Unit 3: Application Areas

3d. Language Processing

Probabilistic language models

- Sometimes we want to know "how likely is this string of words W"?
  - Example: language model in ASR
  - n-gram model: Markov assumption
    - bigram: $P(W)=P(w_1)P(w_2|w_1)P(w_3|w_2)P(w_4|w_3)\ldots$
- A language model is a model of language
  - Can evaluate model by computing probability of held out test set
  - Better model = higher probability

Training Language Models

- Get a whole bunch of text data (corpus)

  …go to the market…

  …want to go to…

  …give the ball to him…

  …have to leave now…

  …supposed to go on Monday…

- Simplest model: Learn $P(w_{t+1}|w_t)$ (bigram)

  $P(\langle the,go,him,leave \rangle | to) = \langle 0.2,0.4,0.2,0.2 \rangle$

Training Language Models

- Problem: what's the probability of P(have|to)?

  P(<the,go,him,leave>|to) = <0.2,0.4,0.2,0.2>

  With this model, P(have|to)=0!

- Smoothing techniques needed to estimate unseen word pairs
  - Add small counts for all words (bad idea in practice)
  - Backoff smoothing: use P(have) to help estimate P(have|to)

    P(have|to) = P(have) BackoffWeight(to)
  - Interpolation:
    P(have|to)= α P(have|to) + (1-α) P(have)

N-grams vs. CFGs

- N-grams lack structure
  - The dog
  - The big dog
  - The big red dog
  - The big red smelly dog
- Plain CFGs don’t assign probabilities
  - Solution: add probabilities to CFG rules

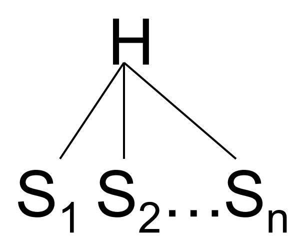
Probabalistic CFGs

- If W is a word sequence, and T is a tree:

$$P(W) = \sum_T P(W,T) = \sum_T P(W|T)P(T)$$

  - Why multiple trees? There can be multiple parses of a sentence.
  - How to calculate P(T)? If T is headed by rule R: H->$S_1 \ldots S_n$

$$P(T) = P(R)\prod_i P(S_i)$$



Rule probabilities

- If $R_1 \ldots R_n$ are the only rules with the same LHS nonterminal, then

$$\sum_{i=1}^{n} P(R_i) = 1$$

  - 0.96 S -> NP VP
  - 0.04 S -> VP

Example

- Non-terminal rules (P(T)):
  - 1 S -> NP VP
  - 0.8 VP->V NP
  - 0.2 VP->V NP PP
  - 0.1 NP -> NP PP
  - 0.9 NP -> Det N
  - 1 PP->P NP
- Terminal (lexical) rules (P(W|T)):
  - Det-> 0.5 the | 0.5 a
  - N-> 0.4 man | 0.3 boy | 0.3 binoculars
  - V-> 1 saw
  - P-> 1 with

The man saw the boy with the binoculars

How do you get rule probabilities?

- Use a corpus of text
  - Must be specially marked up for parses
  - English: Penn Treebank

Penn Treebank example

```
( (S
   (NP-SBJ
     (NP (NNP Pierre) (NNP Vinken) )
     (, ,)
     (ADJP
       (NP (CD 61) (NNS years) )
       (JJ old) )
     (, ,) )
   (VP (MD will)
     (VP (VB join)
       (NP (DT the) (NN board) )
       (PP-CLR (IN as)
         (NP (DT a) (JJ nonexecutive) (NN
     director) ))
       (NP-TMP (NNP Nov.) (CD 29) )))
   (. .) ))
```

```
( (S
   (NP-SBJ (NNP Mr.) (NNP Vinken) )
   (VP (VBZ is)
     (NP-PRD
       (NP (NN chairman) )
       (PP (IN of)
         (NP
           (NP (NNP Elsevier) (NNP N.V.) )
           (, ,)
           (NP (DT the) (NNP Dutch) (VBG
     publishing) (NN group) )))))
   (. .) ))
```

What if you don't have a treebank?

