Hons. Degree in Computing
H4016 Text Analysis

Unit 3 – Document Vectors

Ref for this section: *Konchady, Text Mining Application Programming, chapter 4*

Recap / Context

The last units (2.1 & 2.2) looked at techniques for selecting the keywords to improve on a simple bag of words approach. We now move on to step 3 . . .

5. Analyzing Results

4. Text/Data Mining

- Classification- Supervised Learning
- Clustering- Unsupervised Learning

3. **Feature Selection**

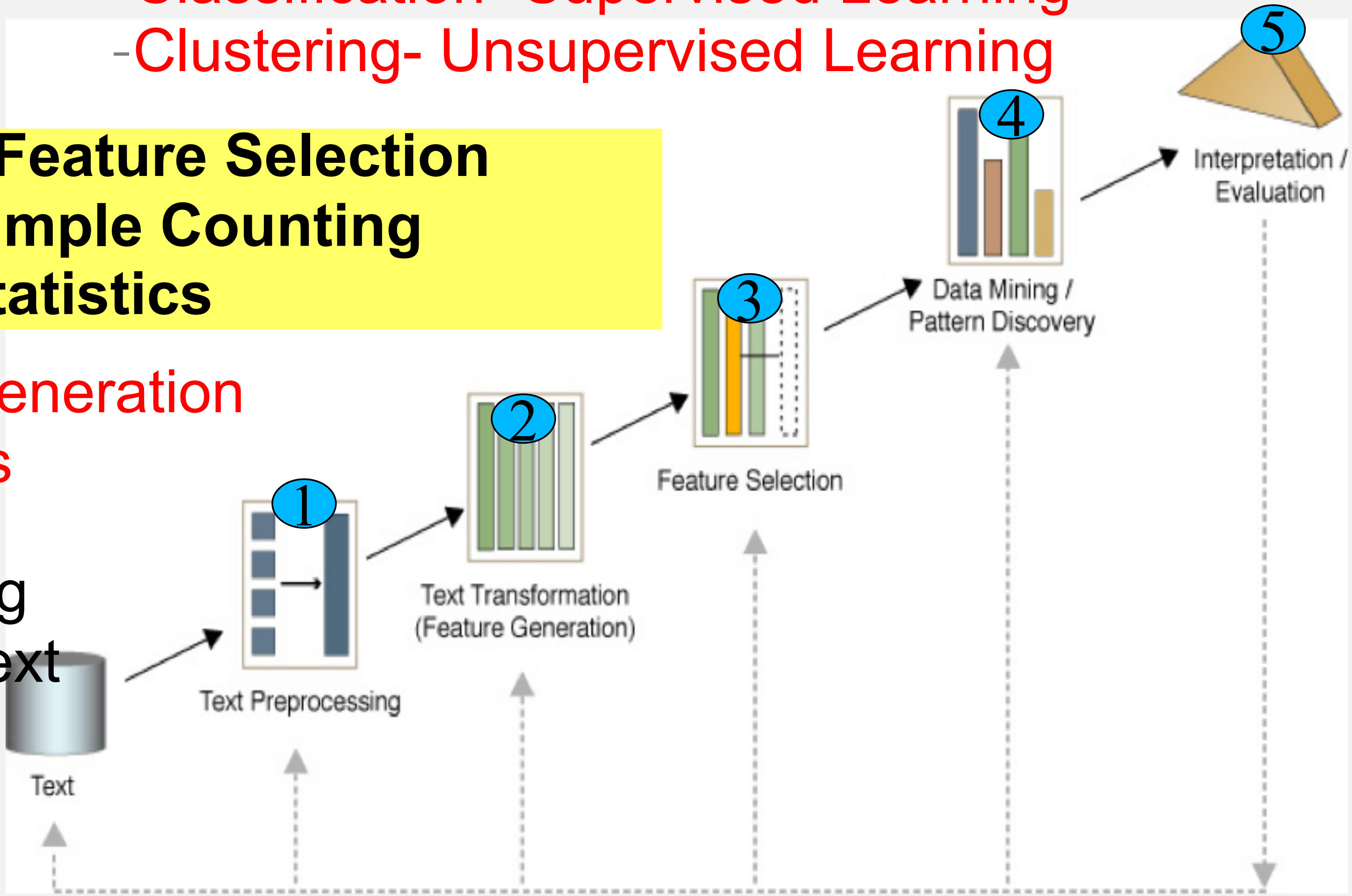
- **Simple Counting**
- **Statistics**

2. Features Generation

- Bag of Words

1. Text Preprocessing

- Syntactic/Semantic Text Analysis



5
4
Interpretation / Evaluation
Data Mining / Pattern Discovery
3
Feature Selection
2
Text Transformation (Feature Generation)
1
Text Preprocessing
Text

