Санкт-Петербургский государственный университет

Фундаментальная информатика и информационные технологии
Профиль Информационные технологии

Шиндарев Никита Андреевич

Моделирование социоинженерных атак: синтез сцен по результатам анализа социальных сетей (проектная работа)

Бакалаврская работа

Научный руководитель:
доцент кафедры информатики, к. пс. н., доцент Тулупьева Т. В.

Рецензент:
начальник отдела информатизации и связи администрации Центрального района Санкт-Петербурга, к. т. н. Азаров А. А.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Fundamental Informatics and Information Technology
Information Technology

Nikita Shindarev

Social Engineering Attacks: situation modelling based on social networks analysis (joint project)

Bachelor's Thesis

Scientific supervisor:
Associate Prof., Computer Science Department, PhD in Psychology, Associate Prof.
Tatyana Tulupyeva

Reviewer:
Head of the IT Department of the Administration of Central District, St. Petersburg,
PhD in Computer Science Artur Azarov

Saint-Petersburg
2017

Оглавление

Введение **5**

1. Описание предметной области **9**
1.1. Социоинженерные атаки: термины и понятия 9
1.2. Задача классификации: термины и понятия 9
1.3. Описание используемых программных средств 12

2. Построение классификатора для пользовательских страниц **13**
2.1. Формализация задачи 13
2.2. Описание выбранного классификатора и алгоритма построения 13
2.3. Анализ полученных результатов 15
2.4. Выводы по главе 19

3. Описание разработанного программного решения **20**
3.1. Разработанный подход к сбору общедоступной информации 20
3.2. Сбор данных для обучения классификатора 23
3.2.1. Раздел «Карьера» 23
3.2.2. Текстовые записи официальной группы организации 25
3.2.3. Счётчик пользовательских отметок «мне нравится» в группе 27
3.2.4. Раздел «Друзья» пользовательской страницы компании 28
3.2.5. Анализ топологии сети 29
3.2.6. Целевая переменная 30
3.3. Программная реализация классификатора 30
3.4. Схема базы данных 33
3.5. Сбор страниц на основе обученного классификатора . . . 33
3.6. Описание дистрибутива программного модуля 35

4. Заключение **37**

Список литературы **39**

Введение

Актуальность темы исследования.

Информационные технологии сегодня являются неотъемлемой частью нашей жизни. Системы, хранящие и обрабатывающие информацию, используются повсеместно. Информация сегодня становится ключевым предметом интереса. Публикации в СМИ об очередном инциденте нарушения информационной безопасности стали традиционными [2]. Количество кибератак возрастает с каждым годом, они приносят всё большие убытки, требует значительно больше времени для расследования преступлений такого рода [2]. При этом большая часть исследований посвящена программно-техническим атакам, в этом срезе проблема информационной безопасности достаточно хорошо изучена, существуют разработки, снижающие вероятность успеха атак [20, 21]. В то же время пользователи информационных систем являются её самым уязвимым местом, атаки через них считаются одними из самых эффективных [1, 12, 13, 14]. Атаки, направленные не на поиск программно-технических уязвимостей, а на уязвимости пользователей информационной системы называются социоинженерными. Такие атаки основаны на использовании манипулятивных техник.

Коллектив исследователей на базе лаборатории теоретических и междисциплинарных проблем информатики Санкт-Петербургского института информатики и автоматизации Российской академии наук разработал прототип программного комплекса, имитирующий социоинженерную атаку на основе деревьев атак и позволяющий делать оценку успеха атаки на пользователя [13, 14]. В основе разработки лежит комплекс моделей: «критические документы – информационная система – персонал – злоумышленник». На основании этих моделей разработан подход для получения численных выражений оценок защищенности пользователя. В основе модели пользователя лежит профиль его уязвимостей, который представляет собой набор пар уязвимость – выраженность уязвимости. Профиль уязвимостей пользователя строится на основе его психологических особенностей, которые, в свою очередь,

выявляются через анкетирование сотрудников. В то же время известны подходы, согласно которым злоумышленник может получать информацию о компании «извне» из общедоступных источников.

Различные социальные сети являются одним из таких источников информации о пользователях. На страницах социальных сетей можно найти такую информацию как списки новостных подписок пользователя, добавленные аудиозаписи, списки друзей и контактов, личную информацию об увлечениях и интересах и т.п. Данная работа направлена на уточнение оценки успешности социоинженерной атаки на пользователя информационной системы через изменение подхода к построению профиля уязвимостей пользователя. Изменённый подход предполагает синтез используемых методик, основанных на получении информации о пользователях системы через анкетирование сотрудников, а также методик, связанных с анализом общедоступных данных, преимущественно получаемых на сайтах социальных сетей. Для сбора информации о сотрудниках через социальные сети необходимо решить задачу поиска аккаунтов сотрудников организации. Задача, решаемая в данной работе, заключается в реализации программного модуля, осуществляющего автоматизацию первичного поиска сотрудников задаваемой организации. Сбор и анализ пользовательских страниц сотрудников реализован с помощью социальной сети «ВКонтакте», которая согласно различным источникам является одной из самых распространённых на территории Российской Федерации [16, 17, 24].

Научная новизна. В данной выпускной квалификационной бакалаврской работе реализована модель, позволяющая выявлять сотрудников задаваемой организации на основе ряда признаков, наблюдаемых у пользователей социальной сети. Разработан подход, допускающий полностью автоматизированное её построение. Также, в данной работе представлены оценки эффективности данной модели.

Постановка целей и задач. Как уже было отмечено ранее, на сегодняшний день единственным разработанным подходом к составлению психологического профиля пользователя является подход на основе анкетирования сотрудников. Теоретическая цель работы состоит в выяв-

лении отличительных признаков, на основе которых можно будет сделать вывод о принадлежности конкретная страница в социальной сети сотруднику компании. Практическая цель состоит в программной реализации модуля, выявляющего страницы сотрудников компании. В перспективе внедрение такой модели в существующий программный комплекс позволит автоматизированно составлять психологический профиль пользователя, что либо дополнит существующий подход с опросами новыми данными, либо полностью его заменит.

В рамках данной работы для достижения конечной цели был поставлен ряд задач:

1. Произвести обзор представленных классификаторов, на основании которого выбрать наиболее подходящий для обозначенной цели;

2. Разработать метод сбора и классификации пользовательских страниц в социальной сети «ВКонтакте».

3. Составить модель, позволяющую получить классификатор для каждой компании без затрат на оценку значений целевого параметра.

4. Программно реализовать модуль, выявляющий страницы сотрудников в социальной сети «ВКонтакте».

5. Сохранить результаты в реляционной СУБД для организации дальнейшего доступа к страницам пользователей.

6. Внедрить программный модуль в разработанный ранее программный комплекс.

Теоретическая и практическая значимость исследования. На основе исследования был разработан новый подход, позволяющий создавать модель классификатора для конкретной компании, что позволяет строить более точную структуру для анализа пользовательских страниц без этапа предварительного «ручного» анализа конечного набора пользовательских страниц.

Структура и объём работы. Представленная работа состоит из введения, трёх глав, заключения и используемой литературы.

Глава 1 содержит общие сведения и термины по теме социоинженерных атак и задачи классификации, а также перечень используемых программных средств.

Глава 2 посвящена выбранному классификатору для анализа пользовательских страниц: приведена формализация задачи, при которой условия исходной задачи сводятся к задаче бинарной классификации, описана программная реализация классификатора на языке C#. В 4 разделе представленной главы представлен анализ полученных результатов.

Глава 3 описывает этапы разработки программного модуля: описание параметров для классификатора, реализация сбора этих параметров для пользовательских страниц, схема базы данных и её реализация. Также в данной главе содержится описание внешнего интерфейса дистрибутива внедряемого программного модуля и приведён алгоритм сбора страниц сотрудников на основе обученного классификатора.

