Faculty for Informatics

Technical University of Munich

Natural Language Processing IN2361

Prof. Dr. Georg Groh

Social Computing Research Group

Chapter 3
Language Modelling with N-Grams

- content is based on [1] and [2]
- certain elements (e.g. equations or tables) were taken over or taken over in a modified form from [1] or [2]
- citations of [1] and [2] or from [1] or [2] are omitted for legibility
- errors are fully in the responsibility of Georg Groh
- BIG thanks to Dan and James for a great book!

Assigning a Probability to a Sequence of Words

Motivation:

- Simple Prediction:
  - *please turn your homework…*: P(*in*) > P(*the*)
  - P(*all of a sudden I notice three guys standing on the sidewalk*) > P(*on guys all I of notice sidewalk three a sudden standing the*)
- Machine Translation:
他	向	记者	介绍了	主要	内容
He	to	reporters	introduced	main	content

    he introduced reporters to the main contents of the statement
    he briefed to reporters the main contents of the statement
    he briefed reporters on the main contents of the statement
- Spell Correction
  - *The office is about fifteen minuets from my house:*
    P(*about fifteen minutes from*) > P(*about fifteen minuets from*)
- Speech Recognition
  - P(*I saw a van*) >> P(*eyes awe of an*)

Assigning a Probability to a Sequence of Words

- **Language model**: assignment of (joint) probabilities to sequences of words:

$$P(w_1, w_2, ..., w_n) = P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

notation: $w_n^m = w_{n:m}$

or (equivalently) modelling conditional probabilities
(e.g. for predicting next word)

$$P(w_n|w_1^{n-1})$$

- MLE based probability estimation: counting instances of sequences in corpora: e.g.

$$P(the|its\ water\ is\ so\ transparent\ that) = \frac{C(its\ water\ is\ so\ transparent\ that\ the)}{C(its\ water\ is\ so\ transparent\ that)}$$

problem: language is creative, combinatorial explosion
→ only very few instances of given sequence → sparsity, poor estimates

N-Gram Models

- Solution: Assumption: restrict chain rule to Markov order N
  → N-Gram Models:

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$$

- N=2 : Bigram model

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

- MLE for Bigram model: Count $C(w_{n-1}w_n)$ of bigram $w_{n-1}w_n$ in corpus →

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

N-Gram Models

- → MLE for N-Gram model:

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

- Example: sample sentence from unigram model $P(w_1^n) \approx \prod_{k=1}^{n} P(w_k)$

  fifth an of futures the an incorporated a a the inflation most dollars quarter in is mass

- Example: sample sentence from bigram model $P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$

  texaco rose one in this issue is pursuing growth in a boiler house said mr. gurria mexico 's motion control proposal without permission from five hundred fifty five yen

Example Bigram Model

	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. Probabilities capture syntactic, pragmatic and cultural aspects of language

$$
\begin{aligned}
P(\texttt{<s> i want english food </s>}) \\
&= P(\texttt{i}|\texttt{<s>})P(\texttt{want}|\texttt{i})P(\texttt{english}|\texttt{want}) \\
&\qquad P(\texttt{food}|\texttt{english})P(\texttt{</s>}|\texttt{food}) \\
&= .25 \times .33 \times .0011 \times 0.5 \times 0.68 \\
&= \; = .000031
\end{aligned}
$$

N-Gram Models

- extending to trigram, 4-gram, 5-gram: better quality but: not sufficient because language has long-term dependencies

- further aspect: probabilities can / should be modelled depending on current usage context

- numerical issue: use log-space to prevent underflows

$$p_1 \times p_2 \times p_3 \times p_4 = \exp(\log p_1 + \log p_2 + \log p_3 + \log p_4)$$

- large N-Gram models available (e.g. Google N-Grams (2006) computed from 10^{13} words of running text, 10^{10} 5-grams appearing ≥ 40 times)

Evaluating / Comparing N-Gram Models

- Best: extrinsic evaluation (on actual application): often costly;
  → Intrinsic evaluation: on data only

- intrinsic evaluation: split data (corpus) into
  - training set/part TrSet,
  - validation/development set/part DevSet,
  - test set/part TeSet (e.g. 80-10-10)

