LINFO2263: N-grams

Pierre Dupont

Outline

Outline

A probabilistic approach to language modeling

A fragment of the ATIS (Air Travel Information System) corpus, after tokenization, capitalization, and some simple standardization

I NEED TWO TICKETS ROUND TRIP FROM **BOSTON** TO SAN FRANCISCO THAT COST LESS THAN ONE THOUSAND SEVEN HUNDRED AND TWENTY DOLLARS

I WOULD LIKE TO GO FROM BALTIMORE TO **BOSTON**

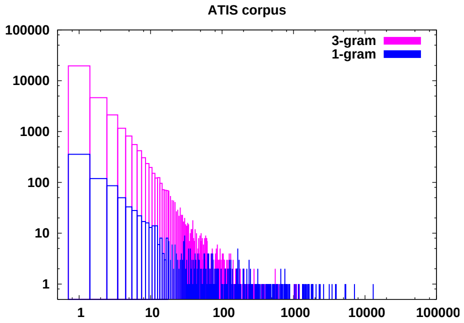
WHAT TIME DOES UNITED FLIGHT TWO OH ONE STOP **IN DENVER**

I WOULD LIKE TO FLY FROM OAKLAND TO **BOSTON** STOPPING **IN DENVER**

- Consider each word occurrence as a random event
- The probability of an event is estimated from its count or frequency of occurrence: $C(BOSTON) = 3$
- Those notions are generalized to N consecutive word types also called N-grams: $C(IN\ DENVER) = 2$ and $C(I\ WOULD\ LIKE) = 2$

Count histograms

How many events appear how many times?



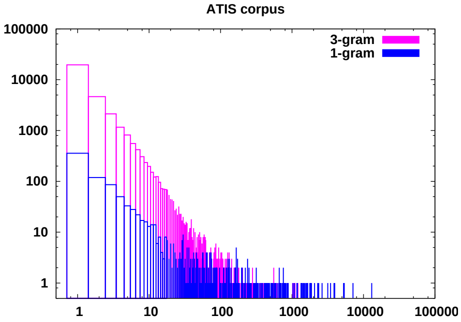
</s>	13044
TO	7123
THE	5378
FROM	5262
FLIGHT	3959
...	...
WORLD	1
WRONG	1
YET	1
ZONES	1

FROM BOSTON TO	1206
<s> SHOW ME	1199
TO SAN FRANCISCO	1101
...	...

- Few events appear very frequently
- Most events appear only a few times
- Many possible events never appear

Zipf’s law

Count histogram (frequency of frequencies) follows a specific distribution



Linear function in log-log scale defines a power law:

$\log(y) = -a \log(x) + \log b$
$\Leftrightarrow \log(y) = \log(bx^{-a})$
$\Leftrightarrow y = bx^{-a}$

N-gram probability

The conditional probability of observing a word w at a position i given a history $h = \ldots w_{i-2} w_{i-1}$ (or N-gram context) of preceding words

$$P(X_i = w_i \mid \ldots, X_{i-2} = w_{i-2}, X_{i-1} = w_{i-1}) \overset{notation}{=} P(w_i|h)$$

- The random variables X_i, X_{i-1}, X_{i-2}
  - are implicit in the shorter notation $P(w_i|h)$
  - take their value in a fixed vocabulary $w_i, w_{i-1}, w_{i-2}, \dots \in V$
- The model is assumed stationary
  - it does not depend on the position i in the text: $P(w_i|h) = P(w|h), \forall i$
- The history h (or left context) for an N-gram is made of the $N-1$ preceding words
- The word w is said to be predicted given the known history h

The term N-grams both refers to this probabilistic model and to the N-tuples used to estimate it

Sentence probability

$P(\textit{The rain in Spain falls mainly in the plain}) = ?$

Chain rule + N-gram assumption

For a general sentence of n words

$$\begin{aligned} P(w_1 \ldots w_n) &= P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1}) \\ &= \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \approx \prod_{k=1}^{n} P(w_k|w_{k-N+1}^{k-1}) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1 w_2)\,\underbrace{P(w_4|w_2 w_3)\ldots P(w_n|w_{n-2} w_{n-1})}_{\text{3-gram approximation}} \end{aligned}$$

3-gram sentence probability

$$\begin{aligned}&P(\textit{The rain in Spain falls mainly in the plain}) \approx \\ &\quad P(\textit{The}).P(\textit{rain} \mid \textit{The}).P(\textit{in} \mid \textit{The rain}).P(\textit{Spain} \mid \textit{rain in}). \\ &\quad P(\textit{falls} \mid \textit{in Spain}).P(\textit{mainly} \mid \textit{Spain falls}).P(\textit{in} \mid \textit{falls mainly}). \\ &\quad P(\textit{the} \mid \textit{mainly in}).P(\textit{plain} \mid \textit{in the})\end{aligned}$$

