

Introduction to NLP

211.

Language Models

Probabilistic Language Models

- Assign a probability to a sentence
 - $P(S) = P(w_1, w_2, w_3, \dots, w_n)$
- The sum of the probabilities of all possible sentences must be 1.
- Predicting the next word
 - Let's meet in Times ...
 - General Electric has lost some market ...
- Formula
 - $P(w_n | w_1, w_2, \dots, w_{n-1})$

Predicting the Next Word

- What word follows “your”?

http://norvig.com/ngrams/count_2w.txt

your abilities 160848
your ability 1116122
your ablum 112926
your academic 274761
your acceptance 783544
your access 492555
your accommodation 320408
your account 8149940
your accounting 128409
your accounts 257118
your action 121057
your actions 492448
your activation 459379

your active 140797
your activities 226183
your activity 156213
your actual 302488
your ad 1450485
your address 1611337
your admin 117943
your ads 264771
your advantage 242238
your adventure 109658
your advert 101178
your advertisement 172783

Uses of Language Models

- Speech recognition
 - $P(\text{"recognize speech"}) > P(\text{"wreck a nice beach"})$
- Text generation
 - $P(\text{"three houses"}) > P(\text{"three house"})$
- Spelling correction
 - $P(\text{"my cat eats fish"}) > P(\text{"my xat eats fish"})$
- Machine translation
 - $P(\text{"the blue house"}) > P(\text{"the house blue"})$
- Other uses
 - OCR
 - Summarization
 - Document classification
- Usually coupled with a translation model (later)

Probability of a Sentence

- How to compute the probability of a sentence?
 - What if the sentence is novel?
- What we need to estimate:
 - $P(S)=P(w_1, w_2, w_3 \dots w_n)$
- Using the chain rule:
 - $P(S)= P(w_1) P(w_2 | w_1) P(w_3 | w_1, w_2) \dots P(w_n | w_1, w_2 \dots w_{n-1})$
- Example:
 - $P(\text{"I would like the pepperoni and spinach pizza"})=?$

N-gram Model

N-gram Models

- Predict the probability of a word based on the words before:
 - $P(\text{square} | \text{Let's meet in Times})$
- Markov assumption
 - Only look at limited history
- N-gram models
 - Unigram – no context: $P(\text{square})$
 - Bigram: $P(\text{square} | \text{Times})$
 - Trigram: $P(\text{square} | \text{in Times})$
- It is possible to go to 3,4,5-grams
 - Longer n-grams suffer from sparseness
- Used for predicting the next word and also for random text generation

Approximating Shakespeare

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.

4
gram

–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;

–It cannot be but so.

Approximating the Wall Street Journal

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

N-Grams

- Shakespeare unigrams
 - 29,524 types, approx. 900K tokens
- Bigrams
 - 346,097 types, approx. 900K tokens
 - How many bigrams are never seen in the data?
- Notice!
 - very sparse data!

Google 1-T Corpus

- 1 trillion word tokens
- Number of tokens
 - 1,024,908,267,229
- Number of sentences
 - 95,119,665,584
- Number of unigrams
 - 13,588,391
- Number of bigrams
 - 314,843,401
- Number of trigrams
 - 977,069,902
- Number of fourgrams
 - 1,313,818,354
- Number of fivegrams
 - 1,176,470,663

Parameter Estimation

- Can we compute the conditional probabilities directly?
 - No, because the data is sparse
- Markov assumption

- $P(\text{"musical"} \mid \text{"I would like two tickets for the"}) = P(\text{"musical"} \mid \text{the})$

or

- $P(\text{"musical"} \mid \text{"I would like two tickets for the"}) = P(\text{"musical"} \mid \text{for the})$

Maximum Likelihood Estimates

- Use training data
 - Count how many times a given context appears in it.
- Unigram example:
 - The word “pizza” appears 700 times in a corpus of 10,000,000 words.
 - Therefore the MLE for its probability is $P'(\text{“pizza”}) = 700/10,000,000 = 0.00007$
- Bigram example:
 - The word “with” appears 1,000 times in the corpus.
 - The phrase “with spinach” appears 6 times
 - Therefore the MLE for $P'(\text{spinach} | \text{with}) = 6/1,000 = 0.006$
- These estimates may not be good for corpora from other genres

