Правительство Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Национальный исследовательский университет
«Высшая школа экономики»

Отделение программной инженерии

Кафедра Управления разработкой программного обеспечения

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по направлению 231000.68 Программная инженерия
подготовки магистра

На тему «Система комплексного анализа русскоязычных текстовых сообщений на платформе *IBM InfoSphere Streams*»

Студента 271мУРПО _____ Карнаухов М.В.
Подпись (Ф.И.О.)
06.06.2014
(Дата)

Научный руководитель
проф., к.т.н. _____ Авдошин С. М.
(должность, звание) Подпись (Ф.И.О.)
06.06.2014
(Дата)

Москва, 2014

Аннотация

Разработана система анализа русскоязычных текстовых сообщений на платформе IBM InfoSphere Streams. Система применяется для категоризации и сентимент анализа потребительских отзывов о Сбербанке России, его услугах и продуктах. Система прошла приемно-сдаточные испытания и соответствует ожиданиям заказчика.

Проведен сравнительный анализ современных средств и методов контент-анализа текстовых сообщений, работающих как с русскоязычными сообщениями, так и с английским языком. Для решения задачи выбраны методы предварительной обработки и классификации. Обучены наивные байесовские классификаторы для решения задачи определения тональности потребительских отзывов, определения категории, к которой принадлежит отзыв (банкоматы, мобильный банкинг, Сбербанк Онлайн) и характеристик упомянутых в сообщении (доступность сервисов, качество обслуживания, скорость обслуживания).

Помимо разработанных операторов, в среду InfoSphere Streams интегрированы сторонние средства статистического анализа (Weka) и автоматический обработки естественного языка (АОТ). Сохранен принцип модульности и модифицируемости системы. Выбор сторонних средств обусловлен предварительно проведенными исследованиями и сравнительным анализом. Достигнуты показатели точности: 82%, 80%, 77% для определения тональности, продукта и характеристики соответственно.

Для улучшения качества анализа предложена методика формирования признаков как нормализованных синтаксических групп, что привело к улучшению точности предсказания в среднем на 5-7%. Исследован и адаптирован для русского языка RNN классификатор, обученный на банке деревьев синтаксического разбора. Выявлены преимущества метода и его недостатки.

Выпускная квалификационная работа состоит из трех частей: обзорно-постановочной, теоретической и практической.

Объем работы: 104 страницы, 35 иллюстраций и 18 таблиц. Использовано 50 источников.

Ключевые слова: *text mining, анализ тональности, классификация текстовых сообщений, наивный байесовский классификатор, машинное обучение, рекурсивные нейронные сети, деревья синтаксического разбора*.

Оглавление

Введение

Человек в информационном обществе – постоянный источник и приемник информации: в процессе образования, в рамках трудовой деятельности, как участник экономических и торговых взаимоотношений, как обладатель нескольких персональных страниц в социальных сетях. Будет ли он рационально применять полученные знания или утонет в океане ненужных фактов – решает каждый сам, ведь ресурсы сети Интернет практически безграничны и противоречивы. Мировое исследовательское сообщество ежегодно издает свыше полутора миллионов статей. Крупнейшие социальные сети *Facebook* и *Twitter* ежедневно пополняются миллиардом пользовательских записей [1,2]. Ежесекундно совершаются миллионы валютных сделок по всему миру, влияющих на котировки на бирже. Все эти данные несут в себе неограниченный скрытый потенциал, который возможно реализовать, используя *data mining* и *text mining.* Корпорации применяют подобный анализ для изыскания новых целевых рынков и повышения конкурентоспособности; фармацевтическая индустрия изучает патенты и исследовательские статьи для способствования открытию новых лекарственных средств; в академической среде анализ больших массивов данных способствует появлению междисциплинарных областей, таких как биоинформатика [3].

Обнаружением скрытых зависимостей и извлечением полезных сведений из больших объемов информации занимаются специалисты области *data mining*. Старейшая и самая крупная международная организация ACM (*Association of Computing Machinery,* Ассоциация Вычислительной техники) основала 37 SIG (*Special Interest Group*) – специальных групп по интересам, каждая из которых проводит исследования в определенной области информационных технологий. В 1998 году, в связи с растущей популярностью области, была создана SIGKDD (Special Group on Knowledge Discovery in Databases). Целями подразделения являются: ведение постоянных разработок в области анализа данных, обобщение вводимых методологий и терминологий, междисциплинарное обучение среди членов группы KDD, в том числе во время ежегодных конференций (*Conference on Knowledge Discovery and Data Mining*)[4]. *Data mining* включает в себя множество подотраслей – наибольший интерес сейчас представляют потоковый анализ (*data stream mining*) и текстовый анализ (*text mining*). Анализ потоков данных применяется

при обработке больших данных (*Big data*), позволяющий исследовать огромные объемы данных. В то время как популярность текстового анализа связана с возможностью обработки постоянно растущих объемов информации на естественном языке.

Анализ текстовой информации, или *text mining*, сочетает в себе подходы разнообразных областей знаний, находясь на стыке таких дисциплин, как машинное обучение, компьютерная лингвистика и программная инженерия. Целью его является получение структурированной информации из неструктурированного текста. Основная часть этого процесса посвящена извлечению объектов, их отношений и свойств из текста.

Задачи, которые решает text mining включают в себя нахождение шаблонов данных (*pattern recognition*), получение структурированной информации, построение иерархий объектов, классификация и кластеризация данных, определение тематики или области знаний, автоматическое реферирование документов, задачи автоматической фильтрации контента, определение семантических связей и другие.

Анализ текстовой информации используется для исследований по всему миру. Крупнейшие центры находятся в Великобритании и США [3]. Национальный центр по анализу текста Великобритании (NaCTem –*The National Centre for Text Mining)* – первый подобный центр, финансируемый из государственного фонда и действующий под эгидой Университета Манчестера. Центр предоставляет услуги *text mining* на запросы академического содружества Англии[5]. Список образовательных центров, которые используют методы анализа текста для своих исследований, включает в себя такие университеты, как Стэнфорд, Кембридж, Оксфорд, Королевский колледж Лондона, Университет Ливерпуля, Университет Калифорнии.

Основные сферы применения *text mining* в научно-академической сфере:

- Систематические обзоры литературы: анализ данных служит для автоматического отбора литературы, которую необходимо осветить исследователю, описывающему текущее состояние какой-либо предметной области. К примеру, Дж. Томас и А.О’Мара-Ивс показали в своей работе[6], что с текстовым анализом обработка источников литературы занимает лишь 25 % от времени, которые было бы потрачено при отборе вручную;

- Выдвижение новых гипотез: зачастую публикации из несопоставимых областей при текстовом анализе оказываются взаимосвязаны по средствам промежуточных связей, дальнейшее изучение которых может привести к появлению новых теорий. Например, в биомедицине Свансон использовал подходы text mining для построения гипотез о том, как уже существующие лекарственные средства могут влиять на различные болезни от болезни Рейнада до Альцгеймера [7];
- Проверка существующих гипотез: анализ архивов и коллекций данных может быть применен при проверке давно составленных гипотез. В частности, текстовый анализ оцифрованной корреспонденции эпохи Просвещения поставил под вопрос широко известное мнение о влиянии Англии на Францию в данный период истории [8];
- Оценка качества текстов: может быть применена в образовании для проверки лексических и синтаксических структур педагогических материалов на соответствие стандартам, или, как предложили японские ученые[10], для определения сложности текстов, используемых при изучении иностранных языков;
- Получение выборок для исследований: выгрузка информации социальных сетей, которая изначально представлена в форме, неструктурированной для анализа человеком, позволит расширять базу для исследований социальных и экономических событий. Например, анализ сообщений в *Twitter*, появившихся в период беспорядков в Великобритании в 2011-м году доказал непричастность социальных сетей к их развитию. [11].

Text mining применяет методы информационного поиска (*information retrieval*) и лексического анализа для получения машинного представления информации. Ключевым этапом преобразования неструктурированных данных в структурируемые является обработка естественных языков.

Обработка естественного языка (*NLP – Natural Language processing*) – междисциплинарная подотрасль области изучения искусственного интеллекта (ИИ), направленная на разработку и построение вычислительных машин, которые могут взаимодействовать с людьми на одном уровне. NLP представляет собой совокупность подходов и вычислительных техник для обработки и представления естественных текстов

и речи на одном или нескольких уровнях лингвистического анализа с целью применения результатов в различных областях.

Последнее десятилетие может считаться расцветом области обработки естественных языков и интеллектуального анализа данных. Развитие информационных технологий позволило расширить спектр задач по поиску, преобразованию, анализу и представлению речевой и другой неструктурированной информации. Сегодня NLP успешно применяется для текстового поиска, извлечения фактов, синтеза и распознавания речи или рукописных текстов.

Фактором, повлиявшим на становление развития систем Искусственного Интеллекта, стал значительный прыжок в вычислительных мощностях, доступных для исследователей. Если в 1960е-1980е разработки приостанавливались по причине нерентабельных временных затрат [12], и такие исследования, как многослойные нейронные сети, NLP, глубокое машинное обучение ушли в тень, то в двадцать первом веке, с развитием аппаратных ресурсов, эксперименты в данных областях возобновились с новой силой.

Глубокое обучение (чаще употребляется английский термин «deep learning») – область машинного обучения, в которой рассматриваются алгоритмы обучения, основанные на многоуровневых/многослойных моделях. Данные алгоритмы имитируют деятельность нейронов неокортекса или новой коры головного мозга, в которой осуществляется более 80% всех мыслительных процессов. Исследования в области *deep learning* начали активно развиваться с 2006 года, после публикации работы Хинтона «*A fast learning algorithm for deep belief nets*», в которой был представлен жадный алгоритм, позволяющий обучать многослойные байесовские сети [13].

За последние несколько лет работа над алгоритмами многоуровневого обучения привела к значительным результатам во многих реальных задачах (распознавание изображений, классификация текстов, распознавание речи, предсказание временных рядов, моделирование естественных изображений и др.). Одним из ярких примеров являются исследования в университете Стэнфорд в области NLP с применением подходов *deep learning* [14]. В результате исследований был разработан инструмент для автоматического анализа тональности текстов. Объектом исследований изначально

являлся лишь английский язык, но уже сейчас возможен анализ арабского, китайского, французского и немецкого языков.

В отличие от мировой компьютерной лингвистики, российская наука пока делает только первые шаги в направлении сравнения и оценки систем автоматического анализа текстов. В течение нескольких лет в рамках конференции «Диалог» проводятся форумы по оценке систем автоматической обработки текста [15]. Форум посвящен независимой оценке методов и алгоритмов лингвистического анализа разного уровня, ориентированных на работу с русскоязычной информацией. С 2010-го года были проведены состязаний в следующих областях:

- 2010 г.: проводилось соревнование систем автоматического *морфологического* анализа русского языка (систем, которые умеют делать грамматический разбор слов).
- 2011 г.: было проведено состязание *синтаксических* анализаторов (парсеров), результаты которого были представлены на конференции в 2012 году.
- 2012 г.: подвели итоги первых соревнований по *анализу тональности* текстов (*sentiment analysis*) на материале русского языка.
- 2013 г.: проводились состязания по *машинному переводу и анализу тональности*.

Таким образом, область автоматического анализа русскоязычных текстов активно развивается в последние годы. Разнообразие методов и сложная структура русского языка значительно расширяют спектр задач для анализа и исследований. Перспективы развития отрасли нужно связывать с передовыми инновационными решениями. Среди них находятся подходы глубокого машинного обучения. Использования алгоритмов *deep learning*, таких как многослойные нейронные сети, может стать следующим шагом в становлении русскоязычной компьютерной лингвистики.

Обзорно-постановочная часть

Цель работы

Целью работы является разработка системы комплексного анализа русскоязычных текстовых сообщений на платформе *IBM InfoSphere Streams*. Под русскоязычными текстовыми сообщениями понимаются короткие сообщения социальных сетей, форумов, блогов и микроблогов, написанные преимущественно на русском языке. В сообщениях могут встречаться имена собственные, англоязычные выражения, гиперссылки и эмотиконы. Как правило, подобные сообщения состоят из одного или нескольких предложений.

Понятие комплексного анализа, с точки зрения text mining, связано с возможностью решения широкого спектра задач: кластеризация, классификация, анализ тональности текста (частный случай классификации), извлечение ключевых словосочетаний, имен собственных, различных паттернов, решение задач информационного поиска. И этот список не является полным. Однако рассматривать разрабатываемую систему можно не только с точки зрения задач text mining, но и с позиции предметной области, а также в разрезе методов, которые можно реализовать в системе.

Построение систем анализа текстовой информации с учетом применяемой предметной области приводит к повышению качества анализа. Очевидно, что в банковской сфере и в медицине используется различная терминология, имеют место и различия в частотных словарях указанных областей, строении предложений, оборотах.

Также система комплексного анализа должна позволить применять различные методы text mining. Другими словами, для реализации и внедрения нового алгоритма кластеризации архитектура и модель данных не должны претерпевать коренных изменений.

Перечисленные признаки подталкивают к необходимости реализации модульной системы, которая будет устойчива к изменениям и доработкам.

Достижение цели возможно при правильном проектировании системы, построении модели данных, внимательном выборе средств разработки и сторонних компонентов.

Таким образом, целью работы является не только реализация системы, но и глубокий анализ существующих алгоритмов и методов text mining, сравнительный анализ существующих решений, проектирование системы и модели данных.

Задачи работы

Исходя из цели проекта, можно сформировать список задач и подзадач, в который входят как аналитические работы, так и задачи проектирования и разработки.

1. Обзор существующих систем анализа текстовой информации, выявление критериев сравнения систем и их сравнительный анализ.
    a. Графематические системы.
    b. Системы выявления морфологических признаков и канонизации текста.
    c. Системы анализа синтаксических взаимосвязей и синтаксического парсинга.
    d. Системы анализа тональности текста
    e. Интегрированные пакеты, предоставляющие возможность анализа текста на разных уровнях языка.
2. Обзор способов представления текстовой информации и методов, работающих с этими представлениями
    a. Текст в неструктурированной форме – raw text
    b. Векторная модель данных – *Vector Space Model (VSM)*
    c. Вероятностная векторная модель данных – *Vector Semantic Spaces*
    d. Деревья синтаксического разбора
3. Обзор современного состояния методов классификации и кластеризации текстовой информации.
    a. K-means, K-medoids, X-means
    b. Наивный Байесовский классификатор (Naïve Bayes)
    c. K-NN кластеризация
    d. Иерархические кластеризации (аггломеративная и дивизимная)

e. Нейронные сети: многослойные персептроны, рекурсивные нейронные сети, нейронные сети глубокого обучения (Deep Learning)

4. Выгрузка и составление обучающей и тестовой выборок.
5. Сравнительный анализ и реализация методов предварительной обработки текста.
    a. Алгоритмы канонизации текста: стемминг, лемматизация
    b. Алгоритмы определения морфологических признаков
    c. Синтаксические парсеры
6. Проектирование модели данных, компонентов системы и их взаимосвязей.
    a. Выбор программных средств
    b. Проектирование БД
    c. Проектирование архитектуры системы с учетом специфики IBM Info Sphere Streams;
7. Реализация спроектированного решения на платформе IBM Info Sphere Streams.
8. Анализ полученных результатов. Улучшение решения.

Таким образом, работа состоит из нескольких частей, которые включают обзорно-постановочную, теоретическую и практическую составляющие.

Уровни представления естественного языка

Естественный язык – система, которая может быть разложена и проанализирована на разных уровнях (Рис. 1)

Рисунок 1 Представление ярусов языков в виде иерархии

Процесс анализа текстовой информации точно также можно представить в виде уровней (Рис.2).

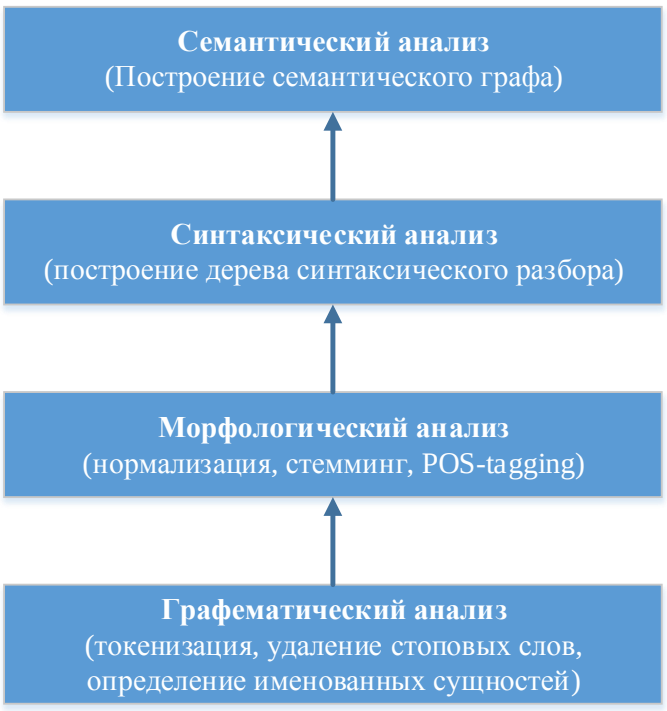


Рисунок 2 Процесс анализа текстовой информации

Сложность анализа текста возрастает по мере повышения уровня языка. Анализ на верхнем уровне невозможен без ранее проведенного анализа на предыдущих уровнях. Например, проведение синтаксического анализа (построение дерева синтаксического разбора) невозможно без морфологического разбора. В свою очередь, определение семантических связей невозможно без ранее построенного дерева синтаксического разбора.

Современные средства анализа текстовой информации можно разделить на два большие категории:

1. *Специализированные средства* – инструменты, для проведения анализа на конкретном уровне языке (морфологические анализаторы, синтаксические парсеры и т.д.);
2. *Интегрированные пакеты* – программные средства, предоставляющие инструменты для анализа текста на разных уровнях.

Фонетический анализ

На фонетическом уровне, как правило, сложно проводить какие либо виды анализа текста. Это связано с тем, что элементарная частица на этом ярусе, фонема, не несет в себе никакой смысловой информации. Стоит отметить, что в лингвистике существует спорное направление *фоносемантика*. Дисциплина находится на стыке *фонетики* (план выражения) и *семантики* (план содержания), которое предполагает, что звуки могут нести смысл сами по себе. Подобные предположения высказывались неоднократно. Например, Михаил Ломоносов писал, что «твердые звуки *к, п, т* имеют тупое произношение и в них нет ни силы, ни сладости».

Фоносемантическое средство анализа текста ВААЛ [16] основывается на схожих предположениях советского ученого А.П. Журавлева. Программа оценивает текст с позиции содержания в нем звуков русской речи и их характеристик. Анализ происходит сразу по 23 шкалам. Например, «*громкий - тихий*» или «*безопасный - страшный*». Подобный анализ может быть полезен для составления текста ораторских выступлений, где выразительная составляющая послания зачастую может быть важнее его содержательной части.

Помимо задачи фоносемантического анализа, экспертная система ВААЛ решает задачу контент-анализа. В качестве категорий выступают семантические примитивы А. Вежбицкой. Например, «думать» (обработка) или «хорошо» (позитив).

Психолингвистическая система является платной. Стоимость пакета фоносемантического анализа – 350 евро. Пакеты контент анализа – 450 евро, 450 евро и 400 евро для английского, русского и украинских языков соответственно.

Достаточно сложно оценить качество анализа средства. Например, слова, которые обычно вызывают привычные ассоциации, приобретают иную окраску (Рис. 2). Так например, слово честь по результатам анализа «производит впечатление чего-то шероховатого, угловатого, низменного, …».

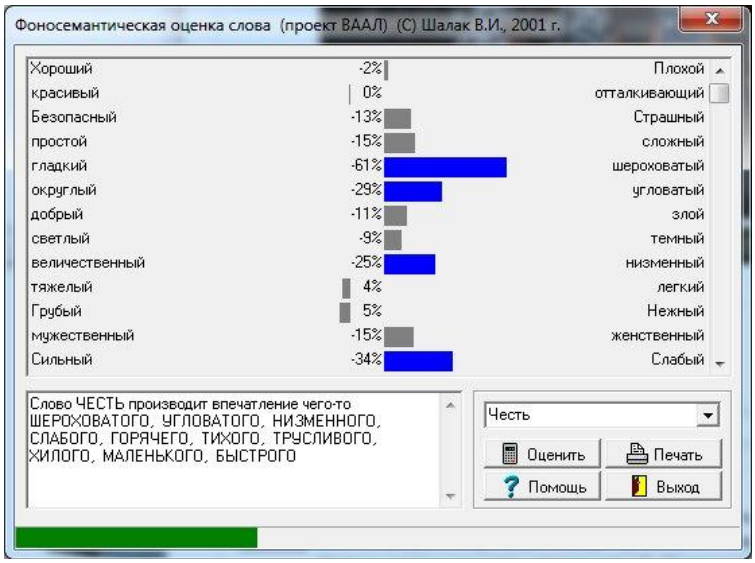


Рисунок 3 Фотосемантическая оценка слова «честь». ВААЛ

Таким образом, анализ на фонетическом уровне не стоит рассматривать как задачу text mining. Во-первых, из-за сложности интерпретации результата, так как восприятие тех или иных звуков может отличаться у индивидов. Во-вторых, из-за спорной применимости методов к анализу печатного текста: в случае анализа звука очень важно слуховое восприятие, интонация, произношение и прочее.

Графематический анализ

Для того, чтобы начать морфологический анализ текста требуется разделить исходный неструктурированный текст на предложения и слова. Эта, на первый взгляд, простая задача имеет свои особенности и играет важную роль при дальнейшем анализе текста.

Графематический анализ включается в себя:

- разделение исходного текста на элементы (слова, разделители);
- удаление нетекстовых элементов (теги, метаинформация);
- выделение и оформление нестандартных элементов:
  - структурные элементы: заголовки, абзацы, примечания;
  - числа, даты, буквенно-цифровые комплексы;
  - имена, отчества, фамилии;
  - элементы форматирования: курсив, подчеркивание, жирность;

- выделение электронных адресов;
- выделение имен файлов;
- выделение устойчивых оборотов, слова который не употребляются раздельно друг от друга;

В англоязычных источниках можно встретить определение *tokenization* (токенизация), которое, по своему содержанию, схоже с графематическим анализом. Токенизация – это процесс разбиение текстового потока (text stream) на токены: слова, словосочетания и предложения. [37]

Таким образом, графематический анализ является начальным анализом неструктурированного текста, представленного в виде цепочки символов в какой-либо кодировке, вырабатывающий информацию, необходимую для дальнейшей обработки текста.

Средства, специализирующиеся исключительно на графематическом анализе практически отсутствуют. В основном, графематика включена в интегрированные пакеты анализа текста: *NLTK, Stanford CoreNLP, Apache NLP, AOT, MBSP* и проч. Также функционал разбиения на токены включен в программы разметки текста, например, в part-of-speech taggers. В *табл. 1* приведены некоторые из средств, решающих эту задачу.

Таблица 1

Программы для графематического анализа

Название	**Метод**	**Языки**	**Лицензия**	**Платформа**
Tokenizer	правила	русский, английский, немецкий	GPL	C
Greeb	регулярные выражения	русский, английский	MIT	Ruby
Twitter NLP and Part-of-Speech Tagger	машинное обучение	Английский	GPL	Java
Lemmatizer	словарный	русский, английский	GPL	GNU/Linux

В большинстве случаев, задачу разбиения можно решить тривиальным способом: используя словарь разделителей и словарь устойчивых выражений. Помимо этого, задачу можно решить с помощью регулярных выражений.

Морфологический анализ

Морфологический анализ обеспечивает определение нормальной формы, от которой была образована данная словоформа, и набора параметров, приписанных данной словоформе [18]. В *табл. 2* представлены средства для анализа русскоязычных текстов.

Таблица 2

Программы для морфологического анализа русского языка

Название	**Методы**	**Лицензия**	**Платформа**	**Console**	**API**	**Модуль-ность**	**Стоимость (ком.лиц.)**
Свободно распространяемое							
AOT	словарный	LGPL	GNU/Linux, Microsoft Windows	-	+		-
MAnalyzer	словарный	MIT	GNU/Linux	-	-	Библиотека	-
Myaso	алгоритм Витерби	MIT	Ruby	-	+	Библиотека	-
mystem	словарный	Некоммерческая	GNU/Linux, Microsoft Windows	+	+		-
phpmorphy	словарный	LGPL	PHP	-	+	Библиотека	-
Pullenti SDK	н/д	Условно бесплатная	.NET	-	+	модуль SDK	100 000 руб
pymorphy	словарный	MIT	Python	-	+	Библиотека	-
RussianMorphology	словарный	Apache License	Java	-	+	Библиотека	-
RussianPOSTagger	словарный	GPL	Java	+	+	модуль GATE	-
Snowball	алгоритм Портера	BSD	GNU/Linux, Microsoft Windows	+	+		-
Stemka	словарный	Собственная	GNU/Linux, Microsoft Windows	+	+		-
SVMTool	метод опорных векторов	LGPL	Perl	-	+	Библиотека	-
TreeTagger	деревья принятия решений	Некоммерческая	GNU/Linux, Microsoft Windows	+	+		-
FreeLing	словарный	Условно платная	GNU/Linux	-	+	Библиотека	
Проприетарное							
RCO	словарный	Коммерческая	Microsoft Windows	-	+	Пакет для СУБД Oracle RCO	от 35 000 руб

Solarix	словарный	Коммерческая	GNU/Linux, Microsoft Windows	+	+	модуль SDK Грамматического Словаря Русского Языка	н/д
Морфер	словарный	Коммерческая	Microsoft Windows, Веб-сервис	+	+	Web-сервис/ Библиотека	от 50 000 руб
ОРФО	словарный	Коммерческая	Microsoft Windows	-	+	Библиотека	н/д

Snowball

Самым распространенным алгоритмом стемминга является *алгоритм Портера (реализация Snowball)*, который был разработан Мартином Портером в 1979 г. Основная идея стеммера Портера заключается в том, что существует ограниченное количество формо- и словообразующих суффиксов, и стемминг слова происходит без использования каких-либо баз основ.

