МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**РАЗРАБОТКА НЕЙРОННОЙ СЕТИ ДЛЯ КЛАССИФИКАЦИИ ТЕКСТОВ**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. С. Андреева

(подпись)

Направление подготовки 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль) Вычислительные технологии

Научный руководитель

канд. физ.–мат. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. Н. Лапина

(подпись)

Нормоконтролер

канд. техн. наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е. Е. Полупанова

(подпись)

Краснодар

2018

**СОДЕРЖАНИЕ**

Введение4

1 Теория нейронных сетей для решения задачи классификации6

1.1 Архитектура нейронной сети6

1.2 Этапы классификации8

1.3 Методы оценки успешности классификации9

2 Разработка и обучение нейронной сети для классификации новостей по категориям15

2.1 Выбор программного обеспечения15

2.2 Исходные данные для обучения нейронной сети16

2.3 Построение модели и подбор параметров для нейронной   
 сети16

2.4 Экспериментальные результаты19

Заключение22

Список использованных источников 23

Приложение А Файл основной программы разработки нейронной сети network.py 24

Приложение Б Парсинг сайта Яндекс.Новости parsing.py33

# ВВЕДЕНИЕ

Актуальность исследований, связанных с интеллектуальным анализом данных, обуславливается развитием средств сбора и хранения данных, позволившим накапливать большие объемы информации. Интеллектуальный анализ данных связан с поиском в данных ранее не известных и выгодных закономерностей, позволяющих получить новые знания об исследуемых данных. Из-за того, что большинство известных статистических методов позволяют удовлетворить только часть требований к обработке данных, и для их использования необходимо иметь четкое представление о закономерностях, встает вопрос об эффективной обработке проверяемых данных.

Развитие технологии баз данных и систем управления базами данных, способствует росту объема данных, хранящихся в базе. Эти данные содержат в себе много важной информации, которая имеет большой потенциал. Ввиду этого многие компании используют технологию интеллектуального анализа данных (data mining), которая позволяет обрабатывать массивные базы данных и извлекать из них полезную информацию. Преимущества нейронных сетей, такие как, высокая допустимость к зашумленным данным и низкий коэффициент ошибок, непрерывное усовершенствование и оптимизация различных алгоритмов обучения сетей, алгоритма извлечения правил, алгоритма упрощения сетей, делают нейронные сети все более и более перспективным направлением в data mining.

Вышеизложенные положения определяют актуальность темы работы.

Цель работы состоит в создании нейронной сети для классификации текстов и оценка её работы.

Достижение поставленной цели определяет постановку и решение следующих задач:

− поиск и извлечение информации из Интернет-ресурсов,

− парсинг данных с сайтов и сохранение материалов в базу данных,

− изучение и выбор модели нейронной сети для автоматической классификации текстов,

− построение нейронной сети и её обучение с использованием полученных данных,

− тестирование полученной модели,

− оценка успешности классификации.

Объектом исследования являются статьи из Интернет-ресурсов.

Предметом выступает нейронная сеть.

Теоретическая значимость работы заключается в проведении анализа корреляции объёма и качества входных данных с точностью классификации.

Практическая значимость работы состоит в автоматической классификации текстов по категориям на большом объёме данных.

**1 Теория нейронных сетей для решения задачи классификации**

**1.1 Архитектура нейронной сети**

[Искусственные нейронные сети](https://en.wikipedia.org/wiki/Artificial_neural_network) представляют собой [вычислительные модели,](https://en.wikipedia.org/wiki/Computational_model) основанные на [биологических нейронных сетях](https://en.wikipedia.org/wiki/Biological_neural_network), и используются для [приближения](https://en.wikipedia.org/wiki/Universal_approximation_theorem)[функций](https://en.wikipedia.org/wiki/Function_(mathematics)), которые обычно неизвестны.

Нейронные сети относятся к классу глубокого машинного обучения.

Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными [алгоритмами](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC). Технически обучение заключается в нахождении коэффициентов связей между нейронами. В общем виде, нейронные сети имеют несколько уровней нейронов, что позволяет выявить скрытые закономерности и более сложносоставные связи между входными данными [1]. Рассмотрим на рисунке 1 схему нейрона.

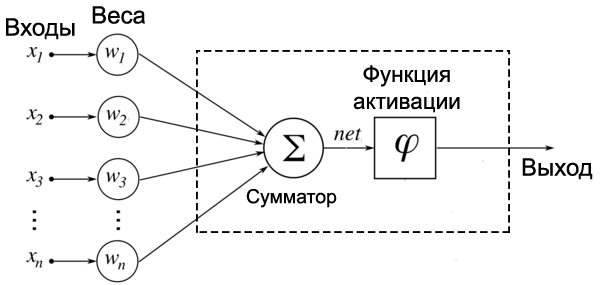


Рисунок 1 – Схема нейрона

Состояние нейрона можно выразить формулой:

, (1)

где n – число входов для нейрона;

xi – состояние входа нейрона;

wi – функция веса синапса.

В самом принципе построения современных нейронных сетей также заложено наличие нескольких уровней, каждый из которых состоит из нейронов, работающих с одинаковыми признаками (см. рисунок 2).

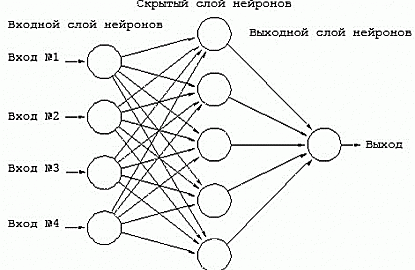


Рисунок 2 – Базовая структура нейронной сети

На вход нейросети подается вектор признаков, а на выходе мы получаем результат принадлежности примера тому или иному классу. При этом выходной уровень должен содержать столько нейронов, сколько классов у нас есть. В дальнейшем, в зависимости от поставленной перед нейронной сетью задачи, выбираются функции оптимизации и потерь. После чего, нейронная сеть итеративно и многократно проходит обучение на тестовой выборке, сохраняя лучшие результаты и корректируя веса признаков, для минимизации ошибок [2].

Многослойный персептрон (MLP) –  нейронная сеть прямого распространения сигнала (без обратных связей), в которой входной сигнал преобразуется в выходной, проходя последовательно через несколько слоев [3].

Первый из таких слоев называют входным, последний - выходным. Эти слои содержат так называемые вырожденные нейроны и в количестве слоев не учитываются. Кроме входного и выходного слоев, в многослойном персептроне есть один или несколько промежуточных слоев, которые называют скрытыми.

Метод обратного распространения ошибки (Back propagation, backprop) - алгоритм обучения многослойных персептронов, основанный на вычислении градиента функции ошибок. В процессе обучения веса нейронов каждого слоя нейросети корректируются с учетом сигналов, поступивших с предыдущего слоя, чтобы затем каждый вес скорректировать в соответствии с алгоритмом градиентного спуска. Вычисляется рекурсивно в обратном направлении от последнего слоя к первому.

