**Задание 4. Text mining средствами R.**

Выполнила студентка 2 курса

группы 09-715(1)

Анисимова Татьяна.

**Текст задания:**

В этом задании нужно применить методы Text mining к данным, загруженным из Интернет-источников. Для загрузки данных желательно использовать средства R. Можно использовать API социальных сетей, например, Twitter. Желательно загружать тексты на русском языке. Возможные задачи для анализа:

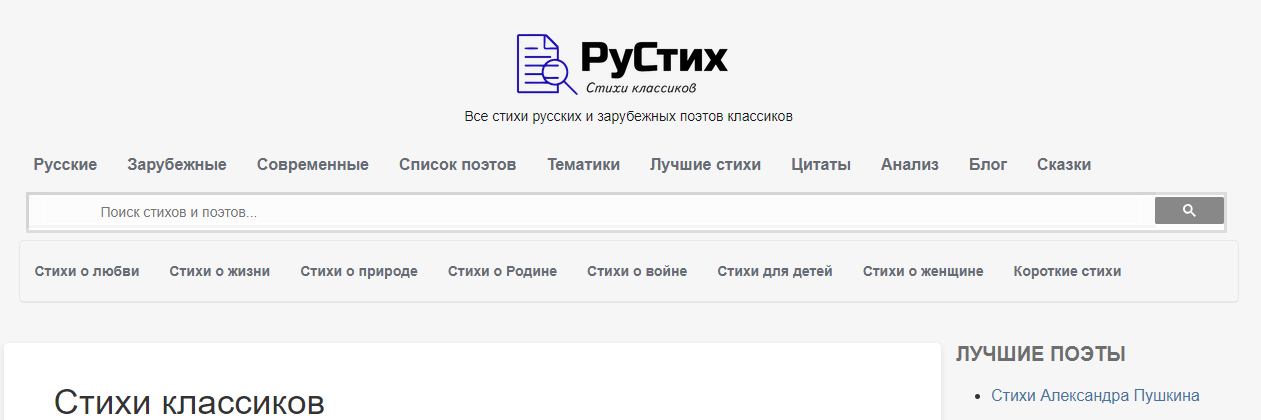
* задача классификации текстов,
* задача кластеризации текстов,
* задача анализа тональности текстов.

**Описание выполненной работы:**

Рассмотрим задачу классификации стихотворений классиков по следующим тематикам:

* стихи о любви;
* стихи о жизни;
* стихи о природе;
* стихи о Родине;
* стихи о войне;
* стихи для детей.

Стихи русских и зарубежных классиков были взяты с сайта <https://rustih.ru>.



Подключим необходимые библиотеки:

> library(RCurl) # основной пакет для веб-скрепинга

> library(XML) # пакет для работы с XML, HTML, Xpath

> library(stringr) # пакет для работы с регулярными выражениями

**# стихи русских и зарубежных классиков**

> main = "https://rustih.ru/"

**# создаем объект для работы по протоколу SSL**

> signatures = system.file("CurlSSL", cainfo = "cacert.pem", package = "RCurl")

> url = getURL(main, cainfo = signatures, encoding="UTF-8")

**# создаем html-объект**

> webpage <- htmlParse(url, encoding="UTF-8")

Рассмотрим кусок HTML – кода:

|  |
| --- |
| <div class="**collapse navbar-collapse main-navigation shadow-menu**" id="bs-example-navbar-collapse-1">  <ul class="nav navbar-nav uppercase">  <li class="menu-item dropdown"><a title="Стихотворения о любви" href="<https://rustih.ru/stihi-o-lubvi/>" class="**dropdown-toggle**" aria-haspopup="true" data-hover="dropdown">Стихи о любви</a>  <ul role="menu" class=" dropdown-menu">  ……  </ul>  </li>  <li class="menu-item dropdown"><a title="Стихотворения о жизни" href="<https://rustih.ru/stihi-o-zhizni/>" class="**dropdown-toggle**" aria-haspopup="true" data-hover="dropdown">Стихи о жизни</a>  <ul role="menu" class=" dropdown-menu">  ……  </ul>  </li  ……  </ul>  </div> |

Для получения ссылок на тематики стихотворений нам нужен тег a с атрибутом class="dropdown-toggle":

**# вытащим информацию о тематике стихотворений**

> category <- xpathSApply(webpage, "//div[@class='collapse navbar-collapse main-navigation shadow-menu']//a[@class='dropdown-toggle']", xmlGetAttr, "href");

Уберем из списка категории короткие стихи и стихи о женщине. Получим 6 тематик:

> category = category[-8];

> category = category[-7];

> category

[1] "https://rustih.ru/stihi-o-lubvi/"

"https://rustih.ru/stihi-o-zhizni/"

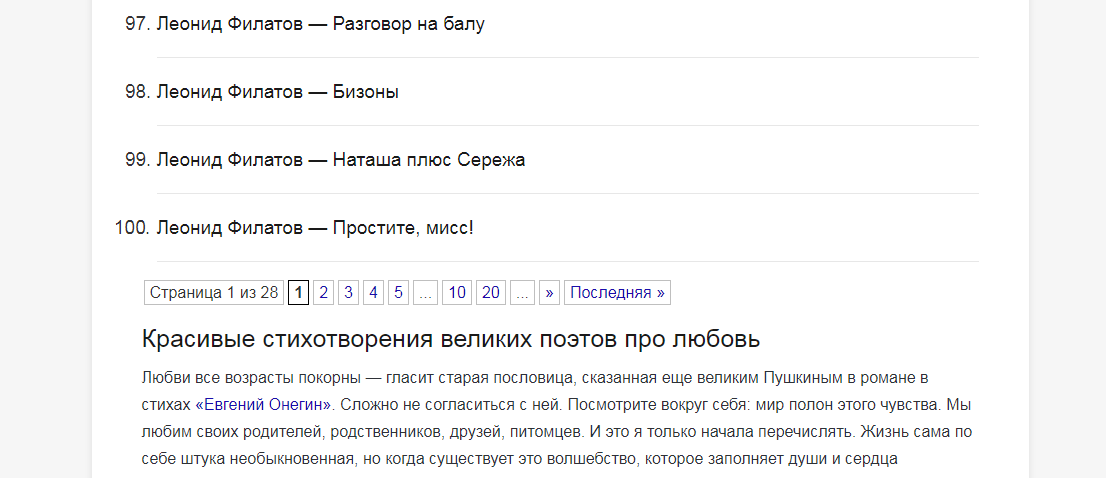
"https://rustih.ru/stixi-o-prirode/"

[4] "https://rustih.ru/stixi-o-rodine-rossii/"

"https://rustih.ru/stixi-o-vojne/"

"https://rustih.ru/stixi-dlya-detej/"

По каждой тематике предлагается несколько страниц стихотворений:



Выполним действия для стихотворений первой тематики (стихи о любви).

