ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
”ВЫСШАЯ ШКОЛА ЭКОНОМИКИ”»

Московский институт электроники и математики

Юткин Дмитрий Игоревич, группа БИВ-144
Вдовкин Василий Алексеевич, группа БИВ-144

Классификация и агрегация новостных статей, используя методы и модели обработки естественного языка

Междисциплинарная курсовая работа
по направлению 09.03.01 Информатика и вычислительная техника
студентов образовательной программы бакалавриата
«Информатика и вычислительная техника»

Студент _____ Д.И. Юткин
Студент _____ В.А. Вдовкин

Руководитель
старший преподователь
Л.Л. Волкова

_____

Москва 2017 г.

Аннотация

В данной работе изучаются алгоритмы классификации и агрегации текста, основанные на методах и моделях обработки языка, и применяются на русскоязычных новостных статьях. В процессе работы удалось собрать и обработать большой корпус новостей, на котором обучены следующие модели: TFIDF, SVM, FastText, word2vec, Kmeans. Классификаторы показали точность 86-88%. Для визуализации работы перечисленных алгоритмов на практике реализован новостной агрегатор в виде web-сервиса, агрегирующий и классифицирующий актуальные новости с сайтов российских СМИ.

Abstract

The main purposes of this work are the natural language processing models and algorithms of the text aggregation and classification and application of these models and algorithms on the russian media news articles. In the course of this work a dataset with the significant number of news was gathered and processed. On this dataset TFIDF, SVM, FastText, word2vec, Kmeans models were trained. The accuracy of the classifiers is between 86 and 88 depending on a model. To show how these algorithms are working on practice the news aggregation web-service was implemented, which aggregates and classifies articles in real-time.

Оглавление

1 Введение

С каждым годом вычислительные мощности современных компьютеров и сервисов, предлагающие облачные вычисления, позволяют обрабатывать всё большие массивы данных. Благодаря этому происходит быстрое развитие алгоритмов анализа данных и машинного обучения. Результат работы этих алгоритмов можно увидеть в нашей повседневной жизни: сервисы прогноза погоды, которые предсказывают направление движения облаков, персонализированная реклама, подстраивающиеся под интересы пользователя, автомобили с автопилотом и т.д. — в основе всех этих разработок лежат алгоритмы интеллектуального анализа данных и машинного обучения.

Одна из важнейших проблем, возникающая перед исследователями и разработчиками подобных систем, заключается в поиске данных для обучения моделей, которые в будущем будут использоваться для анализа новой информации.

Решением данной проблемы являются тексты из сети Интернет. Петабайты информации, записанной на естественном языке за последние несколько десятилетий, доступны любому исследователю. Книги, новостные статьи, блоги, посты в социальных сетях — всё это является источником легкодоступных данных.

Именно поэтому обработка естественного языка (Natrual Language Processing, NLP) получила большое развитие. За последние несколько лет появилось множество специализированных библиотек и сервисов для анализа естественного языка. К таким сервисам, например, относится IBM Watson — набор платных продуктов, предлагающий различные инструменты работы с текстом: от извлечения важной информации и классификации, до его генерации. Яркими примерами свободнораспространяемых библиотек являются: NLTK, Gensim, MyStem, pymorphy и многие другие.

В данной работе решаются две задачи:

а) классификация тем новостных статей;

б) агрегация новостных статей по смысловой близости.

Для придания работе новизны и оригинальности задача решается для русскоязычных новостей.

Практическая значимость работы доказывается на примере реализации в виде веб-сервиса новостного агрегатора, который в реальном времени, с помощью

алгоритмов и подходов предложенных в данной работе, обрабатывает публикации русскоязычных интернет-СМИ.

2 Цель и задачи курсовой работы

Целью курсовой работы является разработка веб-сервиса, который:

а) в реальном времени получает статьи из русскоязычных новостных источников;
б) классифицирует полученные статьи по общим темам;
в) агрегирует по схожести содержания статьи из различных источников.

Агрегация и классификация основана на исследуемых в работе алгоритмы.

Для достижения поставленной цели, должны быть выполнены следующие задачи:

а) Изучить методы и модели автоматической обработки текста и естественного языка;
б) Собрать корпус новостных статей для обучения моделей;
в) Исследовать и реализовать различные подходы к классификации и агрегации текстовых документов;
г) Разработать back-end и front-end инфраструктуру сервиса.

3 Сбор и подготовка данных

3.1 Получение данных из новостных источников

Для получения робастных моделей машинного обучения, требуется достаточно большой корпус новостных статей, содержащий несколько сотен тысяч документов. Обычно существуют готовые коллекции текстов, но из-за выбранных ограничений к документам, а именно: новости на русском языке вместе с тег — словом, описывающим тему документа, найти такой корпус не удалось, поэтому было принято решение собрать данные самостоятельно.

В качестве новостных источников были выбраны следующие популярные СМИ: «Газета.Ru», «Lenta.ru», «ТАСС», «Новая Газета», «ВЕДОМОСТИ», «РИА Новости» и «СПОРТ-ЭКСПРЕСС». Последнее было выбрано по причине малого количества спортивных новостей от других источников.