1. Описание предметной области

1.1. Социоинженерные атаки: термины и понятия

Определение 1.1 *Социоинженерная (социотехническая) атака — набор прикладных психологических и аналитических приёмов, которые злоумышленники применяют для скрытой мотивации пользователей публичной или корпоративной сети к нарушениям устоявшихся правил и политик в области информационной безопасности [14].*

Определение 1.2 *Уязвимость пользователя — некоторая характеристика пользователя, которая делает возможным успех социоинженерного атакующего действия злоумышленника [14].*

Определение 1.3 *Профиль уязвимостей пользователя — совокупность пар «уязвимость пользователя» — «степень выраженности уязвимости» [14].*

Определение 1.4 *Граф социальных связей — граф, узлы которого представлены социальными объектами, такими как пользовательские профили с различными атрибутами (например, имя, день рождения, родной город и т.д.), сообщества, медиаконтент и т.д., а рёбра — социальными связями между ними[14].*

Определение 1.5 *Информационная система — организованно упорядоченная совокупность документов (массив документов) и информационных технологий, в том числе с использованием средств вычислительной техники и связи, реализующих информационные процессы [14].*

1.2. Задача классификации: термины и понятия

Задачи классификации встречаются в самых различных областях деятельности человека: распознавание образов, рукописного текста, фотографий, и т.д.. В контексте данной работы будет решаться задача би-

нарной классификации страниц пользователей социальной сети «ВКонтакте».

Определение 1.6 *Классификация — это системное распределение изучаемых предметов, явлений, процессов по родам, видам, типам, по каким-либо существенным признакам для удобства их исследования [25].*

Для получения оптимальных деревьев принятия решений нужно на каждом шаге выбирать атрибуты,которые наилучшим образом характеризуют целевую функцию. Это требование формализуется через понятие энтропии. В дискретном случае это приводит к следующему определению:

Определение 1.7 *Энтропия случайной величины $\hat{\xi}$ со множеством возможных исходов $\{\xi_1, \dots \xi_n\}$, определяется как [23]:*

$$H(\hat{\xi}) = -\sum_{i=1}^{n} p(\xi_i) \log_2 p(\xi_i)$$

При выборе атрибута для классификацм нужно его выбирать так, чтобы после классификации энтропия в потомках стала как можно меньше. На основе этого определяется прирост информации:

, элементы которого характеризуются свойством S. Пусть при этом элемента A классифицируются посредством атрибута Q, который имеет q возможных значений. Тогда прирост информации (informaton gain) определяется следующим образом [23]:

$$Gain(A, Q) = H(A, S) - \sum_{i=1}^{q} \frac{|A_i|}{A} H(A_i, S)$$

Главной проблемой критерия прироста информации является то, что прирост информации часто выбирает атрибуты, у которых больше всего значений:

Определение 1.8 *Для решения этой проблемы был предложен критерий Gain Ratio [25], который учитывает количество информации, требуемое для разделения по текущему атрибуту:*

$$SplitInfo(A, Q) = H - \sum_{i=1}^{q} \frac{|A_i|}{A} log_2 \frac{|A_i|}{A}$$

а сам критерий — максимизация величины

$$GainRatio(A, Q) = \frac{Gain(A, Q)}{SplitInfo(A, Q)};$$

Далее необходимо ввести ряд определений для оценки качества построенного бинарного классификатора:

Определение 1.9 *Точность (precision) — показывает, сколько из предсказанных позитивных объектов, оказались действительно позитивными. [19]*

Определение 1.10 *Полнота (recall) — показывает, сколько от общего числа реальных позитивных объектов, было предсказано, как позитивный класс. [19]*

Определение 1.11 *F-мера (F-measure) — характеристика, которая позволяет дать оценку одновременно по точности и полноте. [19]*

$$F_{measure} = \frac{1}{\alpha \frac{1}{Precision} + (1 - \alpha) \frac{1}{Recall}}, \alpha \in [0, 1]$$

где α задаёт соотношение весов точности и полноты.

Определение 1.12 *False Positive Rate (FPR) - показывает какая часть от общего числа объектов, оказались предсказанными неверно.*

$$FPR = \frac{FP}{FP + TN}$$

Определение 1.13 *В случае $\alpha = 0.5$ характеристики полноты и точности принимают одинаковый вес. Такая мера называется сбалансированной (F_1):*

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Определение 1.14 *K-fold cross validation — выборка разбивается на k частей, после чего каждая из частей по очереди используется в качестве тестовой выборки, а остальные k-1 частей — в качестве обучающей. Значение параметра k может быть произвольным.*

1.3. Описание используемых программных средств

В силу того, что разработанный программный комплекс «критичные документы — информационная система — персонал — злоумышленник» был разработан на платформе .NET с использованием языка программирования C#, было решено использовать те же язык программирования и платформу с целью возможной интеграции разрабатываемого модуля в программный комплекс. В ходе разработки программного модуля были использованы следующие инструменты:

1. среда разработки Visual Studio 2015 [8];
2. менеджер пакетов NuGet [7];
3. библиотека VkNet [9], предназначенная для взаимодействия с REST-сервисом API социальной сети «ВКонтакте»;
4. фреймворк Accord.Net [6] для построения классификатора;
5. набор библиотек Morpher.Net SDK, с помощью которого осуществляется генерация строковых шаблонов для поиска. Используется на этапе анализа параметра для пользовательской страницы;
6. РСУБД MySql для хранения результатов, а также для возможности обращаться к ним с помощью SQL-запросов;
7. библиотека KBCsv для записи csv-файлов;
8. библиотека LightInject с помощью которой реализуется паттерн проектирования ApplicationController;
9. библиотека QuickGraph для получения графа социальных связей между выявленными сотрудниками.

2. Построение классификатора для пользовательских страниц

2.1. Формализация задачи

Поставленная задача выявления страниц сотрудников может быть сведена к задаче классификации при следующей формализации:

Пусть X — множество страниц пользователей социальной сети vk.com, а Y — множество наименований классов (в данном случае $|Y| = 2$, т.к. мы имеем в результате всего 2 класса: сотрудники и не сотрудники). Существует целевая зависимость:

$$y^* : X \to Y$$

При этом значения для неё известны только на конечном числе объектов обучающей выборки:

$$X^m = \left\{ (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \right\}$$

Тогда задача сводится к построению алгоритма

$$a : X \to Y$$

способного классифицировать любой $x \in X$. Классификация при $|\mathrm{Y}| = 2$ называется бинарной. [5, 19]

2.2. Описание выбранного классификатора и алгоритма построения

Для решения задачи классификации в рамках данной работы применяется структура дерева принятия решений. Причина подобного выбора объясняется рядом преимуществ деревьев принятия решений как классификатора:

- для обучения данной структуры не требуется предварительная подготовка данных;

- качество построенной модели можно оценить при помощи статистических тестов. Тесты приведены в разделе

Подобный подход требует детального анализа пользовательской страницы с выявлением характерных признаков поведения, свойственных сотрудникам заданной компании. Признаки представлены в виде набора параметров, которые подаются на вход. Процесс принятия решения начинается в корне дерева и в зависимости от значения переменной на узле принимается решение следовать далее к левому или к правому потомку (т.е. узлу, располагающемуся на уровне ниже данного). Таким образом, для классификации множества всех страниц потенциальных сотрудников компании мы пройдём путь от корня к одному из листьев и получим значение целевого параметра для этой страницы. Более детально описание выбранных параметров, а также программная реализация алгоритмов рассчитывания этих параметров на языке программирования C# представлены в главе 3 данной работы.

Ниже представлена общая схема алгоритма построения дерева принятия решений на основе обучающей выборки [22]:

1. Выбирается очереднуй атрибут Q, помещается в корень.
2. Для всех его значений i:
   (a) Из элементов выборки остаются только те, у которых значение атрибута Q = i
   (b) Осуществляется рекурсивное построение дерева в данном потомке.

