

SDS Exam 2 Notes

Table of Contents

- [SDS Exam 2 Notes](#)
- [1. 7 - Kappa](#)
 - [1.1. Inter-Rater Reliability](#)
 - [1.2. Agreement Calculations](#)
 - [1.3. Cohen's Kappa](#)
 - [1.4. Applications](#)
 - [1.5. Weighted Kappa](#)
 - [1.6. Other Inter-Rater Reliability Methods](#)
- [2. 8 - Dialogue System Evaluation](#)
 - [2.1. Dialogue Evaluation](#)
 - [2.2. PARADISE Framework](#)
 - [2.3. Experimental Procedures](#)
 - [2.4. Success Metric](#)
- [3. 9 - Basic Text Processing](#)
 - [3.1. Regular Expressions](#)
 - [3.2. Word Tokenization](#)
 - [3.3. Sentence Segmentation and Decision Trees](#)
 - [3.4. Minimum Edit Distance](#)
- [4. 10 - Language Modeling](#)
 - [4.1. Probabilistic Language Models](#)
 - [4.2. Chain Rule](#)
 - [4.3. Applied Chain Rule](#)
 - [4.4. Markov Assumption](#)
 - [4.5. Estimating Bigram Probabilities](#)
 - [4.6. Evaluation](#)
 - [4.7. Perplexity](#)
 - [4.8. Generalization](#)
 - [4.9. Zeros](#)
 - [4.10. Laplace Add-One Smoothing](#)
- [5. 11 - Naive Bayes](#)
 - [5.1. Text Classification](#)
 - [5.2. Naive Bayes Classifier](#)
 - [5.3. Multinomial Naive Bayes](#)
 - [5.4. Learning](#)
 - [5.5. Relationship to Language Modeling](#)

- [5.6. Naive Bayes Evaluation](#)
- [5.7. Cross-Validation](#)
- [5.8. Harms in Classification](#)

1. 7 - Kappa

1.1. Inter-Rater Reliability

- Dialogue Act Classification
 - can be straightforward, i.e. question, declaration, apology
 - can be subject to interpretation
 - yeah, right - agreement or sarcasm?
 - what!? - question, exclamation, or reaction?
 - **solution** - test how well two people agree on given dialogue acts
 - **inter-rater reliability**
- **inter-rater reliability** - degree of agreement between raters where raters work independently of each other
 - application - *validation* of rating protocols
- useful when rating protocols are ambiguous
 - applying dialogue act tags
 - codes from thematic analysis
 - judging the quality of something

1.2. Agreement Calculations

- **agreement** - probability that you and your partner selected the same tag for an item on the list
 - $agreement = \frac{count(item\ rated\ the\ same)}{count(item)}$
- **observed vs. expected agreement** - determine what agreement was likely due to chance
 - **observed agreement** - probability that items were rated the same

$$P(items\ rated\ the\ same)$$
 - **expected agreement** - sum over all ratings
 - $P(item\ rated\ by\ both\ as\ X)$
 - $= P(judge\ 1\ rated\ X \cap judge\ 2\ rated\ X)$
 - if judges rated independently
 - $P(judge\ 1\ rated\ X) * P(judge\ 2\ rated\ X)$
- example
 - rate 20 items good or bad
 - rater 1 rated 1 item bad rest good
 - rater 2 rated 2 items bad rest good

- all the bad rates, the other rater rated that item as good
- observed agreement = $17 / 20 = 0.85$
- expected agreement - make table where entry is the count that the rater rated items that class out of all items

	Rater 1	Rater 2
Bad	0.05	0.10
Good	0.95	0.90

- bad = $0.05 \times 0.10 = 0.005$
- good = $0.95 \times 0.90 = 0.855$
- total = $0.855 + 0.005$

1.3. Cohen's Kappa

- measures the degree to which two raters' agreement exceeds chance
 - $k = \frac{O-E}{1-E}$
- O is observed agreement, E expected agreement
- from previous example

Raw Frequencies				Relative Frequencies			
		Rater 2				Rater 2	
		B	G			B	G
Rater 1	B	0	1		B	0	.05
	G	2	17		G	.1	.85
		2	18			.1	.9

