Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

Лабораторная работа №2 по курсу «ЕЯзИИС» на тему:
«Представление результатов синтаксического анализа в памяти интеллектуальной системы»

Выполнили студенты группы 821702: Астапович О.С., Зайцев Н.С.

Проверил: Крапивин Ю.Б.

МИНСК
2021

1. Цель:

Закрепить знания синтаксического анализа текста.

2. Постановка задачи:

Изучить теоретический материал, необходимый для решения задачи автоматического синтаксического анализа текста естественного языка.

Закрепить навыки программирования естественно-языковых систем и вспомогательных прикладных программ на одном из языков программирования.

3. Ход работы:

Целью разработанной системы является автоматический синтаксический анализ текста естественного языка, подаваемого на вход системы из .html файла. Результатом ее работы является дерево синтаксического разбора.

Для хранения входных данных был использован class 'str', для хранения выходных данных был использован class 'nltk.tree.Tree'.

Для построения деревьев синтаксического разбора предложений предложение полностью считывается, из него убираются переводы строк. Далее предложения разбиваются на слова и получаются теги частей речи для каждого слова. Из массива убираются знаки препинания и слова связываются с подходящими им частями речи и строим дерево.

```
def draw_syntax_tree():
    text = calculated_text.get(1.0, END)
    text = text.replace('\n', '')
    if text != '':
        doc = nltk.word_tokenize(text)
        doc = nltk.pos_tag(doc, tagset='universal')
        text_without_punct = []
        for item in doc:
            if item[1] != ',' and item[1] != '.':
                text_without_punct.append(item)
        cp = nltk.RegexpParser(grammar)
        result = cp.parse(text_without_punct)
        result.draw()
```

Для открытия ранее сохраненных предложений было использовано расширение .html. Пользователь открывает выбранный файл, из него считывается текст и он записывается в переменную, которая потом будет обрабатываться.

```
def open_file_and_input_text():
    file_name = fd.askopenfilename(filetypes=(("Html files", "*.html"),))
    if file_name != '':
        with open(file_name, 'r') as file:
            text = file.read()
            text = html_parser(text)
            calculated_text.delete(1.0, END)
            calculated_text.insert(1.0, text)
```

Для получения информации о системе пользователю необходимо нажать на кнопку “Help” после чего открывается окно с подробной информацией.

```
def information():
    messagebox.askquestion("Help", "1. Input text or open file.\n"
                                   "2. Send button 'CREATE'.\n"
                                   "3. Look at the painted syntax tree.", type='ok')
```

Для сохранения дерева синтаксического разбора была использована функция print_to_file, она сохраняет дерево в расширении .ps.

