

Natural Language Processing

“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.

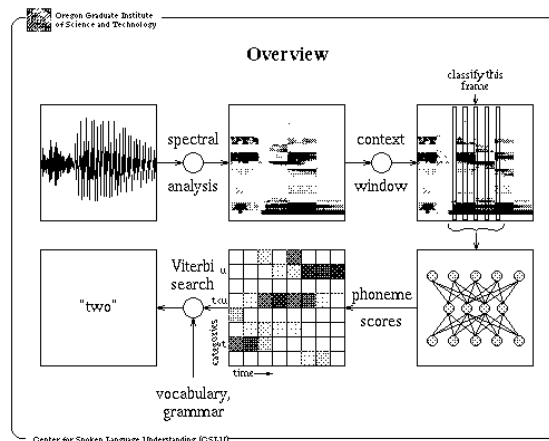
"Open the pod bay doors, HAL."

What is involved in NLP?

Phonetics / Phonology:

Recover sequence of words from audio signal.

"Open the pod bay doors, HAL."



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>-| |
      <NSTGO> |
      <NSTG> |
      <LNR> |
      <LN>-| |
      <TPOS>-| | |
      <LTR> | | |
the .....<*T> | | |
      <NPOS>-| | |
      <NNN> | | |
pod .....<*N>-| | | n: pod
bay .....<*N>-| | | n: bay
      <NVAR>-| | |
doors .....<*N> | | | n: door
      <COMMASTG>-| | |
, .....| | |
      <RN>-| | |
      <APPOS> | | |
      <LNR> | | |
      <NVAR> | | |
HAL .....<*N> | | |
      <ENDMARK>-|

```

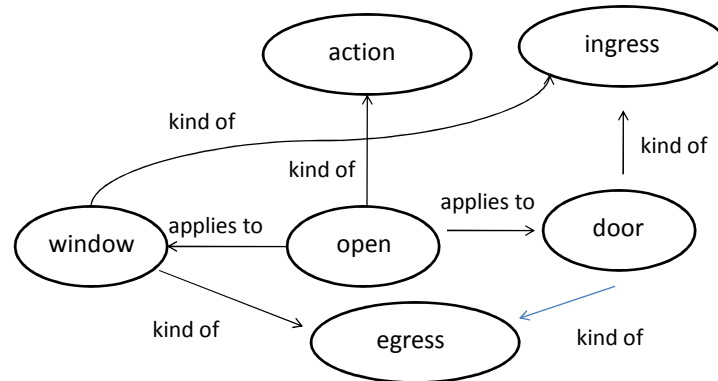
Lexical Semantics:

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?

NLP does all this, plus dealing with ambiguity in language

- “I made her duck”
- Meanings?
 - I cooked waterfowl for her.
 - I cooked waterfowl belonging to her
 - I created the (plaster?) duck she owns
 - I caused her to quickly lower her head or body
 - I waved my magic wand and turned her into an undifferentiated waterfowl

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:
 - “Would you like a drink of [garbled]?”
 - “That cake tasted goop!”

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

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

Typically, bigrams are used:

Let candidate utterance $s = w_1 w_2 \dots w_n$

Then $P(s) = \prod_{k=1}^n P(w_k | w_{k-1})$

$P(w_k | w_{k-1}) = \frac{C(w_{k-1} w_k)}{C(w_{k-1})}$ where C stands for "count"

<s> That cake tasted goop </s>

<s> That cake tasted good </s>

N-gram approximation to Shakespeare (Jurafsky and Martin, 2000)

- Trained unigram, bigram, trigram, and quadrigram model on complete corpus of Shakespeare's works (including punctuation).
- Use these models to generate random sentences

Unigram model

1. *To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have.*
2. *Every enter now severally so, let*
3. *Hill he late speaks; or! a more to leg less first you enter*
4. *Are where exeunt and sighs have rise excellency took of...Sleep knave we. near; vile like.*

Bigram model

1. *What means, sir: I confess she? then all sorts, he is trim, captain.*
2. *Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.*
3. *What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?*
4. *Thou whoreson chops. Consumption catch your dearest friend, well, and I know where many mouths upon my undoing all but be, how soon, then; we'll execute upon my love's bonds and we do you will?*

Trigram model

1. *Sweet prince, Falstaff shall die. Harry of Monmouth's grave.*
2. *This shall forbid it should be branded, if renown made it empty.*
3. *Indeed the duke; and had a very good friend.*