Divide by
total ratings

- $O = 0 + 0.85 = 0.85$
- $E = (0.05 \times 0.1) + (0.95 \times 0.9) = 0.86$
- $k = (0.85 - 0.86) / (1 - 0.86) = -0.071$, poor agreement
- kappa ranges from -1 to 1
 - $k > 0$ indicates agreement better than chance
 - $k = 1$ perfect agreement
 - $k < 0$ indicates agreement worse than chance
 - $k = -1$ perfect disagreement and 50% expected agreement
 - applicable when data are *nominal* and *unordered*

- **Score Interpretation**

Score	Interpretation
< 0	poor
0 - 0.2	slight
0.2 - 0.4	fair
0.41 - 0.6	moderate
0.61 - 0.8	substantial
0.81 - 1	almost perfect

- example

		Rater 2			
		B	G	Meh	
Rater 1	B	5	1	0	6
	G	1	9	1	11
	Meh	1	1	1	3
		7	11	2	

		Rater 2			
		B	G	Meh	
Rater 1	B	.25	.05	0	.3
	G	.05	.45	.05	.55
	Meh	.05	.05	.05	.15
		.35	.55	.1	

-
- $O = 0.25 + 0.45 + 0.05 = 0.75$
- $E = (0.3 \times 0.35) + (0.55 \times 0.55) + (0.15 \times 0.1) = 0.4225$
- $k = (0.75 - 0.4225) / (1 - 0.4225) = 0.57$, moderate agreement

1.4. Applications

- dialogue act classification
 - define a set of dialogue tags and detailed descriptions for each one
 - train secondary annotators on how to use your tagging scheme
 - calculate kappa on subset of data (generally around 20%)

- if kappa is too low, retrain and repeat
- standard practices for corpus-based research
 - one or more annotators tag entire corpus split across each annotator
 - kappa computed on double-tagged portion of corpus, around 20%
 - kappa of around 0.8 is generally acceptable for dialogue act tags
 - lower kappas are acceptable depending on the task
 - tagging uncertainty, disengagement, etc

1.5. Weighted Kappa

- **weighted kappa** - accounts for degree of disagreement
- useful when ratings are ordered
 - i.e. disagreement between good and bad should have more weight than disagreement between good and meh
- consists of **3 matrices**
 - observed agreement matrix
 - expected agreement matrix
 - weight matrix
- **observed agreement matrix** - same as the contingency matrix = X
- **expected agreement matrix** - probabilities for each pair of ratings = M
 - $m_{ij} = \frac{(\text{rater 1's } i \text{ ratings}) \times (\text{rater 2's } j \text{ ratings})}{\text{total data points}}$
- **weight matrix** - each cell in the contingency matrix = W
 - matrix diagonal is zero, no penalty for agreement
 - other weights determined by distance between ratings
 - good/meh and meh/bad = 1, good/bad = 2
- $k = 1 - \frac{\sum \sum w_{ij} x_{ij}}{\sum \sum w_{ij} m_{ij}}$
 - sum of products of weight and observed agreement matrices divided by sum of products of weight and expected agreement matrices

1.6. Other Inter-Rater Reliability Methods

- **Fleiss' kappa** - multiple raters, ordinal data
 - alternative - average pairwise Cohen's kappa
- **Pearson's correlation coefficient and Spearman's rank correlation coefficient** - used for continuous data
- **Krippendorff's alpha** - generalizable to multiple raters and data types
- **Cronbach's alpha** - validating psychometric test items

2. 8 - Dialogue System Evaluation

2.1. Dialogue Evaluation

- things we can measure about how well a dialogue went
 - user satisfaction
 - learning
 - task completion
 - how long they stayed with it
- outcomes
 - tell us how well a dialogue went
 - can be represented numerically in some way and then predicted based on what happened within the dialogues themselves
 - you need to keep records of what happened in the dialogues themselves

