

# Языковые модели

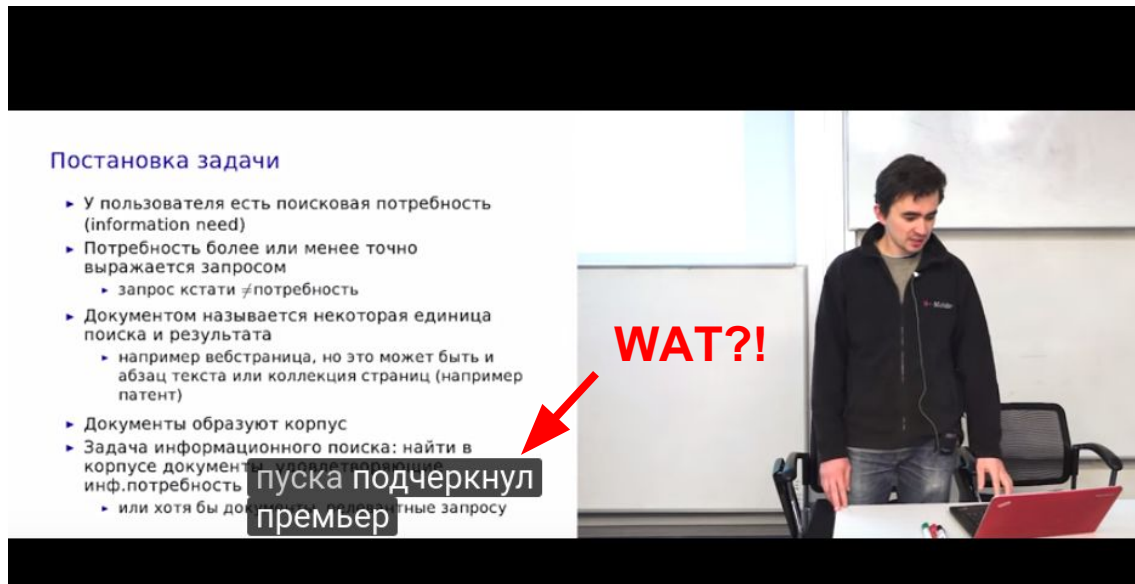
“просто предсказываем следующее слово?”

Антон Алексеев  
ПОМИ РАН, CSCenter, НИУ ИТМО

ЛШ, Дубна, июль 2018

# Мотивация

Во многих задачах бывает нужно проверить “естественность”,  
“правильность”, короче, оценить **вероятность** последовательности слов



Постановка задачи

- ▶ У пользователя есть поисковая потребность (information need)
- ▶ Потребность более или менее точно выражается запросом
  - ▶ запрос к стати  $\neq$  потребность
- ▶ Документом называется некоторая единица поиска и результата
  - ▶ например вебстраница, но это может быть и абзац текста или коллекция страниц (например патент)
- ▶ Документы образуют корпус
- ▶ Задача информационного поиска: найти в корпусе документ, удовлетворяющий инф.потребность
  - ▶ или хотя бы документ, релевантный запросу

WAT?!

пуска подчеркнул премьер

На деле Дмитрий говорит:

*...поиск по патентам, например.*

<https://youtu.be/APcwsxUpGrQ?t=1m38s>

# Мотивация

- **Распознавание речи / машинный перевод / исправление ошибок и опечаток / augmentative communication**  
*декодировали/предсказали “несколько вариантов” реплик, надо выбрать наиболее вероятную с точки зрения языка*
- **Информационный поиск**  
ранжирование: для каждого документа  $d$  строим свою “модель языка” и упорядочиваем документы по  $P(q|d)$
- **Fun!** Генераторы текста “в духе данного корпуса”

# План занятия

1. Интуиция
  2. N-граммное моделирование
  3. Оценка качества языковых моделей
  4. Нули и сглаживание
    - а. Сглаживание Кнезера-Нея
- Библиотеки
  - Наборы данных