Probability of a Sentence

$P("<S> \text{ I will see you on Monday } </S>") =$

$P(I | <S>)$

$\times P(\text{will} | I)$

$\times P(\text{see} | \text{will})$


$\times P(\text{you} | \text{see})$

$\times P(\text{on} | \text{you})$

$\times P(\text{Monday} | \text{on})$

$\times P(</S> | \text{Monday})$

Probability of a Sentence

$$P(X) = \prod_{i=1}^I P(x_i \mid x_1, \dots, x_{i-1})$$


The diagram illustrates the components of the probability formula. A red horizontal line segment is positioned below the term x_i in the denominator, with a red arrow pointing from the text "Next Word" below it to this segment. A blue horizontal line segment is positioned below the sequence x_1, \dots, x_{i-1} in the denominator, with a blue arrow pointing from the text "Context" below it to this segment.

	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.

	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.

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

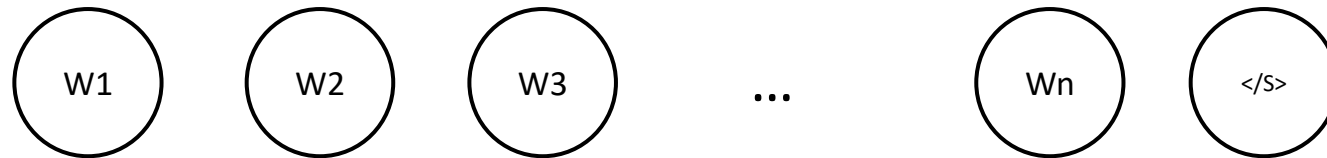
Example from Jane Austen

- $P(\text{"Elizabeth looked at Darcy"})$
- Use maximum likelihood estimates for the n-gram probabilities
 - unigram: $P(w_i) = c(w_i)/V$
 - bigram: $P(w_i | w_{i-1}) = c(w_{i-1}, w_i) / c(w_{i-1})$
- Values
 - $P(\text{"Elizabeth"}) = 474/617091 = .000768120$
 - $P(\text{"looked"} | \text{"Elizabeth"}) = 5/474 = .010548523$
 - $P(\text{"at"} | \text{"looked"}) = 74/337 = .219584569$
 - $P(\text{"Darcy"} | \text{"at"}) = 3/4055 = .000739827$
- Bigram probability
 - $P(\text{"Elizabeth looked at Darcy"}) = .000000001316 = 1.3 \times 10^{-9}$
- Unigram probability
 - $P(\text{"Elizabeth looked at Darcy"}) = 474/617091 * 337/617091 * 4055/617091 * 304/617091 = .000000000001357 = 1.3 \times 10^{-12}$
- $P(\text{"looked Darcy Elizabeth at"}) = ?$

Generative Models

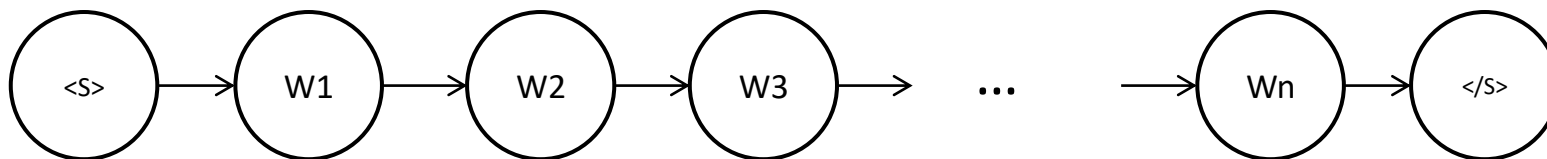
- Unigram:

- generate a word, then generate the next one, until you generate </S> .



- Bigram:

- generate <S> , generate a word, then generate the next one based on the previous one, etc., until you generate </S> .



Engineering Trick

- The MLE values are often on the order of 10^{-6} or less
 - Multiplying 20 such values gives a number on the order of 10^{-120}
 - This leads to underflow
- Use logarithms instead
 - 10^{-6} (in base 10) becomes -6
 - Use sums instead of products

Language Modeling Tools

- http://www.speech.cs.cmu.edu/SLM_info.html
- <http://www.speech.sri.com/projects/srilm/>
- <https://kheafield.com/code/kenlm/>
- <http://htk.eng.cam.ac.uk/>

Introduction to NLP

212.