В первоисточнике указывается, что стеммер Портера способствует снижению размерности словаря на одну треть [17].

Stemka.

Работа стеммера основана на вероятностном подходе: слова из обучающего текста разбираются анализатором на пары «*последние две буквы основы*» + «*суффикс*» и если такая пара уже присутствует в модели – увеличивается ее вес, иначе она добавляется в модель. После чего полученный массив данных ранжируется по убыванию веса и модели, вероятность реализации которых составляет менее 1\10000, отсекаются. Результат - набор потенциальных окончаний с условиями на предшествующие символы - инвертируется для удобства сканирования словоформ "справа налево" и представляется в виде таблицы переходов конечного автомата. При разборе слово сканируется по построенным таблицам перехода [19].

Конкурентным преимуществом средства stemka является нечеткое определение стеммы. Иными словами, программа выдает несколько вариантов, в порядке убывания вероятности.

Mystem

Программа Mystem производит морфологический анализ текста на русском языке. Для слов, отсутствующих в словаре, порождаются гипотезы.

В основе работы программы лежит построение инвертированных префиксных деревьев суффиксов и основ с использованием словаря, содержащего в себе все грамматические формы слова.

На первом шаге за счет использования суффиксного дерева во входном слове определяются возможные границы между стеммой и суффиксом слова. Далее, для каждой потенциальной стеммы (начиная с самой длинной) бинарным поиском по дереву основ проверяется ее наличие в словаре, либо нахождение наиболее близких к ней основ (мерой близости является длина общего «хвоста»). Если слово словарное – алгоритм заканчивает работу, иначе – переходит к следующему разбиению.

Если вариант основы не совпадает ни с одной из «ближайших» словарных основ, то это означает, что анализируемое слово с данным вариантом основы в словаре отсутствует. Тогда по имеющейся основе, суффиксу и модели «ближайшей» словарной основы генерируется гипотетическая модель изменения данного слова. Гипотеза запоминается, а если она уже была построена ранее – увеличивает свой вес. Если слово так и не было найдено в словаре – длина требуемого общего окончания основ уменьшается на единицу, идет просмотр дерева на предмет новых гипотез. Когда длина общего «хвоста» достигает 2, поиск останавливается и идет ранжирование гипотез по продуктивности: если вес гипотезы в пять и более раз меньше самого большого веса – такая гипотеза отсеивается.

Результатом работы стеммера является получившийся набор гипотез для несуществующего слова или одна гипотеза для словарного слова. Кроме того, последняя версия Mystem для каждого варианта начальной формы предлагает всю грамматическую информацию (синтезируемую и для несуществующих слов) и частоту его использования в IPM (instances per million) (если частота неизвестна – выводится 0.00) – эти данные можно использовать в дальнейшем для выбора одной нормальной формы из множества, предложенного программой [20].

Синтаксический анализ

Синтаксический анализ определяет роли слов и их связи между собой, в результате получая набор деревьев, показывающих такие связи. Выполнение задачи осложняется огромным количеством альтернативных вариантов, возникающих в ходе разбора, связанных как с многозначностью входных данных, так и неоднозначностью самих правил разбора [18]. В *табл. 3* далее приведены средства для синтаксического анализа русских текстов.

Таблица 3

Программы для синтаксического анализа русского языка

Название	**Методы**	**Лицензия**	**Платформа**	**Console**	**API**	**Модульность**	**Стоимость (ком.лиц.)**
Свободно распространяемое							
AOT	словарный	LGPL	GNU/Linux, Microsoft Windows	+	+	модуль AOT	-
Link Grammar Parser	грамматика связей	BSD	GNU/Linux, Microsoft Windows	+	+		
AGFL	грамматика аффиксов над конечной решёткой	GPL	GNU/Linux, Microsoft Windows	+	-		
MaltParser	машинное обучение	Собственная	Java	+	-		
ЭТАП-3	Правила	н/д	Microsoft Windows	GUI	-	Модуль ЭТАП-3	
NLTK	машинное обучение	Apache Lisence	Python		+		
Pattern	правила, регулярные выражения	BSD	Python				
Проприетарное							
ABBYY Compreno	Правила	Коммерческая	Microsoft Windows	-	-	модуль ABBYY SDK	н/д
DictaScope	Правила	Коммерческая	FreeBSD, Microsoft Windows	-	+	Библиотека	н/д

Solarix	Правила	Коммерческая	GNU/Linux, Microsoft Windows	+	+	модуль SDK ГС Русского языка	н/д
Синтактико-Семантический Анализ Русского Языка	функциональная модель языка	Коммерческая	Веб-сервис	+	-		н/д

ЭТАП-3

Лингвистический процессор ЭТАП-3 является закрытой разработкой Института проблем передачи информации им. А.А. Харкевича Российской академии наук (ИППИ РАН) [21]. Это компьютерная система, обладающая большим объемом знаний о естественных языках, которая может анализировать тексты, написанные на этих языках, и самостоятельно строить такие тексты по исходному смысловому заданию.

На основе ЭТАПа-3 осуществлены четыре прикладные разработки:

- система машинного перевода,
- конвертор/деконвертор семантического языка UNL,
- компьютерный учебник лексики
- синтаксически размеченный корпус русских текстов СинТагРус [22].

Система машинного перевода умеет переводить тексты с русского языка на английский и с английского на русский. Она располагает словарями этих языков, достигающими 100 тысяч лексических единиц каждый, и несколькими массивами правил анализа, синтеза и перевода текстов. От других систем аналогичного назначения система отличается в первую очередь тем, что опирается на целостную лингвистическую теорию «Смысл ↔ Текст». Далее на рисунке 4 представлен синтаксический разбор предложения:

```
Как сообщили сегодня корреспонденту ИТАР-ТАСС в посольстве РФ в Вене,
сотрудникам российского консульства в Зальцбурге удалось узнать имена
пострадавших.
```

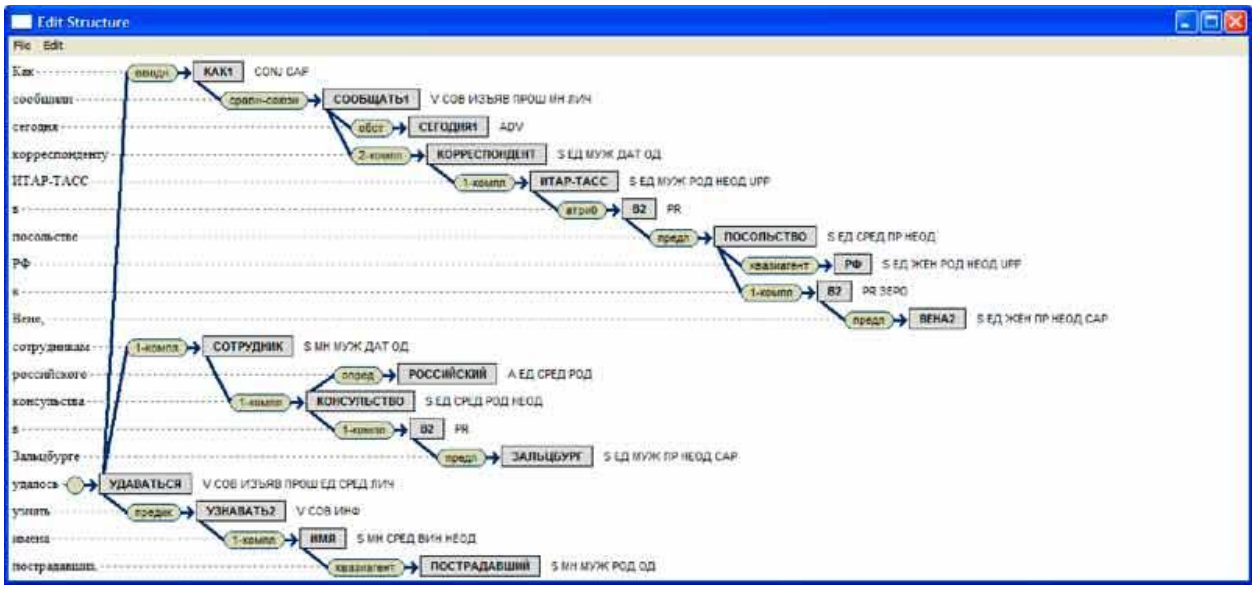


Рисунок 4 Структура предложения в ЭТАП-3

Синтаксическая структура русского предложения в виде дерева зависимостей – главный результат синтаксического анализа, к которому и применяются правила перевода. Все слова предложения представлены своими именами (они в прямоугольниках) и наборами грамматических характеристик – часть речи, род, число, падеж и др. (справа от прямоугольников). Каждое слово, за исключением абсолютной вершины предложения (на рисунке это слово *удалось*), подчиняется какому-то другому слову по некоторому синтаксическому отношению (имена отношений – в полях овальной формы) (Рис. 3).

Конвертор/деконвертор семантического языка UNL способен преобразовывать русские и английские тексты в их семантические представления на языке UNL и осуществлять обратную операцию – синтезировать русские или английские тексты по семантическому представлению. Эти модули входят в состав системы многоязычного общения, в которую помимо них входят конверторы и деконверторы французского, испанского, арабского, хинди и некоторых других языков, разработанные в сотрудничестве с другими командами.

Link Grammar Parser

Link Grammar Parser – это синтаксический парсер английского языка на основе грамматики связей. Работает со словарем, включающем около 60000 словарных форм. Реализован на C для Unix. Есть также версия для Windows API32. Имеет консольный интерфейс [23].

Грамматика связей состоит из слов (терминальных символов грамматики), которые имеют ограничения или требования по связям. Последовательность слов является предложением языка при выполнении трех условий [24]:

1. Проективность: связи между словами не пересекаются
2. Связность: отсутствуют изолированные слова или несвязанные группы слов
3. Требования: выполнены все условия на связи для каждого слова

Для каждого слова в словаре записывается то, какими связями оно может быть связано с другими словами предложения. Для этого используются так называемые коннекторы (специальные последовательности символов), которые обозначают тип связи. Тогда как набор связей, который показывает, что последовательность слов принадлежит языку грамматики связей, называется связкой.

Существует клиент к Web-сервису Link Grammar for Russian, преобразующий вывод анализатора в структуры языка. При модификации парсера для русского языка была разработана формальная грамматика, названная грамматикой связей, с помощью которой предложения кодируются и анализируются с помощью данного контекстно-свободного формализма. Данная грамматика состоит из нескольких сотен базовых правил и более двух десятков базовых типов коннекторов.

Результатом анализа программы является диаграмма связки, где определяются части речи и типы связей:

```
    +--------------Wd---------------+
    |     +IDDL+----EI--+------Sp----+-----------E-----+
    |     |    |        |            |                 |
LEFT-WALL пять минут вечности.ndfpi остались.vsndpp позади.e

(( "LEFT-WALL" RW:6:RIGHT-WALL Wd:4:остались.vsndpp )( "пять" IDDLM:2:минут
)(IDDLM:1:пять "минут" EI:3:вечности.ndfpi )(EI:2:минут "вечности.ndfpi"
Sp:4:остались.vsndpp )(Wd:0:LEFT-WALL Sp:3:вечности.ndfpi "остались.vsndpp"
E:5:позади.e )(E:4:остались.vsndpp "позади.e" )(RW:0:LEFT-WALL "RIGHT-WALL"
))
```

Анализ тональности

Анализ тональности (*sentiment analysis*) — это раздел *text mining*, целью которого является автоматическое извлечение субъективных мнений из текста, дисциплина,

которая исследует не столько содержание текста, сколько его тональность. Говоря о тональности текста, следует выделять три параметра: субъект тональности (автора текста), тональную оценку (позитивность, нейтральность или негативность либо более дискретное деление) и объект тональности (предмет, о котором высказывается мнение, тональная оценка).

Автоматический анализ тональности текста базируется на технологиях лингвистической интерпретации эмоций, машинного обучения, извлечения эмоционального смысла из информации и т.д. Сентимент анализ используется для автоматической оценки новостных событий, продуктов, персоналий, организаций, стран и т.д. В последние годы наибольшей популярностью среди исследователей пользуются текстовые сообщения социальных сетей, таких как *Twitter* и *Facebook*.

Ученые из Вермонтского университета под руководством Питера Доддса с 2009 года анализировали слова в сообщениях пользователей Twitter, на основании чего они делали выводы об уровне счастья авторов сообщений [25]. За время эксперимента «Гедонометр», направленного на измерение исследователи проанализировали сообщения около 63 миллионов пользователей Twitter. Для того, чтобы выяснить эмоциональную окраску того или иного слова, исследователи использовали сервис Mechanical Turk на сайте Amazon, где группа волонтеров оценила «эмоциональную температуру» 10 000 самых употребляемых слов английского языка по 9-балльной шкале. Поскольку все сообщения привязаны к определенной дате и времени, и содержат социально-демографические данные, такие как географическое положение, тенденции изменений в словоупотреблении могут служить основанием для выводов об эмоциональном состоянии тех или иных групп людей. По средствам анализа твиттов ученые опубликовали различные тренды и гипотезы, которые им удалось получить. Например, тенденции смены настроения у людей в течение дня или недели. Или закономерности улучшения эмоционального состояния в праздничные дни (день Святого Валентина, Рождество). На рисунке 5 представлен средний уровень счастья, измеренный за период с 2008 года по настоящее время.

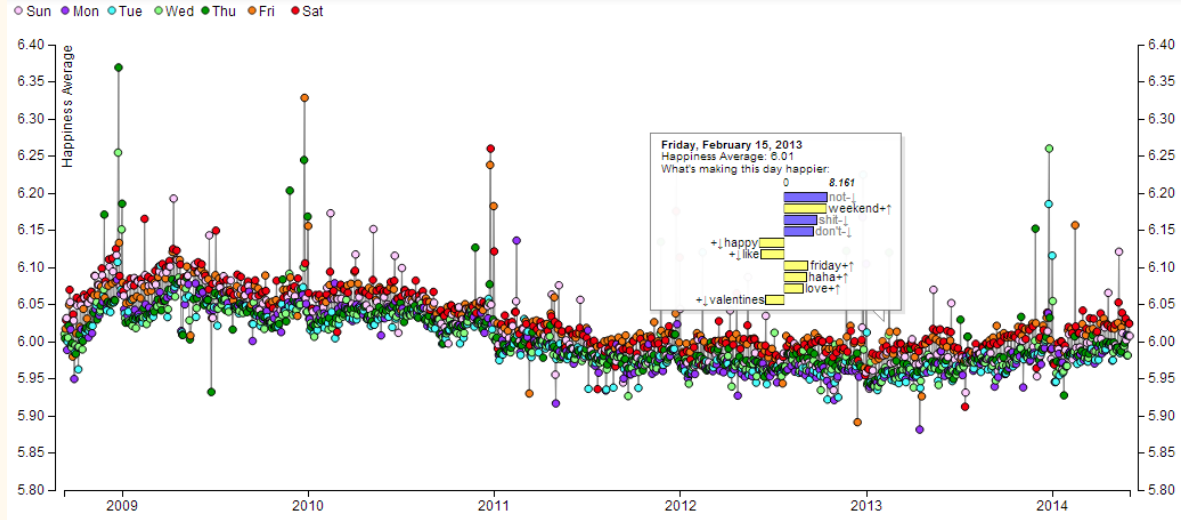


Рисунок 5 Граф уровня счастья

Другим подобным исследованием занимались ученые Великобритании. В мире довольно распространены различного рода индексы, одним из которых является Deprivation Index. Слово “*Deprivation*” обычно сравнимо со словом “*Poverty*”, однако создатели индекса говорят, что он не показывает “нищету”, а скорее показывает отсутствие (“лишение”) возможностей для нормальной жизни:

“*Poverty is not having enough money to get by on where as deprivation refers to a general lack of resources and opportunities*”

Ученые же попытались найти корреляцию между настроением твиттов и индексом “бедности”: чем темнее цвет региона на рисунке 6, тем менее благополучным он считается для проживания [26].

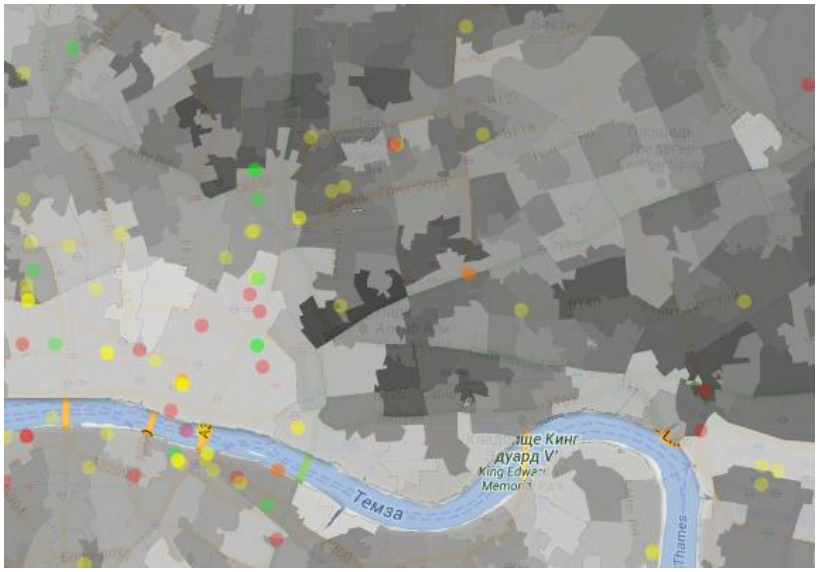


Рисунок 6 The Mood Map (твитты и индекс бедности)

Далее в *табл. 4* представлены программы для анализа тональности русскоязычных текстов. Большинство из них относятся к коммерческим решениям, хотя в ситуации с английским языком на рынке представлено большое число веб-сервисов с бесплатным доступом.

Таблица 4

Программы для сентимент анализа русского языка

Название	**Методы**	**Лицензия**	**Платформа**
Свободно распространяемое			
Sentimental	словарный	MIT	Node.js
Проприетарное			
BrandSpotter	н/д	Коммерческая	Веб-сервис
DictaScope	правила	Коммерческая	FreeBSD, Microsoft Windows
RCO	правила	Коммерческая	Microsoft Windows
SentimentAnalyzer Starget	словарный+правила	Коммерческая	Веб-сервис / Java / .NET
SentiStrength	словарный	Коммерческая	Java, .NET
Аналитический курьер	Правила	Коммерческая	Microsoft Windows

Sentistrength

SentiStrength – хорошо известный инструмент для анализа тональности сообщений социальных сетей. Существуют версии программы для 14 языков, которые показывают высокую точность при использовании.

Подход, реализованный в системе, предполагает определение сентиментной окраски разговорных текстов. Поэтому помимо обычного словаря, он использует и привычные для социальных сетей выражения. Алгоритм применяет для оценки две шкалы: от 1 до 5, и от (-1) до (-5) – для классификации сообщений. Таким способом оцениваются по отдельности составляющие позитивных и негативных эмоций и принимается решение на основе этих отдельных весов. Позднее [27] были добавлены еще две классификации текстов: бинарная (строго негативный/позитивный) и тернарная (позитивный/негативный/нейтральный).

Ресурсы, используемые при разработке алгоритма включают (для английского):

- Коллекция эмоционально окрашенных слов, выражающих эмоции, состоящий из 298 позитивных и 465 негативных слов; каждое позитивное оценено по шкале в диапазоне 2-5, негативное – от (-2) до (-5);
- Алгоритм автокоррекции для английского языка;
- *Booster word list* – коллекция слов, усиливающая или ослабляющая эмоции для словаря эмоционально окрашенных слов;
- Словарь идиом;
- Коллекция отрицающих конструкция для инверсии эмоционально окрашенных выражений;
- Словарь эмотиконов, ассоциируемый с заданным весом эмоциональности;
- Восклицательные предложения оцениваются с минимальным весом.
- Наличие всех словоформ для коллекций.

Sentiment140

Разработчиками данного продукта являются выпускниками Университета Стэнфорда: Alec Go, Richa Bhayani и Lei Huang. Они реализовали систему для анализа тональности сообщений в *Twitter* для английского и испанского языков. Используемый алгоритм сентимент-анализа – классификация методом максимальной энтропии, заявленная точность которого более 80% [28].

Хотя код программы не является свободно распространяемым, программа в виде веб-сервиса доступна в сети Интернет. Среди реализованных функций системы:

- Классификация нескольких твитов за один запрос (Bulk Classification Service) - либо через передачу JSON, либо через передачу простого текстового файла;
- Задание ключевых слов/словосочетаний, имеющих нейтральное значение в анализируемых твитах (они будут исключены из процесса анализа как стоп-слова);
- Возможность специфицировать область, к которой относятся анализируемые твиты (что в теории повышает точность классификации, т.к. одни и те же слова в разных сферах использования могут иметь разные значения);
- Автоматическое определение языка твитов (в настоящее время поддерживаются только английский и немецкий);
- Учет системой наличия в сообщениях эмотиконов.

В то же время программа не обрабатывает встречающиеся в предложениях отрицательные конструкции, что в теории ведет к более низкой точности классификации; а также отсутствуют продвинутые средства работы с твитами: исправления орфографических ошибок, выявления сарказма и т.п.

ConveyAPI

По заявлениям разработчиков, в системе используются комбинированные методы классификации. Применение машинного обучения при наличии качественных обучающих выборок из social media позволяет выбрать специфичную классификацию для анализируемых текстов.

Инновационные алгоритмы и подходы, используемые для реализации *ConveyAPI,* делают программу уникальной среди существующих конкурентов. Точность сентимент-анализа может составлять до 93 %, при этом пополнение обучающих выборок для узкоспециализированных областей может улучшить заявленный коэффициент.

Функционал включает в себя:

- предоставление простого пользовательского интерфейса для создания обучающих выборок на основе актуальных документов
- возможность использовать пользовательские сеты данных для нужд других пользователей
- инструмент корректировки результатов обучения

Определение эмоции человека происходит по 8 категориям согласно классификации Роберта Платчика [29], представленной в *табл. 5* ниже:

Таблица 5

Классификация эмоций по Р. Платчику

Человеческие чувства (результат эмоций)	**Чувства**	**Противоположность**
Оптимизм	Ожидание + Радость	Неодобрение
Любовь	Радость + Доверие	Раскаяние
Покорность	Доверие + Страх	Презрение
Трепет	Страх + Удивление	Агрессивность
Неодобрение	Удивление + Грусть	Оптимизм
Раскаяние	Грусть + Отвращение	Любовь
Презрение	Отвращение +Злость	Покорность
Агрессивность	Злость + Ожидание	Трепет

Извлечение фактов

Группу средств, занимающихся извлечением фактов из текста, стоит рассматривать отдельно. Во-первых, подобные программные средства редко используют техники машинного обучения для достижения результата. Во-вторых, задача *Fact Extraction* (извлечение структурированных данных или фактов) скорее относится к задаче информационного поиска и не связана с data mining и text mining. Схожесть заключается лишь в том, что задача связана с обработкой текста на естественном языке.

Средства заслуживают отдельного рассмотрения и потому, что исследователь может столкнуться с настолько зашумленными данными, что техники data mining не справятся с нахождением значимой информации. В этом случае, быстрее, дешевле и рациональней описать грамматики извлечения фактов вручную, используя, например *Tomita Parser* или язык *AQL* для *IBM InfoSphere Streams*.

Типичными подзадачами задачи извлечения фактов являются:

- Выявление именованных сущностей: ФИО, географические объекты, названия организаций;
- Выявление связей между сущностями: например, родственных отношений;
- Обработка анафор и конференций: определение принадлежности местоимения к ранее упомянутому объекту;

- Выявление ключевых слов и словосочетаний;

Наиболее популярные средства извлечения фактов представлены в *табл. 6*.

Таблица 6

Средства извлечения фактов

Название	**Метод**	**Языки**	**Лицензия**	**Платформа**
Яндекс Томита Парсер	словари и КС грамматики	русский	для некоммерческого использования	GNU/Linux, Microsoft Windows, Mac OS X
TextMF	частотный анализ	русский, английский	н/д	Java
LingPipe	машинное обучение	английский	Коммерческая	Java
FreeLing	конечный автомат	русский, английский, итальянский, испанский, португальский, астурийский, валийский, галисийский, каталанский	GPL + Коммерческая	GNU/Linux

Наиболее популярным и одним из немногих инструментов для извлечения фактов из русскоязычного текста является средство разработанное компанией Яндекс Томита Парсер. Извлечение фактов происходит при помощи контекстно-свободных грамматик и словарей ключевых слов. Парсер позволяет написать свою грамматику, добавить свои словари и запустить на текстах [30].

Для описания грамматики используются *терминальные* и *нетерминальные* символы.

Под терминалами здесь понимаются:

a) части речи – *Noun, Verb, Adj*;
b) некоторые символы разделители – *Comma, Punct, Ampersand, PlusSign*;
c) леммы – *‘лес’, ‘бежать’*;

Другими словами, под терминалы – объекты, присутствующие непосредственно в языке и имеющие неменяющееся значение. К нетерминальным символам относятся объекты, обозначающие какую-либо сущность языка и не имеющие конкретного символьного значения.

