

На правах рукописи

ЛОГАЧЕВА Варвара Константиновна

**ИССЛЕДОВАНИЕ И РАЗРАБОТКА
МЕТОДОВ АВТОМАТИЗАЦИИ ПРОЦЕССОВ
ПРАКТИЧЕСКОЙ ТРАНСКРИПЦИИ ИМЕН СОБСТВЕННЫХ**

05.13.11 — математическое и программное обеспечение
вычислительных машин, комплексов и компьютерных сетей

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:
кандидат технических наук,
доцент Э.С. Клышинский

Москва – 2013

Содержание

Введение	4
Глава 1. Анализ существующих систем практической транскрипции ...	12
1.1 Общая характеристика систем транскрипции	13
1.2 Ручное составление правил	15
1.3 Обучение модели транскрипции	17
1.3.1 <i>Выравнивание</i>	17
1.3.2 <i>Порождающие методы</i>	19
1.3.3 <i>Дифференциальные методы</i>	26
1.4 Соревнование систем транскрипции	29
1.5 Методы транскрипции.....	34
1.6 Смежные задачи	37
1.7 Выводы к главе 1	39
Глава 2. Метод порождения правил транскрипции	43
2.1 Формат правил	43
2.2 Предпосылки создания метода.....	44
2.3 Метод порождения правил.....	46
2.3.1 <i>Первичные правила</i>	47
2.3.2 <i>Сложные правила</i>	52
2.3.3 <i>Дальнейшее усовершенствование правил</i>	57
2.4 Выводы к главе 2	63
Глава 3. Метод транскрипции с помощью конечного автомата.....	65
3.1 Преобразование строки с использованием правил	66
3.2 Структура конечного автомата.....	68
3.3 Построение конечного автомата	70
3.4 Детерминированный конечный автомат	72
3.4.1 <i>Процедура преобразования НКА в ДКА</i>	74
3.4.2 <i>Унификация системы правил</i>	76
3.4.3 <i>Эквивалентность НКА и ДКА</i>	79
3.4.4 <i>Преобразование системы правил в ДКА</i>	81
3.5 Эквивалентность автомата системе правил	86
3.6 Усовершенствование конечного автомата	87
3.7 Скорость работы конечного автомата	89

3.8 Выводы к главе 3	91
Глава 4. Практическая реализация и оценка качества разработанных методов	92
4.1 Описание программной системы	92
4.2 Методы оценки качества систем транскрипции	99
4.3 Обучающие данные	102
4.4 Оценка метода порождения правил	104
4.4.1 Анализ правил транскрипции	104
4.4.2 Численные оценки результатов	109
4.5 Оценка метода транскрипции	112
4.6 Выводы к главе 4	114
Заключение	116
Список литературы	118

Введение

Необходимость правильной организации процессов хранения информации, связанная как с быстрым ростом объемов информации, так и с увеличением количества вовлекаемых предметных областей, привела к появлению в программировании и вычислительной технике целого ряда новых направлений. Это, например, электронный документооборот, а также технологии, сконцентрированные на организационных (CALS-системы¹) или технических (ILM², PDM³ и т.д.) аспектах хранения информации. Применение и развитие электронного документооборота позволило перейти к практической реализации серии крупных проектов (как государственных, так и коммерческих), существенно упрощающих жизнь и работу граждан.

Однако помимо хранения информации остро встает вопрос создания методов и алгоритмов ее обработки. Большинство хранимой документации представлено в текстовом виде, в связи с чем стоит задача разработки специализированных методов и алгоритмов: анализа и синтеза текстов на естественном языке; проверки полноты, корректности документов; проверки отсутствия грамматических ошибок; перевода и др. Так, например, международная торговля предполагает оформление документов, соответствующих международным стандартам, тогда как по российскому законодательству документы должны быть оформлены на русском языке, а значит, возникает необходимость их перевода с/на иностранный. Территориальное распределение технологических процессов требует ведения документации на нескольких языках, что в условиях огромных объемов информации также влечет за собой необходимость использования машинного перевода.

При переводе текста с одного языка на другой имена собственные, встретившиеся в этом тексте, также должны быть каким-то образом переведены, особенно если перевод осуществляется между языками, использующими различные системы письма. Иногда возможен перевод, если у имени есть лексическое значение. Например, мыс Доброй Надежды: африкаанс – Каар die Goeie Hoop, нидерландский

¹ CALS-система (англ. *Continuous Acquisition and Life cycle Support* – непрерывная информационная поддержка поставок и жизненного цикла) – система хранения информации о жизненном цикле продукта.

² ILM-система (англ. *Information Lifecycle Management* – управление жизненным циклом информации) – система управления электронными хранилищами данных.

³ PDM-система (англ. *Product Data Management* – система управления данными об изделии) – система для хранения документации о продукте.

– Каар де Годе Ноор, португальский – Cabo da Boa Esperança, английский – Cape of Good Hope, французский – cap de Bonne-Espérance. Такие случаи относительно редки, поэтому обычно используются другие методы.

Если перевод осуществляется между языками, пользующимися одним и тем же алфавитом, то имя может быть оставлено без перевода. Спорным случаем является использование в имени символов, отсутствующих в алфавите целевого языка. Например, многие языки пользуются различными вариантами расширенного латинского алфавита. При переводе с французского языка имя François, скорее всего, будет оставлено без перевода, хотя символ «ç» используется в алфавитах всего нескольких языков кроме французского (например, турецком и португальском).

Более актуальна проблема передачи иноязычных имен собственных из языков, использующих другую систему письма. На протяжении долгого времени их преобразование осуществлялось с помощью строгой транслитерации, то есть, сопоставлению каждой букве алфавита языка оригинала буквы алфавита целевого языка. У этого подхода есть серьезный недостаток – переведенные таким образом имена часто не сохраняют оригинального звучания в языке перевода. Однако почти до середины двадцатого века это не имело значения по нескольким причинам. Во-первых, не были развиты средства связи, передающие звуковую информацию (телефон, телевизор). Информация, получаемая из-за рубежа, была в основном текстовой, что требовало прежде всего графического, а не звукового сходства перевода имени с оригиналом. Во-вторых, звучание стало иметь значение, только когда помимо передачи иностранных имен на родной язык возникла необходимость передачи имен родного языка на иностранный (например, для оформления международных документов). В этом случае переводчик заинтересован именно в сохранении звучания (особенно, нам кажется, это касается фамильно-именных групп, так как во время пребывания за границей человек вынужден неоднократно называть свое имя, и лучше, если его графическая запись будет соответствовать звучанию).

Все это привело к тому, что к середине двадцатого века во всех сферах распространился новый подход – передача имен собственных с сохранением их звучания. Такой подход в российской лингвистике получил название практической транскрипции. Этот термин впервые применён в 1935 году А. М. Сухотиным [1] и

введён во всеобщее употребление А. А. Реформатским [2]. Практическую транскрипцию следует отличать от:

- фонетической транскрипции, основанной на точной передаче звучания с использованием специального фонетического алфавита;
- транслитерации, определяемой только исходным написанием;
- перевода.

В отличие от фонетической транскрипции, практическая транскрипция использует только символы алфавита языка-приёмника, а возможность введения дополнительных знаков отсутствует.

В зарубежной науке термин «transcription» является не столько лингвистическим, сколько биологическим. Процесс передачи слов некоторого языка средствами другого алфавита (не важно, сохраняется ли при этом оригинальное произношение слова) в англоязычном лингвистическом сообществе принято обозначать термином «transliteration».

Задача практической транскрипции первоначально решалась вручную путем составления систем правил транскрипции для различных пар языков. Такие наборы правил использовались в качестве методических указаний для переводчиков.

Как это отмечалось выше, распространение ЭВМ вызвало появление комплексных систем, решающих различные задачи обработки и хранения информации в автоматическом или полуавтоматическом режиме. В числе прочего перед такими системами ставится задача проведения автоматической транскрипции, например, больших списков имен, ручная обработка которых занимает много времени. Даже при небольших объемах применение средств автоматизации является полезным, так как в этом случае исключается влияние человеческого фактора: устраняется возможность совершения ошибок, допущенных по невнимательности, расхождения в правилах транскрипции, используемых разными пользователями и так далее.

Для этого необходимо разработать программную систему или подсистему, проводящую автоматическую транскрипцию. В связи с этим встает вопрос о принципах построения подобных систем. Первые исследования в этой области использовали уже имевшиеся наработки – составленные исследователями правила практической транскрипции (см., например, [3]). Таким образом, первые системы

машинной транскрипции просто применяли правила транскрипции, написанные вручную. Такие системы широко используются и сейчас, так как справляются со стоящей перед ними задачей и отвечают поставленным перед ними требованиям: транскрибируют имена быстрее и аккуратнее, чем человек. Редактируемые правила транскрипции имеют то важное достоинство, что они позволяют исследователю свободно расширять их список при обнаружении новых правил. Более того, каждое вновь введенное правило расширяет научное знание и может использоваться другими исследователями в дальнейшем.

Однако такая автоматизация процесса транскрипции часто недостаточна. Существующие на сегодняшний день руководства по практической транскрипции содержат правила транскрипции для сравнительно небольшого числа языков (справочник под ред. Гиляревского [4] – транскрипция между русским и 18-ю языками, справочник под ред. Ермоловича [5] – 23 языка, справочник под ред. Клышинского [6] – 33 языка). Ручное составление правил – однократный процесс, но он довольно долг и трудоемок. В условиях постоянно расширяющихся международных контактов, требующих составления правил между все новыми и новыми парами языков, требуется решение задачи автоматического создания правил транскрипции. В целях повышения качества обучения, иностранные исследователи отказываются от правил, записанных в явном виде, в пользу статистических моделей транскрипции. Автор данной работы придерживается противоположного принципа: автоматически порожденные правила должны быть представлены в явном виде, чтобы сделать возможным их ручное редактирование (плюсы которого кратко описаны выше). Но оба подхода едины в понимании того, что автоматическое извлечение правил транскрипции (в явном или в неявном виде) является обязательной частью системы машинной транскрипции. Таким образом, на современном этапе развития программных систем стоит задача разработки комплексных автоматизированных систем транскрипции, позволяющих не только транскрибировать имена собственные по правилам транскрипции, но и проводить автоматическое извлечение таких правил в ходе обучения.

Задача машинной транскрипции и автоматического извлечения правил транскрипции из множества прецедентов особенно актуальна в областях, где

требуется перевод имен собственных с большого количества языков. К таким областям относятся:

- составление электронных каталогов. В крупные библиотеки поступает много материалов на иностранных языках. Для внесения их в единый каталог часто бывает необходимо записать имя автора на русском языке.
- межъязыковой информационный поиск. В поисковом запросе может встретиться имя собственное, чье происхождение не совпадает с языком запроса (то есть, транслитерированное с иностранного языка), или имя, записанное средствами другого алфавита. Оба эти случая требуют применения практической транскрипции, так как иностранное имя должно быть транслитерировано на язык своего происхождения для возможности его поиска в документах на этом языке (если требуется информация о зарубежном ученом, велика вероятность того, что на его родном языке больше информации), а имя, записанное на другом языке, должно быть транслитерировано на язык запроса для попытки поиска информации на языке запроса.
- документооборот. Согласно российским законам, все организации, ведущие торговлю на территории России, должны иметь название, записанное символами русского алфавита.
- машинный перевод. Имена собственные, встречающиеся в тексте, при машинном переводе должны быть транслитерированы. Если система машинного перевода является расширяемой (то есть, предполагает добавление новых языков и направлений перевода), внедрение в нее подсистемы машинной транскрипции имен собственных будет гораздо более эффективным, чем ручное задание правил транслитерации для каждой новой пары языков.
- оформление паспортно-визовых документов. При оформлении визы для въезда в Россию имя въезжающего должно быть записано символами русского алфавита, причем с сохранением фонетического сходства.

В настоящей работе из задачи машинной транскрипции имен собственных намеренно исключена задача перевода географических названий. Дело в том, что передача имен этой категории по-прежнему основывается на традициях,

появлявшихся в разное время и потому с трудом поддающихся формализации. Одной из трудностей можно назвать неоднородное происхождение топонимов: например, на севере Англии очень много топонимов шотландского происхождения, которые записаны в соответствии с правилами шотландской орфографии и, соответственно, должны быть прочитаны и транслитерированы в соответствии с теми же правилами, несмотря на то, что официальный язык Великобритании – английский. Автоматизация же определения происхождения названия – отдельная задача, требующая довольно большого количества обучающих данных и далеко не всегда решаемая с приемлемым качеством. Другой проблемой географических названий является их долгая история. Названия многих современных городов появились несколько сотен лет назад и были записаны согласно действовавшему тогда правилам орфографии. За это время правила могли поменяться, а название – остаться в традиционном написании, которое не соответствует современному звучанию названия. Географические названия часто непригодны и при обучении системы практической транскрипции по некоторому множеству прецедентов (то есть, уже переведенных имен). Если название страны или города часто употребляется в иностранном языке (как, например, название *Москва* в новостных текстах на эстонском), оно «адаптируется» для удобства произнесения носителями этого языка и может отклониться от корректной транслитерации. Можно привести еще один пример из эстонского языка: столица Латвии Рига (лат. Rīga) в эстонском обозначается как Riia. Такая метаморфоза была бы объяснима, если бы в эстонском языке отсутствовал звук [g], но это не так.

По этим причинам рассматривается только практическая транскрипция фамильно-именных групп, причем только современных, так как имена исторических деятелей или литературных героев часто передавались (и передаются сейчас) на иностранный язык не по правилам практической транскрипции.

Таким образом, существует большое количество областей, в которых часто требуется транскрипция имен собственных с/на большое количество языков, причем имена поступают в большом количестве. Причем часто это фамильно-именные группы, которые, как уже было показано выше, обладают бóльшим, чем географические названия, единообразием. Это единообразие делает возможным

извлечение общих закономерностей транскрипции из множества прецедентов, то есть дает возможность автоматизировать не только применение существующих правил, но и порождение новых. Автоматическое обучение транскрипции часто является необходимостью в случаях, когда невозможно ручное: например, при необходимости транслитерации с редкого языка или при невозможности содержать штат экспертов-лингвистов.

Целью диссертационной работы является повышение эффективности обработки документов за счет автоматизации и ускорения процессов практической транскрипции (генерации правил и преобразования имен собственных по этим правилам).

В данной работе рассматриваются различные методы машинной транскрипции, их достоинства и недостатки, с целью показать актуальность создания метода автоматической генерации правил транскрипции в явном виде, а также создания нового метода автоматической транскрипции имен за линейное время с помощью автоматически сгенерированных правил.

Для достижения поставленной цели необходимо решить следующие **задачи**:

- Проанализировать существующие методы машинного обучения практической транскрипции с целью выявления возможности их практического применения;
- Проанализировать существующие методы машинной транскрипции с целью выявления технологий, обеспечивающих наиболее быстрое преобразование строк;
- Разработать метод автоматической генерации правил транскрипции на основе параллельного обучающего корпуса;
- Разработать метод транскрипции имен собственных по набору правил транскрипции за линейное время с помощью конечного автомата.

Основные проблемы, ограничивающие качество автоматической практической транскрипции, следуют из специфики методов обучения транскрипции. Алгоритмы, дающие на сегодняшний день лучший результат, являются статистическими. Эта особенность порождает сразу две проблемы: с одной стороны, необходимость использования больших массивов обучающих данных для обеспечения высокого качества, с другой стороны, неочевидность результата

транскрипции для пользователя. Статистические методы порождают модель транскрипции, недоступную для ручного редактирования, что делает ее негибкой. Модель, имеющая лишь незначительные недостатки, которые могли быть исправлены экспертом, должна быть признана неудовлетворительной, так как возможность коррекции отсутствует. В некоторых системах применяется альтернативный подход: ручное составление правил, – но он, не в пример методам машинного обучения, сложен и трудоемок. Методы автоматической генерации правил транскрипции в явном виде отсутствуют. К тому же, отсутствуют методы применения таких правил.

Для преодоления этих трудностей автором была разработана методика создания системы транскрипции с неизвестного языка по прецедентам. С использованием данной методики становится возможной автоматическая генерация правил транскрипции на основе сравнительно небольшого обучающего множества имен и их переводов для двух произвольных языков, причем правила генерируются в явном виде и в случае необходимости могут редактироваться вручную, а также применение этих правил для транскрипции строк с исходного языка на целевой, скорость транскрипции при этом не зависит от объема системы правил и линейна относительно длины преобразуемой строки. Программная реализация методики позволяет быстро получить относительно корректную модель транскрипции для пары произвольных языков даже при небольшом количестве обучающих данных и применить ее для передачи строк с высокой скоростью.

Содержание работы. Первая глава диссертации посвящена исследованию существующих на сегодняшний день работ в выбранной области: систем машинной транскрипции, методов автоматического обучения транскрипции, методов преобразования строк. Во второй главе изложен предлагаемый автором метод порождения правил транскрипции. Третья глава посвящена новому методу преобразования строк с помощью системы правил. Доказаны утверждения о корректности метода и о линейной скорости преобразования строк с помощью метода. В четвертой главе описана программная система, в которой реализованы предложенные методы. В главе представлены результаты экспериментов, подтверждающих эффективность предложенных методов.

Глава 1. Анализ существующих систем практической транскрипции

Сам термин «практическая транскрипция» введен в 1935 году, когда были предприняты первые попытки стандартизации передачи имен собственных на иностранном языке. До начала активного использования компьютеров при обработке текста задача практической транскрипции решалась посредством создания систем правил транскрипции между парой языков вручную (например, справочные пособия Гиляревского [4] и Ермоловича [5]). Эти системы правил были предназначены для филологов и переводчиков и служили и продолжают служить указаниями по ручному переводу имен собственных.

Когда потребовалось автоматизировать процесс практической транскрипции, первые решения были основаны на использовавшихся ранее правилах, составленных вручную. Они были, как правило, предназначены для фиксированной пары языков и не использовали никаких методов машинного обучения. Однако они решали поставленную задачу: избавляли от необходимости транслитерировать имена вручную. Такие системы пользуются популярностью и в настоящее время из-за относительной простоты их создания. В качестве примера можно привести системы [7, 8, 9], осуществляющие транскрипцию между наиболее популярными европейскими языками и русским.

Необходимость создания систем транскрипции для пар языков, правил транскрипции для которых не были ранее написаны, послужила толчком созданию методов автоматического извлечения правил из множества прецедентов. На сегодняшний день это направление является гораздо более перспективным, чем составление правил вручную, так как обеспечивает системе транскрипции гораздо большую гибкость.

Для решения задачи обучения систем транскрипции применяются как порождающие, так и дифференциальные методы машинного обучения, однако порождающие методы более распространены. Первые системы машинной транскрипции при передаче имен использовали промежуточное фонетическое представление, позднее были предложены методы, работающие с подстроками исходного и целевого имен напрямую.

В данной главе, основанной на работе автора [10 препринт-обзор], будут рассмотрены существующие на сегодняшний день подходы к проектированию и обучению систем машинной транскрипции, их достоинства и недостатки. Помимо этого, дан обзор смежных задач.

1.1 Общая характеристика систем транскрипции

Наиболее развитые системы машинной транскрипции, как правило, имеют следующую архитектуру: они состоят из подсистемы обучения транскрипции, принимающей на вход обучающие данные (как правило, двуязычный корпус⁴ имен) и возвращающей систему правил преобразования имен с исходного языка на целевой, и подсистемы транскрипции, которая с помощью этих правил преобразовывает строки (см. рис 1).

Обучение выполняется обычно в два этапа: выравнивание имен и порождение правил транскрипции. Под выравниванием в данном случае понимается сопоставление символам, подстрокам или звукам исходного имени символов, подстрок или звуков его перевода. Затем на основе установленных соответствий составляются правила.

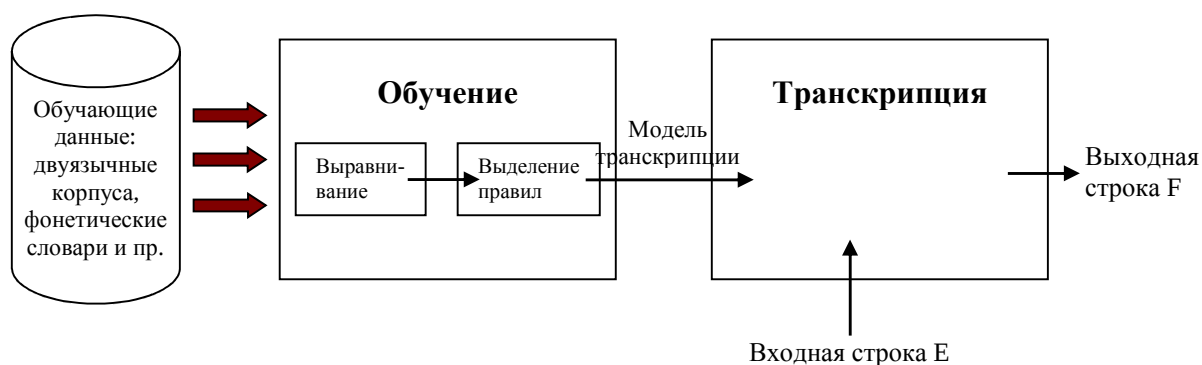


Рис. 1. Структура системы транскрипции

Системы машинной транскрипции создаются исследовательскими группами по всему миру. Эти системы используют различные методы обучения, характеризуются различными требованиями к объему и качеству входных данных,

⁴ Корпусом называют совокупность текстов, объединённых каким-либо общим признаком (языком, жанром, автором, периодом создания текстов). В данной работе под (параллельным) корпусом понимается набор имен собственных на некотором языке и набор их переводов на другой язык.

различными алгоритмами обучения и принципами построения. Системы машинной транскрипции можно классифицировать по следующим параметрам:

- Структура правил транскрипции. Преобразование строк может быть основано на сопоставлении фонем исходного языка фонемам целевого языка, или на сопоставлении символов или подстрок исходного алфавита символам целевого алфавита;
- Методы генерации правил. Самым распространенным подходом к обучению модели транскрипции является использование статистических методов машинного обучения: порождающих или, реже, дифференциальных. Применяются также алгоритмические методы – методы, предполагающие некоторую последовательность действий, и эмпирические методы – то есть, генерация правил вручную;
- Вид правил. Сами правила могут быть стохастическими, то есть, описывать межъязыковые соответствия фонем/подстрок с некоторой вероятностью (что характерно для большинства статистических методов), или детерминированными, то есть, предполагающими однозначные соответствия (такой вид имеют обычно правила, составленные вручную);
- Тип обучающих данных. В некоторых исследованиях используются созданные вручную корпуса имен и их переводов, другие извлекают информацию о переводе имени из двуязычных словарей терминов, из параллельных текстов. В качестве дополнительной информации привлекаются фонетические словари. Существуют методы обучения по одноязычному корпусу.

Первые системы машинной транскрипции использовали правила, основанные на соответствии фонем. Это объяснимо: поскольку требуется фонетическое сходство исходной строки и соответствующей ей строки целевого языка, при обучении требуется сопоставление звуков исходной и результирующей строк примера. К тому же, выбор фонетического соответствия может объясняться и экстралингвистическими причинами: к моменту возникновения задачи автоматизации транскрипции международным языком был английский, поэтому наиболее актуальной задачей был перевод с английского языка и на него. Однако английский язык обладает довольно сложной системой письма, в которой

соответствия графики и фонетики часто неоднозначны. Задача практической транскрипции напрямую с этого языка, без опоры на промежуточное представление, была слишком сложна для первых исследователей.

Первыми методами обучения транскрипции были эмпирические, иначе говоря, первые системы транскрипции не использовали методов машинного обучения для создания правил транскрипции – они просто автоматизировали применение имеющихся правил, составленных вручную.

1.2 Ручное составление правил

Начать обзор существующих на сегодняшний день систем транскрипции стоит с класса систем, отличающихся строением от схемы, приведенной выше. В этих системах первый этап – этап обучения – заменен ручным составлением правил. Такие системы были первыми опытами машинной транскрипции.

В работе Арбаби [11], одной из первых работ в области транскрипции, имена, записанные арабской вязью, преобразуются в звуковое представление, которое затем преобразуется в английские звуки, а после – в запись на английском языке. Передача с арабского на английский осуществляется по написанным вручную правилам. Это объясняется тем, что основной задачей этой работы было создание не правил передачи символов арабского алфавита латинскими буквами, а правил огласовки арабского письменного текста, что представляет главную сложность арабских имен (запись имен с указанными гласными с помощью символов латинского алфавита – довольно простая задача и, действительно, может быть описана небольшим количеством правил).

Работа [12] рассматривает транскрипцию имен между английским и французским языками. Здесь, как и в работе Арбаби, производится транскрипция через промежуточное фонетическое представление. Для преобразования от фонем к звуковому представлению используются методы задачи синтеза речи.

Система «Транскриба» [6] также использует составленные вручную правила. В настоящее время она поддерживает 32 языка, причем возможен перевод в обе стороны для любой пары языков. Эта универсальность достигается за счет использования промежуточного фонетического представления. Правила транскрипции составляются вручную, но нет необходимости определения

соответствий для каждой пары языков: достаточно определить правила преобразования букв и подстрок языка в фонетическое представление (в промежуточный фонетический алфавит по возможности включили все звуки, имеющиеся в представленных в системе языках). Точность транскрипции с помощью этой системы, как и с помощью других систем, использующих созданные вручную правила, довольно высока (около 99%).

Еще одна российская система основана на сходных принципах [13, 14]. При передаче не используется фонетическое представление, то есть правила составляются для каждой пары языков. Еще одним недостатком системы можно назвать ориентацию на машиносчитываемый формат, не допускающий диакритических знаков. Невозможность использования диакритических знаков при записи имен и правил транскрипции заведомо снижает качество передачи, поскольку диакритики часто меняют звучание имени (ср. латышские фамилии Vilciņš – Вилциньш и Vilcins – Вилцинс. В машиносчитываемом формате «ņš» будет записано как «ns», и разница будет потеряна).

Интересен подход, представленный в работе [15], описывающей метод конверсии с таджикского языка, записанного кириллическим алфавитом, на персидский язык, записанный арабской вязью. Модель транскрипции комбинирует правила, определенные вручную, с машинным обучением. Задается список символов, которые передаются между языками единственным образом независимо от контекста (как правило, согласные буквы). Затем производится выравнивание на обучающем корпусе – однозначные символы из оригинала и перевода ставятся в соответствие друг другу, а цепочки неоднозначных символов, находящиеся между однозначными – соответствующим цепочкам из перевода. Работа является примером того, что машинная транскрипция может быть использована не только для передачи имен, но и для перевода между близкородственными языками.

Сходная проблема решается в работе [16] для двух алфавитов языка панджаби: шахмукхи, основанного на арабском алфавите, и гурмукхи. Транскрипция производится на основе соответствий символов двух алфавитов, а также некоторых правил, применяющих эти соответствия с учетом контекстов. И соответствия, и правила составлены вручную.

Таблица 1. Системы, использующие ручное составление правил

Автор, представленные языки	Качество
Арабский – английский [11]	Не указано
Французский – английский [12]	Не указано
Английский – персидский – английский (Karimi, 2006)	56.2%
Shahmukhi-Gurmukhi [16]	99%
«Иностранные имена и названия в русском тексте». 18 языков – русский (Гиляревский, 1985, [4])	Не указано
Основные мировые языки – русский (Бондаренко, 2009, [13])	Около 95%
32 языка – русский (система «Транскриба», Клышинский, 2009, [6])	99%

Системы, построенные вручную, обладают гораздо более высоким качеством транскрипции, чем системы, использующие методы машинного обучения (см. таблицу 1). Однако процесс ручного составления правил долг и трудоемок. Создание таких систем в настоящее время может быть оправданно, только если требуется транскрипция между одной парой языков (или, по крайней мере, небольшим количеством пар), а главное, если нет необходимости в быстром расширении системы.

1.3 Обучение модели транскрипции

Этап обучения системы транскрипции состоит из двух подэтапов: выравнивания и собственно обучения, то есть поиска межъязыковых соответствий звуков или подстрок.

1.3.1 Выравнивание

Обучающие данные представляют собой множество пар: имя на исходном языке и его перевод на целевой язык. Выравнивание – это сопоставление элементов (звуков или подстрок) исходного имени элементам целевого имени (см. пример 1).

Пример 1. Выравнивание

J	o	e	D	a	s	s	i	n
Дж	о		Д	а	с	с	е	н

Во многих статистических системах машинной транскрипции используются методы выравнивания, применяющиеся в статистических системах машинного

перевода. При обучении машинного перевода на параллельном корпусе текстов слова в исходном предложении сопоставляются словам перевода. Самые популярные методы выравнивания – модель IBM [17], использующая для настройки параметров алгоритм ЕМ (Expectation Maximization) [18], и выравнивание, производимое с помощью скрытой цепи Маркова [19, 20].

При применении для задачи транскрипции алгоритмов выравнивания, взятых из систем перевода, в качестве основной рассматриваемой единицы берется вместо предложения слово, а минимальной единицей становится не слово, а символ (или звук, если сопоставление ведется на фонетическом уровне). Во многих системах транскрипции выравнивание производится с помощью системы статистического пофразового машинного перевода GIZA++. Система GIZA была создана в университете Джонса Хопкинса в 1999 году [21], в 2003 году появилась ее усовершенствованная версия GIZA++ [22]. Система часто используется в работах по машинной транскрипции, так как может свободно использоваться для академических исследований.

Необходимо заметить, что некоторые особенности алгоритмов выравнивания, предназначенных для перевода, неприменимы к машинной транскрипции. Например, при выравнивании слов допускается изменение порядка их следования, тогда как передача имени на иностранный язык должна осуществляться с сохранением порядка звуков. Кроме того, при выравнивании слов бывает необходима вставка нулевых слов, которые сопоставляются словам в предложении другого языка, аналог которых в переводе отсутствует (в качестве примера можно привести глагол «быть» в русском языке, имеющий в настоящем времени изъявительного наклонения нулевую форму).

Некоторые исследователи уделяют этапу выравнивания больше внимания, считая, что повышение его качества улучшит качество всей системы. Первоначально за основу алгоритма выравнивания брался алгоритм Ковингтона, предназначенный для поиска в параллельном корпусе слов с одинаковым происхождением [23]. Этот алгоритм при сопоставлении звуков слов позволяет две процедуры: сопоставление звуков или пропуск звука (то есть, продвижение вперед только по одному из слов). Недостоверные процедуры: сопоставление гласному звуку согласного, пропуск звука – штрафуются. Наиболее вероятным выравниванием считается вариант с

минимальной суммой штрафов. В работе [24] этот метод применен к задаче транскрипции, в работе [25] усовершенствован: появилась возможность сопоставить несколько символов одному или нескольким – то есть уже нет необходимости оперировать со звуковым представлением, так как стало возможным сопоставление на уровне подстрок. Главный недостаток такого метода выравнивания – система штрафов, которые должны быть назначены вручную для всех возможных соответствий.

В работе [26] представлен алгоритм выравнивания фонетических последовательностей ALINE. Этот алгоритм сочетает в себе различные техники сравнения последовательностей символов, фонетическое сходство строк определяется на основе набора признаков. К сожалению, авторы не представили численных оценок этого алгоритма.

1.3.2 Порождающие методы

Если сформулировать задачу машинной транскрипции в терминах статистических методов, то по известному входному слову E требуется найти такое (слово на языке перевода), которое максимизирует вероятность $P(F|E)$. Статистические методы порождения правил транскрипции в основном опираются на формулу Байеса [27]:

$$\hat{F} = \operatorname{argmax}_F \frac{P(E|F) \times P(F)}{P(E)} = \operatorname{argmax}_F P(E|F) \times P(F)$$

где E – слово на языке оригинала, F – его перевод, $P(E)$ и $P(F)$ – модели языка оригинала и перевода соответственно.

Если требуется решить задачу обратной транскрипции (задачу восстановления имени на языке оригинала по имеющемуся переводу на другой язык), формула будет противоположной:

$$\hat{E} = \operatorname{argmax}_E \frac{P(F|E) \times P(E)}{P(F)} = \operatorname{argmax}_E P(F|E) \times P(E)$$

Можно разделить методы транскрипции на методы, использующие соответствия букв и подстрок и методы, использующие соответствия фонем. В первом случае правила транскрипции представляют собой правила перевода символов входного языка в символы выходного языка. Во втором случае правила ставят в соответствие звукам входного языка звуки выходного языка. Перед

применением правил проводится преобразование имени из буквенной записи в последовательность фонем. Затем эта последовательность фонем переводится в последовательность фонем целевого языка, которая затем преобразуется в буквенную запись на целевом языке.

Методы, основанные на соответствии фонем

Первые системы машинной транскрипции использовали именно методы, основанные на соответствии фонем. Такой подход отражает суть практической транскрипции: сохранение фонетического облика слова. Схема работы подобных систем показана на рисунке 2. Преобразование производится в три этапа: оригинал имени (**E**) записывается в виде последовательности фонем входного языка (**I_E**), которая затем переводится в последовательность фонем выходного языка (**I_F**), а эта последовательность, в свою очередь, преобразуется в буквенную запись на выходном языке (**F**).



