

Hons. Degree in Computing

H4016 Text Mining & Information Retrieval

Unit 2 – Preparing text data for data mining

Part I: syntax

Recap on last week: Text Mining Process

5. Analyzing Results

4. Text/Data Mining

- Classification- Supervised Learning
- Clustering- Unsupervised Learning

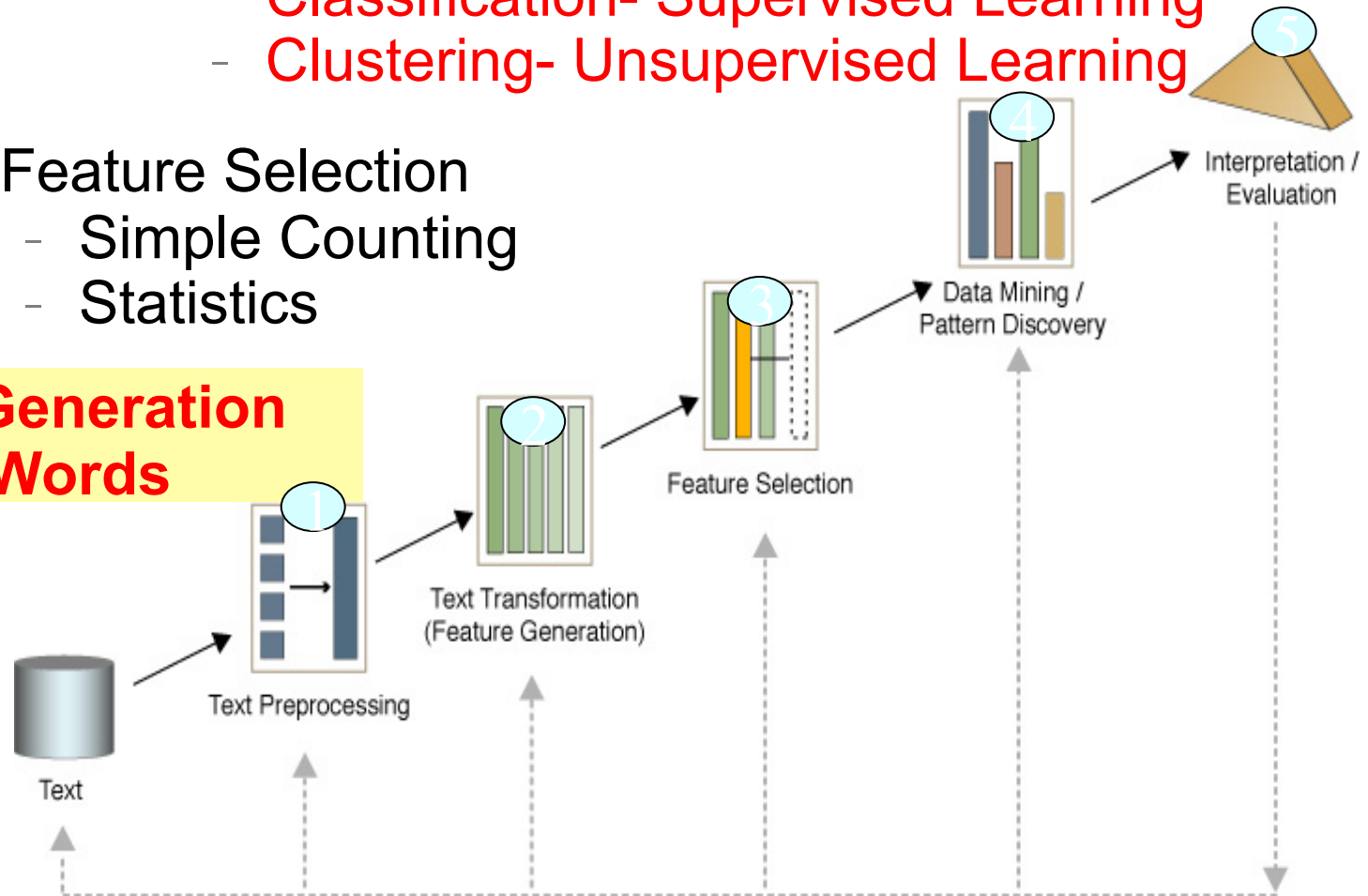
3. Feature Selection

- Simple Counting
- Statistics

2. Features Generation

- Bag of Words

1. Collect text and identify business objective



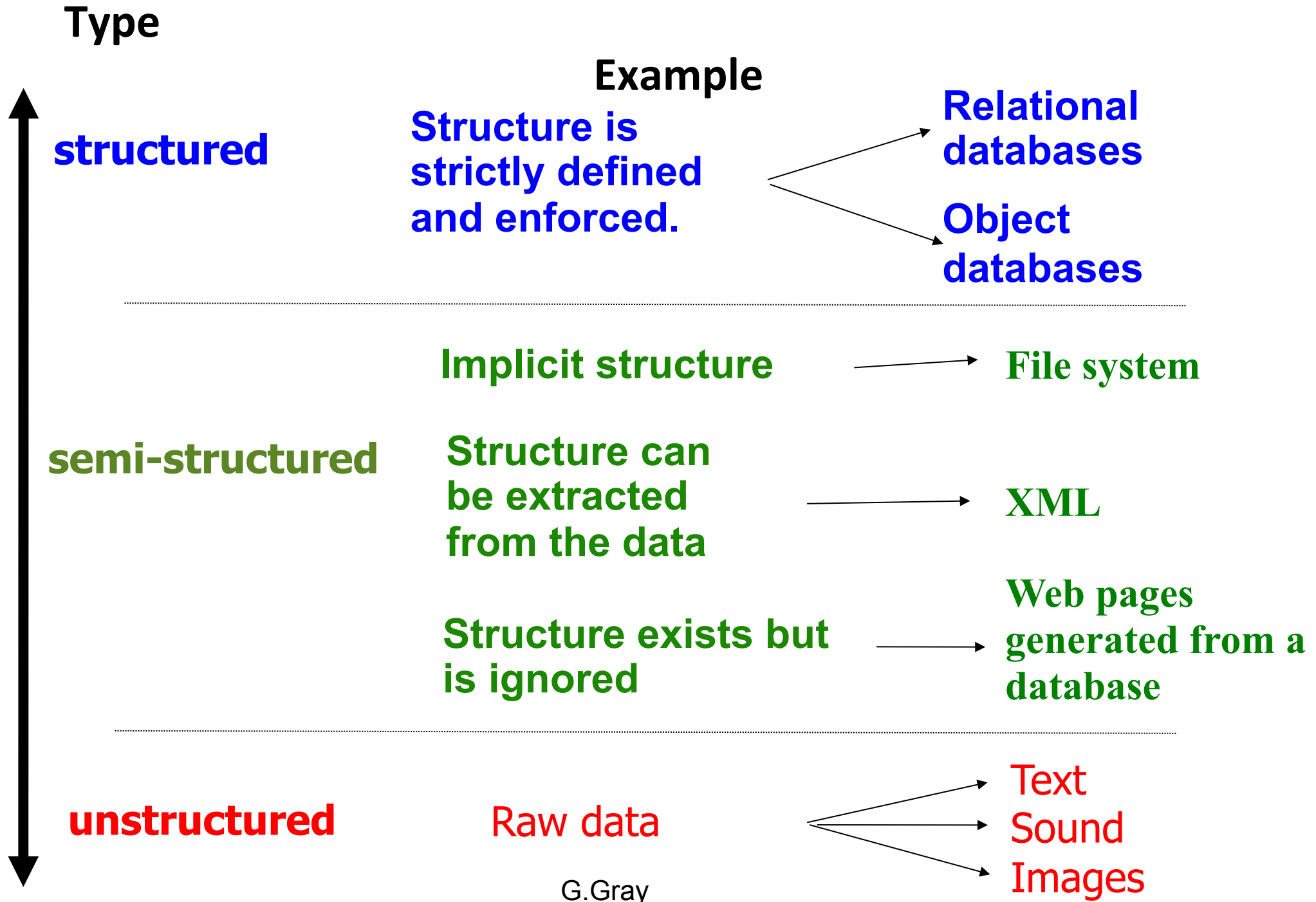
Objectives

- Types of data, and sources of data for text mining
- Text preparation:
 - To understand the NLP techniques used in text mining

Section 1: Setting the context

Types of data
Sources of data for text mining

Types of data



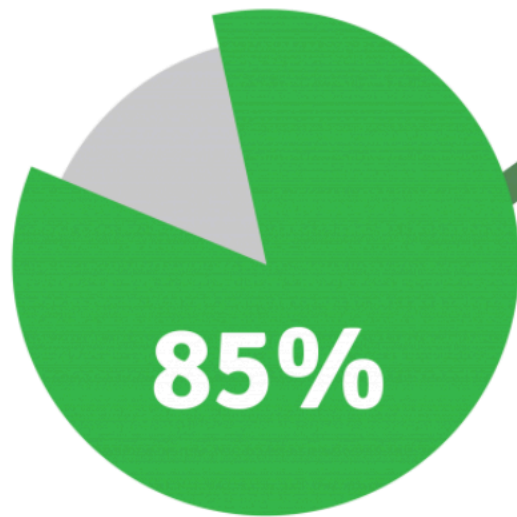
Types of data

- ◆ Structured model: The structure of the data is defined in a schema (data definition), and all data must conform to this schema.
Examples: data in a relational database or object database.
- ◆ Implicit structure: the structure is known in advance, but not stated explicitly in a schema. Examples: file systems, hadoop
- ◆ Structure can be extracted from the data– XML is the easiest form of this, where structure is embedded in the data, other examples may be HTML pages which can be parsed because their structure can be guessed, based on content of data