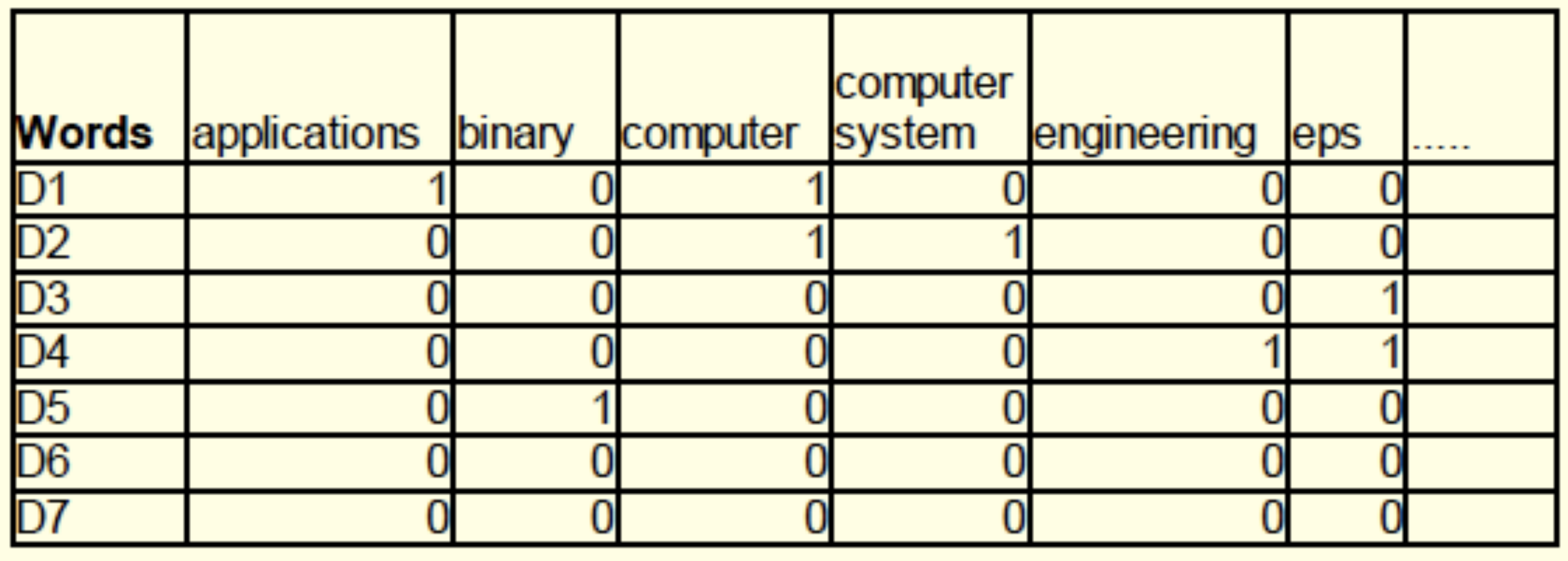
Objectives

To explain a range of alternatives for calculating the data entries in a document vector,

- Unit 2 focused on deciding the attributes themselves (column headings)
- This Unit focus on the data entries, i.e. recording how frequently each attributes occurs in the document.

Vector Generation

The objective in vector generation is to produce document vectors such as shown below:



Words	applications	binary	computer	computer system	engineering	eps	.....
D1	1	0	1	0	0	0	
D2	0	0	1	1	0	0	
D3	0	0	0	0	0	1	
D4	0	0	0	0	1	1	
D5	0	1	0	0	0	0	
D6	0	0	0	0	0	0	
D7	0	0	0	0	0	0	

Vector Generation

Documents are rows.

Features (terms of interest) are columns.

How many features to include will depend on the project.

- For classification, you need at least enough terms to distinguish one class from another

- For clustering, the number of features will influence the number of clusters found.

Vector Generation

Cell entries can be:

- Binary – 0 means term is absent in the document; 1 means term is present in the document.

- Three value system - 0 means term is absent in the document; 1 means term occurs once in the document, 2 means term occurs more than once in the document.

- Normalised frequencies – indicates how often the term appears in the document

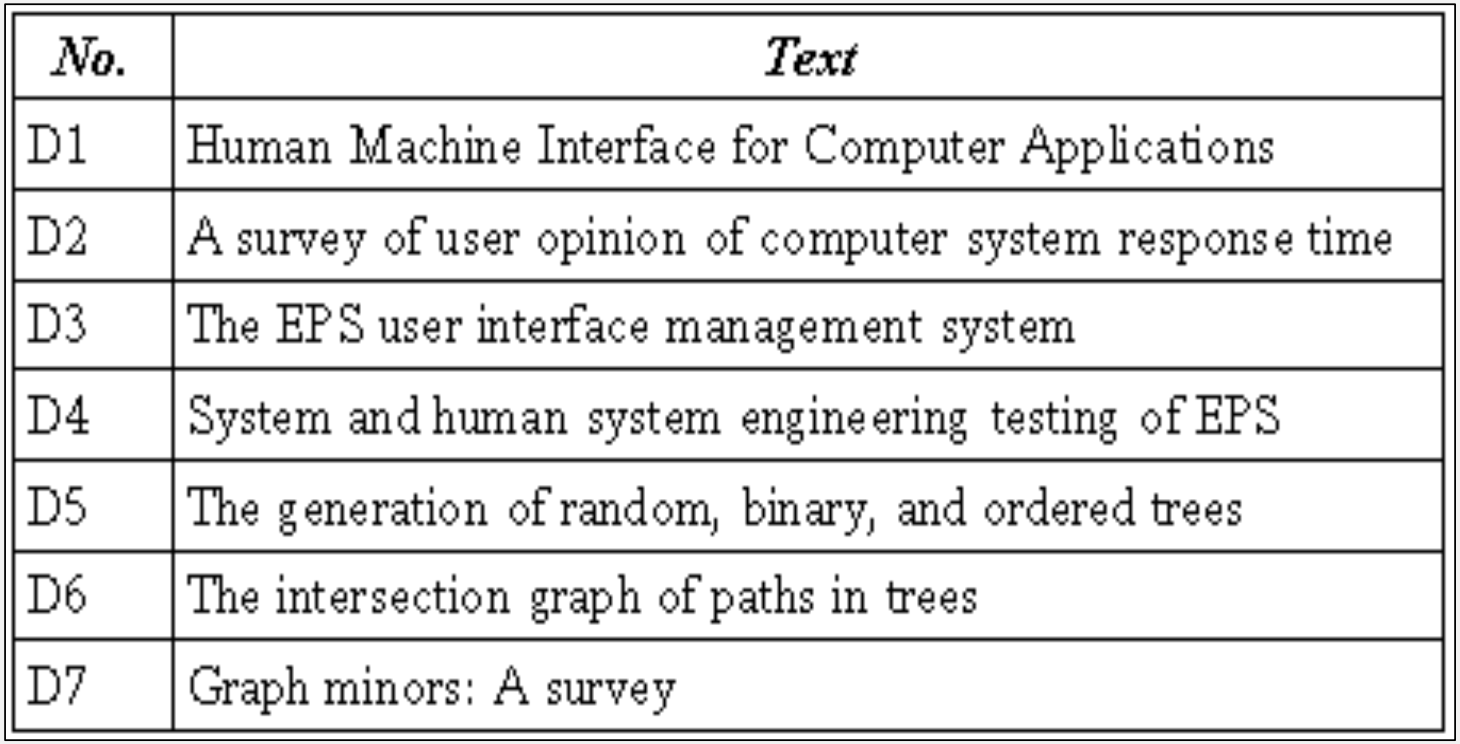
- IDF – infrequent terms are weighted more favourably

- Latent Semantic Indexing include terms from related documents.

