Министерство образования и науки Российской Федерации

Государственное образовательное учреждение
высшего профессионального образования
Московский физико-технический институт
(государственный университет)

Факультет инноваций и высоких технологий
Кафедра компьютерной лингвистики

Японский. Буквенные n-граммы для распознавания

выпускная квалификационная работа
(бакалаврская работа)
по направлению 01.03.02 «Прикладная математика и информатика»

Студент Куликов А.В.
Научный руководитель Андрианов А.И.

Москва, 2017

Оглавление

Введение

История попыток распознать текст началась более века назад. В 1914 году Эмануэль Гольдберг разработал устройство, которой считывало символы и транслировало их в телеграфный код. Примерно в то же время ирландский химик Эдмунд Фурнье д'Альбе создал и запатентовал «оптофон» — прибор, умеющий переводить написанное в систему звуков, различающихся по высоте. Оптофон предназначался для того, чтобы слепые могли «читать».

В 1929 году Густав Таушек (Gustav Tauschek) разработал метод оптического распознавания текста. Машина Таушека представляла собой механическое устройство, которое использовало шрифтовые шаблоны и фотодетектор. Он запатентовал своё изобретение сначала в Германии, а позднее и в США, в 1935 году. Это и положило начало проблеме качественного оптического распознавания символов (Optical Character Recognition, OCR).

Коммерческое производство подобных маних было налажено уже в 1950-х, после войны. Использовавшие наработки военных, производители OCR-машин продвигались всё дальше, увеличивая применимость технологии и качество распознавания.

Постепенно появлялись как универсальные OCR-программы (ABBYY FineReader, Adobe Acrobat), так и специализированные для конкретной области (SmartScore для нотной записи, Persian Reader для фарси и т.д.). При этом точность в задаче распознавания напечатанных латинских символов достигла 99%-100% качества, в то время как корректное распознавание рукописного текста или текста, написанного в другом алфавите, до сих пор является темой множества исследований. Особняком стоит задача распознавания текста на восточных языках (CJK: китайский, японский, корейский), из-за большого размера алфавита в этих языках.

Настоящая работа представляет собой сравнение некоторых методов машинного обучения для исправления ошибок распознавания текста в японском языке.

Спектр способов, которыми можно решать проблему автоматического

исправления ошибок, довольно широк, и включает в себя различные вариации n-граммных моделей (n-gram models), использование нейросетей (Neural Networks, NN), скрытых моделей Маркова (Hidden Markov Models, HMM) и прочих методов машинного обучения. Более подробный обзор основных современных подходов можно найти в **?**.

Среди возможных решений использование n-граммных моделей занимает особую нишу из-за относительной прозрачности и интуитивности принципов работы, и в то же время достаточно широких возможностей по настройке алгоритма.

Подход, предложенный в **?**, использует n-граммные модели, а также различные алгоритмы сглаживания для исправления опечаток, опираясь на словное деление текста.

В работе **?** также даются эвристики для определения границ слов, использующие граф линейного деления (ГЛД). Эти границы слов затем используются в n-граммной модели в качестве вспомогательного контекста.

Более подробно эти и другие подходы разобраны в соответствующем разделе (**??**).

Данное исследование призвано рассмотреть некоторые из n-граммных моделей и сравнить их эффективность в задаче исправления опечаток в японском языке.

Актуальным приложением этой работы является система распознавания восточных языков в ABBYY FineReader.

1 Постановка задачи

Определение 1.0.1. *Оптическое распознавание символов (Optical Character Recognition, OCR)* – процесс считывания текста с физического носителя и его сохранения в цифровом формате. Текст состоит из *символов*.

Определение 1.0.2. *Ошибка OCR* – случай, когда очередной символ текста распознался неверно или не распознался. Ведёт к понижению качества распознавания.

Определение 1.0.3. *N-грамма* – последовательность из n элементов (слов, звуков, символов). Анализируя их частотности, можно строить модели для анализа и синтеза языка.

Определение 1.0.4. *N-граммная модель* – вероятностная модель языка, которая рассчитывает вероятность последнего элемента n-граммы, если известны все предыдущие.
При использовании n-граммных моделей предполагается, что появление каждого элемента зависит только от предыдущих элементов.

Цель работы – сравнить эффективность различных символьных n-граммных моделей в задаче исправления ошибок OCR в японском языке.

Из цели работы вытекают следующие **задачи**:

- Рассмотреть существующие подходы к n-граммному моделированию японского языка;
- Реализовать некоторые символьные n-граммные модели;
- Развернуть систему для тестирования и сравнения моделей.

Чтобы понять специфику цели работы, нужно учесть особенности японского языка.

Очевидно, что устройство японского языка на уровне конкретных символов сложнее, чем устройство языков латино-романской группы, в которых существует всего 25-40 символов, учитывая возможную диакритику.

1.1 Обзор японского языка

Письменный японский текст – это комбинация слогово-фонетических символов (кана) и иероглифов (кандзи). Слоговая азбука кана делится на катакану и хирагану, которые представляют собой разные графические формы одних и тех же слогов.

Рассмотрим эти символы подробнее:

- Хирагана (см. **??**). В основном используется для образования грамматических морфем.

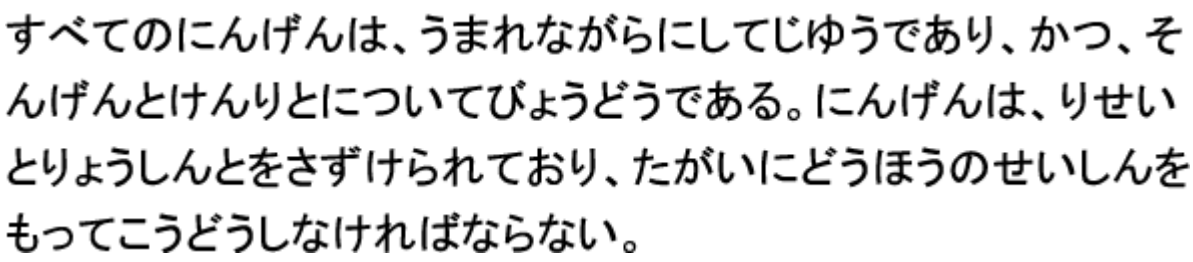


Рис. 1: Хирагана

- Катакана (см. **??**). Используется для транскрибирования иностранных заимствованных слов.

