МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**ИНТЕЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

ОТЧЁТ

По лабораторной работе № 5

Выполнил:

Студент группы ИИ-22

Копанчук Евгений Романович

Проверил:

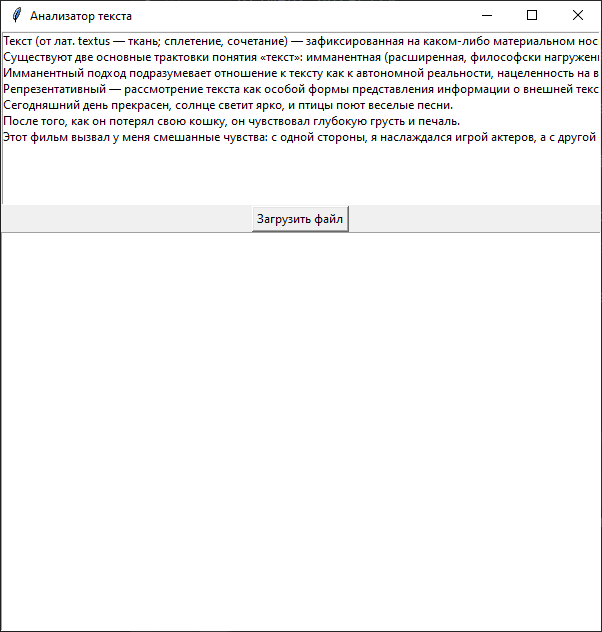
Булей Е. В.

Брест – 2024

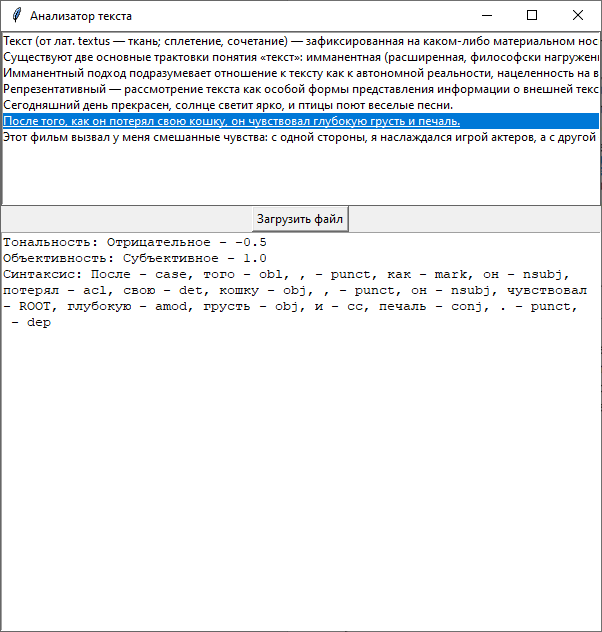
**Ход работы**

**Задание:**

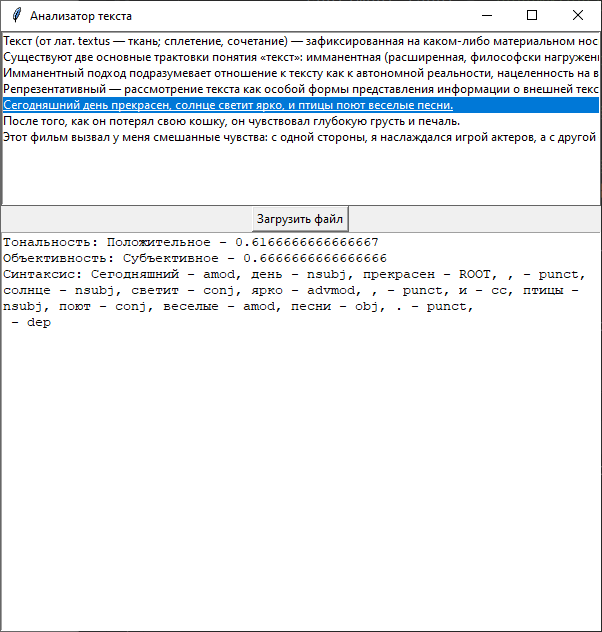
1. *Входные данные* – текст заданного естественного языка ­
2. *Выходные данные* – структуры, полученные при проведении автоматического семантико-синтаксического анализа предложений входного текста
3. Взаимодействие с пользователем посредством графического интерфейса



