

**Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой ИУ-4
(Индекс)
В.А. Шахнов
(И.О.Фамилия)
« ____ » _____ 2022 г.

**З А Д А Н И Е
на выполнение домашнего задания**

по дисциплине Системное программирование

Обработка потоков данных посредством Spark Streaming
(Тема домашнего задания)

Студент Трефилев Александр Алексеевич, ИУ4-22М
(Фамилия, имя, отчество, индекс группы)

График выполнения ДЗ: 25% к ____ нед., 50% к ____ нед., 75% к ____ нед., 100% к ____ нед.

1. Техническое задание

1) Написание программы для подсчета количества сообщений в Телеграм-каналах. 2)
Написание программы для поиска и подсчета имен собственных в сообщениях Телеграм-
каналов.

2. Оформление домашнего задания:

- 2.1. Расчетно-пояснительная записка на ____ листах формата А4.
2.2. Перечень графического материала (плакаты, схемы, чертежи и т.п.)

Дата выдачи задания « 11 » мая 2022 г.

Дата выполнения до « 25 » мая 2022 г.

Руководитель домашнего задания

(Подпись, дата)

С.Ю. Папулин
(И.О.Фамилия)

Студент

(Подпись, дата)

А.А. Трефилев
(И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится у преподавателя

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ4

« » (Индекс)

В.А. Шахнов

(И.О.Фамилия)

« » 2022 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения домашнего задания

Студент Трефилев Александр Алексеевич
(Фамилия, имя, отчество)
Обработка потоков данных посредством Spark Streaming
(Тема домашнего задания)

№ п/п	Наименование этапов домашнего задания*	Выполнение этапов		Отметка об исполнении
		Срок*	Модуль (Объем, %)	
1	Задание 1	14.05	30%	
2	Задание 2	21.05	60%	
3	Итоговое оформление ДЗ	25.05	100%	
	Защита	2 нед.	Оценка:	

* срок выполнения и аттестаций по модулям определяется учебным планом

Руководитель домашнего задания

С.Ю. Папулин

(подпись, дата)

(И.О.Фамилия)

Студент

А.А. Трефилев

(подпись, дата)

(И.О.Фамилия)

** итоговая оценка выставляется как средний бал между оценками аттестаций и оценкой, полученной на защите

АННОТАЦИЯ

Данная работа посвящена обработке потоков данных по средствам Spark Streaming. Spark Streaming используется для обработки непрерывных потоковых данных. В качестве брокера сообщений используется Kafka, для контроля синхронизации распределенных данных будем использовать Zookeeper.

В данном задании с использованием spark streaming требуется написать программу, которая будет считать количество сообщений из telegram – каналов, а также программу, которая будет считать количество имен собственных в сообщениях этих каналов.

Ключевые слова: Python, PySpark, Spark Streaming, Kafka, Zookeeper, Telegram

ABSTRACT

This paper focuses on the processing of data streams using Spark Streaming. Spark Streaming is used to process continuous streaming data. Kafka is used as message broker, and Zookeeper will be used to control synchronization of distributed data.

In this task, using spark streaming, we need to write a program that will count the number of messages from telegram channels, as well as a program that will count the number of proper names in the messages of these channels.

Keywords: Python, PySpark, Spark Streaming, Kafka, Zookeeper, Telegram

ОГЛАВЛЕНИЕ

АННОТАЦИЯ	1
СПИСОК УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СОКРАЩЕНИЙ И ТЕРМИНОВ.....	3
ВВЕДЕНИЕ	4
ЗАДАНИЕ 1	5
ЗАДАНИЕ 2.....	12

СПИСОК УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СОКРАЩЕНИЙ И ТЕРМИНОВ

SparkStreaming – библиотека фреймворка Apache Spark для обработки непрерывных потоковых данных, которая оперирует с дискретизированным потоком DStream, чей API базируется на отказоустойчивой структуре RDD.

Kafka – распределённый программный брокер сообщений, проект с открытым исходным кодом, разрабатываемый в рамках фонда Apache. Написан на языках программирования Java и Scala.

Zookeeper – сервис-координатор, который обеспечивает распределенную синхронизацию небольших по объему данных (конфигурационная информация, пространство имен) для группы приложений. Zookeeper представляет из себя распределенное хранилище ключ-значение (key-value store), гарантирующий надежное консистентное (consistency) хранение информации за счет синхронной репликации между узлами, контроля версий, механизма очередей (queue) и блокировок (lock). За счет использования оперативной памяти и масштабируемости обладает высокой скоростью.

