Федеральное государственное автономное образовательное учреждение

высшего образования «Национальный исследовательский университет   
«Высшая школа экономики»

###### Факультет компьютерных наук

###### Магистерская программа Науки о данных

###### Департамент Департамент анализа данных и искусственного интеллекта

###### КУРСОВАЯ РАБОТА

На тему\_\_\_\_\_«*Оценка качества отчетов о дефектах программного обеспечения в системах отслеживания ошибок*»

Студент группы № мНоД16\_ИССА

Карпец Михаил Валерьевич

(Ф.И.О.)

Руководитель КР

доцент, к. ф.-м. н.\_Яворский Ростислав Эдуардович

(должность, звание, Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Консультант[[1]](#footnote-1)\*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, звание, Ф.И.О.)

Москва, 2017

**0. Содержание**

1. Введение 3

2. Обзор литературы 6

3. Описание данных 8

4. Методология 9

5. Результаты экспериментов 15

6. Прочие этапы работы 18

7. Выводы 20

8. Список литературы 21

**1. Введение**

Системы отслеживания ошибок являются важными элементами работы в рамках крупных проектов по разработке программного обеспечения. Системы отслеживания ошибок, или, как их часто называют, баг-трекинговые системы, это прикладные программы по работе с отчётами о дефектах ПО, а именно их обработке, хранению и отслеживанию. [4].

В данной курсовой работе используются отчёты о дефектах, сформированных в баг-трекинговой системе JIRA.

Ядром баг-трекинговой системы является база данных, в которой хранятся отчёты о дефектах ПО, или, как их часто называют, баг-репорты. Каждый баг-репорт есть набор заполненных полей. В разных баг-трекинговых системах этот набор может различаться, но можно выделить ряд ключевых полей, общих для подавляющего большинства систем:[4]

1) id дефекта

2) author’s name

3) creation time

4) current state

5) program version

6) OS version

7) severity

8) bug description (описание дефекта)

Переходя к вопросу об оценке качества отчёта о дефекте необходимо установить, данные в каких из полей отчёта о дефекте наиболее важны для программистов, занимающихся исправлением этих дефектов.

В исследовании [2] авторы устанавливают, что наиболее важная информация одновременно с этим наиболее сложна для автоматической оценки, так как она содержится в неформализованном виде в поле «описание дефекта», то есть записана на естественном человеческом языке (английском). А именно, это смысловые элементы:

1) шаги для воспроизведения дефекта;

2) ожидаемое/фактическое поведение, то есть описание поведения системы в соответствии с указанными шагами воспроизведения. При этом ожидаемым поведением обычно считается то, которое описано в спецификации.

Некачественное описание дефекта в составе отчёта о дефекте серьёзно затрудняет исправление дефекта ПО.

Так как в больших проектах отчётов о дефектах может быть очень много, некачественные описания дефектов являются серьёзной проблемой так как ограниченные и ценные ресурсы в виде трудозатрат программистов из-за этого необоснованно тратятся, даже если в итоге по некачественному описанию можно будет выявить проблему. Если же проблему не удастся выявить, трудозатраты, дополнительно понесённые программистом из-за некачественного описания, не принесут никакой отдачи, или, что ещё хуже, программист может внести изменения в систему без необходимости, так как он будет предполагать наличие несуществующей проблемы.

В данной курсовой работе основными элементами, определяющими качество отчёта о дефекте, согласно работе [2], считаются шаги для воспроизведения (steps to reproduce, STR) и ожидаемое/фактическое поведение (expected/observed behavior, EOB).

В данной курсовой работе строятся алгоритмы, позволяющие прогнозировать, содержит ли описание дефекта STR и/или EOB, и, таким образом, позволяющие оценивать качество данного отчёта о дефекте. Также оценивается качество этих алгоритмов. Лучший алгоритм в случае удовлетворительного качества работы, может быть использован в качестве одного из средств для формирования признакового описания дефекта. Признаковое описание дефекта должно поступать на вход рекомендательной системы, которая будет предлагать пользователю, создающему отчёт об ошибке, добавить в своё описание дефекта STR и/или EOB в случае, если алгоритм не обнаружит их в описании.