Эпоха – это один полный цикл обучения нейросети на всех примерах обучающей выборки.

## **Этапы классификации**

Суть задачи классификации текста после его предварительной подготовки и очистки сводится к тому, чтобы составить словарь всех слов в текстах, заменить каждое слово на число – уникальный номер слова в словаре, выровнять длину каждого текста до нужного размера (обычно это количество слов в максимально длинном тексте). Алгоритм процесса классификации рассмотрим на рисунке 3.

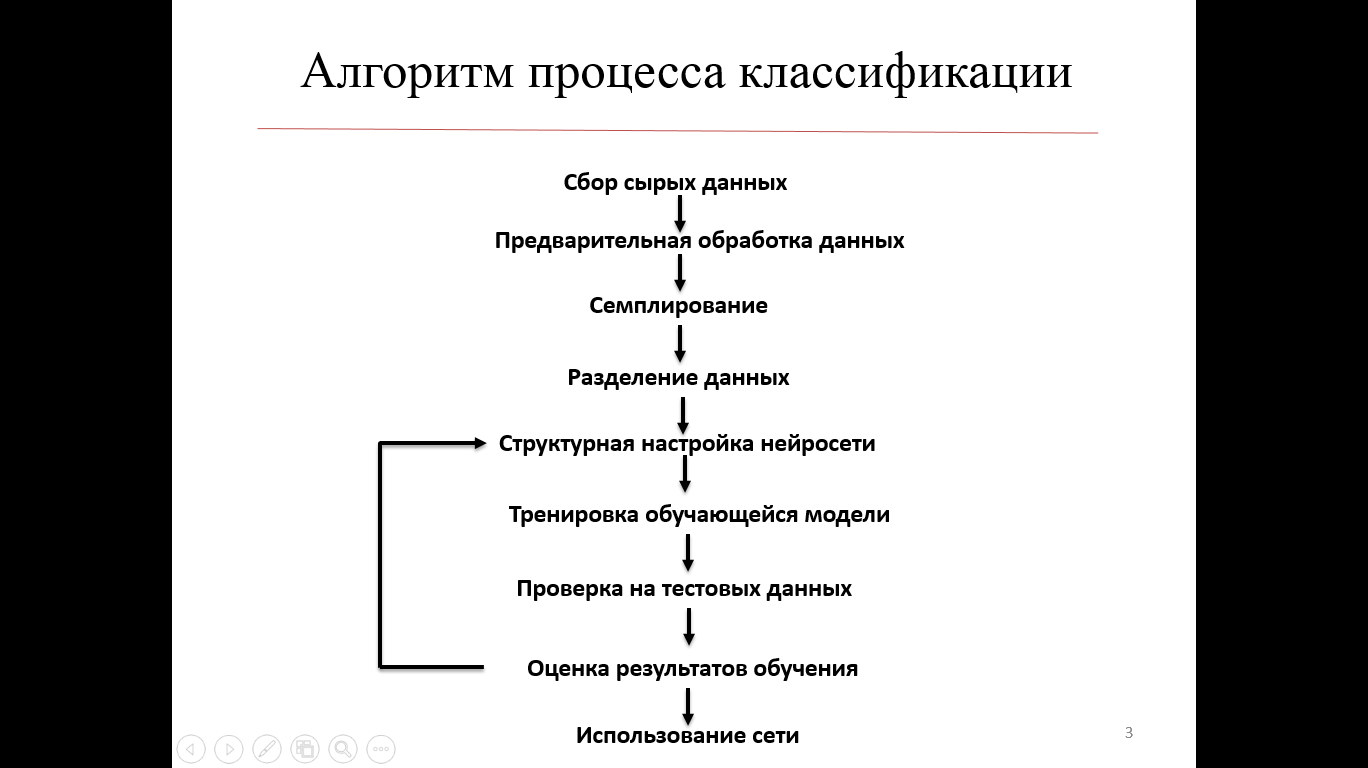


Рисунок 3 – Алгоритм процесса классификации

На этапе предварительной обработки данных мы должны проанализировать данные: посмотреть на их распределение, понять, есть ли в данных, которые мы будем анализировать отсутствующая информация, в каком виде эти данные представлены. Следом необходимо эти данные привести к единому виду.

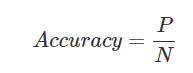
После очистки текста и разделения его на слова, необходимо разделить все строки с обращениями на два набора данных: данные для тренировки модели и данные для проверки ее работы.

## **1.3 Методы оценки успешности классификации**

Основой проверки является тестовая выборка, в которой проставлено соответствие между статьями и их категориями. Также, объем такой выборки не должен быть слишком малым, чтобы имелась возможность провести актуальное сравнение на длительной дистанции.

При наличии тестовой выборки, можно сравнить результаты классификаторов и соотнести их решения с заведомо известным правильным решением. Но для того чтобы принимать решение хуже или лучше справляется с работой новая версия алгоритма, либо формата входных данных, нам необходима численная метрика его качества.

В простейшем случае такой метрикой может быть доля текстов, по которым классификатор принял правильное решение – т.е. точность.

, (2)

где P – количество текстов, по которым классификатор принял правильное решение;

N – размер обучающей выборки.

Однако у этой метрики есть одна особенность, которую необходимо учитывать. Она присваивает всем документам одинаковый вес, что может быть не корректно в случае, если распределение документов в обучающей выборке сильно смещено в сторону какого-то одного или нескольких классов. В этом случае у классификатора есть больше информации по этим классам, и соответственно в рамках этих классов он будет принимать более адекватные решения. На практике это приводит к тому, что вы имеете точность, скажем, 80%, но при этом в рамках какого-то конкретного класса классификатор работает из рук вон плохо, не определяя правильно даже треть документов.

Один выход из этой ситуации заключается в том, чтобы обучать классификатор на специально подготовленном, сбалансированном корпусе документов. Минус этого решения в том, что вы отбираете у классификатора информацию об относительной частоте документов. Эта информация при прочих равных может оказаться очень кстати для принятия правильного решения.

Другой выход заключается в изменении подхода к формальной оценке качества – использование точности и полноты.

Точность (precision) и полнота (recall) являются метриками которые используются при оценке большей части алгоритмов извлечения информации. Иногда они используются сами по себе, иногда в качестве базиса для производных метрик, таких как F-мера или R-Precision. Суть точности и полноты очень проста.

Точность системы в пределах класса – это доля документов, действительно принадлежащих данному классу относительно всех документов, которые система отнесла к этому классу. Полнота системы – это доля найденных классификатором документов, принадлежащих классу относительно всех документов этого класса в тестовой выборке.

Эти значения легко рассчитать на основании таблицы контингентности, которая составляется для каждого класса отдельно (см. рисунок 4).