ВВЕДЕНИЕ

Работа посвящена Spark Streaming — это библиотека фреймворка Apache Spark для обработки непрерывных потоковых данных, которая оперирует с дискретизированным потоком DStream, чей API базируется на отказоустойчивой структуре RDD.

Объектом исследования являются Телеграм – каналы.

Целью работы является подсчет сообщений от Телеграм-каналов и поиск и подсчет в сообщениях от них имен собственных.

Исходными данными для работы является список Телеграм-каналов.

Результатами работы являются программы на языке Python, выполняющие функционал, указанный в целях работы.

Структура и объем работы:

Работа представляет собой 2 задания, в которых описаны этапы разработки требуемых программ на языке Python и листинги их кода.

ЗАДАНИЕ 1

Дано: Напишите программу, которая подсчитывает количество сообщений каждого пользователя в течение 1 мин. и в течение 10 мин. каждые 30 сек (window). Выведите результат в отсортированном по убыванию количества. Формат вывода: username, количество. Названия каналов (usernames): ['tass_agency', 'interfaxonline', 'rian_ru', 'rt_russian', 'rentv_news', 'vestiru24', 'ntvnews', 'news_1tv'].

Для подключения к Телеграм-каналам используем библиотеку telethon. Также необходимо создать свое Телеграм-приложение для получения уникального id и hash (Рисунок 1).

App configuration

App api_id:	<input type="text" value="13803428"/>	🔒
App api_hash:	<input type="text" value="4784f44a9f0f1a436c92f263fcb47cfe"/>	🔒
App title:	<input type="text" value="sysproghomework3"/>	
Short name:	<input type="text" value="sysproghw3"/>	
	alphanumeric, 5–32 characters	

Available MTProto servers

Test configuration:	<input type="text" value="149.154.167.40:443"/>
	DC 2
Public keys:	<pre>-----BEGIN RSA PUBLIC KEY----- MIIBKgKCAQEAymEdY1aR+sCR3ZSJrtztKTKqigvO/vBfqACJLZtS7QMgCGXJ6 XIR yy7mx66w0/sOfa7/1mAZtEoIokDP3ShoqF4fVNB6XeqgQfaUHd8wJpDwHcR2O Fwv p1UUI1PLtkTz9uW2NE23b+ixNwJjJGwBDJPQEQFBE+vfmH0JP503wr5INS1po Wg/ j25sIWeyPHYeOrFp/eXaqhISP6G+q2IeTawTXpwZj4LzXq5YOpk4bYEQ6mvRq 7D1 aHwFym1EGepfaYR8Q0YqvvhYtMte3ITnuSJs171+GDqpdkcSwHnd6FudwG04p cCO j4WcDuXc2CTHgH8gFTNhp/Y8/SpD0hvn9QIDAQAB -----END RSA PUBLIC KEY-----</pre>
Production configuration:	<input type="text" value="149.154.167.50:443"/>
	DC 2

Рисунок 1 – Полученные api-id и api-hash

Далее необходимо запустить Zookeeper и Kafka, а также создать новый топик в Kafka. Листинги операций представлены на рисунках 2, 3.

```
ubuntu@linux:~$ /home/ubuntu/BigData/zookeeper/bin/zkServer.sh start
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /home/ubuntu/BigData/zookeeper/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
ubuntu@linux:~$ $KAFKA_HOME/bin/kafka-server-start.sh $KAFKA_HOME/config/server.
properties
[2022-06-05 12:12:16,584] INFO Registered kafka:type=kafka.Log4jController MBean
(kafka.utils.Log4jControllerRegistration$)
[2022-06-05 12:12:21,608] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotia
tion=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.co
mmon.X509Util)
[2022-06-05 12:12:22,066] INFO Registered signal handlers for TERM, INT, HUP (or
g.apache.kafka.common.utils.LoggingSignalHandler)
[2022-06-05 12:12:22,106] INFO starting (kafka.server.KafkaServer)
[2022-06-05 12:12:22,108] INFO Connecting to zookeeper on localhost:2181 (kafka.
server.KafkaServer)
```

Рисунок 2 – Запуск Zookeeper и Kafka

```
ubuntu@linux:~$ $KAFKA_HOME/bin/kafka-topics.sh --create --bootstrap-server loca
lhost:9092 --partitions 1 --replication-factor 1 --topic telegram
Created topic telegram.
ubuntu@linux:~$
```

