Information-retrieval and classification approaches

Anton Leuski

Chicago - 2035

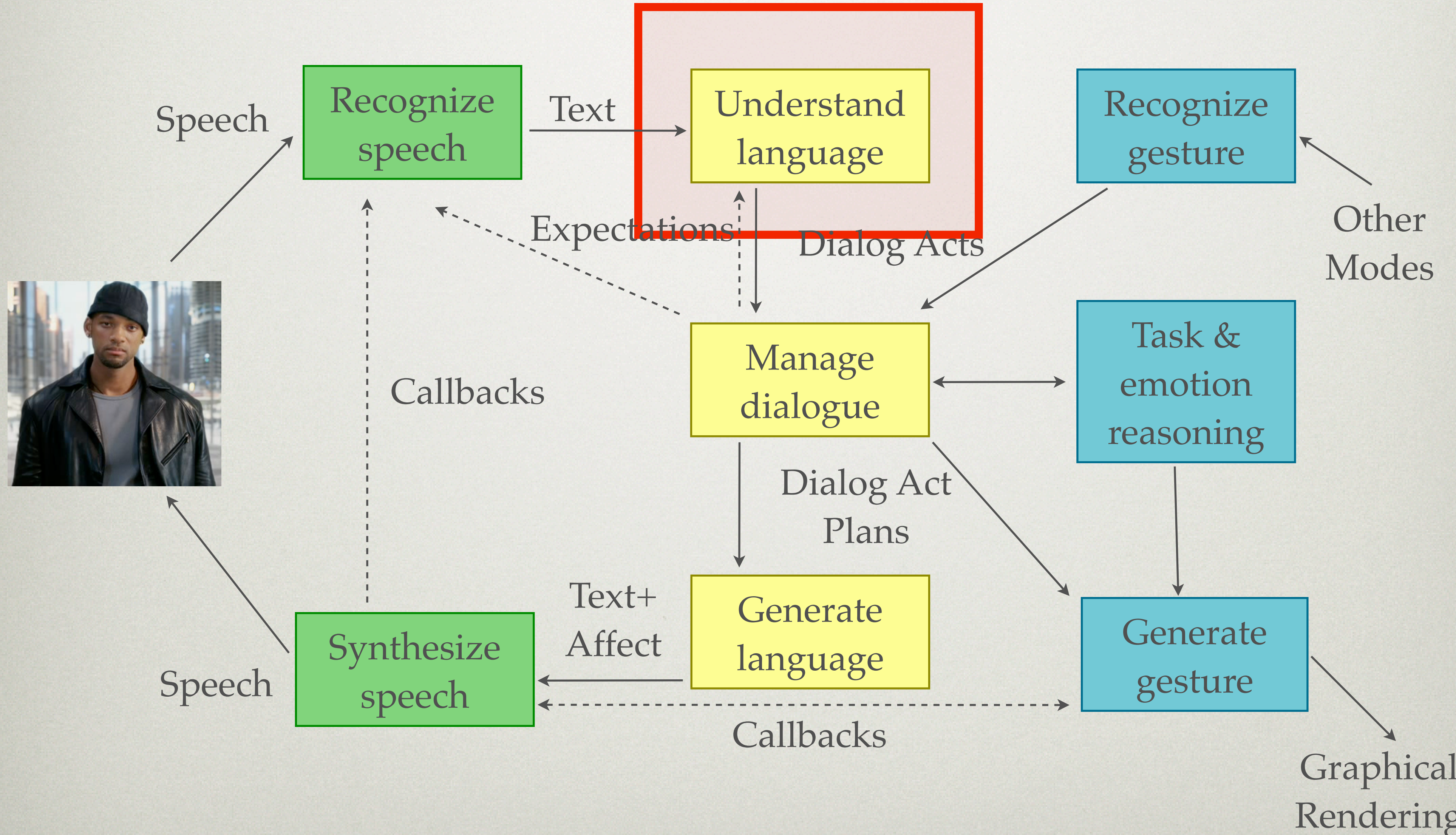
1st Dialog

- Passive system, no initiative
- No context (likely)
- No strategy
- Limited set of responses
- Pre-recorded responses

2nd Dialog

- Initiative
- Emotions
- Strategy
- Response generation
- Unlimited set of responses