2.2. PARADISE Framework

- used to evaluate dialogue systems
- *performance* of a dialogue system is affected by both:
 - *what* gets accomplished by the user and the dialogue agent and
 - *how* it gets accomplished
- maximize user satisfaction
 - maximize task success
 - minimize costs
 - efficiency measures
 - qualitative measures
- regress against user satisfaction
 - questionnaire to assign each dialogue a user satisfaction rating - *dependent* measure
 - cost and success factors - *independent* measures
 - use regression to train weights for each factor

2.3. Experimental Procedures

- subjects given specific tasks
- spoken dialogues recorded
- cost factors, states, dialogue acts automatically logged
- ASR accuracy, barge-in hand-labeled
- users specify task solution via web page
- users complete user satisfaction survey of some kind
- use **multiple linear regression** to model user satisfaction as a function of task success and costs
 - test for significant predictive factors

2.4. Success Metric

- could we use the success metric to drive automatic learning?
- methods for automatically evaluating system performance
- way of obtaining training data for further system development
- can we find intrinsic evaluation metrics that correlate with extrinsic results?

3. 9 - Basic Text Processing

3.1. Regular Expressions

- formal language for specifying text strings
- process based on fixing two kinds of errors
 - matching strings that we should not have matched (there, then, other)
 - *false positives*
 - not matching things that we should have matched (the)
 - *false negatives*
- sophisticated sequences of regular expressions are often the first model for any text processing
 - therefore play a large role
- for many hard tasks, use machine learning classifiers
 - but regular expressions are used as features in the classifiers
 - can be very useful in capturing generalizations