4. *Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.*

Quadrigram model

1. *King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;*
2. *Will you not tell me who I am?*
3. *Indeed the short and long. Marry, 'tis a noble Lepidus.*
4. *Enter Leonato's brother Antonio, and the rest, but seek the weary beds of people sick.*

Smoothing

Needed to overcome problem of sparse data.

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

Whitten-Bell discounting

- Idea: probability of seeing a zero-frequency N-gram can be modeled by probability of seeing an N-gram for the first time. (I.e., assume we just haven't seen it yet.)
- Count the number of times we saw N-grams for the first time in our training corpus.
- Total probability mass of all zero-count N-grams estimated by:

$$\frac{\text{Number of observed types (particular N-grams)}}{\text{Number of tokens} + \text{Number of types}}$$

This is probability of new “type event” occurring.
- Can divide equally among all zero-count N-grams to get individual probabilities

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.

arrows... be... not... or... slings... to
 $\mathbf{d} = (1\dots \quad 2\dots \quad 1\dots \quad 2\dots \quad 1\dots \quad 4\dots)$

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

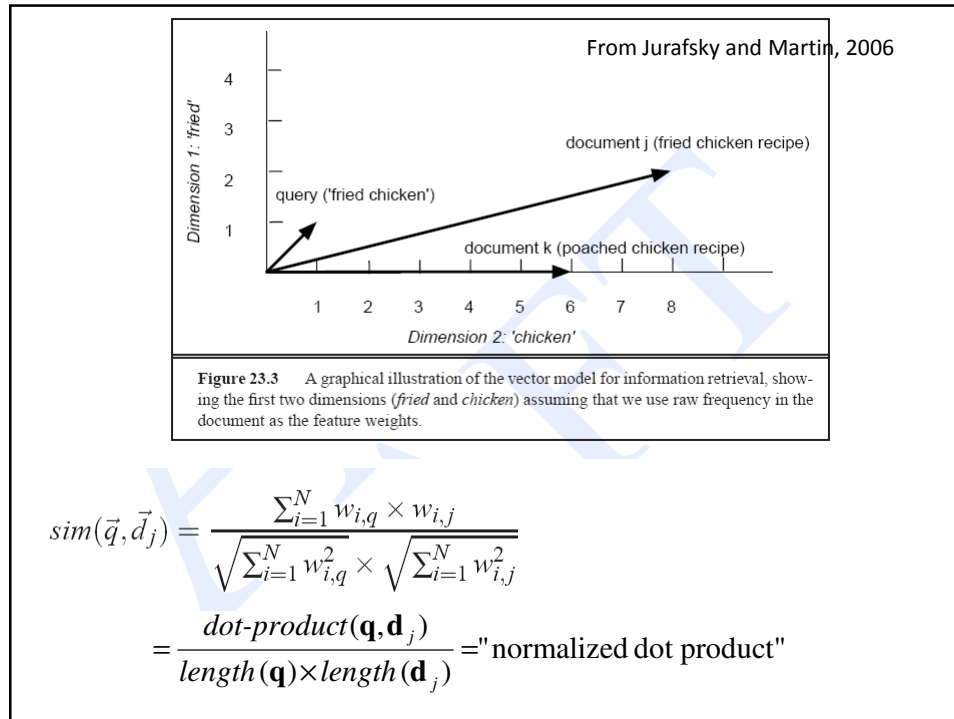
Oil: 0

Pepper: 0

$\mathbf{d}_k = (6, 0, 0, 0)$

Query: fried chicken

$\mathbf{q} = (1, 1, 0, 0)$



Weaknesses of vector space model?

- Homonymy (“lie/lye”, “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:

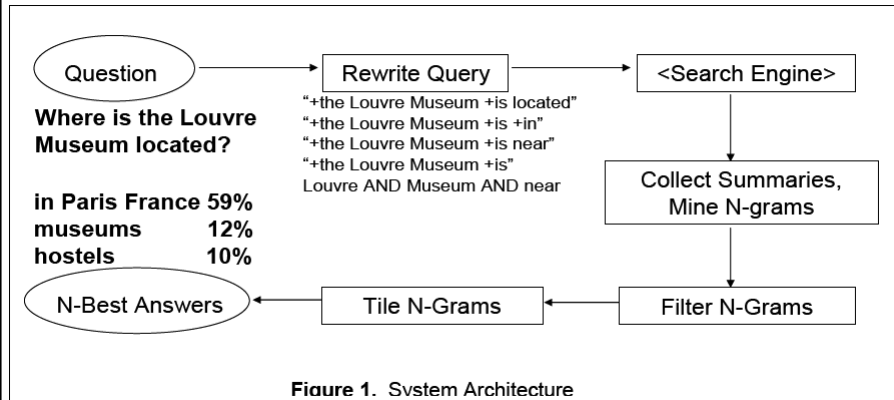
2. Compute all unigrams, bigrams, and trigrams
3. Weight each one as a function of number of snippets the N-gram occurred in

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”

AskMSR Question-Answering System (Brill, Dumais, and Banko, 2002)



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

1. Determine expected answer type of question:

7 question types:

- who-question
- what-question
- where-question
- how-many question
- ...

Filters written manually.

2. Rewrite query as declarative sentence likely to be in text with answer:

“When was the paper clip invented?”

→ “the paper clip was invented”

→ “invented the paper clip”

→ “paper clips were invented”

→ “invented paper clips”

→ paper AND clip AND invented

Can do with simple rewrite rules.

Weights are given to rewrite rules (set manually).

3. N-gram mining:

Send query to search engine.

Collect top M page summaries (snippets)

- less computationally expensive than using entire web page

Extract unigrams, bigrams, and trigrams from each snippet.

4. N-gram filtering:

Filter and reweight N-grams according to how well each candidate N-gram matches expected answer type

(Uses human-constructed filters)

5. N-gram tiling:

Goal is to combine remaining n-grams to produce longer answer

E.g., trigrams ABC and BCD become ABCD

