МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

Национальный исследовательский ядерный университет «МИФИ»

Факультет Кибернетики и Информационной безопасности

КАФЕДРА КИБЕРНЕТИКИ (№ 22)

Направление подготовки 09.04.04 Программная инженерия

Пояснительная записка

к научно-исследовательской работе студента на тему:

Сбор, анализ, разметка данных и создание среды обучения для виртуального актора на нейросетевой основе в контексте юмористических сюжетов

Группа _____ М20-504

Студент _____ Клычков М. Д.

Руководитель _____ Самсонович А. В.

Научный консультант _____

Оценка руководителя _____ Оценка комиссии _____

Члены комиссии

_____ _____

_____ _____

_____ _____

_____ _____

Москва 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

Национальный исследовательский ядерный университет «МИФИ»

Факультет Кибернетики и Информационной безопасности

КАФЕДРА КИБЕРНЕТИКИ (№ 22)

Задание на НИР

Студенту гр. М20-504 Клычкову Матвею Дмитриевичу

ТЕМА НИР

Сбор, анализ, разметка данных и создание среды обучения для виртуального актора на нейросетевой основе в контексте юмористических сюжетов

ЗАДАНИЕ

№ п/п	Содержание работы	Форма отчетности	Срок исполнения	Отметка о выполнении Дата, подпись
1.	Аналитическая часть			
1.1.	Изучение и анализ классических математических моделей Земли и способов нахождения кратчайших расстояний на них	Пункт ПЗ		
1.2.	Изучить материалы описывающие движение воздушного судна	Пункт ПЗ		
1.3.	Изучение и анализ теории компьютерной графики применительно к построению полигональных сеток	Пункт ПЗ		
1.4.	Анализ методов компьютерной графики применительно к задачам построения заданных кривых в пространстве	Пункт ПЗ		
1.5.	Оформление расширенного содержания пояснительной записки (РСПЗ)	Текст РСПЗ	10.10.2021	
2.	Теоретическая часть			
2.1.	Аффинное преобразование			
2.2.	Описание математической модели Земли	Модели		
2.3.	Разработка алгоритмов построения полигональной сетки на основе таблицы высот земной поверхности	Алгоритмы		
2.4.	Разработка алгоритмов построения траектории движения воздушного судна	Алгоритмы		
2.5.	Модификация методов аппроксимации компьютерной графики для построения траектории движения воздушного судна	Алгоритмы		
3.	Инженерная часть			
3.1.	Проектирование методов и алгоритмов компьютерной графики, адаптированных под задачу построения полигональной модели Земли	Макеты		

3.2.	Проектирование классов и функций, для расчёта траектории движения воздушного судна	Макеты		
3.3.	Проектирование классов и функций, для нахождения пересечения кривых и полигональной сетки	Макеты		
3.4.	Проектирование юнит-тестов для проверки корректности работы статической библиотеки	Макеты		
3.5.	Результаты проектирования оформить с помощью UML диаграмм	UML диаграммы		
4.	Технологическая и практическая часть			
4.1.	Реализация статической библиотеки адаптированных методов и алгоритмов компьютерной графики, для построения полигональной сетки	Исполняемые файлы, исходный текст		
4.2.	Реализация статической библиотеки адаптированных методов и алгоритмов компьютерной графики, для построения траектории воздушного судна	Исполняемые файлы, исходный текст		
4.3.	Реализация статической библиотеки адаптированных методов и алгоритмов компьютерной графики, для нахождения пересечения траектории воздушного судна и земной поверхности	Исполняемые файлы, исходный текст		
4.4.	Тестирование статической библиотеки адаптированных методов и алгоритмов компьютерной графики	Исполняемые файлы, исходные тексты тестов и тестовых примеров		
5.	Оформление пояснительной записки (ПЗ) и иллюстративного материала для доклада.	Текст ПЗ, презентация	01.01.2022	

ЛИТЕРАТУРА

- Никулин Е. А. Компьютерная геометрия и алгоритмы машинной графики. — СПб: БХВ-2. Петербург, 2003
- Эдвард Энджел. Интерактивная компьютерная графика. Вводный курс на базе OpenGL = Interactive Computer Graphics. A Top-Down Approach with Open GL. — 2-е изд. — : «Вильямс», 2001
- Ефремов А.В., Захарченко В.Ф., Овчаренко В.Н., Суханов В.Л. Динамика полета: учебник для студентов высших учебных заведений – М. : Машиностроение, 2011
- Bjarne Stroustrup The C++ Programming Language Special Edition. - М.: Издательство «БИНОМ» 2012
- Andrew Koenig, Barbara E. Moo Accelerated C++: Practical Programming by Example.

Дата выдачи задания:	Руководитель	_____	Клычков М. Д.
10.10.2021	Студент	_____	Самсонович А. В.

Реферат

Пояснительная записка содержит 46 страниц, 19 рисунков, – 32 источников литературы.

Ключевые слова: UNITY3D, EBICA, СОЦИАЛЬНО-ЭМОЦИОНАЛЬНЫЙ ИНТЕЛЛЕКТ, ВИРТУАЛЬНЫЙ АКТОР, МАШИННОЕ ОБУЧЕНИЕ, РЕККУРЕНТНЫЕ СЕТИ

Объектом исследования являются экспертные системы.

Предмет исследования - модель когнитивной архитектуры для создания Виртуального Актора.

Целью данной научно-исследовательской работы является создание прототипа Виртуального Актора, работающего по принципу идентификации семантической близости между словами определении действий и их воплощении на пространственной сцене по изначально заданному маршруту в контексте комичного сюжета. Существует два глобальных подхода к созданию социально-эмоционального интеллекта. Один основанный на нейросетях, другой на когнитивных архитектурах. В данной НИР изучаются существующие нейросетевые подходы и наиболее подробно рассматривается совокупности моделей на основе искусственных нейронных сетей Word2vec, используемых в NLP-задачах. В ходе работы над НИР был произведен сбор данных, осуществлена их разметка, разработана и протестирована модель с использованием концепции embeddings, внедрен алгоритм, передающий результаты интеллектуального анализа данных в виртуальное окружение, которое создано при помощи графического движка Unity3d. Данная работа является актуальной поскольку на данный момент эта область находится на начальных этапах развития и активной интеграции в различные индустрии. Созданная и протестированная модель интеллекта затем может быть интегрирована в другие проекты с Виртуальным Актором: виртуальный слушатель, виртуальный клоун, виртуальный танцор.

Содержание

Введение **4**

1 Исследование существующих когнитивных архитектур и анализ их недостатков **5**

1.1 Изучение и анализ существующих когнитивных архитектур 5

1.2 Изучение и анализ когнитивной архитектуры eBICA 10

1.3 Нейронные сети и их типы 12

1.4 Сбор данных, способы и инструменты 13

1.5 Выводы 14

1.6 Цели и задачи НИР 15

2 Описание моделей, отвечающих за генерацию поведения виртуального актора **16**

2.1 Постановка задачи 16

2.2 Сбор данных о юморе 17

2.3 Предварительная обработка полученных данных 18

2.4 Описание работы модели актора в старом приложении 20

2.5 Рекуррентные нейронные сети 23

2.6 Выводы 24

3 Проектирование модели поведения виртуального агента **25**

3.1 Описание предыдущей модели поведения актора и виртуального окружения . . 25

3.2 Сбор данных, метрики и инструменты 26

3.3 Инструменты для анализа текста 28

3.4 Диаграмма классов 29

3.5 Выводы 30

4 Реализация программного продукта **31**

4.1 Использование парсера данных 31

4.2 Анализ и отбор полученных данных 35

4.3 Разметка отобранных данных 38

4.4 Реализация модели поведения виртуального актора 40

4.5 Выводы 42

Заключение 43

Список литературы 44

Список литературы 45

Введение

В последнее время все большую и большую популярность набирают технологии предоставляющие возможность участвовать человеку в виртуальном мире либо технические средства, позволяющие представление виртуальную реальность в реальном мире. Виртуальные Акторы способны в будущем заменить докладчика на конференциях различного характера и представлениях.

Целью исследования является создание модели, позволяющей реализовывать виртуальному Актору действия на основе обучения нейронной сети по текстам, несущим комичный характер. Осуществляется это путем создания Виртуального агента, получающего результаты модели, с помощью которой осуществляется генерация сюжета, действия из которого осуществляет Актор, помещенный в виртуальное окружение. В данной парадигме человек (испытуемый, проводящий сеанс) может просматривать сюжет, сгенерированный моделью, воплощенного в виде аватара в игре с трехмерной графикой и оценивать его по различным параметрам. Такая модель может интерпретировать человеческое поведение и на основе экспериментов можно делать выводы о ее социальной приемлемости.

В первом разделе проводится анализ существующих когнитивных архитектур, выявляются их преимущества и недостатки по сравнению с искусственными нейронными сетями Word2vec. Ставятся цели и задачи научно-исследовательской работы.

Во втором разделе приводятся теоретические выкладки описания экспертной системы. Описание методов и инструментов сбора данных для обучения нейронной сети.

В третьем разделе приводится описание работы алгоритма, на основе которого строится генерация последовательности действий Актора. Показаны схемы реккурентных сетей. Описаны Выводы.

В четвертом разделе приводятся и анализируются программные средства для реализации полученной парадигмы.

1. Исследование существующих когнитивных архитектур и анализ их недостатков

Проводится анализ по выявлению существующих недоработок прототипа. Выявляются недостатки и преимущества по сравнению с другими моделями искусственного интеллекта.

1.1 Изучение и анализ существующих когнитивных архитектур

Одной из наиболее известных когнитивных архитектур является архитектура, составленная Jonathan Gratch и Stacy Marsella, что описано в работе [1]. Цель их исследования - создать общую вычислительную модель механизмов, лежащих в основе человеческих эмоций, которая сможет всецело их описать. Хотя такая модель может давать объяснение человеческого поведения, они рассматривают разработку вычислительных моделей эмоций как ключевой объект исследований для искусственного интеллекта, который будет способствовать развитию большого количества вычислительных систем, которые моделируют, интерпретируют или влияют на человеческое поведение. На рисунке (Рис. 1.1) демонстрируется Когнитивно-мотивационно-эмоциональная система.

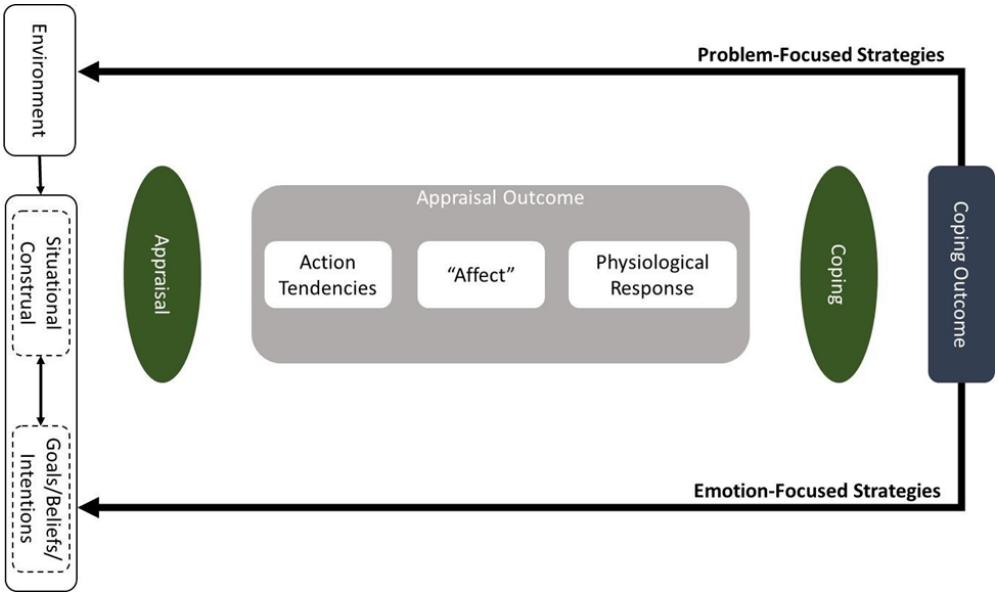

Рис. 1.1 – Когнитивно-мотивационно-эмоциональная система по материалам Smith and Lazarus.

Теория оценки служит концептуальной основой их работы, но эта психологическая теория недостаточно точна, чтобы служить спецификацией вычислительной модели. Для этого они

переделывают теорию с точки зрения методов и представлений искусственного интеллекта. Когнитивно-мотивационно-эмоциональная система Craig Smith и Richard Lazarus, показанная на рисунке 1, является представителем современных теорий оценки. Эмоция концептуализируется как двухступенчатая система контроля. Оценка характеризует отношения между человеком и его физическим и социальным окружением, называемые отношениями человека и окружающей среды, копирование поведения для восстановления или поддержания этих отношений. Поведение возникает в результате тесной связи познания, эмоций и реакций совладения: когнитивные процессы служат для построения индивидуальной интерпретации того, как внешние события соотносятся с его целями и желаниями (отношения человека и окружающей среды). Система использует эти характеристики для изменения отношений между человеком и окружающей средой, мотивируя действия, которые изменяют среду (копирование, ориентированное на проблему), или мотивируя изменения в интерпретации этих отношений (копирование, ориентированное на эмоции).

Модель PAD была разработана Albert Mehrabian и James A. Russell в 1974 году для описания и измерения эмоциональных состояний, как говорится в Работе [2]. В данной модели используются три числовых измерения для представления всех эмоций:

- A — arousal (возбуждение);
- P — pleasure (удовольствие);
- D — dominance (доминирование).