Исходя из метрики качества отчёта о дефектах, основанной на STR/EOB, целями работы являются:

1) Исследование возможности построения классификаторов с качеством работы лучше случайного либо константного выбора, которые должны распознавать наличие STR и/или EOB в тексте описания отчёта. Если такие классификаторы удастся построить, выбрать лучший из этих классификаторов.

2) Исследование наличия взаимосвязей в наличии в описаниях дефектов STR и/или EOB с одной стороны, и других признаков с другой стороны.

3) Проектирование системы рекомендаций для отчётов о дефектах из заданного проекта: описание прецедентов, составление диаграммы прецедентов, составление концептуальной модели.

4) Реализация элементов системы рекомендаций для отчётов о дефектах.

Дальнейшее описание посвящено, в основном, реализации первой цели из поставленных выше, поскольку курсовой проект командный, и автор был сконцентрирован на выполнении этой задачи.

**2. Обзор литературы**

Существует большое количество работ, посвящённых проблематике оценки качества отчётов о дефектах программного обеспечения.

В статье [2] была исследована относительная важность смысловых элементов отчётов. Это важно, потому что между пожеланиями программистов о содержании отчётов о дефектах, и между тем, что представляется важным пользователям при формировании отчётов об ошибках, возможно, существует разница. Необходимость исследования обусловлена тем, что, не зная, что нужно в отчётах программистам, невозможно проинформировать пользователей, составляющих отчёты, о наиболее важных смысловых элементах, которые они должны указать в своём отчёте, чтобы облегчить задачу исправления дефекта программистам. В качестве выборки были использованы результаты опроса, проведённого среди 466 программистов и пользователей, составляющих отчёты. Результаты исследования показали, что между пожеланиями программистов и представлениями пользователей существует различие. Также, что важно для этой курсовой работы, было установлено, что наиболее важными смысловыми элементами в отчётах об ошибках являются steps to reproduce и expected/observed behavior.

В статье [3] были исследованы способы автоматического различения того факта, является ли объект баг-трекинговой системы отчётом о дефекте или запросом на улучшение. Это нужно для того, чтобы в автоматическом режиме распределять объекты баг-трекинговой системы по программистам соответствующих профилей, а также, чтобы более корректно анализировать статистику по объектам систем отслеживания ошибок. В качестве исследуемой выборки были взяты 1800 объектов из проектов Mozilla, Eclipse и JBoss, которые были самостоятельно размечены авторами статьи. Для решения поставленной задачи авторы использовали техники машинного обучения, а именно: Decision Tree, NB Classifier, Log Regression. Результатом данной работы явилось обучение распознающего алгоритма с показателем точности ответов около 80%. Для данной курсовой работы это важно, так как авторы статьи использовали производственную линию работы с текстами описаний объектов систем отслеживания ошибок, использованную в нашем курсовом проекте.

В статье [1] была предложена модель оценки качества отчёта о дефекте, заключающаяся в отождествлении качества отчёта и длины временного отрезка между событиями «создание отчёта о дефекте» и «распределение (triaging) дефекта разработчику». Авторы исследовали возможность прогнозирования попадания или непопадания этого временного отрезка в днях в различные сроки – 25, 30 и т.д. дней. Для решения этой задачи авторы использовали линейную модель, которая показала результат лучше, чем случайный выбор. Также авторы использовали полученную систему классификации как фильтр, отсеивающий отчёты со временем распределения выше заданного порога. При определённых допущениях экономический эффект от экономии трудозатрат программистов-распределителей превышает ущерб от пропуска отчётов о серьёзных дефектах. Эта статья важна для данной курсовой работы, поскольку во-первых даёт альтернативный подход к определению качества отчёта об ошибке, во-вторых же даёт ряд идей, использованных в данной курсовой работе, а именно, использование количества комментариев к отчёту и метрик читабельности в качестве признаков на отчётах о дефектах.

