**Содержание**

[Введение 2](#_Toc86351512)

[1. Исследовательский раздел 4](#_Toc86351513)

[1.1. Понятие фоносемантики 4](#_Toc86351514)

[1.1.1. Назначение и принцип фоносемантики 4](#_Toc86351515)

[1.1.2. Метод семантического дифференциала 5](#_Toc86351516)

[1.2. Сравнение библиотек по машинному обучению 9](#_Toc86351517)

[1.2.1. PyBrain 10](#_Toc86351518)

[1.2.2. Tensorflow 11](#_Toc86351519)

[1.2.3. PyTorch 12](#_Toc86351520)

[1.2.4. Обоснование выбора библиотеки машинного обучения 13](#_Toc86351521)

[2. Конструкторский раздел 15](#_Toc86351522)

[1.3. Представление наборов данных 15](#_Toc86351523)

[1.4. Модуль формализации данных 17](#_Toc86351524)

[1.5. Модуль машинного обучения 19](#_Toc86351525)

[3. Технологический раздел 21](#_Toc86351526)

[1.6. Интерфейс программного обеспечения 21](#_Toc86351527)

[1.7. Тестирование программного обеспечения 23](#_Toc86351528)

Введение

Фоносемантика – это теория содержательности звуковой формы в языке. Фоносемантика находит широкие области применения в маркетинге, рекламе, переводах и т.д.

Для того чтобы лучше обучить машину языку человека, нужно помочь ей в освоении всего сложного спектра семантики. И фоносемантика здесь выступает в роли подходящего инструмента. Хотя это и далеко не главный семантический аспект, все же он тоже играет в семантической системе языка важную роль, и без его учета компьютер может упустить какие-то семантические тонкости, которые могут иногда оказать существенное влияние на формирование общей семантики слова, высказывания и даже целого текста.

Чисто техническая выгода еще и в том, что для обучения компьютера фоносемантике не требуется особых трудов. Все что необходимо это две таблицы да несколько формул. Освоить такую информацию компьютеру ничего не стоит, зато каких результатов можно добиться столь малыми усилиями.

Машина, наделенная такой фоносемантической интуицией, могла бы оказывать помощь человеку везде, где его работа связана с задачами усилить языковое воздействие на читателя, слушателя [1].

Конкретно в данной курсовой работе фоносемантика выступает в роли инструмента с помощью которого предоставляется возможность формализации входных данных для нейронной сети, которая в последствии должна определять объект (класс) по набору фоносемантических признаков.

**Цель работы** – реализовать и настроить программный модуль, позволяющий определять объект по фоносемантическим признакам его названия.

**Задачи курсовой работы:**

* изучить назначение и принцип фоносемантики;
* изучить существующие библиотеки по машинному обучению;
* реализовать программный модуль по формализации данных для их дальнейшего машинного обучения с использованием принципов фоносемантики;
* реализовать программный модуль, позволяющий обучать нейронную сеть, позволяющая определять объект на основе формализованных данных;
* провести тестирование разработанных программных модулей.

1. Исследовательский раздел
   1. Понятие фоносемантики

Фоносемантика - это раздел лингвистики, который изучает мотивированную безусловную связь между двумя сторонами языкового знака. В данной курсовой работе фоносемантика является инструментом для формализации данных передаваемых на вход нейронной сети.

* + 1. Назначение и принцип фоносемантики

Фоносемантика в переводе означает «смысл звука». Каждый звук может вызывать у человека определенную ассоциативную эмоционально-психологическую реакцию, а набор звуков (слово) человек воспринимает по его смысловому символическому значению, что-либо обозначающему.

Например, звук «Р» – говорит о чем-то громком, ярком, горячем, быстром, а звук «М» – о милом, мягком, о матери, о море.

К концу ХХ века профессором Ленинградского государственного университета С.В.Ворониным было обосновано новое направление в науке – фоносемантика, в рамках которой изучается звукоизобразительная лексика, подразделяющаяся на звукоподражательную и звукосимволическую.

Фоносемантику Воронин определяет как новую научную дисциплину языковедческого цикла, возникшую на стыке фонетики (по плану выражения), семантики (по плану содержания) и лексикологии (по совокупности этих планов).

Фоносемантика, как и всякая другая наука, имеет свои проблемы, актуальные на сегодняшний день. При изучении содержательной стороны звука возникают вопросы, ответы на которые вызывают споры у многих ученых. Имеет ли отдельно взятый звук или группа звуков свой особый смысл? Какие ассоциации возникают у человека, когда он слышит тот или иной звук? Почему эти ассоциации возникают?

Один из первых исследователей этой проблемы, И.Н. Горелов, провел такой эксперимент. Он опубликовал несколько картинок в одной из газет с просьбой угадать, кто на какой картинке изображен. На картинках были изображены фантастические существа: круглое, угловатое и др. Названия для этих существ ученый придумал такие: «мамлына» и «жаваруга». Большинство людей решили, что добродушная, толстенькая – это, «мамлына», а колючая и злая – «жаваруга». Отсюда можно сделать вывод, что «М», «Л», «Н», вызывают у нас представление о чем-то круглом, мягком, приятном, а звуки «Ж», «Р», «Г» ассоциируются с чем-то неприятным [1].

* + 1. Метод семантического дифференциала

В 1952 году, американский психолог Чарльз Осгуд, анализируя публичные выступления политиков, он заметил: из двух примерно одинаковых кандидатов (и их столь же идентичных программ) выигрывает тот, кто использует более благозвучную мелодику речи.