- Assumption: you still know rules, just not the probabilities
- Inside-outside algorithm
  - EM for parsing probabilities
  - Like the forward-backward algorithm in HMMs
  - In any EM problem:
    - What are the observed variables?
    - What are the hidden variables?

Inside-outside & EM

- Start with some random probabilities for each rule
- E-step: determine a probability for each parse
  - Same as finding P(T) in “the boy with the telescope”
- M-step: given parse probabilities from entire corpus, update P(Rules)
- Continue around E/M steps until convergence

Unknown structures

- What if you don't even know rules ahead of time?
- Can infer Chomsky Normal Form rules
  - X -> Y Z
  - X -> t
- This becomes a structural-EM problem
- Problems:
  - I-O algorithm slow ($O(n^3t^3)$), structural EM worse
  - Structures learned are often not linguistically plausible
  - PCFGs often not good at local dependencies

Local dependencies

- The man sees the boy with the binoculars

Local dependencies

- The man sees the boy with the binoculars
- The man sees the boy with the bat

- The words are really not independent of the parse tree
- One solution: lexicalized grammars
  - Add dependency on words

Lexicalized grammars

- One word is designated “head”

  P(VP[see] -> V[see] NP PP[binoculars])

  P(NP[boy] -> Det N[boy])

  P(VP[see] -> V[see] NP)

  P(NP[boy]-> Det N[boy] NP[bat])

- Problem: the more specialized the grammar, the fewer data there are to train probabilities

Information retrieval

- Another use for probabilistic language models is information retrieval
  - Given a query Q, find the *relevant* documents D that best satisfy the query
  - relevance (R) is a binary variable
- There are several components to an IR problem
  - Document collection
  - Query (posed in a query language)
  - Result set (all documents for which r is true)
  - Presentation (e.g., ranked list)
- Parsing is too difficult, use simpler measures

Document relevance

- Given document D & query Q, what is probability that D is relevant?

$$P(r \mid D,Q) = \frac{P(D,Q \mid r)P(r)}{P(D,Q)} = \frac{P(Q \mid D,r)P(D \mid r)P(r)}{P(D,Q)}$$

$$= P(Q \mid D,r)P(r \mid D)\frac{P(D)}{P(D,Q)}$$

- Compare against

$$P(\neg r \mid D,Q) = P(Q \mid D,\neg r)P(\neg r \mid D)\frac{P(D)}{P(D,Q)}$$

$$= P(Q \mid \neg r)P(\neg r \mid D)\frac{P(D)}{P(D,Q)}$$

Relevance ratios

- Instead of maximizing P(r|D,Q), maximize ratio of P(r|D,Q)/P(~r|D,Q)

$$\frac{P(r \mid D,Q)}{P(\neg r \mid D,Q)} = \frac{P(Q \mid D,r)}{P(Q \mid \neg r)} \frac{P(r \mid D)}{P(\neg r \mid D)}$$

Relevance ratios

- Instead of maximizing P(r|D,Q), maximize ratio of P(r|D,Q)/P(~r|D,Q)

$$\frac{P(r \mid D,Q)}{P(\neg r \mid D,Q)} = \frac{P(Q \mid D,r)}{P(Q \mid \neg r)} \frac{P(r \mid D)}{P(\neg r \mid D)}$$

Probability that query is generated by relevant document

If document is irrelevant, prob. of query words SAME FOR ALL DOCS

Query-independent odds that document is relevant

Computing query generation probabilities

- Use a "bag of words" model
  - Ordering doesn't matter: unigram model
  - "Bush hates Pelosi" & "Pelosi hates Bush" have same impact
- For every document D, assume it is relevant and compute P(Q|D):

$$P(Q \mid D, r) = \prod_j P(Q_j \mid D, r)$$

Evaluating IR

- Two numbers to describe performance
  - Precision: how many documents were relevant?
  - Recall: how many relevant documents were found?