Модель PAD первоначально использовалась в теории психологии окружающей среды, а основное идеей модели было предположение о том, что физическая среда влияет на людей через их эмоциональное воздействие. На основе данной модели были построены физиологическая теория эмоций и теория эмоциональных эпизодов. Также модель использовалась для изучения невербального общения, в потребительском маркетинге и при создании анимированных персонажей, которые выражают эмоции.

В модели PAD используются трехмерные шкалы, которые в теории могут иметь любые числовые значения:

- шкала удовольствия-неудовольствия показывает, насколько приятно или, наоборот, неприятно человек себя чувствует по отношению к чему-то. Например, радость это — приятная эмоция; гнев и страх — неприятные эмоции;
- шкала возбуждения-неактивности измеряет, насколько человек чувствует возбуждение или его отсутствие. В данном случае оценивается именно возбуждение, а не интенсивность эмоций. Например, горе или депрессия характеризуются слабым возбуждением, но

сильной интенсивностью; а гнев или ярость имеют и высокую интенсивность, и высокое состояние возбуждения;

- шкала доминирования-покорности описывает чувство контроля и доминирования по сравнению со смирением и подчиненностью. Например, гнев — это доминирующая эмоция, а страх
- эмоция покорности, хотя обе они имеют неприятный характер.
- эмоция покорности, хотя обе они имеют неприятный характер.

Еще одна интересная когнитивная архитектура описана в статье трех научных деятелей Ron Sun, Nick Wilson, Michael Lynch. Статья имеет название: “Emotion: A Unified Mechanistic Interpretation from a Cognitive Architecture”. В этой статье рассматривается проект, который пытается интерпретировать эмоции - сложное и многогранное явление с механистической точки зрения, чему способствует существующая комплексная вычислительная когнитивная архитектура - CLARION. Эта когнитивная архитектура состоит из ряда подсистем: подсистем, ориентированных на действие, не ориентированных надействия, мотивационной и метакогнитивной подсистем. С этой точки зрения эмоции в первую очередь основаны на мотивации. Основываясь на этих функциональных возможностях, мы механистически (вычислительно) соединяем части вместе в рамках CLARION и фиксируем множество важных аспектов эмоций, как описано в литературе. На (Рис. 1.2) демонстрируются подсистемы когнитивной архитектуры CLARION.

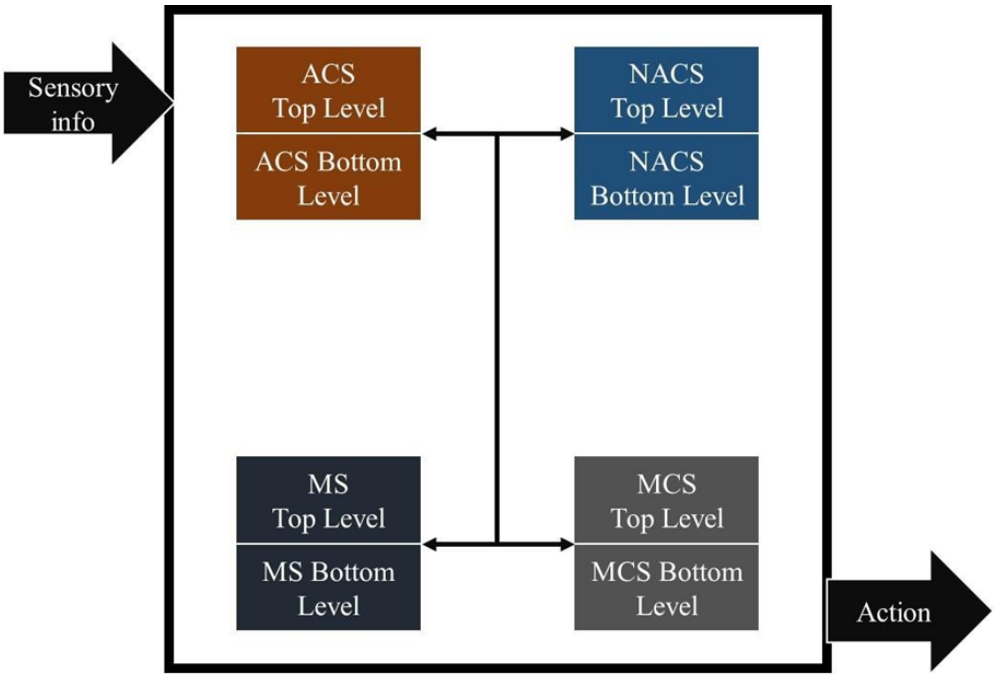

Рис. 1.2 – Подсистемы когнитивной архитектуры CLARION

Основные информационные потоки показаны стрелками. ACS означает подсистему, ори-

ентированную на действия. NACS означает подсистему, не ориентированную на действия. MC — это мотивационная подсистема. MCS означает метакогнитивную подсистему.

Получившая наибольшее распространение из всех формальных моделей представления эмоций является модель OCC (Ortony, Clore, & Collins), которая упоминается в работет [3], предложенная в 1988 году учеными Кембриджского университета. Иерархия содержит три ветви, а именно: эмоции, касающиеся последствий событий (например, радость и жалость), действия агентов (например, гордость и упрек) и аспекты объектов (например, любовь и ненависть). Кроме того, некоторые ветви объединяются в группу сложных эмоций, а именно эмоций относительно последствий событий, вызванных действиями агентов (например, благодарность и гнев). На рисунке (Рис. 1.3) демонстрируется оригинальная модель OOC.

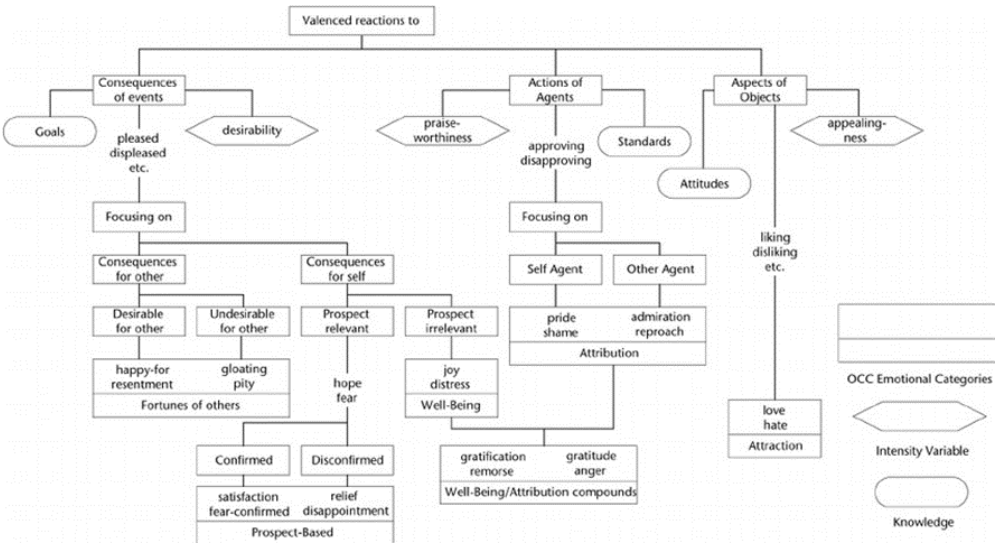

Рис. 1.3 – Оригинальная модель OCC

В основе правил динамики данной модели лежит реакция валентности (Valenced reaction). Под «валентностью» в психологии понимают внутреннюю привлекательность – «хорошую» (положительную валентность) или отвратительность – «плохую» (отрицательную валентность) события, объекта или ситуации. Эмоции формируются под воздействием трех основных факторов — последствий событий (Consequences of events), действий движущих сил (Actions of agents) и аспектов событий.

Рассмотрим левую «ветку» эмоциональной реакции. Последствия событий могут быть приносящими удовольствие (pleased) или доставляющими неудовольствие (displeased). Проведя предварительную оценку, человек фокусируется (Focusing on) на разделении последствий событий для себя (Consequences for self) и для других (Consequences for self), которые, в свою очередь могут оказаться для последних желательными (Desirable for other) или нежелательными (Undesirable for other). По поводу судеб других (Fortunes of others) в зависимости от личного

отношения — положительного или отрицательного — человек может испытывать следующие эмоции: радость за другого (Happy for), обида (Resentment), злорадство (Gloating) или жалость (Pity).

У этой модели есть свои ограничения, заключающиеся как в ее требовании упрощения человеческих эмоций, так и в ее сложном подходе к тому, как надлежит выводить эмоциональные состояния конечных пользователей посредством интерпретации поведения человека через знаки и сигналы, транслируемые людьми. Использование этой модели в ее оригинальном описании затруднено отсутствием математического аппарата, в следствии чего многие исследователи в своих Виртуальных Акторах используют упрощённые версии данной модели.

Также большой интерес представляет когнитивная архитектура, реализованная в физическом роботе, под названием - интегрированное когнитивное универсальное тело (iCub). Это когнитивная архитектура, дизайн которой основан на существующих знаниях в области робототехники, вычислений, нейробиологии и психологии, целью которой является копирование некоторых когнитивных процессов человека для их включения в человекоподобных роботов.

Эта архитектура реализована в человекоподобном роботе. Он был разработан для исследования сообществом когнитивных систем. Кроме того, он имеет лицензию «Стандартная общественная лицензия GNU (GPL)», так что любой человек может свободно использовать все наработки по данному проекту. Данная архитектура реализована в человекоподобном роботе, который имеет 53 степени свободы. По размеру он похож на ребенка трех-четырех лет и ребенка в возрасте 2,5 лет по когнитивным способностям. Кроме того, он может ползать и сидеть. Некоторые особенности, которые описаны в работе [4]

- Не хватает семантической памяти, чтобы помочь ему обобщать события;
- Невозможно сформировать привычки;
- Он учится путем подражания, проб и ошибок;
- Обнаруживает, распознает и отслеживает человеческое лицо, наблюдая за его действиями;
- Действия основаны на жестах рук, таких как встряхивание и манипулирование объектами,например, толкание, подъем и опускание.
- Действия, наблюдаемые роботом, изучаются и сохраняются в базе данных в процессе обучения.

На рисунке (Рис. 1.4) представлена схема работы iCub.

Вспомогательным инструментом при создании актора, наделенного социально- эмоциональным интеллектом, может являться - имитация моторного обучения (IML). IML начинает

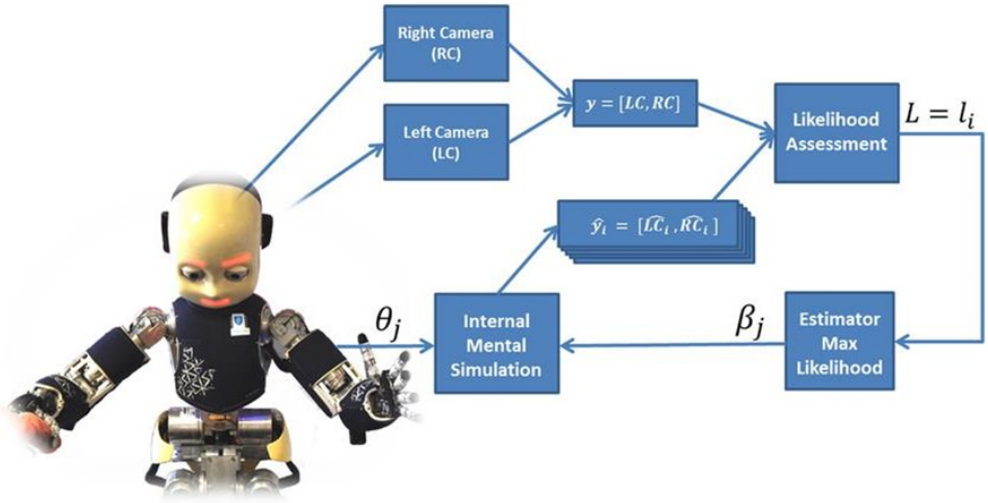

Рис. 1.4 – Схема работы iCub

наблюдать за другим актором, осуществляющим некоторую цепочку действий, затем категоризирует действия (определяет какую цель преследуют данные действия) одновременно отслеживая изменения точки обзора, окружающей среды, положения и типов объектов. Другими словами, когда Виртуальный агент неоднократно наблюдает за определенной новой последовательностью действий, каждый из знакомых элементов действия активирует соответствующее моторное представление через существующие ассоциации. Данное наблюдение формирует связи между элементарными моторными представлениями. Эта связь представлений составляет моторное обучение и улучшает имитационное движение. Способность моторной системы интегрировать разные части организма позволила бы создать обширный репертуар моторного поведения путем смешивания выходных сигналов разных частей организма, чтобы конечный результат отражал относительный и взвешенный вклад каждого в достижении цельной имитации движения. Поскольку невозможно воспроизвести функционирование мозга, были созданы модели, которые пытаются имитировать различные функции и поведение.

1.2 Изучение и анализ когнитивной архитектуры eBICA

Архитектура состоит из семи компонентов: интерфейсный буфер, рабочая, процедурная, семантическая и эпизодическая системы памяти, система ценностей и система когнитивных карт. Три основных строительных блока для этих компонентов — это ментальные состояния, схемы и семантические карты. Семантическая память — это коллекция определений схем. Буфер интерфейса заполняется схемами. Рабочая память включает активные психические состояния. Эпизодическая память хранит неактивные психические состояния, сгруппированные в эпизоды - предыдущее содержимое рабочей памяти. Следовательно, эпизодическая память состоит из структур, аналогичных тем, которые обнаруживаются в рабочей памяти, но которые

«заморожены» в долговременной памяти [5]. Процедурная память включает в себя примитивы. Система ценностей включает в себя шкалы, представляющие основные значения. Система когнитивных карт включает, в частности, семантические карты эмоциональных ценностей. Семантическая карта использует абстрактное метрическое пространство (семантическое пространство) для представления семантических отношений между ментальными состояниями, схемами и их 13 экземплярами, а также для присвоения значений их оценкам. На (Рис.1.5) демонстрируется семантическая карта [5].