Для изучения эмоционального отношения людей к тем или иным понятиям для определения их смысла он разработал метод семантического дифференциала. Методика была описана в работе «The Measurement of Meaning» (1957).

Метод семантического дифференциала является комбинацией метода контролируемых ассоциаций и процедур шкалирования. Метод позволяет строить субъективные семантические пространства, на основании которых можно судить об эмоциональном отношении личности к различным объектам, о ее социальных установках, личностных смыслах, ценностных ориентациях, самооценке.

Для начала объекты (понятия, персонажи, символы) оцениваются по ряду двуполюсных шкал. Для каждого признака на полюсах шкалы присваиваются противоположные выражения(предельно положительным и предельно отрицательным). Например: плохой – хороший, маленький – большой, красивый – безобразный, яркий – тусклый, приятный – неприятный и т. д.

Полюса могут задаваться и невербально через графические оппозиции, фотографии, художественные изображения и даже физические предметы.

Центр шкалы соответствует нулевой выраженности признака как в положительном, так и в отрицательном направлениях. Обе ветви шкалы имеют несколько делений. Обычно применяют 3-, 5- или 7-балльные шкалы.

Оценивая объект-стимул, испытуемый демонстрирует, во-первых, свое понимание этого объекта, во-вторых, степень отношения к данному объекту. Считается, что чем значимее для испытуемого объект, тем он пристрастнее и тем выше (по абсолютной величине) его оценка.

Чем больше использовано шкал, тем многограннее субъективное суждение об объекте. Для наглядности эти шкалы размещают друг под другом, а оценки испытуемого на каждой из них объединяются прямыми отрезками, что в совокупности обычно дает некоторую ломаную линию, именуемую «семантическим профилем».

Каждая шкала может рассматриваться как ось субъективного семантического пространства, а выданные испытуемым оценки – как точки на нем. Положение каждой точки определяется двумя показателями: качественным – направленностью (положительная или отрицательная) и количественным – интенсивностью (абсолютная величина балла).

Сравнение отдельных оценок или семантических профилей, полученных у одного испытуемого, но по разным объектам, дает основание к установлению семантической близости или удаленности этих объектов в семантическом пространстве субъекта. В итоге получаем представление о субъективном восприятии подобия или различия между объектами – понятиями. Степень такого подобия или различия можно выразить и количественно, исходя из модели семантического пространства, с помощью формулы (1):

|  |  |
| --- | --- |
|  | (1) |

где х, у – сравниваемые объекты;

Д – семантическое расстояние между объектами;

d – разность между оценками объектов по каждому признаку.

Таким образом, можно сравнивать между собой данные разных субъектов, но по одному и тому же объекту или прослеживать динамику отношений субъекта к одному и тому же объекту в зависимости от влияния различных факторов (например, времени, социальных условий, здоровья и проч.). При этом в роли субъекта могут выступать как отдельные лица, так и группы людей [1].

Метод семантического шкалирования был применен А.П. Журавлевым. Результатом его исследования оказалась доминирующая экспрессивность звуков по трем основным измерениям: «оценки», «мягкости» и «активности».

При проведении исследований А.П. Журавлевым, была выбрана шкала, имеющая два полюсных значения: «очень хорошее» и «очень плохое». Среднее положение шкалы – качественное понятие «никакое». Промежуточные градации понятий: «хорошее» и «плохое».

Качественным понятиям соответствуют числовые значения «очень хорошее» – «1», «хорошее» – «2», «никакое» – «3», «плохое» – «4», «очень плохое» – «5».

Всего было взято 25 признаковых шкал: «хороший – плохой», «красивый – отталкивающий», «радостный – печальный», «светлый – темный», «легкий – тяжелый», «безопасный – страшный», «добрый – злой», «простой – сложный», «гладкий – шероховатый», «округлый – угловатый», «большой – маленький», «грубый – нежный», «мужественный – женственный», «сильный – слабый», «холодный – горячий», «величественный – низменный», «громкий – тихий», «могучий – хилый», «веселый – грустный», «яркий – тусклый», «подвижный – медлительный», «быстрый – медленный», «активный – пассивный».

Эксперимент по измерению фонетической значимости А.П. Журавлев начал с введения понятия звукобуква. Звукобуква – знак, учитывающий одновременно фонетическое и графическое восприятие звука.

Отобранные для эксперимента звукобуквы ученый расположил в случайной последовательности, и был получен ряд из 46 звукобукв. С этими звукобуквами проводился эксперимент, в котором приняли участие более тысячи респондентов, которые проставили оценку каждой из 46 звукобукв.

В итоге сформировалась таблица, содержащая средние оценки всех русских звукобукв по признаковым шкалам.

Таким образом, учитывая таблицы частойстей звукобукв и их значимостей, любое слово можно представить в виде вектора из 25 элементов, представляющих числовое значение по определенной признаковой шкале [2].

 Если частотность (вероятность) любого (i-того) звука слова обозначить как , а максимальную частотность звука в данном слове как , то коэффициент, учитывающий разницу частотностей звуков слова  , можно вычислить как отношение, представленное в формуле (2):

|  |  |
| --- | --- |
|  | (2) |

Теперь нужно учесть место каждого звука в слове. Для этого коэффициент первого звука слова () увеличивается в четыре раза, так как это показано в формуле (3):

|  |  |
| --- | --- |
|  | (3) |

Для ударного () — в два раза, формула (4):

|  |  |
| --- | --- |
|  | (4) |

Таким образом основная формула (5) будет выглядеть следующим образом:

|  |  |
| --- | --- |
|  | (5) |

где F — фонетическая содержательность слова (его фоносемантика);

— фонетическая содержательность очередного (i-того) звука слова;

 — коэффициент для очередного (i-того) звука слова.

