Национальная Академия Наук

Кыргызской Республики

Институт Автоматики и Информационных Технологий

Лаборатория ИИС

**Реферат**

**Cистема анализа тональности текстов на основе LSTM-классификатора текстов SentiMe,**

**ПС SentiMe**

Автор:

м.н.с. Сороковая А.В.

Бишкек 2019

**Оглавление**

[Введение 3](#_Toc15680617)

[1. Задача классификации текстов 3](#_Toc15680618)

[2. Цели создания системы, решаемые задачи и область применения 5](#_Toc15680619)

[3. Функциональность программного продукта 5](#_Toc15680620)

[4. Ограничения 9](#_Toc15680621)

[5. Архитектура 9](#_Toc15680622)

[5.1. Модуль библиотеки для анализа тональности текстов 9](#_Toc15680623)

[5.1.1. Архитектура 9](#_Toc15680624)

[5.1.2. Основной сценарий использования 11](#_Toc15680625)

[5.2. Модуль Web API для доступа к ПС SentiMe 11](#_Toc15680626)

[5.2.1. Архитектура Web-API 11](#_Toc15680627)

[5.2.2. Архитектура модуля 12](#_Toc15680628)

[5.2.3. Основные сценарии использования 15](#_Toc15680629)

[5.3. Общая архитектура системы анализа тональности текстов на основе LSTM-классификатора текстов SentiMe 16](#_Toc15680630)

[5.3.1. Диаграмма компонентов 16](#_Toc15680631)

[5.3.2. Диаграмма развертывания 17](#_Toc15680632)

[6. Средства разработки 18](#_Toc15680633)

[7. Краткое руководство пользователя 18](#_Toc15680634)

[Заключение 19](#_Toc15680635)

[Список использованной литературы 21](#_Toc15680636)

# **Введение**

Стремительное расширение глобальной информационной сети порождает увеличивающийся поток информации разного рода. Для структуризации информации на естественном языке необходим его автоматический анализ. В задачах обработки естественного языка хорошо зарекомендовали себя искусственные нейронные сети – математические модели, используемые для обработки информации подобно нервной системе человека. Они способны анализировать входящую информацию, запоминать и воспроизводить из памяти образы, и поэтому нашли широкое распространение в анализе текстовой информации.

Cистема анализа тональности текстов на основе LSTM-классификатора текстов SentiMe решает в качестве задачи классификации задачу анализа тональности текстов, заключающуюся в определении эмоциональной оценки авторов текстов к объектам, описывающимся в текстах. Любой пользователь глобальной информационной сети может высказывать свои мнения относительно товаров и услуг, новостей, событий и других тем. Для анализа общей оценки суждений полезно оперативно и качественно анализировать их тональность, выявлять нужные данные в зависимости от сферы деятельности. Исследование субъективного образа, сформированного на основе анализа тональности, является важнейшей составляющей обеспечения эффективной политики и бизнеса, оценки эффективности маркетингового продвижения и рекламных кампаний, выбора целевой аудитории.

# **1. Задача классификации текстов**

Задача классификации в общем смысле представляет собой задачу разделения множества объектов на классы [1]. Классификация требует наличия так называемой обучающей выборки – конечного множества объектов, для которых классы заранее определены экспертом или иным способом. Задача классификации состоит в построении алгоритма для определения классовой принадлежности объектов вне обучающей выборки.

Математическая постановка задачи классификации. Дано множество описаний объектов и множество номеров (наименований) классов . Описание объекта представляет собой вектор признаков , называемый признаковым описанием объекта . Существует неизвестное отображение , значения которой определены на обучающей выборке. Требуется построить алгоритм , способный классифицировать любой объект .

Классификация текстов, или классификация документов, - одна из разновидностей задачи классификации объектов, заключающаяся в отнесении текста к некоторым категориям на основании их содержания [2].

Cистема анализа тональности текстов на основе LSTM-классификатора текстов SentiMe решает задачу классификации как задачу машинного обучения.