“France is in” “is in Europe”

→ “France is in Europe”

Start with highest scoring N-gram as *current answer*

Iterate through remaining N-grams in order of score – for each one, see if it can be tiled with the current answer. If so, *current answer* ← tiling.

Results

- Tested on 500 queries from TREC data set
- 61% answered correctly – relatively good performance at this competition!

Knowing when we don't know

- However.... no answer is usually better than wrong answer. Need option of responding “I don't know”.
- Built decision tree to predict whether system will answer correctly, based on set of features from question string:
 - unigrams, bigrams, sentence length, number of capitalized words, number of stop words, length of longest word.
- Decision tree didn't work very well. (Are you surprised?)
- You can read about other attempts at this in their paper (in optional reading on the class web page)

Beyond N-grams: Latent Semantic Analysis

- Problem: How to capture semantic similarity between documents in a natural corpus (e.g., problems of homonymy, polysemy, synonymy, etc.)
- In general, N-grams often fail to capture semantic similarity, even with query expansion, etc.
- “LSA assumes that there exists a LATENT structure in word usage – obscured by variability in word choice” (<http://ir.dcs.gla.ac.uk/oldseminars/Girolami.ppt>)

Latent Semantic Analysis (Landauer et al.)

- From training data (large sample of documents), create word-by-document matrix.

Technical Memo Example

Titles:

- c1: *Human machine interface* for Lab ABC computer applications
 c2: A survey of user opinion of computer system response time
 c3: The *EPS user interface* management system
 c4: *System* and human system engineering testing of *EPS*
 c5: Relation of user-perceived response time to error measurement

- m1: The generation of random, binary, unordered trees
 m2: The intersection graph of paths in trees
 m3: Graph minors IV: Widths of trees and well-quasi-ordering
 m4: Graph minors: A survey

A sample dataset consisting of the titles of 9 technical memoranda. Terms occurring in more than one title are italicized. There are two classes of documents - five about human-computer interaction (c1-c5) and four about graphs (m1-m4). This dataset can be described by means of a term by document matrix where each cell entry indicates the frequency with which a term occurs in a document.

From Deerwester et al., [Indexing by latent semantic analysis](#)

Terms	Documents									
	c1	c2	c3	c4	c5	m1	m2	m3	m4	
<i>human</i>	1	0	0	1	0	0	0	0	0	
<i>interface</i>	1	0	1	0	0	0	0	0	0	
<i>computer</i>	1	1	0	0	0	0	0	0	0	
<i>user</i>	0	1	1	0	1	0	0	0	0	
<i>system</i>	0	1	1	2	0	0	0	0	0	
<i>response</i>	0	1	0	0	1	0	0	0	0	
<i>time</i>	0	1	0	0	1	0	0	0	0	
<i>EPS</i>	0	0	1	1	0	0	0	0	0	
<i>survey</i>	0	1	0	0	0	0	0	0	1	
<i>trees</i>	0	0	0	0	0	1	1	1	0	
