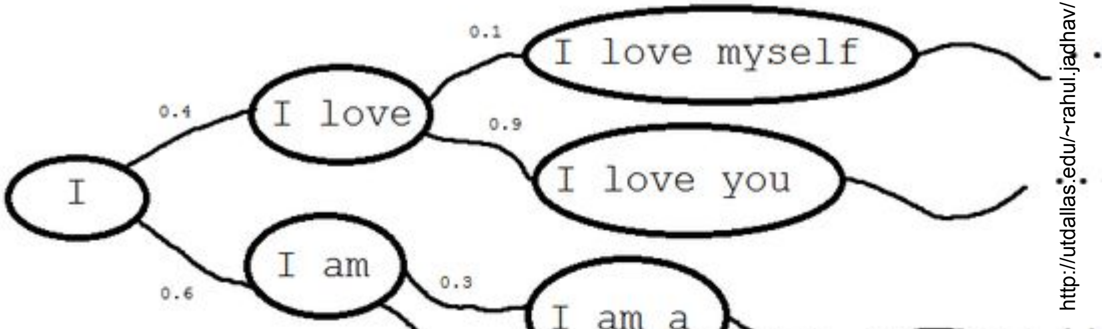
Language modeling-II

Lectures: Anton Alekseev, Steklov Mathematical Institute in St Petersburg
NRU ITMO, St Petersburg, 2018



http://utdallas.edu/~rahul.jadhav/

N-gram model

- Model:

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

  one has to add $N-1$ terms «begin» ^and «end» $ from both sides (padding)
- We can estimate the probability like that

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

- $$P(x_i|x_{i-1}) = Count(x_i, x_{i-1})Count(x_{i-1})$$
- E.g. for bigrams:

$$P(hello, i, love, you) =$$

$$= P(hello|\hat{}\,)P(i|hello)P(love|i)P(you|love)P(\$|you)$$

Plan

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

- Libraries
- Datasets

reminder: 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

reminder: Quality evaluation: data splitting

TRAIN	DEV	TEST

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

$$PP(W) = P(x_1 \ldots x_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(x_1 \ldots x_N)}} =$$

$$= \sqrt[N]{\frac{1}{\prod_{i=1}^{N} P(x_i | x_1 \ldots x_{i-1})}}$$

It is evident that less is better.

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

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

Quality evaluation: example

Training on 38M tokens

Testing on 1.5M

Dataset: Wall Street Journal

	1-gram	2-gram	3-gram
Perplexity	962	170	109

из 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,...) = 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)

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

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

Laplacian smoothing (add-one smoothing)

- So,

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

- 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:

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

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**.

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

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

λ 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

Gale, W. A. and Church, K. W. (1994). What is wrong with adding one?. In Oostdijk, N. and de Haan, P. (Eds.), Corpus-Based Research into Language, pp. 189– 198. Rodopi

Kneser-Ney smoothing: idea №1

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

d - absolute discount

Kneser-Ney smoothing: idea №2

discounted bigram Interpolation weight

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \boxed{\lambda(w_{i-1}) P(w)}$$

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

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

Kneser, R. and Ney, H. (1995). Improved backing-off for M-gram language modeling. In ICASSP-95, Vol. 1, pp. 181–184.

Kneser-Ney smoothing: final formula

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

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

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

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

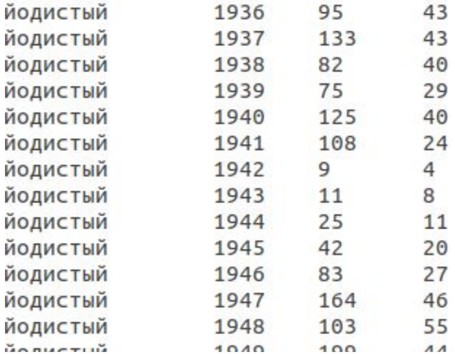
The language model should be trained on a corpus that is suitable to the domain. If the although using additional training data is often beneficial.

Our decoder works with the following language models:

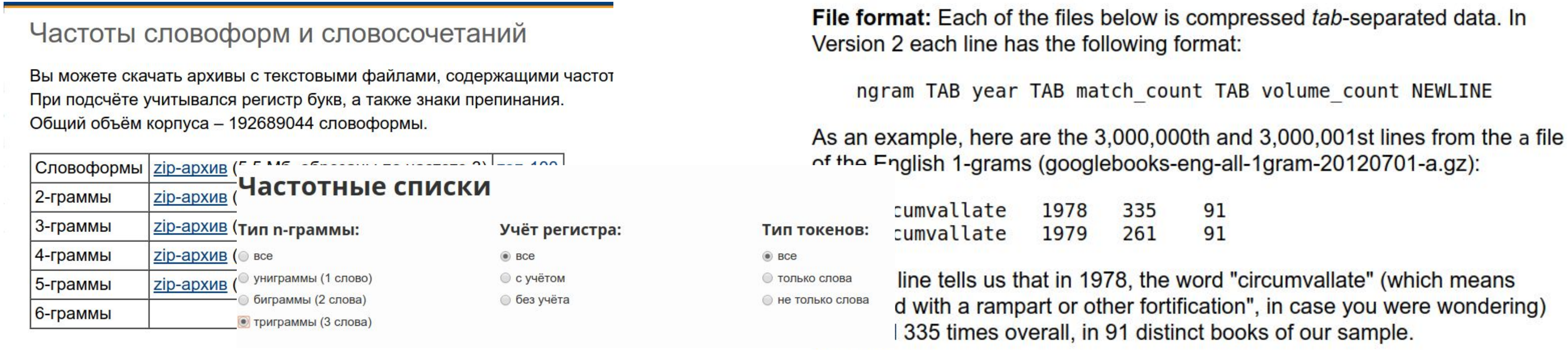
- the SRI language modeling toolkit, which is freely available.
- the IRST language modeling toolkit, which is freely available and open source.
- the RandLM language modeling toolkit, which is freely available and open source.
- the KenLM language modeling toolkit, which is included in Moses by default.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



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, 2018

Markov models, information theory and why we care about it all

Anton Alekseev,
Steklov Mathematical Institute in St Petersburg NRU ITMO,

NRU ITMO, St Petersburg, 2018

anton.m.alexeyev+itmo@gmail.com

Plan for today: theory and applications

1. Markov chains
   a. Language models
   b. Keywords extraction and other applications

2. Elements of information theory
   a. Information
      i. Collocations extraction
      ii. One weird trick to estimate sentiment
   b. Entropy
      i. Connection between entropy and perplexity

Markov property

N-gram models we discussed earlier actually are **Markov models**

Markov property: conditional distribution of the next state of a stochastic process depends only on current state

$$\mathbb{P}(X_{n+1} = i_{n+1} \mid X_n = i_n, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = \mathbb{P}(X_{n+1} = i_{n+1} \mid X_n = i_n)$$

The process with discrete time (or a sequence of random events), that has this property is called a **Markov chain**

A simple and a well-studied probabilistic model suitable for many tasks

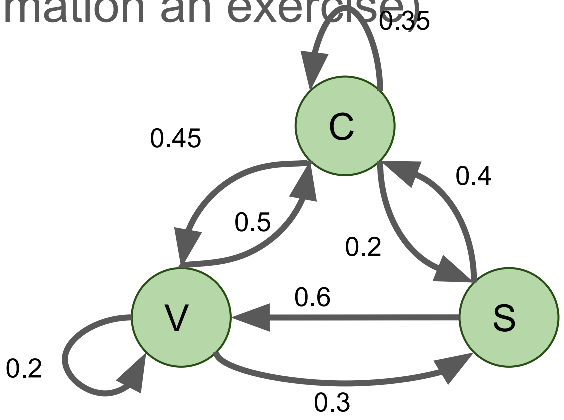
Markov chain

The model is entirely set by the stochastic matrix = transitions probabilities matrix

Example. Events: vowel (v), consonant (c), white**s**pace/punctuation (s) (probabilities are set at random, consider the estimation an exercise)

P_{trans} =

	v	c	s
v	0.2	0.5	0.3
c	0.45	0.35	0.2
s	0.6	0.4	0.0



DEMO: ugly self-promotion: http://antonalexeev.hop.ru/markov/index.html

Markov chains

- So — Markov chain as a process is set by the matrix of transitions probabilities and probabilities of initial states

$$\pi = (p_1^{(0)}, ..., p_n^{(0)})^T$$

$$P_{trans} = \{p_{i\to j},\ i,j \in 1:n, \sum_{j=1}^{n} p_{i\to j} = 1\forall i\}$$

- Probability of a trajectory
  of length one x_i

$$p = p_i$$

  of length two $x_i \to x_j$

$$p = p(x_i)p(x_j|x_i) = \pi_i P_{i,j}$$

  of length three $x_i \to x_j \to x_k$

$$p = p(x_i)p(x_j|x_i)p(x_k|x_i, x_j) = p(x_i)p(x_j|x_i)p(x_k|x_j) = \pi_i P_{i,j} P_{j,k}$$

Markov chains

- Evident enough, probability of trajectory of length n is computed like that

$$p(x_a, ..., x_z) = \pi_a \prod_{i=2}^{|steps|} P_{steps[i], steps[i+1]}, steps = (a, ..., z)$$

- It is easy to prove that the vector of probabilities of the process to be in certain states at $m-$th step can be computed like that

$$\pi^{(m)} = (p_1^{(m)}, ..., p_n^{(m)}) = \pi P_{tr}^m$$

Markov chains: the limit

One can demonstrate that if $P_{trans\ i,j} = p_{i \to j} > 0$, there exist a single asymptotic distribution

$$\hat{\mathbf{p}} = \lim_{m \to \infty} \pi P_{trans}^{m},$$

and

$$\hat{\mathbf{p}} = \hat{\mathbf{p}} P_{trans}, \sum \hat{p}_i = 1$$

Such distribution is called the **stationary** one.