Virtual Human



Language Understanding

- Problem: Speech input is often unpredictable
  - Language ambiguity
  - Speech recognition errors
- Solution: Automatically train machines from input-output pairs

Language Understanding

- Text Mapping
  - "Why did you kill yourself" -> "That detective is the right question"
- Information Extraction
  - "**Alpha one six** this is **Bravo two five** **adjust fire** over" -> "**Bravo two five** **adjust fire** out"
- Semantic parsing
  - "Why did you kill yourself" ->

speech-act **<A213>**
 action *info-req*
 actor *detective*
 addressee *hologram*
  type *question*
  q-slot *cause*
   time *past*
   type *kill*
   object *doctor*

Text Mapping

- How do we do the mapping?
- We have...
- ... a set of Q/A pairs - "**Training**" data
- ... a question - "**Test**" data
- we have to select the "correct" answer

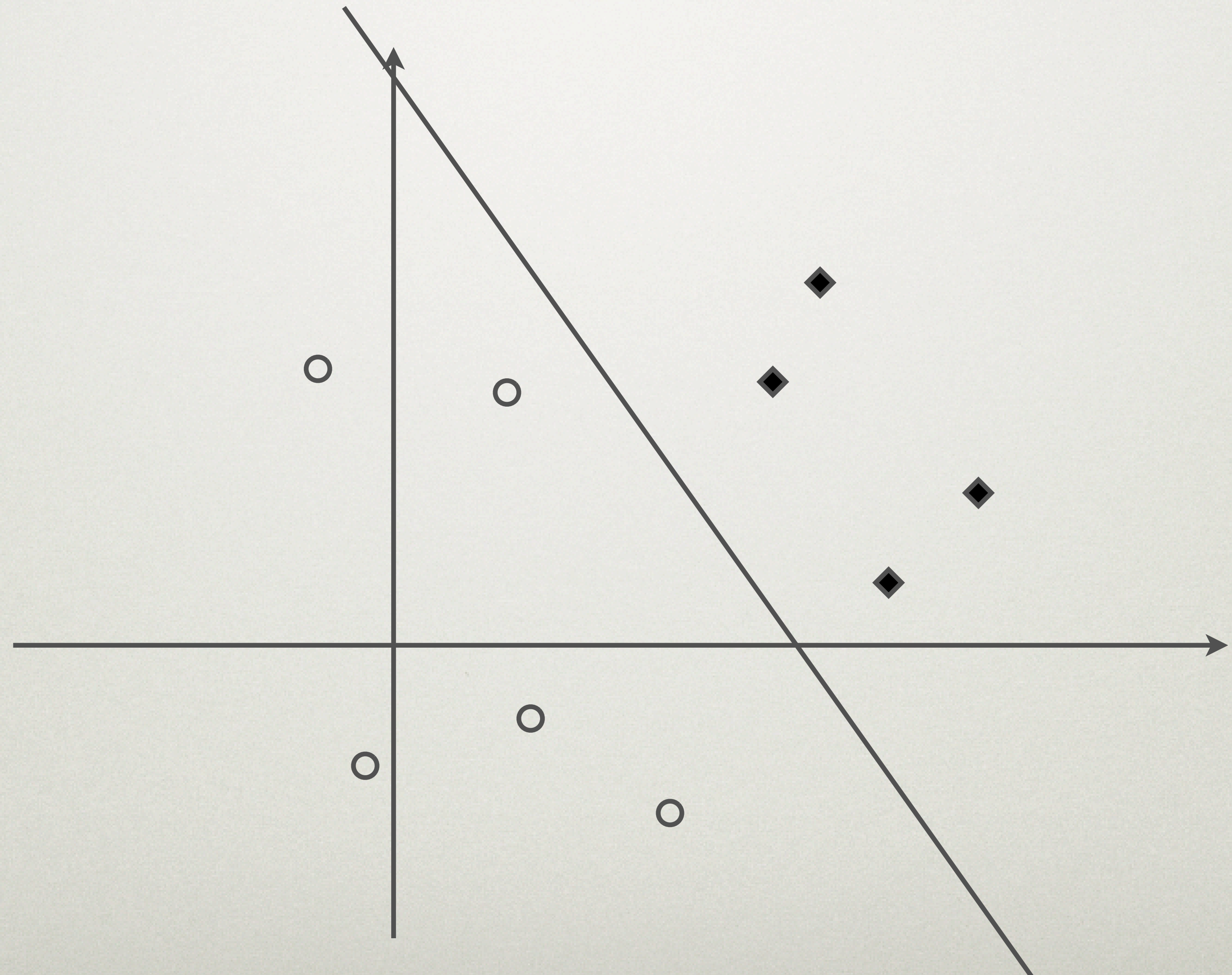
Text Mapping

- Text classification
- Text retrieval

Classification

- Answer = class
- Question = instance
- Training questions = training instances
- Simplest case = 2 classes