With start-of-sentence <s> and end-of-sentence </s> markers:

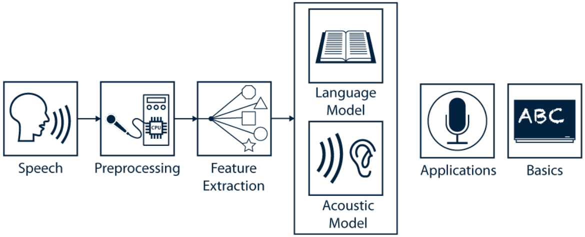
$$\begin{aligned}&P(<\text{s}>\ \textit{The rain in Spain falls mainly in the plain}\ <\text{/s}>) \approx \\ &\quad P(\textit{The} \mid <\text{s}>).P(\textit{rain} \mid <\text{s}>\ \textit{The}).P(\textit{in} \mid \textit{The rain}).P(\textit{Spain} \mid \textit{rain in}). \\ &\quad P(\textit{falls} \mid \textit{in Spain}).P(\textit{mainly} \mid \textit{Spain falls}).P(\textit{in} \mid \textit{falls mainly}). \\ &\quad P(\textit{the} \mid \textit{mainly in}).P(\textit{plain} \mid \textit{in the}).P(<\text{/s}> \mid \textit{the plain})\end{aligned}$$

Some language processing applications of N-grams

- Character or word completion in search engines, query systems, text messaging, source code editors, ...

what time ...	is ... it?	$P(is \mid what\ time)$
	has ... the ... least ... traffic?	$P(has \mid what\ time)$
	from ... now?	$P(from \mid what\ time)$

- Handwriting or speech recognition



- Text categorization
  N-grams can be seen as a specific version of a Naive Bayes classifier

Outline

Maximum likelihood estimation

$C(I\ like\ computers) = 3$ and $C(I\ like) = 10$

$$\Rightarrow \hat{P}(computers \mid I\ like) = \frac{3}{10}$$

Maximum likelihood

$$\hat{P}(w|h) = \begin{cases} \frac{C(h,w)}{C(h)} & \text{if } C(h) > 0 \\ 0 & \text{otherwise} \end{cases}$$

- $C(h, w)$ is the number of times the history h followed by w has been observed in the training set
- $C(h) = \sum_{w \in V} C(h, w)$ is the number of times the history h has been observed in the training set

Consistency property

$$\forall h, \sum_{w \in V} \hat{P}(w|h) = 1$$

Maximum likelihood estimation: Unigram case

Maximum likelihood

$$\hat{P}(w|h) = \begin{cases} \frac{C(h,w)}{C(h)} & \text{if } C(h) > 0 \\ 0 & \text{otherwise} \end{cases}$$

- For a 1-gram (unigram), the history h is empty
- $C(h,w) \Rightarrow C(w)$
- $C(h) = \sum_{w \in V} C(h,w) \Rightarrow \sum_{w \in V} C(w) = M$: the total number of word tokens in the training corpus (= the corpus length)

Example:

the word *plane* appears 10 times in a corpus of 1,000,000 tokens

$$\hat{P}(plane) = \frac{C(plane)}{M} = \frac{10}{10^6} = 10^{-5}$$

Why is smoothing necessary?

Maximum likelihood estimation assigns a zero probability to any unseen event in the training set ($C(h,w) = 0$)

$$\hat{P}(w|h) = \begin{cases} \frac{C(h,w)}{C(h)} & \text{if } C(h) > 0 \\ 0 & \text{otherwise} \end{cases}$$

An N-gram built on a vocabulary V defines $|V|^N$ possible events (combinations of N consecutive words).

- the number of parameters grows exponentially with the model order: $N = 3$ and $|V| = 10,000 \Rightarrow 10^{12}$ parameters
- even for very large training sets, many possible events are assigned a zero probability and the probability of observed events is overestimated
- smoothing techniques are required to correct the ML estimation and the consistency property still need to be satisfied!

Pseudo-counts and Laplace Smoothing

“Bayesian” estimation or additive smoothing

Define a priori pseudo-counts $C^*(h,w) > 0$ for any possible events

$$\hat{P}(w|h) = \frac{C(h,w) + C^*(h,w)}{C(h) + \sum_w C^*(h,w)}$$

- the prior probability of the event (h,w) is defined as $\frac{C^*(h,w)}{\sum_w C^*(h,w)}$ and $\hat{P}(w|h)$ can be interpreted (roughly) as a posterior estimate

Laplace smoothing: the add 1 method

Set $C^*(h,w) = 1$ for all events $\Rightarrow$ uniform priors

$$\hat{P}(w|h) = \frac{C(h,w) + 1}{C(h) + |V|}$$

(1749-1827)