	in resultset	~in resultset
r	30	20
~r	10	40

  - Precision: P(r | in resultset)=30/(10+30)=.75
  - Recall: P(in resultset | r)=.6
- Useful concept in other domains (parsing)

IR extensions

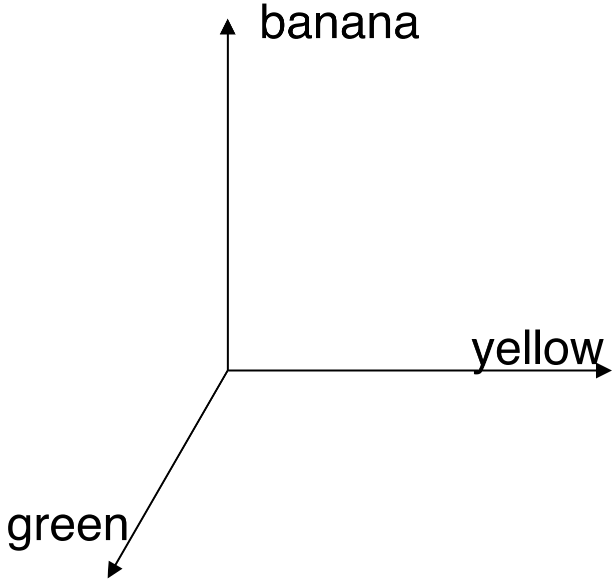
- Several tricks used to find “close” matches
  - Stemming: raining -> rain, pants -> pant
  - Synonyms
  - Relevance feedback
    - Get user to tell you if doc is relevant (more like this)
    - Add document words to query to expand terms

Non-probabilistic IR: Vector Space Model

- Again treat document as a bag of words
- Conceptualize document as a vector in n-space
  - Dimensions determined by vocabulary
  - Vector normalized to have length 1 in n-space
- Query is also a vector in same n-space
  - Find the documents that are closest to the query vector