Рис. 1.5 – Семантическая карта

Для когнитивного семантического отображения может использоваться слабое когнитивное семантическое картирование. Идея заключается в том, чтобы расположить представления на основе очень немногих основных семантических измерениях. Эти измерения могут возникать автоматически, если стратегия состоит в том, чтобы объединить синонимы и антонимы друг от друга. Карта, часть которой показана на рисунке 6 является результатом этого процесса. Эта карта не очень хорошо отделяет различные значения друг от друга: например, основные и сложные чувства. Однако она классифицирует значения в соответствии с их семантикой. Рисунок (Рис. 1.6) демонстрирует примеры простейших эмоциональных элементов в рамках eBICA [6].

(A) Схема имеет оценку в качестве своего атрибута. Это также атрибут головного узла. Значение этого атрибута - «доминантный», что означает, что действие воспринимается как проявление доминирования, или агент воспринимается как «доминантный по отношению ко мне» и т. Д. (B) Психическое состояние имеет оценку атрибут, который представляет собой эмоциональное состояние и самооценку агента в данный момент в данной ментальной перспективе.

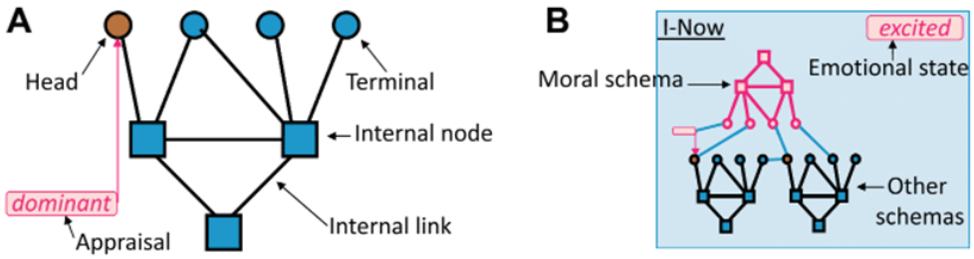

Рис. 1.6 – примеры эмоциональных элементов в рамках eBICA

Показанная ценность этой оценки «взволнована», что означает, что агент находится в возбужденном эмоциональном состоянии. Моральная схема, показанная в B, связывается с частью содержания психического состояния (включая определенный образец оценок) и представляет оценку выбранного образца, например, образец взаимодействий и взаимных оценок двух агентов, упомянутых в ментальном состоянии.

1.3 Нейронные сети и их типы

Нейро́нная сеть (также искусственная нейронная сеть, ИНС) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети У. Маккалока и У. Питтса. После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

Разделяют несколько основных разновидностей Нейронных сетей, согласно работе [7], а именно:

- Нейронные сети прямого распространения
- Сети радиально-базисных функций
- Нейронная сеть Хопфилда (Hopfield network, HN)
- Цепи Маркова (Markov chains, MC или discrete time Markov Chains, DTMC)
- Машина Больцмана (Boltzmann machine, BM)
- Ограниченная машина Больцмана (restricted Boltzmann machine, RBM)
- Автокодировщик (autoencoder, AE)
- Разреженный автокодировщик (sparse autoencoder, SAE)
- Вариационные автокодировщики (variational autoencoder, VAE)
- Шумоподавляющие автокодировщики (denoising autoencoder, DAE)

– Сеть типа «deep belief» (deep belief networks, DBN)
– Свёрточные нейронные сети (convolutional neural networks, CNN)
– Развёртывающие нейронные сети (deconvolutional networks, DN)

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа. Рекуррентные нейронные сети (РНС, англ. Recurrent neural network; RNN) — вид нейронных сетей, где связи между элементами образуют направленную последовательность [8]. Благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки. В отличие от многослойных перцептронов, рекуррентные сети могут использовать свою внутреннюю память для обработки последовательностей произвольной длины. Поэтому сети RNN применимы в таких задачах, где нечто целостное разбито на части, например: распознавание рукописного текста или распознавание речи. Было предложено много различных архитектурных решений для рекуррентных сетей от простых до сложных. В последнее время наибольшее распространение получили сеть с долговременной и кратковременной памятью (LSTM) и управляемый рекуррентный блок (GRU).

В последнее время наибольшую популярность для решения задач тематической классификации, применяемой для выделения семантического смысла текста, приобрели глубокие нейронные сети, так как они позволяют достичь наивысшей точности среди всех известных моделей машинного обучения. В частности, сверточные нейронные сети совершили прорыв в классификации изображений. В настоящее время они успешно справляются и с некоторыми задачами автоматической обработки текстов. Более того, как утверждается в некоторых исследованиях сверточные сети подходят для этого даже лучше рекуррентных нейронных сетей, которые чаще всего используются для анализа текстовых последовательностей. С другой стороны, использование сверточных сетей для классификации текстов мало исследовано. Поэтому исследование применения сверточных нейронных сетей для задачи классификации текстов в качестве альтернативы рекуррентным нейронным сетям представляет практический интерес, что описано в [**neural10**].

Для решения поставленной задачи требуется получить способ представления данных в виде, пригодном для обработки сверточной нейронной сетью. Например, в виде матрицы вещественных чисел. Наиболее распространенным является способ отображения каждого слова в многомерное векторное пространство. В рамках данной работы векторные представления слов строились на основе модели word2vec [9]. sa

1.4 Сбор данных, способы и инструменты

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения — одно из главных особенностей нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искажённых данных.

Сбор данных, и их правильная разметка является неотъемлемой частью качественного обучения нейронной сети. Для реализации сбора данных используются инструменты для анализа данных в зависимости от того в каком формате они находятся.

Одним из примеров является HTML-парсер - инструмент, созданный для чтения HTML-страниц, что рекомендуется применять для решения данной проблемы согласно [10], и извлечения дерева синтаксического анализа(мета-данные). Одним из таких инструментов является Beautiful Soup Beautiful Soup - пакет Python для синтаксического анализа документов HTML и XML (включая искаженную разметку, то есть незакрытые теги, названные так в честь супа тегов). Он создает дерево синтаксического анализа для проанализированных страниц, которое можно использовать для извлечения данных из HTML, что полезно для парсинга веб-страниц. Один из методов актуальный для решения задачи по интеллектуальному анализу данных это - получение сторонних данных предварительно обученных моделью.

1.5 Выводы

Методики для реализации технологии представления визуальных агентов в различных областях, актуальны в наше время и до сих пор нет четкой определенности какая из концепций наиболее подходит для имитации “человеческого поведения”, путем машинной генерации.

Автоматизация извлечения различной информации из текстов, является одной из альтернатив обучения визуальных агентов, однако, так как тексты чаще всего либо являются слабо структурированными, либо вообще не обладают структурой с точки зрения решаемой задачи, особо важным стало направление интеллектуального анализа текстов, включающее в себя методы классификации и анализа текстов на основе алгоритмов машинного обучения.

1.6 Цели и задачи НИР

Целью данной научно-исследовательской работы является создание прототипа Виртуального Актора, работающего по принципу идентификации семантической близости между словами определении действий и их воплощении на пространственной сцене по изначально заданному маршруту в контексте комического сюжета.

1. Определить подход, согласно которому; будет разрабатываться альтернативная линия сюжетов и их воплощения Виртуальным Актором.
2. Собрать и осуществить разметку данных для обучения нейронной сети;
3. Осуществить глубокий интеллектуальный анализ данных по размеченным массивам данных.
4. Разработать, алгоритм передающей результаты анализа виртуальному Агенту.
5. Реализовать Визуального агента и сцену, используя межплатформенную среду разработки компьютерных игр Unity3d.

2. Описание моделей, отвечающих за генерацию поведения виртуального актора

В данном разделе приводится теоретическое описание модели.

2.1 Постановка задачи

В рамках научно-исследовательской работы был выявлен альтернативный путь решения поставленной задачи задачи, который выражается в обучения нейронной сети. Данный метод рассматривается параллельно реализации с использованием когнитивной архитектурой eBICA. eBICA – “emotional biologically inspired cognitive architecture” – “эмоциональная биологически вдохновленная когнитивная архитектура”. В этой архитектуре эмоциональные элементы добавлены практически ко всем процессам за счет модификации основных строительных блоков архитектуры. Ключевым моментом этой когнитивной архитектуры являются оценки, которые связаны со схемами и психическими состояниями как их атрибуты, моральные схемы, которые контролируют модели оценок и представляют социальные эмоции, а также семантические пространства, которые дают значения этих оценок.

Как видно из (Рис. 2.1), архитектура представляет собой конгломерат компонентов: интерфейсный буфер, рабочая, процедурная, семантическая и эпизодическая системы памяти, система ценностей и система когнитивных карт [6]. Три основных строительных блока для этих компонентов - это ментальные состояния, схемы и семантические карты. Семантическая память - это коллекция определений схем. Буфер интерфейса заполняется схемами.

Рабочая память включает активные психические состояния. Эпизодическая память хранит неактивные психические состояния, сгруппированные в эпизоды - предыдущее содержимое рабочей памяти. Следовательно, эпизодическая память состоит из структур, аналогичных тем, которые обнаруживаются в рабочей памяти, но которые «заморожены» в долговременной памяти. Процедурная память включает в себя примитивы. Система ценностей включает в себя шкалы, представляющие основные значения. Система когнитивных карт включает, в частности, семантические карты эмоциональных ценностей. Семантическая карта использует абстрактное метрическое пространство (семантическое пространство) для представления семантических отношений между ментальными состояниями, схемами и их экземплярами, а также для присвоения значений их оценкам.

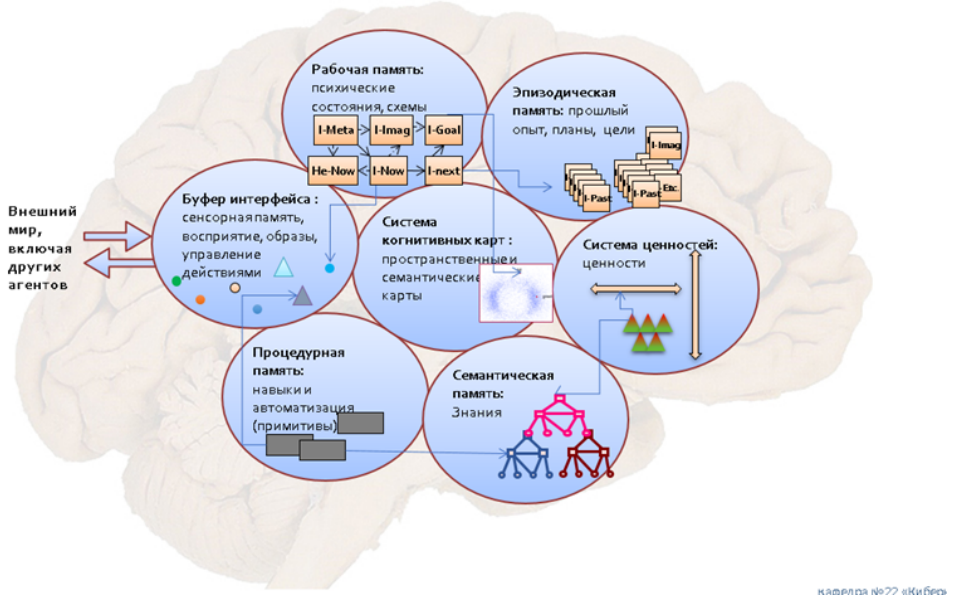

Рис. 2.1 – Структура когнитивной архитектуры eBICA

2.2 Сбор данных о юморе

Для обучения нейронной осуществляется сбор данных. Самой частой альтернативой сбора данных является его получение из открытых ресурсов при помощи парсинга кода HTML.

Для парсинга страниц используются несколько основных методов описанных в [11].

1. Парсинг получаемого JSON при помощи Api
2. Парсинг по XHR запросам в консоли разработчика браузера
3. Поиск JSON в html странице
4. Рендеринг код страницы через автоматизацию браузера

В частности, используется нейросетевой подход для парсинга данных. Такой подход актуален так как владельцы информационных продуктов часто не заинтересованы в том, чтобы пользователи парсили данные с их ресурсов и добавляют различные ограничения и алгоритмы по распознаванию парсинга. Самым простым примером является ограничение на количество запросов к сайту, вставка капчи, проверка IP пользователя на соответствие региону либо количество посещений страницы в минуту. Так же очень частой проблемой является динамическая доработка сайтов, которая в следствии заставляет переделывать алгоритм парсинга.

Для того, чтобы парсинг страниц был более практичным, используют нейросетевой подход. Одним из таких инструментов является TensorFlow, это открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия.

Для решения проблемы с IP используют спуфинг, что представляет из себя принцип маскировки одного юнита под другого путём фальсификации данных и позволяет получить соответствующие преимущества.

Для анализа были взяты данные с различных сайтов с юмористическим посылом. В первую очередь рассматривались сайты со скетчами или сценками (в том числе скетчи из различных юмористических передач). Для каждого скетча и сценки на сайте должно быть их текстовое описание, поясняющее сюжет или сценарий. Далее рассматривались сайты с анекдотами и различными шутками. Такой приоритет был поставлен в связи с тем, что основой юмора в клоунаде должны быть действия акторов на сцене, а не юмористический эффект, полученный в результате “игры слов”.