Рассмотрим кусок HTML – кода:

|  |
| --- |
| <div class='wp-pagenavi'>  <span class='pages'>Страница 1 из 28</span><span class='current'>1</span><a class="page larger" href="<https://rustih.ru/stihi-o-lubvi/page/2/>">2</a><a class="page larger" href="<https://rustih.ru/stihi-o-lubvi/page/3/>">3</a><a class="page larger"  href="<https://rustih.ru/stihi-o-lubvi/page/4/>">4</a><a class="page larger" href="<https://rustih.ru/stihi-o-lubvi/page/5/>">5</a><span class='extend'>...</span><a class="larger page" href="<https://rustih.ru/stihi-o-lubvi/page/10/>">10</a><a class="larger page"  href="<https://rustih.ru/stihi-o-lubvi/page/20/>">20</a><span class='extend'>...</span><a class="nextpostslink" rel="next" href="<https://rustih.ru/stihi-o-lubvi/page/2/>">»</a><a class="**last**" href="<https://rustih.ru/stihi-o-lubvi/page/28/>">Последняя »</a>  </div> |

|  |
| --- |
|  |
|  |

Для получения номера последней страницы для текущей тематики стихотворений нам нужен тег a с атрибутом class="last". Получив ссылку на последнюю страницу, вытаскиваем ее номер:

**# получаем номер последней страницы:**

> last\_page\_numb <- xpathSApply(catpage, "//a[@class='last']", xmlGetAttr, "href");

> last\_page\_numb = as.numeric(str\_extract(last\_page\_numb, "[:digit:]+"));

Заметим, что url первой страницы имеет вид:

* <https://rustih.ru/stihi-o-lubvi/>

Url последующих страниц имеет вид:

* [https://rustih.ru/stihi-o-lubvi/page/**page\_numb**/](https://rustih.ru/stihi-o-lubvi/page/page_numb/), где **page\_numb** – номер страницы.

**# получим список страниц;**

> list\_of\_pages = character();

> list\_of\_pages = append(list\_of\_pages, category[i]);

> if(length(last\_page\_numb) != 0)

+ list\_of\_pages = append(list\_of\_pages, str\_c(category[i], paste0('page/', 2:last\_page\_numb, '/')));

**# загружаем все страницы на диск:**

> for(j in 1:length(list\_of\_pages)){

+ tmp <- getURL(list\_of\_pages[j], cainfo = signatures, encoding="UTF-8")

+ write(tmp, str\_c(paste0("Category", i, "/"), j, ".html"))

+ Sys.sleep(1);

+ }

Рассмотрим кусок HTML – кода:

|  |
| --- |
| <article class="**blog-item**">  <header>  <li class="title">  <span class="title"><a href="<https://rustih.ru/eduard-asadov-esli-lyubov-uxodit/>">Эдуард Асадов &#8212; Если любовь уходит, какое найти решенье</a></span>  </li>  </header>  </article> |

Для получения ссылок на стихотворения текущей тематики нам нужен тег a , перед которым был тег article с атрибутом class="blog-item":

**#ссылки на стихи;**

> ref = character();

> if(length(last\_page\_numb) == 0)

+ last\_page\_numb = 1;

**# проходим по всем загруженным страницам**

**# получаем ссылки на все стихи текущей тематики**

**# и сохраняем их в переменную ref**

> for(j in 1:last\_page\_numb)

+ {

+ **# читаем текущую страницу**

+ tmp <- readLines(str\_c(paste0("Category", i, "/"), j, ".html"))

+ **# убираем переносы строк:**

+ tmp <- str\_c(tmp, collapse = "")

+ **# строим HTML-объект:**

+ tmp <- htmlParse(tmp, encoding="UTF-8")

+

+ **# получаем ссылки на все стихи текущей тематики**

+ poem\_page <- xpathSApply(tmp, "//article[@class='blog-item']//a[@href]", xmlGetAttr, "href")

+ ref = append(ref, poem\_page)

+ }

Рассмотрим кусок HTML – кода:

|  |
| --- |
| <div class="**poem-text**">  <p>Если любовь уходит, какое найти решенье?<br />  Можно прибегнуть к доводам, спорить и убеждать,<br />  Можно пойти на просьбы и даже на униженья,<br />  Можно грозить расплатой, пробуя запугать.</p>  <p>Можно вспомнить былое, каждую светлую малость,<br />  И, с болью твердя, как горько в разлуке пройдут года,<br />  Поколебать на время, может быть, вызвать жалость<br />  И удержать на время. На время — не навсегда.</p>  <p>А можно, страха и боли даже не выдав взглядом,<br />  Сказать: — Я люблю. Подумай. Радости не ломай. —<br />  И если ответит отказом, не дрогнув, принять, как надо,<br />  Окна и двери — настежь! —Я не держу. Прощай!</p>  <p>Конечно, ужасно трудно, мучась, держаться твердо.<br />  И все-таки, чтоб себя же не презирать потом,<br />  Если любовь уходит — хоть вой, но останься гордым.<br />  Живи и будь человеком, а не ползи ужом!</p>  </div> |

Для получения текста стихотворения нам нужен тег div c атрибутом class="poem-text":

> poem\_text <- xpathSApply(poem\_page, "//div[@class='poem-text']", xmlValue);

**# удаляем из текста знаки переноса строки, табуляции, лишние пробелы:**

> poem\_text = str\_replace\_all(poem\_text, "\n", " ");

> poem\_text = str\_replace\_all(poem\_text, "\t", " ");

> poem\_text = str\_replace\_all(poem\_text, "[ ]{2,}", "");

**# все строки стихотворения записываем в одну переменную full\_text:**

> full\_text = "";

> for(k in 1:length(poem\_text))

+ {

+ full\_text = str\_c(full\_text, paste0(" ", poem\_text[k]));

+ }

**# записываем текст стиха на диск:**

> write(full\_text, paste0("Category", i, "/poems.txt"), append = TRUE)

Повторим те же действия для загрузки на диск текстов стихотворений других тематик. Мы имеем на диске 6 папок вида Category**i**, где **i** – номер категории:

* Category1 - стихи о любви;
* Category2 - стихи о жизни;
* Category3 - стихи о природе;
* Category4 - стихи о Родине;
* Category5 - стихи о войне;
* Category6 - стихи для детей.