- Simple performance measure for N-Gram models: likelihood:
  model M_1 is better than other model M_2 (both trained + fine-tuned on TrSet+DevSet) if $P(\text{TeSet}|M_1) > P(\text{TeSet}|M_2)$ where $P(\text{TeSet}|M_i) = P(w_1, w_2, \ldots, w_N|M_i)$

Evaluating / Comparing N-Gram Models

- Another measure: **Perplexity** PP of TeSet: (lower PP = better model):

$$\mathrm{PP}(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

$$\underset{\text{(for a bigram model)}}{=} \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

- Perplexity = **weighted average branching factor** of language / corpus. **Branching factor** = how many different words can follow any word or N-Gram
  - example: string of N digits, unigram model with uniform $P = 1/10$ →

$$\mathrm{PP}(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} = (\frac{1}{10}^{N})^{-\frac{1}{N}} = 10$$

if one digit is much more likely →
branching factor is still 10 but PP is smaller

Corpus Dependence of N-Gram Models

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.

sampled sentences from N-Gram models trained on complete Shakespeare

Perplexity		
Unigram	**Bigram**	**Trigram**
962	170	109

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

sampled sentences from N-Gram models trained on $4 * 10^6$ words Wall Street Journal corpus

Corpus Dependence: Unseen Combinations, Unknown Words

- Occurrence of Out of Vocabulary (OOV) words:
  - approach 1 (preferred):
    - choose fixed vocabulary
    - replace **unknown** words with **<unk>**
    - estimate probabilities as before
  - approach 2:
    - replace all **infrequent** words (set threshold) with **<unk>**
    - then use approach 1

- Black Swan Paradox: MLE → unseen word combinations have probability zero (→ PP diverges): overfitting!

- → work with Priors, MAP. Simple variant of that: smoothing

TrSet

denied the allegations:	5
denied the speculation:	2
denied the rumors:	1
denied the report:	1

TeSet

denied the offer
denied the loan

P(.)=0

Laplace Smoothing

- add-1-smoothing (Laplace smoothing) : each word gets a prior probability of $1/|V|$ (compare Beta priors for coins in ML1) →
  - for unigram models

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

corresponding to an adjusted ("discounted") count

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

  - for bigram models

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \quad \rightarrow \quad P^*_{\text{Laplace}}(w_n|w_{n-1}) =$$

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

(adjusted count: $c^*(w_{n-1}w_n) = \dfrac{[C(w_{n-1}w_n) + 1] \times C(w_{n-1})}{C(w_{n-1}) + V}$)

Laplace Smoothing

- also possible: make prior stronger by adding k instead of 1 (choose k with DevSet)

$$P^*_{\text{Add-k}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+k}{C(w_{n-1})+kV}$$

Laplace Smoothing

- even for k=1 the discount factor $d = c^*/c$ can be very substantial

	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 4.2 Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray. $P(w_{n-1}, \mathrm{w_n})$

	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 3.6 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. $P^*_{Add-1}(w_{n-1}, \mathrm{w_n})$

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Figure 3.1 Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray. $C(w_{n-1}, \mathrm{w_n})$

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Figure 3.5 Add-one smoothed bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Previously-zero counts are in gray. $C(w_{n-1}, \mathrm{w_n}) + 1$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Figure 3.7 Add-one reconstituted counts for eight words (of $V = 1446$) in the BeRP corpus of 9332 sentences. Previously-zero counts are in gray. $C^*(w_{n-1}, \mathrm{w_n})$

d (*want to*) = 0.39

Backoff

- previous slide: simple smoothing → too much probability mass shifted to zero counts → sharp changes in probabilities / counts.
  → simple smoothing techniques alone often do not work well ☹

- Other solution: use less context: **Backoff**: use trigram if evidence sufficient, if not (e.g. true counts for trigram are small or zero) use bigram, if bigram evidence too low use unigram

