Daniel Jurafsky & James H. Martin. Speech and Language Processing.
Глава 2, с. 2-11. https://web.stanford.edu/~jurafsky/slp3/

# Регулярные выражения

# Обработка текста

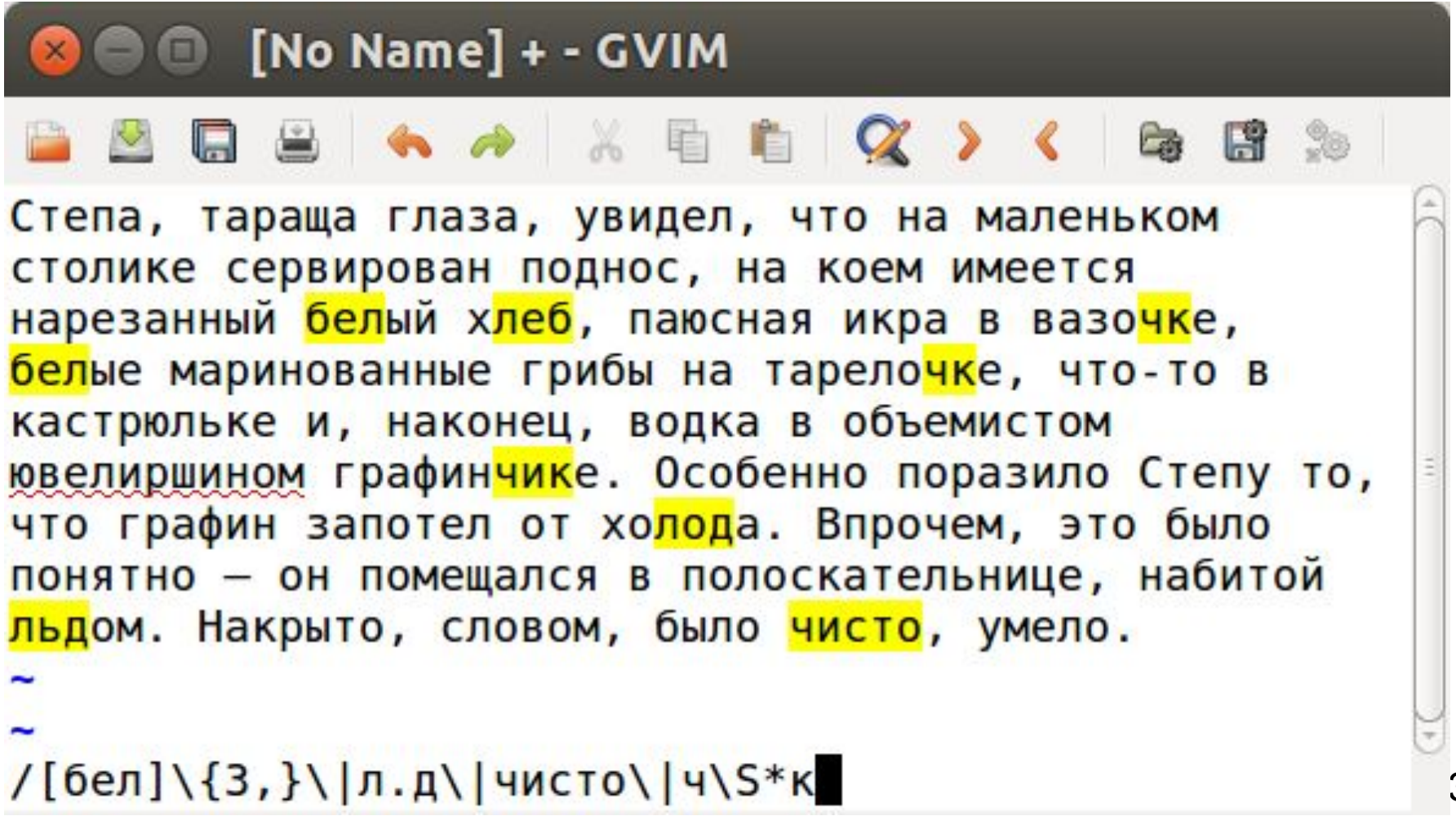
# Basic Text Processing

## Regular Expressions

Крижановский А.А.
Локализация, добавление примеров и вопросов.

# Regular expressions

- A formal language for specifying text strings
- How can we search for any of these?
  - самовар
  - вар
  - варежка
  - Варвара


[No Name] + - GVIM
Степа, тараща глаза, увидел, что на маленьком столике сервирован поднос, на коем имеется нарезанный белый хлеб, паюсная икра в вазочке, белые маринованные грибы на тарелочке, что-то в кастрюльке и, наконец, водка в объемистом ювелиршином графинчике. Особенно поразило Степу то, что графин запотел от холода. Впрочем, это было понятно — он помещался в полоскательнице, набитой льдом. Накрыто, словом, было чисто, умело.
~
~
/[бел]\{3,}\|л.д\|чисто\|ч\S*к

# Формальные языки (ФЯ)

- RE описывают регулярные языки в теории формальных языков.
- ФЯ состоит из слов, порождённых RE.
- RE включают **константы** и **операторы**, которые задают:
  - множество строк,
  - и операций над ними

# Константы, являющиеся RE

- над конечным алфавитом $\Sigma$
- (пустое множество) $\varnothing$
- (пустая строка) $\varepsilon$ — множество, содержащее только пустую строку
- (символьный литерал) $a \in \Sigma$ — множество, содержащее только символ a.

