Dublin City University

School of Computing

CA4009: Search Technologies

Section 6: Question Answering

Gareth Jones

October 2016

Introduction

Information retrieval systems are concerned with searching for and retrieving documents relevant to the user's information need.

The searcher is required to extract information from the retrieved documents in order to satisfy their information need.

In the case of text documents, they do this by reading, while in the case of multimedia content they do this by listening to audio or watching a video.

However, many user requests are essentially questions looking for a single fact or piece of information that can be made in a short statement.

Addressing these information needs by analysing whole retrieved items can be time consuming and highly inefficient.

Introduction

Retrieved text documents can be augmented with simple features such as highlighting of query terms present in the text to aid visual scanning for relevant content, but this still requires the user to read and interpret the text which takes time.

For question type requests, the efficiency of accessing relevant information can be improved by processing the retrieved items to concentrate on information relevant to the question.

This of course will only work for question type requests - but analysis of query logs shows that there are a lot of these!

For broad exploratory information needs, standard IR systems will be more appropriate.

Introduction

Summarization is a means of reducing the user effort in analysing a document,

e.g. the use of documents snippet in a ranked list to enable to users rapidly identify relevant documents without needing to read the whole document.

Sometimes the information necessary to address the information need can be captured in a summary.

A step beyond this for reducing user effort is *question answering (QA)* systems.

Instead of retrieving potentially relevant documents for the user to read or otherwise access, a QA system seeks to provide the answer or answers to the specific user's question.

Introduction

Requests which can be addressed by QA systems have differing levels of difficulty. They may have:

- simple answers to simple questions, e.g. an easily identified name of a person, place, organisation, etc.
- simple answers to difficult questions, e,g. where an amount of money must be totalled from a number of different sources, e.g. how much money did Manchester United spend on players last season?
- difficult answers to difficult questions (sometimes apparently easy ones), e.g. composing a list from multiple documents, or contrasting information in multiple documents.

Question-Answering

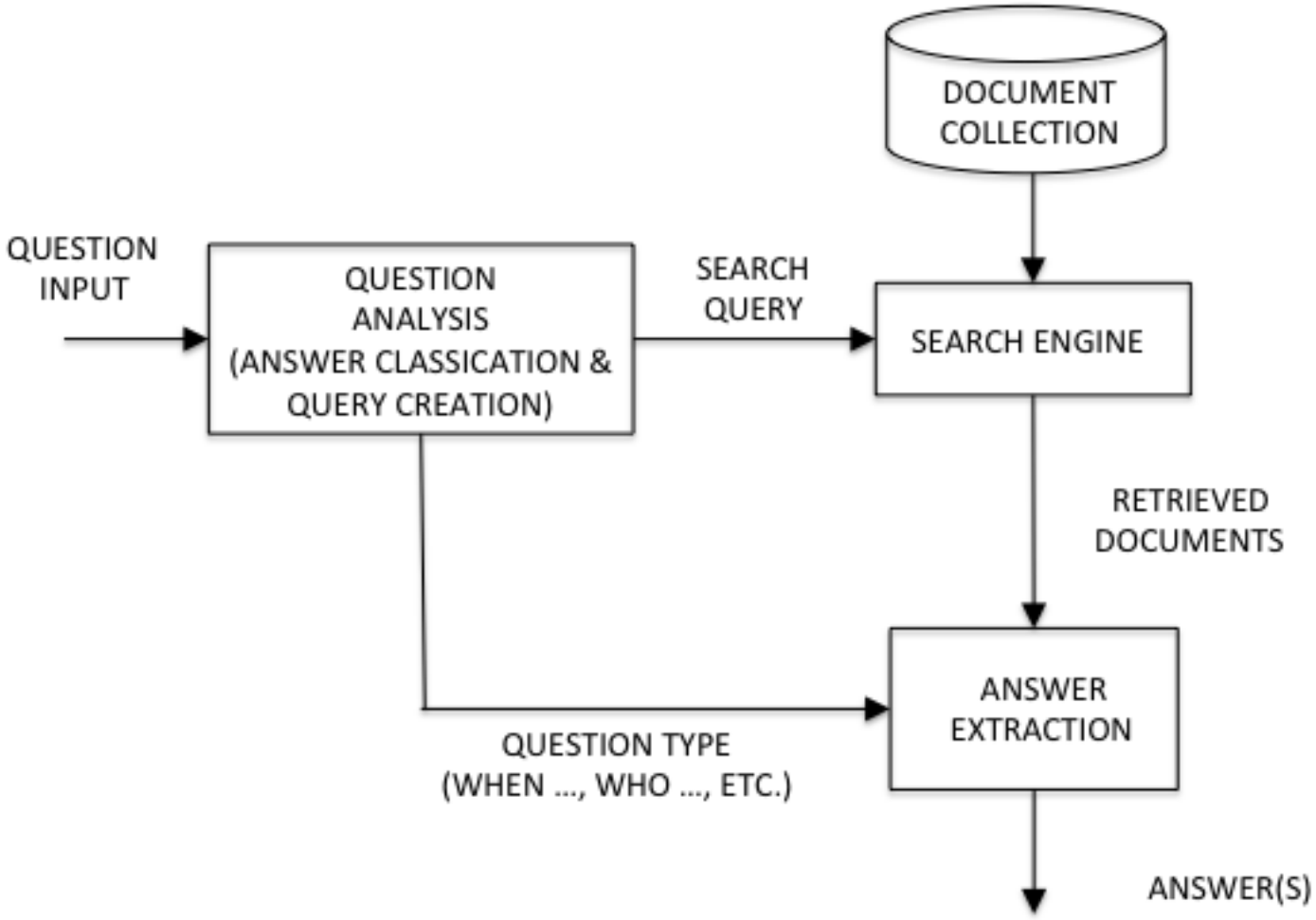
A general purpose QA system or even one for a broad specific domain, e.g. medicine, needs to access a large corpus of information in order to have a reasonable prospective of it containing the answer.