**3. Описание данных**

Отчёты о дефектах, которые размечаются на «имеющие STR и/или EOB», и на которых в дальнейших этапах работы обучаются алгоритмы классификации, суть объекты, обладающие некоторыми атрибутами. В данной работе под словосочетанием «отчёт о дефекте» здесь и в дальнейшем будет пониматься объект, определяемый следующими атрибутами: описание дефекта (на естественном языке), количество комментариев к отчёту о дефекте, определяемое по тексту поля Attachment наличие ссылок на изображения в отчёте о дефекте, определяемое по тексту поля Attachment наличие ссылок на файлы xml, определяемое по тексту поля Attachment наличие ссылок на файлы с расширениями log и/или zip. Также будут использоваться атрибуты, производные от вышеперечисленных (например «наличие стек-трейса», определяемое по описанию дефекта).

Итак, данные, на которых строится алгоритм по распознаванию STR и/или EOB, были взяты следующие:

1) 472 отчёта о дефектах из проекта «JBoss Network-Support (CLOSED)»

2) 985 отчётов о дефектах из проекта «WildFly Core (WFCORE)»

3) 773 отчёта о дефектах из проекта «FUSE Message Broker /MB-1291»

Все три проекта взяты отсюда: <https://issues.jboss.org/browse/MB-1294?jql=project%20%3D%20MB>

Описания дефектов в каждом отчёте о дефекте были размечены экспертом на предмет содержания смысловых элементов STR и/или EOB.

Из 2122 описаний дефектов во всех трёх проектах только у 189 (примерно 9%) есть STR и у 97 (примерно 5%) есть EOB.

**4. Методология**

**4.1.** Наиболее информативным атрибутом отчётов о дефектах (как следует из статьи [2]) является атрибут «описание дефекта». Далее слово «текст» будет использоваться как синоним значения атрибута «описание дефекта» у конкретного отчёта о дефекте.

При работе с описаниями дефекта в случае баг-трекинговых систем, возникает проблема большой зашумлённости неинформативным текстом. В текстах, помимо традиционного шума – знаков препинания, цифр, аббревиатур, стоп-слов - могут встретиться такие сложные для автоматического распознавания виды шума как:

1) Stack Trace

2) Java code

3) Linux commands

4) SQL queries

Чтобы удалить шум, были созданы соответствующие регулярные выражения в количестве 98 экземпляров. Ниже приведён пример регулярного выражения для обнаружения и удаления одного из видов стек-трейса, встречающегося в текстах из отобранных для выборки отчётов о дефектах трёх проектов.

pat = re.compile('.\*[Cc]aused by.\*java.\*\\n(.\*[Cc]aused by.\*java.\*\\n)+')

**4.2.** После удаления шума необходимо привести слова в разных формах к одной единице, объединяющей набор морфологически похожих слов. Известны процедуры, выполняющие эту задачу - стемминг и лемматизация.

Вначале мы создаём два корпусов текстов (из слов после стемминга, и из слов после лемматизации). Далее каждый из этих корпусов «раздвоится» за счёт корпусов биграмм - пар соседних слов, считающихся одной семантической единицей.

В результаты было получено 4 корпуса, на каждом из которых будет производится обучение и оценка качества работы отобранных в последующих этапах работы классификаторов. Будут выбраны лучшие комбинации «корпус-классификатор».

Ниже мы описываем стемминг и лемматизацию.

В лингвистике словом «стемминг» называют сокращение переменной части идентичных слов в разных формах до их основы, то есть стема. Он не обязательно должен совпадать с морфологическим корнем слова. [5]

Мы использовали SnowballStemmer из библиотеки Python NLTK.

В лингвистике лемматизацией называют процесс определения словарной формы слова исходя из его значения. В отличие от стемминга, лемматизация зависит от контекста, в котором находится слово. Проблема лемматизации - открытая область исследований.[6]

Мы использовали WordNetLemmatizer из библиотеки Python NLTK.

