Н И Т У « М И С и С »

ОТЧЁТ

ПО

ЛАБОРАТОРНОЙ РАБОТЕ

«Код Хемминга»

Учебная дисциплина «Глобальные сети сети»

Группа: БПМ-16-2

Студент: Новицкий Дмитрий

Вариант: 12

Преподаватель: доц., к.т.н. Курочкин И.И.

Отметка:

Дата защиты:

2019 г.

Оглавление

Задание

1. С помощью сокетов реализовать передачу сообщений по локальной сети. Для определения хоста назначения необходимо использовать IP-адрес и порт (к примеру, 192.168.123.234:45102).
2. Закодировать передаваемое сообщение кодом Хемминга с определенной длиной слова. (длина слова – 53)
3. Иметь возможность добавлять в сообщение не более n ошибок на каждое слово (этот функционал должен быть реализован на стороне передачи сообщения).
4. Раскодировать сообщение, исправить одиночные ошибки и определить наличие множественных ошибок (>1) в словах.
5. Обратно передать сообщение с информацией о количестве исправленных ошибок, количество правильно и неправильно доставленных слов. Сравнить эту информацию с внесенными ошибками из п.3.

Сообщение должно быть длиной порядка 2000-3000 знаков и пробелов и содержать цифры, латинские и кириллические символы, знаки препинания. (Возьмите русскую статью с английскими терминами из Wikipedia, habr.com и т.д.)

Во время отладки Вы можете использовать специальный диапазон адресов 127.0.0.0/8

Предусмотреть возможность повторного запуска приложения с тем же портом.

Для проверки корректности работы нужно, как минимум, 3 раза отправить сообщение:

1. Без ошибок
2. С возможными ошибками (не более 1 на слово)
3. С множественными ошибками (более 1 на слово, но не обязательно во всех словах)

Загрузка в LMS в одном архиве:

1. Исходных кодов приложения/приложений;
2. Исполняемых файлов с необходимым окружением;
3. Мини-отчет в формате PDF с текстом сообщения, со снимками экрана, пояснениями, настройками и результатами.

Демонстрация лабораторной работы на семинаре на 2 разных ПК/ноутбуках, находящихся в одной локальной сети. Для демонстрации на семинаре могут быть использованы только материалы, предварительно загруженные в LMS.

Алгоритм работы программы

Всего файлов для работы программы 4:

- Рабочий файл со стороны отправления сообщения
- Функции к рабочему файлу со стороны отправления сообщения
- Рабочий файл со стороны приёма сообщения
- Функции к рабочему файлу со стороны приёма сообщения

Алгоритм передачи сообщения

Изначально задаётся путь к тексту, который необходимо отправить. Затем, с помощью метода get_encoging(path) определяем кодировку текстового файла. После чего считываем файл с заданной кодировкой.

```
# Часть 1. Считывание текста с файла
print("Часть 1. Считывание текста с файла")

path_1 = "texts/Задания.txt" # utf8
path_2 = "texts/План работы.docx" # Неизвестная кодировка
path_3 = "texts/Искусственная нейронная сеть.txt"
path_4 = "texts/My_text.txt"

# Определяем кодировку текста в файле
encoding = functions.get_encoding(path_4)
en = encoding['encoding']

if(print_information):
    print("Кодировка файла - ", en)

# Считываем текст с заданной кодировкой
file_text = functions.read_file(path_4, en) # Исходный файл

if(print_information):
    print("file =")
    print(file_text)
    print()

print("Считывание текста с файла окончено")
print()
```

Затем с помощью полученной ранее кодировки переводим каждый символ текста в список десятичных чисел, в котором каждое число соответствует заданному символу заданной кодировке.

```
# Часть 2. Переводим текст в список цифр в десятичной СС
print("Часть 2. Переводим текст в список цифр в десятичной СС")

file_code = list(file_text.encode(en))

if(print_information):
    print("file_text = ")
    print(file_code)
    print()

print("Перевод текста в список цифр в десятичной СС окончен")
print()
```

Далее из полученного списка составляем список двоичных значений полученных ранее десятичных чисел.

```
# Часть 3. Переводим текст из списка цифр в десятичной СС в список цифр в двоичной СС
```

```
print("Часть 3. Переводим текст из списка цифр в десятичной СС в список цифр в двоичной СС")
# Поиск максимального числа
max_length = 0
for i in range(len(file_code)):
    if(file_code[i] > max_length):
        max_length = file_code[i]

if(print_information):
    print("max_length = ")
    print(max_length)
    print()

# Поиск длины слова в доичном виде
word_length = math.ceil(math.log(max_length, 2))

if(print_information):
    print("word_length = ")
    print(word_length)
    print()

# Преобразование в двоичную СС
bit_mas = functions.get_bit_mas(file_code, word_length)

if(print_information):
    print("bit_mas = ")
    print(bit_mas)
    print("len(bit_mas) = ", len(bit_mas))

print("Перевод текста из списка цифр в десятичной СС в список цифр в двоичной СС окончен")
print()
```

Применяем к данному списку алгоритм кода Хемминга.

```
# Часть 4. Применяем кодировку Хемминга
print("Часть 4. Применяем кодировку Хемминга")
start_time = time.time()
# Длина слова Хемминга до вставки контрольных битов
length_hamming_word_input = 53
# Длина слова Хемминга после вставки контрольных битов
length_hamming_word_output = length_hamming_word_input +
math.floor(math.log(length_hamming_word_input, 2)) + 1

Hamming_mas = functions.Hamming_code(bit_mas, length_hamming_word_input, 2)

if(print_information):
    print("Hamming_mas = ")
    print(Hamming_mas)
    print()

    print("Длина Hamming_mas = ")
    print(len(Hamming_mas))
    print()

end_time = time.time()
print("Применение кодировки Хемминга окончено")
print("Время, потраченное на эту часть равно ", end_time - start_time, "секунд")
print()
```

Затем отправляем данные на заданный ip-адрес и заданные порты компьютера (ноутбука).

```
# Отправляем исходный текст
sock_1 = socket.socket()
#sock_1.connect(('192.168.0.102', 9090))
sock_1.connect(('127.0.0.1', 9090))
```

```
sock_1.send(bytes(Hamming_mas))
sock_1.close()

# Отправляем значение длины слова для кода Хемминга
sock_2 = socket.socket()
#sock_2.connect(('192.168.0.102', 49100))
sock_2.connect(('127.0.0.1', 49100))
sock_2.send(bytes(functions.dec_bin(length_hamming_word_input)))
sock_2.close()

# Отправляем значение кодировки исходного текста
# Преобразуем данные в необходимый для отправки вид
en_code = list(en.encode('utf8'))
en_bit_mas = functions.get_bit_mas(en_code, 16)

if(print_information):
    print("en = ", en)
    print("type(en) = ", type(en))
    print("en_code = ", en_code)
    print("en_bit_mas = ", en_bit_mas)

sock_3 = socket.socket()
#sock_3.connect(('192.168.0.102', 49101))
sock_3.connect(('127.0.0.1', 49101))
sock_3.send(bytes(en_bit_mas))
sock_3.close()

# Принимаем значение длины слова для кодировки
sock_4 = socket.socket()
#sock_4.connect(('192.168.0.102', 49102))
sock_4.connect(('127.0.0.1', 49102))
sock_4.send(bytes(functions.dec_bin(word_length)))
sock_4.close()

print("Отправка завершена")
print()
```