Последняя «примерка на манекенах» показывает, что все в порядке — схема расчета в общем верна. Информанты считают, что «слово» незич звучит как нечто «маленькое» и «нежное», а фрыш — как нечто «плохое, грубое, страшное», и компьютер дает примерно те же характеристики. По мнению информантов, хифель и уршух страшное, а лимень и нитис — безопасное; компьютер того же мнения. Вробар и вакам кажутся информантам сильными, и компьютер выдал для них тот же признак.

Значит, способ расчета можно переносить и на настоящие слова. Конечно, спасительная оглядка на информантов теперь невозможна, но компьютер уже научился правильно имитировать человеческое восприятие фоносе-мантического ореола слов. Использовать эти свои умения он может разнообразно, и некоторые из возможностей мы ниже обсудим.

В результате вычислений слово по каждой шкале получает суммарную оценку фоносемантики, выраженную в единицах пягиранговой измерительной шкалы, то есть такую же оценку, как и средняя оценка содержательности отдельного звука [3].

* 1. Сравнение библиотек по машинному обучению

Перед непосредственной разработкой программного обеспечения необходимо проанализировать и выбрать библиотеки позволяющие провести машинное обучение с использованием языка программирования Python.

* + 1. PyBrain

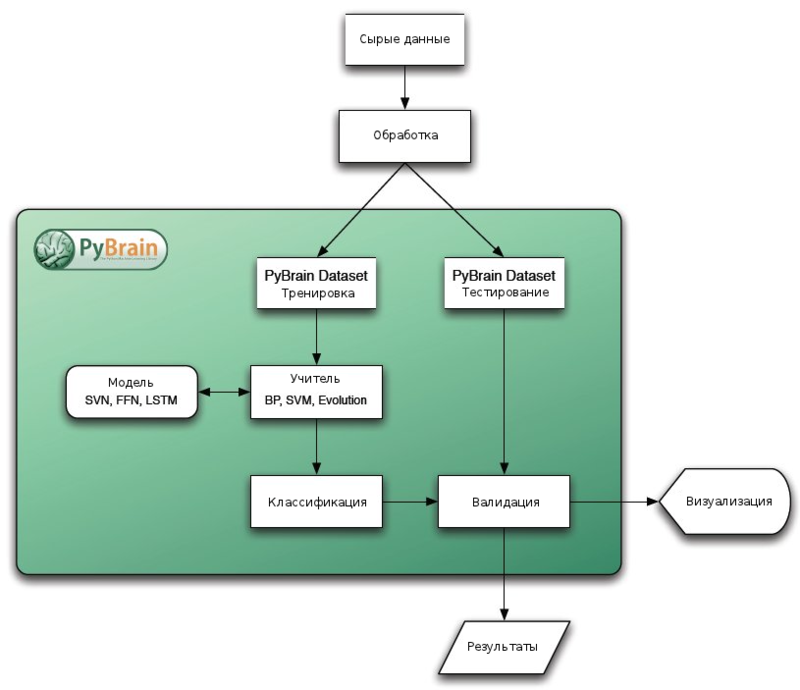
PyBrain — одна из лучших Python библиотек для изучения и реализации большого количества разнообразных алгоритмов связанных с нейронными сетями. Являет собой удачный пример совмещения компактного синтаксиса Python с хорошей реализацией большого набора различных алгоритмов из области машинного интеллекта.

Среди групп людей, которым могут быть полезны возможности PyBrain можно отнести:

* исследователей — предоставляет единообразную среду для реализации различных алгоритмов, избавляя от потребности в использовании десятков различных библиотек. Позволяет сосредоточится на самом алгоритме а не особенностях его реализации;
* студентов — с использованием PyBrain удобно реализовать домашнее задание, курсовой проект или вычисления в дипломной работе. Гибкость архитектуры позволяет удобно реализовывать разнообразные сложные методы, структуры и топологии;
* лекторов — обучение методам Machine Learning было одной из основных целей при создании библиотеки. Авторы будут рады, если результаты их труда помогут в подготовке грамотных студентов и специалистов;
* разработчиков — проект Open Source, поэтому новым разработчикам всегда рады.

PyBrian представляет собой модульную библиотеку предназначенную для реализации различных алгоритмов машинного обучения на языке Python. Основной его целью является предоставление исследователю гибких, простых в использовании, но в то же время мощных инструментов для реализации задач из области машинного обучения, тестирования и сравнения эффективности различных алгоритмов.

Библиотека построена по модульному принципу, что позволяет использовать её как студентам для обучения основам, так и исследователям, нуждающимся в реализации более сложных алгоритмов. Общая структура процедуры её использования приведена на следующей схеме:



Дополнительно присутствуют программные инструменты, позволяющие реализовывать сопутствующие задачи:

* построение / визуализация графиков;
* поддержка netCDF;
* запись/чтение XML [4].
  + 1. Tensorflow

Созданная Google и написанная на Python и C++, TensorFlow является одной из лучших открытых библиотек для численных вычислений. Она просто обязана быть хорошей, поскольку даже такие гиганты как DeepMind, Uber, AirBnB или Dropbox выбрали этот фреймворк для своих нужд.

