Question Answering from text

- An idea originating from the IR community
- With massive collections of full-text documents, simply finding relevant documents is of limited use: we want answers from textbases
- QA: give the user a (short) answer to their question, perhaps supported by evidence.
- The common person’s view? [From a novel]
  - “I like the Internet. Really, I do. Any time I need a piece of shareware or I want to find out the weather in Bogota... I’m the first guy to get the modem humming. But as a source of information, it sucks. You get a billion pieces of data, struggling to be heard and seen and downloaded, and anything I want to know seems to get trampled underfoot in the crowd.”

People want to ask questions...

Examples from AltaVista query log
who invented surf music?
how to make stink bombs
where are the snowdens of yesteryear?
which english translation of the bible is used in official catholic liturgies?
how to do clayart
how to copy px
how tall is the sears tower?
Examples from Excite query log (12/1999)
how can i find someone in texas
where can i find information on puritan religion?
what are the 7 wonders of the world
how can i eliminate stress
What vacuum cleaner does Consumers Guide recommend
Around 12–15% of query logs

The Google answer #1

- Include question words etc. in your stop-list
- Do standard IR
- Sometimes this (sort of) works:
  - Question: *Who was the prime minister of Australia during the Great Depression?*
  - Answer: *James Scullin (Labor) 1929–31.*

But often it doesn’t...

- Question: *How much money did IBM spend on advertising in 2002?*
- Answer: *I dunno, but I’d like to ... 😃*
The Google answer #2

- Take the question and try to find it as a string on the web
- Return the next sentence on that web page as the answer
- Works brilliantly if this exact question appears as a FAQ question, etc.
- Works lousy most of the time
- Reminiscent of the line about monkeys and typewriters producing Shakespeare
- But a slightly more sophisticated version of this approach has been revived in recent years with considerable success…

A Brief (Academic) History

- In some sense question answering is not a new research area
- Question answering systems can be found in many areas of NLP research, including:
  - Natural language database systems
  - A lot of early NLP work on these
  - Spoken dialog systems
  - Currently very active and commercially relevant

- The focus on open-domain QA is new
  - MURAX (Kupiec 1993): Encyclopedia answers
  - Hirschman: Reading comprehension tests
  - TREC QA competition: 1999–

Online QA Examples

- Examples
  - AnswerBus is an open-domain question answering system: www.answerbus.com
  - Ionaut: http://www.ionaut.com:8400/
  - LCC: http://www.languagecomputer.com/
  - EasyAsk, AnswerLogic, AnswerFriend, Start, Quasm, Mulder, Webclopedia, etc.

AskJeeves

- AskJeeves is probably most hyped example of “Question answering”
- It largely does pattern matching to match your question to their own knowledge base of questions
- If that works, you get the human-curated answers to that known question
- If that fails, it falls back to regular web search
- A potentially interested middle ground, but a fairly weak shadow of real QA

Question Answering at TREC

- Question answering competition at TREC consists of answering a set of 500 fact-based questions, e.g., “When was Mozart born?”
- For the first three years systems were allowed to return 5 ranked answer snippets (50/250 bytes) to each question.
  - IR think
    - Mean Reciprocal Rank (MRR) scoring:
      1, 0.5, 0.33, 0.25, 0.2, 0 for 1, 2, 3, 4, 5, 6+ doc
    - Mainly Named Entity answers (person, place, date, …)
- From 2002 the systems are only allowed to return a single exact answer and the notion of confidence has been introduced.
The TREC Document Collection

- The current collection uses news articles from the following sources:
  - AP newswire, 1998-2000
  - Xinhua News Agency newswire, 1996-2000
- In total there are 1,033,461 documents in the collection. 3GB of text
- Clearly this is too much text to process entirely using advanced NLP techniques so the systems usually consist of an initial information retrieval phase followed by more advanced processing.
- Many supplement this text with use of the web, and other knowledge bases

Sample TREC questions

2. What was the monetary value of the Nobel Peace Prize in 1989?
3. What does the Peugeot company manufacture?
4. How much did Mercury spend on advertising in 1993?
5. What is the name of the managing director of Apricot Computer?
6. Why did David Koresh ask the FBI for a word processor?
7. What debts did Qintex group leave?
8. What is the name of the rare neurological disease with symptoms such as: involuntary movements (tics), swearing, and incoherent vocalizations (grunts, shouts, etc.)?

