

Unit 5 Web Mining & Information Extraction

Part 1: Web Content Mining

Geraldine Gray / Markus Hofmann

Overview

- Overview of web mining
 - Web content mining
 - Web usage mining
- Knowledge extraction from less structured text

Mining Data from the Web

The topic of web mining covers a number of data sources including:

- **Web Content Mining**

- Text content on web sites
 - Standard sites
 - Blogs
 - News groups and discussions
 - Software
 - Multimedia content on web sites

- **Web Usage Mining**

- Web logs generated from web activity
- Associated databases recording additional information about that activity.

- **Web Structure Mining (Web Link Mining)**

- Analysis of links between web pages

Web Mining Application Areas

- **Product Mining**: automated scanning of a range of sites for product information and price comparisons.
- **Blog & News mining**: analyse blog content to find out what topics are important, and how people feel about them.
- **Sentiment analysis**: analyse the content of reviews to discover what people are saying about a produce or service.
- **Website optimisation**: analyse usage patterns on your website. Design pages and links to optimise the user experience, and optimise sales.

Which of the examples above will require text processing? Are methods covered to date sufficient?

Web Content Mining

- Covering:
 - Web crawlers and
 - Knowledge extraction from less structured text

Web Content Mining

- The aim of *Web Content Mining* is to extract knowledge from data contained within the content of web pages
- **Stages:**
 1. Crawl the web to gather the data
 2. Extract the relevant information from that data
 3. Pre-process the data for data mining
 4. Mine the data
- The following slides cover steps 1 and 2

Step 1: Web Crawling

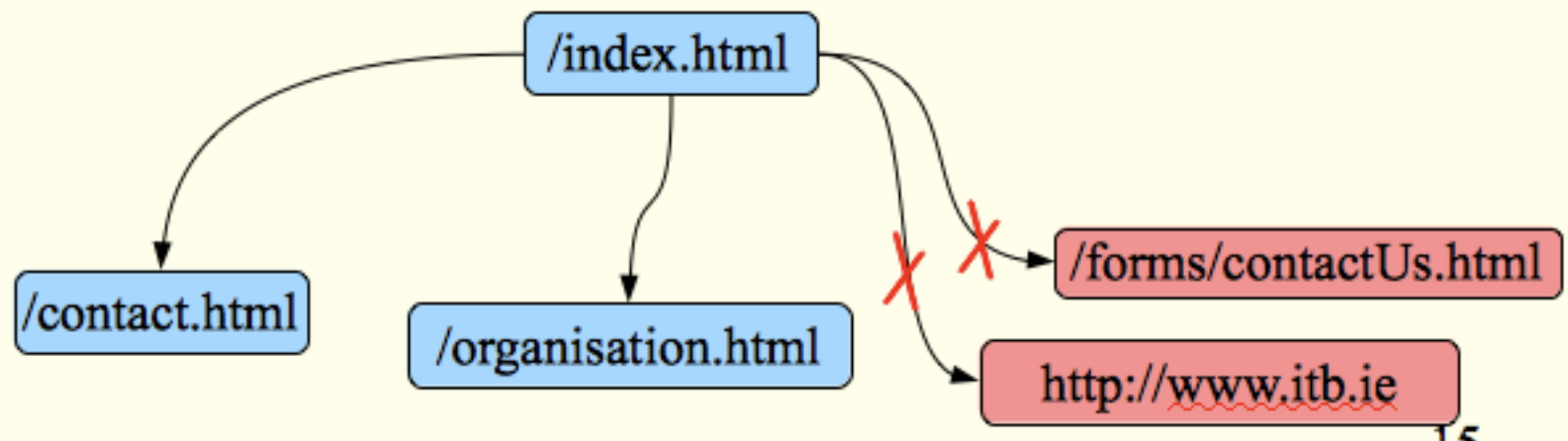
Step 1 in web content mining is to access the content on the web.
This is done using a **web crawler**:

Introduction

- Web crawlers are programs that automatically download web pages
 - Also known as spiders, robots, User Agent, wanderer or bot
- A web crawler can visit many different sites to collect information. This is often a never ending process due to the rapid change and growth of the world wide web
- Well known crawlers used by the search engine include googlebot, bingbot and yahoobot