- ◆ Structure exists but is ignored– web pages/text documents generated from a database, but can not deduce original structure from text.
- ◆ Unstructured – book, research paper, video clip, animated gif, photograph etc . . .

Mining different types of data

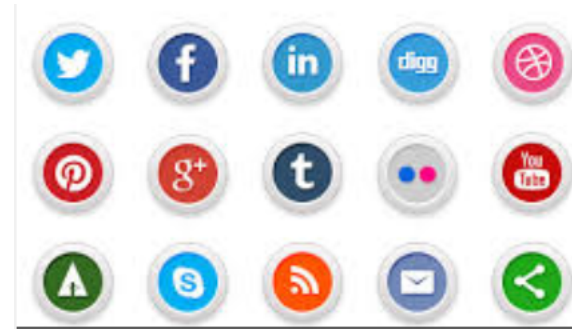
- ◆ **Structured data:**
 - ◆ Most data types can be read directly into a mining tool for data mining.
 - ◆ *Which data types may cause problems?*
- ◆ Data with **implicit** structure:
 - ◆ *Is this data suitable for data mining or text mining?*
- ◆ Structure can be **extracted** from the data
 - ◆ Conversion to structured data can be automated
- ◆ Structure is **ignored**, or there is no **structure**
 - ◆ *. . . now things get more difficult*
 - ◆ Sources of information: Web; intranet; e-mail; text documents; slides; audio files; video files; images; source code; formatted documents;
 - ◆ Data is generally scattered over many computers (unlike structured data which is located in a database or data warehouse).

Str
Un



of BUSINESS DATA is
UNSTRUCTURED

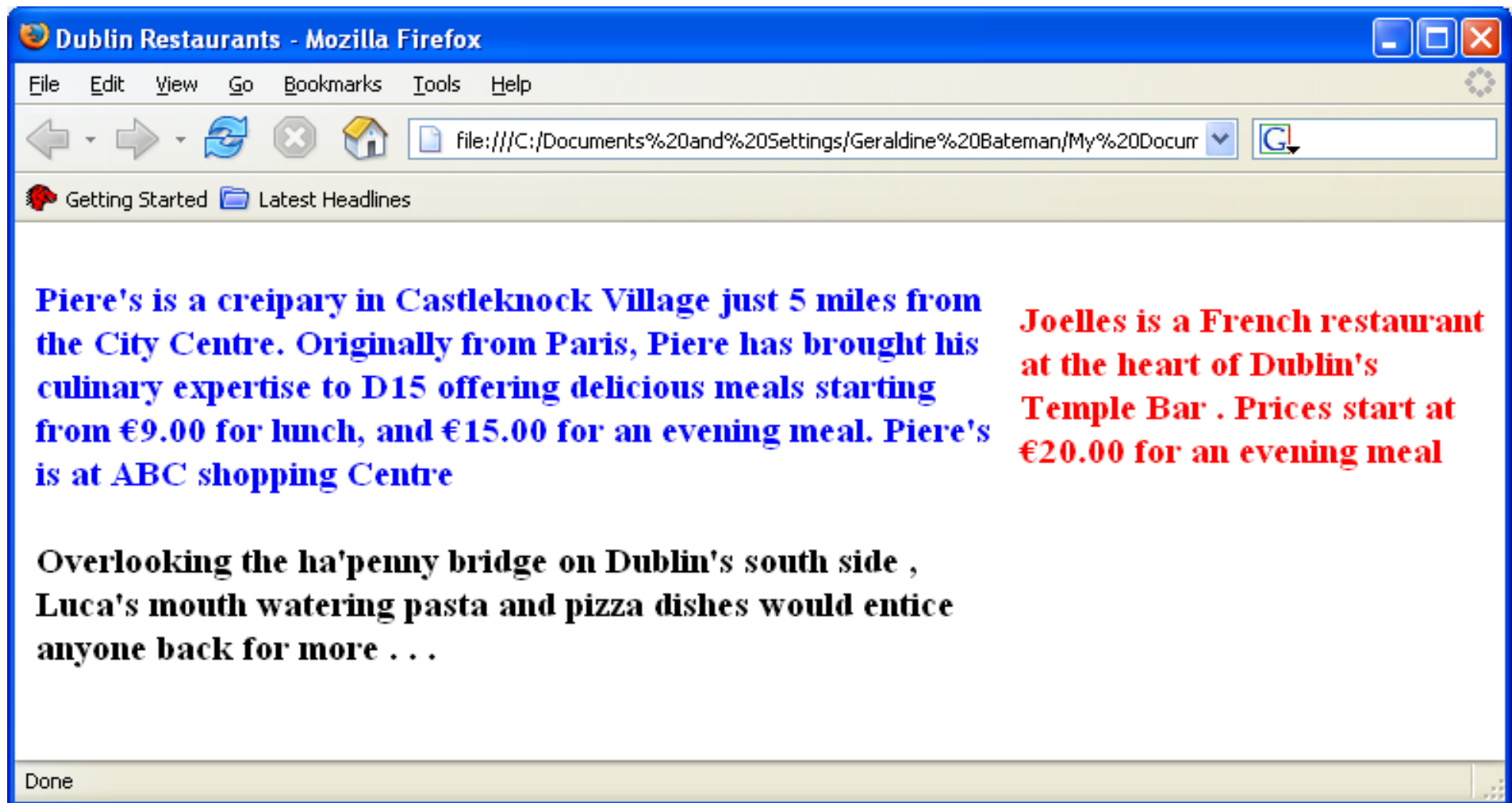
80% - 90% of useful Data is Unstructured!



