МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ

ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

**институт информационных технологий и технологического образования**

**кафедра информационных технологий и электронного обучения**

Основная профессиональная образовательная программа

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль) «Технологии разработки программного обеспечения»

форма обучения – очная

**Курсовая работа**

по дисциплине «Технологии компьютерного моделирования»

Компьютерное и математическое моделирование в лингвистике

Обучающегося 2 курса

Косыгина Кирилла Сергеевича

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель:

к.п.н, доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гончарова С. В.

«\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Санкт-Петербург

2020

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc41826570)

[Глава 1. КОМПЬЮТЕРНАЯ ЛИНГВИСТИКА 4](#_Toc41826571)

[Глава 2. МАТЕМАТИЧЕСКАЯ ТЕОРИЯ ФОРМАЛЬНЫХ ЯЗЫКОВ 6](#_Toc41826572)

[2.1 Регулярные выражения 6](#_Toc41826573)

[2.2 Конечные автоматы 7](#_Toc41826574)

[2.3 Использование теории формальных языков при исправлении опечаток 8](#_Toc41826575)

[2.3.1 Расстояние Левенштейна 8](#_Toc41826576)

[2.3.2 Алгоритм исправления опечаток 9](#_Toc41826577)

[Глава 3. ВЫЧИСЛИТЕЛЬНАЯ МОРФОЛОГИЯ 11](#_Toc41826578)

[3.1 Индексы словообразования 11](#_Toc41826579)

[3.2 Конечные преобразователи 12](#_Toc41826580)

[Глава 4 ОЦЕНКА МАШИННОГО ПЕРЕВОДА 14](#_Toc41826581)

[Глава 5 КОРПУСКАЯ ЛИНГВИСТИКА 16](#_Toc41826582)

[5.1 Меры разброса и частотности 16](#_Toc41826583)

[5.1.1 Уменьшенная частотность (Reduced Frequency) 16](#_Toc41826584)

[5.1.2 Средняя уменьшенная частотность (Average Reduced Frequency) 17](#_Toc41826585)

[5.2 Распределение Ципфа 17](#_Toc41826586)

[5.2 Меры разброса 18](#_Toc41826587)

[Глава 6 СРАВНИТЕЛЬНО-ИСТОРИЧЕСКОЕ ЯЗЫКОЗНАНИЕ 19](#_Toc41826588)

[6.1 Список Сводеша. Формула Сводеша 19](#_Toc41826589)

[6.1.1 Формула для времени расхождения языков 19](#_Toc41826590)

[6.2 Восстановление филогенетических деревьев 20](#_Toc41826591)

[6.2.1 Формула расстояния 20](#_Toc41826592)

[6.2.2 Построение филогенетического дерева 21](#_Toc41826593)

[6.2.3 Алгоритм иерархической кластеризации 21](#_Toc41826594)

[ЗАКЛЮЧЕНИЕ 23](#_Toc41826595)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 25](#_Toc41826596)

[ПРИЛОЖЕНИЯ 27](#_Toc41826597)

ВВЕДЕНИЕ

Лингвистика — это не только наука о том, как грамотно писать и говорить на родном и иностранных языках. Лингвистов гораздо больше интересует, как устроена языковая способность человека, как развиваются и в каких отношениях между собой находятся языки, а также какими свойствами обладают тексты на естественных языках. Для изучения этих вопросов лингвистам часто требуется привлекать математические методы. Идеи и методы математической логики позволяют сделать описание языковых явлений более формальным, и значит, более строгим, а применение количественных методов к большим объёмам лингвистических данных позволяет обнаруживать незаметные на первый взгляд связи и закономерности, а также обрабатывать тексты на естественном языке: например, исправлять опечатки или переводить тексты с одного языка на другой. Помимо очевидных практических применений, лингвистика, начиная с XX в. оказывает большое теоретическое влияние на другие науки, например, принципом минимальной дифференциации.

Объектом исследования данной работы является математическое и компьютерное моделирование.

Предмет исследования - математическое и компьютерное моделирование в лингвистике.

Целью данной работы является резюмирование математических и компьютерных методов моделирования в лингвистике.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучить основы различных направлений лингвистики, акцентируя внимание на математических моделях, используемых в этих направлениях.

2. На основе полученных данных исследовать и предложить математические и компьютерные модели, используемые в каждом разделе.

3. Предложить алгоритмы для реализации этих моделей.

Выделив объект и предмет исследования, можно выдвинуть следующую гипотезу: в лингвистике активно используются математическое и компьютерное моделирование, отражающее свойства языка и текстов.

При проведении исследования используются следующие методы: анализ, синтез, формализация, дедуктивный и индуктивный методы, наблюдение, сравнение, абстрагирование. Теоретическая значимость проводимого исследования заключается в теоретическом обобщении знаний в области лингвистики и технологий компьютерного и математического моделирования. Разработанные математические модели и программы можно будет использовать в практических целях: для моделирования лингвистических процессов и структур, и таким образом, последующих исследований в данной области.

# Глава 1. КОМПЬЮТЕРНАЯ ЛИНГВИСТИКА

Компьютерная лингвистика — это направление в прикладной лингвистике, ориентированное на использование компьютерных инструментов.

