Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине: «Естественно-языковой интерфейс ИС»

# Тема: «Разработка системы автоматического реферирования документов»

Выполнил:

Студент 4 курса

Группы ИИ-21

Ливтинюк Т. В.

Проверила:

Якимук А. В.

Брест 2024

**Цель:** освоить на практике основные принципы автоматического

реферирования документов.



**Ход работы:**

на входе – на входе – текстовые документы одинакового размера (например, 10 страниц

формата А4), содержащие тексты из предметных областей на естественных языках согласно

варианту подлежащие процедуре автоматического реферирования;

• на выходе – активная ссылка на исходный документ и построенный, в соответствии с

вариантом реферат документа, состоящий из 2-х разделов: 1 - классического реферата и

реферата в виде списка ключевых слов (по методу Sentence extraction ) ; 2 – реферата,

построенного с применением машинного обучения (ML ).

• наличие средств сохранения в файл и распечатки полученной на выходе информации;

• интерфейс системы должен быть предельно простым и доступным для пользователей

любого уровня, содержать понятный набор инструментов и средств, а также help-средства.

import re

import nltk

import tkinter as tk

from tkinter import filedialog, Text

from sklearn.feature\_extraction.text import TfidfVectorizer

from transformers import MBartForConditionalGeneration, MBart50TokenizerFast

from docx import Document

import threading

nltk.download('punkt')

nltk.download('punkt\_tab')

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.tokenize import sent\_tokenize, word\_tokenize

# Функция для чтения текста из docx

def extract\_text\_from\_docx(docx\_file):

doc = Document(docx\_file)

full\_text = []

for paragraph in doc.paragraphs:

full\_text.append(paragraph.text)

return '\n'.join(full\_text)

# Функция для очистки текста

def clean\_text(text):

text = re.sub(r'\s+', ' ', text) # Удаление лишних пробелов

text = re.sub(r'\d+', '', text) # Удаление чисел

return text

# Классический метод с использованием TF-IDF

def extract\_sentences\_tfidf(text):

sentences = sent\_tokenize(text)

stop\_words = set(stopwords.words('russian'))

cleaned\_sentences = [' '.join([word for word in word\_tokenize(sentence.lower()) if word.isalpha() and word not in stop\_words]) for sentence in sentences]

vectorizer = TfidfVectorizer()

tfidf\_matrix = vectorizer.fit\_transform(cleaned\_sentences)

return sentences, tfidf\_matrix.sum(axis=1) #type: ignore

def generate\_summary(sentences, sentence\_weights, summary\_length=10):

ranked\_sentences = sorted(((weight, sentence) for sentence, weight in zip(sentences, sentence\_weights)), reverse=True)

summary = ' '.join([sentence for weight, sentence in ranked\_sentences[:summary\_length]])

return summary

# Метод с использованием модели mBART

def generate\_ml\_summary(text):

model\_name = "facebook/mbart-large-50-many-to-many-mmt"

model = MBartForConditionalGeneration.from\_pretrained(model\_name)

tokenizer = MBart50TokenizerFast.from\_pretrained(model\_name)

tokenizer.src\_lang = "ru\_RU"

inputs = tokenizer.encode(text, return\_tensors="pt", max\_length=1024, truncation=True)

summary\_ids = model.generate(inputs, max\_length=250, min\_length=40, length\_penalty=2.0, num\_beams=4, early\_stopping=False, forced\_bos\_token\_id=tokenizer.lang\_code\_to\_id["ru\_RU"])

summary = tokenizer.decode(summary\_ids[0], skip\_special\_tokens=True)

return summary

# Функция для обработки документа и вывода результата

def summarize\_docx(docx\_file, method):

text = extract\_text\_from\_docx(docx\_file)

cleaned\_text = clean\_text(text)

if method == "TF-IDF":

sentences, sentence\_weights = extract\_sentences\_tfidf(cleaned\_text)

summary = generate\_summary(sentences, sentence\_weights)

elif method == "ML":

summary = generate\_ml\_summary(cleaned\_text)

return text, summary

# Функция для выбора файла

def open\_file():

filepath = filedialog.askopenfilename(filetypes=[("Document files", "\*.docx")])

if filepath:

selected\_file.set(filepath)

text = extract\_text\_from\_docx(filepath)

original\_text\_box.delete(1.0, tk.END)

original\_text\_box.insert(tk.END, text)

def run\_summarize():

threading.Thread(target=summarize).start()

def summarize():

if selected\_file.get():

method\_choice = method.get()

original\_text\_box.delete(1.0, tk.END)

summary\_text\_box.delete(1.0, tk.END)

try:

original\_text, summary = summarize\_docx(selected\_file.get(), method\_choice)

# Выводим исходный текст

original\_text\_box.insert(tk.END, original\_text)

# Выводим сводку

summary\_text\_box.insert(tk.END, summary)

except Exception as e:

summary\_text\_box.insert(tk.END, f"Ошибка при создании сводки: {e}")

root = tk.Tk()

root.title("Краткость - С. Т.")

selected\_file = tk.StringVar()

method = tk.StringVar(value="TF-IDF")

file\_button = tk.Button(root, text="Выбрать файл .docx", command=open\_file)

file\_button.pack(pady=10)

file\_label = tk.Label(root, textvariable=selected\_file)

file\_label.pack()

tfidf\_radio = tk.Radiobutton(root, text="TF-IDF", variable=method, value="TF-IDF")

tfidf\_radio.pack(anchor="w")

ml\_radio = tk.Radiobutton(root, text="ML", variable=method, value="ML")

ml\_radio.pack(anchor="w")

summarize\_button = tk.Button(root, text="Сделать сводку", command=run\_summarize)

summarize\_button.pack(pady=10)

original\_text\_label = tk.Label(root, text="Исходный текст:")

original\_text\_label.pack()

original\_text\_box = Text(root, height=10, width=80)

original\_text\_box.pack(pady=5)

summary\_text\_label = tk.Label(root, text="Сводка:")

summary\_text\_label.pack()

summary\_text\_box = Text(root, height=10, width=80)

summary\_text\_box.pack(pady=5)

root.mainloop()

****

**Вывод:** в ходе выполнения лабораторной работы освоил принципы автоматического реферирования документов на естественном языке при помощи метода Sentence Extraction и при помощи ML-методов.