Creating a document vector

Step 1: Simple Frequencies

Take the following sample set of Seven Documents, each is one sentence long:



No.	*Text*
D1	Human Machine Interface for Computer Applications
D2	A survey of user opinion of computer system response time
D3	The EPS user interface management system
D4	System and human system engineering testing of EPS
D5	The generation of random, binary, and ordered trees
D6	The intersection graph of paths in trees
D7	Graph minors: A survey

Taking just the content words (nouns, pronouns, verbs & adverbs), but including single and composite tokens, an index of terms in the seven documents could be as follows:

Creating a document vector

Step 1: Simple Term Frequencies

(occurences)

Table 1: Simple Frequencies

Words	applications	binary	computer	computer system	engineering	eps	generation	graph	human	interface	intersection	machine	management	minors	opinion	ordered	paths	random	response time	survey	system	testing	trees	user	user interface	**Term Total**
D1	1	0	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**5**
D2	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	1	0	**7**
D3	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1	**6**
D4	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	**2**	1	0	0	0	**6**
D5	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	**5**
D6	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	**4**
D7	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	**3**

Creating a document vector

Step 1: Simple Term Frequencies

Actual counts in themselves can only be interpreted with respect to the total number of words in a document.

For example, system appears twice in D4 above, which is the highest frequency of any term in this particular index. However a count of 2 might be relatively small in a collection of larger documents.

A more generic representation of frequency is to normalise counts to values between 0 and 1 such that the total frequency in each document adds up to one.

This is done in the following table (each count is divided by the total number of content words in that document).

Creating a document vector

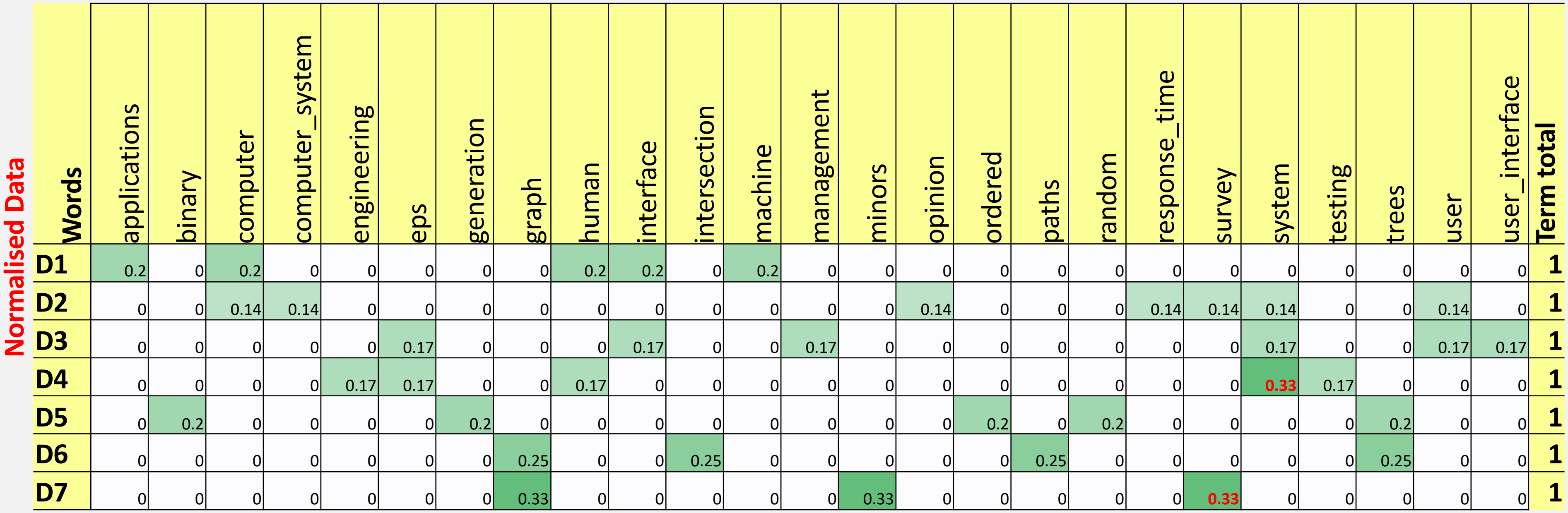
Step 2: Normalised Term Frequencies (TF)

Examples of the calculations:

Document 1 has 5 terms, so each term count was divided by 5: 1/5 = 0.2

Document 4 has 6 terms, so each term count was divided by 6. 1/6 = 0.17 2/6 = 0.33

Document 7 has 3 terms, so each term count was divided by36. 1/3 = 0.33



Normalised Data

Words	applications	binary	computer	computer_system	engineering	eps	generation	graph	human	interface	intersection	machine	management	minors	opinion	ordered	paths	random	response_time	survey	system	testing	trees	user	user_interface	Term total
D1	0.2	0	0.2	0	0	0	0	0	0.2	0.2	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	1
D2	0	0	0.14	0.14	0	0	0	0	0	0	0	0	0	0	0.14	0	0	0	0.14	0.14	0.14	0	0	0.14	0	1
D3	0	0	0	0	0	0.17	0	0	0	0.17	0	0	0.17	0	0	0	0	0	0	0	0.17	0	0	0.17	0.17	1
D4	0	0	0	0	0.17	0.17	0	0	0.17	0	0	0	0	0	0	0	0	0	0	0	0.33	0.17	0	0	0	1
D5	0	0.2	0	0	0	0	0.2	0	0	0	0	0	0	0	0	0.2	0	0.2	0	0	0	0	0.2	0	0	1
D6	0	0	0	0	0	0	0	0.25	0	0	0.25	0	0	0	0	0	0.25	0	0	0	0	0	0.25	0	0	1
D7	0	0	0	0	0	0	0	0.33	0	0	0	0	0	0.33	0	0	0	0	0	0.33	0	0	0	0	0	1

Aside 1: Scaling term frequencies

- The example on the last slide works for short texts, but will produce very small counts for larger texts. The following alternative is often used to calculate Term Frequency (TF) of counts > 0:

$$TF = 0.5 + \frac{0.5 * occurences}{occurences\ of\ the\ most\ frequent\ term}$$

- It scales TF to the range [0.5, 1]. Examples:

Doc x	body	court	crime	defendant	police
Occurrences	5	2	**20**	4	10
TF as above	0.62	0.55	1	0.6	0.75

=0.5 + ((0.5*5) / 20)

=0.5 + ((0.5*2) / 20)

=0.5 + ((0.5*20) / 20)

=0.5 + ((0.5*4) / 20)

=0.5 + ((0.5*10) / 20)

Aside 2: Other term frequencies

FYI - Rapidminer scales TF a little differently as follows:

1. Square the normalised TFs for each term in a document, add them, and get the square root of the answer:

Words	applications	binary	computer	computer_system	engineering	eps	generation	graph	human	interface	intersection	machine	management	minors	opinion	ordered	paths	random	response_time	survey	system	testing	trees	user	user_interface
D4	0	0	0	0	0.17	0.17	0	0	0.17	0	0	0	0	0	0	0	0	0	0	0	**0.33**	0.17	0	0	0

- $(0.17)^2 + (0.17)^2 + (0.17)^2 + (0.33)^2 + (0.17)^2 = 0.22$
- $\mathrm{Sqrt}(0.22) = 0.47$

2. Divide each normalized TF in a document by the answer above. So for D4, this gives:

D4	0	0	0	0	0.36	0.36	0	0	0.36	0	0	0	0	0	0	0	0	0	0	0	0.7	0.36	0	0	0

i.e. 0.17 / 0.47 = 0.36

Explained here: http://community.rapidminer.com/t5/RapidMiner-Studio/How-does-RapidMiner-calculate-Term-Frequency-TF/td-p/13900

Creating a document vector

Step 3: IDF

Returning to our previous seven statements:

- Both computer and application appear in document 1. Computer also appears in document 2.
- Research has shown that if a term's count is adjusted based on **how many documents that term appears** in, the modelling results will be more accurate. This is done as follows:
    - The count of a term that appears in many documents is reduced, by multiplying it by a weight < 1.
    - The count of a term that appears in just a few documents is increased, by multiplying it by a weight > 1.

The most common way to do this is to calculate an **IDF (Inverse Document Frequency)** weighting for each term, and use this to increase/decrease the term counts as follows . . .

Creating Document Vectors

Term ranking using IDF

IDF, or variations of IDF are used in almost all term weighting approaches. Although a heuristic measure, it has been proven to work well.

Formula:

$$\mathrm{idf}(t_i) = \log \frac{N}{n_i}$$

Where t_i is the term, N is the number of documents in the collection, and n_i is the number of documents containing the term t_i.

Any log base can be used, but it's usually $\log_2$

Inverse Document Frequency

$$\boxed{\mathrm{idf}(t_i) = \log \frac{N}{n_i}}$$

IDF generates higher weights for more specific words and lower weights for common words

For example, suppose you have a collection of 10,000 documents:

Log (10000/10000) = 0 (term is in every document)

Log(10000/5000) = 0.301 (term is in half of the documents)

Log(10000/1000) = 1 (term is in one tenth of the documents)

Log (10000/20) = 2.698 (term is in 1/500 of the documents)

Log(10000/1) = 4 (term is in 1/10,000 of the documents)

Using log base 10, terms are positively weighted if they are present in one tenth of the documents or less.

Using log base 2, terms are positively weighted if they appear in 50% of the documents or less.

Calculating IDF . . .

Returning back to the seven sentences used earlier, this table shows the IDF weighting for each term, using $\log_2$ (last row).

Note: ***System*** appears four times, but only in three documents, so n_i for system is 3.

Log(7/3)=1.22; Log (7/2)=1.81; Log (7/1)=2.81

Actual Word Count

Words	applications	binary	computer	computer_system	engineering	eps	generation	graph	human	interface	intersection	machine	management	minors	opinion	ordered	paths	random	response_time	survey	system	testing	trees	user	user_interface	Count
D1	1	0	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	5
D2	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	1	0	7
D3	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1	6
D4	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	6
D5	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	5
D6	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	4
D7	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	3
n_i	1	1	2	1	1	2	1	2	2	2	1	1	1	1	1	1	1	1	1	2	3	1	2	2	1	
IDF	2.81	2.81	1.81	2.81	2.81	1.81	2.81	1.81	1.81	1.81	2.81	2.81	2.81	2.81	2.81	2.81	2.81	2.81	2.81	1.81	1.22	2.81	1.81	1.81	2.81	

N=7

Calculating IDF . . .

$$\boxed{\mathrm{idf}(t_i) = \log \frac{N}{n_i}}$$

Each term appears in either 1, 2 or 3 documents.

Terms that appear in only one document have their count adjusted by a weight of 2.81, since.

- $\mathrm{Log}_2\ (N / n_i) = \log_2\ (7/\mathbf{1}) = \log_2\ (7) = 2.81$

Terms that appear in two documents have their count adjusted by a smaller weight of 1.81.

- $\mathrm{Log}_2\ (N / n_i) = \log_2\ (7/\mathbf{2}) = \log_2\ (3.5) = 1.81$

Terms that appear in three documents have their count adjusted by the smallest weight, 1.22.

- $\mathrm{Log}_2\ (N / n_i) = \log_2\ (7/\mathbf{3}) = \log_2\ (2.33) = 1.22$

The IDF weighting is applied to the original word count for each term. The example illustrated in the next table multiplies the term count by it's IDF weighting giving . . .

Step 4. Applying IDF to term counts

Calculations:

applications has word count of 1, and an IDF of 2.81.

1* 2.81 = 2.81.

Computer also has word count of 1, but an IDF of 1.81.

1* 1.81 = 1.81.

System in D4 has a word count of 2, and an IDF of 1.22

2 * 1.22 = 2.44

Word Count Adjusted by IDF weightings

	applications	binary	computer	computer_system	engineering	eps	generation	graph	human	interface	intersection	machine	management	minors	opinion	ordered	paths	random	response_time	survey	system	testing	trees	user	user_interface	Count
D1	2.81	0	1.81	0	0	0	0	0	1.81	1.81	0	2.81	0	0	0	0	0	0	0	0	0	0	0	0	0	5
D2	0	0	1.81	2.81	0	0	0	0	0	0	0	0	0	0	2.81	0	0	0	2.81	1.81	1.22	0	0	1.81	0	7
D3	0	0	0	0	0	1.81	0	0	0	1.81	0	0	2.81	0	0	0	0	0	0	0	1.22	0	0	1.81	2.81	6
D4	0	0	0	0	2.81	1.81	0	0	1.81	0	0	0	0	0	0	0	0	0	0	0	2.44	2.81	0	0	0	6
D5	0	2.81	0	0	0	0	2.81	0	0	0	0	0	0	0	0	2.81	0	2.81	0	0	0	0	1.81	0	0	5
D6	0	0	0	0	0	0	0	1.81	0	0	2.81	0	0	0	0	0	2.81	0	0	0	0	0	1.81	0	0	4
D7	0	0	0	0	0	0	0	1.81	0	0	0	0	0	2.81	0	0	0	0	0	1.81	0	0	0	0	0	3

