NLP

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
  - $\varnothing$ 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, Ω ={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: $p(1) = 0.3$, $p(2) = 0.2$, ...
- N-dimensional die:
  - $\Omega = \{1, 2, 3, 4, \ldots, N\}$
- Example in modeling text:
  - Toss a die to decide which word to write in the next position
  - Ω = {cat, dog, tiger, …}

Example: Flipping a Coin

- Ω : {Head, 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 = {appear, 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:

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

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

  $P(A|B) = P(B|A)P(A)/P(B)$ (Bayes' Rule)

  For independent events, $P(A \cap B) = P(A)P(B)$, so $P(A|B)=P(A)$
- Total probability:

  If $A_1, \ldots, A_n$ form a partition of S, then

  $P(B) = P(B \cap S) = P(B, A_1) + \ldots + P(B, A_n)$ (why?)

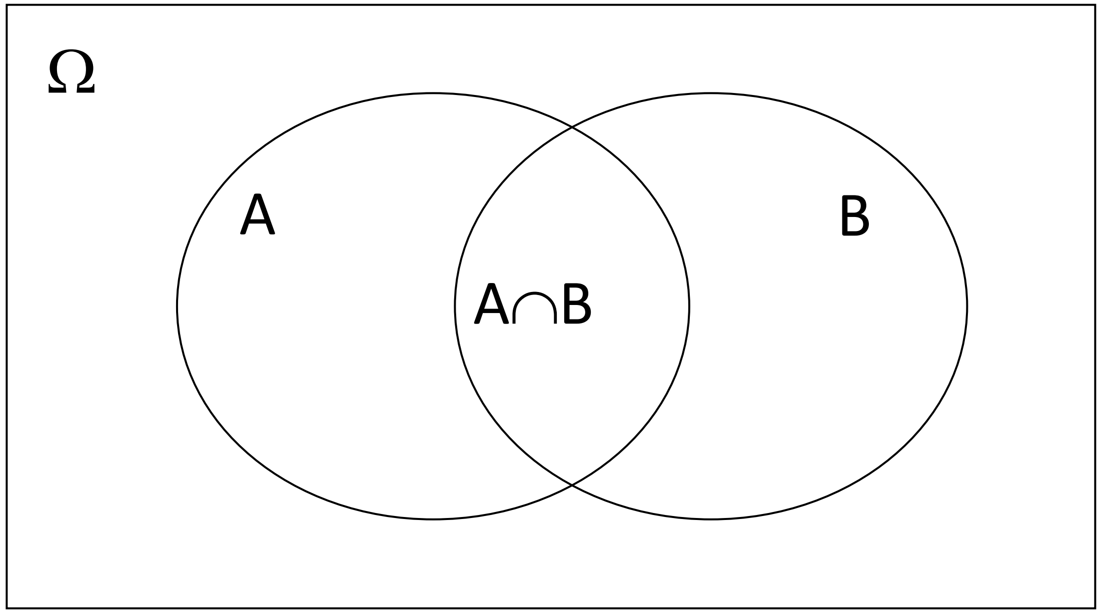
  So, $P(A_i|B) = P(B|A_i)P(A_i)/P(B) = P(B|A_i)P(A_i)/[P(B|A_1)P(A_1)+\ldots+P(B|A_n)P(A_n)$

  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

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

Conditional Probability

- Six-sided fair die

  P(D even)=?

  P(D>=4)=?

  P(D even|D>=4)=?

  P(D odd|D>=4)=?

- Multiple conditions

  P(D odd|D>=4, D<=5)=?

Answers

- Six-sided fair die

  P(D even)=3/6=1/2

  P(D>=4)=3/6=1/2

  P(D even|D>=4)=2/3

  P(D odd|D>=4)=1/3

- Multiple conditions

  P(D odd|D>=4, D<=5)=1/2

The Chain Rule

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

Probability Review

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

Conditional Probability $$= \frac{P(AB)}{P(B)}$$

Chain Rule $$= P(A|B)P(B)$$

Law of Total Probability $$= \sum_b P(A, B=b)$$
$$= \sum_b P(A|B=b)P(B=b)$$

Disjunction (Union) $$P(A \vee B) =$$

Negation (Complement) $$P(\neg A) =$$

[slide from Brendan O'Connor]

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

[slide from Brendan O’Connor]

NLP

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

Example

- $P(\text{disease}|\text{positive}) / P(\neg\text{disease}|\text{positive}) =$
  $(P(\text{positive}|\text{disease}) \times P(\text{disease}))/(P(\text{positive}|\neg\text{disease}) \times P(\neg\text{disease}))$
- 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(disease) is called the prior probability
  - P(disease|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

[slide from Qiaozhu Mei]

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

[slide from Qiaozhu Mei]

NLP

Introduction to NLP

211.

Language Models

Probabilistic Language Models

- Assign a probability to a sentence
  - $P(S) = P(w_1,w_2,w_3,\ldots,w_n)$
- 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
  - $P(w_n|w_1,w_2,\ldots,w_{n-1})$

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(“recognize speech”) > P(“wreck a nice beach”)
- Text generation
  - P(“three houses”) > P(“three house”)
- Spelling correction
  - P(“my cat eats fish”) > P(“my xat eats fish”)
- Machine translation
  - P(“the blue house”) > P(“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:
  - $P(S)=P(w_1,w_2,w_3\ldots w_n)$
- Using the chain rule:
  - $P(S)= P(w_1)\ P(w_2|w_1)\ P(w_3|w_1,w_2)\ldots\ P(w_n|w_1,w_2\ldots w_{n-1})$
- Example:
  - P(“I would like the pepperoni and spinach pizza”)=?

