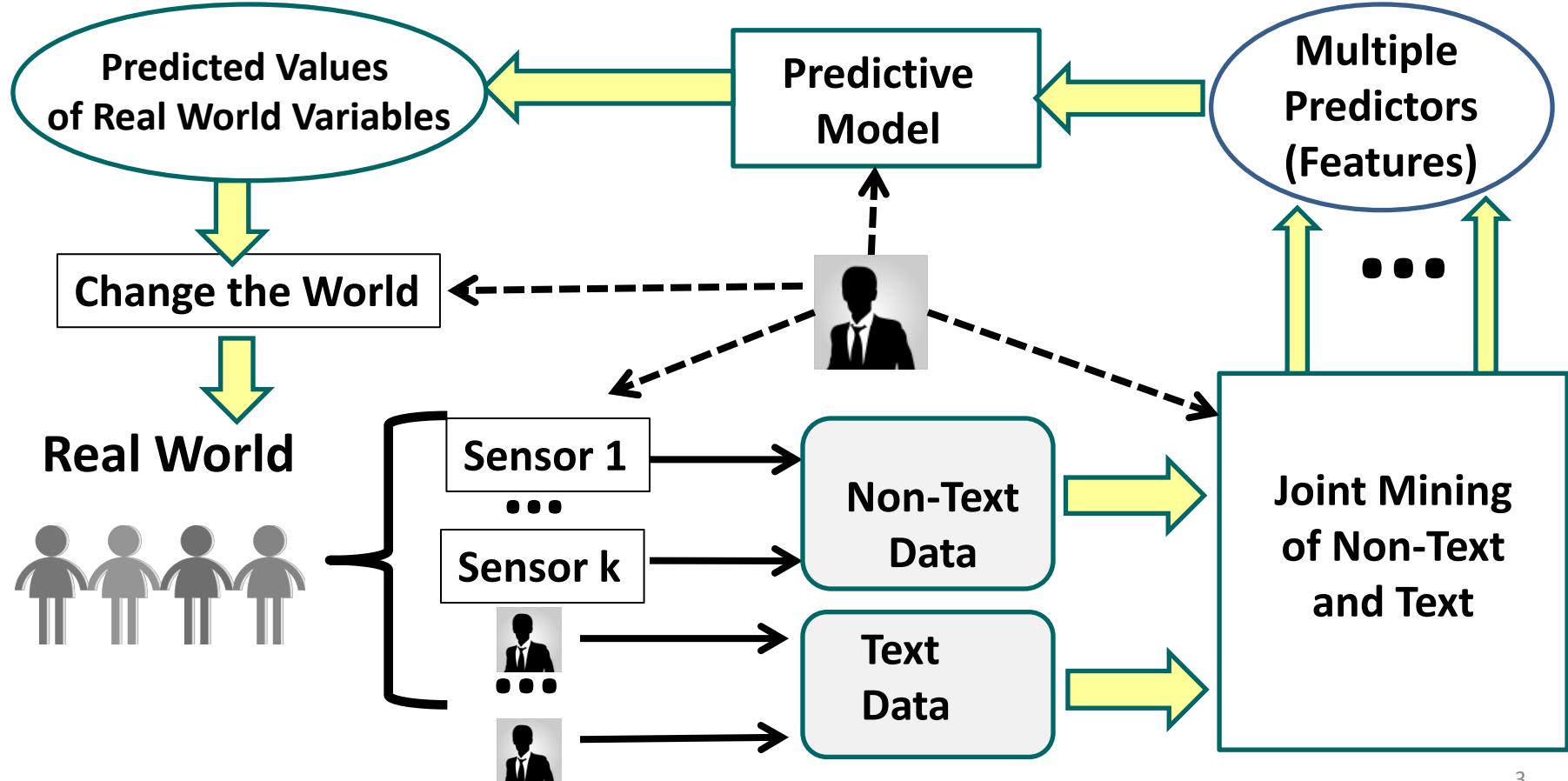
1. Natural language processing and text representation



2. Word association mining and analysis

4. Opinion mining and sentiment analysis

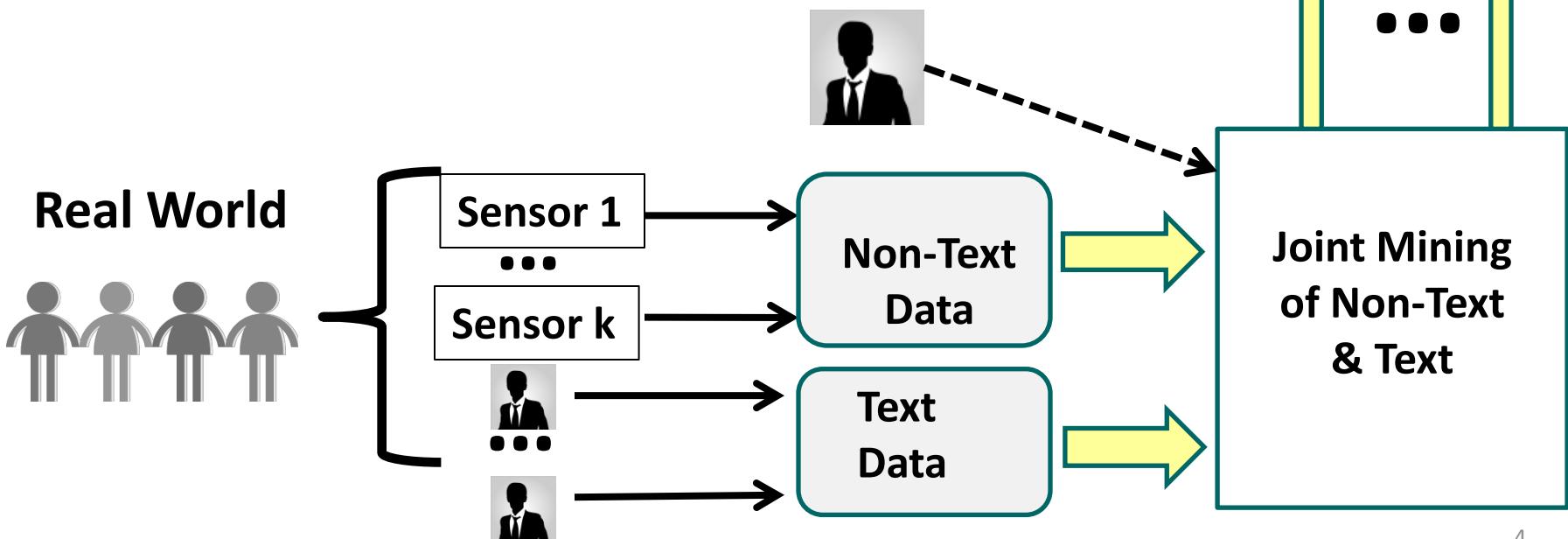
The Big Picture of Prediction: Data Mining Loop