# Операции, генерирующие RE 1

- над конечным алфавитом Σ
- R, S — это RE
- (сцепление, конкатенация) RS
  - Пример: {"ab", "c"}{"d", "ef"} = {"abd", "abef", "cd", "cef"}.
- (дизъюнкция, чередование) R|S
  - {"ab", "c"}|{"ab", "d", "ef"} = {"ab", "c", "d", "ef"}.

# Операции, генерирующие RE 2

- (замыкание Клини, звезда Клини) R*

  минимальное надмножество множества R, которое содержит ε и замкнуто относительно конкатенации

  - {"ab", "c"}* = {ε, "ab", "c", "abab", "abc", "cab", "cc", "abababab", "abcab", … }.
  - {"0","1"}* =

# Области применения RE

- поиск по шаблону
- замена (pattern substitution)
- chatbot (тест Тьюринга)
- токенизация

# Regular Expressions: Disjunctions

- Letters inside square brackets []

| Pattern | Matches |
|---|---|
| `[wW]oodchuck` | Woodchuck, woodchuck |
| `[1234567890]` | Any digit |

- Ranges `[A-Z]`

| Pattern | Matches | |
|---|---|---|
| `[A-Z]` | An upper case letter | `Drenched Blossoms` |
| `[a-z]` | A lower case letter | `my beans were impatient` |
| `[0-9]` | A single digit | `Chapter 1: Down the Rabbit Hole` |

# Regular Expressions: Negation in Disjunction

- Negations [^Ss]
  - Carat means negation only when first in []

| Pattern | Matches | |
|---|---|---|
| [^A-Z] | Not an upper case letter | Oyfn pripetchik |
| [^Ss] | Neither 'S' nor 's' | I have no exquisite reason" |
| [^e^] | Neither e nor ^ | Look here |
| a^b | The pattern a carat b | Look up a^b now |

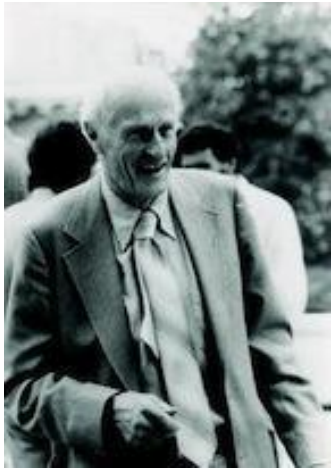
# Regular Expressions: More Disjunction

- Woodchucks is another name for groundhog!
- The pipe | for disjunction

| Pattern | Matches |
|---|---|
| `groundhog\|woodchuck` | |
| `yours\|mine` | yours<br>mine |
| `a\|b\|c` | = [abc] |
| `[gG]roundhog\|[Ww]oodchuck` | |

# Regular Expressions: ? * + .

| Pattern | Matches | |
|---|---|---|
| colou?r | Optional previous char | color colour |
| oo*h! | 0 or more of previous char | oh! ooh! oooh! ooooh! |
| o+h! | 1 or more of previous char | oh! ooh! oooh! ooooh! |
| baa+ | | baa baaa baaaa baaaaa |
| beg.n | | begin begun begun beg3n |



Stephen C Kleene

Kleene *, Kleene +

# Regular Expressions: Anchors ^ $

| Pattern | Matches |
|---|---|
| ^[A-Z] | Palo Alto |
| ^[^A-Za-z] | 1 "Hello" |
| \.$ | The end. |
| .$ | The end? The end! |