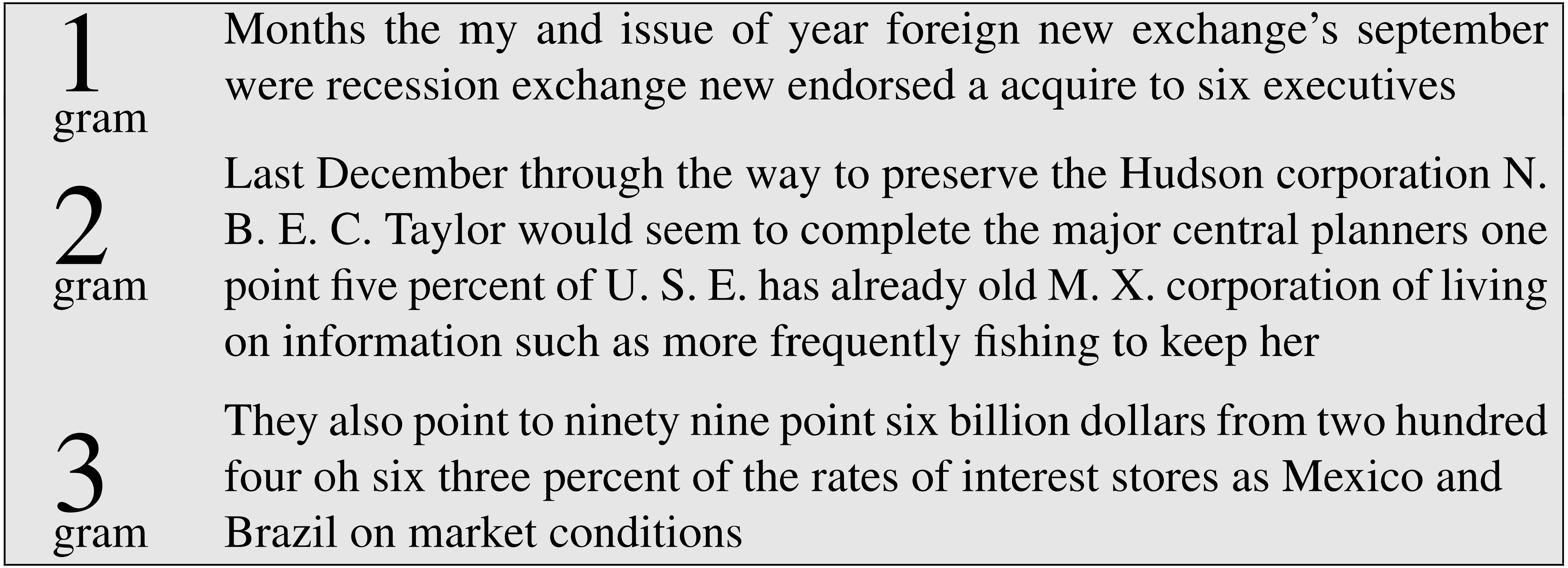
N-gram Models

- Predict the probability of a word based on the words before:
  - P(square|Let’s meet in Times)
- Markov assumption
  - Only look at limited history
- N-gram models
  - Unigram – no context: P(square)
  - Bigram: P(square|Times)
  - Trigram: P(square|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

https://catalog.ldc.upenn.edu/ldc2006t13

Parameter Estimation

- Can we compute the conditional probabilities directly?
  - No, because the data is sparse
- Markov assumption
    - P(“musical” | “I would like two tickets for the”) = P(“musical | the”)

  or

    - P(“musical” | “I would like two tickets for the”) = P(“musical | 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’(“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
  - Therefor the MLE for P’(spinach|with) = 6/1,000 = 0.006
- These estimates may not be good for corpora from other genres

Probability of a Sentence

P("<S> I will see you on Monday</S>") =

P(I|<S>)
x P(will|I)
x P(see|will)
x P(you|see)
x P(on|you)
x P(Monday|on)
x P(</S>|Monday)

Probability of a Sentence

$$P(X) = \prod_{i=1}^{I} P(\underbrace{x_i}_{\text{Next Word}} \mid \underbrace{x_1, \ldots, x_{i-1}}_{\text{Context}})$$

	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.

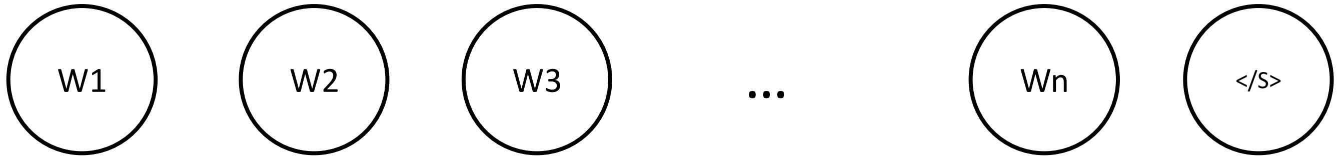
$$
\begin{aligned}
P(\texttt{<s> i want english food </s>}) \\
&= P(\texttt{i}|\texttt{<s>})P(\texttt{want}|\texttt{i})P(\texttt{english}|\texttt{want}) \\
&\qquad P(\texttt{food}|\texttt{english})P(\texttt{</s>}|\texttt{food}) \\
&= .25 \times .33 \times .0011 \times 0.5 \times 0.68 \\
&= .000031
\end{aligned}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