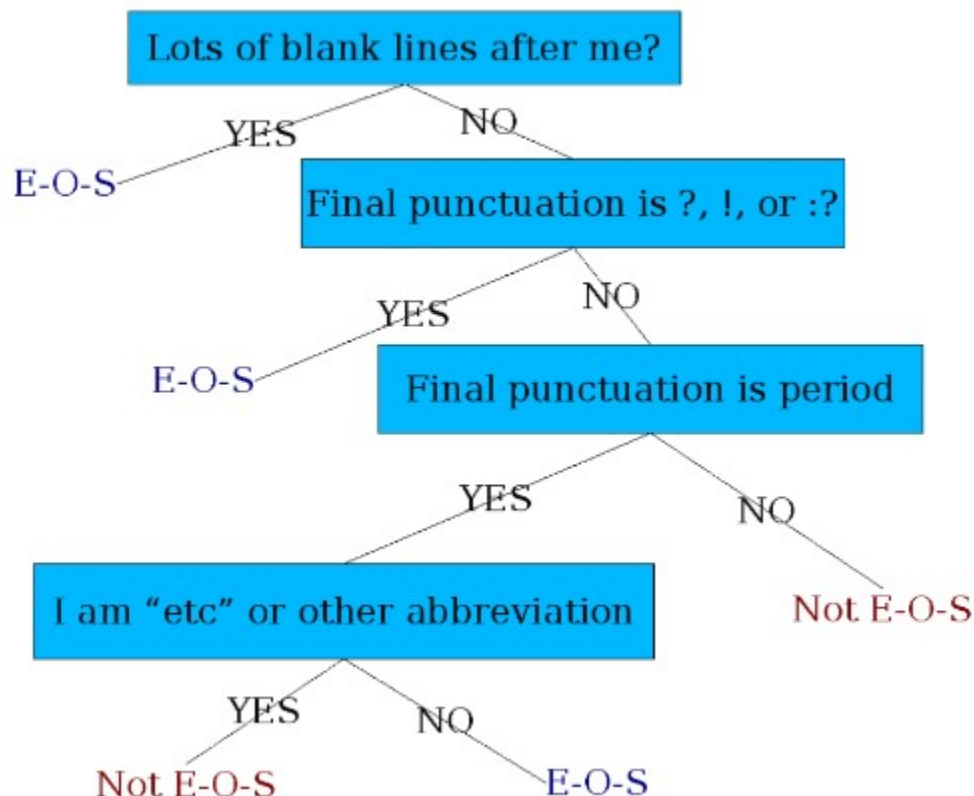
3.2. Word Tokenization

- **text normalization**
 1. segmenting/tokenizing words in running text
 2. normalizing word formats
 3. segmenting sentences in running text
- can be hard to determine how many words are in an utterance
 - "I do uh main- mainly business data processing" - fragments, filled pauses
 - "Suess's cat in the hat is different from other cats!"
 - lemma - same stem, part of speech, rough worse sense
 - cat and cats = same lemma
 - wordform - the full inflected surface form
 - cat and cats = different wordforms
 - "they lay back on the San Francisco grass and looked at the stars and their"
 - **type** - an element of the vocabulary
 - **token** - an instance of that type in running text
 - 15 tokens, 13 types
- issues in tokenization
 - "Finland's capital" -> Finland, Finlands, Finland's?
 - "what're, I'm, isn't" -> what are, I am, is not

- "Hewlett-Packard" -> Hewlett Packard?
- "state-of-the-art" -> state of the art?
- "Lowercase" -> lower-case, lowercase, lower case?
- "San Francisco" -> one token or two?
- "m.p.h., PhD." -> ??
- **normalization** - break words down to their equivalence classes of terms
 - information retrieval - indexed text and query terms must have same form, i.e. match U.S.A and USA as the same
 - implicitly define equivalence classes of terms
 - i.e. deleting periods in a term
 - *alternative* - asymmetric expansion
 - enter: window, search: window, windows
 - enter: windows, search: Windows, windows, window
 - enter: Windows, search: Windows
 - potentially more powerful, but less efficient
- **case folding** - reduce all letters to lower case
 - users tend to use lower case
 - possible exception - upper case in mid-sentence?
 - i.e. General Motors, Fed vs fed, SAIL vs sail
 - for sentiment analysis, MT, information extraction, case is helpful
 - US vs us is important
- **lemmatization** - reduce inflections or variant forms to base form
 - am, are, is -> be
 - car, cars, car's, cars' -> car
 - the boy's cars are different colors -> the boy car be different color
 - have to find correct dictionary headword form
 - machine translation
- **morphology**
 - **morphemes** - small meaningful units that make up words
 - *stems* - core meaning-bearing units
 - *affixes* - bits and pieces that adhere to stems
 - often with grammatical functions
- **stemming** - crude chopping of affixes
 - goal is to reduce terms to their stems in information retrieval
 - language dependent
 - automate, automatic, automation all reduced to automat
 - **Porter's algorithm** - most common English stemmer
 - only strip -ing if there is a verb
 - walking -> walk
 - sing -> sing

3.3. Sentence Segmentation and Decision Trees

- **sentence segmentation** - meaning of punctuation
 - !, ? are relatively unambiguous
 - . is quite ambiguous
 - sentence boundary
 - abbreviations (Dr., Inc, etc)
 - numbers (.02, 4.3)
 - build a binary classifier
 - looks at a .
 - decides end of sentence or not end of sentence
 - *classifiers* - hand-written rules, regular expressions, or machine learning
- use a **decision tree** to determine if a word is end-of-sentence



- - more sophisticated decision tree features
 - word with period - upper, lower, caps, number
- implementing decision trees
 - decision tree is just an *if else* statement
 - interesting research is choosing the features
 - setting up the structure is often too hard to do by hand
 - hand building only possible for very simple features, domains
 - for numeric features, it's too hard to pick each threshold
 - instead, structure usually learned by machine learning from a training corpus

- think of the questions in a decision tree as *features* that could be exploited by any kind of classifier
 - logistic regression
 - SVM
 - neural nets, etc

3.4. Minimum Edit Distance

- **minimum edit distance** - minimum number of editing operations between two strings to transform one into the other
- **editing operations** - insert, delete, substitution
- example


```

      I N T E * N T I O N
      | | | | | | | | |
      * E X E C U T I O N
    
```