# Токенизация (две стратегии разбиения на слова)

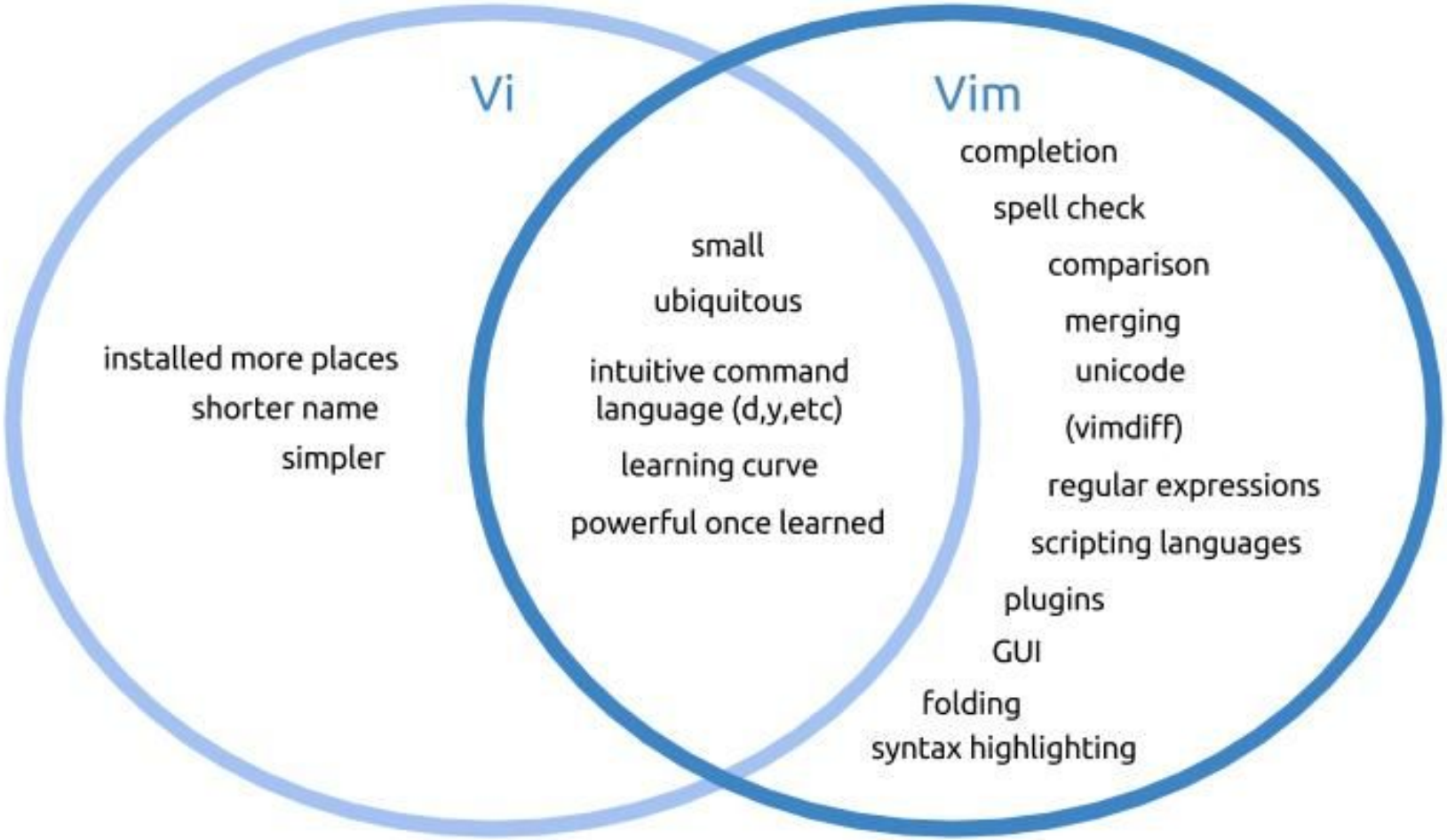
- Пробелы?
- Слова?
  - Какие символы?

# Разбиваем на слова (VIM): (1) ищем границы слов

- Но он не умер. Открыв слегка глаза, он увидел себя сидящим на чем-то каменном. Вокруг него что-то шумело. Когда он открыл, как следует, глаза, он увидел, что шумит море, и что даже больше того, — волна покачивается у самых его ног, и что, короче говоря, он

