

Natural Language Processing

Homework
Due Mon. Oct. 29

Reading assignment:
Vehicles, pp. 62-69
Jurasky & Martin, Chapter 1

Optional:
Jurasky & Martin Chapter 4 (through 4.5.2) (N-grams), Chapter 23 (Question answering and summarization)

Reading questions and exercises

"Open the pod bay doors, HAL."

"I'm sorry Dave, I'm afraid I can't do that."

What is involved in NLP?

Phonetics / Phonology:
Recover sequence of words from audio signal.
"Open the pod bay doors, HAL."

What is involved in NLP?

Phonetics / Phonology:
Recover sequence of words from audio signal.
"Open the pod bay doors, HAL."

© Center for Speech and Hearing Sciences, Harvard Medical School

Take a sequence of words and generate an audio signal.
"I'm sorry Dave, I'm afraid I can't do that."

Morphology:

Recognize plurals, contractions, etc.

*"Open the pod bay **doors**, HAL."*

*"I'm sorry Dave, I'm afraid I **can't** do that."*

Syntax:

Parse utterance

Determine type of utterance (e.g., question, request, command)

```

1: Open the pod bay doors, HAL.
Parses found: 1 [1]
  <SENTENCE>-|
  <CENTER>-|
  <IMPERATIVE>-|
  <VO>-|
  <LVR>-| |
Open ..... <*V>-| | v: open
  <OBJECT>-|
  <NSTG>-|
  <NSTG>-|
  <LNR>-|
  <LNR>-|
  <TPOS>-| |
  <LTR>-| |
the ..... <*T>-| |
  <NPOS>-| |
  <NNNS>-| |
pod ..... <*N>-| | n: pod
bay ..... <*N>-| | n: bay
doors ..... <*N>-| | n: door
  <COMMASTG>-| |
, ..... | |
  <RN>-|
  <APPOS>-|
  <LNR>-|
  <NVAR>-|
HAL ..... <*N>-|
  <ENDMARK>-|
    
```

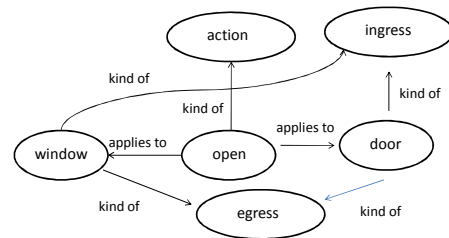
Lexical Semantics:

Determine meaning of component words

- 1: Open the pod bay doors, HAL.
- Parse Nr: 1
- Open
- [v: open](#)
- 1. **open, open_up** -- (cause to open or to become open: "Mary opened the car door")
- pod
- [n: pod](#)
- 4. **fuel_pod, pod** -- (a detachable container of fuel on an airplane)
- bay
- [n: bay](#)
- 1. **bay** -- (an indentation of a shoreline larger than a cove but smaller than a gulf)
- doors
- [n: door](#)
- 1. **door** -- (a swinging or sliding barrier that will close the entrance to a room or building; "he knocked on the door"; "he slammed the door as he left")

Compositional semantics:

Determine meaning from combination of these components.



Pragmatics:

Adapt phrasings to current situation, to accomplish goals.

E.g., politeness:

"I'm sorry Dave, I'm afraid I can't do that."

Discourse:

Conversational behavior follows conventions (don't interrupt, respond to requests and questions, etc.)

NLP does all this, plus dealing with ambiguity in language

- “I made her duck”
- Meanings?

Example of statistical language models: n-grams

- Estimates probability distribution of a word w , given n words that have come before in the sequence.
 $P(w | w_1, w_2, \dots, w_n)$
- Purpose: to guess next word from previous words to disambiguate:
 - “Students have access to a list of course requirements”
 - “Would you like a drink of [garbled]?”
 - “That cake tasted goop!”