Основные этапы построения и обучения классификатора текста:

- предобработка текстов – выделение значимых единиц (слов);

- индексация текстов – получение признакового описания;

- собственно, построение и обучения классификатора;

- оценка качества классификации.

Частной задачей классификации текста является анализ его тональности. Анализ тональности текста - процесс вычислительной идентификации и категоризации мнений, выраженных в части текста для определения того, является ли отношение автора текста к определенной теме, продукту и т.д. положительным, отрицательным или нейтральным [3]. В настоящее время для автоматического анализа тональности текста чаще всего используется одномерное эмотивное пространство: позитивная и негативная составляющие текста. Однако ведутся и уже получены успешные результаты автоматического анализа тональности в многомерных пространствах оценки.

Cистема анализа тональности текстов на основе LSTM-классификатора текстов SentiMe в качестве задачи классификации текстов рассматривает задачу анализа тональности текстов. Анализ тональности текстов полезен для формирования обобщенного эмоционального мнения об объекте исследования и активно применяется в маркетинге для анализа эффективности продвижения товаров и услуг, в бизнесе для анализа реакции целевой аудитории на различные бизнес-идеи, в психологии и других областях человеческой деятельности.

Для решения задачи решено применить методы векторного представления слов для индексации текстов, т.к. их основными преимуществами являются поддержка контекстуального сходства слов и отображение слов в низкоразмерное пространство [4]. Использованы методы векторного представления слов: классический латентно-семантический анализ (LSA) [5], широко используемый Word2Vec [6], мало используемый в русскоязычных проектах Global Vectors (GloVe) [7].

Для классификации решено использовать LSTM-сети как сети, позволяющие учитывать порядок слов в тексте и выявлять зависимости между далеко расположенными словами в контексте [8].

# **2. Цели создания системы, решаемые задачи и область применения**

Неавтоматизированный анализ тональности текстов приводит к большим затратам времени и денежных средств.

Разработанная система позволяет:

* ускорить процесс анализа тональности текстов,
* переложить обязанности аналитика на программное обеспечение,
* сократить затраты на анализ текстовых выборок.

Разработанная система может применяться для анализа комментариев в социальных медиа, для оценки отношения клиентов компаний к их продуктам и услугам, для сбора статистических данных о мнении населения относительно каких-либо тем социальной сферы и в других областях, в которых необходима оценка эмоциональной реакции человека.

# **3. Функциональность программного продукта**

Функциональные требования определяются целями и задачами программной системы. Система выполняет следующие функции:

* Создание и расширение словарей языков;
* Создание и обучение моделей векторного представления слов языка;
* Обучение нового классификатора на основе созданных моделей векторного представления слов языка;
* Использование выбранного классификатора для определения тональности текстов из файлов и/или из комментариев под Instagram-постами;
* Управление конфигурацией системы (подключение новых методов векторного представления слов и классифицирующих алгоритмов).

Исходя из анализа предметной области выявлены актеры, которые будут взаимодействовать с программным продуктом: администратор и клиентское приложение. Система разработана для 2 групп пользователей:

- Администратор системы. Имеет имеет общее представление об аспектах использования и оценки готовых нейросетевых и классифицирующих решений.

- Клиент системы. Особые требования не предъявляются.

Система должна работать в 3 основных режимах работы:

- Управление системой (желтый, код A\*);

- Обучение (синий и серый, коды L\*, LC\*);

- Классификация (красный и серый, коды С\*, LC\*).

Функциональные требования отражены на диаграмме, представленной на рис. 1. Их подробная спецификация представлена в табл. 1.



Рисунок 1. Варианты использования системы определения тональности текста.