Linear Interpolation

Build estimators for several model orders $\Rightarrow$ vary history length and combine them linearly

$$\hat{P}(w|h) = \lambda_0 \hat{P}_{ML}(w|h) + \lambda_1 \hat{P}_{ML}(w|h_{-1}) + \cdots + \lambda_p \hat{P}_{ML}(w|h_{-p})$$

- when $h = X_{i-N+1}, \ldots, X_{i-1}$ (N gram history) $\Rightarrow h_{-1} = X_{i-N+\mathbf{2}}, \ldots, X_{i-1}$ ($N-1$ gram history)
- Example:

  3-gram: $C(I\ like) = 10$ and $C(I\ like\ computers) = 3$

  $$\Rightarrow \hat{P}_{ML}(computers \mid I\ like) = \frac{3}{10}$$

  2-gram: $C(like) = 20$ and $C(like\ computers) = 7$

  $$\Rightarrow \hat{P}_{ML}(computers \mid like) = \frac{7}{20}$$

EM estimation of model weights

A Mixture Model

Linear interpolation smoothing is a model combining K estimators

$$\hat{P}(w|h) = \sum_{k=1}^{K} \lambda_k \hat{P}_k(w|h) \text{ with } \sum_k \lambda_k = 1$$

EM estimation of λ's

Use a validation corpus $S = \{w_1, \ldots, w_M\}$ (which can be seen as a single sequence) different from the training corpus

- Initialize: $\lambda_k = \frac{1}{K}$
  Example: $\lambda_1 = \lambda_2 = \lambda_3 = \frac{1}{3}$ when combining 3-grams, 2-grams and 1-grams
- E-step: $E(Model_k|S) = \sum_{j=1}^{M} \frac{\lambda_k \hat{P}_k(w_j|h)}{\sum_{k'=1}^{K} \lambda_{k'} \hat{P}_{k'}(w_j|h)}$
- M-step: $\lambda_k = \frac{E(Model_k|S)}{M}$

Backoff smoothing (initial idea)

Smoothing by discounting

$$\hat{P}(w|h) = \frac{C(h,w) - d_c}{C(h)} \text{ if } C(h,w) > 0$$

- Discounting corrects for the over-estimated probability of observed N-grams ($C(h,w) > 0$)
- The discounted probability mass will be used to correct for the under-estimated probability of N-grams not observed in the training corpus
  - Without smoothing, $C(h,w) = 0$ implies $\hat{P}_{ML}(w|h) = \frac{C(h,w)}{C(h)} = 0$

Estimation of the discounting parameters I

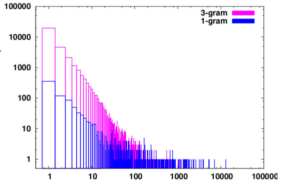
Good-Turing

- Optimal discounting coefficients d_c can be deduced by maximum likelihood estimate on leave-one-out (LOO) samples (N-grams appearing once in the original training set become unobserved)
- The Good-Turing estimate derives from this principled approach

$$C - d_c = (C+1)\frac{n_{C+1}}{n_C}$$

  - n_C the number of N-grams occurring C times

$n_2 \rightarrow$



$$\hat{P}_{ML}(happy|I\ \ am) = \frac{C(I\ \ am\ \ happy)}{C(I\ \ am)} = \frac{2}{1000} = .002$$

$$\hat{P}_{GT}(happy|I\ \ am) = \frac{C(h,w)-d_c}{C(h)} = \frac{C(h,w)+1}{C(h)}\frac{n_{c+1}}{n_c}$$

$$= \frac{3}{1000}\frac{2100}{4900} = 0.00086$$

Good-Turing discounting can be limited to small counts (*e.g.* $C(h,w) \leq 5$) and ML estimates used for larger ones

Estimation of the discounting parameters II

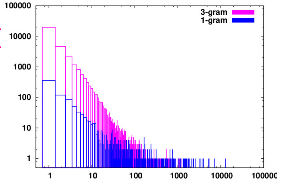
Absolute discounting

$$d_c \overset{\triangle}{=} d_*, \forall c$$

An optimal d_* discounting coefficent can be estimated from n_c the number of N-grams occurring c times

$$d_* = \frac{n_1}{n_1 + 2n_2}$$

$n_1 \rightarrow$
$n_2 \rightarrow$



- $d_* = \frac{20000}{20000+2\times4900} = 0.67$
-

$C(h,w)$	$C(h,w) - d_*$
1	0.33
2	1.33
3	2.33
...	
100	99.33

- In most cases, $n_1 > 0$ and $n_2 > 0 \Rightarrow 0 < d_* < 1$
- As for Good-Turing, the discounted value depends on the model order