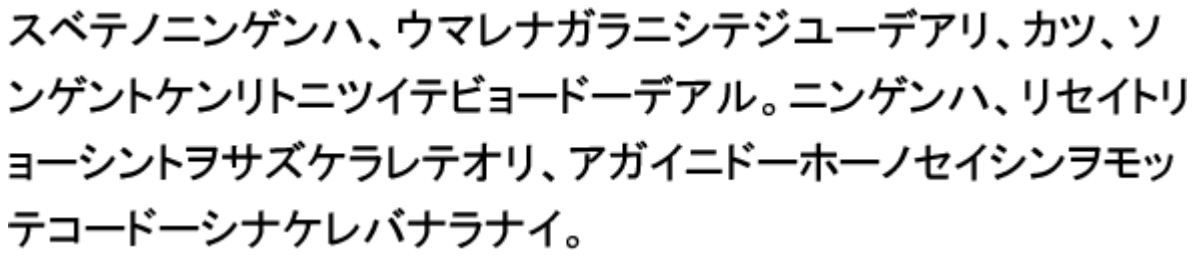


Рис. 2: Катакана

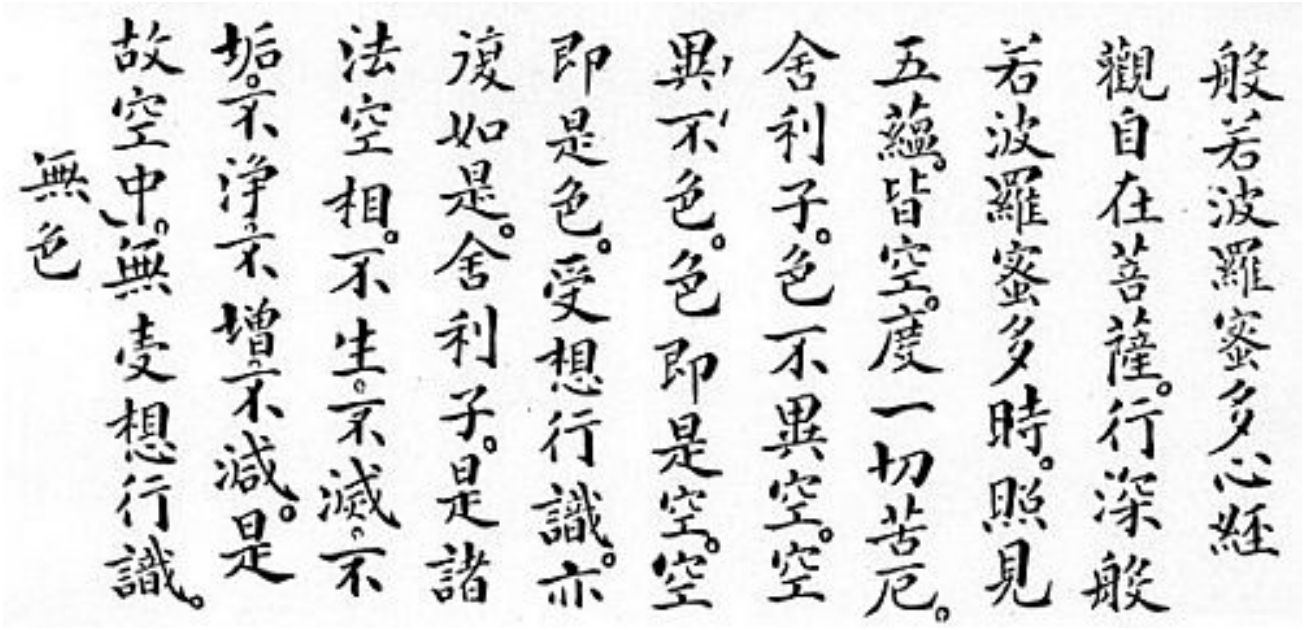
- Также есть диакритические символы – дакутен, хандакутен (см. ?? и ??). Они могут применяться как к катакане, так и к хирагане, и определённым образом влияют на звучание слогов.

ガ ga	ギ gi	グ gu	ゲ ge	ゴ go	が ga	ぎ gi	ぐ gu	げ ge	ご go
ザ za	ジ ji	ズ zu	ゼ ze	ゾ zo	ざ za	じ ji	ず zu	ぜ ze	ぞ zo
ダ da	ヂ ji	ヅ zu	デ de	ド do	だ da	ぢ ji	づ zu	で de	ど do
バ ba	ビ bi	ブ bu	ベ be	ボ bo	ば ba	び bi	ぶ bu	べ be	ぼ bo
パ pa	ピ pi	プ pu	ペ pe	ポ po	ぱ pa	ぴ pi	ぷ pu	ぺ pe	ぽ po

Рис. 3: Дакутен катакана

Рис. 4: Дакутен хирагана

- Кандзи (см. ??). Это символы, несущие семантическую нагрузку.



般若波羅蜜多心経
観自在菩薩。行深般
若波羅蜜多時。照見
五蘊皆空。度一切苦厄。
舎利子。色不異空。空
異不色。色即是空。空
即是色。受想行識亦
復如是。舎利子。是諸
法空相。不生不滅。不
垢不浄。不増不減。是
故空中。無色無受想行識。
無色

Рис. 5: Кандзи

Кана различает 46 слогов, которые могут записываться как катаканой, так и хираганой. А вот иероглифов кандзи существует гораздо больше: 2136 (т.н. jōyō kanji – "обычно используемые кандзи") достаточно для жизни, 6879 используется в кодировке JIS X 0208 (Japanese Industrial Standart, см.

?), а самые большие словари иероглифов (например, Большой китайско-японский словарь) включают в себя от 50000 до 85000.

Кроме перечисленных символов, в японском тексте могут быть и другие: фуригана – маленькие знаки каны в качество фонетических подсказок, ромадзи – система транслитерации японских слов в латиницу и т.д. Однако, в данной работе эти разделы оставлены за кадром.

Японский текст записывается с помощью комбинаций кандзи, кан и пунктуации, при этом отсутствует пробельное деление предложений на слова (см. ??).

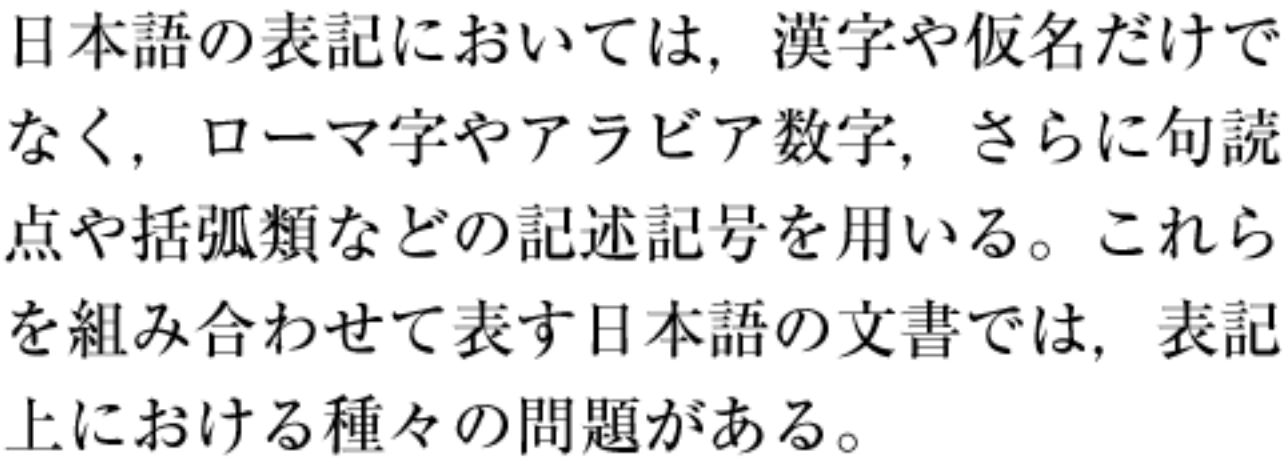


Рис. 6: Японский текст

По сравнению с латино-романскими языками, где алфавит меньше в сотни раз, а деление текста на слова очевидно, задача корректного распознавания символов становится значительно сложнее. Это требует более изощрённых подходов для автоматического анализа распознанного текста и поиска ошибок в нём.

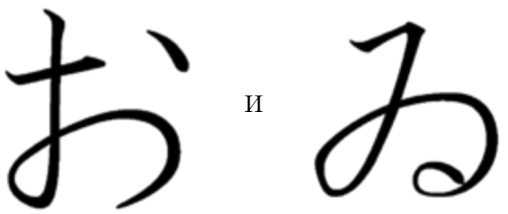
Рассмотрим несколько примеров символов, которые легко спутать.

1.2 Путающиеся символы в японском

При таком большом размере алфавита частые ошибки OCR можно разбить по классам. Вот некоторые из них:

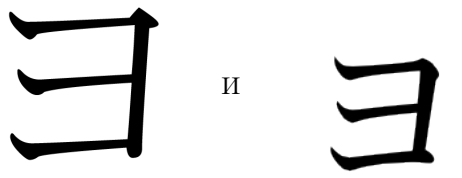
2Kana 2 похожие каны. Таких случаев достаточно мало, а методы их разли-

чения уже существуют (см., например, метод с использованием глубокого обучения и свёрточных нейронных сетей в работе **?**).



KaGa Кана может легко путаться с соответствующим ей дакутен-символом.

BigSmall Существуют большие и маленькие каны, которые нужно различать.



В будущем мы будем рассматривать 3 случая ошибок: **KaGa**, **BigSmall** и **Mix** (смесь **KaGa** и **BigSmall**), более строгое определение которых будет дано в **??**.

1.3 Формальная постановка задачи

Определение 1.3.1. *Алфавит* $\Sigma = \{a, b, c, ..\}$ – множество символов в данном языке.

Определение 1.3.2. *Текст* $Text \in \Sigma^+$ – последовательность символов из алфавита Σ положительной длины.

Определение 1.3.3. Текст делится на конечное число *предложений* $S = \{S_1, S_2, S_3, ...\}$ знаками пунктуации и форматированием. $Text = S_1 S_2 S_3 ....$

Для каждого из предложения текста существует единственно верный вариант написания, а также некоторое (фиксированное) число неверных. Требуется ответить, какой из вариантов верен.

Определение 1.3.4. *Оценивающий алгоритм (estimator)* $\Theta : S \to \mathbb{R}^+$ – функция, возвращающая оценку правильности варианта S.

Среди k вариантов предложения выбирается наилучший: $S_{best} = \underset{S}{\operatorname{argmax}}\, \Theta(S)$, который и считается правильным.

Если S_{best} угадано верно, то на данном предложении алгоритм Θ отработал правильно.

