Введение в компьютерную лингвистику

Предмет изучения. Задачи. Регулярные выражения. Нормализация текста. Расстояние Левенштейна

Предмет и методы

- Теоретическая лингвистика

  изучает язык как систему

- Прикладная лингвистика

  применяет результаты изучения языка для преподавания (например)

- **Компьютерная лингвистика**

  тоже изучает язык как систему

- NLP (Natural language processing)

  применяет результаты изучения языка и методы ИИ для решения практических задач

Методы КЛ

- Моделирование
- Методы теории вероятности
- Методы математической статистики
- А математическая статистика – это машинное обучение

Машинное обучение и КЛ

- Искусственный интеллект:

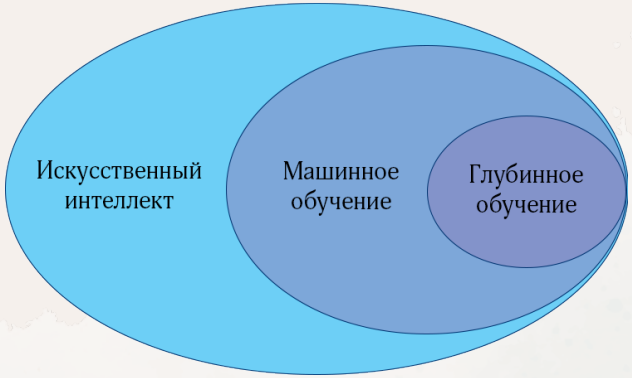
  все о имитации человеческого интеллекта

- Машинное обучение:

  область ИИ, алгоритмы, по которым машины «учатся» решать задачи на основании обучающих данных – примеров, как правильно решать

- Глубинное обучение (нейронные сети):

  подвид алгоритмов машинного обучения



Классическое МО vs нейросети

Классическое МО

- Данные (простые)
- Алгоритмы (тоже простые)

- Признаки (установленные вручную)

Нейросети

- Данные (сложные)
- Алгоритмы (нелинейные, с миллиардами параметров)
- Признаки не нужны: нейросеть устанавливает их сама

Основные задачи

- Совершенства достичь нельзя, поэтому основные виды задач – одни и те же и не изменяются (но могут добавляться новые)
- NLP решает эти задачи, чтобы потом зарабатывать на решениях деньги, КЛ – чтобы в процессе решения исследовать устройство естественного языка
- Классический пример задачи – задача автоматического (машинного) перевода
- Появляются новые решения этих задач, с каждым годом все лучше
- Как оценить, что новое решение лучше?

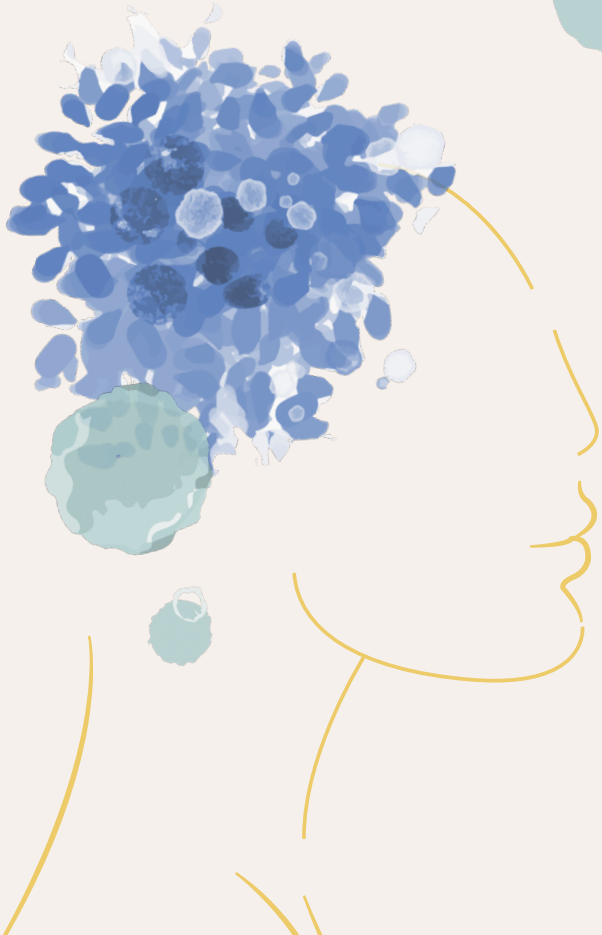
Основные задачи

Создать бейзлайн

Бейзлайн (иногда бейслайн) – базовое, самое простое решение. Наша цель – его побить