Binary classification



Classification

- Text as points?!
- How to compute that line?
- What do we do if the line does not exist?
- What do we do if >2 answers (classes)?

Text as vectors

Why did you kill yourself?

Why	did	you	kill	yourself

Term	tf
why	1
did	1
you	1
...	...

- "Bag of words"
- Stopping
- Stemming

Text as vectors

Why	did	you	kill	yourself

to capture order...

Why did	did you	you kill	kill yourself

Why did you	did you kill	you kill yourself

Term Weights

$$w_{i,j} = \begin{cases} 1 & \text{word } i \text{ is present in string } j \\ 0 & \text{otherwise} \end{cases}$$

$$w_{i,j} = tf_{i,j}$$

$$w_{i,j} = tf_{i,j}/df_i$$

$$w_{i,j} = tf_{i,j}/\log df_i$$

$$w_{i,j} = \frac{tf_{i,j}}{tf_{i,j} + 0.5 + 1.5\frac{doclen}{avgdoclen}} \cdot \frac{\log(\frac{colsize+0.5}{docf_i})}{\log(colsize+1)}$$

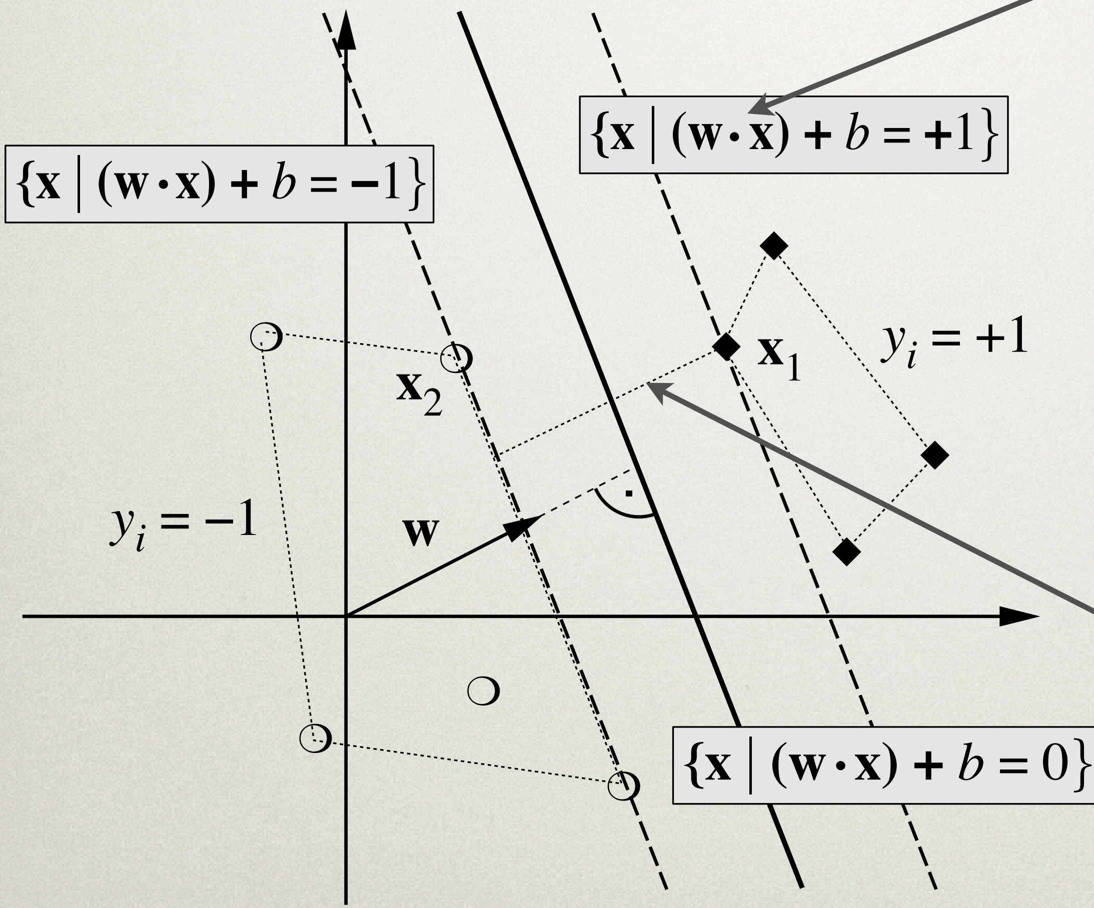
Term	tf	df
why	1	5
did	1	100
you	1	10

Classification

- ~~Text as points?!~~
- How to compute that line?
- What do we do if the line does not exist?
- What do we do if >2 answers (classes)?

Binary classification

dot product



Note:

$$(\mathbf{w} \cdot \mathbf{x}_1) + b = +1$$
$$(\mathbf{w} \cdot \mathbf{x}_2) + b = -1$$