- ◆ Peoples notes:
 - ◆ Minutes of meetings
 - ◆ Patient records
- ◆ e-mails: e-mails and various text files, which now account to 75 to 85 percent of repository content (S. Feldman, IDC)
- ◆ Web content
 - ◆ News & Information
 - ◆ Corporate sites
 - ◆ Social media
- ◆ Corporate information
- ◆ Research & publications
- ◆ Competitive information
 - ◆ Patents
 - ◆ Reports
- ◆ Database text fields:
 - ◆ Call centre
 - ◆ CRM
- ◆ Open-ended survey responses

Processing unstructured data

Take for example the contents of the three web sites below. Contents for the web site were supplied by each restaurant, with no structure imposed, just guidelines stating that the text should include information on prices, addresses, styles of restaurant and reviews.



Processing unstructured data

- ◆ Suppose you want to write a program to query the contents of a **web site** giving information about restaurants in Dublin.

For example:

List all French restaurants

List all restaurants in TempleBar

- ◆ What would be required to extract this information from the text above?
- ◆ Data embedded in unstructured text is the largest source of data. **Automating the processing of such information sources is not trivial.** Parsers have to be tolerant of variations in format, and accept that there will be chunks of text that can not be parsed.

Section 2: NLP techniques for concept extraction – i.e. extract terms from unstructured text

Text mining is generally not concerned with determining the meaning of a sentence, but more with what the document is about, what are its key concepts / key words, so that it can be related to other, similar documents.

NLP techniques are used to extract these concepts, words or phrases, and to recognise terms that mean the same thing.

Bag of Words

The simplest approach to extracting terms from text is to generate a 'bag of words'. This is a list of all words in the text, and can include / exclude numbers and non-alphabetic characters. This list can optionally include:

- ◆ abbreviations
 - ◆ with or without a full stop (Ltd Ltd.)
 - ◆ With numbers (1.5m 1.7bn)
 - ◆ With spaces (Sw F for Swiss Francs)
 - ◆ Emoticons :)
 - ◆ Abbreviations separated by non-alphabetic characters (AT&T)
- ◆ Internet tokens (127.0.0.1 www.itb.ie geraldine.gray@itb.ie)
- ◆ Numbers (500 500.1 5,000 -500.1 +500.1 500.1E15)

Compound tokens make up about 3% of a text on average, hyphen tokens are generally less than 1% of a text

