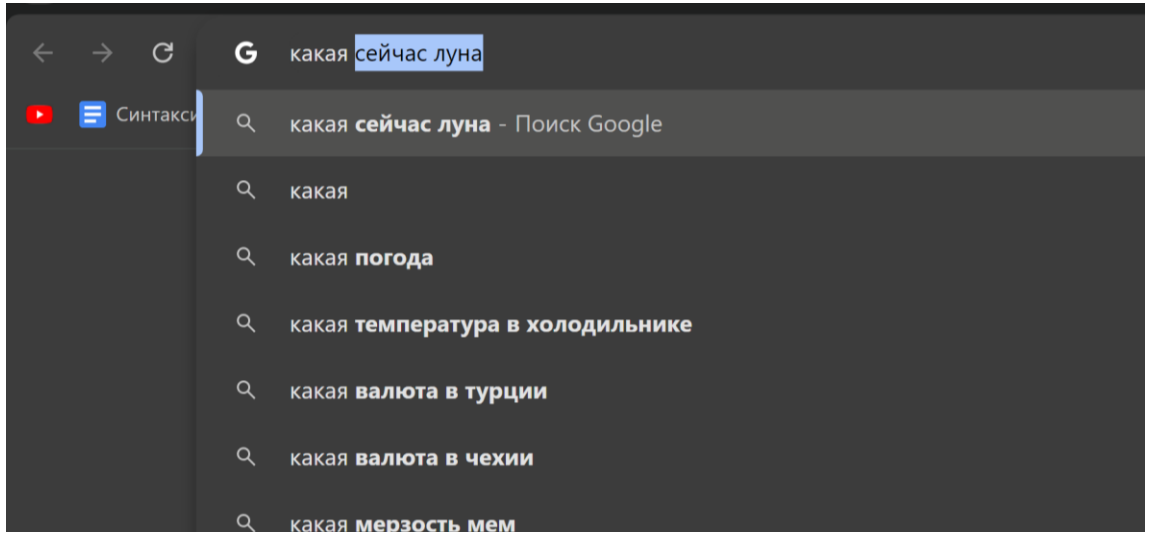
Предсказание следующего слова в контексте...



Предсказание занятие неблагодарное ...

Предсказание занятие неблагодарное …

- Но!
  Лингвистическое – это относительно просто!!!

  Так, первая страница из Журавского по нашей теме предлагает нам предположить, какое слово будет следующим в предложении…

  The water of Walden Pond is so beautifully …

about predicting something that seems much easier, like the next word someone is going to say? What word, for example, is likely to follow

```
The water of Walden Pond is so beautifully ...
```

You might conclude that a likely word is `blue`, or `green`, or `clear`, but probably not `refrigerator` nor `this`. In this chapter we formalize this intuition by introducing

~~Предсказание следующего слова в контексте...~~

Оценка вероятности слова в контексте

Александр Орлов

Для чего это нужно?

Для чего это нужно?

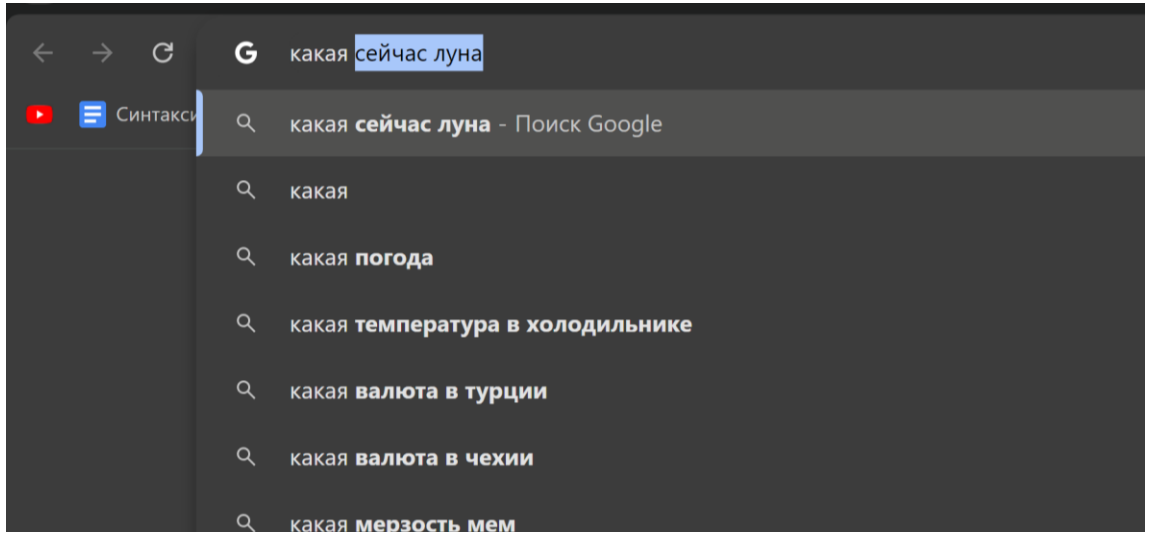
- Спеллчекеры и автоматическое исправление ошибок