$$\Rightarrow \quad (\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2)) = 2$$

$$\Rightarrow \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot (\mathbf{x}_1 - \mathbf{x}_2) \right) = \frac{2}{\|\mathbf{w}\|}$$

margin width

Binary classification

dot product

- subject to constraints

$$y_i \cdot [(\mathbf{w} \cdot \mathbf{x_i}) + b] \geqslant 1, i = 1...m$$

- maximize margin

$$\frac{1}{||\mathbf{w}||^2}$$

- using Lagrange multipliers

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{m} \alpha_i \cdot \{y_i \cdot [(\mathbf{w} \cdot \mathbf{x_i}) + b] - 1\}$$

Binary classification

- extremum at

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) = 0, \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = 0$$

- i.e.

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

- and

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

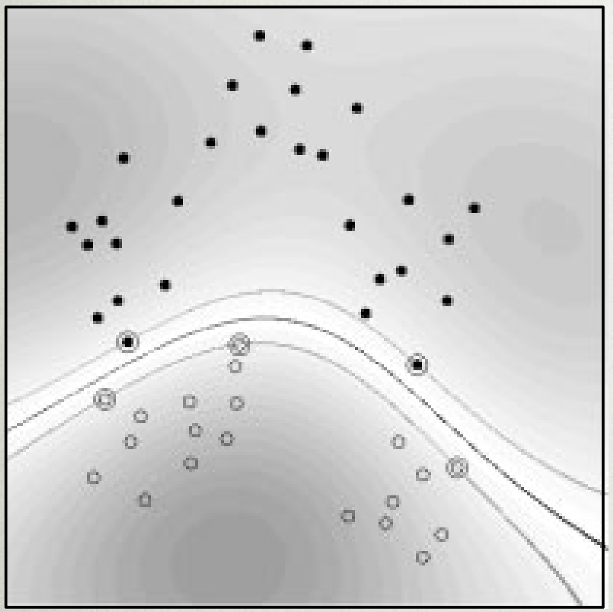
Binary classification

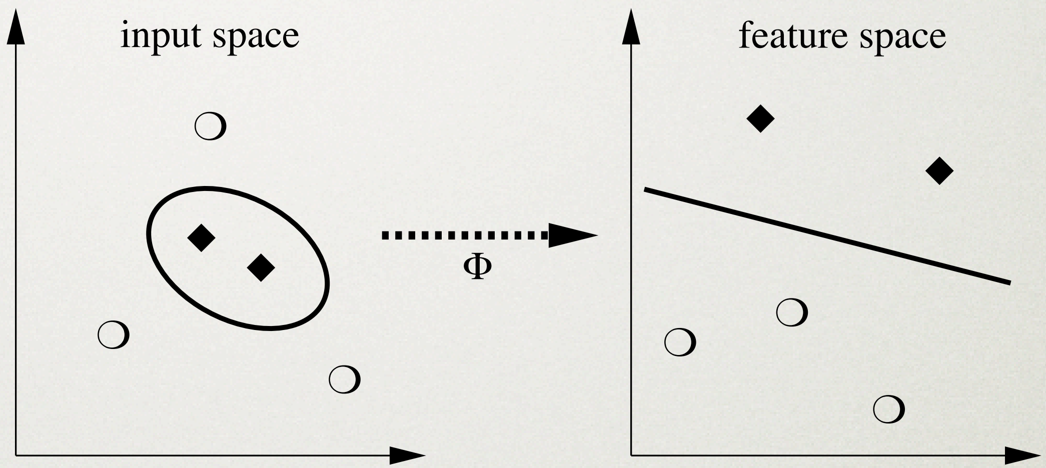
$$\begin{aligned} f(\mathbf{x}) &= \operatorname{sgn}\Big((\mathbf{x}\cdot\mathbf{w}) + b\Big) \\ &= \operatorname{sgn}\Big(\sum_{i=1}^{m} \alpha_i y_i (\mathbf{x}\cdot\mathbf{x}_i) + b\Big) \end{aligned}$$