Определение 1.3.5. *Качество алгоритма* $Q(\Theta) = \dfrac{\#\{\textit{угаданных предложений}\}}{\#\{\textit{всего предложений}\}}$.

Задача – реализовать ряд оценивающих алгоритмов (см. **??**), основанных на n-граммных моделях, и сравнить их по качеству.

2 Обзор источников

Проблема качественного OCR в различных языках (в частности, в японском) стоит достаточно давно, и существует множество различных подходов к её решению. Большинство подходов основываются на применении различных методов машинного обучения, таких как n-граммные модели, нейросети, модели Маркова и т.д.

В силу особенностей японского языка (см. **??**), а именно отсутствия словного деления в текстах, существующие методы исправления ошибок OCR можно разделить на 2 класса: использующие информацию о словном делении и не использующие её.

2.1 Методы, не использующие словное деление

Идея подобных методов заключается в том, чтобы, не тратя ресурсы на определение границ слов в тексте, оперировать предложениями (границы которых выделяются достаточно легко) как единицами трансляции, и исправлять возможные ошибки OCR, не углубляясь в членение предложений.

Этот подход не слишком популярен, но применяется, например, в **?** для задачи машинного перевода без привлечения словного деления, что авторы особенно подчёркивают.

Стоит также заметить, что настоящая работа может быть отнесена к этому классу.

2.2 Методы, использующие словное деление

Информация о словном делении текста даёт удобный контекст для выделения признаков и настройки параметров при машинном обучении. Но эту информацию надо откуда-то получать, что приводит к делению алгоритмов на 2 этапа:

- Получение словного деления. Может производиться с помощью специализированных алгоритмов (например, модификаций алгоритма Витерби, как в **?**, или марковских случайных полей (conditional random

fields, CRFs), как в **?**), а также путём анализа конкретных языковых конструкций (например, bunsetsu boundaries, см. **?**).

Кроме того, словное деление может быть получено путём использования в работе предварительно размеченного корпуса, в разметке которого есть словное деление. В качестве примеров таких корпусов можно привести EDR Japanese Corpus (см. **?**), ATR Dialogue Database (см. **?**) и т.д.

- Применение словного деления как контекста для детектирования ошибок OCR. Достоверная информация о словном делении позволяет считать слова единицами трансляции, не теряя контекста предложения. Это даёт возможность получить больше признаков для машинного обучения. Подобный подход представлен в статье **?**, где используются n-граммные модели с backoff и различными подходами к сглаживанию (Good-Turing, Witten-Bell).

Обзору основных актуальных методов исправления ошибок OCR в японском также целиком посвящена статья **?**.

3 Описание моделей оценивания текста

Перед обучением моделей корпус разбивается на независимые и гомогенные части: обучающую и тестовую выборки. Обучающая выборка используется для обучения модели, тестовая – для проверки качества обучения и, собственно, оценки модели.

В работе рассматриваются следующие модели:

- n-граммные с фиксированным n, $n \in \{1, 2, 3\}$
- Backoff-модель, $n_{max} \in \{3, 5, 7\}$
- Модель Катца (Katz), $n_{max} \in \{3, 5, 7\}$

Также из-за большого размера алфавита необходимо использовать сглаживание (smoothing) для учёта символов и n-грамм, не встретившихся в обучающей выборке. Подробнее о механизме сглаживания – см. **??**.

Во всех моделях оценка предложения производится через агрегацию оценок отдельных n-грамм.

3.1 N-граммные модели с фиксированным n

Обучение модели Для данного n по обучающей выборке собираются статистики по всем n-граммам. Эти статистики затем нормализуются и сериализуются для дальнейшего использования.

$$C(w_{i-n+1}, ..., w_{i-1}, w_i) = \frac{N(w_{i-n+1}, ..., w_{i-1}, w_i)}{N(w_{i-n+1}, ..., w_{i-1})}$$

Применение модели В силу простоты модели оценка n-граммы из тестовой выборки берётся напрямую из собранных на предыдущем этапе статистик.

$$P(w_i|w_{i-n+1}, ..., w_{i-1}) = C(w_{i-n+1}, ..., w_{i-1}, w_i)$$

3.2 Backoff-модель

Обучение модели Этап обучения модели практически такой же, как и в случае простой n-граммной модели, с разницей в том, что здесь собираются статистики для всех $n \leq n_{max}$.

Применение модели Идея backoff-подхода состоит в том, что при нехватке данных для оценки какой-либо n-граммы $w_{i-(n-1)}...w_{i-1}w_i$ постепенно уменьшается n, что позволяет увеличить общность алгоритма и оценить n-грамму по частям, но более надёжно. За счёт этого уменьшается вероятность переобучения модели на конкретных данных. Подробнее можно узнать в **?**.

$$P_n(w_i|w_{i-n+1},...,w_{i-1}) = \begin{cases} C(w_i|w_{i-n+1},...,w_{i-1}) & \text{if } C(w_i|w_{i-n+1},...,w_{i-1}) > k \\ P_{n-1}(w_i|w_{i-n+2},...,w_{i-1}) & \text{otherwise} \end{cases}$$

3.3 Модель Катца (Katz)

Обучение модели Этап обучения модели такой же, как и в случае backoff-модели.

Применение модели Модель Катца является улучшенной версией backoff-модели, в которой накладывается динамический дисконт (коэффициенты $d_{w_{i-n+1}...w_i}$ и $\alpha_{w_{i-n+1}...w_{i-1}}$) на оценку n-граммы в случае уменьшения n. Более подробно о модели Катца можно прочитать в **?**.

$$P_n\left(w_i|w_{i-n+1}...w_{i-1}\right) = \begin{cases} d_{w_{i-n+1}...w_i}\dfrac{C(w_{i-n+1}...w_i)}{C(w_{i-n+1}...w_{i-1})} & \text{if } C(w_{i-n+1}...w_i) > k \\ \alpha_{w_{i-n+1}...w_{i-1}}P_{n-1}\left(w_i|w_{i-n+2}...w_{i-1}\right) & \text{otherwise} \end{cases}$$

При этом коэффициенты $d_{w_{i-n+1}...w_i}$ и $\alpha_{w_{i-n+1}...w_{i-1}}$ вычисляются следующим образом:

$d_{w_{i-n+1}...w_i} -$ размер дисконта, вызванного сглаживанием (см. **??**)

$$\alpha_{w_{i-n+1}...w_{i-1}} - \text{ размер дисконта за backoff } -$$
$$= \frac{\beta_{w_{i-n+1}...w_{i-1}}}{\Sigma_{\{w_i:C(w_{i-n+1}...w_{i-1})\leq k\}} P_{n-1}\left(w_i | w_{i-n+2}...w_{i-1}\right)},$$

где $\beta_{w_{i-n+1}...w_{i-1}} -$ остаточная плотность вероятности для $(n-1)$-грамм $-$

$$= 1 - \Sigma_{\{w_i:C(w_{i-n+1}...w_i)>k\}} \frac{C(w_{i-n+1}...w_i)}{C(w_{i-n+1}...w_{i-1})}$$

4 Описание эксперимента

Для исследования способов исправлять ошибки OCR в тексте необходимо либо иметь большой корпус размеченных данных с результатами распознавания, либо построить некоторую модель частотных ошибок OCR и эмулировать их самостоятельно. Для данной работы был выбран второй подход, поскольку это убирало необходимость интегрироваться с какой-либо OCR машиной (например, ABBYY FineReader). Подобная интеграция могла бы быть достаточно трудоёмкой задачей, не являясь при этом основной целью исследования.

4.1 Корпус

Для обучения и сравнения n-граммных моделей использовался корпус html-страниц с ряда японских сайтов (например, www.38shop.jp, www.ewoman.co.jp, www.cat-v.jp и других) общим размером $\approx 8,5GB$.

В корпусе было 163244 html-страницы со средним размером $\approx 52kB$.

Тексты из этого корпуса не были результатом OCR, поэтому в них не должно было быть ошибок, связанных с распознаванием. Эти тексты были признаны верными с точки зрения языка и подходящими для обучения моделей.

В рамках подготовки корпуса к эксперименту тексты были перемешаны, чтобы тематика текста не зависела от его исходного положения в корпусе, из текстов были удалены html-теги, сложное форматирование, небольшое число мусорных символов. Также корпус был единообразно переведён в кодировку Unicode. Подробнее об этих технических этапах – см. **??**.

После вышеперечисленных операций корпус был готов к использованию, его размер составлял $\approx 1,7GB$. При этом размер алфавита в нашем корпусе составлял ≈ 7000 символов, что в 3 раза меньше размера таблиц Unicode.