В каждой папке есть текстовый файл со стихами poems.txt.

Подготовим тексты для классификации стихотворений классиков по тематикам.

**# пакет Text mining**

> library(tm)

**# получим названия тематики стихов**

> name\_category <- xpathSApply(webpage, "//div[@class='collapse navbar-collapse main-navigation shadow-menu']//a[@class='dropdown-toggle']", xmlValue);

> name\_category

[1] "Стихи о любви" "Стихи о жизни" "Стихи о природе" "Стихи о Родине" "Стихи о войне" "Стихи для детей"

**# прочитаем стихотворения с диска в переменную poems,**

**# а категории, которым принадлежат эти стихи, в переменную categ**

> poems = character(0);

> categ = character(0);

**# для классификации выберем стихотворения о любви, о природе и о войне**

> for(i in c(1, 3, 5))

+ {

+ # читаем стихотворения текущей категории

+ poems\_tmp <- readLines(paste0("Category", i, "/poems.txt"))

+

+ categ\_tmp = rep(name\_category[i], length(poems\_tmp))

+

+ poems = append(poems, poems\_tmp)

+ categ = append(categ, categ\_tmp)

+ }

**# сохраним информацию о стихах в data.frame**

> poems\_frame = data.frame("poem" = poems, "category" = categ)

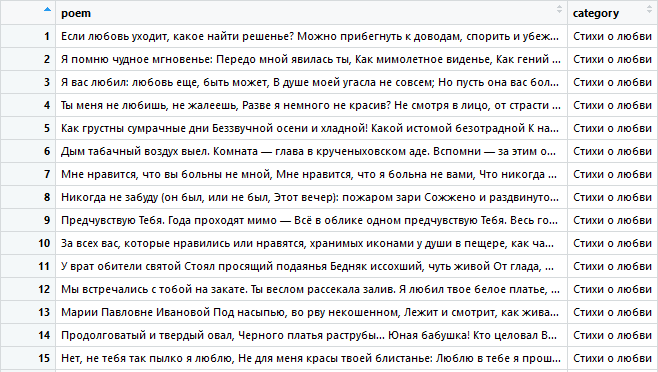
**# общее число стихотворений**

> nrow(poems\_frame)

[1] 4344

Посмотрим содержимое poems\_frame:

> View(poems\_frame)



Заметим, что все стихотворения одной категории расположены подряд. Поэтому перемешаем данные случайный образом:

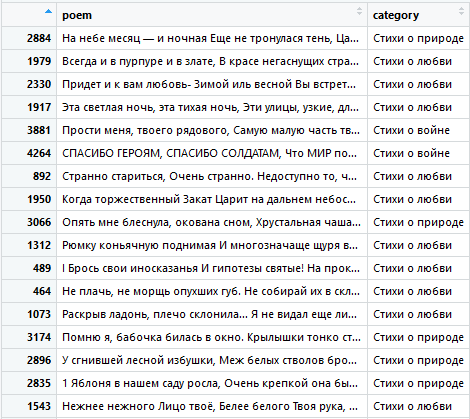
**# перемешаем исходные данные**

> set.seed(0)

> frame = poems\_frame[order(runif(nrow(poems\_frame))),]

**# просмотрим перемешанные данные**

> View(frame)



**# создаем пустой объект корпус**

> poems=list();

> poem\_corpus = VCorpus(VectorSource(poems))

**# добавляем в корпус информацию о стихах всех тематик:**

> for(i in 1:nrow(frame))

+ {

+ **# добавляем в корпус информацию**

+ tmp\_corpus <- VCorpus(VectorSource(frame[i,]$poem))

+ poem\_corpus <- c(poem\_corpus, tmp\_corpus)

+

+ **# добавляем метаинформацию**

+ meta(poem\_corpus[[i]], "category") <- frame[i,]$category

+ meta(poem\_corpus[[i]], "language") <- "ru"

+ }

> poem\_corpus

<<VCorpus>>

Metadata: corpus specific: 0, document level (indexed): 0

Content: documents: 4344

> meta(poem\_corpus[[1]])

author : character(0)

datetimestamp: 2018-11-24 17:29:19

description : character(0)

heading : character(0)

id : 1

language : ru

origin : character(0)

category : Стихи о природе

**# список тематик стихотворений:**

> meta\_data<- data.frame()

> for (i in 1:NROW(poem\_corpus))

+ {

+ meta\_data [i, "category"] <- meta(poem\_corpus[[i]], "category")

+ meta\_data [i, "num"] <- i

+ }

> table(as.character(meta\_data[, "category"]))

Стихи о войне Стихи о любви Стихи о природе

745 2773 826

Выполним преобразования данных.

> poem\_corpus[[16]]$content

[1] " 1 Яблоня в нашем саду росла, Очень крепкой она была. Самой сладкой она слыла, Самым белым цветом цвела. Сучья тяжко к земле склонив, Зрели яблоки белый налив. Зубы врежешь — в гортани мед, Теплым соком гортань зальет. Вот покраснела в лесу листва, Вот забурела в лугах трава, Вот затрещали в печах дрова, Я не перечу — зима права. Онемела земля во льду, Все мертво под луной в саду. Снег подлунный и тот как лед: Голубое сиянье льет. С каждым часом зима сильней, И до нежных живых корней Уж добрался лютой мороз. Спят деревья — не видно слез. Все случилось в глубоком сне, Не помог и глубокий снег. Но расплата близка всегда — В марте месяце с гор вода. Забурлили ручьи-ключи, Заиграли в ручьях лучи, Раскрошились литые льды. Теплый дождик омыл сады. Так ударил расплаты час, Но не все на земле он спас. Что же, яблони, где ваш цвет? Почему же и листьев нет? Вы стоите черны-черны Посреди молодой травы, От дыханья самой весны Не проснулись, деревья, вы. 2 Не сплетаются ветками, Рос не пьют поутру, Но, корявые, редкие, Лишь гремят на ветру. Подгнивают и падают, На дрова их возьмут. Больше солнца не надо им И весна ни к чему. Но выходят из семени Клен, береза, трава. У зеленого племени Не отнимешь права. Глубоки эти корни. Начинается труд. И побеги упорно Пробивают кору. Только выжить до срока, Только на ноги встать, Будет к солнцу дорога — Ни согнуть, ни сломать. Будут сильные листья, Наливные плоды: Только встать, Только выстоять, только быть молодым! "