Рис.2. Схема работы системы транскрипции, основанной на соответствии фонем

Одной из первых работ в рассматриваемой области стала система транскрипции с арабского языка на английский [11], уже упомянутая в предыдущем разделе. Правила транскрипции в ней составлены вручную, однако она работает по той же схеме, что и остальные методы, основанные на соответствии фонем (см. рис.2). То же можно сказать и о системе транслитерации с французского на английский [12], также упоминавшейся в прошлом разделе – используемый в ней подход к задаче транскрипции как к задаче синтеза речи работает по схеме на рис. 2.

Первой полностью автоматической (то есть, не требующей ручного составления правил ни на одном из этапов) системой транскрипции стала система Кевина Найта, впервые описанная в работе [28]. Предложенный им метод был положен в основу многих дальнейших разработок в этой области. Система Найта

выполняла обратную транскрипцию с японского языка на английский (то есть, транскрипцию английских имен, записанных японской азбукой катакана, на английский язык). Транскрипция производится с помощью цепочки статистических конечных автоматов, последовательно выполняющих все этапы транскрипции (рис. 3).

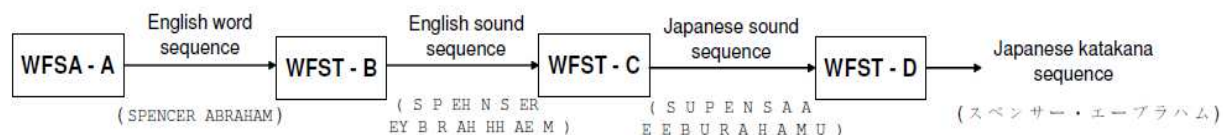


Рис.3. Структура системы К.Найта

Входной информацией для каждого автомата является выход предыдущего автомата. Сначала была сконструирована цепочка автоматов для осуществления прямой транслитерации: с английского языка на японский. Первый автомат делит входную строку на слова, второй преобразует слова в последовательность фонем английского языка, третий преобразует получившуюся цепочку фонем в цепочку фонем японского языка, пятый записывает японские фонемы с помощью азбуки катакана. Соответственно, строится четыре распределения вероятностей:

- $P(E)$ – модель языка;
- $P(I_E | E)$ – генерация цепочки фонем по слову английского языка;
- $P(I_F | I_E)$ – перевод последовательности английских фонем в последовательность японских фонем;
- $P(F | I_F)$ – запись последовательности японских фонем с помощью азбуки катакана.

Требуется построить цепочку автоматов, которая максимизирует произведение

$$P(E) \times P(I_E | E) \times P(I_F | I_E) \times P(F | I_F)$$

Обучение конечных автоматов производится с помощью алгоритма ЕМ (Expectation Maximization) [18, 29, 30, 31]. В качестве обучающих корпусов используются материалы американских печатных изданий, фонетический словарь университета Carnegie Mellon (Carnegie Mellon University Pronouncing Dictionary) [32], составленный вручную англо-японский корпус. Для решения задачи обратной транслитерации эта цепочка конечных автоматов применяется в обратном порядке,

преобразуя запись на азбуке катакана в английское имя, которое с наибольшей вероятностью соответствует этой записи. Этот метод показал довольно высокое качество: система правильно переводила имена с японского на английский в 64% случаев, тогда как человек дал правильный ответ всего в 27% случаев.

Описанная модель, составленная из цепочки конечных автоматов, стала основой множества других методов. Этому способствовало, помимо прочего, создание Джонатаном Граелом свободно распространяемого инструмента CARMEL, реализующего цепочку конечных автоматов, которые могут быть обучены на данных пользователя [33]. В дальнейшем эта система использовалась во многих работах по машинной транскрипции.

Метод Найта также был применен для арабского языка [34, 35]. Правда, в связи с некоторыми особенностями фонетики арабского языка и недостаточным количеством электронных корпусов качество обучения составило всего 56%.

В работе [36] предложен метод обратной транслитерации с корейского на английский. Обучение производится с помощью скрытых Марковских моделей, реализованных в виде нейронной сети с прямым ходом. Определение наиболее вероятных кандидатов в правила производится по следующей формуле:

$$\begin{aligned}
 F &= \operatorname{argmax}_F P(F|E) \\
 &\approx \operatorname{argmax}_F P(f_1 f_2 \dots f_m | e_1 e_2 \dots e_l) \\
 &\approx \operatorname{argmax}_F P(f_1 f_2 \dots f_m) \times P(e_1 e_2 \dots e_l | f_1 f_2 \dots f_m) \\
 &\approx \operatorname{argmax}_F P(I_1 I_2 \dots I_m) \times P(e_1 e_2 \dots e_l | I_1 I_2 \dots I_m) \\
 &\approx \operatorname{argmax}_F \prod_j P(I_{t_j} | I_{t_{j-1}}) \times P(e_j | f_j)
 \end{aligned}$$

Качество транслитерации составило 56% (оценивался процент правильно переведенных слов).

В таблице 2 представлены результаты основных работ по транскрипции, использовавших промежуточное фонетическое представление.

Методы, основанные на соответствии фонем, имеют несколько недостатков. Во-первых, необходимость рассматривать фонетическое представление слова увеличивает количество преобразований. Каждое преобразование порождает ошибки, так что такая система менее надежна, чем система, работающая напрямую с буквенным представлением имени. Во-вторых, порождение правил преобразования во внутреннее фонетическое представление требует особых

обучающих данных – фонетических словарей, таких, как Carnegie Mellon University Pronouncing Dictionary. Такие данные для большинства языков недоступны.

Таблица 2. Сводные результаты работы методов обучения транскрипции, основанных на соответствиях фонем

Языки	Метод обучения	Качество обучения
С японского [28]	ЕМ-алгоритм	64%
С арабского [35]	ЕМ-алгоритм	32%
Корейский [36]	Нейронная сеть	56%
На корейский [37]	Скрытые цепи Маркова	55%
На корейский [25]		52%
Китайский [38]	Widrow-Hoff (метод наименьших квадратов)	83% MRR
На китайский [39]		87% CA
На китайский [40]		36%, 77% CA

Методы, основанные на соответствии подстрок

Для преодоления недостатков методов, использующих соответствия фонем были предложены методы транскрипции, переводящие имя на языке оригинала напрямую в буквенное представление на целевом языке. Эти методы не требуют специфических обучающих корпусов.

Одним из первых примеров транслитерации на основе соответствия подстрок была работа [41] 2000 года. Эта работа развивает метод транскрипции между английским и корейским [36], модель языка строится с помощью биграмм:

$$P(F) = P(f_1) \times \prod_{i=2}^m P(f_i | f_{i-1})$$

В этой работе качество было улучшено до 58%.

Метод n-грамм был также применен для транскрипции между английским и арабским языками [42], результат составил 68% правильно преобразованных имен.

Многие методы, упомянутые в предыдущей секции, были применены и для преобразования строк без промежуточного фонетического представления. В работе [43] представлен метод преобразования с помощью статистического конечного автомата, обучаемого алгоритмом ЕМ. Использование соответствий подстрок вместо соответствий фонем улучшило результат транскрипции с помощью данного метода на 11,9%.

В работе [24], посвященной транскрипции между корейским и английским, использован новый метод выравнивания, который, в отличие от многих предыдущих методов, позволяет вводить соответствия один ко многим. Для обучения транскрипции используются деревья принятия решений. Деревья строятся с помощью алгоритма ID3 [44]. Дерево принятия решений строится для каждой буквы алфавита: 26 для английского, 46 для корейского. В качестве контекстов для каждой буквы используются три предыдущих и три следующих. Метод был опробован на обучающем корпусе, состоящем из 7000 пар имен, точность прямой транслитерации (с английского на корейский) составила 44,9%, обратной – 34,2%.

Стоит обратить внимание на работу [45] хотя бы потому что в ней затрагивается проблема транскрипции между языками, использующими латиницу, которая традиционно считается более простой задачей, чем транскрипция между разными системами письма. Работа посвящена переводу терминов, имеющих в разных языках общее происхождение, но отличающихся написанием и произношением (например, слово «способность»: *saracity* в английском языке, *saracidad* в испанском и *Karazität* в немецком). Основой метода является вычисление обобщенной дистанции редактирования (или расстояния Левенштейна [46]) для соответствующих слов различных языков. Для каждой буквы алфавита исходного языка вычисляется наиболее вероятная операция редактирования (вставки, удаления, замены) в данном контексте. Рассматривается контекст из четырех символов:

$$P(F|E) = \prod_i P(f_i | e_{i-2} e_{i-1} e_i e_{i+1})$$

Модель реализована в виде взвешенного конечного автомата и опробована на 1617 словах на финском, датском, нидерландском, английском, французском, немецком, итальянском, португальском и испанском языках. Качество обучения составило 80 – 91% , качество транскрипции – 64-78%, что на 26% и 22% соответственно превосходит результат обучения с помощью простой дистанции редактирования, который был взят в качестве проверочного метода.

Таким образом, транскрипция может быть применена не только для передачи с одного языка на другой имен собственных, но и для улучшения качества машинного перевода: родственные слова (*cognates*) используются в системах

машинного перевода для улучшения качества выравнивания при обучении на параллельных текстах [47, 48].

В работе [49] представлен алгоритм, основанный на соответствиях согласных и гласных. В работе объясняется, что для арабского, а также других языков, использующих арабский алфавит (в том числе и для персидского), фонетические методы не дают хорошего результата из-за принятого в этих языках сокращения гласных на письме. Таким образом, восстановить фонетический облик по графическому довольно сложно. Методы, преобразующие запись на языке оригинала напрямую в перевод, показывают лучшие результаты. По этой причине под гласными и согласными понимаются не звуки языка, а буквы, обозначающие эти звуки. Слова и их переводы разбиваются на подстроки вида CV, CVC, CC или VC (где C – согласная буква, V – гласная), подстроки оригинала и перевода сопоставляются друг другу, на основе этих соответствий формируется модель транскрипции. Оценка качества проводилась на составленном вручную англо-персидском корпусе из 16760 пар имен. Качество транскрипции составило 51,6%.

В следующей работе [50] этот метод был улучшен: предложен новый алгоритм выравнивания (в первой версии использовалась система Giza++). Кроме того, усовершенствован алгоритм формирования последовательностей гласных и согласных: группируются последовательности букв одного вида (только согласные или только гласные). Испытания на том же корпусе показали качество 55,3%.

В работе [51] представлен алгоритм транскрипции с английского языка на арабский. Представлены два метода: нахождение соответствий подстрок с помощью динамического программирования (алгоритм Витерби [52]) и обучение конечного автомата. Конечный автомат признан более совершенным методом, так как с его помощью может быть реализована статистическая модель языка, а также он не учитывает правила с низкой вероятностью и поддерживает правила перехода подстроки в пустое множество. Система была обучена на корпусе из 2844 пар слов, тестовое множество состояло из 300 пар, модель языка была получена отдельно на 10991 парах. Результат работы системы в проценте правильно переведенных слов не предоставлен, среднее количество ошибок в переведенных именах (средняя дистанция редактирования) – 2,01.

Предпринимались попытки использования при транскрипции личных имен дополнительной информации: языка происхождения и пола [53]. Описанный в работе метод был назван методом «семантической транслитерации», модель транскрипции определялась следующей формулой:

$$P(F|E) = \sum P(F|E, l, g) \times P(l, g|E)$$

где l – язык происхождения имени, g – пол носителя имени. В случае отсутствия какой-либо информации об имени соответствующий компонент просто изымается из модели. Метод был опробован на корпусах японских, китайских и английских имен. Лучшими результатами применения метода стали 49,4% правильно переведенных слов и 69,2% правильно переведенных символов, что хуже результатов применения к данным языкам других методов.

В таблице перечислены основные работы, основанные на соответствиях подстрок. В качестве основной меры использовалась мера Word Accuracy – процент корректно переданных слов тестовой выборки. Некоторые исследователи используют также меру Mean Reciprocal Rank (MRR), оценивающую ранжированный список строк в зависимости от того, насколько высоко в списке правильный ответ.

Таблица 3. Сводные результаты качества работы методов обучения транскрипции, основанных на соответствиях подстрок

Языки	Метод обучения	Качество обучения
Корейский [41]	Марковская модель	58%, 41% (b)
Корейский [24]	Дерево принятия решений	48%, 35% (b)
Арабский [42]	ЕМ-алгоритм	69%
9 европейских языков [45]	Сопоставление расстояний редактирования	70 RR
С арабского [51]	Алгоритм Витерби	2,01 – расстояние редактирования
Персидский [49]	Сопоставление согласных и гласных	74%
Арабский [43]	Алгоритм ЕМ	54.3%
Японский, китайский [53]	Дополнительная информация об имени	49,4% WA 69,2% CA

1.3.3 Дифференциальные методы

Другую разновидность статистических методов обучения составляют дифференциальные методы, использующиеся обычно для решения задач классификации. Поскольку машинная транскрипция не является задачей

классификации, использование таких методов для ее решения затруднительно, поэтому они не получили широкого распространения. Однако в их применении есть некоторые достоинства. При использовании дифференциальных методов нет нужды в этапе выравнивания: выходная последовательность порождается напрямую из входной последовательности.

В работе [54] задача транскрипции рассматривается как задача моделирования последовательности переходов. Обучение производится с помощью алгоритма обучения персептрона, основанного на модели, предложенной в [55]. Модель транскрипции определяет наиболее вероятную последовательность операций замены, удаления или добавления символа. Каждой такой операции для всех символов алфавита в начале обучения присваиваются веса, которые редактируются на основе обучающего множества имен.

Транскрипция была проведена для языковых пар арабский-английский и мандарин (китайский) – английский. Обучающие данные: список из 10084 фамильно-именных групп, распространяемый лингвистическим консорциумом (Linguistic Data Consortium (LDC)). Каждая запись представляет собой арабское имя, преобразованное в ASCII по системе SATTS [56], и его транслитерацию на английский язык. Тестовое множество – 1000 имен. Качество транскрипции (мера Top-1) с помощью модели, обученной предложенным методом, составляет 0,552 (для сравнения берется результат, полученный системой машинного перевода RWTH [57], для него качество составило 0,528).

Эту работу продолжает исследование [58]. Обучение производится с помощью алгоритма MIRA (Margin Infused Relaxed Algorithm, описан в работах [59, 60, 61]) . Алгоритм улучшает классический алгоритм обучения с помощью персептрона по трем параметрам:

- Получает k лучших вариантов транслитерации
- Обучение производится с помощью меры ошибки, заданной пользователем
- Пользователь может настроить параметры обучения для лучшего качества обработки зашумленных данных.

Получение k лучших вариантов возможно, так как алгоритм обучается на словах, для которых существует несколько вариантов транслитерации, отранжированных по заданной пользователем мере (напр., расстояние

Левенштейна). Такой подход хорошо сглаживает ошибки во входных данных, а также полезен для обучения на именах, для которых существует более одной возможной транслитерации. Результат работы усовершенствованного алгоритма на 2,2% лучше результата транскрипции, обученной с помощью первоначального алгоритма, обучение проводилось на том же корпусе имен.

Еще одна система, основанная на использовании персептрона, описана в [62]. Она сочетает в себе черты порождающего и дифференциального подходов. Перевод осуществляется с японского языка (азбука катакана) на английский. Данные – названия параллельных статей из Wikipedia на японском и английском. 56 килобайт пар названий используются для обучения, 2 килобайта составляют тестовое множество. В качестве эталонного метода берется метод, описанный в [51].

Другой дифференциальный подход, используемый для решения задачи машинной транскрипции, состоит в отображении последовательности символов на многомерные пространства признаков. В работе [63] представлены два дифференциальных метода обучения, один из которых использует локальную, а другой – глобальную модель структурированных выходных пространств. Первый, локальный, метод состоит в подборе линейных классификаторов, которые предсказывают символ перевода имени e_i по предыдущим предсказанным символам $e_1, \dots, e_{i-1}$ и оригиналу имени f . Глобальный метод состоит в подборе функции $W(f, e)$, отображающей пару строк (f, e) (оригинал и перевод имени) на множество действительных чисел. Функция линейна относительно набора признаков для пары (f, e) . Метод был применен к трем языковым парам: арабский - английский, корейский – английский и русский – английский. Обучающие данные для каждой языковой пары были разделены на две части: корпус, на котором проводилось обучение и тестовый корпус. Объем корпусов: 935 и 233 пар имен для арабского, 545 и 121 для русского и 11973 и 1363 для корейского. Корпуса были составлены вручную. В процессе обучения также строится вероятностная модель английского языка, для построения которой было использовано 100000 самых распространенных имен, загруженных с сайта US Census [64]. В работе использована вероятностная модель, основанная на 5-граммах с интерполированным сглаживанием Гуд-Тьюринга [65]. Для обучения локальных моделей использована модель персептрона,

описанная в [66]. Лучшие результаты для арабского, русского и корейского составили 32,61%, 46,28% и 35,28% соответственно.

1.4 Соревнование систем транскрипции

Качество различных методов обучения транскрипции с трудом поддается сравнению: разным группам исследователей доступны обучающие данные, отличающиеся по качеству и объему, в разных работах приняты разные меры оценки качества. Сравнение же двух методов, которые не применялись к одной и той же языковой паре, и вовсе лишено смысла – между тем большинство исследователей испытывает методы обучения только на одной паре языков. Возможно, это ограничение связано не с недостатками самих методов, а с недоступностью данных для других языков, однако оно делает невозможным проверку утверждения о том, что метод является независимым от языка.

В связи с этим в 2009 году было организовано соревнование систем машинной транскрипции, прошедшее в рамках конференции Ассоциации Компьютерной лингвистики [67] и получившее название Named Entities Workshop (семинар по именованным сущностям). Участникам предлагалось предоставить свою систему машинной транскрипции для обучения на одном или более из имеющихся параллельных корпусов имен. Было доступно семь корпусов: языком оригинала во всех корпусах был английский, языками перевода были: тамильский, хинди, русский, китайский, каннада (дравидийский язык, распространённый в юго-западной Индии), корейский и японский (азбука катакана). Все языки используют отличную от латинского алфавита систему письма, причем тамильский, хинди, каннада и японский – слоговое письмо, а китайский – идеографическое, что усложняет задачу транслитерации.

Результаты были оценены с помощью шести мер:

- Word Accuracy – точность транскрипции первого из предложенных системой вариантов (учитываются только полные совпадения);
- F-score – мера в данном случае оценивает, насколько отличается первый из предложенных системой кандидатов от ближайшего из эталонных вариантов

(в обучающем корпусе для одного имени может существовать несколько переводов);

- MRR – мера оценивает, какое место в списке выданных кандидатов занимает правильная транскрипция;
- MAP_{ref} – точность транскрипции для n первых выданных системой кандидатов, где n – количество эталонных вариантов транскрипции для данного слова (при $n = 1$ совпадает с Word Accuracy), то есть мера оценивает, насколько близко к началу списка оказались эталонные варианты транскрипции;
- MAP_{10} – точность транскрипции для первых 10 выданных системой кандидатов – то есть количество правильных вариантов среди первых 10 кандидатов (при единственном правильном варианте – наличие или отсутствие правильного варианта среди кандидатов);
- MAP_{sys} – точность транскрипции для всех выданных системой кандидатов.

Все меры нормализованы, то есть принимают значения от 0 до 1. Результаты транскрипции с помощью различных систем представлены в таблице 4. Для подавляющего большинства систем указано только 5 результатов: значение меры MAP_{sys} почти всегда совпадает со значением меры MAP_{10} , поскольку количество предлагаемых системой вариантов транскрипции редко превосходит 10. В случаях, когда указано три меры – это WA, F-score и MRR, в случае одной меры – WA.

Стоит отметить особенность использовавшихся корпусов. В них содержались не наиболее фонетически точные соответствия оригинального имени и перевода, а устойчивые имена одной и той же сущности, использующиеся в разных языках, чаще всего географические названия (например, Moscow – Москва). То есть задача модели транскрипции в данном случае – не передать имя средствами другого языка, как можно точнее сохранив его фонетический облик, а, имея запись имени на некотором языке, записать его на другом языке так, как оно обычно записывается в этом языке. При этом не обязательно, что какой-либо из двух языков будет языком происхождения имени.

Как видно из таблицы 4, популярностью пользуется обучение транскрипции с помощью систем машинного перевода, а также использование дифференциальных

методов обучения – например, обусловленные случайные поля. Из порождающих моделей чаще других встречается скрытая модель Маркова. Стоит также отметить другую тенденцию – во многих системах для обучения используется комбинация нескольких методов.

Таблица 4. Результаты соревнования систем машинной транскрипции. Ссылка на статью с описанием системы – в квадратных скобках перед описанием метода

	Тамильский	Хинди	Русский	Китайский	Каннада	Корейский	Японский
[68] Обучение - обусловленные случайные поля (Conditional Random Fields), статистическая модель выравнивания (алгоритм EM)	0.406 0.894 0.542 0.399 0.193 0.193	0.407 0.877 0.544 0.402 0.195 0.195	0.548 0.916 0.640 0.548 0.210	0.493 0.804 0.600 0.493 0.192	0.350 0.864 0.482 0.344 0.175 0.175		
[69] Выравнивание – скрытая модель Маркова, преобразование строк – вероятностный КА	0.327 0.870 0.458	0.398 0.855 0.515	0.506 0.901 0.609	0.450 0.755 0.514	0.235 0.817 0.353		
[70] Система машинного перевода MOSES [71] + повторное ранжирование результатов, фонетическая информация						0.451 0.720 0.576 0.451 0.181	
[72] Модель канала с помехами				0.652 0.858 0.755 0.652 0.232			
[73] Разделение английских имен на слоги, обучение с помощью системы SRI [74]				0.490 0.782 0.606 0.490 0.189			
[75] Обусловленные случайные поля + source channel model				0.713			0.510
[76] Метод минимальной длины описания (Minimum description length, MDL [77]) + отображение в многомерное пространство признаков	0.374 0.880 0.512	0.409 0.864 0.527	0.566 0.919 0.662	0.619 0.847 0.711	0.345 0.854 0.462	0.413 0.702 0.524	0.420 0.807 0.541
[78] Выравнивание на основе EM, обучение с помощью дифференциального алгоритма		0.509	0.608	0.746		0.387	0.505

Таблица 4. Продолжение

	Тамильский	Хинди	Русский	Китайский	Каннада	Корейский	Японский
[79] Модель исходного канала + модель, основанная на триграммах		0.389 0.831 0.487 0.385 0.16					
[80] Система пофразового статистического машинного перевода, использующая контексты		0.416 0.860 0.493 0.409					
[81] Система пофразового машинного перевода (выравнивание – Moses, обучение – GIZA++), настройка модели языка	0.41 0.89 0.54 0.40 0.18	0.42 0.86 0.54 0.42 0.18			0.36 0.86 0.48 0.35 0.16		
[82] Обусловленные случайные поля	0.365 0.884 0.504 0.360 0.172	0.363 0.864 0.503 0.360 0.170	0.531 0.912 0.635 0.531 0.219	0.580 0.826 0.653 0.580 0.199	0.324 0.856 0.438 0.315 0.148	0.170 0.512 0.218 0.170 0.069	0.457 0.828 0.576 0.445 0.194
[83] Обусловленные случайные поля + повторное ранжирование с учетом происхождения слова и использованием словаря		0.417 0.877 0.546 0.409 0.192			0.335 0.859 0.453 0.327 0.154		
[84] Система пофразового статистического машинного перевода, примененная к подстрокам имен			0.545 0.917 0.596 0.545 0.286	0.411 0.737 0.464 0.411 0.141			
[85] Система пофразового статистического машинного перевода + определение языка происхождения имени	0.404 0.883 0.539 0.398 0.182	0.366 0.854 0.493 0.360 0.164			0.335 0.856 0.457 0.328 0.154		
[86] Метод максимальной энтропии + обусловленные случайные поля + MIRA	0.474	0.483	0.605	0.731	0.398	0.473	0.537
[87] Скрытая Марковская модель, построенная на n-граммах произвольной длины				0.456 0.763 0.587 0.456 0.185			

Таблица 4. Окончание

	Тамильский	Хинди	Русский	Китайский	Каннада	Корейский	Японский
[88] Система пофразового статистического машинного перевода (GIZA++)		0.463 0.876 0.573 0.454 0.201					
[89] Система пофразового машинного перевода + двунаправленная модель транскрипции + эвристические правила	0.316 0.848 0.451 0.307 0.154	0.349 0.829 0.455 0.341 0.151	0.500 0.906 0.613 0.500 0.192	0.643 0.854 0.745 0.643 0.228	0.177 0.799 0.307 0.178 0.109	0.332 0.648 0.425 0.331 0.134	0.406 0.800 0.529 0.393 0.180
[90] Взвешенный перцептрон	0.475	0.465	0.609	0.672	0.396		
[91] Модели, основанные на п-граммах	0.436 0.894	0.415 0.858	0.597 0.925	0.646 0.867	0.370 0.867	0.476 0.742	0.503 0.843
[92] Преобразование через промежуточное фонетическое представление, метод обучения – перцептрон ([93])		0.466					0.469
[94] Графематическое приближение фонетического контекста				0.621 0.852 0.718 0.621 0.220			
[95] Метод обусловленных случайных полей, примененный к подстрокам с использованием контекста	0.435 0.902 0.572 0.430 0.195	0.418 0.879 0.546 0.412 0.183			0.363 0.870 0.482 0.355 0.164		
[96] Скрытая модель Маркова – обучение с помощью алгоритма ЕМ			0.354 0.869 0.394 0.354				

Наилучшие результаты были получены несколькими системами (они выделены в таблице цветом). Система [86], чьи результаты были лучшими для четырех представленных языковых пар, использует комбинацию методов обучения. В ней использованы две модели транскрипции: модель, основанная на подстроках, и модель, основанная на фонемах. Первая переводит подстроки исходного языка напрямую в подстроки целевого языка, вторая же использует промежуточное

фонетическое представление. Для обучения используются: метод максимальной энтропии [97], метод обусловленных случайных полей (CRF) [98, 99, 100], алгоритм MIRA, причем два последних метода реализованы только для модели, основанной на подстроках, а метод максимальной энтропии – для обеих моделей. Таким образом, авторы создали четыре метода транскрипции, которые были обучены на входных данных, а в качестве окончательного результата используется комбинация всех методов.

В работе [90] за основу взята модель на основе взвешенного перцептрона (он описан в уже упоминавшейся работе той же группы [54]). Метод был улучшен: наложены некоторые ограничения на редко встречающиеся n -граммы (запрещается генерация выходных последовательностей с n -граммами, не встретившимися или встретившимися недостаточное количество раз при обучении), а также вместо одной модели были обучены две: одна генерирует выходную последовательность слева направо, другая – справа налево, объединенный результат двух моделей при этом лучше первоначального.

Работа [78] использует метод выравнивания, обучающийся с помощью алгоритма ЕМ и описанный в [101]. Для обучения же используется дифференциальный метод: каждой паре строк из обучающей выборки ставится в соответствие вектор признаков, значения признаков обновляются с помощью алгоритма MIRA.

1.5 Методы транскрипции

Выше были рассмотрены основные методы создания модели транскрипции: ручное составление правил, автоматическое обучение модели по прецедентной базе с помощью различных методов машинного обучения. Следующей не менее важной задачей является задача применения этой модели: частью каждой системы транскрипции является метод преобразования строки исходного языка в строку целевого языка, корректное применение этой модели и приемлемую скорость преобразования.

Проблема скорости транскрипции имен редко рассматривается исследователями. Возможно, это связано с отсутствием необходимости в переводе больших списков имен: система машинной транскрипции часто является частью

другой системы (системы машинного перевода, информационного поиска и пр.), лишь небольшая часть входных данных которой нуждается в транскрипции. При необходимости переводить единичные имена скорость перевода не так важна. О скорости транскрипции упоминается в работе [63]: автор отмечает, что, поскольку многие приложения требуют решать задачу транскрипции в реальном времени, проверка правильности имени с помощью Internet не может быть использована, так как занимает слишком много времени. Однако проект «Перевод имен», представленный на сайте Яндекс.Нано [102], для ранжирования полученных вариантов перевода использует поисковую систему Яндекс (наиболее вероятным считается имя, для которого в результате поиска было найдено наибольшее количество страниц). Такой подход неприемлем при переводе больших списков имен, но данная система не ставит таких целей.

Для преобразования строк в большинстве методов, по примеру работы Найта [28], одной из первых в области машинной транскрипции, используются стохастические конечные автоматы. Конечный автомат обеспечивает линейную скорость обработки строк, то есть системы транскрипции с самого начала не испытывали сложностей, связанных со временем преобразования имен. Другим распространенным подходом является использование для транскрипции систем машинного перевода. Приверженцы этого метода обычно проводят эксперименты на уже готовом программном обеспечении, предназначенном для преобразования строк в режиме реального времени и использующем быстрые алгоритмы преобразования строк по обученной модели (некоторые алгоритмы поиска и декодирования, применяющиеся в статистических системах машинного перевода, описаны в [103]). Таким образом, в системах транскрипции, использующих статистические методы обучения, проблема скорости преобразования не возникает, поэтому метод транскрипции в работах зарубежных исследователей не рассматривается как отдельная задача.

Исследование возможностей и методов преобразования строк требуется в системах, основанных на правилах. Для первых систем машинной транскрипции, основанных на правилах, никаких данных о методе преобразования и скорости его работы не предоставлено – видимо, по той причине, что эти системы ставили целью проверить возможность автоматизации задачи транскрипции, вопрос

эффективности еще не был принципиальным. Но более поздние системы, основанные на правилах, столкнулись с необходимостью выбирать правила из множества, которое может быть довольно большим, причем процесс выбора правила должен повторяться для каждой преобразуемой строки 5-10 раз в зависимости от ее длины.

В системе «Транскриба» [6] используется метод преобразования, формирующий на основе разбираемой строки последовательность применимых к ней правил и последующим применением этих правил. Эта последовательность должна покрывать все символы строки, причем подстроки, разбираемые каждым правилом, не должны пересекаться (то есть не должно существовать подстроки, к которой применяются сразу два правила). Если же невозможно покрыть правилами всю строку (например, если она содержит символы, отсутствующие во входном алфавите), количество разрывов во входном слове должно быть минимально. Алгоритмически этот метод реализуется следующим образом: осуществляется поиск правил, применимых к слову с текущей позиции, затем из найденных правил выбирается правило с исходной подстрокой наибольшей длины, текущая позиция в слове сдвигается на длину исходной подстроки выбранного правила. Последовательность правил считается сформированной, когда достигнут конец разбираемого слова. Этот алгоритм используется в данной работе в качестве эталонного алгоритма, то есть результат его применения к произвольной подстроке является единственным корректным результатом. Сложность этого алгоритма – $O(n \cdot \log(M))$, где n – длина разбираемой подстроки, а M – количество правил в системе.

Другой, более вычислительно сложный метод применения правил к строке предложен в системе Бондаренко [13, 14]. Для разбираемой строки $w = \langle w_1, w_2, \dots, w_n \rangle$ составляется матрица подстрок размерности $n+1$, имеющая следующий вид:

#	w_1	w_2	...	w_n	#
# w_1	w_1w_2	w_2w_3	...	$w_n\#$	#
# w_1w_2	$w_1w_2w_3$	$w_2w_3w_4$	...	#	#
...	...	...	...	...	...
# $w_1w_2\dots w_n$	$w_1w_2\dots w_n\#$	#	...	#	#
# $w_1w_2\dots w_n\#$	#	#	...	#	#

Такая матрица содержит все подстроки разбираемой строки w . Преобразование осуществляется следующим образом: последовательно просматриваются все ячейки матрицы. Для подстроки в ячейке проверяется, существует ли правило, содержащее эту подстроку в качестве исходной. При обнаружении такого правила все подстроки, пересекающиеся с данной, удаляются из матрицы, правило запоминается. Таким образом, в матрице остаются только те подстроки, для которых существуют правила, пересечений применимых правил не происходит. Однако у этого метода есть существенный недостаток: его сложность составляет $O(n^2 \cdot \log(M))$, где n – длина разбираемой подстроки, а M – количество правил в системе.

Таким образом, все существующие на сегодняшний день методы применения правил к строке обладают полиномиальной скоростью и зависят от объема системы правил.

1.6 Смежные задачи

Ядром системы машинной транскрипции являются два метода: метод выделения правил транскрипции и метод преобразования строк по этим правилам. Однако не стоит забывать, что для возможности применения этих методов требуется подготовительная работа. Прежде всего, статистические методы обучения требуют больших обучающих корпусов. Ручное составление корпусов занимает слишком много времени, поэтому предпринимаются попытки **автоматизации сбора обучающих данных**. Для статистических моделей объем и качество корпуса имеют большое значение. Некоторые исследователи довольствуются корпусами, созданными вручную, другие используют двуязычные словари имен и терминов [28, 51, 63], получают параллельные примеры из двуязычных корпусов [104], из поисковых запросов [105], сравнивают термины из разных языков в фонетическом пространстве [106, 107]. В работе [108] описано обучение статистического конечного автомата на одноязычном корпусе.