Помимо этого, Томита Парсер поддерживает *пометы* (конструкции, указывающие на наличие конкретной морфологической информации в слове) и *регулярные выражения*.

Пусть дано текстовое сообщение:

```
В 1976 г. Елене Александровне было 27 лет.
```

Предположим, что грамматика задана следующим образом:

```
S -> YEAR; //вершина грамматики
YearDescr -> "год" | "г." // возможные обозначения года
YEAR  ->    AnyWord<wff=/[1-2]?[0-9]{1,3}г?\.?/>;  //год  в  виде
регулярного выражения с возможной буквой «г» в конце
YEAR -> YEAR YearDescr;
```

В данном случае, грамматика выведет *1976 г*. Если требуется извлечь женские имена и отчества из текста, то можно задать следующую грамматику:

```
S -> FemaleName FemaleMiddleName; //вершина грамматики
FemaleName -> Word<h-reg1, gram="имя, жен">;
FemaleMiddleName -> Word<h-reg1, gram="отч, жен">;
```

Вывод в этом случае «*Елене Александровне*». Стоит обратить внимание, что в описанной выше грамматике используются помета h-reg1 (первая буква слова заглавная) и помета gram – указание грамматических характеристик.

Томита Парсер распространяется бесплатно. Однако компания Яндекс запрещает использование инструмента в коммерческих проектах. На основе результатов Томиты Парсера работает сервис «Яндекс.Новости».

Подобная система извлечения фактов включена в системы потоковой обработки данных IBM InfoSphere BigInsights Text Analytics. Грамматика должна быть описана на языке *AQL – Annotation Query Language*.

Извлечение на фактов на IBM InfoSphere с помощью AQL поддерживает любые языки, однако не поддерживается анализ морфологических признаков для русского языка, что значительно усложняет анализ.

Интегрированные пакеты

CoreNLP

Одним из самых известных пакетов для анализа текстовой информации является Stanford CoreNLP. Релиз средства состоялся 1 ноября 2010 года. Пакет активно дорабатывается. На текущий момент было выпущено 16 релизов.

Стэндфордский Университет распространяет средство под лицензией *GPL* (full GPL). Пакет полностью написан на Java. Исходный код опубликован в открытом доступе, документирован.

CoreNLP включает в себя следующие модули:

- *tokenize* – токенизация текста, справляется с появлением html тегов и прочим шумом возможным в неструктурированном тексте;
- *cleanxml* – очистка текста от XML тегов;
- *ssplit* – разбиение последовательности токенов на предложения (необходимо для построения деревьев разбора);
- *pos* – pos-tagging (Part-of-Speech tagging) аннотирование токенов частями речи;
- *lemma* – лемматизация токенов;
- *ner* – (от англ. *Named Entity Recognition* – определение именованных сущностей) определяет имена, адреса, название организаций, даты, время, числовые и денежные форматы;
- *regexner* – модуль позволяющий определять именованные сущности с помощью регулярных выражений;

- *sentiment* – модуль определения тональности текста по шкале от 0 (очень негативный) до 4 (очень позитивный);
- *parse* – построение дерева синтаксического разбора;
- *dcref* – модуль, находящий упоминания каких-либо сущностей (например, сущность может представлена местоимениями или синонимами);

Изначально, средство CoreNLP разрабатывалось для английского языка, но позже были выпущены расширения, позволяющие работать с немецким, арабским, китайским, болгарским, итальянским, португальским языком. Помимо этого, были выпущены обертки (wrapper) на языках программирования C#, F#, Python, Ruby, Scala, Clojure, Node.js (JavaScript) и Perl.

Протестировать возможности пакета можно на демонстрационных web-площадках [31].

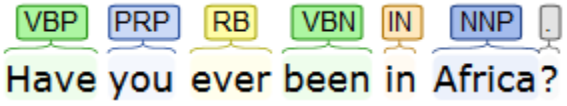


Рисунок 7 Определение частей речи

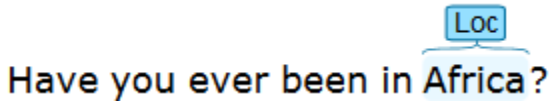


Рисунок 8 Определение именованных сущностей

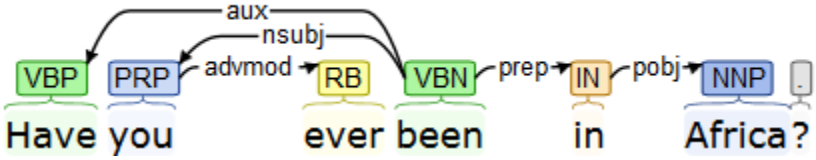


Рисунок 9 Построение графа зависимостей

На Рисунках 7-9 показаны результаты работы некоторых модулей Stanford CoreNLP. В частности POS-аннотации, синтаксический разбор, выявление именованных сущностей.

Особого внимания заслуживает модуль сентимент анализа – определение тональности текстовых сообщений. Для решения этой задачи CoreNLP использует новаторский подход: комбинацию представление текста в виде дерева синтаксического разбора и «глубокое» машинное обучение (*Deep Learning*).

Подобный классификатор использует 5 категорий тональности, в то время как аналоги используют бинарную классификацию (POS - NEG). Точность классификатора - ~80%, но создатели уверены, что за счет расширения обучающей выборки смогут повысить точность до ~96%. Сейчас классификатор обучен на отзывах о кинофильмах, полученных с вебсайта *Rotten Tomatoes*. Обучающая выборка включала около 11000 тысяч отзывов.

Ознакомиться с принципами работы анализа тональности текста от CoreNLP можно на демонстрационном сайте [32].

На Рисунке 10 результат работы определения тональности. Исходное предложение:

```
There are slow and repetitive parts, but it has just enough spice to keep it
interesting
```

представлено в виде дерева синтаксического разбора. Стоит обратить внимание на две аспекта: во-первых, все узлы дерева являются размеченными, во-вторых инструмент справился с наличием противопоставления (<…>, НО <…>). Таким образом, даже при наличии слов с негативной коннотацией slow (медленный, занудный), repetitive (повторяющийся) тональность предложения в целом определяется корректно.

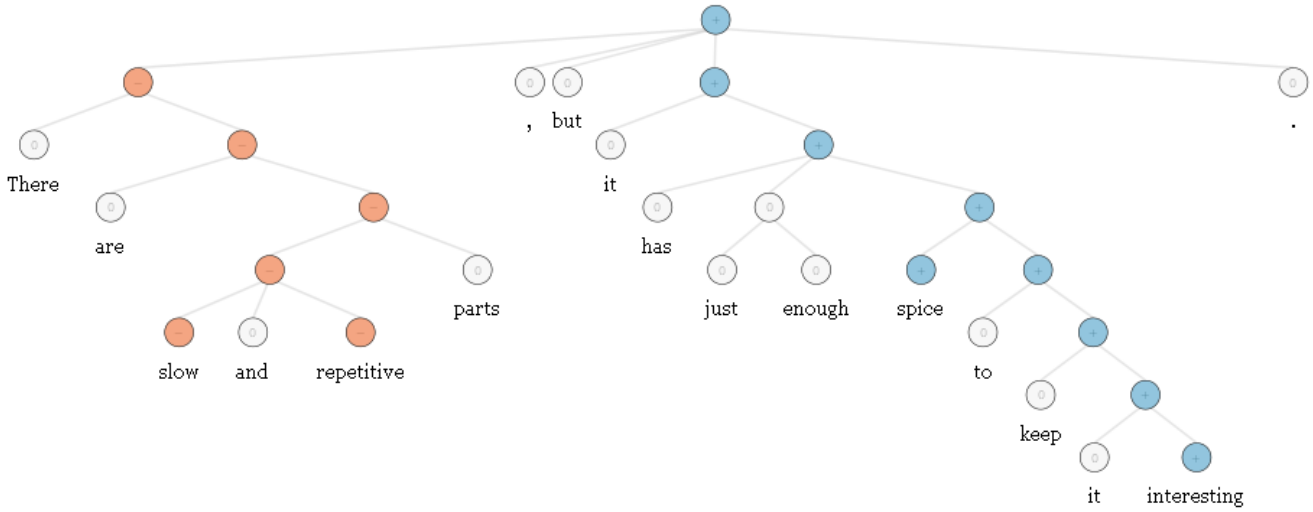


Рисунок 10 Представление предложения в виде дерева синтаксического разбора, аннотированное оценками тональности. Stanford CoreNLP.

К недостаткам пакета CoreNLP можно отнести:

a) Отсутствие поддержки русского языка;
b) Необходимость объемной обучающей выборки (в текущем исследовании было около 700 тысяч ручных аннотирований);
c) Относительно большой расход оперативной памяти (от 2 до 4 Gb для одного процесса);

AOT

Технологии системы АОТ основаны на системы ФРАП (Система французско-русского автоматического перевода была разработана коллективом лаборатории машинного перевода Всесоюзного центра переводов совместно с коллективом лаборатории машинного перевода МГПИИЯ им М. Тореза. 1976-1986 ГГ.) [33].

Компоненты, составляющие языковую модель, - лингвистические процессоры, которые друг за другом обрабатывают входной текст. Вход одного процессора является выходом другого. Выделяются следующие компоненты:

1. Модуль графематического анализа: для выделения слов, цифровых комплексов, формул и т.д;

2. Модуль морфологического анализа: для построения морфологической интерпретации слов входного текста;
3. Модуль синтаксического анализа: для построения дерева зависимостей предложений;
4. Модуль семантического анализа: для построения поверхносто-семантического графа текста;

Абстрактная модель обработки естественного языка представлена на Рисунке 11. На схеме отображены не только последовательности анализа текста, но и используемые ресурсы (например, *словарь оборотов*).

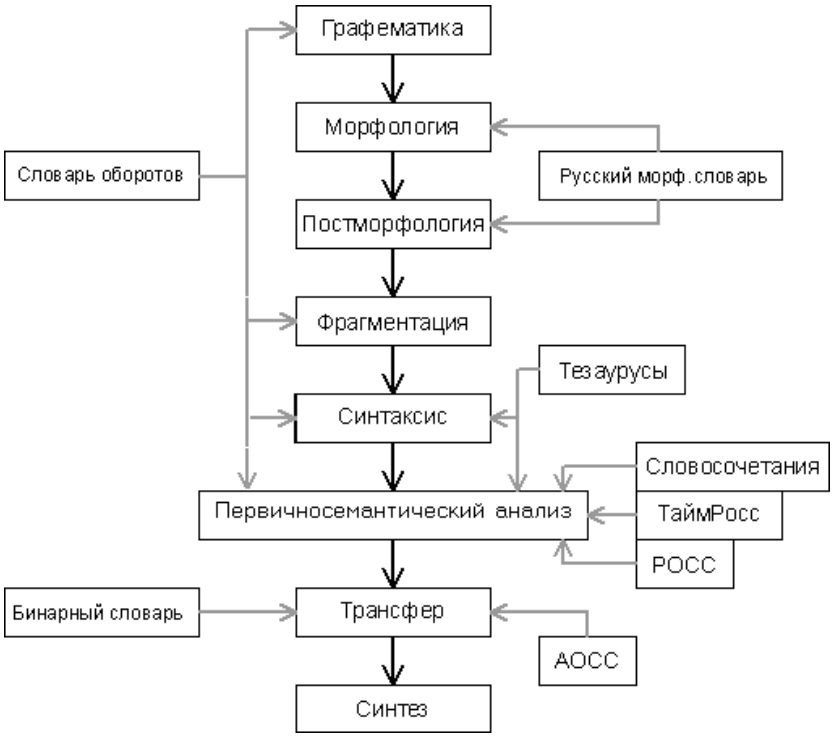


Рисунок 11 Модель системы АОТ

Нормализация слов происходит на ранней стадии обработки средствами АОТ – на стадии морфологии. Работа алгоритма для решения этой задачи аналогична работе средства Mystem и основана на морфологических словарях и частотных словарях.

Основным преимуществом комплекса является возможность синтаксического анализа, т.е. АОТ может определять, в какой зависимости друг от друга находятся слова в предложении и на основе этого выделять члены предложения (подлежащее, сказуемое, дополнения) и словосочетания.

Дерево синтаксического разбора для предложения «Целесообразна ли модернизация энергетического комплекса, если нет средств?» представлено на рисунке 12.

Рисунок 12 Синтаксический разбор АОТ

Еще одной отличительной способностью средства АОТ является возможность построения поверхностно-семантического графа. Семантика – раздел языкознания, который изучает взаимосвязи между языковыми единицами, т.е. семантический анализ указывает, в каких отношениях находятся между собой слова. Например, построение семантического графа по предложению: «Роман Толстого Война и Мир был написан за шесть долгих лет» представлено на Рисунке 13.

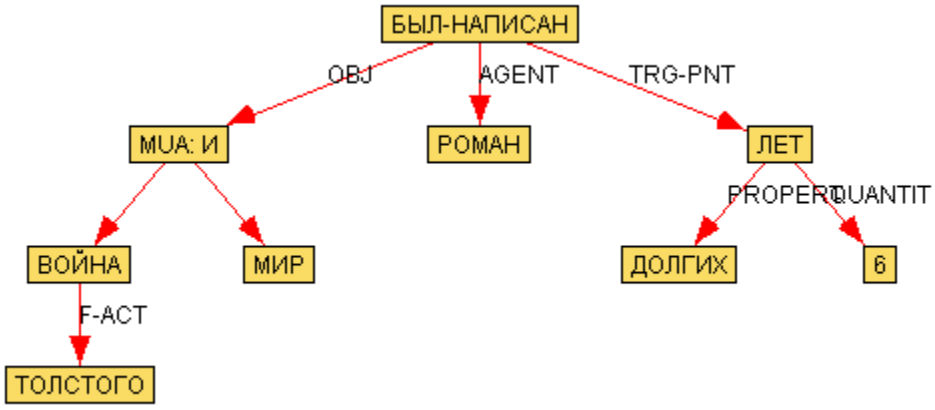


Рисунок 13 Поверхностно семантический граф

Apache OpenNLP

Apache OpenNLP — интегрированный пакет инструментов обработки текста, работающих на основе машинного обучения.

Пакет работает на платформе Java и содержит решения большинства основных задач обработки естественного языка:

- средства токенизации текста;
- разбиения на предложения;

- морфологического анализа;
- извлечения именованных сущностей;
- категоризации текстов;
- синтаксического разбора предложения.

Метод машинного обучения, реализованный в библиотеке, предполагает, что для выполнения любой из задач пользователю изначально необходимо обучить модель.

OpenNLP предоставляет интерфейс в виде командной строки, с помощью которого можно обучить доступные модели.

NLTK

Natural Language Toolkit - пакет библиотек и программ для обработки естественного языка, написанных на языке программирования *Python.* Библиотека сопровождается обширной документацией, включая книгу с описанием основных методов и подходов, реализованных в NLTK, которая одновременно позволяет изучать и понимать реализацию процессов обработки. NTLK является свободным программным обеспечением.

Основные характеристики библиотеки:

- простота – интуитивно понятная реализация алгоритмов, дающая представление о выполняемых процессах;
- последовательность – универсально реализованные интерфейсы и структуры данных с говорящими названиями;
- расширяемость – структура пакетов может быть расширена новыми модулями с альтернативными реализациями;
- модульность – предоставлять компоненты, которые легко использовать независимо друг от друга, не вдаваясь в детали всей системы.

Авторы позиционируют свою разработку как алгоритмический базис для исследования методов NLP, который пополняется с развитием отрасли. Выбор Python в качестве языка программирования обосновывается легкостью восприятия кода, в отличие от низкоуровневых языков. Это подразумевает, что пользователи могут не просто

использовать готовые реализации, но изучать и модифицировать их при необходимости. Далее в *табл. 7* представлены основные модули NLTK с указанием их применения.

Таблица 7

Функциональность NLTK

Задача обработки ЕЯ	**Модуль NLTK**	**Функциональность**
Оценка корпуса языка	nltk.corpus	Стандартизованные интерфейсы для корпусов и словарей
Токенизация	nltk.tokenize, nltk.stem	Токенизаторы слов/предложений, стеммеры
Определений словосочений	nltk.collocations	Тест Стьюдента, Хи-квадрат, point-wise mutual information
POS (определение частей речи)	nltk.tag	*n*-граммы, backoff, *Brill*,скрытые модели Маркова
Классификация	nltk.classify, nltk.cluster	Дерево решений, максимизация энтропии, наивный Байес, *EM, k-means*
Удаление шума	nltk.chunk	Регулярные выражения, *n*-граммы, распознавание имен собственных
Парсинг	nltk.parse	грамматика связей, унификация, вероятностный подход, зависимости
Семантические интерпритации	nltk.sem, nltk.inference	Лямбда-исчисление, логика первого порядка, проверка моделей
Метрики оценки	nltk.metrics	точность, на основе опыта, согласованные допущения
Вероятность и оценка	nltk.probability	Частотные распределения, сглаженные распределения вероятностей
Приложения	nltk.app, nltk.chat	*graphical concordancer*, парсеры, *WordNet* браузер, построители графиков
Лингвистика	nltk.toolbox	Манипуляция данными в формате *SIL Toolbox*

Неспециализированные интегрированные пакеты

Помимо специализированных пакетов и средств text mining, существуют программы широкого профиля для статистической обработки данных, включающие в себя функционал по обработке текста на естественном языке. К подобным проектам относятся *SPSS Modeler*, Rapid Miner, SAS (SAS Text Miner), Weka.

SPSS Modeler

Компания SPSS (Statistic Package for Social Science) с 1983 года разрабатывает средства обработки статистической информации. В 2010 году компания была приобретена

корпорацией IBM, средства data mining *SPSS Clementine* было переименовано в *IBM SPSS Modeler*. Программный продукт обладает полным функционалом для анализа больших объемов информации: выгрузка информации из различных источников (базы данных Oracle, MySQL, Postgres, .xls и .csv файлы), фильтрование шума в данных (*data cleaning*), кластеризация и категоризация. Помимо этого средство позволяет идентифицировать в тексте именованные сущности и другие последовательности слов предопределенной структуры (например, адреса и телефоны). В решение встроены библиотеки для анализа тональности текста, а также библиотеки для визуализации результатов.

Основным недостатком системы является сложность импорта информации: мастер импорта не всегда верно определяет тип данных. SPSS Modeler выпускается в трех версиях: Professional – 27 606$, Premium – от 44 084$ до 110 210$ и Server. Лишь версия *Премиум* предоставляет полный функционал, включающий кластеризацию и категоризацию [34][35].

SAS Text Miner

SAS Text Miner разработан для обнаружения значимой информации в больших объемах «сырого», неструктурированного текста. Решение, как заверяют разработчики [36], помогает экономить время и деньги за счет автоматизации задач извлечения информации из текста.

Система поддерживает импорт данных из документов в основных форматах (.txt, .doc, .pdf), а также загрузку непосредственно с web-страниц. В SAS Text Miner включены опции предобработки и представления информации в векторной форме. Операции над векторами производит система *SAS Enterprise Miner* (Text Miner является расширением этой системы).

К достоинствам системы можно отнести опции по сокращению пространства признаков, наличие модуля для определения эмоциональной окраски документа и поддержку множества языков (в том числе и русского). Стоимость системы начинается от 20 000$.

Weka

Weka – программное обеспечение, в котором реализованы алгоритмы для дата майнинга, разработанное в Университете Вайкато. Алгоритмы могут быть как применены к коллекциям документов, так и быть использованы в пользовательском Java коде. Функциональные возможности *Weka* включают в себя предобработку, обучающие алгоритмы, методы оценки, классификацию, регрессионный и ассоциативный анализы, кластеризацию и визуализацию. Помимо этого предоставляется среда для сравнения методов обучения на выборках, которые входят в пакеты: деревья решений, классификаторы, метод опорных векторов, многослойный персептрон, логистическая регрессия, сети Байеса.

Rapid Miner

RapidMiner – свободно распространяемая среда для проведения экспериментов и решения задач машинного обучения и интеллектуального анализа данных. Программа может работать, как и отдельное приложение, так и быть встроенной в другие программные продукты, в том числе коммерческие. К достоинствам *RapidMiner* можно отнести модульную систему и интуитивно-понятный интерфейс, который позволяет пошагово отобразить процесс обработки данных. В программе реализовано большинство известных алгоритмов кластеризации и классификации, в том числе k-means, c-means, x-means, алгомеративной и дивизимной кластеризаций, Наивный байесовский классификатор, метод опорных векторов. Нейронные сети так же используются при классификации – персептрон, многослойный персептрон и самоорганизующиеся карты признаков Кохонена. Данные могут быть загружены из базы данных, используя *Connector JDBC*, непосредственно с веб-страниц и текстовых файлов (xls, csv).

Теоретическая часть

В теоретической части рассмотрены основные методы предварительной обработки текстовых сообщений, а также методы классификации и кластеризации.

Предварительная обработка

На самой первой стадии анализа текстовой информации документы необходимо избавить от шума. Под шумом обычно понимаются слова, не несущие информации (предлоги, союзы, артикли), слова с опечатками и, иногда, имена собственные. Также необходимо преобразовать все слова в начальную форму, убрать знаки пунктуации и т.д. Пример коротких текстовых сообщений типичных для пользователей социальных сетей:

```
1:  Это лучший в мире аэропорт: бесплатный wi-fi, небольшие очереди и
    дружелюбный персонал.
2:  Из этого аэропорта можно улететь хоть куда!!!
3:  Бесплатный wi-fi и быстрое соединение
4:  Хороший и удобный полет :)
5:  Меры безопасности ужасны ((((
```

Стандартный набор операций предварительной подготовки текстовой информации (Рис. 13):

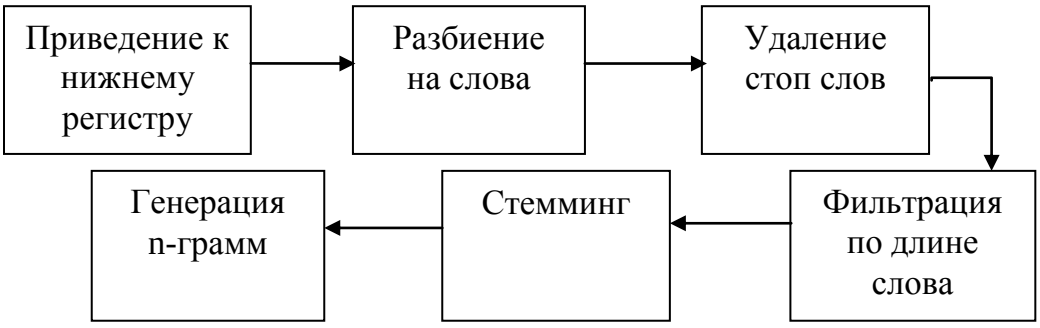


Рисунок 14 Диаграмма процессов предобработки

1) Приведение к нижнему регистру, удаление знаков пунктуации, удаление цифр, удаление лишних пробелов.

Данная процедура очень полезна в отношении сообщений социальных сетей. Пользователи иногда пренебрегают принятой пунктуацией и добавляют в свои сообщение *эмотиконы* ("смайлики") или разбивают слово знаками пунктуации.

```
1:    это лучший в мире аэропорт бесплатный wifi небольшие очереди и
      дружелюбный персонал
2:    из этого аэропорта можно улететь хоть куда
3:    бесплатный wi-fi и быстрое соединение
4:    хороший и удобный полет
5:    меры безопасности ужасны
```

2) Процесс разделения текста на отдельные слова.

Данная процедура достаточно проста. Строка-документ преобразуется в массив слов. Разделение происходит по пробелу.

3) Фильтрация слов по длине.

Чтобы избежать попадания случайных букв, или предложений без пробелов целесообразно применять фильтрацию по длине слова и удалять из документов чрезмерно короткие и неправдоподобно длинные слова.

4) Удаление стоповых слов (stop word).

Стоповыми словами называют слова, настолько часто встречающиеся в языке, что их информационная ценность почти равна нулю, другими словами их энтропия очень мала. Поэтому стоповые слова обычно удаляют перед дальнейшим анализом [37].

Обычно стоповыми словами являются предлоги, союзы и частицы (в случае английского языка – артикли). Существуют стандартные, "общепризнанные" наборы стоповых слов, как для русского [38], так и для английского языка [39].

```
1:    это лучший в мире аэропорт бесплатный wifi небольшие очереди и
      дружелюбный персонал
2:    из этого аэропорта можно улететь хоть куда
3:    бесплатный wifi и быстрое соединение
4:    хороший и удобный полет
5:    меры безопасности ужасны
```

В данном случае, жирным выделены стоповые слова.

5) Стемминг. Приведение к начальной форме.

Процедура стемминга – процедура удаления суффиксов и префиксов слова для извлечения его корня. Это техника часто используется при интеллектуальном анализе текста, так как помогает снизить сложность (прим. размерность вектора признаков) без каких-либо значительных потерь информации [37].

```
1:    лучший аэропорт бесплатный wifi небольшой очередь дружелюбный персонал
2:    аэропорт улететь
3:    бесплатный wifi быстрый соединение
4:    хороший удобный полет
5:    меры безопасность ужасный
```

Самым распространенным алгоритмом стемминга является *алгоритм Портера*. Помимо алгоритма Портера существуют *алгоритм Ланкастера* (для английского языка) и алгоритмы, работающие по принципу "*снежного кома*" (*snowball stemmers*) для других языков.

6) Определение частей речи (*Part-of-Speech tagging*).