При анализе сайтов СМИ стало понятно, что они имеют схожую структуру: для отображения ссылок на статьи, используются страницы со списком новостей

(«Лента новостей»), по которым можно итерироваться, изменяя параметры запросов, например, дату последней новости на странице или количество показанных статей. Для извлечения данных из источников реализован набор алгоритмов, которые опираются на описанную структуру.

Стоит отметить, что во многих случаях для получения чистого текста приходится ждать ответа от сервера и обрабатывать HTML содержание страниц, что сильно замедляет работу, поэтому алгоритмы получения новостей одновременно обрабатывают в параллельных процессах множество веб-страниц, что значительно ускоряет работу. Данные алгоритмы также используются в основном web-сервисе для получения недавних новостей.

При формировании корпуса, все новостные статьи сопровождались различными метаданными: название СМИ, ссылка на статью, дата публикации, тег и заголовок. В результате было получено более 1,1 млн. новостей с 1999 по 2017 год, многие из которых имели неправильно проставленные темы или не имели тем вовсе. Причин тому может быть несколько, например, ошибки редакторов или технические ограничения веб-сайтов новостных агентств. Например, большинство новостей на сайте « Новой Газеты» помечены тегом «политика», что зачастую не совпадает с истинным содержанием статьи. После детального анализа тегов стало ясно, что относить новости к рубрикам редакторы стали только после определённого времени, а все уже имеющие статьи на сайте отнесли в «политику». Некорректные данные пришлось удалить.

В итоговую выборку, которая в дальнейшем использовалась для обучения и тестирования алгоритмов, вошло 133 529 статей, помеченных 32 различными тегами. Распределение СМИ и тем на отобранных данных отражено на рисунке 1.

3.2 Нормализация новостных статей

Нормализация является одной из важнейших стадий обработки естественного языка. Не формально, нормализация приводит текст в более информативный для моделей вид. В зависимости от языка процесс нормализации может отличаться. Например, для русского языка зачастую удаляют пунктуацию и стоп-слова, а также производят лемматизацию. Стоп-слова — это слова, которые примерно одинаково распределены по всему корпусу языка, чаще всего ими являются место-

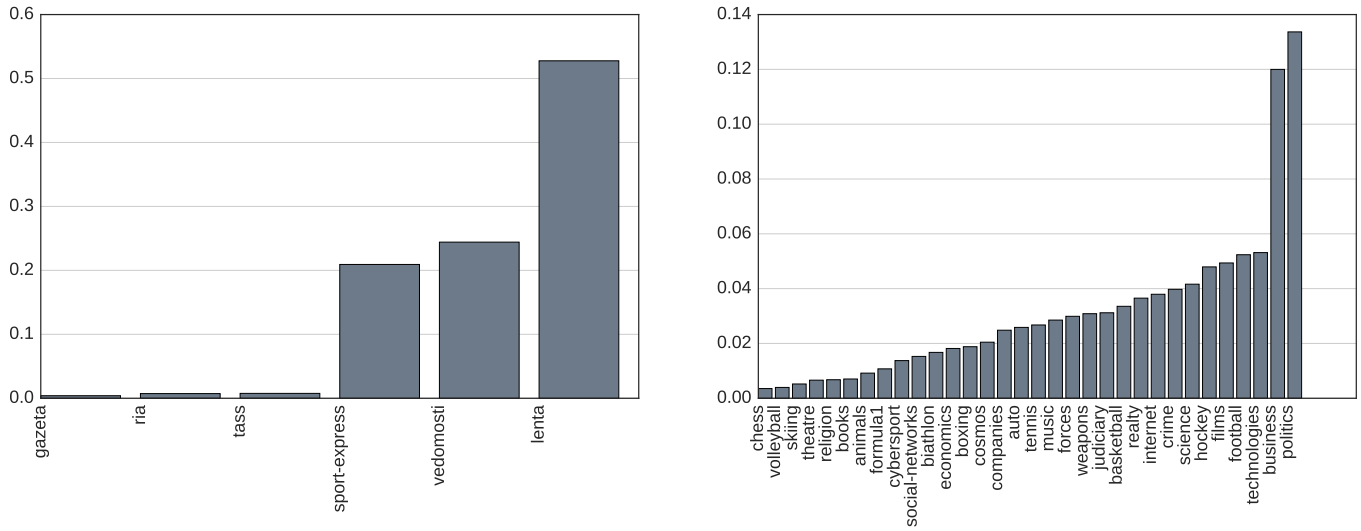


Рисунок 1 — Распределение СМИ и тем в данных

имения, предлоги и союзы. Лемматизация — приведение слова в начальную форму (лемма):

- для существительных — именительный падеж, единственное число;
- для прилагательных — именительный падеж, единственное число, мужской род;
- для глаголов, причастий, деепричастий — глагол в инфинитиве.

Часто вместо лемматизации используется стемминг — алгоритм, который убирает части слова, влияющие на его форму, например, окончание. В результате применения данной процедуры однокоренные слова, как правило, преобразуются к одинаковому виду. Данные алгоритмы не только опираются на словари, но и на определённые правила, зависящие от языка, так как в корпусе могут встречаться слова в разных формах, которых нет в словаре, например, неологизмы, но образованы они по правилам языка.

Процесс обработки текста в данной работе состоит из нескольких последовательных этапов:

а) Приведение текста в нижний регистр.

