

CSE 454

Inverted Indices (with Compression & LSI)

4/7/2005 1:35 PM Copyright © Kambhampati / Weld 2002 1

Review

- **Precision & Recall metrics**
- **Vector Space Representation**
 - Dot Product as Similarity Metric
- **TF-IDF for Computing Weights**
 - $w_{ij} = f(i,j) * \log(N/n_i)$

• **But How Process Efficiently?**

4/7/2005 1:35 PM Copyright © Kambhampati / Weld 2002 2

Today's Class

- **Efficient query processing**
 - Inverted indices (creation & query processing)
 - Compression
- **Latent Semantic Indexing (LSI)**

4/7/2005 1:35 PM Copyright © Kambhampati / Weld 2002 3

Course Overview

4/7/2005 1:35 PM Copyright © Kambhampati / Weld 2002 4

Standard Web Search Engine Architecture

4/7/2005 1:35 PM Copyright © Kambhampati / Weld 2002 / UC Berkeley 5

Search Engine Components

- **Spider**
 - Getting the pages
- **Indexing**
 - Storing (e.g. in an inverted file)
- **Query Processing**
 - Booleans, ...
- **Ranking**
 - Vector space model, PageRank, anchor text analysis
- **Summaries**
- **Refinement**

4/7/2005 1:35 PM Copyright © Kambhampati / Weld 2002 6

Efficient Retrieval

- Document-term matrix

	t_1	t_2	...	t_j	...	t_m	nf
d_1	w_{11}	w_{12}	...	w_{1j}	...	w_{1m}	$1/ d_1 $
d_2	w_{21}	w_{22}	...	w_{2j}	...	w_{2m}	$1/ d_2 $
d_i	w_{i1}	w_{i2}	...	w_{ij}	...	w_{im}	$1/ d_i $
d_n	w_{n1}	w_{n2}	...	w_{nj}	...	w_{nm}	$1/ d_n $

- w_{ij} is the weight of term t_j in document d_i
- Most w_{ij} 's will be zero.

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

7

Naïve retrieval

Consider query $q = (q_1, q_2, \dots, q_j, \dots, q_n)$, $nf = 1/|q|$.

How to evaluate q (i.e., compute the similarity between q and every document)?

Method 1: Compare q with every document directly.

- document data structure:

$d_i : ((t_1, w_{i1}), (t_2, w_{i2}), \dots, (t_j, w_{ij}), \dots, (t_m, w_{im}), 1/|d_i|)$

- Only terms with positive weights are kept.
- Terms are in alphabetic order.

- query data structure:

$q : ((t_1, q_1), (t_2, q_2), \dots, (t_j, q_j), \dots, (t_m, q_m), 1/|q|)$

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

8

Naïve retrieval

Method 1: Compare q with documents directly (cont.)

initialize all $\text{sim}(q, d_i) = 0$;

for each document d_i ($i = 1, \dots, n$)

{ for each term t_j ($j = 1, \dots, m$)

if t_j appears in both q and d_i

$\text{sim}(q, d_i) += q_j * w_{ij}$;

$\text{sim}(q, d_i) = \text{sim}(q, d_i) * (1/|q|) * (1/|d_i|)$; }

sort documents in descending similarities and display the top k to the user;

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

9

Observation

- Method 1 is not efficient
 - Needs to access most non-zero entries in doc-term matrix.
- Solution: Inverted Index
 - Data structure to permit fast searching.
- Like an Index in the back of a text book.
 - Key words --- page numbers.
 - E.g. "Etzioni, 40, 55, 60-63, 89, 220"
 - Lexicon
 - Occurrences

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

10

Search Processing (Overview)

- Lexicon search
 - E.g. looking in index to find entry
- Retrieval of occurrences
 - Seeing where term occurs
- Manipulation of occurrences
 - Going to the right page

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

11

Inverted Files

POS	FILE
1	A file is a list of words by position
10	First entry is the word in position 1 (first word)
20	Entry 4562 is the word in position 4562 (4562 nd word)
30	Last entry is the last word
36	An inverted file is a list of positions by word!

a (1, 4, 40)
 entry (11, 20, 31)
 file (2, 38)
 list (5, 41)
 position (9, 16, 26)
 positions (44)
 word (14, 19, 24, 29, 35, 45)
 words (7)
 4562 (21, 27)