Таблица 1

Спецификация вариантов использования

|  |  |  |  |
| --- | --- | --- | --- |
| **Код** | **Наименование** | **Основной актер** | **Формулировка** |
| A1 | Подключить метод векторного представления | Администратор | Для подключения метода векторного представления администратор должен иметь файл-плагин (.py – код на языке Python), реализующий интерфейс метода векторного представления слов. |
| A2 | Подключить архитектуру сети | Администратор | Для подключения архитектуры сети администратор должен иметь файл-плагин (.py – код на языке Python), реализующий интерфейс класса сети. |
| L1 | Создать словарь языка | Администратор | Администратор выбирает метод предобработки текстов (надо ли удалять стоп-слова, приводить к нижнему регистру, выделять основу слова, а также метод выделения основы слова), загружает файлы с текстами (если это csv файлы, то указывает разделитель и номер поля, в котором указываются тексты). После этого система считывает тексты из файлов, выделяет слова в соответствии с правилами словаря и формирует словарь языка. |
| L2 | Расширить/ сократить словарь языка | Администратор | Администратор выбирает слова для удаления из словаря языка или загружает дополнительные файлы с текстами, после чего система считывает тексты и добавляет новые найденные слова с словарю языка. |
| L3 | Удалить словарь языка | Администратор | Администратор удаляет словарь из системы, вместе со словарем не должны удаляться модели векторного представления, созданные на их основе. |
| L4 | Создать модель векторного представления | Администратор | Администратор выбирает метод обучения модели векторного представления слов, вводит все необходимые для метода параметры, выбирает словарь языка (слова вне языка не будут иметь векторного представления), загружает файлы с текстами (если это csv файлы, то указывает разделитель и номер поляс текстами). После этого система считывает тексты из файлов, выделяет слова в соответствии с правилами словаря и обучает модель векторного представления текстов. |
| L5 | Посмотреть данные о модели векторного представления текста | Администратор | Система отображает данные, введенные пользователем при создании модели векторного представления и отображает вектора слов на двумерной плоскости. |
| L6 | Удалить модель векторного представления | Администратор | Администратор удаляет модель векторного представления из системы, вместе с ней удаляются все классификаторы, обученные на ее основе. |
| L7 | Обучить классификатор | Администратор | Администратор выбирает архитектуру сети для обучения классификатора, вводит все необходимые для архитектуры параметры, выбирает модель векторного представления слов, загружает файлы с обучающей выборкой, указывает разделитель и номера полей с текстом и меткой тональности, указывает какая метка соответствует негативной и позитивной тональностям. После этого система обучает классификатор на основе введенных данных. |
| L8 | Посмотреть данные о классификаторе | Администратор | Администратору отображаются введенные при создании классификатора данные, а также точность и полнота обучения на каждой из эпох обучения классификатора, общий прогресс обучения по эпохам. |
| L9 | Удалить классификатор | Администратор | Администратор удаляет классификатор из системы, вместе с ним не удаляются результаты определения тональности, полученные с его помощью. |
| C1 | Определить тональность текстов | Клиентское приложение | Клиентское приложение отправляет запрос на определение тональности текстов, указывает название набора, а также набор ссылок и файлов, откуда система считывает тексты. |
| C2 | Посмотреть результаты определения тональности текстов | Клиентское приложение | Клиентское приложение получает тексты из набора с определенной тональностью. |
| C3 | Удалить результаты определения тональности текстов | Клиентское приложение | Клиентское приложение отправляет запрос на удаление результатов определения тональности. |

# **4. Ограничения**

*Аппаратно-программные ограничения:*

* ЭВМ, не менее 4 процессоров, 8 ГБ оперативной памяти, 100 ГБ свободного дискового пространства для сервера.
* Компьютер, имеющий доступ в сеть для клиента. На компьютере должно быть установлено программное обеспечение, позволяющее отправлять HTTP-запросы на сервер программной системы для анализа тональности текстов. Приложение может быть любого рода – командная строка, специализированные средства формирования HTTP-запросов или полноценное приложение, использующее Web API разработанной системы в своей основе.

*Архитектурные ограничения:*

* Трехуровневая архитектура, предполагающая наличие в ней трёх компонентов: клиентов (любая сторонняя система, использующая Web API системы анализа тональности текстов SentiMe), сервера приложений (к которому подключены клиентские приложения) и сервера баз данных (с которым работает сервер приложений).