При необходимости проведения транскрипции имен в режиме реального времени с большого количества языков часто необходимо предварительное **определение языка происхождения имени**. Для решения этой задачи используется два класса методов: методы, основанные на правилах, и статистические методы,

основанные на *n*-граммах [109]. Определение языка с помощью набора правил предполагает ручное задание некоторых характеристических свойств языка: если речь идет об определении языка происхождения отдельного имени, а не текста, это, как правило, сочетания символов, характерные или нехарактерные для данного языка. Так, например, символ *k*, характерный для немецкого языка, почти не используется в романских языках. Сочетание *zs* с большой вероятностью указывают на венгерское происхождение имени, *rz* – на польское. Главный недостаток этого метода в том, что такие правила охватывают далеко не все имена языка.

Методам, основанным на использовании *n*-грамм [53, 110], требуется обучающее множество имен для всех языков, которые должна распознавать система. В каждом имени обучающей выборки выделяются все подстроки длины *n*, затем проводится настройка коэффициентов, характеризующих каждую *n*-грамму (чаще всего в качестве коэффициента выступает вероятность появления *n*-граммы в обучающей выборке). Решение о принадлежности имени к тому или иному языку принимается на основе метрики сходства *n*-грамм анализируемого имени с множеством *n*-грамм, характерным для имеющихся в базе языков. Вид метрики зависит от конкретной реализации. Согласно результатам, сообщаемым в работе [111], такой метод с высоким качеством распознает язык происхождения имени в системе с небольшим (до пяти) количеством языков, однако при добавлении новых языков качество уменьшается. Было также установлено, что группировка языков по родству (языковым семьям или группам) не увеличивает качество определения языка имени.

Задача **обратной транслитерации** – восстановления имени на языке оригинала из его записи на языке перевода – является комбинацией задачи практической транскрипции и задачи восстановления правильного написания имени. Если при решении задачи практической транскрипции целью является выдача наиболее точной записи данного имени средствами другого алфавита, в случае обратной транслитерации это должна быть не только и даже не столько точная, сколько наиболее вероятная запись. Например, при обратной транслитерации имени «Джордж Буш», выделенном из текста о политике США, предпочтительным будет результат George Bush, а не Boosh или Boushe, хотя по правилам английского языка все три имени читаются одинаково и могут быть

переданы на русский язык одной и той же строкой «Буш». Именно эта задача положила начало череде работ по машинной транскрипции. К ранним работам в этой области относятся работы группы Найта [28, 34].

Некоторые специфические задачи не требуют генерации транскрипции имени, для их решения достаточно **выбрать правильную передачу имени на целевой язык из множества строк**. Такой подход возможен, например, при наличии параллельного корпуса на исходном и целевом языках, или хотя бы двух одноязычных корпусов, в которых с большой вероятностью появятся одни и те же имена собственные.

Эта задача является задачей классификации, поэтому для ее решения применялись дифференциальные методы, описанные в предыдущем разделе, например, обучение перцептрона [55], примененное к подбору транслитерации с арабского на английский. В системе [112] для обнаружения соответствующих друг другу пар имен используется SVM-классификатор. Имена извлекаются из корпусов новостей, в качестве дополнительной информации используются даты статей.

Решается эта задача и порождающими методами. В работе [113] применяется модель на основе расстояния редактирования, обучающаяся по алгоритму ЕМ. Она распознает японские имена и соответствующие им английские переводы, извлекаемые из поисковых запросов. Модель, описанная в [114], использует распределение вероятностей и фонетическое подобие для поиска подходящей транслитерации имен в английском и японском корпусах сходной тематики (сборнике аннотаций статей на английском и японском языках). Работа [115] также использует вероятностную модель, основанную на расстоянии редактирования, но для транслитерации с арабского на английский, имена-кандидаты выбираются из параллельного корпуса.

В работе [116] используется модель, основанная на подоби, подобие считается по разработанной авторами метрике. В качестве кандидатов используются словосочетания из корпуса n-граммов Google [117].

1.7 Выводы к главе 1

Автором были исследованы существующие подходы к решению задачи машинной транскрипции имен собственных. Задача стала актуальной относительно

недавно (первые работы относятся к 1994 году), но ее решению уже посвящено большое количество научных работ.

Следует отметить, что далеко не все языки охвачены существующими методами. Наибольшей популярностью пользуются китайский, японский, корейский и арабский языки. Это можно объяснить непривычностью написания этих языков для носителей языков, использующих латинский алфавит. К тому же, стоит признать, что эти языки представляют наибольшую сложность для транслитерации. Японское письмо (а точнее, азбука катакана, используемая для записи имен собственных) является слоговым, согласные не могут, в отличие от европейских языков, сочетаться с произвольными гласными, что создает трудности при транслитерации европейских имен на японский язык, а в особенности при их обратной транслитерации. Китайское иероглифическое письмо требует особого подхода, к нему часто неприменимы методы, подходящие для всех остальных языков. Арабский язык использует алфавит, но его письмо до сих пор сохраняет частичную консонантность, то есть гласные звуки, находящиеся в слабой позиции, на письме не передаются. По этой причине транслитерации с арабского языка должна предшествовать вокализация.

Целевым языком при транслитерации в подавляющем большинстве работ является английский язык. Это объясняется его популярностью и статусом международного языка, однако существует необходимость перевода и на другие языки.

Первоначально системы транскрипции были основаны на написанных вручную правилах, но с развитием методов машинного обучения системы правил, составленные вручную, были заменены на автоматически обучаемые модели транскрипции. Наиболее эффективными и популярными методами обучения транскрипции являются статистические порождающие методы. У статистических методов есть недостаток – их качество зависит от объема обучающих данных, но с появлением сети Internet и электронных корпусов текстов большие объемы данных стали доступны исследователям. Неоспоримым достоинством статистических методов является тот факт, что они не требуют знания обрабатываемого языка и введения в систему лингвистической информации. Сложно назвать метод машинного обучения, наилучшим образом подходящий для решения задачи

транскрипции. Разные типы и объемы данных, разные языки требуют разных методов обучения. К тому же, сравнение методов затруднительно из-за того, что все исследователи проводят эксперименты на разных множествах имен.

Соревнование систем машинной транскрипции, проводимое в рамках семинара по именованным сущностям (Named Entities Workshop), позволило лучше понять современное состояние исследуемой области: так, наименее трудоемким методом является использование для транскрипции систем статистического машинного перевода, а наилучший результат достигается комбинацией нескольких различных методов обучения. Кроме того, было показано, что дифференциальные методы обучения могут создать модель, преобразующую строки с достаточно высокой точностью, особенно в сочетании с другими методами.

Надо отметить, что подавляющее большинство описанных в главе методов решают специфическую задачу, в решении которой часто нет необходимости при создании системы машинной транскрипции. Основная задача, решаемая большинством зарубежных исследователей – узнать в имени, записанном на иностранном языке, некоторую известную им сущность (географическое название, имя политика и пр.). Между тем во многих сферах требуется решение другой задачи: как можно более точно передать звучание имени средствами другого языка. Очевидно, что решение этих задач требует принципиально различных обучающих данных и подходов.

Исследование методов транскрипции имен собственных показало, что из существующих моделей наилучший результат дают системы, построенные вручную или полученные с помощью статистических методов машинного обучения. Первый подход слишком долг и трудоемок. Что же касается второго подхода, он также имеет недостатки. Например, для достижения высокого результата требуется обучающая база большого объема и хорошего качества, которая во многих случаях недоступна. Однако более серьезным недостатком статистических методов обучения машинной транскрипции является невозможность ручного редактирования получаемой модели транскрипции. Ошибки транскрипции, произведенной с помощью такой модели, почти недоступны для анализа, для их исправления требуется переобучение всей системы, причем единственный способ исправить ошибки – расширить обучающую базу (при этом расширение базы не

всегда гарантирует улучшения результата). В целях получения контроля над моделью транскрипции и возможностью ее ручного редактирования было принято решение создать метод, автоматически порождающий по обучающему корпусу систему правил транскрипции, записанных в явном виде.

По этим причинам был создан предлагаемый в данной работе метод автоматического порождения системы правил, записанных в явном виде, которые при наличии в них незначительных недостатков могут быть исправлены вручную экспертом без повторного обучения всей модели и без расширения обучающей выборки. Кроме того, исследование выявило отсутствие эффективных методов преобразования строк с помощью таких систем правил. В данной работе предлагается также метод применения правил транскрипции, обладающий линейной скоростью, не зависящей от количества правил.

Глава 2. Метод порождения правил транскрипции

В главе описан метод порождения правил межъязыковой транскрипции, которые могут быть в дальнейшем использованы для автоматического преобразования строк или служить справочным материалом для лингвиста.

Метод основан на анализе двуязычного корпуса имен. Метод состоит из двух этапов: на этапе порождения первичных правил производится выравнивание имен по новому алгоритму, предложенному автором, затем на основе полученных соответствий составляются простые правила транслитерации. Правила, порожденные на первом этапе, используются для порождения сложных правил.

Метод не зависит от языка: он может быть применен для любой пары языков, при условии, что оба языка используют алфавитное письмо. Единственная дополнительная информация, необходимая для работы метода – список гласных букв входного и выходного алфавитов.

2.1 Формат правил

Тема настоящей работы – передача слов с одного языка на другой. Таким образом, описываемый метод работает с двумя языками, один из которых является языком-источником (язык, из которого взято имя), другой – языком-приемником (язык, на который переводится имя). Алфавит языка-источника обозначим через V_I , алфавит языка-приемника – через V_O .

Целью работы является создание метода, позволяющего генерировать межъязыковые соответствия подстрок, другими словами, определить отображение множества подстрок языка-источника в множество подстрок целевого языка. Каждое из таких соответствий подстрок назовем правилом транскрипции.

Правила генерируются из обучающего множества пар имен исходного языка и их записей на целевом языке. Обучающее множество составляется вручную экспертом.

Определим правило как пару $r = \langle p, \beta \rangle$, где:

p – левая часть правила – преобразовываемая цепочка символов входного алфавита, возможно, с контекстами;

β – правая часть правила – цепочка символов выходного алфавита (возможно, пустая), которая соответствует левой части правила в целевом языке.

Левая часть правила определяется следующим образом:

$\mathbf{p} = \langle \mathbf{p}_l, \alpha, \mathbf{p}_r \rangle$, где $\mathbf{p}_l$ и $\mathbf{p}_r$ – левый и правый контексты соответственно, α – преобразовываемая строка, $\mathbf{p}_l = \{\gamma_1, \gamma_2, \dots, \gamma_n\}$, $\gamma_i \in \mathbf{V}_I^+$, $\mathbf{p}_r = \{\delta_1, \delta_2, \dots, \delta_m\}$, $\delta_i \in \mathbf{V}_I^+$;

Исходя из этого, правило *применимо* с текущей позиции, если на текущей позиции находится подстрока α , перед ней полностью представлена одна из подстрок из $\mathbf{p}_l$, после нее – подстрока из $\mathbf{p}_r$. Считается, что к входной строке последовательно ищутся применяемые правила. При нахождении такого правила текущая позиция сдвигается вправо на $|\alpha|$ символов, а на выход подается β (под обозначением $|\alpha|$ здесь и далее понимается длина подстроки α).

При обучении системы машинной транскрипции необходима дополнительная информация о правиле: как часто и в каких словах обучающей выборки оно встретилось. Поэтому при обучении используется расширенный формат правила: $\mathbf{r} = \langle \mathbf{p}, \beta, \mathbf{w} \rangle$, где $\mathbf{p}$ и β те же, что в основном формате, а $\mathbf{w} = \{ \langle w_1, pos_1 \rangle, \dots, \langle w_n, pos_n \rangle \}$, где w_i – слово из обучающего множества, к которому применимо данное правило, pos_i – позиция, на которой находится первый символ из $\mathbf{p}_l$ в слове w_i .

2.2 Предпосылки создания метода

В начале работы перед автором была поставлена задача порождения правил практической транскрипции имен собственных по обучающему множеству имен и их переводов. Очевидным решением представляется комбинаторный метод: перебор всех возможных соответствий подстрок оригинала и перевода слова. Но этот вариант был отвергнут из-за его сложности: количество всех таких сочетаний слишком велико и их анализ займет слишком много времени. Чтобы снизить количество вариантов, нужны априорные знания о транскрипции.

Было решено в качестве основы будущей системы правил использовать правила простой транслитерации: правила вида $S_I \rightarrow S_O$, где S_I – символ из исходного алфавита, S_O – символ из алфавита перевода. Эти правила порождаются на основе имен, у которых в оригинале и в переводе одинаковое количество символов. При этом каждый i -й символ из оригинала имени ставится в соответствие i -му символу из перевода имени (см. рис. 4). Данный подход описан в публикациях автора [118, 119].

K u r t ↓ ↓ ↓ ↓ K y p t	K o b a i n ↓ ↓ ↓ ↓ ↓ ↓ K o б е й н
---	---

Рис. 4. Получение правил строгой транслитерации

Однако одинаковое количество символов еще не гарантирует их соответствия. Полученные таким образом правила соответствия могут быть некорректными (см. рис. 5).

Г ж е л ь ↓ ↓ ↓ ↓ ↓ G z h e l	C a t e ↓ ↓ ↓ ↓ K e й т
---	---

Рис. 5. Некорректные соответствия символов в словах с одинаковым количеством символов в оригинале и переводе

По этой причине было предложено для создания первичных правил использовать не целые слова, а более мелкие единицы. В качестве такой единицы в данной работе используется слог. Необходимо заметить, что слог здесь понимается не в классическом лингвистическом значении минимальной артикуляторной единицы. Слог в данной работе – формальное понятие. Это подстрока, имеющая форму C^*V^+ (где C – согласная, V – гласная), то есть, состоящая из подстроки согласных (возможно, пустой) и следующей за ней непустой подстроки гласных. Граница слога проходит между буквами, левая из которых является гласной, а правая – согласной. Длина слога меньше длины целого слова, поэтому при сопоставлении символов оригинала и перевода слога меньше вероятность порождения некорректных соответствий. Правила простой транслитерации порождаются на основе пар слогов вида $CV \rightarrow CV$. После с помощью этих правил осуществлялся пробный разбор остальных слогов и дополнение системы новыми правилами.

Затем было обобщено первоначальное допущение о том, что в имени, содержащем одинаковое количество букв в оригинале и переводе, буквы находятся во взаимно-однозначном соответствии. Количество букв в записи имени на различных языках может не совпадать из-за того, что некоторые звуки могут быть выражены несколькими символами (диграфами, триграфами и пр.). Но вероятность совпадения количества звуков в оригинале и переводе имени весьма велика.

Поэтому было решено для создания первоначальной системы соответствий делить слова не на слоги, а на группы гласных и согласных, и ставить в соответствие группе из оригинала имени группу из перевода с тем же номером. Некорректные соответствия при подобном подходе почти не порождаются, поскольку ди- и триграфы редко состоят из символов разного типа, значит, границы звуков не разрывают границ выделенных групп.

2.3 Метод порождения правил

На рисунке 6 представлена общая схема работы алгоритма порождения правил транскрипции. На вход алгоритм получает обучающее множество: пары имен на языке оригинала и их запись на языке перевода, результатом работы алгоритма является система правил транскрипции имен собственных с языка оригинала на целевой язык.



Рис.6. Схема работы метода порождения правил

Алгоритм порождения правил транскрипции, описанный автором также в работах [120, 121], состоит из двух этапов:

1. Порождение первичных правил, включающее в себя разделение слов обучающей выборки на группы гласных и согласных и собственно порождение первичных правил;

2. Порождение сложных правил, включающее разделение слов на слоги и пробный разбор слогов, в результате которого к порожденным на предыдущем этапе правилам добавляются новые. Пробный разбор может быть повторен несколько раз.

2.3.1 Первичные правила

Первичными правилами называются правила транскрипции, получаемые с помощью простых операций, совершаемых на первом этапе. Порождение первичных правил основывается на нескольких предположениях.

Суть практической транскрипции состоит в том, чтобы передать слово исходного языка на целевой язык с сохранением фонетического облика. То есть, в идеальном случае слово на языке-источнике и языке-приемнике должны состоять из одинаковых последовательностей фонем. Зачастую это невозможно в силу того, что наборы фонем в разных языках различны. В этом случае фонема исходного языка, отсутствующая в целевом языке, передается ближайшей по звучанию фонемой. Например, отсутствующий в японском языке звук [l] при переводе на японский иностранных имен и терминов передается с помощью звука [r], потому что это единственный в японском сонорный неносовой согласный. Фонемный состав оригинала и перевода слова не всегда совпадает, однако можно с уверенностью сказать, что согласные фонемы языка-источника передаются согласными фонемами языка-приемника, а гласные фонемы – гласными.

Исключение составляют, пожалуй, только аппроксиманты (или глайды) – подтип сонорных согласных, при образовании которых речевой тракт принимает промежуточное положение между положением для образования гласных и согласных шумных звуков. Они считаются согласными, так как при их произнесении все же образуется шум, и преграды в речевом тракте мешают аппроксимантам достигнуть уровня звучности гласных звуков. Тем не менее, звуковые колебания у этих звуков более интенсивны, чем у остальных согласных [122]. Поэтому нередко, если в языке-приемнике отсутствует аппроксимант, который есть в языке-источнике, этот аппроксимант может передаваться как согласной буквой, так и гласной. Например, английский звук [w] при записи английских имен создает

неоднозначности транскрипции, разрешавшиеся по-разному в зависимости от господствующей традиции перевода: «Вильям» или «Уильям», «Ватсон» или «Уотсон» и пр. Но поскольку глайды составляют довольно небольшой процент фонем языка, на этапе порождения первичных правил можно пренебречь их особенностями и считать глайды согласными звуками.

Далее предположим, что в большинстве случаев гласный звук передается одной или несколькими гласными буквами, согласный звук – одной или несколькими согласными буквами. А поскольку ранее мы уже предположили, что каждой фонеме исходного языка соответствует фонема целевого языка того же типа (гласная или согласная), можно с некоторой степенью уверенности предположить, что каждой группе согласных букв в оригинале слова соответствует группа гласных букв в переводе слова, группе гласных букв – группа гласных. На этом предположении и основывается первый этап порождения правил.

Для каждой буквы $v \in \mathbf{V_I} \cup \mathbf{V_O}$ определим предикат $\text{isVowel}(v)$, который возвращает **true**, если буква является гласной и **false** в противном случае (таким образом, буквы русского алфавита «ь» и «ъ», не обозначающие никакого звука, а потому не относимые ни к гласным, ни к согласным буквам, а также символы начала и конца слова будут также рассматриваться как согласные, так же как и описанные выше глайды).

Разделим каждое имя из обучающего множества и его перевод на группы гласных и согласных. Каждая группа должна содержать буквы одного типа. Границы групп в слове $\tau = v_1 v_2 \dots v_n$ находятся между v_i и v_{i+1} такими, что $\text{isVowel}(v_i) \neq \text{isVowel}(v_{i+1})$.

Таким образом, каждое имя может быть представлено как пара $\mathbf{q} = \langle \mathbf{in}, \mathbf{out} \rangle$, где

$\mathbf{in} = \{in_1, in_2, \dots, in_n\}$ – множество непустых цепочек букв из алфавита $\mathbf{V_I}$, множество групп оригинала имени;

$\mathbf{out} = \{o_1, o_2, \dots, o_m\}$ – множество непустых цепочек букв из алфавита $\mathbf{V_O}$, множество групп перевода имени.

Для каждой пары $\mathbf{q}$ из обучающего множества, для которого количество групп в оригинале и переводе имени совпадают, порождается n правил транскрипции ($n = |\mathbf{in}| = |\mathbf{out}|$): $\mathbf{r}_i(\mathbf{p_l}) = \mathbf{r}_i(\mathbf{p_r}) = \emptyset$, $\mathbf{r}_i(\alpha) = in_i$, $\mathbf{r}_i(\beta) = o_i$, $i \in [1, n]$.

Иными словами, i -я группа оригинала имени ставится в соответствие i -й группе перевода, при соблюдении двух условий:

- (1) в оригинале и переводе одинаковое количество групп
- (2) в 1-й группе оригинала содержатся буквы того же типа, что и в i -й группе перевода.

Подобная операция возможна, так как справедливо

Утверждение 1 о генерации первичных правил.

Для возможности генерации первичных правил из пары **q** необходимо и достаточно, чтобы $|\mathbf{in}| = |\mathbf{out}|$ и $\text{isVowel}(v_1) = \text{isVowel}(u_1)$, $v_1 \in \mathbf{in}_1$, $u_1 \in \mathbf{o}_1$.

Доказательство.

Под возможностью использования пары **q** понимается совпадение типов букв всех соответствующих групп из оригинала и перевода, то есть выполнение условия $\text{isVowel}(v_1) = \text{isVowel}(u_1)$, $v_1 \in \mathbf{in}_i$, $u_1 \in \mathbf{o}_j$, $i = 1, \dots, |\mathbf{in}|$, $j = 1, \dots, |\mathbf{out}|$.

Через $\text{isVowel}(\mathbf{in}_x)$ будем обозначать тип букв, составляющих группу $\mathbf{in}_x$, через $\text{isVowel}(\mathbf{o}_x)$ – тип групп, составляющих $\mathbf{o}_x$.

По определению операции разделения слова на группы гласных и группы согласных для любого $\mathbf{in}_i$ из **in** $\text{isVowel}(\mathbf{in}_i) \neq \text{isVowel}(\mathbf{in}_{i-1})$ и $\text{isVowel}(\mathbf{in}_i) \neq \text{isVowel}(\mathbf{in}_{i+1})$, то есть для любой группы тип составляющих ее букв не равен типу букв, составляющих соседние группы. Для любой группы $\mathbf{o}_i$ из **out** это утверждение также верно.

Таким образом, если $\text{isVowel}(\mathbf{in}_i) = \text{isVowel}(\mathbf{o}_i)$, то $\text{isVowel}(\mathbf{in}_{i+1}) = \text{isVowel}(\mathbf{o}_{i+1})$.

Из этого следует, что если в данном слове w выполняется равенство $\text{isVowel}(\mathbf{in}_1) = \text{isVowel}(\mathbf{o}_1)$, то для него верно также, что $\text{isVowel}(\mathbf{in}_i) = \text{isVowel}(\mathbf{o}_i)$, где $i = 1, \dots, n$, n – мощность меньшего из множеств **in** и **out**. Если же $|\mathbf{in}| = |\mathbf{out}|$, то $n = |\mathbf{in}| = |\mathbf{out}|$, а значит, условие $\text{isVowel}(\mathbf{in}_i) = \text{isVowel}(\mathbf{o}_i)$ выполняется для всех $\mathbf{in}_i$ из **in** и $\mathbf{o}_i$ из **out**. Невыполнение начальных условий приводит к невозможности генерации правил на основе пары **q**. Таким образом, данные условия являются необходимыми. Пара **q** может не отвечать дополнительным условиям, то есть, начальные условия являются достаточными. $\square$

Применение данного утверждения на практике позволяет существенно сэкономить вычислительные затраты при нахождении кандидатов в правила, так как нет необходимости проверять соответствие типов букв во всех группах пары **q**.

Неравное количество групп в оригинале и переводе означает наличие нечитаемой буквы или сочетания букв (например, Kate → Кейт, Atheret → Атере, Hélène → Элен). Если в обучающем множестве содержится большое количество слов с неравным количеством групп, это может снизить качество обучения на первом этапе. В этом случае слово также может участвовать в порождении правил, пока соблюдено условие (2).

Образованные таким образом правила составляют множество **R**, множество кандидатов в правила (см. пример 2).

Пример 2. Пример порождения первичных правил.

R | u | gg | ie | r | o M | a | cch | i

R | u | dzh | e | r | o M | a | kk | i

$r \rightarrow r, u \rightarrow u, gg \rightarrow dzh, ie \rightarrow e, o \rightarrow o, m \rightarrow m,$
 $a \rightarrow a, cch \rightarrow kk, i \rightarrow i$

Границы групп обозначены вертикальными линиями. Соответствующие группы из оригинала и перевода имени объединяются в правила.

Пример 3. Избыточность системы правил.

Возьмем систему правил шведско-русской транскрипции $R = \{a \rightarrow a, l \rightarrow л, e \rightarrow e, rs \rightarrow ш, rst \rightarrow шт, t \rightarrow т, i \rightarrow и, g \rightarrow г\}$. Эта система избыточна, так как для слова *Allerstig* → Аллештиг из обучающего множества существует две последовательности правил, с помощью которых оригинал может быть преобразован в перевод: $K_1 = \{a \rightarrow a, l \rightarrow л, e \rightarrow e, rs \rightarrow ш, t \rightarrow т, i \rightarrow и, g \rightarrow г\}$ и $K_2 = \{a \rightarrow a, l \rightarrow л, e \rightarrow e, rst \rightarrow шт, i \rightarrow и, g \rightarrow г\}$, $K_1, K_2 \subseteq R$.

На данном этапе множество **R** не может использоваться в качестве системы правил транскрипции, так как оно чаще всего избыточно и неоднозначно. Под избыточностью понимается существование для строки **ф** на языке оригинала и ее

перевода ψ более одной последовательности правил $r_1, \dots, r_n \in \mathbf{R}$ таких, что $\varphi = r_1(\alpha) + r_2(\alpha) + \dots + r_n(\alpha)$ и $\psi = r_1(\beta) + r_2(\beta) + \dots + r_n(\beta)$ (см. пример 3). Неоднозначной система правил называется, если для строки φ на языке оригинала существует более одной строки ψ на языке перевода, которая может быть получена применением к φ правил из $\mathbf{R}$ (см. пример 4).

Пример 4. Неоднозначность системы правил.

Возьмем систему правил французско-русской транскрипции $\mathbf{R} = \{ a \rightarrow a, l \rightarrow л, l \rightarrow ль, m \rightarrow м, e \rightarrow е, d \rightarrow д \}$. Эта система неоднозначна, так как для слова *Almeda* из обучающего множества существует два варианта перевода: Альмеда и Алмеда, – которые могут быть получены применением к оригиналу последовательностей правил $K_1 = \{ a \rightarrow a, l \rightarrow ль, m \rightarrow м, e \rightarrow е, d \rightarrow д \}$ и $K_2 = \{ a \rightarrow a, l \rightarrow л, m \rightarrow м, e \rightarrow е, d \rightarrow д \}$ соответственно, $K_1, K_2 \subseteq \mathbf{R}$.

Для избавления от избыточности из $\mathbf{R}$ удаляются следующие правила:

- Редкие правила – правила, встретившиеся в обучающем множестве менее χ раз. Велика вероятность того, что выделенное соответствие подстрок оригинала и перевода является исключением из правил или ошибкой эксперта. χ – параметр, который может быть настроен в зависимости от объема обучающего множества, в настоящей системе принято $\chi = 3$;
- Правила, которые могут быть объяснены с помощью других правил. Правило r_0 будет удалено, если найдутся такие $r_1, \dots, r_n \in \mathbf{R}$, что $r_0(\alpha) = r_1(\alpha) + r_2(\alpha) + \dots + r_n(\alpha)$ и $r_0(\beta) = r_1(\beta) + r_2(\beta) + \dots + r_n(\beta)$;
- Слишком длинные правила – правила, левая часть которых состоит более чем из ζ символов (в описываемой системе используется $\zeta = 3$). Скорее всего, такое правило может быть объяснено с помощью других правил, или будет объяснено на следующих этапах обучения. Исключение составляют правила, правая часть которых состоит из одного символа – они не могут быть разложены на более мелкие правила. Например, при сокращении множества $\mathbf{R}$ правило «ouia $\rightarrow$ уиа» будет удалено, правило «tsch $\rightarrow$ ч» – не будет.

Для избавления системы правил от неоднозначностей в неоднозначные правила вводятся контексты. Система правил, полученная на первом этапе, неоднозначна, если в ней присутствует хотя бы одна пара правил r_1 и r_2 таких, что

$r_1(\alpha) = r_2(\alpha)$, $r_1(\beta) \neq r_2(\beta)$ $r_1(p_l) = r_1(p_r) = r_2(p_l) = r_2(p_r) = \emptyset$. Появление таких правил может быть вызвано неоднозначностями правил чтения входного языка, но нередко правила чтения для данной буквы зависят от ее позиции в слове и соседних букв, и неоднозначности можно разрешить включением в правила контекстов (см. пример 5). Контекстами в данном случае называются буквы, которые предшествуют (p_l , левый контекст) строке α правила в словах, где встречается это правило, или следуют за ней (p_r , правый контекст).

Пример 5. Включение контекстов (на примере французского языка).

При порождении первичных правил генерируется два правила для буквы l : « $l \rightarrow л$ » и « $l \rightarrow ль$ ». Для разрешения неоднозначности, то есть определения случаев, когда должно употребляться одно из этих правил, к правилам добавляются контексты – символы, предшествующие строке и следующие за ней в именах обучающей выборки, где употребляется это правило (то есть, в именах, где на i -м месте в оригинале содержится группа « l », а в переводе – группы « $л$ » или « $ль$ »):

Примеры употребления правила « $l \rightarrow л$ »: Anjela – Анжела, Cella – Селла, Ecclesianne – Экклезианн, Flora – Флора, Hilange – Иланж, Lacene – Ласен. Правило, дополненное контекстами:
 $\{c, e, f, i, l, <\} l \{a, c, e, l, o\} \rightarrow л$;

Примеры употребления правила « $l \rightarrow ль$ »: Almelda – Альмельда, Algie – Альжи, Avital – Авиталь, Michel – Мишель, Dasilva – Дазильва, Elbert – Эльбер. Правило, дополненное контекстами:
 $\{a, e, i\} l \{b, d, g, m, v, >\} \rightarrow ль$

2.3.2 Сложные правила

Система правил, порожденная на первом этапе, для многих обучающих множеств является полной. Однако существует несколько типов правил, которые не могут быть сгенерированы с помощью деления слова на группы гласных и согласных и их анализа:

- Правила, строки α и/или β которых состоят из букв разных типов (гласных и согласных). Например, правило $qu \rightarrow к$ для франко-русской транскрипции;
- Правила для произносимых букв в конце или в начале слова.

Порождение таких правил осуществляется путем анализа слов обучающего множества с помощью уже существующей системы правил. Этап порождения сложных правил можно разделить на следующие подэтапы:

1. Разделение слов из обучающего множества на слоги;
2. Пробный разбор слогов, порождение правил;
3. «Склеивание» неразобранных слогов;
4. Возврат к пункту 2.

Разделение слов на слоги

Как это было отмечено выше, термин «слог» в данной работе используется не в классическом лингвистическом значении минимальной единицы речевого потока. Слог в данной работе – это группа согласных и следующая за ней группа гласных. За группой гласных может следовать группа согласных, если ею заканчивается слово. Слог может состоять также и из одной группы гласных, если это первая группа слова. Но наиболее частотная форма слога – группа гласных и группа согласных. Теперь, когда введен предикат `isVowel`, может быть дано формальное определение слога. Слог – это подцепочка слова, границы которой определяются по следующим правилам:

- Граница слога в слове $w = l_1l_2...l_n$ ставится между гласной и согласной, то есть после буквы l_i такой, что `isVowel( $l_i$ ) = true` и `isVowel( $l_{i+1}$ ) = false`;
- цепочка букв, среди которых нет гласной, не выделяется в отдельный слог, а присоединяется к предыдущему.

Слова обучающего корпуса делятся на слоги, так как слоги меньше целых слов, и при работе с ними легче выделить соответствия букв входного и выходного алфавитов.

Пробный разбор и порождение правил

На первом этапе алгоритма составляется система правил **R**, которая уже может быть использована для преобразования строк, пусть и не всегда результат ее применения к строке будет корректен с точки зрения правил чтения языка оригинала.

Эта система правил также используется и самим алгоритмом для порождения новых правил. Проводится пробный разбор слогов по правилам из **R** для выявления подстрок, не объясняемых системой правил.

Рассмотрим слог и его перевод $\lambda \rightarrow \mu$, где $\lambda = \langle u_1, \dots, u_n \rangle$, $\mu = \langle v_1, \dots, v_m \rangle$, $\lambda \in \mathbf{V}_I^+$, $\mu \in \mathbf{V}_O^+$. Требуется найти последовательность правил транскрипции $K = \langle r_1, \dots, r_k \rangle$, принадлежащих системе правил **R**, таких, что последовательным применением K к строке λ можно получить строку μ . В случае, если **R** не объясняет данный слог, требуется показать, что такой последовательности K для данного слога не существует.

Пробный разбор осуществляется следующим образом. Сначала слог разбирается слева направо. Ищется правило, левая часть которого совпадает с начальной подстрокой строки λ длины i , а правая часть – с начальной подстрокой строки μ длины j (где i и j – длина левой и правой частей правила соответственно) и не существует правила с левой частью длиннее i , удовлетворяющего тому же условию. Если существует несколько правил с левой частью одинаковой длины, применимых к разбираемой подстроке с начальной позиции, предпочтение отдается правилу с правой частью наибольшей длины. От строки λ «отрезается» начальная подстрока длины i , от строки μ – начальная подстрока длины j и разбор цепочек λ и μ продолжается с $i+1$ -го и $j+1$ -го символов соответственно.

