1 Анализ предметной области

1.1 Статистические модели текстовых документов

Были протестированы две модели векторизованного представления текста – «мешок слов» и модель TF-IDF. Модель «мешок слов» представляет документ в виде матрицы, представленной на рисунке **??**. Здесь слова каждого абзаца подсчитываются и сопоставляются с абзацами, в которых они встретились.

Рисунок 1 – Bag-of-Words матрица

Модель TF-IDF представляет документ в виде матрицы, представленной на рисунке **??**. Формула (**??**) показывает, как можно получить метрику TF-IDF.

$$tfidf(t,d,D) = \frac{n_t}{\sum\limits_k n_k} \times log\frac{\big|D\big|}{\big|\left\{d_i \in D : t \in d_i\right\}\big|}\,, \tag{1}$$

где t – термин или слово;

d – конкретный абзац;

D – набор абзацев.

Итак, модель TF-IDF придает больший вес словам которые использованы меньше раз. Это может быть полезно, когда тексты схожи с точки зрения используемых слов, как в нашем случае, для политик безопасности.

Рисунок 2 – Матрица TF-IDF

1.2 Подход основанный на латентно-семантическом анализе текста

Современные методы кластеризации текстов позволяют определять тематику текстов с высокой точностью. Однако большинство из этих методов принимают тексты с самыми разными темами как вход для алгоритмов. Но тексты со схожими тематиками можно проанализировать с помощью латентно-семантического анализа дважды: группировать тексты по темам один раз, и предоставить еще более детальное разделение их по подтемам во второй раз. Такой подход можно использовать для более точной классификации абзацев с точки зрения их характеристик и аспектов использования персональных данных. Следует отметить, что латентно-семантический поиск сильно зависит от глобального текстового контекста с потерями информации о локальных контекстных отношениях между словами. Были выделены девять тем конфиденциальности, которые следует сопоставить с абзацами согласия пользователя сайта – «сбор личных данных», «сбор данных третьими лицами», «управление личными данными», «механизмы защиты персональных данных» и др. Очевидно, что аспекты обращения с данными состоят из нескольких слов, и в некоторых случаях перекрываются. На основании этих фактов была выдвинута гипотеза о том, что латентно-семантический поиск способен обнаружить даже незначительную разницу в тексте абзацев при пропуске частых слов. Перед применением латентно-семантического анализа требуется предварительная обработка входных данных. Обычно эта процедура включает очистку данных, удаление гиперссылок, пунктуации и т. д. Также текст политик конфиденциальности был разбит на набор абзацев. Каждый абзац был преобразован в массив слов, которые он содержит. Следующим шагом было удаление наиболее частых, но не столь значимых слов, так называемых стоп-слов. Также была применена операция стемминга, чтобы рассматривать только основную часть всех слов полученных от единого корня.

Пусть A – это матрица абзацев и слов, тогда используя формулу (**??**)

$$A = U \times S \times V^T \tag{2}$$

где A – матрица слов и параграфов;

U – ортонормированная матрица U;

V – ортонормированная матрица V;

S – диагональная матрица S, значения которой сингулярны для A.

После того, как матрица была разделена на три компонента, матрица U содержит n-мерные векторы, которые можно интерпретировать как координаты в n-мерном пространстве [**??**]. Документы могут быть распределены по кластерам по значениям этих координат. Проведенные эксперименты с латентно-семантическим анализом выполнялись с использованием набора данных с открытым исходным кодом, который включает 115 политик безопасности, которые были размечены вручную, и все абзацы присвоены одному или нескольким сценариям использования персональных данных [**??**]. Результаты экспериментов для модели «мешок слов» представлены в таблице **??**, в ней показаны полученные кластеры и соответствующие значения координат.

Таблица 1 – Кластеры политик безопасности для модели Bag-of-Words

	№	Coordinate 1	Coordinate 2	Coordinate 3	Coordinate 4
0	0.634”inform”	0.28”may”	0.276”use”	0.232”servic”
1	0.202“cooki”	0.466“inform”	0.336“site”	0.257“use”
2	0.524“privaci”	0.433“polici”	0.388“cooki”	0.219“site”
3	-0.589“servic”	0.344“site”	0.244“parti”	-0.240“third”
4	-0.504“parti”	0.486 “third”	-0.449“servic”	0.235“advertis”

