

Генерация текста

План

- Задачи генерации
- N-грамм генерация
- Уменьшение размера словаря, токенизация

Зачем это нужно?

Автодополнение слова

В лесу родилась елочка

Зачем это нужно?

Автодополнение слова

В лесу родилась елочка

Автодополнение фразы

Почему птицы летают
поют
летят клином

Зачем это нужно?

Автодополнение слова

В лесу родилась елочка

Автодополнение фразы

Почему птицы летают
поют
летят клином

Диалоговые системы

– Какая погода в Москве?

В Москве сейчас 8 градусов –

Постановка задачи

Текст должен удовлетворять требованиям:

- Логическая связность
- Соблюдение языковых норм

Можно попытаться задать правила вручную.

Но правил слишком много, поэтому ничего хорошего не выйдет.

Постановка задачи

Будем учиться имитировать человеческую речь.

Для этого научимся оценивать вероятность текстов.

$p(\text{В лесу родилась елочка})$ $p(\text{У лукоморья дуб зеленый})$

Постановка задачи

Будем учиться имитировать человеческую речь.

Для этого научимся оценивать вероятность текстов.

$p(\text{В лесу родилась елочка}) \quad \vee \quad p(\text{У лукоморья дуб зеленый})$

$$\frac{\parallel 1}{|\text{dataset}|}$$

$$\frac{1 \parallel}{|\text{dataset}|}$$

Постановка задачи

Будем учиться имитировать человеческую речь.

Для этого научимся оценивать вероятность текстов.

$p(\text{В лесу родилась елочка}) \quad \vee \quad p(\text{У лукоморья дуб зеленый})$

$$\frac{\parallel 1}{|\text{dataset}|}$$

$$\frac{1 \parallel}{|\text{dataset}|}$$

Работать с текстом как с одним целым невозможно!

Постановка задачи

Разделим текст на слова.

x – текст из m слов.

Обучим модель оценивать вероятность набора слов.

$$p(x) = p(x_1, \dots, x_m)$$

Постановка задачи

Разделим текст на слова.

x – текст из m слов.

Обучим модель оценивать вероятность набора слов.

$$p(x) = p(x_1, \dots, x_m)$$

Заменим совместную вероятность на произведение условных вероятностей.

$$p(x_1, \dots, x_m) = p(x_1) \cdot p(x_2 | x_1) \cdot p(x_3 | x_1, x_2) \cdot \dots \cdot p(x_m | x_1, \dots, x_{m-1}) = \prod_{i=1}^m p(x_i | x_{<i})$$

Постановка задачи

$$p(x_1, \dots, x_m) = \prod_{i=1}^m p(x_i | x_{<i})$$

Достаточно обучить модель оценивать вероятность $p(x_i | x_{<i})$.

Постановка задачи

$$p(x_1, \dots, x_m) = \prod_{i=1}^m p(x_i | x_{<i})$$

Достаточно обучить модель оценивать вероятность $p(x_i | x_{<i})$.

Все еще сложно, потому что надо учитывать все предыдущие слова.

Упростим задачу – будем смотреть только на предыдущие n слов.

$$p(x_1, \dots, x_m) \approx \prod_{i=1}^m p(x_i | x_{i-1}, \dots, x_{i-n})$$

Постановка задачи

$$p(x_1, \dots, x_m) = \prod_{i=1}^m p(x_i | x_{<i})$$

Достаточно обучить модель оценивать вероятность $p(x_i | x_{<i})$.

Все еще сложно, потому что надо учитывать все предыдущие слова.

Упростим задачу – будем смотреть только на предыдущие n слов.

$$p(x_1, \dots, x_m) \approx \prod_{i=1}^m p(x_i | x_{i-1}, \dots, x_{i-n})$$

Такая модель называется n -граммной.