Example from Jane Austen

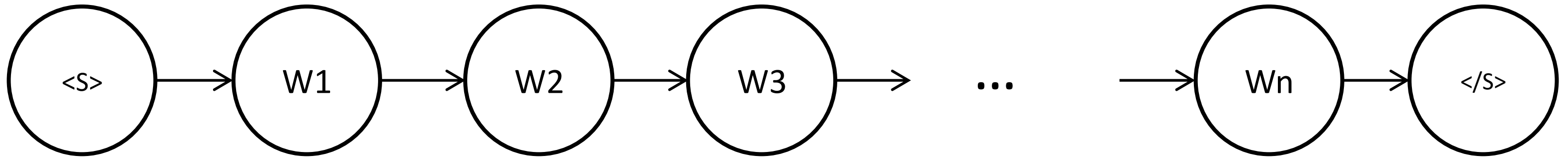
- P("Elizabeth looked at Darcy")
- Use maximum likelihood estimates for the n-gram probabilities
  - unigram: $P(w_i)=c(w_i)/V$
  - bigram: $P(w_i|w_{i-1}) = c(w_{i-1},w_i)/c(w_{i-1})$
- Values
  - P("Elizabeth") = 474/617091 = .000768120
  - P("looked|Elizabeth") = 5/474 = .010548523
  - P("at|looked")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

NLP

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.

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

	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 $\lambda_1+\lambda_2+\lambda_3=1$, $\lambda_1,\lambda_2,\lambda_3 \leq 1$, $\lambda_1,\lambda_2,\lambda_3 \geq 0$
  - 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

NLP

Introduction to NLP

213.

Evaluation of Language Models

Evaluation of LM

- Extrinsic
  - Use in an application
- Intrinsic
  - Cheaper
  - Based on information theory
- Correlate the two for validation purposes

Information Theory

- It is concerned with data transmission, data compression, and measuring the amount of information.
- Applies to statistical physics, economics, linguistics.

Information and Uncertainty

- The decrease in uncertainty is called information
- Example
  - we know that a certain event will happen next week
  - then we learn that it is more likely to happen on a workday
  - the new information reduces the uncertainty
  - the more new information we get, the smaller the remaining uncertainty

Entropy

- Entropy tells us how informative a random variable is.

$$H(p) = H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

Examples

- One symbol (a)
  - uncertainty is 0
- Two symbols (a,b)
  - uncertainty is 1
  - we can reduce it to 0 by using one bit of information (a=0,b=1)
- Four symbols (a,b,c,d)
  - we need two bits of information (e.g., a=00,b=01,c=10,d=11)
- In general we need
  - $\log_2 k$ bits, where *k* is the number of symbols
  - note: this only holds if all symbols are equiprobable

Amount of Surprise

- Amount of surprise (given a general prob. distribution)

  $-\log_2 p(x)$ - for a specific outcome x

- If the distribution is uniform:

  $p(x) = 1/k$

  $k = 1/p(x)$

  $\log_2 k = \log_2 (1/p(x)) = -\log_2 p(x)$

- Average surprise

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x) = E(\log_2 \frac{1}{p(X)})$$

- Information need

  H(X) = 0 means that we have all the information that we need
  H(X) = 1 means that we need one bit of information, etc.

Entropy Example

- Sample 8-character language: A E I O U F G H

$$H(X) = -\sum_{i=1}^{8} p(i) \log_2 p(i) = -\sum_{i=1}^{8} \tfrac{1}{8} \log_2 \tfrac{1}{8} = \log_2 8 = 3$$

- Three bits per character if the characters are equiprobable

A	E	I	O	U	F	G	H
000	001	010	011	100	101	110	111

Simplified Polynesian

- Six characters: P T K A I U, not equiprobable

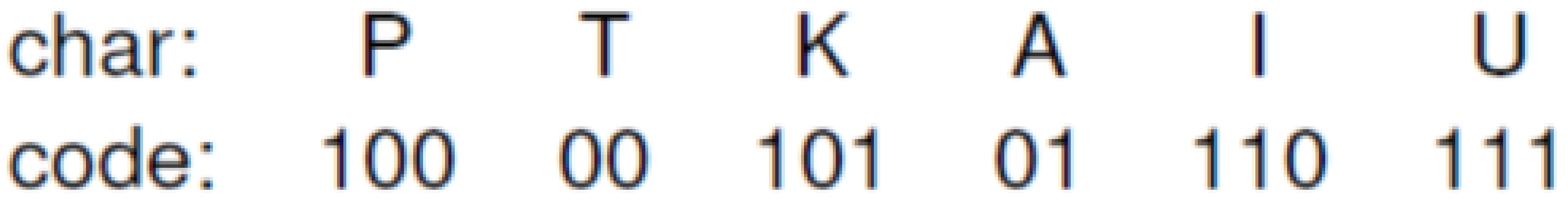
char:	P	T	K	A	I	U
prob:	1/8	1/4	1/8	1/4	1/8	1/8

$$\begin{aligned} H(X) &= -\sum_{i \in L} p(i) \log p(i) \\ &= -[4 \times \frac{1}{8} \log \frac{1}{8} + 2 \times \frac{1}{4} \log \frac{1}{4}] \\ &= 2.5 \end{aligned}$$

- This number (2.5) can lead one to believe that 3 bits per character are needed
  - e.g. 000, 001, 010, 100, 101, 111

Simplified Polynesian

- More efficient encoding



char:	P	T	K	A	I	U
code:	100	00	101	01	110	111

- Longer codes for less frequent characters
- This can lower the average number of bits per character to the theoretical estimate of 2.5
- Under what assumption, though?

Joint Entropy

- Amount of information to specify both *x* and *y*.

$$H(X,Y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_2 p(x,y)$$

- Measures the amount of surprise of seeing a specific tag bigram.