# **5. Архитектура**

## **5.1. Модуль библиотеки для анализа тональности текстов**

**5.1.1. Архитектура**

Для простоты использования библиотеки для анализа тональности текстов ее интерфейс решено строить на примитивных типах данных. Таким образом компоненты, использующие ее, не знают о типах данных, которые она использует внутри.

Диаграмма классов, отображающая основные граничные классы и представлена на рис. 2. Классы, названные со строчной буквы, соответствуют модулям библиотеки. Классы, названные с заглавной буквы, соответствуют классам библиотеки.



Рисунок 2. Диаграмма классов модуля библиотеки для анализа тональности текста.

Краткое описание классов представлено в табл. 2.

Таблица 2

Краткое описание классов и модулей библиотеки

|  |  |  |
| --- | --- | --- |
| **Класс или модуль** | **Тип** | **Описание** |
| helpers | Управляющий | Содержит различные вспомогательные методы для работы с данными |
| csv | Управляющий | Содержит методы для чтения текстов и меток тональности из CSV-файлов, используя для чтения соответствующие атрибуты. |
| tokenizing | Управляющий | Описывает логику предобработки текста. |
| BaseEmbedding | Граничный | Определяет интерфейс, который должны реализовывать классы с логикой обучения моделей векторного представления слов. |
| LSTMCategorical | Управляющий | Описывает логику обучения классификатора на основе LSTM-сети. |
| ValidationCallback | Управляющий | Используется для записи результатов валидации модели на каждой эпохе обучения. |
| LoggingCallback | Управляющий | Используется для записи выходов и метрик модели на каждой эпохе обучения. |
| confusion\_matrix | Управляющий | Описывает логику построения матрицы ошибок из набора действительных и полученных классификатором меток классов, а так же логику расчета метрик на основе матрицы ошибок. |

Система может расширяться любыми методами предобработки и классификации, реализующими соответствующие интерфейсы.

### **5.1.2. Основной сценарий использования**

Для использования библиотеки необходимо использовать граничные классы. Основные варианты использования библиотеки:

* считывание из файла обучающей выборки;
* токенизация текста;
* обучение модели векторного представления текста;
* обучение классификатора для анализа тональности текста;
* использования классификатора для анализа текста.

## **5.2. Модуль Web API для доступа к ПС SentiMe**

**5.2.1. Архитектура Web-API**

Cистема анализа тональности текстов на основе LSTM-классификатора текстов SentiMe предоставляет Web API в качестве интерфейса для ее использования клиентскими приложениями. Использована REST (REpresentational State Transfer) [9] архитектуру для реализации Web-API. Архитектура REST разработана чтобы соответствовать [протоколу HTTP](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) используемому в сети Интернет. Вызов удалённой процедуры представляет собой обычный HTTP-запрос (такой запрос называют «REST-запрос»), а необходимые данные передаются в качестве параметров запроса.

**5.2.2. Архитектура модуля**

Web API разработан в соответствии с шаблоном MVC (Model-View-Controller) [10].

Для разделения процессов обработки запроса и обработки данных в системе из слоя контроллеров обработка данных выносится в слой бизнес-логики. Слой представления обрабатывается в стороннем клиентском приложении.

Таким образом для построения системы реализованы модели и контроллеры шаблона MVC. Общая архитектура Web API разделена на три слоя:

* слой доступа к базе данных, основанный на объектно-ориентированном отображении (ORM), содержит классы-сущности, отображающие состояние базы данных в состояние объектов этих классов-сущностей;
* слой бизнес-логики, содержащий всю логику обработки и изменения сущностей, проецируемых в базу данных;
* слой контроллеров, представляющий собой контроллеры для обработки запросов; представления определяются форматом данных, используемым для обращения к контроллерам.