Лингвистика обычно работает с большими объемами данных (как синхронные, так и диахронические направления), поэтому с появлением компьютеров все математические и компьютерные модели сконцентрировались в направлении «компьютерная лингвистика» [4]. Набор задач, решаемых в данной области, с момента появления значительно расширился. Решаются как теоретические задачи, так и практические, результаты первых воздействуют на вторые и наоборот. Примерами таких практических задач являются: распознавание текстов; распознавание и синтез речи; машинный перевод, проверка орфографии, грамматики и стиля; информационный поиск; извлечение информации и мнений и т.д.

Решение данных задач осложняют сами свойствами языка, а именно:

1. неоднозначность: одно и то же выражение, форма, конструкция может означать разное, следовательно, разрешение может требовать знаний о мире, контексте и т.п.

2. несимметричность: разные языки часто имеют принципиально различающиеся способы кодирования некоторого смысла;

3. избыточность (вариативность): в языке, как правило, имеется множество способов выразить некоторый смысл;

4. конвенциональность: часто правильным и единственным возможным способом выражения некоторого смысла является лишь один из теоретически возможных;

5. эллиптичность: в языке действует множество умолчаний, понимание требует восстановления опущенной информации;

6. непрозрачность: язык активно использует сложные средства референции.

В истории компьютерной лингвистики четко выделяются два этапа с разными методами и подходами к изучаемому предмету [12].

1. 1950-е — 1990-е годы (взгляд лингвиста). Компьютерная лингвистика рассматривается как формальная, полная и логически непротиворечивая теория языка, которая может использоваться для компьютерного анализа текстов. Это особым образом устроенная теория языка — формализованная, полная и логически непротиворечивая до такой степени, что описания языков, сделанные в соответствии с этой теорией, могут быть использованы для компьютерного анализа языка. Например, для определения правильности предложения, построения его языковой структуры (как математического объекта), определения смыслового тождества высказываний и т.п.

2. 1990-е – 2010-e годы. Компьютерная лингвистика рассматривается в качестве технологии и методологии для решения конкретных практических задач без претензий на общую теорию языка (взгляд инженера и пользователя).

Основное различие данных этапов в контексте математического моделирования заключается в том, что первый этап ориентирован на детерминированные модели и структурное моделирование, а второй этап ориентирован на стохастические модели и функциональное моделирование.

С недавнего времени (2010-е годы – настоящее время) выделяется третий этап, амальгамирующий два предыдущих подхода. Активно используются стохастические модели, например, машинное обучение, но снова происходят попытки воссоздать структуру языка, а не решить практическую задачу с плохо интерпретируемыми результатами [10].

# Глава 2. МАТЕМАТИЧЕСКАЯ ТЕОРИЯ ФОРМАЛЬНЫХ ЯЗЫКОВ

При описании фонологических и морфологических процессов часто используются контекстные замены вида (1). Здесь U идентифицирует единицу, которая подвергается действию правила, V указывает какую форму эта единица должна принять, X и Y означают левый и правый контексты для U; X и Y могут опускаться [5].

|  |  |
| --- | --- |
|  | (1) |

Требуется подобрать математическую модель для их описания таких переходов. То есть требуется задать условия вида:

- последний прочитанный символ — a;

- символ b находится между символами a и c;

- в слове содержится ровно 2 символа d, и т.д.

Тогда пусть Σ — алфавит (произвольное конечное множество). Элементы алфавита называются буквами, но не тождественны буквам алфавита естественного языка. Множества букв могут содержать любые фонологические, грамматические и тому подобные единицы. Например,

- - для букв естественного алфавита;

- – для суффиксов. При этом следует обратить внимание, что символ означает не пустое множество, а пустой элемент.

Слово — конечная последовательность элементов .

- множество всех слов.

— пустое слово (слово длины 0).

Язык — произвольное подмножество .

Функция — множество упорядоченных пар , таких, что пары существуют для всех элементов множества X, и, если первые элементы пар совпадают, то и вторые их элементы тоже. То есть это подмножества вида .

## 2.1 Регулярные выражения

Пусть зафиксирован конечный алфавит , тогда:

- базовые регулярные выражения — это элементы алфавита;

- существуют константы 0 (пустой язык) и 1 (язык, содержащий только пустое слово );

- над алфавитом возможны бинарные операции: | (объединение) и (конкатенации или приписывание): ;

- над алфавитами возможна унарная операция \* (итерация): состоит из слов вида , где ;

- если α — регулярное выражение, то L(α) – задаваемый им язык. Например .

Например, пусть (согласный, гласный, слогораздел), тогда слово с 2 гласными, разделенными хотя бы одним согласным задается следующим регулярным выражением: C\*VC+VC\*; корректное разбиение на слоги задается следующим регулярным выражением: 1|(C\*VC\*-)\*(C\*VC\*) и т.п.

Большинство языков программирования имеют встроенные библиотеки для работы с регулярными выражениями, а синтаксис языка регулярных выражений PCRE в немалой степени повторяет математический аппарат регулярных выражений. В компьютерной и других направлениях лингвистики, например, фонологии или компаративистике, регулярные выражения используются для описания переходных процессов.

## 2.2 Конечные автоматы

Конечный автомат — это помеченный граф (2).

|  |  |  |
| --- | --- | --- |
|  | | (2) |
| где | Q – конечное множество состояний (вершин графа); | |
|  | ∆ - конечное множество переходов вида ; | |
|  | – стартовое состояние; | |
|  | – завершающие состояния. | |

Классы автоматных и регулярных языков совпадают [11]. Пример конечного автомата для слогоделения ровно с одним открытым слогом представлен на рисунке 1.