TensorFlow хороша для сложных проектов, таких как создание многослойных нейронных сетей. Она используется для распознавания голоса или картинок и приложений для работы с текстом, таких как Google Translate, например.

Конечно же, эксперты назвали множество ее преимуществ:

* для нее написано большое количество руководств и документации;
* предлагает мощные средства мониторинга процесса обучения моделей и визуализации (Tensorboard);
* поддерживается большим сообществом разработчиков и техническими компаниями;
* обеспечивает обслуживание моделей;
* поддерживает распределенное обучение;
* TensorFlow Lite обеспечивает вывод на устройства с низкой задержкой для мобильных устройств.

Помимо плюсов можно выделить и недостатки:

* проигрывает по скорости работы в эталонных [тестах](https://arxiv.org/pdf/1608.07249v7.pdf), в сравнении с CNTK и MXNet;
* имеет более высокий входной порог для начинающих, чем PyTorch или Keras.
* «чистая» Tensorflow достаточно низкоуровневая и требует много шаблонного кода, и режим «определить и запустить» для Tensorflow значительно усложняет процесс дебага.
* единственный полностью поддерживаемый язык – Python [4].
  + 1. PyTorch

PyTorch является преемником Python для библиотеки Torch, написанной на Lua, и большим конкурентом TensorFlow. Он был разработан Facebook и использовался Twitter, Salesforce, Оксфордским Университетом и многими другими компаниями.

PyTorch используется в основном, чтобы обучать модели быстро и эффективно, поэтому [это выбор](https://www.netguru.com/blog/top-machine-learning-frameworks-%20compared) большого количества разработчиков.

У него есть множество важных преимуществ:

* благодаря архитектуре фреймворка, процесс создания модели достаточно прост и прозрачен;
* режим по умолчанию «define-by-run» – отсылка к традиционному программированию. Фреймворк поддерживает популярные инструменты для дебага, такие как pdb, ipdb или дебаггер PyCharm;
* поддерживает декларативный [параллелизм данных](https://towardsdatascience.com/pytorch-vs-tensorflow-spotting-the-%20difference-25c75777377b);
* имеет много предварительно обученных моделей и готовых модульных частей, которые легко комбинировать;
* распределенное обучение доступно с версии 0.4.

Но есть у этого фреймворка и несколько неоспоримых минусов:

* недостаточная поддержка моделей;
* недостает интерфейсов для мониторинга и визуализации, как TensorBoard – однако он имеет внешнее подключение к Tensorboard [4].
  + 1. Обоснование выбора библиотеки машинного обучения

Для разработки программного обеспечения по опеделению объекта по фоносемантическим признакам его названия было решено использовать библиотеку с открытым исходным кодом для машинного обучения PyBrain. Данная библиотека была выбрана в силу того, что она предоставляет удобную документацию, набор гибких, простых в использовании, но в то же время мощных инструментов для реализации задач из области машинного обучения, тестирования и сравнения эффективности различных алгоритмов. Библиотеки Tensorflow и PyTorch также бы подошли для решения поставленной задачи, но их огромные возможности были бы излишними в данном случае и поребовали бы больше времени на их изучение.

1. Конструкторский раздел

Перед непосредственной разработкой программного обеспечения определения объекта по фоносемантческим признакам его названия необходимо определить форму представления наборов данных подаваемых в нейронную сеть. Среди основных модулей для поставленной задачи были выделены: модуль формализации данных, т.е. представления исходного слова в векторной форме и их, а также модуль машинного обучения, представляющий обучаемую нейронную сеть, которая по итогу должна предсказывать принадлежность произвольного объекта (слова) к какому-либо классу.

* 1. Представление наборов данных

Для представления наборов данных был использован формат csv. Для решения задачи определения объекта по фоносемантическим признакам его названия было решено сформировать четыре набора данных:

1. значимость звукобукв (рисунок 1);

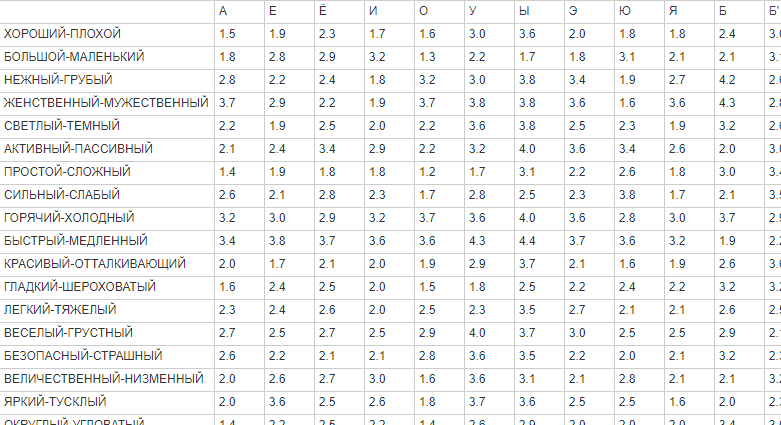


Рисунок 1 - Фрагмент набора данных значимости звукобукв

1. частость звукобукв (рисунок 2);



Рисунок 2 - Фрагмент набора данных частости звукобукв

1. слова и ударения в них (рисунок 3);



Рисунок 3 - Фрагмент набора данных слов и их ударений

1. слова и определяющие их классы (рисунок 4).