Conditional Entropy

- If we know *x*, how much additional information is needed to know *y*.

$$
\begin{aligned}
H(Y|X) &= \sum_{x \in X} p(x) H(Y|X = x) \\
&= \sum_{x \in X} p(x) [- \sum_{y \in Y} p(y|x) \log_2 p(y|x)] \\
&= - \sum_{x \in X} \sum_{y \in Y} p(x) p(y|x) \log_2 p(y|x) \\
&= - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(y|x)
\end{aligned}
$$

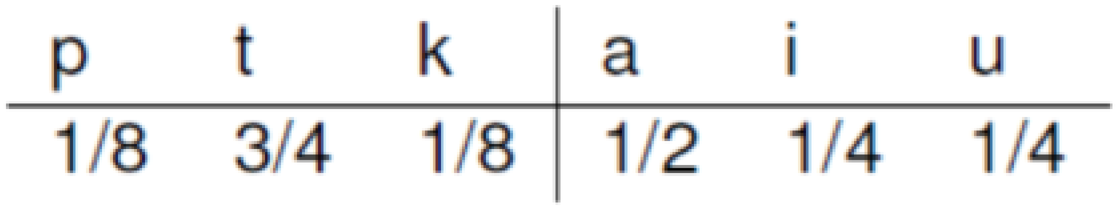
Chain Rule for Entropy

- Chain rule for entropy

$$
\begin{aligned}
H(X,Y) &= H(X) + H(Y|X) \\
H(X_1,\ldots,X_n) &= H(X_1) + H(X_2|X_1) \\
&\quad + H(X_3|X_1,X_2) + \ldots \\
&\quad + H(X_n|X_1,\ldots,X_{n-1})
\end{aligned}
$$

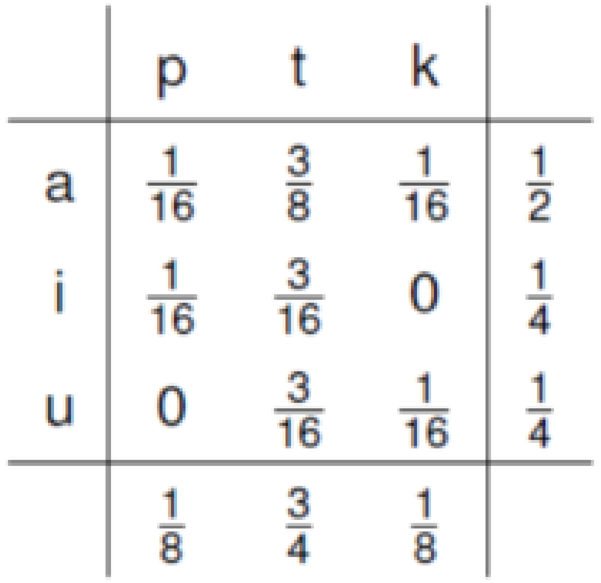
Probabilities of Syllables

- P(C,·)   and P(·,V)   - marginal probabilities



p	t	k	a	i	u
1/8	3/4	1/8	1/2	1/4	1/4

- P(C,V)



	p	t	k	
a	$\frac{1}{16}$	$\frac{3}{8}$	$\frac{1}{16}$	$\frac{1}{2}$
i	$\frac{1}{16}$	$\frac{3}{16}$	0	$\frac{1}{4}$
u	0	$\frac{3}{16}$	$\frac{1}{16}$	$\frac{1}{4}$
	$\frac{1}{8}$	$\frac{3}{4}$	$\frac{1}{8}$	

Surprise in Syllables

$$H(C,V) = H(C) + H(V|C) \approx 1.061 + 1.375 \approx 2.44$$

a. $H(C) = -\sum_{c \in C} p(c) \log_2 p(c) \approx 1.061$

b. $H(V|C) = -\sum_{c \in C} \sum_{v \in V} p(c,v) \log p(v|c) = 1.375$

- Example

$$p(V = a|C = p) = \tfrac{1}{2} \text{ because } \tfrac{1}{16} \text{ is half of } \tfrac{1}{8}$$

Polynesian Syllables (cont'd)

$$\begin{aligned}
H(C) &= -\sum_{i\in L} p(i)\, log\, p(i) \\
&= -[2 \times \frac{1}{8}\, log\, \frac{1}{8} + \frac{3}{4}\, log\, \frac{3}{4}] \\
&= 2 \times \frac{1}{8}\, log\, 8 + \frac{3}{4}(log\, 4 - log\, 3) \\
&= 2 \times \frac{1}{8} \times 3 + \frac{3}{4}(2 - log\, 3) \\
&= \frac{3}{4} + \frac{6}{4} - \frac{3}{4}\, log\, 3 \\
&= \frac{9}{4} - \frac{3}{4}\, log\, 3 \approx 1.061
\end{aligned}$$

Polynesian Syllables (cont'd)

$$\begin{aligned}
H(V|C) &= -\sum_{x\in C}\sum_{y\in V} p(x,y)\, log\, p(y|x) \\
&= -[1/16\, log\, 1/2 + 3/8\, log\, 1/2 + 1/16\, log\, 1/2 \\
&\quad +1/16\, log\, 1/2 + 3/16\, log\, 1/4 + 0\, log\, 0 \\
&\quad +0\, log\, 0 + 3/16\, log\, 1/4 + 1/16\, log\, 1/2] \\
&= 1/16\, log 2 + 3/8\, log\, 2 + 1/16\, log\, 2 \\
&\quad +1/16\, log\, 2 + 3/16\, log\, 4 \\
&\quad +3/16\, log\, 4 + 1/16\, log\, 2] \\
&= 11/8 \\
&= 1.375
\end{aligned}$$

Pointwise Mutual Information

- Measured between two points (not two distributions)

$$I(x;y) = \log \frac{p(x,y)}{p(x)p(y)}$$

- If *p(x,y) = p(x)p(y),* then *I(x;y)* = log 1 = 0 (independence)

Mutual Information

- Same, but for two distributions (not points)
- How much information does one of the distributions contain about the other one.

