Answer Extraction

Ling573
NLP Systems and Applications
May 17, 2011
Roadmap

• Deliverable 3 Discussion
  • What worked

• Deliverable 4

• Answer extraction:
  • Learning answer patterns
  • Answer extraction: classification and ranking
  • Noisy channel approaches
Reminder

- Steve Sim
  - Career Exploration discussion
    - After class today
Deliverable #3

- Document & Passage Retrieval
- What was tried:
  - Query processing:
Deliverable #3

- Document & Passage Retrieval

- What was tried:
  - Query processing:
    - Stopwording: bigger/smaller lists
Deliverable #3

- Document & Passage Retrieval
- What was tried:
  - Query processing:
    - Stopwording: bigger/smaller lists
    - Stemming: Y/N: Krovetz/Porter/Snowball
Deliverable #3

- Document & Passage Retrieval

- What was tried:
  - Query processing:
    - Stopwording: bigger/smaller lists
    - Stemming: Y/N: Krovetz/Porter/Snowball
    - Targets: Concatenation, pronoun substitution
Deliverable #3

- Document & Passage Retrieval

- What was tried:
  - Query processing:
    - Stopwording: bigger/smaller lists
    - Stemming: Y/N: Krovetz/Porter/Snowball
    - Targets: Concatenation, pronoun substitution
    - Reformulation
Deliverable #3

- Document & Passage Retrieval

- What was tried:
  - Query processing:
    - Stopwording: bigger/smaller lists
    - Stemming: Y/N: Krovetz/Porter/Snowball
    - Targets: Concatenation, pronoun substitution
  - Reformulation
  - Query expansion:
    - WordNet synonym expansion
    - Pseudo-relevance feedback
    - Slight differences
Deliverable #3

- What worked:
  - Query processing:
    - Stopwording: bigger/smaller lists
    - Both - apparently
Deliverable #3

- What worked:
  - Query processing:
    - Stopwording: bigger/smaller lists
    - Both - apparently
    - Stemming: Y/N: Krovetz/Porter/Snowball
Deliverable #3

- What worked:
  - Query processing:
    - Stopwording: bigger/smaller lists
    - Both - apparently
    - Stemming: Y/N: Krovetz/Porter/Snowball
    - Targets: Concatenation, pronoun substitution: Both good
Deliverable #3

- What worked:
  - Query processing:
    - Stopwording: bigger/smaller lists
      - Both - apparently
    - Stemming: Y/N: Krovetz/Porter/Snowball
    - Targets: Concatenation, pronoun substitution: Both good
    - Reformulation: little effect
Deliverable #3

• What worked:
  • Query processing:
    • Stopwording: bigger/smaller lists
      • Both - apparently
    • Stemming: Y/N: Krovetz/Porter/Snowball
    • Targets: Concatenation, pronoun substitution: Both good
    • Reformulation: little effect
  • Query expansion:
    • Generally degraded results
    • One group had some improvement in MRR
Deliverable #3

- What was tried:
  - Indexing:
    - Lucene & Indri
Deliverable #3

- What was tried:
  - Indexing:
    - Lucene & Indri
  - Passage retrieval:
    - Indri passages: different window sizes/steps
Deliverable #3

- **What was tried:**
  - **Indexing:**
    - Lucene & Indri
  - **Passage retrieval:**
    - Indri passages: different window sizes/steps
    - Passage reranking:
      - Multitext
      - SiteQ
      - ISI
      - Classification
Deliverable #3

- What worked:
  - Indexing:
    - Lucene & Indri: Generally comparable: 0.30-0.35 MAP
Deliverable #3

- What worked:
  - Indexing:
    - Lucene & Indri: Generally comparable: 0.30-0.35 MAP
  - Passage retrieval:
    - Indri passages: different window sizes/steps
      - Generally works well with moderate overlap: best results
Deliverable #3

- What worked:
  - Indexing:
    - Lucene & Indri: Generally comparable: 0.30-0.35 MAP
  - Passage retrieval:
    - Indri passages: different window sizes/steps
      - Generally works well with moderate overlap: best results
  - Passage reranking:
    - Multitext
    - SiteQ
    - ISI
    - Classification: hard to beat Indri
Deliverable #4

- End-to-end QA: Answer extraction/refinement
  - Jeopardy!-style answers: beyond scope
  - Instead:
    - More fine-grained passages
Deliverable #4