После чего принимаем информацию о количестве верно и неверно принятых слов.

```
# Часть 6. Приём ответной информации о переданном файле

print("Часть 6. Ожидаем приёма ответной информации о переданном файле")
start_time = time.time()

# Принимаем информацию о количестве правильно принятых слов
sock_1 = socket.socket()
#sock_1.bind(('192.168.0.103', 49110))
sock_1.bind(('127.0.0.1', 49110))
sock_1.listen(1)
conn_1, addr_1 = sock_1.accept()

count_of_true_words = functions.bin_dec(list(conn_1.recv(1024)))

conn_1.close()

# Принимаем информацию о количестве неправильно принятых слов
sock_2 = socket.socket()
#sock_2.bind(('192.168.0.103', 49111))
sock_2.bind(('127.0.0.1', 49111))
sock_2.listen(1)
conn_2, addr_2 = sock_2.accept()

count_of_false_words = functions.bin_dec(list(conn_2.recv(1024)))

conn_2.close()
```

```
# Принимаем информацию о количестве исправленных слов
sock_3 = socket.socket()
#sock_3.bind(('192.168.0.103', 49112))
sock_3.bind(('127.0.0.1', 49112))
sock_3.listen(1)
conn_3, addr_3 = sock_3.accept()

count_of_corrected_words = functions.bin_dec(list(conn_3.recv(1024)))

conn_3.close()

# Принимаем информацию о количестве неисправленных слов
sock_4 = socket.socket()
#sock_4.bind(('192.168.0.103', 49113))
sock_4.bind(('127.0.0.1', 49113))
sock_4.listen(1)
conn_4, addr_4 = sock_4.accept()

count_of_uncorrected_words = functions.bin_dec(list(conn_4.recv(1024)))

conn_4.close()

end_time = time.time()
print("Данные приянты")
print("Время ожидания отклика составило", end_time - start_time, "секунд")
print()

# Вывод на экран полученных данных
if(print_information):
    print("Количество правильно переданных слов -", count_of_true_words)
    print("Количество неправильно переданных слов -", count_of_false_words)
    print("Количество исправленных слов -", count_of_corrected_words)
    print("Количество неисправленных слов -", count_of_uncorrected_words)
else:
    file = open("texts/accepted_text_information.txt", "w")
    s = "Количество правильно переданных слов - " + str(count_of_true_words) + "\n"
    s = s + "Количество неправильно переданных слов - " + str(count_of_false_words) +
"\n"
    s = s + "Количество исправленных слов - " + str(count_of_corrected_words) + "\n"
    s = s + "Количество неисправленных слов - " + str(count_of_uncorrected_words)
    file.write(s)
    file.close()
```

Алгоритм приёма сообщения

Принимаем отправленное сообщение.

```
# Часть 1. Работа с сокетами (приём данных)

print("Ожидаем приёма текста")
start_time = time.time()

# Принимаем исходный текст
Hamming_mas = []

sock_1 = socket.socket()
#sock_1.bind(('192.168.0.103', 9090))
sock_1.bind(('127.0.0.1', 9090))
sock_1.listen(1)
conn_1, addr_1 = sock_1.accept()

while True:
    data = list(conn_1.recv(1024))
    for i in range(len(data)):
        Hamming_mas.append(data[i])
    if not data:
```

```
        break

conn_1.close()

# Принимаем значение длины слова для кода Хемминга
sock_2 = socket.socket()
#sock_2.bind(('192.168.0.103', 49100))
sock_2.bind(('127.0.0.1', 49100))
sock_2.listen(1)
conn_2, addr_2 = sock_2.accept()

length_hamming_word_input = functions.bin_dec(list(conn_2.recv(1024)))

conn_2.close()

# Принимаем значение кодировки исходного текста
sock_3 = socket.socket()
#sock_3.bind(('192.168.0.103', 49101))
sock_3.bind(('127.0.0.1', 49101))
sock_3.listen(1)
conn_3, adrr_3 = sock_3.accept()

encoding = functions.get_ascii_code((conn_3.recv(1024)), 16)
encoding = functions.get_text(encoding, 16, 'utf8')

conn_3.close()

# Принимаем значение длины слова для кодировки
sock_4 = socket.socket()
#sock_4.bind(('192.168.0.103', 49102))
sock_4.bind(('127.0.0.1', 49102))
sock_4.listen(1)
conn_4, adrr_4 = sock_4.accept()

word_length = functions.bin_dec(list(conn_4.recv(1024)))

end_time = time.time()
print("Данные приняты")
print("Время ожидания составило", end_time - start_time, "секунд")
print()
```

Затем проверяем и расшифровываем текст.

```
# Часть 3. Проверка и расшифровка текста
print("Часть 3. Проверка и расшифровка текста")
start_time = time.time()
# Обнаруживаем ошибки и исправляем, если они есть
bit_mas_new, count_of_true_words, count_of_false_words, count_of_corrected_words,
count_of_uncorrected_words = functions.Repeat_Hamming_code(Hamming_mas,
length_hamming_word_input)

# Переводим текст в список цифр в десятичной СС
file_code_2 = functions.get_ascii_code(bit_mas_new, word_length)

if(print_information):
    print("file_code_2 = ")
    print(file_code_2)
    print()

max = 0
for i in range(len(file_code_2)):
    if(file_code_2[i] > max):
        max = file_code_2[i]

if(print_information):
```

```
    print("max = ", max)
    print()

# Декодируем исходное сообщение
try:
    text = bytes(file_code_2)
    text = text.decode(encoding)
except UnicodeDecodeError:
    print("Из-за проблем с кодировкой расшифровать сообщение, в котором хотя бы в одном
слове более двух ошибок, невозможно.")
    print("Мне очень жаль.")
    count_of_true_words = 0
    count_of_false_words = 0
    count_of_corrected_words = 0
    count_of_uncorrected_words = 0
    text = ""

if(print_information):
    print("text = ")
    print(text)
    print()
else:
    file = open("texts/accepted text.txt", "w", encoding = encoding)
    file.write(text)
    file.close()

print("Проверка и расшифровка текста окончена")
print("Время проверки и расщифровки составило", end_time - start_time, "секунд")
print()
```

После чего отправляем информацию о принятом текстовом файле.

```
# Часть 4. Отправка информации о принятом файле
print("Часть 4. Отправка информации о принятом файле")
# Отправляем информацию о количестве правильно принятых слов
sock_1 = socket.socket()
#sock_1.connect(('192.168.0.102', 49110))
sock_1.connect(('127.0.0.1', 49110))
sock_1.send(bytes(functions.dec_bin(count_of_true_words)))
sock_1.close()

# Отправляем информацию о количестве неправильно принятых слов
sock_2 = socket.socket()
#sock_2.connect(('192.168.0.102', 49111))
sock_2.connect(('127.0.0.1', 49111))
sock_2.send(bytes(functions.dec_bin(count_of_false_words)))
sock_2.close()

# Отправляем информацию о количестве исправленных слов
sock_3 = socket.socket()
#sock_3.connect(('192.168.0.102', 49112))
sock_3.connect(('127.0.0.1', 49112))
sock_3.send(bytes(functions.dec_bin(count_of_corrected_words)))
sock_3.close()

# Отправляем информацию о количестве неисправленных слов
sock_4 = socket.socket()
#sock_3.connect(('192.168.0.102', 49113))
sock_4.connect(('127.0.0.1', 49113))
sock_4.send(bytes(functions.dec_bin(count_of_uncorrected_words)))
sock_4.close()

print("Отправка завершена")
```

Проверка работы программы

Проверим работу программы при передаче сообщения без ошибок, с не более, чем одной ошибкой и не более, чем с двумя ошибками.