$$I(X;Y) = \sum_{x,y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}$$

Kullback-Leibler (KL) Divergence

- Measures how far two distributions are from one another

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

- It measures the number of bits needed to encode p by using q.
- Always non-negative
- D(p||q) = 0, iff p=q
- D(p||q) = ∞, iff $\exists x \in X$ such that p(x)>0 and q(x)=0
- Not symmetric

Divergence as Mutual Information

$$I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

$$I(X;Y) = D(p(x,y) || p(x)p(y))$$

Perplexity

- Does the model fit the data?
  - A good model will give a high probability to a real sentence
- Perplexity
  - Average branching factor in predicting the next word
  - Lower is better (lower perplexity -> higher probability)
  - N = number of words

$$Per = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Perplexity

- Example:
  - A sentence consisting of N equiprobable words: $p(w_i) = 1/k$

$$Per = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

  - $Per = ((k^{-1})^N)^{(-1/N)} = k$
- Perplexity is like a (weighted) branching factor
- Logarithmic version
  - the exponent is = #bits to encode each word

$$Per = 2^{-(1/N)\sum \log_2 P(w_i)}$$

The Shannon Game

- Consider the Shannon game:
  - Connecticut governor Ned Lamont said ...
- What is the perplexity of guessing a digit if all digits are equally likely? Do the math.
  - 10
- How about a letter?
  - 26
- How about guessing A ("operator") with a probability of 1/4, B ("sales") with a probability of 1/4 and 10,000 other cases with a probability of 1/2 total
  - example modified from Joshua Goodman.

Perplexity Across Distributions

- What if the actual distribution is very different from the expected one?
- Example:
  - All of the 10,000 other cases are equally likely but P(A) = P(B) = 0.
- Cross-entropy = log (perplexity), measured in bits

$$\mathrm{H}(p,q) = -\sum_x p(x)\,\log q(x).$$

Sample Values for Perplexity

- Wall Street Journal (WSJ) corpus
    - 38 M words (tokens)
    - 20 K types
- Perplexity
    - Evaluated on a separate 1.5M sample of WSJ documents
    - Unigram 962
    - Bigram 170
    - Trigram 109
    - More recent results – 47.7 (Yang et al. 2017 using AWD-LSTM

Word Error Rate

- Another evaluation metric
  - Number of insertions, deletions, and substitutions
  - Normalized by sentence length
  - Same as Levenshtein Edit Distance
- Example:
  - governor Ned Lamont met with the mayor
  - the governor met the senator
  - 3 deletions + 1 insertion + 1 substitution = WER of 5

Issues

- Out of vocabulary words (OOV)
  - Split the training set into two parts
  - Label all words in part 2 that were not in part 1 as <UNK>
- Clustering
  - e.g., dates, monetary amounts, organizations, years

Long Distance Dependencies

- This is where n-gram language models fail by definition
- Missing syntactic information
  - **The students** who participated in the game **are** tired
  - **The student** who participated in the game **is** tired
- Missing semantic information
  - **The pizza** that I had last night was **tasty**
  - **The class** that I had last night was **interesting**

Other Ideas in LM

- Skip-grapm models
  - **Ms.** Jane **Doe**, **Ms.** Mary **Doe**
- Syntactic models
  - Condition words on other words that appear in a specific syntactic relation with them
- Caching models
  - Take advantage of the fact that words appear in bursts

Limitations

- Still no general solution for long-distance dependencies
- Cannot handle linear combinations, e.g.,
  - Cats eat mice
  - People eat broccoli
  - Cats ear broccoli
  - People eat mice
- Possible solution – use phrases or n-grams as features (combinatorial explosion)

External Resources

- Google n-gram corpus
  - http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html
- Google book n-grams
  - http://ngrams.googlelabs.com/

N-gram External Links

- http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html
- http://norvig.com/mayzner.html
- http://storage.googleapis.com/books/ngrams/books/datasetsv2.html
- https://books.google.com/ngrams/
- http://www.elsewhere.org/pomo/
- http://pdos.csail.mit.edu/scigen/
- http://www.magliery.com/Band/
- http://www.magliery.com/Country/
- http://johno.jsmf.net/knowhow/ngrams/index.php
- http://www.decontextualize.com/teaching/rwet/n-grams-and-markov-chains/
- http://gregstevens.com/2012/08/16/simulating-h-p-lovecraft
- http://kingjamesprogramming.tumblr.com/

NLP

Introduction to NLP

214.

The Noisy Channel Model

The Noisy Channel Model

- Example:
  - Input: Written English (X)
  - Encoder: garbles the input (X->Y)
  - Output: Spoken English (Y)
- More examples:
  - Grammatical English to English with mistakes
  - English to bitmaps (characters)
- $P(X,Y) = P(X)P(Y|X)$

Encoding and Decoding

- Given f, guess e

$$\begin{tikzcd}
e \arrow[r] & \underset{\text{encoder}}{\boxed{E \rightarrow F}} \arrow[r, "f"] & \underset{\text{decoder}}{\boxed{F \rightarrow E}} \arrow[r] & e'
\end{tikzcd}$$

$$e' = \underset{e}{\operatorname{argmax}} P(e|f) = \underset{e}{\operatorname{argmax}} \underbrace{P(f|e)}_{\text{translation model}} \underbrace{P(e)}_{\text{language model}}$$

Example

- Translate “la maison blanche”

	P(f|e)	P(e)
cat rat piano		
house white the		
the house white		
the red house		
the small cat		
the white house		

Example

- Translate “la maison blanche”

	P(f|e)	P(e)
cat rat piano	-	-
house white the	+	-
the house white		
the red house		
the small cat		
the white house		