Существует несколько различных способов выбирать атрибут Q. Среди таких способов хотелось бы выделить два основных:*ID3* и *C4.5*, так как данные подходы используются для построения дерева принятия решений в фреймворке Accord.Net, который используется . Для программной реализации построения дерева принятия решений на языке C# используется фреймворк *Accord.NET*, подробное описание использования данного фреймворка есть в пункте 3.3 представленной работы.

Фреймворк *Accord.NET* предоставляет на выбор два альтернативных алгоритма для обучения дерева принятия решений:

- *ID3 (Iterative Dichotomiser 3)* — выбор атрибута осуществляется на основе Gain.
- *C4.5* — является усовершенствованной версией алгоритма *ID3*. Выбор атрибута осуществляется на основе Gain Ratio.

Выбранные для анализа пользовательского профиля параметры не всегда являются дискретными, что не позволяет использовать алгоритм ID3 в рамках решения задачи анализа пользовательских страниц. Важным преимуществом алгоритма C4.5 перед ID3 является то, что в нём добавлена возможность использования непрерывных параметров. Также, основной проблемой алгоритма ID3 является склонность к переобучению дерева. В C4.5 эта проблема решается за счет введения дополнительного показателя Split-Info, что позволяет модифицировать критерий прироста информации следующим образом:

$$SplitInfo(S) = \frac{Gain(T)}{SplitInfo(T)}$$

Обновлённый критерий считает долю полезной информации, что, как правило, позволяет выбрать более удачный атрибут, чем тот, который будет выбран на основе обычного критерия прироста. Таким образом, на основе вышеперчисленных преимуществ, для программной реализации структуры было решено использовать алгоритм C4.5.

2.3. Анализ полученных результатов

Задача выявления страниц сотрудников сводится к задачи бинарной классификации, если условиться, что полученное значение целевого параметра для страницы интерпретируется следующим образом: Y — множество значений целевого параметра, характеризующего принадлежность страницы сотруднику компании. Для $y \in Y$ справедливо:

- y = 0, если страница не принадлежит сотруднику компании;

- y = 1 в противном случае.

В данном разделе рассматриваются существующие метрики оценки эффективности бинарного классификатора. Существуют различные метрики оценки качества построенного дерева принятия решений [5]. К примеру, в задачах, связанных с медициной, зачастую используют показатели *sensitivity* и *specificity* [4]. Для задач, связанных с информационным поиском рекомендуется использовать параметры *precision* и *recall* [3, 5].

Для удобства рассчёта эффективности введём следующие обозначения:

- TP — количество верно идентифицированных страниц сотрудников в тестовой выборке, (true positive);
- FP — количество неверно выявленных пользовательских страниц в тестовой выборке, которые определены как страницы, принадлежащие сотрудникам заданной компании, (true positive);
- TN — количество верно идентифицированных страниц в тестовой выборке, не принадлежащих сотрудникам, (true negative);
- FN — количество неверно упущенных страниц сотрудников в тестовой выборке, (false negative);

Таким образом, параметры *precision* и *recall* вычисляются следующим образом:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

Для оценки качества классификатора используется ROC-кривая, которая при варьировании порога решающего правила показывает то, как зависит recall от FPR. Ниже представлена ROC-кривая для анализа эффективности полученного классификатора на примере одной Российской IT-компании с заявленной численностью штата в 1200 сотрудников:

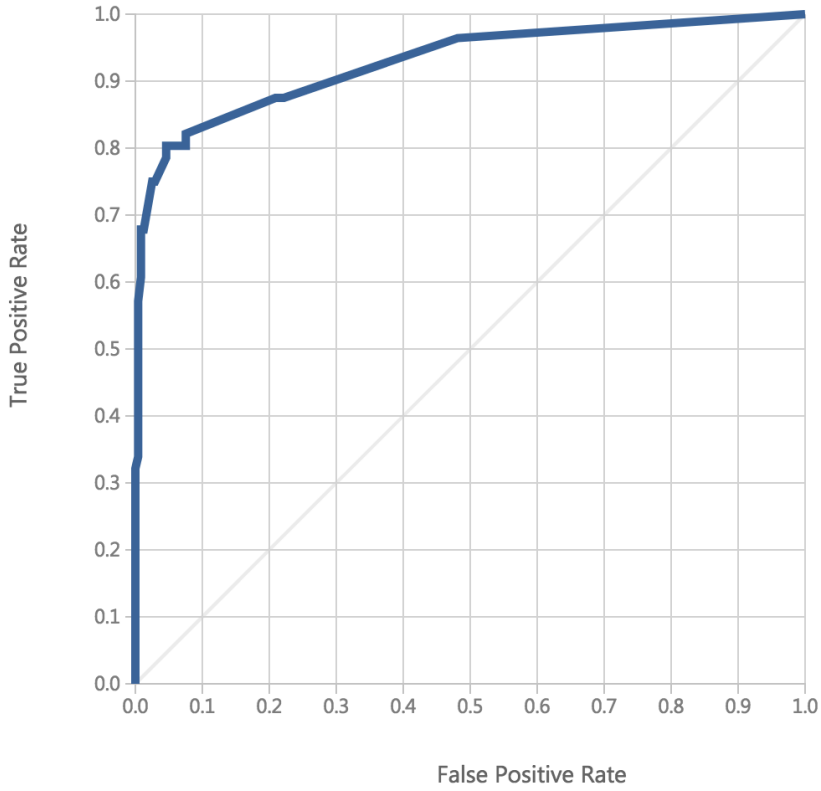

Рис. 1: Пример полученной ROC-кривой

Для избежания получения неточной оценки из-за переобучения в случае построения ROC-кривой на обучающей выборке используется меод перекрёстной проверки (K-fold cross validation), при котором выборка делится на k частей, после чего на k-1 частях дерево обучается, а последняя часть используется для тестирования. Данная операция проводится k раз, в итоге каждая из частей участвует в тестировании. В результате получается более точная оценка качества для дерева.

Для получения числовой характеристики ROC-кривой используется площадь под графиком AUC (Area Under Curve). При $AUC = \frac{1}{2}$ считается, что дерево выдает случайные значения, и, соответственно, чем ближе значение к 1, тем лучше получился классификатор. В примере на рис.1 полуившийся показатель $AUC = 0.928$, что является очень хорошим результатом.

Для анализа результирующего дерева аиболее часто используется мера F_1, которая придаёт один и тот же вес характеристикам полноты и точности. В контексте данной задачи нас в большей степени интересу-

Таблица 1: precision, recall и f-measure в зависимости от порога

Порог	Точность	Полнота	$F_1 - score$	$F_{measure}$
0.1	0.719	0.821	0.767	0.787
0.2	0.714	0.804	0.756	0.774
0.3	0.800	0.786	0.793	0.790
0.4	0.927	0.679	0.784	0.738
0.5	0.950	0.679	0.792	0.742
0.6	0.950	0.679	0.792	0.742
0.7	0.949	0.661	0.779	0.727
0.8	0.944	0.607	0.739	0.679
0.9	0.964	0.482	0.643	0.567
Наилучший показатель				
0.3	0.800	0.786	0.793	0.790

ет показатель полноты, поскольку нам нужно собрать как можно больше страниц, принадлежащих сотрудникам компании. Для этого было решено отказаться от использования сбалансированной меры в пользу F-меры со значением $\alpha = 0.3$:

$$F_{measure} = \frac{1}{0.3\frac{1}{precision} + 0.7\frac{1}{recall}}$$

.

Далее необходимо определить значение порога решающего правила, при котором показатель $F_{measure}$ будет максимальным. В рассматриваемом примере оказалось, что наилучший показатель $F_{measure} = 0.79$ достигается при значении $threshold = 0.3$, где $threshold$ — значение порога решающего правила. Более детально зависимость $F_{measure}$ и $threshold$ отображены в таблице 2.3.