**4.3.** По итогам пунктов 4.1 и 4.2 мы получили 4 корпуса. Они моделируются техникой «мешок слов» [7]. В результате обработки корпусов получились следующие словари для их описания: 3613 токенов после стемминга, 5082 токенов после лемматизации, 38569 биграмм после стемминга, 41496 биграмм после лемматизации. Чтобы индексировать слова в корпусах, мы использовали индексы tf-idf [8] .

После этой процедуры имеем векторизацию каждого текста из каждого корпуса:

di = ( tf-idf(D,di,t1), tf-idf(D,di,t2) … tf-idf(D,di,tn) ) (1)

где di – i-й текст, D – набор текстов, tk – k-й терм, n – мощность множества уникальных термов в корпусе.

Таким образом каждый корпус имеет образ в множестве вещественных матриц объекты-признаки (объекты – тексты, признаки - термы):

||gik||m×n (2)

где gik есть tf-idf(D,di,tk) (см. (1)), m – число текстов в корпусе, n – мощность множества уникальных термов в корпусе.

Каждая строка матрицы (2) это вещественный вектор (1), репрезентирующий описание определённого дефекта и каждый столбец матрицы (2) представляет собой значения индексов tfidf определённого терма в каждом описании дефекта.

В результате пунктов 4.1, 4.2, 4.3 мы получили четыре репрезентации каждого корпуса из 2122 описаний дефектов в виде четырех матриц типа (2):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Матрицы-репрезентации корпусов*** | **№1** | **№2** | **№3** | **№4** |
| **Размер** | 2122 x 3613 | 2122 x 5082 | 2122 x 38569 | 2122 x 41496 |
| **Токены** | да | да |  |  |
| **Биграммы** |  |  | да | да |
| **После стемминга** | да |  | да |  |
| **После лемматизации** |  | да |  | да |

**Таблица 0. Репрезентация различных вариантов корпусов матрицами типа (2)**

Эти репрезентации описаний 2122-х дефектов, а также два вектора-столбца целевых значений (есть/нет STR и/или EOB) будут использоваться в качестве данных в следующих этапах курсового проекта.

**4.4** Отбор признаков.

Следующий этап курсового проекта – снижение размерности признакового пространства. Это делается для улучшения точности работы классификаторов, а также снижения влияния шума. Были использованы следующие методы: bi-normal separation, mutual information classification , information gain.

Опишем метод BNS.

BNS: Bi-Normal Separation это метрика для отбора признаков, определяемая как F\_1(tpr) – F\_1(fpr), где F\_1 это Standard Normal distribution’s inverse cumulative probability function (z-score); tp: true positives = число положительных случаев, содержащих слово; fp: ложные срабатывания = число отрицательных случаев, содержащих слово; tpr: sample true positive rate = tp / pos, fpr: sample false positive rate = fp / neg; pos – количество объектов положительного класса; neg – количество объектов отрицательного класса. [9]

Пусть появление данного признака (терма) в каждом документе моделируется событием: случайно распределённая нормальная величина превышает некоторый порог. Показатель распространенности признака соответствует площади под кривой за порогом. Если этот признак более распространен в положительном классе, то порог дальше от хвоста распределения, чем от отрицательного класса. Метрика BNS измеряет разделение между этими двумя порогами.

Опишем метод information gain (IG). Сначала нужно рассчитать показатель:

Ĥ(X) = -

где X – признак, xi X – значение признака, p(xi) – частота признака.

Далее нужно определить Ĥ( Y | X=xi )

И показатель: Ĥ( Y | X ) = - \* Ĥ( Y | X=xi )

Показатель information gain равен:

IG (Y|X) = Ĥ(Y) – Ĥ( Y | X )

[10]

Mutual Information (MI) между двумя случайными величинами является неотрицательным числом, которое измеряет зависимость между случайными величинами Показатель равен нулю тогда и только тогда, когда две случайные величины независимы, а большие значения означают большую зависимость.[11]

Функция основана на непараметрических методах, основанных на оценке энтропии k-ближайших соседей [11] .