An IR system can be good at identifying a small number of documents from within a large collection which are potentially relevant to a requests, and from which the answer to a question can be extracted.

Finding the answer within a retrieved item will require the application of an answer extraction component.

Standard QA pipeline: system typically combines information retrieval (IR) to create a reduced answer space of retrieved documents and answer extraction to locate the answer within these documents.

Question-Answering



Typical Question Answering workflow

Question-Answering

Typical workflow of a QA system:

- Analyze question to determine what type of question it is.
- Form a search query from the question. Query type, e.g. “when”, is usually left out of the query since it is not related to the topic and identification of potentially useful documents.
- Enter the query into an IR system to retrieve a number documents which are potentially relevant and potentially contain the answer.
- Enter output of question analysis module and top ranked retrieved documents into answer extraction module.

Question-Answering

- Answer extraction module analyses retrieved documents to try to identify answer or answers to the question.

The IR component is generally robust and cheap to apply.

The answer extraction module is typically more expensive to apply.

The retrieved list is truncated either after a fixed number number of documents or based on the rate of decrease in the matching scores.

There is a tradeoff in precision and recall in deciding the length of the list.

- Too short: answers to the question might not be included (so the recall of the possible answer will be too low).
- Too long: too much irrelevant information may be included increasing the computation time and chance that the wrong answer might be found.

Question-Answering

Two contrasting approaches to QA:

- knowledge-based: apply formal natural language processing (NLP) and extensive linguistic resources.
- data-based: use of large document collections with shallow informal language processing, and exploitation of information redundancy.

Both approaches adopt the same standard QA architecture.

The IR stage identifies a number of documents from which the answer might be extracted, but they differ fundamentally in the technology of the answer extraction module.

We will look at both of these approaches.

Knowledge-Based Question-Answering

Retrieved documents are parsed to find the answers to the question.

- locate examples of *Expected Answer Type (EAT)* in the retrieved documents.
- examine relationship between words from the question and the possible answers to try to identify the answer to the question.

The NLP methods used here generally fall into the category of information extraction (IE) methods which are designed to extract specific pieces of information from a text, e.g. analyse a text to determine that Dublin is the capital of Ireland.

The approach can fail if the NLP methods used or the dictionaries are not sufficient to cover the details in the questions or the answers.

Knowledge-Based Question-Answering

Process the question to determine the EAT.