N-грамм модель генерации

Предполагаем, что следующее слово зависит только от n предыдущих

$$p(x_1, \dots, x_m) \approx \prod_{i=1}^m p(x_i | x_{i-1}, \dots, x_{i-n})$$

N-грамм модель генерации

Предполагаем, что следующее слово зависит только от n предыдущих

$$p(x_1, \dots, x_m) \approx \prod_{i=1}^m p(x_i | x_{i-1}, \dots, x_{i-n})$$

Если у слова меньше, чем n предыдущих, дополним пропуски символом <PAD>.

$$\begin{aligned} p(\text{В, лесу, родилась, елочка}) = & \\ & p(\text{В} \mid \text{<PAD>, <PAD>}) \\ & \cdot p(\text{лесу} \mid \text{<PAD>, В}) \\ & \cdot p(\text{родилась} \mid \text{В, лесу}) \\ & \cdot p(\text{елочка} \mid \text{лесу, родилась}) \end{aligned}$$

N-грамм модель генерации: обучение

- Нужно оценить вероятность $p(x_i | x_{i-1}, \dots, x_{i-n})$.
- Посчитаем вручную, сколько раз x_i встретилось после x_{i-1}, \dots, x_{i-n} .

$$p(x_i | x_{i-1}, \dots, x_{i-n}) = \frac{p(x_i, x_{i-1}, \dots, x_{i-n})}{p(x_{i-1}, \dots, x_{i-n})} = \frac{N(x_i, x_{i-1}, \dots, x_{i-n})}{N(x_{i-1}, \dots, x_{i-n})}$$

N-грамм модель генерации: обучение

- Нужно оценить вероятность $p(x_i | x_{i-1}, \dots, x_{i-n})$.
- Посчитаем вручную, сколько раз x_i встретилось после x_{i-1}, \dots, x_{i-n} .

$$p(x_i | x_{i-1}, \dots, x_{i-n}) = \frac{p(x_i, x_{i-1}, \dots, x_{i-n})}{p(x_{i-1}, \dots, x_{i-n})} = \frac{N(x_i, x_{i-1}, \dots, x_{i-n})}{N(x_{i-1}, \dots, x_{i-n})}$$

$$N(\text{В, лесу, родилась}) = 2$$

$$N(\text{В, лесу, обитает}) = 5$$

$$N(\text{В, лесу, было}) = 1$$

$$N(\text{В, лесу}) = 8$$

N-грамм модель генерации: обучение

- Нужно оценить вероятность $p(x_i | x_{i-1}, \dots, x_{i-n})$.
- Посчитаем вручную, сколько раз x_i встретилось после x_{i-1}, \dots, x_{i-n} .

$$p(x_i | x_{i-1}, \dots, x_{i-n}) = \frac{p(x_i, x_{i-1}, \dots, x_{i-n})}{p(x_{i-1}, \dots, x_{i-n})} = \frac{N(x_i, x_{i-1}, \dots, x_{i-n})}{N(x_{i-1}, \dots, x_{i-n})}$$

$$N(\text{В, лесу, родилась}) = 2 \qquad p(\text{родилась} | \text{В, лесу}) = \frac{2}{8}$$

$$N(\text{В, лесу, обитает}) = 5 \qquad \longrightarrow \qquad p(\text{обитает} | \text{В, лесу}) = \frac{5}{8}$$

$$N(\text{В, лесу, было}) = 1 \qquad p(\text{было} | \text{В, лесу}) = \frac{1}{8}$$

$$N(\text{В, лесу}) = 8$$

N-грамм модель генерации: генерация

Генерируем слова по одному в соответствии с вероятностями.

N-грамм модель генерации: генерация

Генерируем слова по одному в соответствии с вероятностями.

<PAD> <PAD> _

$$p(\text{В} \mid \text{<PAD> <PAD>}) = 0.3$$

$$p(\text{Я} \mid \text{<PAD> <PAD>}) = 0.2$$

$$p(\text{Из} \mid \text{<PAD> <PAD>}) = 0.15$$

N-грамм модель генерации: генерация

Генерируем слова по одному в соответствии с вероятностями.

<PAD> <PAD> B _

$$p(\text{доме} \mid \text{<PAD> B}) = 0.34$$

$$p(\text{лесу} \mid \text{<PAD> B}) = 0.23$$

$$p(\text{машине} \mid \text{<PAD> B}) = 0.09$$

N-грамм модель генерации: генерация

Генерируем слова по одному в соответствии с вероятностями.

<PAD> <PAD> В лесу _

$$p(\text{было} \mid \text{В лесу}) = 0.4$$

$$p(\text{воют} \mid \text{В лесу}) = 0.23$$

$$p(\text{родилась} \mid \text{В лесу}) = 0.09$$

N-грамм модель генерации: генерация

Генерируем слова по одному в соответствии с вероятностями.