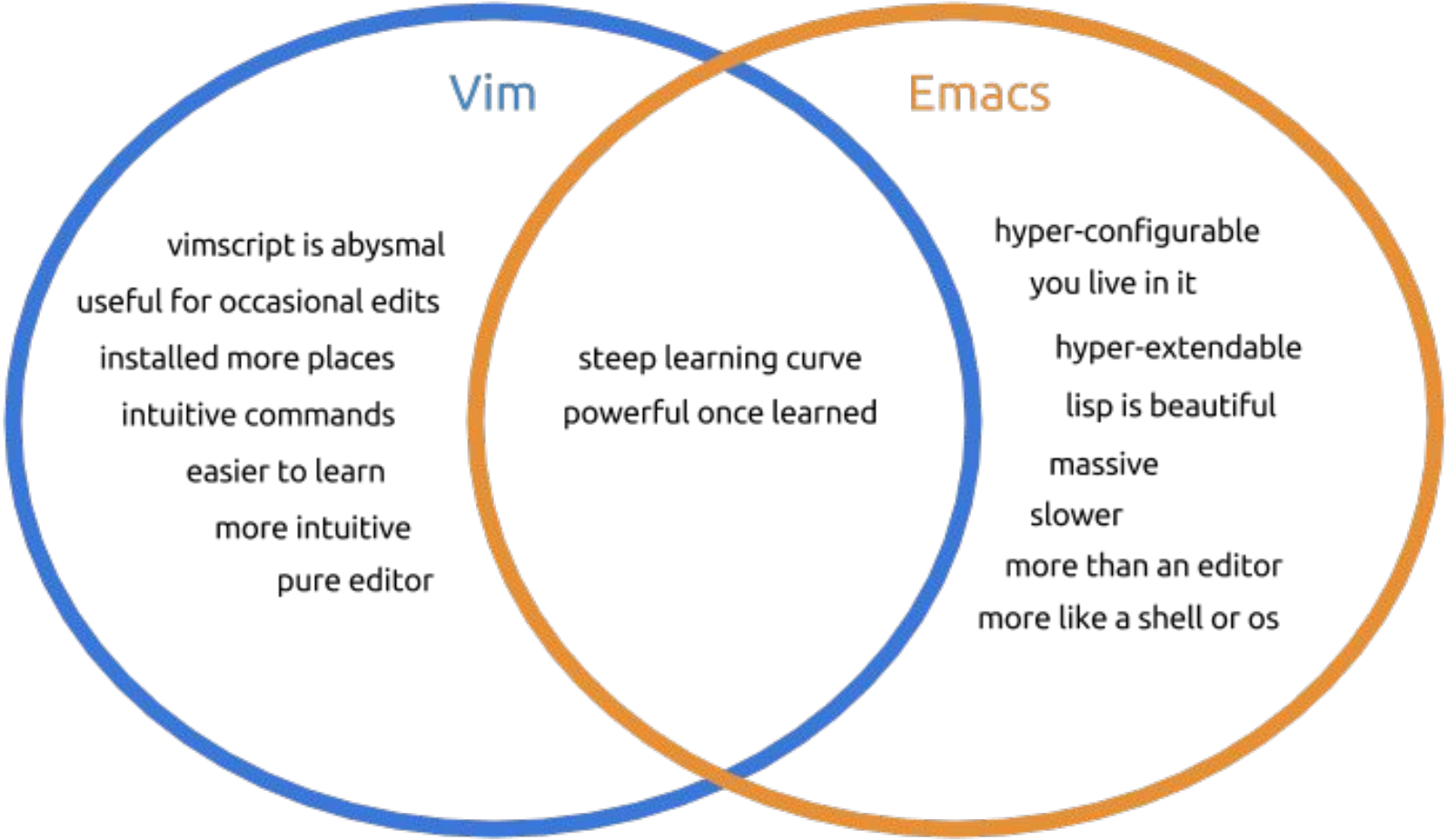
# VIM

Vi, Vim and Emacs






Miessler D. The Differences Between Vi, Vim, and Emacs.https://danielmiessler.com/blog/differences-vi-vim-emacs/





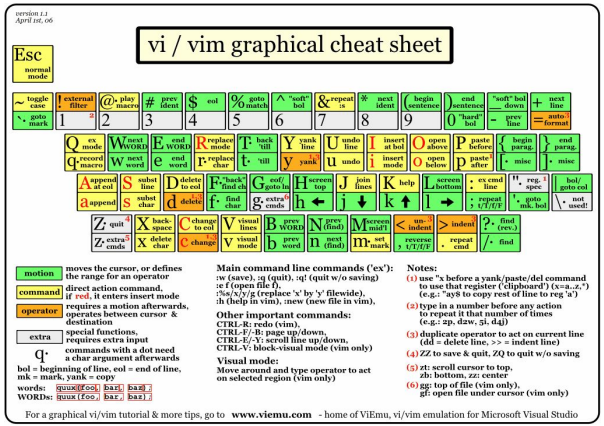

Miessler D. The Differences Between Vi, Vim, and Emacs.https://danielmiessler.com/blog/differences-vi-vim-emacs/