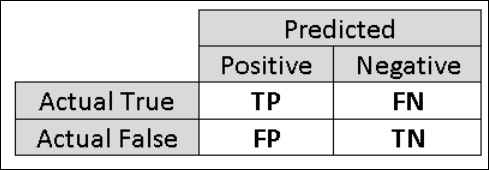
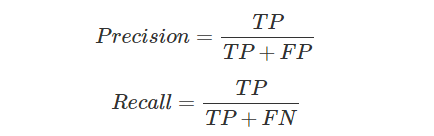


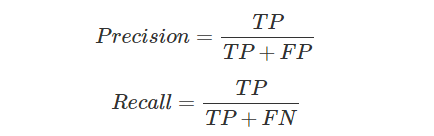
Рисунок 4 – Таблица контингентности

В таблице содержится информация сколько раз система приняла верное и сколько раз неверное решение по документам заданного класса. А именно:

* TP — истино-положительное решение,
* TN — истино-отрицательное решение,
* FP — ложно-положительное решение,
* FN — ложно-отрицательное решение.

Тогда точность и полнота соответствуют следующим формулам:

, (3)

 (4)

На практике значения точности и полноты гораздо удобнее рассчитывать с использованием матрицы неточностей. В случае если количество классов относительно невелико (не более 100-150 классов), этот подход позволяет довольно наглядно представить результаты работы классификатора.

Матрица неточностей – это матрица размера N на N, где N — это количество классов. Столбцы этой матрицы резервируются за экспертными решениями, а строки за решениями классификатора. Когда мы классифицируем документ из тестовой выборки мы инкрементируем число, стоящее на пересечении строки класса, который вернул классификатор и столбца класса к которому действительно относится документ.

Как видно из примера (см. рисунок 5), большинство документов классификатор определяет верно. Диагональные элементы матрицы явно выражены.

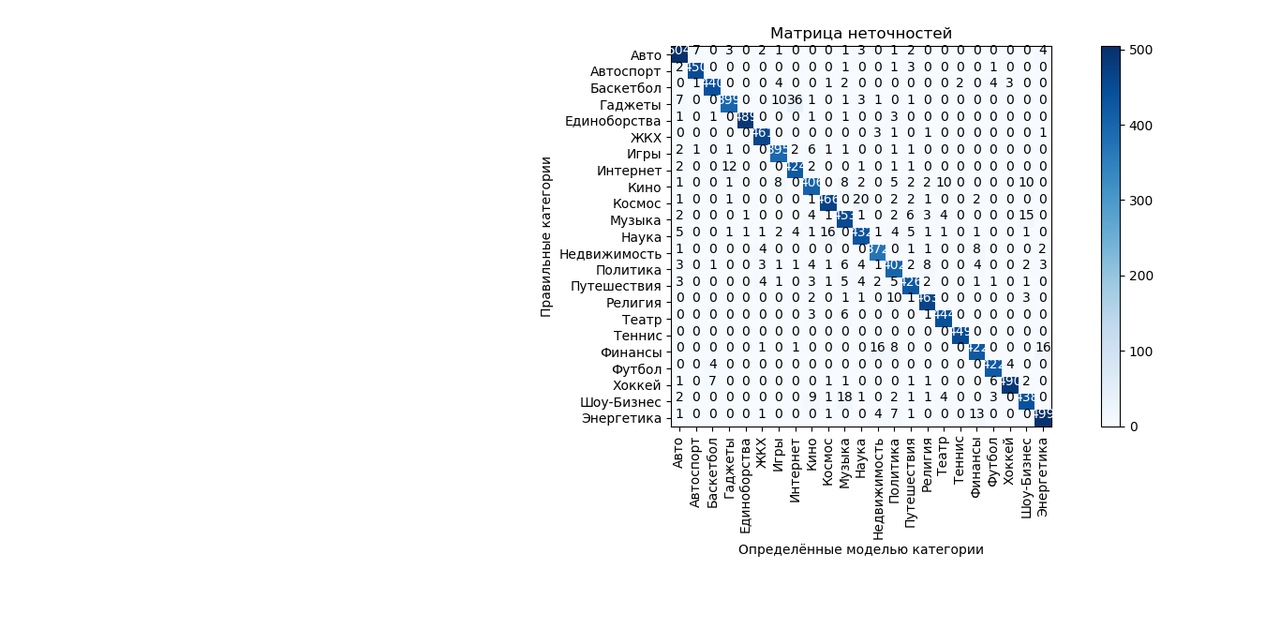
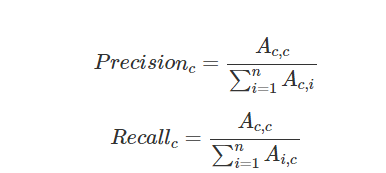
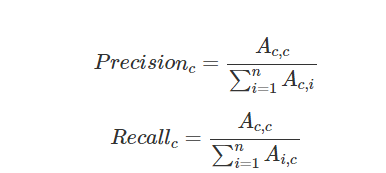


Рисунок 5 – Пример матрицы неточностей

Точность равняется отношению соответствующего диагонального элемента матрицы и суммы всей строки класса. Полнота – отношению диагонального элемента матрицы и суммы всего столбца класса. Формально:

 , (5)

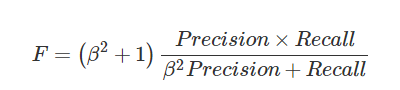
 (6)

Результирующая точность классификатора рассчитывается как арифметическое среднее его точности по всем классам. То же самое с полнотой. Технически этот подход называется macro-averaging.

Понятно, что чем выше точность и полнота, тем лучше. Но в реальной жизни максимальная точность и полнота не достижимы одновременно и приходится искать некий баланс. Поэтому, хотелось бы иметь некую метрику, которая объединяла бы в себе информацию о точности и полноте нашего алгоритма. В этом случае нам будет проще принимать решение о том, какую реализацию следует принять к использованию. Именно такой метрикой является F-мера.

F-мера представляет собой гармоническое среднее между точностью и полнотой. Она стремится к нулю, если точность или полнота стремится к нулю.

Возможно рассчитать F-меру придав различный вес точности и полноте, если вы осознанно отдаете приоритет одной из этих метрик при разработке алгоритма.

 (7)

Здесь β принимает значения в диапазоне если вы хотите отдать приоритет точности, а при β>1 приоритет отдается полноте. При β=1 вы получаете сбалансированную F-меру (также ее называют F1).

F-мера является хорошим кандидатом на формальную метрику оценки качества классификатора. Она сводит к одному числу две других основополагающих метрики: точность и полноту. Имея в своем распоряжении подобный механизм оценки гораздо проще принять решение о том является ли алгоритм обучения более подходящим для данной задачи.