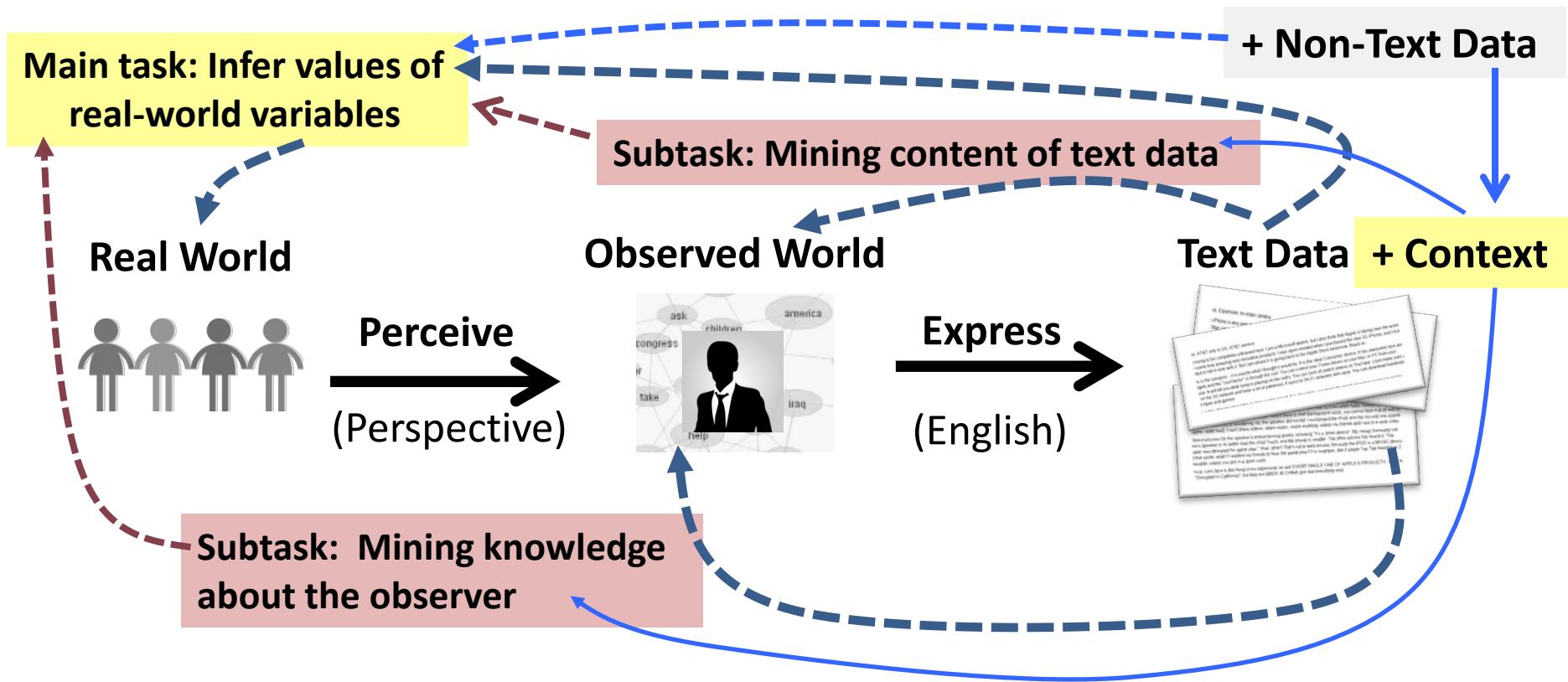
Text-Based Prediction

How can we generate effective predictors from text?

How can we jointly mine text and non-text data?



Text-Based Prediction = a Unified View of Text Mining and Analysis



Joint Mining and Analysis of Text and Non-Text Data

- Non-text data help text mining
 - Non-text data provide context for mining text data
 - **Contextual Text Mining:** Mining text in the context defined by non-text data (see [Mei 2009] for a large body of work)
- Text data help non-text data mining
 - Text data help interpret patterns discovered from non-text data
 - **Pattern Annotation:** Using text data to interpret patterns found in non-text data (see [Mei et al. 2006] for detail)

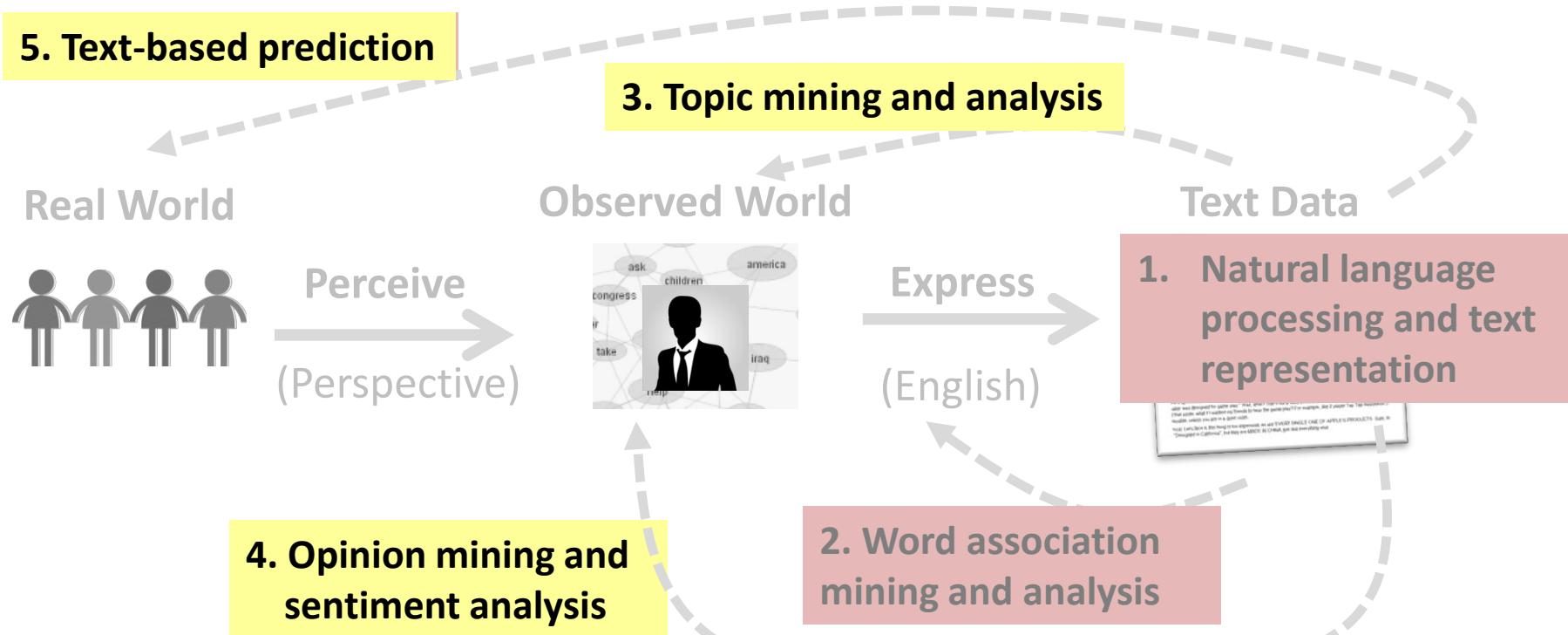
Suggested Reading

- [Mei et al. 2006] Qiaozhu Mei, Dong Xin, Hong Cheng, Jiawei Han, and ChengXiang Zhai. 2006. Generating semantic annotations for frequent patterns with context analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (KDD 2006). ACM, New York, NY, USA, 337-346. DOI=10.1145/1150402.1150441
- [Mei 2009] Qiaozhu Mei, Contextual Text Mining, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 2009.
<http://hdl.handle.net/2142/14707>