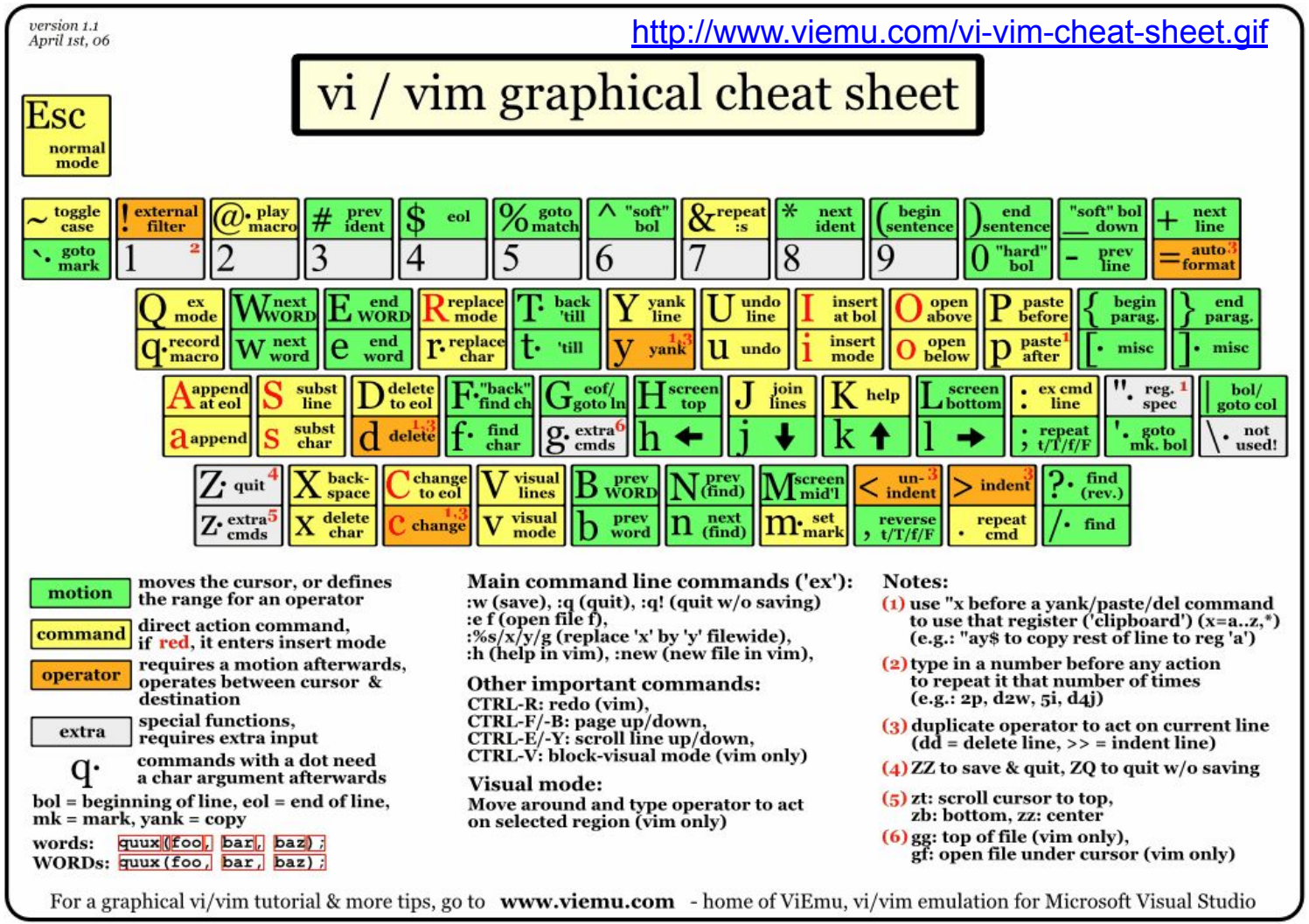
# Vim, Emacs

- Bill Joy
- 1976
- Unix modularity
- work like a language =>? intuitive
- installed nearly

- Richard Stallman
- 1976
- power and configurability
- live in the system as much as possible

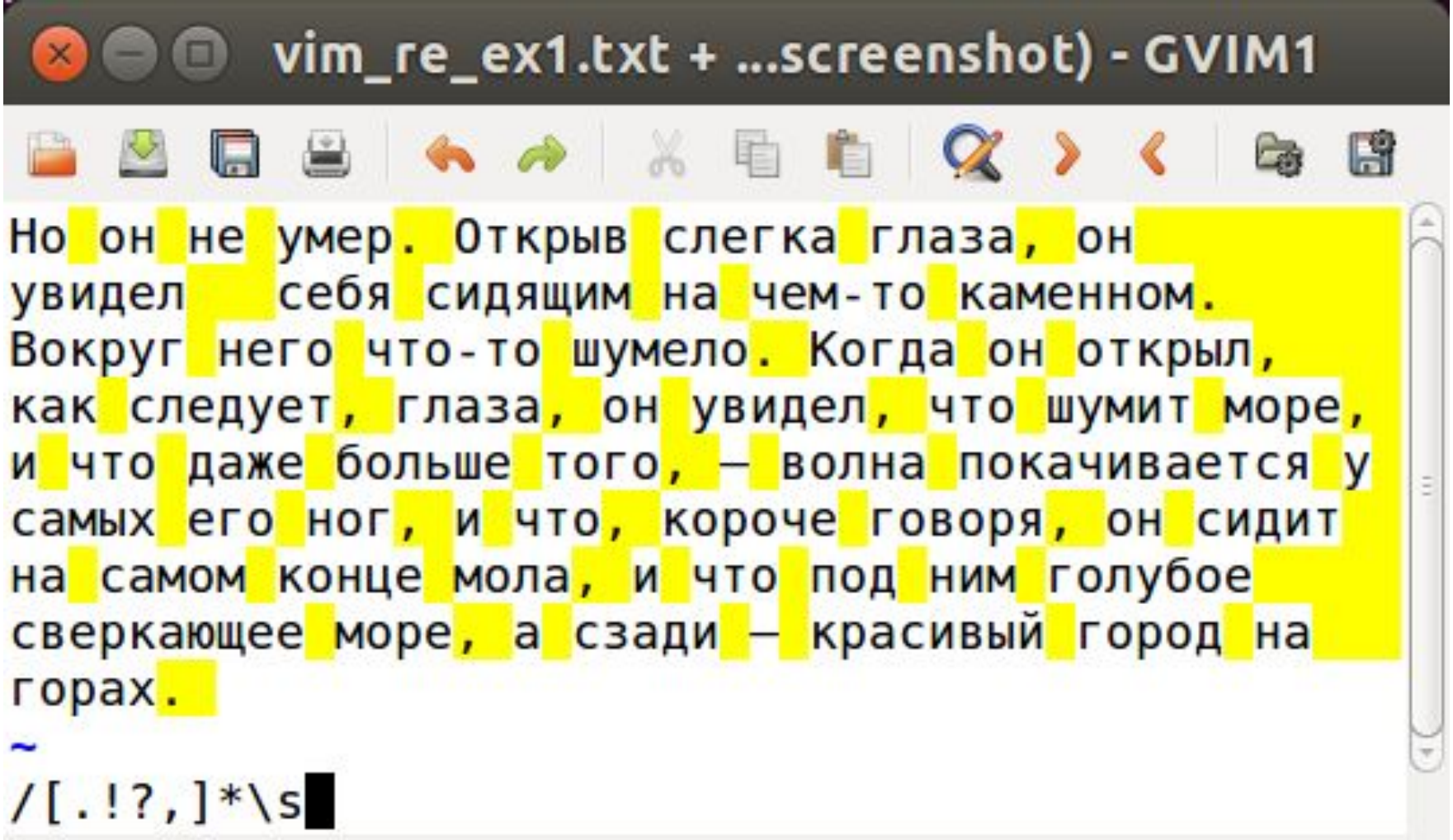
Just memorize this vi / vim cheat sheet and you're ready for lightning-quick editing.