Проверку будем осуществлять на двух файлах:

- Собственноручно написанный небольшой текстовый файлик
- Статья с Википедии об искусственных нейронных сетях

Собственноручно написанный файлик

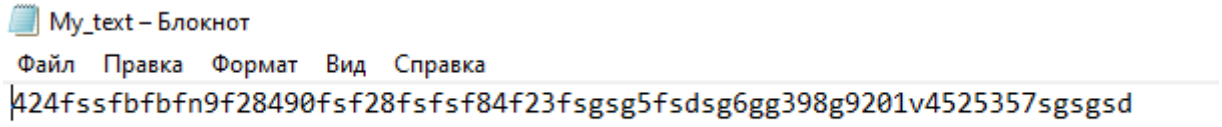

Рис. 1. Мой текст.

Передача файла без ошибок

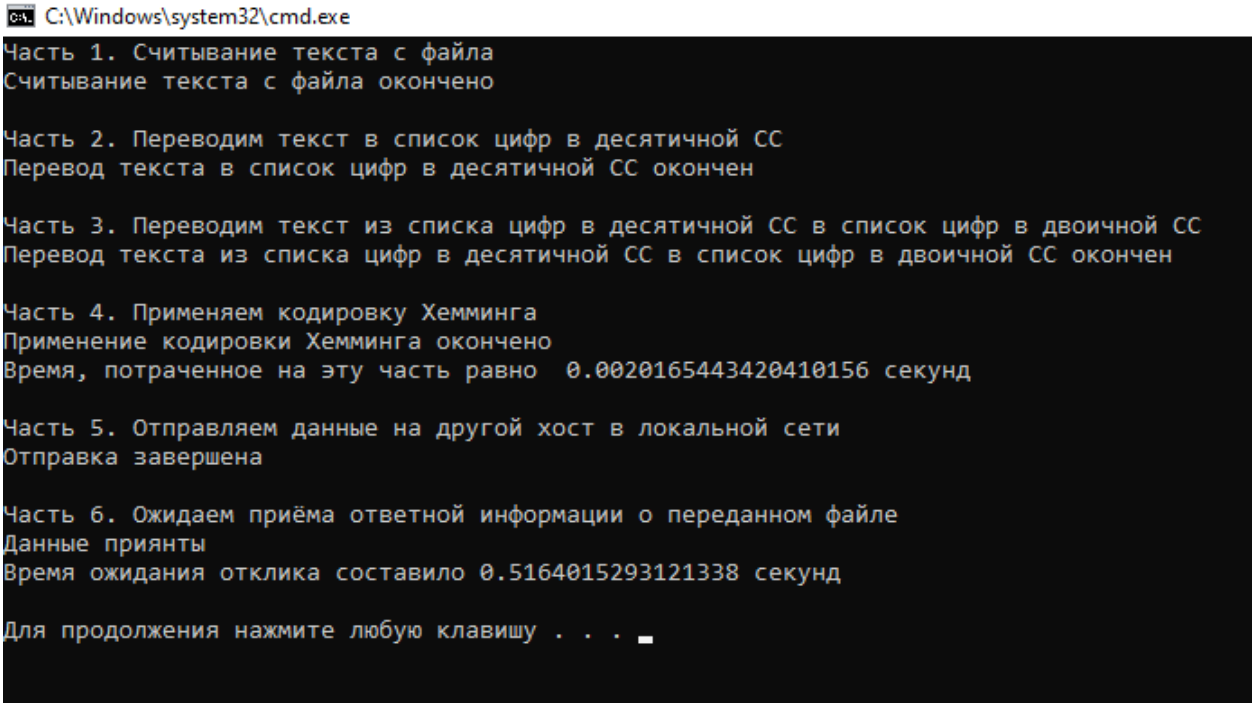

Рис. 2. Результаты работы на стороне передачи.

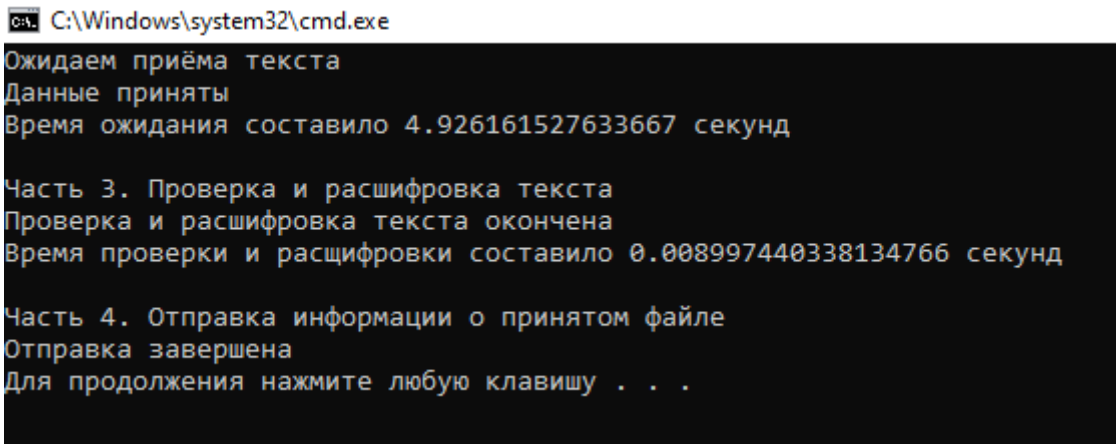

Рис. 3. Результаты работы на стороне приёма.

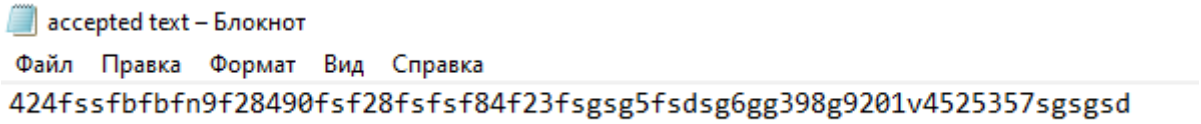

Рис. 4. Принятый текст.

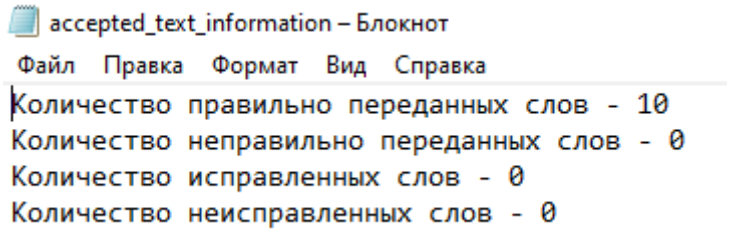

Рис. 5. Информация о переданном тексте.