Имея данные, готовые к использованию, проанализируем их.

4.2 Обработка корпуса

После обработки корпуса его можно было описать следующими цифрами (**??**):

Параметр	Значение
Размер (kB)	1 721 504
Символов	640 604 961
среди них уникальных	6 861
Предложений	79 497 345

Таблица 1: Характеристики корпуса

Если посмотреть на распределение частот отдельных символов (**??**), то оно выглядит так:

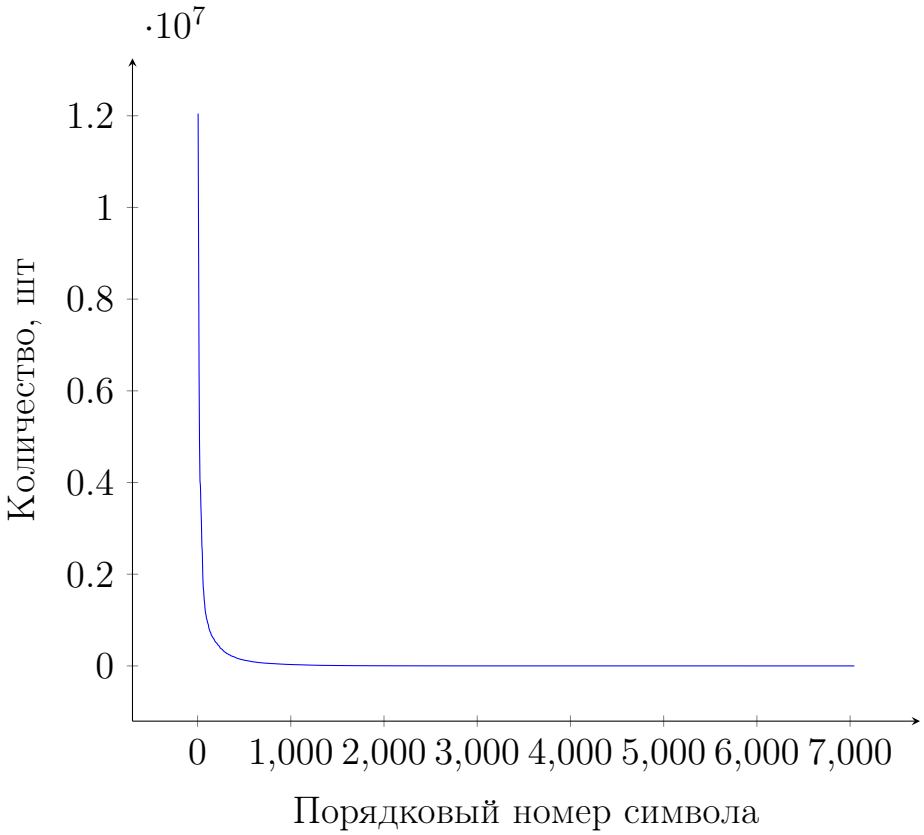


Рис. 7: Распределение частот униграмм

Видно, что распределение похоже на обратное экспоненциальное (кстати, это же утверждает закон Ципфа (Zipf, см. **?**)). Проверим эту гипотезу, построив график обратного логарифма (**??**):

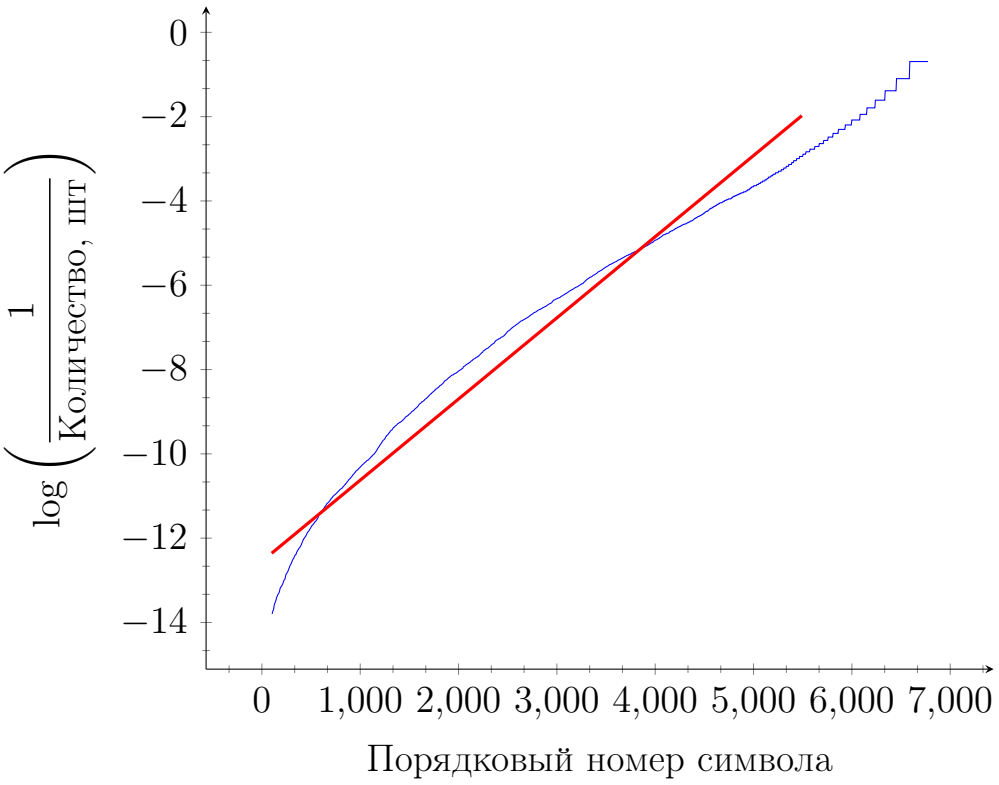


Рис. 8: Проверка закона Ципфа

Действительно, этот график с достаточной точностью ложится на прямую, выбиваясь из неё только в начале списка символов (там находятся самые частотные кандзи, а также практически вся хирагана/катакана, подробнее про распределения различных подмножеств символов см. **??**). Тем самым, эмпирический закон Ципфа был проверен на ещё одном корпусе.

Посмотрев на **??**, можно также заметить, что только очень малая часть символов появляется большое число раз. Посмотрим поближе на "голову" того же распределения (**??**):

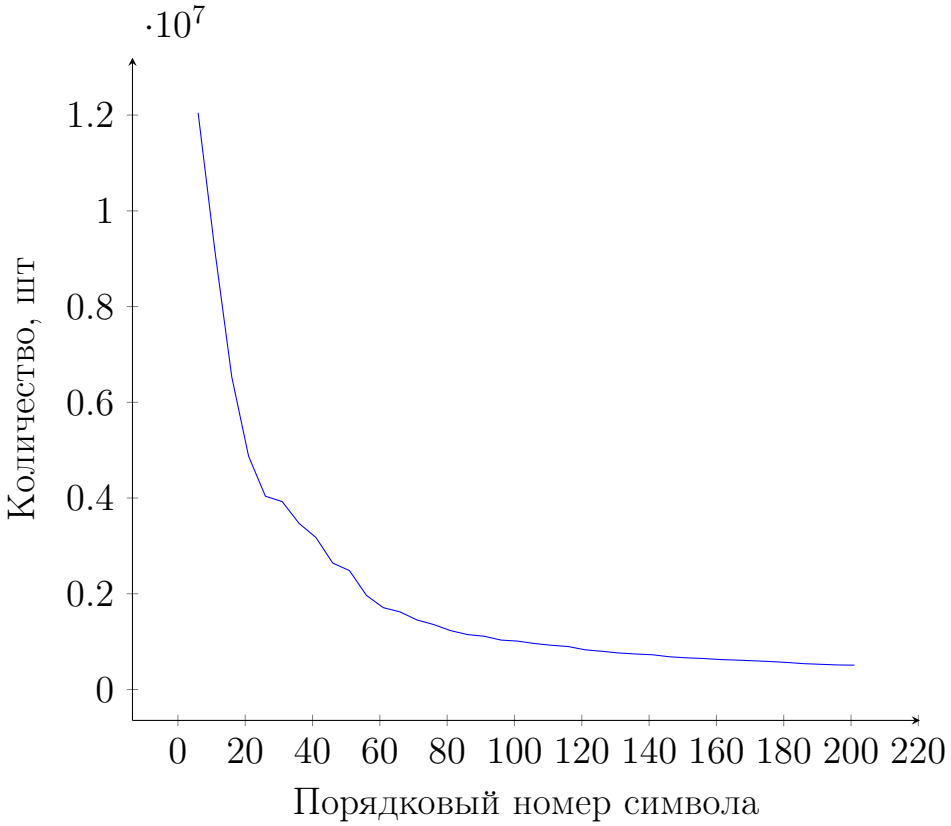


Рис. 9: Частоты униграмм – голова распределения

Действительно, лишь ≈ 200 символов встречаются достаточно часто.