- Applications throughout NLP
- E.g., speech recognition, machine translation, intelligent spell-checking,

- What is $P(\text{good} | \text{that cake tasted})$?
- What is $P(\text{goop} | \text{that cake tasted})$?
- Can estimate from a large corpus:
 - $P(w | w_1, \dots, w_n) = \text{frequency of } w_1 \dots w_n w \text{ divided by frequency of } w_1 \dots w_n$
 - Example: Use Google

Problem! Web doesn't give us enough examples to get good statistics.

One solution:

- Approximate $P(w | w_1, \dots, w_n)$ by using small n (e.g., $n=1$: bigrams).

Bigram example:

$P(\text{good} | \text{tasted})$ vs. $P(\text{goop} | \text{tasted})$

(calculate using Google)

Trigram:

$P(\text{good} | \text{cake tasted})$ vs. $P(\text{goop} | \text{cake tasted})$

Now can calculate probability of utterance:

$$\begin{aligned}
 &P(\text{that cake tasted goop}) \\
 &= P(\text{that} | \langle s \rangle) P(\text{cake} | \text{that}) P(\text{tasted} | \text{cake}) P(\text{goop} | \text{tasted}) \\
 &P(\langle /s \rangle | \text{goop}).
 \end{aligned}$$

$\langle s \rangle$ = sentence start marker

$\langle /s \rangle$ = sentence end marker

Smoothing

Needed to overcome problem of sparse data.

E.g., even in a large corpus, can get zero probability for valid bigrams).

Laplace smoothing (or “add-one” smoothing):

Add 1 to all the bigram counts

$$P(\text{good} | \text{tasted}) = 311,001 / 4,260,001$$

Problem : if data is sparse, Laplace smoothing gives too much probability to the (originally) zero counts.

Good-Turing discounting

- Idea: estimate amount of probability to assign to zero-count n-grams by looking at one-count n-grams.
- Define n_c as number of n-grams that occur c times
- n_0 = number of n-grams that occur zero times.
- n_1 = number of n-grams that occur once.
- Original count is c . Replace with

$$c^* = (c+1) n_{c+1} / n_c$$

Why is this a good thing to do?

Results in assigning count of n_1 / n_0 to originally zero count n-grams

What are the weaknesses of the n-gram method for predicting the next word in an utterance?

Learning n-gram models

- Use training corpus; evaluate by predicting bigrams in test corpus
- Modify parameters (e.g., weights) using “hold out” or “validation” set.

Question answering, or retrieval of information from queries

- “Bag of words” approach

“I see what I eat” = “I eat what I see”

“To be or not to be” = “be be not or to to”

Vector space model

A document is represented as a vector of weighted word counts. E.g.,

“To be or not to be, that is the question. Whether 'tis nobler in the mind to suffer the slings and arrows of outrageous fortune, or to take arms against a sea of troubles, and by opposing end them?”

Vector $\mathbf{d} = (c_1, c_2, \dots, c_M)$, where M is the total number of words in the system's dictionary.

$$\mathbf{d} = (1 \dots \quad 2 \dots \quad 1 \dots \quad 2 \dots \quad 1 \dots \quad 4 \dots)$$

Suppose these terms are found
in two on-line recipes

Recipe 1:

Chicken: 8
Fried: 2
Oil: 7
Pepper: 4
 $\mathbf{d}_i = (8, 2, 7, 4)$

Recipe 2:

Chicken: 6
Fried: 0
Oil: 0
Pepper: 0
 $\mathbf{d}_k = (6, 0, 0, 0)$

Query: fried chicken
 $\mathbf{q} = (1, 1, 0, 0)$

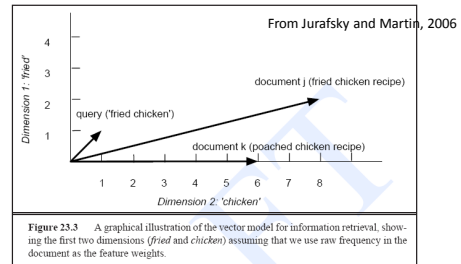


Figure 23.3 A graphical illustration of the vector model for information retrieval, showing the first two dimensions (fried and chicken) assuming that we use raw frequency in the document as the feature weights.

$$\text{sim}(\vec{q}, \vec{d}_j) = \frac{\sum_{i=1}^N w_{i,q} \times w_{i,j}}{\sqrt{\sum_{i=1}^N w_{i,q}^2} \times \sqrt{\sum_{i=1}^N w_{i,j}^2}}$$

$$= \frac{\text{dot-product}(\mathbf{q}, \mathbf{d}_j)}{\text{length}(\mathbf{q}) \times \text{length}(\mathbf{d}_j)} = \text{"normalized dot product"}$$

Weaknesses of vector space model?