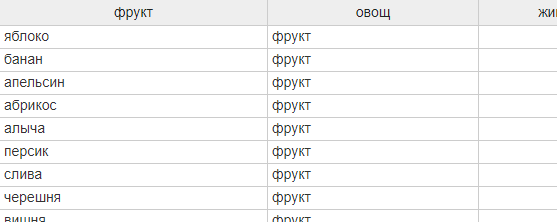


Рисунок 4 - Фрагмент набора данных слов и определяющих их классы

* 1. Модуль формализации данных

Для формализации данных был написан соответствующий модуль. При разработке данного модуля был использован теоретический материал, описанный в разделе 1.1 данного отчета.

Листинг класса, реализующего формализацию слова в векторную форму:

|  |
| --- |
| import sys  sys.path.insert(0, "e:/Files/University/Project/CourseWorkPython/sound\_convert")  from extractor import SoundLetterExtractor  from analyzer import PhonosemanticAnalyzer  from letter import SoundLetter  from word import SoundWord  from utils import read\_csv  from pprint import pprint  import re  class WordToVectorConvert:      """      Модуль для конвертирования слова в вектор.      """      def GetVector(self, analyzer, word, stress):          """          Метод преобразования слова в вектор          """          pass          sound\_extraxtor = SoundLetterExtractor()          sound\_word = sound\_extraxtor.extract(word, stress)          vector = analyzer.analyze\_sound\_word(sound\_word)          return (sound\_word.word, sound\_word.sound\_word, vector)        def ConvertWordsInVector(self, sound\_letters\_frequency\_file, sound\_letters\_significance\_file, classes\_words\_file, words\_stress\_file):          """          Метод преобразования слова в векторную форму для набора данных          :param sound\_letters\_frequency\_file: Таблица частотности звукобукв.          :param sound\_letters\_significance\_file: Таблица значимости звукобукв.          :param classes\_words\_file: Таблица слов и ударений в них.          :param classes\_words\_file: Таблица слов и определяющих их классы.          :return: кортеж (список всех классов, словарь (ключ: звуко-слово, значение: вектор))          """          analyzer = PhonosemanticAnalyzer(sound\_letters\_frequency\_file, sound\_letters\_significance\_file)          classes\_words\_f = read\_csv(classes\_words\_file) # Получение списка слов и их классов          stress\_words\_f = read\_csv(words\_stress\_file) # Получение списка слов и их ударений            words\_stress\_list = list()          for elem in stress\_words\_f:              word = elem[0]              stress = list(map(int, list(filter(None, elem[1:]))))              words\_stress\_list.append(self.GetVector(analyzer, word, stress))          # pprint(words\_stress\_list)          # Список кортежей (вектор слова, список векторов классов, к которым принадлежит слово)          words\_classes\_vector\_list = list()            # Список всех существующих классов          all\_classes\_list = list()          # Перебор списка слов и их классов          for i, elem in enumerate(classes\_words\_f):              # Получение списка всех классов              if i == 0:                  all\_classes\_list = elem                  continue              classes\_vector = [0] \* len(all\_classes\_list)              # Получение слова              word = elem[0]              # Получение классов которым принадлежит слово              classes = list(filter(None, elem[1:]))              for ii, item in enumerate(all\_classes\_list):                 if item in classes:                     classes\_vector[ii] = 1              word\_vector\_res = list()              # Перебор списка слов и их векторов              for item in words\_stress\_list:                  # Получение слова                  word\_str = item[0]                  # Получение вектора                  word\_vector = item[2]                  if (word\_str == word):                      word\_vector\_res = word\_vector              words\_classes\_vector\_list.append((word\_vector\_res, classes\_vector, word, classes))          return (all\_classes\_list, words\_classes\_vector\_list)      def ConvertWordInVector(self, sound\_letters\_frequency\_file, sound\_letters\_significance\_file, word, stress):          """          Метод преобразования слова в векторную форму.          :param sound\_letters\_frequency\_file: Таблица частотности звукобукв.          :param sound\_letters\_significance\_file: Таблица значимости звукобукв.          :param sound\_letters\_significance\_file: Таблица слов и ударений в них.          :return: словарь (ключ: звуко-слово, значение: вектор).          """          analyzer = PhonosemanticAnalyzer(sound\_letters\_frequency\_file, sound\_letters\_significance\_file)          word\_vector = self.GetVector(analyzer, word, stress)          return word\_vector |

* 1. Модуль машинного обучения

Для определения объекта по фоносемантическим признакам его названия было решено задействовать методы машинного обучения и соответствующую библиотеку PyBrain.

Листинг класса, реализующего машинное обучение:

|  |
| --- |
| import sys  sys.path.insert(0, "e:/Files/University/Project/CourseWorkPython/sound\_convert")  from pybrain3.tools.shortcuts import buildNetwork  from pybrain3.datasets import ClassificationDataSet, SupervisedDataSet  from pybrain3.structure import TanhLayer, SoftmaxLayer, LinearLayer  from pybrain3.supervised.trainers import BackpropTrainer  from vector import WordToVectorConvert  class NeuNet:      """      Модуль для обучения нейронной сети.      """      net = None      words\_classes\_vector\_list = list()        def training\_network(self, words\_classes\_vector\_list, number\_epoch):          """          Метод преобразования слова в векторную форму.          """          self.words\_classes\_vector\_list = words\_classes\_vector\_list          # Параметры нейронной сети          num\_input\_neuron = 25          num\_output\_neuron = len(words\_classes\_vector\_list[0])          num\_hidden\_neuron = int((num\_output\_neuron + num\_output\_neuron) \* 2/3)          \_bias = False          \_hiddenclass = SoftmaxLayer          \_nb\_classes = num\_output\_neuron          num\_epoch = number\_epoch            # Создание нейронной сети и датасетов          self.net = buildNetwork(num\_input\_neuron,num\_hidden\_neuron,num\_output\_neuron, bias = \_bias, hiddenclass = \_hiddenclass)          norgate = ClassificationDataSet(num\_input\_neuron, num\_output\_neuron, nb\_classes=\_nb\_classes)          nortrain = ClassificationDataSet(num\_input\_neuron, num\_output\_neuron, nb\_classes=\_nb\_classes)            # Заполнение нейронной сети          print("\n Данные для обучения: ")          for elem in words\_classes\_vector\_list[1]:              word = elem[0]              \_class = elem[1]                print(str(elem[2]) + "  " + str(elem[0]) + "  " + str(\_class))              norgate.addSample(tuple(word), tuple(\_class))              nortrain.addSample(tuple(word), tuple(\_class))            # Обучение нейронной сети          trainer = BackpropTrainer(self.net, norgate) # , verbose=True, batchlearning=False, momentum= 0.1, learningrate= 0.01, weightdecay= 0.01, lrdecay=1.0          trnerr,valerr = trainer.trainUntilConvergence(dataset=nortrain,maxEpochs=num\_epoch, verbose=False)          return (trnerr, valerr)      def testing\_network\_on\_dataset(self):          """          Метод тестирования нейронной сети на наборе данных          """          pass          res\_list = list()          for elem in self.words\_classes\_vector\_list[1]:              word\_vector = elem[0]              word = elem[2]              res\_test = self.net.activate(tuple(word\_vector))              res\_list.append((word, res\_test))          return (self.words\_classes\_vector\_list[0], res\_list)        def testing\_network\_on\_word(self, word, word\_vector):          """          Метод тестирования на базе одного слова          """          pass          res\_list = list()          res\_test = self.net.activate(tuple(word\_vector))          res\_list.append((word, res\_test))          return (self.words\_classes\_vector\_list[0], res\_list) |

Исходный код всех модулей программного обеспечения размещен на веб-сервисе GitHub: <https://github.com/AndreiBarsukov/PhonosemanticsPyBrain>

1. Технологический раздел

Результатом данной курсовой работы является разработанное программное обеспечение, позволяющие обучать нейронную сеть определять объект по фоносемантическим признакам его названия, предсказывать принадлежность объекта к определенному классу, а также редактировать исходные наборы данных необходимые для обучения нейронной сети.

* 1. Интерфейс программного обеспечения

Интерфейс разработанного программного обеспечения, представленного на рисунке 5, содержит набор необходимых элементов управления для редактирования исходных наборов данных, отображения ошибок обучения нейронной сети на графике, а также обучения и тестирования нейронной сети.

Кнопка «Удалить выделенный элемент» позволяет удалить выбранную пользователем строку из соотвествующей таблицы.

Кнопка «Добавить элемент» позволяет добавить в наборы данных новую информацию, которая вводится в поля «Слово» и «Классы», расположенные над кнопкой. Для успешного добавления новых данных необходимо заполнить оба поля, при чем в поле «Слово» также необходимо указать ударения, используя символ «’» (например: «до’м»), а в поле «Классы» можно перечислить их перечень через запятую без пробелов (например: «постройка,жильё»).

Кнопка «Сохранить изменения в наборе данных» позволяет конвертировать информацию представленную в соотвествующих таблицах в файл формата «\*.csv».

Кнопка «Обучить сеть» запускает процесс обучения нейронной сети, выбранное в поле «Количество эпох обучения» количество раз. В результате обучения нейронной сети будет построен график, содержащий информацию об ошибках обучения и проверки.

Кнопка «Протестировать сеть» позволяет проверить возможность нейронной сети предсказать к какому классу принадлежит объект, который пользователь ввел в поле «Слово», руководствуясь тем же правилом, что и для добавления информации в наборы данных. Результат тестирования предоставляется в виде таблицы.

Кнопка «Протестировать сеть на исходном наборе данных» позволяет проверить возможность нейронной сети предсказать к какому классу принадлежат объекты расположенные в исходном наборе данных, т.е. происходит проверка соотвествия данных которые подавались изначально в сеть и данных которые она выдает в результате.

