Answer Extraction

NLP Systems and Applications
Ling573
May 13, 2014

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
  - Task: Pin-point specific answer string

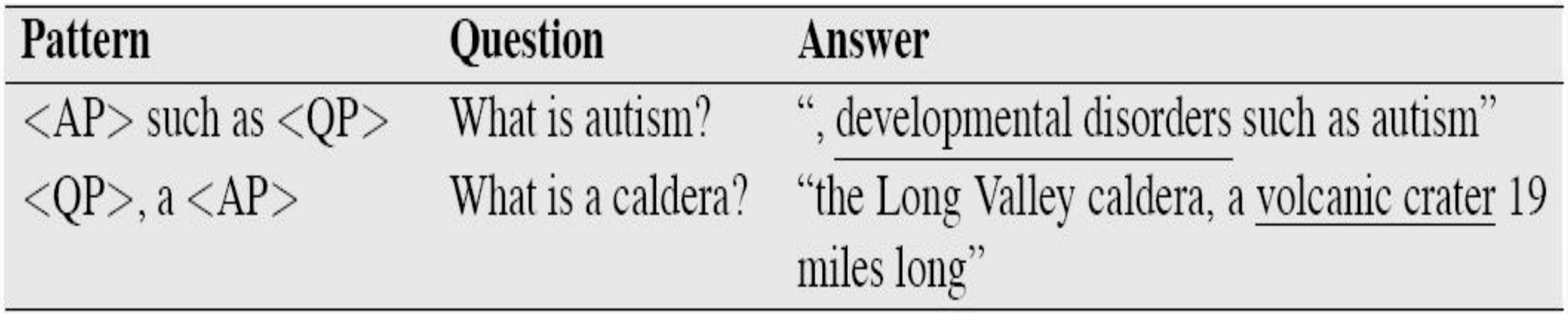
  - Accuracy:
    - Systems: 68.2%, 63.4%, 56.7%
      - Still missing 30%+ answers
    - Oracle (any of 3 right): 78.9% (20% miss)

