INF5830 – 2013 FALL
NATURAL LANGUAGE PROCESSING

Jan Tore Lønning, Lecture 7, 26.9

Today

- (Maximum Likelihood Estimation, last week)
- N-gram models (FSNLP, sec. 6.1)
- Naive Bayes (NLTK, 6.1, 6.2, 6.5, FSNLP 7.0-7.2.1)
- NB two approaches (Information retrieval, ch. 13)
- Evaluation (NLTK, 6.3, FSNLP, 8.1, IR 13.6)
- Feature selection (IR 13.5)
- Smoothing (FSNLP 6.2)

N-gram models

FSNLP, sec. 6.1

Language models

- Goal:
  - Predicting a word given the words seen so far
  - $P(w_n \mid w_1\ w_2 \ldots w_{n-1})$
  - P(w | *this needs salt and*)
  - P(w | *she ate*)
- Useful for:
  - Speech recognition
  - Machine translation
  - Tagging
    - (tag sequences rather than words)

N-gram models

- Language model

$$P(w_1, w_2, ..., w_n) = P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_1 w_2) \times ... \times P(w_n \mid w_1, w_2, ..., w_{n-1}) = \prod_{i=1}^{n} P(w_i \mid w_1^{i-1})$$

- Markov assumption
  - Bigram model

$$P(w_1, w_2, ..., w_n) \approx P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_2) \times ... \times P(w_n \mid w_{n-1}) = \prod_{i=1}^{n} P(w_i \mid w_{i-1})$$

  - Trigram model

$$P(w_1, w_2, ..., w_n) \approx \prod_{i=1}^{n} P(w_i \mid w_{i-2} w_{i-1})$$

  - Unigram model

$$P(w_1, w_2, ..., w_n) \approx \prod_{i=1}^{n} P(w_i)$$

ML Estimation

- A training corpus with N words
- $$\hat{P}(w_i) = \frac{C(w_i)}{N}$$
  - where $C(w_i)$ is the number of occurrences of w_i
- $$\hat{P}(w_i \mid w_j w_k) = \frac{C(w_j w_k w_i)}{C(w_j w_k)}$$
  - where $C(w_i w_k w_i)$ is the number of occurrences of $w_i w_k w_i$, etc.

Naive Bayes

NLTK, 6.1, 6.2, 6.5

FSNLP 7.0-7.2.1

IR, ch 13

Naive Bayes

- Given an object
  - $\langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle$
- Consider
  - $P(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle)$ for each class sm
- Choose the class with the largest value, in symbols

$$\arg\max_{s_m \in S} P(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle) \approx \arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P(f_i = v_i \mid s_m)$$

Naive Bayes (Bernoulli): Training

- Maximum Likelihood
- $\hat{P}(s_m) = \frac{C(s_m, o)}{C(o)}$
  - where $C(s_m, o)$ are the number of occurrences of objects o in class s_m
- $\hat{P}(f_i = v_i \mid s_m) = \frac{C(f_i = v_i, s_m)}{C(s_m)}$
  - where $C(f_i=v_i, s_m)$ is the number of occurrences of objects o
    - where the object o belongs to class s_m
    - and the feature f_i takes the value v_i
  - $C(s_m)$ is the number of occurrences belonging to class s_m
- Observe: this is different from FSNLP

The two models

- Bernoulli
  - The standard form of NB
    - NLTK,
    - Jurafsky and Martin
- Multinomial model
  - For text classification
    - FSNLP,
    - ”Introduction to information retrieval” (both)
  - Related to n-gram models

Multinomial text classification

- Build a language model for each class
- Score the document according to the different classes
- Choose the class with the best score

Multinomial NB: Training

- $\hat{P}(s_m) = \frac{C(s_m, o)}{C(o)}$
  - where $C(s_m, o)$ is the number of occurrences of objects o in class s_m
- $\hat{P}(w_i \mid s_m) = \frac{C(w_i, s_m)}{\sum_j C(w_j, s_m)}$
  - where $C(w_i, s_m)$ is the number of occurrences of word w_i in all texts in class s_m
  - $\sum_j C(w_j, s_m)$ is the total number of words in all texts in class s_m
- Bernoulli counts the number of objects/texts where w_i occurs
- Multinomial counts the number of occurrences of w_i.

Multinomial NB: Decision

$$\arg\max_{s_m \in S} P\left(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle\right) \approx \arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P\left(f_i = v_i \mid s_m\right)$$

- In the multinomial model
  - f_i refers to position i in the text
  - v_i refers to the word occurring in this position
- We assume a word to be equally likely in all positions:

$$\arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P\left(f_i = v_i \mid s_m\right) = \arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P\left(v_i \mid s_m\right)$$

Comparison

Bernoulli

- Registers whether a term is present or not
- Considers both
  - The present terms
  - The absent terms
- Suitable for various tasks

Multinomial

- Counts how many times a term is present
- Considers
  - only the present terms
  - Ignores absent terms
- Tailor-made for text classification

Evaluation

NLTK, 6.3, FSNLP, 8.1, IR 13.6

Evaluation measures

		Is in C	
		Yes	NO
Classifier	Yes	tp	fp
	No	fn	tn

- Accuracy: (tp+tn)/N
- Precision:tp/(tp+fp)
- Recall: tp/(tp+fn)
- F-score $F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}}$

$$F = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{R+P} = \frac{2\,\frac{tp}{tp+fp}\,\frac{tp}{tp+fn}}{\frac{tp}{tp+fp} + \frac{tp}{tp+fn}} = \frac{2\,tp}{tp+fn+tp+fp}$$

Feature selection

IR 13.5

Feature selection

- To include all words as features is too costly
  - – in particular for the Bernoulli model
- Select features which
  - Separate between the classes
  - Expected to occur with a reasonable frequency

