Lifecycle of a Jeopardy Question Answered by Watson DeepQA

Leonard Swiezinski
firstname at lastname dot de

Abstract

In early 2011 IBM's question answering system Watson won in a Jeopardy contest against some of the most capable champions of the TV show. This is considered a milestone in language technology and artificial intelligence. In this work a brief overview of the DeepQA system and a hypothetical walk through the process of Watson answering a question are given.

1 Introduction

"We are asking questions and trying to find an answer, but the answer is in the question itself."

(Jiddu Krishnamurti 1980) [1]

In order to explain what Watson DeepQA is and how it works, some terms and some key concepts need to be explained in advance. This introduction defines some of them and gives a basic summary of the topic.

1.1 Question Answering vs. Information Retrieval

1.1.1 Information Retrieval

The famous free book on information retrieval by Manning and Schütze proposes following definition of information retrieval [6]:

> *Information retrieval (IR)* is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

In other words: For a *question*, IR is about finding needles in a haystack, where the needles are informational fragments that are relevant to the question, while the haystack is a larger set of informational fragments that don't have to be relevant to the question. The question or *information need* is usually given in the form of a *search query*. *Relevance* might loosely be defined as the amount of informational context that query and result share.

IR is prevalent in many fields of daily life, such as web search, public transportation schedules, libraries and many more. Like we know it from web search engines, IR systems can come up with a list of relevant documents for a particular search query. Such result lists are often ordered by relevance that is calculated by a *weighting scheme* with respect to the terms in the search query. A common example for such a weighting measure for a result is *Tf-idf*:

$$w_{t,d} = tf_{t,d} \times log \frac{N}{df_t}$$

[1] jiddu-krishnamurti.net (2013)

where $tf_{t,d}$ is the term frequency (count) of a term t in the document d, N is the total number of documents and df_t is the frequency of documents that contain t. This measure ensures, that if a term occurs often in a document, the documents gets a higher score, while if a term occurs often in many documents, it gets a lower score. This way, very frequent words, or stop words (like in English "is", "to", ...), lead to a lower weight than more specific words.

1.1.2 Question Answering

In contrast to IR, the term *question answering (QA)* refers to a slightly different concept, where the result is not a list of documents, but the actual answer to the question.[2] It seems fair to say, that QA is an extension of IR in a way, that it performs additional tasks and adds semantics:

- Determine the question *focus.* (e.g. "When" in "When was Mozart born?")
- Determine the *lexical answer type (LAT)* for a question. (e.g. date of birth)
- Retrieve the single correct information fragment that answers the question (*answer*). (e.g. 27 January 1756)
  - Retrieve nothing else.
- Optionally present the answer in *natural language*. (e.g. "Wolfgang Amadeus Mozart was born on 27 January 1756.")

Just like in IR, QA systems can either be bound to a specific domain, or otherwise follow a general purpose approach. Usually the latter is less efficient with respect to *result quality*. In order to judge the performance of a QA system, meaningful evaluation is needed. While *precision,* the percentage of correctly answered questions, is a common measure, in some cases answer can be *ambiguous*. In this case the answers need to be normalized by automatic methods or by human labor.

Some of the publicly available general purpose QA systems are Wolfram Alpha [3], START [4] and Answer Bus [5]. Some of the other available systems combine IR and QA approaches, for example by highlighting an answer in a list of IR search results. Another player that recently arose in this domain is Apple's Siri [6].

1.2 The DeepQA Approach

As described in their research report [3], the Watson team decided early in the process of creating its QA system, to employ some key concept that together form the idea of *DeepQA*.

1.2.1 Massive Parallelism

Watson works parallel in many ways. For a question, Watson creates *multiple interpretations*, and for each of them multiple *answer candidates (hypotheses)* from multiple sources. Later in the process the answer candidates get ranked and the answer with the highest correctness probability is chosen as final answer.

1.2.2 Many Experts

To judge the quality of an answer, many different scorers are employed by the system. According to their research report on the system, Watson includes more than fifty of such scoring methods. [3]

1.2.3 Extensible architecture

The previously mentioned parallel approach with many experts makes it desirable, to have a an architecture that lets methods easily be plugged in and out, to evaluate different combinations of methods. Also the handling of parallel computation needs to be designed adequately, so an appropriately engineered software architecture is needed.

[2] DeepQA FAQ (2013)
[3] Wolfram Alpha
[4] START QA
[5] Answer Bus QA
[6] Apple Siri

1.2.4 Pervasive Confidence

Instead of relying on single elaborate methods for judging the quality of a hypotheses, DeepQA lets the work be done by multiple methods. This way *it never has to trust a single component*. Instead, the system relies on *ensembles of methods*, where each method contributes to the final decision. However, the impact a single method has on the final decision may be weighted, based on LAT or the method itself. The set of generated evidence scores serves as input parameter for the final calculation of confidence for an answer.

1.2.5 Integration of Deep and Shallow Knowledge

As previously mentioned, Watson tries to not rely on single sources. For backing an answer with evidence, Watson tries to balance *strict semantics and shallow semantics*. Strict semantics might be contextual, geospatial, or temporal relations, while shallow semantics can include phonetic relations or corpus-probability.