Discounted probability mass

- $\hat{P}(happy|I\ \ am) = \frac{C(h,w)-d_c}{C(h)} = \frac{2-0.67}{1000} = \frac{1.33}{1000} = 0.00133$
  $\Rightarrow$ discounted probability = $\frac{0.67}{1000} = 0.00067$
- $\hat{P}(satisfied|I\ \ am) = \frac{C(h,w)-d_C}{C(h)} = \frac{3-0.67}{1000} = \frac{2.33}{1000} = 0.00233$
  $\Rightarrow$ discounted probability = $\frac{0.67}{1000} = 0.00067$
- $\hat{P}(\ldots|I\ \ am)$ = ...
  $\Rightarrow$ discounted probability = $\frac{0.67}{1000} = 0.00067$

Total discounted probability mass

- for a given history h (for example, $I\ \ am$), the total discounted mass is

$$\gamma(h) = \sum_{w:C(h,w)>0} \frac{d_c}{C(h)}$$

- the sum runs over all words observed at least once, after this history, in the training set

Backoff smoothing (Katz version)

Distribution of the discounted probability mass

$$\hat{P}(w|h) = \begin{cases} \frac{C(h,w)-d_c}{C(h)} & \text{if} \quad C(h,w) > 0 \\ \frac{1}{Z}\,\gamma(h)\;\hat{P}_{back}(w|h) & \text{if} \quad C(h,w) = 0 \end{cases}$$

- $\gamma(h) = \sum\limits_{w:C(h,w)>0} \frac{d_c}{C(h)}$ is the discounted probability mass from observed N-grams
- $\gamma(h)$ is distributed over **un**observed N-grams proportionally to a back-off distribution $\hat{P}_{back} = \hat{P}(w|h_{-1})$ (simplest case: 3-gram backoff = 2-gram)
- Z is a normalization factor to satisfy consistency ($\sum_{w\in V} \hat{P}(w|h) = 1$)

Backoff smoothing (alternative version)

Distribution of the discounted probability mass

$$\hat{P}(w|h) = \begin{cases} \frac{C(h,w)-d_c}{C(h)} + \gamma(h)\hat{P}_{back}(w|h) & \text{if} \quad C(h,w) > 0 \\ \gamma(h)\hat{P}_{back}(w|h) & \text{if} \quad C(h,w) = 0 \end{cases}$$

- the discounted mass can also be distributed over observed N-grams
  - because it gives a complementary estimate based on a lower order (less specific but more robustly estimated) model ($\hat{P}_{back}$)
- in this case, the normalization factor is simply $Z = 1$ (no longer shown)

An equivalent formulation

$$\hat{P}(w|h) = \max\left(0, \frac{C(h,w)-d_c}{C(h)}\right) + \gamma(h)\hat{P}_{back}(w|h)$$

Backoff smoothing (complete formulation)

$$\hat{P}(w|h) = \begin{cases} \max\left(0, \frac{C(h,w)-d_c}{C(h)}\right) + \gamma(h)\hat{P}_{back}(w|h) & \text{if} \quad C(h) > 0 \\ \hat{P}_{back}(w|h) & \text{if} \quad C(h) = 0 \end{cases}$$

- $\hat{P}_{back}(w|h)$ is smoothed using the same formula **recursively**
  - not with the same counts!
  - simplest case: 2-gram serves as backoff for 3-gram; 1-gram as backoff for 2-gram
- the recursion stops on 1-gram: $\hat{P}(w) = \frac{C(w)}{M}$ (M = # tokens in training) or "0-gram": $\hat{P}(w) = \frac{1}{|V|}$ ($|V|$=the vocabulary size)

Outline

Summary

- N-grams are simple yet efficient probabilistic language models used in continuous speech recognition, word/sentence completion tools, search engines, . . .
- The fewer training data you have, the better the smoothing ought to be
- N-gram models evaluation is directly related to a text completion task (see the perplexity metric)
    - Increasing the model order may result in less predictive models if smoothing is poor
    - $N = 3$ is often an optimal order, with sometimes a marginal gain with $N = 4$ or 5
- Many extensions or alternatives to the N-gram model exist

The dog which I saw across the street was barking at an old lady

They rarely outperform significantly a well-smoothed trigram

Software toolkits

- SRILM - The SRI Language Modeling Toolkit
  `www.speech.sri.com/projects/srilm/`

- NLTK Language Model Package
  `www.nltk.org/api/nltk.lm.html`

Further Reading

Jurafsky D. and Martin J.H.
Speech and Language Processing, 3rd edition (draft), chapter 3.

Katz S. (1987).
Estimation of probabilities from sparse data for the language model component of a speech recognizer.
IEEE Transactions on Acoustic, Speech and Signal Processing, **35**(3), 400–401.

Kneser R. and Ney H. (1995).
Improved backing-off for m-gram language modeling.
In *International Conference on Acoustic, Speech and Signal Processing*, p. 181–184, Detroit, Michigan.