Использовать числовые метрики

Если есть «золотой» стандарт – например, эталонный перевод, выполненный человеком – можем сравнивать работу нашего алгоритма с ним



Метрики

Для разных задач существуют свои метрики. Понятно, как сравнить качество частеречной разметки: угадал алгоритм часть речи или нет? Но как оценить перевод? Генерацию текста? Лемматизацию?

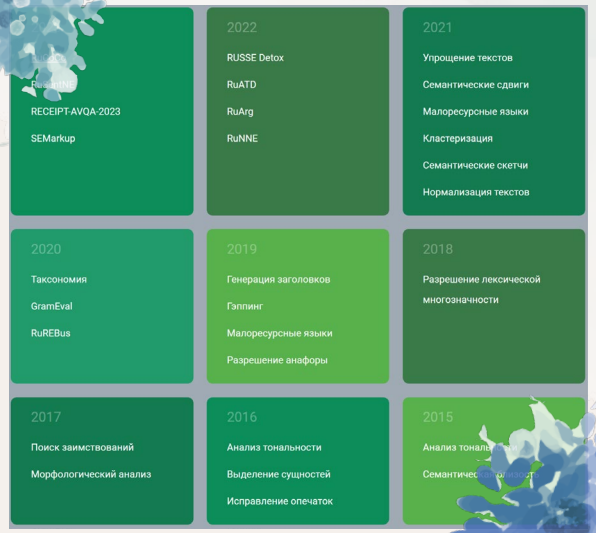
SOTA

Решение, которое показывает самые высокие оценки по метрикам, принятым в данной задаче, называется State of the Art (SOTA)

Соревнования

Если оцениваем качество решения в цифрах, можно соревноваться!
По многим задачам NLP устраиваются соревнования (shared tasks) – на них разные команды пытаются создать самое крутое решение и этим сделать мир лучше

Например, конференция «Диалог» ежегодно проводит соревнования: https://www.dialog-21.ru/evaluation/



Основные задачи nlpprogress.com

- NLU (**understanding**) – понимание естественного языка
- NLP (**processing**) – обработка естественного языка
- NLG (**generation**) – порождение естественного языка

NLU

Natural Language Understanding

MT, NLI, Common Sense, QA, чатботы

LM

Language Modeling

Цепи Маркова, word2vec, BERT, GPT

ASR

Automatic Speech Recognition

Распознание речи

IE

Information Extraction

NER, NEL, Summarization

LA

Linguistic Annotation

Морфология, синтаксис, семантика

Инструменты КЛ

Очевидно, мы не можем жить без программирования (и матана)

Теория автоматов

- Изучает абстрактные автоматы (математические модели вычислительных машин) и задачи, которые они могут решать
- Автомат – это обобщенное представление программы
- Автоматы бывают:
  - Детерминированные и недетерминированные:
  - Конечные (d\n finite state automata)
  - С магазинной памятью (d\n pushdown automata)
  - Машины Тьюринга (d\n Turing machines)

что почитать: учебник по теории автоматов (осторожно, сложно)

Формальные языки

- Формальный язык – это математическая модель реального языка (и часть теории автоматов)
- Основные понятия:

Алфавит – набор символов (конечный и непустой)

Слово (цепочка) – конечная последовательность символов некоторого алфавита

Язык – множество слов из некоторого алфавита

- Язык обычно описывается в виде: $L = \{w | \text{сведения о } w\}$
- Теория автоматов любит решать «проблемы»: относится ли данное слово к языку L или нет
- Хомский придумал свою иерархию, пытаясь приспособить теорию автоматов для описания синтаксиса естественного языка (phrase structure grammar)

Формальные грамматики Хомского

0

Неограниченные грамматики

Вообще все формальные грамматики Phrase Structure Grammars (Машина Тьюринга)