б) Удаление чисел и символов пунктуации. Дефис сохраняется.

в) Удаление стоп-слов. В данный набор входят наиболее часто употребляемые слова русского и английского языков, а также названия новостных агентств («лента», «тасс», «риа» и т.д.), сохранение которых приводит к переобучению моделей.

г) Лемматизация каждого слова с помощью библиотеки MyStem[1].

4 Классификация новостных статей

За последние два десятка лет, в результате активных исследований в области машинного обучения, было изобретено множество успешных алгоритмов классификации. Например, такие модели как support vector machines (SVM) [1], градиентный бустинг над решающими деревьями и нейронные сети [2], были успешно применены к задачам классификации текстов. В данной работе для классификации новостных статей использовались две из перечисленных выше модели — это SVM и градиентный бустинг над решающими деревьями.

Анализ текста является важной частью машинного обучения, однако сырые данные, а именно последовательности символов переменной длины, не могут быть переданы на вход алгоритму в явном виде т.к. большинство моделей ожидают численный вектор признаков фиксированной длинны.

Векторизация — метод трансформации коллекции текстовых документов в числовые вектора признаков. Существуют различные подходы к векторизации текста, в данной работе были применены два самых популярных: TF-IDF и word2vec.

4.1 Линейная модель на TF-IDF признаках

Одним из самых простых подходов к решению задачи классификации текстовых документов является обучение линейного классификатора на TF-IDF признаках, посчитанных на корпусе документов.

TF-IDF — статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса. [3] TF-IDF — это произведение двух статистик: TF (term frequency) и IDF (inverse document frequency). На сегодняшний день, TF-IDF один из самых популярных способов взвешивания слов, входящих в корпус документов. Например, 83% рекомендательных систем цифровых библиотек используют TF-IDF [4].

Существует множество способов подсчёта TF-IDF, в данной работе использовался следующий:

$$\mathrm{tf}(t, d) = \frac{n_t}{\sum_k n_k},$$

[1] https://tech.yandex.ru/mystem/

где n_t есть число вхождений слова t в документ, а $\sum_k n_k$ — общее число слов в данном документе.

$$\text{idf}(t, D) = \log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|},$$

где $|D|$ — число документов в корпусе, $|\{d_i \in D \mid t \in d_i\}|$ — число документов из коллекции D, в которых встречается t (когда $n_t \neq 0$).

Таким образом, мера TF-IDF является произведением двух сомножителей:

$$\text{TF-IDF}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D).$$

Признаковым описанием одного объекта $d \in D$ будет вектор

$$\big(\text{TF-IDF}(t,d,D)\big)_{t \in V},$$

где V — словарь всех слов, встречающихся в коллекции D.

Для преобразования новостных статей в числовые признаки, использовался класс `TfidfVectorizer` из библиотеки машинного обучения Scikit-learn [5]. В качестве параметров векторизации использовались следующие значения: `min_df=3` — учитываются слова, встретившиеся суммарно во всех документах минимум 3 раза и `ngram_range=(1,2)` — учитываются как отдельные слова, так и би-граммы.

После векторизации новостных статей, была получена разрежённая матрица размера $133\,529$ строк на $1\,025\,919$ столбцов. По причине большого количества признаков ($\gg$ обучающих примеров), в качестве классификатора было решено использовать линейный SVM. Не смотря на то, что данный алгоритм был впервые описан более пятидесяти лет назад, сегодня он по прежнему показывает одни из самых высоких результатов в задачах классификации текста. Как было показано в [6], особенно высокое качество удаётся получить при использовании би-грамм.

В данной работе в качестве SVM использовался `SGDClassifier` из библиотеки Scikit-learn. Данный класс реализует различные линейные классификаторы, параметры которых оптимизируются с помощью алгоритма стохастического градиентного спуска [7].

Классификатор обучался со следующими параметрами: `loss="hinge"` — функционал качества линейного SVM, `n_iter=70` — число итераций оптимизационного алгоритма, `alpha=1e-5` — коэффициент регуляризации.

Обучение классификатора происходило на 70% новостных статей, остальные 30% использовались для валидации. Качество классификации оценивалось с помощью метрик Accuracy и F1-меры с макро-усреднением по каждому классу. На валидационном множестве линейному SVM удалось получить accuracy $= 0.8792$ и F1 $= 0.8860$. Из матрицы ошибок, приведённой на рисунке 3, можно увидеть, что модель часто ошибается в классификации новостей на тему
«экономика», относя их к новостям про «бизнес» (в 20% случаев). Такая же ситуация характерна и для классов «компании» – «бизнес» (12%), «силовые структуры» – «оружие» (12%), «социальные сети» – «интернет» (15%), «театр» – «кино» (9%). Перечисленные ошибки являются, в основном, причиной семантической схожести классов, но зачастую одна и та же новость может содержать различные темы, поэтому, возможно, нужно было решать данную задачу методами multilabel классификации, т.е. когда один объект может принадлежать сразу нескольким классам. Напротив, среди всех тем выделяются несколько, которые классифицируются с высокой долей точности. Данные темы являются узкоспециализированными и почти не имеют пересечения с другими. В основном это виды спорта, такие как «футбол» (точность 0.99), «хоккей» (0.99), «формула-1» (1.00), «кибер-спорт» (0.98) и т.д. Более детальные значения метрик по каждому из классов можно найти на графике 4a.