**# удалим числа**

> poem\_corpus <- tm\_map(poem\_corpus, content\_transformer(removeNumbers))

> poem\_corpus[[16]]$content

[1] " Яблоня в нашем саду росла, Очень крепкой она была. Самой сладкой она слыла, Самым белым цветом цвела. Сучья тяжко к земле склонив, Зрели яблоки белый налив. Зубы врежешь — в гортани мед, Теплым соком гортань зальет. Вот покраснела в лесу листва, Вот забурела в лугах трава, Вот затрещали в печах дрова, Я не перечу — зима права. Онемела земля во льду, Все мертво под луной в саду. Снег подлунный и тот как лед: Голубое сиянье льет. С каждым часом зима сильней, И до нежных живых корней Уж добрался лютой мороз. Спят деревья — не видно слез. Все случилось в глубоком сне, Не помог и глубокий снег. Но расплата близка всегда — В марте месяце с гор вода. Забурлили ручьи-ключи, Заиграли в ручьях лучи, Раскрошились литые льды. Теплый дождик омыл сады. Так ударил расплаты час, Но не все на земле он спас. Что же, яблони, где ваш цвет? Почему же и листьев нет? Вы стоите черны-черны Посреди молодой травы, От дыханья самой весны Не проснулись, деревья, вы. Не сплетаются ветками, Рос не пьют поутру, Но, корявые, редкие, Лишь гремят на ветру. Подгнивают и падают, На дрова их возьмут. Больше солнца не надо им И весна ни к чему. Но выходят из семени Клен, береза, трава. У зеленого племени Не отнимешь права. Глубоки эти корни. Начинается труд. И побеги упорно Пробивают кору. Только выжить до срока, Только на ноги встать, Будет к солнцу дорога — Ни согнуть, ни сломать. Будут сильные листья, Наливные плоды: Только встать, Только выстоять, только быть молодым! "

**# заменим на пробелы знаки пунктуации**

> poem\_corpus <- tm\_map(poem\_corpus,

+ content\_transformer(str\_replace\_all),

+ pattern = "[[:punct:]]", replacement = " ")

> poem\_corpus[[16]]$content

[1] " Яблоня в нашем саду росла Очень крепкой она была Самой сладкой она слыла Самым белым цветом цвела Сучья тяжко к земле склонив Зрели яблоки белый налив Зубы врежешь в гортани мед Теплым соком гортань зальет Вот покраснела в лесу листва Вот забурела в лугах трава Вот затрещали в печах дрова Я не перечу зима права Онемела земля во льду Все мертво под луной в саду Снег подлунный и тот как лед Голубое сиянье льет С каждым часом зима сильней И до нежных живых корней Уж добрался лютой мороз Спят деревья не видно слез Все случилось в глубоком сне Не помог и глубокий снег Но расплата близка всегда В марте месяце с гор вода Забурлили ручьи ключи Заиграли в ручьях лучи Раскрошились литые льды Теплый дождик омыл сады Так ударил расплаты час Но не все на земле он спас Что же яблони где ваш цвет Почему же и листьев нет Вы стоите черны черны Посреди молодой травы От дыханья самой весны Не проснулись деревья вы Не сплетаются ветками Рос не пьют поутру Но корявые редкие Лишь гремят на ветру Подгнивают и падают На дрова их возьмут Больше солнца не надо им И весна ни к чему Но выходят из семени Клен береза трава У зеленого племени Не отнимешь права Глубоки эти корни Начинается труд И побеги упорно Пробивают кору Только выжить до срока Только на ноги встать Будет к солнцу дорога Ни согнуть ни сломать Будут сильные листья Наливные плоды Только встать Только выстоять только быть молодым "

**# преобразуем к нижнему регистру**

> poem\_corpus <- tm\_map(poem\_corpus, content\_transformer(tolower))

> poem\_corpus[[16]]$content

[1] " яблоня в нашем саду росла очень крепкой она была самой сладкой она слыла самым белым цветом цвела сучья тяжко к земле склонив зрели яблоки белый налив зубы врежешь в гортани мед теплым соком гортань зальет вот покраснела в лесу листва вот забурела в лугах трава вот затрещали в печах дрова я не перечу зима права онемела земля во льду все мертво под луной в саду снег подлунный и тот как лед голубое сиянье льет с каждым часом зима сильней и до нежных живых корней уж добрался лютой мороз спят деревья не видно слез все случилось в глубоком сне не помог и глубокий снег но расплата близка всегда в марте месяце с гор вода забурлили ручьи ключи заиграли в ручьях лучи раскрошились литые льды теплый дождик омыл сады так ударил расплаты час но не все на земле он спас что же яблони где ваш цвет почему же и листьев нет вы стоите черны черны посреди молодой травы от дыханья самой весны не проснулись деревья вы не сплетаются ветками рос не пьют поутру но корявые редкие лишь гремят на ветру подгнивают и падают на дрова их возьмут больше солнца не надо им и весна ни к чему но выходят из семени клен береза трава у зеленого племени не отнимешь права глубоки эти корни начинается труд и побеги упорно пробивают кору только выжить до срока только на ноги встать будет к солнцу дорога ни согнуть ни сломать будут сильные листья наливные плоды только встать только выстоять только быть молодым "

**# удаляем стоп-слова**

> poem\_corpus <- tm\_map(poem\_corpus, content\_transformer(removeWords), words = stopwords("ru"))