Token Assembly

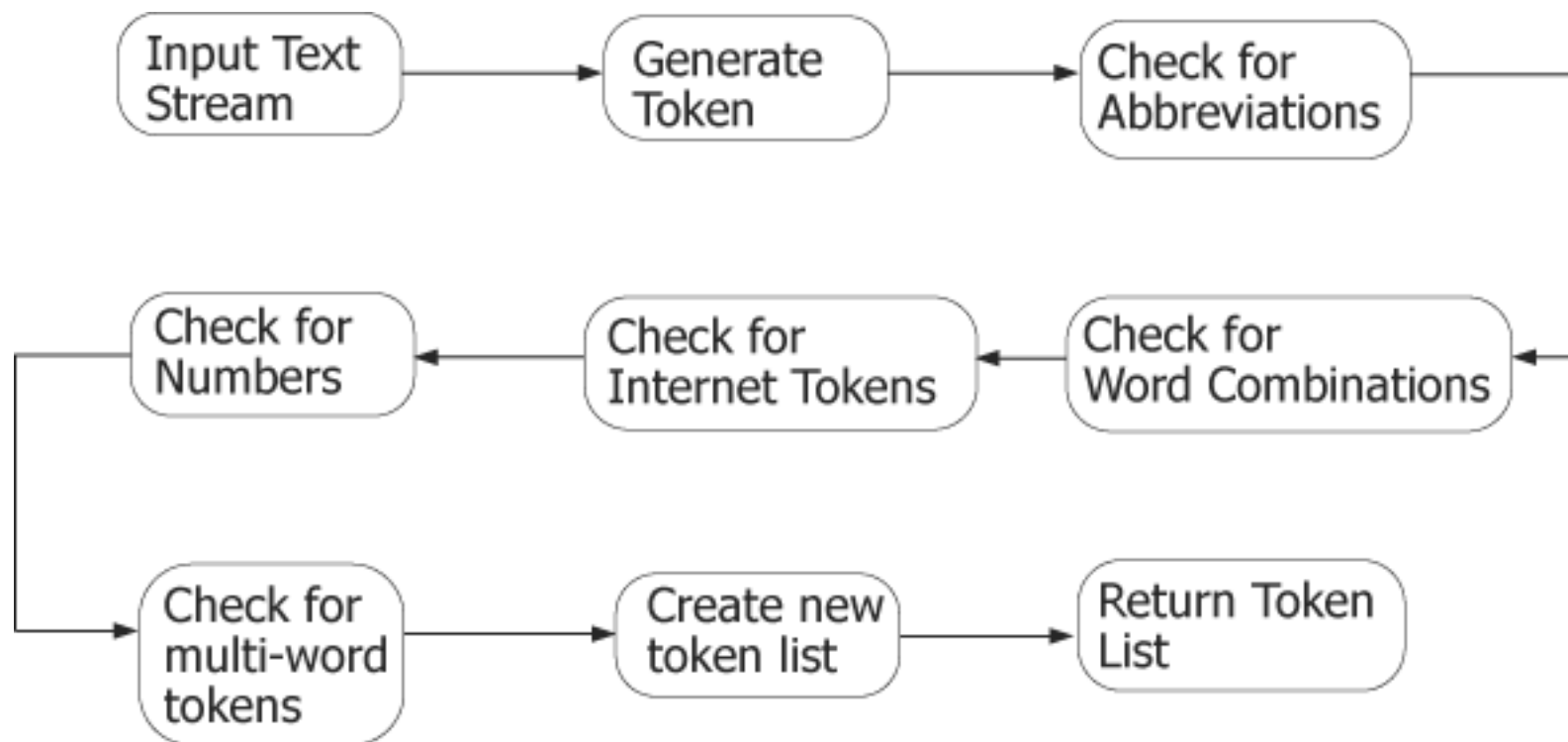


Figure 3.1, *Konchady, Text Mining Application Programming*

Example of Bag of Words

The generated list is generally in alphabetic order.
Immediately you have **lost** meaning

The United
States of
America is at
war

america
at
is
of
states
the
united
war

The phone you
got is not
working

got
is
not
phone
The
working
you

Thks 4 the
fone

4
fone
the
thks

Issues:

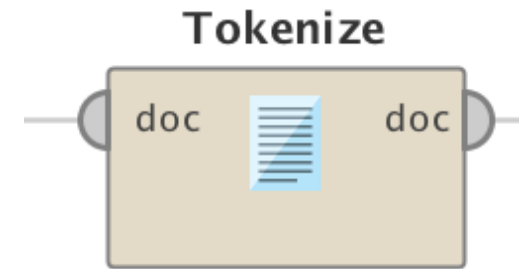
Tokenisers

First draft at a bag of words

Generating a bag of words in Rapidminer - String tokenisers

- ◆ Rapid miner provides three tokenisers:
 - ◆ a standard string **tokeniser** to convert a file of text into words.
 - ◆ **generate n-gram (characters)** to extract strings of a certain length
 - ◆ **generate n-gram (terms)** to extract compound terms of more than one word

Tokeniser



Split a document into individual tokens (words).

The default is to split on any non-letter character, for example:

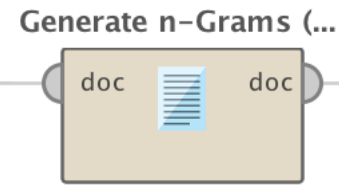
- “*off-campus students can use IP address 194.66.82.11*” becomes:

address, campus, can, IP, off, students, use

- Alternatively, users can provide a list of characters that indicate the end of a token. (*space, . : ;* etc.)

This is generally the first operator used after reading in a document

Character N-gram tokeniser



- ◆ With **character n-gram tokenisers**, words are not considered the basic token unit. Instead, character substrings are used, typically of a fixed length, ***n***.
- ◆ For example, 4-character n-grams (4-grams) for the text 'Johns Hopkins' would be: **#joh, john, ohns, hns#, ns#h, s#ho, #hop, hopk, opki, pkin, kins, and ins#**
- ◆ Storing 12 entries instead of just two (one for each word) means an n-gram approach suffers from performance issues. **However, this expense enables flexible matching.** Web log analysis shows that roughly 30% of users mistakenly type "john hopkins" (missing an s) and yet 8 out of the 12 n-grams still match this string.

So, where the **quality of the text may be an issue, an n-gram tokeniser may perform better than a string tokenisers**

Term N-gram tokeniser

- ◆ A term n-gram tokeniser combines adjacent tokens, and would be used after a string tokeniser. For example a 2-gram term tokeniser combines two adjacent terms; a 3-gram term tokeniser combines three adjacent terms.

Example, using a 2-gram term tokeniser, **after a string tokeniser**:

“off-campus students can use IP address 194.66.82.11” generates:
address, campus, can, IP, off, students, use, off_campus,
campus_student, student_can, can_use, use_IP, IP_address

Filters

Reducing the size of the bag of words

Stop words

- ◆ A stopword is any term that is deemed too common/frequent in text to be of value in text mining.

The stop words in the sentence above are:

a, is, any, that, too, in, the, to, of

- ◆ Stop words are generally removed from the list of tokens, however there is a danger that meaning will be lost if stop words are removed too soon.

[Click to view MySQL stop word list](#)

Example:

- ◆ My phone is not working becomes phone working
- ◆ My phone is working becomes phone working

In a CRM environment, such as a call centre, knowing the difference between the two sentences above is important.

Filters & stopwords

RapidMiner provides a range of filters for removing tokens including:

1. Filters that remove common **stop words** in a number of languages including **English, German, Arabic, French and Czech**.
2. A **custom list of stop words** can be provided in a text file, all of which will be removed from texts (**Filter stopwords (dictionary)**).
3. There is a filter operator that removes words smaller than a specified **length**, or larger than a specified **length**. **For example, remove all words with less than three characters**.
4. And a filter to remove tokens based on their **Part of Speech** (noun, verb, adjective etc.)
5. Filter tokens based on **content** within the token, or before/after (**region**) the token, defined as a regular expression

Filter Tokens (by Length)

Filter Tokens (by Content)

Filter Tokens (by Region)

Filter Tokens (by POS Ratios)

Filter Tokens (by POS Tags)

Filter Stopwords (Dictionary)

Filter Stopwords (English)

Filter Stopwords (German)

Filter Stopwords (French)

Filter Stopwords (Czech)

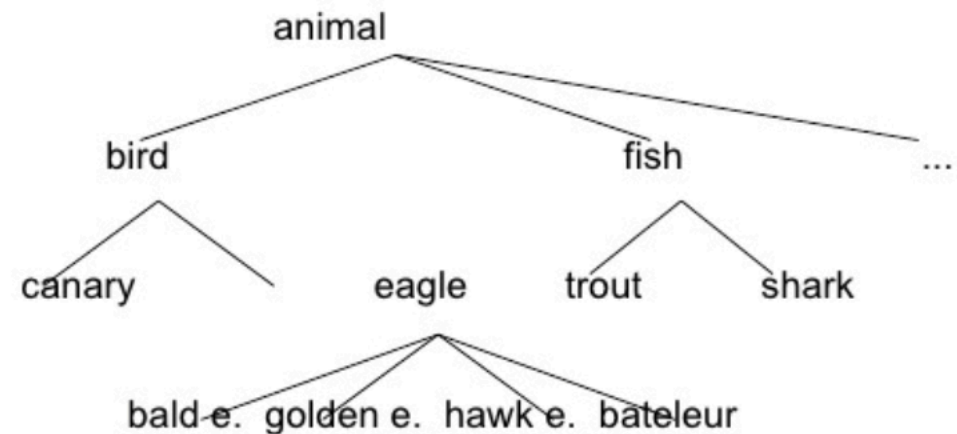
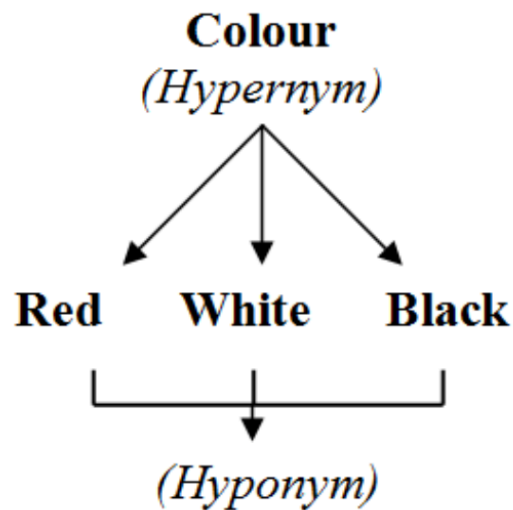
Filter Stopwords (Arabic)