 - d s s i s
 - strings need to be *aligned*
 - if each operation has cost of 1, distance between the two is 5
 - if substitutions cost 2, distance between them is 8
- other uses in NLP
 - evaluating machine translation and speech recognition
 - named entity extraction and entity co-reference
- finding min edit distance
 - search for path (sequence of edits) from the start string to the final string
 - *initial state* - word we are transforming
 - *operators* - insert, delete, substitute
 - *goal state* - word we are trying to get to
 - *path cost* - what we want to minimize, the number of edits
 - space of all edit sequences is huge
 - cannot afford to navigate naively
 - lots of distinct paths wind up at the same state, therefore we don't have to keep track of all of them, just the *shortest path* to each of those revised states
- **dynamic programming** - solving problems by combining solutions to subproblems
 - use it for a tabular computation of $D(n, m)$
 - *bottom-up* - we compute $D(i, j)$ for small i, j , and compute larger $D(i, j)$ based on previously computed smaller values
- Levenshtein
 - initialization - $D(i, 0) = i$, $D(0, j) = j$
 - recurrence relation

```

For each i = 1...M
  For each j = 1...N
    D(i,j) = min {
      D(i-1,j) + 1
      D(i,j-1) + 1
      D(i-1,j-1) + 2; if X(i) ≠ Y(j)
      0; if X(i) = Y(j)
    }

```

-
- termination - $D(N, M)$ is distance
- create an edit distance table
- computing alignments
 - edit distance isn't sufficient
 - often need to *align* each character of the two strings to each other
 - do this by keeping a **backtrace**
 - every time we enter a cell, remember where we came from
 - when we reach the end, trace back the path from the upper right corner to read off the alignment
 - do this through the table
 - label each part of the path with a symbol
 - left = insertion
 - down = deletion
 - diagonal = substitution
 - an optimal alignment is composed of optimal subalignments
 - honestly just look at the slides for these looking at the tables and then transitioning makes it a lot easier to understand
 - **performance**
 - *time* - $O(nm)$
 - *space* - $O(nm)$
 - *backtrace* - $O(n+m)$
- **weighted edit distance** - add weights to the computation
 - *spell correction* - some letters are more likely to be mistyped than others
 - *biology* - certain kinds of deletions or insertions are more likely than others
- alignments in 2 fields
 - **NLP** - generally talk about *distance* (minimized) and *weights*
 - **Computational Biology** - generally talk about *similarity* (maximized) and *scores*
- Needleman-Wunsch - start at top left corner for edit table instead of bottom left
- variant of basic algorithm - might be ok to have unlimited number of gaps in the beginning and end
 - if so, we do not want to penalize gaps at the ends
- Smith-Waterman algorithm
 - ignore badly aligned regions
 - modify Needleman-Wunsch
 - want to have local alignment

4. 10 - Language Modeling

4.1. Probabilistic Language Models

- **goal** - assign a probability to a sentence
 - *machine translation* - $P(\text{high winds tonight}) > P(\text{large winds tonight})$
 - *spell correction* - the office is about fifteen minuets from my house
 - $P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$
 - *speech recognition* - $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$
 - summarization, question-answering, etc
 - $P(W) = P(w_1, w_2, w_3, w_4, w_5)$
- **related task** - probability of an upcoming word
 - $P(w_5 | w_1, w_2, w_3, w_4)$
- **language model (LM)** - model that computes either of the two formulas
 - also called *grammar*
- how do we compute $P(W)$?
 - rely on **Chain Rule of Probability**

4.2. Chain Rule

- definitions of conditional probabilities
 - $P(B|A) = \frac{P(A,B)}{P(A)}$
 - $P(A, B) = P(A)P(B|A)$
- **general equation**
 - $P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)\dots P(x_n|x_1, \dots, x_{n-1})$

4.3. Applied Chain Rule

- **applied** to *joint probability of words* in a sentence
 - $P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$
 - sidenote: latex sucks so if you see \prod_i that means i is the bound, not multiplying the rest of the stuff by i
- example: $P(\text{"its water is so transparent"})$
 - $= P(\text{its}) \times P(\text{water}|\text{its}) \times P(\text{so}|\text{its water is}) \times P(\text{transparent}|\text{its water is so})$
- **naive estimation** - count and divide
 - $P(\text{the}|\text{its water is so transparent that}) = \frac{\text{Count}(\text{its water is so transparent that the})}{\text{Count}(\text{its water is so transparent that})}$
 - but there are *way too many* possible sentences
 - never see enough data for estimating