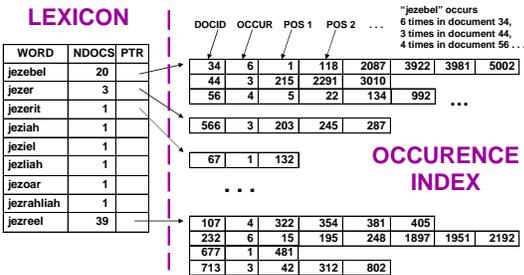
INVERTED FILE

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

12

Inverted Files for Multiple Documents



- One method. Alta Vista uses alternative

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

13

Many Variations Possible

- Address space (flat, hierarchical)
- Position
- TF /IDF info precalculated
- Header, font, tag info stored
- Compression strategies

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

14

Using Inverted Files

Several data structures:

1. For each term t_j , create a list (**inverted file list**) that contains all document ids that have t_j .
 $I(t_j) = \{ (d_1, w_{1j}), (d_2, w_{2j}), \dots, (d_i, w_{ij}), \dots, (d_n, w_{nj}) \}$
 - d_i is the document id number of the i th document.
 - Weights come from **freq** of term in doc
 - Only entries with non-zero weights should be kept.

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

15

Inverted files continued

More data structures:

2. **Normalization factors** of documents are pre-computed and stored in an array: $nf[i]$ stores $1/d_i$.
3. Lexicon: a hash table for all terms in the collection.

t_j	pointer to $I(t_j)$

- Inverted file lists are typically stored on disk.
- The number of distinct terms is usually very large.

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

16

Retrieval using Inverted files

```

initialize all  $\text{sim}(q, d_i) = 0$ ;
for each term  $t_j$  in  $q$ 
    { find  $I(t)$  using the hash table;
      for each  $(d_i, w_{ij})$  in  $I(t)$ 
           $\text{sim}(q, d_i) += q_j * w_{ij}$ ; }
for each document  $d_i$ 
     $\text{sim}(q, d_i) = \text{sim}(q, d_i) * nf[i]$ ;
sort documents in descending similarities and
display the top  $k$  to the user;
    
```

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

17

Observations about Method 2

- If a document d does not contain any term of a given query q , then d will not be involved in the evaluation of q .
- Only non-zero entries in the columns in the document-term matrix corresponding to the query terms are used to evaluate the query.
- Computes the similarities of multiple documents simultaneously (w.r.t. each query word)

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

18

Efficient Retrieval

Example (Method 2): Suppose

$q = \{ (t1, 1), (t3, 1) \}, 1/|q| = 0.7071$
 $d1 = \{ (t1, 2), (t2, 1), (t3, 1) \}, n[f1] = 0.4082$
 $d2 = \{ (t2, 2), (t3, 1), (t4, 1) \}, n[f2] = 0.4082$
 $d3 = \{ (t1, 1), (t3, 1), (t4, 1) \}, n[f3] = 0.5774$
 $d4 = \{ (t1, 2), (t2, 1), (t3, 2), (t4, 2) \}, n[f4] = 0.2774$
 $d5 = \{ (t2, 2), (t4, 1), (t5, 2) \}, n[f5] = 0.3333$
 $I(t1) = \{ (d1, 2), (d3, 1), (d4, 2) \}$
 $I(t2) = \{ (d1, 1), (d2, 2), (d4, 1), (d5, 2) \}$
 $I(t3) = \{ (d1, 1), (d2, 1), (d3, 1), (d4, 2) \}$
 $I(t4) = \{ (d2, 1), (d3, 1), (d4, 1), (d5, 1) \}$
 $I(t5) = \{ (d5, 2) \}$

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

19

$q = \{ (t1, 1), (t3, 1) \}, 1/|q| = 0.7071$

Efficient Retrieval

$d1 = \{ (t1, 2), (t2, 1), (t3, 1) \}, n[f1] = 0.4082$
 $d2 = \{ (t2, 2), (t3, 1), (t4, 1) \}, n[f2] = 0.4082$
 $d3 = \{ (t1, 1), (t3, 1), (t4, 1) \}, n[f3] = 0.5774$
 $d4 = \{ (t1, 2), (t2, 1), (t3, 2), (t4, 2) \}, n[f4] = 0.2774$
 $d5 = \{ (t2, 2), (t4, 1), (t5, 2) \}, n[f5] = 0.3333$