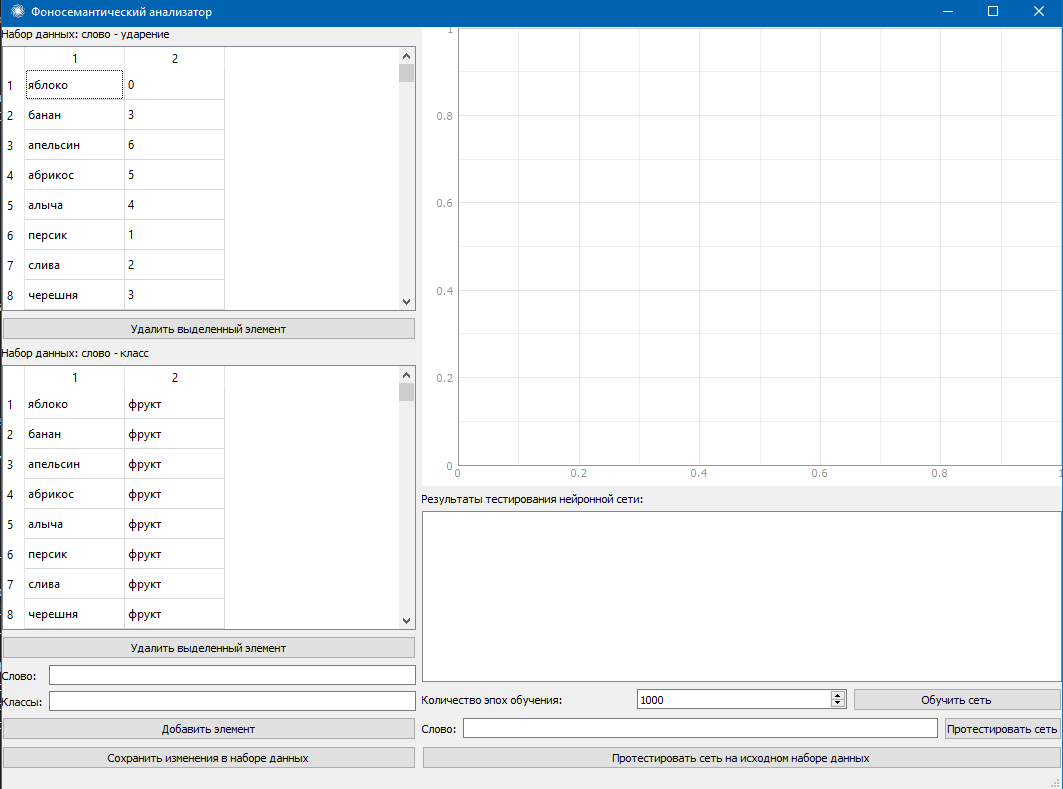


Рисунок 5 – Интерфейс разработанного программного обеспечения

* 1. Тестирование программного обеспечения

Данный раздел сожержит результаты тесирования программного обеспечения.

Рисунок 6 иллюстрируюет результат успешного добавления слова и его классов в наборы исходных данных.

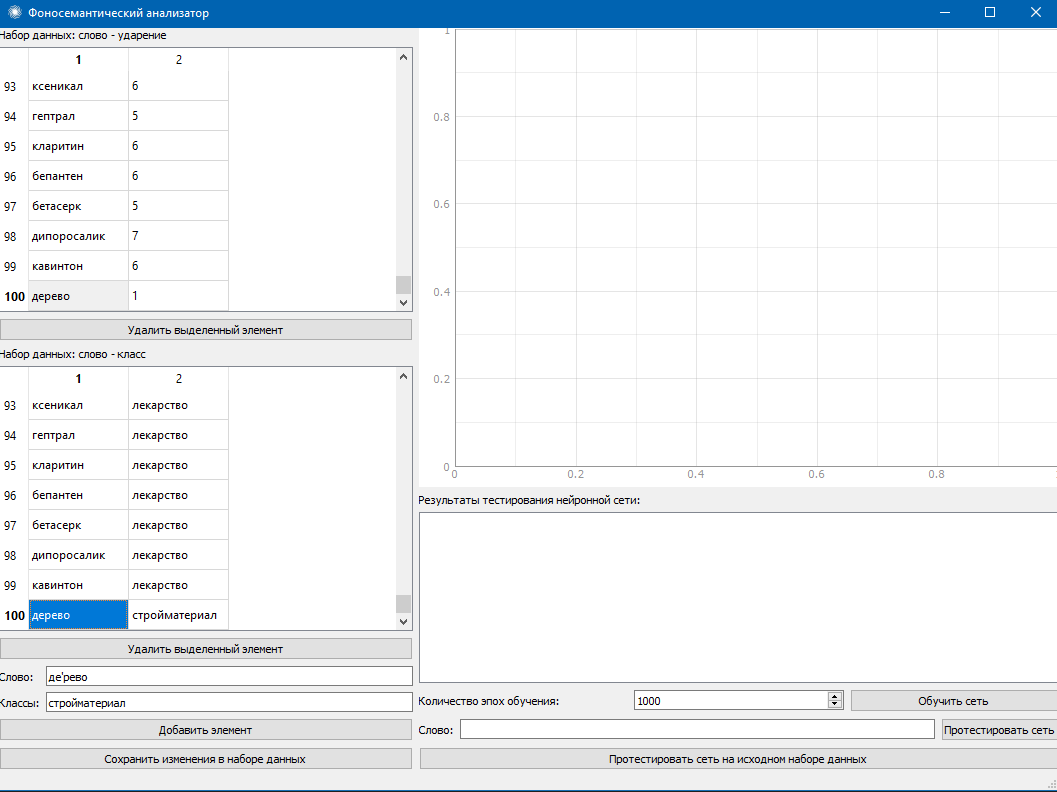


Рисунок – Результат добавления нового слова и его классов в наборы данных

Рисунок 7 иллюстрируюет результат успешного обучения нейронной сети и её тестирования на исходном наборе данных. Из результатов представленных на рисунке 7 можно увидеть, что нейронная сеть в некоторых моментах не верно предсказывает результат. Это может быть обусловленно недостаточностью исходнных данных для обучения, а также многими другими параметрами нейронной сети.