Веса признаков в линейной модели в случае, если признаки отмасштабированы, характеризуют степень их влияния на значение целевой переменной. В задаче классификации текстов, кроме того, признаки являются хорошо интерпретируемыми, поскольку каждый из них соответствует конкретному слову, поэтому из линейного SVM были извлечены слова с наибольшим весом для каждой из 32-ух тем (таблица 1).

4.2 Градиентный бустинг над решающими деревьями. Word2Vec и тематическое моделирование.

Другим популярным подходом к векторизации текста является алгоритм Word2Vec. Реализация алгоритма опубликованная компанией Google [8] в 2013

году, основывается на однослойной нейронной сети, которая пытается «выучить» векторные представления слов (англ. word embeddings). До этого также были предложены различные архитектуры рекуррентных нейронных сетей, которые могли «выучивать» векторные представления, но их проблема заключалась в том, что они требовали намного больше времени для обучения, в отличии от реализации Word2Vec от Google.

Алгоритм Word2Vec является подмножеством более широкого класса алгоритмов, которые называются Vector space models (VSMs) [9]. VSMs представляют слова в непрерывном векторном пространстве, где семантически похожие слова отображаются в близкие точки. VSMs имеют долгую и богатую историю в NLP, но все представители этого семейства моделей опираются на дистрибутивную семантику, которая утверждает, что слова появляющиеся в одном и том же контексте имеют близкое семантическое значение. Все методы основанные на данной гипотезе можно разделить на два класса: статистические или численные (от англ. count-based, пр. Latent Semantic Analysis) и предиктивные модели (пр. neural probabilistic language models).

Первый класс алгоритмов вычисляет как часто слова возникают в контексте своих соседей в больших корпусах текстов. Затем происходит отображение вычисленных статистик в вектора для каждого слова. В свою очередь, предиктивные модели напрямую пытаются предсказать слова по его контексту, «выучивая» вектора таким образом, чтобы снизить ошибку предсказаний.

В частности, Word2Vec является вычислительно-эффективной реализацией предиктивной модели, которая «выучивает» векторные представления слов из необработанных текстовых данных. Есть две вариации алгоритма: Continuous Bag-of-Words (CBOW) и Skip-Gram. Алгоритмически они похожи, за исключением того, что CBOW предсказывает слово (например «раму») основываясь на контексте («мама мыла»), в то время как Skip-Gram наоборот, старается предсказать контекст по исходному слову. На практике CBOW лучше работает на небольших корпусах текстов, однако если корпус достаточно большой, как в данной работе, то рекомендуется использовать модель, основанную на Skip-Gram.

В данной работе, для обучения модели Word2Vec использовалась библиотека Gensim [10], в которой имеется обёртка над Word2Vec от Google для языка Python. Модель обучалась в течении 50 минут со следующими параметрами:

`min_count=3` — в обучении используются только те слова, которые встречаются во всём корпусе не менее трёх раз, `vec_size=300` — размерность векторов слов, `window=5` — размер окна (контекста), `sg=1` — использовать алгоритм Skip-Gram, `iter=10` — количество итераций по коллекции документов.

Результатом обучения Word2Vec является словарь, который отображает слова в вектора. Для представления новостной статьи в виде вектора признаков усреднним все векторы слов в данной статье с idf весами. В результате получим матрицу, в которой $133\,529$ строк и 300 столбцов (признаков), что значительно меньше, чем в случае TF-IDF. Благодаря тому, что удалось снизить размерность с $1\,025\,919$ признаков до 300, становится возможным эффективное использование таких мощных алгоритмов как градиентный бустинг над решающими деревьями.

Градиентный бустинг над решающими деревьями — один из самых универсальных и сильных методов машинного обучения, известных на сегодняшний день. В последнее время этот алгоритм и, особенно его реализация в библиотеке XGBoost [11], пользуются большой популярностью у участников соревнований в области анализа данных на платформе Kaggle.

В данной работе использовалась реализация градиентного бустинга из библиотеки LightGBM, которая является open-source проектом компании Microsoft. Алгоритм построения деревьев в LightGBM немного отличается от того, как это реализовано в XGBoost. Несмотря на это интерфейс библиотек и параметры алгоритмов зачастую полностью совпадают. Однако на практике LightGBM превосходит в скорости работы XGBoost, именно по этой причине эта библиотека использовалась для решения задачи классификации.

Модель градиентного бустинга над деревьями обучалась на векторах, полученных с помощью Word2Vec, в течении 26 минут, со следующими параметрами:

а) `colsample_bytree=0.9` — доля от общего числа признаков, используемая для построения дерева на каждой итерации;
б) `learning_rate=0.1` — шаг градиентного спуска, темп обучения;
в) `max_depth=7` — ограничение на максимальную глубину дерева;
г) `num_leaves=255` — ограничение на максимальное число листьев в дереве;
д) `objective="multiclass"` — функционал качества;

е) `metric="multi_error"` — метрика, по которой оценивается качество и происходит ранняя остановка;

ж) `subsample=0.8` — доля от общего числа объектов, используемых для построения дерева на каждой итерации.

Итоговое качество алгоритма: Accuracy $= 0.8427$, F1 $= 0.8486$.

