Распознавание Спам-Писем Электронной Почты с Помощью Искусственной Нейронной Сети

Алёна Фёдорова, Дориан Саранчук
Департамент Программной Инженерии и Автоматики
Технический Университет Молдовы
Кишинёв, Республика Молдова
aliona.fiodorova@ati.utm.md
dorian.saranciuc@ati.utm.md

***Adstract*—This paper describes the short analysis of spam e-mail detection using artificial neural network process. The requirements for developed software product for recognizing and filtered spam were indicated on the obtained conclusions. Artificial neural network was chosen to implement effective filtering. The main concepts of the developed system and implemented algorithms are described in this work. The necessity of creating anti-spam systems was also shown.**

***Index Terms* — Spam filtering, artificial neural network, recognizing textual messages, N-grams, machine learning.**

I. Введение

На сегодняшний день все больше людей полагаются на электронную почту, чтобы связаться со своими друзьями, семьей, коллегами, клиентами и деловыми партнерами. В современных реалиях трудно представить себе человека, не пользующегося электронной почтой. По всему миру люди владеют от одного и более электронными адресами, на которые ежедневно приходят миллионы писем.

К сожалению, когда использование электронной почты стало массовым, угроза, которую оно в себе несёт, в частности спам, известный также как незапрашиваемая массовая или нежелательная почта, стала все более сложной для обнаружения проблемой, поставляемой в невероятно больших объемах. Например, согласно сообщению информационного сайта, SECURELIST [1], по итогам исследований международной компании «Лаборатория Касперского», работающей в сфере информационной безопасности, средняя доля спама в мировом почтовом трафике составила 58,02% в третьем квартале 2017 года. А средняя доля спама в русскоязычном сегменте интернета составила 61,21%, что выше общемирового показателя.

Переход по ссылкам в электронном письме спама может направить пользователей на веб-сайты фишинга или места, которые заражают вредоносным программным обеспечением.

Спам - серьезная проблема, которая потенциально угрожает дальнейшему использованию почтовых электронных служб. На сегодняшний день, выявление в электронной почте в разделе «Входящие» законных писем является нетривиальной задачей. Спаммеры демонстрируют быструю реакцию на громкие события и адаптацию мошеннических писем в соответствие с новостной повесткой. В последнем квартале 2017 года ими была оперативно использована тематика природных катаклизмов — ураганов Ирма и Харви, землетрясения в Мексике. Не была обойдена вниманием и популярная тема криптовалют: доверчивым жертвам предлагались семинары и «помощь» в гарантированно прибыльных торгах.

В связи с этим, разработка программного продукта, эффективно распознающего спам письма в электронной почте и фильтрующая их, способствующая качественному и безопасному использованию почтовых агентов, является актуальной задачей и на сегодняшний день.

II. Анализ предметной области

Как уже было выяснено, спам - это ненужные адресату электронные послания, например, рекламные письма или письма подделки с коммерческими предложениями, рассылаемые отдельными лицами по интернету или электронной почте. Чтобы предотвратить доставку этого спама используется автоматизированный инструмент называемый фильтр распознавания спама. Спам-фильтр используется для фильтрации входящей электронной почты или размещаемых где-либо ссылок. В своей работе, спам-фильтр оперирует готовыми списками с нежелательными адресатами или адресами сайтов. Соответственно, эти списки необходимо создать, а чтобы работа спам-фильтра была более эффективной, сначала его необходимо «обучить». Проблема с спама кажется актуальной, чтобы упорствовать в её устранении, но современные методы борьбы, похоже, не обеспечивают полной безопасности.

Существует несколько подходов, которые пытаются остановить или уменьшить огромное количество спама от физических лиц. Эти подходы включают законодательные меры, такие как законы о борьбе со спамом существующие по всему миру. Другие методы основаны на использовании сетевой информации и IP-адреса, чтобы определить, является ли сообщение спамом или нет.

Наиболее распространенными методами являются методы фильтрации, пытающиеся определить, является ли сообщение спамом или нет на основе содержимого и других характеристик сообщения.

Одним из самых надёжных методов фильтрации спама является метод с использованием искусственной нейронной сети и правил обучения персептрона.