Рисунок 3 – Создание топика telegram в Kafka

Далее начнем писать программу для подключения и подсчета количества сообщений от Телеграм-каналов в течение 1 минуты (window), используя библиотеку telethon. Для подключения к каналам используем экземпляр класса JoinChannelRequest, для получения сообщений каналов - events.NewMessage, для получения наименований каналов - метод client.get_entity. Для полноценного функционирования программы необходимо создать 2 скрипта, один из которых будет подключаться к каналам (producer) и посылать в Kafka полученные сообщения, и скрипт, который будет обрабатывать сообщения, считая количество сообщений по каналам и ведя поиск имен собственных (consumer). Листинг скриптов представлен в таблицах 1 и 2.

Таблица 1 – Листинг скрипта Producer

```
import time
from telethon.tl.functions.channels import JoinChannelRequest
from telethon import TelegramClient, events
from kafka import KafkaProducer
api_id = 13803428
api_hash = '4784f44a9f0f1a436c92f263fcb47cfe'
# Kafka topic name
TOPIC_NAME = "telegram11"
# Kafka server
KAFKA_HOST = "localhost"
KAFKA_PORT = "9092"
# Here you define the target channel that you want to listen to:
channels = ['https://t.me/tass_agency', 'https://t.me/rt_russian', 'https://t.me/interfaxonline',
            'https://t.me/vestiru24',
            'https://t.me/rentv_news', 'https://t.me/ntvnews', 'https://t.me/news_1tv', 'https://t.me/rian_ru']
client = TelegramClient('tref', api_id, api_hash)
class KafkaCommunicator:
    def __init__(self, producer, topic):
        self.producer = producer
        self.topic = topic
    def send(self, message):
        self.producer.send(self.topic, message.encode("utf-8"))
    def close(self):
        self.producer.close()
def create_communicator():
    """Create Kafka producer."""
    producer = KafkaProducer(bootstrap_servers=KAFKA_HOST + ":" + KAFKA_PORT)
    return KafkaCommunicator(producer, TOPIC_NAME)
communicator = create_communicator()
# Listen to messages from target channel
def main():
    @client.on(events.NewMessage(channels))
    async def newMessageListener(event):
        for channel in channels:
            await client(JoinChannelRequest(channel))
            time.sleep(5)
            message = event.message.message
            sender = await event.get_sender()
            username = sender.username
            messages = {'message': message, 'username': username}
            print(messages)
            communicator.send(str(messages))
    with client:
        client.run_until_disconnected()
if __name__ == '__main__':
    main()
```

Таблица 2 – Листинг скрипта Consumer для подсчета сообщений с window 1 минута

```

from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils
SPARK_APP_NAME = "KafkaWordCount"
SPARK_CHECKPOINT_TMP_DIR = "file:///home/ubuntu/spark3/output"
SPARK_BATCH_INTERVAL = 30
SPARK_LOG_LEVEL = "OFF"
KAFKA_BOOTSTRAP_SERVERS = "localhost:9092"
KAFKA_TOPIC = "telegram11"
def update_total_count(current_count, count_state):
    if count_state is None:
        count_state = 0
    return sum(current_count, count_state)
def create_streaming_context(SPARK_BATCH_INTERVAL):
    sc = SparkContext(appName=SPARK_APP_NAME)
    sc.setLogLevel(SPARK_LOG_LEVEL)
    ssc = StreamingContext(sc, SPARK_BATCH_INTERVAL)
    ssc.checkpoint(SPARK_CHECKPOINT_TMP_DIR)
    return ssc
def create_stream(ssc):
    return (
        KafkaUtils.createDirectStream(
            ssc, topics=[KAFKA_TOPIC],
            kafkaParams={"bootstrap.servers": KAFKA_BOOTSTRAP_SERVERS})
        .map(lambda x: x[1])
    )
def getterUser(info):
    return eval(info)["username"]
def main():
    ssc = create_streaming_context(SPARK_BATCH_INTERVAL)
    messages = create_stream(ssc)
    total_counts_sorted = (
        messages
        .map(getterUser)
        .map(lambda word: (word, 1))
        .reduceByKeyAndWindow(lambda x, y: x + y, lambda x, y: x - y, 60, 30)
        .transform(lambda x_rdd: x_rdd.sortBy(lambda x: -x[1]))
    )
    total_counts_sorted.pprint()
    ssc.start()
    ssc.awaitTermination()
if __name__ == "__main__":
    main()

