Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ

Кафедра интеллектуальных информационных технологий

Отчет по практической работе №3
по курсу «Естественно-языковые интерфейсы интеллектуальных систем»
на тему «Представление результатов семантического анализа в памяти интеллектуальной системы»

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

Проверил: Крапивин Ю.Б.

МИНСК
2021

1. Цель:

Закрепить знания семантического анализа текста.

2. Постановка задачи:

Изучить теоретический материал, необходимый для решения задачи автоматического семантического 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 семантический анализ текста естественного языка, подаваемого на вход системы из .html файла.

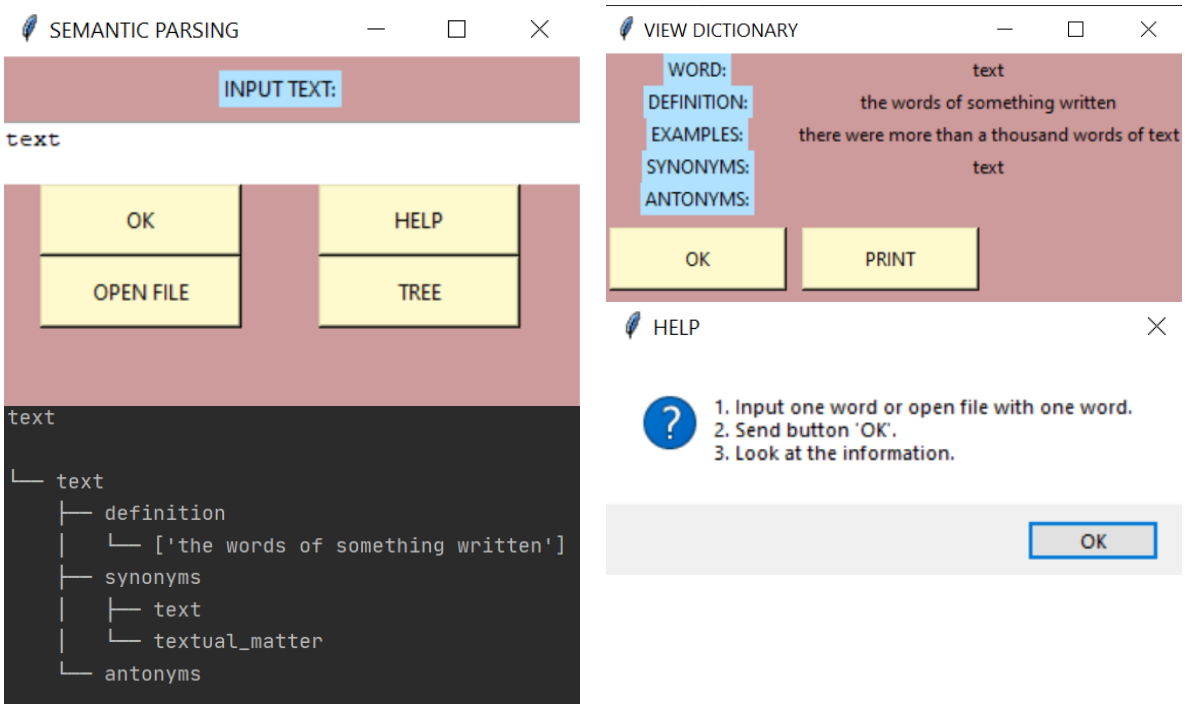
Wordnet - это библиотека для чтения NLTK, лексическая база данных для английского языка. Ее можно определить как семантически ориентированный словарь английского языка. Из WordNet можно выделить информацию о данном слове или фразе, например:

- синоним
- гиперонимы, гипонимов
- голонимы
- меронимы

WordNet также предоставляет информацию о согласованных терминах, производных, смыслах и многом другом. Он используется для поиска сходства между любыми двумя словами. Он также содержит информацию о результатах родственного слова. Статистика показывает, что в English WordNet включено 155287 слов и 117659 наборов синонимов.

Для хранения входных данных был использован class 'list', для хранения выходных данных был использован class 'list'.

При начале работы на вход поступает слово английского языка, либо же любой HTML файл, который содержит одно слово. Для выбора HTML файла с устройства необходимо нажать кнопку “OPEN FILE” и выбрать сам файл. Дальше происходит перенос из файла в поле для ввода.