Как видно из результатов, качество градиентного бустинга оказалось ниже линейного SVM, что, возможно, является причиной усреднения слов в документе, во время которого теряется достаточно много информации. Одним из решений данной проблемы является генерация новых признаков. Для этого в работе было применено тематическое моделирование.

Тематическое моделирование — активно развивающаяся ветвь статистического анализа текстов [12]. Тематические модели стараются раскрыть скрытую тематическую структуру коллекции документов и найти сжатое представление каждого документа в виде тем, которые в нём встречаются. Тематическое моделирование применяется в различных задачах, например, таких как информационный поиск, классификация, кластеризация, сжатие текста без потери информации (summarization) и т.д. Различные способы и идеи применения тематических моделей описаны в обзоре [13].

5 Агрегация новостных статей

В данной работе, в качестве меры семантической схожести двух новостей, используется косинусная мера. Пусть есть две новости, которые представлены в виде двух векторов x и y. Известно, что их скалярное произведение и косинус угла α между ними связаны следующим соотношением:

$$\langle x,y\rangle = \|x\|\|y\| \cdot \cos(\alpha).$$

Тогда, косинусное расстояние определяется как

$$\rho_{\cos}(x,y) = \arccos\left(\frac{\langle x,y\rangle}{\|x\|\|y\|}\right).$$

Косинусное расстояние часто используется для измерения схожести между текстами. Каждый документ описывается вектором, каждая компонента которого соответствует слову из словаря. В данной работе, компонента вектора — это TF-IDF

вес слова, если оно встречается в тексте, и ноль в противном случае. Тогда косинус между двумя векторами будет тем больше, чем больше общих слов в этих двух документах одновременно.

После кластеризации кластеры сортируются по убыванию среднего времени новостей в них. Это необходимо для того, чтобы новые статьи отображались выше старых.

5.1 Кластеризация с помощью алгоритмов машинного обучения

5.2 Объединение в связные компоненты графа

Данный алгоритм основан на алгоритме нахождения связных компонент в графе. Изначально строится полный граф, где вершины — векторное представление новостей, а вес рёбер равен косинусному расстоянию между вершинами. Рёбра, вес которых меньше определённого параметра, игнорируется, и запускается алгоритм поиска в ширину от первой вершины, не находящейся в кластере. Все вершины, по которым прошёл поиск, добавляются в кластеры и помечаются как посещённые, и начинается новый поиск от следующей непомеченной вершины. Такой поиск производится до тех пор, пока все вершины не станут находится в каком-либо кластере.

6 Разработка веб-приложения

Веб-приложение состоит из двух основных частей: back-end и front-end. Back-end – это сам веб-сервер, который осуществляет обработку запросов пользователей, получение и обработку данных. Front-end – это пользовательский интерфейс, визуализирующий полученные от back-end данные в понятный вид. С помощью этого интерфейса пользователь способен не только получать, но и передавать данные на back-end.

Back-end составляющая сервиса реализована на языке Python 3, с использованием библиотек: Flask — обрабатывает HTTP запросы от пользователей, pandas — обработка и хранение данных; sklearn — модели TFIDF, KMeans, SVM; pymystem3 — лемматизация.

Структура приложения показана на рис. 2.

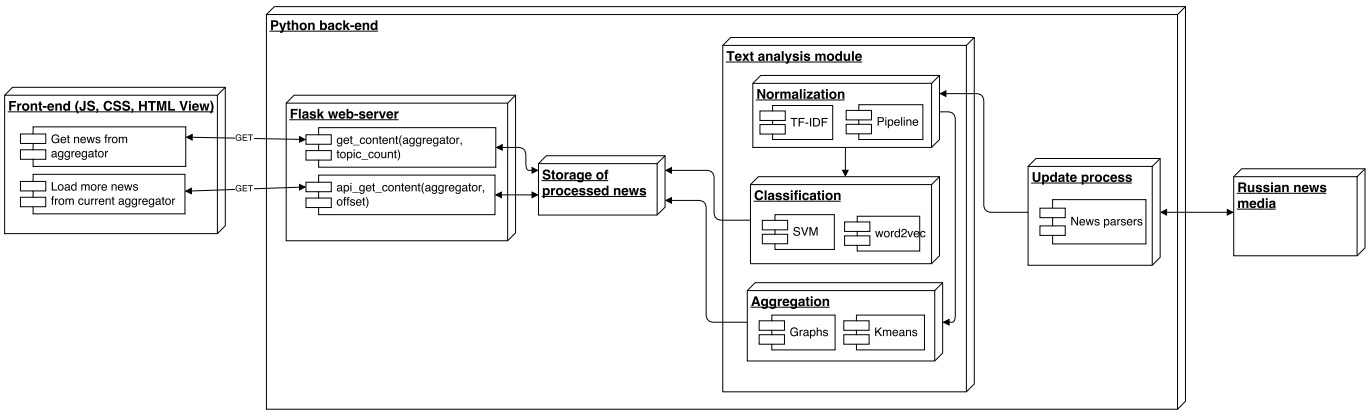


Рисунок 2 — Структура сервера

Back-end разделён на несколько модулей: Flask web-сервер, модуль парсеров новостей, которые с помощью параллельных процессов извлекают данные с сайтов СМИ, модуль анализа данных.