2.3 Предварительная обработка полученных данных

На вход модели должен подаваться заранее обработанный корпус текстов. Предварительная обработка состоит из следующих этапов [12]:

- Удаление всех знаков препинания, чисел и слов «нецелевых» языков (не предназначенных для обработки моделью).
- Разбиение текста на предложения. Для этого был выбран пакет библиотек Natural Language Toolkit (NLTK). Данная библиотека применяет регулярные выражения, а также некоторые алгоритмы машинного обучения для обработки естественного языка. Базовая версия NLTK не поддерживает разбиение русскоязычных текстов на предложения, поэтому использовалась модификация, расширяющая функционал библиотеки.
- Удаление «стоп-слов» – слов, не несущих определенной смысловой нагрузки, но при этом затрудняющих обработку исходного текста. Обычно для каждой специфической задачи применяется свой словарь стоп-слов, однако для нашей задачи достаточно стандартного словаря, содержащего буквы, частицы, предлоги, союзы, местоимения, числительные. Установлено, что удаление стоп-слов из тренировочного набора значительно снижает вычислительную стоимость, а также повышает точность модели.
- К корпусу текстов применяется стемминг или лемматизация. Это позволяет сократить размер словаря и искать семантически близкие слова, а не разные формы одного слова. Стемминг – это поиск основы слова, причем не обязательно совпадающей с корнем. Он имеет высокую скорость работы, но наиболее эффективен для английского языка, так как в нем для нахождения основы слова обычно достаточно удалить окончание. Для русского языка стемминг малоэффективен, поэтому применяется более ресурсоемкий алгоритм лемматизации. Лемматизация – это процесс приведения слова к начальной форме. В дан-

ной работе лемматизация осуществлялась морфологическим анализатором MyStem .

– Дополнение предложений до одинаковой длины с использованием нейтрального слова, так как сверточные нейронные сети способны обрабатывать только последовательности одинаковой длины.

Далее на начальном этапе необходимо перевести слова естественного языка в форму, пригодную для анализа сверточной нейронной сетью. Для этого лучше всего подходит векторное представление слов. Кроме того, среди всех моделей выберем ту, которая наиболее точно отражает реальные взаимосвязи между словами, а именно семантическую близость. Отметим, что модель не должна быть слишком требовательной к вычислительным ресурсам, чтобы было возможно совершать обучение сети на достаточно больших объемах данных.

Для выявления семантических связей между словами воспользуемся предположением лингвистики – дистрибутивной гипотезой: лингвистические единицы, встречающиеся в схожих контекстах, имеют близкие значения. Во многих моделях обработки текстов входные данные кодируются унарным кодом (one-hotencoding) – вектором, размерность которого равна мощности словаря более подробно описанный в [**neural02**]. Элемент, соответствующий И.А. Батраева, А.Д. Нарцев, А.С. Лезгян номеру слова в словаре, равен единице, а остальные элементы равны нулю. Однако у этого метода, согласно [**neural02**] есть ряд существенных недостатков:

– словари естественных языков могут быть достаточно объемными и исчисляться десятками и сотнями тысяч слов; следовательно, если каждое слово кодировать таким вектором, объем данных становится слишком большим;
– при таком способе кодирования теряется связь между словами: все слова считаются разными и никак не связанными между собой.

В силу вышесказанного, one-hot encoding не подходит для анализа семантической близостислов. Поэтому для данной задачи воспользуемся другим способом кодирования – распределенным представлением слов. Распределенное (или векторное) представление слов – это способ представления слов в виде векторов евклидова пространства, размерность которого обычно равна нескольким сотням. Основная идея заключается в том, что геометрические отношения между точками евклидова пространства будут соответствовать семантическим отношениям между словами. Например, слова, представленные двумя близко расположенными точками векторного пространства, будут, скорее всего, синонимами или просто тесно связанными по смыслу словами. Семантическая близость слов вычисляется как расстояние между векторами, для чего используется так называемая косинусная мера.

Word2vec включает в себя две различные архитектуры – CBOW (Continuous Bag of Words

– непрерывный мешок слов) и Skip-gram [13]. CBOW пытается предсказать слово, исходя из текущего контекста, а Skip-gram, наоборот, пытается предсказать контекст по текущему слову. Для реализации модели была выбрана архитектура Skip-gram, которая, несмотря на меньшую скорость обучения, лучше работает с редкими словами. Предварительно обработанный текст можно подавать на вход модели, после чего будут выполнены следующие действия:

- считывается корпус текстов и рассчитывается, сколько раз в нем встретилось каждое слово;
- из этих слов формируется словарь, который сортируется по частоте слов; также из словаря
- для сокращения его размера удаляются редкие слова;
- модель идет по субпредложению (обычно предложение исходного текста или абзац) окном определенного размера; под размером окна понимается максимальная длина между текущим словом и словом, которое предсказывается. Оптимальный размер окна – 10 слов;
- к данным, находящимся в текущем окне, применяется нейронная сеть прямого распространения с линейной функцией активации скрытого слоя и функцией активации softmax для выходного слоя.

Из всего вышесказанного ясно, что матрицы, задающие скрытый и выходной слои, получаются чрезвычайно большими. Это делает обучение сети долгим процессом. Поэтому используются различные оптимизации, которые позволяют существенно снизить, согласно [**neural03**] временные и вычислительные затраты, незначительно потеряв в точности. Одной из таких модификаций является субсемплирование. Чтобы избежать дисбаланса между редкими и часто встречающимися словами, используется простой подход: каждое слово отбрасывается с вероятностью, зависящей от частоты вхождения этого слова в текст.

2.4 Описание работы модели актора в старом приложении

На момент начала выполнения работы уже была реализована система работы виртуальных агентов, и она состоит в том, что сперва считываются действия и объекты с заданными для них параметрами и значениями из Excel файла, а также инициализируются значения. Затем выбор действий происходит в следующем порядке: в радиусе вокруг оценок виртуального актора выбираются действия, которые попадают в этот радиус, а также проверяются различные условия, необходимые для выполнения действия. Если условия не выполняются, то действие не может быть выбрано. Затем, после того как в список добавлены все действия, которые могут быть выполнены, рассчитываются вероятности на основе оценок и рассчитанных констант. Также, если действие повторное, его вероятность несколько занижается. После расчета вероятностей

выбирается действие, которое влияет на оценки самого клоуна, а также, если есть, на цель клоуна.

Помимо этого происходит замена состояний объектов и виртуальных акторов. После этого происходит перерасчет оценок Appraisals и Feelings [6].

Основная модель eBICA определяет поведение виртуального актора исходя из следующих факторов:

- соматический;
- рациональный;
- когнитивный.

Нравственный фактор регулирует отношения первого актора со вторым на основе системы ценностей (представленной семантической картой) и моральных схем. Под когнитивным фактором понимается учет соображений нравственности, этики и морали, общей системы ценностей, понятий о добре и зле, о собственном достоинстве, эмпатии, соображений эстетики, стремлений к простоте и элегантности, и т.д. Учет этих соображений возможен на основе когнитивных оценок (appraisals) всех релевантных агентов, событий, их возможных действий и последствий этих действий, фактов, свойств, отношений, и т.д. Возможен вариант модели, в которой ответное действие может выбираться лишь из двух вариантов: положительная реакция на действие человека и отрицательная. Данная версия модели весьма неплохо работает даже с таким ограничением. Но невозможно придерживаться данной парадигмы при увеличении количества возможных вариантов для взаимодействия между акторами. В данной модели необходимо учесть пересчет оценок Appraisals и Feelings. Для пересчета оценок Appraisals используется следующая формула 2.1:

$$Appraisals = (1 - r) * Appraisals + r * Action \tag{2.1}$$

где Appraisals - оценка, r - эмпирически вычисленная константа экспоненциального затухания, Action - оценка совершаемого действия на семантической карте.

Одновременно с Appraisals пересчитываются так называемые “чувства” Feelings согласно режиму работы моральной схемы. Аффективное пространство VAD – это трехмерное векторное пространство, точки которого соответствуют определенным эмоциональным состояниям, или аффектам, представленным триплетами значений (Valence, Arousal, Dominance). Существуют и сходные модели: PAD (Pleasure, Arousal, Dominance), EPA (Evaluation, Potency, Arousal) и другие. Здесь мы используем модель VAD. Соответственно, под «семантической картой» здесь часто понимается ее конкретная разновидность: аффективная карта (или когнитивная семантическая

карта).

Шкалы имеют следующие значения:

- dominance – варьируется при значении от 0 (покорность) до +1 (доминантность) и описывает соответствующие чувства;
- valense – при значениях от -1 до 0 показывает уровень негатива или радости соответственно;
- arousal – значения от -1 до 1 показывают уровень возбуждения (заинтересованности), к примеру, гнев по уровню возбуждения сильнее раздражительности, но слабее ярости.

Оценки представлены в виде векторов на трехмерной семантической карте [5], 1.5. Моральная схема определяет общую установку на оценку поведения акторов, согласно их ролям и типу ситуации. Ее целью (как агента) является достижение и поддержание «нормального» положения дел, определенного набором Feelings. Вообще говоря, моральная схема состоит из двух частей: части, распознающей тип ситуации и осуществляющей привязку (binding), и части, реализующей динамику схемы. В случае парадигмы актора можно считать, что моральная схема одна, уже привязана, и потому первая часть ее не актуальна.

Субъективные оценки (Feelings) генерируются по определенным правилам на основании истории объективных оценок и состояний системы. Грубо говоря, Feelings – это субъективное представление о том, каким оцениваемый актор является «на самом деле», и, следовательно, какого поведения от него нужно ожидать и на какое место его нужно ставить своим поведением. Следовательно, выбор поведения актора должен осуществляться так, чтобы приблизить Appraisals к Feelings.

Значение Feelings определяет моральная схема, которая может работать в одном из трех режимов. Первый режим основывается на формуле 2.2:

$$Feelings = beta * Appraisals \tag{2.2}$$

где beta – эмпирически вычисленная константа.

В данном режиме схема говорит, что если актор ведет себя хорошо, то к нему нужно относиться как к хорошему, и т.д.

Цель данного процесса – распознать и классифицировать актора, выработать отношение к нему и приписать ему определенную роль во взаимоотношениях.

В данном режиме моральная схема работает пока разница между квадратами норм Feeling и Appraisals не станет меньше некоторого значения.

Суть второго режима заключается в том, что значение Feeling фиксировано и экстремаль-

но по абсолютной величине, т.е. находится на сфере, ограничивающей семантическую карту (предположим, что есть такая сфера). Направленность вектора Feeling может быть либо произвольной, определенной предысторией, либо дискретной – вдоль одной из осей.

Третий режим состоит в том, что значения Feelings меняются 2.3, подстраиваясь под текущие значения Appraisals (здесь r1 может быть отличным от r):

$$Feelings = (1 - r_1) * Feelings + r1 * (Appraisal - Feelings) \tag{2.3}$$

Соответственно значения Appraisals и Feelings как говорится в работе [14] пересчитываются после каждого действия первого актора, направленного на второго актора. Также пересчет оценок происходит после определения и совершения одним из акторов ответного или самостоятельного действия. В данном контексте под термином “самостоятельное действие” имеется в виду действие, основанное лишь на текущем состоянии мира и значений векторов Appraisals и Feelings акторов, отобрежнные на (Рис. 2.2).

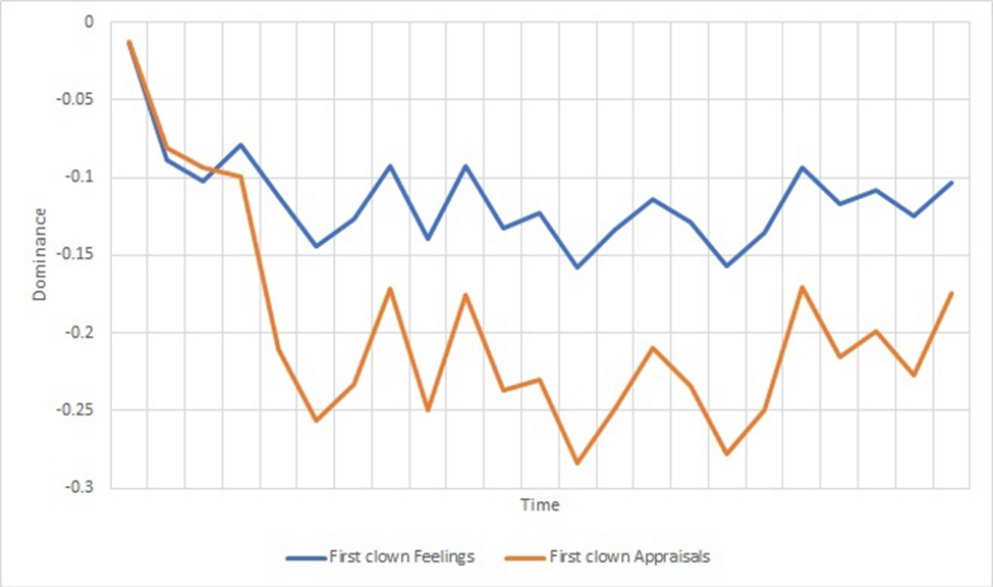

Рис. 2.2 – Корреляция значений Appraisals (оранжевая) и Feelings (синяя) для показателя доминантности на протяжении времени/действий с шагом в 5 секунд

Согласно (Рис. 2.2) мы видим, что работа модели сводится к выбору действия, которое будет максимально приближать Appraisals к Feelings и вектор соматического состояния к начальному положению.

2.5 Рекуррентные нейронные сети

LSTM — это класс возвратных нейронных сетей. Поэтому, прежде чем мы сможем перейти к LSTM, важно понять нейронные сети и рекуррентные нейронные сети [8].