Передача информации с не более, чем одной ошибкой в слове

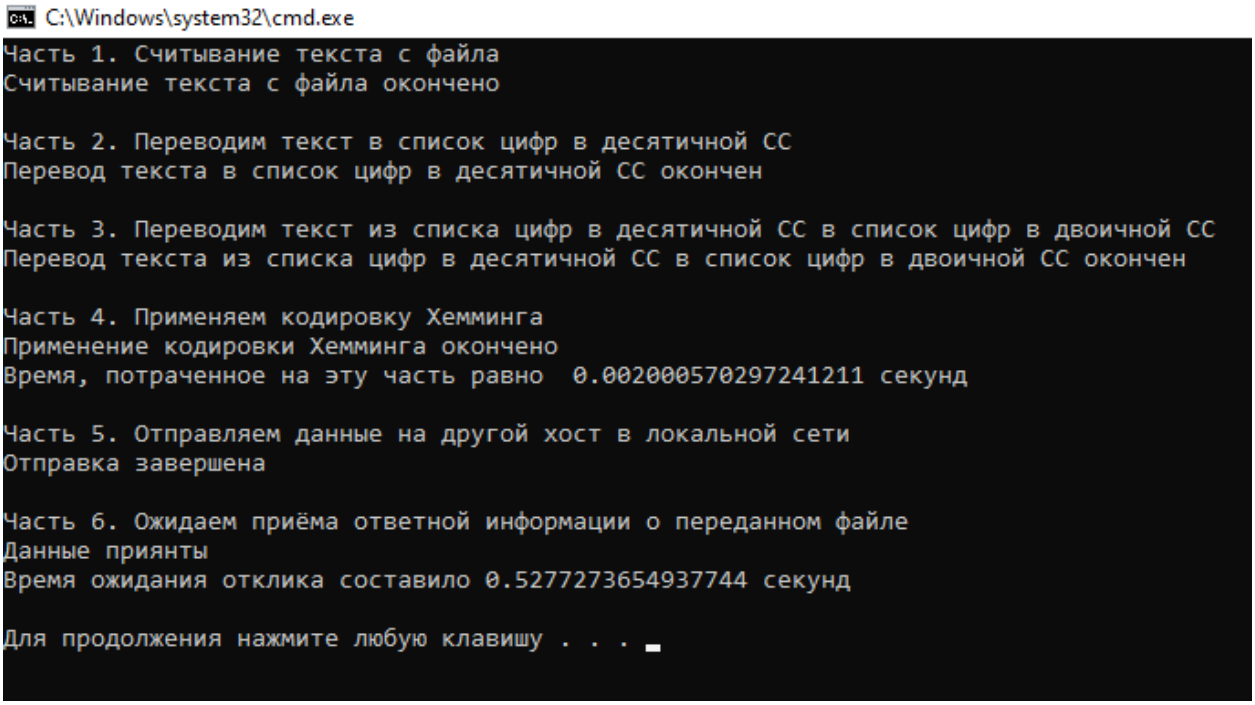

Рис. 6. Результаты работы на стороне передачи.

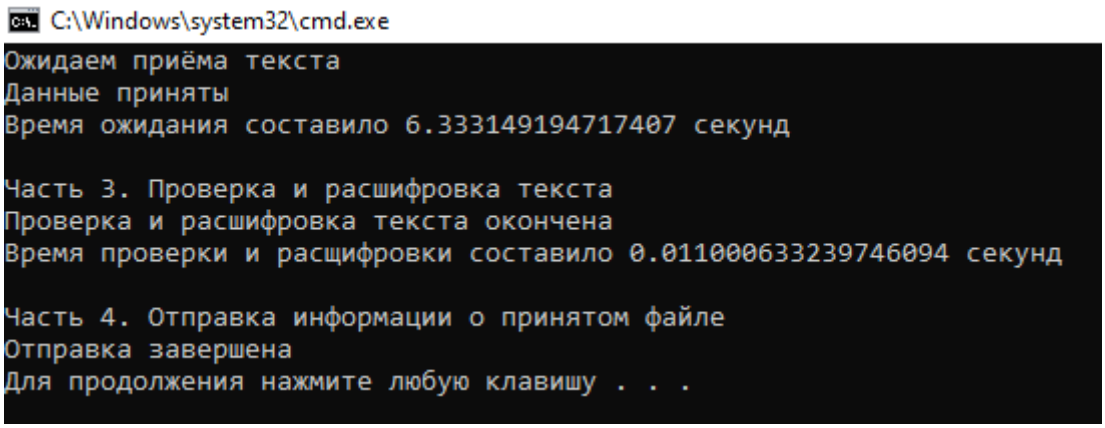

Рис. 7. Результаты работы на стороне приёма.

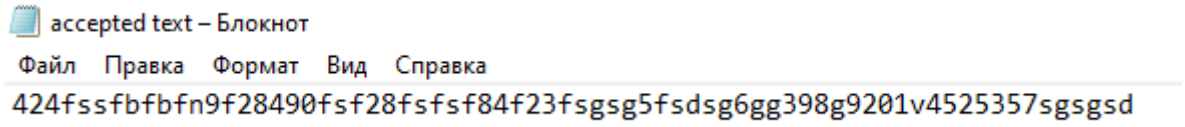

Рис. 8. Принятый текст.

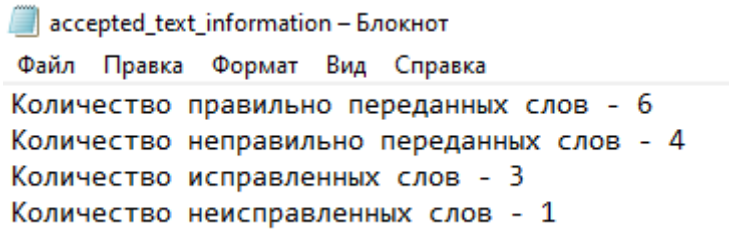

Рис. 9. Информация о переданном тексте.