Осталюся ещё примерно 6500 символов, которые входят в алфавит, но статистически мало отличаются от тех символов, что вовсе не встретились в нашем корпусе. Для оптимизации времени работы и занимаемой памяти эти символы можно представить более сжато.

Определение 4.2.1. *Корзина (бакет, bucket)* – множество символов, которые считаются статистически малозначимыми и заменяются на U+FFFD (Unicode Replacement Character).

Бакет B_i характеризуется числом $|\Sigma_{B_i}|$ – размером алфавита, который остаётся после сливания некоторого хвоста распределения в бакет. Было решено рассматривать бакеты с алфавитами размером $|\Sigma_{B_i}| = \{7000, 4800, 2600, 200\}$, поскольку примерно на эти размеры алфавитов приходятся изменения в

характере убывания частот символов. При этом бакет $|\Sigma_{B_i}| = 7000$ представляет собой исходный корпус.

Основным бакетом впоследствии был выбран $B^* : |\Sigma_{B^*}| = 4800$, поскольку при нём, с одной стороны, реализуется **сглаживание** хвоста распределения, что косвенно даёт возможность учитывать символы, не встретившиеся в обучающей выборке. С другой стороны, размер алфавита остаётся достаточно большим, что будет важно при рассмотрении шумов (см. **??**)

На **??** схематично изображено распределение частот после применения бакета с $|\Sigma_{B_i}| = 2600$.

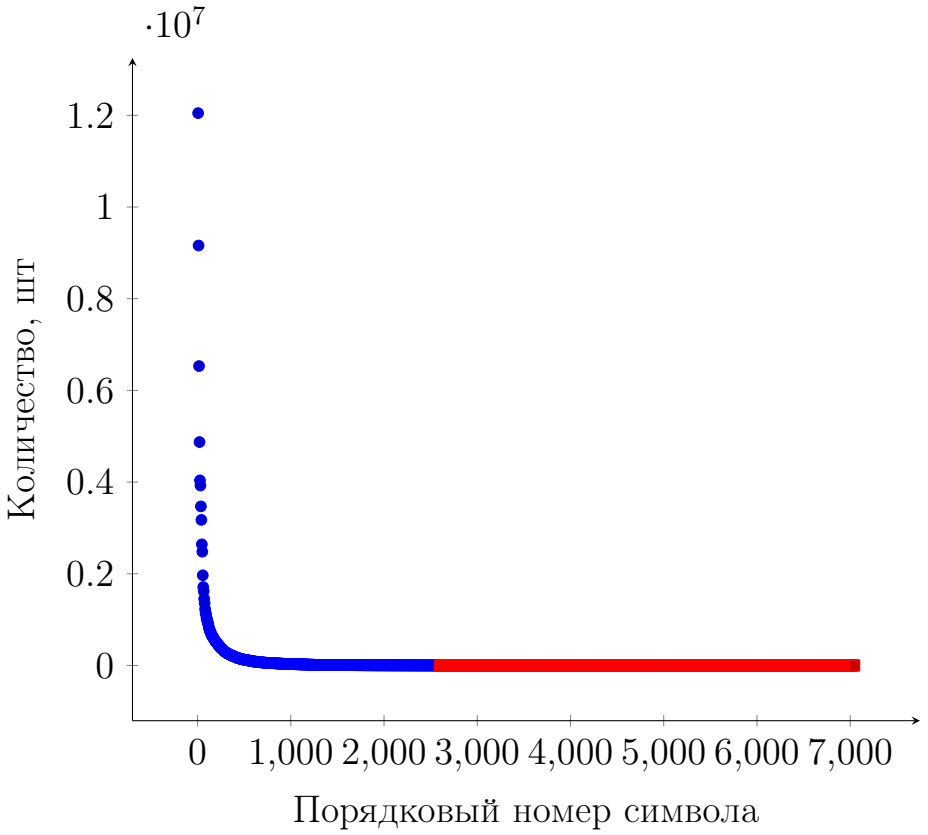


Рис. 10: Распределение частот униграмм: бакет с $|\Sigma_{B_i}| = 2600$

Поскольку нам были недоступны корпуса текстов, распознанные какой-либо OCR машиной, было принято решение эмулировать ошибки OCR самим. Это делалось при помощи генератора шума.

4.3 Генератор шума и режимы его работы

Определение 4.3.1. *Шум* $Noise = \{(a_1, a_2), (b_1, b_2, b_3), (c_1, c_2), ...\}$ – множество наборов символов алфавита Σ, которые легко спутать при распознавании. Конкретные шумы определяются эмпирически.

Для эмуляции ошибок OCR был разработан скрипт – генератор шума. Он параметризуется конкретным шумом и частотой его применения.

Определение 4.3.2. *Генератор шума* – настраиваемый скрипт, который принимает эталонное предложение S, находит в нём символы-представители наборов конкретного шума $x \in S \mid \exists \xi = \{\xi_1, \xi_2, ..., \xi_l\} \in Noise : x \in \xi$, и случайным образом меняет эти символы x на "шумовые"из соответствующего набора ξ.

С помощью шума $Noise$ случайным образом генерируются ошибки в предложениях текста $Text$. Таким образом происходит стохастическая эмуляция ошибок OCR.

Тестовая часть корпуса была разбита на предложения (см. формальную постановку задачи в **??**), которые независимо друг от друга зашумлялись. Эти предложения после зашумления подавались на вход оценивающему алгоритму Θ, который выбирал лучший из предложенных вариантов.

Были определены следующие шумы, обоснование выбора см. в разделе **??**:

KaGa Наборы симвопов, соответствующие добавлению диакритики. Например,

かが	しじ	たな
きぎ	すず	だな
くぐ	ふぶぷ	んだ
けげ	そぞ	ちぢ
こご	ただ	ほぼぽ
さざ	なに	. . .

BigSmall Большие/маленькие написания букв:

あぁ	つっ	アァ
いぃ	やゃ	イィ
うぅ	ゆゅ	ウゥ
えぇ	よょ	エェ
おぉ	わゎ	. . .

Mix Комбинация предыдущих режимов.

かが	ふぶぷ	んだ
あぁ	そぞ	ちぢ
いぃ	ただ	ほぼぽ
うぅ	なに	. . .
えぇ	たな	
おぉ	だな	

Интересно понимать, как выглядит результат работы генератора шума. Предположим, на вход генератору было дано следующее предложение:

キャンペーンは終了致しました。

Тогда для различных шумов и режима "1 символ на предложение"получались такие результаты (**??**), которые затем фиксировались.

Шум	Текст
Эталон	キャンペーンは終了致しました。
KaGa	ギャンペーン**ば**終了致しました。
BigSmall	キ**ヤ**ンペーンは終了致しました。
Mix	**ギ**ャンペーンは終了致しました。

Таблица 2: Различные шумы

4.4 Статистика по шумовым символам

Посмотрим на то, где в общем распределении символов лежат шумовые символы. На всех графиках этого раздела пунктиром показана граница основного бакета B^* с $|\Sigma_{B^*}| = 4800$.

KaGa Рассмотрим, как выглядят **KaGa** шумовые символы на фоне всех остальных в корпусе (шкала логарифмическая, **??**).