Нейронные сети - Искусственная нейронная сеть представляет собой слоистую структуру из связанных нейронов, вдохновленную биологическими нейронными сетями. Это не один алгоритм, а комбинация различных алгоритмов, которая позволяет нам выполнять сложные операции с данными. Рекуррентные нейронные сети - это класс нейронных сетей, предназначенных для работы с временными данными. Нейроны RNN имеют состояние / память ячейки, и ввод обрабатывается в соответствии с этим внутренним состоянием, которое достигается с помощью петель в нейронной сети. В RNN существуют повторяющиеся модули «tanh» слоев, которые позволяют им сохранять информацию. Однако ненадолго, поэтому нам нужны модели LSTM.

LSTM - Это особый вид рекуррентной нейронной сети, способной изучать долгосрочные зависимости в данных. Это достигается за счет того, что повторяющийся модуль модели имеет комбинацию четырех слоев, взаимодействующих друг с другом [8].

На (Рис. 2.3) отображены структуры вышеупомянутые виды рекуррентных сетей:

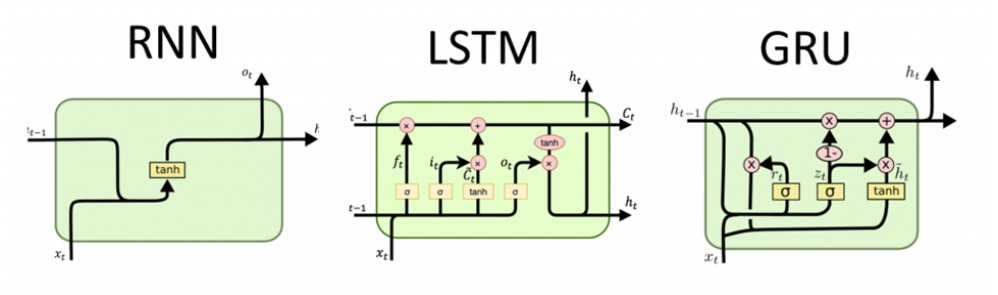

Рис. 2.3 – Структуры рекуррентных нейронных сетей

2.6 Выводы

В данном разделе была сформулирована постановка задачи, рассмотрены возможные ее решения с помощью архитектуры eBICA и нейронных сетей. Обозначена проблема сбора данных для обучения нейронной сети с веб-сервисов, и были выявлены способы решения данной проблемы. А также рассмотрена проблема обработки данных применительно к модели действий виртуальных агентов. Обозначены основные положения поведения виртуальных агентов. Выбран вид нейронной сети, которая будет моделировать поведение виртуальных акторов, а именно рекуррентная нейронная сеть.

3. Проектирование модели поведения виртуального агента

В этом разделе описывается и обосновывается выбор инструментария для проектирования и программного воплощения модели поведения актора в заданной парадигме. Описываются ключевые моменты проектирования и программной реализации модели поведения актора.

3.1 Описание предыдущей модели поведения актора и виртуального окружения

В ходе выполнения программы записывались данные с оценками для первого и второго клоуна, а также с выводом сообщения совершенного действия, данные представлены на (Рис. 3.1).

Message	First clown Appr	First clown App	First clown App	First clown F	First clown Feel	First clown Fe
First clown bumped his head off the chandelier ouch	-0.012	0.015	-0.009	-0.0132	0.0165	-0.0099
Second clown ruffles first clown's hair	-0.02176	0.0147	0.00518	-0.023936	0.01617	0.005698
First clown takes the rat	-0.02176	0.0147	0.00518	-0.023936	0.01617	0.005698
Second clown pats first clown on a shoulder	-0.0205424	0.009553	0.0071282	-0.0225966	0.0105083	0.00784102
First clown ruffles second clown's hair	-0.030131552	0.00936194	0.020985636	-0.0331447	0.010298134	0.0230842
Second clown ruffles first clown's hair	-0.039528921	0.009174701	0.034565923	-0.0434818	0.010092171	0.03802252
First clown pats second clown on a shoulder	-0.038133632	0.004082954	0.036220264	-0.041947	0.00449125	0.03984229
Second clown stepped on the shotgun the gun shoots	-0.060608286	0.031919636	0.022771453	-0.0666691	0.0351116	0.0250486
First clown stumbled on the hammer screams of pain	-0.073790038	0.045962047	0.00708831	-0.081169	0.050558252	0.00779714
Second clown declines the - offered by first clown	-0.110838436	0.044123565	0.006804777	-0.1219223	0.048535922	0.00748526
First clown leaves rat -	-0.110838436	0.044123565	0.006804777	-0.1219223	0.048535922	0.00748526
Second clown stumbled on the bottle of vodka the bottle rolls	-0.114621668	0.049241094	0.002668682	-0.1260838	0.054165203	0.00293555
First clown bumped his head off the chandelier ouch	-0.123183018	0.062763861	-0.006411379	-0.1355013	0.069040247	-0.00705252
Second clown declines the - offered by first clown	-0.158255697	0.060253307	-0.006154923	-0.1740813	0.066278637	-0.00677042
First clown ruffles second clown's hair	-0.165090583	0.05904824	0.007968175	-0.1815996	0.064953064	0.00876499
Second clown bumped his head off the chandelier ouch	-0.172137865	0.072276793	-0.00127087	-0.1893517	0.079504472	-0.00139796
First clown stepped on the shotgun the gun shoots	-0.189252351	0.097385721	-0.013220035	-0.2081776	0.107124294	-0.01454204
Second clown stumbled on the bottle of vodka the bottle rolls	-0.191467304	0.101438007	-0.016955635	-0.210614	0.111581808	-0.0186512
First clown stumbled on the hammer screams of pain	-0.200723285	0.113394867	-0.031446966	-0.2207956	0.124734353	-0.03459166
Second clown ruffles first clown's hair	-0.206708819	0.111126969	-0.016818026	-0.2273797	0.122239666	-0.01849983
First clown stumbled on the hammer screams of pain	-0.215507554	0.12279316	-0.031313486	-0.2370583	0.135072476	-0.03444483
Second clown stumbled on the hammer screams of pain	-0.224042328	0.134109366	-0.045374081	-0.2464466	0.147520302	-0.04991149
First clown stumbled on the hammer screams of pain	-0.232321058	0.145086085	-0.059012859	-0.2555532	0.159594693	-0.06491414
Second clown pats first clown on a shoulder	-0.228997847	0.138635224	-0.05642273	-0.2518976	0.152498746	-0.062065
First clown stepped on the shotgun the gun shoots	-0.243837934	0.161089815	-0.066165821	-0.2682217	0.177198796	-0.0727824
Second clown takes the rat	-0.243837934	0.161089815	-0.066165821	-0.2682217	0.177198796	-0.0727824
First clown stepped on the shotgun the gun shoots	-0.258084416	0.182646222	-0.075519188	-0.2838929	0.200910844	-0.08307111
Second clown stumbled on the hammer screams of pain	-0.265341884	0.192166836	-0.088253612	-0.2918761	0.211383519	-0.09707897
First clown stumbled on the bottle of vodka the bottle rolls	-0.266035046	0.194323499	-0.09048854	-0.2926386	0.213755849	-0.09953739

Рис. 3.1 – Оценки и действия совершенные первым виртуальным актором

Для осуществления контроля за действиями акторов и их анализом система ведет записи в журнал событий. Журнал представляет собой текстовую таблицу. Записи выглядят следующим образом: каждая строка журнала соответствует своему событию, строка начинается с указания времени, когда была произведена запись. После указания времени указывается тип сообщения. Далее в сообщении показывается исполнитель действия, цель действия и номер действия из таблицы действий. После идет содержание сообщения, поясняющее произошедшее событие. После чего показаны оценки Appraisals и Feelings для доброжелательности, возбужденности и доминантности.

На (Рис. 3.2) представлены гистограммы частот действий, совершаемых испытуемыми и модельным человеком.

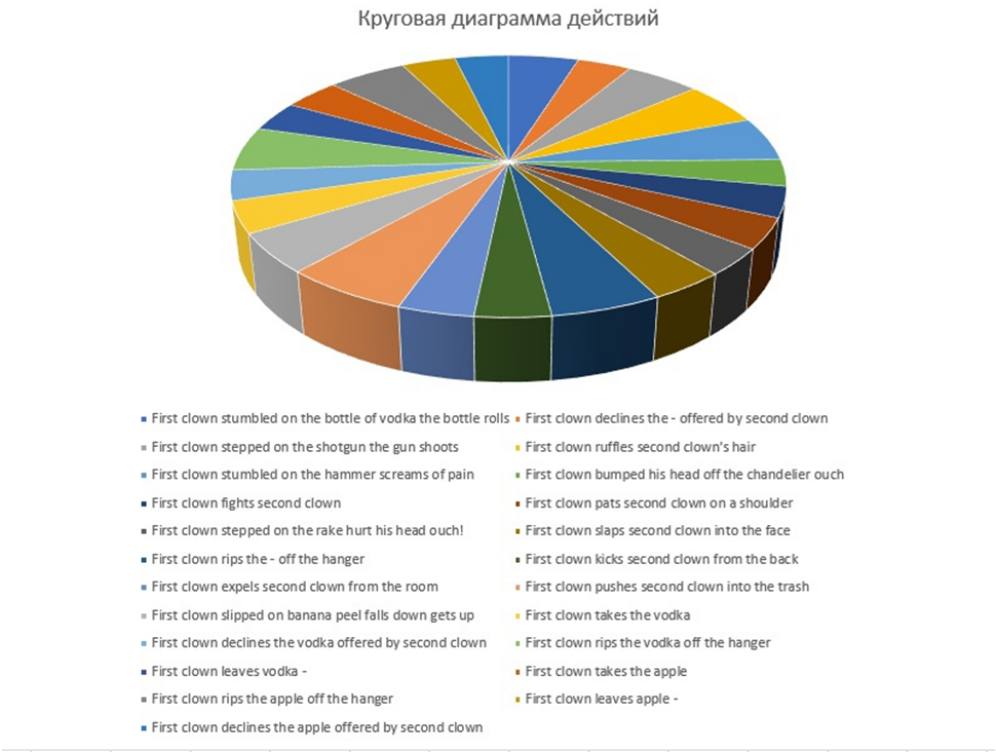

Рис. 3.2 – Диаграмма частот действий первого актора

Исходя из диаграммы, можно заметить, что большое количество действий не использовалось. Данная ситуация возникла в ходе тестирования взаимодействия роботов с использованием моральной схемы, описанной в формуле 2.1.

Существенным фактором, повлиявшим на работу моральной схемы, является оценка каждого действия отдельно по VAD, о чем говорися в работе [4]. В ходе корректировки действий, основанной на опросе фокус группы, были произведены корректировки оценок. Результат корректировок виден на (Рис. 3.2).

3.2 Сбор данных, метрики и инструменты

Для реализации сбора информации обычно используются следующие инструменты, что отображено в работае [10] :

- HTTP Client – клиент является библиотекой передачи, он находится на стороне клиента, отправляет и получает сообщения HTTP.
- Beautiful Soup - это пакет Python для анализа документов HTML и XML. Он создает дерево синтаксического анализа для проанализированных страниц, которое можно использовать для извлечения данных из HTML, что полезно для парсинга веб-страниц.

– Requests - это HTTP-библиотека для языка программирования Python. Цель проекта - сделать HTTP-запросы более простыми и удобными для человека.
– AsyncIO – модуль, предназначенный для упрощения использования корутин и футур в асинхронном коде — чтобы код выглядел как синхронный, без коллбэков.

Основной принцип парсинга ресурсов для сбора информации осуществлялся при помощи написания алгоритмов на языке программирования Python. Для этого использовалась библиотека ButifulSoup, которая была выбрана для использования для чтения HTML, а также библиотека Requests, которая является стандартным инструментом для составления HTTP-запросов в Python. Простой и аккуратный API значительно облегчает трудоемкий процесс создания запросов. Таким образом, можно сосредоточиться на взаимодействии со службами и использовании данных в приложении. Так же из за большого количества реквестов было рационально использовать корутину, используя библиотеку Asyncio.

Такие HTTP методы, как GET и POST, определяют, какие действия будут выполнены при создании HTTP запроса. GET является одним из самых популярных HTTP методов. Метод GET указывает на то, что происходит попытка извлечь данные из определенного ресурса. Для того, чтобы выполнить запрос GET, используется requests.get(). Положительным ответом сервера является код 200.

В метод requests.get() помещается URL, в которую может быть помещен JSON как текст запроса, либо при наличии API более упрощенный текстовый параметр, при наличии токена как идентификатора аккаунта – токен.

response.status_code, который равен 200, означает то что ответ от сервера получен положительный, текст запроса обработка, идентификация пользователя прошла успешна, возвращен ответ в формате JSON.

Иной же случай, когда нет доступа к API, оно платное либо его не существует. В таком случае скрипт будет выглядеть следующим образом.

В данном примере осуществляется request по URL, добавляется User-Agent, который имитирует пользователя, декодируются данные HTML в формат Unicode-8, который представляет собой распространённый стандарт кодирования символов, позволяющий более компактно хранить и передавать символы Юникода, используя переменное количество байт, и обеспечивающий полную обратную совместимость с 7-битной кодировкой ASCII.

Далее находим все классы таблиц и присваиваем переменным генератор, вытаскивающий из классов таблиц данные с соответствующим тегом.

С помощью BS4 можно найти в коде HTML все что требуется для создания нудного мас-

сива данных, далее для того, чтобы унифицировано хранить данные, следует использовать объектно-ориентированные форматы данных, такими бывают JSON и XML. JSON - текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми и в итоге был выбран как формат хранения данных в работе.