Smoothing and Interpolation

Smoothing

- If the vocabulary size is $|V|=1M$
 - Too many parameters to estimate even a unigram model
 - MLE assigns values of 0 to unseen (yet not impossible) data
 - Let alone bigram or trigram models
- Smoothing (regularization)
 - Reassigning some probability mass to unseen data

Smoothing

- How to model novel words?
 - Or novel bigrams?
 - Distributing some of the probability mass to allow for novel events
- Add-one (Laplace) smoothing:
 - Bigrams: $P(w_i | w_{i-1}) = (c(w_{i-1}, w_i) + 1) / (c(w_{i-1}) + V)$
 - This method reassigns too much probability mass to unseen events
- Possible to do add- k instead of add-one
 - Both of these don't work well in practice

	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.

	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.

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

	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.

Thus, each of the unigram counts given in the previous section will need to be augmented by $V = 1446$. The result is the smoothed bigram probabilities in Fig. 3.6.

	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.

Dealing with Sparse Data

- Two main techniques used
 - Backoff
 - Interpolation

Backoff

- Going back to the lower-order n-gram model if the higher-order model is sparse (e.g., frequency ≤ 1)
- Learning the parameters
 - From a development data set

Interpolation

- If $P'(w_i | w_{i-1}, w_{i-2})$ is sparse:
 - Use $\lambda_1 P'(w_i | w_{i-1}, w_{i-2}) + \lambda_2 P'(w_i | w_{i-1}) + \lambda_3 P'(w_i)$
 - Ensure that $\lambda_1 + \lambda_2 + \lambda_3 = 1, \quad \lambda_1, \lambda_2, \lambda_3 \leq 1, \quad \lambda_1, \lambda_2, \lambda_3 \geq 0$
 - Better than backoff
 - Estimate the hyper-parameters $\lambda_1, \lambda_2, \lambda_3$ from held-out data (or using EM), e.g., using 5-fold cross-validation
- See [Chen and Goodman 1998] for more details
- Software:
 - http://www.speech.cs.cmu.edu/SLM/toolkit_documentation.html

Introduction to NLP

123.

Probabilities

Probabilities in NLP

- Very important for language processing
 - “Let’s meet in the conference ...”
- Example in speech recognition:
 - “recognize speech” vs “wreck a nice beach”
- Example in machine translation:
 - “l’avocat general”: “the attorney general” vs. “the general avocado”
- Example in information retrieval:
 - If a document includes three occurrences of “stir” and one of “rice”, what is the probability that it is a recipe
- Probabilities make it possible to combine evidence from multiple sources in a systematic way

Probabilities

- Probability theory
 - predicting how likely it is that something will happen
- Experiment (trial)
 - e.g., throwing a coin
- Possible outcomes
 - heads or tails
- Sample spaces
 - discrete (number of occurrences of “rice”) or continuous (e.g., temperature)
- Events
 - Ω is the certain event
 - \emptyset is the impossible event
 - event space - all possible events

Sample Space

- Random experiment: an experiment with uncertain outcome
 - e.g., flipping a coin, picking a word from text
- Sample space: all possible outcomes, e.g.,
 - Tossing 2 fair coins, $\Omega = \{HH, HT, TH, TT\}$

Events

- Event: a subspace of the sample space
 - $E \subseteq \Omega$, E happens iff outcome is in E, e.g.,
 - $E = \{HH\}$ (all heads)
 - $E = \{HH, TT\}$ (same face)
 - Impossible event (\emptyset)
 - Certain event (Ω)
- Probability of Event : $0 \leq P(E) \leq 1$, s.t.
 - $P(\Omega) = 1$ (outcome always in Ω)
 - $P(A \cup B) = P(A) + P(B)$, if $(A \cap B) = \emptyset$ (e.g., A=same face, B=different face)

Example: Tossing a Die

- Sample space: $\Omega = \{1, 2, 3, 4, 5, 6\}$
- Fair die:
 - $p(1) = p(2) = p(3) = p(4) = p(5) = p(6) = 1/6$
- Unfair die example: $p(1) = 0.3, p(2) = 0.2, \dots$
- N-dimensional die:
 - $\Omega = \{1, 2, 3, 4, \dots, N\}$
- Example in modeling text:
 - Toss a die to decide which word to write in the next position
 - $\Omega = \{\text{cat}, \text{dog}, \text{tiger}, \dots\}$

Example: Flipping a Coin

- $\Omega : \{\text{Head}, \text{Tail}\}$
- Fair coin:
 - $p(H) = 0.5, p(T) = 0.5$
- Unfair coin, e.g.:
 - $p(H) = 0.3, p(T) = 0.7$
- Flipping two fair coins:
 - Sample space: $\{HH, HT, TH, TT\}$
- Example in modeling text:
 - Flip a coin to decide whether or not to include a word in a document
 - Sample space = $\{\text{appear}, \text{absence}\}$

Probabilities

- Probabilities
 - numbers between 0 and 1
- Probability distribution
 - distributes a probability mass of 1 throughout the sample space Ω .
- Example:
 - A fair coin is tossed three times.
 - What is the probability of 3 heads?
 - What is the probability of 2 heads?