## 

**2 Разработка и обучение нейронной сети для классификации новостей по категориям**

## **2.1 Выбор программного обеспечения**

Python является общепринятым языком для многих сфер применения науки о данных (data science). В Python есть библиотеки для загрузки данных, визуализации, статистических вычислений, обработки естественного языка, обработки изображений и многого другого. Этот обширный набор инструментов предлагает специалистам по работе с данными (data scientists) большой набор инструментов общего и специального назначения. Машинное обучение и анализ данных – это в основном итерационные процессы, в которых данные задают ход анализа. Крайне важно для этих процессов иметь инструменты, которые позволяют оперативно и легко работать. В качестве языка программирования общего назначения Python позволяет создавать сложные графические пользовательские интерфейсы (GUI) и веб-сервисы, а также легко интегрироваться в уже существующие системы [4].

TensorFlow — фреймворк для глубокого машиного обучения, разработанный компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия. Применяется как для исследований, так и для разработки собственных продуктов Google. Основное API для работы с библиотекой реализовано для Python.

Keras — открытая высоко модульная библиотека нейронных сетей, написанная на Python и способная работать поверх TensorFlow или Theano. Позволяет на более высоком уровне работать с нейросетями. Нацелена на оперативную работу с сетями глубинного обучения. Эта библиотека содержит многочисленные реализации широко применяемых строительных блоков нейронных сетей, таких как слои, целевые и передаточные функции, оптимизаторы, и множество инструментов для упрощения работы с изображениями и текстом.

## **2.2. Исходные данные для обучения нейронной сети**

Для успешного обучения нейронной сети требуется очень большой объём данных. Мы будем использовать статьи с сайта Яндекс.Новости, поскольку это крупнейший в рунете агрегатор новостных сообщений. Каждый день этот сервис получает материалы от нескольких тысяч СМИ и более 6000 партнёров. В среднем они выпускают более 110 тысяч сообщений в будний день. Всё, что мы видим на страницах Яндекс.Новостей, — это результат работы алгоритмов, таких как классификация статьей по категориям, объединение их с помощью кластеризации в сюжеты и их ранжирование [5].

В нашем курсовом проекте мы повторим работу алгоритмов по классификации статей, разбив их по 23 категориям (спорт, политика, культура и т.д.). Для этого нам необходимо произвести парсинг этого сайта с сохранением в базу данных SQLite news.db таблицы articles.

Для парсинга сайта https://news.yandex.ru будем использовать COM-объект web-браузера "InternetExplorer". Функцию ie(URL) будем вызывать в цикле с задержкой в 7 секунд, чтобы обойти защиту от роботов. Параметром этой функции является адрес интернет-страницы. Результатом является текст в формате html, который мы разбираем и выделяем следующие блоки информации: id новости, её дата-время, ссылка на первоисточник, заголовок и текст статьи.

Экранная форма и результат работы модуля парсинг продемонстрированы на рисунке 5.

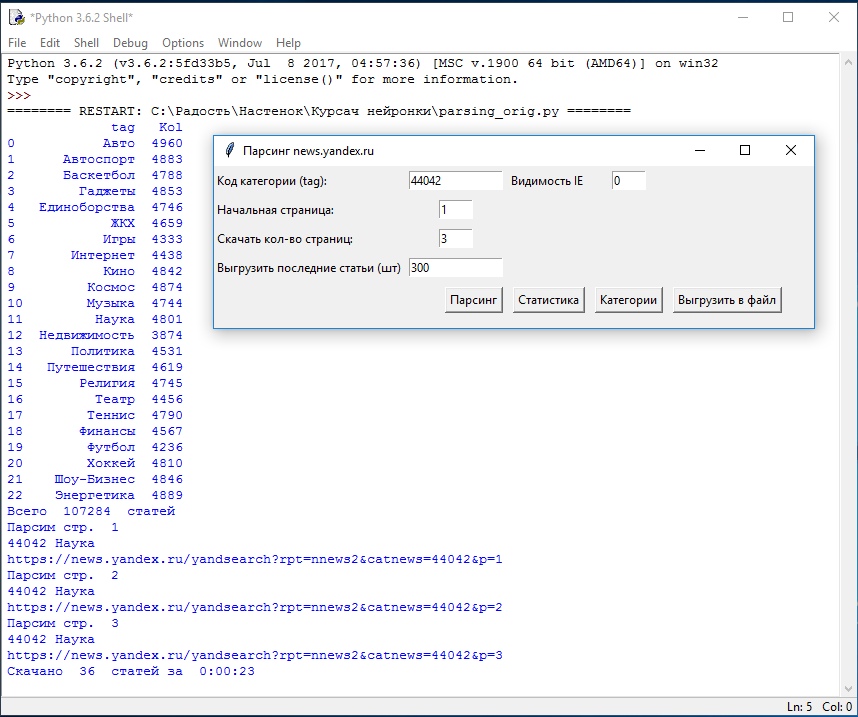


Рисунок 5 – Результат работы модуля парсинг

Обучение проводилось на 3600, 20070 и 107284 статей для выявления динамики точности от объема данных и составила 47%, 89% и 94% соответственно. Время обучения на разных объёмах данных составило 2 мин., 8 мин. и 2 ч. соответственно.

На этапе предобработки текста мы удаляем спецсимволы и знаки препинания, приводим все слова к нижнему регистру и преобразуем в нормальную форму при помощи библиотеки pymorphy2. В результате предобработки текста удалось повысить точность классификации на 1%.

Мы разделяем данные на 2 массива: train и test. 90% статей используем для обучения, а оставшиеся – для тестирования.

**2.3 Построение модели и подбор параметров для нейронной сети**

В данной работе в качестве нейронной сети был выбран многослойный персептрон MLP. Это простейший вид нейросети, но хорошо зарекомендовавший себя в решении задач классификации. Он имеет несколько отличительных признаков: каждый нейрон имеет нелинейную функцию активации (всюду дифференцируемую), сеть содержит несколько слоев скрытых нейронов и обладает высокой степенью связности.

Мы реализуем нейронную сеть MLP с двумя скрытыми слоями. Определим некоторые параметры нашей модели. Эти параметры часто называют гиперпараметрами, так как они должны быть уточнены еще до начала обучения:

* 1. количество обучающих образцов, обрабатываемых одновременно за одну итерацию алгоритма градиентного спуска, нашем случае batch\_size (512);
  2. количество итераций обучающего алгоритма по всему обучающему множеству, в нашем случае epochs (25);
  3. количество нейронов в каждом из двух скрытых слоев MLP, в нашем случае hidden\_size (512).

Мы будем использовать функцию активации LeakyReLU для нейронов первых двух слоев, и softmax для последнего слоя. Функция softmax разработана, чтобы превратить любой вектор с реальными значениями в вектор вероятностей. Данная функция предназначена для максимизации уверенности модели в правильном определении класса (в то время как функция квадратичной ошибки стремится к тому, чтобы вероятность попадания в остальные классы была как можно ближе к нулю). Приведём используемые параметры:

* 1. optimizer – используемый оптимизатор, в нашем случае Adam;
  2. metrics – метрики, по которым считается качество модели, в нашем случае – это точность (accuracy), доля верно угаданных ответов.

