уценыш

исходя из текущей архитектуры мы можем делать морфемный и деривационный анализ (если постараться). анализ структуры составляющих словоформы/основы.

без имени – 2 слова vs безымянный – 1 слово

структура составляющих:

словоформа = основа + флексия

словоформа = мотивирующая основа + словообразующий формант

до сих пор мы пробегались по словоформе, пробегались с конца, знали максимальную длину флексий – это несложно.

если нам нужно разбить текст на энное кол-во минимальных значимых единиц. цэ из эн по ка (пока). (=примерно факториал, т.е выше экспоненты). недетерминированный полином, приведет к комбинаторному взрыву – неприменимо в реальном программировании.

перебор всех подстрок заданного текста. Эн-количество символов в строке. Сперва перебираем все возможные началы, потом все возможные концы. Это Эн квадрат пополам. Разбивая на подстроки, мы пробегаемся по словарю морфем, если находим – то норм разбили. Но это квадратичная функция – тоже неэффективно. Представьте ВиМ в квадрате..

Что же делать? Придумали \_\_\_ и Маргаретт Корасик. Алгоритм Ахо-Корасик (его структура сети переходов). Имеет не квадратичное, а линейное время. У нас не происходит перемножение текста само на себя.

Алгоритм уже реализован (на языке python!) в модуле работы со строками - string\_algorithms (если не хотим имплементировать). Более стандартная реализация этого алгоритма для питона/винды(?) – теоретически на винде должна работать, но практике – грустно.

#конечные автоматы #язык ассессор

**Алгоритм ахо-корасик.** На входе у него строка. У него в качестве данных уже есть список морфем – подстрок. На выходе он выдает все подстроки этой строки, которые есть в словаре.

В «мамами» – мам ам а ами ми и

На входе – мамами. Алгоритм видит пожстроку **мам** с позицией 0 , **ам** – позиция 1, а – 1, м – 2, мам -2, ам – 3, а – 3, м – 4, ми – 4, и – 5. (позиция первых элементов подстроки, может выдавать и иначе). Пересечение/вложение морфем друг в друга и их совместимость алгоритм не проверяет.

мам

ам

а

м

мам

ам

а

м

ми

и

Алгоритм применяется в антивирусах. Там «множество морфем» = множество вирусов, а «строка» = память компьютера.

Предыдущий стеммер выдает много вариантов не пойми чего.

Далее превратить разбиения алгоритма в сегментации. (зд: мам+а, мам+ам, мам+ами).

Если морфемы нет в нашем словаре нам нужны сегментации максимальной длины, полные, без пробелов, без пересечений.

Далее мы будем строить граф, в котором каждая морфема будет начинаться там, где заканчивается предыдущая. (вершины графа – морфемы).

Сперваа нам надо подготовить массив морфем на вход – массив строк.

нулевая задача – импортировать и заставить работать словарь

первая – сегментировать

далее – фильтрование сегментации

в итоге мы сделаем конечный автомат.

#грамматика конечных составляющий (первый тип в иерархии Хомского), теория порождающих грамматик

подвывернутый – под вы вер ну т ый

1. **короче ннада сделать рекурсивный генератор, который будет вызывать сам себя, когда у нас есть морфемы по остатку слова. если есть параллельные варианты, то он их принимает на вход параллельно в одном массиве. например (А и АБ) в «абвгд». в этот массив уже ничего не добавляем. потом что-то йилдим. Йилдим списки с образующими слово морфемами. если недособирали комбо до конца, всё равно йилдим, потом при необходимости их можно будет отсортировать.**

**СОЗДАЕМ КЛАСС МОРФЕМ, ГДЕ ХРАНИМ ИХ ИНДЕКСЫ, чтобы сохранить че-то там в памяти и в непонятных структурах.**

1. **конечный автомат делать классом. у него метод – словарь переходов. метод перехода в след состояние, автомат либо переходит в след состояние, либо возвр фолс.**