МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
<<БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ>>
КАФЕДРА ИИТ

Лабораторная работа №5
По дисциплине: ЕЯИИС
за седьмой семестр
Тема: «Разработка системы прямого машинного перевода документов»

Выполнил:
Студент 4 курса ФЭИС
Группы ИИ 12(1)
Блошук С.Г.
Проверил:
Крапивин Ю.Б.

Брест 2017

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

**Требования к разрабатываемой системе**

Система должна обеспечивать следующие минимальные возможности:

• на входе – естественно-языковой текст, подлежащий процедуре прямого машинного перевода;

• на выходе – переведенный документ;

• интерфейс системы должен быть предельно простым и доступным для пользователей любого уровня, содержать понятный набор инструментов и средств, а также help-средства.

• в качестве словаря система должна использовать БД PEREV.MDB, содержащую не менее 65 предикатных функций и не менее 1230 слов и выражений.

• в случае если в словаре программы не окажется какого-либо слова, она должна сообщить об этом пользователю и запросить перевод.

В отчет по работе необходимо включить:

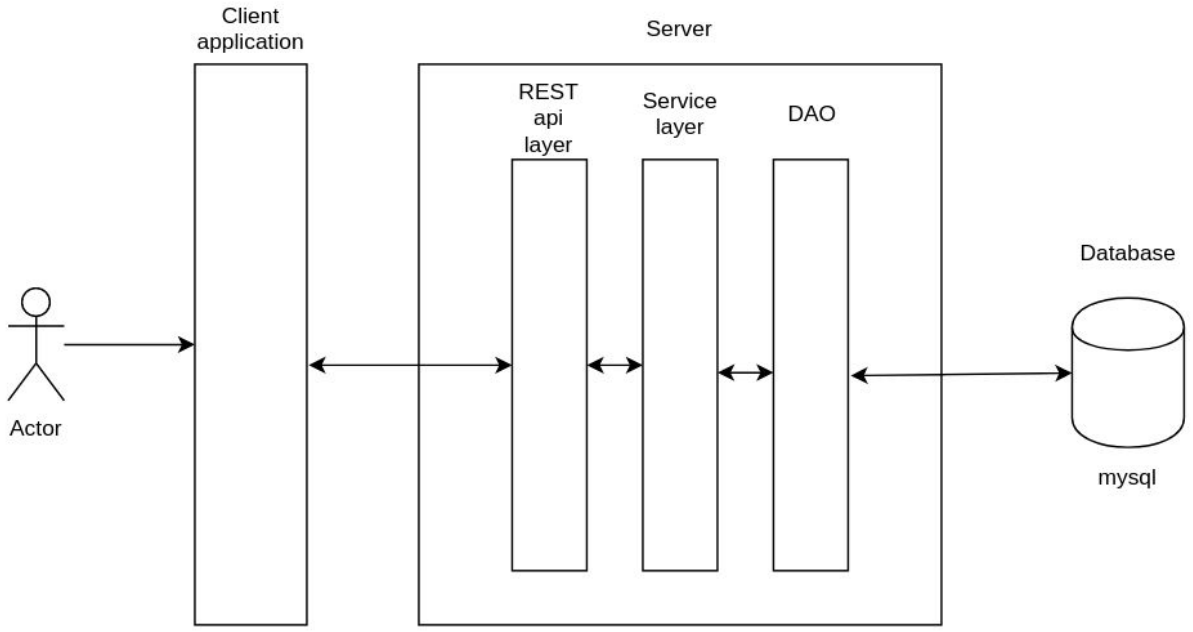
• описание структуры разработанной системы;

• основные алгоритмы реализации компонентов системы;