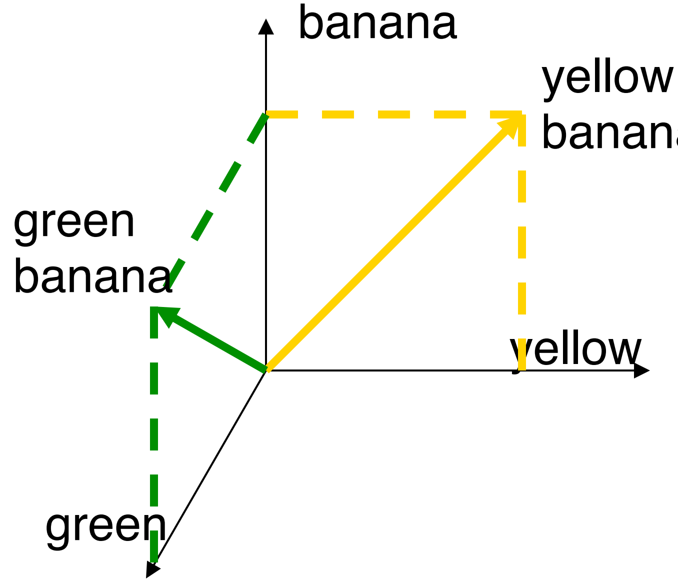
Vector-space IR

- Vocab:
  yellow, green,
  banana



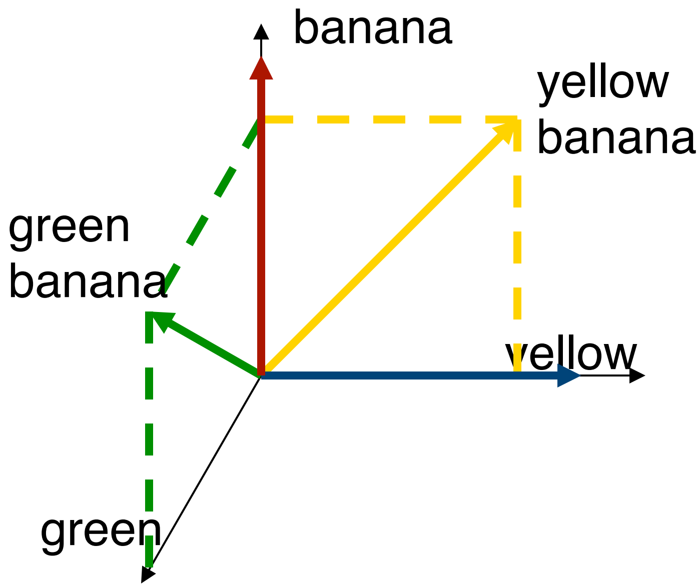
Vector-space IR

- D1: green banana
- D2: yellow banana



Vector-space IR

- D1: green banana
- D2: yellow banana
- Q1: yellow
  - retreives D2
- Q2: banana
  - retreives D1 + D2
- Metric: cosine distance between vectors



Machine Translation (MT)

- One of the original AI holy grails
- Difficult problem: you can't just translate word for word from one langauge to another
  - English: no
  - German: nein, doch (used to contradict)
    - Brauchst du Eis? Nein.
    - *Do you want some ice cream? No.*
    - Hast du kein Eis gegessen? Doch.
    - *Haven't you had any ice cream? No, actually, I have.*
  - also: Eis = ice, ice cream
- Word order can also be different

Machine language strategies

- Rough translation: get the gist across, clean up afterwards
- Restricted source translation: use special domains and formats
  - Weather reports
- Preedited translations: like restricted source, reformat input document into simplified English (or other language)
  - Choose structures that are easy to translate
- Literary translation: capture all meanings
  - Beyond current MT capabilites

Translating at various levels

- Words
  - John loves Mary -> Jean aime Marie
- Syntax
  - S(NP(John) VP(loves NP(Mary)) -> S(NP(Jean) VP(aime NP(Marie))
- Semantics
  - Loves(John,Mary) -> Aime(Jean,Marie)
- Interlingua
  - Attraction(NamedJohn, NamedMary,High)
  - semantic representation that is lang. independent

Statistical MT

- Mostly takes a word-level approach
- Uses probabilistic language models to learn relationship between languages
  - E.g. English string E, French string F
- Let's say that you want to translate E to F

$$\arg\max_F P(F \mid E) = \arg\max_F \frac{P(E \mid F)P(F)}{P(E)}$$

$$= \arg\max_F P(E \mid F)P(F)$$

Why Bayes rule?

- Could compute P(F|E) directly instead of using P(E|F)
- However, we can use a simpler model of P(E|F) and then clean up using P(F)
  - P(F): model of “good French”
- Simple model: $P(E \mid F) = \prod_i P(E_i \mid F_i)$
  - P(the dog|le chein) = P(the|le) P(dog|chien)

What about non 1–1 mappings?

- Sometimes there are different numbers of words in each language
  - Eis/ice cream
  - home/à la maison
- Fertility: words get copied 0 or more times
  - P(home|à la maison)=
    P(0|à)P(0|la)P(home|maison)
  - P(à la maison|home)=
    P(à|home)P(la|home)P(maison|home)

What about different word orders?

- chien brun = brown dog (+1 -1)
- Offset: how far do you have to move a word?
  - P(offset|position, englishlen, frenchlen)
- Combine all of this information together to get P(E|F)

Example combination

Source French	le	chien	brun	n’	est	pas	allé	à	la	maison
Fertility	1	1	1	1	1	0	1	0	0	1
Transformed French	le	chien	brun	n’	est		allé			maison
Word choice	the	dog	brown	not	did		go			home
Offset model	0	+1	-1	+1	-1		0			0
Target	the	brown	dog	did	not		go			home

Putting it all together

- Now we have P(E|F)
  - But we wanted P(F|E)!
- Hypothesize French sentence F
  - Evaluate P(E|F)P(F)
- But… can’t just enumerate all F
  - Use the models to make some intelligent guesses, search over possible sequences
  - Like ASR problem