Если достигнут конец цепочек λ и μ , значит, слог может быть объяснен системой правил **R**. В этом случае он больше не используется алгоритмом, так как с его помощью не удастся породить новых правил.

В противном случае слог аналогично разбирается справа налево: ищется самое длинное правило, левая часть которого совпадает с конечной подстрокой λ , правая – с конечной подстрокой μ .

Отметим, что при пробном разборе допускается нестрогое соответствие разбираемой строки правилу, а именно отсутствие в правиле нужного контекста. Если левая и правая части правила совпадают с начальными подстроками оригинала и перевода имени, но контекстов строки α в оригинале имени нет в списке контекстов правила, контексты добавляются в правило (см. пример 6). Таким

образом, пробный разбор служит не только для порождения новых правил, но и для расширения существующих.

Пример 6. Добавление контекстов в правило при пробном разборе.

Рассмотрим пробный разбор пары «Elphea – Эльфеа». Текущий символ – 2-й, то есть *l* в оригинале и л в переводе. В системе правил было найдено два применимых к строке правила: «{с, е}l{е, i} → л» и «{а, i}l{d, m} → ль». Левые части правил одинаковой длины, но правая часть второго правила длиннее, поэтому из двух правил выбирается второе. Контексты строки «*l*» в разбираемой строке – «*е*» и «*р*». У выбранного правила нет таких контекстов, поэтому они добавляются, и в результате разбора строки правило будет дополнено: «{а, е, i}l{d, m, p} → ль»

Если обозначить через $u_1, \dots, u_n$ и $v_1, \dots, v_m$ символы, объясненные правилами, а через λ_x и μ_x – подстроки оригинала и перевода слова соответственно, не объясненные правилами, слог, не полностью объясненный существующими правилами, может быть записан одним из четырех способов:

- $\langle u_1, \dots, u_i, \lambda_x \rangle \rightarrow \langle v_1, \dots, v_j, \mu_x \rangle$ – слог был частично разобран слева;
- $\langle \lambda_x, u_{i+1}, \dots, u_n \rangle \rightarrow \langle \mu_x, v_{j+1}, \dots, v_m \rangle$ – слог был частично разобран справа;
- $\langle u_1, \dots, u_i, \lambda_x, u_{i+1}, \dots, u_n \rangle \rightarrow \langle v_1, \dots, v_j, \mu_x, v_{j+1}, \dots, v_m \rangle$ – слог был частично разобран справа и слева;
- $\langle \lambda_x \rangle \rightarrow \langle \mu_x \rangle$ – слог не был разобран.

Пример 7. Слово, в котором границы правил и слогов не совпадают. В первой строке показано разделение слова на слоги. Во второй – границы применяемых правил транслитерации: $m \rightarrow m$, $y \rightarrow и$, $r \rightarrow p$, $ill \rightarrow ий$, $a \rightarrow а$. Одно из правил (*ill*–ий) находится на границе двух слогов: и левая, и правая часть его делятся границей слога на две части. Слоги этого слова не могут быть корректно преобразованы по отдельности.

М у | r и | l l а

М | у | r | и l l | а

М и | р и | й а

М | и | р | и й | а

Последовательность правил, применяемых к данной строке, создает разбиение этой строки на подстроки, каждая из которых является левой частью какого-либо правила. Если слог не был разобран целиком, значит, какие-то из правил, создающих разбиение этого слога, отсутствуют в системе правил. Если же слог был разобран только с одной стороны, то, возможно, границы слогов в слове, из которого был взят данный слог, не совпадают с границами правил, с помощью которых можно осуществить корректный перевод этого слова (см. пример 7).

По этой причине слоги, разобранные только с одной стороны или не разобранные вообще, не участвуют в порождении правил на данном этапе.

Слог же, частично разобранный с двух сторон, как уже говорилось, имеет форму $\langle u_1, \dots, u_i, \lambda_x, u_{i+1}, \dots, u_n \rangle \rightarrow \langle v_1, \dots, v_j, \mu_x, v_{j+1}, \dots, v_m \rangle$, где $\lambda_x \rightarrow \mu_x$ – подстрока с переводом, не удовлетворяющая ни одному правилу системы **R**.

Можно выделить три случая несоответствия слога правилам:

- $\lambda_x = \emptyset, \mu_x \neq \emptyset$. В систему **R** добавляется новое правило Γ_k такое что $\Gamma_k(\mathbf{p}_l) = u_{i-1}, \Gamma_k(\alpha) = u_i, \Gamma_k(\mathbf{p}_r) = u_{i+1}, \Gamma_k(\beta) = v_j + \mu_x$, другими словами, правило транслитерации для символа, предшествующего λ_x , который в некоторых контекстах передается двумя или более символами.
- $\lambda_x \neq \emptyset, \mu_x = \emptyset$. Такая ситуация значит, что некоторые символы из V_I в определенном контексте не читаются. К системе **R** добавляется правило Γ_k такое что $\Gamma_k(\mathbf{p}_l) = u_i, \Gamma_k(\alpha) = \lambda_x, \Gamma_k(\mathbf{p}_r) = u_{i+1}, \Gamma_k(\beta) = \emptyset$.
- $\lambda_x \neq \emptyset, \mu_x \neq \emptyset$. В этом случае добавляется правило Γ_k такое что $\Gamma_k(\mathbf{p}_l) = \Gamma_k(\mathbf{p}_r) = \emptyset, \Gamma_k(\alpha) = \lambda_x, \Gamma_k(\beta) = \mu_x$. Если в системе **R** есть правило Γ_q такое что $\Gamma_q(\alpha) = \lambda_x$ (то есть, если введение правила Γ_k в систему нарушает ее однозначность), то к правилу Γ_k добавляются контексты.

«Склеивание» слогов

В предыдущем разделе было показано, что можно с достаточной уверенностью породить на основе слога, разобранного не целиком, новое правило, только если этот слог частично разобран и справа, и слева. В противном случае существует вероятность того, что правило, которое должно быть применено к данному слогу, распространяется не только на этот слог, но и на соседний.

Необходимым условием того, что границы слога не разрывают правило, надо убедиться, что слог разобран с обеих сторон или что в соседнем слоге нет неразобранной подстроки, примыкающей к данному слогу.

Для этого применяется процедура агглютинации («склеивания») слогов.

Если слог не разобран справа, проверяется следующий за ним слог. Если следующий слог не разобран слева, то два эти слога объединяются в один слог, который будет частично разобран с обеих сторон и на основе которого можно будет породить новое правило. Если следующий слог разобран полностью (или, по крайней мере, разобран слева), значит, граница слога не разрывает правило, и для порождения нового правила достаточно будет текущего слога. Если же следующий слог вообще не разобран, он присоединяется к текущему и проверяются все следующие за ним слоги, пока не будет найден частично разобранный справа или полностью разобранный слог, чтобы процесс склеивания мог быть остановлен.

Заметим, что, поскольку у каждого слова обучающего множества есть маркеры начала и конца слова – «<» и «>», а в систему правил перед началом обучения добавляются правила $\leftarrow<$ и $>\rightarrow$, первый слог каждого слова является частично разобранным слева, а последний слог – частично разобранным справа.

После перегруппировки слогов повторяется их пробный разбор и порождение правил.

2.3.3 Дальнейшее усовершенствование правил

Как показано в работе автора [123], полученная в результате двух этапов система правил может также иметь недостатки. Несмотря на учет контекстов, она может оставаться неоднозначной. Под неоднозначностью понимается возможность передачи строки с использованием системы правил более чем одним способом. Контексты не снимают этой проблемы: возможно существование двух правил r_1 и r_2 таких, что $r_1(\alpha) = r_2(\alpha)$, $r_1(\beta) \neq r_2(\beta)$, $r_1(p_l) \cap r_2(p_l) \neq \emptyset$ и $r_1(p_r) \cap r_2(p_r) \neq \emptyset$, то есть, правил для одной и той же строки с различными правыми частями, множества контекстов которых пересекаются и не дают однозначного ответа на вопрос о том, какое правило должно быть применено в данном случае. У этой неоднозначности может быть две причины:

- неоднозначность правил чтения языка, двоякое чтение некоторой буквы или сочетания букв, объясняющееся сложившейся традицией, а не правилами (ср. англ. *sweat* – [swet], *bear* – [beə], *meat* – [mi:t]).
- недостаточность выделенных контекстов: возможно, на подстроку *a* влияет не одна, а две следующие за ней буквы (см. пример 8).

Пример 8. Двухбуквенные контексты.

Рассмотрим правила чтения буквы *i* во французском языке. В открытых слогах (оканчивающихся гласной) она читается как [i], соответственно, должна быть записана в русской транскрипции как «и». Но в закрытых слогах в сочетании с согласными *m* или *n* (диграфы *im*, *in*) эта буква образует носовой звук [ɛ̃], который при транскрипции на русский, в силу отсутствия в русском носовых звуков, будет записан как «эн». Чтобы слог был закрытым, нужно, чтобы он заканчивался на согласную, то есть, чтобы после *i* шло не менее двух согласных букв (одна из них будет принадлежать слогу, в котором содержится *i*, прочие – следующему слогу), или чтобы слог, содержащий *i*, был последним и при этом заканчивался на согласную. Если переформулировать это правило в формате, используемом в данной работе, получится два правила: $i\{m\#, n\# \} \rightarrow \text{э}$, $i \rightarrow \text{и}$, где # - любая согласная буква или символ конца слова.

С выделением контекстов, состоящих из двух букв, связаны некоторые трудности. Прежде всего, сложно отличить случаи неоднозначности от случаев недостаточности контекстов. В качестве усовершенствования системы правил, порожденной предложенным в главе методом, был предложен дополнительный этап – порождение сложных контекстов, то есть контекстов, состоящих из строк длиннее одного символа, для неоднозначных правил. Сложные контексты предполагалось порождать по алгоритму 1. Его основная идея состоит в том, что каждому неоднозначному контексту добавляется еще одна буква. Если же расширение контекста не снимает неоднозначность, правило остается без изменений.

При практической реализации данного алгоритма могут быть порождены ложные правила, в связи с чем целесообразным представляется введение некоторых дополнительных ограничений:

- дополнительный контекст добавляется только в одно из правил с пересекающимися контекстами;
- дополнительный символ добавляется только к одному из контекстов – правому или левому.

Эти ограничения, позволяющие снизить вероятность порождения неверных контекстов, использованы в дополненной версии алгоритма 1 – алгоритме 1б.

Алгоритм 1. Порождение сложных контекстов

Пусть $R_A = \{r_1, \dots, r_n\}$ – множество правил, таких, что $r_1(\alpha) = \dots = r_n(\alpha)$, $r_1(p_l) \cap \dots \cap r_n(p_l) \neq \emptyset$, $r_1(p_r) \cap \dots \cap r_n(p_r) \neq \emptyset$, иными словами, у правил совпадают левые части и существуют общие правые и левые контексты, то есть при преобразовании некоторых строк все правила будут применимы.

Напомним, что правило при обучении может быть записано как тройка $\langle p, \beta, w \rangle$, где w – множество пар $\langle w_i, pos_i \rangle$, определяющих слово w_i из обучающей выборки, в котором применимо данное правило, и pos_i , позицию в слове, с которой применимо правило. Напомним также, что позиция, с которой применимо правило в слове w_i – это позиция начала строки α в слове w_i , даже если у правила есть левый контекст. Обозначим через $w_i(pos_i)$ символ слова w_i , находящийся на позиции pos_i .

Тогда для каждой пары правил r_i, r_j из R_A :

1. Пусть $K = r_i(p_l) \cap r_j(p_l)$. $\forall \gamma \in K$:
 - если $\gamma = w_m(pos_m-1)$, $w_m \in w$, γ заменяется на $\gamma' = w_m(pos_m-2) + w_m(pos_m-1)$.
 - если после замены γ на γ' и пересчета $K \neq \emptyset$, γ' заменяется на γ .
2. Пусть $K = r_i(p_r) \cap r_j(p_r)$. $\forall \gamma \in K$:
 - если $\gamma = w_m(pos_m+|\alpha|)$, $w_m \in w$, γ заменяется на γ' , $\gamma' = w_m(pos_m+|\alpha|) + w_m(pos_m+|\alpha|+1)$.
 - если после замены γ на γ' и пересчета $K \neq \emptyset$, γ' заменяется на γ .

Алгоритм 1б

Требуется определить, к какому из двух неоднозначных правил будет добавляться дополнительный контекст, а также определить, какой из контекстов будет добавляться. Обозначим через K пересекающуюся часть пополняемого контекста (то есть те строки, к которым добавится дополнительный символ), через r_k – правило, к которому добавляются сложные контексты.

1. Выбор контекста: пусть $L = r_i(p_l) \cap r_j(p_l)$, $R = r_i(p_r) \cap r_j(p_r)$. Если $L > R$, $K = R$. Если $L < R$, $K = L$.
2. Выбор правила: пусть $K_i = r_i(p_l) \cap K$, $K_j = r_j(p_l) \cap K$. Если $|K_i| > |K_j|$, $r_k = r_i$, иначе $r_k = r_j$.
3. Добавление контекста: $\forall \gamma \in K$:
 - если $\gamma = w_m(pos_m-1)$, $w_m \in w$, γ заменяется на γ' , $\gamma' = w_m(pos_m-2) + w_m(pos_m-1)$.
 - γ заменяется на γ' только в правиле r_k .

В программной системе, где автором работы реализованы предложенные методы, данный алгоритм порождения сложных контекстов используется, однако остается экспериментальным. Эксперименты показали, что применение этого алгоритма увеличивает количество строк, переданных однозначно, но уменьшает общее качество системы.

Есть и другой недостаток в представленном методе порождения правил. При снятии неоднозначности (этап порождения первичных правил) к неоднозначным правилам добавляются и правый, и левый контексты, тогда как зачастую на то, как произносится буква, влияет только один контекст. К сожалению, по одному или нескольким прецедентам сложно определить, какой именно контекст имеет значение в данном случае. В редких случаях на букву могут влиять оба контекста (см. пример 9).

Пример 9. Влияние различных контекстов на чтение буквы (на примере французского языка).

Влияние правого контекста. Буква **c** после гласных переднего ряда (записываются буквами **e, i, y**) читается как [s]: **Nancy** – [nãsi], в остальных случаях как [k]: **Cannes** – [kan], **Clermont** – [klɛ̃mɔ̃].

Влияние левого контекста. Сочетание **ll** читается как [l], кроме случаев, когда ему предшествует **i**. В этих случаях все сочетание **ill** читается как [ij]: **Montpellier** – [mɔ̃pɛljɛ], **Gaillac** – [gaijak].

Влияние обоих контекстов. Буква **s** между двумя гласными читается как [z]: **Toulouse** – [tuluz], в остальных случаях, кроме положения в конце слова – как [s]: **Marseille** – [maʁsɛj], **Saint-Malo** – [sɛ̃malo]. Звук [s] между двумя гласными звуками записывается с помощью двойного **s**: **Carcassonne** – [kaʁkasɔ̃n].⁵

С помощью описанного в данной главе метода порождения правил можно определить, какой из контекстов в правиле является определяющим: в процессе разбора слогов с помощью существующих правил эти правила дополняются новыми контекстами, и при достаточно большой обучающей выборке во всех правилах с

⁵ Знаком ɥ в международной фонетической транскрипции обозначается увулярное «г», знаками ã, ã̃, ẽ̃ – носовые а, о и е соответственно.

контекстами незначимый контекст будет включать в себя все или почти все символы алфавита, то есть станет равносильным отсутствию контекста, в значимом же контексте останутся только те символы, которые меняют чтение левой части правила. При использовании большой обучающей выборки и алгоритм порождения сложных контекстов, представленный выше в текущем разделе, будет давать лучшие результаты, поскольку многие незначимые контексты будут удалены, и можно будет избежать появления шума и лишних пересечений контекстов.

Неоднозначна и процедура добавления к правилам простых контекстов. Она может привести к появлению неверных правил. Напомним, что для избавления от неоднозначности к правилам с совпадающей строкой α добавляются в качестве контекстов все символы, которые предшествуют строке α или следуют за ней во всех именах обучающей выборки, к которым применимо это правило. Однако, существует вероятность того, что какие-либо комбинации контекстов из разных имен могут оказаться неверными для этого правила (см. пример 10).

Пример 10. Некорректное пересечение контекстов (на примере французского языка).

Буква латинского алфавита **S** при транслитерации французских имен может быть передана, в зависимости от контекста, тремя способами: как «з», «с» или перейти в пустую строку. Рассмотрим случаи перевода **S** как «з»:

Deserai – Дезере

Gratieuse – Грасьез

Levisienne – Левизьенн

Нетрудно заметить, что во всех этих случаях **S** находится между двумя гласными буквами. Обозначим множество всех гласных букв алфавита **V_I** через **V_{Iv}**, а множество всех согласных букв – через **V_{Ic}**, то есть **V_I = V_{Iv} ∪ V_{Ic}**. В результате добавления к правилу **S → з** контекста на основе обучающего множества будет получено правило **V_{Iv} S V_{Iv} → з**.

Символ **S** переходит в пустую строку в конце слова:

Athais – Ате

то есть правило **S → ∅** будет заменено на **V_I s{>} → ∅**.

Случаи, когда *S* переходит в «с», требуют более подробного рассмотрения. Коротко говоря, это все случаи, кроме упомянутых выше, то есть *S* в начале слова:

Silvie – Сильви

Segalene – Сегален

S между гласной и согласной:

Elesbed – Элесбе

Fausta – Фоста

S между согласной и гласной:

Darsie – Дарси

Ponset – Понсе

Таким образом, согласно этим примерам, правило $S \rightarrow c$ должно быть заменено тремя правилами: $\{<\} S V_I \rightarrow c$, $V_{IV} S V_{IC} \rightarrow c$, $V_{IC} S V_{IV} \rightarrow c$. Однако этого не происходит. Все случаи, когда *S* переходит в «с», считаются примерами употребления одного и того же правила, и их контексты объединяются, образуя одно правило $V_I \cup \{<\} S V_I \rightarrow c$. В контексты этого правила включаются контексты правила $V_{IV} S V_{IV} \rightarrow z$, что, во-первых, создает неоднозначность в системе правил, а во-вторых, и вовсе неверно, так как *S* между двумя гласными буквами должна переходить только в «з».

Алгоритм добавления контекстов, описанный в данной главе, уже является упрощением корректного алгоритма. Наиболее правильным способом добавления контекстов было бы создание нового правила для каждого имени из обучающей выборки, таким образом, после анализа пар «*Elesbed* – Элесбе» и «*Fausta* – Фоста» из примера 10 вместо правила « $s \rightarrow c$ » в системе появилось бы два правила: « $\{e\}s\{b\} \rightarrow c$ » и « $\{u\}s\{t\} \rightarrow c$ ». Тогда при достаточно большом размере обучающей выборки были бы найдены все существующие в языке сочетания правого и левого контекстов, при которых справедливо правило « $s \rightarrow c$ ». Однако неизвестно, какого объема должна быть выборка, чтобы содержать все возможные случаи (тем более, невозможно определить, полна выборка или нет), к тому же, одним из требований к созданному методу была возможность обучения на небольших множествах имен. По этой причине было введено правило объединения правил транскрипции с совпадающими контекстами.

Правило объединения

Пусть существует два правила r_1 и r_2 таких, что $r_1(\alpha) = r_2(\alpha)$, $r_1(\beta) = r_2(\beta)$ и $r_1(p_l) \cap r_2(p_l) \neq \emptyset$ или $r_1(p_r) \cap r_2(p_r) \neq \emptyset$, то есть, правые и левые части правил совпадают, и хотя бы один из контекстов содержит общие символы. Тогда правила r_1 и r_2 объединяются в правило r_3 такое, что $r_3(\alpha) = r_1(\alpha) = r_2(\alpha)$, $r_3(\beta) = r_1(\beta) = r_2(\beta)$, $r_3(p_l) = r_1(p_l) \cup r_2(p_l)$, $r_3(p_r) = r_1(p_r) \cup r_2(p_r)$, то есть правила r_1 и r_2 считаются частными случаями одного и того же правила и объединяются в одно. Это правило может вызвать появление некорректных контекстов, но такие случаи редки.

Это правило, хоть и повышает универсальность системы правил транскрипции, на практике оказывается недостаточным – оно не охватывает многие допустимые контексты. Поэтому было принято следующее упрощение – добавлять все контексты в одно правило. Такие контексты охватывают большую часть всех возможных в языке контекстов, но могут породить и некорректные сочетания.

Из вышесказанного ясно, что проблема контекстов правил может быть полностью устранена только обучением на больших (свыше 10 тысяч) множествах имен. Однако этот метод имеет выгодное отличие от прочих существующих, поскольку полученная модель транскрипции может быть отредактирована вручную. В частности, могут быть удалены или отредактированы незначащие контексты и добавлены недостающие.

2.4 Выводы к главе 2

Метод порождения правил транскрипции, представленный в главе, позволяет автоматически генерировать правила межъязыковой транскрипции на основе множества прецедентов (имен на языке оригинала и их транскрипций на целевой язык). Метод не зависит от языка, не требует дополнительной лингвистической информации. Для корректной работы метода необходимы только списки всех символов языков и списки букв, обозначающих в языках гласные звуки.

Правила порождаются в два этапа. На первом этапе – порождении первичных правил – происходит выравнивание имен и их переводов и порождение правил на основе этого выравнивания. Выравнивание производится на основе информации о типе букв (гласные или согласные), поэтому вероятность порождения некорректных соответствий невелика. На втором этапе – порождении сложных правил – происходит разбор слогов с помощью существующих правил. При этом

существующие правила дополняются, а также создаются новые правила. На втором этапе основной единицей анализа является слог, а не слово, потому что в слогe легче найти не удовлетворяющую существующим правилам подстроку.

Кроме того, был добавлен экспериментальный третий этап обучения – добавление контекстов длины 2. Экспериментальным этот этап назван потому, что его вклад в систему правил транскрипции спорен: с одной стороны, он помогает снимать неоднозначности, с другой стороны, создает опасность появления новых, к тому же, на маленьком обучающем множестве многие возможные в языке контексты могут быть не представлены, что сделает правила слишком строгими и неприменимыми при транскрипции.

Все межъязыковые соответствия подстрок могут быть порождены даже на совсем небольшом обучающем множестве (150-200 имен). Порождение контекстов на таком количестве обучающих данных составляет проблему – как уже было сказано, при маленьком количестве примеров могут быть не охвачены многие из возможных в языке контекстов. С другой стороны, контексты могут быть добавлены или отредактированы экспертом вручную при обработке системы правил.

Глава 3. Метод транскрипции с помощью конечного автомата

Одной из задач данной работы является создание метода применения системы правил транскрипции, преобразующего строки за линейное относительно длины строки время. Как было показано в главе 1, все существующие решения обладают полиномиальной сложностью, некоторые из них зависят от объема системы правил, который может колебаться от нескольких десятков до нескольких тысяч правил, в зависимости от степени фонетичности орфографии языка.

Конечный автомат – один из немногих методов, позволяющих обработать строку за время, линейное относительно длины этой строки. Он используется для решения различных задач автоматической обработки текста, таких, как морфологический анализ (в частности, лемматизация, определение части речи и снятие частеречной омонимии) [124]. Применяются они и для решения задачи машинной транскрипции [28]. Существуют приложения для построения конечных автоматов на основе заданной пользователем регулярной грамматики, такие, как *lexc* [125] и *xfst* [126], разработанные финскими исследователями компании XEROX, или уже упоминавшийся инструмент CARMEL [33], позволяющий реализовать цепочку стохастических конечных автоматов для осуществления преобразований текста.

Конечный автомат был выбран по причине обеспечиваемой им линейной скорости обработки строк. Поскольку конечный автомат уже неоднократно использовался ранее в различных приложениях, созданы инструменты, позволяющие пользователю самостоятельно задать грамматику для конечного автомата (*lexc*, *xfst*) или параметры для обучения статистического автомата (CARMEL). Однако все упомянутые средства не подходят для разбора строк с помощью предложенной в главе 2 системы правил транскрипции. *Lexc* может работать только с классической регулярной грамматикой, под определение которой не попадает система правил с контекстами. CARMEL же – система, которая строит статистический конечный автомат, совмещающий, к тому же, этапы обучения и транскрипции – то есть строит автомат не по регулярной грамматике, а по корпусу текстов, заданному пользователем. Поэтому в данной работе строится новый

конечный автомат и предлагается новый алгоритм его построения, принимающий на вход КЗ-грамматику, состоящую из детерминированных правил.

В данной главе описана предложенная автором в работе [127] структура расширенного конечного автомата, позволяющего, в отличие от классического конечного автомата, применить к строке правила преобразования с контекстами. Приведена процедура построения автомата на основе системы правил транскрипции, приведения этого автомата к детерминированному виду, а также некоторые преобразования системы правил (унификация), делающие это приведение возможным. Далее показано, что из предварительно унифицированной системы правил может быть сразу построен детерминированный автомат, без промежуточного недетерминированного. Доказана эквивалентность этого ДКА автомату, построенному из НКА. Кроме того, доказана линейная скорость обработки строк с помощью расширенного КА. В начале главы приведена процедура преобразования строки с помощью системы правил, без использования конечного автомата. Эта процедура в данной работе принята в качестве эталона, то есть, множество строк, получаемых в результате применения этой процедуры к некоторой строке, является единственно верным результатом применения системы правил к этой строке. Доказательство корректности работы конечного автомата производится путем сравнения результатов работы автомата (то есть, множества выдаваемых им строк) с результатами работы процедуры преобразования строк с помощью системы правил.

3.1 Преобразование строки с использованием правил

В данной работе в качестве основного метода транскрипции используется конечный автомат. Однако необходимо понимать, что конечный автомат – только метод применения правил, соответственно, к нему предъявляется требование корректной и непротиворечивой передачи строк с учетом всех предоставленных ему правил. Для проверки корректности преобразования строки с помощью конечного автомата требуется некоторый эталонный метод применения правил, с результатом работы которого можно сравнить строку, выдаваемую автоматом. В качестве такого эталона был принят алгоритм преобразования строки с помощью системы правил (алгоритм 2), являющийся обобщением алгоритмов для существующих систем,

основанных на правилах. В разделе 3.2 представлен алгоритм преобразования строки с помощью конечного автомата (алгоритм 3), который учитывает все особенности применения системы правил к строке. Таким образом, доказываемое ниже в данной главе утверждение об эквивалентности системы правил расширенному конечному автомату эквивалентно утверждению об эквивалентности результатов преобразования любой строки над алфавитом V_I с помощью алгоритма 2 и алгоритма 3.

**Алгоритм 2. Преобразование строки
с помощью системы правил**

Предварительное замечание: как уже было определено в главе 2, правило r *применено* к строке с текущей позиции, если на текущей позиции находится подстрока $r(\alpha)$, перед ней полностью представлена одна из подстрок из $r(p_i)$, после нее – подстрока из $r(p_r)$.

Преобразование строки с помощью системы правил R производится следующим образом:

1. Текущей позицией входной строки назначается первый символ.
2. Формируется пустая выходная строка.
3. В системе правил R ищутся все правила, применимые к строке с данной позиции.
4. Из найденных правил выбирается правило с самой длинной левой частью. Если таких правил более одного, формируется более одной выходной строки (по количеству найденных правил). Если не найдено ни одного правила, текущий символ без изменения в обрамлении специальных символов, обозначающих отсутствие перевода (например, символов подчеркивания «_») приписывается ко всем выходным строкам, текущая позиция сдвигается на 1.
5. Ко всем выходным строкам присоединяется подстрока β найденного правила. Если существует более одной выходной строки и при этом было найдено более одного правила, множеством выходных строк после данного шага будет множество конкатенаций всех пар из декартова произведения существующих выходных строк и строк β всех найденных правил (см. пример 11).
6. Текущая позиция сдвигается на $|\alpha|$ символов вправо.
7. Пункты 3-6 повторяются, пока не достигнут конец слова.

Пример 11. Поиск множества выходных строк в случае неопределенности.

Пусть существует множество уже порожденных выходных строк $\{\langle a \rangle, \langle o \rangle\}$ и множество правых частей применяемых правил $\{\langle b \rangle, \langle v \rangle\}$. К каждой строке из первого множества должна быть добавлена каждая строка из второго множества. Таким образом, результирующее множество выходных строк будет равно $\{\langle ab \rangle, \langle av \rangle, \langle ob \rangle, \langle ov \rangle\}$.

3.2 Структура конечного автомата

В практических приложениях используются конечные автоматы двух типов: распознающие и преобразующие. Распознающие конечные автоматы по данной строке определяют, принадлежит ли она заданному языку. На вход подается строка, разбор начинается с первого символа строки из начального состояния автомата. По каждому следующему символу осуществляется переход в очередное состояние конечного автомата. Если из данного состояния не существует перехода по текущему символу, или достигнут конец строки, когда автомат находится не в конечном состоянии, строка не принадлежит языку, который распознает данный автомат. Если же по достижении конца строки текущим состоянием автомата является конечное состояние, строка принадлежит языку.

Преобразующие конечные автоматы помимо продвижения вперед по строке при переходе в новое состояние осуществляют формирование выходной строки или преобразование текущей. Таким образом, они возвращают не только ответ на вопрос, является ли данная строка словом некоторого языка, но и новую строку [128, 129].

Предлагаемый в работе конечный автомат является расширенным. Он представляет собой преобразователь, то есть, получая на вход строку, преобразует ее в соответствии с правилами, на основе которых построен, и возвращает в преобразованном виде. Расширенным же он является потому, что, в отличие от классических конечных автоматов, может осуществлять движение по строке не на один символ, а на несколько, причем в любом направлении. Подобный конечный автомат можно представить как множество $g = \langle V_I, V_O, Q, q_0, F, \theta \rangle$, где:

V_I – входной алфавит (алфавит языка оригинала);

V_O – выходной алфавит (алфавит языка перевода);

Q – множество состояний автомата;

$q_0 \in Q$ – начальное состояние автомата;

$F \subset Q$ – множество конечных состояний;

θ – функция переходов $Q \times V_I \rightarrow \langle q, n, A \rangle$.

Здесь $q \in Q$ – это состояние, в которое должен быть произведен переход, $n \in \mathbb{Z}$ – количество символов, на которое должен быть произведен сдвиг по разбираемой строке при переходе, $A \subset \{V_O^*\}$ – множество выходных строк, приписанных переходу. .

Разбор строки с помощью описанного КА ведется по алгоритму 3.

Алгоритм 3. Преобразование строки
с помощью конечного автомата

1. Текущим символом l назначается первый символ входной строки, текущим состоянием q назначается начальное состояние q_0 .
2. Вычисляется значение функции переходов $\theta(q, l) \rightarrow \langle q', n, A \rangle$. Из текущего состояния производится переход в состояние q' . При переходе производится сдвиг по разбираемой строке на n символов.
3. Если $q' \in F$, то есть, достигнуто конечное состояние, происходит переход по пустому символу в начальное состояние. То есть $\forall f \in F, \theta(f, \varepsilon) = \langle q_0, 0, \emptyset \rangle$.
4. Если $A \neq \emptyset$ (это возможно только если $q' \in F$), следует обновить результат транскрипции: он составит множество конкатенаций всех пар из декартова произведения существующих выходных строк и выходных строк, приписанных к данному переходу (аналогично с декартовым произведением из описания функции TrR).
5. Если для пары $\langle q, l \rangle$ функция θ не определена, символ l присоединяется к выходной строке (или строкам) без изменения в обрамлении специальных символов, обозначающих отсутствие перевода (аналогично определению функции TrR), и осуществляется переход в q_0 .
6. Пункт 2 повторяется, пока не достигнут конец слова.

В принципе, при разборе не важно, оказалось ли текущим состоянием конечное в тот момент, когда был достигнут последний символ строки. Но если

этого не произошло, то, возможно, для последнего символа не была возвращена выходная подстрока, а значит, транскрипция была проведена не вполне корректно.

3.3 Построение конечного автомата

Теоретически, конечный автомат описанного выше формата может быть построен вручную. Но на практике это является затруднительным, а главное, в этом нет необходимости, так как можно использовать правила, полученные по описанному выше методу. Далее будем считать, что конечный автомат строится на основе системы правил. В этом случае каждое правило преобразуется в цепочку переходов конечного автомата. Первым состоянием каждой такой цепочки является единственное начальное состояние автомата, последнее же состояние цепочки – конечное. Из каждого конечного состояния проводится переход по пустому символу в начальное состояние. Таким образом, переход по цепочке от начального состояния к конечному при разборе слова равносителен применению правила транскрипции. После применения правила автомат возвращается в начальное состояние – то есть, готов применить следующее правило.

Напомним, что правило записывается как кортеж $\langle \mathbf{p_l}, \alpha, \mathbf{p_r} \rangle \rightarrow \beta$, где α – входная строка, β – выходная строка, замещающая строку α при переводе, $\mathbf{p_l} = \{\gamma\}$ и $\mathbf{p_r} = \{\delta\}$ – контексты.