Для того чтобы спарсить сайт полностью и извлечь весь требуемый тип данных, используются рекурсивный подход. Для этого в работе используется пакет Python - html5lib, который реализует алгоритм парсинга HTML5, на который сильно влияют современные браузеры. Как только парсер получает нормализованную структуру содержимого, становится доступным поиск данных в любом дочернем элементе тега html. Искомые данные чаще всего находятся в теге “table”. После нахождения родительского тега, рекурсивно проходим по дочерним элементам. Для ссылок чаще всего используется тег “href”, далее из полученных ссылок извлекаем требуемый тип данных на сайте, в нашем случае это текст.

Так как на сайте слишком много текста и достаточно большой процент содержания лишнего текста по типу маркировок, контекстной рекламы и прочего, ставится задача определения выявления значимого текста на сайте.

Значимый текст выбирается на основании анализа его принадлежности к заранее определенным классам текста (или тематикам). Как правило, методы автоматической классификации основаны на методе машинного обучения: сначала получают обученную с помощью какого-либо алгоритма модель, качество которой определяет точность классификации. Таким образом, процесс обучения зависит от выбранного алгоритма и «чистоты» обучающей выборки, согласно работе о [15]. Одним из фреймворков, который используется для того чтобы обучать модель по заранее определенным классам – это lingvo, реализованный для .NET Framework, в котором используется Gradient Sign Dropout (GradDrop).

3.3 Инструменты для анализа текста

После сбора данных для их последующего использования в нейронной сети требуется осуществить разметку данных, для этого требуется выделить те слова определяющие какое-либо действие или предмет, которые будут семантически близки к заранее предопределенным типам действий или типам предметов.

Возможность идентификации семантической близости между словами сделала модель word2vec широко используемой в NLP-задачах, которые подробно описываются в [16]. Идея word2vec основана на контекстной близости слов. Каждое слово может быть представлено в виде вектора, близкие координаты векторов могут быть интерпретированы как близкие по смыслу слова [17].

Таким образом, извлечение семантических отношений (отношение синонимии, родови-

довые отношения и другие) может быть автоматизировано. Установление семантических отношений вручную считается трудоемкой и необъективной задачей, требующей большого количества времени и привлечения экспертов. Но среди ассоциативных слов, сформированных с использованием модели word2vec, встречаются слова, не представляющие никаких отношений с главным словом, для которого был представлен ассоциативный ряд [18].

В работе рассматриваются дополнительные критерии, которые могут быть применимы для решения данной проблемы. Наблюдения и проведенные эксперименты с общеизвестными характеристиками, такими как частота слов, позиция в ассоциативном ряду, могут быть использованы для улучшения результатов при работе с векторным представлением слов в части определения семантических отношений для русского языка.

Представление слов в виде векторов позволяет применять математические операции. В большинстве примеров можно встретить вычитание векторов, когда результат вычисления vec('Madrid') - vec('Spain') + vec('France') будет ближе к vec('Paris'), чем к другим векторам из распределения. Таким образом, разница векторов может быть использована для поиска семантических отношений между словами [19].

Word2vec не возвращает напрямую семантические отношения между словами. В ассоциативном ряду, который может быть возвращен в качестве близких слов к запрашиваемому (главному) слову, отражаются слова, которые часто употребляются рядом в контексте. Бесспорно, в ассоциативном ряду встречаются синонимы, антонимы, гипонимы, гиперонимы, холонимы, меронимы, ассоциации и другие типы, которые могут быть определены как семантические отношения.

3.4 Диаграмма классов

На (Рис. 3.3) изображена диаграмма классов (Построение такой диаграммы рекомендуется в работе [20]):

1. В Web Site Manager попадают ссылки на все сайты, каждому сайту указываются теги.
2. Далее Web Site Manager передает ссылки в Request Manager.
3. Request Manager асинхронно посылает запросы на сайт, чтобы получить весь контент сайта, по запросу он получает html-страницу и передает ее в Web Html Parser.
4. Web Html Parser по тегам парсит html так, что выделяет нужный текст, который передается в Web Text.
5. Context Text Analisator - использует консольное приложение по работе с LingvoNET, с которым класс общается посредством стандартного ввода/вывода с целью классификации текста на предмет семантической принадлежности.

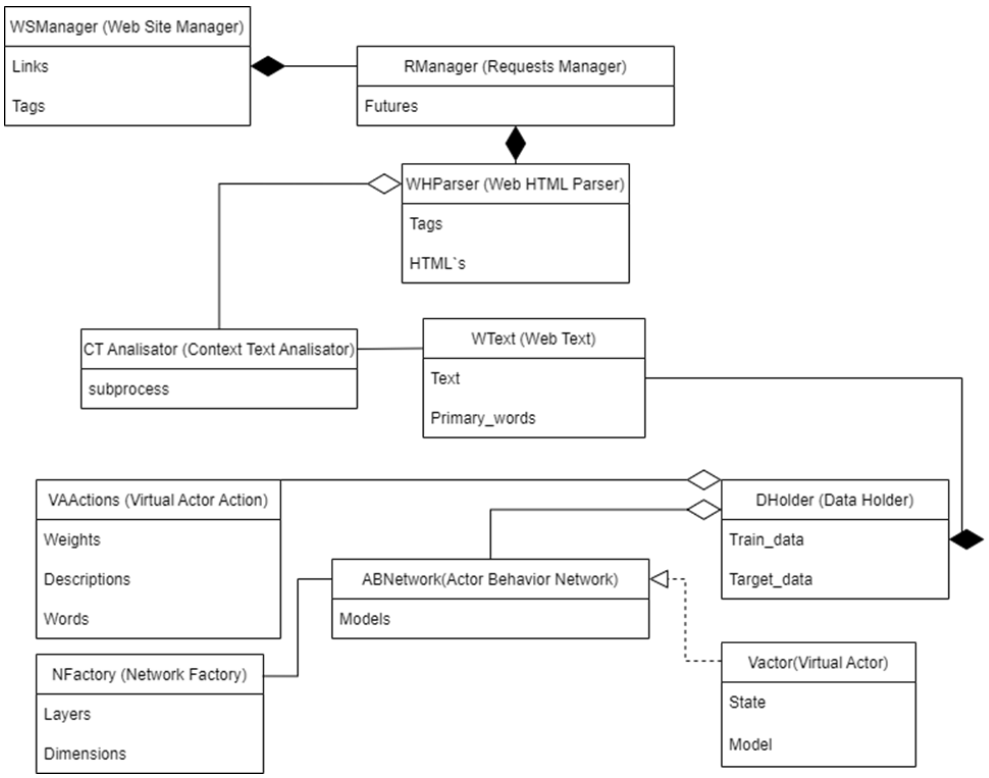

Рис. 3.3 – Связь между акторами и объектами

6. В Web Text выделяются ключевые слова при помощи Virtual Actor Action.
7. Virtual Actor Action берутся из excel-таблицы действий со значением весов и описаниями.
8. В Data Holder на основании Virtual Actor Action и самого текста генерируются сценарии (10 действий, 11 состояний), моделью пытаемся предсказать 11 состояние на основе 10 предыдущих.
9. Actor Behavior Network хранит все обученные модели, из которых потом выбирается оптимальная.
10. Network Factory – шаблон проектирования, создает по описанию нейронной сети создает экземпляр сети (Actor Behavior Network).
11. Одному Virtual Actor соответствует одна нейронная сеть, у Virtual Actor меняется текущее состояние на основе модели.

3.5 Выводы

В процессе работы были выделены часто используемые инструменты для сбора данных со сторонних веб-сервисов, а именно программные модули языка Python requests, Asyncio, ButifulSoup. Произведен анализ инструментария для решения задач семантической близости слов. Также была составлена диаграмма классов программного модуля. А для моделирования поведения виртуального агента составлена архитектура рекуррентной нейронной сети.

4. Реализация программного продукта

В этой главе описаны практические методы и их частичное представлении реализации, которые мы использовали для достижения поставленной цели.

4.1 Использование парсера данных

Так как для обучения нейронной сети статическим контентом, а именно: сайты с анекдотами и сайты со сценариями юмористических сценок. Для того чтобы собрать все эти данные потребуется решить задачу парсинга нужны большие массивы данных, возникает задача сбора данных относящихся к юмору(тории юмора). Источников таких данных не так много, основынми будут веб ресуры с юморивеб сайтов.

Поэтому опишем алгоритм решающий такую задачу. В качестве вводных возьмем url ссылки на веб ресурсы. Так как веб сайты могут иметь разную архитектуру возникает задача построения унификационного решения данного вопроса.

Как было описано ранее для полного парсинга страницы необходимо использовать рекурсивный метод, принцип которого подробно описан в parser02, который выражен в том, что мы проходим все ссылки на выбранном ресурсе. Основная функциональность заключается в следующих строчках :

```
class WSManager:
    def __init__(self, base_url):
        self.base_url = base_url
    def get_urls(self, level):
        urls = []
        try:
            links_1 = []
            start_link = self.base_url
            links_1.append(start_link)
            for i in links_1:
                response = requests.get(i)
                soup = BeautifulSoup(response.content, "html.parser",
parse_only=SoupStrainer(['a', 'img']))
                full_list = [link['href'] for link in soup if link.get('href')] +
[img['src'] for img in soup if img.get('src')]
                full_list = list(set(full_list))
                for url in full_list:
                    if not url.startswith('https:/'):
                        if url.startswith('/'):
                            if url.find('.org') == -1:
                                url = start_link + url[1:]
                                full_list.append(url)
                            elif url.find('.org'):
                                url = 'https:' + url
                                full_list.append(url)
                        elif url.startswith('//'):
                            url = start_link + url[2:]
                            full_list.append(url)
                        else:
                            pass
                    elif url.startswith('https:/'):
                        full_list.append(url)
                        urls.append(full_list)
                self.get_urls(level - 1)
                links_1 = full_list
                links_1 = list(set(links_1))
                return links_1
        except MemoryError as e:
            print(e)

        return urls
```

Рис. 4.1 – Класс WSManager - Основная функциональность рекурсивного парсинга

Имея ссылку на страницу, мы можем полуть html код страницы. По выше указанному коду найдем все ссылки на ресурсы данного веб сервиса предоставленные на этой странице. Осмотрим все полученные ссылки таким же образом. В итоге мы получим html коды практически всех страницы доступных на принятом в рассмотрение ресурсе.

URL ссылки на веб сервисы предоставляющие, данные о юмористических текстах, агрегируются в экземплярах класса WSManager (Web Site Manager). Далее каждый экземпляр класса создает и запускает экземпляры класса RManager (Request Manager). Которые занимаются асинхронным получением данных с сайтов.

Для классического получения данных при наличии API, код будет выглядеть следующим образом:

```
url = self.base_url
data = {"content": message}
header = {"authorization": token}
r = requests.post(url, data=data, headers=header)
```

Рис. 4.2 – Получения данных при наличии API

При получении положительного ответа 200, запрос отработал успешно. В данной работе класс RManager использует корутины, что позволяет существенно сэкономить время ожидания всех ответов веб сервисов. Указанная экономия целиком и полностью объясняется тем, что пока класс ожидает ответ на первые запросы, он не перестает слать последующие. А те запросы, ответ на которые уже был получен, передаются в экземпляры класса WHParser так, что на каждую полученную html страницу создается один экземпляр такого класса.

```
def t(self):
        if self.token is None or self.expiration is None or self.expiration <
datetime.now():
            loop = asyncio.get_event_loop()
            loop.run_until_complete(self. __get_smth_for_smth__())
        if self.expiration > datetime.now():
            return self.token

async def __get_smth_for_smth__(self):
        loop = asyncio.get_event_loop()
        future = loop.run_in_executor(None, functools.partial(requests
self.base_url))
        resp = await future
```

Рис. 4.3 – функция получения данных HTML класса WHParser

Предположим, что на сайте еще стоит ограничение по запросам, поэтому предварительно, мы имеем лист проверенных и подготовленных рабочих прокси. При этом статусы готовности, которых получены в отдельных скриптах языка python3 с использованием корутин из программного модуля asyncio.

```
new_proxies = [i for i in [proxies_list[i:i+2] for i in
range(0,len(proxies_list),2)]]
       proxies = [{'IP': new_proxies[i][0], 'Port': new_proxies[i][1], 'Status':
'unknown'} for i in range(len(new_proxies))]
```

Рис. 4.4 – Обработка proxy

{'IP': '138.201.120.214', 'Port': '1080', 'Status': 'Alive'}
{'IP': '51.222.40.26', 'Port': '9090', 'Status': 'Alive'}
{'IP': '92.42.109.189', 'Port': '1080', 'Status': 'Alive'}
{'IP': '54.37.160.92', 'Port': '1080', 'Status': 'Alive'}
{'IP': '20.105.253.176', 'Port': '8080', 'Status': 'Alive'}
{'IP': '103.73.194.2', 'Port': '80', 'Status': 'Alive'}
{'IP': '54.37.160.90', 'Port': '1080', 'Status': 'Alive'}
{'IP': '134.119.206.106', 'Port': '1080', 'Status': 'Alive'}
{'IP': '176.241.89.244', 'Port': '53583', 'Status': 'Alive'}

Рис. 4.5 – Проверка статуса прокси серверов

Далее помещаем наши прокси в ассинхронные запросы так, что при неудовлетворительном ответе с сервера, класс RManager меняет используемую прокси.

```
some_req = Request('https://www.example/.org/', proxies =
proxies[random.randint(len(proxies))], timeout=5,verify=False)
some_req.add_header('User-Agent', ua.random)
some_doc = urlopen(some_req).read().decode('utf8')
```

Рис. 4.6 – пример работы RManager

WHParser, получив код веб страницы, обрабатывает его с помощью модуля BaeautifulSoup. Выделяет текста на основании тегов, определенных экспертами для каждого веб сервиса. soup = BeautifulSoup(some_doc, 'html.parser')

Для вытаскивания данных из таблиц используем обращения к классам HTML. Атрибут class указывает одно или несколько классов для элемента. Атрибут class используется в основном для того, чтобы указывать на класс в таблице стилей. Однако он также может использоваться JavaScript (через HTML DOM) для внесения изменений в элементы HTML с указанным классом. Так же надо учесть то, что текстовые элементы традиционно заключают в элементы с теговым названием “p”, но полный семантически связанный текст может быть разбит множества предложений, таким образом, что отдельные элементы одного и того же уровня вложенности будут представлять эти разбитые множества. В связи с этим возникает задача восстановления семантически связанного текста из элементов одинаковой вложенности и одинакового типа, что следует учитывать при дальнейшей разработке.

```
for table in soup.find_all('table'):
        try:
            if 'table-striped' in table['class']:
                parsed = [[i] for i in [i.text for i in table.find_all('p')]]
        except KeyError:
            pass
```

Рис. 4.7 – Обращение к атрибутам HTML

Получаем данные в формате:

```
for table in soup.find_all('table'):
        try:
            if 'table-striped' in table['class']:
                parsed = [[i] for i in [i.text for i in table.find_all('p')]]
        except KeyError:
            pass
```

Рис. 4.8 – Полученные данные

4.2 Анализ и отбор полученных данных

После получения текстовых данных с выбранных ресурсов, требуется провести просеивание массива на данные, излишне затронутые в процессе парсинга, так как на сайте слишком много текста и достаточно большой процент содержания лишнего текста по типу маркировок, контекстной рекламы и прочего, а наша задача получить интересующие нас юмористические текстовые данные в совокупности html страниц.