# Интуиция

- ▶ Итак, **языковая модель** позволяет вычислить вероятность любой последовательности слов (альтернативная формулировка — вычислить вероятность очередного слова).
- ▶ Как оценить вероятность последовательности «*Всё смешалось в доме...*»?
- ▶ Прибегнем к формуле условной вероятности

# Интуиция: напоминание

- ▶ Определение условной вероятности

$$P(Y|X) = \frac{P(X, Y)}{P(X)} \Rightarrow P(X, Y) = P(Y|X)P(X)$$

- ▶ Chain rule для большего числа переменных:

$$P(x_1 x_2 \dots x_n) = P(x_n | x_1 \dots x_{n-1}) \dots p(x_2 | x_1) p(x_1)$$

- ▶ Выходит, можем легко посчитать?

$$P(x_i | x_1 \dots x_{i-1}) = \frac{\text{Count}(x_1 \dots x_{i-1} x_i)}{\text{Count}(x_1 \dots x_{i-1})}$$

\* Здесь и далее  $\text{Count}(\dots)$  - это то же, что  $C(\dots)$  и  $c(\dots)$

# Интуиция: напоминание

- ▶ Определение условной вероятности

$$P(Y|X) = \frac{P(X, Y)}{P(X)} \Rightarrow P(X, Y) = P(Y|X)P(X)$$

- ▶ Chain rule для большего числа переменных:

$$P(x_1 x_2 \dots x_n) = P(x_n | x_1 \dots x_{n-1}) \dots p(x_2 | x_1) p(x_1)$$

(обратите внимание, что мы везде вычисляем вероятность очередного слова!)

- ▶ Выходит, можем всё легко посчитать?

$$P(x_i | x_1 \dots x_{i-1}) = \frac{\text{Count}(x_1 \dots x_{i-1} x_i)}{\text{Count}(x_1 \dots x_{i-1})}$$

$$P(\text{happy families are all}) = P(\text{all} | \text{happy families are}) \times \\ \times P(\text{are} | \text{happy families}) \times P(\text{families} | \text{happy}) \times P(\text{happy})$$

# Интуиция: напоминание

- ▶ Определение условной вероятности

$$P(Y|X) = \frac{P(X, Y)}{P(X)} \Rightarrow P(X, Y) = P(Y|X)P(X)$$

- ▶ Chain rule для большего числа переменных:

$$P(x_1 x_2 \dots x_n) = P(x_n | x_1 \dots x_{n-1}) \dots p(x_2 | x_1) p(x_1)$$

- ▶ Выходит, можем легко посчитать?

$$P(x_i | x_1 \dots x_{i-1}) = \frac{\text{Count}(x_1 \dots x_{i-1} x_i)}{\text{Count}(x_1 \dots x_{i-1})}$$

(нет! т.к. отдельные длинные цепочки редки)



## Что делать?

- ▶ На помощь приходит допущение: текст обладает марковским свойством

$$P(x_i | x_1 \dots x_{i-1}) = P(x_i | x_i - K \dots x_{i-1})$$

...то есть очередное событие зависит не более, чем от  $K$  предыдущих

- ▶ Примеры:
  - ▶  $K = 0$  (униграммная модель)

$$P(\textit{happy families are all}) =$$

$$P(\textit{all}) \times P(\textit{are}) \times P(\textit{families}) \times P(\textit{happy})$$

- ▶  $K = 1$  (биграммная модель)

$$P(\textit{happy families are all}) = P(\textit{all} \mid \textit{are}) \times$$

$$\times P(\textit{are} \mid \textit{families}) \times P(\textit{families} \mid \textit{happy}) \times P(\textit{happy})$$

# План занятия

- ~~1. Интуиция~~
  2. N-граммное моделирование
  3. Оценка качества языковых моделей
  4. Нули и сглаживание
    - а. Сглаживание Кнезера-Нея
- Библиотеки
  - Наборы данных