- End-to-end QA: Answer extraction/refinement
  - Jeopardy!-style answers: beyond scope
  - Instead:
    - More fine-grained passages

- Specifically, passages of:
  - 100-char,
  - 250-char, and
  - 1000-char
Deliverable #4

- End-to-end QA: Answer extraction/refinement
  - Jeopardy!-style answers: beyond scope
  - Instead:
    - More fine-grained passages

- Specifically, passages of:
  - 100-char,
  - 250-char, and
  - 1000-char

- Evaluated on QA 2004, 2005 (held out) data
Deliverable #4

- Output Format:
  - Factoids only please

- Top 20 results

- Output lines:
  - Qid run-tag DocID Answer_string

- Answer string: different lengths
  - Please no carriage returns....
Answer Extraction

- Goal:
  - Given a passage, find the specific answer in passage
  - Go from ~1000 chars -> short answer span
Answer Extraction

• Goal:
  • Given a passage, find the specific answer in passage
  • Go from ~1000 chars -> short answer span

• Example:
  • Q: What is the current population of the United States?
  • Pass: The United States enters 2011 with a population of more than 310.5 million people, according to a U.S. Census Bureau estimate.
Answer Extraction

- Goal:
  - Given a passage, find the specific answer in passage
  - Go from ~1000 chars -> short answer span

- Example:
  - Q: What is the current population of the United States?
  - Pass: The United States enters 2011 with a population of more than 310.5 million people, according to a U.S. Census Bureau estimate.
  - Answer: 310.5 million
Challenges

- ISI’s answer extraction experiment:
  - Given:
    - Question: 413 TREC-2002 factoid questions
    - Known answer type
    - All correct answer passages
Challenges

• ISI’s answer extraction experiment:
  • Given:
    • Question: 413 TREC-2002 factoid questions
    • Known answer type
    • All correct answer passages
  • Task: Pin-point specific answer string

• Accuracy:
  • Systems: 68.2%, 63.4%, 56.7%
    • Still missing 30%+ answers
Challenges

- ISI’s answer extraction experiment:
  - Given:
    - Question: 413 TREC-2002 factoid questions
    - Known answer type
    - All correct answer passages
  - Task: Pin-point specific answer string

- Accuracy:
  - Systems: 68.2%, 63.4%, 56.7%
    - Still missing 30%+ answers
  - Oracle (any of 3 right): 78.9% (20% miss)
Basic Strategies

- **Answer-type matching:**
  - Build patterns for answer locations
  - Restrict by answer type
Basic Strategies

- Answer-type matching:
  - Build patterns for answer locations
    - Restrict by answer type

- Information for pattern types:
Basic Strategies

• Answer-type matching:
  • Build patterns for answer locations
    • Restrict by answer type

• Information for pattern types:
  • Lexical: word patterns
  • Syntactic/structural:
    • Syntactic relations b/t question and answer
  • Semantic:
    • Semantic/argument relations b/t question and answer
Basic Strategies

- **Answer-type matching:**
  - Build patterns for answer locations
    - Restrict by answer type

- **Information for pattern types:**
  - **Lexical:** word patterns
  - **Syntactic/structural:**
    - Syntactic relations b/t question and answer
  - **Semantic:**
    - Semantic/argument relations b/t question and answer

- Combine with machine learning to select
Pattern Matching Example

- Answer type: Definition
<table>
<thead>
<tr>
<th>Pattern</th>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;\text{AP}&gt;) such as (&lt;\text{QP}&gt;)</td>
<td>What is autism?</td>
<td>“developmental disorders such as autism”</td>
</tr>
<tr>
<td>(&lt;\text{QP}&gt;), a (&lt;\text{AP}&gt;)</td>
<td>What is a caldera?</td>
<td>“the Long Valley caldera, a \text{volcanic crater} 19 miles long”</td>
</tr>
</tbody>
</table>
## Pattern Matching Example

- **Answer type:** Definition

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;AP&gt;) such as (&lt;QP&gt;)</td>
<td>What is autism?</td>
<td>“, developmental disorders such as autism”</td>
</tr>
<tr>
<td>(&lt;QP&gt;, a &lt;AP&gt;)</td>
<td>What is a caldera?</td>
<td>“the Long Valley caldera, a volcanic crater 19 miles long”</td>
</tr>
</tbody>
</table>