Также мы будем использовать 3 приема для тонкой настройки нашей нейронной сети.

Батч-нормализация - метод ускорения глубокого обучения. Он решает проблему, препятствующую эффективному обучению нейронных сетей: по мере распространения сигнала по сети, даже если мы нормализовали его на входе, пройдя через внутренние слои, он может сильно исказиться как по мат ожиданию, так и по дисперсии, что чревато серьезными несоответствиями между градиентами на различных уровнях. Батч-нормализация - предлагает весьма простое решение данной проблемы: нормализовать входные данные таким образом, чтобы получить нулевое мат ожидание и единичную дисперсию. Нормализация выполняется перед входом в каждый слой [6].

Одной из основных проблем машинного обучения является проблема переобучения, когда модель в погоне за минимизацией затрат на обучение теряет способность к обобщению. Существует простой способ держать переобучение под контролем - метод dropout. Есть и другие регуляризаторы, которые можно применить к нашей сети. Cамый популярный из них - L2-регуляризация (также называемая сокращением весов,), которая использует более прямой подход к регуляризации, чем dropout. Обычно первопричиной переобучения является сложность модели (в смысле количества ее параметров), слишком высокая для решаемой задачи и имеющегося обучающего множества. В некотором смысле, задача регуляризатора - понизить сложность модели, сохранив количество ее параметров. L2-регуляризация выполняется посредством наложения штрафов на веса с наибольшими значениями, минимизируя их L2-норму с использованием параметра λ - коэффициента регуляризации, который выражает предпочтение минимизации нормы относительно минимизации потерь на обучающем множестве.

В Keras есть такая важная особенность, как колбеки. Мы используем их в том случае, если сеть тренируется в течение очень долгого времени. Нам нужно понять, что пора остановиться, когда ошибка на dataset перестала уменьшаться. Мы задали максимальное количество эпох 25, и с помощью ранней остановки обучение ограничилось 7 эпохами, сократив время обучения до 2 часов на объеме данных 107 тыс. статей.

Модель нашей нейронной сети использует связку библиотек Keras и TensorFlow от Google.

**2.4. Экспериментальные результаты**

Классификация происходит при помощи нейронной сети MLP, содержащей два слоя. Модель создает единый словарь tokenizer (слово -> число), преобразует все описания заявок в числовые последовательности, заменяя слова на числа по словарю. По результатам экспериментов оптимальный размер словаря для наших данных составил 10 000 слов. Далее запускается обучение и тренировка нейронной сети и производится оценка результатов на тестовых данных. По окончании работы обученная модель сохраняется в файлы: «mlp\_model.h5», «tokenizer», «encoder.pickle».

В общем виде, результаты финального обучения продемонстрированы на следующем рисунке (см. рисунок 6):

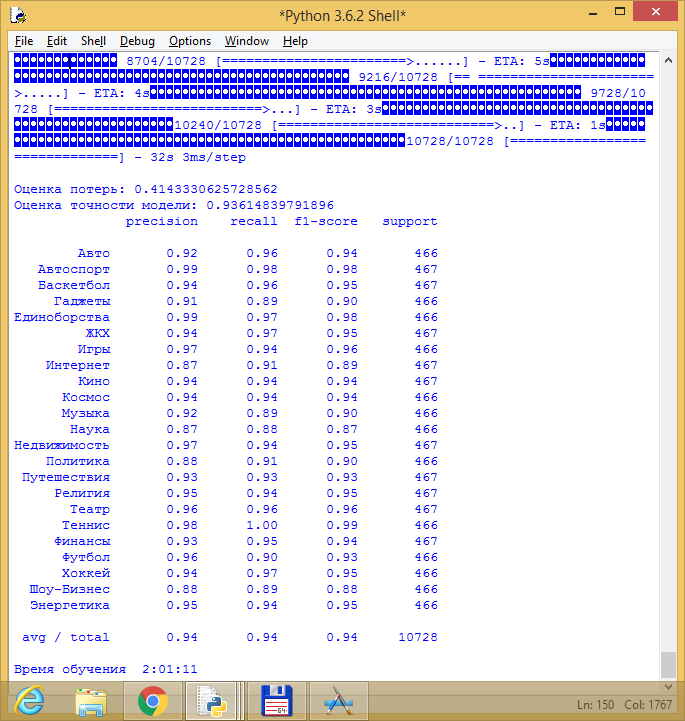


Рисунок 6 – Результаты финального обучения

Можно проследить эффективность обучения нейронной сети по графику точности (см. рисунок 7). По оси абсцисс откладываются эпохи, а по оси ординат – точность.

Чёрным показан результат обучения тренировочных данных, а красным - тестовых.

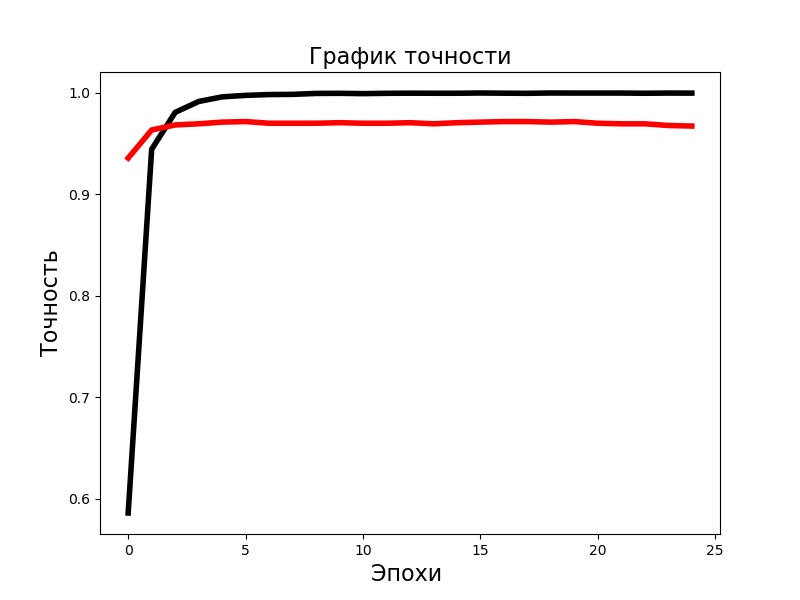


Рисунок 7 – График точности

**ЗАКЛЮЧЕНИЕ**

Данная курсовая работа направлена на демонстрацию современных возможностей нейросетей. Итогом работы стали следующие основные научные и практические результаты.

На основе бесплатных библиотек была построена двухслойная нейронная сеть MLP, которая выполняет функции по классификации новостей Яндекса. По признанию самого Яндекса проект yandex.news.ru был сложным, трудоёмким и дорогостоящим, поскольку его развитие начиналось с 2000 года и в настоящее время действует третья версия самообучаемого алгоритма-робота. Мы получили достаточно высокий результат 94% точности классификации, несмотря на ограниченный объём данных (107 тысяч записей) и небольшую вычислительную мощность.