При запуске сервера происходит получение новостей за последние 12 часов и их обработка. После этого запускается отдельный процесс, отвечающий за актуальность данных: каждую минуту он проверяет наличие новых статей, и, если такие есть, отправляет их на обработку. При запросе от Front-end, сервер обращается к уже обработанным данным, хранящимся в кэше.

Основной частью сервера является класс Analizer, обрабатывающий полученные новостные статьи.

Методы класса:

– `Конструктор`. Параметры: список новостных статей с мета-данными.
  В конструкторе класса производится загрузка сохраненных моделей, определяются параметры моделей агрегации, вызываются функции функции `_data_to_pandas`, `_classify`, `_classify`, `_aggregate`, `_form_output`.
– `append_data`. Параметры: список новостных статей с мета-данными.
  Функция отвечает за добавление новых данных. Вызываются те же методы, что и в конструкторе. Дополнительно удаляются устаревшие данные.
– `_data_to_pandas`. Параметры: список новостных статей с мета-данными. Возвращаемое значение: `Pandas Dataframe`, в котором хранится вся информация о актуальных новостях.

Список новостей конвертируется в таблицу `Pandas Dataframe`, из которой удаляются дубликаты, лишние мета-данные, происходит сортировка по дате публикации статьи. Вызывается функция `_norimalize`.

- `_norimalize`. Параметры: `Pandas Dataframe`.
  Вызываются конвеер с функциями нормализации. К таблице добавляются нормализованные текст и заголовок статей.
- `_classify`. Параметры: `Pandas Dataframe`.
  Вызываются функции классификации. Для каждого классификатора реализована отдельная функция, в которой к таблице добавляются колонка с тегами, полученными от данного классификатора.
- `_aggregate`. Параметры: `Pandas Dataframe` и конфигурация агрегаторов.
  Вызываются функции агрегации: `_aggregate_Kmeans`, `_aggregate_graphs`.
  Для каждого агрегатора реализована отдельная функция, которые возвращают список кластеров, каждый кластер представляет собой список ID — индекс новости в таблице `Pandas Dataframe`.
- `_sort_groups`. Параметры: список кластеров, полученный от агрегаторов. Возвращает отсортированный список кластеров.
  Фильтрует кластеры, в которых меньше двух новостей, сортирует кластеры по среднему времени публикации, и сортирует новости внутри кластера по времени.
- `_get_avg_time`. Параметры: кластер. Возвращает среднее время в кластере.
  Преобразовывает даты публикации каждой новости класстера в Unix-секунды и считает от них среднее.
- `_get_TFIDF`. Параметры: `Pandas Dataframe`. Возвращает модель TF-IDF и матрицу векторизованных новостей.
- `_class_to_str`. Параметры: id класса. Возвращает название класса.
- `_cut_text`. Параметры: текст. Возвращает первый абзац текста.
- `_date_to_str`. Параметры: дата публикации. Возвращает дату в виде строки в определённом формате.
- `_form_output`. Параметры: `Pandas Dataframe`, список кластеров.

Формирует данные в формате JSON, используемым веб-сервером, и сохраняет их.

- `get_data`. Параметров нет. Возвращает готовые данные в формате JSON, используемы веб-сервером, и дату самой актуальной новости.

7 Заключение

Список литературы

1. *Weston Jason, Watkins Chris*. Support Vector Machines for Multi-class Pattern Recognition // Proc. European Symposium on Artificial Neural Networks. — 1999. — Pp. 219–224.

2. Very Deep Convolutional Networks for Natural Language Processing / Alexis Conneau, Holger Schwenk, Loïc Barrault, Yann LeCun // *CoRR*. — 2016. — Vol. abs/1606.01781. http://arxiv.org/abs/1606.01781.

3. *Jones Karen Sparck*. A statistical interpretation of term specificity and its application in retrieval // *Journal of Documentation*. — 1972. — Vol. 28, no. 1. — Pp. 11–21. http://dx.doi.org/10.1108/eb026526.

4. Research-paper recommender systems: a literature survey / Joeran Beel, Bela Gipp, Stefan Langer, Corinna Breitinger // *International Journal on Digital Libraries*. — 2016. — Vol. 17, no. 4. — Pp. 305–338. http://dx.doi.org/10.1007/s00799-015-0156-0.

5. Scikit-learn: Machine Learning in Python / F. Pedregosa, G. Varoquaux, A. Gramfort et al. // *Journal of Machine Learning Research*. — 2011. — Vol. 12. — Pp. 2825–2830.

6. *Wang Sida I., Manning Christopher D.* Baselines and Bigrams: Simple, Good Sentiment and Topic Classification // Proceedings of the ACL. — 2012. — Pp. 90–94.

7. *Bottou Léon*. Large-Scale Machine Learning with Stochastic Gradient Descent // Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers / Ed. by Yves Lechevallier, Gilbert Saporta. — Heidelberg: Physica-Verlag HD, 2010. — Pp. 177–186. http://dx.doi.org/10.1007/978-3-7908-2604-3_16.

8. Efficient Estimation of Word Representations in Vector Space / Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean // *CoRR*. — 2013. — Vol. abs/1301.3781. http://arxiv.org/abs/1301.3781.

