УНИВЕРСИТЕТ ИТМО

«Анализ и разработка алгоритмов сжатия коротких текстов»

Минаев Борис Юрьевич

Научный руководитель:
канд. техн. наук, доцент Буздалов Максим Викторович

Кафедра КТ

2017

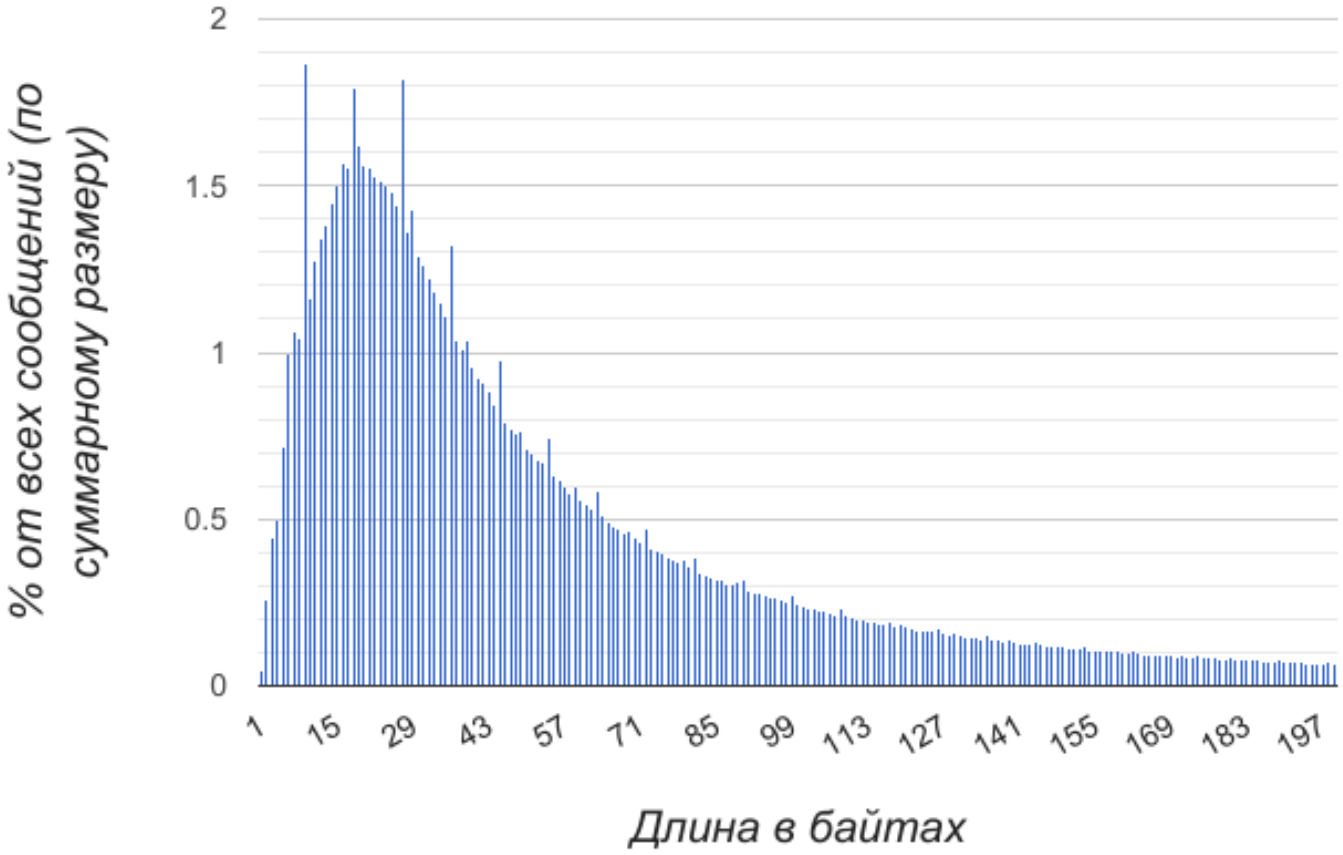
Постановка задачи

- Реализовать алгоритм сжатия сообщений
- Каждое сообщение длиной в среднем 30 символов
- Необходимо уметь разжимать отдельные сообщения
- Можно заранее предподсчитать общую дополнительную информацию

Актуальность задачи

- Алгоритм разрабатывался для хранения сообщений в социальной сети ВКонтакте
- 5 миллиардов сообщений в сутки
- 4 триллиона сообщений всего
- 400 терабайт данных
- Хотим использовать как можно меньше серверов, но при этом хранить как можно больше сообщений в оперативной памяти

Распределение сообщений по длинам



Существующие решения

- Большинство алгоритмов сжатия работают хорошо только на длинных текстах
- На коротких сообщениях они не успевают обучиться
- Вывод: нужно использовать алгоритм, который можно «обучить» на старых сообщениях

Оценка качества алгоритма

$$\text{compression ratio} = \frac{\text{messages size}}{\text{compressed messages size}+\text{additional info}}$$