<PAD> <PAD> В лесу было _

$$p(\text{темно} \mid \text{лесу было}) = 0.22$$

$$p(\text{холодно} \mid \text{лесу было}) = 0.2$$

$$p(\text{свежо} \mid \text{лесу было}) = 0.13$$

N-грамм модель генерации: генерация

Генерируем слова по одному в соответствии с вероятностями.

<PAD> <PAD> В лесу было темно

Преимущества n-грамм

- Тексты состоят из существующих n-грамм.
- Поэтому предложения грамматически верные.
- Модель проста в реализации и очень быстрая.

Недостатки n -грамм

- При генерации смотрит только на последние n слов.
- Из-за этого получаются логически несвязные тексты.
- При увеличении n вероятности слов оцениваются хуже.
- Из-за большого размера словаря многие n -граммы встречаются очень редко.

Недостатки n -грамм

- При генерации смотрит только на последние n слов.
- Из-за этого получаются логически несвязные тексты.
- При увеличении n вероятности слов оцениваются хуже.
- Из-за большого размера словаря многие n -граммы встречаются очень редко.

Как уменьшить размер словаря?

Уменьшение размера словаря

- Удаление стоп-слов
- Лемматизация
- Стемминг
- Изменение способа токенизации

Токенизация

- Любой язык содержит много уникальных слов.
(Около **200к** в русском и **170к** в английском)
- Токенизация по **словам** приводит к получению **огромного словаря**.

"В", "лесу", "родилась", "елочка".

Токенизация

- Букв в алфавите гораздо меньше.
(33 в русском языке и 26 в английском)
- Токенизируя по **буквам** можно значительно **уменьшить словарь**.

"В", " ", "л", "е", "с", "у", " ", "р", "о", "д", "и", "л", "а", "с", "ь".

Задачи алгоритма токенизации

- Возможность токенизировать любой текст.
- Размер словаря не превышает определенное значение.
- Токены несут в себе как можно больше смысла.
- Текст токенизируется наименьшим числом токенов.

Алгоритмы токенизации

- Byte-Pair Encoding
- WordPiece
- Unigram

Byte-Pair Encoding (BPE)

Корпус: ["вол", "стол", "стул", "ствол"]

- Стартовый словарь – все возможные буквы.
Словарь: ["в", "о", "л", "с", "т", "у"]
- Итеративно увеличиваем словарь, склеивая пары токенов в один.
- Склеиваем ту пару, которая встречается чаще всего в корпусе.

Byte-Pair Encoding: алгоритм

Корпус: ["вол", "стол", "стул", "ствол"]

Шаг 1

Словарь: ["в", "о", "л", "с", "т", "у"]

Пары	Встречаемость
с, т	3
о, л	3
в, о	2

Byte-Pair Encoding: алгоритм

Шаг 1

Пары	Встречаемость
с, т	3
о, л	3
в, о	2

Корпус: ["вол", "стол", "стул", "ствол"]

Словарь: ["в", "о", "л", "с", "т", "у", "ст"]

Добавляем "ст" 

Byte-Pair Encoding: алгоритм

Шаг 2

Пары	Встречаемость
о, л	3
в, о	2
ст, о	1
ст, у	1
ст, в	1

Корпус: ["вол", "стол", "стул", "ствол"]

Словарь: ["в", "о", "л", "с", "т", "у", "ст",
"ол"]

Добавляем "ол"



Byte-Pair Encoding: алгоритм

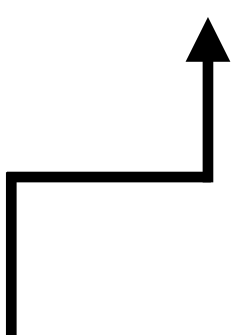
Шаг 3

Пары	Встречаемость
В, О	2
В, ОЛ	2
СТ, ОЛ	1

Корпус: ["вол", "стол", "стул", "ствол"]

Словарь: ["в", "о", "л", "с", "т", "у", "ст",
"ол", "вол"]

Добавляем "вол"



Byte-Pair Encoding: алгоритм

Шаг 4

Пары	Встречаемость
В, О	2
СТ, ВОЛ	1

Корпус: ["вол", "стол", "стул", "ствол"]

Словарь: ["в", "о", "л", "с", "т", "у", "ст",
"ол", "вол", "ВО"]

Добавляем "во" 

Byte-Pair Encoding: алгоритм

Корпус: ["вол", "стол", "стул", "ствол"]

Словарь: ["в", "о", "л", "с", "т", "у", "ст",
"ол", "вол", "во"]

Останавливаемся, когда словарь достиг нужного размера.