Probabilities

- Joint probability: $P(A \cap B)$, also written as $P(A, B)$

- Conditional Probability:

$$P(B|A) = P(A \cap B)/P(A)$$

$$P(A \cap B) = P(A)P(B|A) = P(B)P(A|B)$$

$$P(A|B) = P(B|A)P(A)/P(B) \quad (\text{Bayes' Rule})$$

For independent events, $P(A \cap B) = P(A)P(B)$, so $P(A|B) = P(A)$

- Total probability:

If A_1, \dots, A_n form a partition of S , then

$$P(B) = P(B \cap S) = P(B, A_1) + \dots + P(B, A_n) \quad (\text{why?})$$

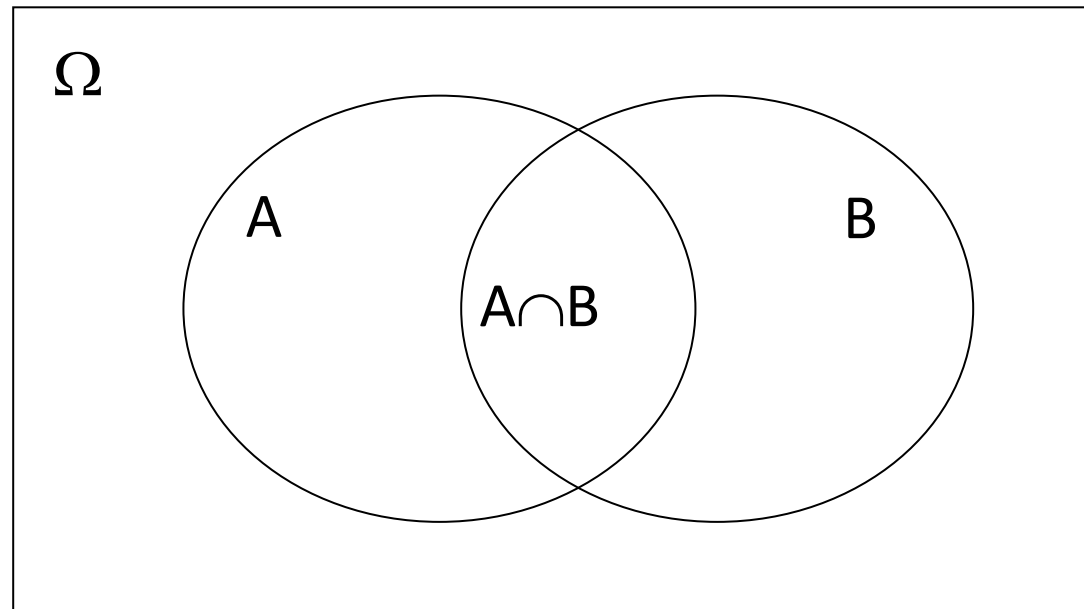
$$\text{So, } P(A_i|B) = P(B|A_i)P(A_i)/P(B) = P(B|A_i)P(A_i)/[P(B|A_1)P(A_1) + \dots + P(B|A_n)P(A_n)]$$

This allows us to compute $P(A_i|B)$ based on $P(B|A_i)$

Conditional Probability

- Prior and posterior probability
- Conditional probability

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$



Properties of Probabilities

- $p(\emptyset) = 0$
- $P(\text{certain event}) = 1$
- $p(X) \leq p(Y)$, if $X \subseteq Y$
- $p(X \cup Y) = p(X) + p(Y)$, if $X \cap Y = \emptyset$