Нейронная сеть — это последовательность нейронов, соединенных между собой синапсами. Структура нейронной сети пришла в мир программирования прямиком из биологии. Благодаря такой структуре, машина обретает способность анализировать и даже запоминать различную информацию. Нейронные сети также способны не только анализировать входящую информацию, но и воспроизводить ее из своей памяти. Другими словами, нейро-сеть - это машинная интерпретация мозга человека, в котором находятся миллионы нейронов, передающих информацию в виде электрических импульсов.

Нейронные сети используются для решения сложных задач, которые требуют аналитических вычислений подобных тем, что делает человеческий мозг. Самыми распространенными применениями нейронных сетей является: классификация — распределение данных по параметрам; предсказание — возможность предсказывать следующий шаг; распознавание — в настоящее время, самое широкое применение нейронных сетей.

Современные спам-фильтры применяют нейронные сети для решения минимум нескольких из вышеперечисленных задач.

III. Цели создания системы

Основной целью создания системы является защита почтовых пользователей от нежеланной корреспонденции.

Система для распознавания спам-писем электронной почты при помощи искусственной нейронной сети представляет собой оригинальную систему, совмещающую в себе наиболее важные качества её аналогов, при этом она не содержит избыточного функционала. Таким образом, описываемая система охватывает всю область применения существующих аналогов, при этом упрощая их применение.

Преимуществами данной системы являются:

- простота в использовании;
- универсальный подход ко всем почтовым агентам;
- наличие только необходимого функционала;
- надежный уровень безопасности;
- исключение вероятности утечки внутренних данных частных лиц/компании;
- проста в настройке конфигураций;
- низкая стоимость;
- использование актуального и наиболее эффективного метода фильтрации.

Также данная система разрабатывалась сразу как универсальный инструмент, который не будет жестко привязан к каким-то конкретным характеристикам и к числу этих характеристик. В результате должен получится инструмент, умеющий классифицировать текст по произвольным характеристикам.

Спам-фильтр реализуется с целью сокращения различных убытков, таких как потерянное время на просмотр и удаление нежеланной почты, потеря данных и нарушение безопасности, при нанесении вреда компьютеру пользователя от вредоносных писем и многие другие.

Также, эта система используется для освобождения памяти на почтовом сервере клиента посредством фильтрации и дальнейшего удаления спама.

IV. Техническое задание

Система для распознавания спам-писем электронной почты при помощи искусственной нейронной сети должна состоять из следующих узлов: синтаксический анализатор текста, статический анализатор текста, искусственная нейронная сеть, встроенная база данных и сервисы.

Искусственная нейронная сеть необходима для обнаружения и распознавания спам-сообщений. Копируя модель поведения человека, выполняющего то же действие, нейронная сеть обрабатывает большое количество входных предварительно выявленных статических параметров, таких как:

- удельное число слов с подозрением на спам в сообщении;
- удельное число словосочетаний и фраз с подозрением на спам в сообщении.

Также нестатистических входных параметров:

- семантические признаки;
- направленность текса;
- морфологические признаки – правильность построения предложения и установления связей между частями речи (формализм Бекуса-Наура);
- орфографические признаки – правильность написания слова, наличие замены сходных по написанию символов (например, замена буквы «О» цифрой «0» для обмана сигнатурной (шаблонной) фильтрации).

Поскольку нейронная сеть оперирует численными значениями, необходимо сформировать из вышеописанных признаков числовой входной вектор значений.

Для получения статистических признаков используется специальный словарь, содержащий в себе слова, наиболее характерные для спама. В исходном сообщении производится поиск и подсчет слов, которые совпадают с содержимым данного словаря. Для улучшения точности принятия решений дополнительно производится подсчет наиболее часто употребляемых в спаме словосочетаний.

Анализ статистических признаков нейронной сетью напоминает байесовскую фильтрацию спама, где для каждого слова или словосочетания можно установить коэффициент «спамности». Однако, в отличие от байесова фильтра, здесь коэффициенты - синаптические связи (веса) между нейронами, способные динамически изменяться в

процессе обучения, что позволяет эффективно обнаруживать новый и ранее неизвестный спам за счет умения нейронной сети обобщать накопленный опыт. Исходя из этого, можно оценивать текст на принадлежность к спаму комплексно, полагаясь на множество разнородных параметров, которые дополняют друг друга и уточняют оценку при принятии решения.

База данных используемая в этой системе ничто иное как готовые списки с ip-адресами серверов уже известных спамеров, списки электронных адресов, заголовков и тому подобной информации, необходимой для обнаружения спама, заранее созданной и часто обновляемой.

