**Федеральное государственное автономное образовательное учреждение высшего образования**

**КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**

**ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И**

**ИНФОРМАЦИОННЫХ СИСТЕМ**

Направление подготовки: 09.03.03 – Прикладная информатика

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**Разработка программного комплекса для автоматизированной модерации сообщений в социальных сетях**

Работа завершена:

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2016г.

Студент группы 11-204 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Р. Бикеев

Работа допущена к защите:

Научный руководитель,

Научный сотрудник кафедры ИТП

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2016г.                  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Р.М. Гареев

Директор Высшей школы ИТИС

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2016г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Ф. Хасьянов

Казань – 2016г.

# Содержание

[Содержание](#h.87adz57xcj7l)

[Введение](#h.mcwm6puk7efn)

[Проблема](#h.ue1ufaiingc)

[Актуальность](#h.2rbph05ju33w)

[Мотивация](#h.50l2ed7q7d7t)

[Постановка задачи](#h.fi7v1l5tdd8a)

[Обзор близких работ](#h.ru71u7ixu08a)

[Наиболее близкие научные работы](#h.qshqerqh1dad)

[Коммерческие продукты](#h.qzuylnnyc49n)

[Разработка метода](#h.yw18v82czwts)

[Визуализация размеченных данных в разрезе различных характеристик.](#h.uhqxhjdwlc1u)

[Программная реализация](#h.jppm9nzcs4dm)

[Описание основных компонентов](#h.3m3rqn7r9lzp)

[Сложности](#h.v7yipyayu0db)

[Тестирование и оценка эффективности](#h.3aaoetic6ang)

[Заключение](#h.i1w8v3g7otp5)

[Рекомендации ко внедрению](#h.l28dm2fugzwo)

[Ссылки](#h.tepx94dwlwg2)

[Приложения](#h.m6whajnh3yf6)

[№1. Алгоритм подбора коэффициентов и степени полинома логистической регрессии (Matlab/Octave).](#h.hfeog1pe60vq)

[№2. Пример размеченного набора данных:](#h.cxky4nkix9i6)

# Введение

## Проблема

В потоке создаваемых пользователями сообщений в социальных сетях очень сложно отсеивать нерелевантные, неактуальные или некорректные сообщения.

## Актуальность

С ростом числа пользователей сообществ в социальных сетях увеличивается и количество создаваемого ими контента, что приводит к усложнению процесса модерации. Данный программный комплекс предназначен для автоматизации модерации пользовательских сообщений в социальных сообществах.

## Мотивация

Реализация программного комплекса заинтересовала масс-медиа сообщество РБК Вконтакте, где используется на сегодняшний день.

# Постановка задачи

Создать систему для анализа и фильтрации пользовательских комментариев в социальной сети Вконтакте. Для этого необходимо:

1. Разметить набор данных.
2. Исследовать возможные характеристики качества контента.
3. Разработать метод оценки качества комментариев на основе исследованных характеристик.
4. Реализовать систему фильтрации комментариев на основе разработанного метода.
5. Интегрировать систему в конкретное сообщество в социальной сети.

## 

# Обзор близких работ

Достаточно распространённый способ решения данной задачи - вручную. Например, критерии, которыми пользуются модераторы сообщества РБК Вконтакте, удалению подлежат:

* Нерелевантные комментарии
* Некорректные комментарии - оскорбления, слишком длинные комментарии или большей частью в верхнем регистре
* Реклама и спам

## Наиболее близкие научные работы

* Validates\_text\_content [4] - небольшое исследование возможностей автоматической модерации контента, основанное на статистическом анализе следующих характеристик текста:
  + Отношение количества знаков препинания к среднему для языка.
  + Отношение среднего количества букв в слове или слов в предложении к средним значениям.
  + Количество заглавных букв относительно всего текста (как минимум < 75% от общего объёма).
* SentiScan [5] - система анализа тональности, которая способна давать оценку объективности/субъективности текста.
* Creating Subjective and Objective Sentence Classifiers from Unannotated Texts [6] - исследование алгоритма классификации объективности/субъективности текста на основе неаннотированного набора данных.