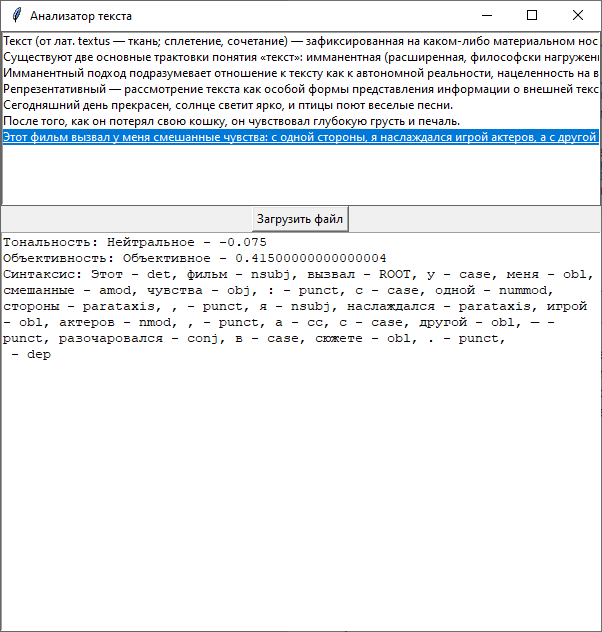
*Приложение с загруженным текстом*



*Анализ предложения 1*



*Анализ предложения 2*



*Анализ предложения 3*

**Код программы:**

import tkinter as tk

from tkinter import filedialog

from docx import Document

import spacy

from textblob import TextBlob

from transformers import MarianMTModel, MarianTokenizer

class TextAnalyzerApp:

def \_\_init\_\_(self, master):

self.master = master

master.title("Анализатор текста")

master.geometry("600x600")

self.sentences\_listbox = tk.Listbox(master, width=50)

self.sentences\_listbox.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

self.sentences\_listbox.bind("<<ListboxSelect>>", self.show\_analysis)

self.analysis\_text = tk.Text(master, wrap="word")

self.analysis\_text.pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

self.load\_button = tk.Button(master, text="Загрузить файл", command=self.load\_file)

self.load\_button.pack(side=tk.BOTTOM)

self.tokenizer = spacy.load('ru\_core\_news\_sm')

self.model\_ru\_en = MarianMTModel.from\_pretrained("Helsinki-NLP/opus-mt-ru-en")

self.tokenizer\_ru\_en = MarianTokenizer.from\_pretrained("Helsinki-NLP/opus-mt-ru-en")

def load\_file(self):

file\_path = filedialog.askopenfilename(filetypes=[("Word files", "\*.docx")])

if file\_path:

doc = Document(file\_path)

text = ""

for paragraph in doc.paragraphs:

text += paragraph.text + "\n"

self.analyze\_text(text)

def analyze\_text(self, text):

self.sentences\_listbox.delete(0, tk.END)

doc = self.tokenizer(text)

for sentence in doc.sents:

self.sentences\_listbox.insert(tk.END, sentence.text)

def translate\_ru\_en(self, text):

inputs = self.tokenizer\_ru\_en(text, return\_tensors="pt", padding=True, truncation=True)

outputs = self.model\_ru\_en.generate(\*\*inputs)

translated\_text = self.tokenizer\_ru\_en.batch\_decode(outputs, skip\_special\_tokens=True)

return translated\_text[0]

def show\_analysis(self, event):

selected\_sentence\_index = self.sentences\_listbox.curselection()[0]

selected\_sentence = self.sentences\_listbox.get(selected\_sentence\_index)

en\_selected\_sentence = self.translate\_ru\_en(selected\_sentence)

blob = TextBlob(en\_selected\_sentence)

sentiment = blob.sentiment

sentiment\_label = sentiment.polarity

if sentiment\_label > 0.1:

sentiment\_label = "Положительное - " + str(sentiment.polarity)

elif sentiment\_label < -0.1:

sentiment\_label = "Отрицательное - " + str(sentiment.polarity)

else:

sentiment\_label = "Нейтральное - " + str(sentiment.polarity)

subjectivity\_label = "Субъективное - " + str(sentiment.subjectivity) if sentiment.subjectivity > 0.5 else "Объективное - " + str(sentiment.subjectivity)

analysis\_result = f"Тональность: {sentiment\_label}\nОбъективность: {subjectivity\_label}"

syntax\_analysis = self.get\_syntax\_analysis(selected\_sentence)

analysis\_result += f"\nСинтаксис: {syntax\_analysis}"

self.analysis\_text.delete("1.0", tk.END)

self.analysis\_text.insert(tk.END, analysis\_result)

def get\_syntax\_analysis(self, sentence):

doc = self.tokenizer(sentence)

syntax\_info = ', '.join([f"{token.text} - {token.dep\_}" for token in doc])

return syntax\_info

def main():

root = tk.Tk()

app = TextAnalyzerApp(root)

root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

main()