Basic Strategies

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



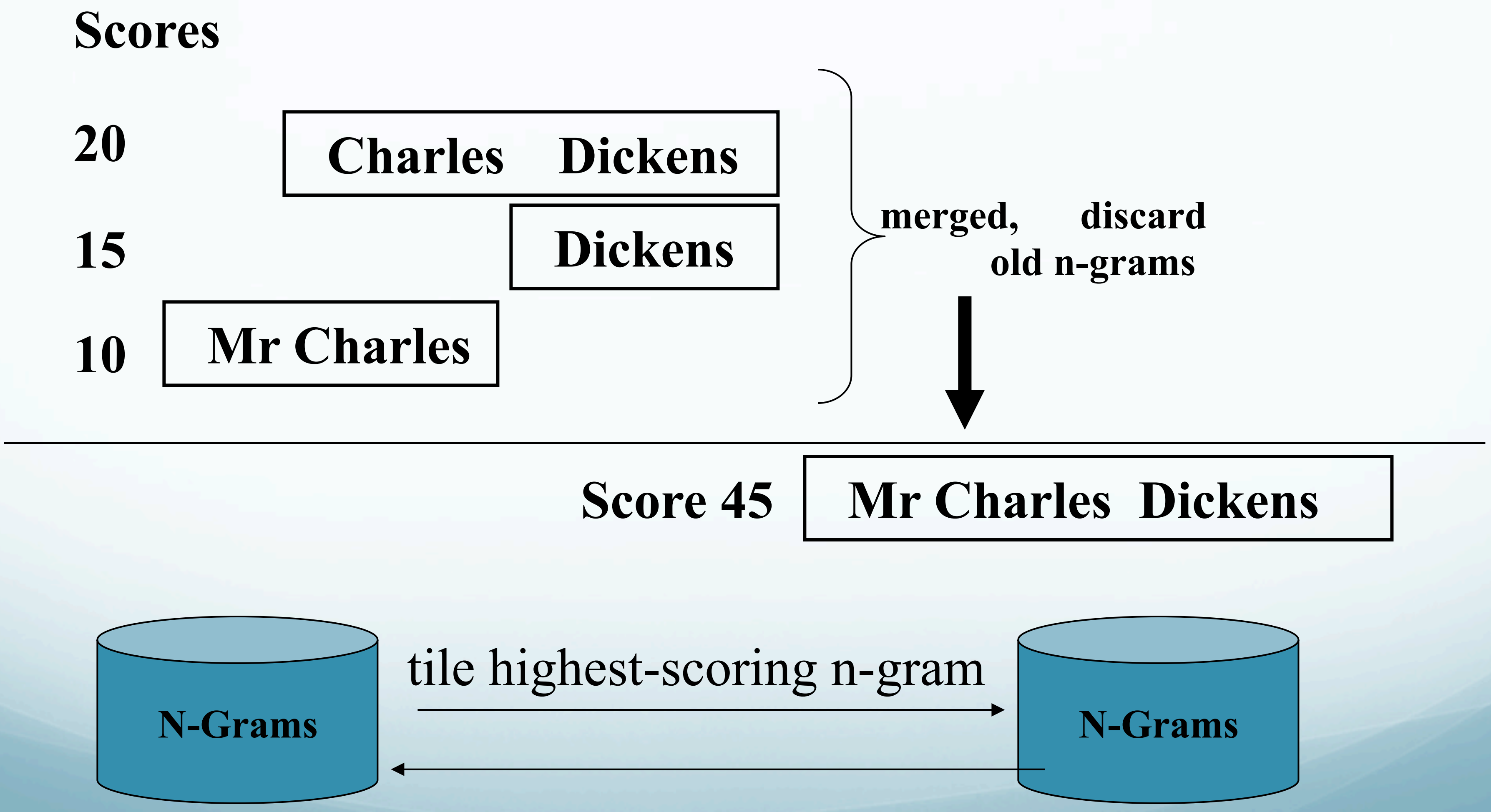
Pattern	Question	Answer
<AP> such as <QP>	What is autism?	“, developmental disorders such as autism”
<QP>, a <AP>	What is a caldera?	“the Long Valley caldera, a volcanic crater 19 miles long”

- 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



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:
    - 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
  - 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
  - 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 –
- 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
- 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
  - 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 $P = C_a/C_o$
  - Retain if match > 5 examples

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
- 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 <ANSWER> <NAME> called

  1.0 laureate <ANSWER> <NAME>

  1.0 by the <ANSWER> , <NAME> ,
  1.0 <NAME> - the <ANSWER> of

  1.0 <NAME> was the <ANSWER> of

- BIRTHYEAR

  1.0 <NAME> (<ANSWER> -)