## Коммерческие продукты

* Nexgate[[1]](#footnote-2) - система анализа и фильтрации пользовательского контента в сети интернет, созданная, преимущественно для защиты компаний от использования бренда в нелегальных целях, таких как:
  + Рассылка спама от лица бренда
  + Мошенничество
  + Оскорбления
* Microsoft Content Moderator[[2]](#footnote-3) - схожа по принципу работы с Nexgate.
* IoSquare[[3]](#footnote-4) - система способная обучаться на принимаемых модераторами решениях для дальнейшей автоматизации.
* Validates\_text\_content[[4]](#footnote-5) - библиотека на Ruby для автоматической пре-модерации контента на основе исследования [4].

# Разработка метода

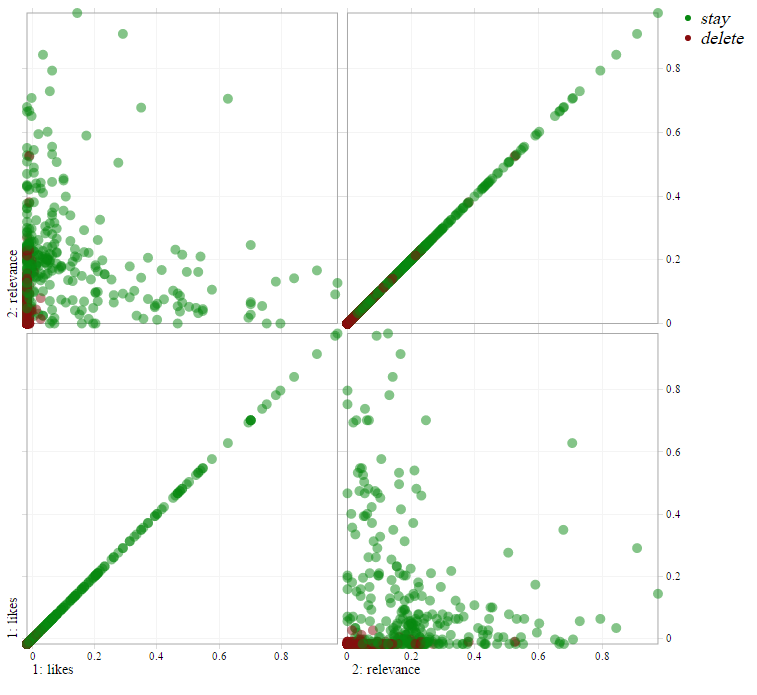
С течением времени, автором данной работы, совместно с модераторами РБК, был собран, размечен и проанализирован набор данных размером 1250 комментариев. Пример такого набора данных можно посмотреть в приложении 2.

На основе данных были исследованы следующие характеристики комментариев:

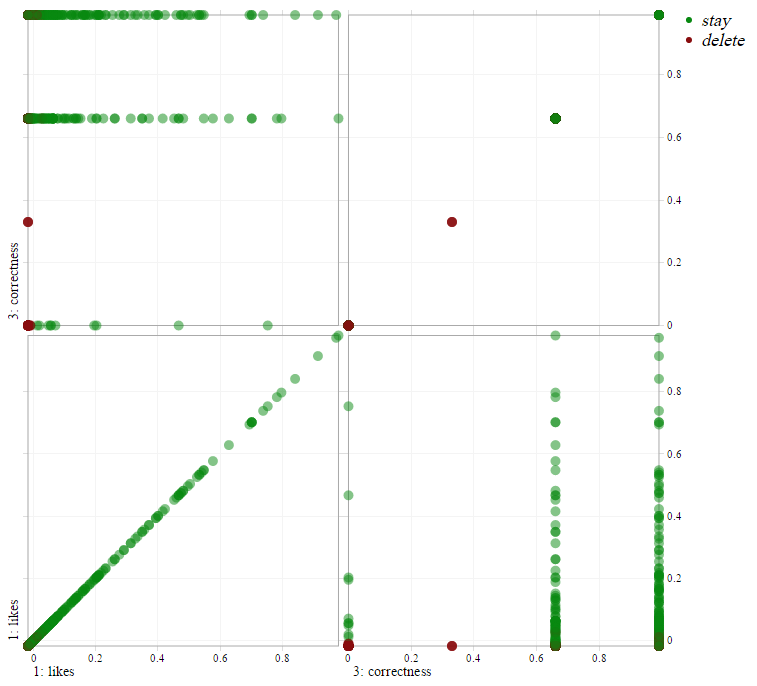
1. Привлекательность - базовая, наиболее весомая и самая тривиальная характеристика, её источником служат отметки “Мне нравится” от других пользователей.
2. Обратная релевантность комментарий-пост (в дальнейшем просто релевантность) - характеристика, основанная на текущих постах в сообществе. Пологая, что комментарий это запрос, а посты это документы - определяем числовое значение релевантности наиболее подходящего поста, используя функцию ранжирования (в данном случае была использована BM25 [2]). Таким образом, полученное значение так же может быть трактовано наоборот, как оценка релевантности самого запроса (комментария) относительно документа (поста).
3. Корректность - опираясь на размеченные данные, были определены следующие критерии корректности комментария:
   * Количество символов пунктуации по отношению к количеству слов в комментарие должно находиться в диапазоне от 0.1 до 1.
   * Как минимум 50% текста должно быть в нижнем регистре.
   * Количество слов должно быть в диапазоне от 2 до 180.