Представленный в данной работе подход к построению классификатора для пользовательских страниц был протестирован на ряде Российских компаний, обладающих следующими характеристиками:

Таблица 2: Оценка реализованных классификаторов на примере различных компаний

Организация	Штат	$F_{measure}$	AUC
A	1200	0.790	0.928
B	500	0.732	0.817
C	100	0.642	0.762
D	30	0.593	0.632

- Организация A: крупная IT-организация с заявленной численностью штата ≈ 1200 сотрудников.
- Организация B: IT-компания средних размеров, штат насчитывает ≈ 500 сотрудников.
- Организация C: школа, численность сотрудников ≈ 100 человек.
- Организация D: дизайнерская команда с собственной разработкой, численность < 30 сотрудников.

В таблице .2.3 представлены полученные оценки эффективности на основе числовой характеристики Area Under Curve.

2.4. Выводы по главе

Проведённый эксперимент показал, что для различных организаций показатель эффективности построенного дерева принятия решений оказывается достаточно высоким при анализе компаний, численность штата которых $>= 100$

3. Описание разработанного программного решения

Введение

В данной главе описываются подходы ко сбору и обработки данных со страниц пользователей социальной сети «ВКонтакте».

3.1. Разработанный подход к сбору общедоступной информации

Для взаимодействия с API, который социальная сеть «ВКонтакте» поставляет для разработчиков, необходимо создать standalone приложение. С помощью данного приложения в веб-браузере осуществляется авторизация пользователя в «ВКонтакте». После прохождения эта-

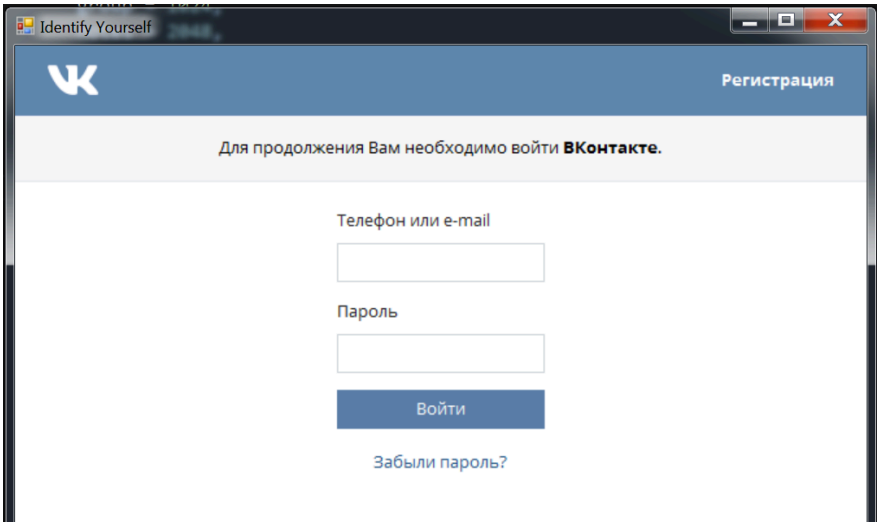

Рис. 2: Окно авторизации пользвателя

па авторизации, взаимодействие с API осуществляется по протоколу HTTP с помощью POST и GET запросов. В ответ на запрос по умолчанию сервер возвращает ответ в формате JSON, также существует возможность изменить схему ответов на XML. Для упрощения взаимодействия с API в рамках данной работы было решено использовать общедоступную библиотеку VkNet [9], которая распространяется под

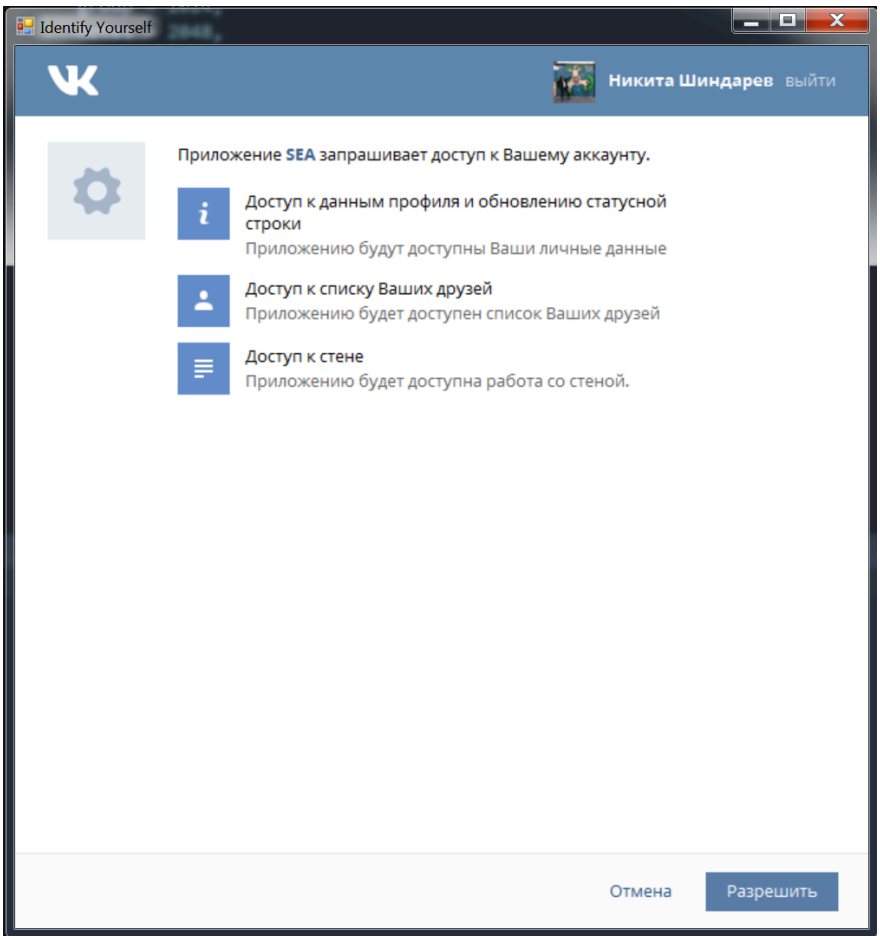

Рис. 3: Окно с запросом прав доступа приложения

лицензией MIT, что позволяет свободно использовать её в реализуемом программном модуле. На данный момент с помощью зарегистрированного приложения можно осуществлять вплоть до 5-ти запросов к API в секунду. Это ограничение было выставлено для приложений, которыми пользуются менее 10000 человек, что является приемлимым значением в рамках решения представленной задачи по сбору страниц сотрудников. В результате подключения библиотеки VkNet всё взаимодействие с API сайта «ВКонтакте» сводится к обращению к полям класса VkApiHolder, которые полученную JSON схему конвертируют в список объектов.

Было решено реализовать программный продукт с возможностью интеграции в существующий программный модуль в качестве библиотеки и при этом реализовать GUI с целью удобного анализа полученных результатов, что позволяет использовать функционал данной библио-

теки вне контекста анализа организаций на предмет уязвимостей от социоинженерных атак.

Для полной изоляции визуальной составляющей от бизнес-логики использовался паттерн проектирования Model-View-Presenter[10]. Суть данного паттерна в разделении программной логики на 3 составляющие:

- модель — отвечает за бизнес-логику приложения;
- представление — отображает на экран из модели и обращается к представителю;
- представитель — связующее звено между представлением и моделью.

Существует несколько различных путей реализации данного паттерна, главным отличием в этих методах является степень изоляции представления от модели. В рамках данной работы был выбран метод Passive View : при данном подходе представление содержит в себе только самую примитивную логику отображения данных [18].