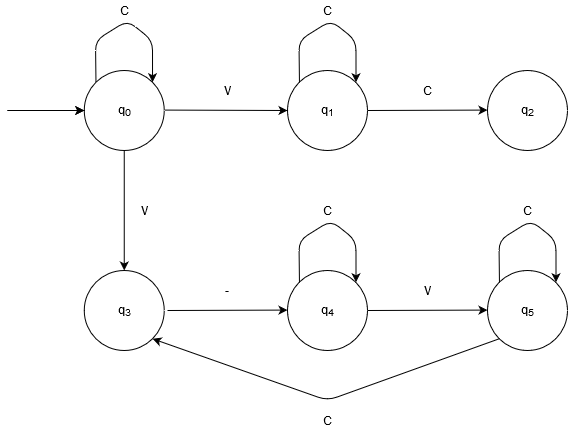


Рисунок 1 - Конечный автомат для слогоделения ровно с одним открытым слогом

Метка пути — это конкатенация всех меток на входящих в него ребрах. Язык L(M) – метки путей из начального состояния в завершающие. Язык называется автоматным, если задается некоторым конечным автоматом. Любой автоматный язык распознается автоматом с однобуквенными переходами. Автомат с однобуквенными переходами — детерминированный, если ни из какого состояния нет двух переходов по одной и той же букве. Любой автоматный язык распознается детерминированным конечным автоматом.

## 2.3 Использование теории формальных языков при исправлении опечаток

Опечатки и орфографические ошибки содержаться в любом достаточно длинном тексте. Исправление опечаток — это выявление ошибок правописания и поиск возможных вариантов их исправления. Опечатки можно трактовать шире: как эволюционные изменения в словах языка или различия между родственными языками: аппарат для выявления и исправления опечаток можно использовать в сравнительно-историческом языкознании для воссоздания языков [13].

Опечатки возникают как следствия: орфографических ошибок, типографских ошибок, когнитивных ошибок (смешения понятий), ошибок при записи речи «на слух», транслитерационных ошибок, региональной или стилистической вариативности (“colour”, “color”). Подавляющее большинство ошибок локально, они затрагивают один-два символа, но не исключены и более масштабные ошибки.

### 2.3.1 Расстояние Левенштейна

Для исправления опечаток можно искать «близкие» слова в словаре. Понятие «близкие» требует оценки, то есть необходимо задать функцию расстояния на множестве слов. Такая функция определяется через расстояние Левенштейна [9].

Расстояние Левенштейна между словами u и v – минимальное число замен, вставок и удалений, необходимых, чтобы получить v из u. При этом можно присвоить разным типам изменений разные веса и добавить перестановку соседних символов.

Используются следующие обозначения:

– слово, – длина слова.

– i-ый символ слова, – подслово с i-ой по j-ую позицию (не включая j).

– префикс по j-ую позицию (не включая j).

– суффикс с i-ой позицию (включая i).

Основная идея алгоритма — вычислять значения матрицы рекурсивно через значения для меньших i, j. Если , то ответом будет . На каждом шаге решается задача с учетом предыдущих решений, то есть это задача динамического программирования. Алгоритмическая сложность - .

Рекурсивная формула расстояния Левенштейна (3):

|  |  |  |
| --- | --- | --- |
|  |  | (3) |
|  |  |  |
|  |  |
|  | |  |

Пример реализации алгоритма для расчета расстояния Левенштейна на Python3 представлен в приложении 1.

Умея оценивать похожесть слов, можно реализовать непосредственно алгоритм для исправления опечаток, например, для каждого слова в словаре рассчитать расстояние Левенштейна от текущего слова. Очевидно, что такой алгоритм имеет большую алгоритмическую сложность и его использование не оправдано. Нужно сократить перечень слов для которых рассчитывается расстояние Левенштейна: породить все слова, расстояние до которых меньше порога d и найти их в словаре. Для порождения слов необходимо прибегнуть к конечным автоматам и преобразователям. Словари обычно хранятся в форме ациклического автомата с элементами двоичного дерева, то есть будет возможно итерировать одновременно по двум автоматам: по автомату словаря и по порождающему автомату. Это позволит уменьшить алгоритмическую сложность как по количеству операций, так и по затратам памяти.

### 2.3.2 Алгоритм исправления опечаток

Алгоритм поиска слов, на которые можно заменить гипотетическую опечатку состоит в следующем [2]:

1. Построить взвешенный автомат (преобразователь) для данного слова w и порога расстояния d.
2. Вернуть список слов из языка , где – автомат для словаря вместе с весами слов в автомате .
3. Выбрать наиболее удачные слова наименьшего веса.

Для нахождения наиболее удачных слов необходимо использовать стохастические модели.

На ребрах заданы операции вида , соответствующие опечаткам вида . У каждой такой операции есть вероятность, которую можно оценить: . То есть маленьким вероятностям соответствуют большие веса.

Пусть – путь наименьшего веса в преобразователе M, исправляющий слово на, а – стоимость этого пути (4).

|  |  |
| --- | --- |
|  | (4) |
|  |  |
|  |  |

То есть вес преобразования приблизительно равен отрицательному логарифму вероятности получить v из u.