Значимый текст выбирается на основании анализа его принадлежности к заранее определенным классам текста (или тематикам). Как правило, методы автоматической классификации основаны на методе машинного обучения: сначала получают обученную с помощью какого-либо алгоритма модель, качество которой определяет точность классификации. Таким образом, процесс обучения зависит от выбранного алгоритма и «чистоты» обучающей выборки. Для решения такой проблемы выделим тэги и классы элементов, содержащих юмористические данные. Чтобы понять какие данные нам подходят, а какие нет.

Следует учесть, что большое количество классов (десятки и сотни) приводит к увеличению трудоемкости обучения и понижению точности классификации, которая описана в [21]. Тематики в юморе, близки по своей сути (например, экономика и бизнес), что приводит к тому, что классы в обучающей модели начинают пересекаться, приводя к снижению точности.

В таких случаях, классификация проходит по принципу объединения классов в один, а затем используют под классификацию или повторную классификацию документов внутри класса.

В данной системе автоматической классификации используется популярный метод опорных векторов (или SVM – Support Vector Machine) с мерой TFiDF [22]. Написано консольное приложение на языке программирования C#, целью которого ставится использование библиотеки dll - LingvoNET, которая представляет собой модель специально обученную на нескольких классах, определенных заранее, принцип проектирования описан в [**neural09**]. Для использования данного ресурса в проекте реализуется класс написанные на языке программирования Python, который взаимодействует с консольным приложением через стандартный вход и стандартный выход.

Классы содержащиеся в библиотеке LingvoNET – представлены ниже:

- Экономика и бизнес
- Шоу-бизнес и развлечения
- Семья
- Мода
- Компьютерные игры
- Здоровье и медицина
- Политика
- Наука и технологи
- Спорт
- Туризм, путешествия
- Недвижимость
- Авто

Согласно этим классам, происходит классификация каждого входящего текста с учетом его меры близости к тому или иному классу, так же как и в юморе чаще всего есть семантическая принадлежность к тому или иному классу представленному выше, однако такие классы как “Игры”, “Спорт”, “Мода”, “Авто” и “Недвижимость”, чаще всего относятся к контекстной рекламе. Если документ близок к двум тематикам, то он попадает в соответствующие два класса. Если текст похож сразу на несколько тематик, то, скорее всего, это шум и не семантически не подходит.

Качество классификации чаще всего оценивается по двум критериям: точностью и полнотой классификации согласно работе [23].

Точность - показывает, насколько точно тексты с ресурсов попадают в определенный класс.

Полнота - определяется соотношением текстов, релевантных данному классу, к общему количеству релевантных текстов. Точность можно повышать, задавая порог прохода текста в тот

или иной класс, при этом полнота классификации будет уменьшаться. Как правило, стараются найти оптимальное соотношение этих критериев.

Представим пример полученного текста с ресурса:

– Мам, помоги с сочинением. Нужно не меньше 350 слов.
– А что за тема?
– Некрасов: ”Кому на Руси жить хорошо”.
– Напиши — ”депутатам”.
– Но надо 350 слов.
– Пиши поименно!

В данном случае получим следующие результаты анализа представленных выше данных:

Таблица 4.1 – Результат семантического анализа подходящего текста

Семья	46.81%
Политика	19.17%
Другие	менее 10%

Текст будет отнесен к классу “Семья” и семантически подходит нам в качестве искомого вида юмористического текста.

Иной же случай, это пример контекстной рекламы: “Горячие предложения контрафактного алкоголя на каждой подворотне! Тема, которая волнует репортерское агентство в этом городе. Бодяжная сивуха убивает людей, о чем думает администрация???”

Таблица 4.2 – Результат семантического анализа не подходящего текста

Недвижимость	14.18%
Здоровье и медицина	12.25%
Политика	11.55%
Семья	10.91%
Экономика и бизнес	10.45%
Шоу-бизнес и развлечения	10.02%
Другие	менее 10%

В данном случае текст будет определяться как шум, так как не проходит ни один порог принадлежности к тому или иному заранее заданному классу. В процессе анализа текста по принадлежности, одновременно запоминаются пути на html странице, а именно, названия атрибутов к которым относится текст на странице html, если в процессе анализа были найдены тексты с одинаковыми путями и при этом все они проходят порог классификации как юмористические, то текстовые данные под такими путями и атрибутами считаются юмористическими и подлежат дальнейшей работе без семантического анализа. Таким образом, оптимизирован поиск требуемых данных на ресурсах и с помощью выбора вышеперечисленных инструментов определяется востребованность данных и отсеивается лишний, для данного проекта, контент.

В результате получаем следующий набор данных, представленный на (Рис. 4.9):

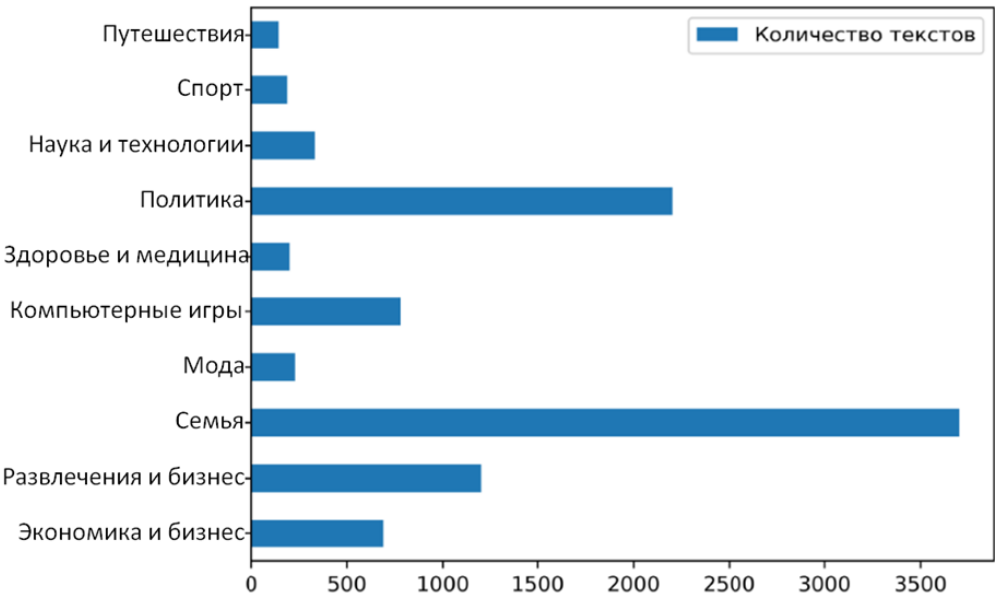

Рис. 4.9 – Диаграмма распределения по количеству, собранных текстов с сайтов

4.3 Разметка отобранных данных

Для реализации моделей из третьего раздела, используются такие пакеты языка программирования python как: ufal.udpipe и wget. Как было сказано ранее для нахождения близости слов мы используем готовую модель, которую взяли с ресурса https://rusvectores.org/static/models с помощью пакета wget, принцип работы описан в [24]. Данный пакет позволяет скачивать данные с веб ресурсов посредством GET запроса.

Для того чтобы модель word2vec, описанная в [25] могла определить расстояние между словами, ей следует передать слово, ставится задача определить его часть речи, для того, чтобы автоматизировать такой процесс по определению части речи слова из текста, мы должны с помощью системы word2vec понять по слову какими категориями частей речи оно обладает,

самые распространённые варианты – это существительные и глагол, далее после определения, мы передаем их на анализ дистанции. Принцип работы модели представлен на (Рис. 4.10):

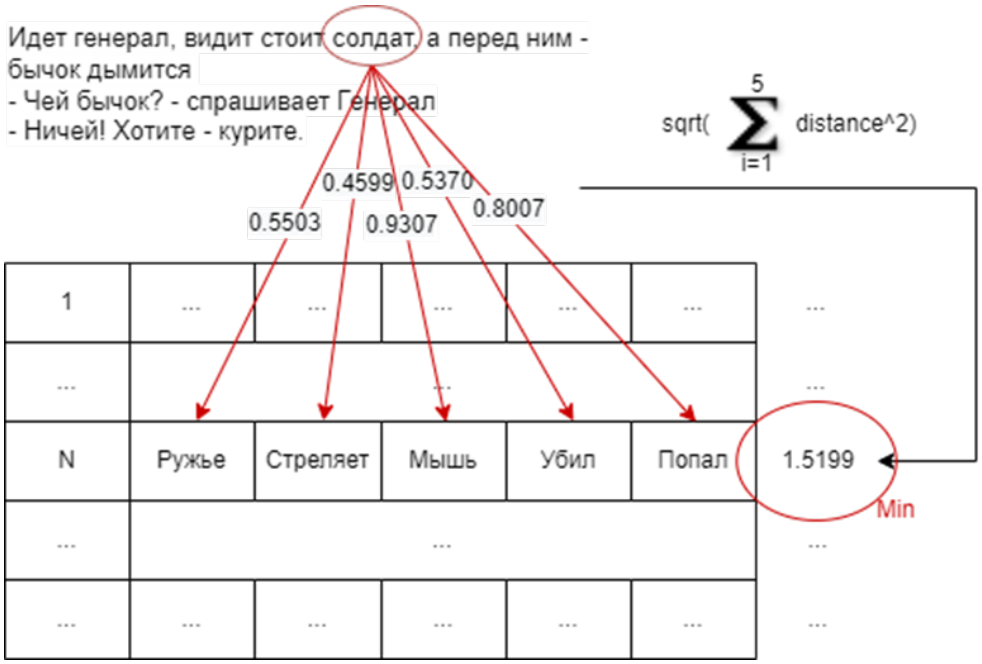

Рис. 4.10 – принцип работы word2vec

В рамках данной работы используется модель ruwikiruscorpora_tokens_elmo_1024_2019 - модель UDPipe для нахождения синонимов. Данная модель позволяет узнать семантическую близость слов.

Чтобы начать применять указанную выше нейронную сеть непосредственно, следует подготовить действия виртуальных акторов так, чтобы ими могла оперировать частично или полностью сама нейронная сеть. Для этого возьмем разобьем описание каждого действия на ключевые слова так, что каждое действие будет ассоциировано с набором слов. При этом создан класс отражающий действие актора как сущность - VAAction(Virtual Actor Action), принцип которой описан в [26].

Данный класс содержит в себе описание действия и ключевые слова, описывающие данное действие. Среди таких слов отсутствуют предлоги, а сами слова представлены в нормальной форме (для существительных - именительный падеж единственное число).

Для построения сценариев берутся текста, полученные в разделе выше. Для каждого текста выделяются ключевые слова, которые наиболее близки к ключевым словам, описывающим действия акторов. Что происходит в процессе итерации через все слова текста так, что для каждого слова применяется считается значение близости слова к действию виртуального актора. Для работы с текстом создан класс WText (Web Text), который является ответственным за итерацию через все слова текста. С помощью методов класса задается функция, которая будет приме-

няться к каждому слову. В качестве такой функции берется функция, которая находит близость слова с ключевыми словами экземпляра класса VAAction.