Classification

- ~~Text as points?!~~
- ~~How to compute that line?~~
- What do we do if the line does not exist?
- What do we do if >2 answers (classes)?

SVM





- That “transformation” function can be very expensive to compute

SVM

- Kernels to the rescue

$$
\begin{aligned}
f(\mathbf{x}) &= \operatorname{sgn}\Big(\sum_{i=1}^{m} \alpha_i y_i (\mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{x}_i)) + b\Big) \\
&= \operatorname{sgn}\Big(\sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\Big)
\end{aligned}
$$

- Kernel function, e.g.,

$$K(\mathbf{x}, \mathbf{x}_i) = \exp(-||\mathbf{x} - \mathbf{x}_i||^2)$$

SVM

- Subject to constraints

$$y_i \cdot [(\mathbf{w} \cdot \mathbf{x_i}) + b] \geqslant 1 - \xi_i$$

$$\xi_i \geqslant 0, i = 1...m$$

- minimize

$$\tau(\mathbf{w}, \xi) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} \xi_i$$

SVM

- www.support-vector.net
- www.kernel-machines.org
- svmlight.joachims.org
- www.csie.ntu.edu.tw/~cjlin/bsvm/

Classification

- ~~Text as points?!~~
- ~~How to compute that line?~~
- ~~What do we do if the line does not exist?~~
- What do we do if >2 answers (classes)?

N-class Classification

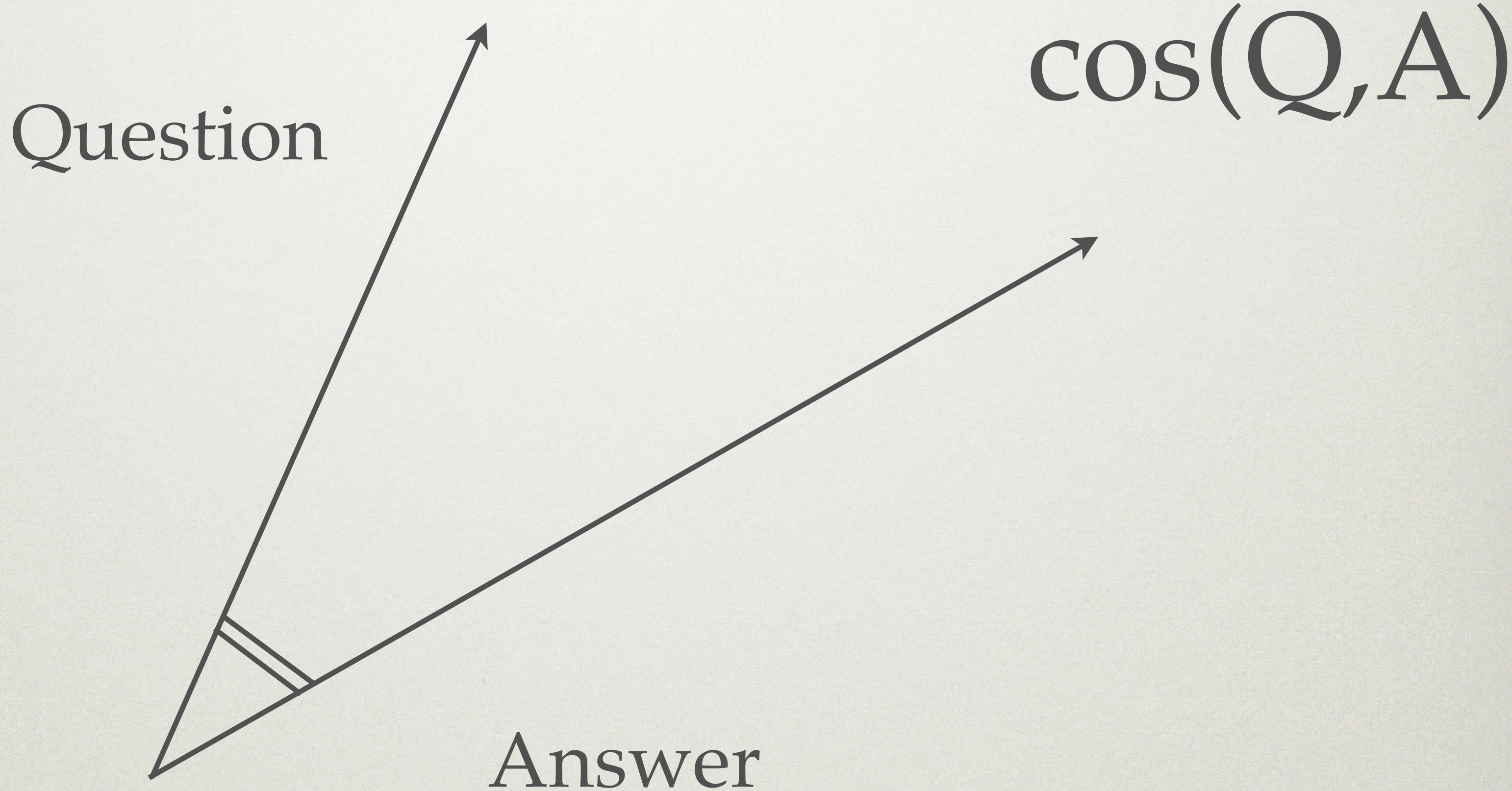
- one-against-all (N)
  - select the class with the highest f(x)
- one-against-one (N(N-1)/2)
  - voting: the class with largest number of wins