Top Performing Systems

- Currently the best performing systems at TREC can answer approximately 70% of the questions
- Approaches and successes have varied a fair deal
- Knowledge-rich approaches, using a vast array of NLP techniques stole the show in 2000, 2001
- Notably Harabagui, Moldovan et al. – SMU/UTD/LCC
- AskMSR system stressed how much could be achieved by very simple methods with enough text (and now various copycats)
- Middle ground is to use large collection of surface matching patterns (B4)

AskMSR

- Web Question Answering: Is More Always Better?
  - Dumais, Banko, Brill, Lin, Ng (Microsoft, MIT, Berkeley)
  - Q: “Where is the Louvre located?”
  - Want “Paris” or “France” or “75058
    Paris Cedex 01” or a map
  - Don’t just want URLs

AskMSR: Shallow approach

- In what year did Abraham Lincoln die?
- Ignore hard documents and find easy ones

AskMSR: Details
Step 1: Rewrite queries

- Intuition: The user’s question is often syntactically quite close to sentences that contain the answer
  - Where is the Louvre Museum located?
  - The Louvre Museum is located in Paris
  - Who created the character of Scrooge?
  - Charles Dickens created the character of Scrooge.

Query rewriting

- Classify question into seven categories
  - Who is/was/are/were...
  - When is/did/will/were...
  - Where is/was/are/were...

a. Category-specific transformation rules
   - “Where is” questions, move “is” to all possible locations
     - “Where is the Louvre Museum located?”
       → “The Louvre Museum located”
       → “the Louvre Museum located”
   - Expected answer “Datatype” (eg, Date, Person, Location,
     When was the French Revolution?, DATE

b. Hand-crafted classification/rewrite/datatype rules
   (Could they be automatically learned?)

Non-sense, but who cares? It’s only a few more queries to Google.

Query Rewriting - weights

- One wrinkle: Some query rewrites are more reliable than others

  Where is the Louvre Museum located?
  Weight 1
  - Lots of non-answers could come back too
  Weight 5
  - if we get a match, it’s probably right

  “the Louvre Museum is located”
  +Louvre +Museum +located

Step 2: Query search engine

- Send all rewrites to a Web search engine
- Retrieve top N answers (100?)
- For speed, rely just on search engine’s “snippets”, not the full text of the actual document

Step 3: Mining N-Grams

- Unigram, bigram, trigram, … N-gram: list of N adjacent terms in a sequence
  - Eg, “Web Question Answering: Is More Always Better”
  - Unigrams: Web, Question, Answering, Is, More, Always, Better
  - Bigrams: Web Question, Question Answering, Answering Is, Is More, More Always, Always Better

Mining N-Grams

- Simple: Enumerate all N-grams (N=1,2,3 say) in all retrieved snippets
  - Use hash table and other fancy footwork to make this efficient
  - Weight of an n-gram: occurrence count, each weighted by “reliability” (weight) of rewrite that fetched the document

Example: “Who created the character of Scrooge?”
  - Dickens - 117
  - Christmas Carol - 78
  - Charles Dickens - 75
  - Disney - 72
  - Carol Banks - 54
  - A Christmas - 41
  - Christmas Carol - 45
  - Uncle - 31
Step 4: Filtering N-Grams

- Each question type is associated with one or more "data-type filters" = regular expression
- When...
- Where...
- What ...
- Who ...

- Boost score of n-grams that do match regexp
- Lower score of n-grams that don’t match regexp
- Details omitted from paper....

Step 5: Tiling the Answers

Scores
20  Charles Dickens
15  Dickens
10  Mr Charles

merged, discard old n-grams

Score 45  Mr Charles Dickens

N-Grams
tile highest-scoring n-gram
N-Grams

Repeat, until no more overlap

Results

- Standard TREC contest test-bed:
  ~1M documents; 900 questions
- Technique doesn’t do too well (though would have placed in top 9 of ~30 participants!)
  - MRR = 0.262 (ie, right answered ranked about #4-#5)
  - Why? Because it relies on the enormity of the Web!
- Using the Web as a whole, not just TREC’s 1M documents... MRR = 0.42 (ie, on average, right answer is ranked about #2-#3)

Issues

- In many scenarios (e.g., monitoring an individuals email...) we only have a small set of documents
- Works best/only for "Trivial Pursuit"-style fact-based questions
- Limited/brittle repertoire of
  - question categories
  - answer data types/filters
  - query rewriting rules

ISI: Surface patterns approach

- Use of Characteristic Phrases
- "When was <person> born"
  - Typical answers
    - "Mozart was born in 1756."
    - "Gandhi (1869-1948)"
  - Suggests phrases like
    - "<NAME> was born in <BIRTHDATE>"
    - as Regular Expressions can help locate correct answer

Use Pattern Learning

- Example:
  - "The great composer Mozart (1756-1791) achieved fame at a young age"
  - "Mozart (1756-1791) was a genius"
  - "The whole world would always be indebted to the great music of Mozart (1756-1791)"
  - Longest matching substring for all 3 sentences is "Mozart (1756-1791)"
  - Suffix tree would extract "Mozart (1756-1791)" as an output, with score of 3
  - Reminiscent of IE pattern learning
Pattern Learning (cont.)