Передача файла с не более, чем двумя ошибками в слове

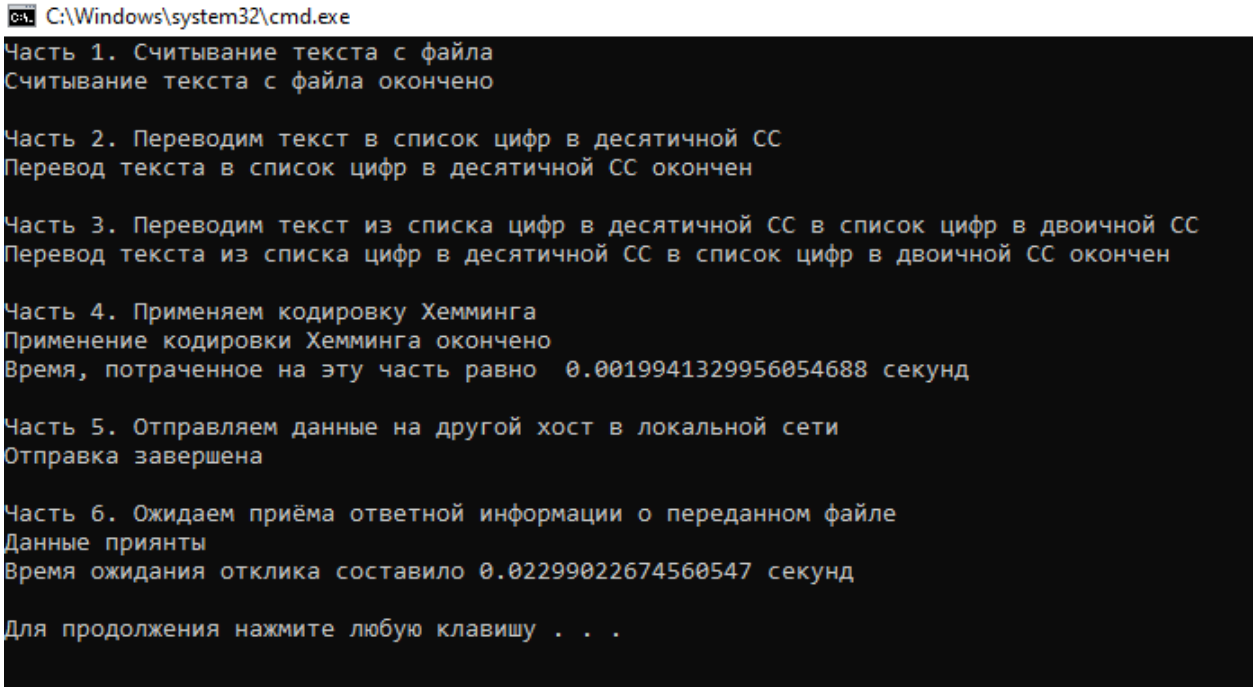

Рис. 10. Результаты работы на стороне передачи.

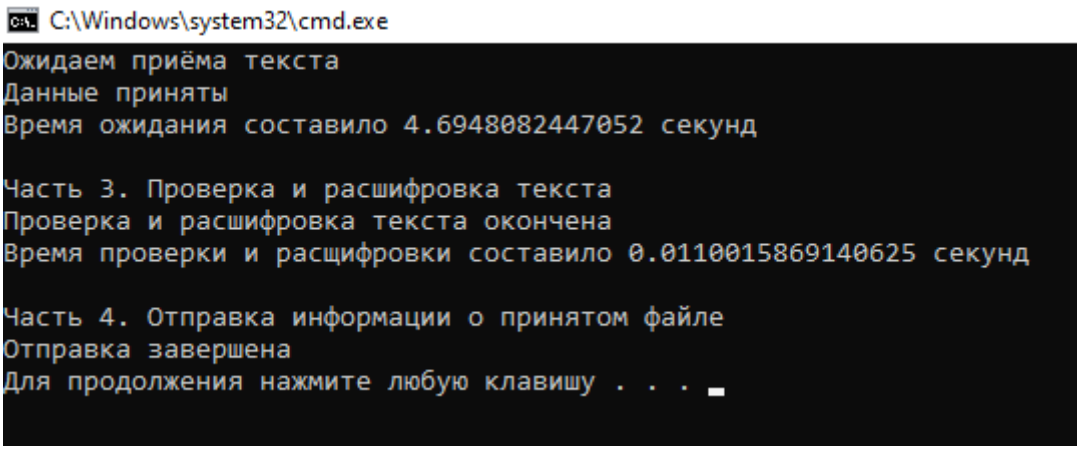

Рис. 11. Результаты работы на стороне приёма.

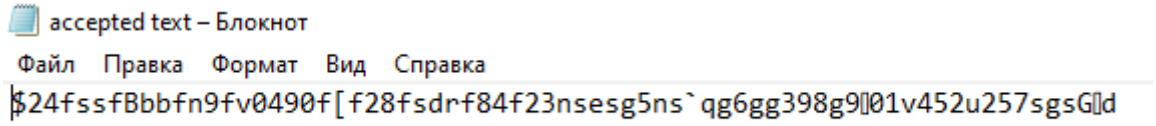

Рис 12. Принятый текст.

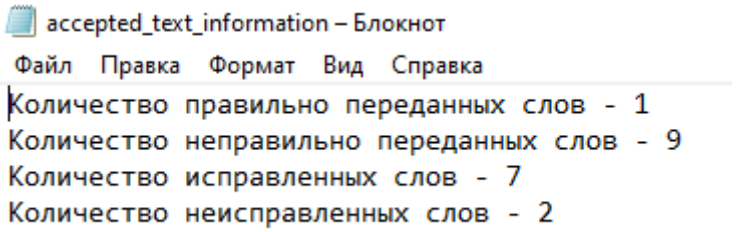

Рис. 13. Информация о переданном тексте.