# N-граммная модель

- ▶ Модель:

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | x_{i-N+1} \dots x_{i-1})$$

при этом важно добавлять по  $N - 1$  терму  
«начало»  $\wedge$  и «конец»  $\$$  слева и справа

- ▶ Можем оценивать вот так

$$P(x_i | x_{i-N+1} \dots x_{i-1}) = \frac{\text{Count}(x_{i-N+1} \dots x_{i-1} x_i)}{\text{Count}(x_{i-N+1} \dots x_{i-1})}$$



$$P(x_i | x_{i-1}) = \text{Count}(x_i, x_{i-1}) / \text{Count}(x_{i-1})$$

- ▶ Пример для биграмм:

$$\begin{aligned} P(\text{hello}, i, \text{love}, \text{you}) &= \\ &= P(\text{hello} | \wedge) P(i | \text{hello}) P(\text{love} | i) P(\text{you} | \text{love}) P(\$ | \text{you}) \end{aligned}$$

# План занятия

- ~~1. Интуиция~~
  - ~~2. N-граммное моделирование~~
  3. Оценка качества языковых моделей
  4. Нули и сглаживание
    - а. Сглаживание Кнезера-Нея
- Библиотеки
  - Наборы данных

# Оценка качества моделей

- **Внешняя**

Проверка путём встраивания модели в способ решения некой полезной задачи (machine translation, spelling correction, ...).

Если есть прирост целевой и “нужной для дела” метрики (*время переводчика, потраченное на корректировку, количество кликов на предложенный вариант исправления, заработанные деньги, наконец*), то модель **стала лучше**

- **Внутренняя**

~~Оценка для бедных~~ - для случая, когда проверять полезность напрямую долго или дорого, или не хочется привязываться к конкретной задаче, если модель в какой-то степени универсальна; тоже *какая-то метрика (см. далее)*, которая скажет нам, насколько “хороша” модель

# Оценка качества моделей

- **Внешняя**

Проверка путём встраивания модели в способ решения некой полезной задачи (machine translation, spelling correction, ...).

Если есть прирост целевой и “важной для дела” метрики (*время переводчика, потраченное на корректировку, количество ошибок на предложенный вариант исправления, заработанные деньги, наконец*), то модель **стала лучше**

- **Внутренняя**

~~Оценка для бедных~~ - для случая, когда проверять полезность напрямую долго или дорого, или не хочется привязываться к конкретной задаче, если модель в какой-то степени универсальна; тоже *какая-то метрика (см. далее)*, которая скажет нам, насколько “хороша” модель

НЕ ГОВОРИМ ОБ ЭТОМ,  
т.к. совсем другая история

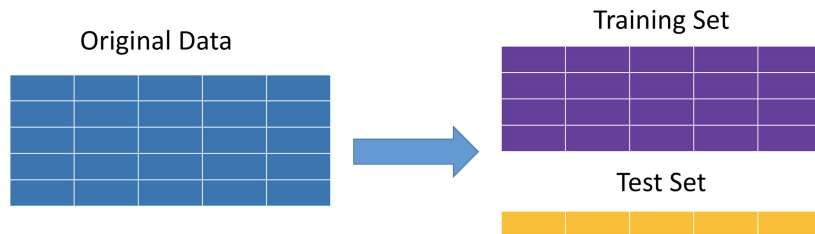
# Оценка качества

У нас есть данные, у нас есть метрика

Делим на

- train set (обучающая выборка; будем настраивать параметры модели) и
- test set (тестовая выборка; оценка качества обученной модели)

И верим в то, что это “семплы из одного и того же распределения”  
(иначе мы и теоретически не сможем обучиться)



# Оценка качества

## Смертный грех №1

Проникновение данных из test в train  
(мы теряем обобщающую способность и валидность оценок)