Сервисы подразумевают набор веб-сервисов, обеспечивающих функционал системы, таким образом, чтобы была возможность использовать отдельные компоненты в других системах.

V. Компоненты разрабатываемого программного продукта UML

На рис. 5.1 представлена диаграмма развертывания программного продукта по распознаванию спам-писем электронной почты при помощи искусственной нейронной сети. На данной диаграмме наглядно представлены узлы и компоненты, участвующие в работе системы.

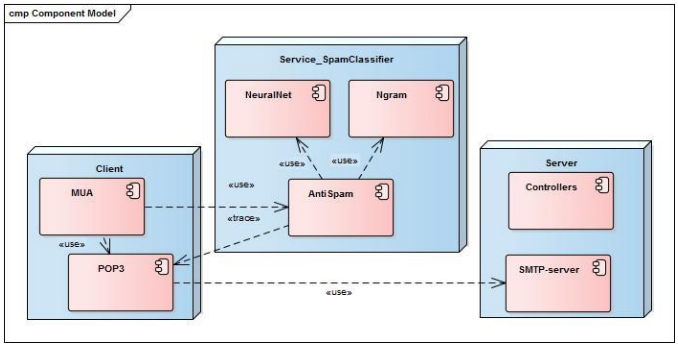

Рис. 5.1. Компоненты программного продукта по распознаванию спам-сообщений

Узел Client

Узел клиентской части системы содержит в себе пользовательский агент, или клиентскую почтовую программу и почтовые серверы для отправки и приема сообщений по электронной почте. POP3 - сервер входящей почты клиента, через него пользователь получает сообщения на свой e-mail адрес.

Узел Services

Узел сервисов объединяет сервисы, используемые системой. Как видно из диаграммы, в узле присутствует отдельный сервис AntiSpam для произведения фильтрации спама. Этот узел может содержать дополнительные сервисы для фильтрации при расширении функционала программы при использовании масштабируемости создаваемой распределённой системы. Расположение всех сервисов на одном узле не является обязательным.

Также AntiSpam сервис обращается к компонентам NeuralNet и Ngram для преобразования входной информации и дальнейшей её фильтрации.

Узел Server

Узел веб сервера содержит в себе такие компоненты, как сам SMTP-сервер для отправки сообщений, контроллеры, обеспечивающие взаимодействие между страницами, являющимися представлениями, а также моделями данных. Кроме этого, узел содержит JavaScript скрипты и файл конфигурации.

VI. Классы компонента Ngram

Компонент Ngram – это набор классов, выделяющих и преобразующих слова из входных данных, для составления обучающего словаря для нейронной сети. Наиболее популярными являются униграммы и биграммы. Бывают так же символьные N-граммы – это когда текст дробится не на отдельные слова, а на отрезки символов определенный длины.

На рис. 6.1 представлены такие классы как: Unigram, Bigram, FiltredUnigram и FiltredBigram, используемые для формирования словаря различными алгоритмами.

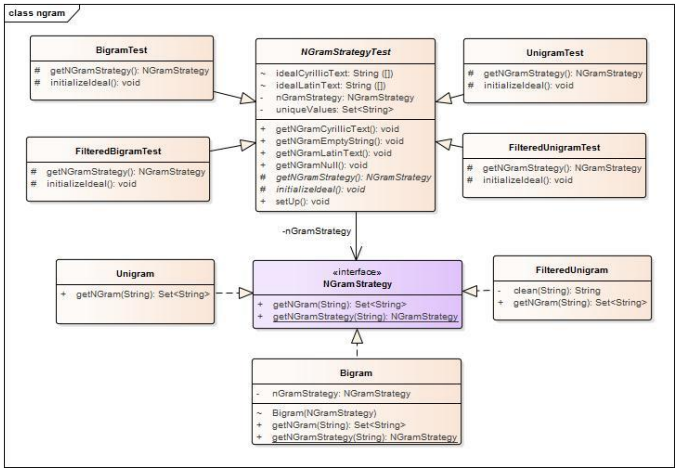

Рис. 6.1. Классы компонента Ngram

Также на диаграмме выше представлены классы, выполняющие тесты N-грамм, для определения более продуктивного варианта: BigramTest, FiltredBigramTest, UnigramTest и FiltredUnigramTest.