Improving terms in the bag of words

Definitions

Synonyms are words that have the same **meaning** or almost the same, e.g. **afraid** & **scared**; or **father** & **dad**

A **hyponym** is a term whose semantic meaning is included in its more general **hyperonym**. For example:



Homonyms are words that sound the same, and are sometimes spelled the same, but have different meanings, e.g. **die** & **dye**; **Brake** & **Break**; **pen** & **pen**

Improving on Bag of Words

Improving on bag of words can include any of the following:

- 1) Recognise **Collocations** (expressions) & phrases, e.g. **with respect to, second hand, drop in**
- 2) Recognise **synonyms** and **hyponyms** – **phone, fone, telephone, mobile.**
- 3) **Concept typing**: recognising concepts as **person, place, business** etc.
- 4) Link words with the same **canonical form** – 'I **am**', 'you **are**', 'he **is**' are all from the verb 'be'.
- 5) Capture **sentiment** – the mood or intent of the text.
- 6) Recognise **polysemy** – e.g. The word '**state**' can have many meanings (next lecture).

1. Recognising phrases

Phrases are found by looking for patterns in the parts of speech (POS).

Content words:

- Nouns
- Adjectives
- Verbs
- Adverbs

Function words:

- Conjunctions (and)
- Determiners (both)
- Pronouns (he)
- Prepositions (by)

Many phrases can be identified by looking for a combination of terms that follow a pattern such as:

- **Noun** followed by a **noun**, e.g. evening news, bird feather
- An **adjective** following by a **noun**, e.g. united states, bifocal lenses
- An **adverb** followed by a **verb**, e.g. abruptly ended, expertly developed

Find POS combinations that make good phrases

The following steps can determine what parts of speech combine to make a phrase. This would be done ONCE for a particular domain (i.e. it is not needed for every text mining project)

Step 1: The first step is to tag all words with their parts of speech. This can be done by reference to a list of terms that are already tagged with their POS. Most text mining tools can do this.

Note: Most words have just one part of speech. We will look later at an approach to parsing words with more than one possible part of speech.

Part of speech tagging

A sentence such as the following . . .

Using a tool like Rapid Miner is a great idea for any organization that is interested in maintaining information on competitive intelligence.

could be tagged using a tag set such as:

a: adjective	b: adverb	c: preposition
d: determiner	n: noun	v: verb
o: coordination	p: participle	s: stop word

Etc.

giving

Part of speech tagging

Using	a	tool	like	Rapid	Miner	is	a	great
V	D	N	C	A	N	V	D	A

idea	for	any	organization	that	is	interested	in	maintaining
N	C	D	N	D	V	P	C	V

information	on	competitive	intelligence.
N	C	N	N

Extracting Concepts

Step 2: Identify phrases, and their corresponding parts of speech.

Rapid Miner	(adjective, noun)
Great idea	(adjective, noun)
Maintaining information	(verb, noun)
Competitive Intelligence	(noun, noun)

This sentence identified three POS combinations that may identify a phrase:

- NN (noun, noun)
- AN (adjective, noun)
- VN (verb, noun)

Concept Extraction

Concepts tend to be Noun based or Verb based, and can be longer than two words.

Exercise:

1. Convert the following statements into parts of speech. You can ignore function words.
 - Do a short survey on student's opinion on machine response time.
 - He was peddling furiously up the north hill
2. Using the patterns from the last slide, what phrases would be identified? (NN, AN, VN)
3. Are there other patterns that might identify phrases?

Parts of Speech in RM

Rapidminer uses the 36 parts of speech tags as defined by the PENN tagging system from **Pennsylvania (Penn) Treebank Tag-set**:

CC	Coordinating conjunction	NNS	Noun, plural	TO	to
CD	Cardinal number	NNP	Proper noun, singular	UH	Interjection
DT	Determiner	NNPS	Proper noun, plural	VB	Verb, base form
EX	Existential <i>there</i>	PDT	Predeterminer	VBD	Verb, past tense
FW	Foreign word	POS	Possessive ending	VBG	Verb, gerund or present participle
IN	Preposition or subordinating conjunction	PRP	Personal pronoun	VBN	Verb, past participle
JJ	Adjective	PRP\$	Possessive pronoun	VBP	Verb, non-3rd person singular present
JJR	Adjective, comparative	RB	Adverb	VBZ	Verb, 3rd person singular present
JJS	Adjective, superlative	RBR	Adverb, comparative	WDT	Wh-determiner
LS	List item marker	RBS	Adverb, superlative	WP	Wh-pronoun
MD	Modal	RP	Particle	WP\$	Possessive wh-pronoun
NN	Noun, singular or mass	SYM	Symbol	WRB	Wh-adverb

Exercise: What regular expression would cover all POS tags for:

Nouns: _____

Adjectives: _____

Verbs: _____

Adverbs: _____

2. Recognising synonyms


Synonym – More than one word can be used to express the same thing, e.g.

◆ **Exuberant** has 4 synonyms: **lush; luxuriant; profuse; riotous.**

- ◆ A general thesaurus, or a specialised lexicon, can be used to identify synonyms, and convert them into one common word.
- ◆ On average, 1.7 words are used to describe a concept in English.

Aside:

Old words are finding new meanings and new words are continually being coined, e.g.

2015 word of the year: 

2014 word of the year - **Vape**: to inhale and exhale vapour produced by an electronic cigarette.

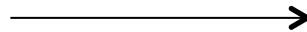
2013 word of the year: **selfie**

2006 word of the year: to be **plutoed**, meaning demoted.