Преобразование правила к конечному автомату производится следующим образом:

- Преобразование левого контекста правила:
 - Определяется значение функции $\theta(q_0, l_0) = \langle q_1, -|\gamma|, \emptyset \rangle$, где l_0 – первый символ α , $\gamma \in \mathbf{p_l}$, q_1 – новое состояние.
 - Для каждой строки $\gamma = \langle t_1 \dots t_k \rangle$ левого контекста создаются пути такие, что $\forall \gamma \in \mathbf{p_l} \rightarrow \theta(q_i, t_i) = \langle q_{i+1}, 1, \emptyset \rangle$, $1 \leq i \leq k$, причем все пути должны начинаться в состоянии q_1 и заканчиваться в состоянии q_k , но их промежуточные состояния q_j , $1 < j < k$, должны быть различными.
 - При отсутствии левого контекста $q_k = q_0$.
- Преобразование строки правила:
 - $\forall l_i \in \alpha \rightarrow \theta(q_{k+i}, l_i) = \langle q_{k+i+1}, 1, \emptyset \rangle$, $0 \leq i < n$,

- При наличии правого контекста: $\theta(q_{k+n}, l_n) = \langle q_{k+n+1}, 1, \emptyset \rangle$,
- При отсутствии правого контекста: $\theta(q_{k+n}, l_n) = \langle q_f, 1, \beta \rangle$, $q_f \in F$.
- Преобразование правого контекста правила:
 - Создаются пути для каждого правого контекста: $\forall \delta \in \mathbf{p_r}$,
 $\delta = \langle s_1 \dots s_m \rangle \rightarrow \theta(q_{k+n+i+1}, s_i) = \langle q_{k+n+i+2}, 1, \emptyset \rangle$, $1 \leq i < m$, $\theta(q_{k+n+m}, s_m) = \langle q_f, -|\delta|+1, \beta \rangle$, $q_f \in F$. Аналогично с процедурой создания путей для левых контекстов промежуточные состояния путей для разных δ должны быть различны.
 - $\forall f \in F, \theta(f, \varepsilon) = \langle q_0, 0, \emptyset \rangle$.

eau $\rightarrow$ o

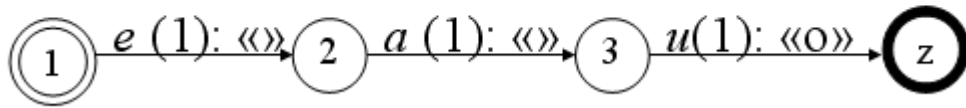


Рис. 7. Преобразование правила без контекста «eau→o» к конечному автомату

{ai, ei}\|{e} $\rightarrow$ й

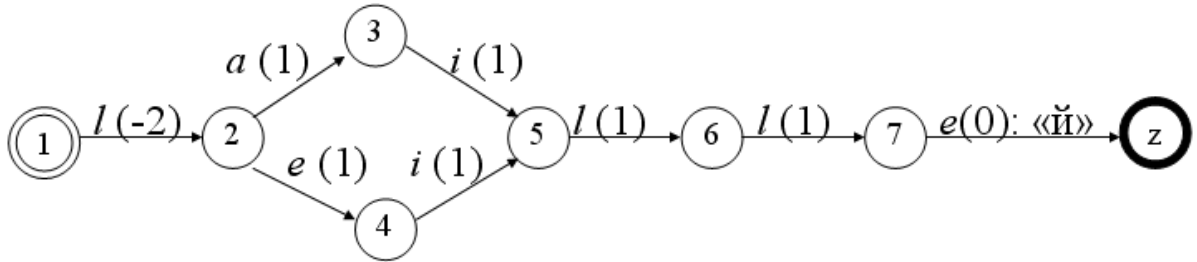


Рис 8. Преобразование правила с контекстом {ai, ei}\|{e} → й к конечному автомату

Таким образом, происходит проверка первого символа из α , возврат назад по цепочке на длину левого контекста, проверка левого контекста, проверка строки α , проверка правого контекста и возврат назад к символу, следующему за последним символом α . В соответствии с этим можно утверждать, что свойства процедуры преобразования строки с помощью системы правил, необходимые для корректной передачи строки, сохраняются и в конечном автомате. Ни один символ из входной строки не пропущен, так как не происходит сдвига вправо по входной строке больше

чем на один символ. Ни один символ из входной строки не рассматривается дважды. Некоторые символы входной строки могут быть просмотрены более одного раза, если они являются контекстами других символов, но при преобразовании данного символа его контекст не преобразовывается, после проверки правого контекста происходит возврат к его началу, чтобы он мог быть рассмотрен по своим правилам. Эквивалентность процедуры преобразования строки с помощью построенного конечного автомата процедуре преобразования строки с помощью системы правил формально доказана в пункте 3.4.

Заметим, что для корректного преобразования правила с контекстом к конечному автомату все цепочки в каждом из контекстов должны иметь одинаковую длину. При этом цепочки из левого контекста не обязаны быть такой же длины, как цепочки правого контекста. Кроме того, правила должны быть проверены на предмет непротиворечивости, то есть не должно существовать правил r_1 и r_2 , таких, что $r_1(\alpha)$ является префиксом $r_2(\alpha) + r_2(p_r)$.

Заметим также, что контексты являются полностью независимыми друг от друга: не предъявляется требования к существованию обоих контекстов. В системе правил вполне допустимы правила вида $p_l \alpha \rightarrow \beta$, $\alpha p_r \rightarrow \beta$ или $\alpha \rightarrow \beta$, то есть без правого, без левого контекста или без обоих контекстов.

3.4 Детерминированный конечный автомат

Конечный автомат является детерминированным (ДКА), когда для каждой входной цепочки символов существует только одна последовательность переходов от начального состояния до конечного. Это значит, что для каждого состояния КА должно существовать не более одного перехода по каждому из символов входного алфавита. При построении конечного автомата согласно процедуре, описанной в предыдущем разделе, это правило может быть нарушено. Из состояния q_0 существуют переходы по первым символам строки α каждого из правил. Если в системе правил содержится хотя бы одна пара правил, у которых совпадают первые символы строк α , то автомат, построенный на основе этой системы, будет недетерминированным (НКА) (см. рис. 9). Разбор строк может быть проведен и с помощью недетерминированного конечного автомата, но будет потеряно главное преимущество КА – линейная скорость обработки строк.

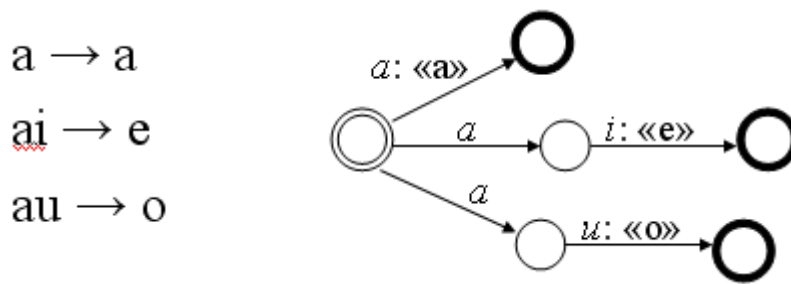


Рис 9. Недетерминированный автомат, построенный на основе системы правил $\{a \rightarrow a, ai \rightarrow e, au \rightarrow o\}$

Недетерминированность конечного автомата следует отличать от неоднозначности самой системы правил, на основе которой он строится. Под неоднозначностью понимается существование для строки φ на языке оригинала более одной строки ψ на языке перевода, которая может быть получена применением к φ правил из **R** (см. пример 4).

Неоднозначность системы правил может объясняться несколькими причинами. Во-первых, правила транскрипции часто неоднозначны по своей природе. Буква или сочетание букв может читаться двояко в силу традиции. Например, в норвежском языке буква “о” может обозначать звуки [o] или [u], и единственный способ получить однозначную транскрипцию имени, содержащего букву “о” – включить в систему **R** правило, левой строкой которого будет это имя, а правой – его перевод. Но такая система будет отражать не правила чтения языка, а лишь особенности обучающей выборки.

Другая причина неоднозначности системы правил состоит в том, что правила в используемой нами форме не учитывают морфологию. Оказавшийся на стыке двух морфем диграф, как правило, теряет устойчивое произношение, составляющие его буквы должны передаваться каждая по своему правилу (см. пример 12). При автоматическом порождении правил транскрипции, если в обучающей выборке для какого-либо диграфа нашлось достаточно случаев его появления на стыке морфем, для этого диграфа может быть порождено два правила.

Пример 12. Влияние морфологии на правила чтения.

Почти во всех языках существуют устойчивые сочетания букв – диграфы (сочетания из двух букв) или триграфы (сочетания из трех букв), которые читаются особым образом – звуки, которые они обозначают, не являются последовательностью всех звуков, которые обозначают все составляющие ди- или триграф буквы в нейтральной позиции. Яркими примерами являются

английский диграф **sh**, обозначающий звук [ш], а также французский триграф **eau**, который читается как [о:]. Однако если такой ди- или триграф находится на стыке морфем, он теряет свои особые свойства, так как разные его части тяготеют к разным морфемам: приставке и корню, корню и суффиксу и пр.

Рассмотрим следующий пример: имена английского языка **Sherlock** и **Bellshouse**. В первом диграф **sh** находится в корне, в другом – на стыке двух корней. Поэтому в первом случае корректной транслитерацией будет «ш», а во втором – каждая буква должна быть переведена по своему правилу – «зх»:

Sherlock – Шерлок

Bellshouse – Беллзхаус

Наконец, неоднозначность можно объяснить человеческим фактором, если правила были составлены экспертом, или, если правила были порождены автоматически, недостатками метода генерации правил.

Если система правил неоднозначна, в результате преобразования некоторых строк с ее помощью может получиться более одной выходной строки. Однако недетерминированность результата транскрипции вовсе не означает недетерминированности конечного автомата. Верно и обратное: из однозначной системы правил может быть получен НКА. Ниже будет показано, что результатом работы используемого нами КА может быть множество строк, причем автомат будет детерминированным. Также будет приведена процедура преобразования расширенного КА к детерминированному виду.

3.4.1 Процедура преобразования НКА в ДКА

В целом процедура преобразования НКА в ДКА, используемая в данной системе, стандартна и описана в работах [128, 129], но из-за специфических особенностей расширенного конечного автомата она была немного изменена (см. рис. 10).

Для каждой вершины q_1 , из которой исходит $n-1$ дуг в вершины $q_2, \dots, q_n$, помеченных одним символом:

- Создается новая вершина q' , в которую входит дуга из вершины q_1 , помеченная тем же символом;
- Из вершины q' проводятся все дуги, которые выходили из вершин $q_2, \dots, q_n$;

- Вершины $q_2, \dots, q_n$ удаляются. В классической процедуре преобразования НКА в ДКА вершины не удаляются, так как в них могут вести и другие переходы. Но процедура построения расширенного КА предполагает, что в каждую вершину ведет не более одного перехода, поэтому удаление будет корректным.



Рис 10. Недетерминированный конечный автомат (слева) и результат его приведения к детерминированному виду

Отметим, что в приведенной процедуре, в отличие от стандартной, не происходит удаления е-дуг (переходов по пустому символу), так как они являются аналогом перехода от одного правила к другому и необходимы для возвращения к начальному состоянию, с которого начинаются все правила.

Следует также отметить, что для конечного автомата, построенного на основе описанной выше системы правил, такая процедура преобразования к ДКА не вполне подходит. Если требуется объединить два перехода, одному из которых приписан кортеж $\langle q, 1, A_1 \rangle$, предполагающий переход по разбираемой строке на 1 символ вправо, а другому – $\langle q, -1, A_2 \rangle$ – переход на один символ влево, не ясно, какое значение n должно быть приписано новому объединенному переходу (см. пример 13).

Пример 13.

На рисунке 11 показана ситуация, когда из одного состояния (начальное, на рисунке помечено двойной незакрашенной рамкой) существует несколько переходов в другие состояния по одному и тому же символу (l), причем эти переходы нельзя соединить в один, так как им присвоены разные значения n , то есть, при разборе нужно будет сдвинуться одновременно на один символ вправо и влево по разбираемой строке, что создает недетерминированность.

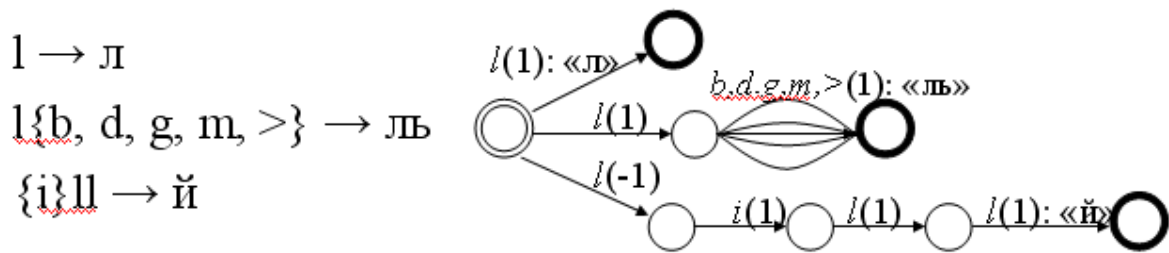


Рис 11. Недетерминированный автомат, построенный на основе системы правил $\{l \rightarrow л, l\{b, d, g, m, >\} \rightarrow ль, \{i\}ll \rightarrow й\}$

Для преодоления этой проблемы требуется провести некоторые преобразования самой системы правил еще до построения конечного автомата. Этот комплекс преобразований представляет собой приведение правил к единообразному виду и потому назван *унификацией* системы правил.

3.4.2 Унификация системы правил

Унификация системы правил включает в себя четыре процедуры:

- добавление контекстов по умолчанию,
- выравнивание контекстов,
- добавление правил по умолчанию,
- удаление неоднозначных контекстов.

Добавление контекстов по умолчанию

Эта процедура служит для предотвращения ситуации, описанной в примере 13.

Если для одной и той же буквы есть правило с контекстом и правило без контекста, правилу без контекста добавляется контекст по умолчанию (пример 14).

Пример 14.

В результате процедуры добавления контекстов по умолчанию правила системы $R = \{l \rightarrow л, \{i\}ll \rightarrow й, l\{b, d, g, m, >\} \rightarrow ль\}$ преобразуются следующим образом:

$l \rightarrow л$	»	$\{*\}l\{*\} \rightarrow л$
$\{i\}ll \rightarrow й$	»	$\{i\}ll\{*\} \rightarrow й$
$l\{b, d, g, m, >\} \rightarrow ль$	»	$\{*\}l\{b, d, g, m, >\} \rightarrow ль$

где * - контекст по умолчанию, то есть, контекст, состоящий из всех символов входного алфавита. Для краткости записывается символом «*».

Что касается символа $*$, необходимы некоторые разъяснения его функции. Выше было сказано, что он обозначает любой символ входного алфавита. Это действительно так, но с некоторыми оговорками. В вышеприведенном примере 14 правила $\langle l \rightarrow л \rangle$ и $\langle l\{b, d, g, m, >\} \rightarrow ль \rangle$ формально вступали в конфликт, так как второе предписывает перевести символ $\langle l \rangle$ как $\langle ль \rangle$ в случае выполнения контекста, тогда как первое предписывает перевести тот же символ как $\langle л \rangle$ независимо от его контекста. Однако неявно правила с контекстом имеют приоритет перед правилами без контекста, так как вторые описывают общий случай, а первые – особые условия. После добавления контекста ко всем правилам, они все становятся равны, и теперь уже налицо явный конфликт не только между первым и третьим, но и между первым и вторым правилами системы. Поэтому символ « $*$ » следует понимать не как все символы алфавита, а все символы алфавита, кроме уже перечисленных. То есть, символ « $*$ » в правом контексте первого правила будет значить все символы, кроме символов $\langle b \rangle$, $\langle d \rangle$, $\langle g \rangle$, $\langle m \rangle$, $\langle > \rangle$ (из третьего правила) и $\langle l \rangle$ (из второго правила), левые контексты первого и третьего правил обозначают все символы кроме $\langle i \rangle$. Такое определение не очень удобно при работе с правилами, но становится необходимым после построения конечного автомата на их основе. Если из некоторого состояния существуют переходы по символам u , v , и $*$, то по любому поступившему на вход символу l следует перейти по переходу, помеченному символом $*$, если только символ l не равен символу u или v .

Выравнивание контекстов

Эту процедуру можно назвать расширением процедуры добавления контекстов по умолчанию. Как уже было отмечено, к контекстам предъявляется требование одинаковой длины всех входящих в них цепочек. Если в контексте правила есть цепочки разной длины, выбирается цепочка наибольшей длины, остальные цепочки дополняются до нужной длины символом по умолчанию (« $*$ »). Причем цепочки левых контекстов дополняются слева, а цепочки правых контекстов – справа, так как значащая часть контекста должна непосредственно соседствовать с цепочкой α правила. То есть, контекст $\{e, ai, o\}$, если это левый контекст, будет преобразован в контекст $\{*e, ai, *o\}$, или в контекст $\{e*, ai, o*\}$, если это правый

контекст. Напомним, что * - символ, заменяющий все символы алфавита, то есть, на самом деле запись цепочки «*е» обозначает множество цепочек {ae, be, ce, ..., ze}.

Добавление правил по умолчанию

Данный этап унификации правил направлен на то, чтобы сделать систему правил универсальной: то есть, способной преобразовать любую цепочку символов входного алфавита. Поскольку система правил служит не для распознавания корректных строк данного языка, а для их преобразования, нет необходимости в строгом ограничении количества и качества правил.

Суть преобразования состоит в том, чтобы преобразовать любой символ входного алфавита независимо от того, в каком он находится контексте. Если для символа $l \in V_I$, не существует правила вида $\alpha \rightarrow \beta$ (без контекста), где $\alpha = l$, составляется правило $\alpha \rightarrow \beta$, где $\alpha = l$, а β – правая часть самого распространенного (имеющего больше всего употреблений в данной обучающей выборке) правила с контекстом для α (см. пример 15).

Если же для символа l в системе правил не существует ни одного правила $p_l \alpha p_r \rightarrow \beta$ (p_l и p_r могут быть пустыми), такого, что $\alpha = l$, никаких усовершенствований системы на этапе унификации не может быть произведено. В этом случае принято считать, что символ l в данном языке может употребляться лишь в строго определенных контекстах (например, символ q во французском языке употребляется только в подстроке qu). Но поскольку такой строгий контекст – довольно редкий случай, он не может сильно ухудшить качество транскрипции. Если же в слове все-таки встретится символ в не предусмотренном для него контексте, он будет передан в результат без перевода, так как появление такого символа, скорее всего, обозначает ошибку во входных данных (например, попытку перевести слово не с того языка).

Пример 15. Добавление правила по умолчанию

В системе правил **R** присутствуют следующие правила для символа «с»:

$c\{e, i\} \rightarrow c$ – встретилось 100 раз
 $c\{a, o, u\} \rightarrow k$ – встретилось 200 раз

На этапе добавления правил по умолчанию в систему **R** будет добавлено правило $c \rightarrow k$

Удаление неоднозначных контекстов

В принципе неоднозначность системы правил транскрипции – нормальное явление, вытекающее из неоднозначности графической системы языка. Но некоторые примеры неоднозначности делают невозможным построение конечного автомата, поэтому при унификации правил мы считаем нужным избавиться от них.

Пусть существуют правила $\mathbf{r}_1$ и $\mathbf{r}_2$, такие что $|\mathbf{r}_1(\alpha)| < |\mathbf{r}_2(\alpha)|$ и $\mathbf{r}_1(\alpha) + \gamma = \mathbf{r}_2(\alpha)$ или $\mathbf{r}_1(\alpha) + \gamma = \mathbf{r}_2(\alpha) + \delta$, где $\gamma \in \mathbf{r}_1(\mathbf{p}_r)$, $\delta \in \mathbf{r}_2(\mathbf{p}_r)$. Иными словами, конкатенация левой часть какого-либо правила с его контекстом совпадает с левой частью другого правила (или с конкатенацией левой части другого правила с контекстом, при условии, что левые части правил не равны). В этом случае предпочтение отдается правилу с более длинной левой частью ($\mathbf{r}_2$), из правого контекста более короткого правила ($\mathbf{r}_1$) удаляется контекст, создающий неоднозначность (γ).

Покажем, что эта процедура не отражается на результате транскрипции. Согласно процедуре преобразования строк с помощью системы правил, из применимых правил выбирается правило с самой длинной левой частью α . Таким образом, если существуют правила $\mathbf{r}_1$ и $\mathbf{r}_2$, такие, что $\mathbf{r}_2(\alpha) = \mathbf{r}_1(\alpha) + \mathbf{r}_1(\gamma)$, $\gamma \in \mathbf{r}_1(\mathbf{p}_r)$, они оба будут применимы к строке, содержащей подстроку $\mathbf{r}_2(\alpha)$. Но в такой ситуации правило $\mathbf{r}_1$ никогда не будет применено. Следовательно, удаление неоднозначных правых контекстов $\mathbf{r}_1$ не изменяет результаты преобразования.

Здесь необходимо заметить, что в связи с выбором самой длинной цепочки сам автомат ограничивает свою выдачу. То есть при наличии правил $\mathbf{r}_1$ и $\mathbf{r}_2$, таких, что $\mathbf{r}_2(\alpha) = \mathbf{r}_1(\alpha) + \mathbf{r}_1(\gamma)$, $\gamma \in \mathbf{r}_1(\mathbf{p}_r)$, выдаваться будет одна цепочка, тогда как правильным было бы выдать обе. Но в большинстве исследованных нами случаев подобное поведение является правильным.

3.4.3 Эквивалентность НКА и ДКА

ДКА, полученный из НКА, должен быть эквивалентен ему, то есть, распознавать ту же грамматику. Однако в настоящей работе используется расширенный конечный автомат, позволяющий переходы по разбираемой строке как вправо, так и влево. Для доказательства утверждения, что преобразование НКА в ДКА сохраняет множество разбираемых цепочек, будет сформулировано

Утверждение 2 об эквивалентности расширенных НКА и ДКА.

Процедура приведения расширенного конечного автомата к детерминированному виду сохраняет множество разбираемых цепочек.

Доказательство.

Используемый в работе автомат отличается от классического возможностью перехода по текущей строке не только вправо, но и влево на произвольное число символов. Покажем, что для этого автомата с учетом накладываемых на правила грамматики ограничений в результате процедуры преобразования НКА в ДКА получится эквивалентный автомат.

По определению автомата переходы влево по строке возможны только перед началом проверки левого контекста правила или после проверки правого контекста.

Переход назад для проверки левого контекста производится после просмотра первого символа преобразуемой подстроки (строки α в правиле). Пусть l – первый символ строки α . Будет создано новое состояние q' , объединяющее все переходы из начального состояния по символу l . Пусть R_l – подмножество системы правил R , включающее все правила, левая часть которых начинается с символа l . Таким образом, состояние q' объединит цепочки всех правил из R_l . Если у правил из R_l контексты разной длины, при создании q' может возникнуть неопределенность, связанная с тем, что разным переходам из начального состояния по l приписаны разные значения функции θ . Напомним, что функция θ переходов КА по текущему состоянию и текущему входному символу возвращает кортеж $\langle q, n, A \rangle$, где q – новое состояние, n – сдвиг по входной строке, выраженный целым числом, A – множество выходных строк. Очевидно, что q для разных переходов по l будет различным. Множество A для перехода в q' будет составлять объединение всех A для исходных переходов. Проблему представляет n , так как детерминированный автомат должен однозначно определять процедуру разбора – то есть, при переходе не должно происходить сдвига сразу на 1 и на 2 символа влево.

Однако в соответствии с процедурой выравнивания контекстов строки, входящие в левый контекст каждого правила класса эквивалентности R_l должны быть одинаковой длины. Значит, при формировании перехода в q' его выходной кортеж $\langle q, n, A \rangle$ будет однозначно определен, так как во всех исходных кортежах, входящих в R_l , значения n совпадают.

Рассмотрим переход назад по строке после проверки правого контекста. Необходимость объединить несколько переходов с отрицательным значением n может возникнуть в нескольких случаях:

- Если существует два правила r_1 и r_2 таких, что $r_1(\alpha) = r_2(\alpha)$;
- Если существует два r_1 и r_2 , таких, что $r_2(\alpha) = r_1(\alpha) + r_1(\gamma)$, $\gamma \in r_1(p_r)$.

В первом случае, если у правил пересекающиеся правые контексты разной длины (например, у одного правила контекст e , у другого – e в конце слова, то есть $e>$), после проверки более короткого контекста происходит переход в конечное состояние со сдвигом влево, тогда как проверка более длинного контекста еще не закончена. Однако процедура выравнивания контекстов предусматривает и такие ситуации, поэтому у правил с совпадающей левой частью должна совпадать длина не только левых, но и правых контекстов. Таким образом, возврат назад будет осуществляться одновременно и на одинаковое расстояние.

Во втором случае мы считаем, что основная часть правила (строка α) имеет приоритет над контекстом, поэтому в процедуру унификации правил входит, помимо прочего, удаление контекстов, которые в объединении с левой строкой правила образуют строку, совпадающую со строкой другого правила. То есть, не возникает необходимости объединять переходы со значениями $\langle q, i, A \rangle$, $\langle q, -j, A \rangle$, где $i, j \in \mathbb{N}$.

Таким образом, при объединении переходов из одного состояния по одному и тому же символу число и направление шагов по разбираемой строке для этих переходов всегда будет совпадать. $\square$

3.4.4 Преобразование системы правил в ДКА

Возможен другой вариант решения проблемы недетерминированности конечного автомата. Система правил может преобразовываться напрямую в детерминированный конечный автомат, минуя стадию недетерминированного автомата.

Процедуры преобразования правил с контекстами и без них остаются прежними, с одним исключением. Перед тем, как создать новый переход по текущему символу из текущего состояния, нужно убедиться, что из этого состояния еще нет перехода по данному символу. Если это так, процедура не меняется. Если

же переход уже существует, новый переход не строится, просто происходит переход по текущему символу в нужное состояние. Если состояние, в которое совершен переход, является конечным, выходная строка добавляется к множеству выходных строк этого состояния. Если же при преобразовании очередного правила было достигнуто конечное состояние, но по процедуре построения данное состояние не должно быть конечным и не предусмотрено выходной строки, значит, система правил, на основе которой строится автомат, не была унифицирована должным образом.

По большому счету процедура построения ДКА из системы правил не очень отличается от процедуры построения НКА по той же системе правил, за исключением того, что для построения ДКА система правил должна быть заранее унифицирована, тогда как НКА можно построить на основе правил с контекстами различной длины. Но поскольку для преобразования НКА в ДКА система правил также должна быть унифицирована, возможность построения НКА по произвольной системе правил становится лишь теоретической, и не может рассматриваться как отличие между двумя способами построения автомата.

Введем обозначения для различных типов КА. ДКА, полученный путем преобразования НКА в ДКА, назовем КА I. ДКА, построенный напрямую из системы правил, назовем КА II. В таком случае может быть доказано

Утверждение 3 Об эквивалентности КА I и КА II.

КА I G_1 и КА II G_2 , полученные по одной и той же системе правил, эквивалентны в смысле $TrA(G_1, \gamma) = TrA(G_2, \gamma)$, где γ – произвольная строка.

Доказательство.

Правила каждого из классов R_l при преобразовании системы правил в конечный автомат образуют независимый подграф, причем это верно как для НКА, преобразованного в ДКА (КА I), так и для ДКА, построенный напрямую из системы правил (КА II). При построении НКА на основе правил из каждого правила создается цепочка. При преобразовании НКА в ДКА цепочки с одинаковыми префиксами объединяются. При построении КА II при преобразовании правила используется уже существующая цепочка для правила с таким же префиксом. Таким образом, цепочки, полученные из правил разных классов эквивалентности, не будут иметь общих состояний, кроме начального.

Покажем теперь, что подграф КА I, построенный на основе R_l эквивалентен подграфу КА II, построенному на основе того же класса. Поскольку процедура унификации правил является обязательной для корректного построения автоматов как первого, так и второго типов, будем считать, что система правил унифицирована и правила единообразны. Можно выделить четыре вида правил в классе:

1. правила без контекстов;
2. правила с контекстами;
3. правила с левым контекстом;
4. правила с правым контекстом.

Покажем, что подграфы будут эквивалентны во всех четырех случаях.

1. Правила без контекстов. Такой случай возможен, только если не существует такого правила, левая часть которого являлась бы подстрокой другого правила. То есть, все правила такой системы характеризуются общим префиксом и конечной подстрокой, уникальной для данного правила. При преобразовании НКА в ДКА будут объединены части цепочек, обрабатывающие эквивалентные подстроки. В состоянии, в которое осуществлен переход по последнему общему символу, цепочка будет расходиться на несколько ветвей (по количеству правил). При построении КА II эквивалентных подцепочек просто не будет создаваться. Цепочка, обрабатывающая общий префикс, будет создана один раз, при преобразовании следующих правил будет совершен проход по ней без создания состояний, новое состояние будет создано только при построении пути для конечной подстроки. Таким образом, подграфы КА I и КА II для независимых подмножеств правил без контекстов, будут эквивалентны.
2. Правила с контекстами. Рассмотрим отличия КА I от КА II, построенных на основе одного и того же подмножества правил с контекстами. При построении КА строится переход из начального состояния по первому символу левой части, а затем создаются цепочки переходов для каждой строки из левого контекста. Отличие КА I от КА II на данном этапе заключается в том, что при построении КА I все цепочки для левых контекстов заканчиваются в одном и том же состоянии, из которого затем выходит цепочка переходов для левой

части и правых контекстов. В КА II пути для разных контекстов не сходятся. Рассмотрим два случая.

- a. В правилах подмножества левые контексты не пересекаются. В этом подграф каждого правила конечных автоматов и I, и II типа для каждого правила независим от других правил того же множества. После проверки первого символа левой части подграф каждого правила разветвляется на несколько цепочек переходов, которые будут различаться только левыми контекстами (см. рис.12 а, б). Цепочки, полученные по левой части правила за исключением левого контекста, эквивалентны, а потому могут быть объединены в одну. В результате подграф КА II станет эквивалентен соответствующему подграфу КА I. В КА II пути для правых контекстов заканчиваются в различных конечных состояниях. Соответственно, конечные состояния, в которые ведут все пути, полученные по символам правого контекста, могут быть объединены в одно. В результате будет получен автомат, эквивалентный КА I (см. рис.13 а, б).
 - b. Существуют правила с пересекающимися левыми контекстами. При преобразовании НКА в ДКА для путей, полученных для левых контекстов, пересекающихся с левыми контекстами других правил, будет сгенерирован отдельный подграф. Они, как и в КА II, образуют отдельную ветвь, проход по которой будет совершен при применении всех правил с совпавшими контекстами. Пути контекстов, которые не пересекаются с контекстами других правил, по-прежнему будут сходиться в одном состоянии. Для приведения КА II к эквивалентному КА I виду так же, как в предыдущем пункте, может быть осуществлено слияние эквивалентных ветвей.
3. Правила с левым контекстом. Этот случай повторяет предыдущий с той лишь разницей, что отсутствует необходимость объединения состояний, в которые совершаются переходы по правым контекстам.
 4. Правила с правым контекстом. Случай повторяет случай 1 с аналогичным отличием: цепочки КА II типа не разветвляются по контекстам.□

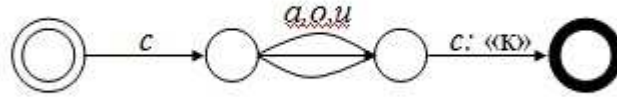


Рис.12а. Левый контекст правила $\{a,o,u\}c \rightarrow k$ для КА I.

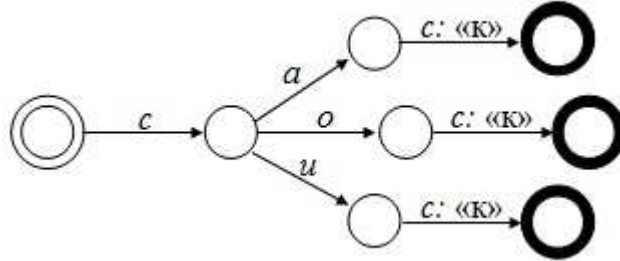


Рис.12б. Левый контекст правила $\{a,o,u\}c \rightarrow k$ для КА II.

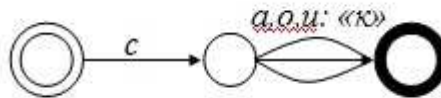


Рис.13а. Правый контекст правила $c\{a,o,u\} \rightarrow k$ для КА I.

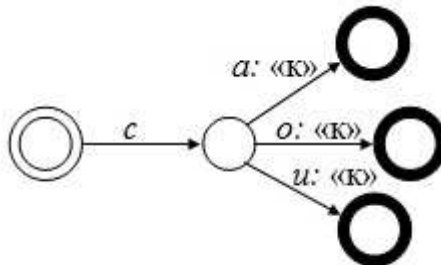


Рис.13б. Правый контекст правила $c\{a,o,u\} \rightarrow k$ для КА II.