Еще одной полезной, но не самой необходимой, процедурой предварительной обработки является маркировка слов в предложении с указанием части речи. Возможно, процедура анализа будет более эффективной, если в словарь будут включены только существительные или же исключены имена прилагательные.

7) *N-граммы*

N-граммой называют непрерывную последовательность из *N* слов. В основном, процедура построения *биграмм* и *триграмм* применяется в системах распознавания речи (где в качестве монограммы используется слог). Последовательность слов не так важна, как последовательность звуков, поэтому N-граммы в text-mining используются реже.

Представление в векторной форме (vector space model)

Для решения задач поиска в документе, задач кластеризации и классификации используется представление документов в векторной форме. Каждый документ d_i представлен в виде вектора, состоящего из весов слов (слова принято называть *термами*)

$d_i = \{w_{1i}, w_{2i}, w_{3i}, \ldots, w_{mi}\}$, w_{ji} - вес *j-го* терма в *i-м* документе, *m* — общее количество различных термов во всех документах коллекции (*размер словаря*).

Существует несколько способов представления слов в векторной форме: *бинарное представление, представление по количеству вхождений (TF), TF-IDF*.

*1)**Бинарное представление*** (Таблица 8)

В случае бинарного представления используются булевские веса, то есть $w_{ji} = 1$, если терм (слово) присутствует в документе, и $w_{ji} = 0$, в противном случае.

Таблица 8

Бинарное представление

№	Текст	аэропорт	wifi	очередь	персонал	полет	безопасность
1	Это лучший в мире аэропорт: бесплатный wi-fi, нет очередей и дружелюбный персонал	1	1	1	1	0	0
2	Из этого аэропорта можно улететь куда угодно	1	0	0	0	1	0
3	Бесплатный wi-fi и быстрое соединение	0	0	0	0	0	0
4	Хороший и удобный полет	0	0	0	0	1	0
5	Меры безопасности ужасны	0	0	0	0	0	1

2) ***Представление по количеству вхождений (TF – term frequency)*** (Таблица 9)

В английском языке такое представление называют "*bag of words*" – "*мешочек слов*".

Основная идея заключается в том, чтобы представить слово в виде вектора, в данном случае в виде мешочка слов. Такое представление содержит только слова, которые принадлежат документы и частоту их появления в документе. Векторная модель игнорирует пунктуацию; предложение разбивается на элементарные частицы (слова, термы), при этом теряется порядок слов и грамматика [37].

Таблица 9

Представление TF

№	Текст	аэропорт	wifi	очередь	персонал	полет	безопасность
1	Это лучший в мире аэропорт: бесплатный wi-fi, нет очередей и дружелюбный персонал	1	1	1	1	0	0
2	Из этого аэропорта можно улететь куда угодно	1	0	0	0	1	0

3	Бесплатный wi-fi и быстрое соединение	0	2	0	0	0	0
4	Хороший и удобный полет	0	0	0	0	1	0
5	Меры безопасности ужасны	0	0	0	0	0	1

3) ***Представление TF-IDF (Term Frequency – Inverse Document Frequency)***

В случае представления, описанном в пункте два, часто встречаемые слова могут повлиять на длину вектора и его направление, что исказит анализ. Чтобы предотвратить это была разработана методика TF-IDF [13] (*Таблица 10*):

$$TF \cdot IDF(t,d,D) = tf(t,d) \cdot idf(t,D), \quad (1)$$

$$tf(t,d) = |t \in d|, \quad (2)$$

$$idf(t,d) = \log \frac{|D|}{\{d \in D : t \in D\}}, \quad (3)$$

где:

t – терм,

d – документ,

D – общее число документов в корпусе.

Таблица 10

Представление TF-IDF

D / t	IDF	d1	TF	d2	TF	d3	TF	d4	TF	d5	TF
f1 (аэропорт)	0.397	0.09925	0.25	0.1985	0.5	0	0	0	0	0	0
f2 (wifi)	0.397	0.09925	0.25	0	0	0.397	1	0	0	0	0
f3 (очередь)	0.698	0.1745	0.25	0	0	0	0	0	0	0	0
f4 (персонал)	0.698	0.1745	0.25	0	0	0	0	0	0	0	0
f5 (полет)	0.397	0	0	0.1985	0.5	0	0	0.397	1	0	0
f6 (безопасность)	0.698	0	0	0	0	0	0	0	0	0.698	1
Кол-во призн.		4		2		2		1		1	

Параметр IDF (обратная частота встречаемости в корпусе) указывает на общую встречаемость слова во всем корпусе. Таким образом, появление в документы часто встречаемых слов нормализуется. Высокие веса будут у термов, которые встречаются в документе часто, но редко во всей коллекции.

Вероятностное векторное представление

Наиболее современным и сложным подходом является использование представлений, учитывающих степень сходства между различными словами (или токенами, в качестве которых могут выступать n-граммы). Основной идеей подхода является возможность определить положение векторного представления слова в зависимости от его ближайших соседей. Причем, слова, векторные представления, которых находятся на наименьшем расстоянии, будут иметь схожий смысл. В литературе встречается определение *Generalized Vector Space Model* – обобщенная векторная модель. Отличием от обычной VSM является учет корреляции между термами, что значительно увеличивает трудоемкость расчета модели.

Информация о связях слов в предложении является абстракцией над порядком слов в нем и может рассматриваться как промежуточный уровень между синтаксисом и семантикой. Более формальное определение дал Тесниэр [41]: «зависимости есть асимметричные бинарные отношения между главным и второстепенным членами предложения». Тогда структура предложения может быть представлена в виде совокупности зависимостей, которые формируют дерево. Именно эти зависимости и формируют контекст, над которым будет определено семантическое пространство.

Механизм задания пространства заключается в идентификации локального контекста целевого слова, который является подмножеством путей зависимостей, начинающихся в этом слове. Пути состоят из граней, отношение зависимостей которых определены как «подлежащее» или «дополнение» и т.д. При этом пути могут быть отсортированы по весу, расчет которого происходит согласно заданной функции (например, может быть оговорено, что подлежащие и сказуемые несут больше семантической информации, нежели определения). Целевые слова тогда представлены в понятиях синтаксиса, которые определяют размерность задаваемого семантического пространства. Впервые формализованное представление семантического пространства было дано Лоуве [40].

Модель семантического пространства представляет собой матрицу $K = B \times T, где\ b_i \in B$ определяет базовый элемент столбца i, $t_j \in T$ обозначает целевое слово

строки j и K_{ij} – ячейка с координатами (i,j). T – множество слов, для которых матрица содержит представления, это могут быть как типы, так и токены слов.

Для построения семантического пространства из совокупной коллекции зависимостей необходимо выполнить последовательность следующих шагов:

1. Определить локальный контекст для слова

Пусть разбор зависимостей p предложения s есть ненаправленный граф $p(s) = (V_p, E_p)$, где набор вершин соответствует словам в предложении: $V_p = \{w_1, \dots, w_n\}$, а $E_p \subseteq V_p \times V_p$ - набор граней.

Класс q есть кортеж, состоящий из двух слов, для которых категорирована часть речи (POS) и отношение r между этими словами. Тогда Q – множество всех классов $<Cat \times R \times Cat>$. Для каждого разбора p, задается функция $L_p: E_p \to Q$, определяющая класс для каждой грани дерева.

Пусть для целевого слова существуют пути зависимости ϕ, где ϕ есть упорядоченный кортеж граней $\langle e_1, \dots, e_n \rangle \in E_p^n$, такой что $\forall i: \big(e_{i-1} = (v_1, v_2) \wedge e_i = (v_3, v_4) \big) \Rightarrow v_2 = v_3$. Другими словами, путь есть кортеж граней, соединенных в граф разбора. Привязанный к слову w путь тот, что начинается в слове w: $\phi = \langle e_1, \dots, e_n \rangle$, где $e_1 = (v_1, v_2)$ и $w = v_1$.

Тогда локальным контекстом для слова w в предложении s считается набор всех привязанных к слову w путей Φ_w. Функция, которая определяет локальный контекст слову, называется функция спецификации контекста (*context specification function*):

$$c(w) = \{\phi \in \Phi_w | \phi = \langle e \rangle \wedge (L_p(e) = \langle V, obj, N \rangle \vee L_p(e) = \langle V, subj, N \rangle)\} \quad (4)$$

Данная формализация отображает, что функция спецификации контекста позволяет детерминировать количество путей на базисе их классов.

2. Количественно оценить контекст слова

Для определения относительной важности различных путей в контексте слова строится функция оценки путей (*path value function*) $v: \Phi \to \mathbb{R}$.

Например, вводится система штрафов: на основе длины пути n можно получить функцию оценки $v(\phi) = \frac{1}{n}$, где $\phi = \langle e_1, \dots, e_n \rangle$.

Тогда максимальный вес самого короткого пути будет равен единице, и дробные значения будут присвоены более длинным путям. Когда оценены все пути локального контекста, необходимо определиться с размерностью множества.

Для этого вводится понятие «отношение эквивалентности путей». Пусть $\sim -$ отношение эквивалентности путей над Ф. Разбиение, вызванное этим отношением, есть множество базисных элементов *B.*

К примеру, возможно соединить все пути, которые заканчиваются в одном слове: путь, который будет начинаться в w_i и заканчиваться в w_j, независимо от его длины будет повторением w_i и w_j. Данное отношение можно формализовать в виде:

$$\langle (v_1, v_2), \dots, (v_{n-1}, v_n) \rangle \sim \langle (v'_1, v_2'), \dots, (v'_{m-1}, v'_m) \rangle, iff \;\; v_n = v'_m \quad (5)$$

После определения локального контекста, можно рассчитать локальную наблюдаемую частоту (*local observed frequency*) повторения базовым элементом *b* целевого слова *w,* как сумму значений всех путей ϕ в контексте, который выражает базовый элемент *b*. Глобальная наблюдаемая частота $\hat{f}$ есть сума всех локальных частот для всех появлений целевого типа *t* и следовательно, мера измерения повторений *t* и *b* во всем корпусе:

$$\hat{f}(b, t) = \sum_{w \in W(t)} \sum_{\phi \in C(w) \wedge \phi \sim b} v(\phi) \quad (6)$$

Согласно распределению типов слов Зифиана, слова с близкими по значению частотами могут быть оценены как более схожие, нежели чем они являются. В таком случае применяют функцию лексической ассоциации (*lexical association function*), чтобы явно сократить повторения. Подобная функция *A* определяет значение в ячейке K_{ij} :

$$K_{ij} = A(\hat{f}(b_i, t_j)) \quad (7)$$

Нормализация значений с помощью функции лексической ассоциации финализирует процесс построения семантического пространства.

К векторным семантическим моделям можно отнести следующие методики.

- Методики четкой кластеризации:
    - *Кластеризация Брауна* (в иностранных источниках встречаются названия *Brown Clustering*, а также *IBM Clustering*);
    - *Обменная кластеризация ;*
- Методики нечеткой кластеризации:

- Латентно-семантический анализ (*Latent Semantic Analysis* - LSA) и Латентно-семантическое индексирование (*Latent Semantic Indexing* - LSI);
- Латентное размещение Дирихле (*Latent Dirichlet allocation* - LDA).

Кластеризация Брауна

Семантическое векторное представление на основе кластеризации Брауна можно отнести к иерархической, а именно агломеративной, кластеризации, которая объединяет слова в бинарные деревья классов.

Пусть W – множество всех слов (словарь) в корпусе документов D $w_1, w_2, \dots w_t$. Также пусть $C: W \to \{1,2, \dots k\}$ – распределение словаря на K классов. Тогда вероятностный семантический вектор:

$$p(w_1, w_2, \dots w_t) = \prod_{i=1}^{n} p(w_i|C(w_i))p(C(w_i)|C(w_{i-1})) \quad (8)$$

Качество распределения по четким кластерам можно оценить с помощью функции:

$$Q(C) = \frac{1}{n}\sum_{i=1}^{n} \log p\big(w_i\big|C(w_i)\big)p(C(w_i)|C(w_{i-1})) \quad (9)$$

Название второе название кластеризации Брауна, IBM Clustering, связано с тем, что группа ученых, во главе с Питером Брауном, работала в IBM T.J. Watson Research Center. Статья была опубликована в 1992 году.

Обменная кластеризация

Алгоритм обменной кластеризации впервые был предложен к использованию для биграммной и триграммной моделей классов [42]. Критерием принадлежности слова к классу является логарифмическая функция подобия.

Если W – словарь корпуса, то $G: w \to g_w$ функция разбиения словаря на фиксированное число классов G, когда каждое слово словаря *w* сопоставляется классу g_w. Функция вероятности принадлежности $p_0(w|g)$ позволяет оценить слово *w* из класса g.

$$p_0(w|g)\begin{cases} > 0, g = g_w \\ = 0, g \neq g_w \end{cases} \quad (10)$$

Для алгоритма кластеризации необходимо получить функцию логарифмического подобия модели. Обменная кластеризация представляет собой обмен слова w между классами для соотнесения w к одному из классов g, при условии, что это оптимизирует функцию правдоподобия. Алгоритм применяет технику локальной оптимизации, циклически проверяя каждый элемент словаря, проверяя его на принадлежность каждого из классов G, до тех пор пока не получает наименьшей нечеткости. Останов процедуры происходит при выполнении критерия прекращения – заданного числа итераций либо не остается слов для обмена между классами.

Латентно-семантический анализ

Латентно-семантический анализ (LSA) – подход в обработке естественных языков, при котором анализируются взаимосвязи между наборами документов и термами, содержащимися в них, путем создания концепций, имеющих отношение к анализируемым текстам. Задачей анализа является извлечение и представление контекстуальных значений слов по средствам статистических вычислений, выполняемых на больших корпусах текстов.

Предположение латентно-семантического анализа заключается в том, что слова, близкие по смыслу, должны встречаться в схожих частях текстов.

В LSA используется матричное представление встречаемости слов в текстах (строки матрицы – уникальные слова, столбец – один параграф текста). Тогда как для уменьшения количества столбцов при сохранении схожей структуры строк применяется сингулярное разложение (SVD – Singular value decomposition). Слова сравниваются по косинуса угла между двумя векторами, представляющими две любые строки. Значение косинуса, близкое к единице, означает высокую степень схожести слов, ноль – отсутствие какой-либо схожести.

Входные данные для анализа представляют собой текст, слова в котором определяются как уникальные символьные строки, а сам текст разделен на смысловые части, такие как предложения или параграфы.

Первым шагом является представление текста в виде прямоугольной матрицы, значения которой далее преобразуются по схеме: для каждой ячейки задаются веса,

значение которых рассчитывается как функция от веса слова в части текста, где оно встретилось, и от степени информационной важности во всем тексте. Пусть *X* – матрица, в которой (*i,j*)-й элемент описывает частотность *i*-го терма в параграфе *j*, причем каждая строка задает вектор для одно терма *t,* а столбец – вектор для одного параграфа(документа) *d.*

$$\begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \quad (11)$$

После этого LSA применяет сингулярное разложение для сокращения размерности матрицы текста.SVD предполагает, что любая прямоугольная матрица может быть разложена на произведение трех матриц:

$$X = U\Sigma V^T, \quad (12)$$

где матрицы *U* и *V* – ортогональные, а Σ – диагональная матрица, значения, на диагонали которой называются сингулярными значениями матрицы *X*. Для получения аппроксимированной матрицы ранга *k* выбираются *k* наибольших сингулярных значений и соответствующие им сингулярные вектора из матриц *U* и *V*.

Имея приближенную матрицу $X_k = U_k\Sigma_k V^T{}_k$, можно:

- Определить связанность двух любых документов d_i и d_q в сниженной размерности, сравнивая вектора $\Sigma_k\widehat{d_i}$ и $\Sigma_k\widehat{d_q}$
- Сравнить любые термы t_i и t_p, используя векторы $\Sigma_k\widehat{t_i}^T$ и $\Sigma_k\widehat{t_p}^T$
- Кластеризовать векторные представления документов и термов, используя, например, алгоритм *k-means.*

В области информационного поиска данный подход называют латентно-семантическим индексированием (LSI).

Латентное размещение Дирихле

Латентное размещение Дирихле (LDA) – порождающая вероятностная модель, основной идеей которой является представление документов в качестве случайных

комбинаций латентных тематик, в то время как каждая тематика характеризуется распределенным множеством слов [43]. Формальные определения LDA включают в себя:

- Слово, как базовая единица представления данных, представляет собой индексированный *v*-ый вектор *w* словаря *V* на отрезке {1,…,*V*}.
  $w^v = 1$ и $w^u = 0$, при $u \neq v$
- Документ – это последовательность *N* слов, обозначаемая

  $\boldsymbol{w} = \{w_1, w_2, \dots, w_N\}$, где w_n- *n*-ое слово в последовательности;
- Корпус – коллекция *M* документов: $D = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_M\}$.

Для каждого документа w в корпусе D выполняется следующий процесс:

1. Выбирает $N \sim Poisson(\xi)$
2. Выбирает $\theta \sim Dir(\alpha)$
3. Для каждого слова $w_n, n \in [1, N]$:
   a. Выбирается тематика $z_n \sim Multinomial(\theta)$
   b. Выбирается w_n из $p(w_n|z_n, \beta)$ с полиномиальной вероятностью по z_n

В итоге функция вероятности для корпуса *D* записывается как:

$$p(D|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha)\left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn}, \beta)\right) d\,\theta_d \qquad (13)$$

где параметр α- *k*-вектор с элементами α_i больше нуля;
параметр β - матрица $\beta_{k \times V}$, где $\beta_{ij} = p(w^j = 1|z^i = 1)$.

Модель LDA представлена как вероятностная модель на рисунке 15. Как видно из схемы, LDA состоит из трех уровней представления. Параметры α и β находятся на уровне корпуса, то есть отбираются во время генерации корпуса. По тому же принципу переменные θ_d вычисляются для документа, w_{dn} и z_{dn} - для каждого слова в документе.

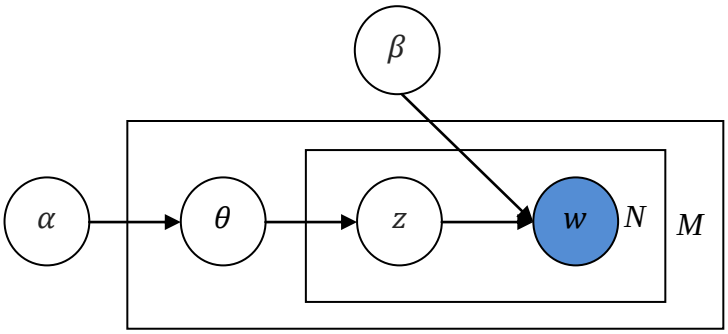


Рисунок 15 Модель LDA

Деревья синтаксического разбора

Деревья синтаксического разбора – форма представления синтаксической структуры предложений. Для описания последней существуют несколько моделей:

- Система зависимостей;
- Система составляющих;
- Система связей.

Грамматика зависимостей

Идея, лежащая в основе анализа через структуры зависимостей, заключается в том, что предложение (текст/сочетание слов) состоит из морфологических слов – словоформ, связанных отношением синтаксической зависимости. Синтаксическая структура цепочки состоит, таким образом, из двух частей:

1) Множества словоформ, содержащихся в цепочке;
2) Множества *n*-арных синтаксических отношений, определенных на множестве словоформ.

Если *S* – конечная цепочка словоформ, то всякое дерево *D* для которого *S* служит множество узлов, называется деревом зависимостей для *S*. Практически для любого предложения соблюдаются три условия, входящие в определение дерева:

- Принцип единственности корневого узла (в предложении есть ровно одно слово, которое не имеет вершины);
- Принцип единственности вершины (слово может иметь только одно вершину);
- Принцип запрета на контур (отсутствие замкнутых структур).

К критериям установления синтаксической зависимости слов и разграничения вершин и зависимых относятся:

- Грамматическая связанность – подчинение какому-то одному грамматическому правилу;
- Линейный порядок – определение линейной позиции одного слова по позиции другого слова;
- Фонетическая слитность – образование словами фонетического комплекса (отрезок предложения, который может выступать в речи изолированно);
- Критерий эндоцентричности – степень совпадения распределения во всем сочетании с распределением хотя бы одного составляющего;
- Морфосинтаксический локус – элемент сочетания, в котором морфологически выражается связь сочетания в внешним контекстом;

В рамках каждого языка могут быть определены виды синтаксических отношений между словами для достижения приемлемой степени детализации в описании структуры предложения.

Если S — линейно упорядоченная цепочка словоформ, а D — дерево подчинения на S, то упорядоченная тройка $< D, Z, \Psi >$, где Z — конечное множество, элементы которого называются метками, а Ψ — отображение множества дуг дерева D в Z («разметка»), называется размеченным деревом подчинения. На рисунке 16 показан разбор предложения «*я вижу высокий дом*» в виде размеченного дерева:

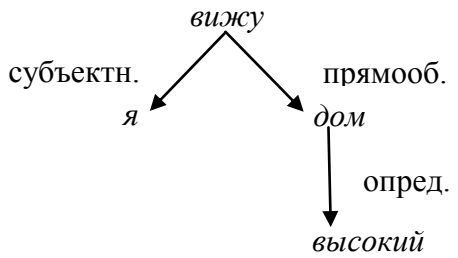


Рисунок 16 Размеченное дерево подчинения

Грамматический смысл пометок таков: *субъектн*. —субъектное отношение (между подлежащим и сказуемым), *прямооб*. — прямообъектное отношение (между глаголом-сказуемым и прямым дополнением), *опред*. — определительное отношение (между определением и определяемым словом).

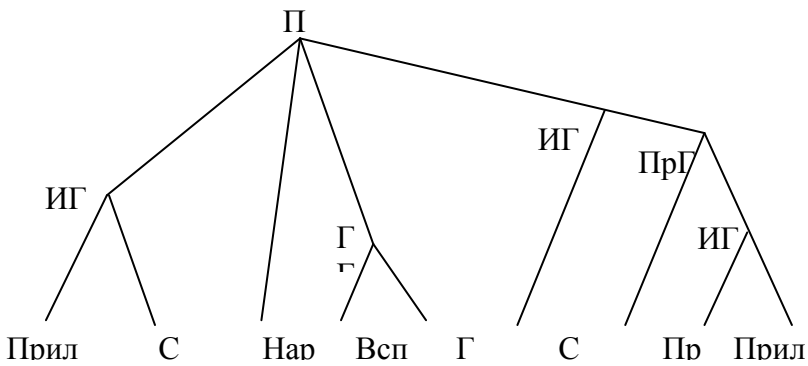
Грамматика составляющих

Помимо отношений между словами в предложении наблюдается еще один, и более важный вид упорядоченных отношений – отношения между группами слов или словосочетаниями [44]. Для их отображения требуется формальная структура другого типа – структура составляющих.

Рассмотрим цепочку (линейно упорядоченное множество) слов *S*. Системой составляющих на *S* является такое множество *C* отрезков *S*, которое содержит в качестве элементов само *S* и каждое слово, входящее в *S,* причем любые два отрезка, входящие в *C,* либо не пересекаются, либо один из них содержится в другом. Элементы *C* называются составляющими. Размеченной системой составляющих на *S* называется множество цепочек вида $<c,k_n...k,,>$, где *c* – элемент системы составляющих *S,* а $k_h...k,,$ – элементы конечного множества категориальных символов *K,* причем каждый элемент *c* входит не более чем в одну такую цепочку.

В грамматике важное значение имеют два вида отношений между составляющими — доминация *A* над *B* (включения составляющей *B* в составляющую *A*) и командование *A* над *B* (когда доминанта над составляющей *A*, также доминирует над *B*, и ни *A*, ни *B* не доминируют одна над другой).

Анализ по составляющим предполагает, что предложение представляет собой иерархическую структуру, состоящую из слов и групп, причем каждый из компонентов принадлежит к некоторому классу – части речи или фразовой категории. На рисунке 17 представлено размеченное дерево – формальная модель категориальной структуры составляющих.



Эти школьники скоро будут писать диктант по русскому

Рисунок 17 Модель структуры составляющих

Части речи, к которым относятся отдельные слова, отображенные с помощью узлов в

основании дерева, обозначены как Прил — прилагательное, С - существительное, Нар — наречие, Всп — вспомогательный глагол, Г — главный глагол, Пр - предлог. Группы, определенные в вершинах – классифицируются по распространяемой в словосочетании части речи. ИГ – именная группа, ГГ – глагольная группа, ПрГ – предложная группа. Верхний класс – класс предложений П.

Грамматика связей

Грамматика связей [24] состоит из слов (терминальных символов грамматики), которые имеют требования по связям. Последовательность слов является предложением языка в случае выполнения следующих условий:

- Проективность – отсутствие пересечений связей между словами;
- Связность – отсутствие изолированных или несвязанных групп слов;
- Требования – выполнение всех условий на связи для каждого слова.

Для каждого слова в словаре записывается то, какими связями оно может быть связано с другими словами предложения. Для этого используются так называемые коннекторы (специальные последовательности символов), которые обозначают тип связи. Для обозначения направления связи справа к каждому коннектору присоединяется либо знак +, либо знак -.

Набор связей, который показывает, что последовательность слов принадлежит языку грамматики связей, называется связкой. На рисунке 18 представлена связка для предложения, которая доказывает принадлежность последнего языку.