После нажатия на кнопку “ОК” слово из ячейки обрабатывается с помощью словаря WordNet. Слово ищется в базе словаря, потом выбираются оттуда только те части связанных понятийно слов, которые нам необходимы. Эти части в последующем выводятся в определенном порядке напротив константных полей, содержащих различные понятия словоформ. Данное действие выполняется столько раз, сколько у нас слов.

```
def semantic_parse(text):
    nltk.download('wordnet')
    syn = wordnet.synsets(text)
    print(syn)
    examples = syn[0].examples()
    defin = []
    defin.append(syn[0].definition())
    definition.append(defin)
    synon=[]
    anton=[]
    hypon=[]
    hypern=[]
    for l in syn[0].lemmas():
        synon.append(l.name())
        if l.antonyms():
            anton.append(l.antonyms()[0].name())

    synonyms.append(synon)
    antonyms.append(anton)

    return dict(examples=examples, definition=definition, synonyms=synonyms,
                antonyms=antonyms)
```

После вывода на экран результатов мы можем либо выйти назад к выбору слова для обработки, либо сохранить результаты нашего семантического анализа, нажав на кнопки "PRINT".

```
def saveFile(data):
    text_list = []
    for key in data:
        text_list.append(data[key])
    print(text_list)
    file_path = filedialog.asksaveasfilename()
    if file_path != "":
        f = open(file_path, 'wb')
        pickle.dump(text_list, f)
        f.close()
```

Для построения деревьев необходимо нажать на кнопку “TREE”. Дерево строится одно для всего предложения, где все вначале делится на слова, а потом на их формы. Итог записывается в файл.

```
def test():
    a = calculated_text.get(1.0, END)
    sentense_par=Node(a)
    text_without_punct = []
    for sentence in nltk.sent_tokenize(a):
        for word in nltk.word_tokenize(sentence):
            if word not in punctuation:
                text_without_punct.append(word)
    for z in range (len(text_without_punct)):
        word = Node(text_without_punct[z], parent=sentense_par)
        definition_tree = Node("definition", parent=word)
        example_definition = Node(definition[z], parent=definition_tree)
        synonyms_tree = Node("synonyms", parent=word)
        for i in range (len(synonyms[z])):
         example_synonyms = Node(synonyms[z][i], parent=synonyms_tree)
        antonyms_tree = Node("antonyms", parent=word)
        for j in range (len(antonyms[z])):
         example_antonyms = Node(antonyms[z][j], parent=antonyms_tree)
    for pre, fill, node in RenderTree(sentense_par):
        print("%s%s" % (pre, node.name))
```

При нажатии на кнопку “HELP” происходит вызов окна с сообщением, содержащим инструкцию к данной лабораторной.

```
def info():
    messagebox.askquestion("HELP", "1. Input one word or open file with one word.\n"
                                   "2. Send button 'OK'.\n"
                                   "3. Look at the information.", type='ok')
```

Выводы.