$I(t1) = \{ (d1, 2), (d3, 1), (d4, 2) \}$
 $I(t2) = \{ (d1, 1), (d2, 2), (d4, 1), (d5, 2) \}$
 $I(t3) = \{ (d1, 1), (d2, 1), (d3, 1), (d4, 2) \}$
 $I(t4) = \{ (d2, 1), (d3, 1), (d4, 1), (d5, 1) \}$
 $I(t5) = \{ (d5, 2) \}$

After t1 is processed:
 $\text{sim}(q, d1) = 2, \text{sim}(q, d2) = 0,$
 $\text{sim}(q, d3) = 1$
 $\text{sim}(q, d4) = 2, \text{sim}(q, d5) = 0$
After t3 is processed:
 $\text{sim}(q, d1) = 3, \text{sim}(q, d2) = 1,$
 $\text{sim}(q, d3) = 2$
 $\text{sim}(q, d4) = 4, \text{sim}(q, d5) = 0$
After normalization:
 $\text{sim}(q, d1) = .87, \text{sim}(q, d2) = .29,$
 $\text{sim}(q, d3) = .82$
 $\text{sim}(q, d4) = .78, \text{sim}(q, d5) = 0$

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

20

Efficiency versus Flexibility

- Storing computed document weights is good for efficiency but bad for flexibility.
 - Recomputation needed if tf and idf formulas change and/or tf and df information change.
- Flexibility is improved by storing raw tf and df information but efficiency suffers.
- A compromise
 - Store pre-computed tf weights of documents.
 - Use idf weights with query term tf weights instead of document term tf weights.

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

21

How Inverted Files are Created

Term	Doc #	Freq
a	2	1
aid	1	1
all	1	1
and	2	1
and	1	1
come	1	1
country	1	1
country	2	1
dark	2	1
for	1	1
for	1	1
good	1	1
in	2	1
is	1	1
is	2	1
is	1	1
manor	2	1
men	1	1
midnight	2	1
night	2	1
now	1	1
of	1	1
past	1	1
stompy	2	1
stompy	2	1
the	1	2
their	2	2
time	1	1
to	1	2
was	2	2

Dictionary

Term	N docs	Tot Freq
a	1	1
aid	1	1
all	1	1
and	1	1
come	1	1
country	2	2
dark	1	1
for	1	1
good	1	1
in	1	1
is	1	1
is	1	1
manor	1	1
men	1	1
midnight	1	1
night	1	1
now	1	1
of	1	1
past	1	1
stompy	1	1
the	2	4
their	1	1
time	2	2
to	1	2
was	1	2

Postings

Term	Doc #	Freq
a	2	1
aid	1	1
all	1	1
and	2	1
and	1	1
come	1	1
country	1	1
country	2	1
dark	2	1
for	1	1
for	1	1
good	1	1
in	2	1
is	1	1
is	2	1
is	1	1
manor	2	1
men	1	1
midnight	2	1
night	2	1
now	1	1
of	1	1
past	1	1
stompy	2	1
stompy	2	1
the	1	2
their	2	2
time	1	1
time	2	1
to	1	2
was	2	2

Should elaborate slide with all process from downloaded docs to files

Copyright © Kambhampati / Weld 2002

22

The Lexicon

- Grows Slowly (Heap's law)**
 - $O(n^\beta)$ where n =text size; β is constant $\sim 0.4 - 0.6$
 - E.g. for 1GB corpus, lexicon = 5Mb
 - Can reduce with stemming (Porter algorithm)
- Store lexicon in file in lexicographic order**
 - Each entry points to loc in occurrence file

4/7/2005 1:35 PM

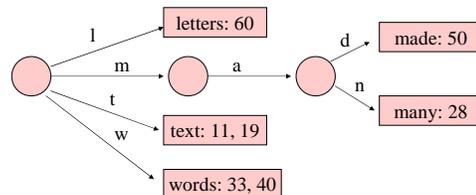
Copyright © Kambhampati / Weld 2002

23

Construction

- Build Trie (or hash table)**

1 6 9 11 17 19 24 28 33 40 46 50 55 60
 This is a text. A text has many words. Words are made from letters.