Crawling the Web

- Information is generally scattered over many pages. A web crawler traverses the graph of linked pages to access the content.
- It is not feasible to gather all content on the web; the web crawling needs to be focused on specific pages which are likely to contain useful information. This can be done by:
 - Limiting the crawler to specific **domains**
 - Limiting the crawler to specific **depths** (the number of slashes in a URL from its site root)



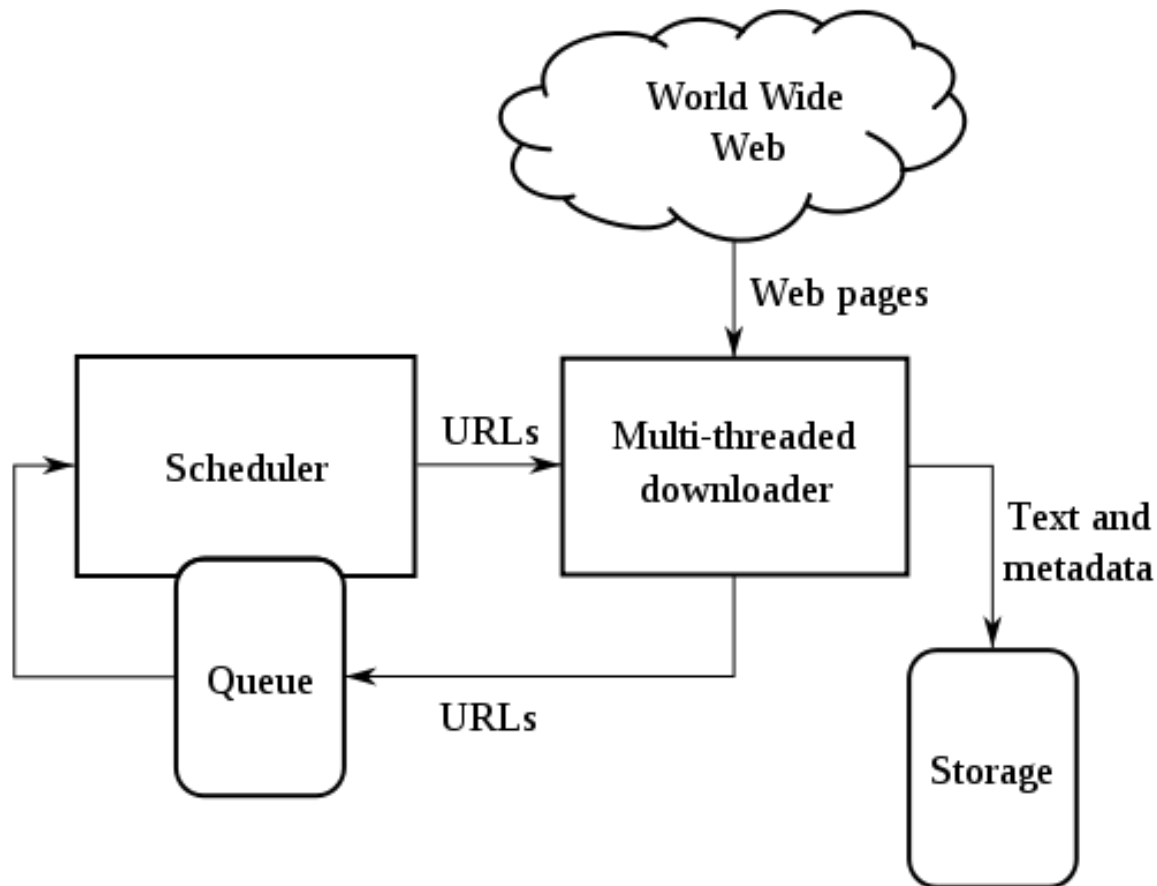
Applications

There are many applications where web crawlers are potentially beneficial

- Collect information from competitors
- Find out about trends on the web
- Build new web services with crawled content
- Harvest email addresses (often malicious)
- Vertical (specialised) search engines
- Most widely used by search engines, with the aim of indexing the entire world wide web

High Level Architecture

- http://en.wikipedia.org/wiki/Web_crawler



Type of Crawlers

1. Universal Crawler

- They are wanderers, designed to gather all pages irrespective of their content
- Large Scale, and Huge Cost in terms of network bandwidth
- Incremental updates to existing data repositories. Heavily indexed.
- Two issues
 - Performance – need to crawl billions of objects
 - Policy – what to prioritise in terms of coverage, freshness and bias towards ‘important’ pages (importance is subjective).