Диаграмма классов, отображающая структуру слоя доступа к базе данных, представлена на рис. 3. Для большей читаемости диаграммы диаграмма сокращена до объема, достаточного для понимания архитектуры. Так же в диаграмме не отображены атрибуты, необходимые для установления соответствующих связей классов. Описание основных классов-сущностей дано в табл. 3.



Рисунок 3. Диаграмма классов-сущностей.

Таблица 3.

Основные классы сущностей в Web API

|  |  |  |
| --- | --- | --- |
| **Класс** | **Тип** | **Описание** |
| Vocabulary | Сущность | Представляет информацию о словаре языка. Для описания используются вспомогательные сущности. |
| Word | Сущность | Представляет информацию о слове языка, в первую очередь символьное представление слова. |
| EmbeddingMethod | Сущность | Представляет информацию о плагине системы, реализующем метод векторного представления слов. Перечисление параметров метода доступно в виде объектов EmbeddingMethodParam. |
| EmbeddingModel | Сущность | Представляет информацию об обученной модели векторного представления слов. Ассоциируется с методом векторного представления текста EmbeddingMethod. Значения параметров метода определяются объектами EmbeddingMethodParamValue. Определяет вектора каждого слова из ассоциированного словаря (WordVector). |
| NetModule | Сущность | Представляет информацию о плагине системы, реализующем метод классификации текста. Перечисление параметров метода доступно в виде объектов NetModuleParam. |
| Net | Сущность | Представляет информацию об обученном классификаторе. Строится на основе конкретной модели векторного представления слов EmbeddingModel. Значения параметров метода классификации NetModule определяются объектами NetModuleParamValue. Каждая эпоха обучения классификатора записывается в соответствующий объект NetEpoch. |
| NetEpoch | Сущность | Представляет информацию об отдельной эпохе обучения классификатора, в первую очередь перечень предсказаний классификатора после обучения на данной эпохе. Предсказания хранятся не в явном виде, а в виде обработанной матрицы ошибок NetEpochConfusion. |
| TextSet | Сущность | Представляет информацию о наборе текстов. Формирует набор текстов из файлов и ссылок на источники информации. Каждая ассоциация с источником информации характеризуется набором извлеченных текстов. Этот набор из текстов содержит тексты в виде объектов Text (наследник LinkText для ссылок содержит уникальный токен текста, наследник FileText содержит номер строки текста в файле). |

Слой бизнес-логики описывает алгоритмы создания, получения, изменения, удаления для каждой сущности, а также алгоритмы обработки данных, представленных объектом сущности. Слой представляет собой набор классов с методами, реализующими логику обработки соответствующего класса-сущности. Таким образом сущности слоя доступа к базе данных не знают ничего о логике своей обработки (за исключением ограничений базы данных и валидации), классы бизнес-логики могут управлять соответствующими сущностями и использовать другие классы управления для обработки сущностей вне своей компетенции. В качестве параметров и возвращаемых результатов классы бизнес-логики используют в основном экземпляры классов-сущностей.

Слой контроллеров представляет собой граничные классы, предоставляющие Web API интерфейс. Каждый граничный класс содержит методы обработки запроса. Запросы соответствуют сценариям использования системы. Методы классов слоя контроллеров расширяются вместе с расширением вариантов использования Web API. Методы этих классов должны только извлекать параметры запроса, использовать методы бизнес-логики для обработки данных и возвращать результаты обработки в соответствующем JSON-формате. Никакой логики обработки данных контроллеры содержать не могут. В качестве параметров и возвращаемых результатов граничные классы используют данные в формате JSON или полученные аргументы URL-запроса.

**5.2.3. Основные сценарии использования**

Для обработки данных необходимо:

1. отправить запрос на обработку данных контроллеру Web API;
2. в контроллере получить необходимые данные по заданным параметрам;
3. если операция выполняется достаточно быстро, обработать данные и возвратить результат обработки клиентскому приложению;
4. если операция выполняется достаточно долго, вернуть клиенту информацию о том, возможно ли выполнение операции (если возможно, обычно возвращается неизмененный объект класса-сущности) и вызвать асинхронную операцию обработки данных;
5. асинхронная операция сигнализирует о завершении путем установки соответствующего статуса в базе данных.