> poem\_corpus[[16]]$content

[1] " яблоня нашем саду росла очень крепкой самой сладкой слыла самым белым цветом цвела сучья тяжко земле склонив зрели яблоки белый налив зубы врежешь гортани мед теплым соком гортань зальет покраснела лесу листва забурела лугах трава затрещали печах дрова перечу зима права онемела земля льду мертво луной саду снег подлунный лед голубое сиянье льет каждым часом зима сильней нежных живых корней добрался лютой мороз спят деревья видно слез случилось глубоком сне помог глубокий снег расплата близка марте месяце гор вода забурлили ручьи ключи заиграли ручьях лучи раскрошились литые льды теплый дождик омыл сады ударил расплаты час земле спас яблони ваш цвет почему листьев стоите черны черны посреди молодой травы дыханья самой весны проснулись деревья сплетаются ветками рос пьют поутру корявые редкие лишь гремят ветру подгнивают падают дрова возьмут солнца весна чему выходят семени клен береза трава зеленого племени отнимешь права глубоки корни начинается труд побеги упорно пробивают кору выжить срока ноги встать солнцу дорога согнуть сломать будут сильные листья наливные плоды встать выстоять молодым "

**# проводим стемминг**

> for(i in 1:NROW(poem\_corpus))

+ poem\_corpus[[i]]$content <- stemDocument(poem\_corpus[[i]]$content, language = "russian")

> poem\_corpus <- tm\_map(poem\_corpus, content\_transformer(removeWords), words = 'котор')

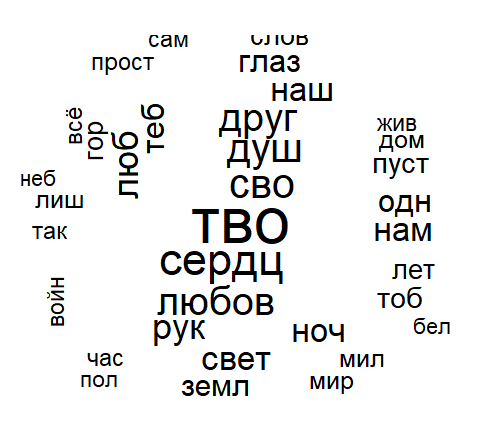
> poem\_corpus[[16]]$content

[1] "яблон наш сад росл очен крепк сам сладк слыл сам бел цвет цвел суч тяжк земл склон зрел яблок бел нал зуб врежеш гортан мед тепл сок гортан зальет покраснел лес листв забурел луг трав затреща печ дров переч зим прав онемел земл льду мертв лун сад снег подлун лед голуб сиян льет кажд час зим сильн нежн жив корн добра лют мороз спят дерев видн слез случ глубок сне помог глубок снег расплат близк март месяц гор вод забурл руч ключ заигра руч луч раскрош лит льды тепл дождик ом сад удар расплат час земл спас яблон ваш цвет поч лист сто черн черн посред молод трав дыхан сам весн проснул дерев сплета ветк рос пьют поутр коряв редк лиш грем ветр подгнива пада дров возьмут солнц весн чем выход семен клен берез трав зелен племен отнимеш прав глубок корн начина труд побег упорн пробива кор выж срок ног встат солнц дорог согнут слома будут сильн лист наливн плод встат выстоя молод"

**# создадим облако слов**

> library(wordcloud)

> wordcloud(poem\_corpus, random.order=F, max.words=40)



**# создаем матрицу терминов-документов**

> tdm <- TermDocumentMatrix(poem\_corpus)

> tdm

<<TermDocumentMatrix (terms: 29665, documents: 4344)>>

Non-/sparse entries: 274517/128590243

Sparsity : 100%

Maximal term length: 35

Weighting : term frequency (tf)

**# создаем матрицу документов-терминов**

> dtm <- DocumentTermMatrix(poem\_corpus)

> dtm

<<DocumentTermMatrix (documents: 4344, terms: 29665)>>

Non-/sparse entries: 274517/128590243

Sparsity : 100%

Maximal term length: 35

Weighting : term frequency (tf)

> dtm <- removeSparseTerms(dtm, 0.95)

> dtm

<<DocumentTermMatrix (documents: 4344, terms: 166)>>

Non-/sparse entries: 63670/657434

Sparsity : 91%

Maximal term length: 8

Weighting : term frequency (tf)

**# список терминов**

> dtm$dimnames$Terms

[1] "бед" "бел" "бог" "бол" "буд" "век"

[7] "вер" "верн" "весел" "весн" "ветер" "ветр"

[13] "вечер" "вечн" "взгляд" "взор" "вид" "внов"

[19] "вод" "войн" "волн" "врем" "всё" "всем"

[25] "глаз" "гляд" "говор" "год" "голов" "голос"

[31] "голуб" "гор" "груд" "губ" "давн" "дал"

[37] "далек" "дел" "ден" "дне" "добр" "долг"

[43] "дом" "дорог" "друг" "дума" "душ" "дым"

[49] "жив" "жизн" "жит" "заб" "зар" "звезд"

[55] "зелен" "земл" "зна" "золот" "кажд" "как"

[61] "кра" "кров" "легк" "лес" "лет" "лист"

[67] "лиц" "лиш" "лун" "луч" "люб" "любв"

[73] "любим" "любл" "любов" "люд" "мечт" "мил"

[79] "мир" "мно" "мог" "молод" "мор" "нам"

[85] "наш" "неб" "нежн" "нем" "нов" "ног"

[91] "ноч" "огн" "одн" "окн" "осен" "перв"

[97] "песн" "плеч" "пок" "пол" "полн" "помн"

[103] "пор" "последн" "прекрасн" "прост" "прошл" "птиц"

[109] "пуст" "пут" "радост" "ран" "рек" "родн"

[115] "рук" "ряд" "сад" "сам" "свет" "светл"

[121] "сво" "сердц" "сил" "син" "сквоз" "след"

[127] "слез" "слов" "словн" "слыш" "смерт" "смотр"

[133] "снег" "снов" "солнц" "сон" "сред" "стал"

[139] "стар" "стих" "сто" "стран" "страст" "страшн"

[145] "судьб" "счаст" "тайн" "так" "тво" "теб"

[151] "тел" "темн" "тен" "тепл" "тих" "тишин"

[157] "тоб" "тоск" "трав" "улыбк" "утр" "холодн"

[163] "цвет" "цел" "час" "черн"

**# дополнительный пакет для работы с текстами**

> library(RTextTools)

**# массив меток – тематики стихотворений**

> org\_labels<-meta\_data[, "category"]

> count\_data <- length(org\_labels)

> count\_data

[1] 4344

**# разделим исходные данные на тестовую и обучающую выборки**

**# число стихов обучающей выборки (70% от исходных данных)**

> count\_train = round(count\_data\*0.7)