Продолжение таблицы **??**

	№	Coordinate 1	Coordinate 2	Coordinate 3	Coordinate 4
5		-0.594“site”	0.278“cooki”	0.272“websit”	0.264“privaci”
6		-0.326“may”	0.311“site”	0.307“servic”	-0.293”email”
7		-0.437”may”	-0.369”advertis”	0.345”person”	0.319”cooki”
8		0.501”may”	-0.315”email”	-0.281”use”	-0.264”address”
9		-0.488”user”	-0.384”use”	0.310”provid”	-0.301”websit”

Как видно, результаты противоречивы, поэтому трудно понять, какая из тем каким смыслом обладает. Затем рассчитывалась метрика принадлежности к теме с помощью библиотеки Gensim [**??**] и результаты снова не были обнадеживающими. Результаты расчета метрики принадлежности кластеру представлены в таблице **??**.

Таблица 2 – Принадлежность кластерам

Topic	0	1	2	3	4
Affiliation	2.27	-0.8	0.15	-0.22	-1.2
Topic	5	6	7	8	9
Affiliation	-0.17	-0.15	-0.2	0.22	-0.07

Другие результаты с параграфами, относящимися к другому аспекту обращения с данными, были почти такими же. Результаты представлены в таблице **??**.

Таблица 3 – Принадлежность кластерам

Topic	0	1	2	3	4
Affiliation	2.59	-0.76	0.64	0.74	0.13
Topic	5	6	7	8	9

Affiliation	0.14	-0.12	0.23	0.12	0.41

Все протестированные абзацы были сопоставлены с кластером 0, что не может быть верным так как абзацы относились к заведомо разным аспектам обращения с персональными данными.

Результаты экспериментов для модели TF-IDF представлены далее, в таблице **??**. Также показывались десять кластеров и значения атрибутов. И, как в первом случае с «мешком слов», по значениям координат невозможно судить о теме кластера.

Таблица 4 – Кластеры политик безопасности для модели Bag-of-Words

№	Coordinate 1	Coordinate 2	Coordinate 3	Coordinate 4
0	0.202“cooki”	0.2“may”	0.198“inform”	0.198“site”
1	0.573“cooki”	0.262“browser”	0.195“advertis”	0.182“web”
2	-0.406“media”	0.291“cooki”	0.282“health”	0.279“advertis”
3	-0.453“health”	0.258“email”	-0.204“kaleida”	0.191“address”
4	0.423“health”	0.215“media”	0.205“kaleida”	-0.199“secur”
5	-0.299“advertis”	0.262“health”	-0.252“media”	-0.213“privaci”
6	-0.325“media”	0.263“polici”	0.249“privaci”	0.197”chang”
7	0.280”cooki”	-0.216”device”	-0.183”health”	-0.166”social”
8	-0.223”advertis”	-0.206”teenag”	-0.206”inelig”	0.176”child”
9	-0.263” child”	-0.26”wireless”	0.245”message”	0.239”parent”

Результаты кластеризации снова противоречивы, поэтому трудно сказать, какая конкретная тема описывает какой аспект политики конфиденциальности. В разных темах встречаются одни и те же слова с изменение веса. Для аспектов политики конфиденциальности, которые мы искали нет тем,

которые могли бы их точно описать, поскольку многие из них могут. Затем с помощью библиотеки Gensim был рассчитан показатель принадлежности к теме, и результаты снова не были обнадеживающими. Результаты расчета аффилированности по абзацу одной из политик конфиденциальности представленные в таблице **??**.

Таблица 5 – Принадлежность кластерам

Topic	0	1	2	3	4
Affiliation	2.18	-0.97	-0.69	-0.27	0.65
Topic	5	6	7	8	9
Affiliation	0.98	-1.17	0.8	0.27	0.01

Результат для другого абзаца, относящегося к другой политике конфиденциальности, был почти такой же. Результаты представлены в таблице **??**.

Таблица 6 – Принадлежность кластерам

Topic	0	1	2	3	4
Affiliation	1.82	0.25	0.49	0.29	-0.04
Topic	5	6	7	8	9
Affiliation	0.74	0.52	-0.04	-0.58	-1.33

Как можно заметить, результаты для модели TF-IDF аналогичны результатам модели «мешка слов», за исключением нескольких незначительных изменений. Все абзацы снова были сопоставлены с кластером 0, что неверно, потому что они на самом деле описывают разные сценарии использования персональных данных. Эти эксперименты позволили сделать вывод, что использование латентно-семантического анализа не дает ценной информации о содержании онлайн-согласия пользователя. Проблема может быть

связана с тем, что сценарии использования персональных данных очень похожи между собой, и для того, чтобы различать разные сценарии необходимо учитывать локальный контекст.