version 1.1
April 1st, 06
http://www.viemu.com/vi-vim-cheat-sheet.gif
vi / vim graphical cheat sheet
Esc
normal mode
~ toggle case
` goto mark
! external filter
1
@ play macro
2
# prev ident
3
$ eol
4
% goto match
5
^ "soft" bol
6
& repeat :s
7
* next ident
8
( begin sentence
9
) end sentence
0 "hard" bol
__ "soft" bol down
- prev line
+ next line
= auto format
Q ex mode
q record macro
W next WORD
w next word
E end WORD
e end word
R replace mode
r replace char
T back 'till
t 'till
Y yank line
y yank
U undo line
u undo
I insert at bol
i insert mode
O open above
o open below
P paste before
p paste after
{ begin parag.
[ misc
} end parag.
] misc
A append at eol
a append
S subst line
s subst char
D delete to eol
d delete
F "back" find ch
f find char
G eof/ goto ln
g extra cmds
H screen top
h
J join lines
j
K help
k
L screen bottom
l
: ex cmd line
; repeat t/T/f/F
" reg. spec
' goto mk. bol
| bol/ goto col
\ not used!
Z quit
z extra cmds
X back-space
x delete char
C change to eol
c change
V visual lines
v visual mode
B prev WORD
b prev word
N prev (find)
n next (find)
M screen mid'l
m set mark
< un-indent
, reverse t/T/f/F
> indent
. repeat cmd
? find (rev.)
/ find
motion
moves the cursor, or defines the range for an operator
command
direct action command, if red, it enters insert mode
operator
requires a motion afterwards, operates between cursor & destination
extra
special functions, requires extra input
q·
commands with a dot need a char argument afterwards
bol = beginning of line, eol = end of line,
mk = mark, yank = copy
words: quux(foo, bar, baz);
WORDs: quux(foo, bar, baz);
Main command line commands ('ex'):
:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:%s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :new (new file in vim),
Other important commands:
CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)
Visual mode:
Move around and type operator to act on selected region (vim only)
Notes:
(1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay$ to copy rest of line to reg 'a')
(2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)
(3) duplicate operator to act on current line (dd = delete line, >> = indent line)
(4) ZZ to save & quit, ZQ to quit w/o saving
(5) zt: scroll cursor to top, zb: bottom, zz: center
(6) gg: top of file (vim only), gf: open file under cursor (vim only)
For a graphical vi/vim tutorial & more tips, go to www.viemu.com - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

# Разбиваем на слова (VIM): (1) ищем границы слов

Но он не умер. Открыв слегка глаза, он увидел себя сидящим на чем-то каменном. Вокруг него что-то шумело. Когда он открыл, как следует, глаза, он увидел, что шумит море, и что даже больше того, — волна покачивается у самых его ног, и что, короче говоря, он сидит на самом конце мола, и что под ним голубое сверкающее море, а сзади — красивый город на горах.


vim_re_ex1.txt + ...screenshot) - GVIM1
Но он не умер. Открыв слегка глаза, он
увидел себя сидящим на чем-то каменном.
Вокруг него что-то шумело. Когда он открыл,
как следует, глаза, он увидел, что шумит море,
и что даже больше того, — волна покачивается у
самых его ног, и что, короче говоря, он сидит
на самом конце мола, и что под ним голубое
сверкающее море, а сзади — красивый город на
горах.
~
/[.!?,]*\s

# Токенизация

- Пробелы,
  но не всегда:
  - Saint Petersburg
  - всё равно
    (частица)
  - ?
- Слова?
  - Поздравляю с 8
    марта :) #весна

# Токенизация

- Пробелы,
  но не всегда:
  - Saint Petersburg
  - всё равно (частица)
  - ?
- Слова?
  - Поздравляю с 8 марта :) #весна
  - + смайлики,
  - + хештеги

!Китайский язык: 1755年1月25日俄罗斯女沙皇伊丽莎白·彼得罗芙娜下令建立莫斯科大学，同年4月26日该大学开始授课。至今为止在俄罗斯1月25日是大学生节。

# Разбиваем на слова:
# (2) ищем сами слова

- The detailed elaborations on the development of even a short program form a long story, indicating that careful programming is not a trivial subject. If this paper has helped to dispel the widespread belief that programming is easy as long as the programming language is powerful enough and the available computer is fast enough, then it has achieved one of its purposes. #NiklausWirth #programming

Niklaus Wirth. Program Development by Stepwise Refinement. 1995.
http://sunnyday.mit.edu/16.355/wirth-refinement.html

# Разбиваем на слова (2): ищем сами слова

1) ищем слова *program* и *programming,*
   - поиск по двум словам, какое раньше?
   - поиск слов с корнем *program*,
   - RE: **Quantifiers, Greedy and Non-Greedy**
     *, *?, +?  VIM: *, \+, \=, \{-}
2) замена (substitute) на WORD,