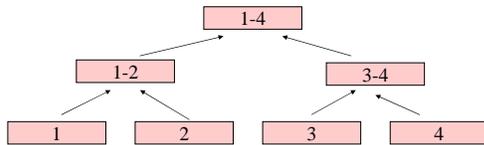


4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

24

Memory Too Small?



- **Merging**

- When word is shared in two lexicons
- Concatenate occurrence lists
- $O(n_1 + n_2)$

- **Overall complexity**

- $O(n \log(n/M))$

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

25

Stop lists

- **Language-based stop list:**

- words that bear little meaning
- 20-500 words
- http://www.dcs.gla.ac.uk/idiom/ir_resources/linguistic_utils/stop_words

- **Subject-dependent stop lists**

- **Removing stop words**

- From document
- From query

From Peter Brusilovsky Univ Pittsburg INFSCI 2140

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

26

Stemming

- **Are there different index terms?**

- retrieve, retrieving, retrieval, retrieved, retrieves...

- **Stemming algorithm:**

- (retrieve, retrieving, retrieval, retrieved, retrieves) \Rightarrow **retriev**
- Strips prefixes of suffixes (-s, -ed, -ly, -ness)
- Morphological stemming

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

27

Stemming Continued

- **Can reduce vocabulary by ~ 1/3**

- **C, Java, Perl versions, python, `c#`**

www.tartarus.org/~martin/PorterStemmer

- **Criterion for removing a suffix**

- Does "a document is about w1" mean the same as
- a "a document about w2"

- **Problems: sand / sander & wand / wander**

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

28

Compression

- **What Should We Compress?**

- Repository
- Lexicon
- Inv Index

- **What properties do we want?**

- Compression ratio
- Compression speed
- Decompression speed
- Memory requirements
- Pattern matching on compressed text
- Random access

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

29

Inverted File Compression

Each inverted list has the form $\langle f_i; d_1, d_2, d_3, \dots, d_j \rangle$

A naïve representation results in a storage overhead of $(f + n) * \lceil \log N \rceil$

This can also be stored as $\langle f_i; d_1 - d_1, d_2 - d_1, \dots, d_j - d_{j-1} \rangle$

Each difference is called a **d-gap**. Since $\sum (d - gaps) \leq N$,

each pointer requires fewer than $\lceil \log N \rceil$ bits.

Trick is encoding since worst case

➡ **Assume d-gap representation for the rest of the talk, unless stated otherwise**

Slides adapted from Tapas Kanungo and David Mount, Univ Maryland

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

30

Text Compression

Two classes of text compression methods

- Symbolwise (or statistical) methods
 - Estimate probabilities of symbols - modeling step
 - Code one symbol at a time - coding step
 - Use shorter code for the most likely symbol
 - Usually based on either arithmetic or Huffman coding
- Dictionary methods
 - Replace fragments of text with a single code word
 - Typically an index to an entry in the dictionary.
 - eg: Ziv-Lempel coding: replaces strings of characters with a pointer to a previous occurrence of the string.
 - No probability estimates needed

➔ Symbolwise methods are more suited for coding d-gaps

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

31

Classifying d-gap Compression Methods:

- **Global:** each list compressed using same model
 - **non-parameterized:** probability distribution for d-gap sizes is predetermined.
 - **parameterized:** probability distribution is adjusted according to certain parameters of the collection.
- **Local:** model is adjusted according to some parameter, like the frequency of the term
- By definition, local methods are parameterized.

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

32

Conclusion

- Local methods best
- Parameterized global models ~ non-parameterized
 - Pointers not scattered randomly in file
- In practice, best index compression algorithm is:
 - Local Bernoulli method (using Golomb coding)
- Compressed inverted indices usually faster+smaller than
 - Signature files
 - Bitmaps

Local < Parameterized Global < Non-parameterized Global

↙
Not by much

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

33

Motivating the Need for LSI

	access	document	external	information	theory	database	indexing	computer	REL	MATCH
Doc 1	x	x	x						R	
Doc 2				x*	x			x*		M
Doc 3			x	x*				x*	R	M

Query: "IDF in computer-based information look-up?"