Также было решено использовать паттерн Application Controller [18] в котором содержится IoC-контейнер [18]. Это необходимо для того, чтобы уровень представителя мог взаимодействовать с интерфейсами представлений. В качестве IoC-контейнера используется библиотека LightInject. Поэтому при запуске достаточно зарегистрировать конкретные реализации интерфейсов представления, модели и представителя, и передать управление представителю, который в рамках нашей задачи отвечает за авторизацию пользователя. Ниже представлен листинг 1 кода регистрации используемых реализаций модели, представления и представителя с привязкой к интерфейсам:

```
1 Application.EnableVisualStyles();
2 Application.SetCompatibleTextRenderingDefault(false);
3
4 var controller =
5 new ApplicationController(new LightInjectAdapter())
6             .RegisterView<IAuthorization, AuthorizationForm>()
```

```
7                 .RegisterView<IMainView, MainForm>()
8                 .RegisterView<ICompanyInfo, CompanyInfoForm_v2>()
9                 .RegisterService<IStudy, CollectingTrainingDataset>()
10                .RegisterService<IParse, MyParser>()
11                .RegisterInstance(new ApplicationContext());
12
13 controller.Run<AuthorizationFormPresenter>();
```

Листинг 1: Регистрация реализаций Model, View и Presenter

3.2. Сбор данных для обучения классификатора

В ходе исследований были исследованы различные характерные признаки, свойственные поведению сотрудников в социальной сети «ВКонтакте» и выявлены 5 основных, на основе которых обучается дерево принятия решений. Ниже подробно приведены все параметры с листингами программного кода, который осуществляет рассчет данных параметров:

3.2.1. Раздел «Карьера»

Зачастую пользователи социальной сети «ВКонтакте» в разделе «Карьера» личной информации указывают одно или несколько мест работы. Первый параметр отвечает за наличие заданной организации в списке мест работы пользователя (далее в тексте работы для этого параметра используется обозначение *hasFirmName*). С помощью запроса к API можно, вызвав метод $Users.Search$, (см. Листинг 2), можно получить полный список пользователей социальной сети «ВКонтакте», указавших заданную компанию в качестве места работы.

```
1 List<User> has_firm_name_employees =
2 VkApiHolder.Api.Users.Search(new UserSearchParams
3 {
4     Company = this.company_name,
5     Count = 1000
6 }).ToList();
```

Листинг 2: Сбор страниц с $hasFirmName = 1$

Параметр *hasFirmName* является определяющим для страниц обучающей выборки: первоначально при помощи запроса, представленного на листинге 2 получается список страниц, для которых определён параметр *hasFirmName* как равный 1. Далее, необходимо собрать страницы, для которых этот параметр составлял бы 0. Было решено в качестве таких страниц использовать «друзей» уже найденных пользователей со значением представленного параметра = 1. Для этих пользователей необходимо детально проверить карьерный список на предмет наличия названия текущей фирмы в графе «карьера».

На тот случай, если название искомой компании содержится в одном из пунктов раздела «Карьера» в виде подстроки, для поиска используется алгоритм Бойера-Мура [15]: для каждого пункта карьеры пользователя поиск осуществляется по шаблону а не по полному совпадению строковых переменных, в которых хранятся название компании и указанная информация о месте работы. Для повышения процента верно классифицированных страниц сотрудников возникает необходимость построения дерева принятия решений для каждой анализируемой компании. Это можно объяснить, например, разницей в численностью штата сотрудников и степенью активности различных пользователей социальной сети «ВКонтакте» в официальной группе компании, что усложняет попытку создания единой структуры для анализа всех компаний. При реализации классификатора под каждую конкретную организацию остро встает проблема автоматизированного анализа целевого параметра обучающей выборки. Для решения этой проблемы в данном исследовании было решено для обучающей выборки использовать параметр *hasFirmName* в качестве значения целевого параметра. Это означает, что если пользователь «ВКонтакте» отметил в качестве своего текущего места работы данную компанию, то страница данного пользователя добавляется в обучающую выборку со значением целевого параметра = 1, (далее в тексте работы для этого параметра используется обозначение *isEmployee*). Данное допущение делается для полной автоматизации процессов сборки тестовой и обучающей выборок: в результате отпадает необходимость в анализе страниц этих выборок экс-

пертами с целью определения значения целевого параметра, поскольку его значение приравнивается к значению *hasFirmName*, а сам параметр *hasFirmName* в обучении классификатора не участвует.

3.2.2. Текстовые записи официальной группы организации

Некоторые компании часто обновляют новости различного характера на стену сообщества, которое является официальным представителем заданной компании во «ВКонтакте». Периодически, среди таких новостных записей встречаются фамилии сотрудников. Третий параметр отвечает за упоминание фамилии сотрудника в новостных записях на стене организации, (далее в тексте этот параметр обозначается как *onWeb*). Поиск параметра *onWeb* осуществляется по полю $User.LastName$. Зачастую официальная группа содержит большое количество текстовых записей на стене, содержащих информацию о каких-либо событиях так или иначе связанных с деятельностью компании. Таким образом, встаёт задача поиска большого количества шаблонов в большом объеме текста. Эту проблему можно было бы решить при помощи алгоритмов поисков строки в тексте, в частности наиболее известного из них, алгоритма Бойера-Мура [15], общая оценка вычислительной сложности которого составляет

$O(n+m)$, где n — длина строки, в которой осуществляется поиск, а m — длина шаблона. Но данный алгоритм в контексте поставленной задачи обладает существенными недостатком: алгоритм не предназначен для реализации поисковой машины, т.к. требуется найти большое множество шаблонов, причём поиск фамилии необходимо осуществлять в формате различных падежей русского языка.

В качестве альтернативы поисковым алгоритмам было решено составить словарь на основе текстовых записей группы. Наиболее подходящей для этого структурой в языке C# является $Dictionary < Key, Value >$, для заполнения которой нам достаточно преобразовать все текстовые записи на стене компании в массив слов.

Далее, после заполнения словаря, необходимо осуществить поиск всех фамилий, представленных в обучающей выборке. Для определе-

ния представлений фамилий в русском языке в различных падежах было решено использовать набор библиотек *Morpher .NET SDK*[11]. Стоит отметить, что в данном SDK присутствуют платные библиотеки, с помощью которых можно производить полностью автоматизированный процесс склонения фамилий сотрудников. Однако, бесплатных библиотек оказывается достаточно для выполнения тех функций, которые предоставляет SDK, с тем ограничением, что все вычисления осуществляются на сервере, а доступ к ним, аналогично библиотеке VkNet, осуществляется с помощью Rest-сервиса, представленного в бесплатной части SDK. Таким образом, в проекте используются следующие библиотеки из *Morpher .NET SDK*:

1. MORPHER.API.dll — содержит уровень абстракции IDeclension для русского языка
2. MORPHER.WebService.V2.dll — содержит реализацию Web-интерфейса для русского языка.

Метод, который с помощью Morpher .NET SDK позволяет просклонять фамилии сотрудников, представлен на листинге 3 ниже:

```
private List<string> makeSurnameValuesToSearch(string surname)
{
  surname = surname.ToLower();
  List<string> surname_dec = new List<string>();

  try
  {
    Morpher.Russian.IDeclension declension =
        Morpher.Factory.Russian.Declension;

    surname_dec.Add(declension.Parse(surname).Nominative);
    surname_dec.Add(declension.Parse(surname).Genitive);
    surname_dec.Add(declension.Parse(surname).Dative);
    surname_dec.Add(declension.Parse(surname).Accusative);
    surname_dec.Add(declension.Parse(surname).Instrumental);
    surname_dec.Add(declension.Parse(surname).Prepositional);
  }
```

```
18    catch (Exception ex)
19    {
20        surname_dec.Add(surname);
21    }
22    return surname_dec;
23 }
```

Листинг 3: Склонение фамилии пользователя

В методе *makeSurnameValueToSearch* из листинга 3 также обработаны вызываемые исключения, когда фамилия пользователя на сайте написана латинскими символами и когда израсходован суточный лимит запросов к API, но данная проблема может быть решена заменой web-интерфейса на платную библиотеку *Morpher.dll*. В случае выбрасываемого исключения мы будем обрабатывать поиск фамилии исключительно в именительном падеже.