Contextual Text Mining: Motivation

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

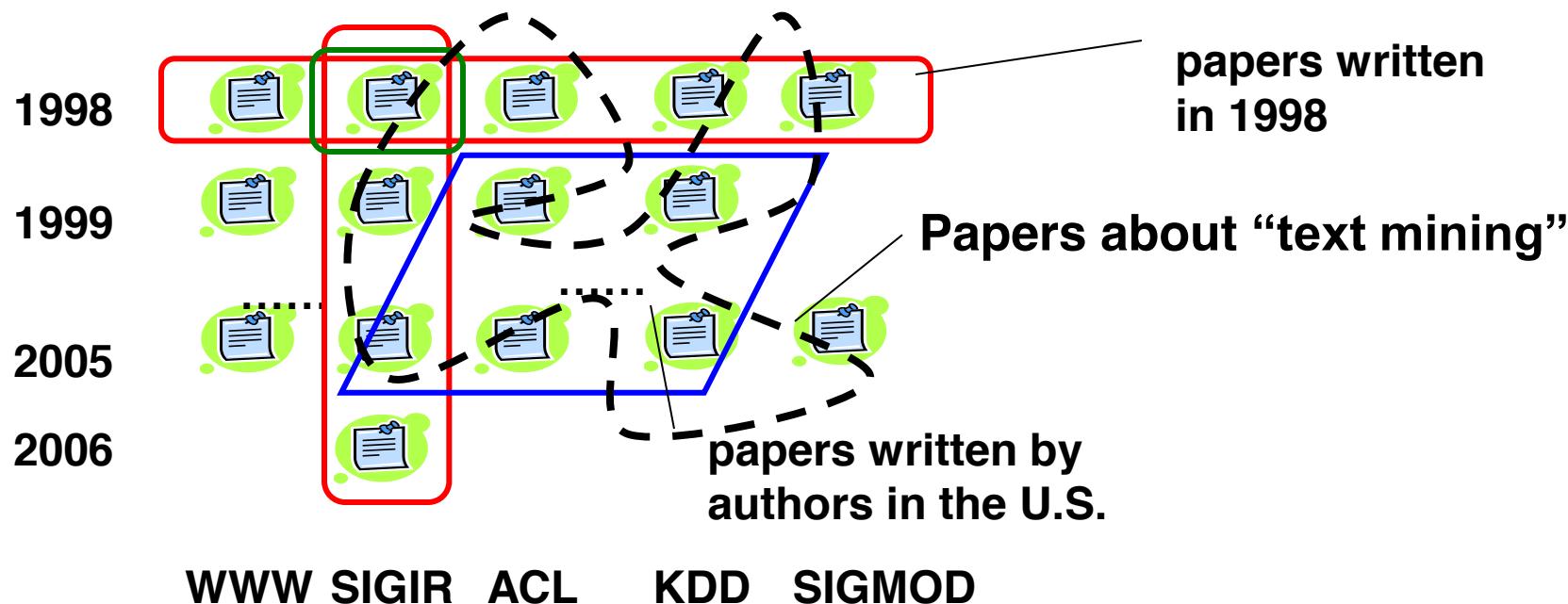
Contextual Text Mining



Contextual Text Mining: Motivation

- Text often has rich context information
 - Direct context (Meta-Data): time, location, authors, source, ...
 - Indirect context (additional data related to meta-data): social network of the author, author's age, other text from the same source, etc.
 - Any related data can be regarded as context
- Context can be used to
 - Partition text data for comparative analysis
 - Provide meaning to the discovered topics

Context = Partitioning of Text



Enables discovery of knowledge associated with different context as needed

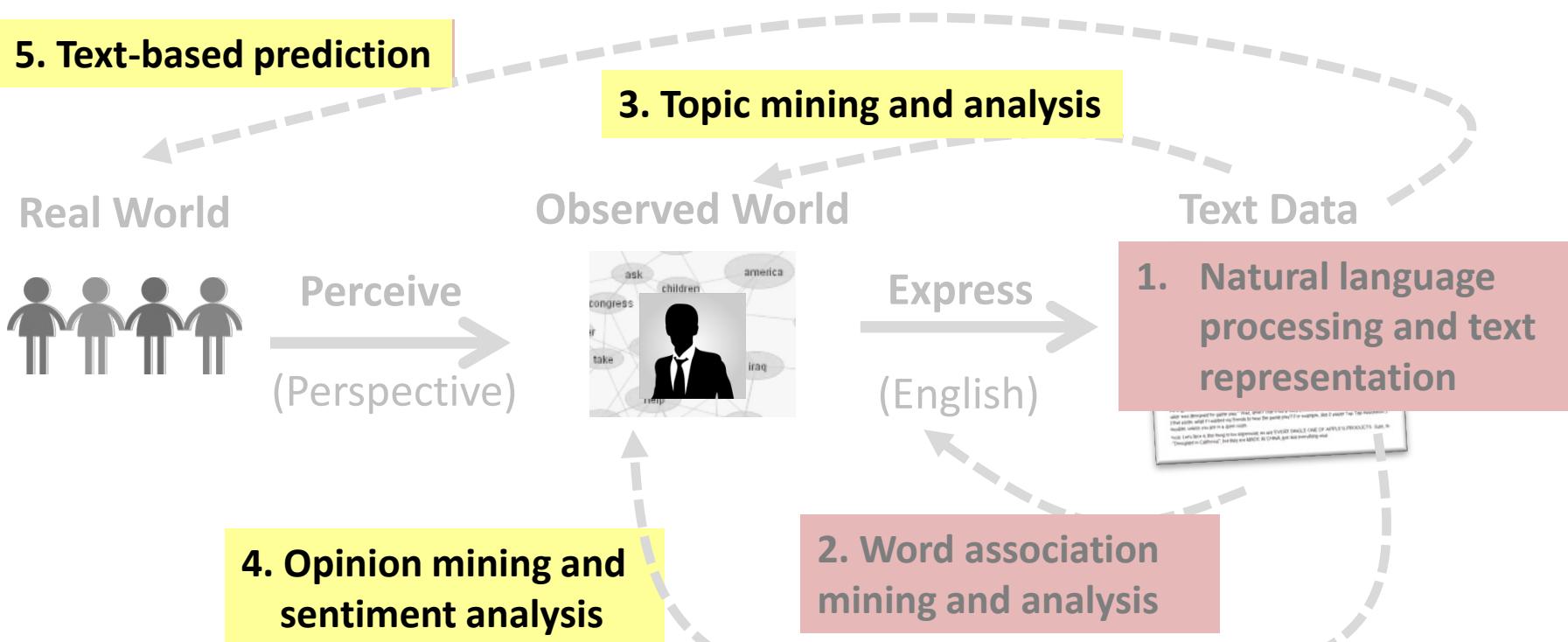
Many Interesting Questions Require Contextual Text Mining

- What topics have been gaining increasing attention recently in data mining research? (time as context)
- Is there any difference in the responses of people in different regions to the event? (location as context)
- What are the common research interests of two researchers? (authors as context)
- Is there any difference in the research topics published by authors in the USA and those outside? (author's affiliation and location as context)
- Is there any difference in the opinions about a topic expressed on one social network and another? (social network of authors and topic as context)
- Are there topics in news data that are correlated with sudden changes in stock prices? (time series as context)
- What issues “mattered” in the 2012 presidential election? (time series as context)

Contextual Text Mining: Contextual Probabilistic Latent Semantic Analysis

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Contextual Text Mining: Contextual Probabilistic Latent Semantic Analysis



Contextual Probabilistic Latent Semantic Analysis (CPLSA)

[Mei & Zhai 06]

- General idea:
 - Explicitly add interesting context variables into a generative model (→ enable discovery contextualized topics)
 - Context influences both coverage and content variation of topics
- As an extension of PLSA
 - Model the conditional likelihood of text given context
 - Assume context-dependent views of a topic
 - Assume context-dependent topic coverage
 - EM algorithm can still be used for parameter estimation
 - Estimated parameters naturally contain context variables, enabling contextual text mining

Generation Process of CPLSA

Choose a topic

Themes

	View1	View2	View3
government			
donation			
New Orleans			

Texas July 2005 sociologist

Choose a view

government 0.3
response 0.2..

donate 0.1
relief 0.05
help 0.02 ..

city 0.2
new 0.1
orleans 0.05 ..

Draw a word for

Criticism of government response to the hurricane primarily consisted of criticism of its response to ... The total shut-in oil production from the Gulf of Mexico ... approximately 24% of the annual production and the shut-in gas production ... Over seventy countries pledged monetary donations or other assistance. ...

Choose a Coverage

Theme coverage:

Texas

July 2005

.....

document

4

Comparing News Articles [Zhai et al. 04]