- Repeat with different examples of same question type
  - "Gandhi 1869", "Newton 1642", etc.
- Some patterns learned for BIRTHDATE
  a. born in <ANSWER>, <NAME>
  b. <NAME> was born on <ANSWER>,
  c. <NAME> ( <ANSWER> -
  d. <NAME> ( <ANSWER> - )

Experiments

- 6 different Q types
  - from Webclopedia QA Typology (Hovy et al., 2002a)
    - BIRTHDATE
    - LOCATION
    - INVENTOR
    - DISCOVERER
    - DEFINITION
    - WHY-FAMOUS

Experiments: pattern precision

- BIRTHDATE table:
  - 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
  - 0.50 - <NAME> ( <ANSWER>
  - 0.36 <NAME> ( <ANSWER>
- INVENTOR
  - 1.0 <ANSWER> invents <NAME>
  - 1.0 the <NAME> was invented by <ANSWER>
  - 1.0 <ANSWER> invented the <NAME> in

Experiments (cont.)

- DISCOVERER
  - 1.0 when <ANSWER> discovered <NAME>
  - 1.0 <ANSWER>'s discovery of <NAME>
  - 0.9 <NAME> was discovered by <ANSWER> in
- DEFINITION
  - 1.0 <NAME> and related <ANSWER>
  - 1.0 form of <ANSWER>, <NAME>
  - 0.94 as <NAME>, <ANSWER> and

Experiments (cont.)

- WHY-FAMOUS
  - 1.0 <ANSWER> <NAME> called
  - 1.0 laureate <ANSWER> - <NAME>
  - 0.71 <NAME> is the <ANSWER> of
- LOCATION
  - 1.0 <ANSWER>'s <NAME>
  - 1.0 regional : <ANSWER> - <NAME>
  - 0.92 near <NAME> in <ANSWER>
- Depending on question type, get high MRR (0.6–0.9), with higher results from use of Web than TREC QA collection

Shortcomings & Extensions

- Need for POS &/or semantic types
  - "Where are the Rocky Mountains?"
  - "Denver’s new airport, topped with white fiberglass cones in imitation of the Rocky Mountains in the background, continues to lie empty"
  - <NAME> in <ANSWER>
- NE tagger &/or ontology could enable system to determine "background" is not a location
Shortcomings... (cont.)

- Long distance dependencies
  - "Where is London?"
  - "London, which has one of the most busiest airports in the world, lies on the banks of the river Thames"
  - would require pattern like: 
    <QUESTION>, <any_word>, lies on <ANSWER>
- Abundance & variety of Web data helps system to find an instance of patterns w/o losing answers to long distance dependencies

Harabagiu, Moldovan et al.

Value from sophisticated NLP – Pasca and Harabagiu 2001)

- Good IR is needed: SMART paragraph retrieval
- Large taxonomy of question types and expected answer types is crucial
- Statistical parser used to parse questions and relevant text for answers, and to build KB
- Query expansion loops (morphological, lexical synonyms, and semantic relations) important
- Answer ranking by simple ML method

QA Typology from ISI (USC)

- Typology of typical Q forms—94 nodes (47 leaf nodes)
- Analyzed 17,384 questions (from answers.com)

Syntax to Logical Forms

- Syntactic analysis plus semantic => logical form
- Mapping of question and potential answer LF's to find the best match
Abductive inference

- System attempts inference to justify an answer (often following lexical chains)
- Their inference is a kind of funny middle ground between logic and pattern matching
- But quite effective: 30% improvement
- Q: When was the internal combustion engine invented?
- A: The first internal-combustion engine was built in 1867.
- invent -> create_mentally -> create -> build

Question Answering Example

- How hot does the inside of an active volcano get?
- get(Temperature, inside(volcano(active)))
- "lava fragments belched out of the mountain were as hot as 300 degrees Fahrenheit"
- fragments(lava, Temperature(degrees(300)), belched(out, mountain))
- volcano ISA mountain
- lava ISPARTOF volcano
- lava inside volcano
- fragments of lava HAVEPROPERTIESOF lava
- The needed semantic information is in WordNet definitions, and was successfully translated into a form that was used for rough 'proofs'

References

- AskMSR: Question Answering Using the Worldwide Web
  - Michele Banko, Eric Brill, Susan Dumais, Jimmy Lin
  - http://www.ai.mit.edu/people/jimmylin/publications/AskMSR/AskMSR.pdf
- Web Question Answering: Is More Always Better?
  - Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, Andrew Ng

References