VII. Реализация проекта

Проект реализован на Java. В качестве СУБД использован SQLite и H2.

Также использовался фреймворк Hibernate. Для чтения Excel-файла с данными для обучения понадобились некоторые библиотеки Apache POI. Для тестов использовались JUnit 4 + Mockito.

Так как размер входного слоя должен быть равен размеру словаря, содержащего слова/фразы, из которых состоят тексты, первым шагом реализации стало создание этого самого словаря.

Один из способов создания словаря это «выдергивание» слов/фраз (даже точнее сказать, фрагментов) из текста при помощи построения N-грамм. Наиболее популярными являются униграммы и биграммы. Трудно заранее сказать, какая из N-грамм

окажется эффективнее в конкретной задаче, поэтому была необходимость провести сравнение и экспериментировать.

Двигаясь от простого к сложному, был разработан класс Unigram (листинг 5.1) для начала.

```
class Unigram implements NGramStrategy {
  @Override
  public Set<String> getNGram(String text) {
    if (text == null) {
      text = "";
    }
    // get all words and digits
    String[] words =
text.toLowerCase().split("[ \\pP\n\t\r$+<>№=]");

    Set<String> uniqueValues = new
LinkedHashSet<>(Arrays.asList(words));
    uniqueValues.removeIf(s -> s.equals(""));

    return uniqueValues;
  }
}
```

Листинг 5.1 – Класс Unigram

В результате, после обработки ~2000 текстов, был получен словарь размером ~32000 элементов. Проанализировав по диагонали полученный словарь, стало ясно, что в нем очень много лишнего, от чего надо бы избавиться.

Для этого было сделано следующее:

- Были убраны все небуквенные символы (числа, знаки препинания, арифметические операции и т.д.), так как никакой смысловой нагрузки они, как правило, не несут.
- Слова были пропущены через процедуру Стемминга (использовался Стеммер Портера для русского языка). Кстати, полезным побочным эффектом этой процедуры является «унисексация» текстов, то есть «сделал» и «сделала» будут преобразованы в «сдела».
- Для определения и исправления опечаток и грамматических ошибок было решено, что если элемент N-граммы встречается меньше чем в 4 текстах из обучающей выборки, то этот элемент в словарь не включает как бесполезный в будущем. Таким образом, мы избавились от большинства опечаток, слов с грамматическими ошибками, «слепленныхСлов» и просто очень редких слов.

Фильтрация от небуквальных символов и удаление цифр и знаков препинания, а также окончания слов, производится в классе FilteredUnigram (листинг 5.2).

```
public class FilteredUnigram implements
NGramStrategy {
  @Override
  public Set<String> getNGram(String text) {
    String[] words = clean(text).split("[
\n\t\r$+<>№=]");
    for (int i = 0; i < words.length; i++) {
      words[i] =
PorterStemmer.doStem(words[i]);
    }
    Set<String> uniqueValues = new
LinkedHashSet<>(Arrays.asList(words));
    uniqueValues.removeIf(s -> s.equals(""));
    return uniqueValues;
  }
  private String clean(String text) {
    if (text != null) {
      return
text.toLowerCase().replaceAll("[\\pP\\d]", " ");
    } else {
      return "";
    }
  }
}
```

Листинг 5.2 – Класс FilteredUnigram

Подсчёт частоты использования каждого слова (преобразуется в N-грамм) из всех классифицируемых текстов, а также конвертация уникальных слов в словарь, исключая нечастые используемые, производится в классе VocabularyBuilder (листинг 5.3).

```
class VocabularyBuilder {
  private final NGramStrategy nGramStrategy;

  VocabularyBuilder(NGramStrategy
nGramStrategy) {
    if (nGramStrategy == null) {
      throw new IllegalArgumentException();
    }
    this.nGramStrategy = nGramStrategy;
  }
  List<VocabularyWord>
getVocabulary(List<ClassifiableText>
classifiableTexts) {
    if (classifiableTexts == null ||
        classifiableTexts.size() == 0) {
      throw new IllegalArgumentException();
    }
    Map<String, Integer> uniqueValues = new
HashMap<>();
    List<VocabularyWord> vocabulary = new
ArrayList<>();
      for (ClassifiableText
classifiableText : classifiableTexts) {
      for (String word :
nGramStrategy.getNGram(classifiableText.getText()))
{
        if (uniqueValues.containsKey(word)) {
          uniqueValues.put(word,
uniqueValues.get(word) + 1);
        } else {
          uniqueValues.put(word, 1);
        }
      }
    }

    for (Map.Entry<String, Integer> entry :
uniqueValues.entrySet()) {
      if (entry.getValue() > 3) {
        vocabulary.add(new
VocabularyWord(entry.getKey()));
      }
    }
   return vocabulary;
```

Листинг 5.3 – Класс VocabularyBuilder

Для построения биграмм тоже реализован класс Bigram, чтобы в итоге после экспериментов остановить выбор на варианте, дающем лучший результат по соотношению размера словаря и точности классификации.