- **Answer type:** Birthdate
### Pattern Matching Example

#### Answer type: Definition

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;AP&gt;) such as (&lt;QP&gt;)</td>
<td>What is autism?</td>
<td>“, developmental disorders such as autism”</td>
</tr>
<tr>
<td>(&lt;QP&gt;), a (&lt;AP&gt;)</td>
<td>What is a caldera?</td>
<td>“the Long Valley caldera, a \textit{volcanic crater} 19 miles long”</td>
</tr>
</tbody>
</table>

#### Answer type: Birthdate

- **Question**: When was Mozart born?
- **Answer**: Mozart was born on ....
Pattern Matching Example

- Answer type: Definition

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;AP&gt; such as &lt;QP&gt;</td>
<td>What is autism?</td>
<td>“, developmental disorders such as autism”</td>
</tr>
<tr>
<td>&lt;QP&gt;, a &lt;AP&gt;</td>
<td>What is a caldera?</td>
<td>“the Long Valley caldera, a volcanic crater 19 miles long”</td>
</tr>
</tbody>
</table>

- Answer type: Birthdate
  - Question: When was Mozart born?
  - Answer: Mozart was born on ....
  - Pattern: <QP> was born on <AP>
  - Pattern: <QP> (<AP> - ......)
Basic Strategies

• N-gram tiling:
  • Typically as part of answer validation/verification
  • Integrated with web-based retrieval
  • Based on retrieval of search ‘snippets’
  • Identifies frequently occurring, overlapping n-grams
    • Of correct type
N-gram Tiling

Scores

20 Charles Dickens
15 Dickens
10 Mr Charles

merged, discard old n-grams

Score 45 Mr Charles Dickens

tile highest-scoring n-gram

Repeat, until no more overlap
Automatic Pattern Learning

- Ravichandran and Hovy 2002; Echihabi et al, 2005
- Inspiration (Soubottin and Soubottin ’01)
- Best TREC 2001 system:
Automatic Pattern Learning

- Ravichandran and Hovy 2002; Echihabi et al, 2005
- **Inspiration** (Soubottin and Soubottin ’01)
  - Best TREC 2001 system:
    - Based on extensive list of surface patterns
      - Mostly manually created
Automatic Pattern Learning

- Ravichandran and Hovy 2002; Echihabi et al, 2005

- Inspiration (Soubottin and Soubottin ’01)

- Best TREC 2001 system:
  - Based on extensive list of surface patterns
    - Mostly manually created

- Many patterns strongly associated with answer types
  - E.g. <NAME> (<DATE>-<DATE>)
    - Person’s birth and death
Pattern Learning

- S & S ‘01 worked well, but
Pattern Learning

- S & S ‘01 worked well, but
- Manual pattern creation is a hassle, impractical
Pattern Learning

- S & S ‘01 worked well, but
  - Manual pattern creation is a hassle, impractical

- Can we learn patterns?
  - Supervised approaches:
Pattern Learning

- S & S ‘01 worked well, but
  - Manual pattern creation is a hassle, impractical

- Can we learn patterns?
  - Supervised approaches:
    - Not much better,
    - Have to tag training samples, need training samples
Pattern Learning

- S & S ‘01 worked well, but
  - Manual pattern creation is a hassle, impractical

- Can we learn patterns?
  - Supervised approaches:
    - Not much better,
      - Have to tag training samples, need training samples
  - Bootstrapping approaches:
    - Promising:
Pattern Learning

• S & S ‘01 worked well, but
  • Manual pattern creation is a hassle, impractical

• Can we learn patterns?
  • Supervised approaches:
    • Not much better,
      • Have to tag training samples, need training samples
  • Bootstrapping approaches:
    • Promising:
      • Guidance from small number of seed samples
      • Can use answer data from web
Finding Candidate Patterns

- For a given question type
  - Identify an example with qterm and aterm
Finding Candidate Patterns

- For a given question type
  - Identify an example with qterm and aterm
  - Submit to a search engine
  - Download top N web docs (N=1000)
  - Select only sentences w/qterm and aterm
Finding Candidate Patterns

- For a given question type
  - Identify an example with qterm and aterm
  - Submit to a search engine
  - Download top N web docs (N=1000)
  - Select only sentences w/qterm and aterm
  - Identify all substrings and their counts
    - Implemented using suffix trees for efficiency
  - Select only phrases with qterm AND aterm
  - Replace qterm and aterm instances w/generics
Example

