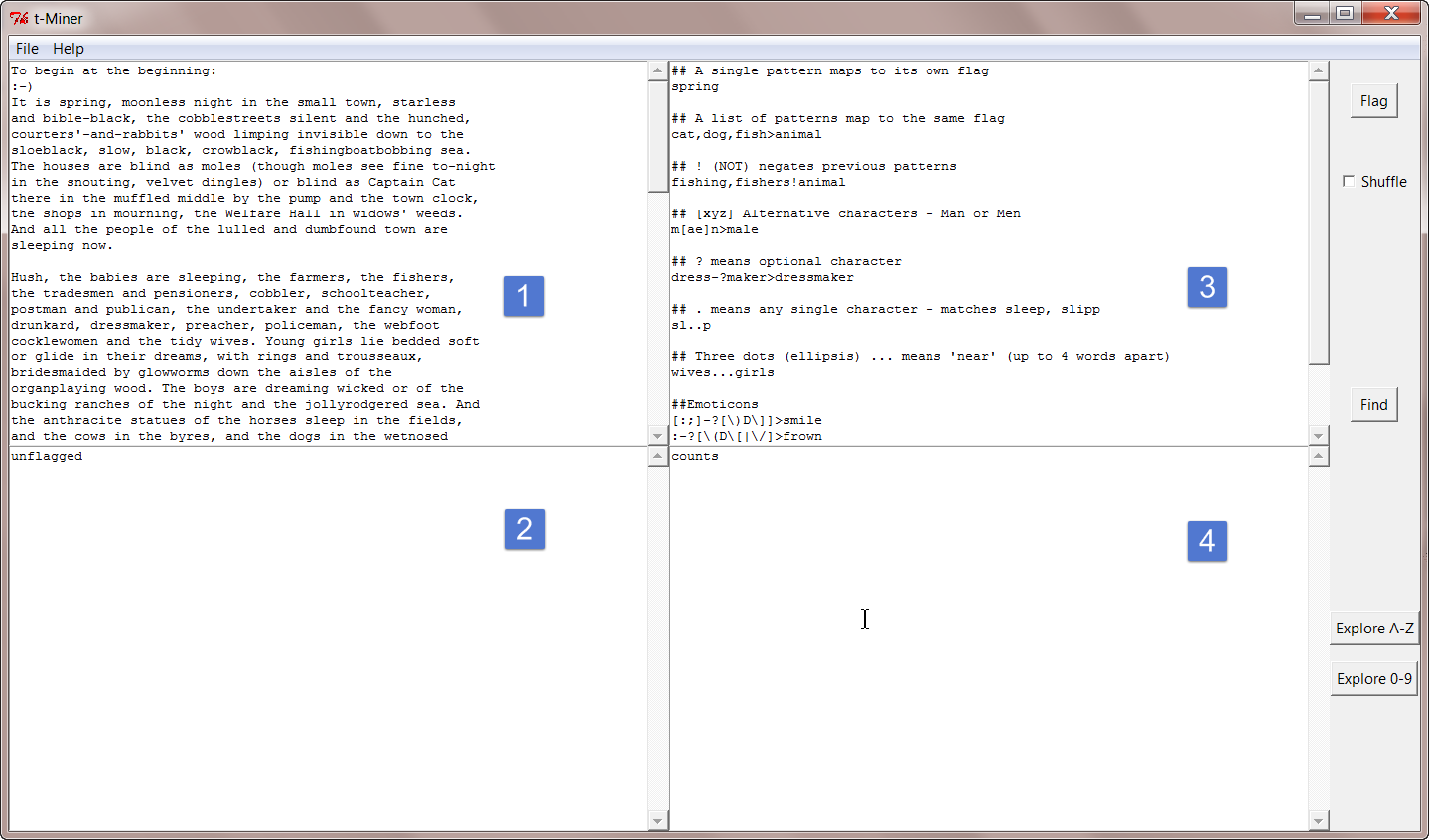
TMine – Text Mining tool

# Overview

TMine is a tool used to quickly generate patterns for flagging content of text files. These patterns can be kept, maintained, and reused. Code for flagging the patterns in Portrait Miner is also generated.

# Running

1. Double click on TMine shortcut



Window 1 is the text to mine, or ‘Verbatims’.