  Their are two midterms –> There are two midterms

  Everything has improve → Everything has improved.

- Автозаполнение:
  - Augmentative and Alternative Communication (AAC) – люди, которые не могут ни писать, ни использовать моторику, общаются на скорости только 10% от «здорового» населения (Trnka et al. 2007, AAC Kane et al. 2017)
- Часть более сложных систем:
  - поиск
  - Распознование речи
  - NLG
  - перевод
  - И т.п.

Предсказание следующего слова в контексте...



Как это делать?

Словарь

- N-gramm (bigram, trigram, -);
- Вероятность/probability;
- P(w|h), the probability of a word w given some history h;

«Идеальный» вариант

$$P(\texttt{blue}|\texttt{The water of Walden Pond is so beautifully}) = \frac{C(\texttt{The water of Walden Pond is so beautifully blue})}{C(\texttt{The water of Walden Pond is so beautifully})}$$

В чем, соответсвенно, проблема?

«Идеальный» вариант

$$P(X_1 \ldots X_n) = P(X_1)P(X_2|X_1)P(X_3|X_{1:2}) \ldots P(X_n|X_{1:n-1})$$
$$= \prod_{k=1}^{n} P(X_k|X_{1:k-1})$$

Математическое правило подсчета вероятности последовательности

Реальный вариант

Мы можем приблизительно оценить вероятность слова в данной последовательности, подсчитав не все предыдущие вероятности, а только часть из них.

Биграммы, к примеру, аппроксимируют вероятность одного слова, исходя из предыдущего слова.

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$$

⇒ **Цепи Маркова**

⇒ *Markov assumption*: Можем предсказать вероятность слова, смотря только на n слов назад;

Реальный вариант

Maximal Likelihood Estimation (MLE) /Метод максимального правдоподобия:

(1) Получение данных из корпуса

(2) Нормализация (чтобы все вероятности были между 0 и 1)

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

Реальный вариант

Maximal Likelihood Estimation (MLE) /Метод максимального правдоподобия:

(1) Получение данных из корпуса

(2) Нормализация (чтобы все вероятности были между 0 и 1)

Подсчитать вероятность слова Y после Z = подсчитать количество употреблений биграммы ZY и разделить её на количество всех биграмм, начинающихся с Z в данном корпусе

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

<s>

<s>

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$P(\text{I}|\text{<s>}) = \frac{2}{3} = 0.67$ $P(\text{Sam}|\text{<s>}) = \frac{1}{3} = 0.33$ $P(\text{am}|\text{I}) = \frac{2}{3} = 0.67$

$P(\text{</s>}|\text{Sam}) = \frac{1}{2} = 0.5$ $P(\text{Sam}|\text{am}) = \frac{1}{2} = 0.5$ $P(\text{do}|\text{I}) = \frac{1}{3} = 0.33$

Зачем нужен символ <s>?

Задачка.

Suppose we didn't use the end-symbol </s>. Train an unsmoothed bigram grammar on the following training corpus without using the end-symbol </s>:

```
<s> a b
<s> b b
<s> b a
<s> a a
```

Demonstrate that your bigram model does not assign a single probability distribution across all sentence lengths by showing that the sum of the probability of the four possible 2 word sentences over the alphabet {a,b} is 1.0, and the sum of the probability of all possible 3 word sentences over the alphabet {a,b} is also 1.0.

Большие модели

- Log:

Большие модели

- Log: так как вероятности по определению меньше единицы, то переумножпение их много раз, даёт нам … Добавление логорифма = умножение в линейном пространстве. Под log обычно подразумевается log (ln)

Оценка вероятности предложения

- P(“Mary has a little lamb .”) = P(Mary | <s> <s>) × P(had | <s>Mary) × P(a | Mary, has) × P(little | has, a) × P(lamb | a, little) × P(. | little, lamb)

- Заметте, что на триграммах мы можем использовать в.т.ч <s><s>

Сглаживание/Smoothing

- Что, если какой-то последовательности не будет в корпусе? Но она будет в тестовом датасете…

- Mary has a little lamb
- Например:

  count (a, little, lamb) = N, но count(the, little, lamb) = 0

Немного терминологии

- *Data sparsity*: даже хороший репрезентативный корпус не позволит идеально оценить вероятности

- *Zeros / OOV* – out of vocabulary words могут встретиться в тестовом корпусе

- Какие решения вы можете предложить?