В качестве исправления c для слова u оправдано выбирать наиболее вероятное слово: . Для перехода от к нужно воспользоваться теоремой Байеса (5).

|  |  |
| --- | --- |
|  | (5) |

Вероятности замен можно настроить по корпусу опечаток или с помощью эвристик: расположения слов на клавиатуре, фонетической близости, графической близости и т.д.

Тогда финальный алгоритм поиска слов-кандидатов выглядит следующим образом:

1. Взять словарь с проставленными вероятностями слов , преобразовать во взвешенный автомат с выходными весами
2. Построить взвешенный автомат для текущего слова w и порога расстояния d.
3. Вернуть список слов из языка вместе с их весами.
4. Вернуть слово или несколько слов с наименьшими весами.

Пример программной реализации примитивного корректора опечаток на Python3 представлен в приложении 2.

# Глава 3. ВЫЧИСЛИТЕЛЬНАЯ МОРФОЛОГИЯ

Морфологический анализ определяет статус элементов в системе языка и вносит больший вклад в лемматизацию. Морфология лежит в основе треугольника перевода (схемы в виде треугольника, описывающей переходные этапы при анализе и синтезе текста) и описывает слова и их формы. Без морфологического анализа дальнейшие этапы дадут если не бессмысленные, то очень сложные для интерпретации результаты [16].

## 3.1 Индексы словообразования

Чем больше в языке грамматических значений, тем больше у каждой лексемы словоформ. Количество грамматических значений коррелирует с количеством морфем в слове. Эти параметры отражаются индексами словообразования, например, индексом синтетичности.

Индекс синтетичности — это среднее количество морфем в слове (6).

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | (6) |
| где | M - число морфем; | |  |
|  | W - число слов. | |  |

Индекс синтетичности служит для описания сложности морфологического состава слов в языке. Значение индекса синтетичности с недавнего времени наиболее показательно рассчитывается с помощью корпуса языка. Разные языки выражают разное количество морфологических значений, например, для русского языка индекс синтетичности колеблется от 2,33 до 2,45, для западногренландского языка — 3,72, а для вьетнамского языка — 1,06. Аналогичным образом, как среднее значение, определяются другие индексы, например, «индекс словосложения», характеризующий использование сложения при словообразовании или «индекс деривации», характеризующий использование аффиксов и т.д.

Индексы словообразования изменяются, во времени и пространстве. Аддитивная модель морфологии иногда требует применения фонологических правил. Существует неконкатенативная морфология (морфология с изменение морфем) [15].

Для представления слов нужно решить две задачи: обеспечить экономное описание и описать реальность. Возможны два механизма:

1. Computation (вычисление) — грамматика;

2. Storage (хранение) — лексикон.

Человек использует двойной механизм: часть слов берется из памяти, часть строится по правилам, для моделирования можно использоваться либо один из механизмов, либо комбинировать оба.

## 3.2 Конечные преобразователи

Определение конечного преобразователе схоже с определением конечного автомата (7), различие заключается в том, что преобразователь не просто переходит в другое состояние, но также изменяет слово.

|  |  |  |
| --- | --- | --- |
|  | M = ⟨ Q, Σ, Г, ∆, q0, F ⟩, | (7) |
| где | Q – конечное множество состояний (вершин графа); |  |
|  | ∆ - конечное множество переходов вида ⟨ q1, u ⟩ → ⟨ q2, v ⟩; q1; q1 ∈ Q, v ∈ Г\*; |  |
|  | – стартовое состояние; |  |
|  | F ⊆ Q – завершающие состояния. |  |

В дальнейшем будем считать, что все переходы имеют вид ⟨ q1, a ⟩ → ⟨ q2, b ⟩, a ∈ Σ ∪ {ε}, v ∈ Г ∪ {ε}. То есть на каждом шаге читается и пишется не более одного символа.

Конечный преобразователь принимает пары слов , которые являются метками путей из начального состояния в завершающее. Конечные преобразователи замкнуты относительно операций конкатенации, объединения, композиции, обращения.

Конечные преобразователи широко используются в ранее упомянутых контекстных заменах (1), то есть заменить X на Y между U и V. Также преобразователи используются при лемматизации и генерации словоформ, образовании прошедших времен глагола, множественного числа и т.п., например, следующие преобразователь удаляет в слове повторяющиеся буквы: (см. рисунок 2).

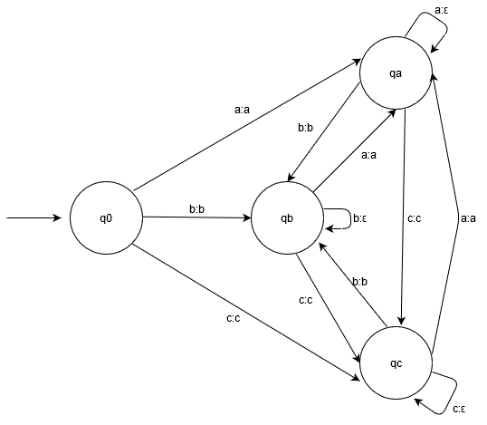


Рисунок 2 – Конечный преобразователь, удаляющий повторяющиеся буквы

Для моделирования подобных структур отлично подходит программа Foma [1], строящая конечные автоматы или язык программирования Prolog. Преимуществом Foma является наличие дополнения Lexicon, которое содержит предикаты для лексических функций. Лексические функции – это зависимости, которые связывают слово с его лексическими коррелятами [8]. Пример конечного преобразователя для множественного числа в английском языке представлен в приложении 3.