## **5.3. Общая архитектура системы анализа тональности текстов на основе LSTM-классификатора текстов SentiMe**

**5.3.1. Диаграмма компонентов**

Диаграмма компонентов системы для анализа тональности текстов представлена на рис. 4. Основные компоненты системы:

* библиотека для анализа тональности текстов SentimentAnalysis; при изменении модулей, ответственных за алгоритмы чтения, токенизации, векторного представления, классификации и оценки качества алгоритмов, должна происходить перекомпиляция библиотеки;
* Web API SentimentWebApi системы как отдельный исполняемый компонент, который обслуживает запросы клиента и использует библиотеку SentimentAnalysis для классификации и сопутствующих процессов. При изменении модулей слоя доступа в базе данных, слоя бизнес-логики или слоя контроллеров исполняемый модуль Web API должен быть перекомпилирован.

База данных как компонент формируется на основе слоя доступа в базе данных средствами Code First (с использованием средств alembic). При изменении сущностей в слое доступа к базе данных необходимо сформировать новую миграцию и применить ее.



Рисунок 4. Диаграмма компонентов системы.

**5.3.2. Диаграмма развертывания**

Самое главное достоинство сервисов в том, что с ними может работать какая угодно система, будь то сайт, flash, программа и др., так как методы парсинга и выполнения запросов HTTP реализованы на любых платформах (рис. 5).



Рисунок 5. Диаграмма развертывания системы.

# **6. Средства разработки**

В качестве языка программирования выбран язык Python. Python является популярным языком для систем с научным уклоном. Открытые репозитории содержат большое количество признанных мировым ученым сообществом пакетов: SciPy, Scikit-Learn, Pandas, NumPy.

Для реализации выбранных методов используются такие популярные библиотеки как NumPy, GenSim, TensorFlow, Keras, Matplotlib. Для предобработки текста так же используется библиотека для стемминга слов PyStemmer.

Python отличается скоростью разработки, в т.ч. за счет элементов функционального программирования, полезных для разработки небольших модулей программной системы. В пользу Python так же говорит его кроссплатформенность.

В качестве системы управления базами данных для хранения информации выбрана свободная реляционная система управления базами данных MySql. Благодаря внутреннему механизму многопоточности быстродействие MySQL весьма высоко. Безопасность обеспечивается встроенными средствами управления доступом и шифрования. Использование базы данных для некоммерческих целей бесплатно. Сообщество пользователей велико, существует огромное количество форумов, на которых легко и быстро можно получить консультацию по возникшим проблемам. Одним из главных преимуществ так же остается наличие версий СУБД для большинства распространенных компьютерных платформ.

Для использования СУБД при разработке используется объектно-ориентированное отображение базы данных в объекты классов-сущностей программ. Для этого применяется библиотека SqlAlchemy и коннектор PyMySql для Python.

Так как используется подход Code First, СУБД легко заменить практически на любую реляционную СУБД, для которой доступен соответствующий коннектор. Механизм миграции позволяет моментально восстановить структуру базы данных в любой из этих СУБД на основе моделей SqlAlchemy. Для создания миграции используется библиотека Alembic.

Web API системы построено с помощью библиотеки Flask.

# **7. Краткое руководство пользователя**

Так как программная система представляет собой REST API, то для каждой сущности системы можно формировать GET-, POST-, PUT-, DELETE-запросы.

* GET-запросы в данной программной системе предназначаются для получения сущности (если задан ее id) или получения всех доступных сущностей в системе.
* POST-запросы в данной программной системе предназначаются для создания новой сущности.
* PUT-запросы в данной программной системе предназначаются для отправки данных, из которых нужно получить необходимую информацию и обновить отдельную сущность.
* DELETE-запросы в данной программной системе предназначаются для удаления сущностей из системы.