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

Исходники кода:

```
import nltk
from nltk.corpus import wordnet
from tkinter import *
from tkinter import messagebox, filedialog
from tkinter import filedialog as fd
from pyquery import PyQuery
from spacy.compat import pickle
from string import punctuation
from anytree import Node, RenderTree
import pickle

root = Tk()
root["bg"] = "dark orchid"
root.title("SEMANTIC PARSING")
root.resizable(width=True, height=True)
root.geometry("330x200+300+300")

global calculated_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 html_parser(html):
    pq = PyQuery(html)
    tag = pq('p')
    return tag.text()


def children_view():
    children = Toplevel(root)
```

```
    children.title('VIEW DICTIONARY')
    children["bg"] = "dark orchid"
    b3 = Button(children, width=15, text="OK", bg='lemon chiffon', command=lambda:
quit_window(children))
    b3.grid(row=7, column=1, sticky=W, padx=5, pady=8)
    b4 = Button(children, width=15, text="PRINT", bg='lemon chiffon', command=lambda:
saveFile(data))
    b4.grid(row=7, column=2, sticky=W, padx=5, pady=8)

    b = Label(children, bg='dark orchid', text='WORD:')
    b.grid(row=0, column=1)
    b = Label(children, bg='dark orchid', text='DEFINITION:')
    b.grid(row=1, column=1)
    b = Label(children, bg='dark orchid', text='EXAMPLES:')
    b.grid(row=2, column=1)
    b = Label(children, bg='dark orchid', text='SYNONYMS:')
    b.grid(row=3, column=1)
    b = Label(children, bg='dark orchid', text='ANTONYMS:')
    b.grid(row=4, column=1)


    return children


def quit_window(window):
    window.destroy()


definition = []
synonyms = []
antonyms = []


def semantic_parse(text):
    nltk.download('wordnet')
    syn = wordnet.synsets(text)
    print(syn)
    examples = syn[0].examples()
    defin = []
    defin.append(syn[0].definition())
    definition.append(defin)
    synon=[]
    anton=[]
    hypon=[]
    hypern=[]
    for l in syn[0].lemmas():
        synon.append(l.name())
        if l.antonyms():
```

```
            anton.append(l.antonyms()[0].name())

    synonyms.append(synon)
    antonyms.append(anton)

    return dict(examples=examples, definition=definition, synonyms=synonyms,
                antonyms=antonyms)


def saveFile(data):
    text_list = []
    for key in data:
        text_list.append(data[key])
    print(text_list)
    file_path = filedialog.asksaveasfilename()
    if file_path != "":
        f = open(file_path, 'wb')
        pickle.dump(text_list, f)
        f.close()


def view_window():
    global data
    text = calculated_text.get(1.0, END)
    text = text.replace('\n', '')

    if text != '':
        text_without_punct = []
        for sentence in nltk.sent_tokenize(text):
            for word in nltk.word_tokenize(sentence):
                if word not in punctuation:
                    text_without_punct.append(word)
        for q in range (len(text_without_punct)):
            print(text_without_punct[q])
            if text_without_punct[q] != '':
                children = children_view()
                data = semantic_parse(text_without_punct[q])
                print(data)

                b = Label(children, bg='dark orchid', text=text_without_punct[q])
                b.grid(row=0, column=2)
                b = Label(children, bg='dark orchid', text=data['definition'][q][0])
                b.grid(row=1, column=2)
                if data['examples']:
                    b = Label(children, bg='dark orchid', text=data['examples'][0])
                else:
                    b = Label(children, bg='dark orchid', text=data['examples'])
                b.grid(row=2, column=2)
```

```
            b = Label(children, bg='dark orchid', height=1, text=data['synonyms'][q][0])
            b.grid(row=3, column=2)
            b = Label(children, bg='dark orchid', height=1, text=data['antonyms'][q])
            b.grid(row=4, column=2)


def info():
    messagebox.askquestion("HELP", "1. Input one word or open file with one word.\n"
                           "2. Send button 'OK'.\n"
                           "3. Look at the information.", type='ok')


def test():
    a = calculated_text.get(1.0, END)
    sentense_par=Node(a)
    text_without_punct = []
    for sentence in nltk.sent_tokenize(a):
        for word in nltk.word_tokenize(sentence):
            if word not in punctuation:
                text_without_punct.append(word)
    for z in range (len(text_without_punct)):
        word = Node(text_without_punct[z], parent=sentense_par)
        definition_tree = Node("definition", parent=word)
        example_definition = Node(definition[z], parent=definition_tree)
        synonyms_tree = Node("synonyms", parent=word)
        for i in range (len(synonyms[z])):
         example_synonyms = Node(synonyms[z][i], parent=synonyms_tree)
        antonyms_tree = Node("antonyms", parent=word)
        for j in range (len(antonyms[z])):
         example_antonyms = Node(antonyms[z][j], parent=antonyms_tree)
    for pre, fill, node in RenderTree(sentense_par):
        print("%s%s" % (pre, node.name))


label = Label(root, bg='dark orchid', text='INPUT YOUR TEXT:')
label.grid(row=1, column=2, padx=5, pady=8)
calculated_text = Text(root, height=1, width=40)
calculated_text.grid(row=3, column=1, sticky='nsew', columnspan=3)
b1 = Button(text="OK", bg='lemon chiffon', width=20, command=view_window)
b1.grid(row=5, column=2)
b2 = Button(text="OPEN FILE", bg='lemon chiffon', width=20, command=open_file_and_input_text)
b2.grid(row=6, column=2)
button3 = Button(text="HELP", bg='lemon chiffon', width=20, command=info)
button3.grid(row=7, column=2)
button3 = Button(text="TREE", bg='lemon chiffon', width=20, command=test)
button3.grid(row=8, column=2)

root.mainloop()
```