В случае если для пользователя хотя бы одно из склонений его фамилии было обнаружено в тексте, обновляется значение его параметра *On Web*.

3.2.3. Счётчик пользовательских отметок «мне нравится» в группе

Кнопка «Мне нравится» в социальной сети «ВКонтакте» используется для выражения отношения заинтересованности пользователя к тому или иному контенту. Для выражения степени заинтересованности пользователей в контенте, выкладываемом группой на стене, используется параметр *likesCounter*.Этот параметр фиксирует число отметок «мне нравится» оставленных пользователем в группе. Для анализа данного параметра также было решено использовать структуру *Dictionary<Key, Value>*, где в качестве ключа используется *id* сотрудника, поставившего отметку «мне нравится» под одним из постов группы, а в качестве значения берётся счётчик. После сбора статистики производится поиск страниц из обучающей выборки в полученной структуре, и, в случае нахождения *id* сотрудника в словаре, для него обновляется значение параметра *likesCounter*. Ниже представлен листинг 4

программного кода, собирающего информацию о данном параметре:

```
private void searchInGroupLikes(List<Post> group_posts,
                List<Photo> group_photos)
{
  string filterExpression, sortOrder;
  makeLikesDictionary(group_posts, group_photos);


  foreach(KeyValuePair<long, int> likes_by_user
              in this.likes_in_group)
  {
    filterExpression = "vk_id = '" + likes_by_user.Key + "'";
    sortOrder = "vk_id DESC";
    DataRow[] users_found_surname =
    training_dataset.Select(filterExpression, sortOrder,
   DataViewRowState.Added);

  foreach (DataRow row in users_found_surname)
  {
    row[3] = likes_by_user.Value;
  }
  }
}
```

Листинг 4: Сбор отметок «мне нравится» в официальной группе компании

Представленный метод работает с экземпляром объекта *Datatable*, в котором сохраняются значения всех параметров для обучающей выборки.

3.2.4. Раздел «Друзья» пользовательской страницы компании

Следующий параметр представляет интерес для классификатора в том случае, если существует страница организации в виде пользовательской страницы. Зачастую данная страница принадлежит администратору группы: таким образом, у пользователей появляется возможность не только «подписаться» на официальное сообщество компании в сети «ВКонтакте», но и добавить в «друзья» пользовательскую страни-

цу администратора. Характер связи «друзья» между пользователями сети «ВКонтакте» является двунаправленным, что означает, что для создания подобной связи требуется согласие обеих сторон, в то время как «подписаться» на сообщество часто может любой желающий. Именно из-за «bidirectional» характера связи между «друзьями» было решено добавить данный параметр для построения классификатора.

3.2.5. Анализ топологии сети

Анализ социального графа пользователей социальной сети также может представлять интерес в контексте выявления сотрудников компании. Зачастую, круги общения для различных сотрудников компании имеют пересечения (в данном случае имеются в виду пользователи, которых он «добавил в друзья»). Параметр *following matches* показывает число друзей сотрудника, которые состоят в официальной группе компании. В листинге 5 представлены основные методы, с помощью которых осуществляется анализ топологии сети.

```
1  private Dictionary<long, List<int>> searchFollowingMatches(List<
       int> group_followers_ids , Dictionary<long, List<int>>
       dataset_ids)
2  {
3    Dictionary<long, List<int>> rez = new Dictionary<long, List<int
       >>();
4
5    group_followers_ids.Sort();
6    foreach (KeyValuePair<long, List<int>> entry in dataset_ids)
7    {
8      entry.Value.Sort();
9
10     rez.Add(entry.Key, GetSimilarID(entry.Value,
       group_followers_ids));
11                   Console.WriteLine("for id:{0}", entry.Key);
12     GetSimilarID(entry.Value, group_followers_ids)
13   }
14   return rez;
15 }
16
```

```
17 private List<int> GetSimilarID(IEnumerable<int> list1,
18                 IEnumerable<int> list2)
19 {
20 return (from item in list1 from item2 in list2
21     where (item == item2)
22         select item).ToList();
23 }
```

Листинг 5: Основные методы для анализа топологии сети

3.2.6. Целевая переменная

Переменная *isEmployee* является целевой в контексте бинарной классификации пользовательских страниц в социальной сети «ВКонтакте». Данный параметр = 1, если страница на основе представленных выше параметров была классифицирована как страница сотрудника компании, иначе = 0.

Для полной автоматизации процесса сбора обучающей и тестовых выборок было необходимо автоматизировать процесс анализа целевой переменной для этих выборок. Для решения этой проблемы было решено приравнять значение целевой переменной параметру *hasFirmName*.

Ниже на рис. 4 проиллюстрирована основная форма, которая использует представленные в данной главе методы по сбору информации о пользователе, чей *id* вводится в *textbox* с подписью «please, insert vk user id to analyze». Далее, на основе обученного классификатора вычисляется значение целевой переменной *isEmployee*.

3.3. Программная реализация классификатора

Как уже было упомянуто ранее в пункте 2.2, в данном проекте используется библиотека Accord.NET, которая используется реализации машинного обучения на языке C#, которая охватывает такие области как численная оптимизация, статистика, и машинное обучение.

На листинге представлен класс *DecisionBuilder.cs*, который отвечает за обучение дерева принятия решений. Обучающая выборка для

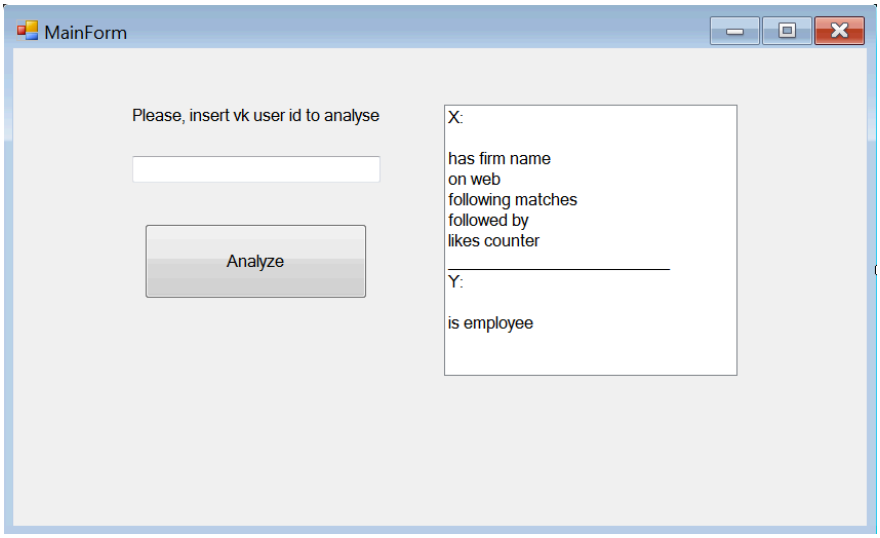

Рис. 4: Окно с запросом прав доступа приложения

дерева содержится в поле *training_dataset* в виде экземпляра класса DataTable. Обученное дерево заполняется в приватное поле *current_DT*, к которому можно обратиться из другого класса через соответствующий get-метод.

```
class DecisionTreeBuilder
{
...

private DecisionTree current_DT;
private DataTable training_dataset;

public void studyDT()
{
// Create a new codification codebook to
// convert strings into integer symbols
Codification codebook = new Codification(training_dataset);

DecisionVariable[] attributes =
{
  new DecisionVariable("on_web",2),
  new DecisionVariable("likes_counter", DecisionVariableKind.
   Continuous),
  new DecisionVariable("followed_by", 2),
  new DecisionVariable("following_matches", DecisionVariableKind.
   Continuous)
```

```
};

int classCount = 2; // 2 possible output values: yes or no

// Create a new instance of the C4.5 algorithm
current_DT = new DecisionTree(attributes, classCount);
C45Learning c45learning = new C45Learning(current_DT);

// Translate our training data into integer symbols using our
   codebook:
DataTable symbols = codebook.Apply(training_dataset);
double[][] inputs =
symbols.ToIntArray("on_web",
           "likes_counter",
                   "followed_by",
                   "following_matches").ToDouble();

int[] outputs =      symbols.ToIntArray("is_employee").GetColumn
   (0);

// Learn the training instances!
c45learning.Run(inputs, outputs);


// Convert to an expression tree
Expression<Func<double[], int>> expression =
          current_DT.ToExpression();

Console.WriteLine(GetDebugView(expression));

// Compiles the expression to IL
var func = expression.Compile();
}
}
```