В результате апробации алгоритма латентно-семантического анализа было выяснено что для кластеризации экстремально схожих между собой текстов он подходит не лучшим образом. В связи с этими обстоятельствами было решено обратить внимание на несколько иной подход анализа текста, основанный на контекстно-свободных грамматиках, тегировании по частям речи и синонимическом поиске.

1.3 Подход основанный на латентном размещении Дирихле

1.4 Подход основанный на применении контекстно-свободных грамматик и синонимическом поиске

Другой предложенный подход – подход, основанный на анализе с помощью контекстно-свободных грамматик и синонимического поиска. Синонимический поиск в данном случае – это подмена ключевых слов и их синонимов метками, например «__FP_A__» означает, что это слово и его синонимы считаются акторами первой стороны. Этот метод можно применить ко многим другим словам. Например, сообщения электронной почты, аватары, местоположение также могут быть объектами и синонимами абстрактной метки «__CN__», которая означает существительное сбора или объект сбора. Так все ключевые слова могут быть преобразованы в их смыслы в контексте предметной области. Маркировка выполняется легко, все слова совпадающие с пулами заменяются метками этих пулов.

Предварительная обработка данных в данном случае состоит из токенизации и лемматизации для более гибкой замены слов на метки их пулов.

При анализе пользовательского согласия сайта недостаточно найти ключевые слова, относящиеся к разным типам персональных данных, например цель и правовую основу распознать гораздо сложнее. Следующий шаг - установить слова отношения в предложениях, чтобы можно было определенно

сказать, что ярлыки пулы синонимов связаны друг с другом и формируют логическая цепочку. Один из возможных способов определения отношений слов в тексте на естественном языке – это синтаксический анализ предложения, основанный на частеречной разметке [**??**]. Имея размеченное по частям речи предложение, парсер грамматики NLTK [**??**] строит деревья предложений по правилам грамматики. Одно из таких деревьев в обозначениях NLTK можно увидеть на рисунке **??** [**??**], где «S» – основа предложения, «NP» – именная фраза, «VP» – глагольная фраза, «Adj» – прилагательное, «НОМ» – именное словосочетание, «ПП» – предлог фраза, «Det» – артикль, «V» – глагол, «N» – существительное, «P» – предлог.

Рисунок 3 – Матрица TF-IDF

В предлагаемом подходе немного другая грамматическая запись. Созданная грамматика представлена в (**??**).

$$\begin{cases} D \to S \mid S\ D \mid S\ \ U\ D \\ S \to NPG\ \ VBG \\ VPG \to VP \mid VP\ \ VPG \mid VP\ \ U\ \ VPG \\ NPG \to NP \mid NP\ \ NPG \mid NP\ \ U\ \ NPG \\ AJPG \to AJ \mid AJ\ \ APG \mid AJ\ \ U\ \ APG \\ AVPG \to AV \mid AV\ \ APG \mid AV\ \ U\ \ APG \\ VP \to VAPG \mid V\ \ PPG \mid V\ \ PP\ \ APG \\ NP \to NOM \mid DET\ \ NOM \\ NOM \to N \mid AJPG\ \ N \\ PP \to NPG \mid P\ \ NPG \end{cases}, \tag{3}$$

где «D» – документ, «S» – предложение, «NPG» – группа именных фраз,

«VPG» – группа глагольных фраз, «AJPG» – группа прилагательных, «AVPG» – группа наречий, «VP» – глагольная фраза, «NP» – именная фраза, «NOM» – именная фраза, «PP» – предлог, «DET» – артикль. Грамматика из формулы (**??**) позволяет рекурсивно выделять основу предложения и последовательности глагола, существительного, прилагательного, наречия и т.д. Это все еще не идеальное решение, но попытка найти более сложные предложения в политиках безопасности. Этот подход требует использования пулов синонимов, которые соответствуют различным ключевым словам. Поэтому в грамматику включены метки пулов синонимов, привязанных к части речи. Метки пулов вручную назначены частям речи для преобразования привязок частей речи NLTK, это показано в формуле (**??**).

$$\begin{cases} U \rightarrow NLTK\_CC \\ DET \rightarrow NLTK\_DT \\ AJ \rightarrow NLTK\_JJ \\ AV \rightarrow NLTK\_RB \\ N \rightarrow \_\_CN\_\_ \mid \_\_FP\_A\_\_ \mid \_\_TP\_A\_\_ \mid NLTK\_N \\ V \rightarrow \_\_CV\_\_ \mid NLTK\_V \end{cases} \tag{4}$$