Table 1

- Relevant docs may not have the query terms
 - but may have many "related" terms
- Irrelevant docs may have the query terms
 - but may not have any "related" terms

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

34

Terms and Docs as vectors in "factor" space

In addition to doc-doc similarity, We can compute term-term distance

Document vector

	a	b	c	d	e	f	g	h	i
Interface	0	0	1	0	0	0	0	0	0
User	0	1	1	0	1	0	0	0	0
System	2	1	1	0	0	0	0	0	0
Human	1	0	0	1	0	0	0	0	0
Computer	0	1	0	1	0	0	0	0	0
Response	0	1	0	0	1	0	0	0	0
Time	0	1	0	0	1	0	0	0	0
EPS	1	0	1	0	0	0	0	0	0
Survey	0	1	0	0	0	0	0	0	1
Trees	0	0	0	0	0	1	1	1	0
Graph	0	0	0	0	0	1	1	1	1
Minors	0	0	0	0	0	0	0	1	1

Term vector

If terms are independent, the T-T similarity matrix would be diagonal
 =If it is not diagonal, we can use the correlations to add related terms to the query
 =But can also ask the question "Are there independent dimensions which define the space where terms & docs are vectors?"

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

35

Latent Semantic Indexing

- Creates modified vector space
- Captures transitive co-occurrence information
 - If docs A & B don't share any words, with each other, but both share lots of words with doc C, then A & B will be considered similar
 - Handles polysemy (adam's apple) & synonymy
- Simulates query expansion and document clustering (sort of)

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

36

LSI Intuition

- The key idea is to map documents and queries into a lower dimensional space (i.e., composed of higher level concepts which are in fewer number than the index terms)
- Retrieval in this reduced concept space might be superior to retrieval in the space of index terms

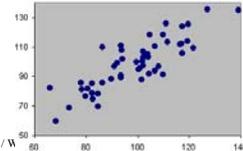
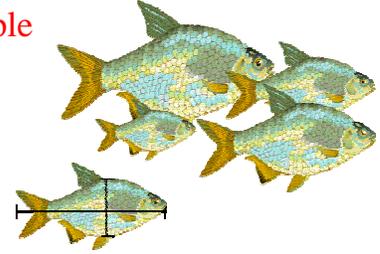
4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

37

Visual Example

- Classify Fish
 - Length
 - Height

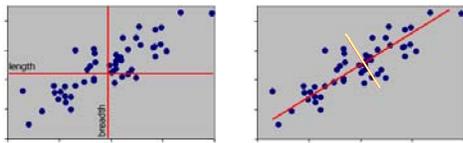
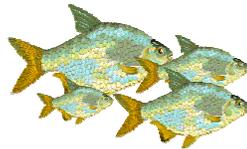


4/7/2005 1:35 PM

Copyright © Kambhampati / W

Move Origin

- To center of centroid
- But are these the best axes?



Better if one axis accounts for most data variation
What should we call the red axis?

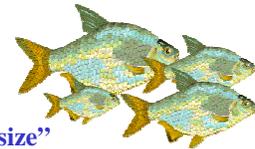
4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

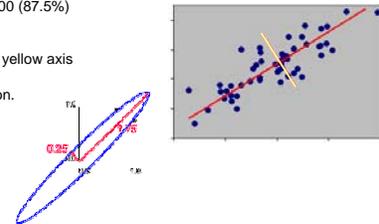
39

Reduce Dimensions

- What if we *only* consider "size"



We retain 1.75/2.00 x 100 (87.5%) of the original variation.
Thus, by discarding the yellow axis we lose only 12.5% of the original information.