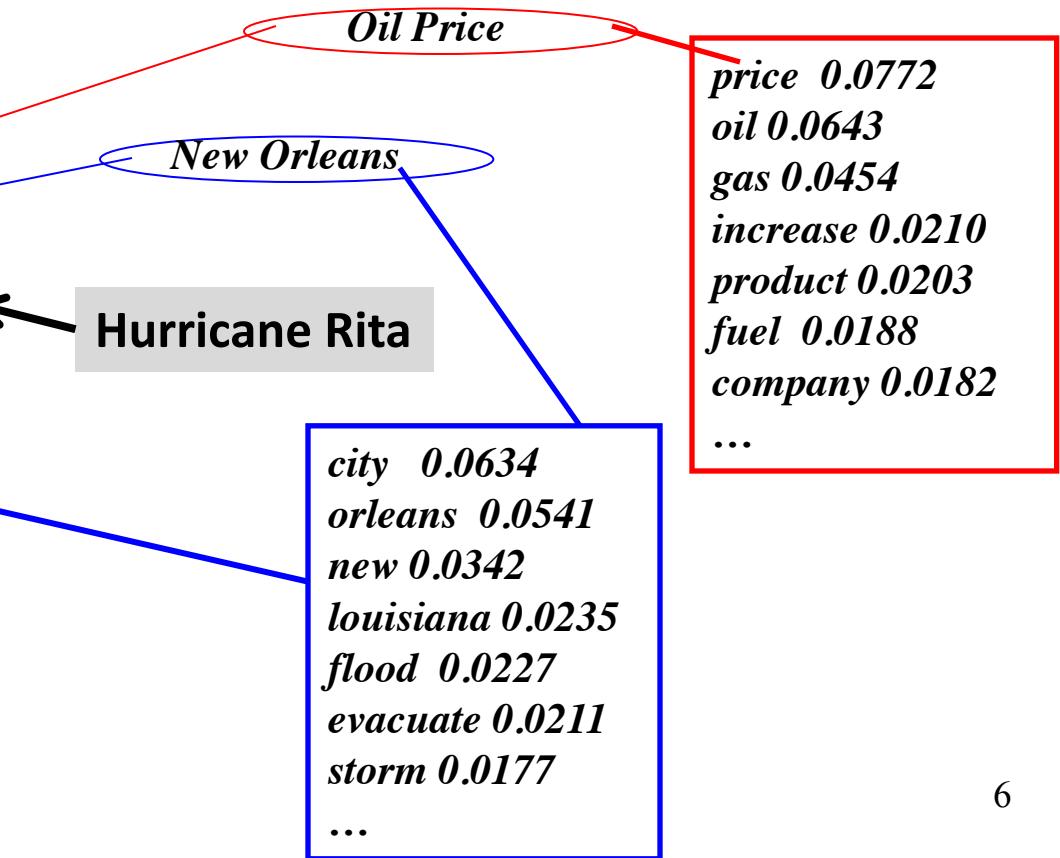
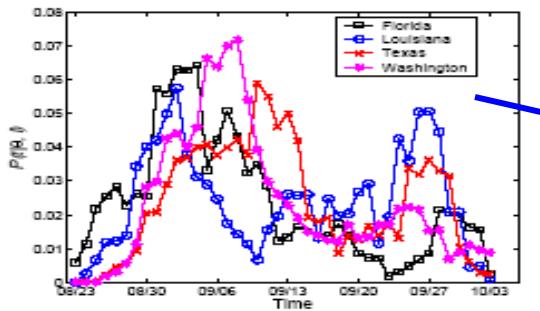
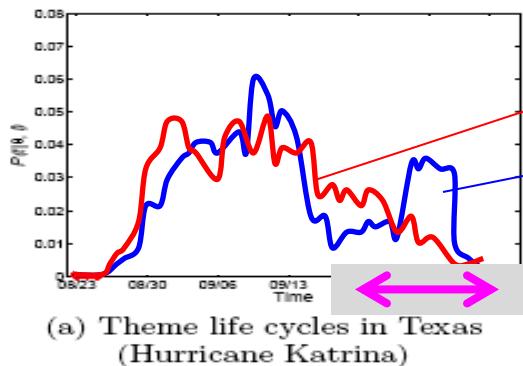
Iraq War (30 articles) vs. Afghan War (26 articles)

The common theme indicates that “United Nations” is involved in both wars

	Cluster 1	Cluster 2	Cluster 3
Common Theme	united nations 0.042 ... 0.04	killed month 0.035 deaths 0.032 ... 0.023	...
Iraq Theme	n Weapons 0.03 Inspections 0.024 ... 0.023	troops hoon 0.016 sanches 0.015 ... 0.012	...
Afghan Theme	Northern alliance 0.04 kabul 0.04 taleban 0.03 aid 0.025 ... 0.02	taleban rumsfeld 0.026 hotel front 0.012 ... 0.011	...

Collection-specific themes indicate different roles of “United Nations” in the two wars

Theme Life Cycles in Blog Articles About “Hurricane Katrina” [Mei et al. 06]



Spatial Distribution of the Topic “Government Response” in Blog Articles About Hurricane Katrina [Mei et al. 06]



(a) Week1: 08/23-08/29

Theme 1	
Government Response	
bush	0.0716374
president	0.0610942
federal	0.0514114
govern	0.0476977
fema	0.0474692
administrat	0.0233903
response	0.0208351
brown	0.0199573
blame	0.0170033
governor	0.0142153



(b) Week Two: 08/30-09/05



(c) Week Three: 09/06-09/12



(d) Week Four: 09/13-09/19

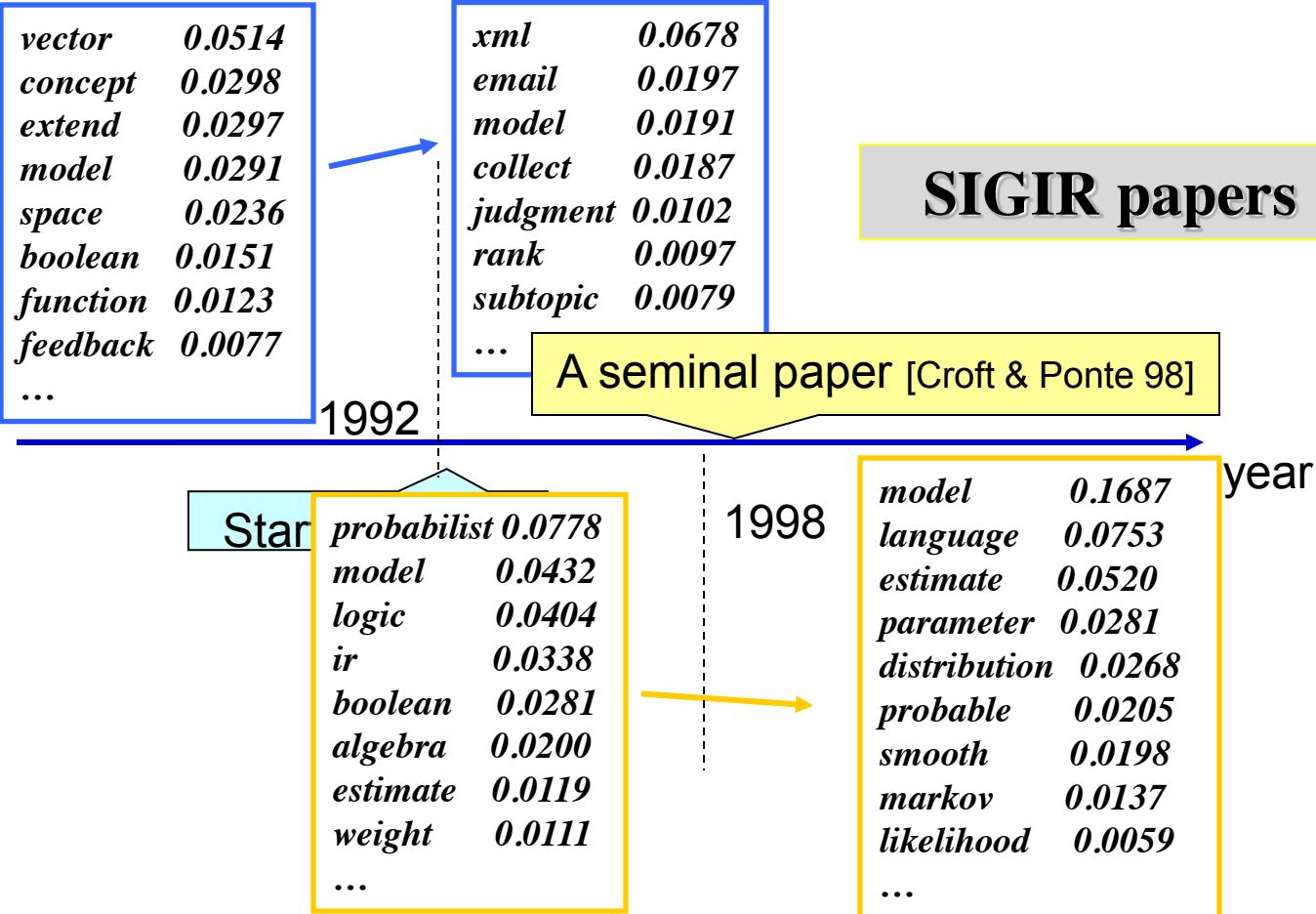


(e) Week Five: 09/20-09/26

Event Impact Analysis: IR Research [Mei & Zhai 06]

Topic: retrieval
models

<i>term</i>	0.1599
<i>relevance</i>	0.0752
<i>weight</i>	0.0660
<i>feedback</i>	0.0372
<i>independence</i>	0.0311
<i>model</i>	0.0310
<i>frequent</i>	0.0233
<i>probabilistic</i>	0.0188
<i>document</i>	0.0173
...	