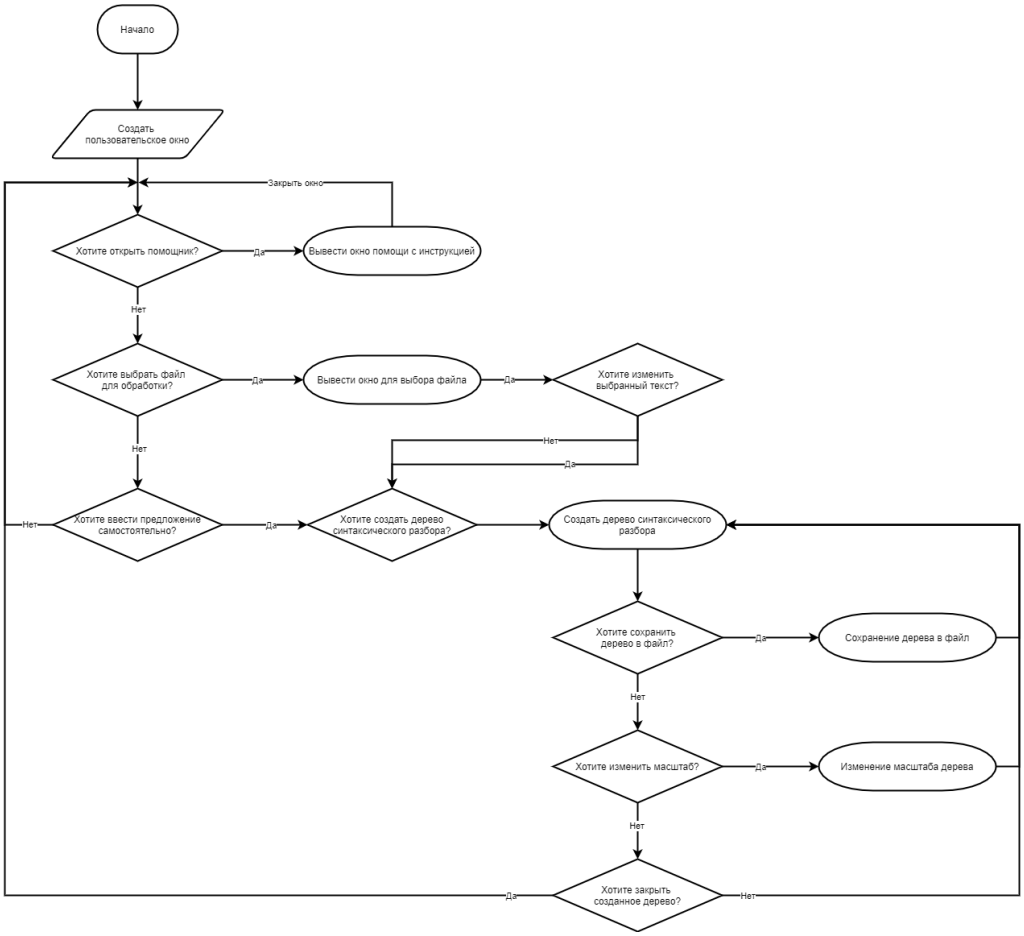
```
def print_to_file(self, filename=None):
    """
    Print the contents of this ``CanvasFrame`` to a postscript
    file.  If no filename is given, then prompt the user for one.

    :param filename: The name of the file to print the tree to.
    :type filename: str
    :rtype: None
    """
    if filename is None:
        ftypes = [("Postscript files", ".ps"), ("All files", "*")]
        filename = asksaveasfilename(filetypes=ftypes, defaultextension=".ps")
        if not filename:
            return
    (x0, y0, w, h) = self.scrollregion()
    postscript = self._canvas.postscript(
        x=x0,
        y=y0,
        width=w + 2,
        height=h + 2,
        pagewidth=w + 2,  # points = 1/72 inch
        pageheight=h + 2,  # points = 1/72 inch
        pagex=0,
        pagey=0,
    )
    # workaround for bug in Tk font handling
    postscript = postscript.replace(" 0 scalefont ", " 9 scalefont ")
    with open(filename, "wb") as f:
        f.write(postscript.encode("utf8"))
```

Анализ текста “I usually have four or five lessons at school.” занял 0.62 секунд.

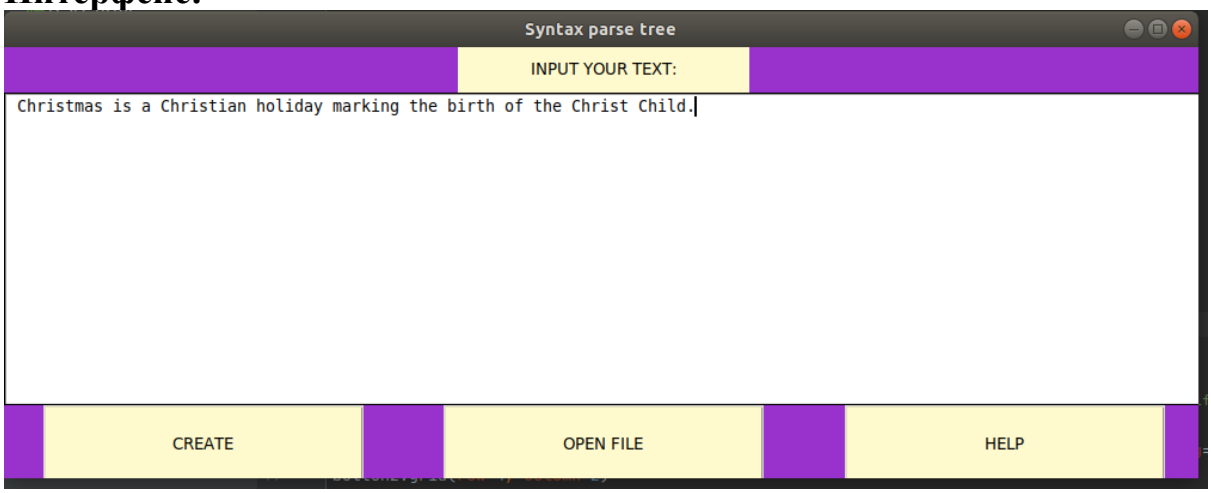
Анализ текста “Thought it was created in the early decades of the 20th century, blues music has had a huge influence on American popular music up to the present days.” занял 0.9 секунд.