Был найден размер входного слоя нейронной сети, который будет равняться количеству элементов в словаре.

Размер выходного слоя для поставленной задачи будет равным количеству возможных значений для характеристики. Например, имеем 2 возможных значения для характеристики «Тип сообщения»: полезное сообщение, спам-сообщение. Тогда количество нейронов выходного слоя будет равно двум. При обучении сети для каждого значения характеристики нужно заранее определить эталонный уникальный набор чисел, который мы ожидаем получить на выходном слое.

Что касается количества и размерности скрытых слоев, то тут каких-то конкретных правил нет. Считается, что оптимальный размер под каждую конкретную задачу можно вычислить только экспериментальным путем, но для сужающейся сети рекомендуется начинать с одного скрытого слоя, размер которого колеблется в интервале между размерами входного слоя и выходного. Для начала был создан один скрытый слой размером 2/3 от входного слоя.

Для конвертации текста в цифры, пригодные для подачи на вход нейронной сети, необходимо преобразовать текст в вектор текста. Предварительно нужно каждому слову в словаре присвоить уникальный вектор слова, размер которого должен быть равен размеру словаря. После обучения сети менять векторы для слов нельзя.

Процедура преобразования текста в вектор текста подразумевает сложение векторов используемых в тексте слов. Этот вектор уже можно подавать на вход нейронной сети: каждое отдельное число на каждый отдельный нейрон входного слоя (число нейронов как раз равно размеру вектора текста).

Метод, вычисляющий вектор текста getTextAsVectorOfWords:

```
        private double[]
getTextAsVectorOfWords(ClassifiableText
classifiableText) {
          double[] vector = new
double[inputLayerSize];
          // convert text to nGram
          Set<String> uniqueValues =
nGramStrategy.getNGram(classifiableText.getText());
          // create vector
            for (String word : uniqueValues) {
            VocabularyWord vw =
findWordInVocabulary(word);
            if (vw != null) { // word found in
vocabulary
              vector[vw.getId() - 1] = 1;
            } }
          return vector;
        }
```

При первом запуске программа запрашивает XLSX-файл с обучающими данными для нейронной сети.

Файл может состоять из одного или из двух листов. На первом листе данные для обучения, на втором – данные для последующего тестирования точности сети. Структура листов совпадает. В первой колонке анализируемый текст. В последующих колонках (их может быть сколько угодно) содержатся значения характеристик текста. В первой строке содержатся названия этих характеристик.

На основании этого файла строится словарь, определяется перечень характеристик и уникальные значения, допустимые для каждой характеристики. Все это сохраняется в хранилище. Создается отдельная нейронная сеть для каждой характеристики. Все созданные нейронные сети обучаются и сохраняются. Полученный текст обрабатывается независимо каждой нейронной сетью, и выдается общий результат в виде значения для каждой характеристики.

После того как был выбран XLSX-файл с обучающими данными в предложенном файловом менеджере, программа подгружает все данные, преобразует их в вектор и подаёт в нейронную сеть для дальнейшего её обучения.

После обучения нейронной сети программа готова к фильтрации и классификации текста. Для наглядного примера работы обученной нейронной сети был реализован простой интерфейс. При вводе сообщения из раздела «не спам» данных для обучения, нейронная сеть определяет сообщение как не вредное и помещает в папку «входящие» (рис. 7.1).