- Q: When was Mozart born?
- A: Mozart (1756-....)
Example

- Q: When was Mozart born?
- A: Mozart (1756 –
- Qterm: Mozart; Aterm: 1756
  - The great composer Mozart (1756–1791) achieved fame
  - Mozart (1756–1791) was a genius
  - Indebted to the great music of Mozart (1756–1791)
Example

- Q: When was Mozart born?
- A: Mozart (1756 –
- Qterm: Mozart; Aterm: 1756
  - The great composer Mozart (1756–1791) achieved fame
  - Mozart (1756–1791) was a genius
  - Indebted to the great music of Mozart (1756–1791)
- Phrase: Mozart (1756-1791); count =3
Example

- Q: When was Mozart born?
- A: Mozart (1756 –

Qterm: Mozart; Aterm: 1756
- The great composer Mozart (1756–1791) achieved fame
- Mozart (1756–1791) was a genius
- Indebted to the great music of Mozart (1756–1791)

Phrase: Mozart (1756-1791); count =3

Convert to: <Name> (<ANSWER>
Patterns

- Typically repeat with a few more examples
Patterns

- Typically repeat with a few more examples

- Collect more patterns:
  - E.g. for Birthdate
    - a. born in <ANSWER>, <NAME>
    - b. <NAME> was born on <ANSWER>,
    - c. <NAME> ( <ANSWER> -
    - d. <NAME> ( <ANSWER> - )

- Is this enough?
Patterns

- Typically repeat with a few more examples

- Collect more patterns:
  - E.g. for Birthdate
    - a. born in <ANSWER>, <NAME>
    - b. <NAME> was born on <ANSWER>,
    - c. <NAME> ( <ANSWER> -
    - d. <NAME> ( <ANSWER> - )

- Is this enough?
  - No – some good patterns, but
    - Probably lots of junk, too; need to filter
Computing Pattern Precision

- For question type:
  - Search only on qterm
Computing Pattern Precision

- For question type:
  - Search only on qterm
  - Download top N web docs (N=1000)
  - Select only sentences w/qterm
Computing Pattern Precision

- For question type:
  - Search only on qterm
  - Download top N web docs (N=1000)
  - Select only sentences w/qterm
  - For each pattern, check if
    - a) matches w/any aterm; $C_o$
    - b) matches w/ right aterm: $C_a$
Computing Pattern Precision

- For question type:
  - Search only on qterm
  - Download top N web docs (N=1000)
  - Select only sentences w/qterm
  - For each pattern, check if
    - a) matches w/any aterm; $C_o$
    - b) matches/w right aterm: $C_a$
  - Compute precision $P = \frac{C_a}{C_o}$
  - Retain if match > 5 examples
Pattern Precision Example

- Qterm: Mozart
- Pattern: <NAME> was born in <ANSWER>
Pattern Precision Example

- Qterm: Mozart
- Pattern: <NAME> was born in <ANSWER>
- Near-Miss: Mozart was born in Salzburg
- Match: Mozart born in 1756.
Pattern Precision Example

- Qterm: Mozart
- Pattern: <NAME> was born in <ANSWER>
- Near-Miss: Mozart was born in Salzburg
- Match: Mozart born in 1756.
- Precisions:
  - 1.0 <NAME> (<ANSWER> - )
  - 0.6 <NAME> was born in <ANSWER>
  - ....
Nuances

- Alternative forms:
  - Need to allow for alternate forms of question or answer
    - E.g. dates in different formats, full names, etc
  - Use alternate forms in pattern search
Nuances

- Alternative forms:
  - Need to allow for alternate forms of question or answer
    - E.g. dates in different formats, full names, etc
  - Use alternate forms in pattern search

- Precision assessment:
  - Use other examples of same type to compute
  - Cross-checks patterns
Answer Selection by Pattern

- Identify question types and terms
- Filter retrieved passages, replace qterm by tag
- Try to match patterns and answer spans
- Discard duplicates and sort by pattern precision
Pattern Sets

- WHY-FAMOUS
  1.0 <NAME> called <NAME> laureate by the, <NAME>, <NAME> - the <NAME> of <NAME> was the <NAME> of