> count\_train

[1] 3041

**# для применения методов классификации создаем контейнер**

> container <- create\_container(

+ dtm,

+ labels = org\_labels,

+ trainSize = 1:count\_train,

+ testSize = (count\_train+1):count\_data,

+ virgin = FALSE

+ )

**# обучение моделей**

**# Support vector machine**

> svm\_model <- train\_model(container, "SVM")

**# Random forest**

> tree\_model <- train\_model(container, "TREE")

**# Maximal entropy**

> maxent\_model <- train\_model(container, "MAXENT")

**# запускаем классификацию**

> svm\_out <- classify\_model(container, svm\_model)

> tree\_out <- classify\_model(container, tree\_model)

> maxent\_out <- classify\_model(container, maxent\_model)

**# создаем общую матрицу меток**

> labels\_out <- data.frame(

+ correct\_label = org\_labels[(count\_train+1):count\_data],

+ svm = as.character(svm\_out[,1]),

+ tree = as.character(tree\_out[,1]),

+ maxent = as.character(maxent\_out[,1]),

+ stringsAsFactors = F)

**# SVM performance**

> table(labels\_out[,1] == labels\_out[,2])

FALSE TRUE

344 959

**# Random forest performance**

> table(labels\_out[,1] == labels\_out[,3])

FALSE TRUE

366 937

**# Maximum entropy performance**

> table(labels\_out[,1] == labels\_out[,4])

FALSE

1303

Таким образом, метод SVM допустил 344 ошибки из 1303 (точность метода составляет 73,6%); метод Random forest допустил 366 ошибок из 1303 (точность метода составляет 71,9%); метод Maximum entropy не подходит для решения данной задачи, он ошибается для любого тестового образца.

Попробуем уменьшить разреженность матрицы документов-терминов:

> dtm <- removeSparseTerms(dtm, 0.9)

> dtm

<<DocumentTermMatrix (documents: 4344, terms: 42)>>

Non-/sparse entries: 26387/156061

Sparsity : 86%

Maximal term length: 5

Weighting : term frequency (tf)

Было 166 терминов, стало 42.

**# список терминов**

> dtm$dimnames$Terms

[1] "бел" "глаз" "говор" "гор" "ден" "дом" "дорог" "друг" "душ"

[10] "жив" "жизн" "земл" "лет" "лиш" "люб" "любв" "любов" "мил"

[19] "мир" "мно" "нам" "наш" "неб" "ноч" "одн" "пор" "прост"

[28] "пуст" "рук" "сам" "свет" "сво" "сердц" "слов" "сто" "счаст"

[37] "так" "тво" "теб" "тих" "тоб" "час"

В этом случае точность методов классификации уменьшится:

**# SVM performance**

> table(labels\_out[,1] == labels\_out[,2])

FALSE TRUE

456 847

**# Random forest performance**

> table(labels\_out[,1] == labels\_out[,3])

FALSE TRUE

479 824

**# Maximum entropy performance**

> table(labels\_out[,1] == labels\_out[,4])

FALSE

1303

Попробуем увеличить разреженность матрицы документов-терминов:

> dtm <- removeSparseTerms(dtm, 0.97)

> dtm

<<DocumentTermMatrix (documents: 4344, terms: 332)>>

Non-/sparse entries: 91483/1350725

Sparsity : 94%

Maximal term length: 8

Weighting : term frequency (tf)

Количество терминов стало 332.

**# список терминов**

> dtm$dimnames$Terms

[1] "бед" "бел" "берег" "бледн" "бог" "бол"

[7] "больн" "больш" "брат" "буд" "бур" "быва"

[13] "ваш" "вдал" "век" "велик" "вер" "верн"

[19] "весел" "весн" "ветер" "ветр" "вечер" "вечн"

[25] "взгляд" "взор" "вид" "видел" "видн" "виж"

[31] "вин" "вмест" "внов" "вод" "воздух" "войн"

[37] "вокруг" "вол" "волн" "вперед" "враг" "врем"

[43] "всё" "всем" "вспомн" "встрет" "встреч" "встреча"

[49] "вся" "высок" "глаз" "глубок" "гляд" "говор"

[55] "год" "голов" "голос" "голуб" "гор" "горд"

[61] "город" "горьк" "горяч" "гроз" "гром" "груд"

[67] "груст" "грустн" "губ" "давн" "дал" "далек"

[73] "дальн" "дар" "двер" "дел" "ден" "дерев"

[79] "дет" "дне" "дни" "дня" "добр" "дожд"

[85] "долг" "дом" "дорог" "дрож" "друг" "друз"

[91] "дума" "душ" "дым" "дыхан" "дыш" "едв"

[97] "жал" "жела" "жен" "женщин" "жив" "жизн"

[103] "жил" "жит" "заб" "зар" "звезд" "звук"

[109] "зелен" "земл" "земн" "зим" "зна" "знаком"

[115] "знал" "знат" "золот" "игра" "кажд" "как"

[121] "ког" "ком" "кон" "конц" "кра" "крас"

[127] "красн" "красот" "кров" "круг" "крыл" "куст"

[133] "ласк" "легк" "лес" "лет" "лист" "лиц"

[139] "лиш" "луг" "лун" "луч" "люб" "любв"

[145] "любим" "любл" "любов" "люд" "мал" "мат"

[151] "мгновен" "меж" "мертв" "мест" "месяц" "мечт"

[157] "миг" "мил" "минут" "мир" "мно" "мог"

[163] "могл" "мол" "молод" "молч" "мор" "мрак"

[169] "мук" "мысл" "навек" "наверн" "надежд" "нам"

[175] "народ" "наш" "неб" "небес" "нежн" "нем"

[181] "нов" "ног" "ноч" "ночн" "нужн" "облак"

[187] "огн" "огон" "одн" "окн" "осен" "оста"

[193] "ответ" "отда" "очен" "пел" "пен" "перв"

[199] "песн" "печа" "печал" "печальн" "пламен" "плеч"

[205] "побед" "пок" "пол" "полн" "помн" "поня"

[211] "пор" "последн" "поэт" "прав" "пред" "прекрасн"

[217] "природ" "приход" "пришл" "прост" "прошл" "птиц"

[223] "пуска" "пуст" "пустын" "пут" "пыл" "равн"

[229] "рад" "радост" "разлук" "ран" "рассвет" "рек"

[235] "реч" "родн" "роз" "рук" "ряд" "сад"

[241] "сам" "свет" "светл" "сво" "свят" "сердц"