Example

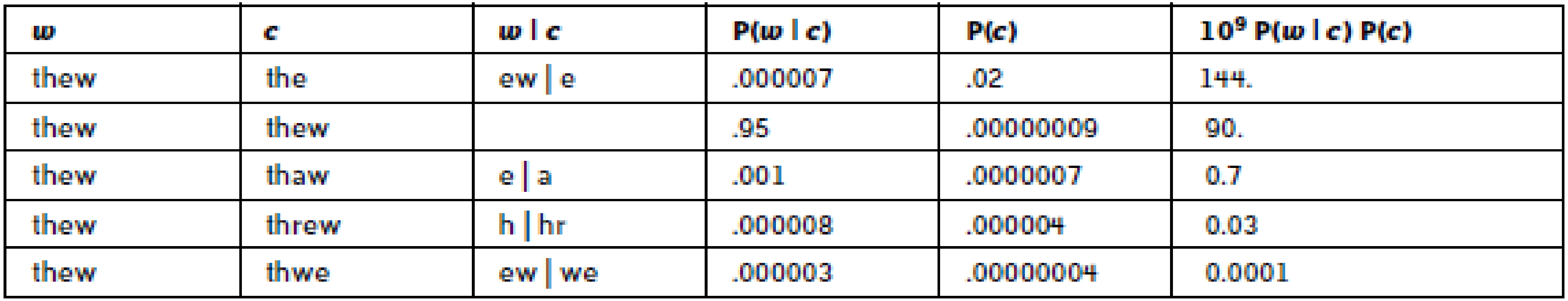
- Translate “la maison blanche”

	P(f|e)	P(e)
cat rat piano	-	-
house white the	+	-
the house white	+	-
the red house	-	+
the small cat	-	+
the white house	+	+

Uses of the Noisy Channel Model

- Handwriting recognition
- Text generation
- Text summarization
- Machine translation
- Spelling correction
  - See separate lecture on text similarity and edit distance

Spelling Correction



w	c	$w \mid c$	$P(w \mid c)$	$P(c)$	$10^9 P(w \mid c) P(c)$
thew	the	ew | e	.000007	.02	144.
thew	thew		.95	.00000009	90.
thew	thaw	e | a	.001	.0000007	0.7
thew	threw	h | hr	.000008	.000004	0.03
thew	thwe	ew | we	.000003	.00000004	0.0001

From Peter Norvig: http://**norvig**.com/ngrams/ch14.pdf

Features

- For each “e”:
  - P(e)
  - P(f|e)
  - what else?
- What about some other task, e.g., POS tagging?

NLP

Introduction to NLP

215.

Part of Speech Tagging

The POS task

- Example
  - Bahrainis vote in second <u>round</u> of parliamentary election
- Jabberwocky (by Lewis Carroll, 1872)

`Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.

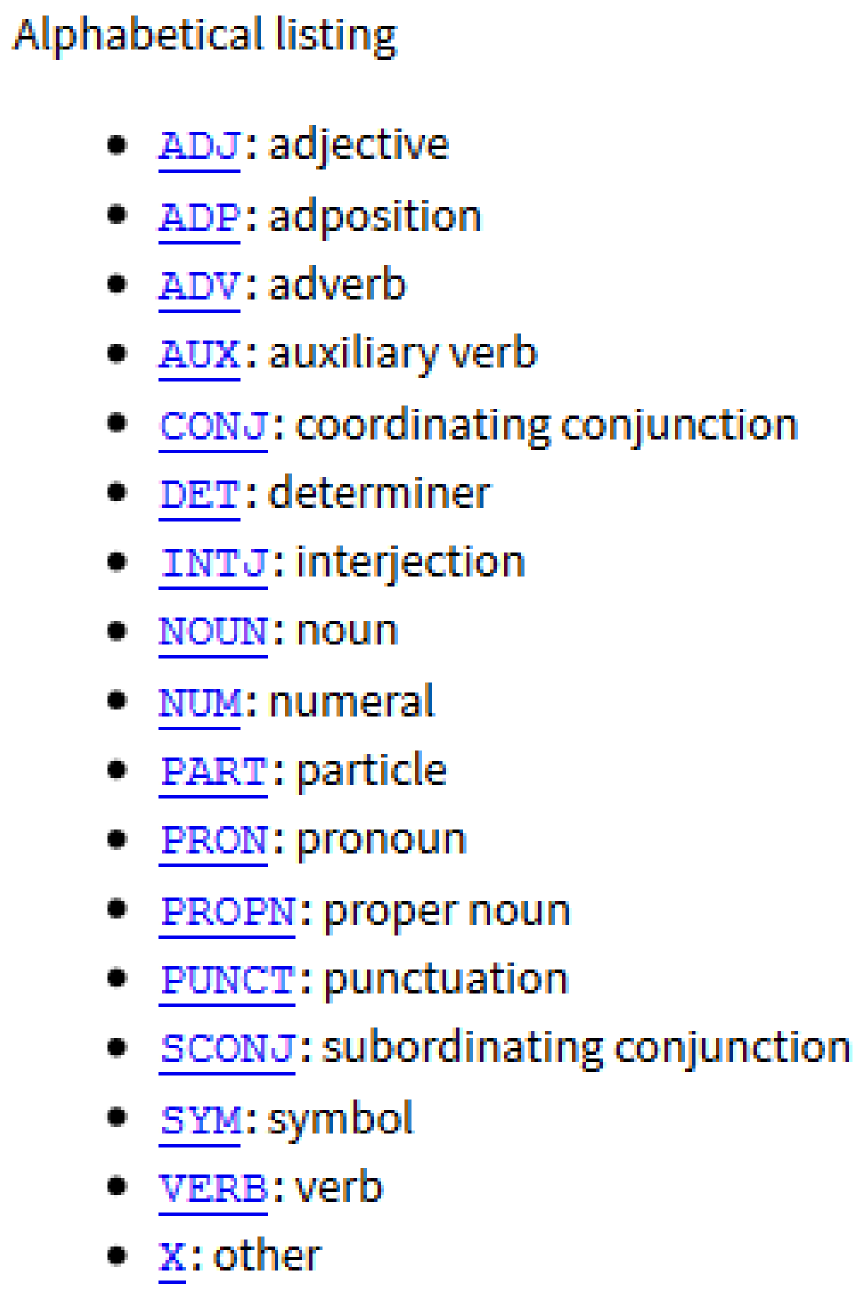
Penn Treebank tagset (1/2)