# Глава 4 ОЦЕНКА МАШИННОГО ПЕРЕВОДА

Разработка машинного перевода началась в 1950-е годы. При разработке таких систем ученые столкнулись с множеством проблем, например, лексическими функциями. Не существует идеального правильного перевода, поэтому с помощью асессоров, владеющих исходным языком и языком перевода, оцениваются:

- Качество перевода;

- Адекватность оригиналу.

Асессоры по-разному понимают шкалу оценки, поэтому необходимо нормирование, также при разработке и тестировании алгоритмов перевода нужно каждый раз оценивать перевод. Оценки производятся столь часто, что становится невозможным привлекать асессоров. Автоматическая оценка машинного перевода ориентируется на образцовые переводы, сделанные вручную. Правильно поставленная оценка машинного перевода должно коррелировать с ручной оценкой по качеству языка и адекватности [14]. Для этого используются параметры:

- Точность – значение, описывающее какая доля слов машинного перевода правильна;

- Полнота – значение, описывающее какая доля слов образцового перевода передана.

Используются следующие обозначения:

– число слов, совпадающих в машинном и в образцовом переводе;

– длина машинного перевода;

– длина образцового перевода.

Тогда определяют параметры «точность» (8) и «полнота» (9).

|  |  |
| --- | --- |
|  | (8) |
|  | (9) |

Эти меры легко завысить, поэтому также используется F-мера (10). F-мера отражает баланс точности и полноты (среднее гармоническое).

|  |  |
| --- | --- |
|  | (10) |

При и ,

Между машинным и образцовым переводом можно измерять расстояние Левенштейна, как в случае опечаток. Word Error Rate (WER) – это взвешенное расстояние Левенштейна между машинным и образцовым переводом. Чем выше значение данного параметра, тем хуже качество перевода.

Можно учитывать не только отдельные слова, но и сочетания n-слов (n-граммы) [3]. Соответствующий параметр называется BLEU (Bilingual Evaluation Understudy) (11).

|  |  |  |
| --- | --- | --- |
|  |  | (11) |
| где | – точность на уровне n-грамм. |

Преимуществами BLUE являются: возможность сравнить перевод с несколькими образцовыми переводами и хорошая корреляция с ручной оценкой качества перевода и адекватности. При этом BLEU не учитывает различия в значимости слов, не учитывает грамматичность, сами значения плохо интерпретируемы и не сравнимы при разных тестах, а постредактирование перевода улучшает ручную оценку, но не BLUE.

# Глава 5 КОРПУСКАЯ ЛИНГВИСТИКА

Корпусная лингвистика – раздел языкознания, занимающийся разработкой, созданием и использованием текстовых корпусов. Корпус текстов – это большой, представленный в машиночитаемом формате, унифицированный, структурированный, размеченный массив данных, предназначенный для решения лингвистических задач. Для корпусов характерно наличие металингвистической информации и репрезентативность. Корпуса – основной инструмент для изучения частотности различных единиц в языке. Меры частотности и разброса вводятся для ситуаций, когда требуется сравнить частотность некоторых единиц в разных корпусах или частях корпуса [6].

## 5.1 Меры разброса и частотности

При работе с корпусами используются несколько мер разброса и частотности.

Абсолютная частота – количество вхождений словоформы в корпус.

Относительная частота – отношение количества вхождений словоформы в корпус к общему количеству словоформ в корпусе.

R (range) – количество сегментов корпуса, поделенного на n частей (n=100 Ляшевская, Шаров), в которых встретилось слово. Это одновременно мера разброса и мера частотности: показывает одновременно на сколько слово обильно представлено в корпусе и насколько оно равномерно распределено. У данной меры есть очевидный недостаток: частотность, измеренная таким образом лежит в очень небольшом множестве значений, потому что многомилионные корпуса делят на небольшое n. Для решения этой проблемы можно делить корпус на нефиксированное количество сегментов.

### 5.1.1 Уменьшенная частотность (Reduced Frequency)

Пусть корпус – это закольцованный список слов длины N (пронумерованный от 1 до N), то есть после N-го слова опять следует первое, второе и т.д.; f – число вхождений в корпус слова w; – позиция i-го вхождения w; – расстояние от (i – 1)-го вхождения до i-го вхождения w, здесь расстояние от 0-го до 1-го вхождения – это расстояние от последнего вхождения до первого (корпус закольцован).

Разделим корпус на f сегментов длины , тогда Reduced Frequency (RF) – количество сегментов, содержащих слово w хотя бы раз. Отсюда максимальная частотность равняется f, минимальная частотность равняется . Но проблема остается нерешенной, например, пусть слово встретилось два раза при количестве сегментов также равном двум, и слово попало в оба этих сегмента, тогда информация не передает всю действительность, потому что слово может быть не распространено равномерно (одно следует за другим). Для решения данной проблемы вводиться усредненная сокращенная частотность.