Text Retrieval

Text Retrieval

- Information Retrieval
- Answer = document
- Question = query
- match query against documents...

Text as vectors



Text Retrieval

- Compute vector for each answer
- Compute vector for the question
- Order answers by the similarity
- Select the top-ranked answer

Vectors are Bad!

- They work... But!
- no model
- ad-hoc weighting schemes
- ad-hoc similarity measure
- difficult to interpret
- impossible to explain
- unclear how to improve

$$w_{i,j} = \frac{tf_{i,j}}{tf_{i,j} + 0.5 + 1.5\frac{doclen}{avgdoclen}} \cdot \frac{\log(\frac{colsize+0.5}{docf_i})}{\log(colsize + 1)}$$

Language Model

Word generator

That	detective	is	the	right	question

Language Model

- Random process
  - M
- Defined by the text probabilities
  - $P(W \mid M) = P(w_1,...,w_N \mid M)$

probability |ˌpräbəˈbilətē| | ˈprɑbəˌbɪlədi| |prɒbəˌbɪlɪti|

noun (pl. **-ties**)
the extent to which something is probable; the likelihood of something happening or being the case : *the rain will make the probability of their arrival even greater.*
• a probable event : *for a time, revolution was a strong probability.*
• the most probable thing : *the probability is that it will be phased in over a number of years.*
• Mathematics the extent to which an event is likely to occur, measured by the ratio of the favorable cases to the whole number of cases possible : *the area under the curve represents probability | a probability of 0.5.*
PHRASES
in all probability used to convey that something is very likely : *he would in all probability make himself known.*
ORIGIN late Middle English : from Latin ***probabilitas***, from ***probabilis 'provable, credible'*** (see **probable**).

Probabilistic Matching

- Estimate language models of question M_Q and answer M_A
- Compare the models (e.g., cross entropy)
  - number of bits to "encode" M_Q with M_A

$$H(M_Q||M_A) = -\sum_w P(w|M_Q) \log P(w|M_A)$$

- Select the most similar answer
  - ... or top N best
  - ... or with entropy below a threshold

ESTIMATION

Models

- Unigram
  - $$P(W) = P(w_1 ... w_n) = \prod_{i=1}^{n} P(w_i)$$
  - word independence
  - P("did you kill") = P("you did kill")
- Higher-order models
  - n-gram: condition on preceding words
  - cache: condition on a window
  - grammar: condition of grammar structure
- Are they useful?
  - parameter estimation expensive
  - need more data