- BIRTHYEAR
  1.0 <NAME> ( <ANSWER> - )
  0.85 <NAME> was born on <ANSWER>, <ANSWER>, <ANSWER>, <NAME> was born in <ANSWER>, <ANSWER>, <NAME> was born <ANSWER> <ANSWER> <NAME> was born
Results

- Improves, though better with web data

### TREC Corpus

<table>
<thead>
<tr>
<th>Question type</th>
<th>Number of questions</th>
<th>MRR on TREC docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIRTHYEAR</td>
<td>8</td>
<td>0.48</td>
</tr>
<tr>
<td>INVENTOR</td>
<td>6</td>
<td>0.17</td>
</tr>
<tr>
<td>DISCOVERER</td>
<td>4</td>
<td>0.13</td>
</tr>
<tr>
<td>DEFINITION</td>
<td>102</td>
<td>0.34</td>
</tr>
<tr>
<td>WHY-FAMOUS</td>
<td>3</td>
<td>0.33</td>
</tr>
<tr>
<td>LOCATION</td>
<td>16</td>
<td>0.75</td>
</tr>
</tbody>
</table>

### Web

<table>
<thead>
<tr>
<th>Question type</th>
<th>Number of questions</th>
<th>MRR on the Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIRTHYEAR</td>
<td>8</td>
<td>0.69</td>
</tr>
<tr>
<td>INVENTOR</td>
<td>6</td>
<td>0.58</td>
</tr>
<tr>
<td>DISCOVERER</td>
<td>4</td>
<td>0.88</td>
</tr>
<tr>
<td>DEFINITION</td>
<td>102</td>
<td>0.39</td>
</tr>
<tr>
<td>WHY-FAMOUS</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>LOCATION</td>
<td>16</td>
<td>0.86</td>
</tr>
</tbody>
</table>
Limitations & Extensions

- Where are the Rockies?
- ..with the Rockies in the background
Limitations & Extensions

- Where are the Rockies?
- ..with the Rockies in **the background**
- Should restrict to semantic / NE type
Limitations & Extensions

- Where are the Rockies?
- ..with the Rockies in the background

- Should restrict to semantic / NE type
  - London, which....., lies on the River Thames
  - <QTERM> word* lies on <ANSWER>
    - Wildcards impractical
Limitations & Extensions

- Where are the Rockies?
- ..with the Rockies in the background
- Should restrict to semantic / NE type
  - London, which....., lies on the River Thames
  - <QTERM> word* lies on <ANSWER>
    - Wildcards impractical
- Long-distance dependencies not practical
Limitations & Extensions

• Where are the Rockies?
• ..with the Rockies in the background

• Should restrict to semantic / NE type
• London, which....., lies on the River Thames
• <QTERM> word* lies on <ANSWER>
  • Wildcards impractical

• Long-distance dependencies not practical
• Less of an issue in Web search
  • Web highly redundant, many local dependencies
  • Many systems (LCC) use web to validate answers
### Limitations & Extensions

- When was LBJ born?
- Tower lost to Sen. LBJ, *who ran for both the*...
Limitations & Extensions

- When was LBJ born?
- Tower lost to Sen. LBJ, *who ran for both the...*

- Requires information about:
  - Answer length, type; logical distance (1-2 chunks)
Limitations & Extensions

- When was LBJ born?
- Tower lost to Sen. LBJ, *who ran for both the...*

- Requires information about:
  - Answer length, type; logical distance (1-2 chunks)

- Also,
  - Can only handle single continuous qterms
  - Ignores case
  - Needs handle canonicalization, e.g. of names/dates
Integrating Patterns II

- Fundamental problem:
Integrating Patterns II

- Fundamental problem:
  - What is there’s no pattern??
Integrating Patterns II

- Fundamental problem:
  - What is there’s no pattern??
    - No pattern -> No answer!!!

- More robust solution:
  - Not JUST patterns
Integrating Patterns II

- Fundamental problem:
  - What is there's no pattern??
    - No pattern -> No answer!!