Recognising synonyms

- ◆ Wordnet from Princeton is a lexical database for English with an emphasis on synonyms: <http://wordnet.princeton.edu>.
- ◆ Nouns, verbs, adjectives and adverbs are grouped into synonym sets.
- ◆ Words are linked according to lexical and conceptual relations, creating a net
- ◆ Run has 57 entries / meanings

Use WordNet
online



Word to search for:

Display Options:

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations
Display options for sense: (gloss) "an example sentence"

Noun

- **S: (n)** **anger**, **choler**, **ire** (a strong emotion; a feeling that is oriented toward some real or supposed grievance)
 - **direct hyponym** / **full hyponym**
 - **S: (n)** **fury**, **rage**, **madness** (a feeling of intense anger) "*hell hath no fury like a woman scorned*"; "*his face turned red with rage*"
 - **S: (n)** **infuriation**, **enragement** (a feeling of intense anger)
 - **S: (n)** **umbrage**, **offense**, **offence** (a feeling of anger caused by being offended) "*he took offence at my question*"
 - **S: (n)** **indignation**, **outrage** (a feeling of righteous anger)
 - **S: (n)** **huffiness** (a passing state of anger and resentment)
 - **S: (n)** **dander**, **hackles** (a feeling of anger and animosity) "*having one's hackles or dander up*"
 - **S: (n)** **bad temper**, **ill temper** (a persisting angry mood)
 - **S: (n)** **annoyance**, **chafe**, **vexation** (anger produced by some annoying irritation)
 - **direct hypernym** / **inherited hypernym** / **sister term**
 - **derivationally related form**
- **S: (n)** **anger**, **angriness** (the state of being angry)
- **S: (n)** **wrath**, **anger**, **ire**, **ira** (belligerence aroused by a real or supposed wrong (personified as one of the deadly sins))

Verb

- **S: (v)** **anger** (make angry) "*The news angered him*"
- **S: (v)** **anger**, **see red** (become angry) "*He angers easily*"

Note: The wordnet extension for Rapidminer can read Wordnet dictionaries directly

Other variations in words . . .

In addition to synonyms, words of the same meaning may vary in other ways, for example:

- ◆ **Spellings can differ by region:**

- ◆ Tumour = tumor

- ◆ **Equivalent expressions:**

- ◆ Cancer of the thyroid = thyroid cancer

- ◆ **Short versions:**

- ◆ ziff-davis inc = ziff davis

- ◆ **Incorrect spellings**

- ◆ Technichal support = technical support

String comparisons using Rapid Miners N-gram (characters) can be used to spot these variants.

Dictionary Stemmer

- ◆ RapidMiner's **DictionaryStemmer** reads in a text file that lists **synonyms**, and the base form that each should be replaced with in the form:

- ◆ **<base_form>:<expression>**

Where **expression** can be a word, or a regular expression (Java's Regex).

Example:

kenya:kenyan.*

kenya:molo.*

tribe:tribal.*

province:munster

province:leinster

province:ulster

province:connacht

Note: The **Replace Tokens** operator allows you do the same thing, without using an external file.

3. Concept Typing

- ◆ After Concepts are extracted, they can optionally be classified using a lexicon or dictionary to identify, for example:
 - ◆ **O : Organisation, company or group**
 - ◆ **D : Product**
 - ◆ **P : Person's name**
 - ◆ **L : Location**
 - ◆ **U : Unknown Type**

Concept Extraction Engines generally return too many concepts for practical use. User needs to:

1. Figure out which concepts are of value
2. Convert these concepts into fields
3. Merge concept data with other structured data in order to perform Data Mining

See IEEE paper: ACE: Improving Search Engines via Automatic Concept Extraction, 2004

Recap on NLP done to date

- ◆ Identify terms
- ◆ Identify phrases
- ◆ Identify synonyms
- ◆ Allocate types to known phrases

How has the list of terms changed?

The United
States of
America is at
war



The phone you
got is not
working



Thks 4 the
fone



4. Link words with the same canonical form

- ◆ It is possible to reduce terms to their base canonical form using stemmers. This would allow, for example, a tool to view **running** and **runner** as both being related to **run**.
- ◆ Opinion on whether or not this is a good idea is divided:
 - ◆ Stemmers are automated. However the inherent irregularities in the English language gives some unusual results. **They can be more successful in other languages.**
- ◆ We will look at two stemmers supported by RapidMiner:
 - ◆ Porters Stemmer
 - ◆ Lovins Stemmer

Rapidminer also has a tool for defining your own stemming rules:
Snowball Stemmer

Porters Stemmers

- ◆ Porters Stemmer is the most popular of the stemming algorithms.
- ◆ **Martin Porter** developed his stemming algorithm in the late '70s, publishing the details in 1979 in his paper “*An algorithm for suffix stripping*” <http://tartarus.org/martin/PorterStemmer/def.txt>
- ◆ “The purpose of the algorithm was to remove common **morphological*** and **inflexional**** endings from words in English.”
- ◆ **Rational:** Terms with a common stem will usually have similar meanings, for example:
 - ◆ CONNECT
 - ◆ CONNECTED
 - ◆ CONNECTING
 - ◆ CONNECTION
 - ◆ CONNECTIONS

***Morphology:** the pattern of word formation in a particular language; the study of form

****Inflexion:** alteration of the form

Porters Stemmer

- ◆ Such term can be converted into a single term by removing the various suffixes **-ED, -ING, -ION, IONS** to leave the single term CONNECT.
- ◆ In addition, the suffix stripping process will **reduce** the total **number of terms**, and hence reduce the number of attributes in the dataset, which is generally a good thing.
- ◆ The **rules** for removing a suffix are given in the form

(condition) S1 -> S2

This means that if a word ends with the suffix S1, and the stem before S1 satisfies the given condition, S1 is replaced by S2. The condition is usually given in terms of **m**, a count of the numbers of consonants and vowels remaining when the suffix is removed.

Porters Stemmer

The details:

- ◆ Let v represent a group of consecutive vowels, and c represent a group of consecutive consonants, then

Decided = $cvcvcvc$

start = cvc

Open = $vcvc$

- ◆ m is the number of vc groups in the word

A word can therefore be defined as

$[c] \ vc \ \{m\} \ [v]$

Decided = $c \ vc\{3\}$ $m=3$

Decide = $c \ vc\{2\} \ v$ $m=2$

start = $c \ vc\{1\}$ $m=1$

Open = $vc\{2\}$ $m=2$

G.Gray

Vowels:

a

e

i

o

u and

y (if preceded
by a consonant)

Porters Stemmer

- ◆ What is **m** in each of the following terms?
 - ◆ TR,
 - ◆ EE,
 - ◆ TREE,
 - ◆ BY,
 - ◆ TREES
 - ◆ TROUBLE,
 - ◆ OATS,
 - ◆ IVY
 - ◆ TROUBLES,
 - ◆ PRIVATE,
- ◆ Other conditions:
 - ◆ *S - the stem ends with S (and similarly for the other letters).
 - ◆ *v* - the stem contains a vowel.
 - ◆ *d - the stem ends with a double consonant (e.g. -TT, -SS).
 - ◆ *o - the stem ends cvc, where the second c is not W, X or Y (e.g. -WIL, -HOP).