Term-Frequency - Inverse Document Frequency (TF-IDF)

worked example

As with simple word frequencies, the adjusted frequencies are normalised (and scaled) to values between 0 and 1. This normalised count, adjusted by an IDF weighting, is called **TF-IDF**

The result is given on the next slide:

First table: TF-IDF

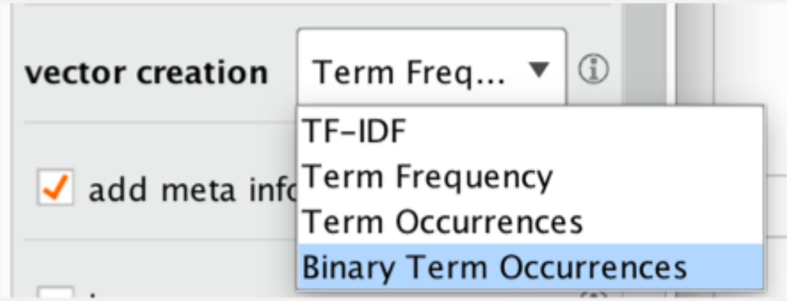
Second table: original normalised counts (TF).

Note: Applications and Computers appear equally frequently in document 1 (both occur once). However Computers also appears in other documents and so is not as useful in determining what D1 is about. Following weighting using IDF, Applications now has a higher rank than computers

Comparing IDF with simple counts

	applications	binary	computer	computer_system	engineering	eps	generation	graph	human	interface	intersection	machine	management	minors	opinion	ordered	paths	random	response_time	survey	system	testing	trees	user	user_interface
	TF-IDF																								
D1	0.25	0	0.16	0	0	0	0	0	0.16	0.16	0	0.25	0	0	0	0	0	0	0	0	0	0	0	0	0
D2	0	0	0.12	0.19	0	0	0	0	0	0	0	0	0	0	0.19	0	0	0	0.19	0.12	0.08	0	0	0.12	0
D3	0	0	0	0	0	0.15	0	0	0	0.15	0	0	0.23	0	0	0	0	0	0	0	0.1	0	0	0.15	0.23
D4	0	0	0	0	0.24	0.15	0	0	0.15	0	0	0	0	0	0	0	0	0	0	0	0.21	0.24	0	0	0
D5	0	0.22	0	0	0	0	0.22	0	0	0	0	0	0	0	0	0.22	0	0.22	0	0	0	0	0.14	0	0
D6	0	0	0	0	0	0	0	0.2	0	0	0.3	0	0	0	0	0	0.3	0	0	0	0	0	0.2	0	0
D7	0	0	0	0	0	0	0	0.28	0	0	0	0	0	0.44	0	0	0	0	0	0.28	0	0	0	0	0
	TF (normalised simple term count)																								
D1	0.2	0	0.2	0	0	0	0	0	0.2	0.2	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0
D2	0	0	0.14	0.14	0	0	0	0	0	0	0	0	0	0	0.14	0	0	0	0.14	0.14	0.14	0	0	0.14	0
D3	0	0	0	0	0	0.17	0	0	0	0.17	0	0	0.17	0	0	0	0	0	0	0	0.17	0	0	0.17	0.17
D4	0	0	0	0	0.17	0.17	0	0	0.17	0	0	0	0	0	0	0	0	0	0	0	0.33	0.17	0	0	0
D5	0	0.2	0	0	0	0	0.2	0	0	0	0	0	0	0	0	0.2	0	0.2	0	0	0	0	0.2	0	0
D6	0	0	0	0	0	0	0	0.25	0	0	0.25	0	0	0	0	0	0.25	0	0	0	0	0	0.25	0	0
D7	0	0	0	0	0	0	0	0.33	0	0	0	0	0	0.33	0	0	0	0	0	0.33	0	0	0	0	0