- probability theory correctness: distribution of probability mass needs to be adjusted by discounting: (e.g. take away mass from trigrams and add it to bigram etc.): Katz Backoff:

$$P_{\mathrm{BO}}(w_n|w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n|w_{n-N+1}^{n-1}), & \text{if } C(w_{n-N+1}^{n}) > 0 \\ \alpha(w_{n-N+1}^{n-1}) P_{\mathrm{BO}}(w_n|w_{n-N+2}^{n-1}), & \text{otherwise.} \end{cases}$$

Interpolation

- Other solution: **Interpolation**: mix all three:

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n) \qquad \sum_i \lambda_i = 1$$

- Weighted Interpolation: λs depend on context: the higher the true counts, the larger λ. (determine λs using validation/hold-out corpus with Expectation Maximization)

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1(w_{n-2}^{n-1})P(w_n|w_{n-2}w_{n-1}) + \lambda_2(w_{n-2}^{n-1})P(w_n|w_{n-1}) + \lambda_3(w_{n-2}^{n-1})P(w_n)$$

Absolute Discounting

- We have seen: introducing priors → adjusted ("discounted") counts are smaller than original counts (shifting some probability mass to unseen words / N-Grams ("zeros"))

- Suppose we wanted to subtract a little from a count of 4 to save probability mass for the zeros: How much to subtract ?

- Church & Gale 1991: Divide $44 * 10^6$ word corpus in two $22 * 10^6$ halves. For all bigrams that occurred exactly n times in first half: how often on average do they occur in second set?

  → subtract ≈ 0.75 (probability mass shifted to zeros)

n	N number of bigrams in first half of data that occurred n times	C_2 total number of occurrences of these bigrams in second half of data	C_2 / N average no of occurrences of these bigrams in second half of data
0	74 671 100 000	2 019 187	0.000027
1	2 018 046	903 206	0.448
2	449 721	564 153	1.25
3	188 933	424 015	2.24
4	105 664	341 099	3.23
5	68 379	287 776	4.21
6	48 190	251 951	5.23
7	35 709	221 693	6.21
8	27 710	199 779	7.21
9	22 280	183 971	8.26

Absolute Discounting

$$P_{\text{AbsoluteDiscounting}}(w_i|w_{i-1}) = \underbrace{\frac{C(w_{i-1}w_i)-d}{\sum_v C(w_{i-1}v)}}_{\substack{\text{discounted bigram}\\ d\approx 0.75}} + \underbrace{\lambda(w_{i-1})}_{\substack{\text{Interpolation}\\ \text{weight}}}\underbrace{P(w_i)}_{\text{unigram}}$$

But should we really just use the raw regular unigram probability $P(w_i)$?

→ better way: **Kneser-Ney smoothing** (often delivers best performance)

Kneser Ney Smoothing

- Example: continuation of sentence

  I can't see without my reading____________?

  since *Hong Kong* is a frequent bigram, $P(Kong) > P(glasses)$
  → in a unigram model, select *Kong* instead of *glasses*

- → instead of $P(w)$: "How likely is w" we really want $P_{\text{continuation}}(w)$: "How likely is w to appear as a novel continuation?".
  Intuition: $P_{\text{continuation}}(w)$ proportional to **number of different** (bigram) **contexts** that w has appeared in in TrSet

$$P_{\text{CONTINUATION}}(w) = \frac{|\{v : C(vw) > 0\}|}{|\{(u', w') : C(u'w') > 0\}|}$$

or

$$P_{\text{CONTINUATION}}(w) = \frac{|\{v : C(vw) > 0\}|}{\sum_{w'} |\{v : C(vw') > 0\}|}$$

now: $P_{\text{continuation}}(Kong) < P_{\text{continuation}}(glasses)$

Kneser Ney Smoothing

- → interpolated Kneser – Ney smoothing:

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

$\lambda(w_{i-1})$ is a normalizing constant ←→ distribute the probability mass we've discounted

$$\lambda(w_{i-1}) = \underbrace{\frac{d}{\sum_v C(w_{i-1}v)}}_{\text{the normalized discount}}\underbrace{|\{w : C(w_{i-1}w) > 0\}|}_{\text{The number of word types that can follow } w_{i-1}}$$

the normalized discount

$= d/C(w_{i-1})$

The number of word types that can follow w_{i-1}

= # of word types we discounted

= # of times we applied normalized discount

General Recursive Kneser Ney

$$P_{\mathrm{KN}}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c_{KN}(w_{i-n+1}^{i}) - d, 0)}{\sum_v c_{KN}(w_{i-n+1}^{i-1}v)} + \lambda(w_{i-n+1}^{i-1}) P_{KN}(w_i|w_{i-n+2}^{i-1})$$

with $c_{KN}(\cdot) = \begin{cases} \text{count}(\cdot) & \text{for the highest order} \\ \text{continuationcount}(\cdot) & \text{for lower orders} \end{cases}$

and termination of recursion (unigrams interpolated with uniform distr):

$$P_{\mathrm{KN}}(w) = \frac{\max(c_{KN}(w) - d, 0)}{\sum_{w'} c_{KN}(w')} + \lambda(\epsilon)\frac{1}{V}$$

where ϵ is the empty string.

Stupid Backoff

- on very large corpora (web-scale): instead of full Kneser-Ney, give up idea of correct probabilities and do simple backoff to lower order N-Gram

$$S(w_i|w_{i-k+1}^{i-1}) = \begin{cases} \frac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if count}(w_{i-k+1}^{i}) > 0 \\ \lambda S(w_i|w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

with constant $\lambda \approx 0.4$

referring to this as S instead of P (because it is not a probability)

Entropy <-> Perplexity

- Entropy of length n word sequences

$$H(W_1^n) = H(w_1, w_2, \ldots, w_n) = -\sum_{W_1^n \in L} p(W_1^n) \log p(W_1^n)$$

- per-word entropy (entropy rate):

$$\frac{1}{n} H(W_1^n) = -\frac{1}{n} \sum_{W_1^n \in L} p(W_1^n) \log p(W_1^n)$$

- entropy rate of a whole language L:

$$\begin{aligned} H(L) &= \lim_{n \to \infty} \frac{1}{n} H(w_1, w_2, \ldots, w_n) \\ &= -\lim_{n \to \infty} \frac{1}{n} \sum_{W \in L} p(w_1, \ldots, w_n) \log p(w_1, \ldots, w_n) \end{aligned}$$

Entropy ←→ Perplexity

- language perceived as a stochastic Markov process : if stationary and ergodic: (Shannon McMillan, Breiman)→ process converges to a limit distribution →

$$H(L) = \lim_{n \to \infty} -\frac{1}{n} \log p(w_1 w_2 \ldots w_n)$$

→ instead of averaging over all possible sequences, it suffices to take long enough sequence (large enough corpus)

- → if $H(W) \approx -\frac{1}{N} \log P(w_1 w_2 \ldots w_N)$

we have

$$\text{Perplexity}(W) = 2^{H(W)}$$

$$\begin{aligned}
\text{Perplexity}(W) &= 2^{H(W)} \\
&= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}} \\
&= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}
\end{aligned}$$

Cross Entropy

- if we do not know true p but instead approximate it with some simpler distribution m , we can use cross entropy as upper limit for entropy:

$$H(p,m) = \lim_{n\to\infty} -\frac{1}{n} \sum_{W\in L} p(w_1,\ldots,w_n) \log m(w_1,\ldots,w_n)$$

because for all m, p $\quad H(p) \leq H(p,m)$

- again: (Shannon McMillan, Breiman) →

$$H(p,m) = \lim_{n\to\infty} -\frac{1}{n} \log m(w_1 w_2 \ldots w_n)$$

Bibliography

(1) Dan Jurafsky and James Martin: Speech and Language Processing (3rd ed. draft, version Oct. 2019); Online: https://web.stanford.edu/~jurafsky/slp3/ (URL Oct 2019) (this slideset is especially based on chapter 3)

(2) Powerpoint slides from Dan Jurafsky and James Martin: Speech and Language Processing (3rd ed. draft); Online: https://web.stanford.edu/~jurafsky/slp3/ (URL Oct 2018)

Recommendations for Studying

- minimal approach:

  work with the slides and understand their contents! Think beyond instead of merely memorizing the contents

- standard approach:

  minimal approach + read the corresponding pages in Jurafsky [1]

- interested students

  == standard approach