- messages size — суммарный размер сообщений, которые помещаются в оперативную память
- compressed messages size — суммарный размер сообщений в сжатом виде
- additional info — размер дополнительной информации, которую надо хранить, чтобы уметь разжимать сообщения

Набор данных для тестирования

- Случайные русскоязычные твиты
- получены с помощью twitter streaming API
- 250 мегабайт
- 2 миллиона сообщений

- Храним набор подстрок
- Находим самую длинную подстроку W, которую уже видели
- Добавляем в словарь подстроку W + последний символ
- Словарь фиксированного размера (использовался 2^{20})
- При переполнении удаляем подстроки по LRU
- Коэффициент сжатия = 2.14

Однобуквенный Хаффман

- Считаем вероятность каждого символа
- Строим дерево Хаффмана с 256 листьями
- Коэффициент сжатия = 1.37

Хаффман по словам

- Разбиваем все символы на буквы и все остальные знаки
- Каждое сообщение разбивается на слова и разделители
- Отдельно строим словарь «слов» и словарь «разделителей»
- Для каждого словаря строим дерево Хаффмана
- Чтобы деревья не были слишком большими, удаляем слова, которые встречаются мало раз и добавляем слово-исключение
- Чтобы закодировать слово, которого нет в словаре, записываем слово-исключение, а потом побайтово нужное слово
- Коэффициент сжатия = 2.57
- Проблема: 3% слов, которые не попали в словарь, генерируют 13% сжатых данных!

Хаффман по словам

- Разбиваем все символы на буквы и все остальные знаки
- Каждое сообщение разбивается на слова и разделители
- Отдельно строим словарь «слов» и словарь «разделителей»
- Для каждого словаря строим дерево Хаффмана
- Чтобы деревья не были слишком большими, удаляем слова, которые встречаются мало раз и добавляем слово-исключение
- Чтобы закодировать слово, которого нет в словаре, записываем слово-исключение, а потом побайтово нужное слово
- Коэффициент сжатия = 2.57
- Проблема: 3% слов, которые не попали в словарь, генерируют 13% сжатых данных!

Хаффман по словам + Однобуквенный Хаффман

- Делаем Хаффман по словам
- Чтобы закодировать слово, которого нет в словаре, записываем слово-исключение, а потом однобуквенным Хаффманом кодируем слово (предварительно посчитав вероятности букв)
- Коэффициент сжатия = 2.72

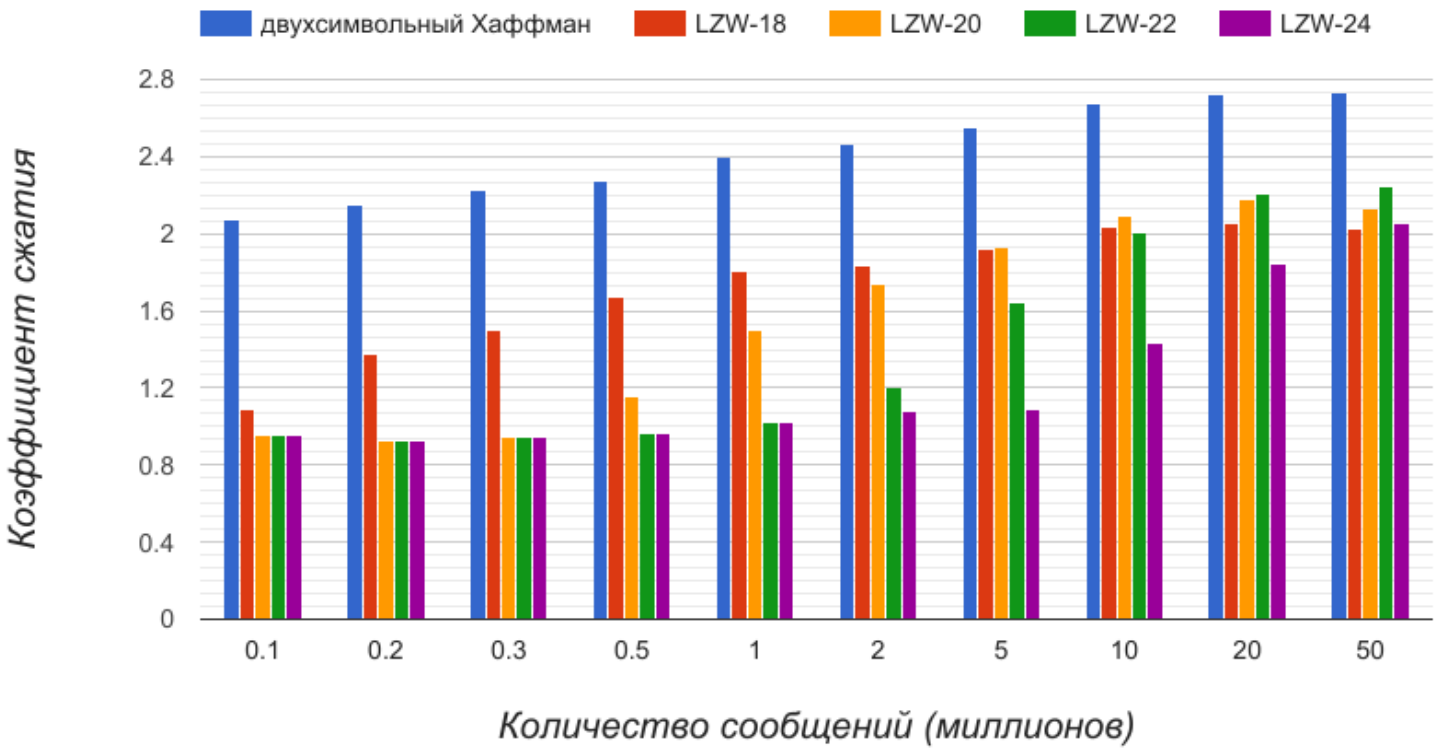
Хаффман по словам + Однобуквенный Хаффман зависящий от предыдущей буквы