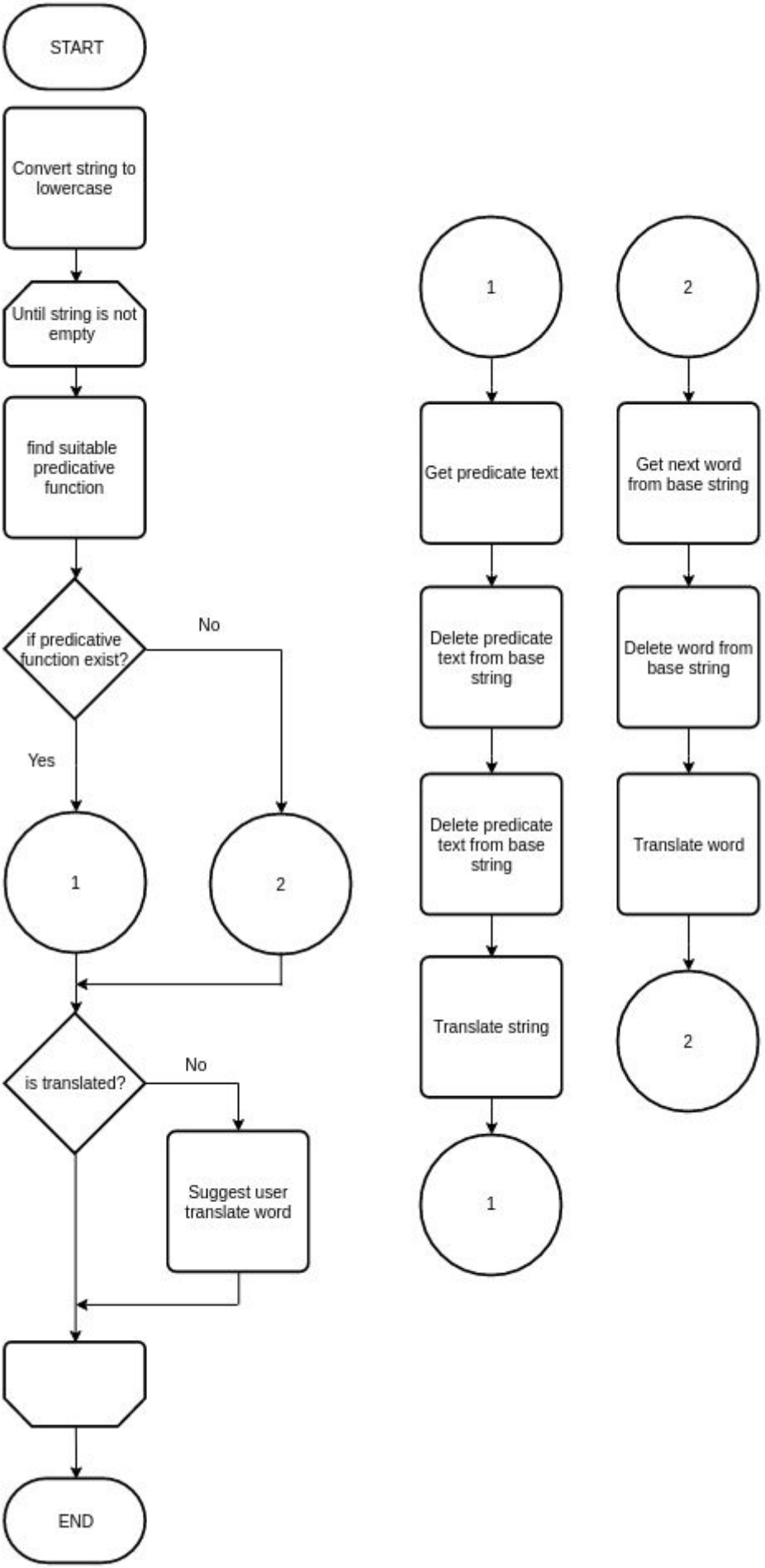
• результаты тестирования;

• результаты анализа полученных данных, и предложения по улучшению работы системы.

Структура системы:



Algorithm:


START
Convert string to lowercase
Until string is not empty
find suitable predicative function
if predicative function exist?
No
Yes
1
2
is translated?
No
Suggest user translate word
END
1
Get predicate text
Delete predicate text from base string
Delete predicate text from base string
Translate string
1
2
Get next word from base string
Delete word from base string
Translate word
2

Описание алгоритма:
Строка: “привет Боб, это Сергей”

1. Поиск предикатной функции for “привет Боб, это Сергей”:
   Rus: “привет x1 это x2”
   Eng: “hello y1 this is nda y2”
2. Просим пользователя перевести:
   [Боб] > [Bob], [Сергей]>[Siarhei]
3. Результат: hello bob this is nda Siarhei

Реализация:

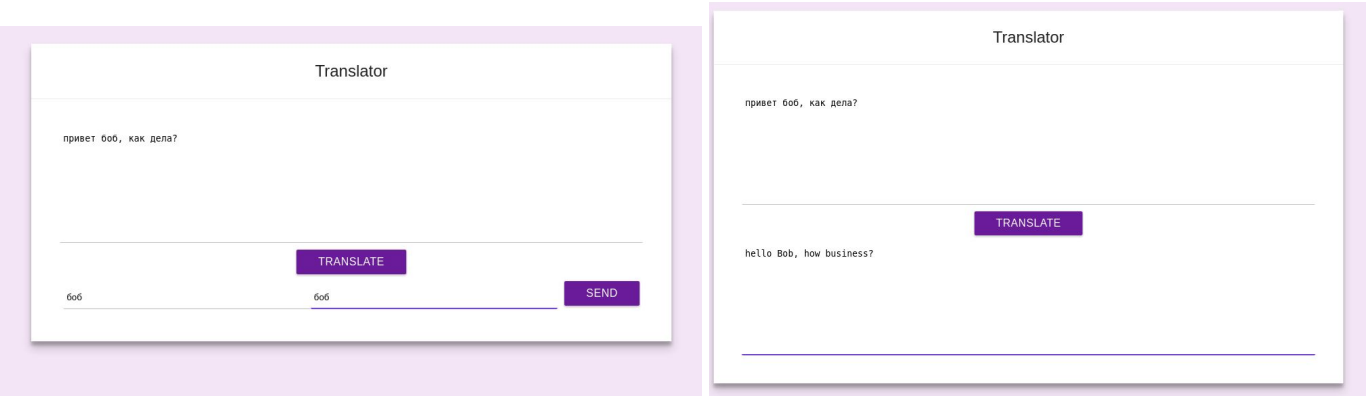
```
@RestController
@CrossOrigin
public class TranslationController {

    private TranslationService translationService;

    @Autowired
    public TranslationController(TranslationService translationService) {
        this.translationService = translationService;
    }

    @PostMapping("/translate")
    @ResponseBody
    public Map<String, Object> translate(@RequestBody TranslateRequest request) {
        Map<String, Object> response = new HashMap<>();

        response.put("text", translationService.translate(request.getText(), request.getDict()));
        response.put("status", "ok");

        return response;
    }

    private static class TranslateRequest {
        private String text;
        private Map<String, String> dict;

        public TranslateRequest() {
        }

        public TranslateRequest(String text, Map<String, String> dict) {
            this.text = text;
            this.dict = dict;
        }

        public String getText() {
            return text;
        }

        public void setText(String text) {
            this.text = text;
        }

        public Map<String, String> getDict() {
            return dict;
        }

        public void setDict(Map<String, String> dict) {
            this.dict = dict;
        }
    }
}

public class TranslationService {

    private NamedParameterJdbcTemplate jdbcTemplate;
```

```
    public TranslationService(DriverManagerDataSource dataSource) {
        jdbcTemplate = new NamedParameterJdbcTemplate(dataSource);

        langFuncs = jdbcTemplate.query("select * from Func", new FuncRowMapper());

        for (LangFunc func : langFuncs) {
            List<String> rusPlaceHolders = jdbcTemplate.queryForList("select PosRus from Pos where IDFunc
= " + func.getFuncId().toString(), new HashMap<>(), String.class);
            func.setRusPlaceHolders(rusPlaceHolders);

            List<String> engPlaceHolders = jdbcTemplate.queryForList("select PosEng from Pos where IDFunc
= " + func.getFuncId().toString(), new HashMap<>(), String.class);
            func.setEngPlaceHolders(engPlaceHolders);

            List<Pair<String, String>> fDict =
                    jdbcTemplate.query(
                            "select * from Func_Wrd fw join Dict d on fw.IDWrd = d.IDWrd where fw.IDFunc
= " + func.getFuncId().toString(),
                            new HashMap<>(),
                            (resultSet, i) -> new Pair<>(resultSet.getString("WrdRus"),
resultSet.getString("WrdEng"))
                    );

            HashMap<String, String> fDictMap = new HashMap<>();

            for (Pair<String, String> pair : fDict) {
                fDictMap.put(pair.getKey(), pair.getValue());
            }

            func.setSuitableWords(fDictMap);
        }

        List<Pair<String, String>> cDict =
                jdbcTemplate.query(
                        "select * from Dict",
                        new HashMap<>(),
                        (resultSet, i) -> new Pair<>(resultSet.getString("WrdRus"),
resultSet.getString("WrdEng"))
                );

        dict = new HashMap<>();

        for (Pair<String, String> pair : cDict) {
            dict.put(pair.getKey(), pair.getValue());
        }

        Collections.sort(langFuncs, new Comparator<LangFunc>() {
            @Override
            public int compare(LangFunc o1, LangFunc o2) {
                Integer o1_c = o1.getfTxtRus().split("\\s+").length;
                Integer o2_c = o2.getfTxtRus().split("\\s+").length;

                return o1_c > o2_c ? -1 : (o1_c < o2_c) ? 1 : 0;
            }
        });
    }

    private List<LangFunc> langFuncs;
    private HashMap<String, String> dict;

    public String translate(String input, Map<String, String> localDict) {
        String normInput = input.toLowerCase().trim();

        // Lang functions block
        for (LangFunc langFunc : langFuncs) {
            Pattern pattern = Pattern.compile(langFunc.getRegexString());
            Matcher matcher = pattern.matcher(normInput);