Unigram Model Revisited

- Unigram model:

$$P(w_1 ... w_n) = \prod_{i=1}^{n} P(w_i)$$

- Exchangeability instead of independence
- de Finetti's theorem

$$P(w_1 ... w_n) = \int_{\Theta} \prod_{i=1}^{n} P_{\theta}(w_i) p(d\theta)$$

- hide dependencies in the parameters

probability measure over all possible parameter settings

Unigram Model Revisited

- Estimating the generative density
  - using N training strings (e.g, answers)
- Kernel-based estimation

$$p(d\theta) = \frac{1}{N}\sum_{l=1}^{N} K_l(d\theta)$$

- Delta kernel (others exist)

$$K_{\delta,l}(d\theta) = \begin{cases} 1 & d\theta \sim P_l(w) \\ 0 & \text{otherwise} \end{cases}$$

- Can show that

$$P(w_1 \ldots w_n) = \frac{1}{N}\sum_{l=1}^{N}\prod_{i=1}^{n} P_l(w_i)$$

Unigram Model Revisited

- LM

$$P(w|w_1...w_n) = \frac{P(w, w_1...w_n)}{P(w_1...w_n)} = \frac{\sum_{l=1}^{N} P_l(w) \prod_{i=1}^{n} P_l(w_i)}{\sum_{l=1}^{N} \prod_{i=1}^{n} P_l(w_i)}$$

- A much better estimate
- Interpretation: averaged (smoothed) over the training strings

P(w) estimations

- Maximum-likelihood
- Discounting
- Interpolation

Maximum-likelihood

- relative word frequency

$$\hat{P}(w|M_W) = u_{W,ml}(w) = \frac{\#(w,W)}{|W|}$$

- unbiased
  - if we repeat estimation an infinite number of times with different starting points, we will get correct probabilities
- Zero-frequency problem

Zero Frequency Problem

- Suppose some word not in the string
  - we get zero probability for the word
  - and any string with that word
- Happens with language

Discounting

- Laplace
  - add 1 to every count, normalize
- Lindstone
  - add a constant
- Absolute discounting
- Leave-one-out discounting
- Good-Turing estimation

Interpolation

- Problem with discounting
  - treats all unseen words equally
- Use background probabilities
  - interpolate ML estimates with General English expectations

Interpolation

- Jelinek-Mercer

$$u_W(w) = \lambda \cdot u_{W,ml}(w) + (1-\lambda) \cdot u_{GE,ml}(w) = \lambda \cdot \frac{\#(w,W)}{|W|} + (1-\lambda) \cdot \frac{\#(w,GE)}{|GE|}$$

- Dirichlet

$$u_W(w) = \frac{|W|}{|W|+\mu} \cdot u_{W,ml}(w) + \frac{\mu}{|W|+\mu} \cdot u_{GE,ml}(w)$$

- Witten-Bell
- Two-stage

LM Summary

- Compute LM for each answer A
  - use unigram model
  - use Dirichlet smoothing

$$p(w|M_A) = \frac{\sum_{l=1}^{N} u_l(w) \prod_{i=1}^{n} u_l(a_i)}{\sum_{l=1}^{N} \prod_{i=1}^{n} u_l(a_i)}$$

- Compute LM for the question
- Compute cross-entropy for each pair

$$H(M_Q||M_A) = -\sum_{w} P(w|M_Q) \log P(w|M_A)$$

- Select answer with the highest value

Discussion

- That's how you do retrieval
- The assumption is that M_Q is similar to M_A
- Is it true?

Discussion

- Not really!
- Questions and answers are generated by different speakers
- Questions have specific form
- They are two different “languages”!

Discussion

- Single-language solution
  - retrieve training questions, not answers
  - individual questions
  - ... or pseudo-questions created by combining all questions appropriate to a single answer
- Cross-lingual solution
  - e.g. retrieve Chinese documents with an English query
  - view questions and answers as coming from two languages

Cross-lingual method

- Question LM is replaced by the "translated" question LM:
  - we iterate over $\{Q_l, A_l\}$

$$p(w|M_Q) = \frac{\sum_{l=1}^{N} u_{A_l}(w) \prod_{i=1}^{n} u_{Q_l}(q_i)}{\sum_{l=1}^{N} \prod_{i=1}^{n} u_{Q_l}(q_i)}$$

