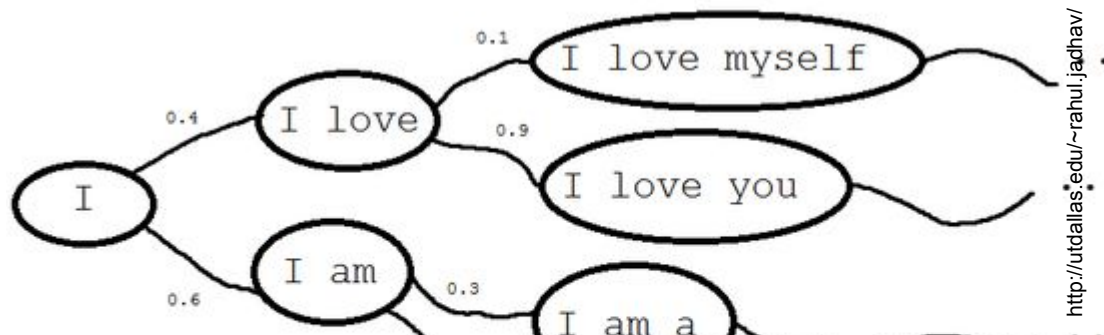


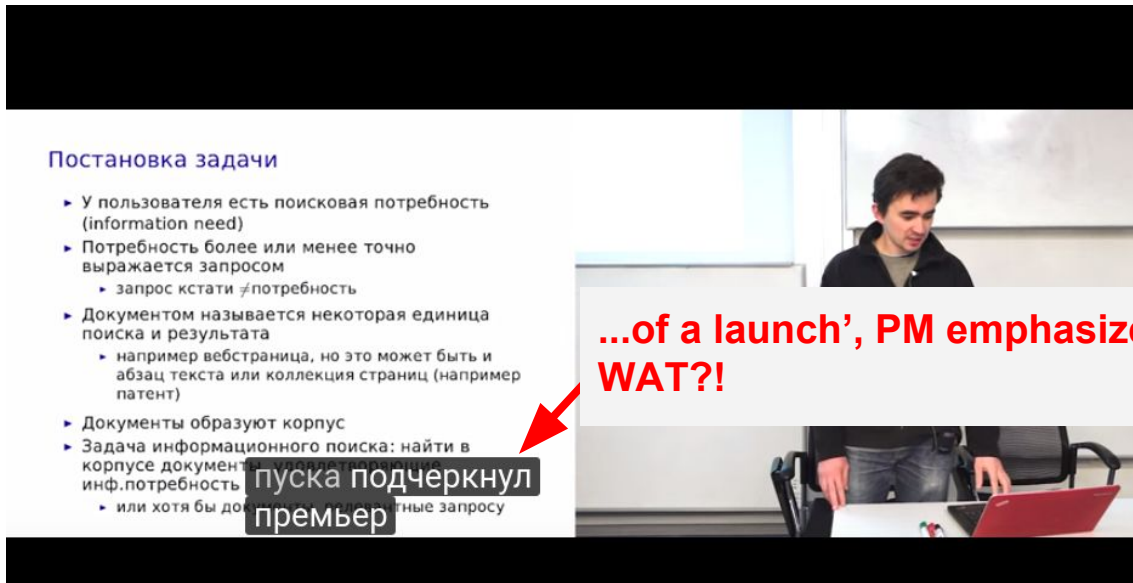
# Language modeling

Lectures: Anton Alekseev, Steklov Mathematical Institute in St Petersburg  
NRU ITMO, St Petersburg, 2019



# Motivation

In many tasks one has to estimate whether the text is 'natural' or 'comprehensible'. Sometimes a clever way to estimate the word sequence probability is enough



Постановка задачи

- ▶ У пользователя есть поисковая потребность (information need)
- ▶ Потребность более или менее точно выражается запросом
  - ▶ запрос кстати ≠ 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 докумен
  - ▶ инф.потребность
  - ▶ или хотя бы до

пуска подчеркнул премьер

...of a launch', PM emphasized WAT?!