4.4. Markov Assumption

- *simplify* assumption
- approximate each component in the product
 - $P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-k} \dots w_{i-1})$
- **unigram model** - simplest case
 - $P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$
- **bigram model** - condition on the previous word
 - $P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-1})$
- **n-gram models** - can extend to trigrams, 4-grams, etc
 - in general this is an *insufficient model of language* because language has **long-distance dependencies**
 - words that have meaning tied with another part of the sentence may be many many words separated
 - we can often get away with n-gram models though

4.5. Estimating Bigram Probabilities

- **maximum likelihood estimate**
 - count abbreviated to c in following formulas
 - $P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$
- example:
 - I am Sam. Sam I am. I do not like green eggs and ham.
 - $P(\text{Sam} | \text{am}) = 1/2$
 - $P(\text{am} | \text{I}) = 2/3$
 - $P(\text{do} | \text{I}) = 1/3$
- **raw bigram count table** - (row, column) is count of times that row column appears in the given sentences
 - to get probabilities, normalize by the unigrams
 - see HW4
- **practical issues** - we do everything in log space
 - avoid *underflow*
 - adding is faster than multiplying
 - $\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$

4.6. Evaluation

- does our language model prefer *good* sentences to *bad* ones?
- assign higher probability to *real* or *frequently observed* sentences than *ungrammatical* or *rarely observed* sentences
- train parameters of the model on a **training set**

- test the model's performance on data it has not seen
 - **test set** - unseen dataset that is different from the training set, totally unused
 - **evaluation metric** - how well the model does on the test set
- training on the **test set**
 - cannot allow test sentences in the training set
 - assign it an artificially high probability when we set it in the test set
 - training on the test set is bad science and violates the honor code
- **extrinsic** evaluation of **n-gram models** - best evaluation for comparing models A and B
 - put each model in a task, such as spelling corrector, speech recognizer, MT system, etc
 - run the task, get an *accuracy* for A and for B
 - how many misspelled words corrected properly
 - how many words translated correctly
 - etc
 - compare accuracy for A and B
 - **difficulty** - time-consuming, can take days or weeks
- **intrinsic** evaluation - **perplexity**
 - bad approximation, unless test data looks just like the training data
 - generally only useful in *pilot experiments*
 - helpful to think about though

4.7. Perplexity

- **Shannon Game** - how well can we predict the next word?
 - unigrams are terrible at this due to only calculating the probability of a word, not with context in sentence
 - a better model of text is one which assigns a higher probability to the word that actually occurs
- **best language model** is one that best predicts an unseen test set, so it gives the highest $P(\text{sentence})$
- **perplexity** - inverse probability of the test set, *normalized* by the number of words
- **!!!** minimizing perplexity is the same as maximizing probability **!!!**
- equations (I hope we don't need to memorize these...)

Chain rule:

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_1 \dots w_{i-1})}}$$

For bigrams:

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_{i-1})}}$$

- perplexity as a **branching factor**

- example: sentence consists of random digits
 - perplexity of the sentence according to a model that assigns $P=1/10$ to each digit?

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

$$PP(W) = \left(\frac{1}{10}\right)^N)^{-\frac{1}{N}} = \frac{1}{10}^{-1} = 10$$

- lower perplexity = *better model*

4.8. Generalization

- **Shannon Visualization Method**

- choose a random bigram ($\langle s \rangle, w$) according to its probability
- now choose a random bigram (w, x) according to its probability
- and so on until we choose $\langle /s \rangle$
- then string the words together

- **perils of overfitting** - N-grams only work well for word prediction if the test corpus looks like the training corpus

- in reality, it often does not
- need to train robust models that *generalize*
- one kind of generalization - **zeros**
 - things that do not ever occur in the training set, but occur in the test set

4.9. Zeros

- training set:

- ...denied the allegations
- ...denied the reports
- ...denied the claims
- ...denied the request

- test set:

- ...denied the offer
- ...denied the loan

- $P(\text{"offer"} \mid \text{denied the}) = 0$

- **zero probability bigrams** - bigrams with zero probability that means we will assign 0 probability to the test set