The EAT is often some type of named entity, e.g. someone's name.

First, identify the word(s) in the question that determine the EAT:

who (easy - looking for a person)

```
Who is the president of Ireland?
```

what (much harder - may refer to many different entities).

```
What do most tourists visit in Dublin?
```

The EAT is "landmark"

Knowledge-Based Question-Answering

When we know the expected type of the answer, we can analyze the documents to try to find potential answers which match this type requirement.

How much did Manchester United spend on players last season?

- EAT = money

- What would the QA system need to be able to understand in a text in order to be able to answer this question?

Knowledge-Based Question-Answering

The contents of the available documents can be augmented with knowledge contained in additional domain specific data structures. These include:

- simple lists
- conventional databases
- purpose built structures - *taxonomies* and *ontologies*

Knowledge Representation using Taxonomies and Ontologies

Taxonomies are a means of capturing information about a particular topic in an hierarchical structure.

A subcategory of a category inherits the properties of its parent.

For example,

motor vehicles $\rightarrow$ cars, vans, lorries, etc

cars $\rightarrow$ family, sports, mini etc

Vehicles can be defined in different taxonomies for different applications.

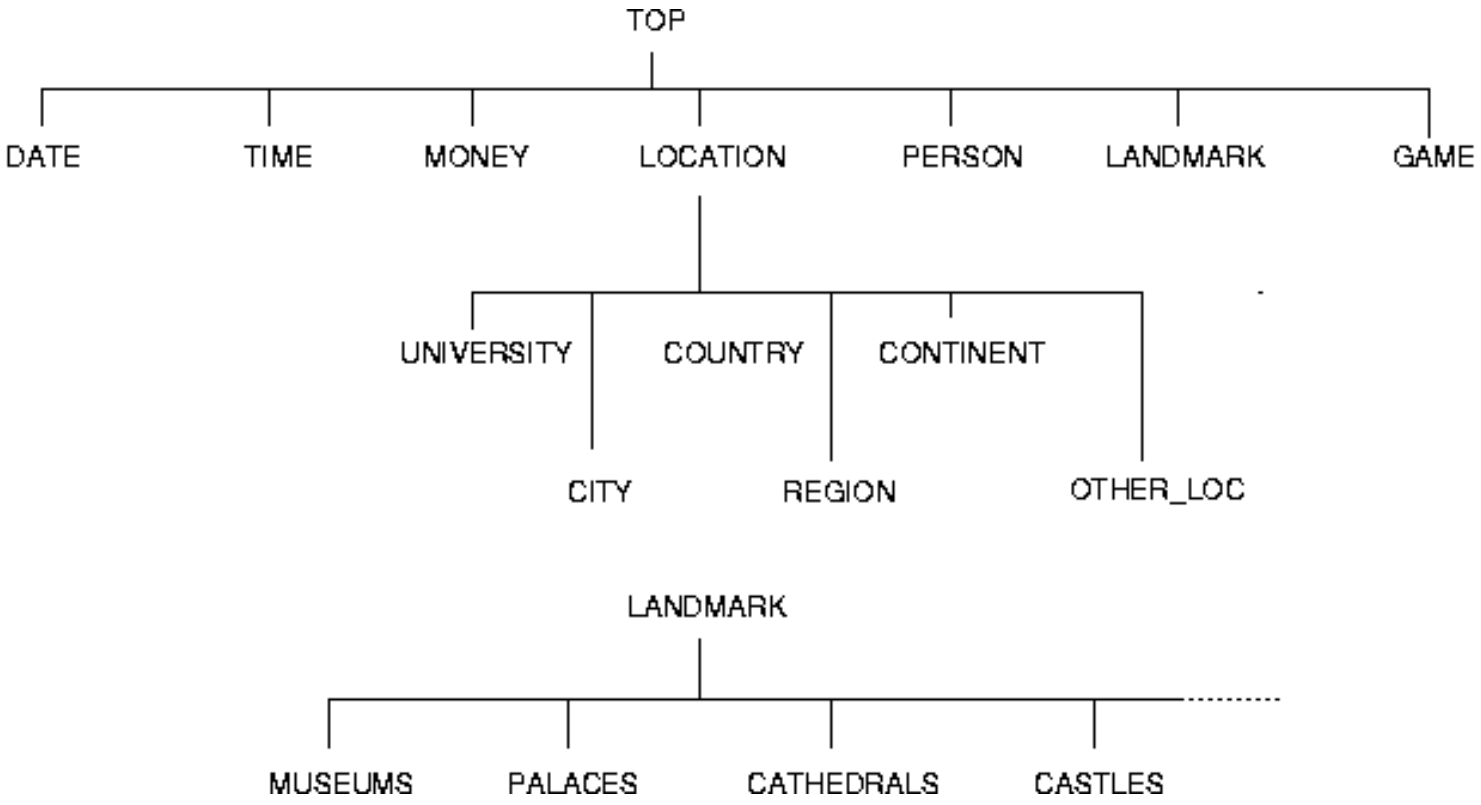
Knowledge Representation using Taxonomies and Ontologies