- To select: use similar measures as for collocations, e.g.
  - Raw frequency
  - Chi-square
  - Mutual information
  - …

Chi square

		Sense musical instrument		
		Yes	No	sums
”guitar” in context	Yes	25	5	30
	No	275	695	970
	Sums	300	700	1000

$$X^2 = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

Chi square

Observations

		Is in class s		
		Yes	NO	sums
w in context	Yes	O_{11}	O_{10}	O_{1x}
	No	O_{01}	O_{00}	O_{0x}
	Sums	O_{x1}	O_{x0}	N

Expectations

		Is in class s		
		Yes	NO	sums
w in context	Yes	$E_{11}=O_{1x}\times O_{x1}/N$	$E_{10}=O_{1x}\times O_{x0}/N$	O_{1x}
	No	$E_{01}=O_{0x}\times O_{x1}/N$	$E_{00}=O_{0x}\times O_{x0}/N$	O_{0x}
	Sums	O_{x1}	O_{x0}	O

Chi square

Observations

		Sense musical instrument		
		Yes	NO	sums
”guitar” in context	Yes	25	5	30
	No	275	695	970
	Sums	300	700	1000

Expectations

		Is in class s		
		Yes	NO	sums
w in context	Yes	9=30×300/1000	21=30×700/1000	30
	No	291	679	970
	Sums	300	700	O

Apply chi square

- Assume a binary classifier: only two classes {yes, no}
- For every word calculate the chi square score between the word and the class
- Select the words with the highest score as features

- Similar to collocations
- We may use other association measures
- Trade-off: discrimination/frequency

Pointwise mutual information

$$I(W=1;C=1) = \log \frac{\ddot{P}(W=1,C=1)}{\hat{P}(W=1)\hat{P}(C=1)}$$

$$= \log \frac{\frac{O_{11}}{N}}{\frac{O_{x1}}{N}\frac{O_{1x}}{N}} = \log \frac{N\,O_{11}}{O_{x1}O_{1x}}$$

$$= \log \frac{O_{11}}{E_{11}}$$

Mutual information

$$I(W;C) = \sum_{i=0,1;\, j=0,1} \hat{P}(W=i, C=j) \log \frac{\hat{P}(W=i, C=j)}{\hat{P}(W=i)\hat{P}(C=j)}$$

$$= \sum_{i=0,1;\, j=0,1} \frac{O_{ij}}{N} \log \frac{NO_{ij}}{O_{xi}O_{jx}} = \sum_{i=0,1;\, j=0,1} \frac{O_{ij}}{N} \log \frac{O_{ij}}{E_{ij}}$$

Smoothing

FSNLP 6.2

Shakespeare as a Corpus

- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams...
  - So, 99.96% of the possible bigrams were never seen (have zero entries in the table)
  - This is the biggest problem in language modeling; we'll come back to it.
- Does that mean that any sentence that contains one of those bigrams should have a probability of 0?

Laplace Smoothing

- Also called add-one smoothing
- Just add one to all the counts!
- Very simple
- MLE estimate: $P(w_i) = \frac{c_i}{N}$
- Laplace estimate: $P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$
- Reconstructed counts: $c_i^* = (c_i + 1)\frac{N}{N + V}$

Laplace-Smoothed Bigram Counts

	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

Laplace-Smoothed Bigram Probabilities

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

	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

Reconstituted Counts

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

	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

Big Change to the Counts!

- C(count to) went from 608 to 238!
- P(to|want) from .66 to .26!
- Discount d= c*/c
  - d for "chinese food" =.10!!! A 10x reduction
  - So in general, Laplace is a blunt instrument
  - Could use more fine-grained method (add-k)
- But Laplace smoothing not used for N-grams, as we have much better methods
- Despite its flaws Laplace (add-k) is however still used to smooth other probabilistic models in NLP, especially
  - For pilot studies
  - in domains where the number of zeros isn't so huge.

Held-out estimation

- Split training set in 2
  - Training: Set1
  - Held-out: Set2
- Instances unseen in Set1:
  - How frequent are they in set2?
    - $T_0 = \sum_{C_1(o)=0} C_2(o)$
  - Probability for each item
    - $P(o) = T_0 / N_0 T$
- For all classes
  - $T_r = \sum_{C_1(o)=r} C_2(o)$
  - $P(o) = T_r / N_r T$

- o an object
- $C_1(o)$ number of occurrences of o in set1
- $C_2(o)$ similarly in set2
- N: number of objects in Set1
- T: number of objects in Set2
- N_r: number of objects with r occurrences in set1

Better Smoothing

- Intuition used by many smoothing algorithms
  - Good-Turing
  - Kneser-Ney
  - Witten-Bell

- Is to use the count of things we've seen once to help estimate the count of things we've never seen

Good-Turing

- Instances unseen in Set1:
  - Give them a part of size N_1
  - The adjusted frequency for an unseen item
    - $0_{GT} = N_1/N_0$
  - The probability for an item
    - $P(o) = N_1/N_0 N$
- For objects with r>0 occurrences:
  - Adjusted score
    - $r^* = (r+1)N_{r+1}/N_r$
  - Probability
    - $P(o) = r^*/N$

- o an object
- $C_1(o)$ number of occurrences of o in set1
- N: number of objects in Set1
- N_r: number of objects with r occurrences in set1

GT-problems and refinements

- What if N_r+1 is empty?
- A simpl(ified) solution:
  - Only smooth for r<k for some k (e.g. 5 or 10)
  - Trust real counts for larger r-s
  - Assuming large data so that N_r is large for r<k
- Consider table 6.4 in FSNLP