WordPiece токенизация

Корпус: ["вол", "стол", "стул", "ствол"]

- Стартовый словарь – все возможные буквы.
Словарь: ["в", "о", "л", "с", "т", "у"]
- Итеративно увеличиваем словарь, склеивая пары токенов в один.

WordPiece токенизация

Корпус: ["вол", "стол", "стул", "ствол"]

- Стартовый словарь – все возможные буквы.

Словарь: ["в", "о", "л", "с", "т", "у"]

- Итеративно увеличиваем словарь, склеивая пары токенов в один.

- Для каждой пары токенов (t_i, t_j) считаем скор $\frac{p(\overline{t_i t_j})}{p(t_i)p(t_j)} = \frac{N(\overline{t_i t_j})}{N(t_i)N(t_j)}$

- Склеиваем пару с наибольшим скором.

Таким образом, мы выбираем ту пару, добавление которой больше всего увеличивает **правдоподобие** выборки

WordPiece: алгоритм

Шаг 1

Пары	score
с, т	$3/(3 \cdot 3) = 1/3$
о, л	$3/(3 \cdot 4) = 1/4$
в, о	$2/(2 \cdot 3) = 1/3$

$$score(t_i, t_j) = \frac{N(\overline{t_i t_j})}{N(t_i)N(t_j)}$$

Корпус: ["вол", "стол", "стул", "ствол"]

Словарь: ["в", "о", "л", "с", "т", "у", "ст"]

Добавляем "ст"



WordPiece: алгоритм

Шаг 2

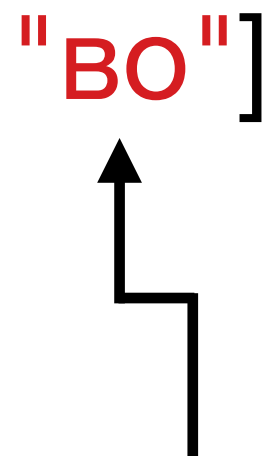
Пары	score
о, л	$3/(3 \cdot 4) = 1/4$
в, о	$2/(2 \cdot 3) = 1/3$
у, л	$1/(1 \cdot 3) = 1/3$

$$score(t_i, t_j) = \frac{N(\overline{t_i t_j})}{N(t_i)N(t_j)}$$

Корпус: ["вол", "стол", "стул", "ствол"]

Словарь: ["в", "о", "л", "с", "т", "у", "ст",
"во"]

Добавляем "во"



WordPiece: алгоритм

Шаг 3

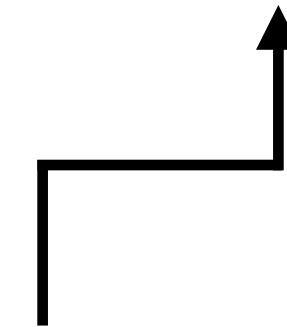
Пары	score
во, л	$2/(2 \cdot 4) = 1/4$
о, л	$3/(3 \cdot 4) = 1/4$
у, л	$1/(1 \cdot 3) = 1/3$

$$score(t_i, t_j) = \frac{N(\overline{t_i t_j})}{N(t_i)N(t_j)}$$

Корпус: ["вол", "стол", "стул", "ствол"]

Словарь: ["в", "о", "л", "с", "т", "у", "ст",
"во", "ул"]

Добавляем "ул"



WordPiece: алгоритм

Шаг 4

Пары	score
т, ул	$1/(3 \cdot 1) = 1/3$
во, л	$2/(2 \cdot 4) = 1/4$
о, л	$3/(3 \cdot 4) = 1/4$

$$score(t_i, t_j) = \frac{N(\overline{t_i t_j})}{N(t_i)N(t_j)}$$

Корпус: ["вол", "стол", "стул", "ствол"]

Словарь: ["в", "о", "л", "с", "т", "у", "ст",
"во", "ул", "тул"]

Добавляем "тул" 

WordPiece: алгоритм

Корпус: ["вол", "стол", "стул", "ствол"]

Словарь: ["в", "о", "л", "с", "т", "у", "ст",
"во", "ул", "тул"]

Останавливаемся, когда словарь достиг нужного размера.