            List<Pair<String, String>> instructions = new ArrayList<>();

            while (matcher.find()) {
                int start = matcher.start();
```

```
                int end = matcher.end();

                List<String> placeholderWords = new ArrayList<>(); // Get x1 x2 ... words

                for (String placeholder : langFunc.getRusPlaceHolders()) {
                    placeholderWords.add(matcher.group(placeholder));
                }

                for (int i = 0; i < placeholderWords.size(); i++) { // Translate them
                    placeholderWords.set(i,
                                         translateWordForFunc(placeholderWords.get(i),
                                                              langFunc.getSuitableWords(),
                                                              localDict));
                }

                String eng = langFunc.getfTxtEng(); // Form english expression

                for (int i = 0; i < langFunc.getEngPlaceHolders().size(); i++) {
                    eng = eng.replaceAll(langFunc.getEngPlaceHolders().get(i), placeholderWords.get(i));
                }

                if (eng.endsWith("?")) {
                    eng = eng.substring(0, eng.length() - 1);
                }

                instructions.add(new Pair<>(normInput.substring(start, end), eng));
            }

            for (Pair<String, String> instruction : instructions) {
                normInput = normInput.replaceAll(instruction.getKey(), instruction.getValue());
            }
        }

        // Other words

        Pattern pattern = Pattern.compile("(?<word>[а-яА-Я\\-]+)");
        Matcher matcher = pattern.matcher(normInput);

        List<Pair<String, String>> instructions = new ArrayList<>();

        while (matcher.find()) {
            int start = matcher.start();
            int end = matcher.end();

            String rus = normInput.substring(start, end);
            String eng = translateWord(rus, localDict);

            instructions.add(new Pair<>(rus, eng));
        }

        for (Pair<String, String> instruction : instructions) {
            normInput = normInput.replaceAll(instruction.getKey(), instruction.getValue());
        }

        return normInput;
    }

    private String translateWordForFunc(String rus, Map<String, String> funcDict, Map<String, String>
localDict) {
        String eng = funcDict.get(rus);

        if (eng == null && localDict != null) {
            eng = localDict.get(rus);
        }

        if (eng == null) {
            throw new NoWordException("Word " + rus + " is not found", rus);
        }

        return eng;
    }
```

```
    private String translateWord(String rus, Map<String, String> localDict) {
        String eng = dict.get(rus);

        if (eng == null && localDict != null) {
            eng = localDict.get(rus);
        }

        if (eng == null) {
            throw new NoWordException("Word " + rus + " is not found", rus);
        }

        return eng;
    }

    private static class FuncRowMapper implements RowMapper<LangFunc> {
        @Override
        public LangFunc mapRow(ResultSet resultSet, int i) throws SQLException {
            LangFunc langFunc = new LangFunc();

            langFunc.setFuncId(resultSet.getInt("IDFunc"));
            langFunc.setfTxtRus(resultSet.getString("FTxtRus"));
            langFunc.setfTxtEng(resultSet.getString("FTxtEng"));

            return langFunc;
        }
    }

}
```

Результат работы:




Недостатки системы:

1) не запоминает пунктуацию
2) Вовлекает пользователя в процесс перевода
3) Маленькая база данных предикатных функций и слов.
4) Не учитывается морфология

Предложения по улучшению системы:

1) Разработать базу данных для учета пунктуации в предикатах, добавить больше слов или хранить словарь всех слов целиком
2) Подключить библиотеки для анализа морфологии, добавить в предикатные функции учет информации о морфологии токенов

Вывод: освоил на практике основные принципы машинного перевода документов.