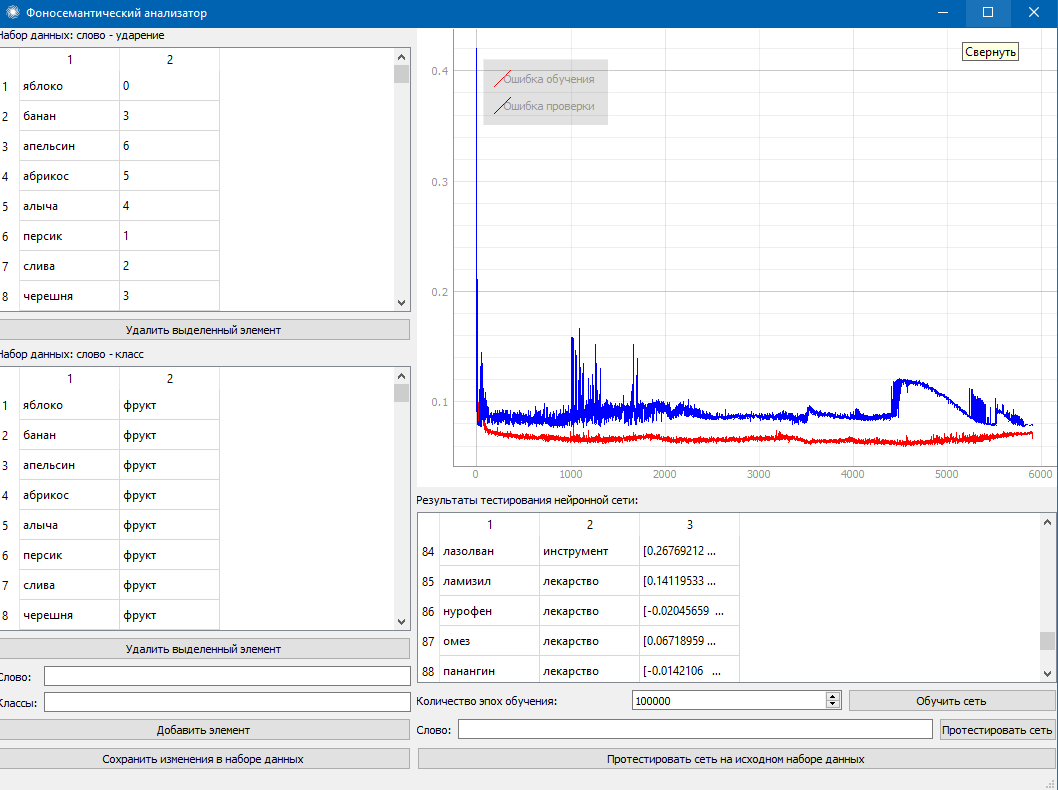


Рисунок – Результат обучения и тестирования нейронной сети на исходном наборе данных

Рисунок 8 иллюстрируюет результат успешного обучения нейронной сети и её тестирования на новом объекте «циркулярка», которого не было в обучающей выборке.

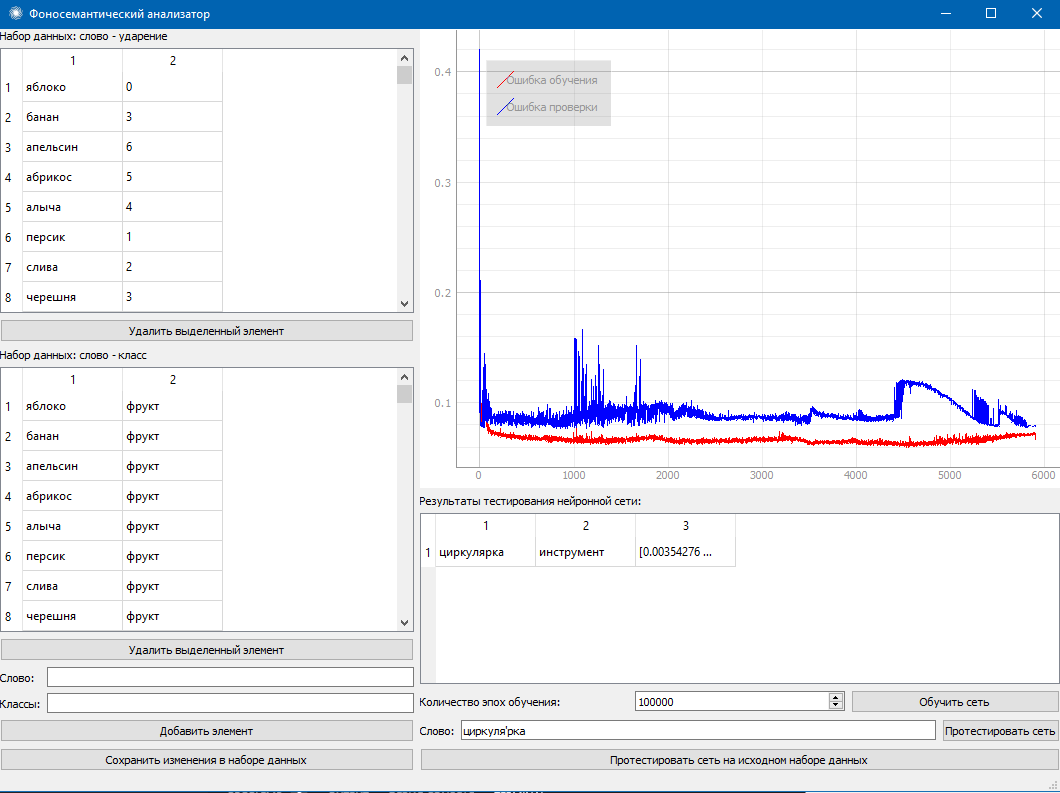


Рисунок – Результат обучения и тестирования нейронной сети на новом объекте