1

Контекстно-зависимые грамматики

Слева и справа может быть контекст $L = \{a^n b^n c^n | n \geq 1\}$ (ограниченная машина Тьюринга)

2

Контекстно-свободные грамматики

Любые языки программирования (автомат с магазинной памятью)

3

Регулярные языки

Регулярные выражения (конечный автомат)

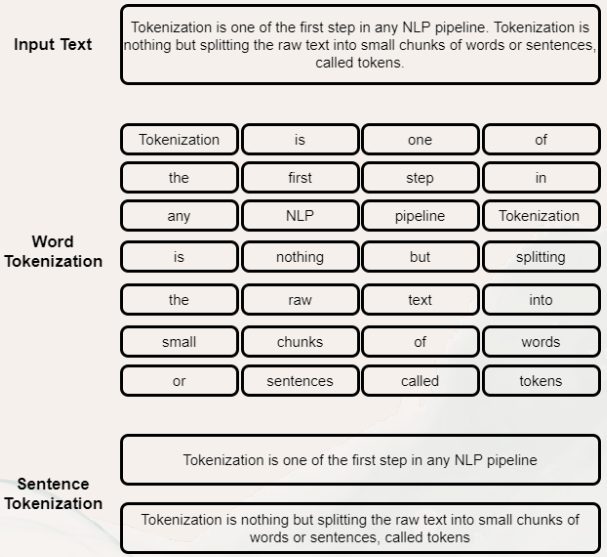
Регулярные выражения

- Регулярный язык – самый простой
- У него есть своя **алгебра** (или синтаксис)
- Основные применения языка:
  - лексические анализаторы (часть компилятора для какого-нибудь языка программирования)
  - **текстовый поиск, подстановка, замена**

- Очень часто и много используется для решения задач КЛ

Пайплайн

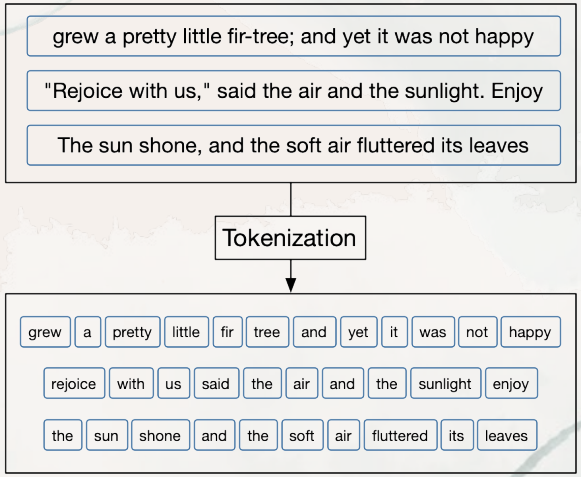
- Работаем всегда с текстом;
- Всегда на компьютере;
- Следовательно, текст нужно обрабатывать:
  - делить на слова
  - делить на предложения
  - нормализовать (?)
- Тут-то нам и пригодятся регулярки
- Этапы обработки текста называются **пайплайн**



Токенизация

- Компьютер воспринимает текст как набор символов
- Нам нужно, чтобы он его умел сегментировать
- На какие кусочки?

- Предложения
- Слова
- Токены
- Token types (уникальные токены)



- **Токен – это значимая для анализа последовательность символов**

Heap's Law

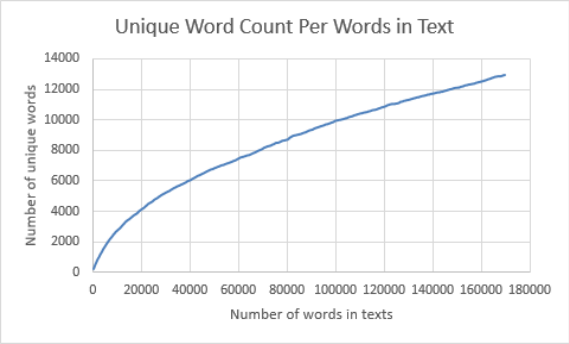
В тексте есть токены и token types (уникальные токены):

In a hole in the ground there lived a hobbit.