Как видно из вышесказанного, процедуры преобразования строки с помощью автомата, о котором шла речь в предыдущих разделах, и автомата, описанного в данном разделе, эквивалентны, так как построены на основе одной и той же грамматики, хотя сами автоматы не эквивалентны. Выбор реализации зависит от имеющихся ресурсов: преобразование автомата из НКА в ДКА занимает дополнительное время, но при этом экономится место на размещении самого автомата. Детерминированный автомат, построенный по описанной процедуре,

занимает больше места, чем детерминированный автомат, преобразованный из недетерминированного.

3.5 Эквивалентность автомата системе правил

Конечный автомат $\mathbf{g}$ может быть назван эквивалентным системе правил $\mathbf{R}$, если для любой строки $\gamma \in \mathbf{V}_1^+$ выполняется равенство $\text{TrR}(\mathbf{R}, \gamma) = \text{TrA}(\mathbf{g}, \gamma)$.

Очевидно, что автомат, не эквивалентный системе правил, не следует использовать для решения задачи, так как требуется преобразовывать строки с помощью системы правил, а конечный автомат был выбран только из-за высокой скорости обработки строк. Поэтому к автомату предъявляется требование эквивалентности системе правил, для доказательства которого будет сформулировано

Утверждение 4 об эквивалентности системы правил конечному автомату.

Процедуры преобразования строки с помощью системы правил и с помощью конечного автомата, построенного на основе этой системы правил по описанному алгоритму, эквивалентны, то есть, для любой строки над алфавитом $\mathbf{V}_1$ результатом их работы будут одинаковые строки или множества строк над алфавитом $\mathbf{V}_0$.

Доказательство.

При преобразовании строки γ с помощью системы правил $\mathbf{R}$ строка делится на подстроки $\gamma = \langle \gamma_1, \gamma_2, \dots, \gamma_n \rangle$, где $\gamma_i = \mathbf{r}_i(\alpha)$. Заметим, что в КА по процедуре построения переход из начального состояния в конечное возможен только по цепочке, сгенерированной на основе какого-либо правила. Этот факт можно доказать следующим образом. При построении недетерминированного КА генерируются пути, соединяющие начальное состояние с конечным по цепочке, включающей в себя символы из $\mathbf{p}_l$, α , $\mathbf{p}_r$, то есть, прохождение по недетерминированному КА возможно только при соблюдении указанного условия.

Теперь покажем, что выбор передаваемой цепочки в КА осуществляется так же, как при разборе с использованием системы правил. Пусть существует два правила $\mathbf{r}_1$ и $\mathbf{r}_2$, таких, что $\mathbf{r}_1(\alpha) = l_0 \dots l_k$, $\mathbf{r}_2(\alpha) = l_0 \dots l_k l_n$, то есть, левая часть второго правила является левой частью первого правила с добавлением одного символа. Пути в графе конечного автомата, построенные для этих двух правил, будут

совпадать до последнего перехода. Пусть после просмотра подстроки $l_0...l_k$ текущим состоянием является состояние A_k . Из него по определению построения КА существует два перехода: переход по символу l_n , ведущий в конечное состояние, которому приписана выходная строка $r_2(\alpha)$, и переход по символу «*», обозначающему все символы кроме l_n . Таким образом, как и в случае разбора по правилам, к подстроке $l_0...l_k l_n$ будет применено r_2 , как обладающее самой длинной строкой α .

Таким образом, входная строка будет разбита на одни и те же подстроки как в случае применения правил, так и в случае КА. Подстроки будут передаваться одними и теми же правилами (в явном виде или преобразованными к КА). Следовательно, и выходные цепочки будут генерироваться одни и те же в обоих случаях, а их конкатенация будет давать один и тот же результат. То есть транскрипция с использованием системы правил и КА будут эквивалентны. $\square$

3.6 Усовершенствование конечного автомата

Несмотря на предварительную подготовку правил (унификацию) порожденный автомат с большой вероятностью не является универсальным – то есть, существуют строки входного языка, для которых не будет найдено не только корректного, но и хоть какого-нибудь перевода. Рассмотрим пример. На рис.14 изображен конечный автомат, полученный из системы правил $R = \{l \rightarrow л, l\{b, d, g, m, >\} \rightarrow ль, \{i\}l \rightarrow й\}$. На рис. 15 представлен тот же конечный автомат, полученный из системы R после ее унификации и приведенный к детерминированному виду.

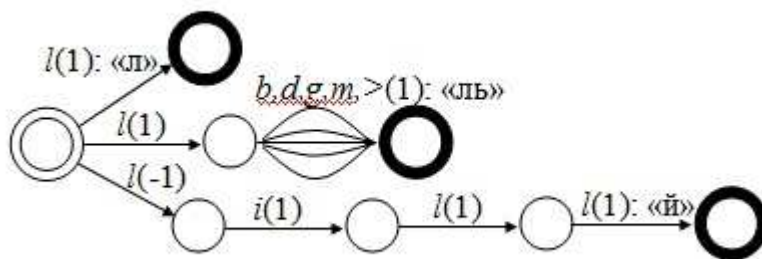


Рис.14. Недетерминированный конечный автомат

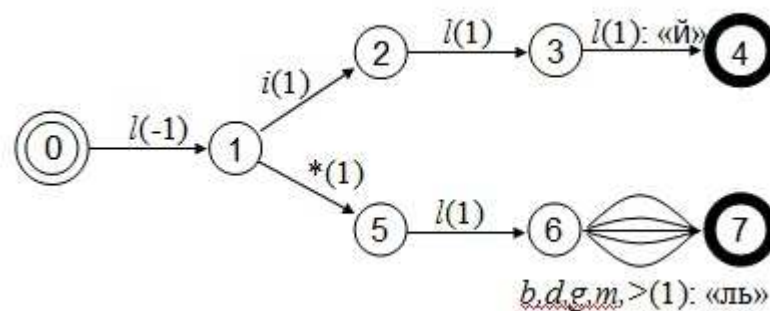


Рис. 15. Детерминированный конечный автомат

Рассмотрим пример. Требуется преобразовать имя TAMILA с помощью конечного автомата с рисунка 15. Пусть текущим символом является пятый символ – L (предыдущие были успешно переведены). В соответствии с процедурой преобразования строки с помощью конечного автомата происходит переход из состояния 0 в состояние 1 со сдвигом по строке на один символ влево, в результате чего текущим символом становится I. Из текущего состояния существует переход по символу *i* и по всем остальным символам («*»). Поскольку текущий символ – *i*, производится переход в состояние 2. Далее по символу *l* переходим в состояние 3. Но из состояния 3 существует только один переход – по символу *l*, и он в данном случае не подходит. В этой ситуации конечный автомат вернет в качестве транскрипции строку «Тами_l_a», то есть, символ *l* так и останется без перевода, хотя в системе, на основе которой был построен конечный автомат, есть правило для передачи символа *l* в таком случае: «*l* → л». Такая ситуация – обратная сторона детерминированности символа «*», описанная в разделе «Унификация правил». Если этот символ обозначает все символы, кроме отмеченных отдельно, процедура преобразования автомата к детерминированному виду становится проще, но такой автомат не является универсальным. Есть способ избавить автомат от этого недостатка уже после его построения. Из каждого неконечного состояния строятся переходы в новые конечные состояния. Каждому такому переходу приписывается выходная строка, которая формируется особым образом.

Для каждого символа входного алфавита определяется правило по умолчанию. Оно определяется так же, как правило по умолчанию в соответствующем этапе процесса унификации правил. Затем для каждого неконечного состояния определяется цепочка символов, по которой были

осуществлены переходы в это состояние из начального. Из этого состояния строится переход в новое конечное состояние по символу «*», переходу приписывается в качестве множества A конкатенация правых частей правил по умолчанию для всех символов цепочки.

Таким образом, из любого состояния существует переход по любому символу входного алфавита, то есть, любая последовательность символов может быть обработана конечным автоматом. Конечный автомат становится универсальным (см. рис.16).

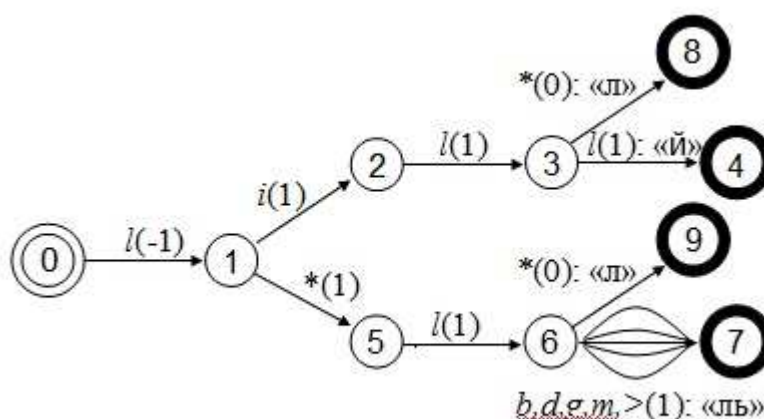


Рис.16. Универсальный конечный автомат

3.7 Скорость работы конечного автомата

Известно, что автомат обрабатывает строки с линейной скоростью [128]. Это верно для использующихся в задачах автоматической обработки текста классических конечных автоматов. Однако описанный выше автомат, использующийся в данной системе, не является классическим. Он был назван расширенным, так как имеет дополнительную функциональность: может сдвигаться по обрабатываемой строке на произвольное число символов, причем как вправо, так и влево. Эта особенность делает неочевидным свойство линейности работы автомата. Доказательству скорости разбора для расширенного автомата посвящено

Утверждение 5 о линейности конечного автомата.

Транскрипция строки длины n с помощью расширенного конечного автомата осуществляется за время, линейное относительно n .

Доказательство.

Известно, что разбор строки конечным автоматом имеет сложность $O(n)$, где n – длина строки. При разборе строки с помощью классического конечного автомата строка просматривается один раз и при каждом переходе происходит сдвиг на один символ, таким образом, разбор состоит из n переходов.

В расширенном конечном автомате, осуществляющем транскрипцию, возможен переход по строке на произвольное количество символов как вправо, так и влево.

Как это было отмечено выше, переход влево (то есть, назад) по строке возможен только при проверке контекстов. Перед проверкой левого контекста автомат сдвигается назад по строке на длину контекста, так как до этого текущим символом является первый символ применяемого правила. После проверки правого контекста автомат также сдвигается назад на длину контекста, так как контекст не входит в правило и после применения текущего правила должен быть разобран отдельно. Таким образом, величина сдвига влево определяется длиной контекста. Если ограничений на длину контекста нет, то сложность разбора строки в худшем случае составит $O(n^2)$, то есть, при проверке каждого символа нужно будет проверить его контекст, включающий все остальные символы строки.

Но система правил, на основе которой строится конечный автомат, не содержит правил с контекстом длиннее двух символов, так как в методе порождения правил введено ограничение на длину контекста. Это значит, что при разборе строки в худшем случае придется применить правило с левым и правым контекстами длины 2 для каждого символа строки. То есть для преобразования каждого символа в худшем случае придется дважды проверить сам символ, два предшествующих и два следующих за ним. То есть сложность разбора составит $O(6n)$. Такая оценка сложности для строк длиннее 6 символов будет меньше квадратичной. Что же касается строк длины 6 символов и меньше, сложность их разбора все равно не достигнет $O(6n)$, так как для крайних символов строки нет возможности проверить контексты длины 2. Если же контексты длины 2 не будут использоваться, максимальная сложность составит $O(4n)$.□

3.8 Выводы к главе 3

В главе описан предлагаемый автором метод транскрипции с помощью конечного автомата. Конечный автомат строится на основе системы правил транскрипции и осуществляет преобразование строк с исходного языка на целевой по этим правилам. Конечный автомат является детерминированным, но может выдавать несколько вариантов транскрипции. Все варианты равнозначны, потому что переходам автомата, как и правилам преобразования, не приписаны вероятности.

В главе изложены алгоритмы построения автомата, преобразования его к детерминированному виду и преобразования (унификации) системы правил, позволяющие корректно построить детерминированный конечный автомат из недетерминированного и избежать неоднозначностей. Помимо этого, представлен алгоритм построения на основе предварительно подготовленной системы правил детерминированного конечного автомата, без промежуточного недетерминированного варианта. Доказана эквивалентность процедуры преобразования строк с помощью расширенного конечного автомата описанной во 2-й главе эталонной процедуре преобразования строк с помощью системы правил. Кроме того, доказана правомерность всех промежуточных процедур: эквивалентность унифицированной системы правил первоначальной системе, эквивалентность детерминированного расширенного КА недетерминированному.

Главная цель создания метода транскрипции – разработка линейной, не зависящей от количества правил процедуры преобразования строки – также достигнута. В главе доказано утверждение о том, что предложенный конечный автомат позволяет преобразовывать строки за линейное время.

Глава 4. Практическая реализация и оценка качества разработанных методов

В главе описана программная реализация изложенных в предыдущих главах методов. Приведено описание программной системы, осуществляющей генерацию правил транскрипции и машинную транскрипцию строк.

Основным содержанием главы является описание экспериментов, проведенных с целью проверки эффективности предложенной в предыдущих главах методики транскрипции. Вводятся некоторые новые меры оценки качества транскрипции, выбранные с учетом особенностей системы. Приведены оценка качества обучения – качества транскрипции на обучающем множестве данных – двух экспериментов с различными настройками, а также оценка качества транскрипции – то есть качества передачи тестового множества данных. Экспериментально доказано и значительное превосходство в скорости предложенного в главе 3 метода транскрипции над аналогичными методами.

Кроме того, оценено качество порождаемых системой правил и зависимость качества и количества правил от объема обучающей выборки.

4.1 Описание программной системы

Изложенные в главах 2 и 3 методы и алгоритмы были реализованы в виде программной системы. Программа была написана на языке C++ в среде Borland C++ [130]. Объем программы составляет около 5000 строк кода.

В соответствии со спецификой задачи машинной транскрипции, распадающейся на две подзадачи – обучение и собственно транскрипция – рассматриваемая система имеет двухчастную структуру. Она состоит из двух подсистем: подсистемы обучения и подсистемы транскрипции. Две подсистемы могут использоваться как независимо друг от друга, так и в составе комплекса.

В подсистеме обучения реализован метод порождения множества правил транскрипции по обучающему множеству, описанный в главе 2. Подсистема принимает на вход обучающее множество имен на исходном языке и их транскрипций на целевой язык. Имена записаны в текстовом файле в формате *исходное_имя* *разделитель* *целевое_имя*, где *исходное_имя* – строка, состоящая из символов исходного алфавита, *разделитель* – знаки «,» или «;», *целевое_имя* –

строка, состоящая из символов целевого алфавита. Исходным алфавитом по умолчанию считается стандартный латинский алфавит, целевым алфавитом – стандартный кириллический алфавит, дополнительные символы, которые должны быть включены в алфавиты, могут быть заданы списком. Помимо обучающих данных, для корректной работы системе требуется список гласных букв исходного и целевого алфавита.

Подсистема включает в себя модуль порождения первичных правил, реализующий алгоритм порождения первичных правил (этап 1 порождения правил), а также модуль порождения сложных правил, реализующий алгоритм пополнения множества правил (этап 2 порождения правил).

Результатом работы подсистемы обучения транскрипции является система правил транскрипции, записанная в текстовом файле, каждая строка которого представляет собой правило транскрипции, записанное в формате $\{\gamma_1, \dots, \gamma_m\} \alpha \{\delta_1, \dots, \delta_n\} \rightarrow \beta$, где $\gamma_1, \dots, \gamma_m, \alpha, \delta_1, \dots, \delta_n \in \mathbf{V_I}$, $\beta \in \mathbf{V_O}$. Заметим, что оба модуля подсистемы возвращают корректный по форме результат – множество правил транскрипции. Таким образом, по желанию пользователя процесс обучения может быть ограничен порождением только первичных правил.

Подсистема транскрипции, осуществляющая преобразование строк, принимает на вход систему правил, записанную в текстовом файле в формате, описанном выше. Таким образом, система правил может быть как сгенерирована подсистемой обучения, так и написана вручную в соответствующем формате. Допускается также ручное редактирование правил, сгенерированных системой.

В состав подсистемы транскрипции входят три модуля. Модуль конверсии правил в конечный автомат осуществляет построение конечного автомата на основе поданной на вход системы правил. Модуль транскрипции осуществляет преобразование строки на исходном языке в строку на целевом языке с помощью конечного автомата. Автомат реализован в виде матрицы переходов, поэтому поиск перехода из данного состояния по данному символу реализуется как доступ к ячейке матрицы с номером столбца, соответствующего номеру текущего состояния и номеру строки, соответствующему номеру текущего символа. В стандартном контейнере vector библиотеки STL, с использованием которого построена матрица, такой доступ осуществляется за время $O(1)$.

Входная строка подается в модуль транскрипции из интерфейсного модуля, который осуществляет взаимодействие подсистемы с пользователем: установку используемой системы правил, запрос входной строки, возврат выходной строки, вычисленной модулем транскрипции.

Общая структура программного комплекса изображена на рисунке 17.

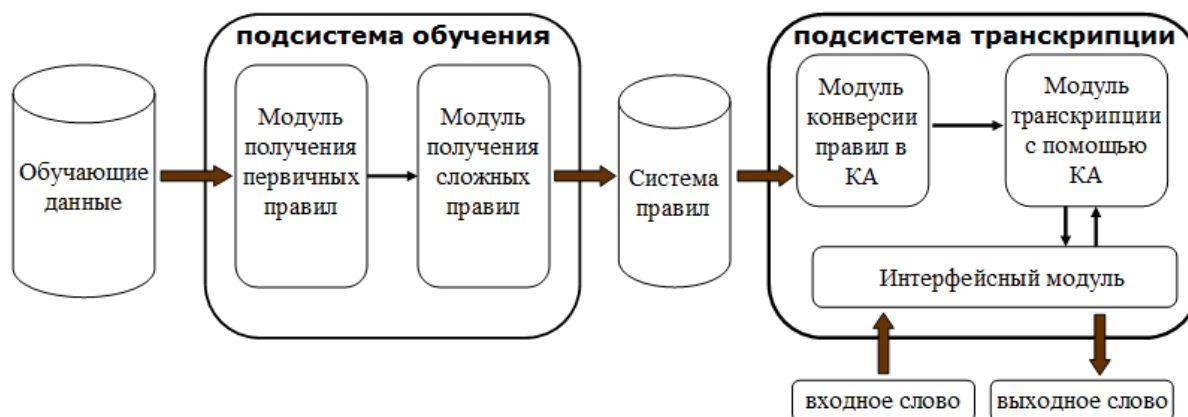


Рис 17. Структура программного комплекса.

В ходе реализации была разработана следующая структура классов, представленная на рисунке 18. Для хранения данных был создан базовый класс *Item*, содержащий в себе строку на исходном языке и соответствующую ей строку на целевом языке. От этого класса наследуются три класса, ядром которых является пара строк: класс *Word* (слово), класс *Syllable* (слог) и класс *Rule* (правило). В объектах класса *Word* хранятся имена из обучающего множества, а также определены операции – разделения слова на группы гласных и согласных и на слоги. Получаемые слоги хранятся в объектах класса *Syllable*. Класс *Rule* создан для хранения правил, которые также представляют собой пару строк: строка исходного языка и соответствующая ей строка целевого языка. Помимо свойств самого правила (левой и правой части, контекстов) в классе хранится массив указателей на слова, в которых было использовано данное правило. В классе также определена операция редактирования контекста правила. Сама же пара строк, составляющая правило, после создания не может быть изменена. Для хранения обучающего множества используется класс *WordSet*, порождаемая система правил транскрипции хранится в экземпляре класса *RuleSet*, который также обладает функциональностью нормализации системы правил (выравнивания контекстов, вычеркивания слишком

редких или длинных правил и т.д.). Обучение осуществляется с помощью объекта класса Converter, предоставляющий метод извлечения множества первичных правил из множества слов, а также метод пробного разбора слов и слогов с использованием системы правил.

Вся функциональность подсистемы транскрипции реализуется в классе Automaton. Его конструктор представляет собой процедуру конверсии системы правил в конечный автомат. Преобразование строк осуществляется с помощью метода этого класса.

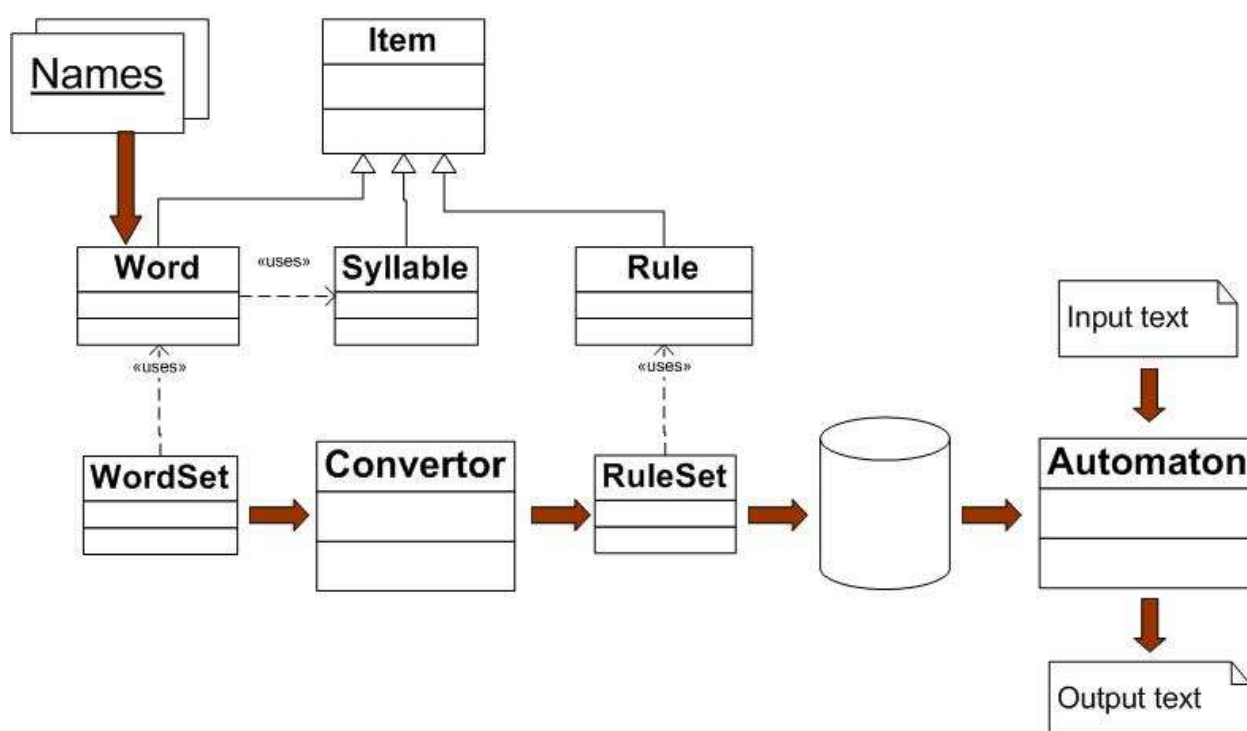


Рис.18. Структура классов системы транскрипции.

Рассмотрим порядок применения программной системы для решения задачи транскрипции. Общая схема технологического процесса представлена на рисунке 19. Первым этапом работы является формирование обучающего корпуса – множества имен исходного языка с корректной записью этих имен на целевом языке. В настоящей реализации использовались обучающие корпуса, составленные вручную, однако они могут быть созданы и автоматически и поданы на вход системе из стороннего приложения, при условии, что соблюден требуемый формат записи.

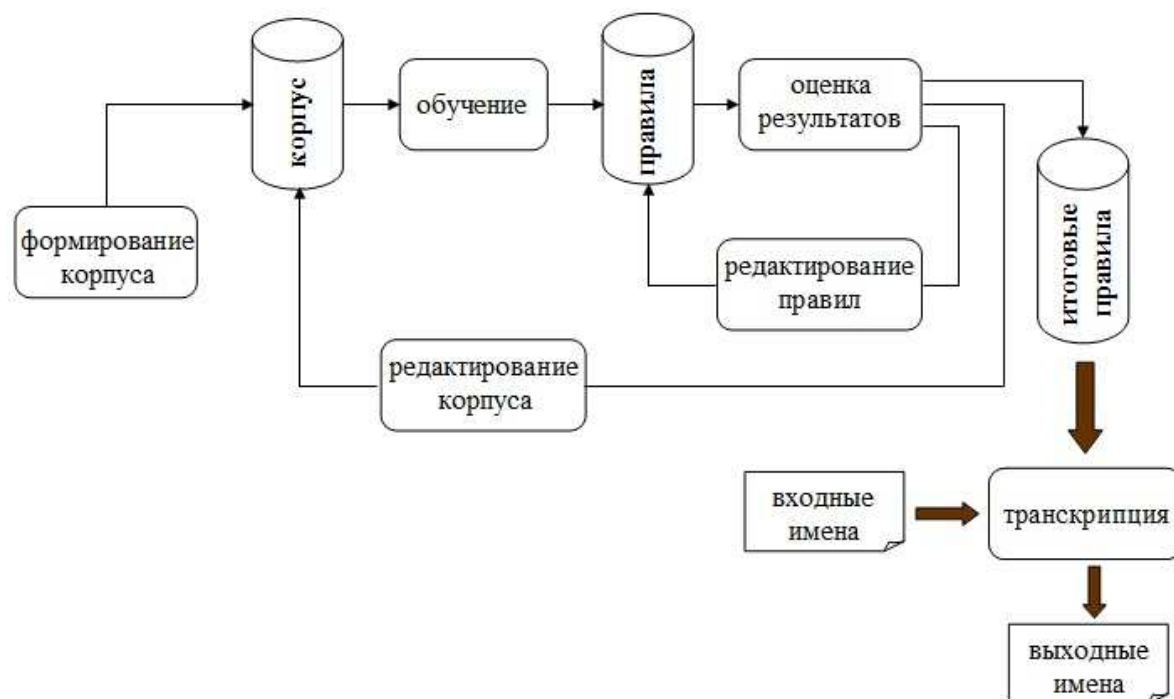


Рис 19. Последовательность применения разработанных компонентов для решения задач обучения системы транскрипции и преобразования имен.

Следующим этапом является обучение системы машинной транскрипции, все этапы которого подробно описаны в главе 2. В системе, помимо обычного хода алгоритма, доступен пошаговый запуск обучения, который в случае возникновения ошибок при обучении помогает определить, на каком этапе они возникают. Результатом обучения является множество правил транскрипции. Они могут быть переданы в подсистему транскрипции для проведения автоматической транскрипции вводимых пользователем строк или записаны в файл. Однако подсистема обучения предоставляет, помимо самого обучения, возможность оценки результатов – проверки сгенерированного множества правил на всем обучающем множестве или на тестовом множестве – десятой части обучающего множества, которая скрывается и не используется при обучении. Пользователю доступны результаты тестовой транскрипции – все варианты транскрипции для каждого имени и статистическая информация (о мерах, используемых для оценки данных, подробнее рассказано в разделе 2 настоящей главы). Интерфейс подсистемы обучения изображен на рисунке 20.

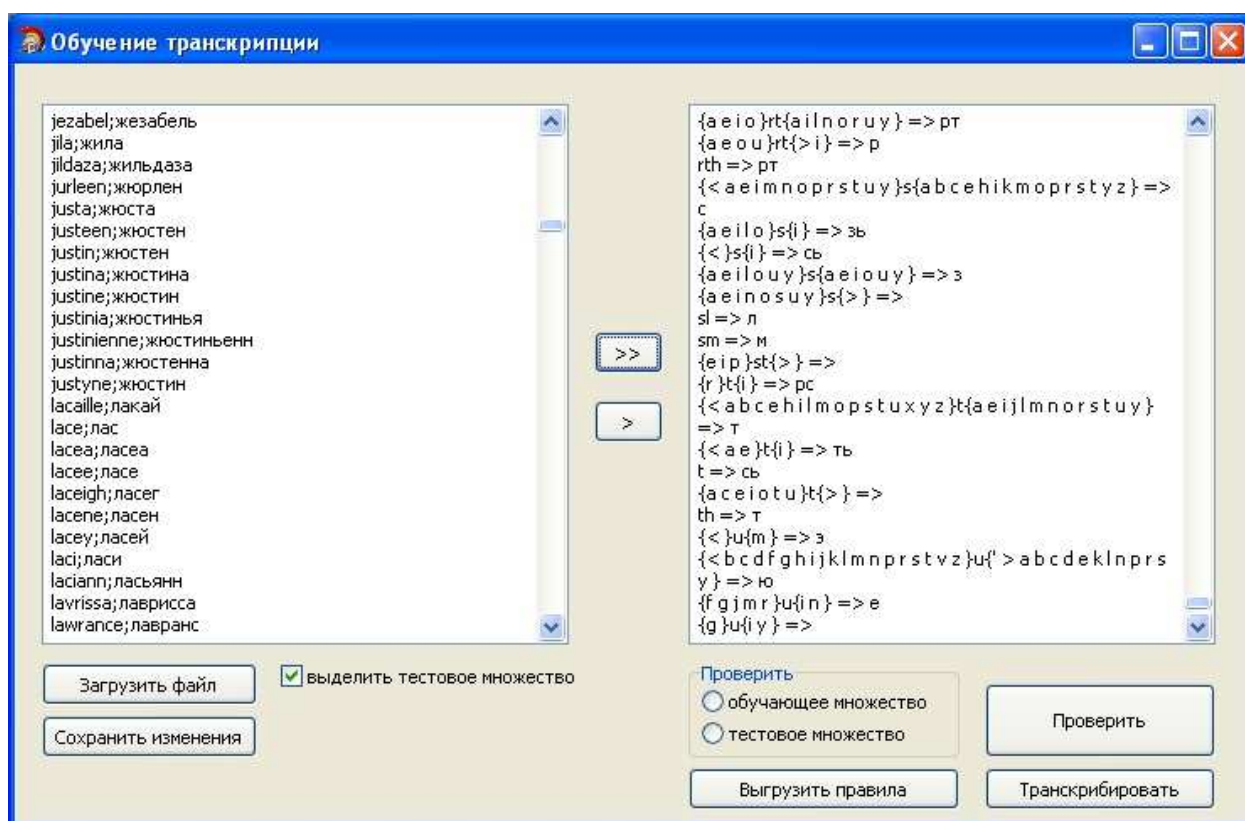


Рис. 20. Интерфейс подсистемы обучения.

В левом редактируемом поле отображается файл с обучающим множеством имен, по которому проводится обучение. Файл загружается по нажатию на кнопку «Загрузить файл». По кнопке «Сохранить изменения» обучающее множество с изменениями, внесенными пользователями, сохраняется на локальном диске. Если выделен флажок «выделить тестовое множество», одна десятая часть обучающего множества выделяется и не используется при обучении. В правом редактируемом поле отображаются правила, порождаемые системой. По нажатию на кнопку «>>» осуществляется **процесс обучения**. По нажатию на кнопку «>» обучение осуществляется пошагово: выполняется только один этап, его результаты отображаются в правом редактируемом поле, следующий этап осуществляется при повторном нажатии на кнопку «>>». С помощью кнопки «Проверить» запускается тестовая транскрипция: осуществляется транскрипция всех имен из указанного множества (обучающего или тестового), результат транскрипции и меры оценки качества для полученного результата выводятся в правом редактируемом поле. По нажатию на кнопку «Выгрузить правила» порожденные правила сохраняются в текстовом файле на локальном диске. Нажатие на кнопку «Транскрибировать» открывает подсистему транскрипции, которой передаются порожденные подсистемой обучения правила.



Рис. 21. Интерфейс подсистемы транскрипции

*В левой редактируемой строке пользователь вводит имя на исходном языке. По нажатию на кнопку «>>» осуществляется его **транскрипция**, результат которой отображается в правой редактируемой строке. По нажатию на кнопку «Загрузить систему правил» открывается диалог выбора файла с множеством правил транскрипции, после выбора файла на основе содержащегося в нем множества строится конечный автомат. После этого транскрипция осуществляется по правилам из выбранного файла. Сами правила отображаются в нижнем редактируемом поле. Правила могут быть отредактированы. По нажатию на кнопку «Сохранить изменения» измененное множество правил сохраняется на локальном диске.*

Доступ к подсистеме транскрипции может быть осуществлен из подсистемы обучения, но она может быть запущена и как независимое приложение. В этом случае перед началом транскрипции потребуется указать имя файла с множеством правил, по которым должна производиться транскрипция. Множество правил отображается в подсистеме в поле с возможностью редактирования, поэтому загруженные в систему правила могут быть изменены и транскрипция будет проведена с учетом изменений. Интерфейс подсистемы транскрипции изображен на рисунке 21.

4.2 Методы оценки качества систем транскрипции

Наиболее естественным и простым способом проверки качества преобразования строк системой транскрипции представляется сравнение результата ее работы с некоторым эталоном. Распространенной практикой при решении различных задач машинного обучения является вычисление ошибки при применении модели к тестовому множеству примеров. Из обучающих данных выделяется некоторое подмножество (тестовая выборка), обучение на котором не проводится, оно используется только для проверки точности полученной в результате обучения модели. Для задачи транскрипции тестовая выборка – это множество имен на исходном языке, перевод которых на целевой язык известен.