Анализ текста “Christmas is a Christian holiday marking the birth of the Christ Child.” занял 0.65 секунд.

Алгоритм реализованной программы, функции представлены черными ящиками.



Интерфейс:



Выводы.

В ходе лабораторной работы была решена задача автоматического синтаксического анализа текста естественного языка. Была разработана система позволяющая пользователю производить синтаксический разбор текстов из .html файла и печатать в .ps файл результат работы программы.

Исходный код:

```
import nltk
from tkinter import *
from tkinter import messagebox
from tkinter import filedialog as fd

from pyquery import PyQuery

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('universal_tagset')

root = Tk()
root["bg"] = "dark orchid"
root.title("Syntax parse tree")
root.resizable(width=True, height=True)
root.geometry("1000x360+300+300")
label = Label(root, text='INPUT YOUR TEXT:', height=2, width=30, bg='lemon chiffon')
label.grid(row=0, column=2)
calculated_text = Text(root, height=15, width=125)
calculated_text.grid(row=1, column=1, sticky='nsew', columnspan=3, rowspan=3)


def html_parser(html):
    pq = PyQuery(html)
    tag = pq('textarea')
    return tag.text()


def open_file_and_input_text():
    file_name = fd.askopenfilename(filetypes=(("Html files", "*.html"),))
    if file_name != '':
        with open(file_name, 'r') as file:
            text = file.read()
            text = html_parser(text)
            calculated_text.delete(1.0, END)
            calculated_text.insert(1.0, text)


def information():
    messagebox.askquestion("Help", "1. Input text or open file.\n"
                                   "2. Send button 'CREATE'.\n"
                                   "3. Look at the painted syntax tree.", type='ok')
```

```
grammar = r"""
      P: {<PRT|ADP>}
      V: {<VERB>}
      N: {<NOUN|PRON>}
      NP: {<N|NP|P>+<ADJ|NUM|DET>+}
      NP: {<ADJ|NUM|DET>+<N|NP|P>+}
      PP: {<P><NP>|<NP><P>}
      VP: {<NP|N><V>}
      VP: {<VP><NP|N||ADV>}
      VP: {<NP|N|ADV><VP>}
      VP: {<VP><PP|P>}
      """


def draw_syntax_tree():
   text = calculated_text.get(1.0, END)
   text = text.replace('\n', '')
   if text != '':
      doc = nltk.word_tokenize(text)
      doc = nltk.pos_tag(doc, tagset='universal')
      text_without_punct = []
      for item in doc:
         if item[1] != ',' and item[1] != '.':
            text_without_punct.append(item)
      cp = nltk.RegexpParser(grammar)
      result = cp.parse(text_without_punct)
      result.draw()


button1 = Button(text="CREATE", height=3, width=30, command=draw_syntax_tree, bg='lemon
chiffon')
button1.grid(row=4, column=1)
button2 = Button(text="OPEN FILE", height=3, width=30, command=open_file_and_input_text,
bg='lemon chiffon')
button2.grid(row=4, column=2)
button3 = Button(text="HELP", height=3, width=30, command=information, bg='lemon chiffon')
button3.grid(row=4, column=3)
root.mainloop()
```