Этого удалось достичь, поскольку программное обеспечение в последнее время развивается очень быстрыми темпами и то, что недавно требовало огромных вычислительных мощностей, сейчас общедоступно и позволяет реализовывать алгоритмы машинного обучения, используя гораздо меньше ресурсов.

По результатам курсовой работы все поставленные цели были достигнуты. Экспериментальное исследование, проведенное с использованием данной системы, подтвердило достоверность и эффективность результатов, полученных в работе. Потенциал системы высок, есть возможность развивать его до следующих уровней. В дальнейшем на её основе можно реализовать и протестировать алгоритмы машинного обучения без учителя (ранжирование, кластеризация, выделение ключевых признаков).

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 Wasserman Ph. Neural Computing: Theory and Practice / Ph. Wasserman / Coriolis Group – 1st ed. – 1989.

2 Вьюгин В. В. Математические основы машинного обучения и прогнозирования. – М:«МЦНМО» – 2013.

3 Саймон Хайкин. Нейронные сети: полный курс. — 2-е изд. — М.: «Вильямс» – 2006.

4 Коэльё Л. П., Ричерт В. Построение систем машинного обучения на языке Python. — М.: ДМК Пресс — 2015.

5 Как устроены Яндекс.Новости [Электронный ресурс]. – URL: https://yandex.ru/blog/company/76641 (дата обращения 1.12.2018).

6 Николенко С., Кадурин А., Архангельская Е.Глубокое обучение. — СПб.: Питер — 2018.

**ПРИЛОЖЕНИЕ А**

**Файл основной программы разработки нейронной сети**

from tkinter import \*

from tkinter import messagebox

import sqlite3

import win32com.client

from time import sleep

import datetime

from timeit import default\_timer as timer

import numpy as np

import pandas as pd

import sys

import re

import pymorphy2

ma = pymorphy2.MorphAnalyzer()

# - - - - - - - - - - - - - - - - - - - -

# Создание COM-объекта "InternetExplorer", переход по ссылке,

# возврат страницы в виде кода HTML для дальнейшего парсинга

def ie(URL):

ie = win32com.client.Dispatch("InternetExplorer.Application")

ie.Visible = field\_visible\_ie.get()

ie.Navigate(URL)

if ie.Busy:

sleep(7)

text = ie.Document.body.innerHTML

ie.Quit()

return text

# - - - - - - - - - - - - - - - - - - - -

# Подключение к таблице "Статьи"

def connect\_articles():

try:

conn = sqlite3.connect('data/news.db')

curs = conn.cursor()

CreateCommand = """

CREATE TABLE IF NOT EXISTS articles (

'id' NVARCHAR PRIMARY KEY,

'tag' NVARCHAR(30),

'txt' NVARCHAR(350),

'clean\_txt' NVARCHAR(350),

'date' TEXT,

'source' NVARCHAR(50),

'url' NVARCHAR(350)

);"""

curs.execute(CreateCommand)

conn.commit()

curs.close()

conn.close()

return True

except:

print('Нет подключения к таблице "articles" базы "data/news.db" ',sys.exc\_info()[0])

return False

# - - - - - - - - - - - - - - - - - - - -

# Очистка таблицы "Статьи"

def del\_articles():

try:

conn = sqlite3.connect('data/news.db')

curs = conn.cursor()

curs.execute("DELETE FROM articles")

conn.commit()

curs.close()

conn.close()

except:

print('Нет подключения к таблице "articles" базы "data/news.db" ',sys.exc\_info()[0])

# - - - - - - - - - - - - - - - - - - - -

# Загрузка статей с файла 'data/articles.csv'

def load\_articles():

try:

#del\_articles()

conn = sqlite3.connect('data/news.db')

curs = conn.cursor()

InsertText = """

INSERT OR REPLACE INTO articles ('id', 'tag', 'txt', 'clean\_txt', 'date', 'source', 'url')

VALUES ("{id}", "{tag}" , "{txt}", "{clean\_txt}", "{date}", "{source}", "{url}");"""

df = pd.read\_csv('data/articles.csv',sep=';',encoding='ANSI')

for index, item in df.iterrows():

txt = item.txt.replace('"', "")

date = item.date

date = date[6:10]+'-'+date[3:5]+'-'+date[0:2]+date[10:]

InsertCommand = InsertText.format(

id=str(item.id),

tag=item.tag,

txt=txt,

clean\_txt=item.clean\_txt,

date=date,

source=item.source,

url=item.url)

curs.execute(InsertCommand)

conn.commit()

curs.close()

conn.close()

except:

print('Нет подключения к таблице "articles" базы "data/news.db" ',sys.exc\_info()[0])

# - - - - - - - - - - - - - - - - - - - -

# Загрузка категорий с файла 'data/tags.csv'

def load\_tags():

try:

conn = sqlite3.connect('data/news.db')

curs = conn.cursor()

CreateCommand = """

CREATE TABLE IF NOT EXISTS tags (

'id' INTEGER PRIMARY KEY,

'name' NVARCHAR(30)

);"""

curs.execute(CreateCommand)

curs.execute("DELETE FROM tags")

InsertText = """

INSERT INTO tags ('id', 'name')

VALUES ("{id}" , "{name}");"""

df = pd.read\_csv('data/tags.csv',sep=';',encoding='ANSI')

for index, item in df.iterrows():

InsertCommand = InsertText.format( id=item["id"], name=item["name"])

curs.execute(InsertCommand)

conn.commit()

curs.close()

conn.close()

except:

print('Нет подключения к таблице "tags" базы "data/news.db" ',sys.exc\_info()[0])

# - - - - - - - - - - - - - - - - - - - -

# Статистика по таблице "Статьи"

def statistic():

try:

conn = sqlite3.connect('data/news.db')

curs = conn.cursor()

query = 'SELECT tag, count(id) AS Kol FROM articles GROUP BY tag ORDER BY tag;'

df = pd.read\_sql\_query(query,conn)

print(df)

print('Всего ',df['Kol'].sum() ,' статей')

df.to\_csv('data/statistic.csv',sep=';',encoding='ANSI')

curs.close()

conn.close()

except:

print('Нет подключения к таблице "articles" базы "data/news.db" ',sys.exc\_info()[0])

# - - - - - - - - - - - - - - - - - - - -

# Категории и коды

def id\_tag():

try:

conn = sqlite3.connect('data/news.db')

curs = conn.cursor()

query = 'SELECT id,name as tag FROM tags ORDER BY tag;'

df = pd.read\_sql\_query(query,conn)

print(df)

curs.close()

conn.close()

except:

print('Нет подключения к таблице "articles" базы "data/news.db" ',sys.exc\_info()[0])