Статья с Википедии об искусственных нейронных сетях

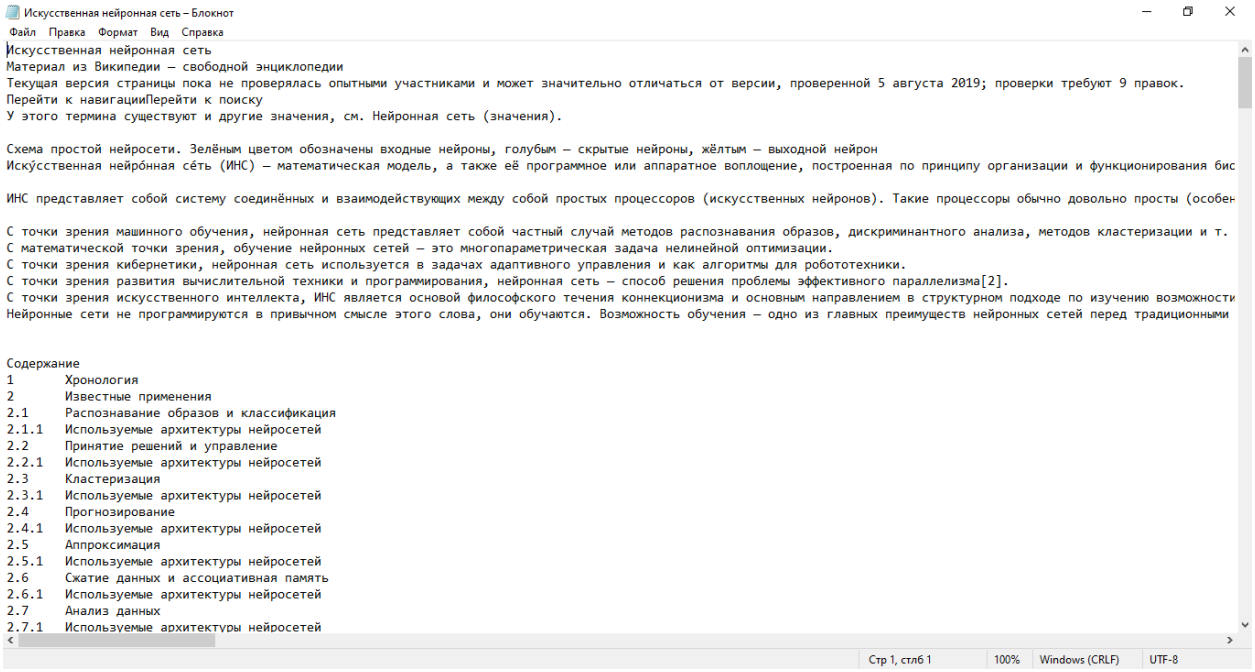

Рис. 14. Статья с Википедии об искусственных нейронных сетях

Передача файла без ошибок

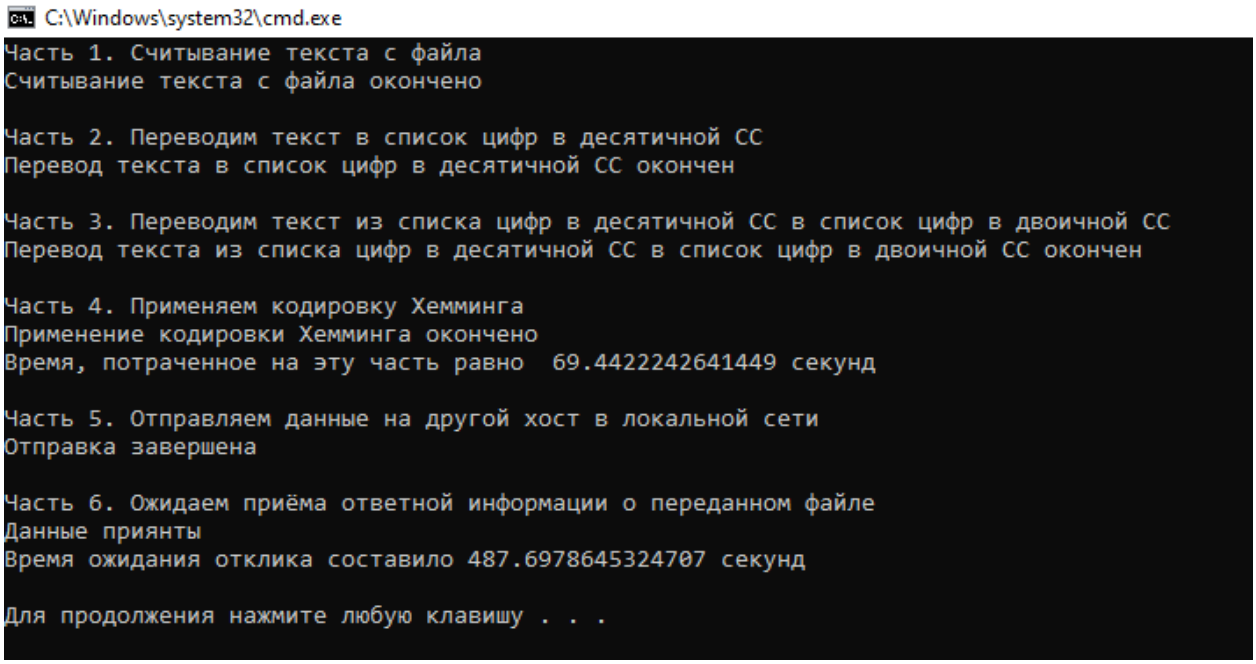

Рис. 15. Результаты работы на стороне передачи.

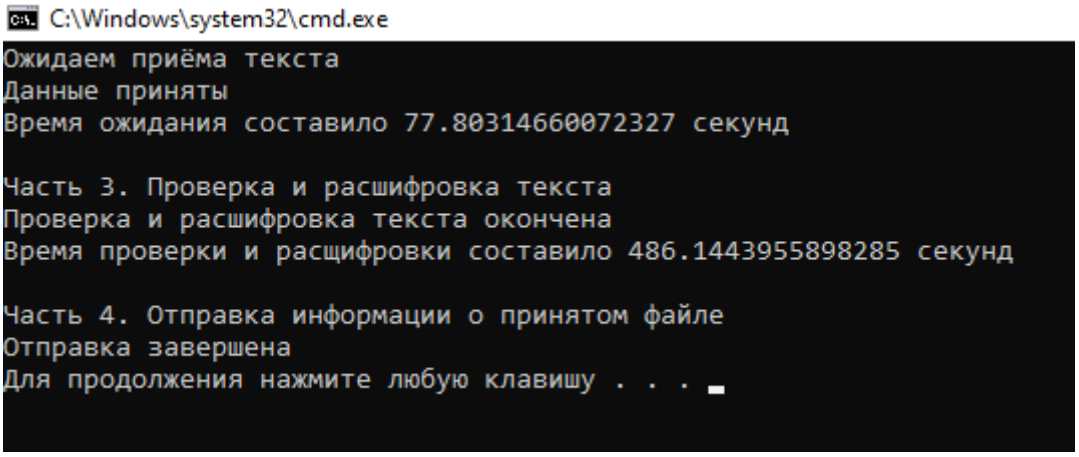

Рис. 16. Результаты работы на стороне приёма.

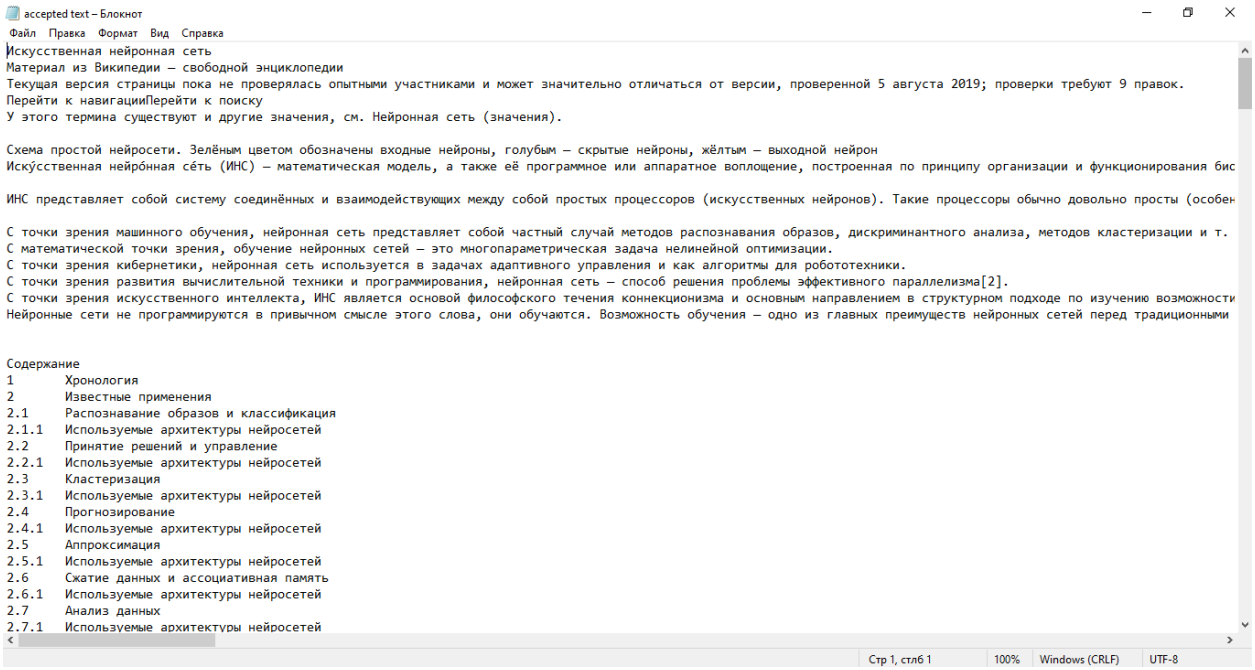

Рис 17. Принятый текст.