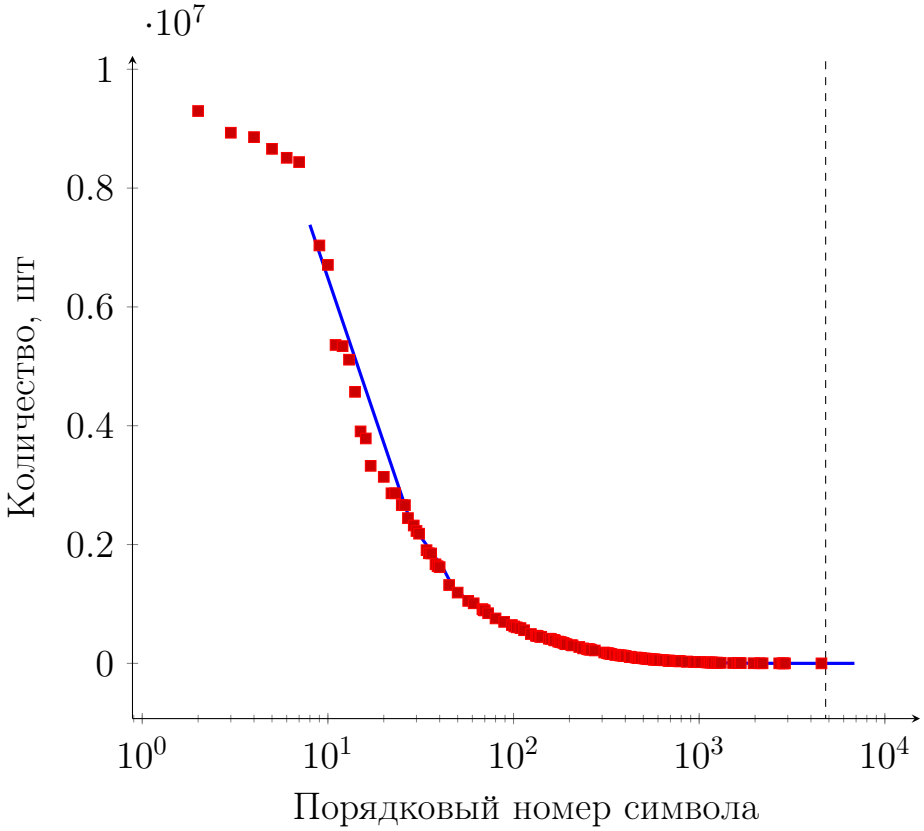


Рис. 11: Распределение частот шума **KaGa**

Видно, что шумовые символы лежат достаточно равномерно по кривой распределения, что может в будущем помешать оцениванию моделей для небольших размеров алфавита.

BigSmall Приведём аналогичный график для шума, состоящего из больших и маленьких кан (шкала логарифмическая, **??**).

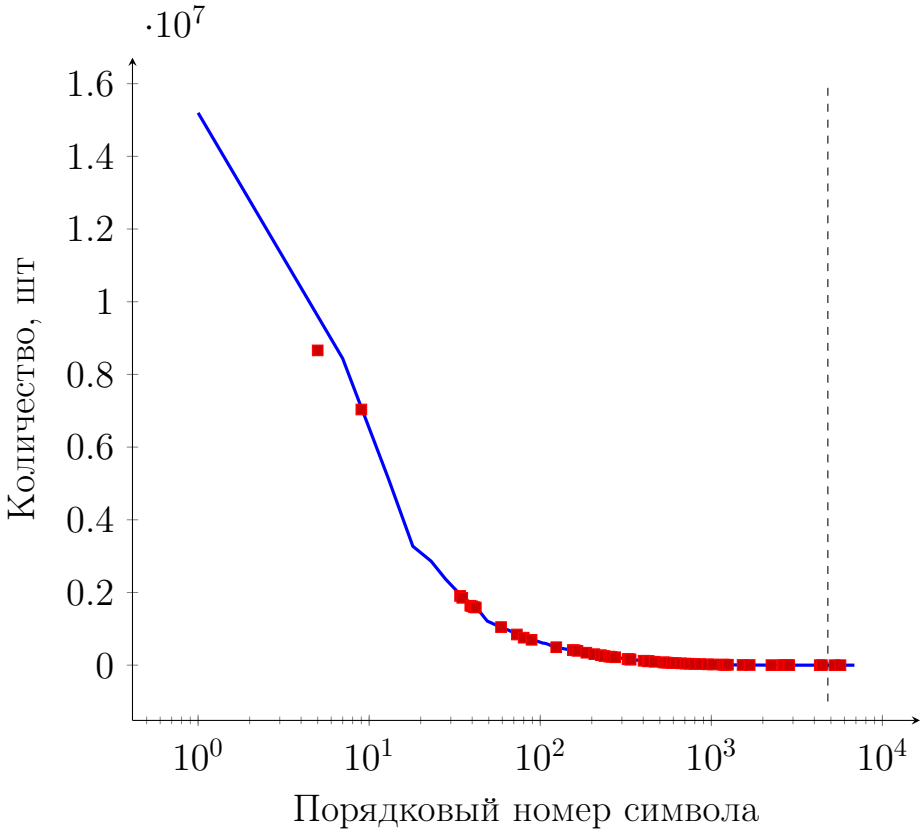


Рис. 12: Распределение частот шума **BigSmall**

В этом случае видно, что шумовые маленькие буквы в основном сконцентрированы в хвосте распределения. Поэтому для малых размеров алфавита работа с этим шумом бесполезна.

Mix Если смешать эти 2 шума (**KaGa** и **BigSmall**) и построить такой же график для смеси, то результат предсказуемо будет являть собой наложение двух предыдущих графиков (шкала логарифмическая, **??**)

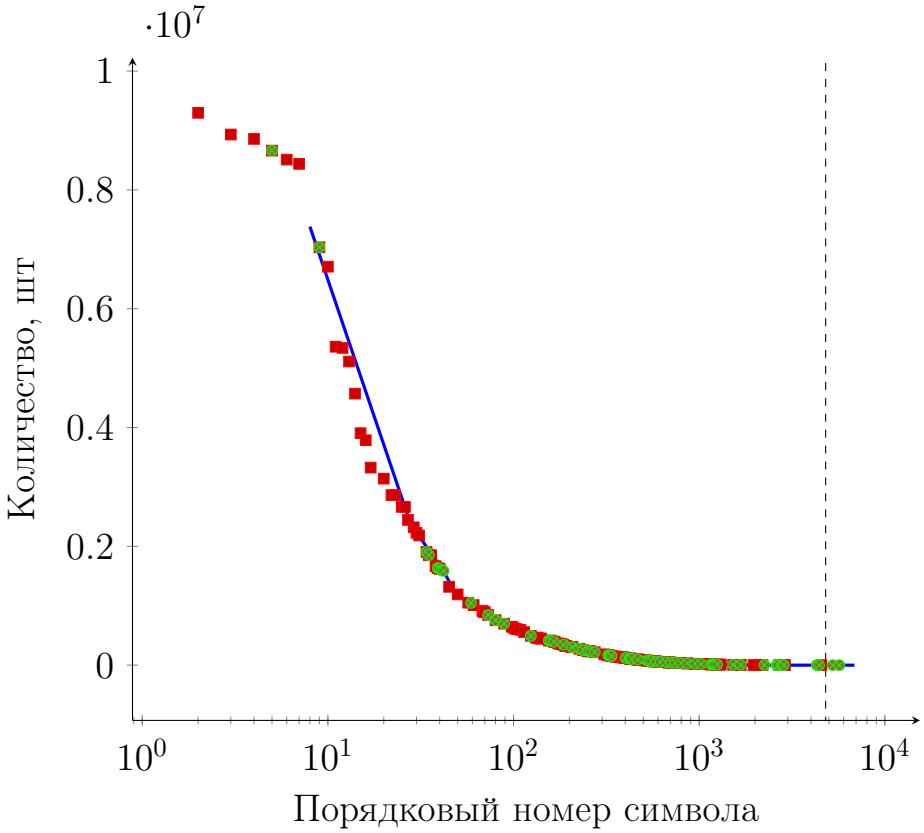


Рис. 13: Распределение частот шума **Mix**

4.5 Ход эксперимента

Глобально эксперимент состоял из следующих этапов:

1. Обучение модели:
   (a) Получение n-грамм для обучающей выборки;
   (b) Подсчёт статистики по n-граммам;
   (c) Сериализация статистики на диск;
2. Применение модели:
   (a) Десериализация обученной модели;
   (b) Получение n-грамм для тестовой выборки;

(c) Оценивание моделью различных вариантов написания для предложения;

(d) Вычисление оценки модели.

Определение 4.5.1. *Уверенность (confidence)* – порог отношения оценок величины $\frac{\max(\xi_1, \xi_2)}{\min(\xi_1, \xi_2)} = Confidence$, при достижении которого оценки ξ_1, ξ_2 величины ξ считаются неразличимыми.

При этом на этапе оценивания вклада результата работы модели на конкретном предложении (**??**) существовало 2 альтернативы:

1. $\Theta(S_{good}) > \Theta(S_{noise}) \Rightarrow$ на этом предложении модель отработала хорошо, иначе – плохо.
2. $\frac{\max(\Theta(S_{good}), \Theta(S_{noise}))}{\min(\Theta(S_{good}), \Theta(S_{noise}))} < Confidence \Rightarrow$ на этом предложении модель отработала уверенно, классификация проводится, далее см. **??**. Если же различие между оценками меньше порога $Confidence$, то результат – отказ от классификации этого случая.