Простейшая мера оценки соответствия результата применения модели транскрипции к каждому из имен тестовой выборки – бинарная: правильно переведенным считается имя, если его перевод, предоставленный системой, полностью совпадает с эталонным переводом, все остальные случаи не учитываются. Точность перевода тестовой выборки имен определяется следующим соотношением:

$$WA = \frac{1}{N} \sum_{i=1}^N \{1, \text{если } \exists r_{i,j}: r_{i,j} = c_{i,k}; 0 \text{ иначе}\},$$

где N – количество слов в тестовой выборке, r_i – множество эталонных переводов для i -го слова тестовой выборки, c_i – множество переводов i -го слова тестовой выборки, предложенное системой транскрипции.

Эта мера в зарубежной литературе по машинной транскрипции обозначается как Word Accuracy (WA) и является одной из самых распространенных мер оценки. В данной работе эта мера в чистом виде не используется, но является основой других используемых мер. К сожалению, часто количество правильно переведенных слов не дает адекватного представления о качестве транскрипции: имя, в котором допущена одна ошибка, считается неправильно переданным, тогда как оно не перестает быть узнаваемым – а для многих задач этого достаточно. К тому же, наличие одного-двух неправильно переданных символов в имени свидетельствует о незначительных недостатках модели транскрипции, которые легче исправить, чем недостатки, вызывающие полностью неправильную передачу строки.

Для более точного учета ошибки системы транскрипции в зарубежной литературе используется мера Character Accuracy (CA) – посимвольная точность. Для имени тестовой выборки она вычисляется как нормированное расстояние редактирования, то есть количество операций удаления, добавления или замены символа, которые нужно применить к переводу, выданному системой, для получения эталонного перевода:

$$CA(\varphi) = \frac{len(\varphi) - EDist(\varphi, \varphi')}{len(\varphi)},$$

где φ – эталонный перевод, φ' – перевод, полученный системой транскрипции, функция $len()$ возвращает длину строки, а функция $EDist(\zeta, \eta)$ – расстояние редактирования между строками ζ и η . Существуют различные разновидности расстояния редактирования. Чаще всего используется расстояние Левенштейна [46], при вычислении которого операции удаления, вставки и замены символа имеют равный вес.

Для множества строк мера CA определяется как

$$CA = \frac{1}{N} \sum_{i=0}^N CA(w_i),$$

где w_i – слово тестовой выборки, N – количество слов в выборке.

Большая часть существующих в настоящее время систем машинной транскрипции пользуется статистическими методами обучения, результатом работы которых является вероятностная модель. Для некоторой строки на языке оригинала эта модель возвращает ее наиболее вероятный перевод на целевой язык. Однако, согласно этой модели, существует несколько допустимых переводов некоторой строки, и в некоторых практических приложениях машинной транскрипции полезно видеть несколько наиболее вероятных выходных строк. Для оценки такого формата результата транскрипции используются различные модификации меры **WA**, в общем случае называемые Тор-N ассигасу, где N – количество наиболее вероятных рассматриваемых результатов. Эта мера определяется как количество слов тестовой выборки, для которых правильный результат транскрипции был выдан среди первых N выданных результатов. Мера Тор-1 эквивалентна мере **WA**, если считать единственным результатом транскрипции наиболее вероятный результат. Наиболее распространена мера Тор-3.

Поскольку при использовании вероятностных правил выходные строки выдаются в виде списка, отсортированного по убыванию вероятности появления строки при заданных входных данных, имеет смысл оценка положения правильного перевода в списке предложенных вариантов. Многие исследователи оценивают результат машинной транскрипции с помощью меры, используемой в работах по информационному поиску для оценки ранжированного списка документов, выдаваемых поисковой машиной. Это мера **MRR** (Mean reciprocal rank), являющаяся средним обратным значением номера правильного перевода в списке предложенных, формально определяется как

$$MRR = \frac{1}{N} \sum_{i=1}^N \frac{1}{R_i},$$

где N – количество слов в тестовой выборке, R_i – номер правильного перевода в списке предложенных вариантов для i -го слова тестовой выборки.

Правила транскрипции, порождаемые предложенным в главе 2 методом, являются детерминированными, то есть в случае, когда правило может быть применено, оно применяется со стопроцентной вероятностью. Напомним, что система правил транскрипции почти всегда остается неоднозначной из-за неоднозначностей правил чтения языка оригинала, поэтому оправданно предложение нескольких возможных вариантов перевода для части слов. И система правил, и конечный автомат, который используется для преобразования строк с помощью этой системы, допускают выдачу более одного варианта транскрипции, но эти варианты не могут быть отсортированы ни по какому критерию, поэтому меры **MRR** и Top-N не могут использоваться как критерий качества транскрипции.

Для оценки качества транскрипции с помощью системы правил были предложены следующие меры:

- процент строк, для которых хотя бы один из предложенных вариантов перевода оказался корректным, то есть совпал с данным экспертом переводом; эта мера обозначена как Correct Transliteration (**CT**). Эта мера по сути является мерой Top-N с неопределенным N ;
- процент строк, для которых был предложен единственный вариант перевода, оказавшийся корректным; мера обозначена как Unique Correct Transliteration (**UCT**);

- мера неоднозначности перевода – среднее количество выдаваемых системой вариантов транскрипции строки; мера обозначена как Average Transliteration Variant (**ATV**).

Мера неоднозначности перевода по сути не является мерой качества транскрипции – это оценка степени неоднозначности языка, или, по крайней мере, степени неоднозначности, которая не может быть разрешена при порождении правил машинной транскрипции с помощью предложенного метода. Однако эта мера в комбинации с мерой **СТ** дает возможность сравнить качество транскрипции с помощью системы правил с качеством транскрипции статистических систем. Мера **СТ** для данной языковой пары можно считать равносильной мере Top-N, где N – значение меры **ATV** для данной языковой пары. Как будет показано в разделе «Численная оценка результатов», **ATV** ни для какого языка не превосходит 3, поэтому мера **СТ** в проведенных экспериментах равносильна мере Top-3.

Для оценки результатов транскрипции был также использован аналог меры **СА**, называемая в данной работе средним нормированным расстоянием Левенштейна и обозначенная как Average Normalized Levenshtein Distance (**ANL**). Для каждого имени тестовой выборки определяется нормированное расстояние Левенштейна: расстояние Левенштейна, деленное на длину имени. Таким образом, если перевод, предложенный системой, полностью совпадает с фактическим переводом, мера для данного слова будет равна нулю. Заметим также, что мера рассчитывается для каждого варианта перевода, предложенного системой, а затем делится на общее количество вариантов (а не на общее количество строк в тестовой выборке). Кроме того, считается средняя ошибка: среднее расстояние Левенштейна для неправильно переданных слов. Эта мера обозначена как Average Error (**AE**).

4.3 Обучающие данные

Метод порождения правил транскрипции для обучения требует параллельного корпуса имен на исходном и целевом языках. Во всех базах, использованных для экспериментов, целевым языком был русский. Базы были составлены вручную экспертами-лингвистами. Используемые базы перечислены в таблице 5.

Таблица 5. Обучающие корпуса

Исходный язык	Объем базы (количество пар)
Японский	7006
Китайский	4671
Немецкий	4208
Арабский	2109
Шведский	1777
Польский	1435
Испанский	1041
Французский	760
Румынский	576
Словенский	504
Тагальский	293
Монгольский	232
Хинди	161

Исходным алфавитом всех баз является латинский алфавит без диакритических знаков (исключение составляет немецкий, в котором помимо стандартной латиницы используются символы ä, ö и ü). Их отсутствие объясняется тем, что многие корпуса составлены из имен, взятых из документов машиносчитываемого формата. Отсутствие диакритических знаков часто создает неоднозначность, снижающую качество правил. Например, в шведском языке есть символ «å», который по правилам практической транскрипции должен передаваться на русский язык символом «о». Машиносчитываемый формат документов не допускает диакритических знаков, поэтому при формировании таких документов символ «å» заменяется на сочетание «aa». Исходя из этого при анализе обучающей выборки формируется правило «aa → о». Но в той же выборке присутствует также сочетание «aa», которое не является заменой символа «å», а составлено из двух «a», поэтому и на русский язык должно передаваться как «aa». Это порождает неразрешимую в рамках данной задачи неоднозначность, тогда как в исходном корпусе подобной неоднозначности нет.

В базах языков, пользующихся не латинским алфавитом (японский, китайский, хинди и пр.), использована их латинская транскрипция (для японского – транскрипция по системе Хепберна, для китайского – по системе Пиньин). Однако латинский алфавит не является принципиальным условием работы системы. В качестве исходного и целевого алфавитов могут быть использованы любые алфавиты. Единственное ограничение метода состоит в том, что письмо должно быть именно алфавитным, но не иероглифическим или слоговым.

4.4 Оценка метода порождения правил

4.4.1 Анализ правил транскрипции

На основе имеющихся корпусов были построены системы правил транскрипции с этих языков на русский. Для оценки правил было проведено сравнение систем правил, сгенерированной автоматически по описанному в статье методу, с системами правил, написанных вручную экспертом для системы «Трансскриба» (см. таблицу 6).

Таблица 6. Сравнение результатов ручной и автоматической генерации правил

Язык	Сгенерировано правил	Правил эксперта
Арабский	63	78
Испанский	88	106
Немецкий	102	121
Монгольский	41	46
Французский	160	356
Шведский	105	423
Японский	52	131

Сокращение количества правил может быть объяснено двумя причинами. Во-первых, некоторые правила не были сгенерированы. Подобная ситуация сложилась по нескольким причинам. Некоторые правила генерировались экспертами исходя из знаний языка, но примера на их употребление в коллекцию помещено не было. Так, во французском языке эксперт поместил правило, что «ai» в начале слова переходит в «э», тогда как подобного примера в коллекции не было. Часть правил отсеивалась

как встретившиеся только один раз и незначимые. И, наконец, некоторые правила просто не были порождены предлагаемым методом. Так, например, для арабского языка экспертом было записано правило «уаа → я», тогда как программой были получены правила «аа → а» и «уа → я» (которые также есть у эксперта), а правило «уаа → я» порождено не было. Для японского языка экспертом были записаны слоги «ha», «sha» и «cha», передаваемые как «ха», «ся» и «тя», соответственно. Системой же было порождено единое правило: «а» с левым контекстом «h» передается как «я». При включении в систему этапа порождения длинных контекстов это правило дополнилось левыми контекстами «sh» и «ch».

Во-вторых, при генерации произошло сжатие правил по сравнению с правилами, составленными экспертами. Так. Например, во французском языке экспертами в связи с особенностями применявшегося ранее метода был порожден целый комплект правил, касающихся прочтения сочетания «ai» в различных контекстах, тогда как в данной версии было сгенерировано лишь одно - «ai → е», покрывающее все эти случаи (кроме «ai» в начале слова). Для японского языка слоговая азбука была редуцирована в правила чтения отдельных сочетаний. Так, для трех слогов, начинающихся с «ch» было сгенерировано единое правило «ch → т». Заметим, что подобный подход является корректным, но может оказаться непривычным для лингвиста, корректирующего правила.

В остальном правила являются корректными, покрывают большую часть тестовой базы и позволяют проводить транскрипцию имен собственных для языков без разработанных правил машинной транскрипции.

Следует также рассмотреть вопрос необходимого и достаточного для обучения объема обучающей выборки и зависимости качества правил от объема данных. К сожалению, оценка качества обучающих данных – сложная задача. Под качеством в данном случае понимается наличие в базе достаточного количества случаев употребления всех символов и устойчивых сочетаний символов, характерных для данного языка, снабженных достаточным количеством контекстов, позволяющих получить полное представление об их употреблении в языке. Оценка данных с этой точки зрения может быть произведена только вручную, к тому же, сложно выразить такую характеристику набора данных количественно, поэтому

сравнение двух корпусов имен по принципу полноты затруднительно. Будем считать, что с увеличением объема корпуса увеличивается и его полнота.

Точно так же и полнота системы правил зависит от ее объема: наиболее вероятно, что система из 60 правил более полно описывает правила чтения языка, чем система из 20 правил. Существует вероятность того, что система правил большего объема не более точна, а избыточна. Но предложенный автором метод выделения правил стремится уменьшить избыточность системы: так, например, метод устраняет все правила, объяснимые с помощью других правил. Приняв это допущение, логично предположить, что с увеличением количества имен в обучающей выборке количество порождаемых системой правил увеличивается, так как ей предоставляются новые прецеденты. Было решено выяснить, как влияет увеличение объема обучающего корпуса на количество правил и их качество: действительно ли увеличивается количество правил при расширении корпуса, какой объем данных является оптимальным для работы метода, являются ли достаточными данные, на которых проводились эксперименты.

Для ответа на эти вопросы была проведена серия экспериментов: на основе большого обучающего корпуса было сформировано несколько обучающих корпусов меньшего размера – от 100 до N , где N – количество имен в большом корпусе. На основе каждого из этих корпусов была сгенерирована система правил транскрипции, а затем проведено сравнение количества правил в системах, обучившихся на корпусах разного объема. Испытания проводились на польском (общий объем корпуса – 1435 имен), китайском (4671 имя), немецком (4204 имени) и японском (7006 имен) языках. На рисунках 22а – 22г изображены графики зависимости объема системы правил от объема обучающей выборки, полученные в результате проведенных экспериментов. Синяя линия обозначает исходные данные экспериментов, красная линия – сглаженный график функции, позволяющий оценить общий тренд. Все графики имеют похожую форму: функция, по которой они построены, возрастает, но по мере увеличения x (количества правил) интенсивность возрастания снижается. Это значит, что происходит насыщение – то есть от увеличения объема обучающей выборки объем системы правил не возрастает или возрастает незначительно. Можно сделать вывод, что для порождения полной системы правил для данного языка достаточно обучающей выборки объемом M .

Число M для разных языков может различаться: для польского оно составляет 1000, для китайского и немецкого – 4000, для японского – 5000. В любом случае такой объем обучающих данных гораздо меньше того, что требуется статистическим методам для создания модели транскрипции, преобразующей строки с высокой точностью. Однако нельзя сказать, что увеличение объема обучающих данных сверх числа M при обучении системы транскрипции с помощью представленного в данной работе метода бесполезно или вредно. Оно не создаст новых правил, но позволит существующим правилам расширить контексты и тем самым снимет часть неоднозначностей.

Что же касается колебаний графиков зависимости количества правил от количества имен, они объясняются следующим образом. При порождении правил не учитываются соответствия, редко (в данной реализации системы – менее 3 раз) встречающиеся в обучающей выборке, так как велика вероятность того, что эти соответствия – результат ошибки в выборке или исключение из правил. При расширении выборки соответствия, которые ранее из-за недостаточной представленности считались исключениями, выделяются в правила, в результате чего количество правил увеличивается. Однако при дальнейшем расширении выборки может оказаться, что два правила, выделявшиеся независимо друг от друга, являются частными случаями одного и того же правила. Это приводит к уменьшению количества правил при незначительном расширении обучающей выборки. При этом общей тенденцией все равно остается увеличение количества правил.

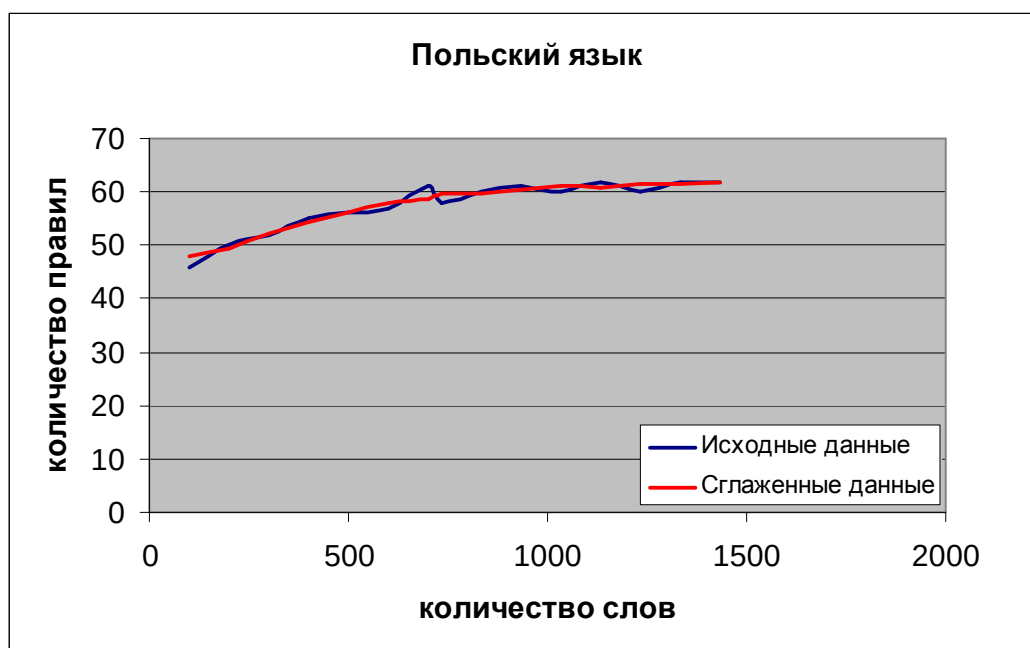


Рис. 22а

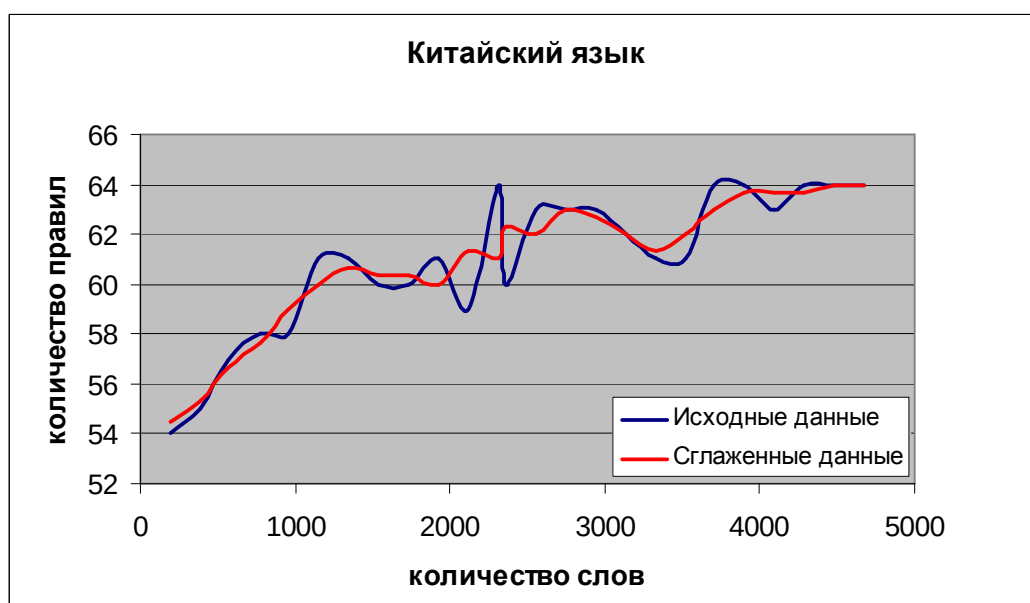


Рис. 22б

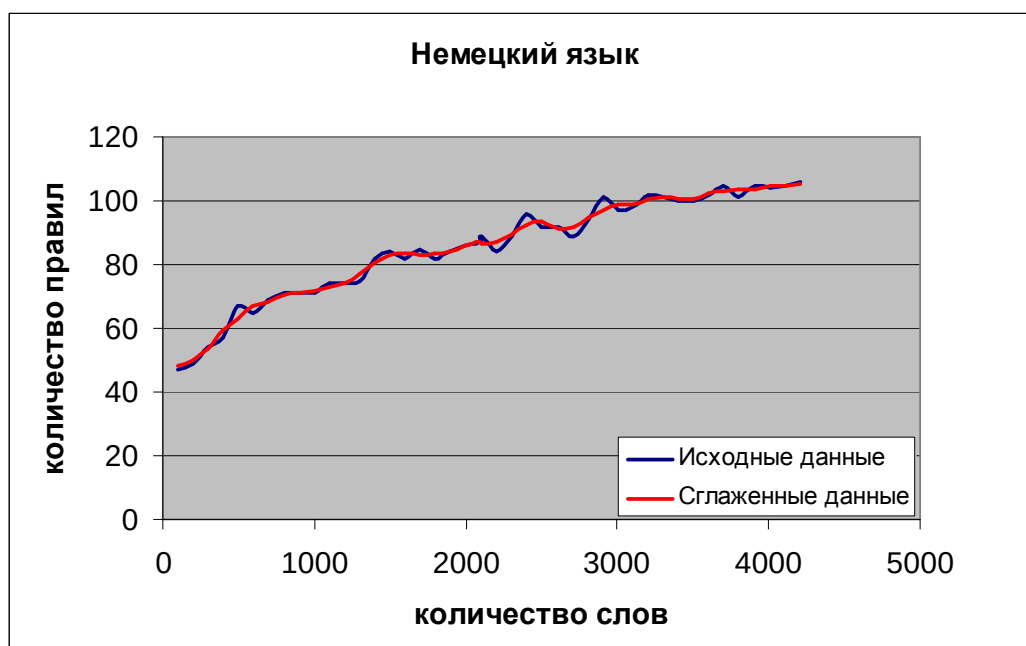


Рис. 22в

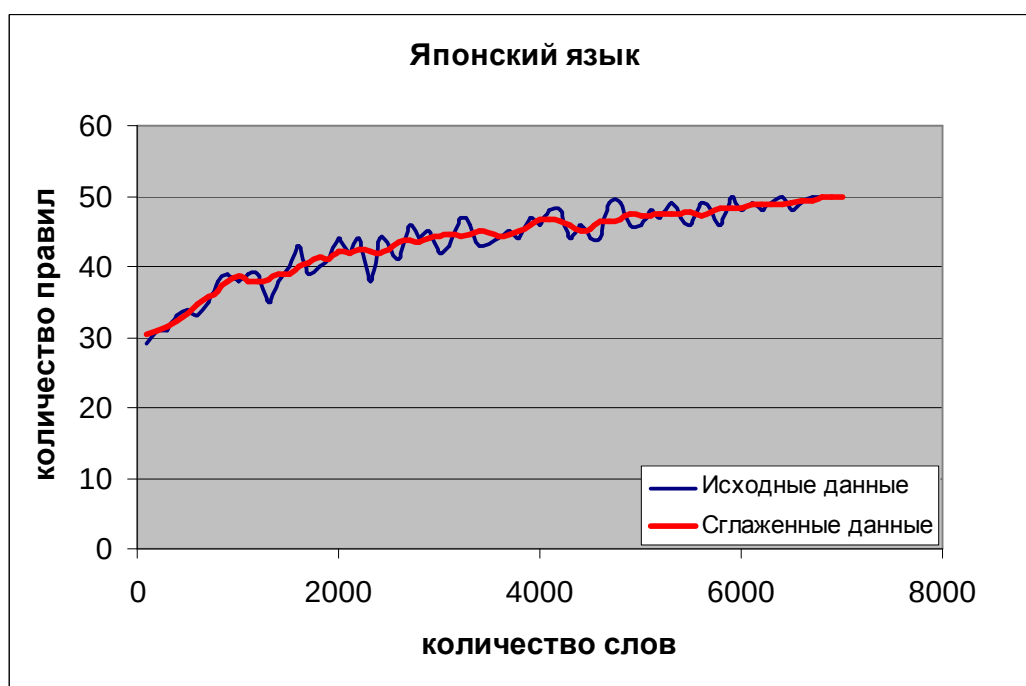


Рис. 22г

Рис. 22а-г. Графики зависимости количества правил в системе транскрипции от количества имен в обучающей выборке.

4.4.2 Численные оценки результатов

Самыми показательными результатами, конечно же, являются численные оценки качества обучения системы. Были проведены эксперименты, позволяющие оценить качество обучения: на некотором тестовом множестве была получена

система правил транскрипции, затем эти правила были применены к тому же тестовому множеству.

Для каждого из языков было проведено два эксперимента:

1. извлечение правил по алгоритму, описанному в главе 2;
2. извлечение правил по тому же алгоритму, дополненному извлечением контекстов длины 2.

Таблица 7. Оценка качества обучения метода. Для мер СТ и UCT указано количество правильно и однозначно переданных слов соответственно, в скобках указано, какой процент тестовой выборки составили эти слова

Исходный язык	Всего имен	СТ		UCT		ATV	
		1	2	1	2	1	2
Японский	7006	7005 (99%)	6892 (98%)	4778 (68%)	5567 (79%)	1,38	1,2
Китайский	4671	4662 (99%)	4589 (98%)	4364 (93%)	4557 (97%)	1,06	1,01
Немецкий	4204	4132 (98%)	-	3853 (91%)	-	1,07	-
Арабский	2109	2102 (99%)	1999 (94%)	1793 (85%)	1797 (85%)	1,18	1,11
Французский	1881	1596 (84%)	1553 (82%)	547 (29%)	855 (45%)	2,8	1,14
Шведский	1777	1720 (96%)	1694 (95%)	985 (55%)	1077 (60%)	1,64	1,47
Польский	1435	1422 (99%)	1179 (82%)	1218 (84%)	1040 (72%)	1,16	1,11
Турецкий	1062	976 (91%)	-	625 (58%)	-	1,56	-
Испанский	1041	1026 (98%)	1011 (97%)	778 (74%)	886 (85%)	1,33	1,14
Румынский	576	565 (98%)	544 (94%)	295 (51%)	373 (64%)	1,78	1,38
Словенский	504	502 (99%)	502 (99%)	354 (70%)	480 (95%)	1,3	1,04
Тагальский	293	257 (87%)	257 (87%)	225 (76%)	227 (77%)	1,14	1,13
Монгольский	231	231 (100%)	-	227 (98%)	-	1,02	-
Хинди	161	160 (99%)	-	152 (94%)	-	1,04	-

Результаты оценки качества обучения представлены в таблице 7. Оценка проводилась по трем мерам: СТ, UCT и ATV. Значение каждой меры указано для

обоих экспериментов: для эксперимента 1 – в столбцах под цифрой 1, для эксперимента 2 – в столбцах под цифрой 2. Для некоторых языков данные по эксперименту 2 отсутствуют (строки, соответствующие этим языкам, помечены в таблице серым цветом). Это значит, что контексты длины 2 для этих языков не были порождены, следовательно, результаты второго эксперимента не отличаются от результатов первого.

Мера СТ, то есть количество правильно переведенных имен почти для всех языков превосходит 95%, что значительно выше результатов, сообщаемых зарубежными исследователями. Что же касается контекстов длины 2, почти на всех обучающих множествах наблюдается следующая тенденция: извлечение длинных контекстов снижает значение меры СТ, но повышает значение меры УСТ, то есть общее количество корректно переведенных имен снижается, но количество имен, для которых правильный ответ был единственным предложенным, возрастает. Это значит, что повышается общая однозначность системы правил. Об этом свидетельствует и значение меры ATV, которое для второго эксперимента меньше, чем для первого.

Для оценки полноты построенной системы правил с точки зрения правил чтения данного языка и качества транскрипции необходимо было провести транскрипцию на тестовой выборке имен – то есть, на параллельном корпусе имен, не использовавшемся при генерации правил. Распространенной практикой является отделение от обучающих данных десятой части и проверка на ней результатов обучения. Однако маленький размер обучающих корпусов для многих языков (напр. хинди, монгольский) не позволил провести такую проверку. Выборка, составляющая десятую часть маленького корпуса, нерепрезентативна, к тому же, ее исключение из обучающего множества может привести к тому, что некоторые правила не будут порождены (например, если какой-либо характерный для данного языка диграф встретился в данных только один раз, и слово, содержащее его, попало в тестовую выборку). Было решено выделить тестовые множества имен только для тех корпусов, в которых больше тысячи слов, так, чтобы тестовая выборка составляла по крайней мере сто слов.

Таблица 8. Оценка качества транскрипции

Язык	Объем тестовой выборки	СТ	UCT	ANL	AE
Японский	701	664 (94%)	496 (70%)	0,043	1,125
Китайский	468	436 (93%)	406 (86%)	0,026	1,23
Немецкий	421	339 (80%)	325 (77%)	0,032	1,176
Арабский	211	169 (80%)	160 (75%)	0,054	1,5
Шведский	178	149 (83%)	82 (46%)	0,131	1,33
Польский	144	138 (95%)	123 (85%)	0,027	1,5
Испанский	105	99 (94%)	76 (72%)	0,038	1,125

При оценке результатов транскрипции тестовой выборки использовались те же меры, что и при оценке качества обучения метода: процент слов, для которых хотя бы один из предложенных вариантов перевода оказался правильным, процент слов, для которых правильный вариант перевода был единственным вариантом, предложенным системой, вероятность правильного перевода.

Кроме этих мер, были введены две новые меры оценки ошибки, основанные на расстоянии Левенштейна. Расстоянием Левенштейна называется количество операций добавления, удаления или замены символа, необходимых для преобразования строки А в строку В [46].

Результаты проверки качества транскрипции представлены в таблице 8.

4.5 Оценка метода транскрипции

Помимо качества транскрипции была оценена и скорость преобразования строк с помощью нового метода. Скорость нового метода транскрипции с помощью конечного автомата, предложенного в данной работе, было решено сравнить со скоростью преобразования строк другой системы транскрипции, основанной на правилах: системы «Транскриба».

Метод транскрипции, реализованный в системе «Транскриба», обладает сложностью $O(\log(M) \cdot N)$, где М – количество правил транскрипции, N – длина строки. Метод транскрипции с помощью конечного автомата, как было показано

выше, обладает линейной сложностью относительно длины разбираемой строки ($O(N)$).

Таблица 9. Результаты вычислительных экспериментов. Китайские имена записаны с помощью латинского алфавита по системе Уэйда

Язык	Количество слов	Транскриба		Конечный автомат	
		Количество правил	Время (мс)	Количество правил	Время (мс)
Шведский	1884	423	780	115	63
Французский	1882	356	970	160	50
Испанский	1066	107	320	87	15
Монгольский	232	46	90	41	<1
Китайский*	2680	337	940	59	60

Была проведена транскрипция имен из тестовых корпусов для различных языков (см. таблицу 9. Языки оригинала перечислены в таблице, язык перевода для всех корпусов – русский). Имена из каждого корпуса были переведены с помощью обоих методов транскрипции. При экспериментах с системой «Транскриба» использовались правила, написанные для этой системы вручную, правила для конечного автомата были сгенерированы с помощью системы автоматического порождения правил транскрипции по методу, описанному в главе 2. Разный объем правил не позволяет точно оценить выигрыш по скорости, но все же дает представление о порядке.

Согласно описанию системы «Транскриба», изложенном в [111], скорость преобразования строк с помощью этой системы составляет $\log(M) \cdot N$, где M – количество правил транскрипции, а N – длина строки. Скорость же предложенного в данной работе метода преобразования строк с помощью конечного автомата, согласно расчетам, произведенным в главе 3, составляет не более $6 \cdot N$, то есть линейна относительно длины преобразуемой строки.

На рисунке 23а представлен график зависимости времени транскрипции с помощью системы «Транскриба» от заявленной скорости работы, аналогичный график представлен на рисунке 23б для конечного автомата – график зависимости

времени от количества преобразуемых строк (для простоты вычислений было принято допущение, что все строки в тестовых выборках одинаковой длины).