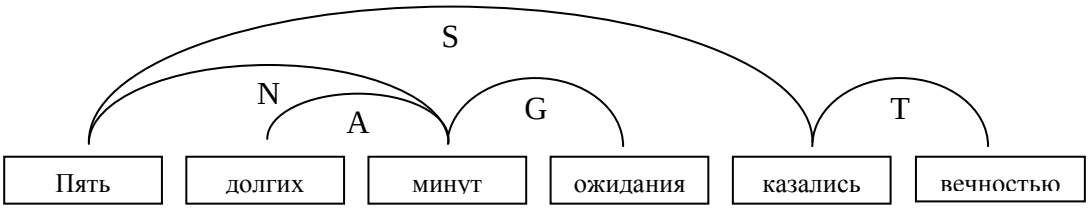


Рисунок 18 Связка предложения

Penn Treebank

Коллекция деревьев разбора, позволяющая покрыть целый язык, называется аннотируемым корпусом языка. Такие коллекции существуют, хотя их не так много. Наиболее ценны глубоко аннотируемые корпуса, которые маркируют синтактико-семантические связи между словами в предложении.

Крупнейший глубоко аннотированный корпус английского языка называется *Penn Treebank* и содержит почти три миллиона слов[45]. Данный корпус был первым масштабным проектом в области *NLP,* и только за первые восемь лет работы (с 1989 по 1996 гг.) было получено:

- Семь миллионов POS-размеченных слов;
- Деревья разбора предложений для трех миллионов слов текста;
- Для более чем двух миллионов слов текста была определена предикат-аргументная структура;
- 1.6 миллиона слов транскрибированных текстов аннотированы для речевых оговорок/оборотов.

Во второй вазе проекта (*Treebank II)* была предикат-аргументная схема анализа, целью которой являлось отнесение аргумента при предикате к семантической категории для определения его роли.

Кластеризация

Кластеризация используется в data mining как метод нахождения новых и значимых наборов данных. Процесс кластеризации очень важен: будучи полностью непредсказуемым, он позволяет структурировать информацию. [37]

Основная задача кластеризации разбиение выборки объектов в группы (кластеры) таким образом, чтобы в каждом кластере находились наиболее похожие объекты.

K-means

Задачей алгоритма является минимизация суммарного квадратичного отклонения точек кластеров от центров этих кластеров:

$$\arg\min_{s} \sum_{i=1}^{k} \sum_{x_j \in S_i} \left\| x_j - \mu_i \right\|^2 \qquad (14)$$

где μ_i - центр масс векторов в кластере S_i

В простейшем случае, алгоритм состоит из четырех частей:
1) Случайным образом задать центры кластеров S_i (*Рис. 17*)

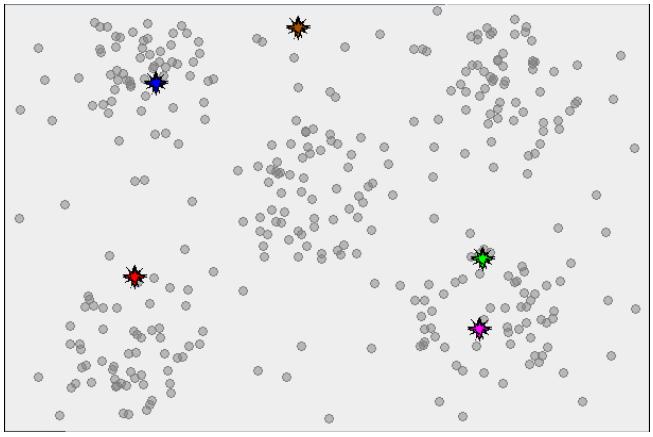


Рисунок 19 Выбор центров кластеров случайный образом

2) Отнести наблюдения к ближайшим центрам кластеризации (*assignment step*):

$$S_i^{(t)} = \left\{ x_P : \left\| x_p \quad m_i^{(t)} \right\| \le \left\| x_p \quad m_j^{(t)} \right\| \forall 1 \le j \le k \right\} \qquad (15)$$

3) Перенести центр кластера в центр масс(*update step*) (Рис. 12):

$$m_i^{(t+1)} = \frac{1}{\left| S_i^{(t)} \right|} \sum x_j \qquad (16)$$

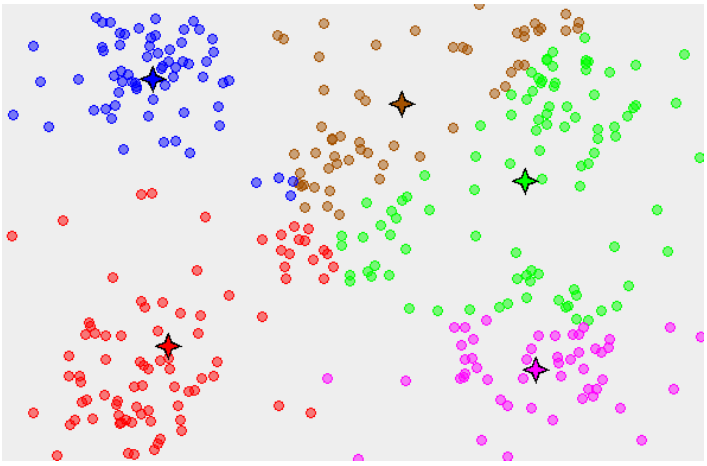


Рисунок 20 Смещение центра масс

4) Повторять шаги 2 и 3 до тех пока алгоритм не сойдется (последующая итерация не приведет к смещению центров кластеризации).

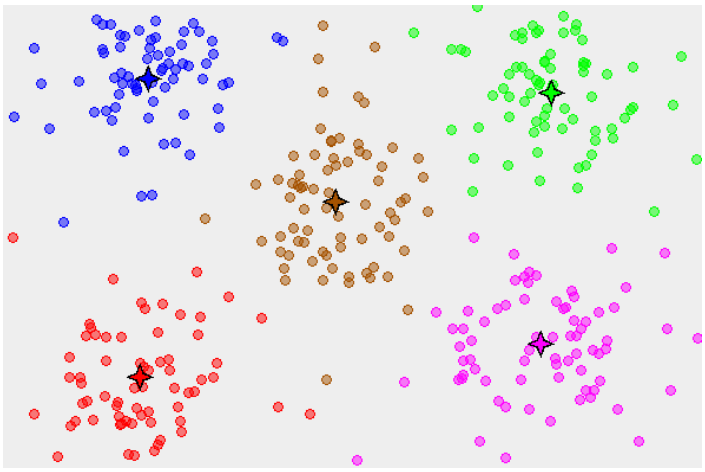


Рисунок 21 Завершенная кластеризация

В зависимости от набора данных и задачи могут быть использованы различные метрики для определения схожести векторов, например, *Евклидово расстояние, Манхэттенская метрика, дистанция Махаланобиса, косинус угла между векторами*.

У кластеризации k-means имеются недостатки (Рис. 19)

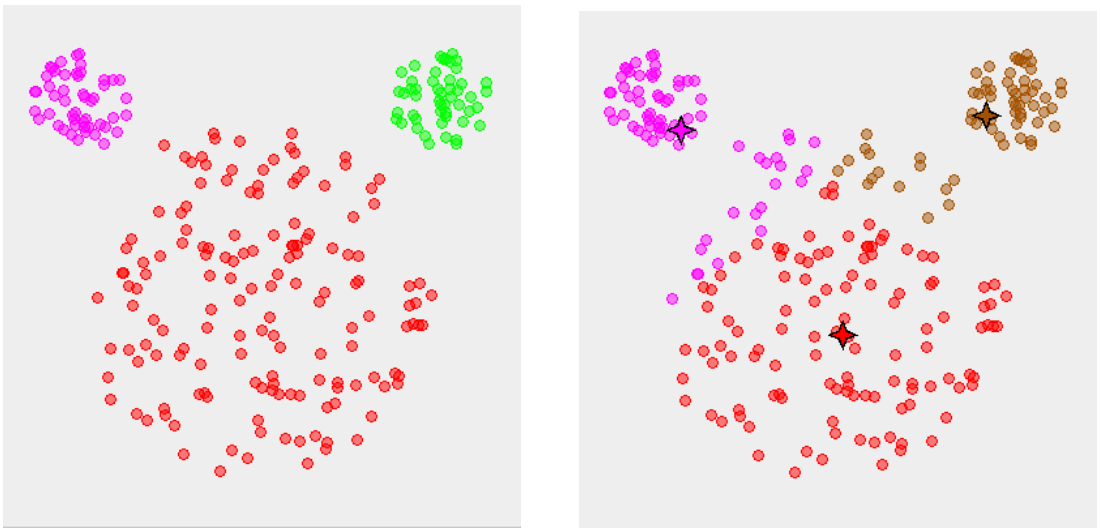


Рисунок 22 Недостатки кластеризации

В случае кластеризации текстовых сообщений указанных выше алгоритм будет работать подобным образом:

1) Пусть количество кластеров k=3

2) Инициализация начальных центров кластеров:

Таблица 11

Начальные центры кластеров

	Инициализация центров кластеров		
	Кластер		
	1	2	3
Аэропорт	0	0	0
Wifi	0	0	0.397
Очередь	0	0	0
Персонал	0	0	0
Полет	0	0.397	0
Безопасность	0.698	0	0

3) Операции присвоения и смещения центра масс повторятся всего 2 раза:

Таблица 12

Итоговые центры кластеров

	Финальные центры кластеров		
	Кластер		
	1	2	3
Аэропорт	0.00000	0.09925	0.04963
Wifi	0.00000	0.00000	0.24813
Очередь	0.00000	0.00000	0.08725
Персонал	0.00000	0.00000	0.08725

Полет	0.00000	0.29775	0.00000
Безопасность	0.69800	0.00000	0.00000

4) Результат кластеризации

Таблица 13

Результат кластеризации

№	Текст	Кластер
1	Это лучший в мире аэропорт: бесплатный wi-fi, небольшие очереди и дружелюбный персонал	3
2	Из этого аэропорта можно улететь хоть куда	2
3	Бесплатный wi-fi и быстрое соединение	3
4	Хороший и удобный полет	2
5	Меры безопасности ужасны	1

K-medoids

K-medoids – алгоритм, схожий с *k-means*, но в отличие от последнего он находит не центры кластеров не как среднее точек, а как медоиды точек. Медоид — объект, принадлежащий набору данных или кластеру, различие которого с другими объектами в наборе данных или кластере минимально. Таким образом, алгоритм проверяет, является ли центр кластера одной из его точек.

$$\arg\min_{s} \sum_{i=1}^{k} \sum_{x_j \in S_i} \left\| x_j \quad \mu_i \right\|^2 \tag{18}$$

Шаги алгоритма аналогичны с *k-means*:

1) Случайным образом задаются центры кластеров S_i
2) Отнести наблюдения к ближайшим медоидам кластера (*assignment step*):

$$S_i^{(t)} = \quad x_P : \left\| x_p \quad X_i^{(t)} \right\| \forall 1 \le j \le k, X_i : \arg\min \sum_{S_i = k} \left\| x_p \quad X_i \right\|^2 \tag{19}$$

3) Перенести центр кластера в медоид (*update step*) :

$$m_i^{(t+1)} = \frac{1}{\left| S_i^{(t)} \right|} \sum x_j \tag{20}$$

4) Повторять шаги 2 и 3 до тех пока алгоритм не сойдется (последующая итерация не приведет к смещению центров кластеризации).

X-means

Алгоритм является расширением *k-means*, в котором реализован механизм разбиения кластеров и получения новых центров. Оценка эффективности разбиения основывается на сравнении существующего и возможных новых центров с помощью байесовского информационного критерия (*BIC*).

X-means может быть разбит на два этапа (части):

1) Оптимизация параметров (*Improve-Params*) – операция, идентичная стандартному алгоритму *k-means*, выполняемая до схождения центров;
2) Оптимизация структуры (*Improve-Structure*) – операция, проверяющая необходимость появления новых центроидов.

После каждой итерации смещения центров выполняется оценка модели M (с *k-means* она имеет представление в виде сферической гауссовой поверхности) : в случае если *BIC* показывает лучшие результаты на новых центрах, происходить разбиение:

$$BIC(M_j) = \hat{l}_j(D) - \frac{p_j}{2} \cdot \log R \qquad (21)$$

где:

$\hat{l}_j(D)$ – логарифмическое подобие информации согласно j-ой модели, взятое в точке максимального правдоподобия;

p_j – количество параметров модели M_j.

Наивный байесовский классификатор

Наивный байесовский классификатор – простой вероятностный классификатор, основанный на теореме Байеса со строгими предположениями о независимости. Он определяет вероятность $P(c|d)$ того, что документ d может быть отнесен к категории c, вычисляя эту вероятность по формуле Байеса:

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)} \qquad (22)$$

Использование предположения о независимости переменных позволяет не изучать взаимодействие всех возможных сочетаний переменных, ограничившись лишь влиянием каждой переменной по отдельности на принадлежность объекта к одной из

категорий. Представляя документ d в векторной форме $d = (w_1, w_2, ...)$, получаем совместную модель вероятности, которая может быть представлена следующим образом:

$$P(c \mid w_1, w_2, ...) = P(c)P(w_1, w_2, ... \mid c) = P(c)P(w_1 \mid c)P(w_2, ... \mid c, w_1) = \\ = P(c)P(w_1 \mid c)P(w_2, ... \mid c, w_1)P(w_3, ... \mid c, w_1, w_2) \quad (23)$$

Используя условие независимости переменных, что каждое свойство документа не зависит от другого свойства, то есть $P(w_i \mid c, w_j) = P(w_i \mid c)$, приводим совместную модель вероятностей к виду:

$$P(c \mid w_1, w_2, ...) = P(c)P(w_1 \mid c)P(w_2 \mid c)P(w_3 \mid c)...P(w_n \mid c) = \prod_i^n P(w_i \mid c)P(c) \quad (24)$$

Для применения классификатора, необходимо обучить его определению вероятности $P(w_i|c)$ имеющемся наборе документов, которые уже распределены по классам. Для этого необходимо посчитать оптимальные и априорные оценки того, что слово встречается в той или иной категории.

Тогда классификатор будет определен следующим образом:

$$classify(w_1, ..., w_n) = \arg\max_c P(C = c) \prod_i P(W_i = w_i \mid C = c) \quad (25)$$

Наивный байесовский классификатор является эффективным на практике, потому что, несмотря на неточности в вероятностных оценках, категоризация объектов часто оказывается верной [46].

KNN кластеризация

Кластеризация на основе алгоритма *k*-ближайших соседей *(k-nearest neighbors*) классифицирует объект на основании того, к какому классу принадлежат *k* ближайших к нему объектов обучающей выборки.

Задачей алгоритма является максимизация значения функции схожести заданного объекта с *k* уже классифицированными объектами обучающей выборки $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$:

$$\arg\max_{y \in Y} \sum_{x=1}^{m} [x_{i;\bar{w}} = y] \hat{\rho}(i, \bar{w}), \quad (26)$$

где $\hat{\rho}(i, \bar{w})$ – заданная весовая функция для оценки степени важности i-го соседа для классификации произвольного объекта $\bar{w}$. В методе k-ближайших соседей функция задается как $\hat{\rho}(i, \bar{w}) = [i \leq k]$. Весовая функция основывается на заданной на множестве объектов функции расстояния ρ, с помощью которой ведется перенумерация выборки относительно удаленности от произвольного $\bar{w}$.

Таким образом, для проведения кластеризации нужно определить, исходя из задачи, параметры для запуска: выбор числа соседей k и используемую метрику расстояния.

Агломеративная кластеризация

На первом шаге агломеративной кластеризации каждый документ – кластер $S_1 = d_1, S_2 = d_2, \ldots, S_n = d_n$.

На втором шаге происходит объединение наиболее похожих кластеров $S_1 = \{d_1\}, S_2 = \{d_2, d_3\}, \ldots, S_{n\ 1} = \{d_n\}$.

Шаг 2 повторяется до тех пор, пока не останется один кластер. Этот кластер будет состоять из всех документов.

Существуют несколько способов определения двух наиболее схожих кластеров. Все они отличаются различными аргументами функции:

$$dist_{rw} = \alpha_p dist_{pw} + \alpha_q dist_{qw} + \beta dist_{pq} + \gamma \left| dist_{pw} \quad dist_{qw} \right| \qquad (27)$$

Таблица 14

Методы агломеративной кластеризации

Method	A	A	β	Γ
Nearest neighbor	½	½	0	~½
Furthest neighbor	½	½	0	½
Median clustering	½	½	¼	0
Between-groups linkage	½	½	0	0
Within-groups linkage	$\frac{k_p}{k_p + k_q}$	$\frac{k_q}{k_p + k_q}$	0	0
Centroid clustering	$\frac{k_p}{k_p + k_q}$	$\frac{k_q}{k_p + k_q}$	$\frac{-k_p k_q}{k_p + k_q}$	0

Ward's method	$\frac{k_r + k_p}{k_r + k_p + k_q}$	$\frac{k_r + k_p}{k_r + k_p + k_q}$	$\frac{-k_r}{k_r + k_p + k_q}$	0

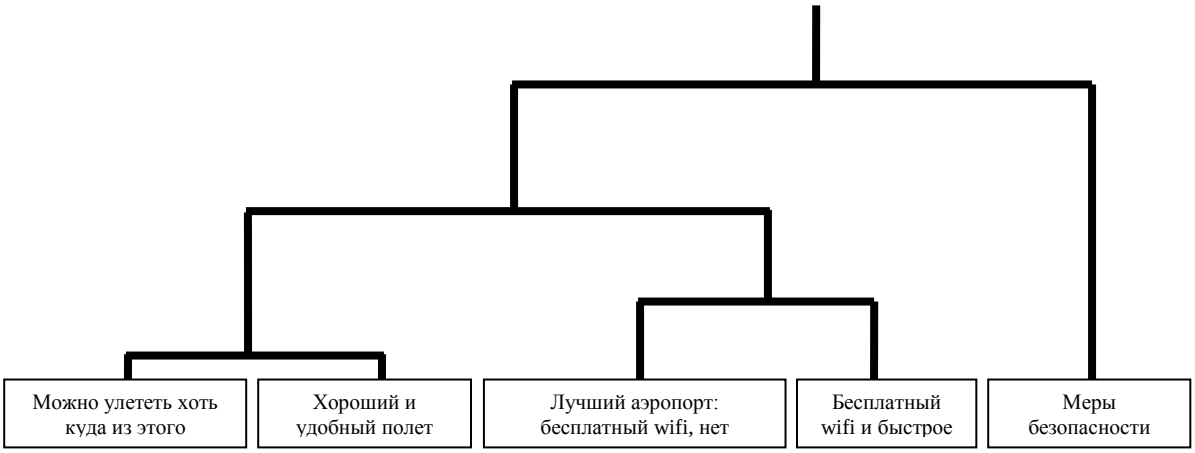


Рисунок 23 Агломеративная кластеризация для документов

Дивизимная кластеризация

В отличие от агломератимных алгоритмов, дивизмные методы представляют на первом шаге все множество элементов как единственный кластер. На каждом шаге алгоритма один существующих кластеров рекурсивно делится на два дочерних. Таким образом, итерационно образуются кластеры сверху вниз. Обычно этот метод применяют, когда необходимо разделить множество объектов на относительно небольшое количество кластеров.

Один из первых дивизимных алгоритмов был предложен Смитом Макнаотом в 1965 году.

1) Все элементы помещаются в один кластер $S_1 = D$

2) Выбирается элемент, у которого среднее значение расстояния от других элементов в этом кластере наибольшее. Среднее значение может быть вычислено по формуле:

$$dist\_s_S = \frac{1}{N_S} \times \sum \sum dist(d_p, d_q) \forall d_p, d_q \in S \qquad (28)$$

3) На каждом последующем шаге элемент в кластере S_1, для которого разница между средним расстоянием до элементов, находящихся в кластере S_2, и средним расстоянием до элементов, остающихся в S_1, наибольшая, переносится в S_2.

Переносы элементов продолжаются до тех пор, пока соответствующие разницы не станут отрицательными, то есть пока существуют элементы расположенные к элементам кластера S_2 ближе, чем к элементам кластера S_1.

Нейронные сети как классификатор

Нейронные сети также могут быть использованы как инструмент классификации. Самая простая нейронная сеть – многослойный персептрон (Рис. 22) хорошо аппроксимирует функции и показывает хорошие результаты для категоризации.

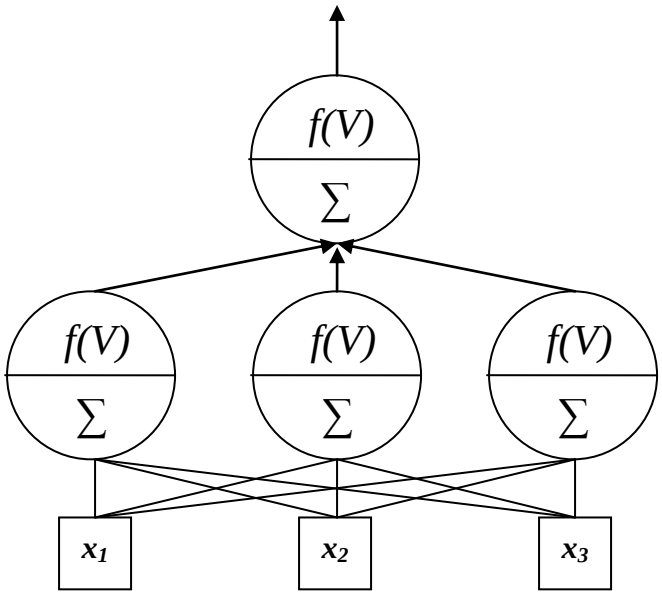


Рисунок 24 Пример сети с прямым распространением сигнала

При обучении нейронной сети прямого распространения сигнала (многослойный персептрон) используется алгоритм обратного распространения ошибки [47].

Нейронная сеть обладает множеством входов x_1,...., x_n , множеством выходов *outputs* и множеством скрытых (внутренних) слоев. Перенумеруем все узлы (включая входы и выходы) числами от *1* до *N* (сквозная нумерация, вне зависимости от топологии слоёв). Обозначим через $w_{i,j}$ вес, стоящий на ребре, соединяющем *i-й* и *j-й* узлы, а через o_i — выход *i-го* узла. При наличии обучающего примера вычисляется функция ошибки по методу наименьших квадратов. Для модификации весов необходимо поправлять веса после каждого обучающего примера и, таким образом, "двигаться" в

многомерном пространстве весов (процесс нахождения глобального минимума в пространстве ошибок). Чтобы "добраться" до минимума ошибки, нужно "двигаться" в сторону, противоположную градиенту, то есть, на основании каждой группы правильных ответов, добавлять к каждому весу $w_{i,j}$

$$\Delta w_{i,j} = \eta \frac{\partial E}{\partial w_{i,j}} , \qquad (29)$$

где $0 < \eta < 1$ - множитель, задающий скорость "движения" (скорость обучения сети). Вычисляемая производная представляет собой поправку, вычисленную для узла следующего уровня. Алгоритм называется алгоритмом обратного распространения ошибки, потому что способен вычислять поправку для узлов последнего уровня и выражать поправку для узла более низкого уровня через поправки более высокого.

Для классификации можно использовать векторное представление, полученное методом TF-IDF. На вход нейронной сети подаются веса, соответствующие термам. То есть, количество нейронов во входном слое будет равно размеру словаря. Выходным слоем является вектор признаков $R = \{r_1, r_2, r_2, \ldots, r_l\}$, где r_i - вероятность документа в категорию.

Рекурсивные нейронные сети

В отличие от обычных нейронных сетей, рекурсивные нейронные сети (RNN – *Recursive Neural Networks*) могут обрабатывать структурированные входные данные путем повторного применения той же самой нейронной сети в каждом узле направленного ациклического графа (DAG – *Directed Acyclic Graph*). Раньше они использовались для настроек только в том случае, если другой компонент первоначально создавал графы. Эти ациклические графы последовательно передавались на вход нейронной сети. При такой архитектуре, каждый узел графа (дерева), который не является листом, ассоциируется с той же нейронной сетью. На рисунке 25 представлен DAG, который представляет собой бинарное дерево. Другими словами, все репликации в сети имеют одинаковый вес. Входные данные ко всем этим повторяющимся сетям представляют собой либо представления дочерних узлов либо вычисленные представления на предыдущем шаге.

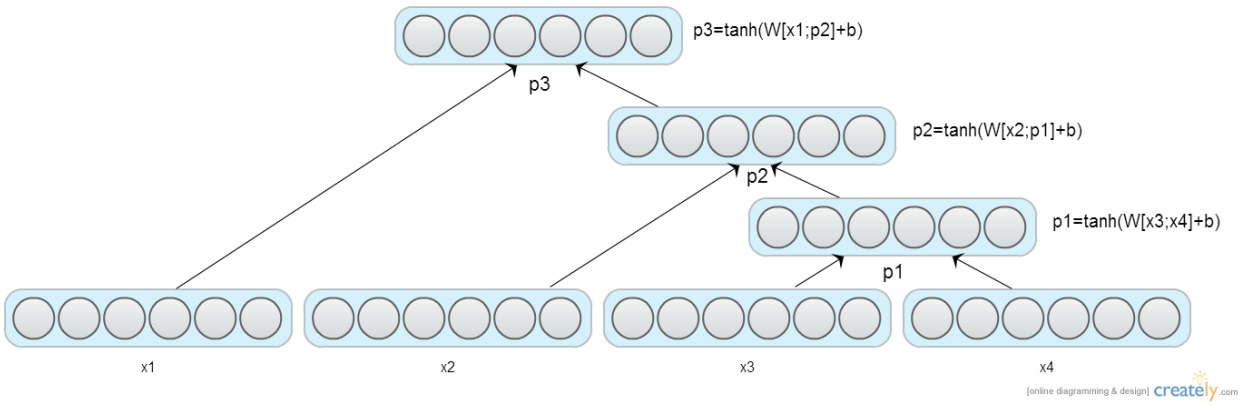


Рисунок 25 Направленный ациклический граф

Пусть есть вектор входных данных $(x_1, \dots, x_n)$, где $x_i \in \mathbb{R}^n$ - имеют одинаковую размерность. Имеется также бинарное дерево в форме ветвления триплетом $(p \to x_1 x_2)$. Тогда каждый триплет обозначает, что родительский узел p имеет два дочерних и каждый c_k может быть либо входом x_i, либо нетерминальным узлом дерева. На рисунке 25 триплетами являются $(p_1 \to x_3 x_4)$, $(p_2 \to x_2 p_1)$, $(p_3 \to x_1 p_2)$. Заметим, для того чтобы повторить нейронную сеть и вычислить представления узлов снизу-вверх по дереву, родительский узел должен иметь ту же размерность, что и каждый из дочерних узлов. Имея подобную структуру дерева, можно вычислить активации для каждого узла, начиная снизу:

$$p = \tanh(W[x_1 x_2] + b), \tag{30}$$

где $x_1, x_2, p \in \mathbb{R}^{n \times 1}$, терм $[x_1, x_2]$ обозначает конкатенацию двух дочерних векторов, результирующую в $\mathbb{R}^{2n \times 1}$-вектор, отсюда $W \in \mathbb{R}^{n \times 2n}$. На вершине далее можно использовать следующий уровень классификации или регрессии.