Unigram

- Unigram работает в противоположную сторону относительно BPE и WordPiece
- Исходный словарь состоит из всех возможных токенов
- На каждом шаге удаляем один токен

Unigram: идея

- Unigram (1-грамм) модель считает, что все токены независимы

$$p(x) = p(x_1, \dots, x_m) = \prod_{i=1}^m p(x_i) \quad p(x_i) = \frac{N(x_i)}{\sum_{t \in V} N(t)}$$

- Мы хотим максимизировать правдоподобие корпуса текстов
- Будем **удалять** из словаря те **токены**, которые **минимально уменьшают правдоподобие** Unigram модели

Unigram: идея

- Unigram (1-грамм) модель считает, что все токены независимы

$$p(x) = p(x_1, \dots, x_m) = \prod_{i=1}^m p(x_i) \quad p(x_i) = \frac{N(x_i)}{\sum_{t \in V} N(t)}$$

- Мы хотим максимизировать правдоподобие корпуса текстов
- Будем **удалять** из словаря те **токены**, которые **минимально уменьшают правдоподобие** Unigram модели
- Остановимся, когда размер словаря достигнет нужного размера

Unigram: алгоритм

Корпус: ["вол", "стол", "кол"]

Словарь: ["в", "о", "л", "с", "т", "к", "во", "ол", "ст", "то", "ко", "вол", "сто", "тол"]

Токен	в	о	л	с	т	к	во	ол	ст	то	ко	вол	сто	тол
Встречаемость	1	3	3	1	1	1	1	3	1	1	1	1	1	1

Всего: 20

Unigram: алгоритм

Корпус: ["вол", "стол", "кол"]

Словарь: ["в", "о", "л", "с", "т", "к", "во", "ол", "ст", "то", "ко", "вол", "сто", "тол"]

Токен	в	о	л	с	т	к	во	ол	ст	то	ко	вол	сто	тол
Встречаемость	1	3	3	1	1	1	1	3	1	1	1	1	1	1

Всего: 20

$$p(\text{"стол"}) \begin{cases} p(\text{"сто"}, \text{"л"}) = 1/20 \cdot 3/20 \\ p(\text{"с"}, \text{"тол"}) = 1/20 \cdot 1/20 \end{cases}$$

Удаление "с" и "тол" не уменьшит вероятность, значит их можно выбросить.

Unigram: алгоритм

Корпус: ["вол", "стол", "кол"]

Словарь: ["в", "о", "л", "т", "к", "во", "ол", "ст", "то", "ко", "вол", "сто"]

Токен	в	о	л	т	к	во	ол	ст	то	ко	вол	сто
Встречаемость	1	3	3	1	1	1	3	1	1	1	1	1

Всего: 18

Unigram: алгоритм

Корпус: ["вол", "стол", "кол"]

Словарь: ["в", "о", "л", "т", "к", "во", "ол", "ст", "то", "ко", "вол", "сто"]

Токен	в	о	л	т	к	во	ол	ст	то	ко	вол	сто
Встречаемость	1	3	3	1	1	1	3	1	1	1	1	1

Всего: 18

Что поменяется, если удалить "л"?

$$p(\text{"вол"}) \begin{cases} p(\text{"во"}, \text{"л"}) = 1/18 \cdot 3/18 \\ p(\text{"в"}, \text{"ол"}) = 1/18 \cdot 3/18 \end{cases}$$

$$p(\text{"кол"}) \begin{cases} p(\text{"ко"}, \text{"л"}) = 1/18 \cdot 3/18 \\ p(\text{"к"}, \text{"ол"}) = 1/18 \cdot 3/18 \end{cases}$$

$$p(\text{"стол"}) \begin{cases} p(\text{"сто"}, \text{"л"}) = 1/18 \cdot 3/18 \\ p(\text{"ст"}, \text{"ол"}) = 1/18 \cdot 3/18 \end{cases}$$

Ничего не меняется => удаляем

Метрики качества генерации

- Статистические метрики
- Перплексия

Статистические метрики

Необходимо измерить, насколько текст похож на человеческий

Идея:

- Посчитаем различные статистики сгенерированных текстов.
- Сравним их с текстами из корпуса.