```

Результатом работы Producer является сообщение с id канала в виде словаря (рисунок 4), выведенное в консоль для визуального представления результата работы, а также с помощью communicator.send() отправка этих сообщений в Kafka.

```
(venv) ubuntu@linux:~/spark3$ python3 producer.py
Please enter your phone (or hot token): 89258641231
Please enter the code you received: 9836
Signed in successfully as Sema
{"message": "Ситуация на Украине складывается не в пользу Киева, поэтому украинцы должны как можно скорее начать переговоры.\n\nОб этом заявил бывший замруководителя Европейского командования ВС США генерал-лейтенант Стэнли Тинтти. Он добавил, что у России «ничего больше боевая мощь».\n\nЧем дольше будет продолжаться конфликт, тем слабее будет становиться позиция Украины на будущих переговорах. Если Киеву будет нечего предложить, то он может потерять по итогу немного больше, чем планировал», — цитирует Тинтти РИА Новости.\n\nОб необходимости для украинских властей пойти на уступки РФ ранее «говарил экс-госсекретарь США Киссинджер. После этого его внесли в базу сайта «Миротворец».\n\n@rt_russian", "username": "rt_russian"}
{"message": "Жители Донецка делятся в соцсетях кадрами последствий обстрела со стороны ВСУ. На них видно, что во дворах горят автомобили, улицы сильно задымлены.\n\n@rentv_news", "username": "rentv_news"}
{"message": "", "username": "rentv_news"}
{"message": "", "username": "rentv_news"}
{"message": "Власть дал интервью Боснийско-сербской телекомпания. Главное: \n\n* учитывая цены на нефть, бюджет России не понесёт никаких потерь из-за ограничений на поставки в ЕС\n\n* РФ в 2022 г существенно увеличила прибыль, получаемую от экспорта своих энергоносителей \n\n* РФ увеличила продажи нефти на альтернативных рынках\n\n* нельзя допустить, чтобы какие-то критические сферы жизнедеятельности РФ зависели от инвестиций и технологий из Запада\n\n* Запад не разрешает Украине вести переговоры с Россией.", "username": "rian_ru"}
{"message": "Сегодня весь день из Донецка и близлежащих населённых пунктов приходило сообщение о массированных обстрелах. Украинские нацисты обрушили на головы мирных жителей настоящий град из ракет, мины и снарядов. Огонь очень плотный и неизбежный, пострадали множество малых домов, учебное заведение, больницы. А один снаряд угодил даже в автомобиль скорой помощи. Есть погибшие и пострадавшие: https://www.itv.ru/r/306461", "username": "news_itv"}
{"message": "Первые кадры последствий украинского обстрела центра Донецка", "username": "rian_ru"}
{"message": "", "username": "rian_ru"}
{"message": "", "username": "rian_ru"}
```

Рисунок 4 – Результат работы Producer

Результатом работы Consumer является отображение в консоли batch, которые обновляются каждые 30 секунд (window) и отображают количество сообщений и канал (рисунок 5).

```
Terminal: producer x consumer x + v
ubuntu@linux:~/spark3$ /home/ubuntu/BigData/spark/bin/spark-submit --master local[2] --packages org.apache.spark:spark-streaming-kafka-0-8_2.11:2.0.2 consumer10.py
```

```
Terminal: producer x consumer x + v
-----
Time: 2022-06-04 18:48:00
-----
('news_itv', 1)
-----
Time: 2022-06-04 18:48:30
-----
('news_itv', 1)
-----
Time: 2022-06-04 18:49:00
-----
-----
Time: 2022-06-04 18:49:30
-----
('rian_ru', 3)
-----
Time: 2022-06-04 18:50:00
-----
('rian_ru', 3)
-----
Time: 2022-06-04 18:50:30
-----
-----
Time: 2022-06-04 18:51:00
-----
```

Рисунок 5 – Запуск и результат работы Consumer

Аналогично напишем скрипт Consumer для подсчета количества слов в течение 10 минут с window 30 секунд. Скрипт для Producer не изменяется. Скрипт Consumer представлен в таблице 3.