Laplace Smoothing

(Add-one smoothing)

Laplace Smoothing

- Добавить один к каждому подсчету (одно вхождение станет двумя, два – тремя и т д)

$$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$$

- Что произошло с формулой? Что такое V?

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Figure 3.2 Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray.

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Figure 3.2 Bigram pr
of 9332 sentences. Zero

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Figure 3.7 Add-one smoothed bigram probabilities for eight of the words (out of $V = 1446$) in the BeRP corpus of 9332 sentences. Previously-zero probabilities are in gray.

Laplace Сглаживание

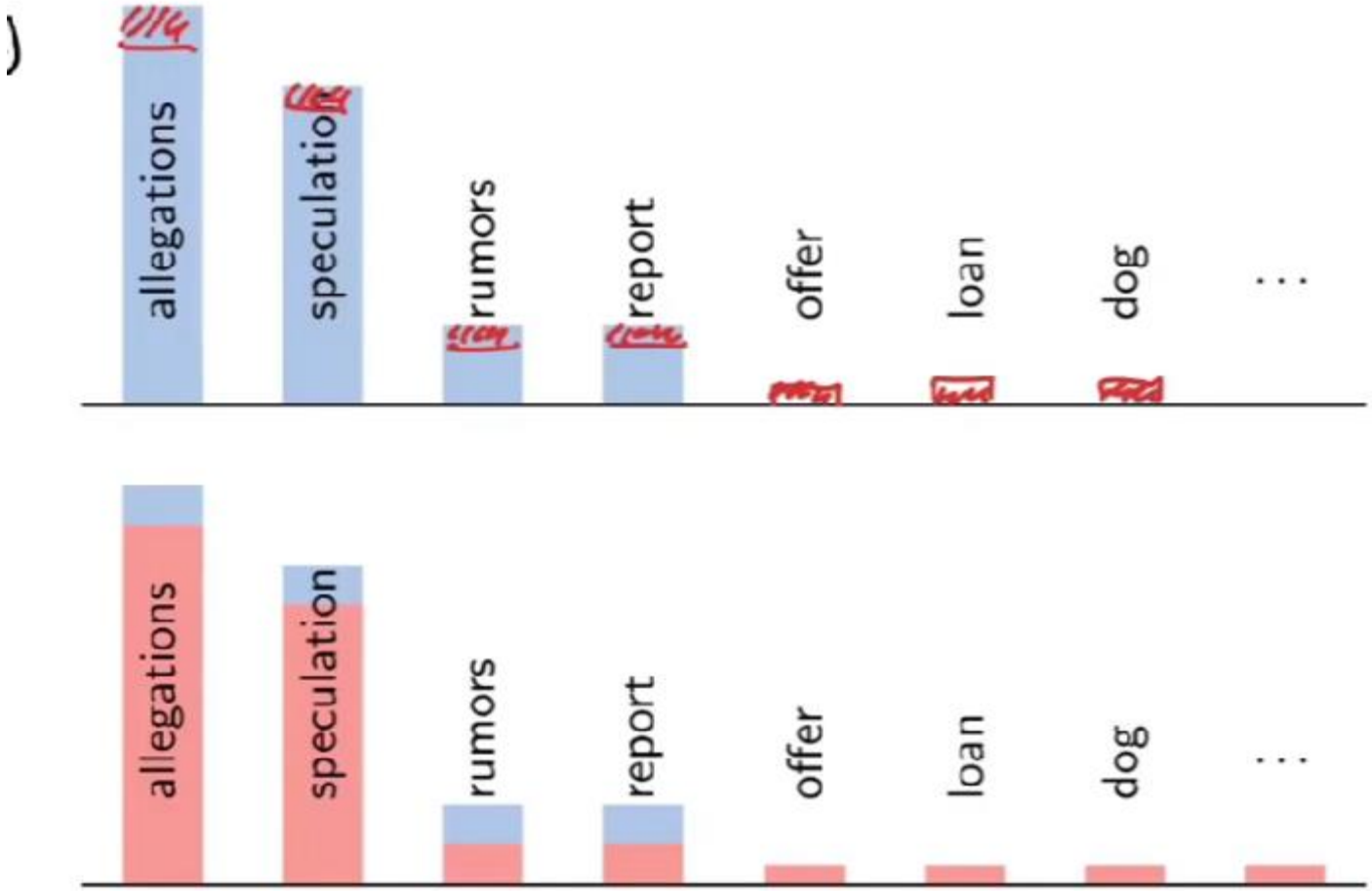
- Любое сглаживание всегда снижает вероятности самых частотных значений и передает их более «маргинальным»
- Laplace smoothing в силу проблем с точностью результатов почти не используется в этой задаче, однако применяется в др. Задачих типа Text Classification

