**НИТУ «МИСИС»**

**Кафедра инженерной кибернетики**

**ОТЧЕТ**

по

**ЛАБОРАТОРНЫМ РАБОТАМ №1 и №2**

*«Изучение технологий разработки чат-ботов на основе программного комплекса (фреймворка) для создания диалоговых систем RASA»*

учебная дисциплина «Методы искусственного интеллекта»

**Учащийся:**

Меремьянина П.С., группа БПМ-20-4

**Преподаватель: Дранга Д.**

**Оценка:**

**Дата защиты:**

**2023 г.**

1. Задание на лабораторные работы
   1. Цель работы

Целью лабораторных работ является:

* освоение технологий создания чат-ботов (диалоговых систем) и фреймворка Rasa на основе изучения и выполнения заранее подготовленных примеров использования Rasa;
* формирование предложений на доработку и непосредственное улучшение; (совершенствование, развитие) существующей конкретной диалоговой системы (чат-бота).
  1. Задание на лабораторную работу 1

В рамках лабораторной работы 1 необходимо:

1. выбрать одного из тестовых чат-ботов, основанных на фреймворке RASA, из предлагаемого списка на странице <https://github.com/Raiffeisen-DGTL/ds-chatbot-test-task-description>;
2. выполнить программную реализацию заданного варианта чат-бота;
3. провести анализ созданного чат-бота;
4. определить направления и/или способы его доработки.
   1. Задание на лабораторную работу 2

В рамках лабораторной работы 2 необходимо:

1) выполнить программную реализацию предложенных доработок для чат-бота, изученного в лабораторной работе №1;

2) подготовить общий отчет по обеим лабораторным работам.

1. Лабораторная работа №1
   1. Выбор чат-бота

Для изучения и доработки был выбран чат-бот базы знаний «knowledgebasebot» из набора примеров фреймворка Rasa:

<https://github.com/RasaHQ/rasa-examples/tree/main/knowledgebasebot>.

* 1. Программная реализация

Для изучения чат-бота был установлен фреймворк Rasa версии 3.1.

Так как выбранный чат-бот был реализован на более ранних версиях фреймворка Rasa, то была выполнена миграция для домена:

rasa data migrate -d domain.yml --out domain1.yml

После обучения модели, чат бот был запущен в режиме командной строки. На рисунке 1 представлен пример диалога с изучаемым чат-ботом.

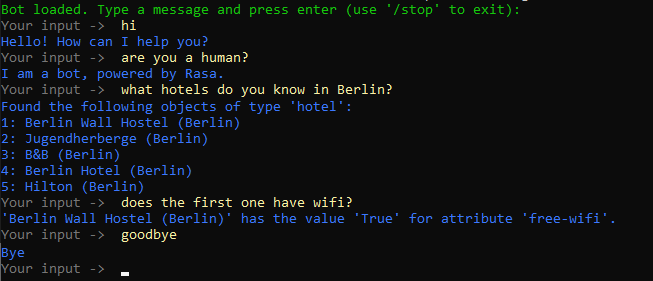


Рисунок 1 – Пример диалога с изучаемым чат-ботом

Как видно из рисунка 1, при запросе к базе знаний бот выводит до пяти элементов (гостиниц) из базы знаний, соответствующих запросу пользователя. При уточняющем запросе пользователь может получить более детальную информацию (например, наличие WiFi).

* 1. Анализ чат-бота

Диалоги с ботом реализованы на английском языке.

Основной задачей бота является предоставление информации о ресторанах и гостиницах. Данные о ресторанах и гостиницах бот получает из базы знаний. База знаний бота определена в файле «knowledge\_base\_data.json» формата JSON.

Домен бота определен в файле «**domain.yml**».