  0.85 <NAME> was born on <ANSWER> ,

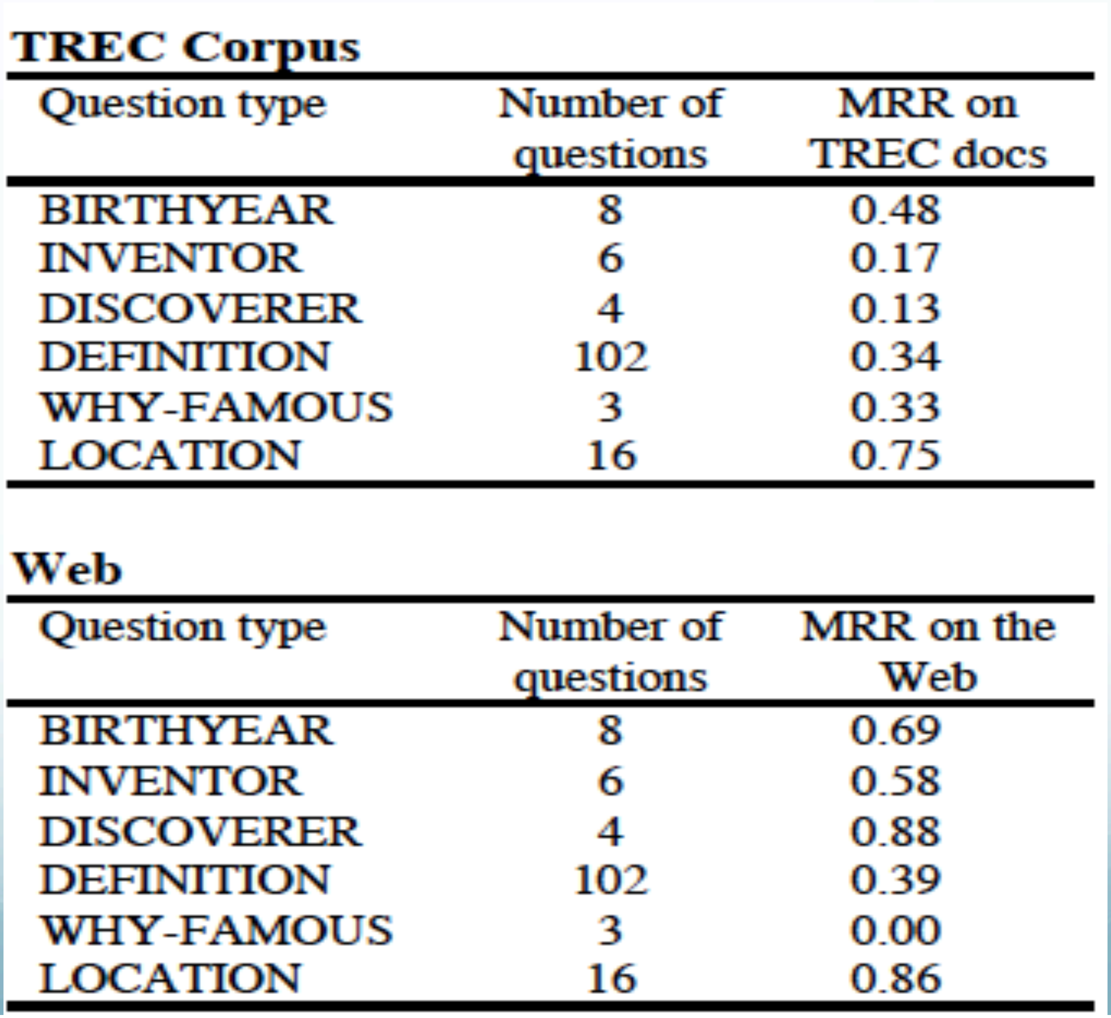
  0.6 <NAME> was born in <ANSWER>

  0.59 <NAME> was born <ANSWER>

  0.53 <ANSWER> <NAME> was born

Results

- Improves, though better with web data



TREC Corpus

Question type	Number of questions	MRR on TREC docs
BIRTHYEAR	8	0.48
INVENTOR	6	0.17
DISCOVERER	4	0.13
DEFINITION	102	0.34
WHY-FAMOUS	3	0.33
LOCATION	16	0.75

Web

Question type	Number of questions	MRR on the Web
BIRTHYEAR	8	0.69
INVENTOR	6	0.58
DISCOVERER	4	0.88
DEFINITION	102	0.39
WHY-FAMOUS	3	0.00
LOCATION	16	0.86

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*…

- 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:
  - What if 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

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

$$\widehat{a} = \underset{a}{\operatorname{argmax}}[\sum_{m=1}^{M} \lambda_m f_m(a, \{a_1, a_2, \ldots a_A\}, q)]$$

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
- 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?

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

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..