Рекурсивные нейронные сети для прогнозирования структуры

Далее возможно расширить RNN, чтобы строить дерево снизу-вверх. Пусть имеется список слов. Входные векторы $(x_1, \dots, x_n)$ появляются из матрицы размерности $L \in \mathbb{R}^{n \times |V|}$, где $|V|$ есть размер словаря. Сама матрица L является параметром обучения для модели. Каждое входное слово в последовательности имеет ассоциируемый с ним индекс k из таблицы. Математически, просмотр таблицы можно представить как

проекционный слой, в котором используется бинарный вектор *b*. Данный вектор содержит нули во всех позициях, кроме *k*-ой, которая указывает на индекс.

$$x_i = Lb_k \in \mathbb{R}^n \tag{31}$$

Полученные векторы слов задействуются в паре процессов в качестве:

- параметров модели, которые требуют корректировки (оптимизации);
- входных данных для *softmax* классификатора.

Необходимость данного слоя при решении задачи классификации очевидна: для определения принадлежности входного образа (в данном случае, предложения) одному из классов, нужно обучить сеть моделировать вероятностное распределение.

Softmax слой представляет собой группу нейронов, количество которых соответствует числу классов *K* для категоризации, с функцией активации представленной далее:

$$y_i = \frac{exp^{x_i}}{\sum_{j=1}^{K} exp^{x_j}}, \tag{32}$$

где x_i – входной вектор для *softmax* слоя, x_j – вектор *softmax* слоя.

Таким образом, основной задачей моделей рекурсивных нейронных сетей является *softmax* классификация каждого входного вектора x_i и порождаемых родительских векторов p_i. Главное же различие моделей рекурсивных нейронных сетей заключается в модификации механизма вычисления скрытых векторов p_i [51]:

- RNN (*Recursive Neural Network*): $p_i = f\left(W \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right)$, где $x_1, x_2 \in \mathbb{R}^n$ – входные векторы, $W \in \mathbb{R}^{n \times 2n}$ – параметр обучения, f – функция tanh.
- MV-RNN (*Matrix-Vector RNN*): в матрично-векторной модели для каждого входного слова x_i задается матрица X_i размерности $n \times n$, параметры которой так же подлежат обучению для минимизации ошибки классификации. Тогда родительский узел p_i для входных векторов x_1, x_2также должен представлять пару «вектор-матрица».

$$p_1 = f\left(W\begin{bmatrix}x_1 X_2\\ x_2 X_1\end{bmatrix}\right),\ P_i = f\left(W_M\begin{bmatrix}X_1\\ X_2\end{bmatrix}\right), \quad (33)$$

- RNTN (*Recursive Neural Tensor Network*): идея тензорной нейронной сети заключается в использовании одной и той же тензорной функции ко всем объектам модели. Пусть $V^{[1:n]} \in \mathbb{R}^{2n\times 2n\times n}$ – тензор, определяющий квадратичные формы, тогда сама функция равна $h = \begin{bmatrix}x_1\\ x_2\end{bmatrix}^T V^{[1:n]}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$, тогда как родительский вектор:

$$p_1 = f\left(\begin{bmatrix}x_1\\ x_2\end{bmatrix}^T V^{[1:n]}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + W\begin{bmatrix}x_1\\ x_2\end{bmatrix}\right) \quad (34)$$

Recursive Neural Model

Рассматривается алгоритм анализа тональности текстового сообщения, т.е. решение частного случая задачи классификации.

Пусть, обучающая выборка состоит всего из двух текстовых сообщений:

```
1:    Не самый дружелюбный сервис, ужасное обслуживание.
2:    Ужасно дружелюбный сервис.
```

В первую очередь, происходит синтаксический разбор предложений, нормализация токенов и запись результатов в специальном формате (Рисунки 26, 27).

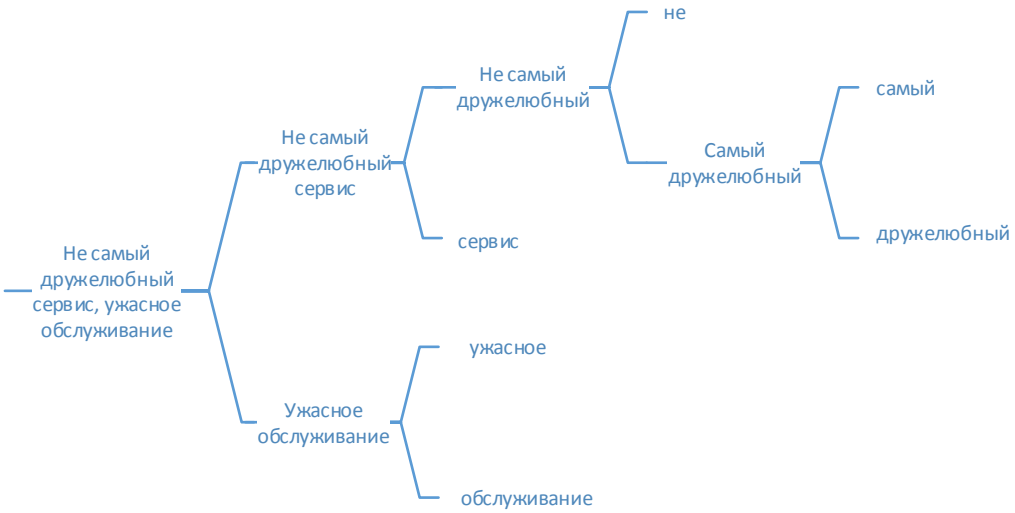


Рисунок 26 Дерево синтаксического разбора для предложения 1.

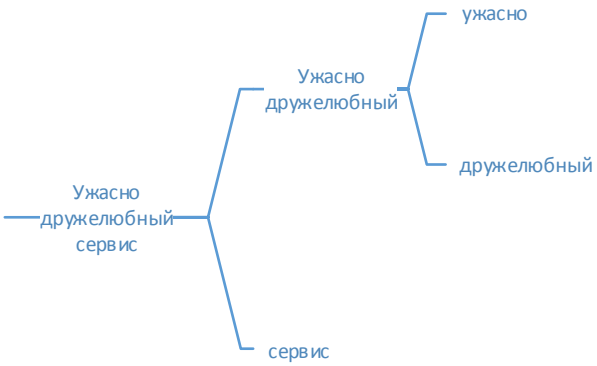


Рисунок 27 Дерево синтаксического разбора для предложения 2.

Далее, происходит нормализация токенов и запись результатов в формате PTB. В качестве примеров приведены безличные предложения, практически все слова находятся в нормальной форме. Для составления обучающей выборки каждый узел дерева разбора необходимо аннотировать. В рассматриваемом случае, в качестве аннотаций будут выступать степени тональности текста:

- 0 – *very negative*, очень негативный;
- 1 – *negative*, негативный;
- 2 – *neutral*, нейтральный;
- 3 – *positive*, позитивный;
- 4 – *very positive*, очень позитивный;

Предложения в аннотированном PTB формате выглядят следующим образом:

```
1:    (1 (1 (1 (1 (2 не) (4 (2 самый) (3 дружелюбный))) (2 сервис)) (2 ,))(1
      (0 ужасное) (2 обслуживание)))

2:    ((4 (0 ужасно) (3 дружелюбный)) (2 сервис))
```

Идея для построения модели определения тональности текста заключается в следующем: оценка должна исходить из анализа каждого слова и его контекста в предложении, нежели чем обобщенная оценка предложения.

В случае обучения нейронной сети, слово в созвучном контексте есть положительный пример обучаемой выборки, тогда как размещение случайного слова в рамках контекста предоставит отрицательный пример. Таким образом, задача, стоящая

перед разрабатываемой моделью – совокупная оценка предложения, основанная на составляющих его словах.

Сначала необходимо получить матричное представление исходных предложений $L \in \mathbb{R}^{n \times |V|}$, где n – заданная размерность одного вектора, а $|V|$ – размер словаря.

Матрица L для каждого из предложений имеет вид, как представлено на рисунке 28:

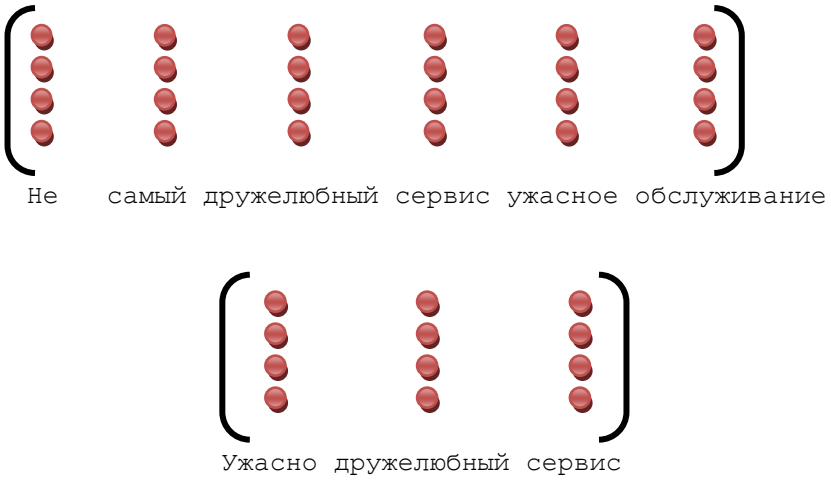


Рисунок 28 Матричное представление предложений

Чтобы получить корректные входные данные для обучаемой нейронной сети, необходимо получить один вектор для предложения (т.е. выполнить конкатенацию векторов). Тогда результирующий вектор x представляет собой (для предложения 1):

Оценка s предложения $x = [x_{\text{не}}, x_{\text{самый}}, x_{\text{дружелюбный}}, x_{\text{сервис}}, x_{\text{ужасное}}, x_{\text{обслуживание}}]$ будет вычислена как:

$$s = U^T a = U^T f(z) = f(Wx + b), \tag{35}$$

где $x \in \mathbb{R}^{24 \times 1}$, $W \in \mathbb{R}^{8 \times 24}$, $U \in \mathbb{R}^{8 \times 1}$.

Заведомо понятно, что получаемые оценки разделяются в зависимости от корректности входных данных, тогда s – есть оценка корректного элемента обучающей выборки, а s_c – ошибочного.

Идея обучения заключается в следующем: используя алгоритм обратного распространения ошибки необходимо минимизировать функцию *J:*

$$J = \max(0,1 - s + s_c) \quad (36)$$

Предполагая, что величина J положительная, можно вычислять производные оценок s и s_c по всем аргументам функции – *U, W, b,x* для работы алгоритма. Беря производные по цепочке от всех переменных, в итоге можно получить необходимое условие минимума.

Далее на одном из слоев сети может использоваться *softmax* классификатор для того, чтобы нейронная сеть могла моделировать вероятность принадлежности объектов к классам.

В случае с анализом тональности текста по пяти степеням эмоциональной окраски маркировка вектора слова может быть записана как:

$$y^{x_i} = softmax\ (W_s x_i), \quad (37)$$

где $W_s \in \mathbb{R}^{5\times n}$ – матрица классификация тональности с размерностью, соответствующей количеству категорий для классификации. Модель для анализа тональности на примере предложения 2 представлена ниже на рисунке 29: оценка тональности каждого вектора позволяет по совокупности оценить положительность предложения.

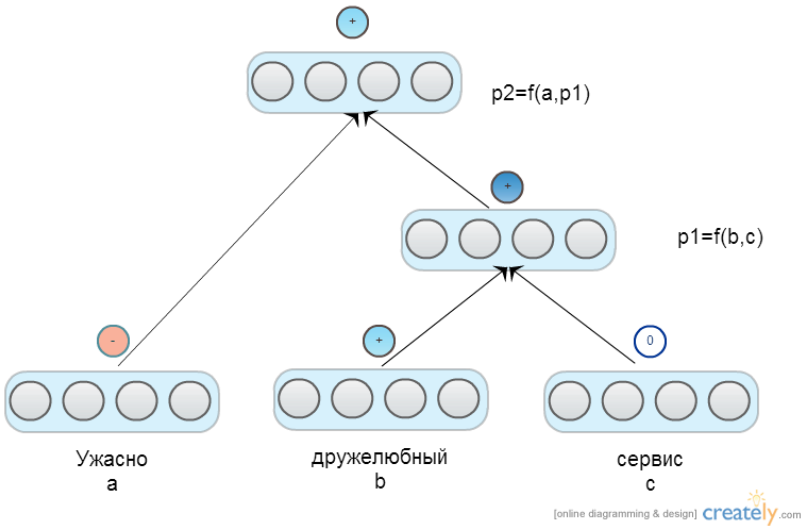


Рисунок 29 Анализ тональности для предложения 2

Практическая часть

Задачей практической части является разработка системы комплексного анализа русскоязычных текстовых сообщений на платформе IBM InfoSphere Streams. В качестве примера комплексного и модульного подхода система решает задачи сентимент-анализа и категоризации текстовых сообщений.

Также в практической части производятся эксперименты по обучению анализатора тональности текста с использованием деревьев синтаксического разбора и многослойных рекурсивных нейронных сетей. Подобный подход был представлен относительно недавно и входит в состав пакета *Stanford CoreNLP,* однако средство не работает с русскоязычными сообщениями и требует адаптации.

Помимо этого, были проведены эксперименты по решению задачи анализа тональности для системы микроблогов *Twitter*. В связи, с отсутствием конкретной предметной области, достаточно сложно обучить классификатор на словарных и вероятностных методах. Поэтому, эта задача решалась с помощью инструмента извлечения фактов Яндекс Томита Парсер путем описания правил контекстно-свободной грамматики.

Далее, в практической части проводится сравнительный анализ основных методов *text mining*. Оценивается зависимость качества от размеров обучающей выборки и делаются выводы.

Анализ методов классификации текста

В ходе эксперимента сравнительного анализа алгоритмов кластеризации участвовали методы, представленные в *табл. 15*.

Таблица 15

Различные методы классификации текста, способы представления и необходимая предварительная обработка

Метод	**Способ представления текста**	**Предварительная обработка**
Naïve Bayes	Tokens	Токенизация, нормализация, частотные фильтры, удаление стоповых слов.

k-NN	VSM (Vector Space Model)	Токенизация, нормализация, частотные фильтры, удаление стоповых слов, TF-IDF.
k-means, c-means, x-means	VSM (Vector Space Model)	Токенизация, нормализация, частотные фильтры, удаление стоповых слов, TF-IDF.
SVM	VSM (Vector Space Model)	Токенизация, нормализация, частотные фильтры, удаление стоповых слов, TF-IDF.
Neural Net *(Perceptron)*	VSM (Vector Space Model)	Токенизация, нормализация, частотные фильтры, удаление стоповых слов, TF-IDF.
Keywords	Raw text	Удаление стоповых слов, нормализация. Подготовка словарей ключевых слов.
Tomita parser	Raw text	Не требуется.
AQL	Raw text	Не требуется.
RNN, RTNN	Treebank	Токенизация, нормализация, частотные фильтры, удаление стоповых слов. Построение дерева синтаксического разбора.

Исходя из предложенной классификации, методы классификации можно разделить по способу представления текстовой информации. Например, способы классификации, основанные на извлечении из текста определенных фактов, не требует предварительной обработки неструктурированного текста.

Методы четкой и нечеткой, иерархической и неирархической кластеризации используют представление текста в векторной форме (бинарное, частотное или нормализованное). Более сложные методы, например рекурсивные нейронные сети, могут работать на деревьях синтаксического разбора.

Эксперимент проводился для решения задачи определения тональности текста. В качестве обучающих и тестовых выборок были выбраны отзывы о популярных мобильных приложениях (Twitter, Foursquare, Facebook) выгруженные из Apple AppStore. Отзывы пользователей написаны на английском языке. Помимо текста отзыва пользователь оставляет заголовок отзыва и оценку приложения от 1 (плохо) до 5

(отлично). Всего было выгружено порядка 100.000 отзывов, которые были разделены на обучающие и тестовые выборки в соотношении 80% к 20%. Распределение оценок в каждой выборке было равномерным, чтобы избежать смещения оценки.

Была выдвинута гипотеза, что методы, работающие на основе векторного представления текста, показывают худшие результаты в сравнении с методами, использующими представление текста в виде дерева синтаксического разбора.

Во-первых, стоит отметить недостатки каждого из методов представления текста.

Методы, требующие ручного составления словаря или написания грамматик, очень трудоемки. Подготовка словаря ключевых слов или написание грамматики может занимать дни, недели или месяцы. Переобучение классификатора потребует таких трудозатрат.

Представление текста в векторной форме имеет сразу несколько недостатков:

1. Теряется информация о связи слов;
2. Увеличение обучающей выборки приводит к неизбежному росту словаря токенов (Рис. 30).

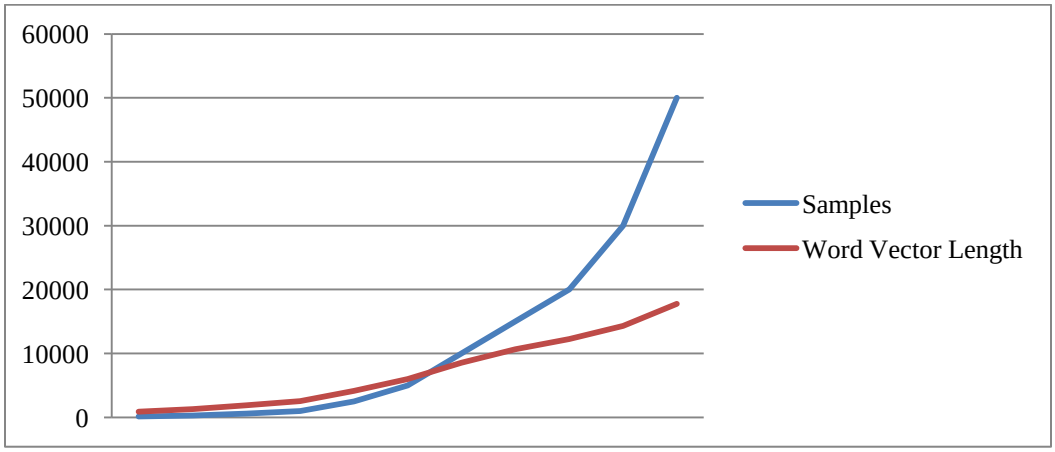


Рисунок 30 Зависимость размерности вектора от количества

Это не только увеличивает затраты на вычисления, но и приводит к ухудшению кластеризации/категоризации. Матрица признаков становится слишком разреженной. Это проблема может быть решена путем применения факторного анализа, однако это процесс тоже требует усилий.

Недостатками представления текста в виде дерева синтаксического разбора являются:

1. Сложность анализа и возможные потери на каждом уровне анализа текстовой информации (графематический анализ, морфологический анализ и синтаксический анализ).
2. Необходимость подготовки огромной обучающей выборки (для каждого предложения необходимо аннотировать все узлы дерева синтаксического парсинга).

Помимо точности классификации, в качестве критерия выступает зависимость точности от размера обучающей выборки. Этот критерий позволяет оценить, возможно ли улучшения качества анализа при увеличении количества документов в обучающей выборке (Рис 30).

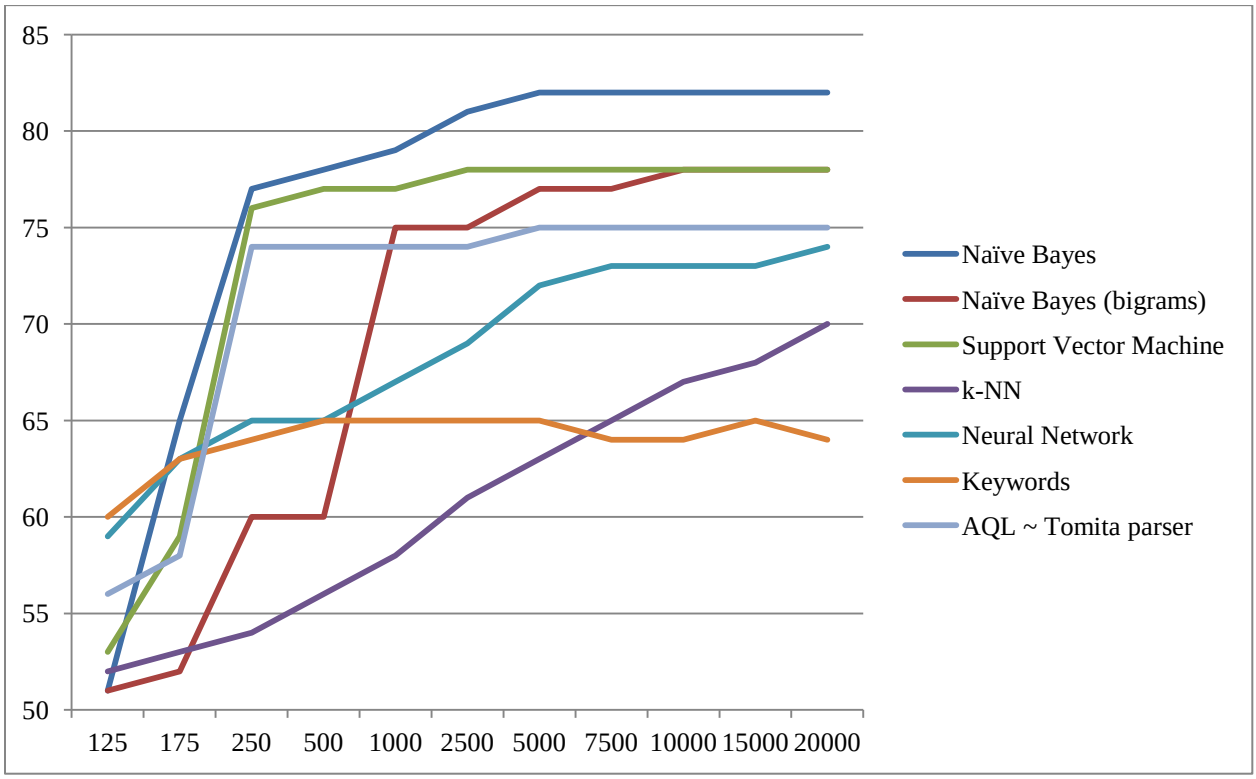


Рисунок 31 На рисунке приведена зависимость точности определения

В случае категоризации с количеством категорий *N*=5 точность анализа значительно падает, и в некоторых случаях ухудшается при росте обучающей выборки.

Результаты сравнения методов классификации приведены в *табл. 16*.

Таблица 16

Результаты обучения классификаторов

Метод	**P (precision), N=2**	**P (precision), N=5**
Naïve Bayes	85	76
Naïve Bayes with bigrams	78	70
Naïve Bayes с синтаксическими группами	87	77
SVM	78	68
Multilayer Perceptron (Neural Network)	74	72
Keywords	65	44
RNN	86	83
Tomita parser	86	-

Наилучшие результаты показал наивный байесовский классификатор, работающий поверх синтаксически связанных признаков. Однако результаты классификации наивного байесовского классификатор ухудшаются с увеличением количества категорий. Данный эффект менее заметен для рекурсивных нейронных сетей, обученных на деревьях синтаксического разбора.

Используя средство Яндекс Томита парсер, были сформированы правила контекстно-свободных грамматик, выявляющие эмоционально окрашенные выражения. Пример извлечения негативно-окрашенных конструкций приведен ниже.

```
S -> NEGSMILE interp (Sentiment.neg_smile) |
     NEGCOLLOC interp (Sentiment.neg_collocation) |
     NEGCOLLOCSTRONG interp (Sentiment.neg_collocation_strong) |
     NEGCOLLOCPERSONAL interp (Sentiment.neg_collocation_personal) |

     POSSMILE interp (Sentiment.pos_smile) |
     POSCOLLOC interp (Sentiment.pos_collocation) |
     POSCOLLOCSTRONG interp (Sentiment.pos_collocation_strong) |
     POSCOLLOCPERSONAL interp (Sentiment.pos_collocation_personal) |

     SWEARING interp (Sentiment.swearing);

NEGCOLLOC -> negColloc | AnyWord<kwtype="negator"> posCollocStrong;
// "не самый хороший фильм"
negColloc -> Adj<kwtype="adj_neg", gnc-agr[1]> Noun<gnc-agr[1]> |
// "плохой фильм"
Verb Adv<kwtype="adv_neg"> |
// "сняли плохо"
Adv<kwtype="adv_neg"> Verb;
// "плохо сняли"
```

Ручное определение правил дает высокие результаты, но требует больших временных затрат. Добавление новой категории в классификатор усложняет процесс в разы. Также требуются словари положительно и негативно окрашенных слов.