### 5.1.2 Средняя уменьшенная частотность (Average Reduced Frequency)

Основная идея следующая: разбиение на сегменты не обязано начинаться с первого слова: можно взять усредненную уменьшенную частотность по всем разбиениям на сегменты (корпус по-прежнему закольцован), сдвигая начало разбиения. Имеет смысл вычислять RF только для разбиений с 1-го по v-е. Тогда средняя уменьшенная частотность – это сумма уменьшенных частот, деленная на количество разбиений (12).

|  |  |
| --- | --- |
|  | (12) |

Очевидно, что перебрать все разбиения корпуса для всех слов таким вычислительным методом – неэффективно из-за большой алгоритмической сложности. В расчёте ARF участвует каждый сегмент от 1-го до N-го. Следует рассмотреть сегменты, которые начинаются сразу после первого вхождения, по последний сегмент включительно. Не больше v сегментов начиная с (-го по -й содержат слово w. Тогда при переходе от v возможных разбиений к количеству сегментов с длинной v, начиная с каждой позиции корпуса в которой это слово встретилось, получиться более оптимальная формула средней уменьшенной частоты (13).

|  |  |
| --- | --- |
|  | (13) |

В каждой такой паре количество сегментов будет равно минимуму из двух возможных значений: расстояния между соседними вхождениями и длины сегмента.

В большинстве случаев частотность выражается в процентах. Частотность позволяет определить равномерность распределения слова в корпусе и даже в конкретных текстах ориентирована на язык, а не на контекст. Пометки в словарях «употребительное», «малоупотребительное» и т.д. основаны на данном параметре.

## 5.2 Распределение Ципфа

Частотность слова обратно пропорциональна его рангу в частотном списке (13). Эта зависимость называется законом Ципфа.

|  |  |
| --- | --- |
|  | (13) |

Бенуа Мандельброт модифицировал закон Ципфа (14).

|  |  |
| --- | --- |
|  | (14) |

Эта модификация обязана тому, что кривая зависимости частоты появления каждого слова от его ранга убывает медленнее.

Когда тексты создаются не естественным путем, в них содержится много ключевых фраз, которые не вписываются в рамки закона Ципфа.

## 5.2 Меры разброса

Общепринятая статистическая мера разброса – стандартное отклонение (15).

|  |  |  |
| --- | --- | --- |
|  | , | (15) |
| где | – среднее арифметическое частот в n сегментах. |  |

Размер зависит от .

С помощью коэффициента Жуйяна (16), и меры разброса – количества сегментов корпуса, в которых встретилось слово, из n = 100, можно сделать выводы по соотношению слова и контекста.

|  |  |
| --- | --- |
|  | (16) |

1. высокое R, высокое D – слово соответствует контексту;

2. низкое R, высокое D – низкочастотное слово;

3. низкое R, низкое D – тематически окрашенное слово;

4. высокое R, низкое D – слово есть во всех сегментах, но с очень разными частотами, то

есть общая частотность завышена за счет выбросов.

# Глава 6 СРАВНИТЕЛЬНО-ИСТОРИЧЕСКОЕ ЯЗЫКОЗНАНИЕ

Сравнительно-историческое языкознание или компаративистика занимается установлением родства языков, восстановлением праязыковых форм, изучает эволюцию языка и праязыка, датирует расхождения родственных языков.

## 6.1 Список Сводеша. Формула Сводеша

Список Сводеша – это предложенный американским лингвистом Моррисом Сводешом инструмент для оценки степени родства между различными языками по признаку схожести наиболее устойчивого словаря. Представляет собой перечень базовых лексем языка, упорядоченный по убыванию их «базовости». Минимальный набор важнейшей лексики содержится в 100-словном списке Сводеша. Список Сводеша внес огромный вклад в историческое языкознание, но он имеет несколько недостатков: стабильность различных слов списка неодинакова; варианты для списка неоднозначны (walk: ходить, идти); слова одинакового происхождения могут обозначать разные понятия.

### 6.1.1 Формула для времени расхождения языков

По аналогии с радиоуглеродным распадом в лингвистике оценивается «лексический полураспад» (17). Данный метод основан на гипотезе, что скорость изменения базового словаря языка остается примерно одинаковой.

|  |  |  |
| --- | --- | --- |
|  |  | (17) |
| где | C – доля общей лексики; |  |
|  | – константа сохранения (0,86 – Swadesh, 0,805 – Less). |  |

Правильнее определять долю общей лексики как долю понятий, обозначенных словами из одного класса когнации. Класс когнации – это множество слов, имеющих одно и то же происхождение, хотя возможно, и разное значение.

Метод не лишен недостатков: доля сохраняемых когнатов зависит от исторических, социокультурных и прочих факторов, слова из различных частей списка Сводеша выпадают с разной частотой, существуют заимствования и т.д. Сергей Старостин предложил следующие правки:

1. Систематически заимствуемые слова должны быть исключены из вычислений;
2. Скорость изменения не постоянна (как предполагал Сводеш) и зависит от периода времени, в течение которого слово представлено в языке, то есть вероятность замены лексемы X лексемой Y возрастает со временем;
3. Отдельные единицы в 100-словном списке имеют другой уровень стабильности.

На основании предложенных правок была получена модифицированная формула Сводеша (18).

|  |  |
| --- | --- |
|  | (18) |

В данной формуле Lc отражает постепенное замедление процесса замещения благодаря различным индивидуальным скоростям (наименее стабильные элементы замещаются первыми и быстрее всего). Обратная тенденция представлена квадратным корнем: ускорение замещения по мере того, как единицы в первоначальном списке слов устаревают и становятся более склонными к изменению своего значения.