Сколько слов в предложении?

Сколько уникальных слов?

Какое между ними соотношение? (Token Type Ratio)



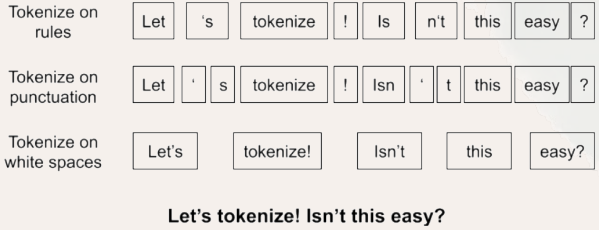
Закон Хипса (Хердана):

$V_R(n) = Kn^{\beta}$, где V_R – число уникальных слов в тексте размера n, K и β – свободные параметры, которые устанавливаются эмпирически.

Устанавливает функцию отношения уникальных слов ко всем словам в документе.

Токенизация: сложности

- New York
- 01.09.2023
- 2 см^2
- doesn’t
- 計算語言學
- اللغويات الحاسوبية



BPE-Токенизация

- Вместо того, чтобы нам придумывать определение токена и самим писать правила, заставим текст сообщить нам, что в нем – токены
- Byte Pair Encoding – алгоритм автоматической токенизации по **подсловам** (subwords)
- Современные нейронные сети используют BPE
- Понадобятся **обучающие данные** – набор сырых текстов, на которых будем учить, какие бывают токены
- «Собираем» токены из символов

BPE-токенизация

Слова в наших обучающих данных: (“hug”, 10), (“pug”, 5), (“pun”, 12), (“bun”, 4), (“hugs”, 5)

Алфавит: [“b”, “g”, “h”, “n”, “p”, “s”, “u”], значит, можем представить слова так:

(“h” “u” “g”, 10), (“p” “u” “g”, 5), (“p” “u” “n”, 12), (“b” “u” “n”, 4), (“h” “u” “g” “s”, 5)

Устанавливаем, какого объема словарь хотим.

Вычисляем самые частотные сочетания: самая частотная пара – “ug”, она встретится 20 раз

Соединим эти два символа и получим новый словарь: [“b”, “g”, “h”, “n”, “p”, “s”, “u”, “ug”]

Тогда наш корпус будет выглядеть так:

(“h” “ug”, 10), (“p” “ug”, 5), (“p” “u” “n”, 12), (“b” “u” “n”, 4), (“h” “ug” “s”, 5)

Какие два “слова” тогда будут чаще всего вместе? Найдите.

Склеим их в одно новое слово и добавим в наш словарь.

И так пока не доведем наш словарь до желаемого объема.

Нормализация

Случаи вида:

- USA, US
- гм-м, гмм
- Газпром, «Газпром»

И такие:

- компьютерная, Компьютерная

Хотим унифицировать! Это и называется нормализация (а еще есть case folding – приведение всех слов к одному регистру)

Вопрос: всегда ли полезно приводить все слова к одному регистру?

Лемматизация и стемминг

В каком предложении выше TTR? Почему?

a) Компьютерная лингвистика отличается от теоретической лингвистики используемыми методами.

b) Computational linguistics differs from theoretical linguistics in the methods used.

Лемматизация и стемминг

- *Лемма*
- Словоформа

Лемматизация – приведение слова к его ***лемме***.

Стемминг – приведение слова к его **псевдооснове** (stem).

Что такое псевдооснова?

Один из самых известных алгоритмов стемминга –

алгоритм Портера (1980)

Стемминг

Лемматизация

Расстояние Левенштейна

- Пользователь напечатал: **лцнь**. Что он имел в виду – *лень* или *луна*?
- Нужно сравнить, какие пары строк более похожи: *лцнь-лень* или *лцнь-луна*
- Посчитаем, сколько нужно произвести посимвольных замен:

- *лцнь* → *лень*: одна замена

- *лцнь* → *луна*: две замены

- Минимальное количество замен для «лень»
- Такое количество односимвольных изменений называется минимальным редакционным расстоянием (minimum edit distance), или расстоянием Левенштейна (Levenshtein distance)