Suggested Reading

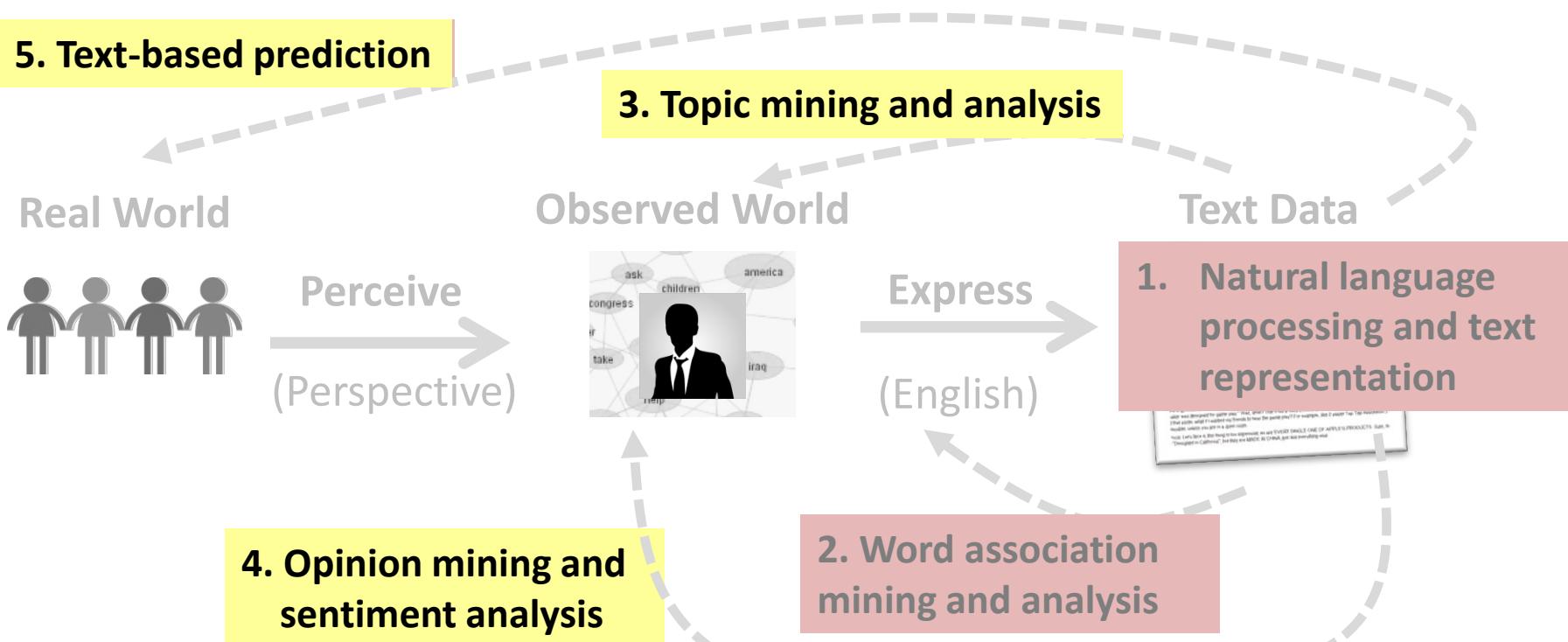
- **[Zhai et al. 04]** ChengXiang Zhai, Atulya Velivelli, and Bei Yu. 2004. A cross-collection mixture model for comparative text mining. In *Proceedings of the 10th ACM SIGKDD international conference on knowledge discovery and data mining* (KDD 2004). ACM, New York, NY, USA, 743-748.
DOI=10.1145/1014052.1014150
- **[Mei & Zhai 06]** Qiaozhu Mei and ChengXiang Zhai. 2006. A mixture model for contextual text mining. In *Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining* (KDD 2006). ACM, New York, NY, USA, 649-655. DOI=10.1145/1150402.1150482
- **[Mei et al. 06]** Qiaozhu Mei, Chao Liu, Hang Su, and ChengXiang Zhai. 2006. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In *Proceedings of the 15th international conference on World Wide Web* (WWW 2006). ACM, New York, NY, USA, 533-542.
DOI=10.1145/1135777.1135857



Contextual Text Mining: Mining Topics with Social Network Context

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Contextual Text Mining: Mining Topics with Social Network as Context



Topic Analysis with Network Context

- The **context** of a text article can form a **network**, e.g.,
 - Authors of research articles may form **collaboration networks**
 - Authors of social media content form **social networks**
 - Locations associated with text can be connected to form a **geographic network**
- **Benefit of joint analysis** of text and its network context
 - Network imposes **constraints** on topics in text (**authors connected in a network tend to write about similar topics**)
 - Text helps **characterize** the content associated with each subnetwork (e.g., difference in opinions expressed in two subnetworks?)

Network Supervised Topic Modeling: General Idea

[Mei et al. 08]

- Probabilistic topic modeling as optimization: maximize likelihood

$$\Lambda^* = \arg \max_{\Lambda} p(\text{TextData} | \Lambda)$$

- Main idea: network imposes constraints on model parameters Λ
 - The text at two adjacent nodes of the network tends to cover similar topics
 - Topic distributions are smoothed over adjacent nodes
 - Add network-induced regularizers to the likelihood objective function

Any generative model

Any network

$$\Lambda^* = \arg \max_{\Lambda} f(p(\text{TextData} | \Lambda), r(\Lambda, \text{Network}))$$

Any way to combine

Any regularizer

Instantiation: NetPLSA [Mei et al. 08]

Network-induced prior: Neighbors have similar topic distribution

Modified objective function

$$O(C, G) = (1 - \lambda) \cdot \left(\sum_d \sum_w c(w, d) \log \sum_{j=1}^k p(\theta_j | d) p(w | \theta_j) \right) + \lambda \cdot \left(-\frac{1}{2} \sum_{\langle u, v \rangle \in E} w(u, v) \sum_{j=1}^k (p(\theta_j | u) - p(\theta_j | v))^2 \right)$$

Annotations pointing to parts of the equation:

- Text collection**: Points to the first term $(1 - \lambda) \cdot \left(\sum_d \sum_w c(w, d) \log \sum_{j=1}^k p(\theta_j | d) p(w | \theta_j) \right)$.
- Network graph**: Points to the second term $\lambda \cdot \left(-\frac{1}{2} \sum_{\langle u, v \rangle \in E} w(u, v) \sum_{j=1}^k (p(\theta_j | u) - p(\theta_j | v))^2 \right)$.
- Influence of network constraint**: Points to the parameter λ in the second term.
- Weight of edge (u, v)** : Points to the weight $w(u, v)$ in the second term.
- PLSA log-likelihood**: Points to the first term $(1 - \lambda) \cdot \left(\sum_d \sum_w c(w, d) \log \sum_{j=1}^k p(\theta_j | d) p(w | \theta_j) \right)$.
- Quantify the difference in the topic coverage at node u and v** : Points to the squared difference term $(p(\theta_j | u) - p(\theta_j | v))^2$ in the second term.

Mining 4 Topical Communities: Results of PLSA

Can't uncover the 4 communities (IR, DM, ML, Web)

Topic 1		Topic 2		Topic 3		Topic 4	
term	0.02	peer	0.02	visual	0.02	interface	0.02
question	0.02	patterns	0.01	analog	0.02	towards	0.02
protein	0.01	mining	0.01	neurons	0.02	browsing	0.02
training	0.01	clusters	0.01	vlsi	0.01	xml	0.01
weighting	0.01	stream	0.01	motion	0.01	generation	0.01
multiple	0.01	frequent	0.01	chip	0.01	design	0.01
recognition	0.01	e	0.01	natural	0.01	engine	0.01
relations	0.01	page	0.01	cortex	0.01	service	0.01
library	0.01	gene	0.01	spike	0.01	social	0.01

Mining 4 Topical Communities: Results of NetPLSA

Uncovers the 4 communities well

Information Retrieval		Data Mining		Machine Learning		Web	
retrieval	0.13	mining	0.11	neural	0.06	web	0.05
information	0.05	data	0.06	learning	0.02	services	0.03
document	0.03	discovery	0.03	networks	0.02	semantic	0.03
query	0.03	databases	0.02	recognition	0.02	services	0.03
text	0.03	rules	0.02	analog	0.01	peer	0.02
search	0.03	association	0.02	vlsi	0.01	ontologies	0.02
evaluation	0.02	patterns	0.02	neurons	0.01	rdf	0.02
user	0.02	frequent	0.01	gaussian	0.01	management	0.01
relevance	0.02	streams	0.01	network	0.01	ontology	0.01

Text Information Network

- In general, we can view text data that naturally “lives” in a rich information network with all other related data
- Text data can be associated with
 - Nodes of the network
 - Edges of the network
 - Paths of the network
 - Subnetworks
 - ...
- Analysis of text should be using the entire network!