...puska podcherknul prem'yer

Actually Dmitriy said:

...poisk po patentam naprimer  
~ *patent search, for example*

*I must admit it was kinda hard to find a good example of lousy generated English subtitles*

<https://youtu.be/APcwsxUpGrQ?t=1m38s> 2

# Motivation

- **Speech recognition / machine translation / spelling correction / augmentative communication**  
e.g.: having generated several possible decodings of the phrase, one has to choose ‘the most probable’ (from the language’s point of view)
- **Information retrieval**  
ranking: for every document  $\mathbf{d}$  we build ‘its language model’ and sort all documents by  $\mathbf{P}(\mathbf{q}|\mathbf{d})$  (where  $\mathbf{q}$  is a query)
- **Fun!** Text generators, imitating the provided text collection’s style

# Plan

1. Intuition
  2. N-gram modeling
  3. Language models quality evaluation
  4. Zeros and smoothing
    - a. Kneser-Ney smoothing
- Libraries
  - Datasets

# Intuition

- **Language model** allows us to estimate the probability of any sequence of words (alternative formulation: to estimate the probability of the next word)
- How to estimate the probability of *'Everything was in confusion in the Oblonskys' house...'*?
- Let us turn to conditional probability

# Intuition: total recall

- ▶ Conditional probability

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

- ▶ Chain rule for greater number of variables:

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

- ▶ So can we compute it all easily?

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

\* Here and further Count(...) is the same as C(...) и c(...)

+

-

## Intuition: total recall

- ▶ Conditional probability

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

- ▶ Chain rule for greater number of variables:

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

- ▶ So can we compute it all easily?

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

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

# Intuition: total recall

- ▶ Conditional probability

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

- ▶ Chain rule for greater number of variables:

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

- ▶ So can we compute it all easily?

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

(nope! long chains are rare events!)



## What do we do?

- ▶ Assumption is here to help: text satisfies the Markov property

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

...which means that current event depends on not more than on  $K$  preceding ones

- ▶ Examples:
  - ▶  $K = 0$  (unigram model)

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

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

- ▶  $K = 1$  (bigram model)

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

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

# Plan

- ~~1. Intuition~~
  2. N-gram modeling
  3. Language models quality evaluation
  4. Zeros and smoothing
    - a. Kneser-Ney smoothing
- Libraries
  - Datasets

# N-gram model

- ▶ Model:

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

one has to add  $N - 1$  terms «begin» ^ and «end» \$ from both sides (padding)

- ▶ We can estimate the probability like that

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

- ▶

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

- ▶ E.g. for bigrams:

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

# Plan

- ~~1. Intuition~~
  - ~~2. N-gram modeling~~
  3. Language models quality evaluation
  4. Zeros and smoothing
    - a. Kneser-Ney smoothing
- Libraries
  - Datasets

# Quality evaluation techniques

- **Extrinsic**

Checking quality by inducing the model into a bigger useful task (machine translation, spelling correction, ...).

If the target metric (where the money is: translators work time, editor's time, clicks count, earned money, etc.) goes up, **the model has become better**

- **Intrinsic**

~~Evaluation for the poor~~ we need estimates when extrinsic evaluation is too expensive or when one doesn't want the results to be related to some specific application (if the model is universal to certain extent); also a metric that shows us how 'good' the model is

# Quality evaluation techniques

- **Extrinsic**

Checking quality (by inducing the model into a bigger useful task (machine translation, spelling correction, ...)).

If the target metric (where the money is: translators work time, editor's time, clicks count, earned money, etc.) goes up, **the model has become better**

- **Intrinsic**

~~Evaluation for the poor~~ when we need estimates when extrinsic evaluation is too expensive or when one doesn't want the results to be related to some specific application (if the model is universal to certain extent); also a metric that shows us how 'good' the model is