Weaknesses of vector space model?

- Homonymy ("lie", "bore")
- Polysemy ("concrete", "right")
- Synonymy ("canine/dog", "party/celebration")

Improving performance
of vector-space models

?

Improving performance
of vector-space models

- Improve query:

Improving performance of vector-space models

- Improve query:
 - Relevance feedback
 - Query expansion

Question answering

Question	Answer
Where is the Louvre Museum located?	in Paris, France
What's the abbreviation for limited partnership?	L.P.
What are the names of Odin's ravens?	Huginn and Muninn
What currency is used in China?	the yuan
What kind of nuts are used in marzipan?	almonds
What instrument does Max Roach play?	drums
What's the official language of Algeria?	Arabic
What is the telephone number for the University of Colorado, Boulder?	(303)492-1411
How many pounds are there in a stone?	14

Figure 23.7 Some sample factoid questions and their answers.

Components of a question answering system

1. Question processing:

- From question, create list of keywords that forms a query

One method: do syntactic parsing; query's keywords formed from terms found in noun phrases

Also: query expansion

2. Question classification

- What is the expected *answer type*?

“Who was the fourth U.S. president?”

Expected answer type is proper noun, name of a person

“What is the name of China's currency?”

Expected answer is a proper noun, name of a currency

“What is the largest city in North America?”

Expected answer is a proper noun, name of a city

“How long does it take to roast a turkey?”

Expected answer is a number, in units of time

- Can use set of hand-coded rules
- Can use supervised machine learning
- In general, need ontologies!

3. Passage retrieval

- Submit query
- Extract set of potential answer passages from retrieved set of documents.
- Run answer-type classification on all passages. Filter out ones that don't provide needed answer type.

- Rank remaining passages based on set of features

E.g.,

- Number of named entities of the right type present
- Number of question keywords present
- Rank of document containing passage
- Proximity of keywords from original query to each other.
- N-gram overlap between the passage and the question

4. Answer processing

- Extract specific answer from the passage
- Use info about expected answer type together with regular expression patterns designed by programmer or learned

E.g.,
 <AP> such as <QP>
 (AP = answer phrase, QP = question phrase)
 Example: "What is **polysemy**?"
 - "linguistic terms such as polysemy"
 <QP> (an <AP>)
 Example: "What is a **caldera**?"
 - "the Long Valley caldera, a **volcanic crater** 19 miles long"

Another method for answer extraction:
 n-gram tiling

n-gram tiling:

1. Present query

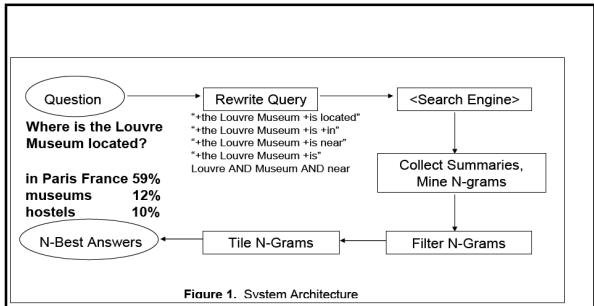
For snippets returned from web search engine:

2. Compute all unigrams, bigrams, and trigrams
3. Weight each one as a function of number of snippets the n-gram occurred in

3. Score n-grams by how well they match predicted answer type (computed by hand-written filters built for each answer type).
4. Concatenate best-scoring overlapping n-gram fragments into longer answers.

"San" and "Francisco"

"light amplification by stimulated emission of radiation"



From Brill et al. (2002), An analysis of the AskMSR question-answering system