Recap to date



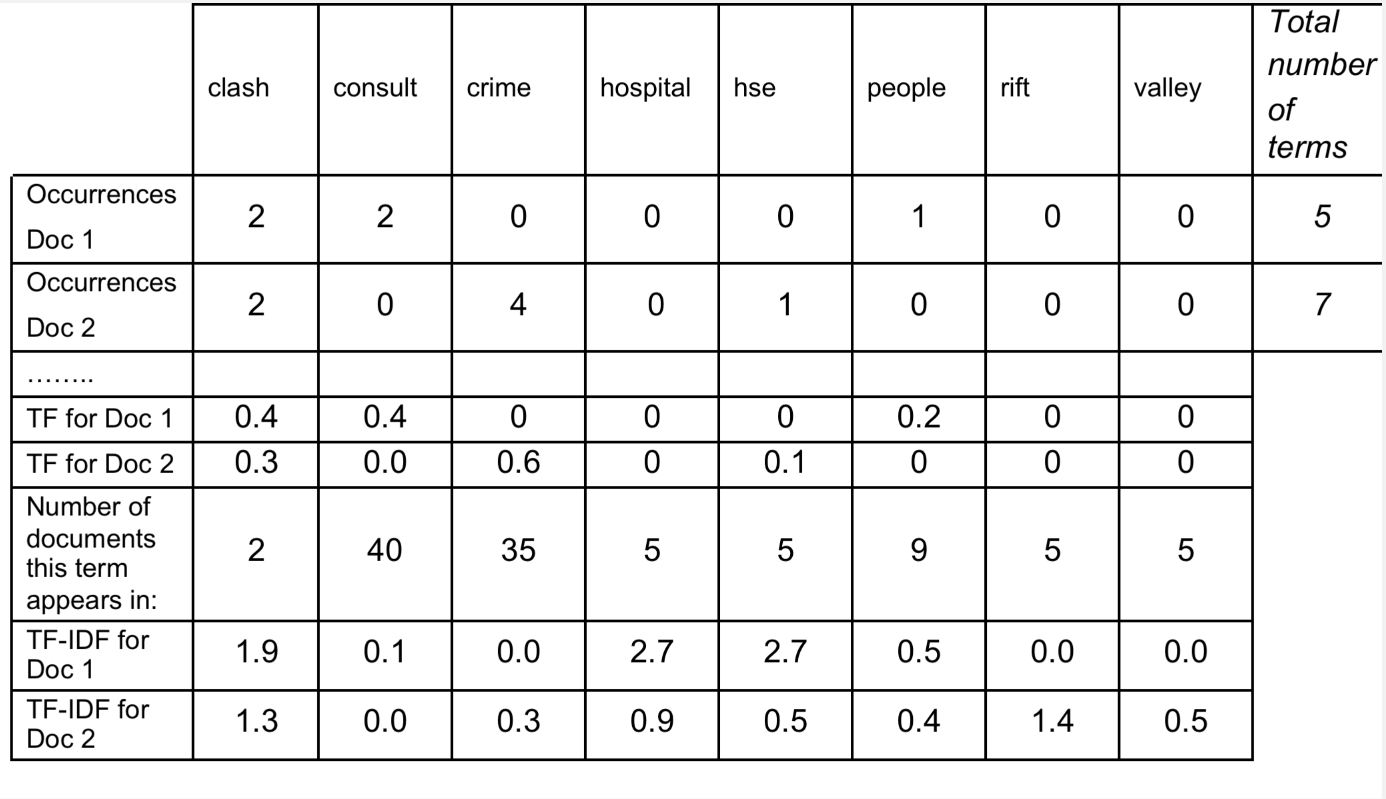
vector creation Term Freq... ▼

☑ add meta infc

1. Binary Term Occurrences is a 0 or 1, indicating presence or absence of term in a document
2. Term Occurrence counts the actual number of times a term appears in a document
3. Term Frequency (TF) normalises term occurrence so that it is a fraction of the number of terms in the document. The results is scaled to the interval [0,1].
4. TF-IDF adds an IDF weight to TF based on the number of documents a term occurs in. Terms that appear in less documents have their TF increased; terms that appear in more documents have their term decreased

Question from past exam paper:

$$\boxed{\mathrm{idf}(t_i) = \log \frac{N}{n_i}}$$



	clash	consult	crime	hospital	hse	people	rift	valley	*Total number of terms*
Occurrences Doc 1	2	2	0	0	0	1	0	0	*5*
Occurrences Doc 2	2	0	4	0	1	0	0	0	*7*
……..									
TF for Doc 1	0.4	0.4	0	0	0	0.2	0	0	
TF for Doc 2	0.3	0.0	0.6	0	0.1	0	0	0	
Number of documents this term appears in:	2	40	35	5	5	9	5	5	
TF-IDF for Doc 1	1.9	0.1	0.0	2.7	2.7	0.5	0.0	0.0	
TF-IDF for Doc 2	1.3	0.0	0.3	0.9	0.5	0.4	1.4	0.5	

The table above illustrates three ways to record term counts in a document vector: Occurrences, Term Frequency (TF) and Term Frequency - Inverse Document Frequency (TF-IDF).
Give an overview of each of these counts. **(7 marks)**

Explain why the term ‘*clash’* has an occurrence of two for both documents, but the corresponding term frequencies are different (0.4 for doc1 and 0.3 for doc2) **(3 marks)**

Explain why ‘*clash’* and *‘consult’* both have an occurrence of 2 in document 1, but their TF-IDF values are quite different (1.9 for *clash* and 0.1 for *consult*)
Note: there are more than two documents in the collection. See row 6 of the table above for a count of how many documents each term appears in. **(4 marks)**

Question from past exam paper:

	rival	experience	competition	candidate
doc 1: occurrences	5	10	5	3
doc 1: TF	0.22	0.43	0.22	0.13
number of documents this term appears in	10	15	25	4
doc 1: TF-IDF	0.27	0.38	0.09	0.27

a) When generating a document vector from unstructured text, there are a number of choices for term count. Explain each of the three choices depicted in Table 1: **occurrences**, **TF**, and **TF-IDF**.

In your answer, explain why the terms ***rival*** and ***competition*** have the same count for occurrences and TF only, while the terms ***rival*** and ***candidate*** have the same count for TF-IDF only. **(12 marks)**

Creating Document vectors

Reducing number of attribues

Using term counts to reduce bag or words:

What words are best at determining the meaning of a document?

- If a word appears in all documents is that useful?
- If a word appears infrequently in a document is that useful?

Aside: G.K. Zipf did extensive research into peoples use of words, claiming that we use few words very often, and rarely use most other words. The effort of using a word is based on familiarity and frequency of use in the language. 20% of words account for the majority of words in a text.

Aside: Zipf’s Law

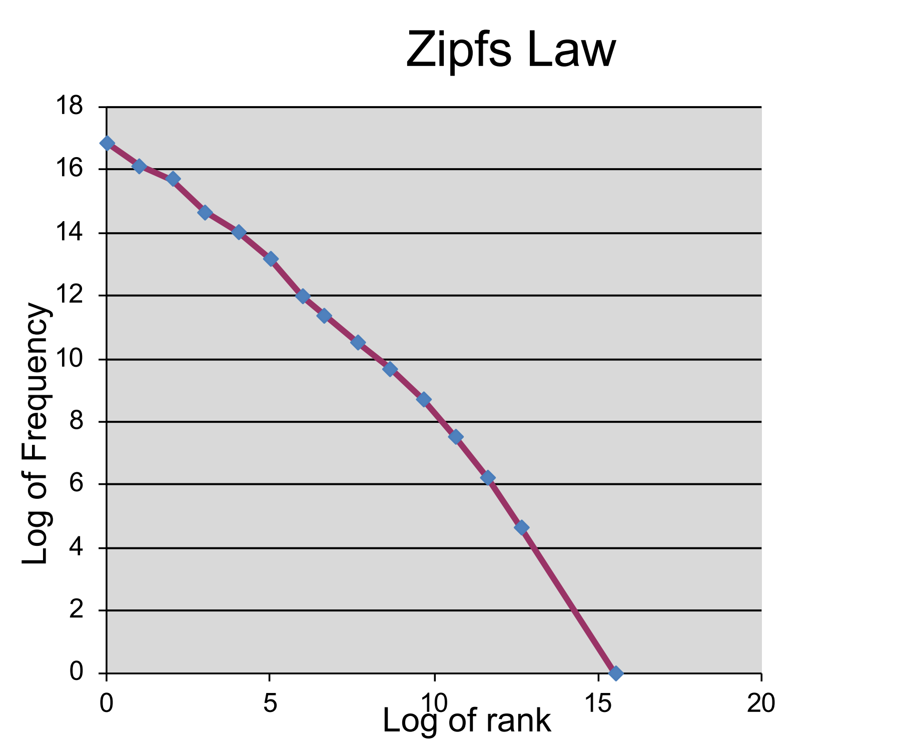
Zipf's law states that:

If words are ranked in descending order of frequency the product of a words frequency, and its rank, is “roughly” the same for all words.