Таблица 3 - Листинг скрипта Consumer для подсчета сообщений с window 10 минут

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils
SPARK_APP_NAME = "KafkaWordCount"
SPARK_CHECKPOINT_TMP_DIR = "file:///home/ubuntu/spark3/output"
SPARK_BATCH_INTERVAL = 30
SPARK_LOG_LEVEL = "OFF"
KAFKA_BOOTSTRAP_SERVERS = "localhost:9092"
KAFKA_TOPIC = "telegram8"
def create_streaming_context(SPARK_BATCH_INTERVAL):
    sc = SparkContext(appName=SPARK_APP_NAME)
    sc.setLogLevel(SPARK_LOG_LEVEL)
    ssc = StreamingContext(sc, SPARK_BATCH_INTERVAL)
    ssc.checkpoint(SPARK_CHECKPOINT_TMP_DIR)
    return ssc
def create_stream(ssc):
    return (
        KafkaUtils.createDirectStream(
            ssc, topics=[KAFKA_TOPIC],
            kafkaParams={"bootstrap.servers": KAFKA_BOOTSTRAP_SERVERS})
        .map(lambda x: x[1])
    )
def getter(info):
    return eval(info)["username"]
#def natasha():
def main():
    ssc = create_streaming_context(SPARK_BATCH_INTERVAL)
    messages = create_stream(ssc)
    total_counts_sorted = (
        messages
        .map(getter)
        .map(lambda word: (word, 1))
        .reduceByKeyAndWindow(lambda x, y: x + y, lambda x, y: x - y, 600, 30)
        .transform(lambda x_rdd: x_rdd.sortBy(lambda x: -x[1]))
    )
    total_counts_sorted.pprint()
    ssc.start()
    ssc.awaitTermination()
if __name__ == "__main__":
    main()
```

Основным различием является изменение длины окна в `reduceByKeyAndWindow` – длина окна (window) выставляется равной 600 секунд. Результат подсчета сообщений за 10 минут показаны на рисунках 6 и 7.

```
2022-06-04 19:45:39,193 INFO storage.BlockManager: Initialized BlockManager: BlockManagerId(driver, linux, 35785, None)
2022-06-04 19:45:39,570 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@3e5a11de{/metrics/json,null,AVAILABLE,@Spark}
-----
Time: 2022-06-04 19:46:00
-----
-----
Time: 2022-06-04 19:46:30
-----
-----
Time: 2022-06-04 19:47:00
-----
('tass_agency', 1)
('rian_ru', 1)
-----
Time: 2022-06-04 19:47:30
-----
('tass_agency', 1)
('rian_ru', 1)
-----
Time: 2022-06-04 19:48:00
-----
('tass_agency', 1)
('rian_ru', 1)
```

Рисунок 6 – Запуск и первые batch