- More robust solution:
  - Not JUST patterns
  - Integrate with machine learning
    - MAXENT!!
    - Re-ranking approach
Answering w/ Maxent

\[ P(a \mid \{a_1, a_2, \ldots a_A\}, q) = \frac{\exp[\sum_{m=1}^{M} \lambda_m f_m(a \mid \{a_1, a_2, \ldots a_A\}, q)]}{\sum_{a'} \exp[\sum_{m=1}^{M} \lambda_m f_m(a' \mid \{a_1, a_2, \ldots a_A\}, q)]} \]

\[ \hat{a} = \arg\max_a \sum_{m=1}^{M} \lambda_m f_m(a \mid \{a_1, a_2, \ldots a_A\}, q) \]
Feature Functions

- Pattern fired:
  - Binary feature
Feature Functions

- Pattern fired:
  - Binary feature

- Answer frequency/Redundancy factor:
  - # times answer appears in retrieval results
Feature Functions

- Pattern fired:
  - Binary feature

- Answer frequency/Redundancy factor:
  - # times answer appears in retrieval results

- Answer type match (binary)
Feature Functions

- Pattern fired:
  - Binary feature

- Answer frequency/Redundancy factor:
  - # times answer appears in retrieval results

- Answer type match (binary)

- Question word absent (binary):
  - No question words in answer span
Feature Functions

- Pattern fired:
  - Binary feature

- Answer frequency/Redundancy factor:
  - # times answer appears in retrieval results

- Answer type match (binary)

- Question word absent (binary):
  - No question words in answer span

- Word match:
  - Sum of ITF of words matching b/t questions & sent
Training & Testing

- Trained on NIST QA questions
  - Train: TREC 8, 9;
  - Cross-validation: TREC-10

- 5000 candidate answers/question

- Positive examples:
  - NIST pattern matches

- Negative examples:
  - NIST pattern doesn’t match

- Test: TREC-2003: MRR: 28.6%; 35.6% exact top 5
Noisy Channel QA

- Employed for speech, POS tagging, MT, summ, etc
- Intuition:
  - Question is a noisy representation of the answer
Noisy Channel QA

- Employed for speech, POS tagging, MT, summ, etc

- Intuition:
  - Question is a noisy representation of the answer

- Basic approach:
  - Given a corpus of \((Q,S_A)\) pairs
  - Train \(P(Q|S_A)\)
  - Find sentence with answer as
    - \(S_{i,Aij}\) that maximize \(P(Q|S_{i,Aij})\)
QA Noisy Channel

- A: Presley died of heart disease at Graceland in 1977, and..
- Q: When did Elvis Presley die?
QA Noisy Channel

- A: Presley died of heart disease at Graceland in 1977, and..
- Q: When did Elvis Presley die?

Goal:
- Align parts of Ans parse tree to question
  - Mark candidate answers
  - Find highest probability answer
Approach

- Alignment issue:
Approach

- Alignment issue:
  - Answer sentences longer than questions
  - Minimize length gap
    - Represent answer as mix of words/syn/sem/NE units
Approach

- Alignment issue:
  - Answer sentences longer than questions
  - Minimize length gap
    - Represent answer as mix of words/syn/sem/NE units
  - Create ‘cut’ through parse tree
    - Every word –or an ancestor – in cut
    - Only one element on path from root to word
Approach

- Alignment issue:
  - Answer sentences longer than questions
  - Minimize length gap
    - Represent answer as mix of words/syn/sem/NE units
  - Create ‘cut’ through parse tree
    - Every word –or an ancestor – in cut
    - Only one element on path from root to word

Presley died of heart disease at Graceland in 1977, and...
Presley died PP PP in DATE, and...
When did Elvis Presley die?
Approach (Cont’d)

- Assign one element in cut to be ‘Answer’
- Issue: Cut STILL may not be same length as Q
Approach (Cont’d)

- Assign one element in cut to be ‘Answer’
- Issue: Cut STILL may not be same length as Q
- Solution: (typical MT)
  - Assign each element a fertility
    - 0 – delete the word; > 1: repeat word that many times
Approach (Cont’d)

- Assign one element in cut to be ‘Answer’
- Issue: Cut STILL may not be same length as Q
- Solution: (typical MT)
  - Assign each element a fertility
    - 0 – delete the word; > 1: repeat word that many times
- Replace A words with Q words based on alignment
- Permute result to match original Question
- Everything except cut computed with OTS MT code
Schematic

- Assume cut, answer guess all equally likely
Training Sample Generation

- Given question and answer sentences
- Parse answer sentence
- Create cut s.t.:
  - Words in both Q & A are preserved
  - Answer reduced to ‘A_’ syn/sem class label
  - Nodes with no surface children reduced to syn class
  - Keep surface form of all other nodes
- 20K TREC QA pairs; 6.5K web question pairs
Selecting Answers