## 6.2 Восстановление филогенетических деревьев

Ручное построении деревьев требует больших усилий: необходимо работать сразу с большим числом языков, требуются предварительные лексикологические исследования, малоизученные семьи языков требуют предварительной классификации и т.п.

Общая схема сводится к следующему:

1. Вычислить расстояние между словами в списках Сводеша;
2. Вычислить расстояние между языками на основе межсловных расстояний;
3. Сохранить результаты в матрицу попарных расстояний;
4. Построить дерево по найденной матрице.

### 6.2.1 Формула расстояния

Расстояние между словами равняется нулю, если они из одного класса когнации, иначе – единице. Тогда расстояние между языками имеет вид (19):

|  |  |  |
| --- | --- | --- |
|  |  | (19) |
| где | – слова для i-го понятия в списке Сводеша для каждого из языков; |  |
|  | N – длина списка Сводеша. |  |

То есть расстояние – это процент несовпадающих слов. Недостатком такого подхода является необходимость знать классы когнации. В таком случае лучше использовать признаковые методы. Например, можно считать расстояние Левенштейна между словами (20).

|  |  |
| --- | --- |
|  | (20) |

Такое решение не учитывает случайные совпадения. В качестве решения можно сравнить расстояние между парами соответствующих и несоответствующих слов. Окончательное решение (21):

|  |  |
| --- | --- |
|  | (21) |

### 6.2.2 Построение филогенетического дерева

Существует множество алгоритмов: UPGMA, NeighborNET, Neighbor-joining и т.д. Общая идея всех алгоритмов:

1. найти двух соседей в дереве;

2. объединить их в новую вершину, посчитать расстояние от новой вершины до прежних;

3. пересчитать расстояния между оставшимися вершинами.

Подобный подход применяется в аггломеративной кластеризации. Кластеризация – задача разделения объектов на группы, состоящие из близких объектов, причем эти группы неизвестны заранее. В аггломеративных алгоритмах объекты последовательно объединяются в группы, пока не останется нужное число групп. Однако при построении деревьев нужны не только расстояния между группами, но и расстояния между группами и их элементами.

### 6.2.3 Алгоритм иерархической кластеризации

1. На каждом шаге есть текущие группы объектов , где i – номер шага, а также матрица D расстояний между ними.

2. Вначале все группы одноэлементны, их высота равно 0.

3. Пока число групп больше 1:

3.1 найти две самых близких группы объединить их в новую .

3.2 Добавить в лес новую вершину . Связать ее с и .

3.3 Положить .

Далее нужно интерпретировать результат, вычисляя расстояния от корня до листьев. Для этого надо найти для каждой вершины ее высоту (22).

|  |  |
| --- | --- |
|  | (22) |

Длина ребра между вершинами – разность высот его концов.

Длина пути между вершинами – сумма длин ребер. То есть длина пути от корня до любого листа одинакова и равна высоте корня.

Таким образом используется методология «молекулярных часов». Такой подход не позволяет учитывать нелексическую информацию, при этом регулярные и нерегулярные соответствия штрафуются одинаково; одинаково наказываются изменения в корне и суффиксе, сложно выбрать общий алфавит (IPA не отражает все детали). Поэтому также используются дискретные методы, аналогичные алгоритмам восстановления генома в биологии. Единственным отличием является наличие для каждого признака своего набора значений, что ведет к большему числу параметров [7].

# ЗАКЛЮЧЕНИЕ

В ходе ознакомления с математическими и компьютерными моделями в лингвистике были изучены основы различных направлений лингвистики и используемые в них математические модели. Для начала было установлено, что с появлением компьютерной лингвистики любые математические модели, включая существовавшие ранее, были включены в это направление. Далее выяснилось, что в компьютерной лингвистике можно выделить три этапа (или подхода), ориентированные на различные модели:

- первый подход ориентирован на дискретные и статические модели, поэтому использует теорию формальных языков и математическую логику, например, конечные автоматы и конечные преобразователи. Таким образом лингвисты пытались воссоздать модель языка и его структуру. Задачи сводились к созданию моделей, выводящих правила или описывающих их. Соответствующие модели ориентированы на морфологию, фонологию и тому подобные направления, работающие с минимально дифференцируемыми единицами;

- второй этап, ориентированный на стохастические модели, доминировал до недавнего времени и позволял получить результаты быстро, но часто плохо интерпретируемые. Для моделей данного этапа характерно использование метрик (расстояние Левенштейна и другие) и корпусов для определения на их основе наиболее вероятных вариантов. Такие модели широко использовались для решения практических задач: перевода, исправления опечаток и т.п. К этим моделям также добавляются различные индексы и параметры для описания гипотез, частотностей, распределений и т.п., например, среднеквадратическое отклонение, закон Ципфа и др. Огромен вклад моделей второго этапа и в теоретическую лингвистику, особенно в компаративистику (сравнительно-историческое языкознание);

- третий этап совмещает два предыдущих подхода: стохастическим моделям стараются привить правила вывода, основанные не только на частотах, но и на детерминированных процессах.