Листинг 6: класс DecisionBuilder.cs

3.4. Схема базы данных

В существующем программном комплексе была реализована схема базы данных для хранения информации о моделях комплекса «критические документы — информационная система — персонал — злоумышленник». Представление данного комплекса при реализации в РСУБД после сбора соответствующей информации позволяет свести ряд задач по анализу защищённости персонала информационной системы от социоинженерных атак к исполнению SQL запросов к реляционной БД. Также, важным требованием для использования разработанного программного продукта является локально настроенный сервер с базой данных MySQL. Поэтому было решено сохранять найденные страницы, обучающую и тестовую выборки с возможностью обращения к полученным результатам с помощью SQL запросов, что позволяет абстрагироваться от логики реализации программного модуля, так как отпадает необходимость обращения к полям классов из внешнего интерфейса библиотеки.

В схеме базы данных, представленной на рис.5 существует разделение пользовательской информации на две таблицы: *employee* и *employee_found*, связанные при помощи внешнего ключа *employee_id* в таблице *employee_found*. В таблицу *employee_found* сохраняются пользователи со значением целевой переменной $isEmployee = 1$.

В таблице *employee* хранится необходимая личная информация, которая возвращаются по результату REST-запроса к vk.com. На данном этапе среди прочей личной информации для хранения в БД были выбраны Ф.И.О. и возраст сотрудников.

3.5. Сбор страниц на основе обученного классификатора

В результате работы алгоритма, представленного в разделе 2.3 представленного исследования был получен классификатор для пользовательских страниц. Для сбора пользовательских страниц используется

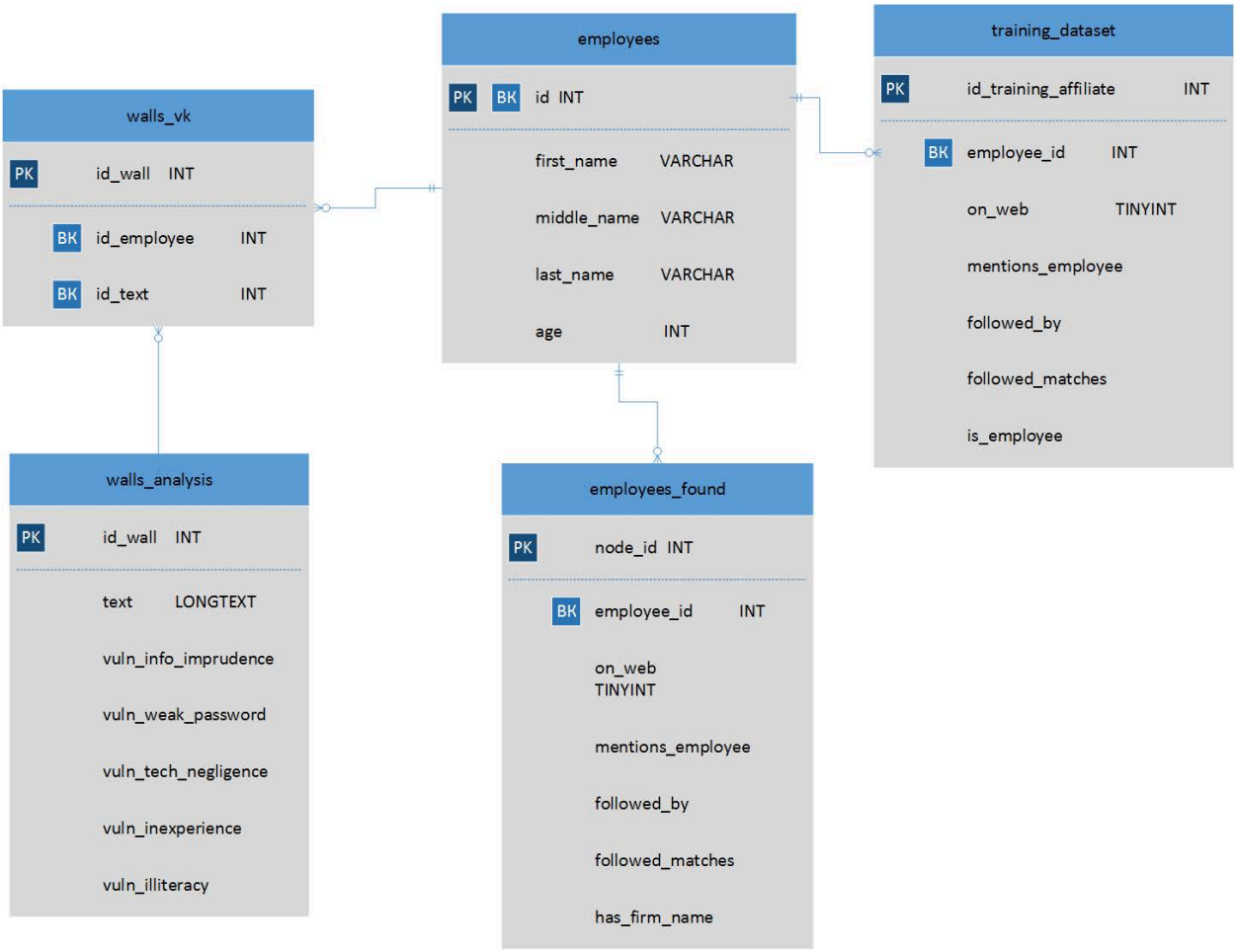

Рис. 5: Представление связей между реляционными таблицами

продвижение в социальном графе пользователя. На листинге представлен псевдокод алгоритма, на основе которого собирается результирую-

щий список сотрудников.

```
begin
    List < User > foundEmployees = newList <> ();
    foundEmployees.Add(X);;
    for c ∈ C do
        if employees_found <= emp_counter then
            x.getFriends();
            x.friends.getParams();
            for y ∈ x.friends do
                collectAllEmpFriends(y);
            end
        end
    end
end
```

Algorithm 1: Алгоритм поиска сотрудников с помощью классификатора

3.6. Описание дистрибутива программного модуля

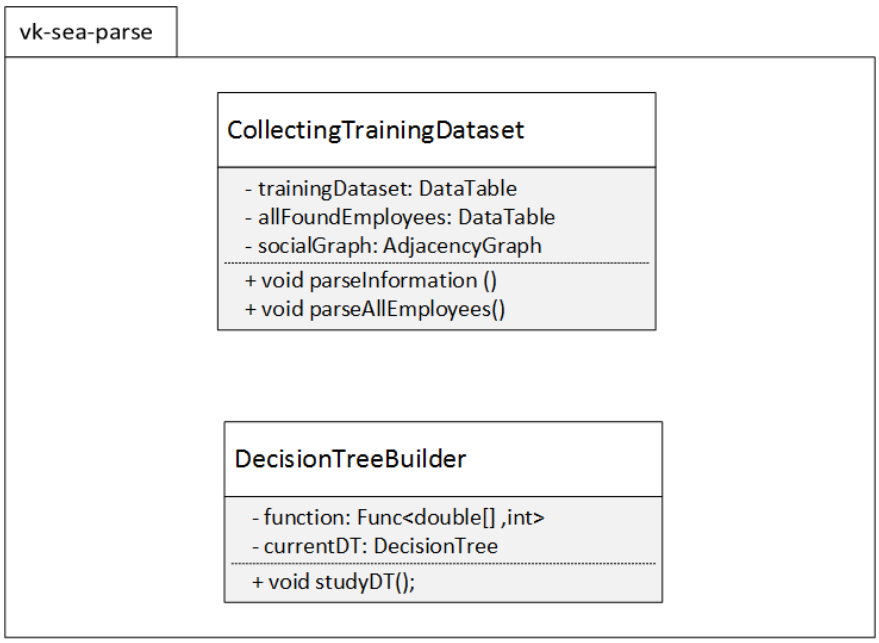

Рис. 6: Внешний интерфейс программного модуля

Для осуществления внедрения разработанного модуля в программ-

ный комплекс «критические документы – информационная система – персонал – злоумышленник» создаётся файл подключаемой библиотеки, её внешний интерфейс для взаимодействия с программным комплексом представлен на рис. 6.