# Жадные (ленивые) квантификаторы:

`?` `*` `+` `.`

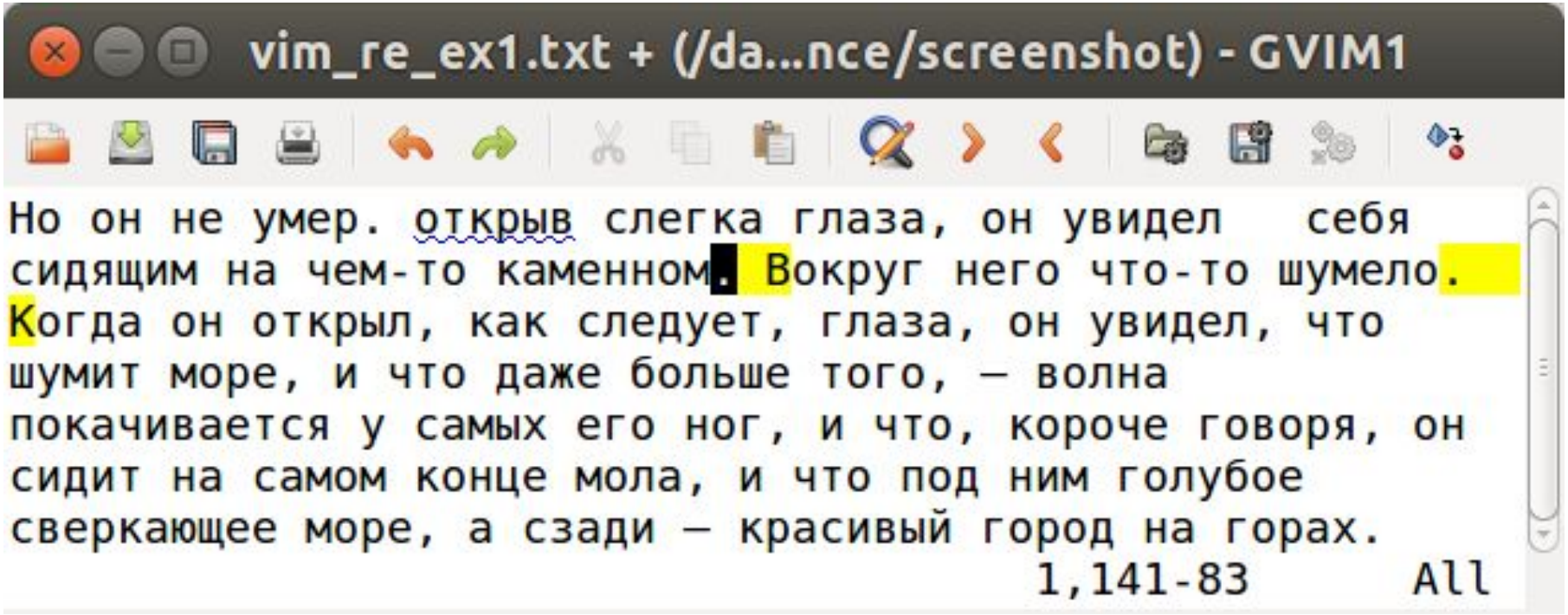
| Vim greedy | Vim non-greedy | Matches | |
|---|---|---|---|
| `*` | `\{-}` | 0 or more of previous char | color colour |
| `\+` | `+?` | 1 or more of previous char | oh! ooh! oooh! ooooh! |
| `{m,n}` | `{-n,m}` | from m to n of previous char | oh! ooh! oooh! ooooh! |
| | | | |
| | | | |