Разработка системы анализа текстовой информации

Программное средство (ПС) является частью системы по анализу отзывов, оставленных участниками на демонстрационном стенде компании Сбербанк России. Реализованные алгоритмы также подходят для анализа отзывов пользователей социальных о сетей о продуктах и услугах Сбербанка России.

ПС обладает гибкими возможностями конфигурирования, однако текущая реализация предназначена именно для анализа отзывов об услугах, предоставляемых банками в частном секторе, в особенности банком Сбербанк России.

ПС написано на языке программирования *Java* и адаптировано (реализовано в виде операторов) под использование на платформе *IBM InfoSphere Streams.*

Для корректной работы ПС необходимы результаты обработки текста с помощью АОТ (Автоматическая Обработка Текста) и результаты обучения классификатора в .arff файле.

Объектом анализа являются короткие текстовые отзывы, оставленные пользователями специализированных форумов и групп социальных сетей о работе Сбербанка России – крупнейшего банка Российской Федерации [48].

Заказчик инструмента заинтересован в трех направлениях:

1. В определении продукта (или услуги), о котором упоминается в отзыве.
2. В определении тональности сообщения.
3. В определении упомянутых характеристик.

Решение выполнено на платформе *IBM InfoSphere Streams*, что является ограничением со стороны подрядчика – компании *IBM.*

Задача решалась поэтапно:

1. Анализ и сбор требований заказчика.
2. Определение границ проекта и сроков.

3. Изучение предметной области.
4. Выгрузка обучающей выборки и тестовой выборки.
5. Проектирование решения.
6. Обучение классификаторов, сравнительный анализ классификаторов.
7. Реализация операторов *IBM InfoSphere Streams*.
8. Выводы и улучшение модели.

Извлечение данных для тестовых и обучающих выборок.

Источниками сообщений для анализа данных являются текстовые сообщения из социальных сетей и специализированных форумов. Социальные сети: *Facebook* и ВКонтакте, а также сервис микроблогов *Twitter* предоставляют API для извлечения информации из публичных групп и блогов:

- dev.twitter.com
- developers.facebook.com/
- vk.com/dev/openapi

В данном случае, ответ приходит в структурированном виде. То есть, не требуется очищать текстовое сообщение от html-тегов и прочей разметки. Пример сообщения полученного от *Twitter* API приведен ниже:


```
{
"created_at": "Wed, 19 Jan 2011 21:12:02 +0000",
"profile_image_url":
"http://a0.twimg.com/profile_images/1142619698/DSC_0195_normal.jpg",
"from_user_id_str": "165544885",
"id_str": "27835698383945728",
"from_user": "Deberamatkin",
"text": "Fetching the number of followers without using any Twitter API
http://pr9.in/4q",
"to_user_id": null,
"metadata": {
    "result_type": "recent"
},
"id": 27835698383945728,
"geo": null,
"from_user_id": 165544885,
"iso_language_code": "en",
"source": "&lt;a href=&quot;http://www.exaspring.com&quot;
rel=&quot;nofollow&quot;&gt;ExaSpring Information Services&lt;/a&gt;",
"to_user_id_str": null
}
```

Стоит отметить, что сообщения социальных сетей могут содержать так называемые хеш-теги, указывающие на имя пользователя (*@Username*) или же указывающие на тематику сообщения (*#TwitterAPI*). Такие пометы могут нести в себе как полезную для анализа информацию, так и вносить шум. Однако благодаря синтаксису, подобные конструкции легко извлекаются. Помимо хеш-тегов, в сообщениях могут встречаться эмотиконы, помогающие определить тональность текста и эмоции, которые испытывает автор при написании. Например,

```
Не так уж все и плохо :)
```

Извлечение данных с форумов, не предоставляющих API к данным, сложнее. Дело в том, что текст сообщений приходится извлекать из html-страниц. В этом случае текст содержит специализированные теги.

В качестве примера можно привести один из самых посещаемых специализированных форумов банковской тематики *Banki.ru*. Извлечение данных с публичных форумов данного ресурса способствуют подготовки выборки для обучения классификатора определения продукта. На форуме присутствуют отдельные ветки, в которых обсуждаются кредитные карты, банкоматы, отделения обслуживания физических и юридических лиц и прочее.

Пример сообщения форума, полученный как исходный код html-страницы.

```
<div class="forum-post-entry">
    <div class="forum-post-text" id="message_text_2786934"> У дебетовых карт
Банка Москвы нравятся лимиты обналичивания. Интересная стоимость годового
обслуживания разных карт...интересный вариант с не именной картой, первые 2
года - бесплатно, за 3-й - 150 руб/5$/5euro, можно открыть счета в этих
валютах...карту выдают сразу, прямо в Банке.
    </div>
    <div class="forum-user-signature">
        <div class="forum-signature-line"></div>
        <span>Глупцы героев строят, бросаются вперед,<br />
Нормальные герои всегда идут...в обход.</span>
    </div>
</div>
```

Извлечь необходимую информацию из данного представления несколькими способами. Во-первых, возможно обрабатывать текстовое сообщение с помощью регулярных выражений. Однако такой метод не является корректным. Гораздо правильнее и проще использовать обход DOM-дерева по средствам написания выражений на языке xpath (XML path query language) [49].

Примером для извлечения всех сообщений со страницы форума является выражение:

```
//div[contains(@id, 'message_text')]//text()
```

Выражение осуществляет поиск всех элементов div, id которых содержит «message_text».

Преимуществом xpath является возможность получения массива элементов, которые могут быть рассмотрены отдельно.

Таким же образом можно извлекать информацию с форума Сбербанка России. Данный ресурс и явился основным источником для обучения классификаторов. Всего было выгружено более 60 MB текстовой информации. Характеристики выгрузки представлены в *табл. 17*.

Таблица 17

Характеристики выгрузки

Ветка форума	**Количество тем**	**Количество сообщений**
Банковские карты	4284	54406
Денежные переводы	1201	11166
Качество обслуживания	3871	38452
Кредитные карты	674	4311
Основной форум	3682	35923
Сбербанк Online	2982	27495
Предложения по работе	339	2541
Итого	**17033**	**174294**

Сообщения выгружены с метаинформацией: датой сообщения, url сообщения, именем автора сообщения и ссылкой на его профиль, а также названием темы. Сообщения по каждой ветке хранятся в JSON. Унифицированный формат хранения делает

выгруженные данные доступными для повторного использования. Пример формата хранения показан на Рисунке 32.

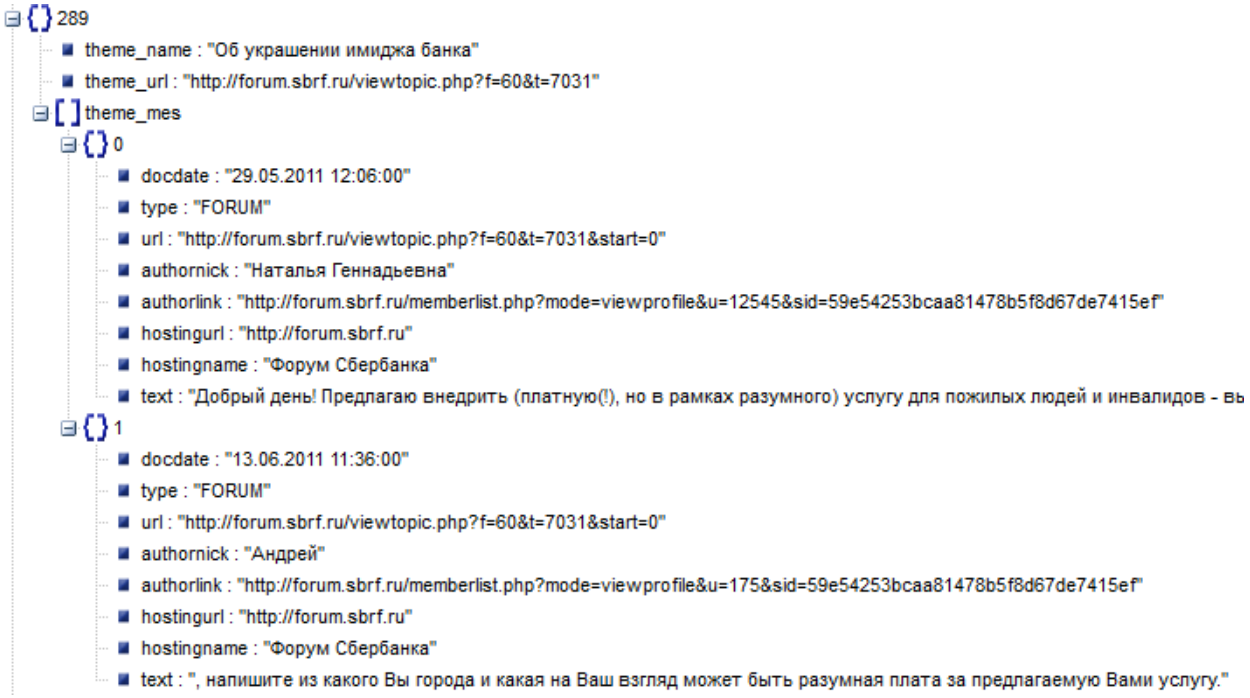


Рисунок 32 Пример формата хранения.

Помимо этого для каждого типа классификатора были составлены свои словари стоповых слов (*табл. 18*)

Таблица 18

Примеры стоповых слов и обучающих выборок.

	Тональность (Sentiment)	**Продукт (Product)**	**Свойтсва (Feature)**
Категории	POSITIVE, NEGATIVE	ATM, MOBILE_BANK, SBER_ONLINE	AVAILABILITY, CONNECTION, COVERAGE, FUNCTIONALITY, SECURITY, SMS, SPEED, SUPPORT, UI
Примеры стоповых слов	*я, ты, он, очень, особенно …*	*банк, я, она, Сбербанк, платеж*	*я, ты, он, очень, особенно …*
Источники обучающих выборок	отзывы на banki.ru, GooglePlay, AppStore	форум сбербанка	форум сбербанка + тематическая литература

Проектирование решения

Решение включает в себя несколько ключевых этапов:

1. Получение текстовых сообщений;
2. Анализ сообщений;
    a. Предварительная обработка сообщения;
    b. Определение продукта;
    c. Определение тональности;
    d. Определение характеристики;
    e. Постобработка сообщения;
3. Запись результатов анализа;
4. Отображение результатов.

Общая схема решения представлена на Рисунке 33:

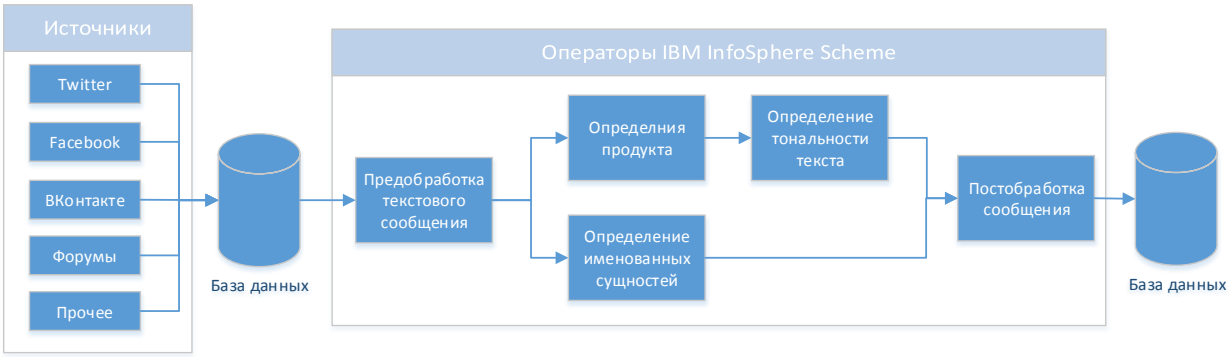


Рисунок 33 Общая схема решения

IBM InfoSphere предоставляет возможность получать сообщения несколькими способами: из БД, из текстовых документов или напрямую из web. Под предварительной обработкой текста подразумевается решение задачи канонизации текста. Постобработка включает в себя аннотирование исходного предложения html тегами для удобства отображение результатов на веб-странице.

Стоит отменить, что оперативная выгрузка сообщений, а также отображение результатов на веб-странице не входит в рамки реализуемого проекта.

Предварительная обработка текстовых сообщений осуществляется за счет сторонних средств. В качестве основных кандидатов выступили две программы – *MyStem*

от Яндекс и АОТ. Выбор был сделан в пользу АОТ в силу нескольких явных преимуществ:

1. *MyStem* доступен только для некоммерческого использования;
2. *MyStem* возвращает только нормальную форму слова, АОТ возвращает не только нормальную форму, но и синтаксически связанные группы;
3. *MyStem* возможно использовать только по средствам обращения к отдельному процессу через stdin/stdout;
4. АОТ можно скомпилировать как нативный оператор *InfoSphere Streams*;

Таким образом, в состав системы анализа будут входит следующие операторы:

1. Оператор предобработки – АОТ
2. Оператор определения продукта: *Банкоматы*, *Мобильный Банк*, *Сбербанк Онлайн;*
3. Оператор определения тональности: *Positive, Negative;*
4. Оператор определения характеристики: *Доступность*, *Скорость соединения*, *Покрытие*, *Функциональность*, *Безопасность*, *Сопровождаемость*, *Пользовательский интерфейс*;
5. Оператор агрегирования результатов;
6. Оператор html аннотирования;

Реализация на IBM InfoSphere Streams

IBM InfoSphere Streams позволяет осуществлять постоянный высокоскоростной анализ больших объемов структурированных и неструктурированных данных в движении, что помогает ускорить процесс обработки бизнес-данных и принятия важных решений. InfoSphere Streams предоставляет отлично масштабируемую среду выполнения, а также широкий выбор интуитивно понятных инструментов, которые помогают пользователям разрабатывать мощные аналитические приложения, которые загружают, фильтруют и сопоставляют данные из одного или нескольких потоков [50].

InfoSphere Streams предоставляет отлично масштабируемую среду выполнения, а также широкий выбор интуитивно понятных инструментов, которые помогают пользователям разрабатывать мощные аналитические приложения, которые загружают, фильтруют и сопоставляют данные из одного или нескольких потоков.

Неструктурированные данные можно суммировать или превращать в структурированные данные для последующего анализа.

Сегмент решений для проведения анализа в реальном времени пересекается с сегментом решений для обработки сложных событий, в котором компания IBM со своим предложением WebSphere Business Events занимает лидирующие позиции. InfoSphere Streams расширяет предложение WebSphere Business Events за счет возможности обработки большего числа событий в секунду в порядке их возрастания, а также возможности обработки неструктурированных данных, таких как аудио- и видеоданные. InfoSphere Streams может выполнять такие функции, как Extract Transform and Load (ETL), там, где компания IBM предлагает ведущие в отрасли решения семейства InfoSphere Server (возможность DataStage). Хотя InfoSphere Streams может справляться с данными рабочими нагрузками, этот продукт не предназначен для приложений ETL, так как они требуют более высокой производительности и возможности выполнения дополнительных функций, таких как одновременная суммаризация. InfoSphere Streams не предоставляет расширения ETL, такие как администрирование, предлагаемые в InfoSphere Server.

Программная реализация на Streams создается как с помощью использования языка программирования Java, так и с помощью DSL (Domain Specific Language), который называется SPL (Streams Processing Language). На языке SPL описываются используемые операторы, их интерфейсы и способы взаимодействия. Разработка приложения происходит с помощью специального модуля, встроенного в IDE Eclipse.

Взаимодействие между операторами осуществляется с помощью структур под названием tuple (или кортеж).

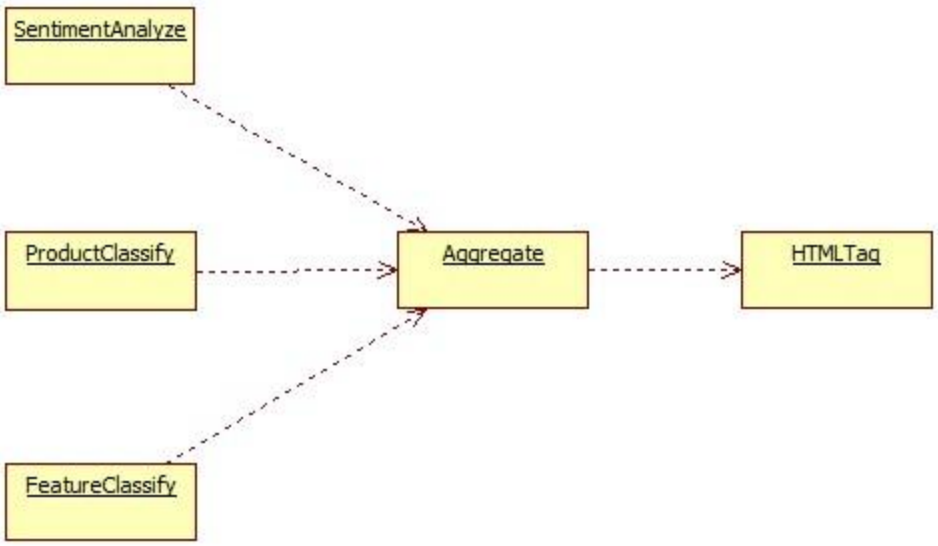


Рисунок 34 Структура операторов Streams

Операторы *SentimentAnalyze*, *ProductClassify*, *FeatureClassify* и *HTMLTag* реализованы как нативные операторы на языке Java. Программные реализации операторов находятся в папке resources*/impl/java/bin/com/ibm/forumspb/operator*.

- Оператор *SentimentAnalyze* предназначен для оценки эмоциональной окраски отзыва и отдельных лемм, его составляющих.
- Оператор *ProductClassify* выполняет классификацию по продуктам отзыва в целом и отдельных лемм, его составляющих.
- Оператор *FeatureClassify* производит классификацию по категориям отзыва в целом и отдельных лемм, его составляющих.

Отзыв, который передается перечисленным выше операторам, проходит предварительную обработку с помощью АОТ.

- Вспомогательный оператор *Aggregate* конструирует обобщенный кортеж на основе переданных кортежей от вышеперечисленных операторов.
- Оператор *HTMLTag* строит HTML–разметку на основе обобщенного кортежа.

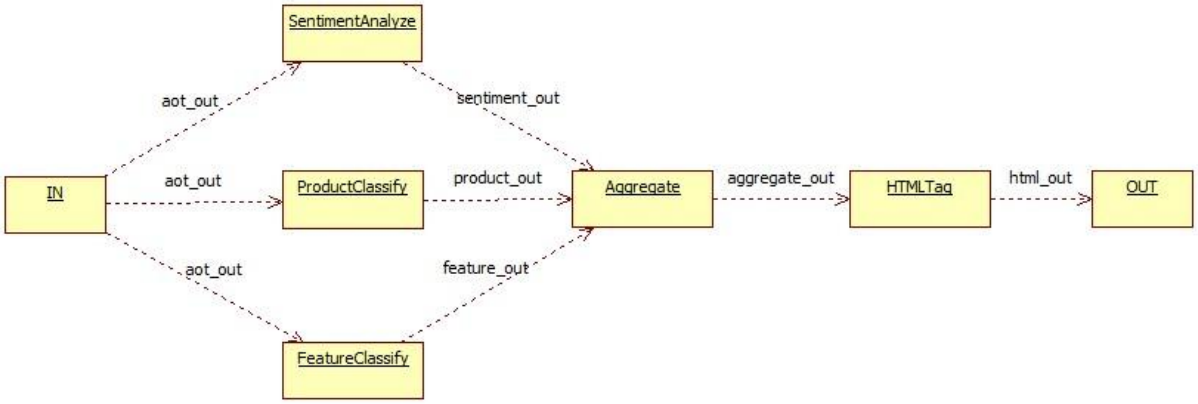


Рисунок 35 Схема входных и выходных данных.

Классификаторы и анализатор тональности текста принимают на вход данные после предварительной обработки и канонизации по средствам АОТ.

```
aot_out:    tuple<rstring aotXml, int32 id>, где
aot_xml – xml-выход АОТ.
id – идентификатор отзыва
```

Далее приведены несколько примеров, иллюстрирующие выходной формат от оператора реализующего методы АОТ.

```
Нравится использовать Сбербанк Онлайн! Быстро и удобно ;)
```

Для данного предложения `aot_xml` будет выглядеть следующим образом:

```
<chunk>
    <input>Нравится использовать Сбербанк Онлайн! Быстро и удобно ;)</input>
    <sent>
        <synvar>
            <rel name=\"ПРИЛ_ПОСТПОС\" > Сбербанк Онлайн </rel>
            <rel name=\"ПЕР_ГЛАГ_ИНФ\" > Нравится использовать </rel>
            <rel name=\"ПРЯМ_ДОП\" > использовать Сбербанк </rel>
        </synvar>
        <lemma_synvar>
            <rel_lemm name=\" \" > СБЕРБАНК ОНЛАЙН </rel_lemm>
            <rel_lemm name=\"ПЕР_ГЛАГ_ИНФ\" > НРАВИТЬСЯ ИСПОЛЬЗОВАТЬ
</rel_lemm>
            <rel_lemm name=\"ПРЯМ_ДОП\" > ИСПОЛЬЗОВАТЬ СБЕРБАНК </rel_lemm>
        </lemma_synvar>
        <lemma>
            <lemm word=\"1\"  > НРАВИТЬСЯ </lemm>
            <lemm word=\"2\"  > ИСПОЛЬЗОВАТЬ </lemm>
            <lemm word=\"3\"  > СБЕРБАНК </lemm>
```

```
            <lemm word=\"4\"  > ОНЛАЙН </lemm>
            <lemm word=\"5\"  > ! </lemm>
        </lemma>
    </sent>
    <sent>
        <synvar>
            <rel name=\"ОДНОР_НАР\" > и Быстро </rel>
            <rel name=\"ОДНОР_НАР\" > и удобно </rel>
        </synvar>
        <lemma_synvar>
            <rel_lemm name=\"ОДНОР_НАР\" > И БЫСТРО </rel_lemm>
            <rel_lemm name=\"ОДНОР_НАР\" > И УДОБНО </rel_lemm>
        </lemma_synvar>
        <lemma>
            <lemm word=\"1\"  > БЫСТРО </lemm>
            <lemm word=\"2\"  > И </lemm>
            <lemm word=\"3\"  > УДОБНО </lemm>
            <lemm word=\"4\"  > ; </lemm>
            <lemm word=\"5\"  > ) </lemm>
        </lemma>
    </sent>
</chunk>
```

Xml-представление содержит следующие узлы:

- `<chunk>` - результат работы АОТ;
- `<input>` - исходный текст;
- `<sent>` - предложение; в исходном тексте их может быть несколько;
- `<synvar>` - массив связанных конструкций, например, прямое дополнение;
- `<lemma_synvar>` - массив связанных конструкций после лемматизации;
- `<rel_lemm>` и `<rel>` - связанные конструкции;
- `<lemma>` - массив слов после лемматизации;
- `<lemm>` - леммы;

Обработанные таким образом данные передаются сразу трем операторам: **SentimentAnalyze, ProductClassify, FeatureClassify.**

Результаты работы `SentimentAnalyze` приведены ниже:

```
sentiment_out:
      tuple<list<ustring> sentiment features,
      list<list<float64>> sentiment_featuresDistributions,
      list<float64> sentiment_distribution, ustring text,
      int32 id>;

sentiment_features - список, включающий составные леммы отзыва.
```

```
sentiment_featuresDistributions – список векторов
      [<вероятность-что-признак-отрицательный>,
      <вероятность-что-признак-положительный>] для каждого элемента
      sentiment_features.

sentiment_distribution –вектор
[<вероятность-что-признак-отрицательный>,
      <вероятность-что-признак-положительный>] для отзыва в целом.

text – текст отзыва (обработанный АОТ).

id – идентификатор отзыва
```

Пример выходной информации приведен ниже:

```
features=["СБЕРБАНКОНЛАЙН НРАВИТЬСЯИСПОЛЬЗОВАТЬ ИСПОЛЬЗОВАТЬСБЕРБАНК
НРАВИТЬСЯ ИСПОЛЬЗОВАТЬ СБЕРБАНК ОНЛАЙН ! ИБЫСТРО ИУДОБНО БЫСТРО И УДОБНО ; )
","СБЕРБАНКОНЛАЙН","НРАВИТЬСЯИСПОЛЬЗОВАТЬ","ИСПОЛЬЗОВАТЬСБЕРБАНК","НРАВИТЬСЯ"
,"ИСПОЛЬЗОВАТЬ","СБЕРБАНК","ОНЛАЙН","ИБЫСТРО","ИУДОБНО","БЫСТРО","УДОБНО"],pr
obabilities=[[0.0031516,0.996848],[0.463127,0.536873],[0.5,0.5],[0.5,0.5],[0.
121282,0.878718],[0.475277,0.524723],[0.5,0.5],[0.5,0.5],[0.209753,0.790247],
[0.5,0.5],[0.215713,0.784287],[0.286516,0.713484]],id=7
```

В первом массиве [0.0031516,0.996848] число 0.996848 означает, что с вероятностью 0.996848 признак "СБЕРБАНКОНЛАЙН НРАВИТЬСЯИСПОЛЬЗОВАТЬ ИСПОЛЬЗОВАТЬСБЕРБАНК НРАВИТЬСЯ ИСПОЛЬЗОВАТЬ СБЕРБАНК ОНЛАЙН ! ИБЫСТРО ИУДОБНО БЫСТРО И УДОБНО ;) ", который является обработанным АОТ'ом сообщением "Нравится использовать Сбербанк Онлайн! Быстро и удобно ;)", является положительным.