Suggested Reading

- [Mei et al. 08] Qiaozhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. 2008. Topic modeling with network regularization. In *Proceedings of the 17th international conference on World Wide Web* (WWW 2008). ACM, New York, NY, USA, 101-110. DOI=10.1145/1367497.1367512

Contextual Text Mining: Mining Causal Topics with Time Series Supervision

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Contextual Text Mining: Mining Causal Topics with Time Series Supervision

5. Text-based prediction

3. Topic mining and analysis

Real World



Perceive
(Perspective)

Observed World



Express
(English)

Text Data

1. Natural language processing and text representation



4. Opinion mining and sentiment analysis

2. Word association mining and analysis

Text Mining for Understanding Time Series

What might have caused the stock market crash?



Any clues in the companion news stream?

Dow Jones Industrial Average [Source: Yahoo Finance]

Analysis of Presidential Prediction Markets

What might have caused the sudden drop of price for this candidate?



What “mattered” in this election?



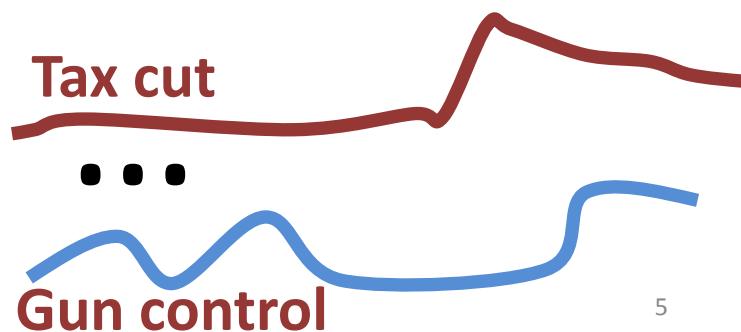
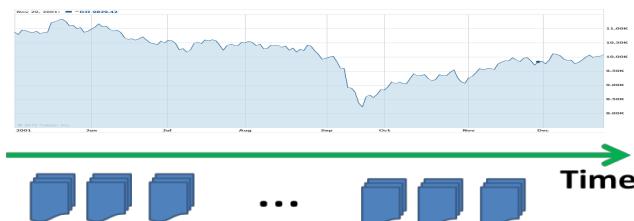
Tax cut?



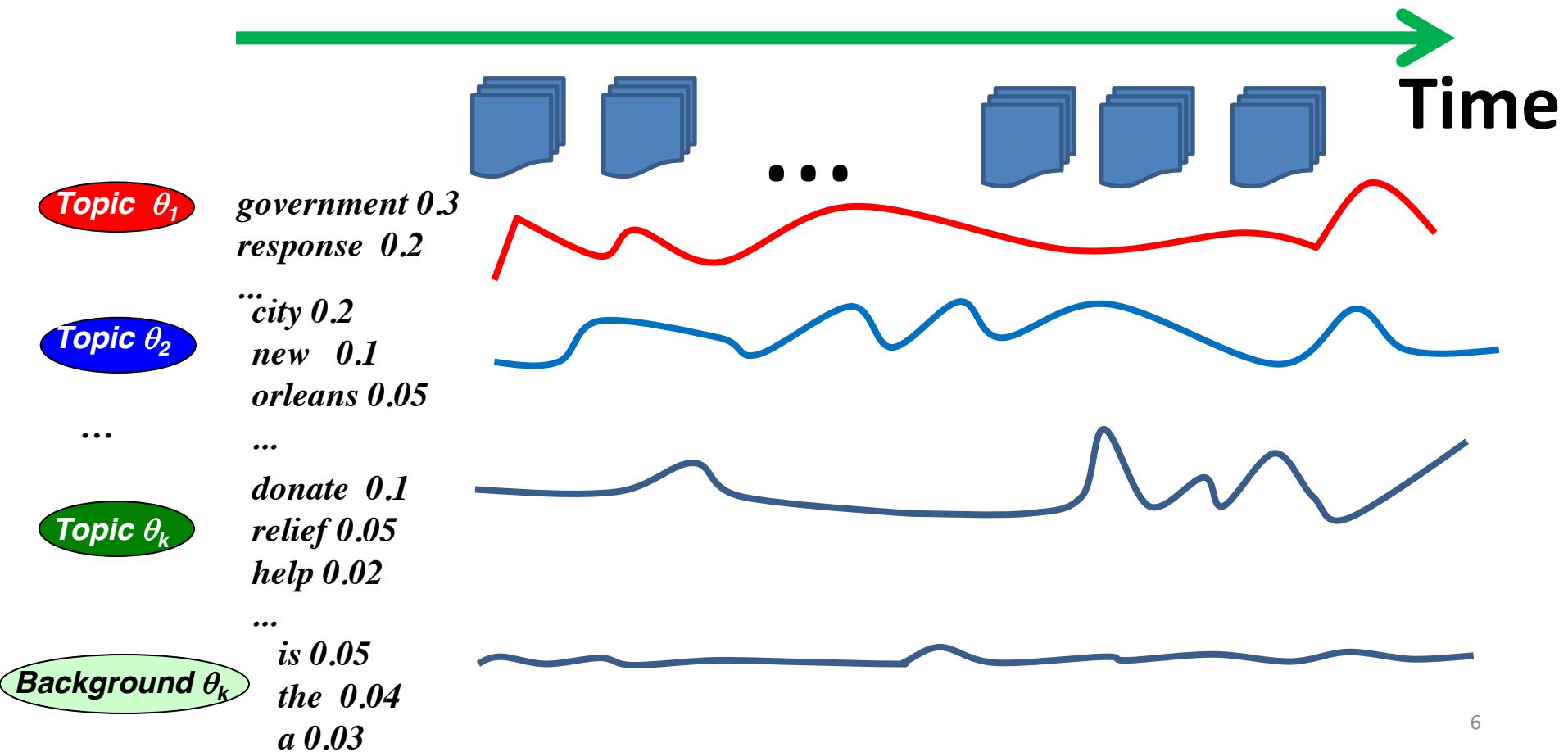
Any clues in the companion news stream?

Joint Analysis of Text and Time Series to Discover “Causal Topics”

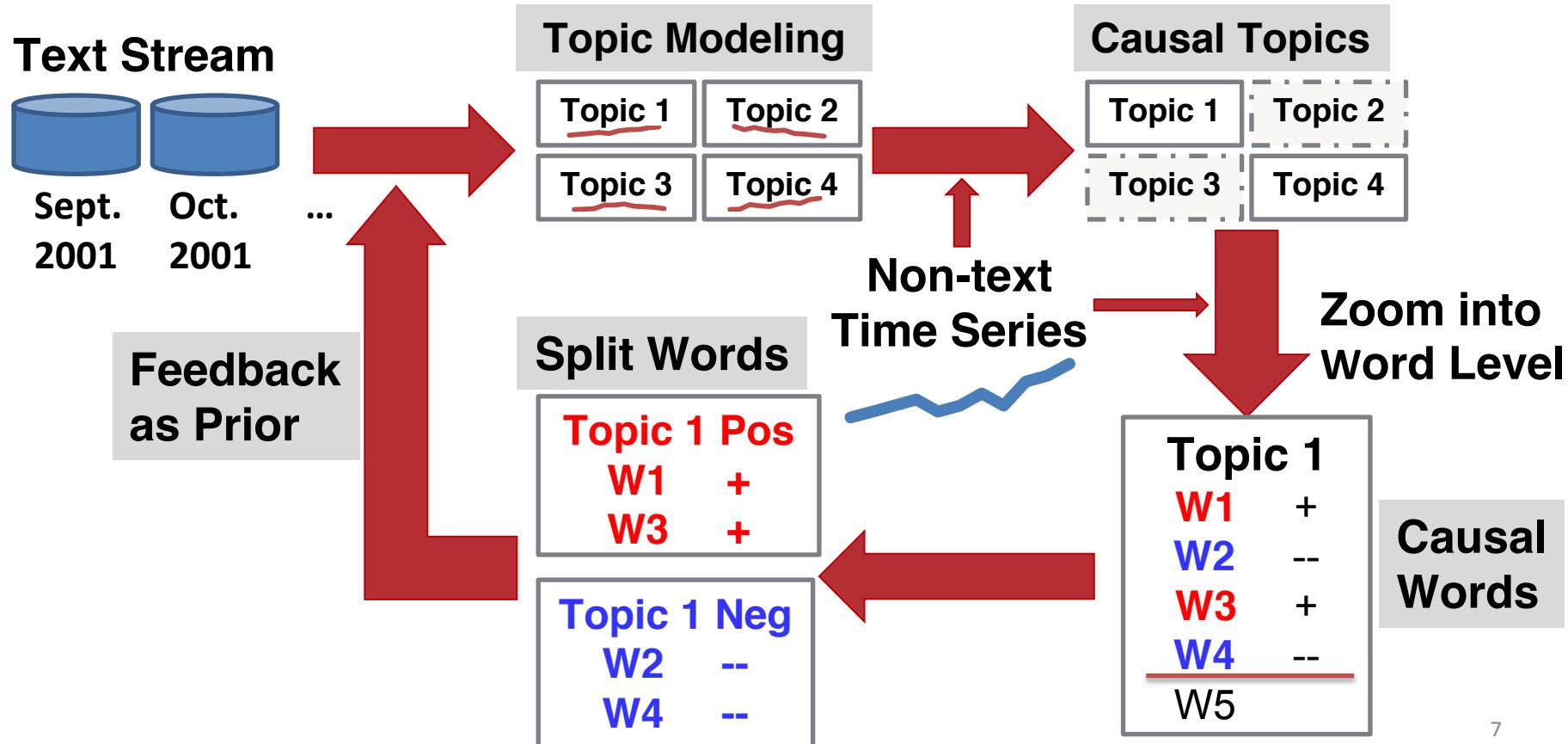
- Input:
 - Time series
 - Text data produced in a similar time period (text stream)
- Output
 - Topics whose coverage in the text stream has strong correlations with the time series (“causal” topics)