*Not this time  
(totally different story)*

**REMINDER!**

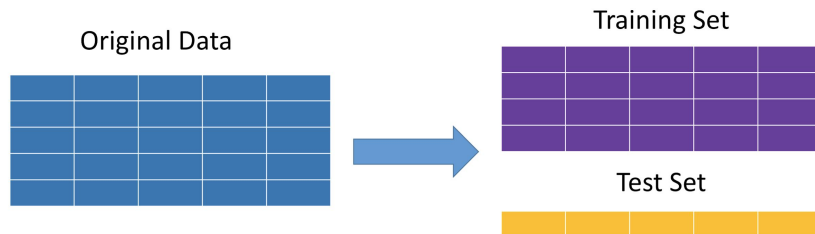
# Quality evaluation

We have the data, we have the metric

We split the data into

- train set (for tuning models) and
- test set (for trained models evaluation)

We have to believe that train and test set data samples are from “the same distribution” (otherwise we won’t be able to train anything useful)



**REMINDER!**

# Quality evaluation

## Deadly Sin №1

Test data leaks into train set  
(this way we lose generalization capability and estimates validity)

## Deadly Sin №2

Tuning hyperparameters on test set

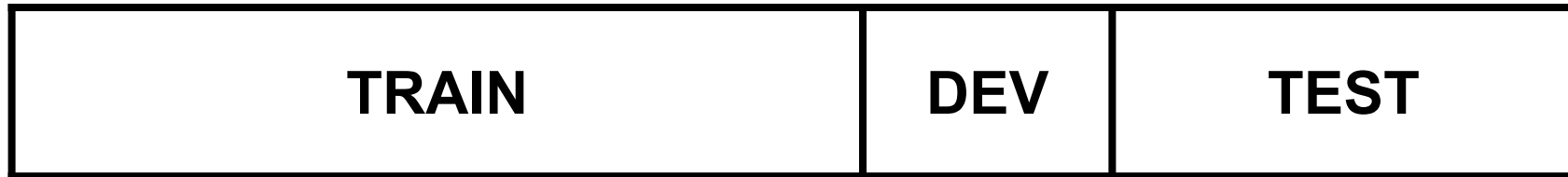
**But how do we tune the parameters? Ideas?**





**REMINDER!**

## Quality evaluation: data splitting



1. TRAIN - training model
2. DEV - evaluating quality + analyzing errors + tuning hyperparameters
3. TEST - blind quality evaluation: looking at quality metric ONLY + not too often, so as not to overfit

# Model quality evaluation

- ▶ The larger the probability of the test text, the closer the model is to life
- ▶ Perplexity — inverse probability of the text normalized by words sequence length

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

It is evident that less is better.

- ▶ To those who know some information theory, the formula may seem familiar:

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

# Quality evaluation: example

Training on 38M tokens

Testing on 1.5M

Dataset: Wall Street Journal

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

*from Martin/Jurafsky*

# Plan

- ~~1. Intuition~~
  - ~~2. N-gram modeling~~
  - ~~3. Language models quality evaluation~~
  4. Zeros and smoothing
    - a. Kneser-Ney smoothing
- Libraries
  - Datasets

# Generalization capability discussion

- There is no such *perfect* corpus where all possible n-grams occur at least once!
- The model we have described returns  $P(x, \dots) = 0$  when run on the text that contains at least one ngram that was not present in train set
- Evident enough, the model must generalize (and not just encode with non-zeros what was present in the train set)

**a very natural solution is to convert zeros to small values**

- Also: words we haven't met before (**OOV = out of vocabulary**) can be replaced with some universal substitutes, e.g.  
<UNKNOWN>/'part-of-speech'/'frequential bucket'

# Laplacian smoothing (add-one smoothing)

- ▶ Let us imagine that all n-grams in concern occur in the text one more time. Then we can re-estimate the probabilities like that (bigrams example)

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

where  $V$  would save probabilities from not being equal to 1 when summed. What does it equal to?