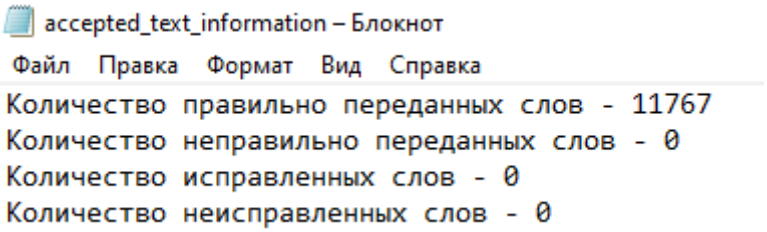

Рис. 18. Информация о переданном тексте.

Передача файла не более, чем с одной ошибкой в слове

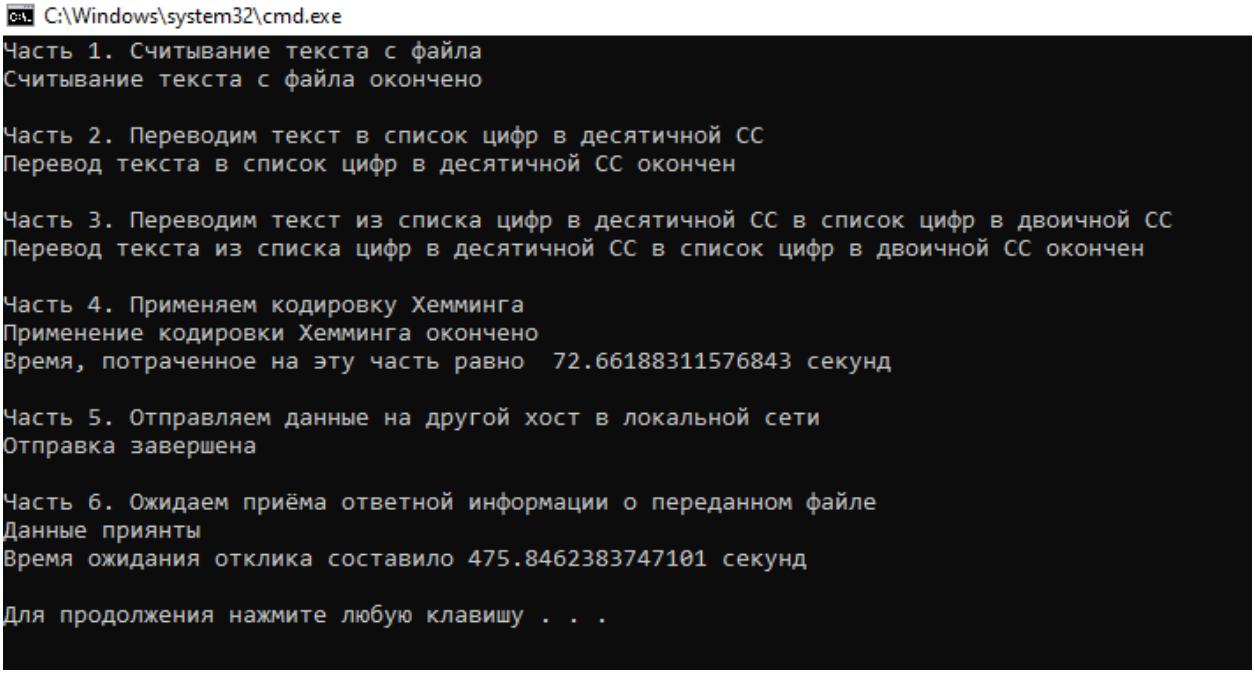

Рис. 19. Результаты работы на стороне передачи.

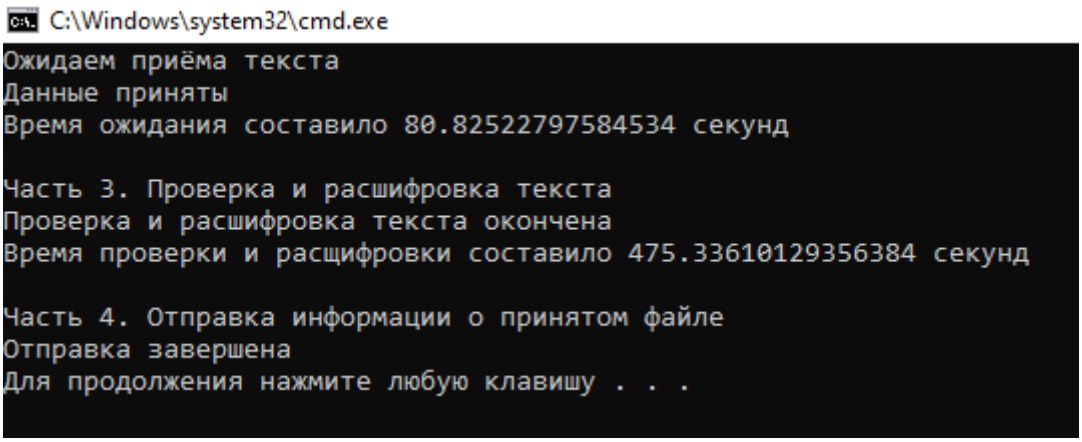

Рис. 20. Результаты работы на стороне приёма.