Статистические метрики

Модель **повторяет** одно и то же или генерирует **разные** тексты?

Обучающий корпус

Вот дом,
Который построил Джек.
А это пшеница,
Которая в тёмном чулане хранится
В доме,
Который построил Джек.

Генерация

Дом, который построил Джек.
Дом, который построил Джек.
Дом, который построил Джек.

Статистические метрики

Модель **повторяет** одно и то же или генерирует **разные** тексты?

Обучающий корпус

Вот дом,
Который построил Джек.
А это пшеница,
Которая в тёмном чулане хранится
В доме,
Который построил Джек.

Генерация

Дом, который построил Джек.
Дом, который построил Джек.
Дом, который построил Джек.

Отношение числа уникальных n-грамм к
числу всех n-грамм.

$$\frac{|\text{unique 2-gram}|}{|\text{all 2-gram}|} = \frac{3}{9}$$

Чем выше, тем больше разнообразие

Статистические метрики

Модель **повторяет** одно и то же или генерирует **разные** тексты?

Обучающий корпус

Вот дом,
Который построил Джек.
А это пшеница,
Которая в тёмном чулане хранится
В доме,
Который построил Джек.

Генерация

Дом,	который	построил	Джек.
Дом,	который	построил	Джек.
Дом,	который	построил	Джек.

Число уникальных токенов среди k токенов

$$\frac{|\text{unique tokens}|}{10} = \frac{4}{10}$$

Чем выше, тем больше разнообразие

Статистические метрики

Что если модель просто **запомнила** обучающую выборку?

Обучающий корпус

Вот дом,
Который построил Джек.
А это пшеница,
Которая в тёмном чулане хранится
В доме,
Который построил Джек.

Генерация

Дом, который построил Джек.
Дом, который построил Джек.
Дом, который построил Джек.

Статистические метрики

Что если модель просто **запомнила** обучающую выборку?

Обучающий корпус

Вот дом,
Который построил Джек.
А это пшеница,
Которая в тёмном чулане хранится
В доме,
Который построил Джек.

Генерация

Дом, который построил Джек.
Дом, который построил Джек.
Дом, который построил Джек.

Доля совпадения сгенерированных n-грамм с
n-граммами обучающего корпуса

$$\frac{|\{\text{train 2-grams}\} \cap \{\text{gen 2-grams}\}|}{|\{\text{gen 2-grams}\}|} = 1$$

Чем меньше, тем меньше запоминание

Статистические метрики

Модель 1

разнообразие: 0.5
запоминание: 0.5

Модель 2

разнообразие: 0.3
запоминание: 0.4

Как сравнивать?

Необходимо ввести единую метрику

Перплексия

- Посчитаем вероятности текстов из тестовой выборки.
- Чем лучше модель, тем больше должна быть вероятность.

Обучающий корпус

Вот дом,
Который построил Джек.
А это пшеница,
Которая в тёмном чулане хранится
В доме,
Который построил Джек.

Тестовый корпус

А это весёлая птица-синица,
Которая часто ворует пшеницу,
Которая в тёмном чулане хранится
В доме,
Который построил Джек.

$p(\text{“А это весёлая птица-синица, ...”}) = ?$

Перплексия

$$PPL(x) = p(x_1, \dots, x_m)^{-\frac{1}{m}}$$

- Чем меньше, тем лучше.
- $\frac{1}{m}$ нужно для нормализации по длине текста.

Перплексия была придумана еще в 1977 году, но она до сих пор популярна

Перплексия сводится к кросс-энтропии

$$PPL(x) = p(x_1, \dots, x_m)^{-\frac{1}{m}}$$

$$CE(x) = -\frac{1}{m} \sum_{i=1}^m \log p(x_i | x_{i-1}, \dots, x_1)$$

$$\begin{aligned} \exp(CE(x)) &= \exp\left(\sum_{i=1}^m \log p(x_i | x_{i-1}, \dots, x_1)\right)^{-\frac{1}{m}} = \left(\prod_{i=1}^m p(x_i | x_{i-1}, \dots, x_1)\right)^{-\frac{1}{m}} = \\ &= p(x_1, \dots, x_m)^{-\frac{1}{m}} = PPL(x) \end{aligned}$$