- and thus we cannot compute perplexity, we cannot divide by 0

4.10. Laplace Add-One Smoothing

- **smoothing intuition** - when we have sparse statistics, steal probability mass to generalize better
- **Laplace Add-One smoothing** - pretend we saw each word one more time than we did

- add one to all counts
- traditional MLE estimate: $P_{MLE}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$
- Add-1 estimate: $P_{Add-1}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$
- **maximum likelihood estimates (MLE)** - maximizes the likelihood of the training set T given the model M based on some parameter of a model M from a training set T
 - example: suppose word "bagel" occurs 400 times in a corpus of a million words
 - probability that a random word from some other text will be "bagel"?
 - MLE estimate = $400/1,000,000 = 0.0004$
 - may be a bad estimate for some other corpus
 - but it is the *estimate* that makes it *most likely* that "bagel" will occur 400 times in a million word corpus
- Add-1 is a *blunt instrument*, so it is not used for N-grams
- used to smooth other NLP models for text classification and in domains where the number of zeros is not huge

5. 11 - Naive Bayes

5.1. Text Classification

- **input** - a document d and a fixed set of classes $C = \{c_1, c_2, \dots, c_j\}$
- **output** - a predicted class $c \in C$
- *classification methods* - **hand-coded rules**
 - rules based on combinations of words or other features
 - accuracy can be high, if rules carefully refined by expert
 - but building and maintaining these rules is *expensive*
- **classification methods* - **supervised machine learning**
 - **input** - a document d , a fixed set of classes $C = \{c_1, c_2, \dots, c_j\}$, and a training set of m hand-labeled documents $(d_1, c_1), \dots, (d_m, c_m)$
 - **output** - a learned classifier $\gamma : d \Rightarrow c$
 - any kind of classifier
 - naive bayes
 - logistic regression
 - neural networks
 - k-nearest neighbors
 - etc

5.2. Naive Bayes Classifier

- **intuition for Naive Bayes** - simple classification method based on Bayes rule
 - relies on very simple representation of document, *bag of words*

- **Bayes' Rule** - for a document d and a class c , $P(c|d) = \frac{P(d|c)P(c)}{P(d)}$
- **Naive Bayes Classifier**
 - MAP = most likely class
 - $c_{MAP} = \arg \max (c \in C) P(c|d)$
 - using Bayes' Rule, can simplify to
 - $c_{MAP} = \arg \max (c \in C) P(d|c)P(c)$
 - where $P(d|c)$ is the *likelihood* and $P(c)$ is the *prior*
 - d can be represented as features x_1, \dots, x_n
 - $O(|X|^n \times |C|)$ parameters
 - could only be estimated if a very, very large number of training examples was available
 - we can just count the relative frequencies in a corpus

5.3. Multinomial Naive Bayes

- **independence assumptions** - $P(x_1, x_2, \dots, x_n|c)$
- **bag of words assumption** - assume position does not matter
- **conditional independence** - assume the feature probabilities $\{P(x_i|c_j)\}$ are independent given the class c
 - $P(x_1, x_2, \dots, x_n|c) = P(x_1|c) \times P(x_2|c) \times P(x_3|c) \times \dots \times P(x_n|c)$
- $c_{NB} = \arg \max (c \in C) \prod (x \in X) P(x|c)$
- **problem** with multiplying lots of probabilities
 - can result in floating point underflow
 - *solution* - use logs since multiplications become additions
 - taking log does not change the ranking of classes
 - linear model - max of a sum of weights, so it is a linear function of the inputs
- Naive Bayes is a **linear classifier**

5.4. Learning

- *first attempt* - use MLE with the frequencies in the data
 - $P(w_i|c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{(w \in V)} \text{count}(w, c_j)}$
 - fraction of times word w_i appears among all words in documents of topic c_j
 - create *mega-document* for topic j by concatenating all docs in this topic
 - use frequency of w in mega-document
 - **problem** - what if we have seen no training documents with a word classified in a class?
 - we will get 0
 - cannot condition away zero probabilities no matter what
 - **solution** - Laplace Add-1 smoothing, same idea as with Markov assumptions, add one to all counts
- **unknown words** - what do we do with them that appear in the test data but not in the training

data or vocabulary?