Результаты работы оператора* `ProductAnalyzer`*:

```
product_out:
      tuple<list<ustring> product_features,
      list<list<float64>> product_featuresDistributions,
      list<float64> product_distribution, ustring text,
      int32 id> ;

product_features – список, включающий составные леммы отзыва.

product_featuresDistributions – список векторов
      [<вероятность-что-признак-принадлежит-продукту-ATM>,
      <вероятность-что-признак-принадлежит-продукту-MOBILE_BANK,
```

```
      <вероятность-что-признак-принадлежит-продукту-SBER_ONLINE>]
      для каждого элемента product_features.

product_distribution – вектор
      <вероятность-что-признак-принадлежит-продукту-ATM>,
      <вероятность-что-признак-принадлежит-продукту-MOBILE_BANK,
      <вероятность-что-признак-принадлежит-продукту-SBER_ONLINE>]
      для отзыва в целом.

text – текст отзыва (обработанный АОТ).

id – идентификатор отзыва
```

Пусть текст сообщения:

```
Очень неудобно пользоваться банкоматами!
```

Тогда, результат работы оператора:

```
features=["НЕУДОБНООЧЕНЬ ПОЛЬЗОВАТЬСЯНЕУДОБНО ПОЛЬЗОВАТЬСЯБАНКОМАТ ОЧЕНЬ
НЕУДОБНО ПОЛЬЗОВАТЬСЯ БАНКОМАТ .
","НЕУДОБНООЧЕНЬ","ПОЛЬЗОВАТЬСЯНЕУДОБНО","ПОЛЬЗОВАТЬСЯБАНКОМАТ","ОЧЕНЬ","НЕУД
ОБНО","ПОЛЬЗОВАТЬСЯ","БАНКОМАТ"],probabilities=[[0.826647,0.0773094,0.0960432
],[0.158722,0.249763,0.591514],[0.333333,0.333333,0.333333],[0.71666,0.187955
,0.0953856],[0.333333,0.333333,0.333333],[0.331997,0.243799,0.424204],[0.2903
36,0.378953,0.330711],[0.715632,0.169195,0.115173]],id=15
```

В первом массиве [0.826647,0.0773094,0.0960432]число 0.826647 означает, что с вероятностью 0.826647 признак "НЕУДОБНООЧЕНЬ ПОЛЬЗОВАТЬСЯНЕУДОБНО ПОЛЬЗОВАТЬСЯБАНКОМАТ ОЧЕНЬ НЕУДОБНО ПОЛЬЗОВАТЬСЯ БАНКОМАТ . " который является обработанным АОТ'ом сообщением "Очень неудобно пользоваться банкоматами.", принадлежит к продукту АТМ.

Результаты работы оператора **`FeatureAnalyzer`*:***

```
feature_out:
      tuple<list<ustring> feature_features,
      list<list<float64>> feature_featuresDistributions,
      list<float64> feature_distribution, ustring text,
      int32 id>;

feature_features – список, включающий составные леммы отзыва.

feature_featuresDistributions – список векторов
      [<вероятность-что-признак-принадлежит-категории-AVAILABILITY>,
```

```
      <вероятность-что-признак-принадлежит-категории-CONNECTION>,
      <вероятность-что-признак-принадлежит-категории-COVERAGE>,
      <вероятность-что-признак-принадлежит-категории-FUNCTIONALITY>,
      <вероятность-что-признак-принадлежит-категории-SECURITY>,
      <вероятность-что-признак-принадлежит-категории-SMS>,
      <вероятность-что-признак-принадлежит-категории-SPEED>,
      <вероятность-что-признак-принадлежит-категории-SUPPORT>,
     <вероятность-что-признак-принадлежит-категории-UI>] для каждого
      элемента feature_features.

feature_distribution – вектор
      [<вероятность-что-признак-принадлежит-категории-AVAILABILITY>,
      <вероятность-что-признак-принадлежит-категории-CONNECTION>,
      <вероятность-что-признак-принадлежит-категории-COVERAGE>,
      <вероятность-что-признак-принадлежит-категории-FUNCTIONALITY>,
      <вероятность-что-признак-принадлежит-категории-SECURITY>,
      <вероятность-что-признак-принадлежит-категории-SMS>,
      <вероятность-что-признак-принадлежит-категории-SPEED>,
      <вероятность-что-признак-принадлежит-категории-SUPPORT>,
      <вероятность-что-признак-принадлежит-категории-UI>] для отзыва в
      целом.

text – текст отзыва (обработанный АОТ).

id – идентификатор отзыва
```

Спецификация вспомогательных операторов Aggregate и HTMLtag:

```
aggregate_out:
      tuple<list<ustring> sentiment_features,
      list<list<float64>> sentiment_featuresDistributions,
      list<float64> sentiment_distribution,
      list<ustring> product_features,
      list<list<float64>> product_featuresDistributions,
      list<float64> product_distribution,
      list<ustring> feature_features,
      list<list<float64>> feature_featuresDistributions,
      list<float64> feature_distribution,
      ustring text, int32 id>;

html_out:
      tuple<list<ustring> sentiment_features,
      list<list<float64>> sentiment_featuresDistributions,
      list<float64> sentiment_distribution,
      list<ustring> product_features,
      list<list<float64>> product_featuresDistributions,
      list<float64> product_distribution,
      list<ustring> feature_features,
      list<list<float64>> feature_featuresDistributions,
      list<float64> feature_distribution,
      ustring text,
      int32 id,ustring html>;

html - html-разметка результатов работы классификаторов
```

Технико-экономические показатели

Основными метриками измерения качества анализа текста являются:

- точность (precision)
- полнота (recall)
- выпадение (fall-out)
- F-measure

В ходе контрольных испытаний в качестве атрибутов качества рассматривались только показатели точности и полноты. Высокий показатель полноты достигается за счет использования обширного словаря термов. В свою очередь, использование большого словаря термов приводит к снижению точности. Также стоит отметить, что недостатком наивного байесовского классификатора является значительная потеря точности при наличии большого (> 4-5) категорий.

Полученные результаты для тестовых выборок из 200 сообщений:

- тональность - 77-82%
- продукт - 75-80%
- свойство - 72-75%

Данные результаты показывают, что работу классификатора можно считать эффективной. Подобные результаты показывает главный поставщик услуг по категоризации сообщений твиттера *Sentiment140*.

Заключение

Результаты проделанной работы можно разбить на несколько основных блоков: сравнительный анализ существующих средств обработки естественного языка и анализа текстовых сообщений, сравнительный анализ алгоритмов классификации текстовых сообщений, применение алгоритма deep learning для русскоязычных предложений и реализация программы комплексного анализа русскоязычных текстовых сообщений на платформе IBM InfoSphere Streams.

За основу классификации доступных программных средств были взяты уровни естественного языка: фонетический, морфологический, лексический, синтаксический и семантический. Отдельным блоком были рассмотрены программы определения тональности текста (сентимент анализ - частный случай задачи классификации), интегрированные программные пакеты и модули анализа текста в составе широко-профильных статистических пакетов. В качестве критериев сравнительного анализа выступили как функциональные, так и нефункциональные требования.

Функциональные критериии сравнительного анализа:

- Корректность выполняемого анализа;
- Нефункциональные критерии:
- Стоимость решения;
- Лицензия, под которой распространяется решение;
- Платформы, на которых решение может быть запущено;
- Наличие пользовательского интерфейса;
- Наличие оберток (wrappers);
- Естественные языки, которые возможно проанализировать с помощью программного средства;

Количество программ и библиотек, доступных для анализа русского языка, уступает количеству программных средств, специализирующихся на задачах анализа английского языка. Часть многообещающих отечественных разработок находится в закрытом доступе: ЭТАП-3, Abbyy Campreno, Solarix. Другая часть хоть и доступная

только для использования в некоммерческих целях: Mystem и Tomita Parser, разработанные компанией Яндекс.

Для решения задачи морфологического анализа, определения части речи и генерации гипотез по словам с опечатками стоит отметить средство Mystem. Для решения задачи построения дерева синтаксического разбора наиболее продвинутым, из доступных средств, является AOT (для построения грамматики зависимостей) и средство LinkGrammar (для построения грамматики составляющих). Из интегрированных пакетов наиболее привлекательными являются AOT, NLTK и Stanford CoreNLP. Последние два упомянутые средства не поддерживают русский язык, но заслуживают внимание из-за наличия в открытом доступе исходного кода, обширной документации, демонстрационных стендов и мощной научной базы, подкрепленной множеством научных публикаций, статей.

Из средств извлечения фактов (именованных сущностей, адресов, дат и прочих шаблонов) стоит выделить Яндекс Томита Парсер. Программа работает с правилами контекстно-свободной грамматики, которые пользователь составляет самостоятельно. Помимо терминальных и нетерминальных символов Томита Парсер позволяет использовать пометы - специальные конструкции, определяющие тип связи между синтаксическими конструкциями, а также позволяющие конкретизировать морфологические признаки, определить формат слов с помощью регулярных выражений. Наличие модуля морфологического анализа в Томита Парсер позволяет сократить словарь нетерминалов в несколько раз. Это происходит за счет приведения к нормальной форме всех словоформ. Подобные инструменты, интегрированные в InfoSphere Streams не позволяют этого делать, что делает их использование трудоемким для русского языка.

Пакет анализа данных RapidMiner, дополненный модулем анализа текст TextMiner, является наиболее интуитивно понятным и и доступным средством анализа текста как для начинающих, так и для продвинутых пользователей. Средство привлекательно с нескольких точек зрения: во-первых, процесс анализа данных представлен в GUI с помощью отдельных блоков-операторов, соединенных в порядке, соответствующем порядку проведения анализа. Во-вторых, программное средство обладает большим набором встроенных операторов, существует возможность менять

параметры обучения, кластеризации, модифицировать имеющиеся операторы или создавать собственные. RapidMiner распространяется бесплатно, но есть ограничение по оперативной памяти.

Перед реализацией программы анализа текстовой информации был проведен сравнительный анализ алгоритмов классификации неструктированного текста. Были рассмотрены различные способы предварительной обработки текста. В качестве контрольной задачи рассматривалась задача определения тональности текста потребительских отзывов. В экспериментах участвовали следующие методы классификации:

- Тривиальный метод, основанный на определении категории по наличию ключевых слов;
- Наивный Байесовский классификатор;
- Классификация, основанная на результатах кластеризации;
- Метод опорных векторов;
- Нейронные сети (многослойные нейронные сети прямого распространения сигнала);

Помимо методов классификации были рассмотрены различные способы структурирования, канонизации и предварительной обработки текста:

- стемминг и лемматизация;
- использование n-грамм;
- удаление стоповых слов;
- частотные фильтры;
- использование синтаксически связанных групп в качестве токенов;
- представление в векторной форме: бинарное, частотное, с использованием нормализации весов TF-IDF;
- представление в виде семантических векторов;
- представление в виде деревьев синтаксического разбора (с формой хранения TreeBank);

Кроме привычных для решения задач классификации и кластеризации метрик оценки качества (точность, F-мера), также рассмотрена зависимость между точностью классификации и размером обучающей выборки, количеством категорий и точностью классификации, размером обучающей выборки и размером словаря.

Стоит отметить, что при выборе метода анализа стоит учесть простоту реализации метода и возможность улучшения модели.

Наилучшие результаты показал многослойный персептрон и наивный байесовский классификатор, работающие с представлением неструктурированного текста в виде набора канонизированных слов и синтаксически связанных групп. Наиболее перспективным для изучения является метод классификации текста, использующий представление текста в виде аннотированного дерева синтаксического разбора и методологии deep learning.

Указанный выше метод, предложенный группой ученых Стенфорского Университета и разрабатываемый в рамках проекта Stanford CoreNLP рассмотривался обособленно от других методов. Метод использует построение деревьев синтаксического разбора, тем самым метод может обнаруживать противоречия и противопоставления в пользовательских текстах. Методы, работающие с векторными представлениями, как правило, не учитывают связь слов в предложении. К недостаткам метода можно отнести: необходимость аннотирования огромного количества деревьев синтаксического разбора, продолжительное обучение многослойной нейронной сети (обучение может достигать нескольких суток), сложная для понимания структура нейронных сетей с множеством скрытых слоев делает процесс интерпретации результатов неконтролируемым.

Система анализа тональности текста от Stanford CoreNLP использует не бинарную категоризацию, а 5 различных категорий от очень негативного до очень позитивного. Однако средство не работает с русским языком. Проведенные исследования пользовали разработать модуль для применения описанного выше метода для русского языка. Обучение проходило на небольшой обучающей выборке, включающей в себя конструкции противопоставления. Тестирование показало, что обучение прошло успешно. Классификатор смог “запомнить” синтаксические шаблоны и успешно идентифицировать их при тестировании.

Проведенные исследования и эксперименты нашли применение в решении задачи комплексного анализа пользовательских отзывов. Заказчик прототипа системы - Сбербанк России. Главный подрядчик - IBM. Изначально задача состояла в разработке стенда для демонстрации конкурентного преимущества компании - анализа потребительского мнения в социальных сетях и на специализированных форумах. Позже проводилось обсуждение по использованию модуля анализа текста в службе поддержки клиентов. Подобная система способна ускорить работу операторов, занимаясь автоматической сортировкой обращений пользователей.

Результатом работы классификатора является:

- Анализ тональности текстового сообщения;
- Определение продукта Сбербанка: работа банкоматов, мобильный банкинг, Сбербанк Онлайн;
- Определение характеристики, упомянутой в сообщении: доступность, качество соединения, покрытие, функциональность, безопасность, скорость работы, качество поддержки. Некоторые из этих параметров являются строго соотносимыми с типами продуктов.

Система была разработана на платформе потоковой обработки информации *IBM InfoSphere Streams.* Решения на этой платформе состоят из операторов, которые разработаны на языке программирования Java. Описание связи между операторами осуществляется на domain specific language - SPL (Stream Processing Language).

Для разработки решения было решено использовать сторонние компоненты, выбранные на основе предварительно проведенного сравнительного анализа методов и программных компонент.

Выбранная методология включала в себя следующие шаги анализа текста и его предобработки:

- Приведение текста к одному регистру;
- Удаление шумовых и стоповых слов;
- Канонизация текста;

- Выявление синтаксически связанных структур;
- Частотная фильтрация;
- Отдельный наивный байесовский классификатор для тональности;
- Отдельный наивный байесовский классификатор для определения продукта;
- Отдельный наивный байесовский классификатор для определения характеристики продукта;

Точность классификации сравнима с аналогами для англоязычных текстовых сообщений:

1. определение тональности - 77-82% (в зависимости от продукта);
2. определение продукта - 75-80%;
3. определение свойств - 72-75%;

Можно отметить, что качестве классификации напрямую зависит от количества категорий.

В качестве сторонних библиотек было решено использовать два компонента:

1) Библиотеки AOT были скомпилированы как нативный оператор InfoSphere Streams. Средство AOT используется для решения задачи нормализации текста и выделения синтаксически связанных групп.
2) Средство интеллектуального анализа Weka было использовано для обучения наивного байесовского классификатора и сохранения классификатора; Weka распространяется под лицензией GNU GPL, обладает исчерпывающей и доступной документацией.

Для обучения классификаторов были выгружены более 50.000 отзывов из социальных сетей, портала Сбербанка, специализированных форумов, а также отзывы по мобильным приложениям из AppStore и Google Play (специально для формирования обучающей выборки по мобильным продуктам); Разработанная система обладает модульной структурой - для использования нового классификатора достаточно заменить несколько текстовых файлов: файл стоповых слов и файл со словарем и весами(получается после обучения наивного байесовского классификатора).

Таким образом, в ходе выполнения выпускной квалификационной работы была достигнута цель работы - реализована система комплексного анализа текстовых сообщений. Полученное средство способно приводить неструктурированный текст в

каноническую форму, очищать от шумов и решать задачи классификации. Решены задачи сравнительного анализа методов и инструментов text mining, которые позволили выбрать сторонние средства и методы для решения практической задачи.

Отдельное внимание стоит уделить исследованию методов глубокого обучения и представления текста в виде банка деревьев синтаксического разбора. Подобная комбинация помогает перейти на качественно новый уровень анализа текста. Представление в виде дерева синтаксического разбора сохраняет информацию о связи слов в предложении, что позволяет выявлять противоречия и отрицания. Впервые подобный классификатор был обучен для русского языка, но для получения корректных оценок и результатов требуется объемные обучающие выборки. Обучение полнофункционального классификатора может стать частью дальнейшей работы. По оценкам Стенфордского Университета точность указанного метода может достигать 95-97%.

Перспективами развития системы является реализация классификации на основе глубокого машинного обучения с учетом синтаксической структуры предложения. Такой подход позволит улучшать результат классификации с ростом обучающей выборки, а так же лучше учитывать взаимосвязь слов в тексте. Перспективами уже разработанного инструмента является ввод в эксплуатацию, расширение обучающих выборок и распространение на другие предметные области.

Источники литературы

1. J. Bullas, *New Twitter Facts, Figures and Growth Statistics plus [INFOGRAPHIC]*. [Электронный ресурс]. Режим доступа: http://www.jeffbullas.com/2011/09/21/11-new-twitter-facts-figures-and-growth-statisticsplus-infographic/.
2. J. Manyika et al., *Big data : The next frontier for innovation , competition and productivity*, 2011.
3. JISC (2012) *The Value and Benefit of Text Mining to UK Further and Higher Education*. Digital Infrastructure. Available at: http://bit.ly/jisc-textm Programme: Digital Infrastructure www.jisc.ac.uk/whatwedo/programmes/di_directions.aspx
4. Data Mining conferences worldwide [Электронный ресурс] /Conference Alerts - Режим доступа: http://www.conferencealerts.com/data.htm
5. Ananiadou, S. *The National Centre for Text Mining: a Vision for the Future*, 2007
6. J. Thomas and A. O'Mara-Eves, *Reconceptualising searching and screening: How new technologies might change the way that we identify studies*, in 19th Cochran Colloquim, 2011.
7. S. Ananiadou, D. B. Kell, and J.-ichi Tsujii, *Text mining and its potential applications in systems biology*., Trends in biotechnology, vol. 24, no. 12, pp. 571-9, Dec. 2006.
8. Stanford Humanities Center, *Digging Into The Enlightenment*. [Электронный ресурс]. Режим доступа: http://enlightenment.humanitiesnetwork.org/.
9. J.-D. Kim, T. Ohta, K. Oda, and J. Tsujii, *From Text To Pathway:Corpus Annotation for Knowledge Acquisition From Biomedical Literature*, Proceedings of The 6th Asia-Pacific Bioinformatics Conference, vol. 11, no. 1, pp. 165-175, 2008.
10. T. Ozasa, G. Weir, and M. Fukui, *A software application for assessing readability in the Japanese EFL context*, Themes in Science and Technology Education, vol. 3, no. 1&2, pp. 91-118, 2010.
11. JISC, *Social media 'not to blame' for inciting rioters*.[Электронный ресурс]. Режим доступа: http://www.jisc.ac.uk/news/stories/2011/12/riot.aspx.
12. K. Sparck Jones, *Natural language processing: a historical review*, in Current Issues in Computational Linguistics: in Honour of Don Walker, Amsterdam: Kluwer, 1994, 3-16

13. G. E. Hinton., *Learning multiple layers of representation*, Trends in Cognitive Sciences, 11, pp. 428–434, 2007.
14. *Unsupervised Feature Learning & Deep Learning*. [Электронный ресурс]. Режим доступа: http://deeplearning.stanford.edu/
15. Конференция Диалог. Соревнования парсеров. [Электронный ресурс]. Режим доступа: http://www.dialog-21.ru/parser/
16. Проект ВААЛ[Электронный ресурс]. Режим доступа: http://www.vaal.ru/
17. Porter, M. An algorithm for suffix stripping. / Porter, M. //Readings in Information Retrieval/San Francisco, CA, 1997: Morgan Kaufmann Publishers. – C. 313-316.
18. Большакова Е.И., Клышинский Э.С., Ландэ Д.В.,Носков А.А., Пескова О.В., Ягунова Е.В. *Автоматическая обработка текстов на естественном языке и компьютерная лингвистика.* : учеб. пособие /. — М.: МИЭМ, 2011. — 272 с.
19. Журнал "Системный Администратор" Архив номеров / 2002 / Выпуск №1 (1) / «Стеммер». Морфологический анализ для небольших поисковых систем
20. Segalovich I. "A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine.", MLMTA-2003
21. Многоцелевой лингвистический процессор ЭТАП-3[Электронный ресурс] / Режим доступа http://www.iitp.ru/ru/science/works/452.htm
22. Ruscorpa. Национальный корпус русского языка. [Электронный ресурс] / Режим доступа: http://www.ruscorpora.ru/
23. Сокирко А.В. *Семантические словари в автоматической обработке текста: По материалам системы ДИАЛИНГ*: диссертация кандидата технических наук: 05.13.17. – Москва, 2001. – 120 с.: ил.
24. Протасов С.В. Преимущества грамматики связей для русского языка // Труды международного семинара Диалог'2005, М., 2005.
25. Dodds PS, Harris KD, Kloumann IM, Bliss CA, Danforth CM (2011) Temporal Patterns of Happiness and Information in a Global Social Network: Hedonometrics and Twitter. PLoS ONE 6(12)
26. The Mood Map [Электронный ресурс] / Режим доступа (http://themoodmap.co.uk/)

27. Lopez R., Tejada J., Thelwall M.: *Spanish Sentistrength as a tool for Opinion Mining Peruvian Facebook and Twitter*. Artificial Intelligence Driven Solutions to Business and Engineering Problems (ITHEA 2012), pp.82-85, 2012.
28. Go, A., Bhayani, R., & Huang, L. *Twitter sentiment classification using distant supervision*. CS224N Project Report, Stanford, 1-12., 2009.
29. Plutchik, R. *Emotions: A general psychoevolutionary theory. Approaches to emotion*, 1984, 197-219, 1984
30. Яндекс Технологии. Томита Парсер. [Электронный ресурс] / Режим доступа: http://api.yandex.ru/tomita/
31. The Stanford Natural Language Processing Group [Электронный ресурс] / Режим доступа: http://nlp.stanford.edu/software/corenlp.shtml
32. Sentiment Analysis. Live demo[Электронный ресурс] / Режим доступа: http://nlp.stanford.edu/sentiment/
33. АОТ Автоматическая Обработка Текстов [Электронный ресурс]/ Режим доступа: http://aot.ru/technology.html
34. SPSS Price [Электронный ресурс]/ IBM – Режим доступа: http://www-142.ibm.com/software/products/ru/ru/spss-modeler/
35. SPSS Modeler [Электронный ресурс]/ IBM – Режим доступа: www-01.ibm.com/software/analytics/spss/products/modeler/professional.
36. Products & Solutions / Text Mining . [Электронный ресурс]/ SAS Institute Inc. – Режим доступа: http://www.sas.com/text-analytics/text-miner/index.html
37. Feinerer, I., Hornik, K. & Meyer, D. Text mining infrastructure in R. / Feinerer, I., Hornik, K. & Meyer, D. //Journal of statistical software, 25(5). – 2008. - American Statistical Association
38. Русские стоп слова. [Электронный ресурс]/ Antula: профессиональная студия веб-дизайна. – Режим доступа: http://www.antula.ru/noise-word_2.htm
39. Stop words List 1 [Электронный ресурс]/ Lextek International. – Режим доступа: http://www.lextek.com/manuals/onix/stopwords1.html
40. W. Lowe *Towards a theory of semantic space* in Proceedings of the Twenty-first Annual Meeting of the Cognitive Science Society LEA pp.576-581, 2001
41. Tesnière, L. *Elements de syntaxe structurale.* Klincksieck, Paris. 1959

42. Martin, S., Liermann, J., & Ney, H. Algorithms for bigram and trigram word clustering1. Speech Communication, 24(1), 19-37. 1998.
43. Blei, D. M., Ng, A. Y., & Jordan, M. I. *Latent dirichlet allocation*. the Journal of machine Learning research, 3, 993-1022., 2003.
44. Теслец Я. Г. *Введение в общий синтаксис*. М.: РГГУ, 2001. — 798 с
45. Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993*). Building a large annotated corpus of English: The Penn Treebank*. Computational linguistics, 19(2), 313-330.
46. Rish, I., *An empirical study of the naive Bayes classifier*./ Rish, I., // IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence – 2001.
47. Rummelhart D.E., Hilton G.E., Williams R.J. Learning internal representations by error propagation./ Rummelhart D.E., Hilton G.E., Williams R.J. – In McClelland ct al., 1986
48. Социальные медиа о Сбербанке [Электронный ресурс] / Режим доступа: http://www.sbforum.ru/board.php http://www.banki.ru/forum/
49. XPath Tutorial [Электронный ресурс] / Режим доступа: http://www.w3schools.com/XPath/
50. IBM Inc. Why InfoSphere? [Электронный ресурс] / Режим доступа: http://www-01.ibm.com/software/data/infosphere/
51. Socher R., Perelygin A., Wu J., Chuang J., Manning C., Ng A. and Potts C.. *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*, Conference on Empirical Methods in Natural Language Processing (EMNLP 2013, Oral)