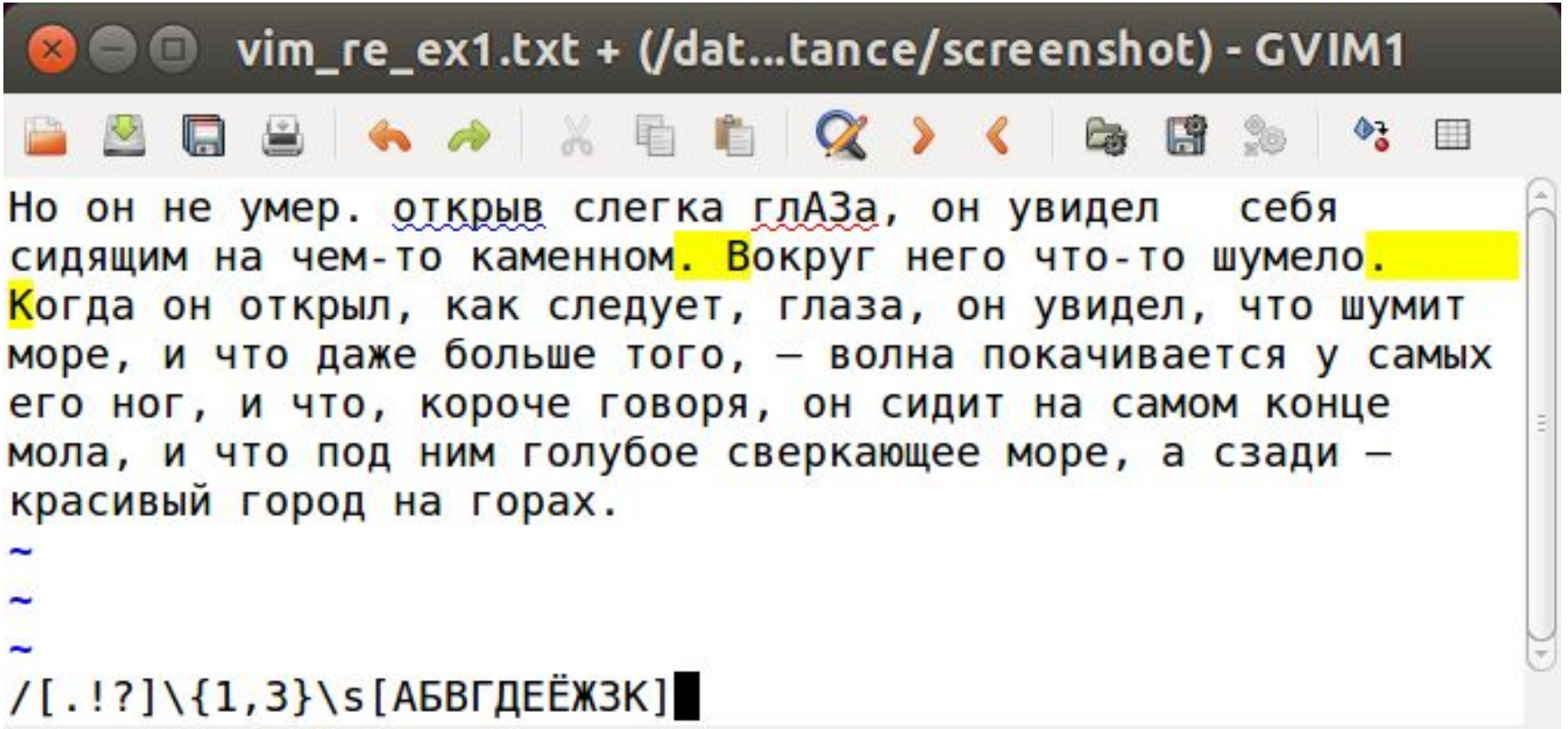
# Разбиваем на слова: ищем сами слова

- The detailed elaborations on the development of even a short program form a long story, indicating that careful programming is not a trivial subject. If this paper has helped to dispel the widespread belief that programming is easy as long as the programming language is powerful enough and the available computer is fast

# Разбить текст на предложения

- Но он не умер. открыв слегка глаза, он увидел себя сидящим на чем-то каменном. Вокруг него что-то шумело. Когда он открыл, как следует, глаза, он увидел, что шумит море, и что даже больше того, — волна покачивается у самых его ног, и что, короче говоря, он сидит на самом конце мола, и что под


vim_re_ex1.txt + (/da...nce/screenshot) - GVIM1
Но он не умер. открыв слегка глаза, он увидел себя
сидящим на чем-то каменном. Вокруг него что-то шумело.
Когда он открыл, как следует, глаза, он увидел, что
шумит море, и что даже больше того, – волна
покачивается у самых его ног, и что, короче говоря, он
сидит на самом конце мола, и что под ним голубое
сверкающее море, а сзади – красивый город на горах.
1,141-83 All


vim_re_ex1.txt + (/dat...tance/screenshot) - GVIM1
Но он не умер. открыв слегка глАЗа, он увидел себя
сидящим на чем-то каменном. Вокруг него что-то шумело.
Когда он открыл, как следует, глаза, он увидел, что шумит
море, и что даже больше того, – волна покачивается у самых
его ног, и что, короче говоря, он сидит на самом конце
мола, и что под ним голубое сверкающее море, а сзади –
красивый город на горах.
~
~
~
/[.!?]\{1,3}\s[АБВГДЕЁЖЗК]

# Example

- Find me all instances of the word “the” in a text.

```
the
```

Misses capitalized examples

```
[tT]he
```

Incorrectly returns `other` or `theology`

```
[^a-zA-Z][tT]he[^a-zA-Z]
```

# Errors

- The process we just went through was based on fixing two kinds of errors
  - Matching strings that we should not have matched (there, then, other)
    - False positives (Type I)
  - Not matching things that we should have matched (The)
    - False negatives (Type II)

# Errors cont.

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves two antagonistic efforts:
  - Increasing accuracy or precision (minimizing false positives)
  - Increasing coverage or recall (minimizing false negatives).

# Summary

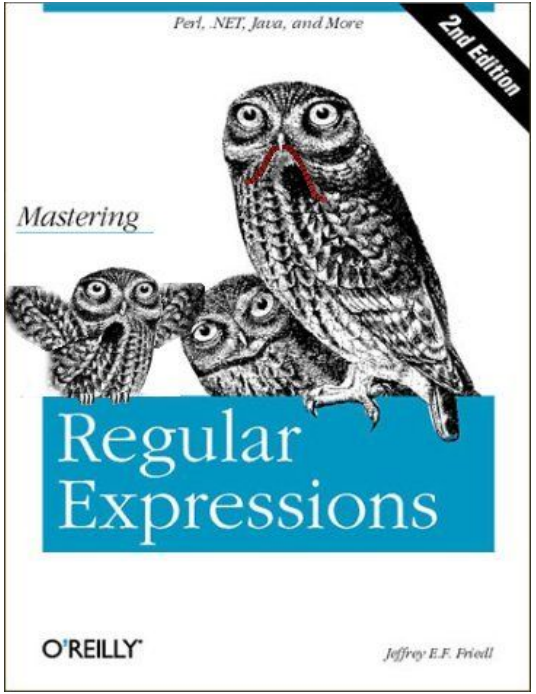
- Regular expressions play a surprisingly large role
  - Sophisticated sequences of regular expressions are often the first model for any text processing text
- For many hard tasks, we use machine learning classifiers
  - But regular expressions are used as features in the classifiers
  - Can be very useful in capturing generalizations



37

# Литература

- Фридл, Дж. Регулярные выражения. — СПб.: «Питер», 2001. — 352 с. (Mastering Regular Expressions)
- Miessler D. The Differences Between Vi, Vim, and Emacs. https://danielmiessler.com/blog/



# Ссылки

- http://vimregex.com
-

# Basic Text Processing

Regular Expressions