- **ignore them** - remove from test document, pretend they were not there
 - do not include any probability for them at all
- building an unknown word model does not help, knowing which class has more unknown words is *not generally helpful*
- **stop words** - very frequent words like "the" and "a"
 - some systems ignore them
 - but usually does not help
 - sort vocabulary by word frequency in a training set
 - call the top 10 or 50 words the *stopword list*
 - remove all stop words from both training and test sets, as if they were never there to begin with

5.5. Relationship to Language Modeling

- **generative model of NB** - graph of the words that are classified to a specific class
- NB classifiers can use any sort of *feature*
 - i.e. URL, email address, dictionaries, network features
 - we use **only** word features
 - we use **all** of the words in the text, not a subset
 - then NB has an *important similarity* to language modeling
- **each class = a unigram language model**
 - assigning each word: $P(\text{word}|c)$
 - assigning each sentence: $P(s|c) = \prod P(\text{word}|c)$
 - example: each word and their probability it is positive class
 - I = 0.1, love = 0.1, this = 0.05, fun = 0.01, film = 0.1
 - $P(\text{sentence} | \text{positive}) = 0.1 \times 0.1 \times 0.05 \times 0.01 \times 0.1 = 0.0000005$
 - example: using positive class from previous and given new negative class, which one assigns the higher probability to sentence?
 - negative: I = 0.2, love = 0.001, this = 0.01, fun = 0.005, film = 0.1
 - $P(\text{sentence} | \text{negative}) = 10^{-9}$
 - $P(\text{sentence} | \text{positive}) > P(\text{sentence} | \text{negative})$

5.6. Naive Bayes Evaluation

- **2 by 2 confusion matrix**

		gold standard labels		
		gold positive	gold negative	
system output labels	system positive	true positive	false positive	precision = $\frac{tp}{tp+fp}$
	system negative	false negative	true negative	
		recall = $\frac{tp}{tp+fn}$		accuracy = $\frac{tp+tn}{tp+fp+tn+fn}$

- **accuracy** - do not use as the evaluation metric
 - useless, does not return what we are looking for
 - can get amazing accuracy for very dumb labeling that is not very representative of the data as a whole
 - use *precision* and *recall* instead
- **precision** - percent of items the system detected (i.e. items the system labeled as positive) that are positive (according to human gold labels)
 - $precision = \frac{true\ positives}{true\ positives + false\ positives}$
- **recall** - percent of items actually present in the input that were correctly identified by the system
 - $recall = \frac{true\ positives}{true\ positives + false\ negatives}$
- precision and recall, not accuracy, emphasize **true positives** - finding the things that we are supposed to be looking for
- **combined measure F** - a single number that combines both *precision* and *recall*

$$F_{\beta} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

- almost always use balanced F_1 ($\beta = 1$)
 - $F_1 = \frac{2PR}{P+R}$

5.7. Cross-Validation

- **devsets** - development test sets
- train on training set, tune on devset, report on test set
 - avoids *overfitting* - tuning to the test set
 - more conservative estimate of performance
 - **paradox** - want as much data as possible for training and as much for dev, so how do you split it?
- **cross-validation** - multiple splits

- pool results over splits, compute pooled dev performance

5.8. Harms in Classification

- can have biases in classifiers that perpetuate negative stereotypes against a certain group of people, etc
 - or censorship of discussion about a group of people
- **causes of harms**
 - problems in the training data, ML systems known to amplify biases in their training data
 - problems in the human labels
 - problems in the resources used (like lexicons)
 - problems in the model architecture (like what the model is trained to optimize)
- **mitigation** of these harms is an *open research area*
- **model cards** - for each algorithm you release, document:
 - training algorithms and parameters
 - training data sources, motivation, and preprocessing
 - evaluation data sources, motivation, and preprocessing
 - intended use and users
 - model performance across different demographic or other groups and environmental situations