# - - - - - - - - - - - - - - - - - - - -

# Выгрузка "Статей" для тестов

def unload():

try:

conn = sqlite3.connect('data/news.db')

curs = conn.cursor()

query = 'SELECT \* FROM articles ORDER BY date DESC limit '+str(field\_row.get())+' ;'

df = pd.read\_sql\_query(query,conn)

#print(df)

df.to\_csv('data/test.csv',sep=';',encoding='ANSI')

print('Выгружено ',df['id'].count() ,' записей в файл "data/test.csv"')

curs.close()

conn.close()

except:

print('Нет подключения к таблице "articles" базы "data/news.db" ',sys.exc\_info()[0])

# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# очистка текста

def text\_cleaner(text):

text = text.lower()

# замена даты и времени

text = re.sub( r'(\d+[/\.]\d+[/\.]\d+)|(\d+ ?гг?)|(\d+:\d+(:\d+)?)', ' ', text)

# удаление знаков

text=re.sub(r'''[!"#$%&’“”'()\*+,./:;<=>?@[\]^`{|}~]''', ' ', text)

text=re.sub(r'\-+', '\_', text)

text=re.sub(r'( \\_+)|(\_+ )', ' ', text)

# приведение к нормальной форме

text = " ".join(ma.parse(word)[0].normal\_form for word in text.split())

return text

# - - - - - - - - - - - - - - - - - - - -

# Парсинг сайта news.yandex.ru и загрузка их в таблицу "articles"

def yandex\_news():

if connect\_articles() :

t = timer()

conn = sqlite3.connect('data/news.db')

curs = conn.cursor()

kol\_page = field\_kol\_page.get()

id\_tag = field\_tag.get()

start\_page = field\_start\_page.get()

if id\_tag == '' : usl = ''

else : usl = ' WHERE id == '+id\_tag

query = 'SELECT id,name as tag FROM tags'+usl+' ORDER BY tag;'

df = pd.read\_sql\_query(query,conn)

kol = 0

for i in range(kol\_page):

page = start\_page+i

print("Парсим стр. ",page)

for index, item in df.iterrows():

URL = 'https://news.yandex.ru/yandsearch?rpt=nnews2&catnews='+str(item.id)+'&p='+str(page)

tag = item.tag

print(item.id,tag)

print(URL)

text = ie(URL)

if text.find('Новостей по вашему запросу не найдено')>=0 or text.find('Ошибка 404')>=0 :

print("Нет данных ...")

else :

text = text.replace("<div", "\n<div")

spis = text.split("\n")

InsertText = """

INSERT OR REPLACE INTO articles ('id', 'tag', 'txt', 'clean\_txt', 'date', 'source', 'url')

VALUES ("{id}", "{tag}" , "{txt}", "{clean\_txt}", "{date}", "{source}", "{url}");"""

for line in spis:

if line.find('document i-bem')>=0 :

id = line.partition('id="')[2].partition('"')[0]

elif line.find('document\_\_provider-name">')>=0 :

source = line.partition(">")[2].partition("</div>")[0]

elif line.find('document\_\_title')>=0 :

url = line.partition('href="')[2].partition('"')[0]

date = int(line.partition('-date=')[2].partition(',')[0])

date = datetime.datetime.utcfromtimestamp(date+10800)

txt = line.partition("]}'>")[2].partition('</')[0]

elif line.find('document\_\_snippet')>=0 :

txt = txt + " "+line.partition(">")[2].partition('</')[0]

txt = txt.replace('"', "")

clean\_txt = text\_cleaner(txt)

source = source.replace('"', "")

InsertCommand = InsertText.format(

id=str(id),

tag=tag,

txt=txt,

clean\_txt=clean\_txt,

date=date,

source=source,

url=url)

#print(InsertCommand)

curs.execute(InsertCommand)

kol+=1

conn.commit()

curs.close()

conn.close()

t = int(timer() - t)

t = str(datetime.timedelta(seconds=t))

print('Скачано ',kol,' статей за ',t)

# - - - - - - - - - - - - - - - - - - - -

root = Tk()

root.title("Парсинг news.yandex.ru")

root.geometry("600x400")

field\_tag = StringVar()

field\_start\_page = IntVar()

field\_kol\_page = IntVar()

field\_row = IntVar()

field\_visible\_ie = IntVar()

field\_kol\_page.set(1)

field\_start\_page.set(1)

field\_row.set(300)

tag\_label = Label(text="Код категории (tag):")

start\_page\_label = Label(text="Начальная страница:")

kol\_page\_label = Label(text="Скачать кол-во страниц:")

row\_label = Label(text="Выгрузить последние статьи (шт)")

visible\_ie\_label = Label(text="Видимость IE ")

tag\_label.grid(row=0, column=0, sticky="w")

start\_page\_label.grid(row=1, column=0, sticky="w")

kol\_page\_label.grid(row=2, column=0, sticky="w")

row\_label.grid(row=3, column=0, sticky="w")

visible\_ie\_label.grid(row=0, column=2, sticky="w")

tag\_entry = Entry(textvariable=field\_tag, width=15)

start\_page\_entry = Entry(textvariable=field\_start\_page, width=5)

kol\_page\_entry = Entry(textvariable=field\_kol\_page, width=5)

row\_entry = Entry(textvariable=field\_row, width=15)

visible\_ie\_entry = Entry(textvariable=field\_visible\_ie, width=5)

tag\_entry.grid(row=0,column=1, padx=5, pady=5)

start\_page\_entry.grid(row=1,column=1, padx=5, pady=5)

kol\_page\_entry.grid(row=2,column=1, padx=5, pady=5)

row\_entry.grid(row=3,column=1, padx=5, pady=5)

visible\_ie\_entry.grid(row=0,column=3, padx=5, pady=5)

parsing\_button = Button(text="Парсинг", command=yandex\_news)

parsing\_button.grid(row=4,column=1, padx=5, pady=5, sticky="e")

statistic\_button = Button(text="Статистика", command=statistic)

statistic\_button.grid(row=4,column=2, padx=5, pady=5, sticky="e")

statistic\_button = Button(text="Категории", command=id\_tag)

statistic\_button.grid(row=4,column=3, padx=5, pady=5, sticky="e")

statistic\_button = Button(text="Выгрузить в файл", command=unload)

statistic\_button.grid(row=4,column=4, padx=5, pady=5, sticky="e")

root.mainloop()

