Проект этап 3. Создайте модель цепи Маркова

Цепь Маркова — это статистическая модель, в которой вероятность каждого события зависит от предыдущего события. Его можно описать как набор состояний и переходов между ними. Каждый переход имеет вероятность, которая определяется какими-то статистическими данными. В этом проекте состояние соответствует токену, и каждый переход представляет собой переход от одного слова предложения к другому. Вероятность переходов рассчитывается из биграмм, собранных нами на предыдущем этапе.

Основная идея этого проекта заключается в том, что из словаря мы можем создать модель, которая будет учитывать все возможные переходы от одного слова к другому и выбирать наиболее вероятный на основе предыдущего слова.

Описание

Это последний шаг, на котором мы будем работать над созданием модели цепи Маркова. Мы будем использовать данные, подготовленные на первых двух этапах, и преобразовать их в модель. Эта модель будет содержать вероятностную информацию, которая скажет нам, каким может быть следующее слово в цепочке.

У нас уже есть список всех биграмм из корпуса. Как мы обсуждали ранее, это уже можно использовать для некоторых наивных предсказаний. Однако есть проблема: прямо сейчас наши данные содержат много повторений.

Как мы видели на первом этапе, общее количество токенов почти в 10 раз превышает количество уникальных токенов. Это соотношение должно быть примерно таким же в списке биграмм. Одни биграммы могут быть очень распространенными, другие — относительно редкими. На данный момент мы не можем сказать, кто есть кто.

Чтобы решить эту проблему, мы сделаем упрощенную версию модели цепи Маркова.

Задачи

1. Данные должны быть реорганизованы таким образом, чтобы каждая головка повторялась только один раз, а ко всем возможным хвостам можно было получить прямой доступ, запросив эту голову.

Например: **`Голов`a — good,** **`Хвост`** **— night, bye, bye, night, to, to, bye, boy.** Как видите, среди хвостов еще есть повторы.

2. Вместо того, чтобы повторять хвосты каждый раз, когда они появляются, каждый хвост должен появляться только один раз, а количество повторений должно храниться как целое число.

Например, предыдущий пример должен выглядеть так: `Голова` — good, `Хвост` — night: 2, bye: 3, to: 2, boy: 1. Вы можете видеть, что данные стали более читаемыми после этого преобразования!

3. Помимо создания модели, мы также должны проверить, правильно ли она работает. Чтобы проверить это, давайте возьмем строку в качестве пользовательского ввода и напечатаем все возможные `хвосты` и соответствующие им значения.

Если модель не содержит указанную `голову`, напечатайте следующее сообщение об ошибке **KeyError. Запрашиваемое слово отсутствует в модели. Пожалуйста, введите другое слово.** и просить другой ввод, пока он не будет действительным. Повторяйте до тех пор, пока не будет введена строка **exit**.

Пример

Вывод программы должен выглядеть примерно так.

```
> corpus.txt
> Night
Head: Night
Tail: King    Count: 17
Tail: gathers Count: 9
Tail: King's  Count: 4
Tail: is      Count: 2

> Jon
Head: Jon
Tail: Snow    Count: 36
Tail: Snow.   Count: 29
Tail: Arryn   Count: 14
Tail: said    Count: 10
Tail: often   Count: 6
Tail: knows   Count: 5
Tail: left    Count: 5

> Northampton
Head: Northampton
KeyError. Запрашиваемое слово отсутствует в модели. Пожалуйста,
введите другое слово.

> King
Head: King
Tail: in      Count: 76
Tail: Robert  Count: 29
Tail: of      Count: 24
Tail: Joffrey Count: 20
Tail: Tommen  Count: 6
Tail: Stannis Count: 5
Tail: Robb    Count: 5

> exit
```