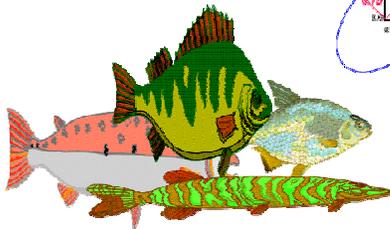


4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

40

Not Always Appropriate



4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

41

Linear Algebra Review

- Let A be a matrix
- X is an Eigenvector of A if
 - $A * X = \lambda X$
- λ is an Eigenvalue
- Transpose:

$$A * X = \lambda X$$

$$A^T =$$

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

42

Latent Semantic Indexing Defns

- Let m be the total number of index terms
- Let n be the number of documents
- Let $[A_{ij}]$ be a term-document matrix
 - With m rows and n columns
 - Entries = weights, w_{ij} , associated with the pair $[k_i, d_j]$
- The weights can be computed with tf-idf

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

43

Singular Value Decomposition

- Factor $[A_{ij}]$ matrix into 3 matrices as follows:
- $(A_{ij}) = (U) (S) (V)^t$
 - (U) is the matrix of eigenvectors derived from $(A)(A)^t$
 - $(V)^t$ is the matrix of eigenvectors derived from $(A)^t(A)$
 - (S) is an $r \times r$ diagonal matrix of singular values
 - $r = \min(m,n)$ that is, the rank of (A_{ij})
 - Singular values are the positive square roots of the eigen values of $(A)(A)^t$ (also $(A)^t(A)$)

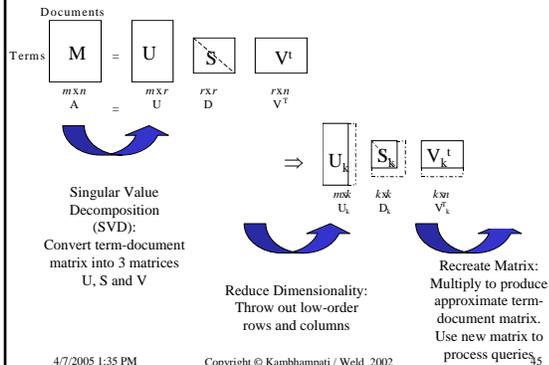
U and V are orthogonal matrices

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

44

LSI in a Nutshell



4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

45

Example

term	ch2	ch3	ch4	ch5	ch6	ch7	ch8	ch9
controllability	1	1	0	0	1	0	0	1
observability	1	0	0	0	1	1	0	1
realization	1	0	1	0	1	0	1	0
feedback	0	1	0	0	0	1	0	0
controller	0	1	0	0	1	1	0	0
observer	0	1	1	0	1	1	0	0
transfer function	0	0	0	0	1	1	0	0
polynomial	0	0	0	0	1	0	1	0
matrices	0	0	0	0	1	0	1	1

U (9x7) =

```

0.3996 -0.1037 0.5606 -0.3717 -0.3919 -0.3482 0.1029
0.4180 -0.0641 0.4878 0.1566 0.5771 0.1981 -0.1094
0.3464 -0.4422 -0.3997 -0.5142 0.2787 0.0102 -0.2857
0.1888 0.4615 0.0049 -0.0279 -0.2087 0.4193 -0.6629
0.3602 0.3776 -0.0914 0.1596 -0.2045 -0.3701 -0.1023
0.4075 0.3622 -0.3657 -0.2684 -0.0174 0.2711 0.5676
0.2250 0.1667 -0.1303 0.4376 0.3844 -0.3066 0.1230
0.2259 -0.3096 -0.3579 0.3127 -0.2406 -0.3122 -0.2611
0.2958 -0.4232 0.0277 0.4305 -0.3800 0.5114 0.2010
    
```

S (7x7) =

```

3.9901 0 0 0 0 0 0
0 2.2813 0 0 0 0 0
0 0 1.6705 0 0 0 0
0 0 0 1.3522 0 0 0
0 0 0 0 1.1818 0 0
0 0 0 0 0 0.6623 0
0 0 0 0 0 0 0.6487
    
```

V (7x8) =

```

0.2917 -0.2674 0.3883 -0.5393 0.3926 -0.2112 -0.4505
0.3399 0.4811 0.0649 -0.3760 -0.6959 -0.0421 -0.1462
0.1889 -0.0351 -0.4582 -0.5788 0.2211 0.4247 0.4346
-0.0000 -0.0000 -0.0000 -0.0000 0.0000 -0.0000 0.0000
0.6838 -0.1913 -0.1609 0.2535 0.0050 -0.5229 0.3636
0.4134 0.5716 -0.0566 0.3383 0.4493 0.3198 -0.2839
0.2176 -0.5151 -0.4369 0.1694 -0.2893 0.3161 -0.5330
0.2791 -0.2591 0.6442 0.1593 -0.1648 0.5455 0.2998
    
```

This happens to be a rank-7 matrix
-so only 7 dimensions required

Singular values = Sqrt of Eigen values of AA^T

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

46

Now to Reduce Dimensions...

