**Алгоритм решения задачи**

Для автоматического построения рефератов текстовых документов можно использовать различные подходы и алгоритмы. Одним из распространенных методов является экстрактивное резюмирование, при котором извлекаются наиболее информативные предложения из исходного текста для создания реферата. Вот общий алгоритм решения задачи:

1. Загрузка и предобработка текстовых данных:
   * Прочитайте входные данные, содержащие массив текстовых документов в формате JSON.
   * Выполните предварительную обработку текстов, включающую удаление лишних символов, приведение текста к нижнему регистру и т.д.
   * Разбейте каждый текстовый документ на предложения.
2. Вычисление рейтинга предложений:
   * Примените методы анализа и оценки текста, чтобы определить наиболее информативные предложения.
   * Оцените каждое предложение на основе его значимости, структуры и связей с другими предложениями в тексте.
   * Рассмотрите различные критерии для оценки значимости предложений, такие как частота ключевых слов, длина предложения, наличие цитат и т.д.
   * Назначьте каждому предложению рейтинг, отражающий его важность в тексте.
3. Извлечение наиболее информативных предложений:
   * Выберите предложения с наивысшими рейтингами, чтобы сформировать реферат для каждого текстового документа.
   * Убедитесь, что суммарная длина выбранных предложений не превышает установленный предел (например, 300 знаков).
   * При необходимости можно использовать алгоритмы оптимизации или эвристики для сокращения и адаптации рефератов.
4. Форматирование рефератов и создание выходных данных:
   * Обработайте выбранные предложения для форматирования реферата, удалите повторяющиеся предложения или фразы.
   * Создайте массив рефератов, где каждый элемент соответствует реферату для соответствующего текстового документа из входных данных.
   * Преобразуйте массив рефератов в формат JSON для выходных данных.

**Листинг программы**

import json

from nltk.tokenize import sent\_tokenize

from nltk.corpus import stopwords

from nltk.probability import FreqDist

from heapq import nlargest

import re

from nltk.tokenize import word\_tokenize

from nltk.probability import FreqDist

def preprocess\_text(text):

# Удаление символов пунктуации

text = re.sub(r'[^\w\s]', '', text)

# Приведение текста к нижнему регистру

text = text.lower()

# Токенизация текста на отдельные слова

tokens = word\_tokenize(text)

# Удаление стоп-слов

stop\_words = set(stopwords.words('english')) # Замените 'english' на язык вашего текста

filtered\_tokens = [token for token in tokens if token not in stop\_words]

# Объединение токенов в предобработанный текст

preprocessed\_text = ' '.join(filtered\_tokens)

return preprocessed\_text

def calculate\_sentence\_scores(sentences):

# Собираем все слова из всех предложений

all\_words = [word for sentence in sentences for word in word\_tokenize(sentence)]

# Вычисляем частоту слов

word\_frequencies = FreqDist(all\_words)

# Вычисляем рейтинг предложений на основе суммы частот слов в каждом предложении

sentence\_scores = {}

for i, sentence in enumerate(sentences):

sentence\_tokens = word\_tokenize(sentence)

sentence\_score = sum(word\_frequencies[word] for word in sentence\_tokens)

sentence\_scores[i] = sentence\_score

return sentence\_scores

def extract\_summary(text, max\_length):

sentences = sent\_tokenize(text)

sentence\_scores = calculate\_sentence\_scores(sentences)

summary\_sentences = nlargest(max\_length, sentence\_scores, key=sentence\_scores.get)

summary = ' '.join(summary\_sentences)

return summary

def generate\_summaries(texts, max\_length=300):

summaries = []

for text in texts:

preprocessed\_text = preprocess\_text(text)

summary = extract\_summary(preprocessed\_text, max\_length)

summaries.append(summary)

return summaries

# Загрузка исходных текстовых данных из файла JSON

with open('input.json', 'r') as file:

input\_data = json.load(file)

# Генерация рефератов

generated\_summaries = generate\_summaries(input\_data)

# Сохранение рефератов в файл JSON

with open('output.json', 'w') as file:

json.dump(generated\_summaries, file)

**Контрольный тест**

Вывод данных осуществляется в файл output.json

**Вывод**

Удалось выполнить задачу резюмирования текста при помощи простого алгоритма и получить результат, который можно распространить на другие тексты при помощи библиотеки nltk.