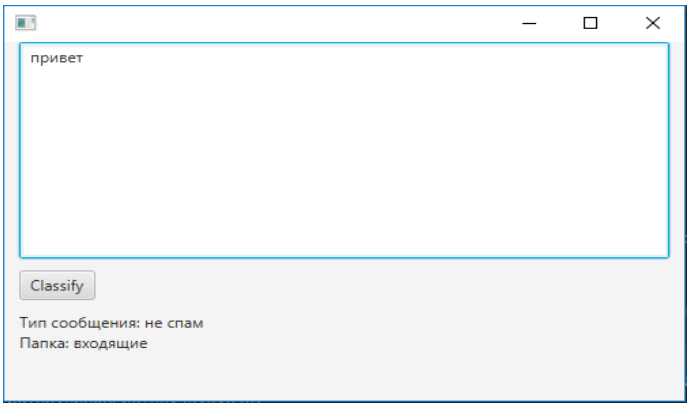

Рис. 7.1. Пример классификации сообщение как «не спам»

При вводе сообщения, например, с рекламным заголовком, нейронная сеть определяет его как спамовое и помещает в папку «спам» (рис. 7.2).

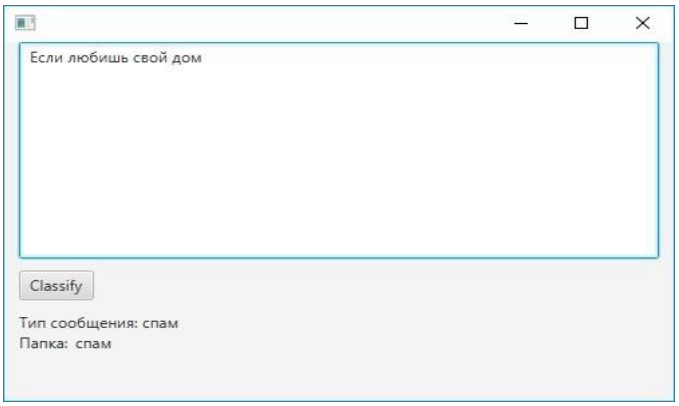

Рис. 3.5. Пример классификации сообщения как «спам».

VIII. Заключение

Информационная система по распознаванию спам-сообщений электронной почты главной целью имеет обеспечение безопасности пользовательского ПО и конфиденциальных данных, а также обеспечение комфорта и удобства при использовании почтовых клиентов. Система автоматизирует процессы, которые занимают значительную часть времени у пользователей, подвергающихся также опасности при работе со спамом.

Была проанализирована предметная область, найдены некоторые аналоги системы, а также спроектирована большая часть проекта. Разработаны диаграммы UML основных процессов, а также структура и архитектура классов основных компонентов системы.

Представлены данные о нейронных сетях, о способах их реализации и обучения, описан способ конвертации данных, поступающих на входной слой нейронной сети в понятном для неё виде.

Основной задачей выполненной работы является реализация части проекта, отвечающей за распознавание данных, их обработку, составление словаря, определение размерности входного слоя сети, конвертации текстовых данных в вектор, обучение нейронной сети для последующей фильтрации сообщений.

Основной целью использованных для распознавания данных алгоритмов является нахождение как можно большего числа угроз.

В результате работы система представляет собой один из основных компонентов – сервис для распознавания сообщений для последующей их фильтрации.

Литература

[1] Colobridge.net, Объем спама в процентах от общего трафика электронной почты, [Электронный ресурс]. – Режим доступа: https://blog.colobridge.net/2017/07/spam-volume-in-your-inbox/.

[2] Хайкин Саймон, Нейронные сети. Полный курс, Том 2: 2-е изд., испр.: Пер. с англ. – М.: ООО «И. Д. Вильямс», 2006. – 1104 с.

[3] Андрей Карпаты, Руководство хакера по нейронным сетям, [Электронный ресурс]. – Режим доступа: https://karpathy.github.io/neuralnets/.

[4] Tariq Rashid. Neural Networks and Deep Learning, Mar 31, 2016.

[5] Swsys.ru, Программные продукты и системы. Система предотвращения массовых рассылок на основе алгоритмов машинного обучения, [Электронный ресурс]. – Режим доступа: http://www.swsys.ru/index.php?page=article&id=51

[6] Скляренко Н. С., Обзор алгоритмов машинного обучения, решающих задачу обнаружения спама, [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/v/obzor-algoritmov-mashinnogo-obucheniya-reshayuschih-zadachu-obnaruzheniya-spama.

[7] Neuralnet, Нейронные сети. Учебник, [Электронный ресурс]. – Режим доступа: https://neuralnet.info/book/.

[8] Habrahabr.ru, Нейронные сети для начинающих. Часть 2, [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/313216/.