COM4511 Speech Technology: Language Models

Anton Ragni

February 26, 2020

Language Models

- Statistical models capable of assigning probabilities to word sequences $\boldsymbol{w}_{1:L}$
  - compare and rank word sequences
  - sample/compute probability of next word
- Not all language models can do both tasks
  - bi-directional language models $P(w_l|\boldsymbol{w}_{1:l-1}, \boldsymbol{w}_{l+1:L})$
  - whole-sentence language models $P(s) = P(\boldsymbol{w}_{1:L})$
- No language model can do that perfectly

  How do you defend yourselves against a man armed with a ...?

	goldfish	banana	fruit	gun	knife	stick	thomas
English News	0.03	0	0	0.15	0.2	0.1	0
Monty Python	0.03	0.2	0.2	0	0	0	0

  - wider context/domain knowledge needed than just the current sentence
- Why language models?
  - find many uses in speech technology and natural language processing
  - examples: speech recognition, dialogue systems, machine translation, etc

SPandH

Anton Ragni

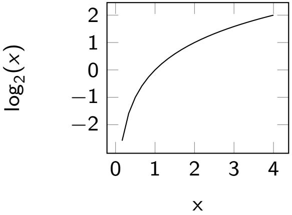
Vocabulary

- Words known to a language model constitute its vocabulary
  - need not be the set of modelling units: words, morphs, syllables, characters
  - unseen or out-of-vocabulary (OOV) words cannot be handled
- Word-based vocabularies
  - advantages:
    - relatively long modelling context
  - disadvantages:
    - new words emerge every day, "infinite" vocabularies
    - lack word relationship information
- Morph-based vocabularies
  - advantages
    - virtually lacks unseen words
    - enables to link related words
  - disadvantages:
    - relatively short modelling context
    - need to know how to decompose words
- Why not to use character based vocabularies?

SPandH

Anton Ragni

Entropy



Lookup table:

$\log_2(0)$	$=$	
$\log_2(1)$	$=$	0
$\log_2(2)$	$=$	1
$\log_2(3)$	$\approx$	1.585
$\log_2(4)$	$=$	2

- Entropy of probability distribution over discrete vocabulary is

$$\mathcal{H}_2(P) = -\sum_{w \in \mathcal{W}} P(w) \log_2(P(w))$$

  - a key measure of information in bits (originated in physics)
  - example: $\min \setminus \max \mathcal{H}_2(P)$, $\mathcal{H}_2(\frac{1}{2}, \frac{1}{2})$, $\mathcal{H}_2(\frac{3}{4}, \frac{1}{4})$, $\mathcal{H}_2(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, $\mathcal{H}_2(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$
  - BUT the "true" distribution P is rarely known

- Cross-entropy between "true" distribution P and estimated distribution Q

$$\mathcal{H}_2(P, Q) = -\sum_{w \in \mathcal{W}} P(w) \log_2(Q(w))$$

  - a key measure of information needed to describe P given Q

Anton Ragni

Perplexity

- Key measure of predictive performance for language models

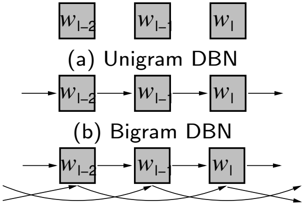
$$\mathrm{PPL}(P, Q) = 2^{\tilde{H}_2(P,Q)}$$

  - average number of choices Q makes to predict P
  - use empirical estimate of $H_2(P, Q)$ normalised by the number of words
- Examples:
  - relate $e^{\tilde{H}_e(P,Q)}$ and $10^{\tilde{H}_{10}(P,Q)}$ to $\mathrm{PPL}(P, Q)$
  - find min\max $\mathrm{PPL}(P)$, $\mathrm{PPL}(\frac{1}{2}, \frac{1}{2})$, $\mathrm{PPL}(\frac{3}{4}, \frac{1}{4})$, $\mathrm{PPL}(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, $\mathrm{PPL}(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$

- Only intrinsic performance measure yet extrinsic measures are of interest
  - speech recognition: word error rate (WER) — important to know correlation

Anton Ragni

N-gram Language Model



(a) Unigram DBN

(b) Bigram DBN

(c) Trigram DBN

Shakespear

1-gram: Every enter now severally so, let Will rash been and by I the me loves gentle me not slavish page, the and hour; ill let

2-gram: What means, sir. I confess she? then all sorts, he is trim, captain. The world shall- my lord!

3-gram: Indeed the duke; and had a very good friend. Sweet prince, Fallstaff shall die. Harry of Monmouth's grave.

4-gram: Enter Leonato's brother Antonio, and the rest, but seek the weary beds of people sick.

Example:

$w_{l-1} \backslash w_l$	a	b	c	d
a	0.1	0.5	0.3	0.1
b	0.7	0	0.3	0
c	0.4	0.1	0.2	0.3
d	0.5	0.4	0	0.1

Compute probability of `c,d,b,b,a,a`

- N-gram language model for general order n

$$P(w_1, \ldots, w_{L-1}, w_L) = \prod_{l=1}^{L} P(w_l | w_{l-1}, \ldots, w_l) \approx \prod_{l=1}^{L} P(w_l | w_{l-1}, \ldots, w_{l-n+1})$$

  - words further than $n-1$ positions in the past does not affect probability (Markov)
- Need to know how to compute n-gram probabilities: ML, neural networks!