Add-k smoothing

- + 0.01, +0.5
- Как это повлияет на данные?
- Как выбирать K?

Linear Interpolation

Robin Hood: Steal from the rich and give to the poor.



There are different ways of moving probability mass.

Example: Drinking beer in Scotland

We have a corpus with the following trigram counts:

`Scottish beer was:`	20
`Scottish beer can:`	16
`Scottish beer awards:`	7
`Scottish beer brands:`	3

The following trigrams are never seen:

`Scottish beer drinkers:`	0
`Scottish beer eaters:`	0

How would the following estimates differ, given this data? Is this reasonable?

$$P_{\text{MLE}}(\texttt{drinkers}|\texttt{Scottish}, \texttt{beer})$$
$$P_{\text{MLE}}(\texttt{eaters}|\texttt{Scottish}, \texttt{beer})$$

But do you think the two bigrams below will have the same count?

`beer drinkers`
`beer eaters`

Example: Drinking beer in Scotland

We have a corpus with the following trigram counts:

`Scottish beer was:`	20
`Scottish beer can:`	16
`Scottish beer awards:`	7
`Scottish beer brands:`	3

The following trigrams are never seen:

`Scottish beer drinkers:`	0
`Scottish beer eaters:`	0

How would the following estimates differ, given this data? Is this reasonable?

$$P_{\text{MLE}}(\texttt{drinkers}|\texttt{Scottish}, \texttt{beer}) \simeq 0$$
$$P_{\text{MLE}}(\texttt{eaters}|\texttt{Scottish}, \texttt{beer}) \simeq 0$$

But do you think the two bigrams below will have the same count?

`beer drinkers` > 0
`beer eaters` ≈ 0

Interpolation

- Если мы пытаемся посчитать $P(w_n|w_{n-2}w_{n-1})$, но у нас нет примеров такой триграммы, мы можем опираться на вероятность более малого контекста

$$P_{\text{int}}(w_t|w_{t-2},w_{t-1}) = \lambda_1 P_1(w_t) + \lambda_2 P_2(w_t|w_{t-1}) + \lambda_3 P_3(w_t|w_{t-2},w_{t-1})$$

- NB! Все эти лямбды должны суммироваться до единицы.

Как устанавливаются эти значения λ?

Held-out corpus (валидационный корпус): корпус, который не входит ни в training, ни в test датасеты.

(1) Определяем вероятности n-gram (по training корпусу);

(2) Определяем значения λ, которые при включении в формулу, дают нам наибольшие вероятность для предложений, входящих в валидационный корпус.

Ряд достаточно сложных алгоритмов (Jelinek and Mercer, 1980).

Когда еще может помочь сглаживание?

- Др. понимание интерполяции:
  Обращение к данным других корпусов за помощью

$$P_{\text{int}}(w_t|w_{t-2},w_{t-1}) = \lambda_1 P_1(w_t|w_{t-2},w_{t-1}) + \lambda_2 P_2(w_t|w_{t-2},w_{t-1})$$

Сценарий: мы работаем на корпусе южно-африканского английского, но данных так мало, что мы хотим прибавить туда данные американского английского, с каким-то коэффициентом

Добавить к данным из нашего корпуса данные из чужого корпуса

- Автозаполнение
  - Соединение общих вероятностей возникновения одного слова после другого с вероятностями, возникшими в результате анализа поведения пользователя.

Stupid Back-off

Схождение на один уровень вниз

Если мы не находим информации среди триграмм, мы смотрим на биграммы, если не видим в биграммах, то смотрим в монограммах

Этот метод называется stupid back-off, потому что он не учитывает поправку на схождение вниз и надёжность

$$P(lamb \mid the, little) = 0$$

> пробуем биграммы

$$P(lamb \mid the)$$

> если снова 0, то пробуем униграммы

$$P(lamb)$$

- В чем математические проблемы такого метода?