- Делаем Хаффман по словам
- Чтобы закодировать слово, которого нет в словаре, записываем слово-исключение, а потом кодируем следующим образом:
- Посчитаем для каждого символа вероятность встретить каждый другой символ после него. Построим на этих вероятностях 256 различных деревьев Хаффмана. Будем использовать их для кодирования.
- Коэффициент сжатия = 2.82
- 6% слов, которые не попали в словарь, генерируют 12% сжатых данных

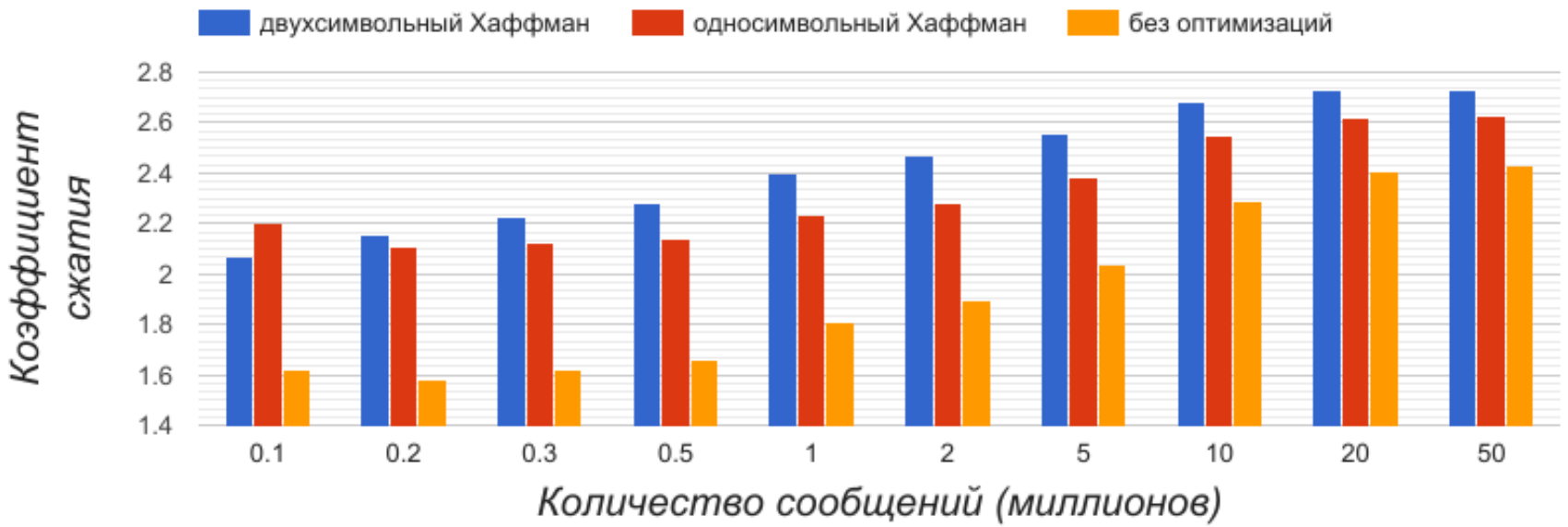
Сравнение

Алгоритм	Коэффициент сжатия
LZW	2.14
Однобуквенный Хаффман	1.37
Хаффман по словам	2.57
Хаффман по словам + Однобуквенный Хаффман	2.72
Хаффман по словам + Однобуквенный Хаффман зависящий от предыдущей буквы	2.82
gzip всех сообщений целиком (не применим)	2.95

Сравнение на реальных данных. LZW



Сравнение на реальных данных. Хаффман



Выводы

- Разработан алгоритм сжатия коротких текстов на основе уже существующих алгоритмов сжатия
- Основное отличие алгоритма — возможность быстро разжимать отдельные сообщения
- По эффективности алгоритм не сильно проигрывает архиваторам, которые сжимают текст целиком
- Алгоритм уже применяется для сжатия сообщений в социальной сети ВКонтакте, что позволило сэкономить несколько терабайт оперативной памяти

Вопросы?