Вышеописанные методы снижения признакового пространства в дальнейшем послужат для отбора оптимальных подмножеств термов, описывающих тексты.

**4.5** Опишем потенциальные преимущества методов машинного обучения, использованных в работе.

**4.5.1.** Мы используем логистическую регрессию. Формула логистической регрессии:

f(z) = 1 / (1 + exp{-z})

где z – скалярное произведение векторного описания классифицируемого объекта на вектор параметров модели.

Во-первых, абсолютная величина модулей коэффициентов логистической регрессии покажет важность соответствующих термов для классификации (и знаки коэффициентов будут указывать на класс классификации). Во-вторых, в нашем распоряжении будет не только ответ классификатора на каждом объекте, но и оценка апостериорной вероятности класса на каждом объекте. Это основное различие между логистической регрессией и другими линейными классификационными моделями [12]

В этой работе была использована реализация метода Python из библиотеки scikit-learn 0.18.1.

**4.5.2.** Мы используем метод опорных векторов. Суть метода заключается в построении оптимальной разделяющей гиперплоскости между объектами двух разных классов. Потенциально полезным побочным результатом метода является получение набора опорных векторов, то есть объектов, находящихся вблизи границы классов. Возможно, если метод показывает хорошие результаты, можно будет сделать некоторые выводы из статистики по объектам вблизи границы классов. [13]

В этой работе мы использовали реализацию метода Python из библиотеки scikit-learn 0.18.1.

**4.5.3.** Мы также используем метод Random Forest. Результаты работы этого метода не интерпретируемы, но он показывает хорошие результаты [14] .

В этой работе мы использовали реализацию случайного леса в Python из библиотеки scikit-learn 0.18.1.

**4.5.4.** Также в работе используется метод Gaussian naive Bayes.

Он основан на использовании формулы Байеса.

P(A | B) = P(B | A)\*P(A) / P(B)

где P(A) – априорная вероятность события A, P(A | B) – апостериорная вероятность события A.

Преимущество Naive Bayes заключается в том, что для оценки параметров, необходимых для классификации, требуется лишь небольшое количество данных. В этой работе мы использовали реализацию Gaussian naive Bayes в Python из библиотеки scikit-learn 0.18.1. [15]

**5. Результаты экспериментов.**

После получения- в п. 4.3 четырёх матриц объекты-признаки на токенах и биграммах после стемминга/лемматизации, с помощью методов отбора признаков, описанных в п. 4.4, количество признаков (столбцов в каждой из матриц) было сокращено. При этом, уровни значимости для классификации были выставлены таким образом, чтобы отсечь более 90% наименее информативных признаков. Уровни значимости для разных методов были выставлены следующим образом:

BNS: 0.1 для токенов и для биграмм при стемминге и лемматизации.

Mutual information classification: 0.005 для токенов и для биграмм при стемминге и лемматизации.

Information Gain: 0.0005 для токенов при стемминге и лемматизации и 0.003 для биграмм при стемминге и лемматизации.

После этого было проведено обучение всех классификаторов из пункта 4.5. Ниже приведены результаты проверки качества работы обученных классификаторов, включая cross-validation [16] на 4-х блоках и  hold-out [16] для класса  "единица", то есть если отчёт содержит шаги для воспроизведения или ожидаемое/фактическое поведение.