Discounting

Back-off N-gram model:

$$P_{\text{BO}}(w_t|w_{t-N+1:t-1}) = \begin{cases} P_d(w_t|w_{t-N+1:t-1}) \\ \qquad\qquad \text{if } C(w_{t-N+1:t}) > 0 \\ \alpha(w_{t-N+1:t}) P_{\text{BO}}(w_t|w_{t-N+2:t-1}) \\ \qquad\qquad \text{if } C(w_{t-N+1:t}) = 0 \end{cases}$$

where

- $P_d(w_t|w_{t-N+1:t-1})$ is some discounted N-gram model.
- The back-off weights $\alpha(w_{t-N+1:t})$ are such that the probability sum to 1.

- ➔ Jurafsky (2)

Kneser-Nay smoothing

Kneser-Ney smoothing

In the Europarl corpus:

- `York` occurs 477 times. As frequent as `foods`, `indicates` and `providers`.
- This leads to a relatively high unigram estimate for $P(\texttt{York})$.
- But, `York` almost always follows `New` (473 times).
- So in unseen bigram contexts, `York` should have a low probability. But when we back-off or interpolate, the unigram probability for `York` will be high! I.e. the $P(\texttt{York}|\texttt{<not New>})$ is too high.

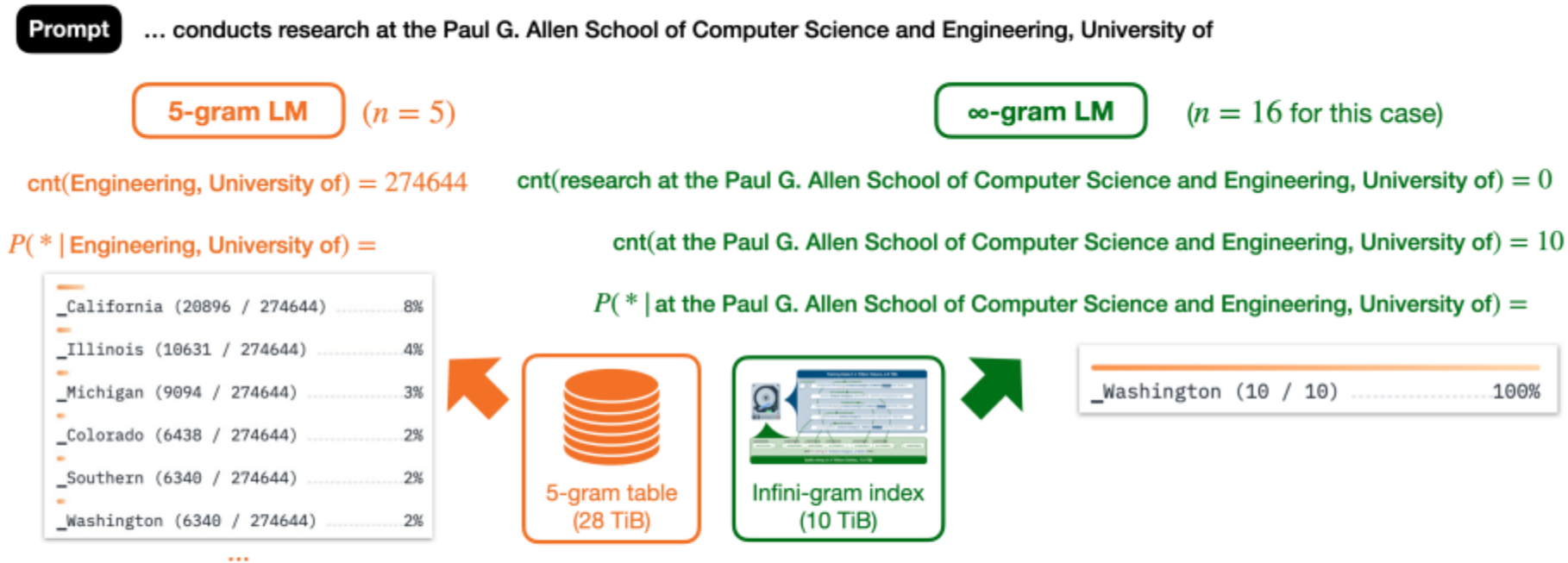


Figure 1: An example where a 5-gram LM gives an incorrect prediction but the ∞-gram gives the correct prediction by using the longest suffix of the prompt that has a non-zero count in the corpus. The counting and distribution estimate in ∞-gram LM are powered by our infini-gram engine.