## Визуализация размеченных данных в разрезе различных характеристик.

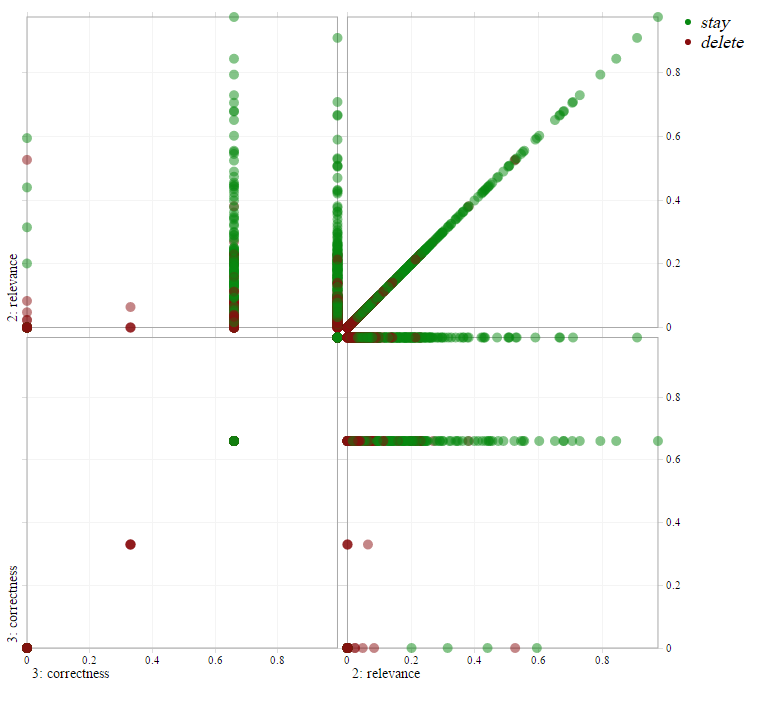
График, описывающий зависимость характеристики привлекательности от релевантности:



Зависимость характеристики привлекательности от корректности:



Зависимость характеристики релевантности от корректности:



Был разработан метод классификации комментариев на два класса: “удалить”, “оставить” на основе алгоритма логистической регрессии [8]. Для применения данного алгоритма, каждый элемент набора характеристик описывающих комментарий представлен в векторной форме: , а значение класса полученное на этапе разметки: ( - “оставить”, - “удалить”) . Далее, используя алгоритм логистической регрессии, была построена разделяющая плоскость, минимизирующая ошибку на части размеченного набора данных (примерно 70% от всего набора). Уравнение, этой плоскости, имеет вид:

где **w** - вектор весов для характеристик , подобранный методом минимизации функции ошибки Cross entropy loss [3].

Таким образом, принятие решения о принадлежности нового, неразмеченного комментария, описываемого вектором характеристик , тому или иному классу происходит по следующей формуле:

где - логистическая функция или сигмоида имеющая следующий вид:

860d15104c9075eebbc6ded06873cd19.png

Исходный код алгоритма подбора весовых коэффициентов на языке Matlab представлен в приложении 1.

# 

# Программная реализация

Программный комплекс реализован на языках Scala и Matlab с использованием интегрированных сред разработки Intelij IDEA и Octave.

Развёртывание комплекса происходило на выделенном сервере в операционной системе CentOS 7.

Использованные библиотеки:

* Akka[[5]](#footnote-6)
* Specs2[[6]](#footnote-7)
* Cats[[7]](#footnote-8)
* Spray[[8]](#footnote-9)
* nscala-time[[9]](#footnote-10)

## Описание основных компонентов

В среде Octave был разработан алгоритм подбора весовых коэффициентов и степени полинома для вектора характеристик комментариев (исходный код представлен в приложении 1 и в репозитории проекта).