Conditional Probability

- Six-sided fair die

$$P(D \text{ even})=?$$

$$P(D \geq 4)=?$$

$$P(D \text{ even} \mid D \geq 4)=?$$

$$P(D \text{ odd} \mid D \geq 4)=?$$

- Multiple conditions

$$P(D \text{ odd} \mid D \geq 4, D \leq 5)=?$$

Answers

- Six-sided fair die

$$P(D \text{ even}) = 3/6 = 1/2$$

$$P(D \geq 4) = 3/6 = 1/2$$

$$P(D \text{ even} | D \geq 4) = 2/3$$

$$P(D \text{ odd} | D \geq 4) = 1/3$$

- Multiple conditions

$$P(D \text{ odd} | D \geq 4, D \leq 5) = 1/2$$

The Chain Rule

- $P(w_1, w_2, w_3 \dots w_n) = ?$
- Using the chain rule:
 - $P(w_1, w_2, w_3 \dots w_n) = P(w_1) P(w_2 | w_1) P(w_3 | w_1, w_2) \dots P(w_n | w_1, w_2 \dots w_{n-1})$
- This rule is used in many ways in statistical NLP, more specifically in Markov Models

Probability Review 2/2

$$1 = \sum_a P(A = a)$$

Conditional Probability

$$P(A|B) = \frac{P(AB)}{P(B)}$$

Chain Rule

$$P(AB) = P(A|B)P(B)$$

Law of Total Probability

$$P(A) = \sum_b P(A, B = b)$$

$$P(A) = \sum_b P(A|B = b)P(B = b)$$

Disjunction (Union)

$$P(A \vee B) = P(A) + P(B) - P(AB)$$

Negation (Complement)

$$P(\neg A) = 1 - P(A)$$

Introduction to NLP

125.

Bayes Theorem

Bayes Theorem

- Formula for joint probability

$$p(A,B) = p(B|A)p(A)$$

$$p(A,B) = p(A|B)p(B)$$

- Therefore

$$p(B|A) = p(A|B)p(B)/p(A)$$

- Bayes' theorem is used to calculate $P(A|B)$ given $P(B|A)$

Example

- Diagnostic test
- Test accuracy

$p(\text{positive} \mid \neg \text{disease}) = 0.05$ – false positive

$p(\text{negative} \mid \text{disease}) = 0.05$ – false negative

So: $p(\text{positive} \mid \text{disease}) = 1 - 0.05 = 0.95$

Same for $p(\text{negative} \mid \neg \text{disease})$

In general the rates of false positives and false negatives can be different

Example

- Diagnostic test with errors

$P(A B)$		A=TEST	
		Positive	Negative
B=DISEASE	Yes	0.95	0.05
	No	0.05	0.95

Example

- What is $p(\text{disease} \mid \text{positive})$?
 - $P(\text{disease} \mid \text{positive}) = P(\text{positive} \mid \text{disease}) * P(\text{disease}) / P(\text{positive})$
 - $P(\neg \text{disease} \mid \text{positive}) = P(\text{positive} \mid \neg \text{disease}) * P(\neg \text{disease}) / P(\text{positive})$
 - $P(\text{disease} \mid \text{positive}) / P(\neg \text{disease} \mid \text{positive}) = ?$
- We don't really care about $p(\text{positive})$
 - as long as it is not zero, we can divide both sides by this quantity

Example

- $P(\text{disease} | \text{positive}) / P(\neg \text{disease} | \text{positive}) = \frac{(P(\text{positive} | \text{disease}) \times P(\text{disease}))}{(P(\text{positive} | \neg \text{disease}) \times P(\neg \text{disease}))}$
- Suppose $P(\text{disease}) = 0.001$
 - so $P(\neg \text{disease}) = 0.999$
- $P(\text{disease} | \text{positive}) / P(\neg \text{disease} | \text{positive}) = (0.95 \times 0.001) / (0.05 \times 0.999) = 0.019$
- $P(\text{disease} | \text{positive}) + P(\neg \text{disease} | \text{positive}) = 1$
- $P(\text{disease} | \text{positive}) \approx 0.02$
- Notes
 - $P(\text{disease})$ is called the prior probability
 - $P(\text{disease} | \text{positive})$ is called the posterior probability
 - In this example the posterior is 20 times larger than the prior

Example: An Unfair Die

- It's more likely to get a 6 and less likely to get a 1
 - $p(6) > p(1)$
 - How likely?
- What if you toss the die 1000 times, and observe “6” 501 times, “1” 108 times?
 - $p(6) = 501/1000 = 0.501$
 - $p(1) = 108/1000 = 0.108$
 - As simple as counting, but principled – maximum likelihood estimate



What if the Die has More Faces?

- Suitable to represent documents
- Every face corresponds to a word in vocabulary
- The author tosses a die to write a word
- Apparently, an unfair die