Type of Crawlers

2. Preferential Crawlers / Focused Crawlers / Topical Crawlers

- These crawlers are more targeted and only download pages of a certain topic or type.
- The crawler works with a priority queue rather than the FIFO (first in first out) concept.
- The crawlers assign each universal link a priority based on an estimate of the value of the linked page. The usefulness of the page has to be estimated BEFORE the page is accessed, and can be evaluated based on some thing simple like the anchor text of the link itself, or more complex estimates that train a classifier to determine which links should be prioritised.
- Preferential crawlers that start with a set of seed pages or URLs are often called topical crawlers.

Crawler Ethics

- Crawlers can upset web servers and administrators
 - E.g. Sending too many requests in quick succession can be seen as a Denial of Service attack.
 - This can lead to blacklisting of the IP

Crawler Rules

- You need to identify yourself
- Alternate requests between different hosts
- Look out for re-direction loops
- Look out for spider traps (also known as crawler traps), which are similar to an infinite loop, e.g.
 - A Calendar - dynamically creates links to the next day/month/year.
 - Can be included as design feature to catch out spambots, or be the result of poor development.
- Honour the [robot.txt](#) file which contains information as to who is allowed to access which files/folders

A recent study based on a large scale analysis of robots.txt files showed that certain web crawlers were preferred over others, with Googlebot being the most preferred web crawler

Robots.txt File

- Check out <http://www.springer.com/robots.txt>

....

User-agent: Googlebot

User-agent: Mediapartners-Google

User-agent: Adsbot-Google

User-agent: slurp

....

User-agent: FeedFetcher

Crawl-delay: 2

Disallow: /covers/

Time delay between requests. Crawlers interpret the number differently
Can not enter this directory

...

Allow: /product-search?cps=*&facet=type__book\$

Allow: /product-search?cps=*&facet=type__journal\$

...

extra lines for bing bot only

User-agent: bingbot

Crawl-delay:0

..

all others

User-agent: *

Disallow: /

All other agents are not allowed crawl the site

You are not legally bound to robot.txt files, but you may be blocked if you ignore it

Crawler Ethics

- You are not legally bound to robot.txt files
 - But you may be blocked if you ignore it
- Some crawlers have an identity crisis and pretend to be someone they are not.
 - They give themselves a different name
 - They pretend to be official users by mimicking users' behaviour.
- Servers also can pretend to be someone else and provide different content based on the User-Agent
 - Known as cloaking

RapidMiner Crawler

- RapidMiner's Crawler is based on WebSphinx (<http://www.cs.cmu.edu/~rcm/websphinx/>)
- There are a number of operators which use the web crawler, all of which have the following parameters:
 - **Crawling rules** use regular expressions to limit which links to follow and which pages to process (explained in a later slide)
 - **Max pages**: max number of pages to be downloaded
 - **Max depth**: max depth of the crawling process
 - **Domain**: Specifies if links should be followed to the whole web, or remain in the current domain.
 - **Max threads**: number of crawling threads run in parallel
 - **Max page size** in KB

Rapidminer Web Crawler

Rapidminer Operators:

1. Crawl Web: crawls the web, as per the identified parameters, starting at a user defined root page. Each page found is added to the specified output directory.

2. Process documents from web: as above, except that additional operators can be embedded within this operator to process each web page (e.g. extract content) and so generate a dataset from the web pages, rather than storing the source web pages themselves.



Crawl Web	
url	<input type="text" value="http://www.itb.ie"/>
crawling rules	 Edit List (0)...
<input checked="" type="checkbox"/> write pages into files	
<input type="checkbox"/> add pages as attribute	
output dir	<input type="text" value="web mining/crawler"/> 
extension	<input type="text" value="txt"/>
max pages	<input type="text" value="20"/>
max depth	<input type="text" value="2"/>
domain	<input type="text" value="web"/>
delay	<input type="text" value="1000"/>
then visiting a page in milliseconds. (integer; 0-+∞;	

Rapidminer **crawling rules**

- **store_with_matching_url**: If the regular expression matches the url, this page will be stored in the resulting ExampleSet.
- **store_with_matching_content**: If the regular expression matches page content, this page will be stored in the resulting ExampleSet.
- **follow_link_with_matching_url**: If the regular expression matches the url, the crawler will follow the link and load the url.
- **follow_link_with_matching_text**: If the regular expression matches the text of the hyperlink, the crawler will follow the link and load the url.