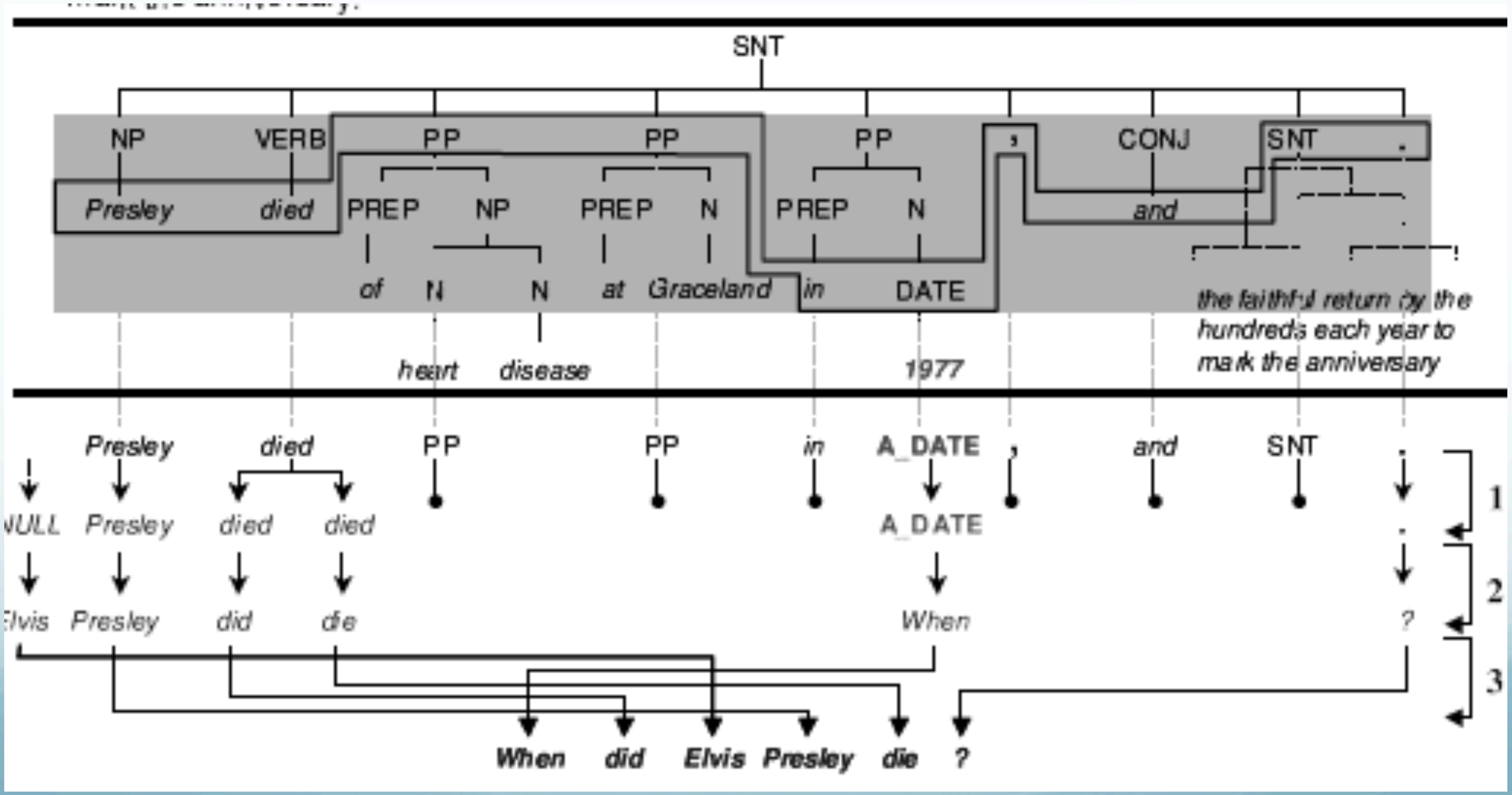
When did Elvis Presley die?

Approach (Cont’d)

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

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
- 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

- 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%
- Combined: 57%+