## Смертный грех №2

Донастройка параметров по test set  
(пример: на train посчитали частоты и, глядя на test, подобрали гиперпараметры или какие-нибудь масштабирующие коэффициенты)

**Но как тогда их настраивать? Идеи?**





# Оценка качества

<b>TRAIN</b>	<b>DEV</b>	<b>TEST</b>
--------------	------------	-------------

1. На TRAIN - обучаем модель
2. На DEV - оцениваем качество + поглядываем, где ошибаемся + настраиваем более высокоуровневые параметры
3. На TEST - оцениванием качество вслепую, т. е. только считаем метрику качества и “не смотрим, где ошиблись”, чтобы исключить (ха-ха) подгонку

# Оценка качества модели

- ▶ Чем больше вероятность тестового текста, тем «правильнее» модель
- ▶ Перплексия — инвертированная вероятность текста, нормализованная числом слов

$$\begin{aligned} PP(W) &= P(x_1 \dots x_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(x_1 \dots x_N)}} = \\ &= \sqrt[N]{\frac{1}{\prod_{i=1}^N P(x_i | x_1 \dots x_{i-1})}} \end{aligned}$$

Очевидно, less is better.

- ▶ Кстати, любителям теории информации кое-что покажется знакомым

$$PP(W) = P(x_1 \dots x_N)^{-\frac{1}{N}} = e^{-\frac{1}{N} \sum_{i=1}^N \log P(x_i | x_1 \dots x_{i-1})}$$

# Оценка качества: пример

Обучение на 38M слов

Тестирование на 1.5M

Тексты: Wall Street

Journal

	<b>1-gram</b>	<b>2-gram</b>	<b>3-gram</b>
<b>Perplexity</b>	962	170	109

*из Martin/Jurafsky*

# План занятия

- ~~1. Интуиция~~
  - ~~2. N-граммное моделирование~~
  - ~~3. Оценка качества языковых моделей~~
  4. Нули и сглаживание
    - а. Сглаживание Кнезера-Нея
- Библиотеки
  - Наборы данных

# Рассуждение об обобщающей способности

- Нет идеального корпуса, в котором все  $n$ -граммы встречаются хотя бы один раз!
- Модель, которую мы описали:  $P(x, \dots) = 0$  текст, в котором есть хотя бы одна  $N$ -грамма, которой не было в обучающей выборке
- При этом модель должна **обобщать**, а не просто описывать “что встретилось в обучающей выборке, а что нет”

**Естественное решение -- превращать эти нули в малые величины**

- Also: если раньше не видели отдельных слов (**OOV = out**), можно заменять их меткой <UNKNOWN>/частью речи/”частотным бакетом”

## Лапласовское сглаживание (add-one smoothing)

- Представим, что любую из нграмм мы увидели в тексте ещё по одному разу, тогда пересчитываем оценки так (на примере биграмм)

$$P(w_i|w_{i-1}) = \frac{\text{Count}(w_i, w_{i-1}) + 1}{\text{Count}(w_i) + V},$$

где  $V$  не позволит вероятностям перестать давать в сумме 1. Чему оно равно?

# Лапласовское сглаживание (add-one smoothing)

- ▶ Итак,

$$P(w_i|w_{i-1}) = \frac{\text{Count}(w_i, w_{i-1}) + 1}{\text{Count}(w_i) + V}$$

- ▶ Если просуммировать по всем  $w_i$ , будет видно, что  $V$  — это мощность множества униграмм, иначе  $P$  перестанет быть вероятностью.
- ▶ Обычно работает не очень хорошо (слишком много массы переносим на нули!)
- ▶ Фикс для бедных:

$$P(w_i|w_{i-1}) = \frac{\text{Count}(w_i, w_{i-1}) + \alpha}{\text{Count}(w_i) + \alpha V}$$