[247] "сил" "сильн" "син" "скаж" "сквоз" "склон"

[253] "скольк" "скор" "слав" "сладк" "след" "слез"

[259] "слов" "словн" "слуша" "слыш" "смерт" "смотр"

[265] "снег" "снов" "соб" "солдат" "солнц" "сон"

[271] "спокойн" "сред" "стал" "стан" "стар" "стен"

[277] "стих" "сто" "сторон" "стоя" "стран" "страст"

[283] "страшн" "стру" "судьб" "счаст" "счастлив" "сын"

[289] "тайн" "так" "тво" "теб" "тел" "темн"

[295] "тен" "тепл" "тех" "тих" "тишин" "тоб"

[301] "том" "тоск" "трав" "труд" "трудн" "тума"

[307] "туч" "тяжел" "узна" "улыбк" "уст" "утр"

[313] "уход" "ход" "холодн" "хорош" "хот" "хотел"

[319] "хоч" "цвет" "цел" "час" "част" "черн"

[325] "черт" "чист" "чувств" "чуж" "шаг" "шум"

[331] "ярк" "ясн"

**# SVM performance**

> table(labels\_out[,1] == labels\_out[,2])

FALSE TRUE

297 1006

**# Random forest performance**

> table(labels\_out[,1] == labels\_out[,3])

FALSE TRUE

366 937

**# Maximum entropy performance**

> table(labels\_out[,1] == labels\_out[,4])

FALSE

1303

Как видим, точность метода SVM повысилась до 77,2%, точность метода Random forest не изменилась. Дальнейшее увеличение разреженности матрицы не привело к увеличению точности классификации.

Попробуем запустить классификацию стихов для 5 тематик:

* стихи о любви;
* стихи о жизни;
* стихи о природе;
* стихи о Родине;
* стихи о войне.

**# общее число стихотворений**

> nrow(poems\_frame)

[1] 5629

> table(as.character(meta\_data[, "category"]))

Стихи о войне Стихи о жизни Стихи о любви Стихи о природе

745 727 2773 826

Стихи о Родине

558

> dtm <- DocumentTermMatrix(poem\_corpus)

> dtm

<<DocumentTermMatrix (documents: 5629, terms: 33360)>>

Non-/sparse entries: 350890/187432550

Sparsity : 100%

Maximal term length: 35

Weighting : term frequency (tf)

> dtm <- removeSparseTerms(dtm, 0.97)

> dtm

<<DocumentTermMatrix (documents: 5629, terms: 331)>>

Non-/sparse entries: 115590/1747609

Sparsity : 94%

Maximal term length: 8

Weighting : term frequency (tf)

**# SVM performance**

> table(labels\_out[,1] == labels\_out[,2])

FALSE TRUE

617 1072

**# Random forest performance**

> table(labels\_out[,1] == labels\_out[,3])

FALSE TRUE

698 991

**# Maximum entropy performance**

> table(labels\_out[,1] == labels\_out[,4])

FALSE

1689

Метод SVM допустил 617 ошибок из 1689 (точность метода составляет 63,5%); метод Random forest допустил 698 ошибок из 1689 (точность метода составляет 58,7%); метод Maximum entropy не подходит для решения данной задачи, он ошибается для любого тестового образца.

**Код программы:**

# назначим рабочий директорий

setwd("C:/Users/TatyanaAnisimova/Documents/Rprojects")

library(RCurl) # основной пакет для веб-скрепинга

library(XML) # пакет для работы с XML, HTML, Xpath

library(stringr) # пакет для работы с регулярными выражениями

# стихи русских и зарубежных классиков

main = "https://rustih.ru/"

# создаем объект для работы по протоколу SSL

signatures = system.file("CurlSSL", cainfo = "cacert.pem", package = "RCurl")

url = getURL(main, cainfo = signatures, encoding="UTF-8")

# создаем html-объект

webpage <- htmlParse(url, encoding="UTF-8")

webpage

# вытащим информацию о тематике стихотворений

category <- xpathSApply(webpage, "//div[@class='collapse navbar-collapse main-navigation shadow-menu']//a[@class='dropdown-toggle']", xmlGetAttr, "href");

category = category[-8];

category = category[-7];

category

for(i in 1:length(category))

{

categ\_url=getURL(category[i], cainfo = signatures, encoding="UTF-8")

catpage <- htmlParse(categ\_url, encoding="UTF-8")

catpage

# получаем номер последней страницы:

last\_page\_numb <- xpathSApply(catpage, "//a[@class='last']", xmlGetAttr, "href");

last\_page\_numb = as.numeric(str\_extract(last\_page\_numb, "[:digit:]+"));

last\_page\_numb

# получим список страниц;

list\_of\_pages = character();

list\_of\_pages = append(list\_of\_pages, category[i]);

if(length(last\_page\_numb) != 0)

list\_of\_pages = append(list\_of\_pages, str\_c(category[i], paste0('page/', 2:last\_page\_numb, '/')));

list\_of\_pages

# загружаем все страницы на диск:

for(j in 1:length(list\_of\_pages)){

tmp <- getURL(list\_of\_pages[j], cainfo = signatures, encoding="UTF-8")

write(tmp, str\_c(paste0("Category", i, "/"), j, ".html"))

Sys.sleep(1);

}

#ссылки на стихи;

ref = character();

if(length(last\_page\_numb) == 0)

last\_page\_numb = 1;

# проходим по всем загруженным страницам

# получаем ссылки на все стихи текущей тематики и сохраняем их в переменную ref

for(j in 1:last\_page\_numb)

{

# читаем текущую страницу

tmp <- readLines(str\_c(paste0("Category", i, "/"), j, ".html"))

# убираем переносы строк:

tmp <- str\_c(tmp, collapse = "")

# строим HTML-объект:

tmp <- htmlParse(tmp, encoding="UTF-8")

# получаем ссылки на все стихи текущей тематики

poem\_page <- xpathSApply(tmp, "//article[@class='blog-item']//a[@href]", xmlGetAttr, "href")

ref = append(ref, poem\_page)

}

# записываем на диск тексты всех стихотворений текущей тематики

for(j in 1:length(ref))