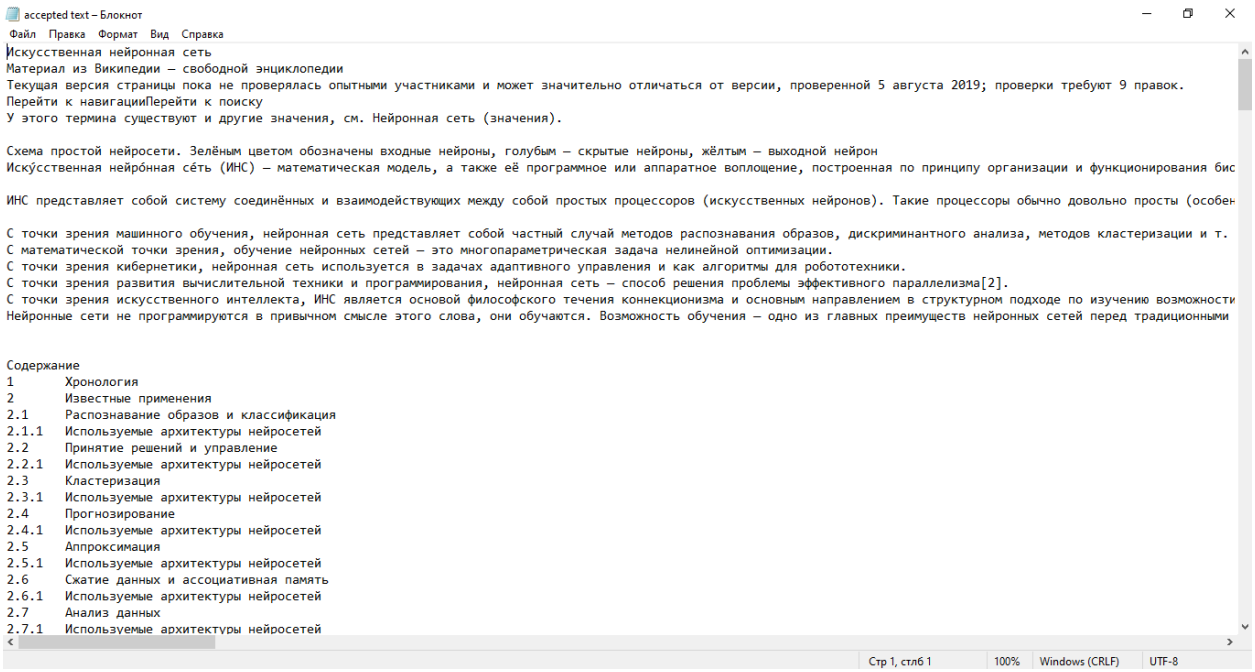

Рис 21. Принятый текст.

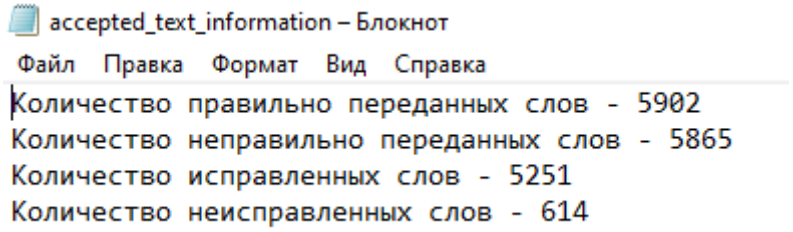

Рис. 22. Информация о переданном тексте.

Передача файла с не более, чем двумя ошибками в слове

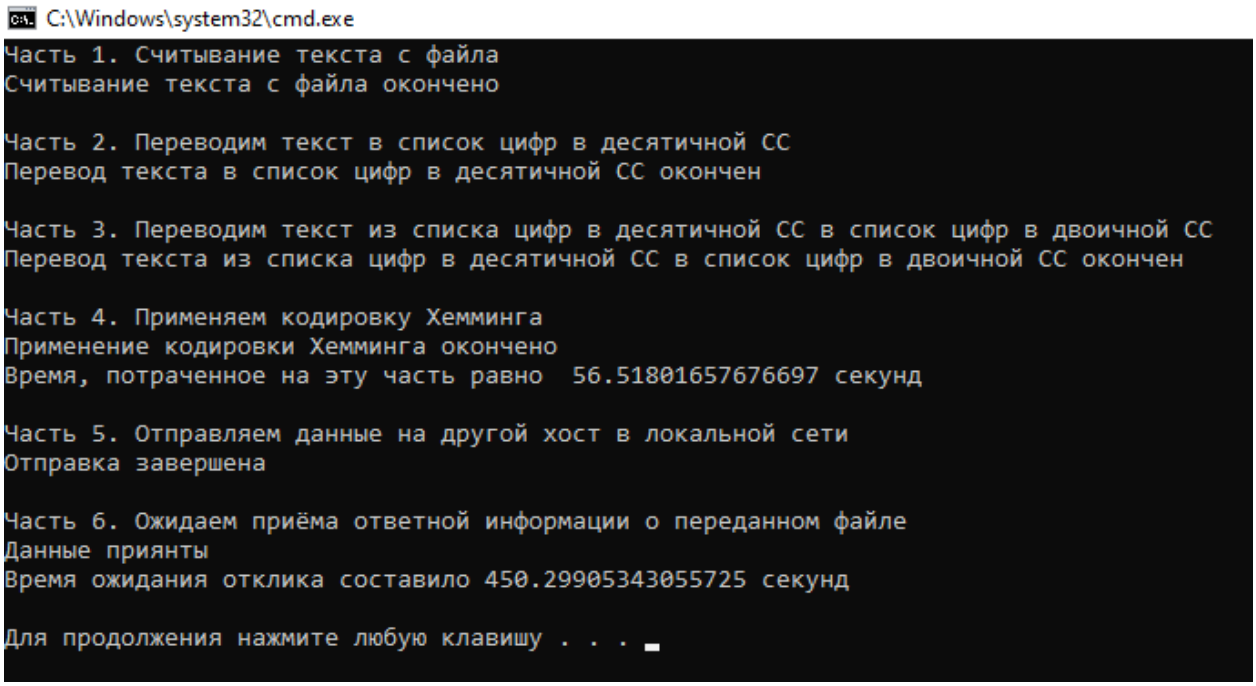

Рис. 23. Результаты работы на стороне передачи.

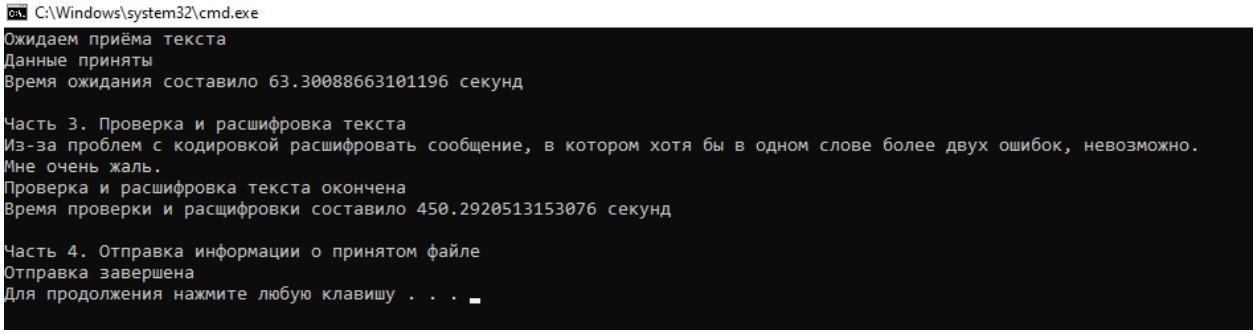

Рис. 24. Результаты работы на стороне приёма.

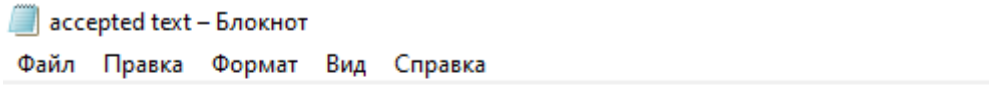

Рис 25. Принятый текст.

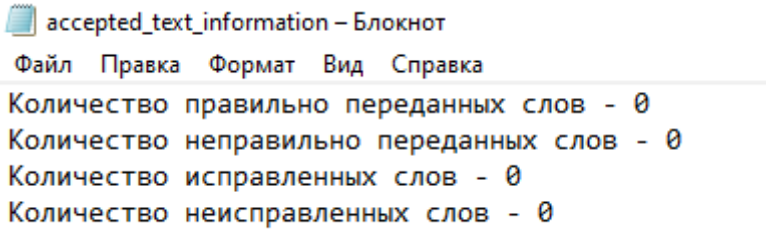

Рис. 26. Информация о переданном тексте.