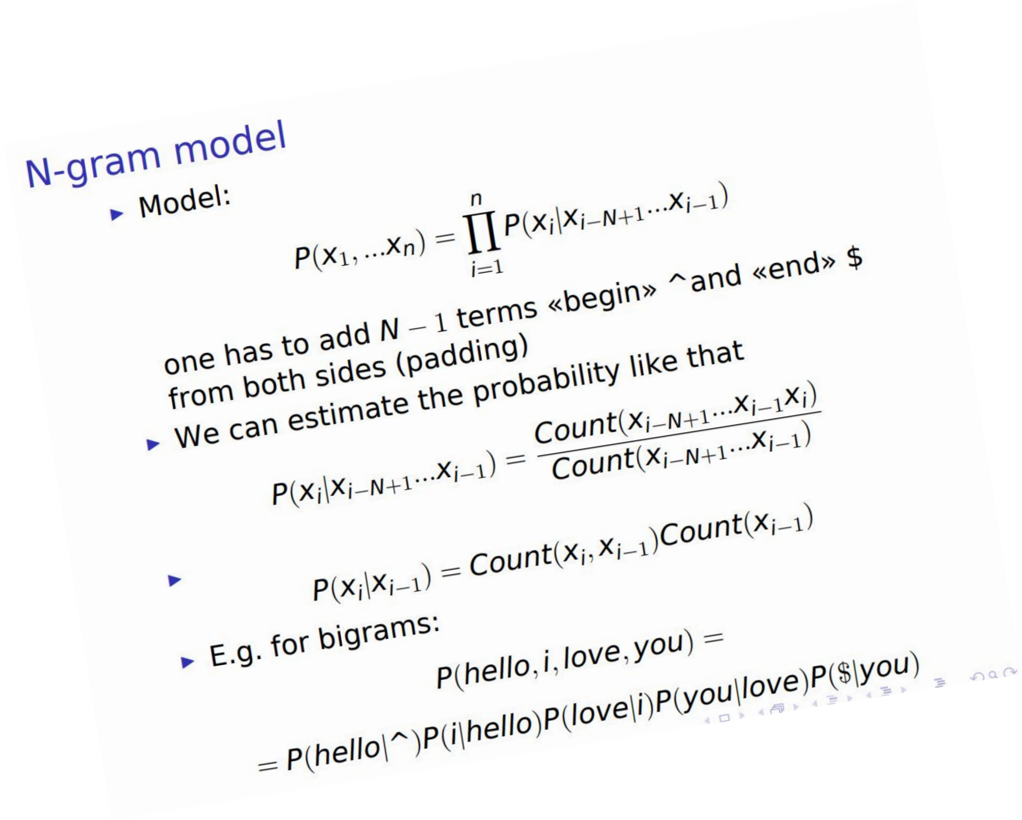
Stationary distribution: interpretation

Suppose we are watching random [web] surfer, who moves from state to state **eternally**, making decisions where to glide using the distribution of states in the current row

http://slideplayer.com/slide/8080871/

Then each value in the vector of stationary distribution is **the fraction of total time** spent in the corresponding state

Application example №1 (previous lecture)



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

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

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

- E.g. for bigrams:

$$P(hello, i, love, you) =$$

$$= P(hello|\hat{}\,)P(i|hello)P(love|i)P(you|love)P(\$|you)$$

Application example №2: PageRank

The ‘value’ of the web page is defined by

- the ‘value’ of the pages that refer to it,
- a number of pages those pages refer to
(less = better)

Let $L_{ij} = 1$ if webpage j links to webpage i (written $j \to i$), and $L_{ij} = 0$ otherwise

Also let $m_j = \sum_{k=1}^{n} L_{kj}$, the total number of webpages that j links to

First we define something that's almost PageRank, but not quite, because it's broken. The BrokenRank p_i of webpage i is

$$p_i = \sum_{j \to i} \frac{p_j}{m_j} = \sum_{j=1}^{n} \frac{L_{ij}}{m_j} p_j$$

http://www.stat.cmu.edu/~ryantibs/datamining/lectures/03-pr.pdf

To be continued

Markov models, information theory and why we care about it all

Anton Alekseev,
Steklov Mathematical Institute in St Petersburg NRU ITMO,

NRU ITMO, St Petersburg, 2018

anton.m.alexeyev+itmo@gmail.com