Examples of rules (condition) S1 -> S2:

- | | | |
|----|--|-------------------------|
| 1) | SSES -> SS | caresses -> caress |
| 2) | IES -> I | ponies -> poni |
| | | ties -> ti |
| 1) | S -> | cats -> cat |
| 2) | (m>0) EED -> EE | feed -> feed |
| | | agreed -> agree |
| 1) | (*v*) ED -> | plastered -> plaster |
| | | bled -> bled |
| 1) | (*v*) ING -> | motoring -> motor |
| | | sing -> sing |
| 1) | AT -> ATE | conflat(ed) -> conflate |
| 2) | BL -> BLE | troubl(ed) -> trouble |
| 3) | IZ -> IZE | siz(ed) -> size |
| 4) | (*d and not (*L or *S or *Z)) -> single letter | |
| | | hopp(ing) -> hop |
| | | tann(ed) -> tan |
| | | fall(ing) -> fall |
| | | hiss(ing) -> hiss |
| | | fizz(ed) -> fizz |

Porters rules

See moodle for:

- ◆ A more complete list of rules (from <http://tartarus.org/martin/PorterStemmer/def.txt>)
- ◆ Pseudocode for the algorithm (not in the attachment, but can be found at: <http://people.ischool.berkeley.edu/~hearst/irbook/porter.html>)
- ◆ An implementation in Java (from <http://tartarus.org/martin/PorterStemmer/>)

Lovins Stemmer

- ◆ The first ever published stemming algorithm was: *Lovins JB (1968) Development of a Stemming Algorithm. Mechanical Translation and Computational Linguistics*
- ◆ The design of the algorithm was much influenced by the technical vocabulary with which Lovins found herself working (materials science and engineering field).
- ◆ The subject term list may be **slightly limiting** in that certain common endings are not represented (**ements** and **ents** for example), and also in that the **algorithm's treatment of short words**, or words with short stems, can be rather **destructive**.
- ◆ The Lovins algorithm is bigger than the Porter algorithm, because of its **very extensive endings list**. It is also **faster**. It has effectively traded space for time, and with its large suffix set it needs just two major steps to remove a suffix, compared with the eight of the Porter algorithm.

Lovins stemmer

- ◆ The Lovins stemmer has 294 endings, 29 conditions and 35 transformation rules.
- ◆ Each ending is associated with one of the conditions.
- ◆ In the first step, the longest ending is found which satisfies its associated condition, and is removed.
- ◆ In the second step the 35 rules are applied to transform the ending. The second step is done whether or not an ending is removed in the first step.
- ◆ Example: endings and their conditions include:

ationally	B	ionally	A	izationally	B
allically	C	ationally	B	eableness	E
less	A	lily	A	ness	A
				ogen	A

- | | |
|---|------------------------------|
| A | No restrictions on stem |
| B | Minimum stem length = 3 |
| C | Minimum stem length = 4 |
| D | Minimum stem length = 5 |
| E | Do not remove ending after e |

Lovins stemmer

Example – Step 1 (endings and conditions):

- ◆ the word **Nationally** has an ending of *ationally* which is in the list. However this is associated with condition B, which states the remaining stem must be at least length 3. Therefore this ending can not be removed.
- ◆ The word also has an ending of *ionally* which is in the list with condition A – no limitation on stem length.
- ◆ Therefore **Nationally** is stemmed to **Nat**

Example - Step 2 (rules):

- ◆ The transformation rules are similar to Porters, but without the condition. The first three are:

- 1 remove one of double b, d, g, l, m, n, p, r, s, t
- 2 iev -> ief
- 3 uct -> uc

e.g.

- ◆ 1 rubb[ing] -> rub, embedd[ed] -> embed etc
- ◆ 2 believ[e] -> belief
- ◆ 3 induct[ion] -> induc[e]

Lovins stemmer

- ◆ See moodle for a full list of endings and conditions, taken from:
<http://snowball.tartarus.org/algorithms/lovins/stemmer.html>

Stemming and stop words

Stemming stopwords is not useful.

- ◆ Generally speaking, **inflectional*** stopwords exhibit many irregularities, which means that stemming is not only useless, but not possible, unless one builds tables of exceptions into the stemmer.

Stop words and other filters are applied before stemmers

- ◆ More background information on stemmers can be found at <http://snowball.tartarus.org/texts/introduction.html>

*inflectional stop words – stop words that change the meaning or intent of the statement

5. Sentiment analysis

Sentiment analysis, also known as opinion mining, is the analysis of the *feelings* (i.e. attitudes, emotions and opinions) behind words, using natural language processing tools.

It is a subjective impression rather than a fact, and so less precise than matching terms.

Sentiment analysis answers questions such as:

- ◆ Is a product review positive or negative?
- ◆ Is a customer email satisfied or dis-satisfied?
- ◆ Based on tweets, how are people responding to this ad?
- ◆ Have bloggers attitudes about the government changed since the election?

- ◆ Is there an identified bias in a news source?
- ◆ Is this video suitable for children based on comments?

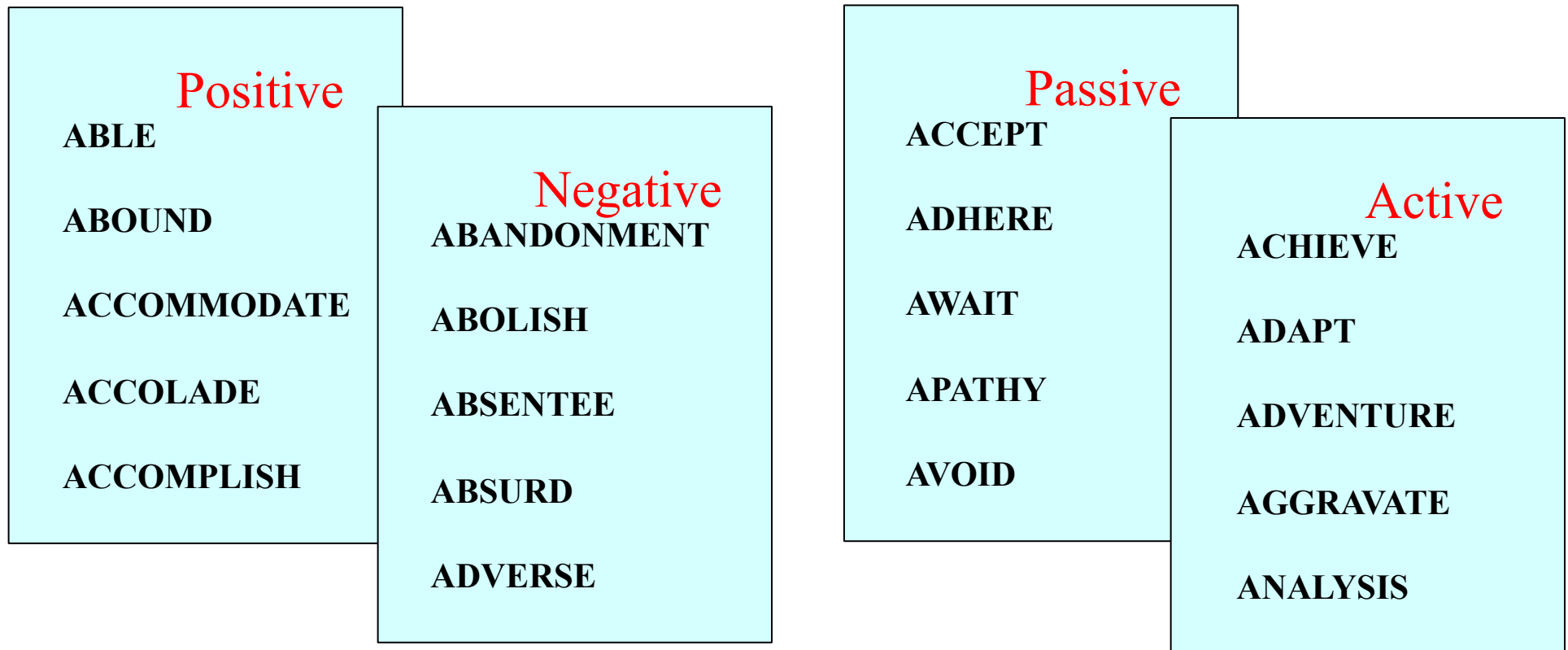
- ◆ Why are customers not buying our product? What do they think of our competitors product?