После лемматизации и после стемминга результаты получились почти одинаковые, и расположились следующим образом (остальные значительно хуже):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | *Цель распознавания* | **Steps to reproduce** | | | | | |
|  | *Качество результата (по убыванию)* | **1** | | **2** | | **3** | |
|  | *Классификатор* | ***Random Forest*** | | ***Random Forest*** | | ***Gaussian Naïve Bayes*** | |
|  | *Метод отбора признаков* | ***Information Gain*** | | ***Mutual information classification*** | | ***Information Gain*** | |
|  | *Единицы текста* | ***Биграммы*** | | ***Биграммы*** | | ***Биграммы*** | |
|  | *Лемматизация/стемминг* | ***Лем.*** | ***Стем.*** | ***Лем.*** | ***Стем.*** | ***Лем.*** | ***Стем.*** |
| **4-folds cross-validation** | **Precision** | 0,76 | 0,7 | 0,66 | 0,7 | 0,68 | 0,64 |
| **Recall** | 0,48 | 0,49 | 0,47 | 0,45 | 0,31 | 0,32 |
| **F-measure** | 0,59 | 0,59 | 0,57 | 0,55 | 0,41 | 0,4 |
| **Hold-out 70%/30%** | **Precision** | 0,76 | 0,76 | 0,67 | 0,71 | 0,74 | 0,71 |
| **Recall** | 0,65 | 0,56 | 0,5 | 0,56 | 0,48 | 0,52 |
| **F-measure** | 0,7 | 0,64 | 0,57 | 0,62 | 0,58 | 0,6 |

**Таблица 1 Результаты распознавания Steps to reproduce**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | *Цель распознавания* | **Expected/observed behavior** | | | | | |
|  | *Качество результата (по убыванию)* | **1** | | **2** | | **3** | |
|  | *Классификатор* | ***Random Forest*** | | ***Random Forest*** | | ***Gaussian Naïve Bayes*** | |
|  | *Метод отбора признаков* | ***Information Gain*** | | ***Mutual information classification*** | | ***Information Gain*** | |
|  | *Единицы текста* | ***Биграммы*** | | ***Биграммы*** | | ***Биграммы*** | |
|  | *Лемматизация, стемминг* | ***Лем.*** | ***Стем.*** | ***Лем.*** | ***Стем.*** | ***Лем.*** | ***Стем.*** |
| **4-folds cross-validation** | **Precision** | 0,81 | 0,8 | 0,74 | 0,92 | 0,63 | 0,55 |
| **Recall** | 0,45 | 0,46 | 0,46 | 0,47 | 0,48 | 0,47 |
| **F-measure** | 0,55 | 0,53 | 0,55 | 0,53 | 0,52 | 0,49 |
| **Hold-out 70%/30%** | **Precision** | 0,93 | 0,93 | 0,81 | 0,76 | 0,78 | 0,69 |
| **Recall** | 0,43 | 0,43 | 0,43 | 0,43 | 0,6 | 0,6 |
| **F-measure** | 0,59 | 0,59 | 0,57 | 0,55 | 0,68 | 0,64 |

**Таблица 2 Результаты распознавания Expected/observed behavior**

При кросс-валидации на 4-х блоках на всех признаках, без отбора, максимальное значение f-measure достигается на Random Forest и равно 0.26 (0.32). При hold-out максимум f-measure 0.45.

**6. Прочие этапы работы.**

В рамках выполнения второй цели, поставленной во введении, была проведена кластеризация методом k – means. Для этого признаковое описание отчёта о дефектах было расширено следующим образом: помимо признаков STR и EOB были введены также признаки: количество комментариев к отчёту о дефекте, количество слов в описании, доля глагольных форм в описании, наличие изображений в отчёте о дефекте, наличие Stack Trace, наличие файлов xml, наличие файлов с расширениями log и/или zip. Метрики читабельности текста, используемые в начальных экспериментах с кластеризацией, были впоследствии исключены из рассмотрения из-за отсутствия информативности. В зависимости от вариаций в признаковом описании (например выбора различных метрик читабельности) силуэтный индекс колеблется вокруг значений 0,4 - 0,5 на 5-9 кластерах.

В каждом варианте кластеризации был выявлен кластер величиной порядка 70%, значения всех признаков в котором были «негативными», т.е. «нет STR», «нет EOB», «нет Stack Trace», «нет изображений» и т.д. При этом, естественное предположение об обусловленности этого слишком коротким описанием дефекта, не получает подтвержения в данных.

