Information-retrieval and classification approaches

Anton Leuski
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

- Recognize speech
- Understand language
- Manage dialogue
- Generate language
- Generate gesture
- Task & emotion reasoning
- Graphical Rendering

Flow:
- Speech
- Text
- Expectations
- Dialog Acts
- Dialog Act Plans
- Text+ Affect
- Callbacks
- Other Modes
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
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?

<table>
<thead>
<tr>
<th>Term</th>
<th>tf</th>
</tr>
</thead>
<tbody>
<tr>
<td>why</td>
<td>1</td>
</tr>
<tr>
<td>did</td>
<td>1</td>
</tr>
<tr>
<td>you</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- “Bag of words”
- Stopping
- Stemming
Text as vectors

<table>
<thead>
<tr>
<th>Why</th>
<th>did</th>
<th>you</th>
<th>kill</th>
<th>yourself</th>
</tr>
</thead>
</table>

to capture order...

<table>
<thead>
<tr>
<th>Why did</th>
<th>did you</th>
<th>you kill</th>
<th>kill yourself</th>
</tr>
</thead>
<tbody>
<tr>
<td>Why did you</td>
<td>did you kill</td>
<td>you kill yourself</td>
<td></td>
</tr>
</tbody>
</table>
Term Weights

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

\[ w_{i,j} = t f_{i,j} \]

\[ w_{i,j} = t f_{i,j} / d f_i \]

\[ w_{i,j} = t f_{i,j} / \log d f_i \]

\[ w_{i,j} = \frac{t f_{i,j}}{t f_{i,j} + 0.5 + 1.5 \frac{d o c l e n}{a v g d o c l e n}} \cdot \frac{\log(\frac{c o l s i z e + 0.5}{d o c f_i})}{\log(\text{colsize} + 1)} \]

<table>
<thead>
<tr>
<th>Term</th>
<th>tf</th>
<th>df</th>
</tr>
</thead>
<tbody>
<tr>
<td>why</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>did</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>you</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

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

\[
\{x \mid (w \cdot x) + b = 0\}
\]

\[
\{x \mid (w \cdot x) + b = -1\}
\]

\[
\{x \mid (w \cdot x) + b = +1\}
\]

\[
\{x \mid (w \cdot x) + b = 1\}
\]

\[
\{x \mid (w \cdot x) + b = +1\}
\]

\[
\{x \mid (w \cdot x) + b = -1\}
\]

Note:

\[
(w \cdot x_1) + b = +1
\]

\[
(w \cdot x_2) + b = -1
\]

\[
\Rightarrow (w \cdot (x_1 - x_2)) = 2
\]

\[
\Rightarrow \left( \frac{w}{\|w\|} \cdot (x_1 - x_2) \right) = \frac{2}{\|w\|}
\]

**dot product**

**margin width**
Binary classification

- subject to constraints
  \[ y_i \cdot [(w \cdot x_i) + b] \geq 1, \ i = 1 \ldots m \]
- maximize margin
  \[ \frac{1}{||w||^2} \]
- using Lagrange multipliers
  \[ L(w, b, \alpha) = \frac{1}{2} ||w||^2 - \sum_{i=1}^{m} \alpha_i \cdot \{y_i \cdot [(w \cdot x_i) + b] - 1\} \]
**Binary classification**

- extremum at

\[
\frac{\partial}{\partial b} L(w, b, \alpha) = 0, \quad \frac{\partial}{\partial w} L(w, b, \alpha) = 0
\]

- i.e.

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

- and

\[
w = \sum_{i=1}^{m} \alpha_i y_i x_i
\]
Binary classification

\[ f(x) = \text{sgn} \left( (x \cdot w) + b \right) \]

\[ = \text{sgn} \left( \sum_{i=1}^{m} \alpha_i y_i (x \cdot x_i) + b \right) \]
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)?
That “transformation” function can be very expensive to compute
• Kernels to the rescue

\[
f(x) = \text{sgn} \left( \sum_{i=1}^{m} \alpha_i y_i (\Phi(x) \cdot \Phi(x_i)) + b \right)
\]

\[
= \text{sgn} \left( \sum_{i=1}^{m} \alpha_i y_i K(x, x_i) + b \right)
\]

• Kernel function, e.g.,

\[
K(x, x_i) = \exp(-||x - x_i||^2)
\]
• Subject to constraints

\[ y_i \cdot [(w \cdot x_i) + b] \geq 1 - \xi_i \]
\[ \xi_i \geq 0, \ i = 1...m \]

• minimize

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

\[ \cos(Q,A) \]

Question

Answer
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

\[
    w_{i,j} = \frac{tf_{i,j}}{tf_{i,j} + 0.5 + 1.5 \frac{doclen}{avgdocen}} \cdot \frac{\log(\text{colsize} + 0.5)}{\log(\text{colsize} + 1)}
\]
That detective is the right question
**Language Model**

- Random process
  - $M$
- Defined by the text probabilities
  - $P(W | M) = P(w_1, ..., w_N | M)$
probability |ˈpræbəˈbiləti| |ˈpræbəˌbɪli| |ˈpræbəˌbɪli| 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

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

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

- 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 \( \{Q_l, A_l\} \)

\[
p(w|M_Q) = \frac{\sum_{i=1}^{N} u_{A_i}(w) \prod_{i=1}^{n} u_{Q_i}(q_i)}{\sum_{i=1}^{N} \prod_{i=1}^{n} u_{Q_i}(q_i)}
\]

- **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: $P(Y \mid X)$
Conditional Random Fields

\[ X = X_1, \ldots, X_{n-1}, X_n \]

• CRF defines an expression for \( P(Y \mid X) \):

\[
P(y \mid x) = \frac{1}{Z(x)} \exp \left\{ \sum_i \lambda_i f_i(y, x) \right\}
\]

• 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

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

- as

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

- with empirical distribution over training \( \tilde{p}(Y, X) \)

- 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


Semantic Parsing

• “Why did you kill yourself” ->

• 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?