Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа № 3

По дисциплине ЕЯИИС

Тема: «Разработка системы прямого машинного перевода документов»

Выполнил:

Студент 4 курса

Группы ИИ-16 (2)

Пешко А.С.

Проверил:

Скарубо А.О.

Брест, 2021

**Цель работы:** освоить на практике основные принципы машинного перевода документов.

**Ход работы**

Сначала мы импортировали базу данных, предоставленную к лаботраторной работе (файл PEREV.MDB) в свою локальную бд.

**Код программы:**

public class ConstantUtils {

public static final String TRANSLATION\_ERROR = "Translation error";

public static final String WORD\_NOT\_FOUND\_REGEX = "Word %s not found in the dictionary";

public static String toRegex(final String text){

return text

.replaceAll("x1", ".\*")

.replaceAll("x2", ".\*")

.replaceAll("x3", ".\*")

.replaceAll("y1", ".\*")

.replaceAll("y2", ".\*")

.replaceAll("y3", ".\*");

}

public static String removePunctuation(final String text){

return text

.replaceAll("[.,/;!?]", "");

}

public static boolean isStringMock(final String word) {

return word.equals("x1")

|| word.equals("x2")

|| word.equals("x3")

|| word.equals("y1")

|| word.equals("y2")

|| word.equals("y3");

}

}

@Entity

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

@Table(name = "dict")

public class Dictionary {

@Id

@Column(name = "id")

private Integer id;

@Column(name = "russian\_word")

private String russianWord;

@Column(name = "english\_word")

private String englishWord;

}

@Service

public class Translator {

private final DictionaryRepository dictionaryRepository;

private final FunctionRepository functionRepository;

private final FunctionWordRepository functionWordRepository;

private final PositionRepository positionRepository;

public Translator(DictionaryRepository dictionaryRepository, FunctionRepository functionRepository, FunctionWordRepository functionWordRepository, PositionRepository positionRepository) {

this.dictionaryRepository = dictionaryRepository;

this.functionRepository = functionRepository;

this.functionWordRepository = functionWordRepository;

this.positionRepository = positionRepository;

}

public String translate(final String text){

List<Function> allFunctions = functionRepository.findAll();

String textWithoutPunctuation = ConstantUtils.removePunctuation(text);

Optional<Function> matchedFunctionOptional = allFunctions.stream()

.filter(function -> {

String russianTextRegex = ConstantUtils.toRegex(function.getRussianText());

return textWithoutPunctuation.matches(russianTextRegex);

})

.findFirst();

if(!matchedFunctionOptional.isPresent()){

return TRANSLATION\_ERROR;

}

Function function = matchedFunctionOptional.get();

String russianText = function.getRussianText();

String[] russianTextWords = russianText.split("([.,;/?! ])+");

String[] textWords = text.split("([.,;/?! ])+");

if(textWords.length != russianTextWords.length){

return TRANSLATION\_ERROR;

}

String result = function.getEnglishText();

for (int i = 0; i < textWords.length; i++) {

String russianWord = russianTextWords[i];

String textWord = textWords[i];

if(ConstantUtils.isStringMock(russianWord)){

Position position = positionRepository.findByFunctionIdAndRussianWordPosition(function.getId(), russianWord);

Optional<Dictionary> dictionaryOptional = dictionaryRepository.findByRussianWord(textWord);

if(!dictionaryOptional.isPresent()){

return String.format(WORD\_NOT\_FOUND\_REGEX, textWord);

}

result = result.replaceAll(position.getEnglishWordPosition(), dictionaryOptional.get().getEnglishWord());

}

}

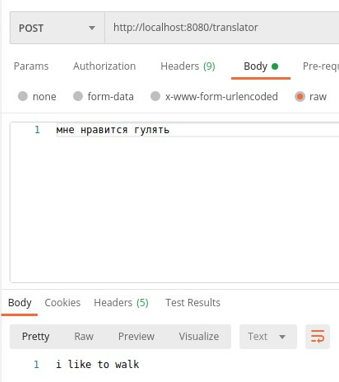
return result;

}

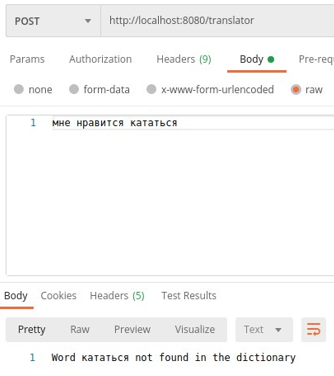
}}}

**Тестирование:**

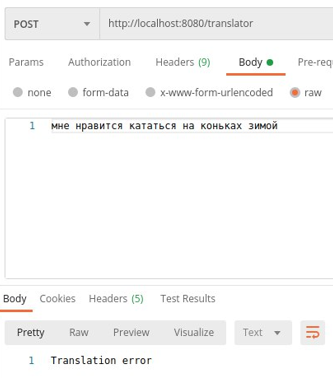
Перевод с фразой имеющейся в базе и словами имеющимися в базе:



При попытке перевода не существующего в базе слова:



При попытке перевода фразы не существующей в базе:



**Вывод:** освоили на практике основные принципы машинного перевода документов.