- For any candidate answer sentence:
  - Do same cut process
Selecting Answers

- For any candidate answer sentence:
  - Do same cut process
  - Generate all candidate answer nodes:
    - Syntactic/Semantic nodes in tree
Selecting Answers

- For any candidate answer sentence:
  - Do same cut process
  - Generate all candidate answer nodes:
    - Syntactic/Semantic nodes in tree
  - What’s a bad candidate answer?
Selecting Answers

- For any candidate answer sentence:
  - Do same cut process
  - Generate all candidate answer nodes:
    - Syntactic/Semantic nodes in tree
  - What’s a bad candidate answer?
    - Stopwords
    - Question words!
  - Create cuts with each answer candidate annotated
  - Select one with highest probability by model
Example Answer Cuts

- Q: When did Elvis Presley die?
- $S_{A1}$: Presley died $A_{PP}$ PP PP, and ...
- $S_{A2}$: Presley died PP $A_{PP}$ PP, and ....
- $S_{A3}$: Presley died PP PP in $A_{DATE}$, and ...

- Results: MRR: 24.8%; 31.2% in top 5
Error Analysis

- Component specific errors:
  - Patterns:
    - Some question types work better with patterns
    - Typically specific NE categories (NAM, LOC, ORG..)
    - Bad if ‘vague’
  - Stats based:
    - No restrictions on answer type – frequently ‘it’
  - Patterns and stats:
    - ‘Blatant’ errors:
      - Select ‘bad’ strings (esp. pronouns) if fit position/pattern
Error Analysis

- Component specific errors:
  - Patterns:
    - Some question types work better with patterns
    - Typically specific NE categories (NAM, LOC, ORG..)
    - Bad if ‘vague’
Error Analysis

- Component specific errors:
  - Patterns:
    - Some question types work better with patterns
    - Typically specific NE categories (NAM, LOC, ORG..)
    - Bad if ‘vague’
  - Stats based:
    - No restrictions on answer type – frequently ‘it’
Error Analysis

- Component specific errors:
  - Patterns:
    - Some question types work better with patterns
    - Typically specific NE categories (NAM, LOC, ORG..)
    - Bad if ‘vague’
  - Stats based:
    - No restrictions on answer type – frequently ‘it’
  - Patterns and stats:
    - ‘Blatant’ errors:
      - Select ‘bad’ strings (esp. pronouns) if fit position/pattern
Combining Units

- Linear sum of weights?
Combining Units

- Linear sum of weights?
  - Problematic:
    - Misses different strengths/weaknesses
Combining Units

- Linear sum of weights?
  - Problematic:
    - Misses different strengths/weaknesses

- Learning! (of course)
  - Maxent re-ranking
    - Linear
Feature Functions

- 48 in total
- Component-specific:
  - Scores, ranks from different modules
  - Patterns. Stats, IR, even QA word overlap
Feature Functions

- 48 in total

- Component-specific:
  - Scores, ranks from different modules
    - Patterns. Stats, IR, even QA word overlap

- Redundancy-specific:
  - # times candidate answer appears (log, sqrt)
Feature Functions

- 48 in total

- Component-specific:
  - Scores, ranks from different modules
    - Patterns, Stats, IR, even QA word overlap

- Redundancy-specific:
  - # times candidate answer appears (log, sqrt)

- Qtype-specific:
  - Some components better for certain types: type+mod
Feature Functions

- 48 in total

- Component-specific:
  - Scores, ranks from different modules
    - Patterns, Stats, IR, even QA word overlap

- Redundancy-specific:
  - # times candidate answer appears (log, sqrt)

- Qtype-specific:
  - Some components better for certain types: type+mod

- Blatant ‘errors’: no pronouns, when NOT DoW
Experiments

- Per-module reranking:
  - Use redundancy, qtype, blatant, and feature from mod
Experiments

- Per-module reranking:
  - Use redundancy, qtype, blatant, and feature from mod

- Combined reranking:
  - All features (after feature selection to 31)
Experiments

- **Per-module reranking:**
  - Use redundancy, qtype, blatant, and feature from mod

- **Combined reranking:**
  - All features (after feature selection to 31)

- **Patterns:** Exact in top 5: 35.6% -> 43.1%
- **Stats:** Exact in top 5: 31.2% -> 41%
- **Manual/knowledge based:** 57%