В домене:

* список намерений (**intents**):

|  |  |
| --- | --- |
| Наименование | Описание |
| bot\_challenge | Намерение для обращения к боту |
| Greet | Намерение приветствия |
| goodbye | Намерение прощание |
| query\_knowledge\_base | Намерение-запрос к базе знаний |

* список сущностей (**entities**):

|  |  |
| --- | --- |
| Наименование | Описание |
| object\_type | Тип объекта |
| mention | Ссылка на ранее упомянутый объект |
| attribute | Атрибут |
| hotel | Гостиница, отель |
| restaurant | Ресторан |
| cuisine | Кухня |
| city | Город |

* список слотов (**slots**):

|  |  |
| --- | --- |
| Наименование | Описание |
| object\_type | Тип объекта |
| mention | Ссылка на ранее упомянутый объект |
| attribute | Атрибут |
| hotel | Гостиница, отель |
| restaurant | Ресторан |
| cuisine | Кухня |
| city | Город |

Все слоты сопоставляются с одноименными сущностями и заполняются из сущностей. У всех слотов определен параметр «influence\_conversation: false» для отключения влияния слота на разговор.

* в списке действий (**actions**) только одно действие:   
  action\_query\_knowledge\_base – это встроенное действие фреймворка, обеспечивающее доступ к заданной базе знаний. В составе фреймворка это действие реализовано с помощью класса ActionQueryKnowledgeBase. Для настройки параметров этого действия создается класс-наследник. Его реализация приведена в файле «**actions.py**». В конструкторе указывается файл базы знаний и переопределяется формат вывода для объекта «hotel».
* список ответов (**responses**):

|  |  |
| --- | --- |
| Наименование | Описание |
| utter\_greet | Ответ на приветствие |
| utter\_goodbye | Ответ на прощание |
| utter\_ask\_rephrase | Ответ с просьбой перефразировать сообщение (когда бот не понял) |
| utter\_iamabot | Ответ: «Я бот» |

* в разделе «**session\_config**» определены параметры сессии.

Конфигурация бота определена в файле «**config.yml**»:

* **language** – определяет поддерживаемый язык;
* в разделе **pipeline** – определяются компоненты конвейера и параметры этих компонентов;
* в разделе **policies** – определяются политики;
* assistant\_id – идентификатор чат-бота.

В файле «**endpoints.yml**» указаны параметры подключения к серверу действий.

В папке «**data**» располагаются данные для обучения.

В файле «**nlu.yml**» для каждого намерения (**intent**) определяется набор обучающих сообщений. Это сообщения, с которыми, возможно, пользователь будет обращаться к боту. В сообщениях выполнена маркировка сущностей (**entity**). Это необходимо для обучения бота находить и выделять сущности.

В файл «**rules.yml**» перечислен список правил. Правила определяют следующие шаги (действия, **action**) бота после выделения определенного намерения (**intent**).

В файле «**stories.yml**» перечислены истории (**stories**). Истории определяют возможное развитие (ход) диалога. Истории пишутся в терминах намерений (**intent**), сущностей(**entity**) и действий (**action**).

* 1. Предложения по доработке чат-бота

На примере изученного чат-бота предлагаю создать чат-бота для выбора гостиницы.

Чат-бот на основе базы знаний предлагает варианты гостиниц в Москве и Санкт-Петербурге. Гостиницы можно выбрать по городу, по рейтингу (количеству звезд), по близости к станции метро.

Пользователь может запросить у бота более подробную информацию об одной из предложенных ему гостиниц:

- адрес;

- телефон;

- ближайшая станция метро;

- рейтинг;

- наличие завтрака, включенного в стоимость.

Язык диалога с чат-ботом – русский.

1. Лабораторная работа №2
   1. Доработки чат-бота

Для поддержки диалогов на русском языке в конфигурации бота в качестве языка выбрали русский (language: ru).

Все сообщения, как обучающие, так и ответные приведены на русском языке.

Для того, чтобы убрать текс на английском языке из ответов бота, сформированных из базы знаний, были переопределены функции «*utter\_objects*» и «*utter\_attribute\_value*» класса «*ActionQueryKnowledgeBase*» (см. файл «actions.py»).

В файле базы знаний был определен требуемый набор полей (атрибутов). База знаний заполнена подборкой гостиниц Москвы и Санкт-Петербурга.

Для обеспечения требуемой функциональности были определены следующие сущности (entities):

|  |  |
| --- | --- |
| Наименование | Описание |
| object\_type | Тип объекта |
| mention | Ссылка на ранее упомянутый объект |
| attribute | Атрибут |
| hotel | Гостиница, отель |
| city | Город |
| rating | Рейтинг (количество звезд) |
| metro | Ближайшая станция метро |

Также в домен включены одноименные слоты (slots). Все слоты сопоставляются с одноименными сущностями и заполняются из сущностей.