Таким образом гипотеза о том, что в лингвистике широко используются математические модели, описывающие язык и отдельные его процессы, подтвердилась. Следует отметить, что сегодня инженеры вынуждены все больше полагаться на лингвистов в вопросах моделирования лингвистических процессов, потому что сугубо стохастические функциональные модели, дают неполноценные результаты с практической точки зрения и плохо интерпретируются лингвистами. Симбиоз математической логики и стохастических методов с недавнего времени дает положительные результаты и, возможно, является перспективным направлением. Некоторые алгоритмы были реализованы на python3, соответствующие программы решают и моделируют некоторые практические задачи и процессы.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Foma. [Электронный ресурс]. URL: https://github.com/mhulden/foma/. (Дата обращения 04.06.2020).

2. How to Write a Spelling Corrector. [Электронный ресурс]. URL: http://norvig.com/spell-correct.html. (Дата обращения 04.06.2020).

3. Ying Zhang, Stephan Vogel, and Alex Waibel Interpreting BLEU/NIST Scores: How Much Improvement Do We Need to Have a Better System? // Language Technologies Institute, Camegie Mellon University – 2004.

4. Алпатов В. Языкознание: От Аристотеля до компьютерной лингвистики / Владимир Алпатов; - М.: Альпина нон-фикшн, 2018. – 253 с. Ил. – (Серия «Библиотека ПостНауки»).

5. Журавлев Владимир Константинович Диахроническая фонология / Отв. Ред. В.А. Виноградов. - М.: Едиториал УРСС, 2016. – 264 с.

6. Козлова Наталья Валерьевна Лингвистические корпуса: определение основных понятий и типология // Вестник НГУ. Серия: Лингвистика и межкультурная коммуникация. 2013. №1.

7. Лингвистическое время: глоттохронология, лексикостатистика. [Электронный ресурс]. URL: https://studopedia.ru/16\_94573\_glava--lingvisticheskoe-vremya-glottohronologiya-leksikostatistika.html. (Дата обращения 04.06.2020).

8. Медведева Т.Н. Формальные модели в лингвистике: Учебное пособие / Т.Н. Медведева. – Саратов: Научная книга, 2010. – 56 с.

9. Нечёткий поиск в тексте и словаре. [Электронный ресурс]. URL: https://habr.com/ru/post/114997/. (Дата обращения 04.06.2020).

10. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. – СПБ.: Питер, 2018. – 480 с.: ил. – (Серия библиотека программиста).

11. Ожиганов А.А. Теория автоматов. Учебное пособие. – СПб: НИУ ИТМО, 2013. - 84 с.

12. От автоматической обработки текста к машинному пониманию – аналитический портал ПО-ЛИТ.РУ. [Электронный ресурс]. URL: https://polit.ru/article/2013/03/26/vladimir\_selegey/. (Дата обращения 04.06.2020).

13. Пиперски А. Конструирование языков: От эсперанто до дотракийского / Александр Пиперски; - М.: Альпина нон-фикшн, 2017. – 224 с. Ил. – (Серия «Библиотека ПостНауки»).

14. Хакимов М.Х. Математические модели слов и предложений по типам английского языка для системы машинного перевода // Проблемы вычислительной и прикладной математики. — 2017. — № 5. — С. 50.

15. Черноглазова М. В. Методы системной фонологии // Ярославский педагогический вестник. 2003. №1.

16. Ясулова Хадижат Сиражудиновна Математические модели синтаксиса // Известия ДГПУ. Естественные и точные науки. 2011. №3.

# ПРИЛОЖЕНИЯ

Приложение 1

Код программы для расчета расстояния Левенштейна на Python3

def levenstein(u: str, v: str) -> int:

"""Calculate the Levenshtein distance between u and v."""

n, m = len(u), len(v)

if m < n:

u, v = v, u

n, m = m, n

cur\_row = range(n+1)

for i in range(1, m+1):

prev\_row, cur\_row = cur\_row, [i] + [0]\*n

for j in range(1, n+1):

add, delete, change = prev\_row[j] + 1, cur\_row[j-1] + 1, prev\_row[j - 1]

if u[j-1] != v[i-1]:

change += 1

cur\_row[j] = min(add, delete, change)

return cur\_row[n]

Приложение 2

Код программы примитивного корректора опечаток на Python3

import re

from collections import Counter

def words(text):

return re.findall(r'\w+', text.lower())

WORDS = Counter(words(open('big.txt').read()))

def P(word, N=sum(WORDS.values())):

"Probability of `word`."

return WORDS[word] / N

def correction(word):

"Most probable spelling correction for word."

return max(candidates(word), key=P)

def candidates(word):

"Generate possible spelling corrections for word."

return (known([word]) or known(edits1(word)) or known(edits2(word)) or [word])

def known(words):

"The subset of `words` that appear in the dictionary of WORDS."

return set(w for w in words if w in WORDS)

def edits1(word):

"All edits that are one edit away from `word`."

letters = 'abcdefghijklmnopqrstuvwxyz'

splits = [(word[:i], word[i:]) for i in range(len(word) + 1)]

deletes = [L + R[1:] for L, R in splits if R]

transposes = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R)>1]

replaces = [L + c + R[1:] for L, R in splits if R for c in letters]

inserts = [L + c + R for L, R in splits for c in letters]

return set(deletes + transposes + replaces + inserts)

def edits2(word):

"All edits that are two edits away from `word`."

return (e2 for e1 in edits1(word) for e2 in edits1(e1))

Приложение 3

Конечный преобразователь для вывода формы множественного числа в английском языке