Здесь в формуле (4) «NLTK_CC» – соединение NLTK, «NLTK_N» – все формы существительных NLTK, «NLTK_B» – все формы глаголов NLTK, «NLTK_DET» – определители NLTK, «NLTK_JJ» – все NLTK формы прилагательных, «NLTK_RB» – все формы наречий NLTK. Теги, начинающиеся с подчеркивания, являются метками пулов синонимов. Синтаксический анализ выполняет библиотека NLTK. На основе предложенной грамматики, описанной (**??**) и (**??**) и разметки лейблами пулов было построено дерево предложения, результат на рисунке **??**.

Рисунок 4 – Матрица TF-IDF

Здесь на рисунке **??** «__FP_A__» - метка актора-обладателя персональных данных, «__TP_A__» - третья сторона, «__CV__» - глагол сбора, «__CN__» - существительное сбора. Когда было построено дерево предложений последовательность меток ключевых слов может быть распознана. В этом случае представленная на рисунке **??**, последовательность «__FP_A__», «__CV__», «__CN__» хорошо видна. Такие атомарные последовательности, раскрывают значения частей предложения и могут быть объединены в список, после этого весь смысл документов будет описан этим список. Сочетание маркировки ключевых слов и синтаксического анализа дает значения ключевых слов с отношениями между этими словами, определенными в виде древовидных структур. Дерево структура данных более гибкая, чем строка предложения, деревья и особенно поддеревья показывают важные отношения между словами. Запросы к таким структурам могут дать необходимую информацию для построения логических последовательностей действующих лиц, их действий, субъектов этих действий и, наконец, обстоятельств. Предлагаемый подход определенно имеет такие недостатки, как низкая производительность, вручную определенные пулы синонимов и т.д.

В результате апробации алгоритма латентно-семантического анализа было выяснено что для кластеризации экстремально схожих между собой текстов он подходит не лучшим образом. В связи с этими обстоятельствами было решено обратить внимание на несколько иной подход анализа текста, основанный на контекстно-свободных грамматиках, тегировании по частям речи и синонимическом поиске.

1.5 Выводы по строгим методам текстового анализа

Эксперименты показали, что оба рассмотренных метода имеют как преимущества, так и определенные недостатки. Хотя предложенные подходы, оказались противоречивыми, окончательные результаты заслуживают вни-

мания. Подход с латентно-семантическим поиском оказался не слишком эффективным. Однако, подход основанный на грамматическом анализе предложений и синонимическом поиске дал определенные результаты. Хоть он и не является производительным, с его помощью возможно производить выделение логических цепочек из предложений для получения более формального описания политик безопасности нежели их текстовые варианты.

1.6 Подход основанный на глубоком обучении

Исходя из проведенных исследований стало понятно, что более предпочтительным вариантом решения задачи будет подход с применением моделей с глубоким обучением. Реализация подобного проекта – комплексная задача, ее можно разделить на несколько этапов. Сначала необходимо собрать датасет, потом его разметить для обучения модели, далее обучить модель и получить результаты. Однако сбор датасета тоже является непростой задачей. Для того чтобы осуществить сбор датасета необходим инструмент для поиска и скачивания вэб-страниц из сети интернет. Затем необходимо произвести очистку данных, удалить все теги со страниц, чтобы можно было передать текст аннотаторам. Все этапы сбора датасета полагаются на базу данных. Она лишена сложного объектно-реляционного моделирования, так как в ней по сути необходимо только хранить промежуточные результаты обработки текстовых файлов.

2 Проектирование инструментария

2.1 Техническое задание «Инструментарий для сбора датасета»

2.1.1 Основные положения технического задания

2.1.2 Скрейпер вэб-страниц

Скачивание веб-страниц будет производиться инструментом написанном на языке Python, с помощью библиотек можно скачивать страницы анализировать данные с них, переходить по гиперссылкам и много другое. Такой инструмент позволит просматривать и сохранять содержимое страниц в автоматическом режиме без вмешательства пользователя. Таким образом в автоматическом режиме можно сохранить и проанализировать огромное количество текстовой информации.

2.1.3 Очистка скачанных страниц политик

Для очистки страниц от кода разметки планируется использовать библиотеку «html sanitizer». Очистка кода необходима для того, чтобы аннотаторы могли максимально сфокусироваться на анализе текста, таким образом получая чистый текст они не будут отвлекаться на не имеющие значения в контексте задачи фрагменты.