Sentiment analysis

There are many dictionaries of terms that are tagged with a particular intent, e.g. General Inquirer from Harvard:

<http://www.wjh.harvard.edu/~inquirer/homecat.htm>

When analysing text, a count of the number of terms under each category/intent will give an indication of the sentiment of the text.

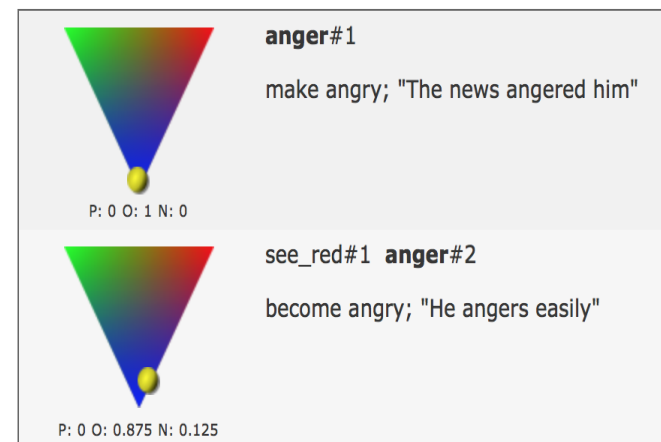
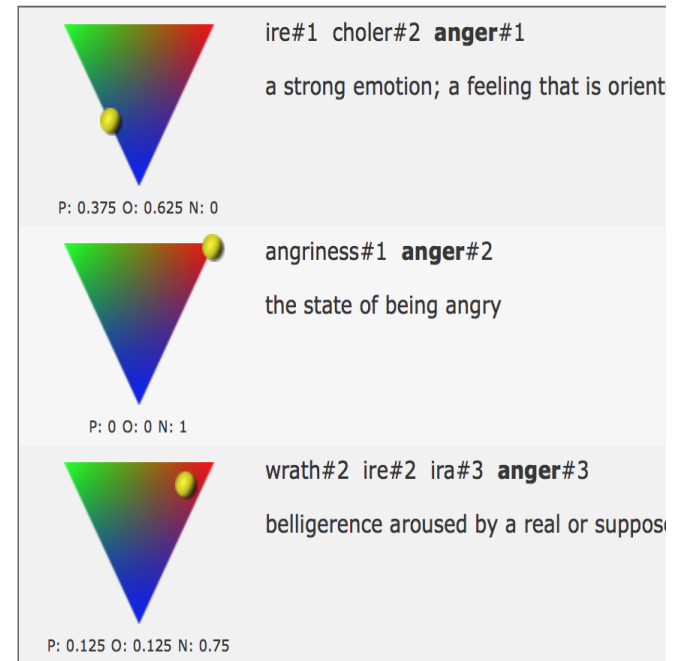


Sentiwordnet

- ◆ One of the most popular is Sentiwordnet, based on Wordnet, which assigns one of three sentiments to words in wordnet, **positive**, **objective** or **negative**.
- ◆ <http://sentiwordnet.isti.cnr.it>












 Extract Sentiment (English)

The Extract Sentiment operator in the Wordnet extension for Rapidminer extracts sentiment using SentiWordNet.



Sentiment analysis

- ◆ Tweets and short texts can be difficult to work with, but if they includes emoticons, sentiment can be derived more easily:

icon	text	text	full text
	:)	:-)	:smile:
	:D	:-D	:grin:
	:(:-(" data-bbox="468 484 491 507"/>	:sad:
	:o	:-o	:oek:
	8O	8-O	:shock:
	:?	:-?	:???:
	8)	8-)	:cool:
	:x	:-x	:mad:
	:P	:-P	:razz:
	:	:-	:neutral:
	:)	;-)	:wink:

Sentiment Analysis

Lists of terms are useful for general comments.

However for a particular domain, such as movie reviews or product reviews, models can be trained to recognise positive and negative reviews based on a set of reviews already labeled as positive, negative or neutral. Including phrases can improve accuracy (e.g. **disappointed with, not good enough, delighted with**).

Take a look at: <http://text-processing.com/demo/sentiment/>
(sentiment analysis using Python)

Analyze Sentiment

Language

english ▾

Enter text

The film tells the inspiring true story of how Nelson Mandela joined forces with the captain of South Africa's rugby team to help unite their country. Newly elected President Mandela knows his nation remains racially and economically divided in the wake of apartheid. Believing he can bring his people together through the universal language of sport, Mandela rallies South Africa's underdog rugby team as they make an unlikely run to the 1995 World Cup

Sentiment Analysis Results

The text is **neutral**.

The final sentiment is determined by looking at the classification probabilities below.

Subjectivity

- **neutral: 0.8**
- polar: 0.2

Recap

1. Read in a corpus is text files (Process document)
2. Tokenise to extract the terms in each document as dataset attributes (tokenise / character n-gram)
3. Remove stop words
4. Experiment with other filters to remove useless terms
5. Experiment with stem dictionaries and stemmers so that terms that mean the same thing can be recognised as the same term

For sentiment analysis, terms can be reduced to emotions such as positive, negative, angry etc.

6. Identify phrases:
 - Reduce the list to nouns, verbs and adjectives
 - Create term n-grams to combine adjacent terms

Can the steps above be done in any order?