Также класс WText формирует набор определяющих текст слов, эти слова выбираются так, что значение семантической близости больше, чем заданный заранее порог, данной работе порог равен 0.08. После того как все тексты были переведены в экземпляры класса WText, была произведена оценка кол-ва текстов с одинаковым кол-вом определяющих слов. Такое распределение приведено на (Рис. 4.11):

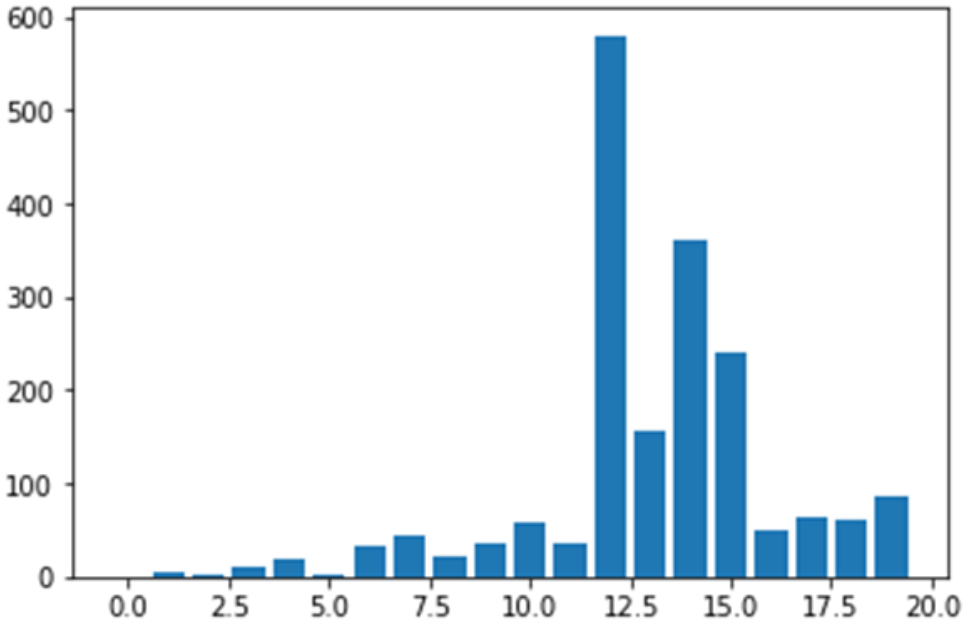

Рис. 4.11 – Распределение количества ключевых слов на текст

Исходя из картинки видно, что подавляющее большинство текстов имеют более 10 определяющих слов. Каждому определяющему слову ставится в соответствие экземпляр класса VAAction, таким образом из текстов набираются сценарии из 10 последовательных действий. Чтобы было удобнее формировать данные для обучения модели, определяющей поведение виртуальных акторов, реализуется метод получения последовательности состояний виртуальных агентов так, что переход в каждое последующее состояние.

4.4 Реализация модели поведения виртуального актора

По полученным размеченным данным на основании семантической близости слов выделенных и сопоставленных с заранее описанными действиями для акторов были получены следующие результаты. Суть первого эксперимента, состояла в предсказании 50 действий производимых акторами, результаты эксперимента представлены на (Рис. 4.12), (Рис. 4.13):

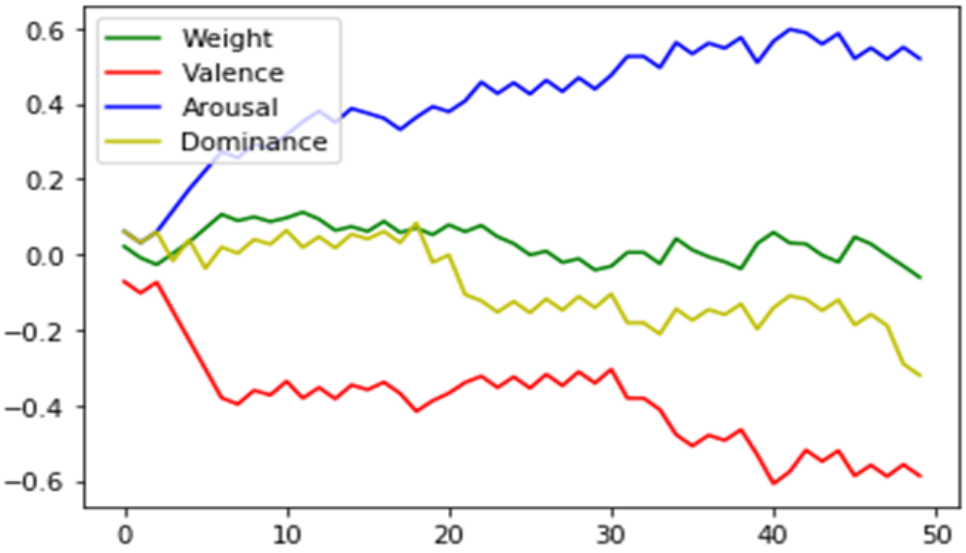

Рис. 4.12 – Оценки виртуального актора1

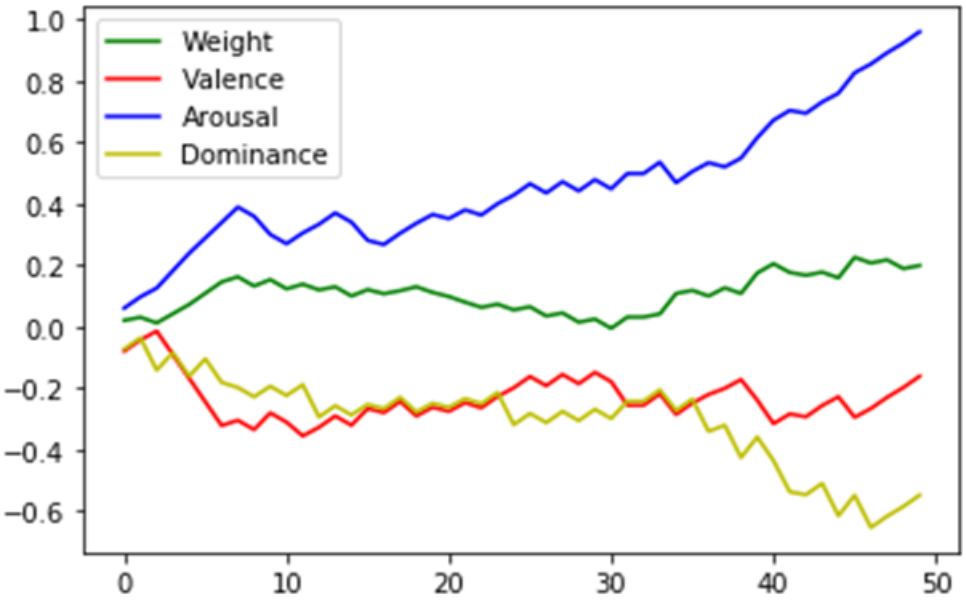

Рис. 4.13 – Оценки виртуального актора2

В следующем эксперименте было проведено 10000 действий и была построена гистограмма распределения частоты действий выполняемых первым виртуальным актором., результаты представлены на (Рис. 4.14):

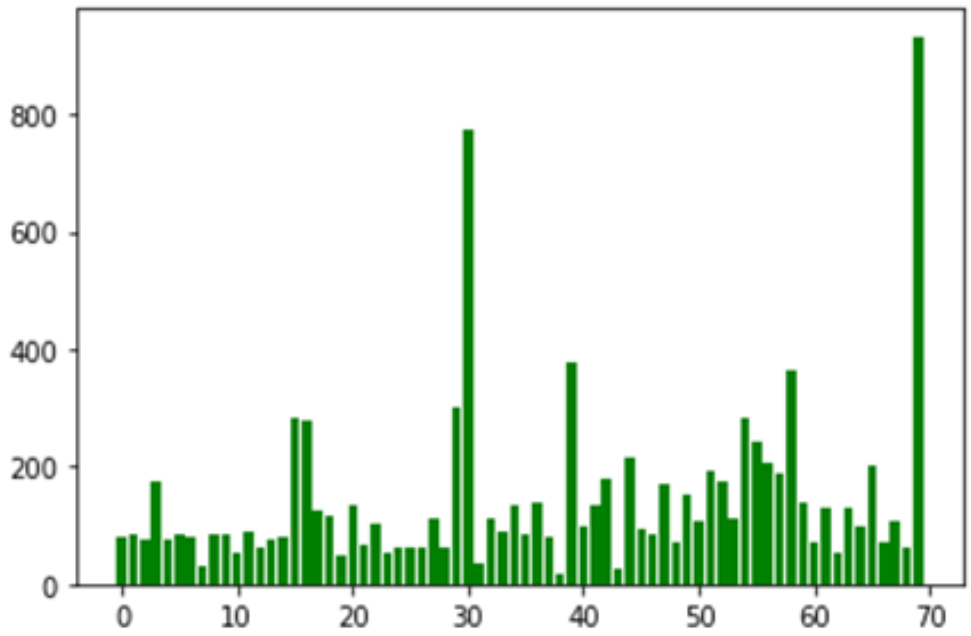

Рис. 4.14 – Частоты действий акторов

4.5 Выводы

В данном разделе осуществляется описание реализации получения и разметки данных с ресурсов содержащих юмористические сюжеты. А так же последующим обучением модели, генерирующей поведение виртуального агента.

Была разработано программное обеспечение, позволяющее определять семантическую принадлежность текста. А так же последующее определение семантической близости слов с заранее описанными действиями для виртуальных акторов. Группы действий которых впоследствии используются для обучения нейронной сети для последующей генерации юмористических сюжетов.

Заключение

В рамках представленной работы были изучены проблемы парсинга сайтов и классические методы их решения, были выделены основные нейросетевые подходы для анализа текстов, выделены основные проблемы связанные с семантитеческой классификацией, а так же изучены библиотеки и плагины, которые применяются для реализации данных подходов. Был применен метод опорных векторов (или SVM – Support Vector Machine) для определения семантитеческой принадлежности текста к юмористическимо сюжету, а так же, оптимизирован алгоритм поиска требуемых данных на юмористических ресурсах.

По заврешению получения юмористических данных в процессе парсинга, были получены данные, которые в последствии были размечены по принципу определения близости слов с заранее описанными действиями для виртуальных акторов.

Размеченные данные были использованы в последствии для обучения модели задача которой генерировать юмористические сюжеты, что подтверждается результатами обучения.

1. Были определены подходы, согласно котором разрабатывалась альтернативная линия сюжетов и их воплощения Виртуальным Актором.
2. Был осуществлен сбор и осуществлена разметкуа данных для обучения нейронной сети;
3. Был осуществлен глубокий интеллектуальный анализ данных по размеченным массивам данных.
4. Был разработан, алгоритм передающей результаты анализа виртуальному Агенту.
5. Было Реализован визуальный агент и сцена, используя межплатформенную среду разработки компьютерных игр Unity3d.

Список литературы

1. *V. S. A.* Comparative analysis of implemented cognitive architectures //Biologically Inspired Cognitive Architectures 2011. – IOS Press. — 2011.
2. *V. S. A.* Emotional biologically inspired cognitive architecture //Biologically Inspired Cognitive Architectures. — 2013.
3. *P. I. E.* Emotions and feelings. — 2007.
4. *V. S. A.* Socially emotional brain-inspired cognitive architecture framework for artificial intelligence //Cognitive Systems Research. — 2020.
5. *Langley P. Laird J. E. R. S.* Cognitive architectures: Research issues and challenges //Cognitive. — 2009.
6. *V. S. A.* Emotional biologically inspired cognitive architecture //Biologically Inspired Cognitive. — 2013.
7. *Bai S. Kolter J.Z. K. V.* An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. — 2018.
8. *ресурс] W. [* Рекуррентная нейронная сеть. — 2020.
9. *И.А. Батраева А.Д. Нарцев А. Л.* ИСПОЛЬЗОВАНИЕ АНАЛИЗА СЕМАНТИЧЕСКОЙ БЛИЗОСТИ СЛОВ ПРИ РЕШЕНИИ ЗАДАЧИ ОПРЕДЕЛЕНИЯ ЖАНРОВОЙ ПРИНАДЛЕЖНОСТИ ТЕКСТОВ МЕТОДАМИ ГЛУБОКОГО ОБУЧЕНИЯ. — 2020.
10. *Standard L.* HTML Standard - whatwg. —.
11. *Wrobel L.* Simple mathematical expressions parser. —.
12. *Mikolov T.* Efficient Estimation of Word Representations in Vector Space. — 2013.
13. *Kuznetsov S. O.* Fitting Pattern Structures to Knowledge Discovery in Big Data. ICFCA. — M., 2013.
14. *V. S. A.* Goal reasoning as a general form of metacognition in BICA //Biologically Inspired Cognitive Architectures. — 2014.
15. *П. Ф.* Машинное обучение. Наука и искусство построения алгоритмов, который извлекают знания из данных. — 2015.
16. *Y. K.* Convolutional neural networks for sentence classification // Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP 2014). — 2014.

17. *Селезнев К.*, *Владимиров А.* Лингвистика и обработка текстов // Открытые системы. — М., 2013.

18. *Mirkin B.* Core Concepts in Data Analysis: Summarization, Correlation and Visualisation, DOI. — 2011.

19. *Manning C.*, *Schuetze H.* Foundations of Statistical Natural Processing. MIT. — М., 1999.

20. *Weisfeld. M.* The Object-Oriented Thought Process. — Fourth Edition. — Addison-Wesley Professional. — 2013.

21. *Константин Симаков И. К.* Особенности очистки адресных данных // Открытые системы. СУБД. — 2013.

22. *Ильвовский Д.*, *Черняк Е.* Системы автоматической обработки текстов // Открытые системы. СУБД. — М., 2014. — URL: https://www.osp.ru/os/2014/01/13039687.

23. *Н. К. В.* Элементы теории ассоциативной семантики // Управление большими системами. — 2012.

24. *М. Л. Ю.* Люди и знаки. — 2010.

25. *MyStem. Я. технология.* MyStem. —. — URL: https://tech.yandex.ru/mystem.

26. *Xin. R.* Word2vec parameter learning explained. 2014. arXiv preprint arXiv: 1411.2738. — 2010.