2.1.4 Инструмент разметки датасета

Инструмент разметки датасета планировалось реализовать с помощью веб-технологий. Серверная часть будет полагаться на приложение, написанное на PHP, которое будет регулировать порядок выдачи текста на аннотирование. Процесс разметки высокодинамичен, поэтому невозможно избежать написания качественной клиентской части приложения на языке javascript. Это позволит сделать работу аннотаторов максимально производительной, в «одну сессию», так как страница не будет перезагружаться, однако все изменения, которые будут вносится, сохранятся.

2.1.5 Фреймворк глубокого обучения

Для создания и тренировки модели анализа текста планируется использовать фреймворк машинного обучения «Keras». Он позволяет быстро создавать классификаторы с самыми разными конфигурациями и любых типов.

После того как классификатор будет сконфигурирован останется лишь обучить его на датасете, полученном ранее.

Обученный классификатор будет в состоянии определять различные характеристики политики безопасности и аспекты обращения с данными, что позволит в автоматическом режиме формировать краткие отчеты о безопасности предоставляемого соглашения.

2.2 Первичная декомпозиция и планирование

Начальным этапом решения задачи является первичная декомпозиция, в ее результате выделяются подзадачи различной важности, которые должны быть решены для доведения цикла разработки до конца. В данном случае можно выделить следующие подзадачи:

1) определение источника информации о различной IoT-продукции,

2) отправка поискового запроса,

3) получение результатов запроса (список IoT-продуктов),

4) определение производителей IoT-продукции,

5) поиск официальных сайтов производителей в сети интернет,

6) поиск раздела «политика безопасности» на сайтах производителей,

7) скачивание политик безопасности,

8) очистка скачанных веб-документов от лишних элементов разметки.

Получение списка производителей возможно на электронных торговых площадках, типовая разметка веб-страниц располагает для получения такой информации, так как существует лишь несколько вариантов наполнения страницы продукции.

Получение официальных веб-сайтов производителей задача на первый

взгляд сложная, однако результаты ручной проверки показали, что лучшим вариантом является поисковый запрос с названием производителя «как есть». В таком случае веб-сайт производителя оказывается на первой строчке результата поискового запроса, а если не оказывается, значит у этой компании его с очень большой вероятностью нет.

2.3 Приложение вэб-скрейпер

2.3.1 Структура приложения вэб-скрейпера

Исходя из результатов декомпозиции, эффективным подходом выглядит представление приложения в виде последовательно выполняющихся подпрограмм так, что входом модуля является результат работы предыдущего модуля, то есть в виде конвейера. Схема организации приложения представлена на рисунке **??**.

Рисунок 5 – Схема организации приложения

Далее была разработана композиционная модель приложения, на ней присутствуют все необходимые для решения задач модули. Схема представлена на рисунке **??**.

Рисунок 6 – Композиционная модель приложения

Здесь на рисунке **??** модуль «main» отвечает за запуск программы, развертывание основных ее частей. Там же происходит инициализация пула потоков для мультипроцессинга затратных задач таких как, например, взаимодействие с «безголовым» браузером.

Модуль «pipeline» отвечает за последовательное исполнение подпрограмм элементов конвейера. Он осуществляет прием выходных и передачу входных данных.

Модуль «scraping» отвечает за получение данных с веб-страниц. Ин-

формация полученная с помощью этого модуля записывается в csv-файл для большей прозрачности и возможности сохранения результатов между запусками приложения, например, для пропуска данного этапа и использования его сохраненных результатов работы. Он включает в себя 3 основных действия:

1) получение производителей IoT-продуктов (соответствует модулю «emarts»),

2) получение официальных сайтов производителей (соответствует модулю «websites»),

3) получение веб-ссылок на политики безопасности (соответствует модулю «policies»).

Модуль «downloader» отвечает за скачивание страниц и их сохранение в отведенную для этого директорию. Информация полученная с помощью этого модуля записывается в html-файлы для большей прозрачности и возможности сохранения результатов между запусками приложения, например, для пропуска данного этапа и использования его сохраненных результатов работы.

Модуль «sanitizer» отвечает за очистку скачанных веб-страниц от ненужных тегов и ссылок. Информация полученная с помощью этого модуля записывается в html-файлы для большей прозрачности и возможности сохранения результатов между запусками приложения, например, для пропуска данного этапа и использования его сохраненных результатов работы.