Anton Ragni

Maximum Likelihood Estimate

- Probability of word w_l occurring after w_{l-1}

$$P(w_l|w_{l-1}) \triangleq \frac{c(w_{l-1}, w_l)}{\sum_w c(w_{l-1}, w)} = \frac{c(w_{l-1}, w_l)}{c(w_{l-1})}$$

  - $c(w_{l-1}, w)$ count of w_{l-1}, w (frequency of occurrence)
  - any unseen n-gram will be assigned zero probability — data sparsity
- Example: estimate conditional probabilities from

  `c,d,b,b,a,a, d,b,a,b,b,a, a,d,d,a,b,b,`

  - frequency table

aa	2	ba	3	ca	0	da	1
ab	2	bb	3	cb	0	db	2
ac	0	bc	0	cc	0	dc	0
ad	2	bd	0	cd	1	dd	1

  - alternatively, estimate joint probabilities and apply Bayes' rule

Anton Ragni

Smoothing

- Refine ML estimate by smoothing it with a prior

$$P(w_l|w_{l-1}) \triangleq \frac{c(w_{l-1}, w_l) + \tau Q(w_{l-1}, w_l)}{\sum_w c(w_{l-1}, w) + \tau}$$

  - multiple forms of prior probabilities/counts possible
- Popular examples:
  - Laplace smoothing

$$P(w_l|w_{l-1}) \triangleq \frac{c(w_{l-1}, w_l) + 1}{\sum_w c(w_{l-1}, w) + 1}$$

    - redistributes $\frac{|\mathcal{V}|}{N+|\mathcal{V}|}$ among unseen events
  - Add-δ smoothing
    - set $\tau < 1$ to be less "generous" to unseen events

Discounting

- Allocate probability to unseen events by reducing probability of seen events
  - event can be n-gram (joint) or seeing word w_l given past $n-1$ words (conditional)

$$P(w_{l-1}, w_l) = d(w_{l-1}, w_l)\frac{c(w_{l-1}, w_l)}{\sum_w c(w)} \quad \text{or} \quad P(w_l|w_{l-1}) = d(w_{l-1}, w_l)\frac{c(w_{l-1}, w_l)}{c(w_{l-1})}$$

  - numerous schemes available for setting $d(w_{l-1}, w_l)$
- Good-Turing discounting
  - assume unseen events are equiprobable to singleton events $P_0 \equiv P_1$
  - probability mass to redistribute among $r = 1, \ldots, R$ times occurring events

$$1 - P_1 = \sum_{r=1}^{R} P_{r+1} = \sum_{r=1}^{R} \frac{P_{r+1}}{P_r} P_r$$

  - equivalent to rescaling estimates by $\frac{P_{r+1}}{P_r}$, where $P_r = \frac{n_r r}{N}$, or using $r^* = (r+1)\frac{n_{r+1}}{n_r}$
- Example: apply Good-Turing discounting to the previous example
  - counts of counts $\boldsymbol{n} = [\qquad]$, adjusted counts $\boldsymbol{r}^* = [\qquad]$, mass $\boldsymbol{n}^\mathsf{T}\boldsymbol{r}^* =$
  - compare to ML estimates and discuss potential issues

Anton Ragni

Backoff

- Alternatively, obtain an estimate using less specific statistics ($n-1, n-2, \ldots$ order)

$$P(w_l|w_{l-1}) = \begin{cases} \frac{c(w_{l-1},w_l)}{c(w_{l-1})}, & \text{if} \quad c(w_{l-1},w_l) \geq C \\ d(w_{l-1},w_l)\frac{c(w_{l-1},w_l)}{c(w_{l-1})}, & \text{if} \quad 0 < c(w_{l-1},w_l) < C \\ \alpha(w_{l-1})P(w_l), & \text{otherwise} \end{cases}$$

  - typically includes discounting of any n-gram occurring less than cutoff value C
  - back-off weights are estimated to ensure valid probabilities (how?)
- Default approach for all n-gram language models used today

Anton Ragni

Interpolation

- Alternatively, interpolate ML-estimates of all available orders

$$P(w_l|w_{l-1}) = \lambda_2 \frac{c(w_{l-1}, w_l)}{c(w_{l-1})} + \lambda_1 \frac{c(w_l)}{\sum_w c(w)}$$

  - weights $\boldsymbol{\lambda} = \begin{bmatrix}\lambda_1 & \lambda_2\end{bmatrix}^{\mathsf{T}}$ must obey standard stochastic constraints
- Options available for estimating interpolation weights
  - greedy search: sweep through a range of values
  - deleted interpolation: enables optimal global or context-based weights
- Practical task will examine learning optimal interpolation weights
  - BUT we will interpolate different language models

Summary

- This lecture examined statistical language models
  - choice of modelling units
  - standard performance measures
- Focused on n-gram models
  - maximum likelihood estimates
  - estimate refinements
- Next lecture will examine more advanced language models
  - including using neural networks to estimate n-gram probabilities

Anton Ragni