Ontologies generally include all the information captured in a taxonomy, but also details about rules and relationships between categories.

An ontology may include semantic information that can be used in problem solving and decision making.

Knowledge-Based Question-Answering

Using a taxonomy of places to identify location types which can be classified as landmarks.



Data-Based Question-Answering

Within small document collections often only one document may answer the question. (Of course, sometimes no document will answer the question!)

To answer a question from such a collection, the QA system must locate this answer.

For very large document collections (e.g. the WWW) many documents may contain the answer to the question.

For small collections both the question and the documents often need to be analysed in detail to understand the question and to match it with the answer(s).

Data-Based Question-Answering

The answer to the question: *Who killed Adraham Lincoln?*,

can be extracted from the text:

John Wilkes Booth altered history with a bullet. He will forever be known as the man who ended Abraham Lincoln's life.

but it will be quite difficult.

Thus in small collections it is more likely that sophisticated language processing will be needed to locate and extract the answer, since it may only be stated once in a complex statement.

The QA system must be sophisticated and robust enough to handle any way in which the answer may appear in the text.

Data-Based Question-Answering

In a large collection, particularly one such as the Web with many authors writing about the same topic, the answer to a question will often be stated in alrenative ways in a number of different documents.

In some of these documents the answer may be written in a simple way which can be matched very easily to the question, e.g. one of the documents may include the statement: *John Wilkes Booth killed Abraham Lincoln.*

Much shallower linguistic processing can be applied in this case, as shown in the following example.

Data-Based Question-Answering

Rather than attempting to "understand" the relevant contents of the question and the document, data-based QA systems rely on simple pattern matching.

For example, the question can be rewritten according to simple rules to form possible answer patterns.

For the question: *Where is the Louvre Museum located?*

The obvious answer pattern is: *The Louvre Museum is located in*, so we only need to find the pattern: *The Louvre Museum is located in Paris.*, in one document to be able to answer the question.

For the general question form *Where is* $w_1 w_2 \ldots w_n$, potential answer patterns can be generated by moving the verb to all possible position, e.g. w_1 is $w_2\ w_3 \ldots w_n$, $w_1\ w_2$ is $w_3 \ldots w_n$, etc.

Data-Based Question-Answering

Some of these patterns will be nonsense, e.g. *The is Louvre Museum located in*, *The Louvre is Museum located in*, but these are unlikely to match with the content of any retrieved document.

We can reasonably expect to find the statement *The Louvre Museum is located in ...* in at least one document on the Web.

Having found the statement we are looking for in one of the documents, we can reasonably hypothesize that the word following the statement will be the answer to the question.

In this case of course the answer will mostly probably be “Paris”, although we might also find the answer “France” or the district of Paris where the Louvre is located. If we have multiple possible answers, then we could use the method in next example to help choose the “best” answer.

Data-Based Question-Answering

Even for very large collections, we may not find the answer in exactly the form that we looking using this exact string matching method.

When no obvious answer can be found in the text, redundancy can often provide a reasonable “guess” at the correct answer, or to help us choose from among multiple possible answers (France, Paris, ... - see previous example).

We can look for words from the question in close proximity to a word of the correct form for the answer, e.g. a number or a place.