На языке Scala был реализован программный комплекс, использующий подобранный полином для классификации новых комментариев и их модерации через API (исходный код представлен в репозитории проекта [1]).

Основным элементом на уровне данных в программном комплексе является субъект - хранилище комментариев и их характеристик.

На уровне логики находятся:

* Системы извлечения характеристик - функции отображающие субъекты в нормированное (в пределах от -1 до 1) числовое значение, например релевантности. Для каждой исследованной характеристики присутствует своя, одноимённая система извлечения.
* Система принятия решения - функция, обобщающая извлечённые характеристики и присваивающая метку класса субъекту, используя весовые коэффициенты и степень полинома, полученные на этапе их подбора.

Отдельно стоит выделить компонент для непрерывного взаимодействия с API, так как к нему предъявлялись особые требования надёжности и стабильности - он должен работать бесперебойно на протяжении длительного времени. Реализован он был в формате алгоритма Producer-Consumer [7] на фреймворке Akka.

Дополнительные сервисы:

* Серверный клиент для Vk API
* Клиент для системы полнотекстового поиска Elasticsearch
* Web интерфейс

Исходный код программного сервиса имеет гибкую, модульную архитектуру и покрыт unit-тестами.

## 

## 

## Сложности

Учитывая большие объёмы данных (модерируется 50+ верхних постов на странице сообщества) и то, что характеристика пользовательской привлекательности зависит от времени (для правильной оценки - нужно постоянно оценивать время, прошедшее с момента написания комментария), появляется необходимость очень частого обращаться к API, что ограничивается сервисом Вконтакте. Частичным решением данной проблемы стало использование метода API `execute`[[10]](#footnote-11), что позволило отправлять множетсво запросов за раз в формате скрипта на языке в формате ECMAScript. В дальнейшем есть смысл использовать недавно реализованный callback API[[11]](#footnote-12) Вконтакте.

Кроме того Scala - сложный язык, особенно если писать в функциональном стиле. За время работы на нём, были определены следующие преимущества и недостатки.

Плюсы:

* Хорошая поддержка функционального программирования.
* Много готовых и удобных функциональных комбинаторов.
* Мощная система типов.
* Статическая типизация.

Минусы:

* Недостаточная поддержка функционального программирования на уровне языка, а основные функциональные библиотеки либо слишком теоретизированны (ScalaZ[[12]](#footnote-13)), либо ещё сырые (Cats)
* Сложная для освоения.
* Долгая компиляция.
* Плохая поддержка со стороны IDE.
* Плохая обратная совместимость.
* Субъективно неудобные регулярные выражения.

# Тестирование и оценка эффективности

Ниже представлен сравнительный анализ разработанных характеристик для части размеченного набора данных (148 элементов).

Для анализа были использованы:

* Таблицы смежности, вида:

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Оценка асессора: | |
| Удалить | Оставить |
| Оценка системы: | Удалить | TP | FP |
| Оставить | FN | TN |

В таблице содержится информация о том, сколько раз была дана верная оценка класса и сколько раз неверная. А именно:

* + TP — истинно-положительное решение;
  + TN — истинно-отрицательное решение;
  + FP — ложно-положительное решение;
  + FN — ложно-отрицательное решение.
* Основные меры качества классификации:
  + Аккуратность:

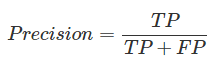
Accur.PNG

где, P – количество документов, по которым классификатор принял правильное решение, а N – размер всего множества.

* + Полнота:

Recall.PNG

* + Точность:



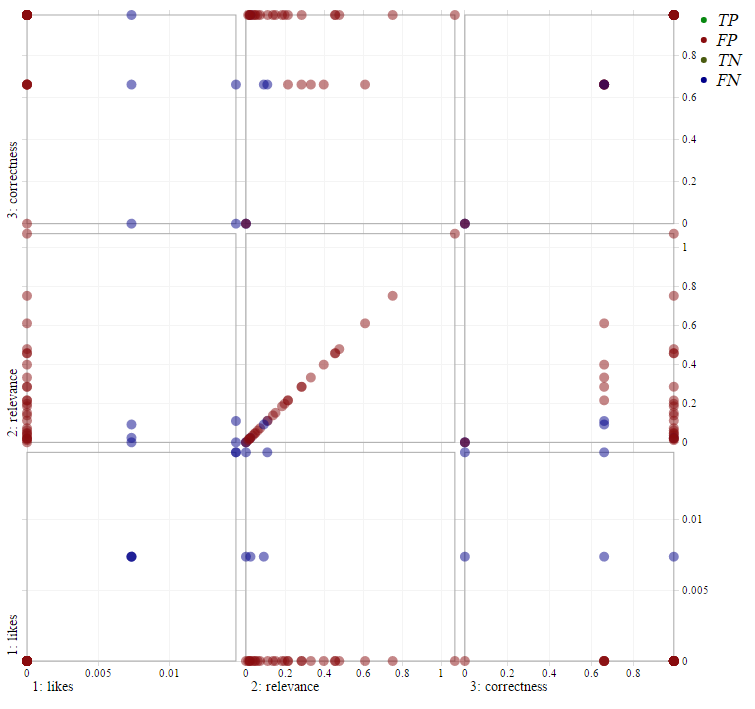
* Так же представлен вес полинома подобранный с помощью алгоритма оптимизации представленного в приложении 1.
* Кроме того, представлена визуализация зависимости ошибок от различных характеристик комментариев (для наглядности представлены именно ошибки классификатора).

1. Тестирование системы на основе одной метрики привлекательности.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Оценка асессора: | |
| Удалить | Оставить |
| Оценка системы: | Удалить | 73 | 27 |
| Оставить | 5 | 43 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***w*** |  |  |  |
|  |  |  |  |

Распределение ошибок классификатора, построенного на основе одной метрики привлекательности:

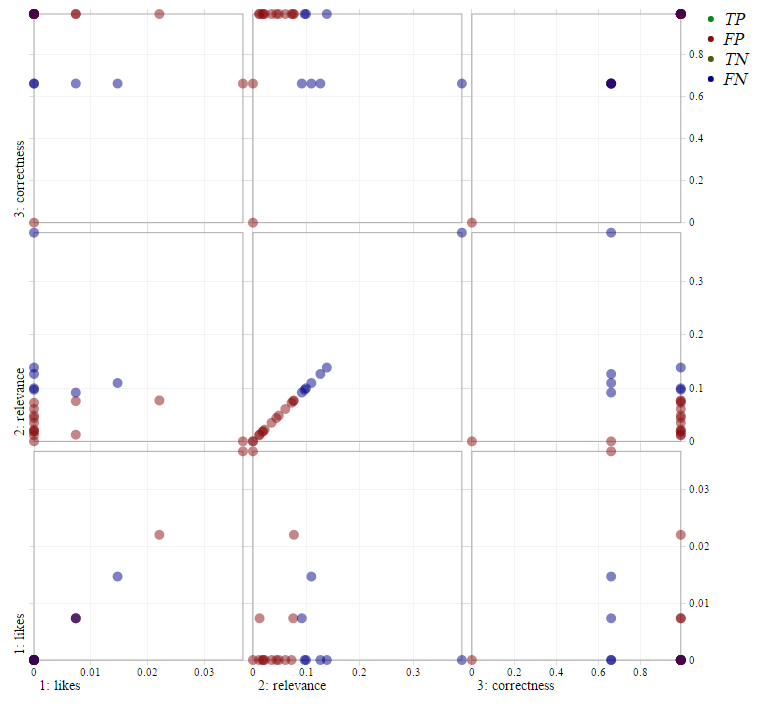


1. Тестирование системы на основе метрик привлекательности и релевантности.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Оценка асессора: | |
| Удалить | Оставить |
| Оценка системы: | Удалить | 73 | 14 |
| Оставить | 7 | 56 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***w*** |  |  |  |
| (0.94742, -6.38366, -10.03590) |  |  |  |

Распределение ошибок классификатора, построенного на основе метрик привлекательности и релевантности:

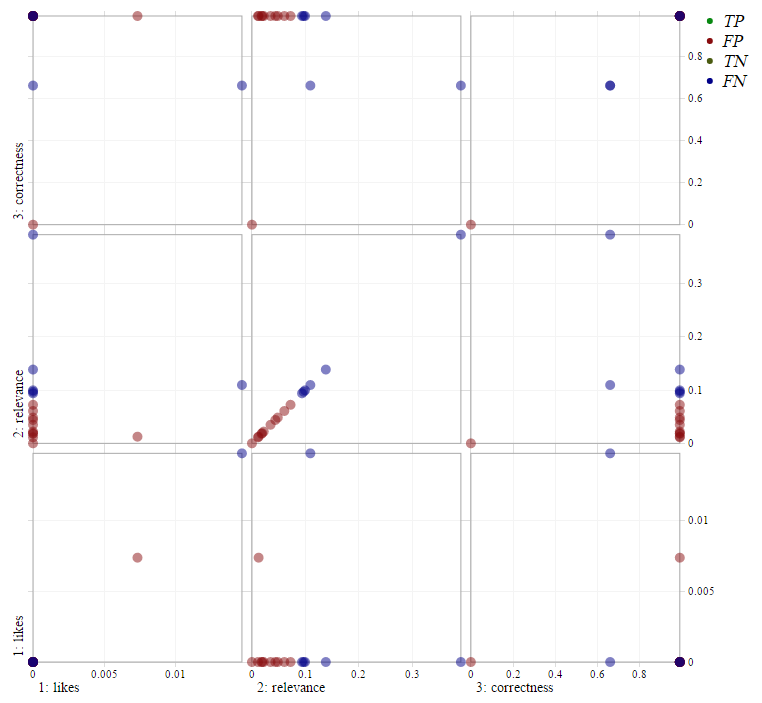


1. Тестирование системы на основе всех метрк: привлекательности, корректности и релевантности.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Оценка асессора: | |
| Удалить | Оставить |
| Оценка системы: | Удалить | 72 | 11 |
| Оставить | 6 | 59 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***w*** |  |  |  |
| (3.1642, -61.6405, -15.6417, -1.7404) |  |  |  |

Распределение ошибок классификатора, построенного на основе всех метрик: привлекательности, релевантности и корректности:



По итогам тестов, можно заключить, что результаты достаточно оптимистичные - по сравнению с базовой характеристикой привлекательности, удалось добиться повышения аккуратности - на 12%, полноты на 8% и точности на 14%.

# Заключение

Была проделана большая работа:

* Размечен набор данных.
* Успешно исследованы и протестированы новые характеристики качества пользовательских комментариев.
* Создан алгоритм автоматического подбора вектора весов характеристик и степени полинома логистической регрессии для классификации.
* Программный комплекс реализован и протестирован.
* Система внедрена в социальное сообщество РБК Вконтакте, где используется на сегодняшний день.

## Рекомендации к внедрению

Минимальные системные требования:

* Операционная система поддерживающая JDK версии 1.7\_10 и выше, а так же среду полнотекстового поиска ElasticSearch 2.3.
* Процессор: Pentium 4 2.0 ГГц.
* Оперативная память: 2 ГБ.
* Жёсткий диск: Не менее 1 Гб свободного места.

Для развёртывания системы необходимо следующее программное обеспечение:

* + Компилятор для языка Scala
  + Система сборки SBT
  + ElasticSearch 2.3
  + Консольный клиент Git

Установка и запуск.

1. Запустить ElasticSearch по адресу: localhost:9201
2. Открыть консоль в нужной папке и выполнить команду:

git clone --branch stage <https://github.com/kell18/CAS.git>

1. Настроить параметры используемого API и социального сообщества. Для Vk API:
   1. Настроить нужные параметры и получить OAuth ключ, запустив приложение и пройдя по ссылке: http://localhost:8080/vk-auth/
   2. Скопировать полученный ключ из адресной строки и вставить в поле token в файле /resources/cas/dealers/VkApi.json
   3. В файле /resources/cas/dealers/UsingDealer.json, в поле id указать VkApi.json
2. Опционально. Запустить unit-тесты: sbt test
3. Запустить приложение: sbt re-start
4. Остановить приложение можно командой: sbt re-stop

# 

# 

# Ссылки

1. Репозиторий проекта: <https://github.com/kell18/CAS>
2. S. Robertson and H. Zaragoza “The Probabilistic Relevance Framework: BM25 and Beyond”: <http://www.staff.city.ac.uk/~sb317/papers/foundations_bm25_review.pdf>
3. Shie Mannor, Dori Peleg, Reuven Rubinstein, “The cross entropy method for classification”: <http://www.machinelearning.org/proceedings/icml2005/papers/071_CrossEntropy_MannorEtAl.pdf>
4. Aaron Gough, “Automatic content moderation with 'validates\_text\_content'”: <http://thingsaaronmade.com/blog/validates_text_content.html>
5. Dmitry Kan, “SentiScan: блог-пост о технологии распознавания сентимента (тональной окрашенности сообщений)” : [http://mathlingvo.ru/2013/10/sentiscan-блог-пост-о-технологии-распознаван](http://mathlingvo.ru/2013/10/sentiscan-)
6. Janyce Wiebe and Ellen Riloff “Creating Subjective and Objective Sentence Classifiers from Unannotated Texts”: <https://www.cs.utah.edu/~riloff/pdfs/cicling05.pdf>
7. Nikolay Kuznetsov, “Шаблон проектирования Producer-Consumer”: <https://software.intel.com/ru-ru/articles/producer-consumer>
8. Николай Паклин, “Логистическая регрессия и ROC-анализ — математический аппарат”: <https://basegroup.ru/community/articles/logistic>