```
Terminal: producer x consumer x + v
-----
Time: 2022-06-04 19:56:00
-----
('vestiru24', 1)
('tass_agency', 1)
('rian_ru', 1)
('rt_russian', 1)
('rentv_news', 1)
-----
Time: 2022-06-04 19:56:30
-----
('rentv_news', 2)
('vestiru24', 1)
('tass_agency', 1)
('rian_ru', 1)
('rt_russian', 1)
-----
Time: 2022-06-04 19:57:00
-----
('rentv_news', 2)
('vestiru24', 1)
('rt_russian', 1)
('tass_agency', 0)
('rian_ru', 0)
```

Рисунок 7 – Результат подсчета сообщений за 10 минут и начало нового окна

ЗАДАНИЕ 2

Дано: Напишите программу, которая подсчитывает количество собственных имен существительных в сообщениях Телеграм-каналов (см. задание 1) в течение 1 мин. с накоплением. Каждую 1 мин. сохранять результат в файл в отсортированном по убыванию количества виде. Накапливать не менее 30 мин. При завершении программы вывести в консоль 10 наиболее популярных слов. Формат вывода: слово, количество. Опционально можно удалять первые слова в приложении, добавить фильтрацию на стоп-слова, также необходимо нормализовать слова.

Первые слова полученных сообщениях удалять не будем, так как по наблюдениям в большинстве случаев это слова – искомые имена существительные. Реализуем проверку на стоп-слова, а нормализацию проведем с помощью библиотеки `ru morphology2` и ее методов.

Листинг `Producer` не изменяется, в скрипт `Consumer` добавляется загрузка стоп-слов, функция для нормализации, а также реакция на завершение программы от пользователя `KeyboardInterrupt`, после которого будет выведены топ-10 часто попадающихся найденных имен-собственных. Листинг представлен в таблице 4.

Таблица 4 – Листинг `Consumer` для поиска и подсчета имен-собственных

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils
import pymorphy2
import glob
import os.path
SPARK_APP_NAME = "KafkaWordCount"
SPARK_CHECKPOINT_TMP_DIR = "file:///home/ubuntu/spark3/output"
SPARK_BATCH_INTERVAL = 30
SPARK_LOG_LEVEL = "OFF"
KAFKA_BOOTSTRAP_SERVERS = "localhost:9092"
KAFKA_TOPIC = "telegram11"
with open('stop-words-russian.txt') as f:
    stopwords = f.read()
def create_streaming_context(SPARK_BATCH_INTERVAL):
    sc = SparkContext(appName=SPARK_APP_NAME)
    sc.setLogLevel(SPARK_LOG_LEVEL)
    ssc = StreamingContext(sc, SPARK_BATCH_INTERVAL)
    ssc.checkpoint(SPARK_CHECKPOINT_TMP_DIR)
    return ssc
def update_total_count(current_count, count_state):
    if count_state is None:
        count_state = 0
    return sum(current_count, count_state)
```

```

def create_stream(ssc):
    return (
        KafkaUtils.createDirectStream(
            ssc, topics=[KAFKA_TOPIC],
            kafkaParams={"bootstrap.servers": KAFKA_BOOTSTRAP_SERVERS})
        .map(lambda x: x[1])
    )
def getterUser(info):
    return eval(info)["username"]
def getterMessage(info):
    return eval(info)["message"]
def fstopwords(word):
    if word.lower() not in stopwords:
        return True
    else:
        return False
def normalize(word):
    if fstopwords(word):
        morph = pymorphy2.MorphAnalyzer()
        word = word.strip("\n\n").strip(".").strip(",").strip(":")
        if word[0].isupper():
            word = morph.parse(word)[0]
            word = word.normal_form
            word = word.capitalize()
            return word
        else:
            return None
    else:
        return None
def main():
    try:
        ssc = create_streaming_context(60)
        messages = create_stream(ssc)
        total_counts_sorted = (
            messages
            .map(getterMessage)
            .flatMap(lambda line: line.split())
            .map(normalize)
            .filter(lambda x: x is not None)
            .map(lambda word: (word, 1))
            .reduceByKey(lambda x, y: x + y)
            .updateStateByKey(update_total_count)
            .transform(lambda x_rdd: x_rdd.sortBy(lambda x: -x[1]))
            .saveAsTextFiles('file:///home/ubuntu/spark3/upperW/upperW')
        )
        ssc.start()
        ssc.awaitTermination()
    except KeyboardInterrupt:
        folder_path = r'/home/ubuntu/spark3/upperW'
        files = glob.glob(folder_path + '/upperW-*' + '/part-00000')
        max_file = max(files, key=os.path.getctime)
        with open(max_file, 'r') as myfile:
            words = myfile.read().split('\n')
            print("Top 10 used words:")
            for x in range(10):
                print(words[x])

```

```
if __name__ == "__main__":  
    main()
```

Результатом работы является файл, где хранятся слова, а также они выводятся на консоль (рисунок 8).

The screenshot shows an IDE with a project explorer on the left containing folders like 'upperW-165443490000' and files like 'part-00000'. The main editor displays a list of words and their counts, such as ('Европа', 4), ('Эрдоган', 3), ('Турция', 3), ('Президент', 2), ('Тайип', 2), ('Сша', 2), ('Венесуэла', 2), ('Украина', 2), ('Роман', 2), ('Мариуполь', 1), ('Корреспондент', 1), ('Литвинов', 1), ('Рудник', 1), ('Сила', 1), ('Жительница', 1), ('Нижний', 1), ('Диснеевский', 1), ('Замок', 1), ('Выхватить', 1), ('Диснейленд', 1), and ('Швеция', 1). The terminal window at the bottom shows the command 'cat part-00000' and the output of the file, which matches the list in the editor.

```
2022-06-05 12:52:33,232 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@61e65fa4{/metrics/json,null,AVAILABLE,@Spark}  
^C  
Top 10 used words:  
(('Европа', 4)  
(('Эрдоган', 3)  
(('Турция', 3)  
(('Президент', 2)  
(('Тайип', 2)  
(('Сша', 2)  
(('Венесуэла', 2)  
(('Украина', 2)  
(('Роман', 2)  
(('Мариуполь', 1)  
(venv) ubuntu@linux:~/spark3$
```

Рисунок 8 – Результат работы программы поиска и подсчета имен собственных в течение 30 минут.