<i>graph</i>	0	0	0	0	0	0	1	1	1	
<i>minors</i>	0	0	0	0	0	0	0	1	1	

- Now apply “singular value decomposition” to this matrix
- SVD is similar to principal components analysis (if you know what that is)
- Basically, reduce dimensionality of the matrix by re-representing matrix in terms of “features” (derived from eigenvalues and eigenvectors), and using only the ones with highest value.
- Result: Each document is represented by a vector of *features* obtained by SVD.
- Given a new document (or query), compute its representation vector in this *feature space*, compute its similarity with other documents using cosine between vector angles. Retrieve documents with highest similarities.

Result of Applying LSA

From <http://lair.indiana.edu/courses/i502/lectures/lect6.ppt>

	DOC1	DOC2	DOC3	DOC4	DOC5	DOC6	DOC7	DOC8	DOC9
HUMAN	0.16	0.4	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
INTERFACE	0.14	0.37	0.33	0.4	0.16	-0.03	-0.07	-0.1	-0.04
COMPUTER	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
USER	0.26	0.84	0.61	0.7	0.39	0.03	0.08	0.12	0.19
SYSTEM	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
RESPONSE	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
TIME	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.2	-0.11
SURVEY	0.1	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
TREES	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
GRAPH	-0.06	0.34	-0.15	-0.3	0.2	0.31	0.69	0.98	0.85
MINORS	-0.04	0.25	-0.1	-0.21	0.15	0.22	0.5	0.71	0.62

In the above matrix we can now observe:

$$r(\text{human.user}) = 0.94$$

$$r(\text{human.minors}) = -0.83$$

How does it find the latent associations?

From <http://lair.indiana.edu/courses/i502/lectures/lect6.ppt>

- By analyzing the contexts in which the words appear
- The word *user* has co-occurred with words that *human* has co-occurred with (e.g., system and interface)
- It downgrades associations when such contextual similarities are not found

Some General LSA Based Applications

From <http://lsa.colorado.edu/~quesadaj/pdf/LSATutorial.pdf>

Information Retrieval

Text Assessment

Compare document to documents of known quality / content

Automatic summarization of text

Determine best subset of text to portray same meaning

Categorization / Classification

Place text into appropriate categories or taxonomies

**Application: Automatic Essay Scoring
(in collaboration with
Educational Testing Service)**

Create domain semantic space

Compute vectors for essays, add to vector database

To predict grade on a new essay, compare it to ones
previously scored by humans

From <http://lsa.colorado.edu/~quesadaj/pdf/LSATutorial.pdf>

Mutual information between two sets of grades:

human – human .90

LSA – human .81

From <http://lsa.colorado.edu/~quesadaj/pdf/LSATutorial.pdf>

Demo

(<http://www.pearsonkt.com/cgi-bin/prenhall/phMenu.cgi?rubric=6&demo=1>)