# Laplacian smoothing (add-one smoothing)

- ▶ So,

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

- ▶ If we sum over  $w_i$ , we'll see that  $V$  should be the cardinality of unigrams set, otherwise  $P$  couldn't be called probability.
- ▶ Doesn't work well  
(to much useful weight is transferred to zeros!)
- ▶ The Fix for the poor:

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



## Backoff and interpolation

- ▶ No occurrences of «somewhat young specialist», yet a few «young specialist» bigrams (if none — unigram «specialist»)
- ▶ One can use probabilities of smaller n ngrams for computing estimates of probabilities for target ones with zero counts. This is called **backoff**.
- ▶ Every n-gram probability can be treated as a weighed sum of probabilities of ngrams it contains: n-1-grams, n-2-grams, etc. This is called **interpolation**.

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

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

$\lambda$  weights are tuned on the separate held out dev set, may depend on different contexts.



# Kneser-Ney smoothing: idea №1

- choose bigrams, counts of which equal to  $k$  in the train set
- look at their counts in the held out set

We'll see that difference is ~ **constant!**  
(excluding rare n-grams in both sets)

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

Hence let us remember the shift  $d = 0.75$  for all the n-grams or  $0.75$  for **2...9** and  $0.5$  for **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

# Kneser-Ney smoothing: idea №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 - absolute discount

# Kneser-Ney smoothing: idea №2

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

- Why should we interpolate? Which n-grams are rare guests?
- “Despite he begged for \_\_\_\_\_”  
“stockings”? “Lanka”? -- different yet equally frequent
- **Idea:** the larger the **cardinality of set of n-grams that contain the word**, the more **useful for interpolation** this word is
- *Intuition: should we consider ‘Francisco’ as a filler for this particular ‘gap’ if it usually goes **only after the word ‘San’?***

## Kneser-Ney smoothing: idea №2

- **Idea:** the larger the **cardinality of set of n-grams containing the word**, the more **useful for interpolation this word** (hopefully) is

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

# Kneser-Ney smoothing: final formula

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

Lambda helps to preserve properties of probabilities distributing the ‘weight’ between ngrams correctly

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

There is a recursive formula for ngrams for any **n**  
(see Martin-Jurafsky, Chapter 4)


# Summary: which is the best?

[Philip Koehn's slides](#)

## Evaluation

Evaluation of smoothing methods:

Perplexity for language models trained on the Europarl corpus

See the literature 

<b>Smoothing method</b>	<b>bigram</b>	<b>trigram</b>	<b>4-gram</b>
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

# Plan

- ~~1. Intuition~~
  - ~~2. N-gram modeling~~
  - ~~3. Language models quality evaluation~~
  - ~~4. Zeros and smoothing~~
    - a. ~~Kneser-Ney smoothing~~
- Libraries
  - Datasets

# Tools

`nltk` has some LM-related code (`nltk.models`)

Here's what Moses can use (open source 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.



# Datasets

- \*Huge unlabeled texts collection for your specific task
- Datasets for tasks that use LM, e.g. WMT
- Google NGrams
- National corpora (e.g. 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 частот. При подсчёте учитывался регистр букв, а также знаки препинания. Общий объём корпуса – 192689044 словоформы.

Словоформы	<a href="#">zip-архив</a> (5.5 МБ)
2-граммы	<a href="#">zip-архив</a> (1.5 МБ)
3-граммы	<a href="#">zip-архив</a> (1.5 МБ)
4-граммы	<a href="#">zip-архив</a> (1.5 МБ)
5-граммы	<a href="#">zip-архив</a> (1.5 МБ)
6-граммы	<a href="#">zip-архив</a> (1.5 МБ)

**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:

```
ngram 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):

```
circumvallate 1978 335 91
circumvallate 1979 261 91
```

Тип токенов:  все  только слова  не только слова

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.

# LM lectures takeaways

- We have discussed machine learning models evaluation
- We've learnt how to estimate word sequence probabilities using a practical mainstream method

# Sources and recommendations

Slides are heavily based on Jurafsky/Martin book and Daniel Jurafsky's course slides + a few peeks at P. Braslavsky's course were taken

Recommended:

- Martin-Jurafsky, edition 3, chapter 4
- "Statistical Machine Translation" Philip Koehn

# Language modeling

Lectures: Anton Alekseev, Steklov Mathematical Institute in St Petersburg  
NRU ITMO, St Petersburg, 2019