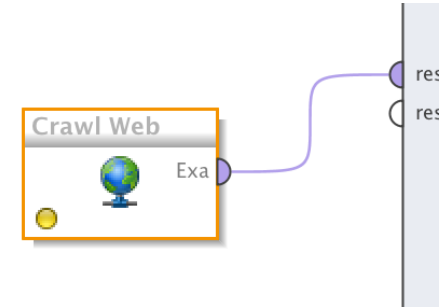
Rapidminer lab work

- Note: the following slides will cover three alternative ways to input data from the web:
 1. Use a **web crawler**, and store web pages in a folder
 2. Use **process documents from web** to create a document vector directly from the results of a web crawler
 3. Create document vectors from a list of URLs using **Get Pages**

Lab work #1

- Create a folder to hold webpages found
- Open up Rapidminer and start a new process.
- Add a **crawl web** operator.
 - Set the URL to <http://www.itb.ie>
 - Set **output dir** to the folder created above
 - Set max pages to 10
 - Leave remaining parameters at their default values
- Connect crawl web to the output port and **run the process**.

As the process is running, look at the log window to see what pages are visited.



url	<input type="text" value="http://www.itb.ie"/>
crawling rules	<input type="button" value="Edit List (0)..."/>
<input checked="" type="checkbox"/> write pages into files	
<input type="checkbox"/> add pages as attribute	
output dir	<input type="text" value="eraldine/Desktop/crawl"/> <input type="button" value="Folder icon"/>
extension	<input type="text" value="txt"/>
max pages	<input type="text" value="10"/>
max depth	<input type="text" value="2"/>

Lab work #2: only store pages that include the words 'student' or 'Student'

Return to the design view, and click on [the crawling rules](#) Edit List parameter.

As the rule application, select 'store_with_matching_content'

Under rule value type: **.*[sS]tudent.***

rule application	rule value
store_with_matching_content ▼	.*[sS]tudent.* 

- **Run** the process. Again, as the process is running, look at the log window to see what pages are visited.
- Once web pages are in folder(s), they can be read into Rapidminer using 'Process Document from File' (as you used in labs 2 and 3).
- The 'Extract Content' operator extracts the content from the HTML page.

Lab work #3

The operator '**Process Documents from Web**' has a built in crawler, and is used if you don't want to store the entire web page, just the key words and other meta data.

1. Start a new process, and add the operator 'process documents from the web'.
 - Set the URL to <http://www.itb.ie>
 - Select add pages as attribute to store the web page content
 - Set max pages to 10
 - Leave remaining parameters at their default values
2. Double click on process documents from the web, and add "**extract content**" as an inner operator.
3. Run the process to view the output

☒ add pages as attribute

Lab work #4

Note: **Process Documents from Web** outputs an object of type **dataset**, which is not a document vector, and so does not work with text mining preparation operators like tokenise etc. To use these:

1. Return to the top level design view window.
2. Add a 'Process Documents from Data' operator which accepts a dataset.
3. Double click on this operator, and add a tokeniser as an inner operator.
 - You can also add filters and stemmers here.
4. Run the process again.

Lab work #5: Get Page(s)

- The Get Page and Get Pages operators download predefined web pages
 - Get Page: supply one URL to download a single page
 - Get pages: supply a list of URLs, each page is downloaded. The input will be a dataset that includes a list of URLs as one of its attributes. Make sure that the attribute that stores the URLs has a datatype of **file_path**.
 - See [UsingGetPages.rmp](#) using [urlList.xlsx](#)

Step 2: Extract the relevant information from that data

Pages downloaded by the web crawler can be mined as a text file, as you have been doing with unstructured news articles.

Alternatively, specific information can be extracted from those pages e.g. product price, company name etc

How to extract specific information is the focus of the following slides.

Information Extraction

- There are **two choices** when extracting information:
 1. Limit which part of the text to consider for text mining, e.g. Only the first paragraph; only product descriptions; only customer comments; only hotel reviews etc. **assuming the region of interested can be defined**
 2. Extract specific elements of text to fill a predefined template, **however defining what text to extract can be tricky.**
- The **extract content** operator in RapidMiner extracts pre-defined meta data (URL, TITLE, etc – see next slide).
- The **extract information** operator in RapidMiner can be used to define specific tokens or regions within a text. Areas of interest can be identified using **Regular Expression** or **XPath**.