Для получения списка сотрудников компании после прохождения авторизации вызывается метод *parseInformation()*, который отвечает за сбор обучающей выборки, затем совершается построение классификатора в классе *DecisionTreeBuilder*, а результирующий граф найденных сотрудников хранится в поле *socialGraph*, где в вершинах располагаются найденные сотрудники компании, а наличие рёбер между вершинами свидетельствует о статусе «друзей» для данных пользователей в рамках сети «ВКонтакте».

4. Заключение

В представленой работе были выполнены следующие задачи:

1. были выбраны деревья принятия решений в качестве классификатора,
2. разработан метод сбора и классификации пользовательских страниц,
3. разработана модель, позволяющая осуществлять автоматизированную оценку целевого параметра обучающей выборки,
4. разработана и реализована схема БД с возможностью получения информации о значениях параметров идентифицированных сотрудников,
5. программный модуль внедрён в разработанный ранее программный комплекс.

Все поставленные задачи выполнены и цель работы достигнута.
Результаты работы были представлены на 3 конференциях:

1. Юбилейная XV Санкт-Петербургская международная конференция «Региональная информатика (РИ-2016)»;
2. XX Международная конференция по мягким вычислениям и измерениям (XX International Conference on Soft Computing and Measurements, SCM 2017);
3. Всероссийская научная конференция по проблемам информатики «СПИСОК-2017», Санкт-Петербург, 25–27 апреля 2017 года.

По теме дипломного проекта было подготовлено 5 публикаций:

1. Шиндарев Н.А., Абрамов М.В. Построение вероятностной графической модели для оценки успешности социоинженерной атаки// Региональная информатика (РИ-2016). XV Санкт-Петербургская

международная конференция «Региональная информатика (РИ-2016)» (Санкт-Петербург, 26-28 октября 2016 г.): Материалы конференции. СПб.: СПОИСУ, 2016, С. 514 (опубликована)

2. Багрецов Г.И., Шиндарев Н.А., Абрамов М.В., Тулупьева Т.В. Подходы к разработке моделей для анализа текстовой информации в профилях социальной сети в целях построения профиля уязвимостей пользователя // XX Международная конференция по мягким вычислениям и измерениям (принято к печати, индексация в РИНЦ)

3. G. Bagretsov, N. Shindarev, M. Abramov, T. Tulupyeva. Approaches to development of models for text analysis of information in social network profiles in order to evaluate user's vulnerabilities profile // XX International Conference on Soft Computing and Measurements, SCM 2017 (поданы материалы, индексация в Scopus)

4. N. Shindarev, G. Bagretsov, M. Abramov, T. Tulupyeva. Constructing an intellectual system for social network's user's profile analysis aimed at structuring user's vulnerabilities profile // 2nd International Scientific Conference «Intelligent Information Technologies for Industry» (IITI), Sep 14, 2017 – Sep 16, 2017 (поданы материалы, индексация в Scopus)

5. Шиндарев Н.А., Абрамов М.В., Тулупьев А.Л. Анализ страниц пользователей социальной сети "ВКонтакте" с целью выявления сотрудников заданной компании // Всероссийская научная конференция по проблемам информатики «СПИСОК-2017», Санкт-Петербург, 25–27 апреля 2017 года (принято к печати, индексация в РИНЦ)

Список литературы

[1] Information security business. studies of current trends in information security business. URL: http://media.kaspersky.com/pdf/IT_risk_report_Russia_2014.pdf.

[2] The losses from cybercrime continue to grow. URL: http://www8.hp.com/ru/ru/software-solutions/ponemon-cyber-security-report/index.html.

[3] Precision and recall. URL: https://en.wikipedia.org/wiki/Precision_and_recall.

[4] Sensitivity and specificity. URL: https://en.wikipedia.org/wiki/Sensitivity_and_specificity.

[5] Wikipedia: Binary classification. URL: https://en.wikipedia.org/wiki/Binary_classification.

[6] Сайт проекта accord.net. URL: http://accord-framework.net/.

[7] Сайт проекта nuget. URL: https://www.nuget.org/.

[8] Сайт проекта visual studio 2015. URL: https://msdn.microsoft.com/ru-ru/library/dd831853.aspx.

[9] Сайт проекта vknet. URL: https://vknet.github.io/vk/.

[10] Model-view-presenter, 2016. URL: https://en.wikipedia.org/wiki/Model-view-presenter.

[11] Morpher .net sdk, 2016. URL: http://morpher.ru/dotnetapi/.

[12] M. V. Abramov and A. A. Azarov. Social engineering attack modeling with the use of bayesian networks. *Soft Computing and Measurements (SCM), 2016 XIX IEEE International Conference on*, pages 58–60, IEEE, 2016.

[13] M.V. Abramov, A.A. Azarov, T.V. Tulupyeva, and A.L. Tulupyev. Model of malefactor profile for analyzing information system personnel security from social engineering attacks. *Information & Control Systems/Informazionno-Upravlyaushie Sistemy*, 83(4):77–84, 2016.

[14] A.A. Azarov, T.V. Tulupyeva, A.V. Suvorova, A.L. Tulupyev, M.V. Abramov, and R.M. Usypov. *Social Engineering attacks: problem of analisys.* Science, SPb, 2016.

[15] R.S. Boyer and J.S. Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, 1977.

[16] Analytics Brand. Социальные сети в России, зима 2015–2016. Цифры, тренды, прогнозы., 2016. URL: `https://blog.br-analytics.ru/socialnye-seti-v-rossii-zima-2015-2016-cifry-trendy-prognozy/`.

[17] Digital. Социальные сети в России, осень 2016. Цифры, тренды, прогнозы., 2016. URL: `https://adindex.ru/publication/analitics/100380/2016/12/8/156545.phtml`.

[18] M. Fowler and K. Beck. *Refactoring: improving the design of existing code.* Addison-Wesley Professional, 1999.

[19] E.A. Freeman and G.G. Moisen. A comparison of the performance of threshold criteria for binary classification in terms of predicted prevalence and kappa. *Ecological Modelling*, 217(1):48–58, 2008.

[20] I.V. Kotenko and M.V. Stepashkin. Systems-simulators: purpose, functions, architecture and implementation approach. *Information & Control Systems/Informazionno-Upravlyaushie Sistemy*, 83(4):77–84, 2016.

[21] I.V. Kotenko and R.M. Yusupov. Future research directions in the field of computer security. *Information & Control Systems/Informazionno-Upravlyaushie Sistemy*, 83(4):46, 2006.

[22] Е.Р. Горяинова and Т.И. Слепнёва. Методы бинарной классификации объектов с номинальными показателями. *ЖурналНЭА№2*, 2012.

[23] С.И. Николенко and А.Л. Тулупьев. *Самообучающиеся системы*. Litres, 2016.

[24] Е. Фролова. Самые популярные социальные сети в России 2016., 2016. URL: `http://www.pro-smm.com/populyarnye-socialnye-seti-v-rossii-2016/`.

[25] И. Чубукова. Data mining: Информация, 2016. URL: `http://morpher.ru/dotnetapi/`.