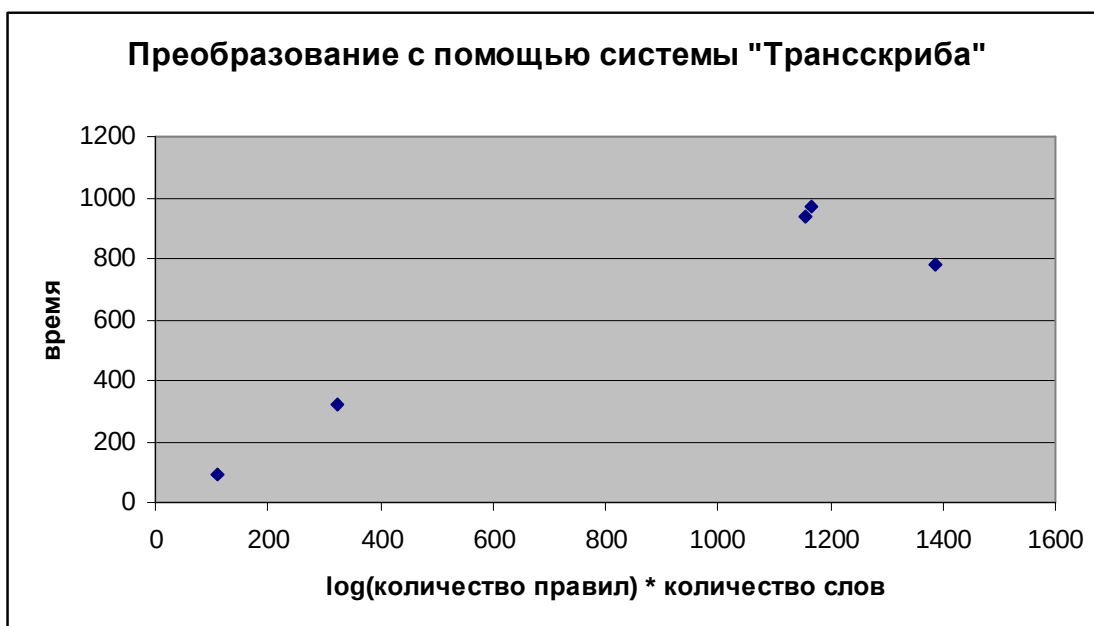


Рис. 23а. Зависимость времени транскрипции от количества правил и количества преобразуемых слов

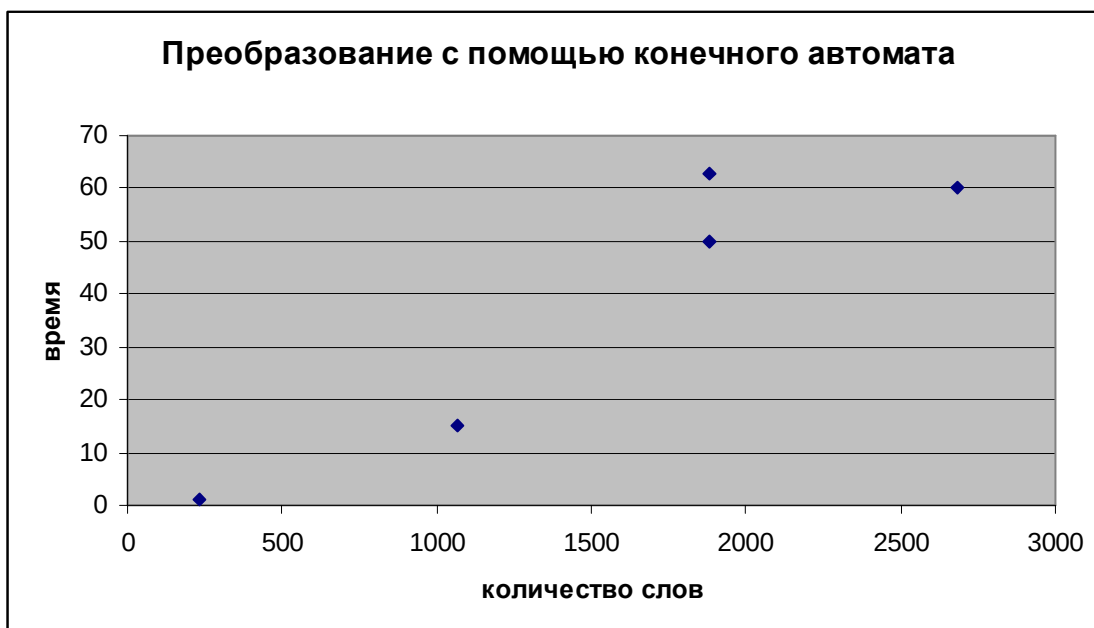


Рис. 23б. Зависимость времени транскрипции от количества преобразуемых слов

4.6 Выводы к главе 4

В рамках исследования была создана система машинной транскрипции, использующая методы, предложенные в главах 2 и 3. Создание системы позволило оценить качество предложенных методов. Были проведены эксперименты, оценивающие качество и скорость транскрипции. Качество обучения (транскрипция

имен из обучающего множества) для подавляющего большинства языков превышает 95%, качество транскрипции (транскрипция имен из тестового множества) для большинства языков составляет 90 – 95%, что также является довольно высоким результатом. Было проведено сравнение двух модификаций метода порождения правил транскрипции: с контекстами длины 2 и без них. Эксперименты показали, что введение длинных контекстов увеличивает однозначность системы правил, но ухудшает ее качество: понижается процент слов, для которых был предложен правильный вариант перевода, но при этом возрастает количество слов, для которых был предложен единственно правильный вариант перевода, а также уменьшается количество предлагаемых вариантов, что в некоторых случаях может быть важнее, чем абсолютное количество правильно переданных строк.

Было проведено сравнение правил, порождаемых системой, с правилами, составленными экспертом вручную. Автоматически сгенерированное множество содержит меньше правил, причем в большинстве случаев это объясняется большей компактностью правил, порожденных системой (то есть одно правило заменяет несколько правил, составленных вручную).

Было оценено необходимое для обучения количество имен в обучающем множестве. Проведены эксперименты, показавшие, что существует порог насыщения обучающего корпуса, после достижения которого количество правил, порождаемых системой, не увеличивается или увеличивается незначительно.

Кроме того, было экспериментально доказано значительное увеличение скорости транскрипции с помощью системы правил при использовании в качестве модели транскрипции расширенного конечного автомата, описанного в главе 3. Ускорение по сравнению с методом, основанном на последовательном выборе применимых правил из множества, составило от 3 до 90 раз в зависимости от средней длины имен в тестовом множестве и сложности правил.

Заключение

Имена собственные не могут быть переведены так же, как обычные слова, поэтому для их передачи на другой язык используется практическая транскрипция – передача с сохранением звучания имени. Развитие электронного документооборота, а также систем машинного перевода, информационного поиска потребовало автоматизировать процесс практической транскрипции. Компьютер может осуществлять транскрипцию между парой языков с помощью системы написанных вручную правил. Однако во многих приложениях требуется практическая транскрипция имен с большого количества языков. Составление правил для каждой пары – долгий и трудоемкий процесс. Поэтому очень многих исследователей привлекает задача автоматического обучения систем практической транскрипции.

Как показал анализ, существующие на сегодняшний день системы машинной транскрипции обладают некоторыми недостатками, которые часто делают невозможным их применение. Во-первых, большая часть систем использует для обучения статистические методы, что влечет за собой сразу несколько проблем: необходимость использования больших корпусов обучающих данных, которые недоступны для многих языков, невозможность ручного редактирования полученной модели, непредсказуемость получаемых при ее использовании ошибок. Во-вторых, несмотря на заявленную универсальность методов обучения, редкие из них способны работать более чем с двумя языками, для обучения на новых языках требуется введение дополнительной информации и настройка параметров.

По этим причинам требуется создание нового метода машинной транскрипции, свободной от недостатков существующих методов. Метод должен порождать правила, записанные в явном виде и поддающиеся последующему ручному редактированию, чтобы в случае недостатков была возможность улучшить систему без повторного обучения. Кроме того, метод должен применять полученные правила за линейное время, что недоступно существующим на сегодняшний день системам.

В данной работе описана методика автоматизации практической транскрипции имен собственных, включающая в себя два метода: метод автоматического порождения правил транскрипции и метод транскрипции имен собственных с помощью системы правил транскрипции.

Метод порождения правил независим от языка, он не требует дополнительной лингвистической или иной информации. Метод чувствителен к ошибкам в обучающем множестве, но для обучения ему достаточно небольшого множества прецедентов (имен и их переводов). Метод является алгоритмическим, а не стохастическим. Он состоит из двух основных этапов: порождение первичных правил и порождение сложных правил. Первичные правила порождаются на основе выравнивания обучающего корпуса. Выравнивание производится на основе информации о типе составляющих имя букв (гласные и согласные), буквы сопоставляются друг другу, только если они принадлежат к одному типу, поэтому процент некорректных соответствий на этом этапе очень низок. На этапе порождения сложных правил производится анализ имен обучающего множества с помощью существующих правил, в результате которого, во-первых, выделяются новые правила, а во-вторых, редактируются существующие.

Преобразование строк осуществляется с помощью расширенного конечного автомата. За счет возможности передвижения по строке как вперед, так и назад на произвольное количество символов автомат имеет возможность учитывать контексты правила, однако сохраняет линейную относительно длины строки скорость разбора. При этом скорость работы автомата не зависит от объема системы правил, что очень важно для языков со сложными соответствиями фонетики и графики (например, французский, немецкий). Автомат является детерминированным, но может выдавать несколько вариантов разбора, что также очень важно при существовании в языке нескольких норм произношения или при наличии в обучающем множестве имен исключений.

Список литературы

1. Сухотин А.М. О передаче иностранных географических названий // Вопр. географии и картографии. – М., 1935. – Сб. 1. – С. 144-145.
2. Реформатский А.А. Введение в языкознание. – М., 1947.
3. Гиляревский Р.С. Иностранные имена и названия в русском тексте. Справочник / Р.С. Гиляревский, Б.А. Старостин, 3 изд. – М., 1985.
4. Ермолович Д.И. Имена собственные: теория и практика межъязыковой передачи. – М.: Р.Валент, 2005.
5. Лингвистический транслятор/транскриптор [Электронный ресурс] // 2007. – Режим доступа: <http://www.lingvoconverter.com/>.
6. Система транскрипции «Transcriptor.ru» [Электронный ресурс] // 2008. – Режим доступа: <http://transcriptor.ru/>.
7. Транскриптор студии Артемия Лебедева [Электронный ресурс] // 2012. – Режим доступа: <http://www.artlebedev.ru/tools/transcriptor/>.
8. Arbabi M. Algorithms for Arabic name transliteration / M. Arbabi, S.M. Fischthal, V.C. Cheng, E. Bart, // IBM Journal of research and Development. – 1994. – Vol.38, Issue 2. – P. 183–194.
9. Divay M. Algorithms for grapheme-phoneme translation for English and French: applications for database searches and speech synthesis / M. Divay, A.J. Vitale // Computational Linguistics. –1997. – Vol. 23, issue 4. – P. 495–523.
10. Логачева В.К. Современные методы практической транскрипции. [Электронный ресурс] / В.К. Логачева, Э.С. Клышинский, В.А. Галактионов // Препринты ИПМ им. М.В.Келдыша. – 2012. – № 13. 18 с. Режим доступа: <http://library.keldysh.ru/preprint.asp?id=2012-13>.
11. Практическая транскрипция личных имен в языках народов мира, под ред. Клышинского. – М.: Наука, 2010. – 679 с.
12. Бондаренко А.В. Формальный метод транскрипции иностранных имен собственных на русский язык / А.В. Бондаренко, Ю.В. Визильтер, В.И. Горемычкин, Э.С. Клышинский // Программные продукты и системы. – 2010. – № 1.
13. Бондаренко А.В. Автоматическая транскрипция именных групп в процессах обработки машиносчитываемых проездных документов / А.В. Бондаренко,

- А.А. Герасименко // Вестник компьютерных и информационных технологий. – 2008.
14. Гращенко Л.А. Математические основы автоматизированной таджикско-персидской конверсии графических систем письма: диссертация на соискание ученой степени кандидата физико-математических наук: 05.13.18. / Гращенко Леонид Александрович – Душанбе, 2010. – 115 с.
 15. Malik M.G.A. Punjabi machine transliteration. // Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL. – Sydney, Australia, 2006. – P.1137–1144.
 16. Brown P.F. The mathematics of statistical machine translation: Parameter estimation / P.F. Brown, V.J.D. Pietra, S.A.D. Pietra, R.L. Mercer // Computational Linguistics. – 1993. – Vol.19, issue 2. – P. 263–311.
 17. Dempster A. Maximum likelihood from incomplete data via the EM algorithm / A. Dempster, N. Laird, D. Rubin // Journal of the Royal Statistical Society. – 1977. – Vol.39, issue 1. – P. 1-38.
 18. Vogel S. HMM-based word alignment in statistical translation / S. Vogel, H. Ney, C. Tillmann // Proceedings of the 16th Conference on Computational linguistics. – Copenhagen, Denmark, 1996. – P. 836-841.
 19. Toutanova K. Extensions to HMMbased statistical word alignment models / K. Toutanova, H.T. Ilhan, C.D. Manning // Proceedings of the Conference on Empirical methods in Natural Language Processing. – Pennsylvania, Philadelphia, 2002. – P. 87-94.
 20. Al-Onaizan Y. Statistical machine translation / Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, D. Melamed, F. J. Och, D. Purdy, N. Smith, D. Yarowsky, // Tech. rep., Johns Hopkins University. – 1999.
 21. Och F.J., Ney H. A systematic comparison of various statistical alignment models / F.J. Och, H.A. Ney // Computational Linguistics. – 2003. – Vol.29, issue 1. – P. 19-51.
 22. Covington M.A. An algorithm to align words for historical comparison // Computational Linguistics. – 1996. – Vol.22, issue 4. – P. 481-496.

23. Kang B.-J. Automatic transliteration and backtransliteration by decision tree learning / B.-J. Kang, K.-S. Choi // Conference on Language Resources and Evaluation. – Athens, Greece, 2000. – P. 1135–1411.
24. Oh J.-H. An English-Korean transliteration model using pronunciation and contextual rules / J.-H. Oh, K.-S. Choi // Proceedings of the 19th International Conference on Computational linguistics. –Taipei, Taiwan, 2002.
25. Kondrak G. A new algorithm for the alignment of phonetic sequences // Proceedings of 1st Conference of the North American Chapter of the Association for Computational Linguistics (NAACL). – Seattle, Washington. – 2000. – P. 288-295.
26. Bayes T. An Essay towards solving a Problem in the Doctrine of Chance. By the late Rev. Mr. Bayes, communicated by Mr. Price, in a letter to John Canton, M.A. and F.R.S. / T. Bayes, R. Price // Philosophical Transactions of the Royal Society of London. – 1763. – Vol.53. P. 370-418.
27. Knight K. Machine transliteration / K. Knight, J. Graehl // Computational Linguistics. – 1998. – Vol.24, issue 4. – P. 599-612.
28. Sundberg R. Maximum likelihood theory and applications for distributions generated when observing a function of an exponential family variable. // Dissertation. – Institute for Mathematical Statistics, Stockholm University. – 1971.
29. Sundberg R. Maximum likelihood theory for incomplete data from an exponential family. // Scandinavian Journal of Statistics. – 1974. – Vol.1, issue 2. P. 49-58.
30. Sundberg R. An iterative method for solution of the likelihood equations for incomplete data from exponential families. // Communications in Statistics – Simulation and Computation. – 1976. – Vol.5, issue 1. – P. 55-64.
31. The Carnegie Mellon University Pronouncing Dictionary. [Электронный ресурс] // 2012. – Режим доступа: <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>.
32. Graehl J. Carmel finite-state toolkit. [Электронный ресурс] / J. Graehl // 1997. – Режим доступа: <http://www.isi.edu/licensed-sw/carmel>.
33. Al-Onaizan Y. Machine transliteration of names in Arabic text / Y. Al-Onaizan, K. Knight // Proceedings of the Association for Computational Linguistics (ACL) workshop on Computational approaches to Semitic languages. – Philadelphia, PA, 2002. – P. 1-13.

34. Stalls B. Translating names and technical terms in Arabic text / B. Stalls, K. Knight // Proceedings of the Conference on Computational Linguistics (COLING) Workshop on Computational Approaches to Semitic Languages. – Montreal, Canada, 1998. – P. 34–41.
35. Jeong K. Automatic identification and back-transliteration of foreign words for information retrieval / K. Jeong, S. Myaeng, J. Lee, K. Choi // Information Processing and Management. – 1999. – Vol.35, issue 4. – P. 523-540.
36. Jung S. Y. An English to Korean transliteration model of extended Markov window / S.Y. Jung, S.L. Hong, E. Paek // Proceedings of the 18th Conference on Computational linguistics. – Saarbrücken, Germany, 2000. – P. 383-389.
37. Lin W.-H. Backward machine transliteration by learning phonetic similarity / W.-H. Lin, H.-H. Chen // Proceeding of the 6th Conference on Natural Language Learning. – Taipei, Taiwan, 2002. – P. 1–7.
38. Virga P. Transliteration of proper names in cross-lingual information retrieval / P. Virga, S. Khudanpur // Proceedings of the ACL Workshop on Multilingual and Mixed-Language Named Entity Recognition. – Sapporo, Japan, 2003. – P. 57-64.
39. Gao W. Phoneme-based transliteration of foreign names for OOV problem / W. Gao, K.-F. Wong, W. Lam // Proceedings of the 1st International Joint Conference on Natural Language Processing. Lecture Notes in Computer Science. – Springer, 2004. – Vol.3248. – P. 110-119.
40. Kang I.-H. English-to-Korean transliteration using multiple unbounded overlapping phoneme chunks / I.-H. Kang, G. Kim // Proceedings of the 18th Conference on Computational Linguistics. – Saarbrücken, Germany, 2000. – P. 418-424.
41. AbdulJaleel N. Statistical transliteration for English-Arabic cross language information retrieval / N. AbdulJaleel, L.S. Larkey // Proceedings of Conference on Information and Knowledge Management. – New Orleans, Louisiana, 2003. – P. 139-146.
42. Al-Onaizan Y. Translating named entities using monolingual and bilingual resources. / Y. Al-Onaizan, K. Knight // Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. – Philadelphia, PA, 2002. – P. 400-408.

43. Quinlan J. R. Induction of Decision Trees. // Machine Learning. –1986. – Vol.1, issue 1. – P. 81-106.
44. Lindén K. Multilingual modeling of cross-lingual spelling variants. // Information Retrieval. – 2005. – Vol.9, issue 3. – P. 295-310.
45. Левенштейн. В. И. Двоичные коды с исправлением выпадений, вставок и замещений символов. // Доклады Академий Наук СССР. – 1965.
46. Meng H. Generate phonetic cognates to handle name entities in English-Chinese cross-language spoken document retrieval. / H. Meng, W.-K. Lo, B. Chen, T. Tang // Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) workshop on Automatic Speech Recognition and Understanding. – Madonna di Campiglio, Italy, 2001. – P. 311-314.
47. Oh J.-H. Recognizing transliteration equivalents for enriching domain-specific thesauri / J.-H. Oh, K.-S. Choi // Proceedings of the 3rd International WordNet Conference. – 2006. – P. 231-237.
48. Karimi S. English to Persian transliteration. / S. Karimi, A. Turpin, F. Scholer // String Processing and Information Retrieval. Lecture Notes in Computer Science – Glasgow, UK, 2006. – vol. 4209. – P. 255-266.
49. Karimi S. Machine transliteration of proper names between English and Persian. // Ph.D. thesis. – RMIT University, Melbourne, Australia. 2008.
50. Sherif T., Kondrak G. Substring-based transliteration / T. Sherif, G. Kondrak // Proceedings of Conference of Association for Computational Linguistics (ACL) – Prague, Czech Republic, 2007. – P. 944–951.
51. Viterbi A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. // Institute of Electrical and Electronics Engineers (IEEE) Transactions on Information Theory. – 1967. – Vol.13, issue 2. – P. 260-269.
52. Li H. Semantic transliteration of personal names / H. Li, K. Sim, J.-S. Kuo, M. Dong // Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics. – Prague, Czech Republic, 2007. – P. 120-127.
53. Freitag D. A Sequence Alignment Model Based on the Averaged Perceptron / D. Freitag, S. Khadivi // Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. – Prague, 2007. – P. 238–247.

54. Collins M. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms // Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP) – Philadelphia, PA, USA, 2002.
55. Standard Arabic Technical Transliteration System [Электронный ресурс] // 2012. – Режим доступа: <http://en.wikipedia.org/wiki/SATTS>.
56. Zens R. The RWTH phrase-based statistical machine translation system / R. Zens, O. Bender, S. Hasan, S. Khadivi, E. Matusov, J. Xu, Y. Zhang, H. Ney. // Proceedings of the International Workshop on Spoken Language Translation (IWSLT). – Pittsburgh, PA, USA, 2005. – P. 155-162.
57. Bellare K. Loss-Sensitive Discriminative Training of Machine Transliteration Models / K. Bellare, K. Crammer, D. Freitag // Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT) Student Research Workshop and Doctoral Consortium. – Boulder, Colorado, 2009. – P 61-65.
58. Crammer K., Online Passive-Aggressive Algorithms / O. Dekel, J. Keshet, S. Shalev-Shwartz, Y. Singer // Journal of Machine Learning Research. – 2006. – Vol.7 – P. 551-585.
59. Crammer K., Singer Y. Ultraconservative Online Algorithms for Multiclass Problems / K. Crammer, Y. Singer // Journal of Machine Learning Research. – 2003. – Vol.3. P. 951-991.
60. McDonald R. Online Large-Margin Training of Dependency Parsers / R. McDonald, K. Crammer, F. Pereira // Proceedings of the 43rd Annual Meeting of the ACL. – Ann Arbor, 2005. – P. 91-98.
61. Cherry C. Discriminative Substring Decoding for Transliteration / C. Cherry, H. Suzuki // Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing. – Singapore, 2009. – P. 1066-1075.
62. Zelenko D. Discriminative methods for transliteration / D. Zelenko, C. Aone // In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing. – Sydney, Australia, 2006. – P. 612-617.
63. US Census [Электронный ресурс] // 2012. – Режим доступа: <http://www.census.gov/>.

64. Gale W. Good-Turing frequency estimation without tears / W. Gale, G. Sampson // Journal of Quantitative Linguistics. – 1995. – Vol.2. – P. 217-235.
65. Freund Y. Large margin classification using the perceptron algorithm / Y. Freund, R. Shapire // Machine Learning. – 1999. – Vol.37. – P. 277-296.
66. Association for Computational Linguistics [Электронный ресурс] // 2012. – Режим доступа: <http://www.acl-ijcnlp-2009.org/>.
67. Shishtla P. A Language-Independent Transliteration Schema Using Character Aligned Models At NEWS 2009 / P. Shishtla, S.V. Ganesh, S. Subramaniam, V. Varma // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 40–43.
68. Vardarajan B. ϵ -extension Hidden Markov Models and Weighted Transducers for Machine Transliteration / B. Vardarajan, D. Rao // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 120–123.
69. Hong G. A Hybrid Approach to English-Korean Name Transliteration / G. Hong, M.-J. Kim, D.-G. Lee, H.-C. Rim // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 108-111.
70. Koehn P. Moses: Open Source Toolkit for Statistical Machine Translation / P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, E. Herbst // Annual Meeting of the Association for Computational Linguistics (ACL), demonstration session. – Prague, Czech Republic, 2007.
71. Jia Y. A Noisy Channel Model for Grapheme-based Machine Transliteration / Y. Jia, D. Zhu, S. Yu // Proceedings of the 2009 Named Entities Workshop, Joint

- conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 88–91.
72. Jiang X. A Syllable-based Name Transliteration System / X. Jiang, D. Zhang // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 96–99.
73. Chen S.F. SRI toolkit: An empirical study of smoothing techniques for language modeling / S.F. Chen, J. Goodman // Technical Report TR-10-98. – Harvard University, 1998.
74. Yang D. Combining a Two-step Conditional Random Field Model and a Joint Source Channel Model for Machine Transliteration / D. Yang, P. Dixon, Y.-C. Pan, T. Oonishi, M. Nakamura, S. Furui // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 72–75.
75. Zelenko D. Combining MDL Transliteration Training with Discriminative Modeling // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 116–119.
76. Grunwald P. The Minimum Description Length principle // MIT Press. – 2007.
77. Jiampojarn S. DIRECTL: a Language-Independent Approach to Transliteration / S. Jiampojarn, A. Bhargava, Q. Dou, K. Dwyer, G. Kondrak // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint

- Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 28–31.
78. Das A. English to Hindi Machine Transliteration System at NEWS 2009 / A. Das, A. Ekbal, T. Mandal, S. Bandyopadhyay // Proceedings of the 2009 Named Entities Workshop, ACL-IJCNLP. – Suntec, Singapore, 2009. – P. 80–83.
 79. Haque R. English—Hindi Transliteration Using Context-Informed PB-SMT: the DCU System for NEWS 2009 / R. Haque, S. Dandapat, A.K. Srivastava, S.K. Naskar, A. Way // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 104–107.
 80. Chinnakotla M.K., Damani O.P. Experiences with English-Hindi, English-Tamil and English-Kannada Transliteration Tasks at NEWS 2009 / M.K. Chinnakotla, O.P. Damani // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 44–47.
 81. Aramaki E. Fast decoding and Easy Implementation: Transliteration as Sequential Labeling / E. Aramaki, T. Abekawwa // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 65–68.
 82. Khapra M.M. Improving transliteration accuracy using word-origin detection and lexicon lookup / M.M. Khapra, P. Bhattacharyya // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 84–87.

83. Noeman S. Language Independent Transliteration system using phrase based SMT approach on substrings // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 112–115.
84. Bose D. Learning Multi Character Alignment Rules and Classification of training data for Transliteration / D. Bose, S. Sarkar // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 61–64.
85. Oh J.-H. Machine Transliteration using Target-Language Grapheme and Phoneme: Multi-engine Transliteration Approach / J.-H. Oh, K. Uchimoto, K. Torisawa // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 36–39.
86. Zhou Y. Maximum N-gram HMM-based Name Transliteration: Experiment in NEWS 2009 on English-Chinese Corpus // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 128–131.
87. Rama T. Modeling Machine Transliteration as a Phrase Based Statistical Machine Translation Problem / T. Rama, K. Gali // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 124–127.

88. Song Y. Transliteration of Name Entity via Improved Statistical Translation on Character Sequences / Y. Song, C. Kit, X. Chen // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 57–60.
89. Freitag D. Name Transliteration with Bidirectional Perceptron Edit Models / D. Freitag, Z. Wang // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 132–135.
90. Jansche M. Named Entity Transcription with Pair n-Gram Models / M. Jansche, R. Sproat // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 32–35.
91. Cherry C. NEWS 2009 Machine Transliteration Shared Task System Description: Transliteration with Letter-to-Phoneme Technology / C. Cherry, H. Suzuki // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 69–71.
92. Jiampojarn S. Joint processing and discriminative training for letter-to-phoneme conversion / S. Jiampojarn, C. Cherry, G. Kondrak // Proceedings of the 46th Annual meeting of the Association for Computational Linguistics (ACL). – Columbus, Ohio, 2008. – P. 905-913.
93. Kwong O.Y. Phonological Context Approximation and Homophone Treatment for NEWS 2009 English-Chinese Transliteration Shared Task // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the

- Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 76–79.
94. Reddy S. Substring-based Transliteration with Conditional Random Fields / S. Reddy, S. Waxmonsky // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 92–95.
95. Nabende P. Transliteration System using pair HMM with weighted FSTs // Proceedings of the 2009 Named Entities Workshop, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP). – Suntec, Singapore, 2009. – P. 100–103.
96. Oh J.-H. A comparison of different machine transliteration models / J.-H. Oh, K.-S. Choi, H. Isahara // Journal of Artificial Intelligence Research (JAIR). – 2006. – Vol.27. – P. 119-151.
97. Wallach H.M. Conditional Random Fields: An Introduction // Technical Report MS-CIS-04-21. – University of Pennsylvania CIS, 2004.
98. Klinger R. Classical Probabilistic Models and Conditional Random Fields / R. Klinger, K. Tomanek // Algorithm Engineering Report TR07-2-013. – 2007.
99. Sutton C. An Introduction to Conditional Random Fields for Relational Learning / C. Sutton, A. McCallum // Introduction to Statistical Relational Learning. – MIT Press. – 2006.
100. Jiampojamarn S. Applying many-to-many alignments and Hidden Markov Models to letter-to-phoneme conversion / S. Jiampojamarn, G. Kondrak, T. Sherif. // Proceedings of the Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT). – Rochester, NY, USA. – 2007. – P. 372-379.
101. Транскриптор [Электронный ресурс] // 2011. – Режим доступа: <http://nano.yandex.ru/project/anthroponym/>.

102. Zens R. Improvements in phrase-based statistical machine translation / R. Zens, H. Ney // Proceedings of the Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT). – Boston, USA. – 2004. – P. 257-264.
103. Sproat R., Named entity transliteration with comparable corpora / R. Sproat, T. Tao, C. Zhai // Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL). – Sydney, Australia – 2006.
104. Wu J. Learning to find English to Chinese transliterations on the web. / J. Wu, J. Chang // Proceedings of Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning (EMNLP/CoNLL). – Prague, Czech Republic. – 2007.
105. Goldberg Y. Identification of transliterated foreign words in Hebrew script / Y. Goldberg, M. Elhadad // Proceedings. of 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing). – Haifa, Israel. – 2008.
106. Tao T., Unsupervised named entity transliteration using temporal and phonetic correlation / T. Tao, S. Yoon, A. Fister, R. Sproat, C. Zhai // Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP). – Sydney, Australia. – 2006.
107. Ravi S., Knight K. Learning Phoneme Mappings for Transliteration without Parallel Data. // Human Language Technology Conference archive Proceedings of Human Language Technologies: the 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics. – 2009.
108. Pervouchine V. Improving Name Origin Recognition with Context Features and Unlabelled Data / V. Pervouchine, M. Zhang, V. Liu, H. Li // Proceedings of COLING. – Beijing, 2010. – P. 972-978.
109. Qu Y., Finding ideographic representations of Japanese names written in Latin script via language identification and corpus validation. / Y. Qu, G. Grefenstette // Proceedings of 42nd Annual Meeting of Association for Computational Linguistics (ACL). – Barcelona, Spain 2004 – P.183-190.

110. Клышинский Э.С. Методика определения языка происхождения имени собственного // Сб. трудов 14 научно-практического семинара «Новые информационные технологии». – М., 2011. – С. 107-112.
111. Klementiev A. Named entity transliteration and discovery from multilingual comparable corpora / A. Klementiev, D. Roth // Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT). – New York City, USA, 2006. – P. 82-88.
112. Brill E. An improved error model for noisy channel spelling correction / E. Brill, R.C. Moore // Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL). – Morristown, NJ. 2000. –P.286-293.
113. Bilac S. Extracting transliteration pairs from comparable corpora / S. Bilac, H. Tanaka // Proceedings of the Annual Meeting of the Natural Language Processing Society. – Japan, 2005.
114. Sherif T. Bootstrapping a stochastic transducer for Arabic-English transliteration extraction / T. Sherif, G. Kondrak // Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL). – Prague, Czech Republic, 2007. – P. 864–871.
115. Hermjakob U. Name translation in statistical machine translation - learning when to transliterate / U. Hermjakob, K. Knight, H. Daumé // Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL). – Columbus, Ohio, 2008. – P. 389-397.
116. Google n-gram viewer [Электронный ресурс] // 2012. – Режим доступа: <http://books.google.com/ngrams/>.
117. Кодзасов С. В. Общая фонетика, учебник / С.В. Кодзасов, О.Ф. Кривнова. – М.:РГГУ. – 2001.
118. Логачева В.К. Автоматическое порождение правил транскрипции фамильно-именных групп // Сб. трудов 13 научно-практического семинара «Новые информационные технологии», М., 2010, С. 117-121.
119. Клышинский Э.С. Автоматическое порождение правил транскрипции фамильно-именных групп / Э.С. Клышинский, В.К. Логачева // Сб. трудов 12 национальной конференции по искусственному интеллекту КИИ-2010 (Тверь, 20-24 сентября 2010 г), том 1, М.: Физматлит, 2010. С. 274-282.

120. Логачева В.К. Non-stochastic learning of cross-language transliteration rules from a small dataset / В.К. Логачева, Э.С. Клышинский // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (Бекасово, 25-29 мая 2011 г.). Вып. 10 (17).- М.: Изд-во РГГУ, 2011. С. 448-457.
121. Логачева В.К. Метод порождения правил межъязыковой транскрипции // НТИ, серия 2, № 9. М.: 2011. – С. 26-33.
122. Yu S. Applications of Finite-State Transducers in Natural Language Processing. In Implementation and Application of Automata, (eds.) / S. Yu, A. Paun // Lecture Notes in Computer Science. Springer Verlag, Heidelberg, 2001. – Vol.2088. – P. 34-46.
123. Логачева В.К. Автоматическая генерация правил транскрипции и машинная транскрипция имен собственных с использованием конечного автомата. [Электронный ресурс] / В.К. Логачева, Э.С. Клышинский, В.А. Галактионов // Препринты ИПМ им. М.В.Келдыша. – 2012. – № 14. 24 с. Режим доступа: <http://library.keldysh.ru/preprint.asp?id=2012-14>.
124. Karttunen L. Finite-State Lexicon Compiler. [Электронный ресурс] / L. Karttunen // Technical Report. ISTL-NLTT-1993-04-02. – Xerox Palo Alto Research Center. Palo Alto, California, 1993. Режим доступа: <http://www.cis.upenn.edu/~cis639/docs/lexc.html>
125. Axelson E. Helsinki Finite-State Transducer Technology (HFST) [Электронный ресурс] / E. Axelson // 2003. – Режим доступа: <https://kitwiki.csc.fi/twiki/bin/view/KitWiki/HfstHome>.
126. Ахо А., Ульман Д., Теория синтаксического анализа, перевода и компиляции. –М.: «Мир». –1978.
127. Логачева В.К. Метод генерации конечного автомата для задач машинной транскрипции / В.К. Логачева, Э.С. Клышинский // НТИ, серия 2, №1. М.: 2012. – С. 22-29.
128. Карпов Ю.Г. Теория автоматов. – СПб.: Питер. – 2003.
129. C++ Builder XE2 [Электронный ресурс]. – 2011. – Режим доступа: <http://www.embarcadero.com/products/cbuilder>.