Tutorial on regular expressions: <http://www.javaregex.com/tutorial.html>

Extracting information in Rapidminer

1. Extract Content:

- Nested within 'process documents', 'extract content' will extract the following from an webpage:
 - URL
 - Title
 - Language
 - Description
 - Keywords
 - Robots
 - All text blocks of a minimum length (default is 5) can then be tokenised using **Tokenize**.

Note: 'Process Documents' can be used on any attribute of type TEXT. 'Nominal to Text' will convert a nominal attribute to text.

Extracting Information in Rapidminer

1. **Extract Information**, also nested within **process documents**, can be used to identify sections to extract from a web page (or any text attribute). Limit what text to consider using String Matching, Regular Expression, Regular Region, XPath, or Index Queries:
 - **string matching queries**: Creates new attributes, giving each the value of a text block identified by the start and end string, e.g.

attribute name	query expression		
heading	<h1>	</h1>	Returns content within a H1 tag.
main	<div id="main">	</div>	Return content with main div tag
email	mailto:		Returns text following mailto: up to the next double quotes.

- **regular expression queries**: As above, but instead of giving a start and end strings, you define the text block using a regular expression, e.g. **mailto:.***
- **regular region queries**: As above, but a regular expression is used to define the start and the end of a block of text.
- **xpath queries**: The text block is identified using an XPath expression.

Recap on Xpath & Regular Expression

Sections from a webpage can be identified using:

Xpath – which identifies text based on its position within XML or HTML tags

Regular expression – which identifies text based on the surrounding text, or providing a matching template for the text itself

XPath

- XPath uses path expressions to select nodes, node-sets, attribute names or attribute values in an XML document. This allows the navigation in a XML document as well as the extraction of sections from the document.
- In XPath, there are seven kinds of nodes: **element**, **attribute**, **text**, **namespace**, **processing-instruction**, **comment**, and **document** nodes.
- XML documents are treated as trees of nodes. The topmost element of the tree is called the root element.
- <http://www.w3.org/TR/xpath/>

XPATH

<!-- XML Structure of a College Course -->.

<Degree>

<Computing>

<title lang="en">MSc. in Science in Computing</title>

<lecturer>Geraldine Gray</lecturer>

<lecturer>Markus Hofmann</lecturer>

<duration>2 years</duration>

<fee cur="Euro">4000</fee>

</Computing>

</Degree>

- Example of nodes from the sample XML file:
 - **Root Element:** <Degree>
 - **Element Node:** <lecturer>Geraldine Gray</ lecturer>
 - **Attribute Node:** lang="en"
 - **Text Node:** Geraldine Gray
 - **Comment Node:** XML Structure of a College Course

XPath

- Selecting Nodes in RapidMiner

Expression	Description
<i>h:nodename</i>	Selects all child nodes of the <i>nodename</i> node
/	Selects from the root node
//	Selects nodes in the current document no matter where they are within the document
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