Tag	**Description**	**Example**
CC	coordinating conjunction	and
CD	cardinal number	1
DT	determiner	the
EX	existential there	*there* is
FW	foreign word	d‘oeuvre
IN	preposition/subordinating conjunction	in, of, like
JJ	adjective	green
JJR	adjective, comparative	greener
JJS	adjective, superlative	greenest
LS	list marker	1)
MD	modal	could, will
NN	noun, singular or mass	table
NNS	noun plural	tables
NNP	proper noun, singular	John
NNPS	proper noun, plural	Vikings
PDT	predeterminer	*both* the boys
POS	possessive ending	friend*'s*

Penn Treebank tagset (2/2)

Tag	**Description**	**Example**
PRP	personal pronoun	I, he, it
PRP$	possessive pronoun	my, his
RB	adverb	however, usually, naturally, here, good
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	particle	give *up*
TO	to	*to* go, *to* him
UH	interjection	uhhuhhuhh
VB	verb, base form	take
VBD	verb, past tense	took
VBG	verb, gerund/present participle	taking
VBN	verb, past participle	taken
VBP	verb, sing. present, non-3d	take
VBZ	verb, 3rd person sing. present	takes
WDT	wh-determiner	which
WP	wh-pronoun	who, what
WP$	possessive wh-pronoun	whose
WRB	wh-abverb	where, when

Universal POS

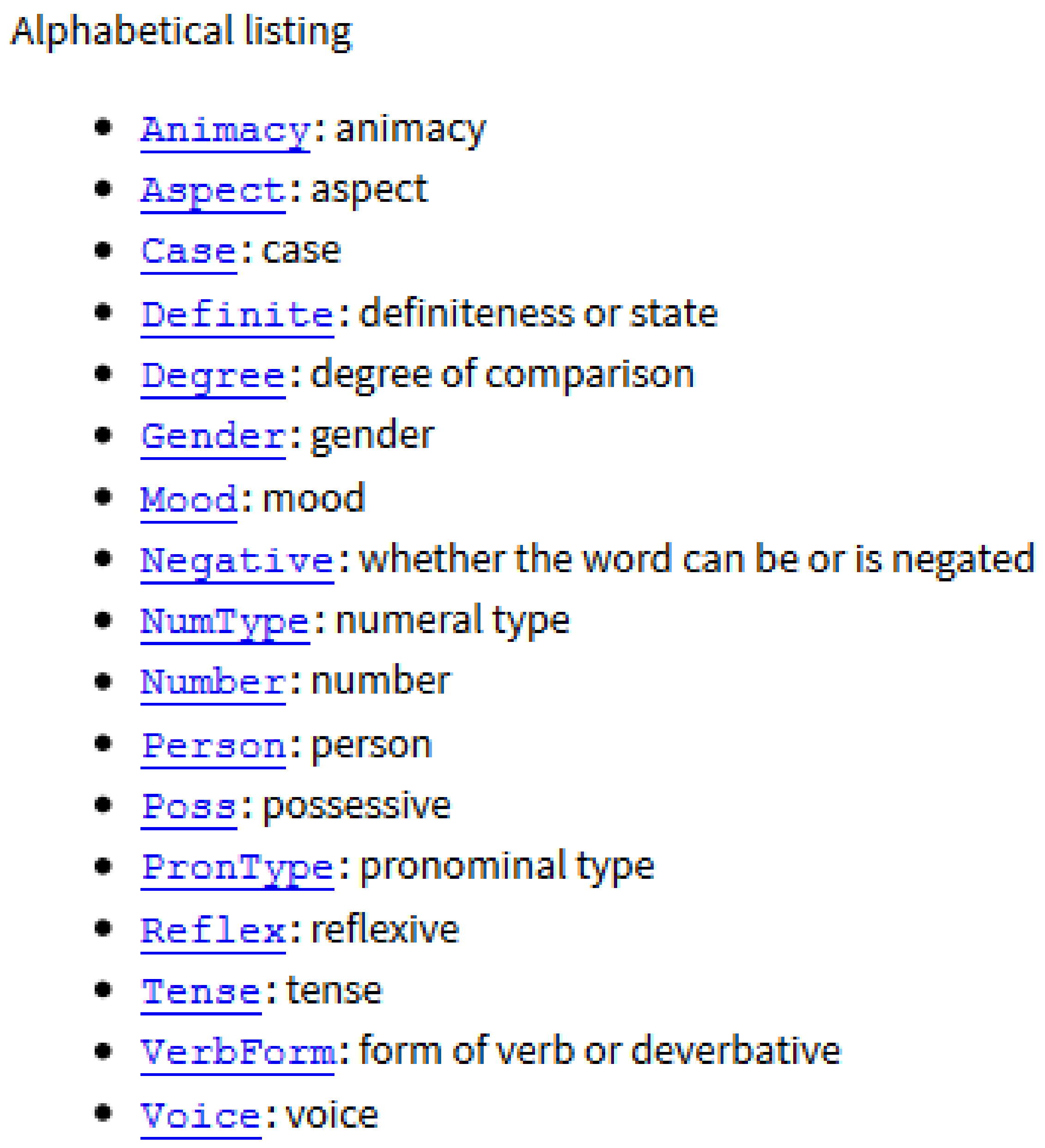


Alphabetical listing

- ADJ: adjective
- ADP: adposition
- ADV: adverb
- AUX: auxiliary verb
- CONJ: coordinating conjunction
- DET: determiner
- INTJ: interjection
- NOUN: noun
- NUM: numeral
- PART: particle
- PRON: pronoun
- PROPN: proper noun
- PUNCT: punctuation
- SCONJ: subordinating conjunction
- SYM: symbol
- VERB: verb
- X: other