В обучающие данные были добавлены синонимы. Синонимы нормализуют обучающие данные, сопоставляя извлеченный объект со значением, отличным от извлеченного буквального текста. Примером синонимов является слова «гостиница» и «отель». Эти слова определяют одну и туже сущность в контексте диалога пользователя с ботом. Синонимы определены в файле «nlu.yml».

Намерения (intents), действия (action), истории (stories), правила (rules), конвейер (pipeline) и политики (policies) не изменились.

После выполненных изменений была переобучена модель.

* 1. Диалоги с чат-ботом

Рассмотрим примеры работы чат-бота после выполненных доработок. На рисунке 2 приведен пример диалога.

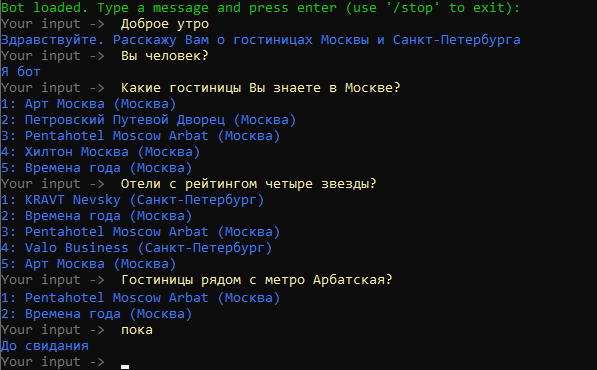


Рисунок 2 – Варианты гостиниц с отбором по городу, по рейтингу,   
по близости к станции метро

Бот по запросам от пользователя предлагает варианты для выбора гостиницы в указанном городе или по заданному рейтингу (количеству звезд), или по близости к станции метро.

Выделяя в намерениях сущность «mention» бот может ссылаться на одну из предложенных ранее в диалоге гостиниц. При этом, выделяя в намерениях сущность «attribute», бот предоставляет запрошенные параметры выбранной гостиницы. Пример такого диалога представлен на рисунке 3.

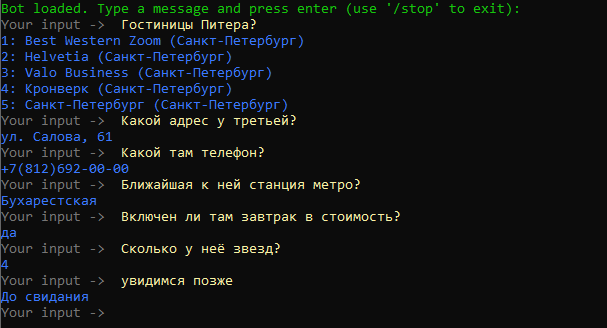


Рисунок 3 – Диалог с получением параметров выбранной гостиницы

В данном диалоге из вопроса «Какой адрес у третьей?» выделяются: сущность «mention» со значением «3» и сущность «attribute» со значением «address». После выделения этих сущностей Rasa заполняет одноименные слоты полученными значениями и выполняет действие «action\_query\_knowledge\_base». Это действие из базы знаний выбирает адрес третьей гостиницы из списка предложенных. Таким образом отрабатывает механизм выбора конкретной гостиницы из списка предложенных.

Следующие вопросы в этом диалоге «Какой там телефон?», «Ближайшая к ней станция метро?», «Включен ли там завтрак в стоимость?» и «Сколько у неё звезд?» также используют механизм ссылок, но ссылка происходит на уже выбранную гостиницу. При этом, в зависимости от вопроса, из базы знаний возвращаются запрошенные параметры выбранной гостиницы.

1. Отчетные материалы

Исходный код доработанного чат-бота по выбору гостиницы размещен по адресу:

<https://github.com/AnilopaM/selecthotel.git>

Там же в папке «doc» размещены: задание на доработку бота, данный отчет и презентация.

**Выводы**

В результате выполнения данных лабораторных работ были изучены возможности фреймворка Rasa.

Были получены практические навыки по разработке чат-ботов на базе фреймворка Rasa.

По результатам опробования доработанный чат-бот обеспечивает выполнение требований задания на доработку.