Zipf's Law has been shown to hold true across languages

The table below is data from a Reuters document illustrating Zipf's Law.

Word	Rank	Frequency	Rank*Frequency
the	1	120,021	120,021
of	2	72,225	144,450
and	4	53,462	213,848
for	8	25,578	204,624
is	16	16,739	267,824
company	32	9,340	298,880
Co.	64	4,005	256,320
quarter	100	2,677	267,700
unit	200	1,489	297,800
investors	400	828	331,200
head	800	421	336,800
warrant	1,600	184	294,400
Tehran	3,200	73	233,600
guarantee	6,400	25	160,000
tabulating	47,075	1	47,075



Rank for our seven documents:

Where two are more words have the same frequency, rank is assigned randomly. So the word with four occurrences (system) is ranked first. All words with a frequency of 2 are ranked randomly from 2 to 9. The remaining words, with a frequency of 1, are ranked randomly from 10 to 25.

Actual Count of Words

Words	D1	D2	D3	D4	D5	D6	D7	Frequency	Rank
applications	1	0	0	0	0	0	0	1	25
binary	0	0	0	0	1	0	0	1	24
computer	1	1	0	0	0	0	0	2	2
computer_system	0	1	0	0	0	0	0	1	23
engineering	0	0	0	1	0	0	0	1	22
eps	0	0	1	1	0	0	0	2	3
generation	0	0	0	0	1	0	0	1	21
graph	0	0	0	0	0	1	1	2	4
human	1	0	0	1	0	0	0	2	5
interface	1	0	1	0	0	0	0	2	6
intersection	0	0	0	0	0	1	0	1	20
machine	1	0	0	0	0	0	0	1	19
management	0	0	1	0	0	0	0	1	18
minors	0	0	0	0	0	0	1	1	17
opinion	0	1	0	0	0	0	0	1	16
ordered	0	0	0	0	1	0	0	1	15
paths	0	0	0	0	0	1	0	1	14
random	0	0	0	0	1	0	0	1	13
response_time	0	1	0	0	0	0	0	1	12
survey	0	1	0	0	0	0	1	2	7
system	0	1	1	2	0	0	0	4	1
testing	0	0	0	1	0	0	0	1	11
trees	0	0	0	0	1	1	0	2	8
user	0	1	1	0	0	0	0	2	9
user_interface	0	0	1	0	0	0	0	1	10

Creating Document vectors

Step 5: using counts to reducing bag of words

. . .so how can rank be used to prune terms in the document vector?

Words with a high ranking will tend to appear in all documents, and so are not useful in determining what a document is about.

These words are the stop words, and can be eliminated from the dataset by removing the top *n*% of words from the list.

Question: up to what frequency should you include in the list of stop words?

- If *n* is too high, useful words will be eliminated as stop word
- If *n* is too low, words that are not useful will be included in the index

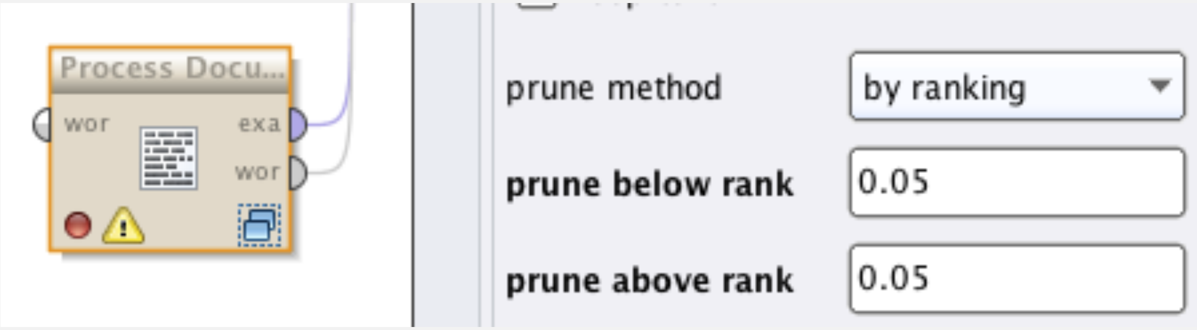
Creating Document vectors

using counts to reduce bag of words

What about words with low ranking ?

- If words have too low a ranking, particularly in large document collections, it means they are specific to a small number of documents, but are unlikely to be useful in classifying larger groups.
- These terms can therefore also be removed.

Process documents in Rapid-miner allows for term pruning by rank, absolute count, or percentage of documents which have the term.



prune method: by ranking

prune below rank: 0.05

prune above rank: 0.05

Recap on steps done so far:

１．Extract content words

２．Do a simple count of the frequencies of terms

３．Adjust the frequencies by the IDF weighting to increase rank of less frequent terms.

４．Normalise the frequency to values between zero and one (TF-IDF).

５．Remove low and high frequency terms to reduce the size of the bag of words

Rapid Miner

- Open up your Rapid Miner process from Lab 2.

- Select the Process Document from Data operator and view the options for its vector creation operator:

  - TF-IDF – normalised counts adjust by IDF

  - TF – normalised counts

  - Term occurrences – basic term count.

  - Binary Term occurrences – binary value of 0 (term not in documnet) or 1 (term in document).

Latent Semantic Indexing

At this stage we have an index which contains terms that heuristically are the most useful terms for describing a document by:

1. Calculating word frequencies
2. Weighting those frequencies based on the number of documents those words appear in
3. Removing the highest and lowest ranked terms

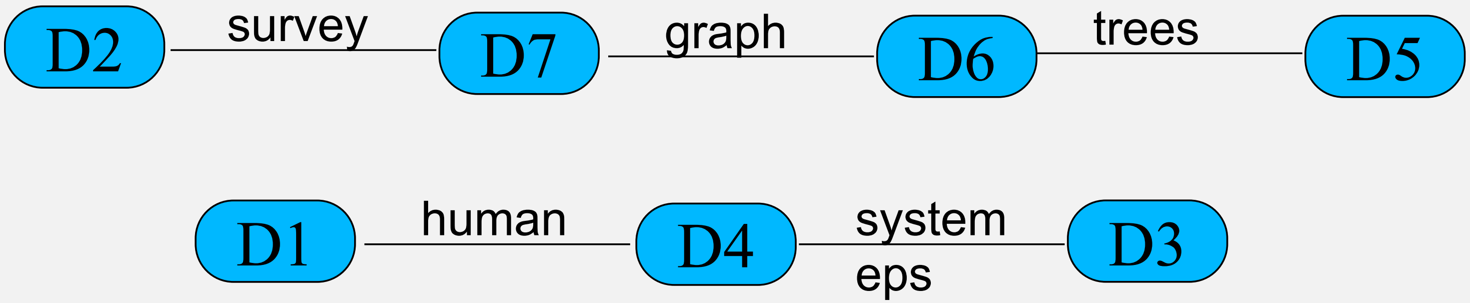
This approach captures global relationships, by counting the occurrences of words.