Расстояние Левенштейна

- Виды посимвольных преобразований:
- удаление
- вставка
- замена

H	O		N	D	A	
H	Y	U	N	D	A	I

H	Y	U	N	D	A	I
H		O	N	D	A	

- Есть вариант – расстояние Дамерау-Левенштейна, где есть еще операция перестановка
- Это расстояние может помочь нам вычислить выравнивание (alignment) – что с чем сопоставляется в двух строках.
- Выравнивание важно, например, в машинном переводе (когда нужно сопоставить слова в предложениях)

Алгоритм Вагнера-Фишера

$$D(i,j) = \begin{cases} 0, & i = 0, j = 0 \\ i, & j = 0, i > 0 \\ j, & i = 0, j > 0 \\ \min\{ & \\ D(i, j-1) + 1, & \\ D(i-1, j) + 1, & j > 0, i > 0 \\ D(i-1, j-1) + m\big(S_1(i), S_2(j)\big) & \\ \} & \end{cases}$$

Выглядит жутко! Что это значит?

- i и j – длины наших двух строчек. Если обе строки содержат 0 символов, расстояние тоже равно 0!
- Если одна из строк содержит 0 символов, то расстояние равно длине другой строки.
- Во всех остальных случаях вычисляем тот страшный минимум. $m\big(S_1(i), S_2(j)\big)$ равно 0, если два символа равны друг другу, и 1, если не равны.

Алгоритм Вагнера-Фишера

		Л	А	Б	Р	А	Д	О	Р
	0	1	2	3	4	5	6	7	8
Г	1								
И	2								
Б	3								
Р	4								
А	5								
Л	6								
Т	7								
А	8								
Р	9								

Чтобы посчитать, будем построчно заполнять такую таблицу.

D(1, 1) – это расстояние между символами «л» и «г».

Они не равны, значит,

$$min\begin{cases} D(0,1) + 1 \\ D(1,0) + 1 \\ D(0,\ 0) + m('г',\ 'л') \end{cases} = D(0,\ 0) + 1$$

Значит, вписываем 1 в желтую клеточку.

Алгоритм Вагнера-Фишера

		Л	А	Б	Р	А	Д	О	Р
	0	1	2	3	4	5	6	7	8
Г	1	1							
И	2								
Б	3								
Р	4								
А	5								
Л	6								
Т	7								
А	8								
Р	9								

Сдвинемся вправо на клеточку и посчитаем по той же формуле.

$$D(1,2) = min\begin{cases} D(0,2)+1 \\ D(1,1)+1 \\ D(0,1)+1 \end{cases} = min\begin{cases} 2+1 \\ 1+1 \\ 1+1 \end{cases} = 2$$

Алгоритм Вагнера-Фишера

		Л	А	Б	Р	А	Д	О	Р
	0	1	2	3	4	5	6	7	8
Г	1	1	2	3	4	5	6	7	8
И	2	2	2	3	4	5	6	7	8
Б	3	3	3						
Р	4								
А	5								
Л	6								
Т	7								
А	8								
Р	9								

Таким образом, значения в ячейках будут увеличиваться на единицу, пока не дойдем до букв «б», «б», которые совпадают.

$$D(3,3) = min\begin{cases} D(2,3)+1 \\ D(3,2)+1 \\ D(2,2)+0 \end{cases} = min\begin{cases} 3+1 \\ 3+1 \\ 2+0 \end{cases} = 2$$

Попробуйте самостоятельно заполнить таблицу до конца.

Алгоритм Вагнера-Фишера

		Л	А	Б	Р	А	Д	О	Р
	0	1	2	3	4	5	6	7	8
Г	1	1	2	3	4	5	6	7	8
И	2	2	2	3	4	5	6	7	8
Б	3	3	3	2	3	4	5	6	7
Р	4	4	4	3	2	3	4	5	6
А	5	5	4	4	3	2	3	4	5
Л	6	5	5	5	4	3	3	4	5
Т	7	6	6	6	5	4	4	4	5
А	8	7	6	7	6	5	5	5	5
Р	9	8	7	7	7	6	6	6	5

Совпало?

Расстояние Левенштейна по этой таблице – нижняя правая ячейка, D(9,8).