## Откат (backoff) и интерполяция

- ▶ Нет «довольно молодой специалист», но есть «молодой специалист» (если нет — униграмма «специалист»)
- ▶ Можно использовать вероятность меньших n-грамм для вычисления больших с нулевой частотой. Это называется **откатом**.
- ▶ Каждую n-грамму можно рассматривать как взвешенную сумму вероятностей n-1-граммы, n-2-граммы и так далее. Это называется **интерполяцией**.

$$P(w_i | w_{i-2} w_{i-1}) = \lambda_2 P(w_i | w_{i-2} w_{i-1}) + \lambda_1 P(w_i | w_{i-1}) + \lambda_0 P(w_i)$$

$$\sum_{i=0}^N \lambda_i = 1$$

Веса  $\lambda$  подбираются на отдельном «отложенном» множестве, могут зависеть от конкретных контекстов



# Сглаживание Кнезер-Нея: идея №1

- выберем биграммы, число которых на train set равно  $k$
- посмотрим, сколько их в held out set

Увидим, что разница  $\sim$  **константа!**  
(кроме редких и там, и там биграмм)

The intuition is that since we have good estimates already for the very high counts, a small discount  $d$  won't affect them much. It will mainly modify the smaller counts, for which we don't necessarily trust the estimate anyway

Тогда запомним поправку  $d = 0.75$  для всех  
Или  $0.75$  для **2...9** и  $0.5$  для **1**

Bigram count in training set	Bigram count in heldout set
0	0.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

# Сглаживание Кнезер-Нея: идея №1

$$P_{\text{AbsoluteDiscounting}}(w_i | w_{i-1}) = \frac{\overset{\text{discounted bigram}}{c(w_{i-1}, w_i) - d}}{c(w_{i-1})} + \overset{\text{Interpolation weight}}{\lambda(w_{i-1})} \overset{\text{unigram}}{P(w)}$$

$d$  - та самая абсолютная поправка (штраф?)

# Сглаживание Кнезер-Нея: идея №2

$$P_{\text{AbsoluteDiscounting}}(w_i | w_{i-1}) = \frac{\overset{\text{discounted bigram}}{c(w_{i-1}, w_i) - d}}{c(w_{i-1})} + \overset{\text{Interpolation weight}}{\lambda(w_{i-1})} \overset{\text{unigram}}{P(w)}$$

- Зачем мы вообще интерполируем? Каких n-грамм обычно мало?
- “Хотя он умолял меня о \_\_\_\_\_”  
“чулках”? “-Петербург”? -- совсем разные, но одинаково частотные
- **Идея:** чем больше **мощность множества n-грамм**, в которых встречается слово, тем оно **полезнее для интерполяции**
- *Какой смысл рассматривать “Франциско” как замену для “дырки в данных”, если оно чаще всего идёт после “Сан”?*

# Сглаживание Кнезер-Нея: идея №2

- **Идея:** чем больше **мощность множества n-грамм**, в которых встречается слово, тем оно **полезнее для интерполяции**

$$P_{CONTINUATION}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|}$$

# Сглаживание Кнезера-Нея: итоговая формула

$$P_{\text{KN}}(w_i|w_{i-1}) = \frac{\max(C(w_{i-1}w_i) - d, 0)}{C(w_{i-1})} + \lambda(w_{i-1})P_{\text{CONTINUATION}}(w_i)$$

Лямбда здесь помогает сохранить свойства вероятности - правильно раскидывая “вес” по нграммам

$$\lambda(w_{i-1}) = \frac{d}{\sum_v C(w_{i-1}v)} |\{w : C(w_{i-1}w) > 0\}|$$

Есть рекурсивная формула для n-грамм для произвольного n (см. Martin-Jurafsky, Chapter 4)

# Итог: что лучше?


Слайды Филипа Коэна

## Evaluation

Evaluation of smoothing methods:

Perplexity for language models trained on the Europarl corpus

См. литературу



Smoothing method	bigram	trigram	4-gram
Good-Turing	96.2	62.9	59.9
Witten-Bell	97.1	63.8	60.4
Modified Kneser-Ney	95.4	61.6	58.6
Interpolated Modified Kneser-Ney	94.5	59.3	54.0

# План занятия

- ~~1. Интуиция~~
  - ~~2. N-граммное моделирование~~
  - ~~3. Оценка качества языковых моделей~~
  - ~~4. Нули и сглаживание~~
    - ~~а. Сглаживание Кнезера-Нея~~
- Библиотеки
  - Наборы данных

# Инструменты

Что-то есть в **nltk** (nltk.models; будете делать д/з -- увидите)

А вот что использует Moses (OS SMT engine)

## Language Models in Moses

The language model should be trained on a corpus that is suitable to the domain. If the although using additional training data is often beneficial.

Our decoder works with the following language models:

- the SRI language modeling toolkit, which is freely available.
- the IRST language modeling toolkit, which is freely available and open source.
- the RandLM language modeling toolkit, which is freely available and open source.
- the KenLM language modeling toolkit, which is included in Moses by default.
- the DALM language modeling toolkit, which is freely available and open source.
- the OxLM language modeling toolkit, which is freely available and open source.
- the NPLM language modeling toolkit, which is freely available and open source.



# Данные

- \*Большая коллекция текстов под вашу задачу
- Датасеты под задачи, где требуется LM, например, данные конференций WMT
- Google NGrams
- НКРЯ, OpenCorpora

йодистый	1936	95	43
йодистый	1937	133	43
йодистый	1938	82	40
йодистый	1939	75	29
йодистый	1940	125	40
йодистый	1941	108	24
йодистый	1942	9	4
йодистый	1943	11	8
йодистый	1944	25	11
йодистый	1945	42	20
йодистый	1946	83	27
йодистый	1947	164	46
йодистый	1948	103	55
йодистый	1949	100	44

## Частоты словоформ и словосочетаний

Вы можете скачать архивы с текстовыми файлами, содержащими частоты. При подсчёте учитывался регистр букв, а также знаки препинания. Общий объём корпуса – 192689044 словоформы.

Словоформы	<a href="#">zip-архив</a> (75 MB)
2-граммы	<a href="#">zip-архив</a> (15 MB)
3-граммы	<a href="#">zip-архив</a> (25 MB)
4-граммы	<a href="#">zip-архив</a> (35 MB)
5-граммы	<a href="#">zip-архив</a> (45 MB)
6-граммы	<a href="#">zip-архив</a> (55 MB)

### Частотные списки

Тип n-граммы:

- ☒ все  
☐ униграммы (1 слово)  
☐ биграммы (2 слова)  
☐ триграммы (3 слова)

Учёт регистра:

- ☒ все  
☐ с учётом  
☐ без учёта

**File format:** Each of the files below is compressed *tab*-separated data. In Version 2 each line has the following format:

ngm TAB year TAB match\_count TAB volume\_count NEWLINE

As an example, here are the 3,000,000th and 3,000,001st lines from the a file of the English 1-grams (googlebooks-eng-all-1gram-20120701-a.gz):

Тип токенов:	cumvallate	1978	335	91
	cumvallate	1979	261	91

line tells us that in 1978, the word "circumvallate" (which means d with a rampart or other fortification", in case you were wondering) | 335 times overall, in 91 distinct books of our sample.

# Очевидный недостаток подхода

как думаете?

# Очевидный недостаток подхода

- There are A LOT of n-grams!  
→ Gigantic RAM requirements!
- Recent state of the art: *Scalable Modified Kneser-Ney Language Model Estimation* by Heafield et al.:  
“Using one machine with 140 GB RAM for 2.8 days, we built an unpruned model on 126 billion tokens”

[https://www.youtube.com/watch?v=Keqep\\_PKrY8](https://www.youtube.com/watch?v=Keqep_PKrY8)