Также в кластере величиной порядка 20% отчётов нет ничего, кроме Stack Trace.

Для разрешения вопроса о выяснениях отличий отчётов этих групп отчётов, составляющих в общей сложности около 90% всех отчётов, от более информативных отчётов, было произведено расширение признакового описания для повторной кластеризации и изучения медианных характеристик отчётов о дефектах из вновь полученных кластеров. Новые признаки были добавлены следующим образом: из каждого кластера (обозначим его K), полученного в первоначальном варианте кластеризации были отобраны 50 слов с наибольшим tf-idf среди всех текстов данного кластера. Далее был установлен процент (в разных экспериментах от 5 до 15%), обозначим его P. Итак, новый признак, соответствующий кластеру K в первоначальном варианте, принимает значение «1» если в описании дефекта содержится процент P из 50 слов кластера K с наибольшим tf-idf и «0» в противном случае. Таким образом были сформированы новые признаки, по одному для каждого кластера из первоначального варианта кластеризации.

По результатам новой кластеризации был сделан вывод о необходимости проведения углублённого лингвистического анализа для понимания отличий «плохих» 90% отчётов от оставшихся 10%. Эта задача выходит далеко за рамки этой работы, и может явиться предметом дальнейших исследований соответствующих специалистов.

В рамках проектирования рекомендательной системы была разработана подробная схема взаимодействия системы и трёх акторов: пользователя, аналитика и администратора.

Также были реализованы некоторые элементы рекомендательной системы в виде консольного приложения под Windows.

**7. Выводы.**

1) Лучший результат был достигнут классификатором Random Forest на корпусе биграмм после лемматизации, после отбора лучших признаков методом Information Gain. f-measure для STR составила 0.59, для EOB 0.55, что лучше произвольного присвоения всем отчётам метки «ноль» (нет STR или EOB) или случайного предсказания.

2) В результаты проведённой кластеризации не удалось выявить значимых взаимосвязей между наличием в описании дефекта STR и/или EOB и значениями других признаков.

3) Было осуществлено проектирование системы рекомендаций для отчётов о дефектах из заданного проекта, а именно: было создано описание прецедентов системы, составлена диаграмма прецедентов, составлена концептуальная модель.

4) Были реализованы элементы системы рекомендаций, а именно: загрузка и предобработка проекта или отчёта, взаимодействие системы с аналитиком, проставляющим на отчётах из проекта метки STR/EOB, формирование файловой структуры для хранения информации о проекте.

**8. Список литературы**

1. P. Hooimeijer., W. Weimer. Modeling bug report quality, 2007.

2. N. Bettenburg., T. Zimmermann. What makes a good bug report?, 2008.

3. G. Antoniol., K. Ayari. Is it a bug or an enhancement? A text-based approach to classify change requests, 2008.

4. J. Corbet "Distributed bug tracking", 2008

5. J. Beth «[Development of a Stemming Algorithm](http://mt-archive.info/MT-1968-Lovins.pdf)», 1968

6. M. Johnson [“Modern Stemmer’s look"](http://tartarus.org/~martin/PorterStemmer/), 2016

7. H. Peterson “Bag of words model – new approaches”, 2013

8. J. Bowl «Tf-idf indexing», 1986

9. U. Chen “Quality of predictor’s assesment”, 1992

10. Shannon “C. E. [A mathematical theory of communication](http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6773024)”, 1948

11. A. Kraskov, H. Stogbauer and P. Grassberger, “Estimating mutual information”. Phys. Rev. E 69, 2004.

12. Andrew Ng. Stanford CS229 Lecture Notes

13. К. В. Воронцов. Лекции по методу опорных векторов

14. Breiman, Leo «Random Forests», 2001

15. Rish, Irina, «An empirical study of the naive Bayes classifier», 2001

16. Kohavi R. “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection”, 1995

1. \* указывается в случае назначения консультанта [↑](#footnote-ref-1)