Модуль «emarts» включает в себя набор плагинов, каждый из которых адаптирован для получения требуемой информации с определенного шаблона веб-страничной разметки. Некоторое поведение инкапсулировано в абстрактном плагине для увеличения «reusability» кода. Получая адрес на вход, данный плагин производит скачивание страницы и с помощью набора шаблонов пытается извлечь информацию. Данный модуль записывает полученную с помощью плагинов информацию в csv-файл для большей прозрачно-

сти и возможности сохранения результатов между запусками приложения, например, для пропуска данного этапа и использования его сохраненных результатов работы.

Модуль «web» отвечает за взаимодействие с веб-сайтами будь то торговые площадки или сайты производителей IoT-продуктов. В нем используется geckodriver для управления «безголовым» браузером.

Модуль «tools» содержит вспомогательные функции, в частности для ввода и вывода данных в формате csv.

2.3.2 Потенциальные проблемы

Потенциально возможные проблемы при реализации приложений подобного типа:

1) блокировка из-за подозрительных заголовков браузера,

2) блокировка из-за слишком частого обращения с запросами,

3) как следствие 2-х предыдущих пунктов требование подтвердить, что это не попытка автоматического доступа (ввод капчи).

Упомянутые проблемы решаются использованием разных заголовков браузера попеременно. Также отправка запросов ограничена по частоте от 15 до 30 секунд, ограничение выбирается случайным образом. Такие решения позволяют крайне редко попадать под подозрения, соответственно процент успеха при попытке получить данные с веб-страницы значительно повышается.

2.4 Инструмент разметки датасета

Инструмент разметки датасета планировалось реализовать с помощью веб-технологий. Серверная часть будет полагаться на приложение, написанное на PHP, которое будет регулировать порядок выдачи текста на аннотирование. Процесс разметки высокодинамичен, поэтому невозможно избежать написания качественной клиентской части приложения на языке javascript. Это позволит сделать работу аннотаторов максимально производительной, в «одну сессию», так как страница не будет перезагружаться, однако все изме-

нения, которые будут вносится, сохранятся.

2.4.1 Объектное моделирование приложения

Объектная модель инструмента представлена на рисунке **??**.

Рисунок 7 – Объектная модель

2.4.2 Реляционная модель приложения

Реляционная модель инструмента представлена на рисунке **??**.

Рисунок 8 – Реляционная модель

2.4.3 Проектирование пользовательского интерфейса

Презентационный прототип интерфейса инструмента представлен на рисунке **??**.

Рисунок 9 – Презентационный прототип интерфейса

2.5 Выбор средств разработки

Для реализации приложения были выбраны следующие средства:

1) python 3.8.5,

2) «безголовый» браузер Firefox,

3) драйвер для управления «безголовым» браузером geckodriver,

4) библиотека html sanitizer для очистки скачанных веб-документов.

Выбор «безголового» браузера обусловлен потребностью в отрисовке страниц, так как на некоторых веб-страницах разметка генерируется с помощью javascript. Это делает невозможным использование простого скачивания, необходима страница именно с исполненными скриптами, в противном случае будет невозможно получить требуемую информацию. В то же время браузер лишен графического интерфейса, чем снижается потребление вычислитель-

ных ресурсов.

3 Реализация инструментария

3.1 Полученные в результате реализации исходные коды

В соответствии с результатами декомпозиции, выбора средств и проектирования приложение было реализовано. Характеристики полученных классов и функций приведены далее, в таблицах ?-?. Исходные коды представлены в приложении А.

3.2 Полученный в результате реализации пользовательский интерфейс инструмента разметки

3.3 Результаты решения поставленной задачи с помощью разработанного инструментария

В результате работы программы были найдены 65 политик безопасности, разумеется среди них имеется определенный процент промахов, если производитель имеет сходство с каким либо другим более крупным. Поиск осуществлялся по торговой площадке amazon, брались результаты поискового запроса по первым 10-ти страницам, по категориям «smart scales», «smart watches», «smart locks» и «smart bulbs». Всего производителей было найдено приблизительно 160. Стоит отметить, что результат является приемлемым, так как многие производители на данной торговой площадке не имеют выделенного веб-сайта, а пользуются услугами amazon, то есть на таких страницах действует политика безопасности amazon, а не производителя. Также стоит отметить что у некоторых продуктов явно не указан производитель, что сократило количественно результат поиска.