9. *Salton G., Wong A., Yang C. S.* A Vector Space Model for Automatic Indexing // *Commun. ACM*. — 1975. — nov. — Vol. 18, no. 11. — Pp. 613–620. http://doi.acm.org/10.1145/361219.361220.

10. *Řehůřek Radim, Sojka Petr*. Software Framework for Topic Modelling with Large Corpora // Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. — Valletta, Malta: ELRA, 2010. — May. — Pp. 45–50. — http://is.muni.cz/publication/884893/en.

11. *Chen Tianqi, Guestrin Carlos*. XGBoost: A Scalable Tree Boosting System // *CoRR*. — 2016. — Vol. abs/1603.02754. http://arxiv.org/abs/1603.02754.

12. *Blei David M.* Probabilistic Topic Models // *Commun. ACM*. — 2012. — apr. — Vol. 55, no. 4. — Pp. 77–84. http://doi.acm.org/10.1145/2133806.2133826.

13. Knowledge discovery through directed probabilistic topic models: a survey / Ali Daud, Juanzi Li, Lizhu Zhou, Faqir Muhammad // *Frontiers of Computer Science in China*. — 2010. — Vol. 4, no. 2. — Pp. 280–301. http://dx.doi.org/10.1007/s11704-009-0062-y.

Приложение

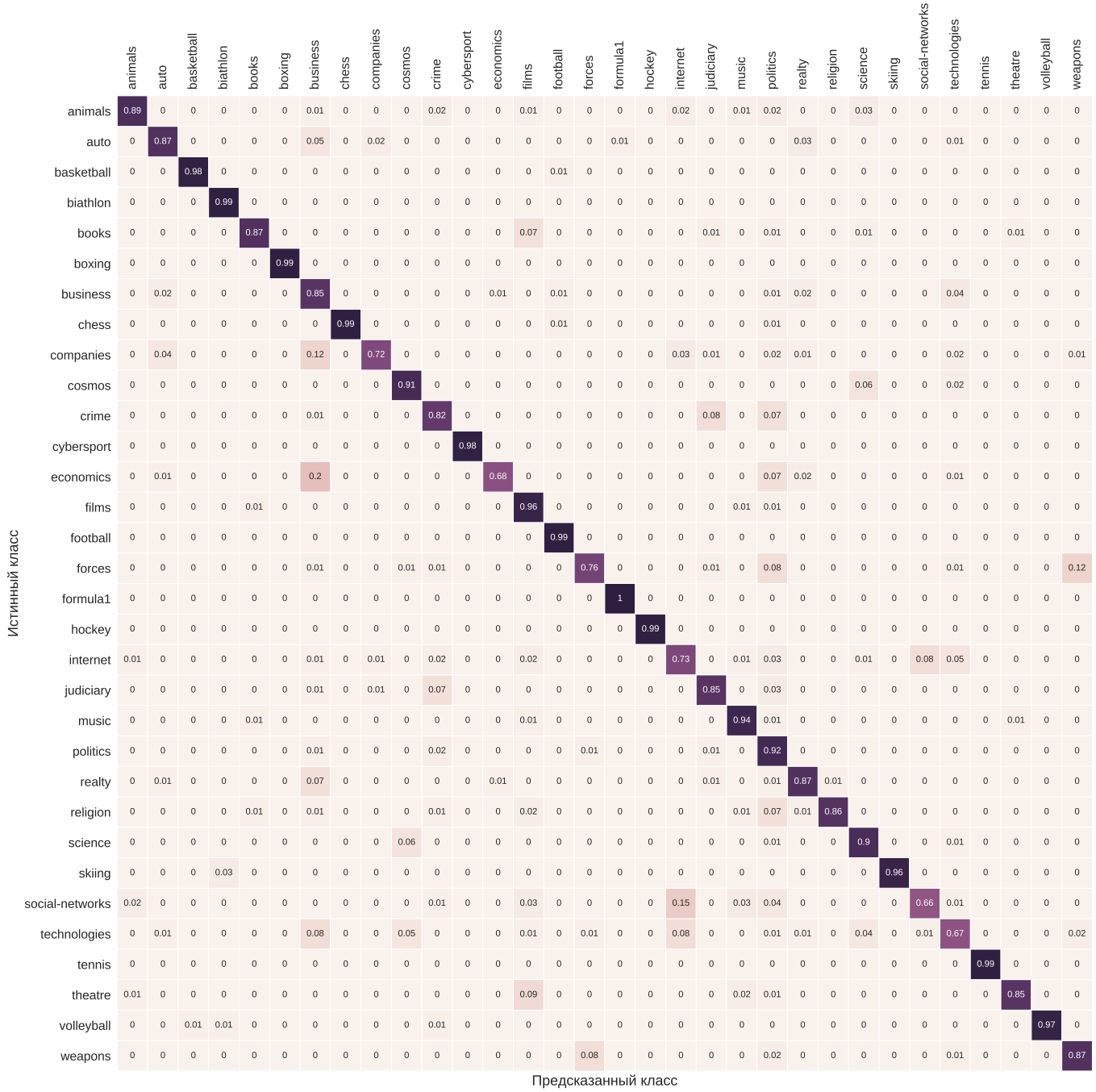
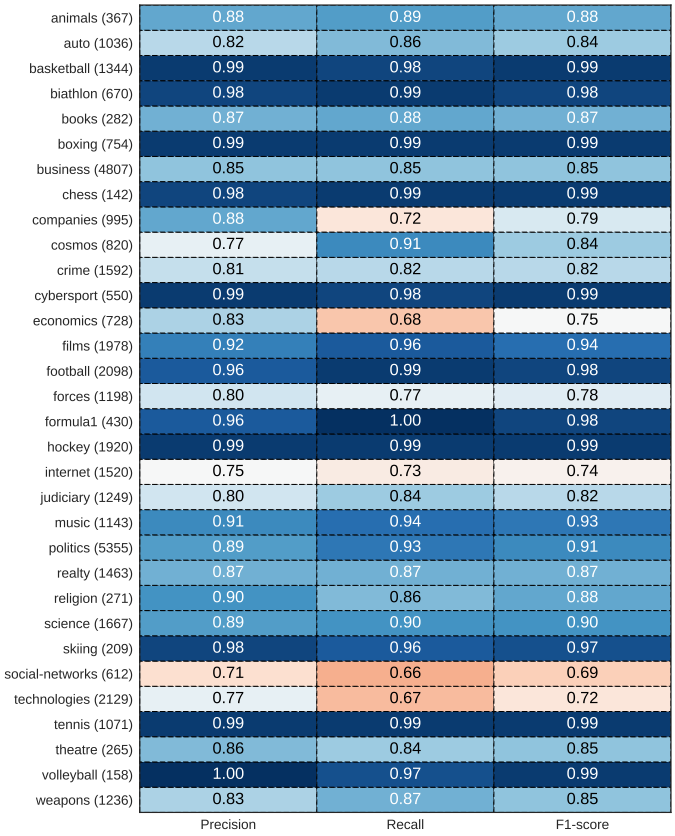