ersaldependencies.org/u/pos/

Universal Features



Alphabetical listing

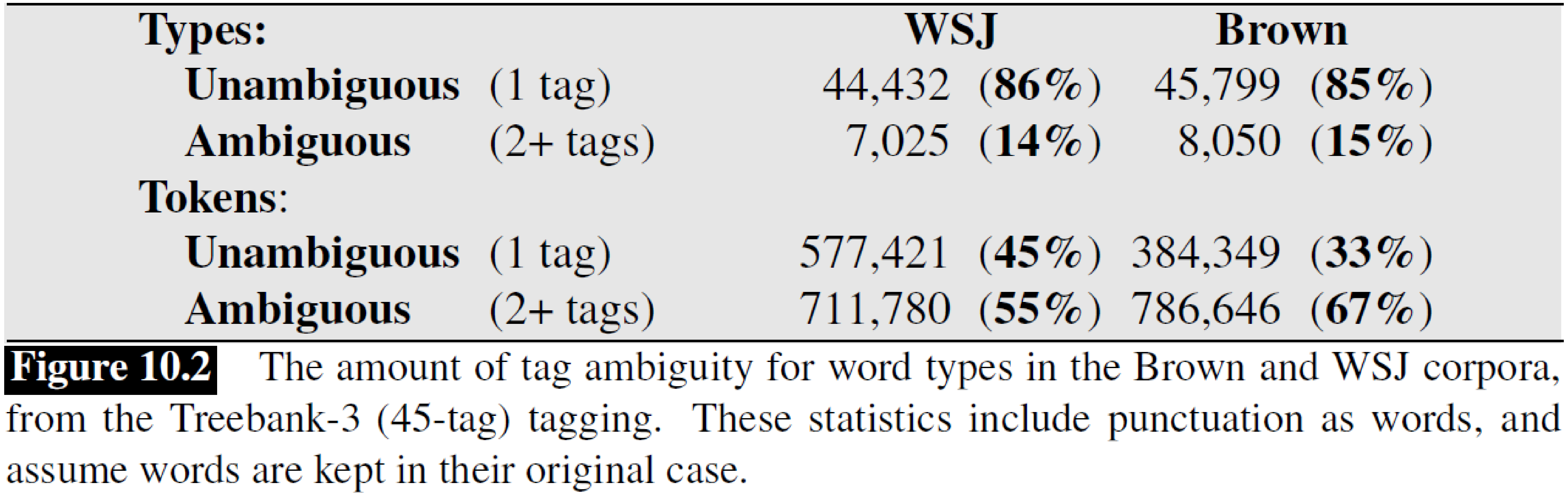
- Animacy: animacy
- Aspect: aspect
- Case: case
- Definite: definiteness or state
- Degree: degree of comparison
- Gender: gender
- Mood: mood
- Negative: whether the word can be or is negated
- NumType: numeral type
- Number: number
- Person: person
- Poss: possessive
- PronType: pronominal type
- Reflex: reflexive
- Tense: tense
- VerbForm: form of verb or deverbative
- Voice: voice

http://universaldependencies.org/u/feat/

Some Observations

- Ambiguity
  - count (noun) vs. count (verb)
  - 11% of all types but 40% of all tokens in the Brown corpus are ambiguous.
  - Examples
    - *like* can be tagged as ADP VERB ADJ ADV NOUN
    - *present* can be tagged as ADJ NOUN VERB ADV

POS Ambiguity



Types:		**WSJ**	**Brown**
Unambiguous	(1 tag)	44,432 (**86%**)	45,799 (**85%**)
Ambiguous	(2+ tags)	7,025 (**14%**)	8,050 (**15%**)
Tokens:			
Unambiguous	(1 tag)	577,421 (**45%**)	384,349 (**33%**)
Ambiguous	(2+ tags)	711,780 (**55%**)	786,646 (**67%**)

Figure 10.2 The amount of tag ambiguity for word types in the Brown and WSJ corpora, from the Treebank-3 (45-tag) tagging. These statistics include punctuation as words, and assume words are kept in their original case.

Example from J&M

Example

- Bethlehem/NNP Steel/NNP Corp./NNP ,/, hammered/VBN by/IN higher/JJR **costs/NNS**
- Bethlehem/NNP Steel/NNP Corp./NNP ,/, hammered/VBN by/IN higher/JJR **costs/VBZ**

Classifier-based POS Tagging

- A baseline method would be to use a classifier to map each individual word into a likely POS tag
  - Why is this method unlikely to work well?

Sources of Information

- Bethlehem/NNP Steel/NNP Corp./NNP ,/, hammered/VBN by/IN higher/JJR **costs/NNS**
- Bethlehem/NNP Steel/NNP Corp./NNP ,/, hammered/VBN by/IN higher/JJR **costs/VBZ**

- Knowledge about individual words
  - lexical information
  - spelling (-or)
  - capitalization (IBM)
- Knowledge about neighboring words

Evaluation

- Baseline
  - tag each word with its most likely tag
  - tag each OOV word as a noun.
  - around 90%
- Current accuracy
  - around 97% for English
  - compared to 98% human performance

NLP