- Two estimation functions $u()$
  - one for questions and one for answers with their own parameters
- Interpretation
  - estimate how the answer would look like and compare that estimation to the existing answers

Text Mapping Summary

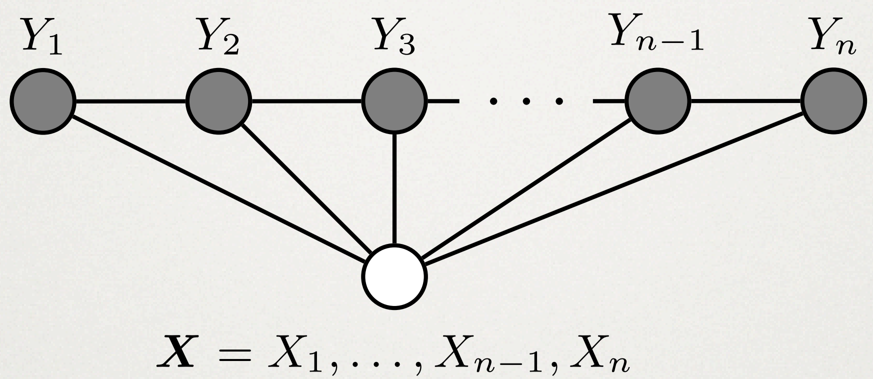
- Classification methods
  - well-defined
  - well-studied
  - require feature vectors
- Retrieval methods
  - vector-based
  - probability-based
  - estimation
  - single-language and cross-language approaches

Information Extraction

Y:	**FDC**	**FDC**	**FDC**	other	other	**FO**	**FO**	**FO**	**WO**	**WO**	K
X:	Alpha	one	six	this	is	Bravo	two	five	adjust	fire	over

- Markup important word sequences
- Maximize likelihood of observing a sequence of labels given a sequence of words: P(Y | X)

Conditional Random Fields



- CRF defines an expression for P(Y|X):

$$P(y|x) = \frac{1}{Z(x)} \exp\left\{ \sum_i \lambda_i f_i(y, x) \right\}$$

- Markov CRF: iff

$$f_i(y, x) = f_i(y_{j-1}, y_j, x, j)$$

- The CRF is determined by the parameters

CRF on Text

- Feature functions?
  - generally binary
  - word
  - word class (digit)
  - word modification (capitalization)
  - part of speech
  - presence of a feature in position j, j+1, j+2, j-1, j-2

Training CRF

- Maximizing log-likelihood

$$\mathcal{L}(\boldsymbol{\lambda}) = \sum_k \left[\log \frac{1}{Z(\boldsymbol{x}^{(k)})} + \sum_j \lambda_j F_j(\boldsymbol{y}^{(k)}, \boldsymbol{x}^{(k)}) \right]$$

- as

$$\frac{\partial \mathcal{L}(\boldsymbol{\lambda})}{\partial \lambda_j} = E_{\tilde{p}(\boldsymbol{Y},\boldsymbol{X})}\left[F_j(\boldsymbol{Y},\boldsymbol{X})\right] - \sum_k E_{p(\boldsymbol{Y}|\boldsymbol{x}^{(k)},\boldsymbol{\lambda})}\left[F_j(\boldsymbol{Y},\boldsymbol{x}^{(k)})\right]$$

- with empirical distribution over training $\tilde{p}(\boldsymbol{Y},\boldsymbol{X})$
- it might not have a closed solution

Training MCRF

- Chained CRF are much easier to train
- Beyond the scope of this lecture :-)
- see for example

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *International Conference on Machine Learning*, 2000.

Semantic Parsing

- "Why did you kill yourself" ->

speech-act **<A213>**
action *info-req*
actor *detective*
addressee *hologram*
type *question*
q-slot *cause*
time *past*
type *kill*
object *doctor*

- Translation from text to frames
- Note: Frame creation, not retrieval
- Likelihood, recall the cross-lingual technique

$$P(f|W) = \frac{\sum_s \phi_{F_s}(f) \prod_{i=1}^{m} \pi_{W_s}(w_i)}{\sum_s \prod_{i=1}^{m} \pi_{W_s}(w_i)}$$

Semantic Parsing

- Rank all slot-value pairs by the likelihood
- Cut the top part of the ranking
  - determine threshold from the training data
- That’s the frame
- How to use the frames?