Data-Based Question-Answering

Question: *How many times did Bjorn Borg win Wimbledon?*

Assume that we are not able to find an answer of the form *Bjorn Borg won Winbledon 5 times.*, but we can find a number of documents containing phrases of the form.

(1) **Bjorn Borg** blah blah **Wimbledon** blah blah **5** blah . . .

(2) **Wimbledon** blah blah blah **Bjorn Borg** blah **37** blah . . .

(3) blah **Bjorn Borg** blah blah **5** blah blah **Wimbledon** . . .

(4) **5** blah blah **Wimbledon** blah blah **Bjorn Borg** . . .

Data-Based Question-Answering

In order to answer the question we could simply look for words in close proximity (remember Luhn's hypothesis about word proximity from earlier in the module), i.e. in this case Bjorn Borg and Wimbledon, and look for possible answers nearby.

Some more analysis could show that "How many ...?" questions will have a numerical answer. So we could look for numbers in close proximity to the question words.

The majority answer is most likely to relate to both the concepts here, i.e. Bjorn Borg and Wimbledon.

Data-Based Question-Answering

Since someone asks the question ”How many” in relation to Bjorn Borg and Wimbledon, we know that we are looking for a number.

If we find a number located close consistently to “Bjorn Borg” and “Wimbledon” - there is a reasonable chance that this is the answer to the question.

The number 37 probably relates to at least one of the entities (Borg or Wimbledon), and possibly both, but is probably less important since it only appears once. It may have been Borg’s age when the article was written, or may it relate to some specific incident involving him and Wimbledon, perhaps the aces served in a match.

But the most frequent number relating to both concepts is 5, so this can be proposed as the answer to the question.

Data-Based Question-Answering

In an interactive system, what could we do if the user knows or thinks that the answer to the question proposed by the QA system may be wrong or wants to confirm that it is correct?

Evaluation

As with information retrieval systems, evaluation is important in the development of QA systems.

What metrics should we use?

Correct: percentage of questions answered correctly

Precision: percentage of questioned answered where the answer is correct.

Correct may equal Precision - but it may not! why?

Recall: percentage of questions that could be answered correctly from the document collection that have been answered correctly.

No answer: percentage of questions that cannot be answered from the document collection, that have been identified as such. Identifying questions which cannot be answered is an important feature of QA systems.

Evaluation

Evaluation needs:

- Set of documents which will be used to answer the questions.
- Set of questions[a].
- The answers to the questions.

As with evaluation of IR systems, the documents and questions should be representative of the QA task for which the system is being developed.

[a] It may not be possible to answer all the questions from the document set.

Phone A Friend

One research group investigated the effectiveness of using WWW data to answer questions from the quiz show *Who wants to be a millionaire?*.

Their experiment used a collection of around 100 million web pages.

Results were as follows:

Question Value	Total Questions	Number Correct	% Correct
$100-$1,000	48	36	75 %
$2,000-$8,000	29	20	69%
$16,000-$64,000	22	15	68%
$125,000-$1,000,000	9	5	56%
Overall	108	76	70%

Phone A Friend

It is generally assumed that web data is of low quality.

Identifying high quality web pages is a major problem.

For QA from the WWW, quality issues are less important since results from many low quality pages can be combined to produce a consensus answer.

- *Quantity* can substitute for *quality*.

Homework Question: How might this observation interact with page importance measures such as *PageRank*?

IBM Watson

IBM developed *Watson* as an example of a system which aggregates large amounts of data of multiple types from many different sources to achieve an objective.

The development of Watson is an example of a “Grand Challenge” in computing which sets out to promote rapid development in a new area

IBM set out to develop a system to beat human champions at the TV game show “Jeopardy”!

Watson combines multiple established technologies - all based on research in AI related topics - including IR, NLP, machine learning, etc.

These work together to leverage a variety of unstructured and structured data and information sources in combination.

Cognitive Computing

Watson is an example of what IBM refers to as a *cognitive computing* application.

Cognitive computing systems are defined as incorporating the following fundamental principles:

- *Learn* - use data to make inference about something, e.g a domain, topic, person, ..., based on training and observation.
- *Model* - create a model of something and dynamically incorporate data into the model as it is observed.
- *Generate hypotheses* - assume that there is not a angle correct answer, the system is probabilistic and uses data to score multiple hypotheses which potentially address a particular problem.