# Приложения

## №1. Алгоритм подбора коэффициентов и степени полинома логистической регрессии (Matlab/Octave).

[X, y] = prepareClassData('data/data-train\_oct.txt');

[tX, ty] = prepareClassData('data/data-test\_oct.txt');

[vX, vy] = prepareClassData('data/data-validation\_oct.txt');

ch = X(:, 2:end);

opts = optimset('GradObj', 'off', 'MaxIter', '100');

d = 0; maxD = 20; rndIters = 100;

minVal = 1500.0;

rFP = 100; rFN = 100;

thR = [];

info = 10;

dR = 1; jR = -1;

errHist = [];

while (d < maxD)

d += 1;

dX = raiseDegree(X, d);

costFn = @(th) costFunction(th, dX, y);

sz2 = 1 + size(ch)(2) \* d;

for i = 1:rndIters

th0 = (rand(sz2, 1) .- 0.5) .\* 10;

[th, fV, inf, out, grad, hess] = fminunc(costFn, th0, opts);

[j, g, FP, FN] = costFunction(th, raiseDegree(vX, d), vy);

if (FP + FN < minVal)

minVal = FP + FN;

rFP = FP;

rFN = FN;

thR = th;

info = inf;

dR = d;

jR = j;

end

end

errHist = [errHist; d, FP, FN];

end

disp('Validation min: ')

rFP, rFN, thR, info, dR, jR

errHist

disp('Testing: ')

[tj, tg, tFP, tFN] = costFunction(thR, raiseDegree(tX, dR), ty);

tFP, tFN

tj

function [J, grad, FP, FN] = costFunction(theta, X, y)

m = length(y);

J = 0;

grad = zeros(size(theta));

identity = ones(m, 1);

hX = sigmoid(X \* theta);

J = -1/m \* sum(0.9 .\* y .\* log(hX) + 1.1 .\* (identity - y) .\* log(identity - hX));

grad = 1/m \* (hX - y)' \* X;

predicted = hX >= 0.5;

errors = y - predicted;

FP = length(find(errors < 0));

FN = length(find(errors > 0));

end

function[X, y] = prepareClassData(path)

D = load(path);

X = D(:, 1:end-1);

y = D(:, end);

end

%% raiseDegree: Change degree of regression polynomial

function [dX] = raiseDegree(X, degree)

ch = X(:, 2:end);

dX = X;

for d = 2:degree

dX = [dX, (ch) .^ (1/d)]; % Mb .^ (1/d)

end

end

## №2. Пример размеченного набора данных.

{

"data": [{

"post": "Содержание поста",

"text": "Содержание комментария",

"likes": 1,

"id": "4541060\_4541186",

"date": "14/06/2016 01:51:02",

"relevance": 0.14523743,

"isToDelete": 1

}, … ]

}

1. http://nexgate.com/ [↑](#footnote-ref-2)
2. https://www.microsoft.com/en-us/moderator [↑](#footnote-ref-3)
3. http://iosquare.com/ [↑](#footnote-ref-4)
4. https://github.com/aarongough/validates\_text\_content [↑](#footnote-ref-5)
5. http://akka.io/ [↑](#footnote-ref-6)
6. https://etorreborre.github.io/specs2/ [↑](#footnote-ref-7)
7. https://github.com/typelevel/cats [↑](#footnote-ref-8)
8. http://spray.io/ [↑](#footnote-ref-9)
9. https://github.com/nscala-time/nscala-time [↑](#footnote-ref-10)
10. https://new.vk.com/dev/execute [↑](#footnote-ref-11)
11. https://new.vk.com/dev/callback\_api [↑](#footnote-ref-12)
12. https://github.com/scalaz/scalaz [↑](#footnote-ref-13)