4.6 Baseline эксперимента

В качестве baseline эксперимента были выбраны результаты работы униграммной модели (n-граммная при $n = 1$) (результаты показаны для основного бакета, $|\Sigma_{B^*}| = 4800$). Оценка производилась без учёта уверенности.

Для разных шумов baseline показал следующие результаты (**??**):

Шум	Оценка модели
KaGa	0.75
BigSmall	0.88
Mix	0.76

Таблица 3: Baseline

5 Реализация модели

Из-за обилия задач, связанных с обработкой текста в Unicode, а также по причине наличия удобных специализированных библиотек в качестве основного языка программирования был выбран Python 3. Впрочем, отдельные задачи, связанные с предобработкой корпуса, писались на Bash. Подробные спецификации использованного софта и аппаратные характеристики используемой машины см. в **??**.

5.1 Общее окружение: nltk, pygtrie

Поскольку в ходе эксперимента считались статистики по 7-граммам для корпуса из $79 \cdot 10^6$ предложений, было необходимо хранить статистики эффективно. Для хранения строк подобного вида лучше всего подходит такая структура данных, как префиксное дерево (бор, trie).

Поскольку задачи писать эффективное и масштабируемое префиксное дерево не было, была выбрана его реализация, предоставленная Google в библиотеке pygtrie (см. **?**, **?**).

Для получения n-грамм, а также различных вспомогательных задач был выбран популярный python-пакет nltk (Natural Language ToolKit, см. **?**)

5.2 Полезные утилиты

Pickle Удобный модуль для сериализации/десериализации сложных Python-объектов.

BeautifulSoup.UnicodeDammit Спасительный модуль для работы с разнообразными кодировками в составе пакета BeautifulSoup. Особенно мощно работает в связке с библиотеками chardet и cchardet. Без него привести корпус к читабельному виду было бы невозможно.

В качестве подтверждения – статистика по различным кодировкам в исходном корпусе (**??**):

Кодировка	Количество файлов
utf-8	68789
iso-8859-2	46870
shift_jis	42015
euc-jp	2562
cp932	1575
ascii	544
windows-1253	436
iso-8859-7	256
windows-1252	78
iso-8859-5	9
gb2312	4
tis-620	4
ibm866	1
maccyrillic	1
Всего	163144

Таблица 4: Кодировки в корпусе

Документацию можно найти в **?**.

6 Результаты эксперимента

Напомним условия эксперимента. Модели Ngram($n = 1$), Backoff($n = 5$) и Katz($n = 5$) запускались на тестовых выборках размером $\approx 300MB$, что соответствует $\approx 8 \cdot 10^6$ предложений, или же $\approx \frac{1}{10}$ части корпуса. При этом размер алфавита был равен $|\Sigma B^*| = 4800$, выборки были зашумлены 3 различными способами: **KaGa**, **BigSmall** и **Mix**.

6.1 Результаты для различных моделей

Первая часть эксперимента проводилась без оценивания уверенности ответов (см. **??**) и показала следующие результаты (**??**). В этой и последующих таблицах $M = Model, N = Noise, C = Confidence$.

$M \backslash N$	**KaGa**	**BigSmall**	**Mix**
Ngram(1) (Baseline)	0.75	0.88	0.76
Backoff(5)	0.89	0.93	0.90
Katz(5)	0.961	0.965	0.962

Таблица 5: Результаты эксперимента: accuracy

После того, как в первой части эксперимента хорошие результаты показала модель Katz(5), для неё были проведены испытания по поиску оптимального уровня уверенности $Confidence, C$:

$C \backslash N$	**KaGa**	**BigSmall**	**Mix**
0.9	0.86	0.94	0.86
0.95	0.979	0.988	0.981
0.97	0.966	0.986	0.967
0.99	0.94	0.98	0.95

Accuracy

$C \backslash N$	**KaGa**	**BigSmall**	**Mix**
0.9	0.53	0.43	0.52
0.95	0.61	0.69	0.62
0.97	0.77	0.85	0.77
0.99	0.91	0.94	0.91

Доля классифицированных

Таблица 6: Результаты эксперимента с $Confidence$

В итоге оптимальная конфигурация – модель Катца с $n = 5$ и уверенностью $= 0.97$ (Katz$(n = 5, C = 0.97)$). Она даёт высокие результаты исправления ошибок $(0.97 - 0.99)$, классифицируя достаточно большую часть выборки $(77 - 85\%)$. Результаты работы этой конфигурации в будут более детально проанализированы в **??**.

Приведём также часть статистик (Precision, Recall, F-measure, Accuracy) по конкретным символам (**??**):

ノ	2731	104	12	1366	0.995625228	0.963315697	0.979204016	0.972466176
う	37286	777	6	1210	0.999839108	0.979586475	0.989609183	0.980065684
つ	58704	3698	446	16500	0.992459848	0.940739079	0.965907595	0.947774361
ず	1912	4	110	34437	0.945598417	0.997912317	0.971051295	0.996873543
す	34437	110	4	1912	0.999883859	0.996815932	0.998347539	0.996873543
た	5420	34	74	14682	0.986530761	0.993766043	0.990135185	0.994656111
あ	68445	438	22	1852	0.999678677	0.993641392	0.996650892	0.993498876
お	164748	170	17	2812	0.999896823	0.998969185	0.999432788	0.998885226
人	24364	42	647	29013	0.974131382	0.998279112	0.98605743	0.987256316
ワ	1177	10	0	0	1	0.9915754	0.995769882	0.9915754
ヴ	0	0	2	393	0	0	0	0.994936709
ッ	665	7	41	2015	0.941926346	0.989583333	0.965166909	0.982404692
画	5778	12	2	1088	0.999653979	0.997927461	0.998789974	0.997965116
由	1088	2	12	5778	0.989090909	0.998165138	0.993607306	0.997965116
し	28300	212	18	572	0.999364362	0.992564534	0.995952842	0.992096763
じ	572	18	212	28300	0.729591837	0.969491525	0.832605531	0.992096763

Рис. 14: Статистики по символам

Из полной версии этой таблицы, чо в целом хирагана исправляется чаще (0.96-0.97), чем катакана (0.93-0.95). Это можно объяснить местами их употребления: хирагана используется для подчиняющихся правилам грамматических аффиксов, а катакана – для записи заимствованных слов, которые могут быть достаточно хаотичны. Полная версия таблицы в **??**.

6.2 Затраты ресурсов

Ресурсы делятся на время и память, затраты которых определяются, в нашем случае, глубиной бора n и размером тестовой выборки $size$.

Ниже представлены сводные таблицы по ресурсам (**??**) для различных моделей.

Модель	Место на диске	RAM
1-gram	100 KB	$\approx$ 1 MB
3-gram trie	178 MB	$\approx$ 7 MB
5-gram trie	1.5 GB	$\approx$ 40 GB
7-gram trie	3.7 GB	$\approx$ 110 GB

Таблица 7: Затраты ресурсов, память, $|\Sigma| = 4800$

Модель	10000 примеров	1000000 примеров
1-gram	$<$ 1 мин	$\approx$ 20 мин
3-gram trie	$\approx$ 1 мин	$\approx$ 1.5 ч
5-gram trie	$\approx$ 2 мин	$\approx$ 3 ч

Таблица 8: Затраты ресурсов, время, $|\Sigma| = 4800$

6.3 Статистики по обученным моделям

Обученные деревья даже для маленьких обучающих выборок оказываются весьма большими. Целесообразно привести статистику по 7-граммному

бору (цифры для более мелких ему соответствуют, т.к. n-граммный бор аддитивен). См. **??**.

n	Различных n-грамм
1	4430
2	482607
3	3436987
4	10025503
5	19051342
6	28679559
7	37723274

Таблица 9: Бор, $|\Sigma| = 4800$

7 Анализ результатов

В итоге оптимальной была признана модель Катца с глубиной бора $n = 5$, результаты которой ещё улучшились после добавления опции отказа от классификации при недостаточной уверенности (**??**).

$M \backslash N$	**KaGa**	**BigSmall**	**Mix**
Katz($n = 5, C = 1.0$)	0.961	0.965	0.962
Katz($n = 5, C = 0.97$)	0.966	0.986	0.967

Таблица 10: Результаты модели Катца

Проанализируем эти цифры глубже.

7.1 Сравнение с другими методами

Сравним наши результаты с полученными в работе **?**. В этой работе ошибки OCR исправлялись с помощью примерно такого же инструментария (n-граммы, Backoff, сглаживание). Несмотря на то, что эта работа является самой близкой по характеру, есть существенное отличие: в не йиспользовалась информаци о словном делении. См. **??**.