XPath examples

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bookstore>

  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>

  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>

</bookstore>
```

- `/bookstore/book/title`: returns all book titles, provided the title tag is nested with `/bookstore/book`
- `//title`: returns all titles, regardless of where the tag is located (e.g. could return CD titles as well)
- `/bookstore/book[1]/title`: returns the title of the first book
- `/bookstore/book[price>35]/title`: returns the title of books whose price is > 35
- `/bookstore/book[@lang="eng"]/title`: returns the title of books that have the attribute of `lang="eng"`

XPath–Functions

- There are a large number of different functions available. For example:
- `name()`
 - Returns the name of the current node or the first node in the specified node set
- `contains()`
 - Searches for the occurrence of a string in the XML document

Using Google Chrome for XPath

- Chrome gives you the XPATH query for any element on the page:
 - Simply right click on the element you want to extract
 - Select “Inspect” element
 - Right click on the highlighted code that contains the information you want to extract
 - Select “Copy XPath”
 - This puts the XPATH query into your clipboard memory. Simply paste it into RapidMiner.
 - You need to add a ‘h:’ before every XHTML element (e.g. *div* becomes *h:div*)

- XPATH expressions tend to be specific to each web page, and may not generalise to all web pages in your collection.
- Regular expressions focus on the page content rather than page layout, and so are more likely to generalise to a number of pages.

Regular Expression: Java's REGEX

REGEX allows you to define a template that text can be matched against.

Character Classes

[abc]	a, b, or c (simple class)
[^abc]	Any character except a, b, or c (negation)
[a-zA-Z]	a through z, or A through Z, inclusive (range)
[a-dm-p]	a through d, or m through p
[a-z&&[^bc]]	a through z, except for b and c: [ad-z]
[a-z&&[^m-p]]	a through z, and not m through p: [a-lq-z]

Unless preceded by a backslash, the following characters are treated as **meta characters** :

([{ \ ^ - \$ |] }) ? * + .

Regular Expression: Java's REGEX

Predefined Character Classes

.	Any character
\d	A digit: [0-9]
\D	A non-digit: [^0-9]
\s	A whitespace character: [\t\n\x0B\f\r]
\S	A non-whitespace character: [^\s]
\w	A word character: [a-zA-Z_0-9]
\W	A non-word character: [^\w]

Quantifiers

?	once or not at all, e.g [a-z]?	
*	zero or more times e.g [a-z]*	[A-Z].*
+	one or more times, e.g [a-z]+	
{n}	exactly n times, e.g [a-z]{5}	\w{4}
{n,}	at least n times, e.g [a-z]{5,}	\w{4,}
{n,m}	at least n but not more than m times, e.g [a-z]{5,10}	

Regular Expression: Java's REGEX

- **Boundary Matchers**

- ^ The beginning of a line
- \$ The end of a line
- \b A word boundary
- \B A non-word boundary
- \A The beginning of the input
- \G The end of the previous match
- \Z The end of the input

Regular Expression: Java's REGEX

Matches against numeric values

- > greater than
- >= greater than or equal to
- < less than
- <= less than or equal to
- && and
- || or
- != not equal to

Other:

(?i) ignore case, e.g. (?i)dublin matches with dublin and Dublin

Regular expression examples

Matching with text:

Term called 'cost' : `cost`

Lines starting with the string 'cost' : `^cost.*`

Terms containing the string 'cost' or 'Cost':

`.*[cC]ost.*`

Lines that start with the letter g: `^g\w*`

Matching with numeric values:

Numeric values greater than 20: `> 20`

Numeric values less than 15 or greater than 25: `< 15 | > 25`

Numeric values between 15 and 25: `> 15 && < 25`

Regular expression exercise

If you are defining the token to be extracted, put round brackets around it, e.g. (cost)

If you are defining the token that proceeds the term to be extracted, put (.*) after it, e.g. cost (.*)

Define the following:

The token that comes after the term 'price':

The token that comes after the terms 'price' or 'Price':

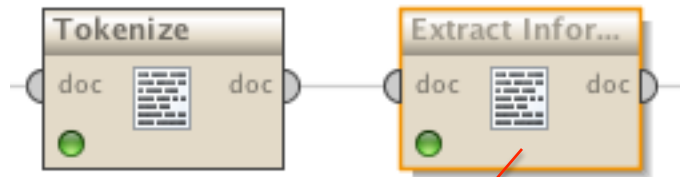
The token that comes after the terms 'price' , 'Price' , 'Cost' or 'cost':

The subject line of an email:

Examples:


Extracting information from an e-mail:

Nested in Process Documents:



query type: Regular Expression ▼ ⓘ

attribute type: Nominal ▼ ⓘ

regular expression que...:  Edit List (3)... ⓘ

attribute name	query expression
subject	Subject:(.*)
from2	From:[\s]*([^\s]*)
from	From:(.*)

gives . . .

See the RapidMiner process 'InfoExtract-regexexpression'

From: rych@festival.ed.ac.uk (R Hawkes)
Subject: Re: 3DS: Where did all the texture rules go?

eric.vitiello@tfd.coplex.com (Eric Vitiello) writes:

>TO: rych@festival.ed.ac.uk (R Hawkes)

>RH>I've noticed that if you only save a model (with all y
>RH>positioned carefully) to a .3DS file that when you rel
>RH>3DS they are given a d

**Original texts
(graphics folder
on moodle):**

From: peng@cipserv1.physik.uni-ulm.de (WEIGUO PENG)
Subject: SW convert plot to ASCII file

I am looking for software that reads a plot in PCX or other format and
converts it into x,y coordinate.

From: SITUNAYA@IBM3090.BHAM.AC.UK
Subject: Best FTP Viewer please.

=====
Could someone please tell me the Best FTP'able viewer available for MSDOS
I am running a 486 33mhz with SVGA monitor.
I need to look at gifs mainly and it would be advantageous if it ran
under windows.....thanks

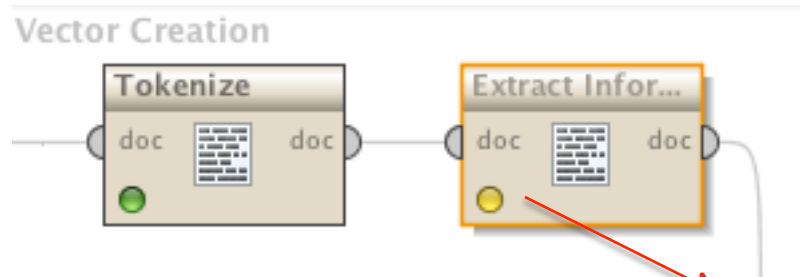
Output: three additional attributes/features

subject	from2	from
Re: 3DS: Where did all the texture rules go	rych@festival.ed.ac.uk	rych@festival.ed.ac.uk (R Hawkes)
SW convert plot to ASCII fil	peng@cipserv1.physik.uni-ulm.de	peng@cipserv1.physik.uni-ulm.de (WEIGUO PENG)
Best FTP Viewer please	SITUNAYA@IBM3090.BHAM.AC.UK	SITUNAYA@IBM3090.BHAM.AC.U
RGB to HVS, and bac	remcoha@htsa.aha.nl	remcoha@htsa.aha.nl (Remco Hartog)

Examples:

What would the following extract from the healthcare, crime and kenya files?

- Query string:
 - `((Dr | President | Ms | General)\s[A-Z]\w*\s[A-Z]\w*)`



Exercise: Change the expression above to also include names where only the surname is given, e.g. Ms Harney

attribute name	query expression
Name_Title	<code>((Dr President Ms General)\s[A-Z]\w*\s[A-Z]\w*)</code>

See RapidMinerProcess: InfoExtract-name

Results . . .

label	metadata_...	metadata_...	metadata_...	Name
crime	C1.txt	/Users/gera	Feb 21, 200	?
crime	C2.txt	/Users/gera	Feb 21, 200	?
crime	C3.txt	/Users/gera	Feb 21, 200	Dr Diarmuid Martin
crime	C4.txt	/Users/gera	Feb 21, 200	?
crime	C5.txt	/Users/gera	Feb 21, 200	?
kenya	K1.txt	/Users/gera	Feb 21, 200	?
kenya	K2.txt	/Users/gera	Feb 21, 200	General Kofi Annan
kenya	K3.txt	/Users/gera	Feb 21, 200	?
kenya	K4.txt	/Users/gera	Feb 21, 200	?
kenya	K5.txt	/Users/gera	Feb 21, 200	President Mwai Kibaki
healthcare	H1.txt	/Users/gera	Feb 21, 200	?

More complex methods to extract information:

There are a number of more complex methods to extract content from a text documents, we will mention two of these briefly:

- Hidden Markov models

- Named Entity Recognition

Extract specific elements using a Hidden Markov model

The idea here is similar to using HMM for identifying part of speech.

The objective is to find patterns to sentences which are likely to have the concepts we're looking for.

For example. Suppose you want to extract information about people and the company / organisation they work for . . .

- The training data could be labeled as follows:
 - John works in Dublin for E-bay
 - [person] [-] [-] [Location] [-] [Company]
 - Mary Daly works at Google headquarters in CA
 - [person-firstname] [person-lastname] [-] [-] [Company] [-] [-] [Location]
- Other information such as parts of speech, upper/lower case, etc. could also be added to the model for greater accuracy.
- The HMM could then be used to identify the concept associated with a new word, based on the words before and after it.

Extract specific elements using **Named Entity Recognition (NER)**

- Named Entity Recognition is an alternative approach to identifying concepts.
- Its a way of presenting information to a learning algorithm to allow it learn what a word is, based on attributes such as the word that comes after it, the POS that comes after it, etc.
- A dataset of training data is created as follows:

Term	Label	Next word	Next word POS	Previous word ...
John	Person	Works	Verb	.
Works	-	at	prep	John
...				

- The table can be extended to store more information about each attribute.

Summary