- In the matrix (S), select k largest singular values
- Keep the corresponding columns in (U) and $(V)^t$
- The resultant matrix is called $(M)_k$ and is given by
 - $(M)_k = (U)_k (S)_k (V)_k^t$
 - where $k, k < r$, is the dimensionality of the concept space
- The parameter k should be
 - large enough to allow fitting the characteristics of the data
 - small enough to filter out the non-relevant representational details

The classic over-fitting issue

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

47

Formally, this will be the rank- k (2) matrix that is closest to M in the matrix norm sense

U (9x7) =

```

0.3996 -0.1037 0.5606 -0.3717 -0.3919 -0.3482 0.1029
0.4180 -0.0641 0.4878 0.1566 0.5771 0.1981 -0.1094
0.3464 -0.4422 -0.3997 -0.5142 0.2787 0.0102 -0.2857
0.1888 0.4615 0.0049 -0.0279 -0.2087 0.4193 -0.6629
0.3602 0.3776 -0.0914 0.1596 -0.2045 -0.3701 -0.1023
0.4075 0.3622 -0.3657 -0.2684 -0.0174 0.2711 0.5676
0.2250 0.1667 -0.1303 0.4376 0.3844 -0.3066 0.1230
0.2259 -0.3096 -0.3579 0.3127 -0.2406 -0.3122 -0.2611
0.2958 -0.4232 0.0277 0.4305 -0.3800 0.5114 0.2010
    
```

S (7x7) =

```

3.9901 0 0 0 0 0 0
0 2.2813 0 0 0 0 0
0 0 1.6705 0 0 0 0
0 0 0 1.3522 0 0 0
0 0 0 0 1.1818 0 0
0 0 0 0 0 0.6623 0
0 0 0 0 0 0 0.6487
    
```

V (7x8) =

```

0.2917 -0.2674 0.3883 -0.5393 0.3926 -0.2112 -0.4505
0.3399 0.4811 0.0649 -0.3760 -0.6959 -0.0421 -0.1462
0.1889 -0.0351 -0.4582 -0.5788 0.2211 0.4247 0.4346
-0.0000 -0.0000 -0.0000 -0.0000 0.0000 -0.0000 0.0000
0.6838 -0.1913 -0.1609 0.2535 0.0050 -0.5229 0.3636
0.4134 0.5716 -0.0566 0.3383 0.4493 0.3198 -0.2839
0.2176 -0.5151 -0.4369 0.1694 -0.2893 0.3161 -0.5330
0.2791 -0.2591 0.6442 0.1593 -0.1648 0.5455 0.2998
    
```

U2 (9x2) =

```

0.3996 -0.1037
0.4180 -0.0641
0.3464 -0.4422
0.1888 0.4615
0.3602 0.3776
0.4075 0.3622
0.2250 0.1667
0.2259 -0.3096
0.2958 -0.4232
    
```

S2 (2x2) =

```

3.9901 0
0 2.2813
    
```

V2 (8x2) =

```

0.2917 -0.2674
0.3399 0.4811
0.1889 -0.0351
-0.0000 -0.0000
0.6838 -0.1913
0.4134 0.5716
0.2176 -0.5151
0.2791 -0.2591
    
```

$U2 * S2 * V2$ will be a 9x8 matrix
The approximation of original matrix

4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

48

What should be the value of k?

$U_2 S_2 V_2^T$

5 components ignored

$K=2$

$USV^T = U_2 S_2 V_2^T$

term	ch3	ch4	ch5	ch6	ch7	ch8	ch9
controllability	1	1	0	0	1	0	1
observability	1	0	0	1	1	0	1
realization	1	1	0	0	0	1	0
feedback	0	1	0	0	1	0	0
controller	0	1	0	0	1	1	0
observer	0	1	1	1	1	0	0
transfer function	0	0	0	1	1	0	0
polynomial	0	0	0	0	1	1	0
matrices	0	0	0	0	1	0	1