Модель	Accuracy
Katz	0.96–0.986
Nagata	0.97–0.98

Таблица 11: Сравнение с результатами **?**

Учитывая, что наш метод не использует информацию о словном делении, можно считать показатели модели Катца хорошими, осознавая при этом неравенство корпусов, моделей шума и прочей вспомогательной информации.

7.2 Анализ ошибок

Если доля правильных выборов – 97%, то доля неправильных – 3%. Чем можно объяснить 3 % ошибок?

Нехватка информации В условиях ручной генерации тестовых выборок (т.е. отсутствия отрицательных примеров для обучения) и n-граммной модели при рассмотрении эталонного и зашумлённого вариантов предложения зашумлённый всегда предполагается неверным. При этом, безусловно, существуют случаи, когда зашумлённый вариант встречается чаще, чем эталонный (значит, он наверняка возможен), из-за этого побеждает эталонный вариант при оценивании и решение классификатора считается неверным.

Приведём пример (3-граммная модель Катца), см. **??** (здесь и далее оценка приведена во внутренних единицах):

Тип	Предложение	Оценка
Эталон	ご要望**な**あわせて、勉強会を開催。	56
Шум	ご要望**に**あわせて、勉強会を開催。	63

Таблица 12: Пример ошибочной классификации

Рассмотрим процесс получения такой оценки.

Поскольку предложения отличаются 1 символом, а модель 3-граммная, то на итоговую оценку влияют следующие n-граммы:

Эталон:

- 要望**な** $\rightarrow 15$
- 望**な**あ $\rightarrow 7$
- **な**あわ $\rightarrow 22$

Зашумлённое предложение:

- 要望**に** $\rightarrow 0 \Rightarrow$ Backoff to bigrams

- 要望 → 45
- 望に → 23
- 望にあ → 6
- にあわ → 21

После применения штрафов за Backoff второе предложение всё равно выигрывает, решение считается неверным, но зашумлённое предложение встречается на самом деле.

8 Заключение

В работе проанализированы некоторые модели машинного обучения, основанные на символьных n-граммах, эти модели были сравнены в качестве решений для автоматического исправления ошибок OCR. Исследования были проведены для различных характеров ошибок OCR, которые имитировались в процессе эксперимента.

Также были проведены измерения производительности этих подходов с точек зрения времени и памяти. В силу особенностей японского языка затраты памяти на работу с обученной 7-граммной моделью впечатляют.

Однако, менее глубокие модели требуют разумных объёмов ресурсов, показывая при этом хорошие результаты, сравнимые с показателями аналогичных работ, в которых результаты были получены с привлечением информации о словном делении, а также POS-тегов.

Возможными направлениями дальнейшего развития данной темы могли бы стать исследования с реальными ошибками OCR, оптимизация структур данных с целью использовать более полные статистики (например, 7-граммное префиксное дерево), реализация других моделей и алгоритмов (например, сглаживание по Кнезеру-Нею (Kneser-Ney)).

Литература

Soumendu Das et al. Survey of Pattern Recognition Approaches in Japanese Character Recognition //
(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5(1), 2014. P. 93–99.

Nagata, Masaaki Japanese OCR Error Correction using Character Shape Similarity and Statistical Language Model //
Proceedings of the 17th International Conference on Computational Linguistics - Volume 2, 1998. P. 922–928.

Nagata, Masaaki Context-based Spelling Correction for Japanese OCR //
Proceedings of the 16th Conference on Computational Linguistics - Volume 2, 1996. P. 806–811.

S. Katz Estimation of probabilities from sparse data for the language model component of a speech recognizer. //
IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 35, no. 3, 1987. P. 400–401.

Pygtrie documentation [Электронный ресурс].
URL: http://pygtrie.readthedocs.io/en/latest/ – 2014

Pygtrie github [Электронный ресурс].
URL: https://github.com/google/pygtrie – 2017

Natural Language Toolkit [Электронный ресурс].
URL: http://www.nltk.org/ – 2017

Beautiful Soup Documentation [Электронный ресурс].
URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ – 2015

Tsai, Charlie Recognizing Handwritten Japanese Characters Using Deep Convolutional Neural Networks //
URL: http://cs231n.stanford.edu/reports/2016/pdfs/262_Report.

Update Registration 87 Japanese Graphic Character Set for Information Interchange [Электронный ресурс].
URL: https://web.archive.org/web/20150318013247/
http://kikaku.itscj.ipsj.or.jp/ISO-IR/168.pdf – 1992

Christopher D. Manning, Hinrich Schütze Foundations of Statistical Natural Language Processing // MIT Press, 1999.

EDR Japanese Corpus Version 1.0 [Электронный ресурс].
URL: http://www2.nict.go.jp/ipp/EDR/ENG/E_Struct/E_Struct-CPS.html – 1991

ATR Dialogue Database [Электронный ресурс].
URL: http://shachi.org/resources/3450 – 1990

Sungyup Chung and Keikichi Hirose and Nobuaki Minematsu N-gram language modeling of Japanese using bunsetsu boundaries //
INTERSPEECH, 2004.

Kudo, Taku and Yamamoto, Kaoru and Matsumoto, Yuji Applying Conditional Random Fields to Japanese Morphological Analysis //
Proceedings of EMNLP 2004, 2004. P. 230–237.

Graham Neubig et al. Machine translation without words through substring alignment //
ACL '12 Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1 , 2012. P. 165–174.

Приложение А. Условия эксперимента

А.1 Аппаратные характеристики платформы

Архитектура	x86-64
Частота процессора	3.30 GHz
Количество процессоров	32
Оперативная память	192 GB

А.2 Используемое программное обеспечение

Python	3.5.2 64-bit
nltk	3.2.4
pygtrie	2.1
BeautifulSoup	4.4

Приложение Б. Результаты модели Катца для символов

Б.1 Результаты модели Катца для символов

Symbol	TP	FN	FP	TN	Precision	Recall
な	20554	185	802	127893	0.96244615096459996	0.99107960846713905
だ	789	25	50	9853	0.94040524433849804	0.96928746928746901
ビ	1300	80	10	306	0.99236641221374	0.94202898550724601
ピ	403	6	67	712	0.85744680851063804	0.98533007334963296
ミ	481	12	19	1229	0.96199999999999997	0.97565922920892401
マ	1229	19	12	481	0.99033037872683305	0.98477564102564097
び	43633	9	59	117	0.99864963837773502	0.99979377663718405
ひ	229	63	5	21435	0.97863247863247804	0.784246575342465
の	497614	9348	10	6194	0.99997990450621299	0.98156074814285799
め	6194	10	9348	497614	0.398533007334963	0.99838813668600901
さ	10916	24	0	228	1	0.99780621572212003
ざ	228	0	24	10916	0.90476190476190399	1
で	42499	944	4	6954	0.99990588899607002	0.97827037727596999
て	6954	4	944	42499	0.88047606989111105	0.99942512216154
に	249152	1310	94	6978	0.999622862553461	0.99476966565786396
ぬ	52	0	541	123404	8.7689713322090995E-2	1
ス	15085	220	11	206	0.99927133015368297	0.98562561254491998
ヌ	170	0	99	3765	0.63197026022304803	1
シ	6379	152	122	10915	0.98123365636055904	0.97672638187107597
ジ	4498	99	49	2415	0.98922366395425498	0.97846421579290799
オ	2704	138	49	1149	0.98220123501634504	0.95144264602392603
エ	3296	127	24	1458	0.99277108433734895	0.962898042652643
り	4047	30	10	4588	0.99753512447621395	0.99264164827078705
ら	4588	10	30	4047	0.99350368124729305	0.99782514136581102
か	25338	40	262	23583	0.98976562499999998	0.99842383166522097
が	29978	864	31	20930	0.99896697657369404	0.97198625251280701
ン	4144	55	4	338	0.99903567984570796	0.98690164324839202
ヅ	2	0	17	2458	0.105263157894736	1
フ	3495	53	74	1457	0.97926590081255205	0.98506200676437405
ブ	499	60	54	1653	0.90235081374321802	0.89266547406082197
プ	2373	83	48	763	0.98017348203221799	0.96620521172638396
を	81025	1882	22	3931	0.99972855256826199	0.97729986611504405
ま	3931	22	1882	81025	0.67624290383622898	0.99443460662787697
ア	4610	492	30	6751 40	0.99353448275862	0.903567228537828
イ	21349	204	103	2670	0.99519858288271401	0.99053496033034805