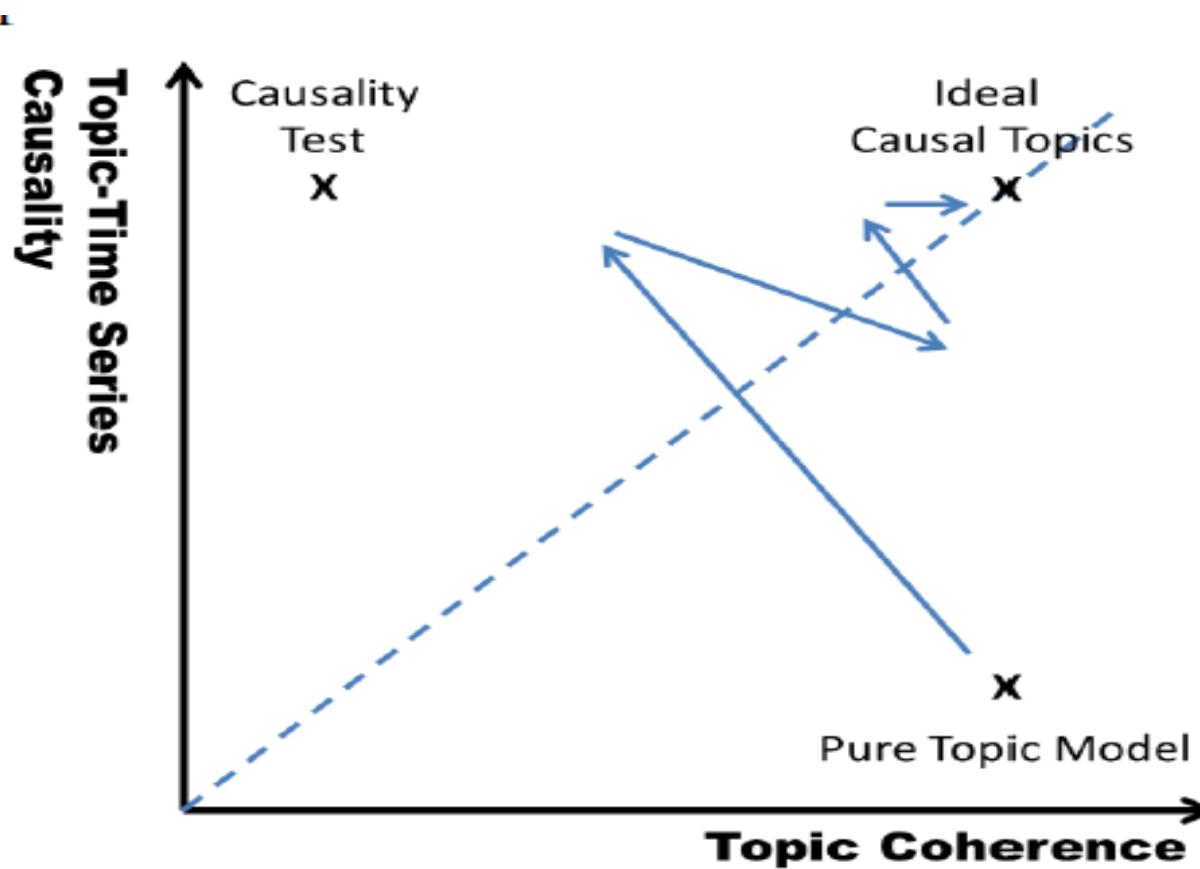
When a Topic Model Applied to Text Stream



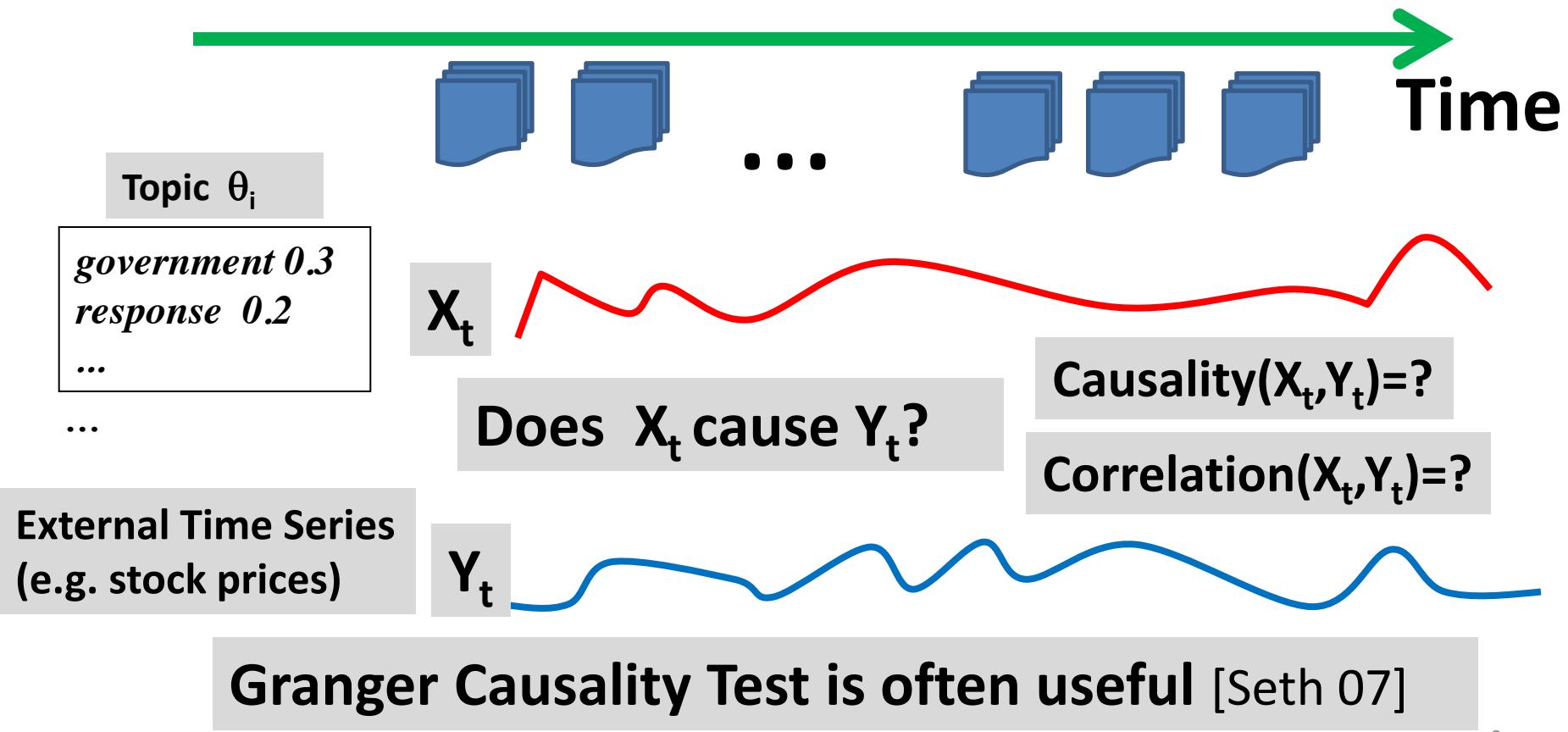
Iterative Causal Topic Modeling [Kim et al. 13]