$U_4 S_4 V_4^T$

3 components ignored

$K=4$

$U_6 S_6 V_6^T$

One component ignored

$K=6$

4/7/2005 1:35 PM Copyright © Kambhampati / Weld 2002 49

Coordinate transformation inherent in LSI

$M = U S V^T$

Mapping of keywords into LSI space is given by US

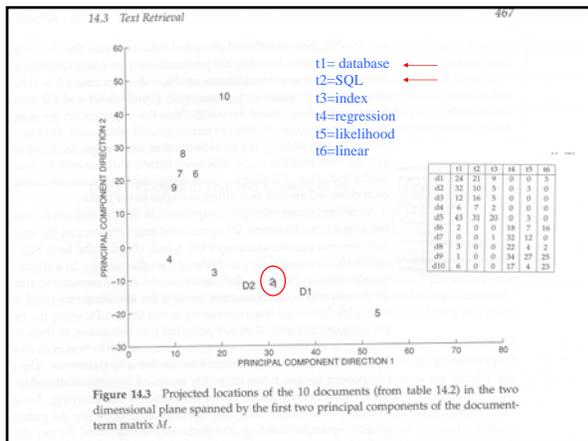
Mapping of a doc $d=[w_1 \dots w_k]$ into LSI space is given by dUS^{-1}

The base-keywords of The doc are first mapped To LSI keywords and Then differentially weighted By S^{-1}

For $k=2$, the mapping is:

	LSx	LSy
controllability	1.5944439	-0.2365708
observability	1.6678618	-0.14623132
realization	1.3821706	-1.0087909
feedback	0.7333309	1.05282
controller	1.4372339	0.86141896
observer	1.6259657	0.83262685
Transfer function	1.0972775	0.38029274
polynomial	0.90136355	-0.7062905
matrices	1.1802715	-0.96544623

4/7/2005 1:35 PM Copyright © Kambhampati / Weld 2002 50



Calculating Information Loss

$\lambda_1, \dots, \lambda_6 = \{77.4, 69.5, 22.9, 13.5, 12.1, 4.8\}$

In agreement with our intuition, most of the variance in the data is captured by the first two principal components. In fact, if we were to retain only these two principal components (as two surrogate terms instead of the six original terms), the fraction of variance that our two-dimensional representation retains is $(\lambda_1^2 + \lambda_2^2) / \sum_{i=1}^6 \lambda_i^2 = 0.925$; i.e., only 7.5% of the information has been lost (in a mean-square sense). If we represent the documents in the new two-dimensional principal component space, the coefficients for each document correspond to the first two columns of the U matrix:

d1	30.8998	-11.4912
d2	30.3131	-10.7801
d3	18.0007	-7.7138
d4	8.3765	-3.5611
d5	52.7057	-20.6051
d6	14.2118	21.8263
d7	10.8052	21.9140
d8	11.5080	28.0101
d9	9.5259	17.7666
d10	19.9219	45.0751

Should clean this up into a slide summarizing the info loss formula

4/7/2005 1:35 PM Copyright © Kambhampati / Weld 2002 52

SVD Computation complexity

- For an $m \times n$ matrix SVD computation is
 - $O(k^2 n + k' n^3)$ complexity
 - $k=4$ and $k'=22$ for best algorithms
 - Approximate algorithms that exploit the sparsity of M are available (and being developed)

4/7/2005 1:35 PM Copyright © Kambhampati / Weld 2002 53

What LSI can do

- LSI analysis effectively does
 - Dimensionality reduction
 - Noise reduction
 - Exploitation of redundant data
 - Correlation analysis and Query expansion (with related words)
- Any one of the individual effects can be achieved with simpler techniques (see thesaurus construction). But LSI does all of them together.

4/7/2005 1:35 PM Copyright © Kambhampati / Weld 2002 54

LSI is not the most sophisticated
dimensionality reduction technique

- **Dimensionality reduction is a useful technique for any classification/regression problem**
 - Text retrieval can be seen as a classification problem
- **Many other dimensionality reduction techniques**
 - Neural nets, support vector machines etc.
- **Compared to them, LSI is limited because it's *linear***
 - It cannot capture non-linear dependencies between original dimensions
 - E.g.



4/7/2005 1:35 PM

Copyright © Kambhampati / Weld 2002

55