Список доступных сущностей (GET, POST, PUT, DELETE):

* vocabulary – частотный словарь языка;
* embedding-model – модель векторного представления слов языка, формируется на основе словаря языка, содержит только представления слов из словаря;
* net – нейросетевой классификатор;
* text-set – набор текстов, анализируемых отдельным классификатором.

Список доступных сущностей (GET):

* language – доступный в системе язык словаря;
* normalization – доступный в системе метод нормализации слов;
* embedding-method – доступный в системе метод получения модели векторного представления слов;
* net-module – доступная в системе архитектура нейронной сети для классификации.

# **Заключение**

Cистема анализа тональности текстов на основе LSTM-классификатора текстов SentiMe предоставляет Web API для анализа тональности текстов. Для реализации системы анализа тональности текстов выбран язык Python в сочетании с популярными научными библиотеками: NumPy, GenSim, Keras и др.

Разработанная система позволяет:

* Обучать и использовать модели векторного представления слов;
* Обучать и использовать классификаторы для анализа тональности текстов;
* Подключать в виде плагинов другие методы векторного представления слов и классификации текстов, реализующие соответствующие интерфейсы.

В дальнейшем планируется наращивать функционал системы для анализа тональности текстов. Теоретически система может быть использована не только для анализа тональности текстов, но и для любых других задач классификации текстов.

Система была использована для исследования эффективности методов векторного представления слов. Для исследования эффективности методов векторного представления слов был разработан отдельный программный продукт. С его помощью проведены вычислительные эксперименты с использованием различных моделей векторного представления, проведен анализ эффективности этих моделей для русскоязычного и англоязычного корпусов текстов, получены результаты и описаны дальнейшие перспективы исследования [11].

Web API системы может быть использовано любым приложением. Web API уже использовано для разработки Web-приложения, позволяющего классифицировать наборы текстов. Web-приложение предоставляет графический интерфейс пользователя к Web API системы анализа тональности текстов [12].

# **Список использованной литературы**

1. Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. Методы и модели анализа данных: OLAP и Data Mining. – СПб.: БХВ-Петербург, 2004. – 336 с.: ил.
2. Классификация документов. Виды и методы. – URL: <https://studfiles.net/preview/4513655/page:6/>. (дата обращения 16.12.2017)
3. Кузнецов А.Л., Кочуров Д.Н. Анализ тональности текста в социальных медиа с использованием искусственных нейронных сетей // Студенческий научный форум, УрФУ, Екатеринбург, Россия. 2018.
4. Word Bags vs Word Sequences for Text Classification. – URL: <https://towardsdatascience.com/word-bags-vs-word-sequences-for-text-classification-e0222c21d2ec>. (дата обращения 01.02.2019)
5. Landauer T.K., Foltz P.W., Laham D. Introduction to Latent Semantic Analysis // Discourse Processes. 1998. Vol. 25. Pp. 259-284.
6. Le Q., Mikolov T. Distributed Representations of Sentences and Documents // Proceedings of the 31stInternational Conference on Machine Learning, Beijing, China. 2014. Vol. 32. Pp. 1188-1196.
7. Pennington J, Socher R, Manning CD. GloVe: Global Vectors for Word Representation // In EMNLP. 2014. Vol. 14. Pp. 1532-1543.
8. Understanding LSTM Networks. – URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. (дата обращения 14.09.2018)
9. Wilde E., Pautasso C. REST: From Research to Practice. // Springer Science & Business Media, 2011. — 528 p.
10. Model-View-Controller. – URL: https://ru.wikipedia.org/wiki/Model-View-Controller. (дата обращения 16.11.2018)
11. Сороковая А.В. Модели Векторного Представления Слов Для Обучения Классификатора На Основе Lstm-Сети // XI Международная конференция студентов и молодых ученых, НС РАН, Бишкек. 2019.
12. Сороковая А.В, Разработка нейросетевого классификатора текстов // Магистерская диссертация, КРСУ, Бишкек. 2019. – 116 с.