{

print(j)

poem\_url= getURL(ref[j], cainfo = signatures, encoding="UTF-8");

poem\_page <- htmlParse(poem\_url, encoding="UTF-8")

poem\_text <- xpathSApply(poem\_page, "//div[@class='poem-text']//h2", xmlValue)

if(length(poem\_text) != 0){

poem\_text <- xpathSApply(poem\_page, "//div[@class='poem-text']//p[following-sibling::style]", xmlValue)

}else

{

poem\_text <- xpathSApply(poem\_page, "//div[@class='poem-text']", xmlValue);

}

# удаляем из текста знаки переноса строки, табуляции, лишние пробелы:

poem\_text = str\_replace\_all(poem\_text, "\n", " ");

poem\_text = str\_replace\_all(poem\_text, "\t", " ");

poem\_text = str\_replace\_all(poem\_text, "[ ]{2,}", "");

# все строки стихотворения записываем в одну переменную full\_text:

full\_text = "";

for(k in 1:length(poem\_text))

{

full\_text = str\_c(full\_text, paste0(" ", poem\_text[k]));

}

# записываем текст стиха на диск:

write(full\_text, paste0("Category", i, "/poems.txt"), append = TRUE)

}

}

# пакет Text mining

library(tm)

# получим названия тематики стихов

name\_category <- xpathSApply(webpage, "//div[@class='collapse navbar-collapse main-navigation shadow-menu']//a[@class='dropdown-toggle']", xmlValue);

name\_category=name\_category[-8]

name\_category=name\_category[-7]

name\_category

# прочитаем стихотворения с диска в переменную poems,

# а категории, которым принадлежат эти стихи, в переменную categ

poems = character(0);

categ = character(0);

# для классификации выберем стихотворения о любви, о природе и о войне

for(i in c(1, 3, 5))

{

# читаем стихотворения текущей категории

poems\_tmp <- readLines(paste0("Category", i, "/poems.txt"))

categ\_tmp = rep(name\_category[i], length(poems\_tmp))

poems = append(poems, poems\_tmp)

categ = append(categ, categ\_tmp)

}

# сохраним информацию о стихах в data.frame

poems\_frame = data.frame("poem" = poems, "category" = categ)

# общее число стихотворений

nrow(poems\_frame)

# перемешаем исходные данные

set.seed(0)

frame = poems\_frame[order(runif(nrow(poems\_frame))),]

# создаем пустой объект корпус

poems=list();

poem\_corpus = VCorpus(VectorSource(poems))

# добавляем в корпус информацию о стихах всех тематик:

for(i in 1:nrow(frame))

{

# добавляем в корпус информацию

tmp\_corpus <- VCorpus(VectorSource(frame[i,]$poem))

poem\_corpus <- c(poem\_corpus, tmp\_corpus)

# добавляем метаинформацию

meta(poem\_corpus[[i]], "category") <- frame[i,]$category

meta(poem\_corpus[[i]], "language") <- "ru"

}

poem\_corpus

meta(poem\_corpus[[1]])

# список тематик стихотворений:

meta\_data<- data.frame()

for (i in 1:NROW(poem\_corpus))

{

meta\_data [i, "category"] <- meta(poem\_corpus[[i]], "category")

meta\_data [i, "num"] <- i

}

table(as.character(meta\_data[, "category"]))

poem\_corpus[[16]]$content

# удалим числа

poem\_corpus <- tm\_map(poem\_corpus, content\_transformer(removeNumbers))

poem\_corpus[[16]]$content

# заменим на пробелы знаки пунктуации

poem\_corpus <- tm\_map(poem\_corpus,

content\_transformer(str\_replace\_all),

pattern = "[[:punct:]]", replacement = " ")

poem\_corpus[[16]]$content

# преобразуем к нижнему регистру

poem\_corpus <- tm\_map(poem\_corpus, content\_transformer(tolower))

poem\_corpus[[16]]$content

# удаляем стоп-слова

poem\_corpus <- tm\_map(poem\_corpus, content\_transformer(removeWords), words = stopwords("ru"))

poem\_corpus[[16]]$content

# проводим стемминг

for(i in 1:NROW(poem\_corpus))

poem\_corpus[[i]]$content <- stemDocument(poem\_corpus[[i]]$content, language = "russian")

poem\_corpus <- tm\_map(poem\_corpus, content\_transformer(removeWords), words = 'котор')

poem\_corpus[[16]]$content

# создадим облако слов

library(wordcloud)

wordcloud(poem\_corpus, random.order=F, max.words=40)

# создаем матрицу терминов-документов

tdm <- TermDocumentMatrix(poem\_corpus)

tdm

# создаем матрицу документов-терминов

dtm <- DocumentTermMatrix(poem\_corpus)

dtm

dtm <- removeSparseTerms(dtm, 0.97)

dtm

# список терминов

dtm$dimnames$Terms

# дополнительный пакет для работы с текстами

library(RTextTools)

# массив меток – тематики стихотворений

org\_labels<-meta\_data[, "category"]

count\_data <- length(org\_labels)

count\_data

# разделим исходные данные на тестовую и обучающую выборки

# число стихов обучающей выборки (70% от исходных данных)

count\_train = round(count\_data\*0.7)

count\_train

# для применения методов классификации создаем контейнер

container <- create\_container(

dtm,

labels = org\_labels,

trainSize = 1:count\_train,

testSize = (count\_train+1):count\_data,

virgin = FALSE

)

container

# проверим репрезентативность выборки

round(prop.table(table(frame[2]))\*100, digits = 1)

train\_data = frame[1:count\_train,]

round(prop.table(table(train\_data[2]))\*100, digits = 1)

# обучение моделей

# Support vector machine

svm\_model <- train\_model(container, "SVM")

# Random forest

tree\_model <- train\_model(container, "TREE")

# Maximal entropy

maxent\_model <- train\_model(container, "MAXENT")

# запускаем классификацию

svm\_out <- classify\_model(container, svm\_model)

tree\_out <- classify\_model(container, tree\_model)

maxent\_out <- classify\_model(container, maxent\_model)

# создаем общую матрицу меток

labels\_out <- data.frame(

correct\_label = org\_labels[(count\_train+1):count\_data],

svm = as.character(svm\_out[,1]),

tree = as.character(tree\_out[,1]),

maxent = as.character(maxent\_out[,1]),

stringsAsFactors = F)

# SVM performance

table(labels\_out[,1] == labels\_out[,2])

# Random forest performance

table(labels\_out[,1] == labels\_out[,3])

# Maximum entropy performance

table(labels\_out[,1] == labels\_out[,4])