Наша сегментация совершенно не учитывает типы морфем друг с другом. То есть: флексии могут оказаться в начале слова, а префиксы в конце... Это плохо. Нам нужен конечный автомат, который будет контролировать выдачу.

(См. конспект.)

Нужно, чтобы у нас был словарь с информацией о типе каждого морфа. При этом у одного морфа м. б. несколько типов (ср. «а»); словарь вида {морф: [множество типов]}. [Сами морфы типизировать!]

Конечный автомат (граф) тж. можно представить в виде словаря: {'Pr': {'Pr', 'R'}}. Потом будем проверять, можем ли мы перейти или нет.

(Задача 0.) Сам автомат сделать через класс, экземпляр которого изначально находится в состоянии start, а в каждый момент времени может находиться только в одном состоянии. Обладает сетью переходов и пытается её обходить. Если не может — поднимает исключение.

(Задача 1.) Нашу «слепую» сегментацию превратить в пару объектов вида (последовательное множество типов, последовательное множество морфов). Разберём пример «ами»:

«а» — Pr, R, S, F.

«ми» — R, F.

Нужно их декартово произведение; функцию импортировать — itertools.product. Для каждого из вариантов выдачи (в данном случае — пар) запускаем конечный автомат и пытаемся осуществить переход из начала в конец. В данном случае проходят следующие интерпретации:

Pr R,

R F.

Здесь это не есть хорошо, но чтобы было лучше, нужна более строгая типизация.

Декартово произведение — это плохо: вычислительная сложность экспоненциально растёт в зависимости от количества морфем. Засим на очень длинном слове возможен комбинаторный взрыв. Но конечный автомат примитивен и не может одновременно находиться в нескольких состояниях.

(Задача 2.) Улучшить реализацию таким образом, чтобы маршруты с нулевой флексией генерировали ещё и саму нулевую флексию.

(Задача 3.) Интегрировать в стеммер — реализовать удаление флексии, а не просто слайс. Например: «по|на|смотр|е|вш|ий|ся»; в данном случае можно удалить ещё и словоизменительный суффикс.

(Задача 4.) Подключить такой стемминг к поисковой системе в качестве фолбэка для лемматизатора (он более надёжен).

(Задача 5.) Попытаться реализовать конечный автомат так, чтобы в тестовом примере не было некорректных (с лингвистической точки зрения) сегментаций.