Heuristic Optimization of Causality + Coherence



Measuring Causality (Correlation)



Topics in NY Times Correlated with Stocks

[Kim et al. 13]: June 2000 ~ Dec. 2011

AAMRQ (American Airlines)	AAPL (Apple)
<p>russia russian putin</p> <p>europe european</p> <p>germany</p> <p>bush gore presidential</p> <p>police court judge</p> <p><u>airlines airport air</u></p> <p><u>united trade terrorism</u></p> <p>food foods cheese</p> <p>nets scott basketball</p> <p>tennis williams open</p> <p>awards gay boy</p> <p>moss minnesota chechnya</p>	<p>paid notice st</p> <p>russia russian europe</p> <p>olympic games olympics</p> <p>she her ms</p> <p>oil ford prices</p> <p>black fashion blacks</p> <p><u>computer technology software</u></p> <p><u>internet com web</u></p> <p>football giants jets</p> <p>japan japanese plane</p>

Topics are biased toward each time series

Major Topics in 2000 Presidential Election

[Kim et al. 13]

Top Three Words in Significant Topics from NY Times

tax cut 1

screen pataki guiliani

enthusiasm door symbolic

oil energy prices

news w top

pres al vice

love tucker presented

partial abortion privatization

court supreme abortion

gun control nra

Text: NY Times (May 2000 - Oct. 2000)

Time Series: Iowa Electronic Market

<http://tippie.uiowa.edu/iem/>

Issues known to be
important in the
2000 presidential election

Suggested Reading

- [Kim et al. 13] Hyun Duk Kim, Malu Castellanos, Meichun Hsu, ChengXiang Zhai, Thomas Rietz, and Daniel Diermeier. 2013. Mining causal topics in text data: Iterative topic modeling with time series feedback. In *Proceedings of the 22nd ACM international conference on information & knowledge management* (CIKM 2013). ACM, New York, NY, USA, 885-890.
DOI=10.1145/2505515.2505612
- [Seth 07] Anil Seth, Granger Causality. 2007. *Scholarpedia*, 2(7): 1667, doi: 10.4249/scholarpedia.1667

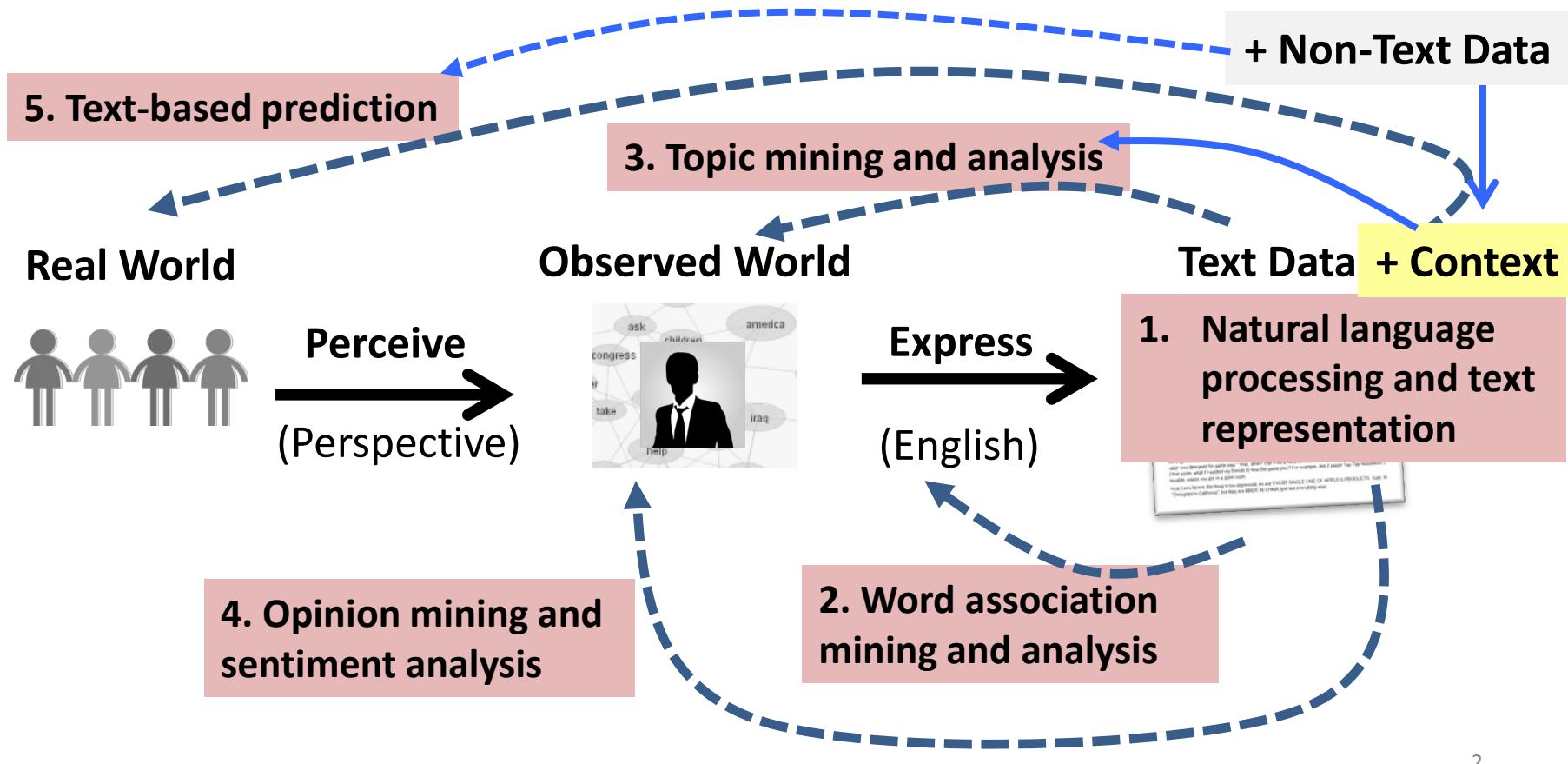
Summary of Text-Based Prediction

- Text-based prediction is very useful for “big data” applications:
 - Inferring new knowledge about the world
 - Optimizing decision making
- Text data is often combined with non-text data for prediction
 - Joint analysis of text and non-text is necessary and useful
 - Non-text data provide context for mining text data (contextual text mining)
 - Text data help interpret patterns discovered from non-text data (pattern annotation)
- An active research topic with many open challenges

Course Summary

ChengXiang “Cheng” Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Topics Covered in This Course



Key High-Level Take-Away Messages

- 13. Joint mining of text and non-text
- 14. Contextual PLSA
- 15. NetPLSA
- 16. Causal topic mining



- 6. Probabilistic Topic Model (PLSA, LDA)
- 7. Generative model; ML estimate; EM
- 8. Text clustering: model vs. similarity-based
- 9. Text categorization: generative vs. discriminative
- 10. Evaluation of clustering and categorization

- 11. Sentiment classification: ordinal regression
- 12. Latent Aspect Rating Analysis

- 1. NLP → Text representation → Knowledge discovery
- 2. Robust TM = Word-based rep + Statistical analysis



(English)

REPRESENTATION



- 3. Paradigmatic and syntagmatic relations
- 4. Text similarity: Vector space, BM25
- 5. Co-occurrence analysis: Entropy, MI

What to Learn Next

- **Natural Language Processing**
 - Foundation for all text-based applications
 - More NLP → Deeper knowledge discovery
- **Statistical Machine Learning**
 - Backbone techniques for NLP and text analysis
 - Key to predictive modeling and “big data” applications
- **Data Mining**
 - General data mining algorithms can always be applied to text
- **Text/Information Retrieval**
 - Essential system component in any text-based application (human in the loop)
 - Some techniques useful for text data mining

Main Techniques for Harnessing Big Text Data: Text Retrieval + Text Mining