Рисунок 3 – Матрица ошибок линейного SVM



а) Линейный SVM б) LightGBM + Word2Vec + bigARTM

Рисунок 4 — Precision, Recall и F1 мера по каждому из классов

animals	жить	вольер	хозяин	животный	зоопарк	питомец	кличка	животное
auto	авторынок	осаго	автомобильный	автопроизводитель	автопром	камаз	автоваз	автомобиль
basketball	рфб	евробаскет	центровой	кубок европа	баскетбол	баскетболист	евролига	нба
biathlon	эстафета	биатлонистка	шипулин	сбр	хохфильцен	биатлон	ibu	биатлонист
books	библиотека	произведение	писательница	роман	книга	литературный	поэт	писатель
boxing	алоян	лебзяк	мма	поединок	поветкин	бой	бокс	боксер
business	россельхознадзор	fifa	ржд	газпром	туроператор	formula	ритейлер	оао
chess	карякин	шахматист	карякина	магнус	шахматы	карлсен	фид	шахматный
companies	тысяча автомобиль	компания	миллиард кубометр	миллиард	тысяча	процент акция	ретейлер	процент
cosmos	космонавт	светить	прогресс	вселенная	космос	астрофизик	марс	астронавт
crime	грабитель	изымать	группировка	полиция	убивать	летний	преступник	тюрьма
cybersport	gaming	team	valve	киберфутбол	dota	киберспорт	киберспортивный	киберспортсмен
economics	мрот	греция	бюджет	пенсия	ввп	минфин	экономика	инфляция
films	мультфильм	сериал	актриса	кино	картина	режиссер	актер	фильм
football	поле	стадион	нападающий	матч тур	фифа	уефа	футболист	полузащитник
forces	развертывать	выполнение	военнослужащий	военный	шойгу	генштаб	конашенков	минобороны
formula1	манор	цитировать	рено	феррари	макларен	пилот	мерседес	формула
hockey	авангард	ска	шайба	нападающий	хоккей	хоккеист	нхл	кхл
internet	википедия	сервис	ресурс	youtube	сайт	хакер	блогер	интернет
judiciary	стража	статья	арестовывать	колония	следственный комитет	следствие	скр	комитет россия
music	композитор	евровидение	песня	певец	концерт	альбом	певица	музыкант
politics	лидер	кремль	парламентарий	партия	депутат	госдума	глава	мид
realty	объект	строительный	жилищный	строительство	жкх	ипотека	жилье	недвижимость
religion	монастырь	собор	церковный	муфтий	святой	христиан	митрополит	патриарх
science	университет	журнал	математик	физик	научный	археолог	исследователь	ученый
skiing	вяльбе	нортуг	лыжник	fis	легков	йохауг	лахти	устюгов
social-networks	некоторые	юзер	facebook	twitter	пользователь сеть	пользователь	вконтакте	соцсеть
technologies	vimpelcom	apple	оператор	wifi	говорить	мтс	робот	контакт
tennis	кубок федерация	open	шарапов	теннис	корт	теннисист	теннисистка	кубок дэвис
theatre	росгосцирк	цирковой	балет	постановка	театральный	мюзикл	театр	спектакль
volleyball	факел	маричев	алекно	суперлига	казанский	белогорье	волейболист	волейбол
weapons	jane	использоваться	defense news	министерство оборона	миллиметр	миллиметровый	defense	тип

Таблица 1: Слова с наибольшим весом по каждой теме (веса линейного SVM)