**ПРИЛОЖЕНИЕ Б**

**Файл программы парсинга сайта news.yandex.ru**

import sys

import numpy as np

import pandas as pd

import sqlite3

import pickle

import matplotlib.pyplot as plt

import datetime

from timeit import default\_timer as timer

from sklearn.metrics import classification\_report

# neuro

import keras

from keras.models import Sequential, Model

from keras.layers import Dense, Dropout, Activation, Input

from keras.preprocessing.text import Tokenizer

from sklearn.preprocessing import LabelEncoder

from keras.layers.normalization import BatchNormalization

from keras.callbacks import EarlyStopping

from keras.layers.advanced\_activations import LeakyReLU

from keras.regularizers import l2

from sklearn.metrics import confusion\_matrix

from itertools import \*

# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# загрузка данных

def load\_data():

try:

conn = sqlite3.connect('data/news.db')

query = 'SELECT \* FROM articles;'

df = pd.read\_sql\_query(query,conn)

conn.close()

except:

df = { 'id':[], 'txt':[], 'tag':[] }

print("Ошибка: ", sys.exc\_info()[0])

return df

# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# разделение набора текстов на тестовый и учебный наборы

def load\_data\_from\_arrays(strings, labels, train\_test\_split=0.9):

data\_size = len(strings)

test\_size = int(data\_size - round(data\_size \* train\_test\_split))

x\_train = strings[test\_size:]

y\_train = labels[test\_size:]

x\_test = strings[:test\_size]

y\_test = labels[:test\_size]

return x\_train, y\_train, x\_test, y\_test

# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

#матрица неточностей

def plot\_confusion\_matrix(cm, classes,

title='Confusion matrix',

cmap=plt.cm.Blues, normalize=True):

cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title, fontsize=30)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=45, fontsize=22)

plt.yticks(tick\_marks, classes, fontsize=22)

fmt = '.2f'

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, format(cm[i, j], fmt),

horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.ylabel('Правильная категория', fontsize=25)

plt.xlabel('Определенная моделью категория', fontsize=25)

y\_softmax = model.predict(X\_test)

y\_test\_1d = []

y\_pred\_1d = []

for i in range(len(y\_test)):

probs = y\_test[i]

index\_arr = np.nonzero(probs)

one\_hot\_index = index\_arr[0].item(0)

y\_test\_1d.append(one\_hot\_index)

for i in range(0, len(y\_softmax)):

probs = y\_softmax[i]

predicted\_index = np.argmax(probs)

y\_pred\_1d.append(predicted\_index)

text\_labels = encoder.classes\_

cnf\_matrix = confusion\_matrix(y\_test\_1d, y\_pred\_1d)

plt.figure(figsize=(48,40))

plot\_confusion\_matrix(cnf\_matrix, classes=text\_labels, title="Confusion matrix")

plt.show()'''

# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

df = load\_data()

#df = pd.read\_csv('data/articles1.csv',sep=';',encoding='ANSI')

#df = df.sample(frac=1).reset\_index(drop=True)

x\_train, y\_train, x\_test, y\_test = load\_data\_from\_arrays(df['clean\_txt'],df['tag'])

print('\nРазмерность x\_train :', len(x\_train))

print('Размерность x\_test :', len(x\_test))

t = timer()

print('\nОбучаем нейронную сеть ...')

descriptions = df['clean\_txt']

categories = df['tag']

# количество эпох\итераций для обучения

epochs = 25

batch\_size = 512

# создаем единый словарь (слово -> число) для преобразования

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts(descriptions)

# Преобразуем все описания в числовые последовательности, заменяя слова на числа по словарю.

textSequences = tokenizer.texts\_to\_sequences(descriptions)

max\_words = len(tokenizer.word\_index)

max\_words = min(10000,max\_words)

print('В словаре {} слов'.format(max\_words))

tokenizer = Tokenizer(num\_words=max\_words)

tokenizer.fit\_on\_texts(descriptions)

x\_train\_ = tokenizer.texts\_to\_matrix(x\_train, mode='binary')

x\_test\_ = tokenizer.texts\_to\_matrix(x\_test, mode='binary')

print('Размерность x\_train :', x\_train\_.shape)

print('Размерность x\_test :', x\_test\_.shape)

# Записываем словарь в файл

with open('model/tokenizer.pickle', 'wb') as handle:

pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST\_PROTOCOL)

# encoder

encoder = LabelEncoder()

encoder.fit(y\_train)

y\_train1 = encoder.transform(y\_train)

y\_test1 = encoder.transform(y\_test)

num\_classes = np.max(y\_train1) + 1

print('\nКол-во категорий :',num\_classes)

# Записываем encoder в файл

with open('model/encoder.pickle', 'wb') as handle:

pickle.dump(encoder, handle, protocol=pickle.HIGHEST\_PROTOCOL)

# Преобразуем категории в матрицу двоичных чисел для использования categorical\_crossentropy

y\_train\_ = keras.utils.to\_categorical(y\_train1, num\_classes)

y\_test\_ = keras.utils.to\_categorical(y\_test1, num\_classes)

print('Размерность y\_train :', y\_train\_.shape)

print('Размерность y\_test :', y\_test\_.shape)

print('\nСтроим модель ...')

reg\_alpha = 0.0001

model = Sequential()

model.add(Dense(512, input\_shape=(max\_words,), W\_regularizer=l2(reg\_alpha)))

model.add(BatchNormalization())

model.add(LeakyReLU(alpha=.001))

model.add(Dropout(0.5))

model.add(Dense(256, W\_regularizer=l2(reg\_alpha)))

model.add(BatchNormalization())

model.add(LeakyReLU(alpha=0.001))

model.add(Dropout(0.5))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(x\_train\_, y\_train\_,

batch\_size=batch\_size,

epochs=epochs,

verbose=1,

callbacks=[EarlyStopping(monitor='val\_loss', patience=5)],

validation\_split=0.1)

score = model.evaluate(x\_test\_, y\_test\_, batch\_size=batch\_size, verbose=1)

print('\nОценка потерь: {}'.format(score[0]))

print('Оценка точности модели: {}'.format(score[1]))

#Plot the Loss Curves

plt.figure(figsize=[8,6])

plt.plot(history.history['loss'],'r',linewidth=3.0)

plt.plot(history.history['val\_loss'],'b',linewidth=3.0)

plt.legend(['Training loss', 'Validation Loss'],fontsize=18)

plt.xlabel('Epochs ',fontsize=16)

plt.ylabel('Loss',fontsize=16)

plt.title('Loss Curves',fontsize=16)

#Plot the Accuracy Curves

plt.figure(figsize=[8,6])

plt.plot(history.history['acc'],'r',linewidth=3.0)

plt.plot(history.history['val\_acc'],'b',linewidth=3.0)

plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=18)

plt.xlabel('Epochs ',fontsize=16)

plt.ylabel('Accuracy',fontsize=16)

plt.title('Accuracy Curves',fontsize=16)

# Записываем модель в файл

model.save('model/mlp\_model.h5')

# Результат на тестовых данные

text\_labels = encoder.classes\_

predicted=[]

for i in range(len(x\_test)):

prediction = model.predict(np.array([x\_test\_[i]]))

predicted\_label = text\_labels[np.argmax(prediction)]

predicted.append(predicted\_label)

#print(x\_test[i])

#print(y\_test[i])

#print(predicted\_label)

rep=classification\_report(y\_test, predicted)

print(rep)

t = int(timer() - t)

t = str(datetime.timedelta(seconds=t))

print('Время обучения ',t)

plt.show()