IBM Watson

From IBM's perspective Watson is a means to improve business processes through natural human interaction with machines. In Watson's case by using natural language.

Users have become proficient in obtaining information using standard IR systems, e.g. web search engines, Watson seeks to support users in a different way via QA.

Watson promotes dialogue in information search - it can either give a potential answer or answers, or ask a follow up question seeking more information, in order to identify potential answers.

The cognitive computing approach seeks to ensure that the accuracy of results continues to improve through offline and online iterative training. Watson becomes "'smarter" each time new data is ingested.

IBM Watson

Exploiting all the available information for a task or domain requires the availability of diverse information processing technologies.

Thus, Watson combines both shallow and deep information analysis and processing technologies.

In QA terms, Watson combines elements of both knowledge-based and data-based type QA methods.

All available technologies within Watson can potentially produce hypotheses from which the answers(s) might be selected.

Jeopardy: the rules of game

Jeopardy is a TV quiz show in which contestants are required to answer general knowledge questions.

The game proceeds as follows:

- The host reads the question.
- When the question has been read in its entirety, a “ready” signal light is illuminated.
- Contestants are now allowed to active their buzzer to answer.
- The first contestant to active their buzzer has the chance to respond.

Jeopardy: the rules of game

Watson received the questions as electronic texts at the same moment they they were made visible to the human players.

Watson first had to determine whether it was sufficiently confident in its potential answer to activate the buzzer to get the right to answer.

For each question, Watson's three most probable answers were displayed.

After it had activated the buzzer, Watson gave its answer using speech synthesis.

Watson won the competition, but had problems with short questions containing only a few words.

Planning to Win at Jeopardy

To win Jeopardy by beating the best human contestants, IBM determined:

- You need to answer 70% of the questions, and get the answer right 80% of the time.
- You need to be able to answer in 3 seconds.

There is thus likely to be a tradeoff in accuracy and speed.

The correct answer is more likely to be identified using more sophisticated analysis and NLP methods, but the answer may come too late to be counted.

Planning to Win at Jeopardy

The main challenge to answering the questions in Jeopardy is their diversity.

Watson thus needed to ingest a wide range of information in order to be able to answer questions on a wide range of topics.

The data sources ingested by Watson included encyclopaedias, Wikipedia, dictionaries, historical documents, textbooks, news articles, music databases, literature, etc.

Watson had access to 200 million pages of structured and unstructured content while playing the game.

It was not connected to the internet during the game!

Planning to Win at Jeopardy

The general Watson strategy was:

- Question classification: fact-based, puzzle, pun - different answering approach used depending on question class.
- Identify many possible answers in parallel. These possible answers need to be broad, but not so broad as to be confusing.
- Determine a confidence level for each answer. Combine different sources of evidence and scoring to produce confidence score - similar idea to learning-to-rank in web retrieval.
- Choose the best answer.

IBM Watson

Strategy to make Watson fast enough to meet the 3 second response deadline.

- Operate many machines in parallel.
- Use powerful individual computers.
- Store knowledge base in RAM to minmize access time.
- Use really fast networking!

Commercial Application of Watson

IBM didn't ultimately build Watson as a cool toy to win Jeopardy!

The Jeopardy application was an important technical grand challenge in development of Watson technology.

Winning at Jeopardy was great publicity for IBM and Watson.

IBM see Watson technology as an important direction for development of commercial cognitive computing applications.

Commercial Application of Watson

The questions posed to Watson are likely to be much more complex in commercial applications, often without a specific correct answer.

Watson needs to be able to identify many potential answers, and will sometimes need to return alternative answers ranked by confidence, possibly with supporting evidence made available.

In any application Watson can make use of all available information sources, e.g. an application to support medical diagnostics can include published reports and papers together with knowledge structures such as taxonomies and ontologies.

Commercial Application of Watson

In general, include domain specific databases, ontologies and taxonomies, as well as domain specific unstructured sources, books, articles, specialist web content, etc.

Knowledge-base can be continuously updated during operation by ingesting new content as it appears.