LSI builds relationships based on co-occurring words in multiple documents.

Rather than relying on exact word matches to link documents, it links documents based on common topics or concepts. This gets around the issue of **synonyms**.

Latent Semantic Indexing

Latent semantic indexing works by linking documents that share common words. Although the calculations are done using matrices, what is happening can be illustrating using a graph. Take the graph below of the seven documents used in previous examples:



This network of documents has two subnetworks, illustrating that there are possibly two topics covered in the seven documents.

The two subnetworks are unconnected, illustrating that the two topics under discussion are unrelated.

Looking at the original seven documents, are the assumptions above about two unrelated topics correct?

Latent Semantic Indexing

D7 shares the term survey with D2 and the term graph with D6.

As a result, all content words in both D2 and D6 are added to the vector for D7, but with a reduced weighting.

Content words in D5 are also included in the vector for D7 but with a further reduction in weight. This is because D5 is two links from D7 through its common term trees with D6.

The full document vector is given on the next slide.

Notice there are less zero entries now.

D7, which had three content words (graph, minor, survey) now has 14.

Latent Semantic Indexing:

resulting dataset for our 7 documents.

Clearly highlights two clusters of documents

Words	applications	binary	computer	computer_system	engineering	eps	generation	graph	human	interface	intersection	machine	management	minors	opinion	ordered	paths	random	response_time	survey	system	testing	trees	user	user_interface
D1	0.011	0	0.011	0	0.074	0.197	0	0	0.085	0.011	0	0.011	0.123	0	0	0	0	0	0	0	0.271	0.074	0	0	0.123
D2	0	0.014	0	0.032	0	0	0.014	0.233	0	0	0.072	0	0	0.162	0.032	0.014	0.072	0.014	0.032	0.193	0	0	0.086	0.032	0
D3	0.011	0	0.011	0	0.074	0.197	0	0	0.085	0.011	0	0.011	0.123	0	0	0	0	0	0	0	0.271	0.074	0	0	0.123
D4	0.011	0	0.011	0	0.074	0.197	0	0	0.085	0.011	0	0.011	0.123	0	0	0	0	0	0	0	0.271	0.074	0	0	0.123
D5	0	0.014	0	0.032	0	0	0.014	0.233	0	0	0.072	0	0	0.162	0.032	0.014	0.072	0.014	0.032	0.193	0	0	0.086	0.032	0
D6	0	0.014	0	0.032	0	0	0.014	0.233	0	0	0.072	0	0	0.162	0.032	0.014	0.072	0.014	0.032	0.193	0	0	0.086	0.032	0
D7	0	0.014	0	0.032	0	0	0.014	0.233	0	0	0.072	0	0	0.162	0.032	0.014	0.072	0.014	0.032	0.193	0	0	0.086	0.032	0

ASIDE:

In Rapidminer, Wordnet’s synonyms operator does something like this – synonyms of terms are added to the bag of words. It will add synonyms, with counts, for all words in a document vector, so is increase the number of terms rather.

Indexing formatted documents

- Most documents are formatted (word docs; pdf; web pages).
- Formatting allows the author to highlight words or phrases of significance, which can inform term counts.
- However many document filters convert formatted text to raw text before processing the document, thus losing the information provided by formatting.

- One solution to this is to repeat words which are highlighted by formatting.

Indexing formatted documents

$16{,}384 = 16KB = (2^{14})$

Example of rules for web pages:

R = 2 * (length of web page / 16384) + 1

If text is in bold, big, em, i, h4, strong or font size 4 tag, repeat the term R times.

If text is from title, h3, for font size 5 tag repeat the term (2xR) times.

If text is h2, or font size 6 tag repeat the term (4xR) times.

If text in h1, or font size 7 tag repeat the term (8xR) times.

Indexing formatted documents

www.itb.ie home page source has 1,266 words:

R = 2 * (1,266/16,384) + 1 = **1.15**

msdn.microsoft.com home page source has about 8,252 words:

R = 2 * (8,252/16,384) + 1 = **2.01**

http://www.w3.org/TR/2006/REC-xml11-20060816/ has a word count of about 19,620:

R = 2 * (19,620/16,384) + 1 = **3.4**

A larger document will by definition have an average higher word count for words, and so formatted words will need to be repeated more often.

Evaluation of Indexing methods

1. Normalised frequencies are easily computed, but do not consider relationships between documents, just the frequency in the document itself.
2. Inverse document frequencies take global relationships into account. Words that appear in many documents are given a lower weighting than words that appear in only a few documents. The additional computation required to compute this global relationship is minimal.
3. Latent Semantic indexing focuses on documents related by concepts. It is more computationally intensive, but addresses the problem of synonyms.
4. Word counts for words highlighted by formatting can be increased in accordance with their apparent importance.

None of the methods above address the issue of polysemy (one word, multiple meanings), as they do not attempt to understand a term based on it’s context..

Vector Generation

Note: IDF (and latent semantic indexing) tend to:

✓ result in more accurate results,

but

x the rules generated from such a data set are not as intelligible.

The rules generated by a classification algorithm in text mining can be as informative as the model itself, providing the raw data is in simple frequencies or binary data, e.g. If freq(rugby) > 5 then document in class sport.

x and they are computationally more expensive.

Summary

1.Term Occurrences
Count the occurrences
of a term in
each document

2. Term Frequencies
Normalised the count with respect
to the total number of words in
that document

4. **TF/IDF**
Normalise the IDF
weights

3. IDF
Weight the term based on
how useful it is at describing
the document

Alternative: Latent Semantic Indexing – addresses synonyms

Note: Additional preprocessing needed for formatted documents