Статья

- Прочитайте стр. 1-6 включительно
- Пропуская цифры и технические детали про байты и биты

https://arxiv.org/pdf/2401.17377

- Почему n-grammы обычно ограничиваются небольшими числами типа пяти? (2)
- Как устроен Suffix array? (3)
- Что такое data (de-)contamination?
- Какое у автоpo:к получилось качество? (4)
- Какое количество слов в цепочке минимально достаточно для достижения хорошего качества? (4)

Тестирование

оценки

- В целом оценка любой модели NLP:

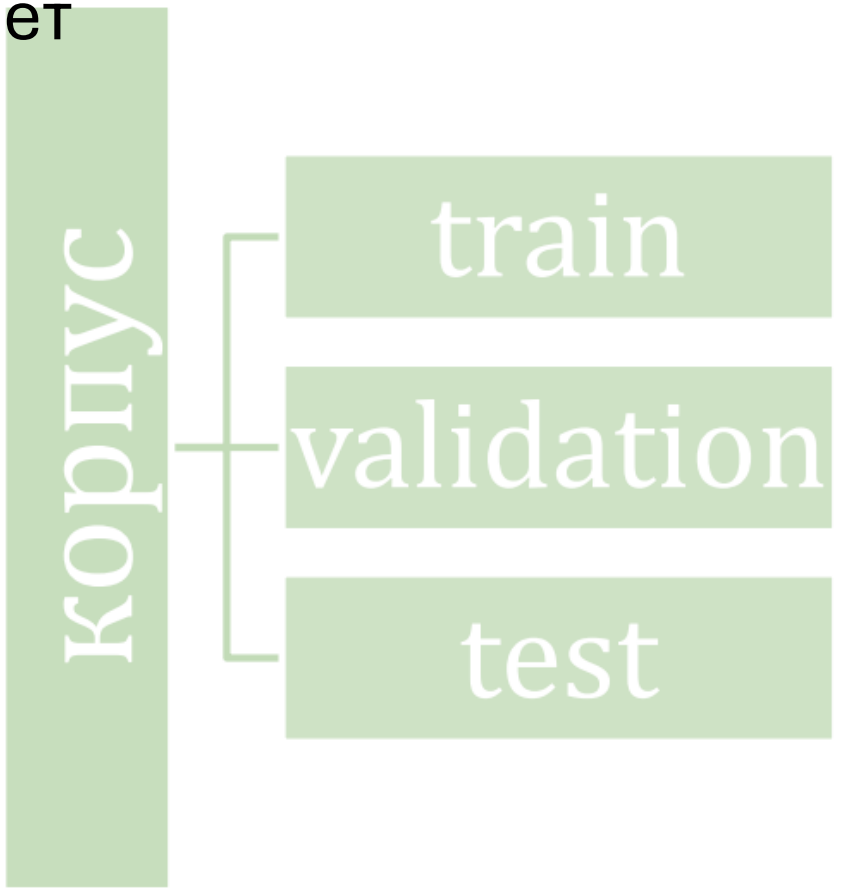
- **внешняя (extrinsic)**

насколько улучшает качество работы какой-нибудь системы NLP (информационный поиск, машинный перевод, чат боты и т.д.)

- **внутренняя (intrinsic)**

специальные метрики для конкретной задач

Валидационный корпус – в т ч для «подгонки» модели, если мы много раз тестируем модель на тестовых данных, а потом ее фиксим, то мы можем невольно подстроить модель под этот датасет



Perplexity

- Причина в том, что вероятность тестового набора (или любой последовательности) зависит от количества слов или токенов в нем; вероятность тестового набора становится меньше, чем длиннее текст.
- Мы бы предпочли метрику, которая является пословной, нормализованной по длине, чтобы мымогли сравнивать тексты разной длины.
- Perplexity (иногда сокращенно PP или PPL) нормализованна по количеству слов (или токенов).

Perplexity

$$\begin{aligned}\text{perplexity}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}\end{aligned}$$

- N – количество слов

NB! Чем ниже метрика, тем лучше

Почитаем 4 -6

Анализ n-граммных моделей

Достоинства:

- простые, быстро обучаются
- не требуют размеченных данных (возможно, хороший корпус для оценки)

Недостатки:

- не моделируют дистантные отношения (согласование, управление, анафора, … отделяемые приставки и пр.)
- не учитывают морфологию и т.п.
- то есть не обеспечивают связность (fluency) текста