Window 2 shows unflagged lines of text after “Flagging”

Window 3 holds your search patterns

Window 4 shows counts of occurances of patterns in the Verbatims

1. Load text:
   * Copy and paste into window 1, or
   * Click *File/Load Verbatims* and select a text file
   * Click *File/Load Project* and select any text file from your previous project (it just looks at the filestem and loads the matching patterns and text file)
2. Explore text:
   * Click on *Explore A-Z* or *Explore 0-9*
3. Create patterns:
   * Type or paste patterns into window 3
   * Select a word with the mouse, and right-click to push it to window 3
   * Click *File/Load Patterns* and select a text file
4. Flag your text:
   * Click *Flag* button
   * Observe the flagged lines in window 1
   * Observe the unflagged lines in window 2
   * Tick *Shuffle* box for window 2 to be randomized, so you get a better sample of unflagged lines showing
5. Save your project:
   * Click *File/Save Project,* select a directory and enter a project name. It will save:
     + name.txt- the original text
     + name.patterns.txt *–* your pattern set
     + name.words.txt – a list of words in frequency order
     + name.counts.txt – a list of patterns and frequency of occurance
     + name.unflagged.txt – lines of text yet to be flagged
     + name.fdl – code for Portrait Miner

Note that only the first two files (name.txt and name.patterns.txt) are loaded when you load a project – the rest can be regenerated at will.

# Tag Patterns

All patterns are matched line by line, using regular expressions:

https://en.wikipedia.org/wiki/Regular\_expression#Basic\_concepts

A line is labelled on the left side of the ## if it matches the pattern e.g. for the pattern ‘milk>dairy’

* dairy ## This is my text about milkshakes

All matches are case insensitive.

Matches and negations are carried out in the order of the pattern file.

Certain characters have a special meaning: e.g. \* $ [ ] ( ) { } | . + To use them literally, you need to put a backslash before them (e.g. ‘\?’ to look for a question mark)

This tool has a couple of extra expressions ‘…’ and ‘&’ – see below.

A single pattern maps to its own label

* spring

A list of patterns can be mapped to the same label using a ‘>’ (greater-than symbol, here meaning arrow)

* cat,dog,fish>animal

More patterns can be added to a label

* cat,dog,fish>animal
* cow,horse,goat>animal

A pattern can undo a mapping, using a ‘!’ (NOT)

* fishing,fisherman!animal

Alternative characters can be provided: [xyz] matches ‘x’ or ‘y’ or ‘z’

* m[ae]n>male

‘?’ means optional character

* dress-?maker>handicraft (matched ‘dressmaker’ or ‘dress-maker’

Three dots (ellipsis) ... means 'near' (up to 4 words apart)

* wives...girls>wivesandgirls

‘&’ means within the same sentence

* holiday&France>FrenchHoliday

Some emoticons

* [:;]-?[\)D\]]>smile
* :-?[\(\[|\/]>frown

Search for a label – all the above search the right-hand side for patterns. You can look for labels already created by using a leading dollar. This can be used to build *A not B* patterns.

* $A>AnotB
* $B!AnotB

# Regular Expression (Regex) cheat sheet

Googling regex will give many results, but for quick reference:

**.** Any single character (but see ‘… = near’ below)

*m.n* matches *man, men, min, m3n,* and many more

**(…)** All-of group - used for option groups

**[…]** Any-of group

*m[ae]n* matches *man or man* only

t[ -]mobile matches *t mobile* or *t-mobile*

**?** Optional:

*hot-?dog* matches *hot-dog* or *hotdog* only

*zufried(en)?* matches *zufried* or *zufrieden* only

*m.?n* matches *man, men, min, m3n,* and many more, and also *mn*

t[ -]?mobile matches *t mobile, t-mobile* or *tmobile*

**(a|b)** Or, within brackets:

(hot|cold)dog matches hotdog and colddog

**\** Escape character – means ‘literally the next character’. Used for including symbols used above, without them being interpreted as control codes

t\.?mobile matches *t.mobile* or *tmobile*

t[ -\.]?mobile matches *t mobile, t-mobile, t.mobile* or *tmobile*

*…* Ellipsis – means “near”, or 0-3 words between the terms within the same sentence *(Note, custom for this this application only)*

speed…internet matches “slow speed of internet”, “speed of my internet” etc.

**Complex example** using several of the above:

*[:;][0o-]?[\)\]D]* matches :-), :), :-], ;-D and several others.

First the eyes: [:;] means either : or ;

Next the nose: *[Oo-]?* means one of O, o, or -, or no nose, optionally

Finally the mouth: *[\)\]D]* means one of

* \) a literal right bracket),
* \[ (a literal right brace), or
* D a toothy D.