1.3 Why Jeopardy?

Jeopardy, "America's Favorite Quiz Show", according to its website [7], is on screen since 1984. The Watson team chose the show as a challenge for the DeepQA system, because it has a number of hard requirements for the candidates:

- Jeopardy is not multiple choice, so the candidates have to find the answer by themselves.
- Players need to be confident about their answers, because Jeopardy penalizes wrong answers.
- Questions can have ambiguous wording.

These requirements make it necessary, that Watson excels at various fields of QA technology, such as parsing of natural language sentences, question type classification, relation detection and logical reasoning. [3]

A property that distinguishes Jeopardy from other quiz shows is, that questions are given in the form of a *clue*, while the answer has to be given in the form of a question. However, in this work the term "question" also refers to the clue that is presented by the host of the show and "answer" refers to the question the candidate replies with. Also no other kinds of Jeopardy-specific game tactics are covered in this work.

1.4 Technology And Knowledge Resources

1.4.1 Watson's Technology Stack

As reported by the Watson team [3], the system is based on following partial list of software applications:

- Apache UIMA (Unstructured Information Management Architectur) [8]
  - Used as architecture for linguistic analysis pipelines
- Apache Hadoop [9]
  - Used for distributed parallel computing
  - UIMA-AS (UIMA Asynchronous Scaleout) is based on UIMA and Hadoop to perform linguistic computing in a scalable, distributed and asynchronous way.
- Lucene [10], INDRI [11]

[7] jeopardy.com (2013)
[8] Apache UIMA
[9] Apache Hadoop
[10] Apache Lucene
[11] INDRI

  - As search engines for retrieval of candidate answers and for evidence retrieval
- Triplestores and SPARQL [12]
  - Another source for candidates and evidence

1.4.2 Knowledge

Besides learning custom knowledge, such as answer patterns, various knowledge sources are included. [3] Some of them are listed as follows:

- Wikipedia [13]
- Freebase [14]
- Geo- and weather-data
- Dictionaries and thesauri
- Wordnet [15]
- DBPedia [16]

1.4.3 Hardware

Doing massively parallel computing comes at the price of appropriately powerful hardware. While early versions of the software, running on a single processor, needed about two hours to answer questions, the version that powered Watson during the Jeopardy shows had more impressive numbers:

- Run-time latency between 3 and 5 seconds [3]
- 2500 computing cores [3]
- 16 Terabyte RAM [17]

[12] SPARQL
[13] Wikipedia
[14] Freebase
[15] Wordnet
[16] DBpedia
[17] csee.umbc.edu (2013)

2 Lifecycle of a Jeopardy-Question answered by Watson

2.1 Example Clue

In the first show of the series of Jeopardy shows with Watson that was broadcasted in february 2011, Watson received following clue as a text-file [18]:

A piece of a tree, or to puncture with something pointed.

Brad Rutter, one of Watson's human opponents, previously chose the question for $600 from the category "Alternate Meanings".

Approximately five seconds later Watson buzzered and replied with the correct answer "What is stick?". Since the top three of Watson's answers for each question were displayed during the show, we can see, that Watson gave this answer a score of 96% while the following answers "lumber" and "bark" received only 20% and 10%.

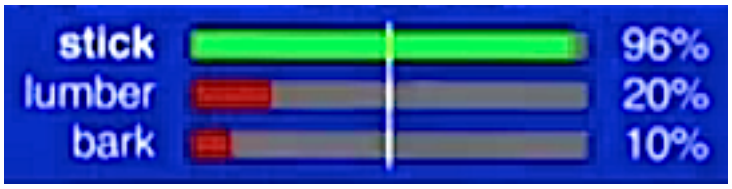

Figure 1: Watson's answer ranking for the example clue

While this question will be used, to illustrate a hypothetical walkthrough for the question answering process, another question will be used to explain points where the previous question doesn't fit:

This famous composer was born on January 27 1756 in Salzburg.

We already learned, that a correct answer in Jeopardy would be "Who is Wolfgang Amadeus Mozart?".

2.2 Intuition

What did Watson in these five seconds before his answer? While the intuitive way, a human would answer a question, could be described as a sequence of reading, understanding, memorizing (or solving a puzzle), judging correctness and finally answering, Watson needs to take a computational approach that is less intuitive.

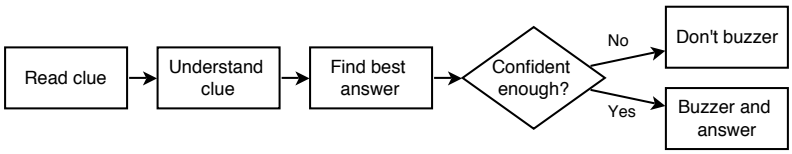

Figure 2: Jeopardy question answering intuition

[18] Youtube (2013): IBM Watson Jeopardy Full Episode (Day 1)

3 Steps to Solve a Jeopardy Clue Computationally

3.1 Step 1 - Linguistic preprocessing

3.1.1 Tokenization

Tokenization is the basic step that most applications in *natural language processing (NLP)* start with: an input text is segmented into smaller units, such as words, numbers and punctuation. Even if most people have an intuitive understanding of this concept, there is no general rule for building tokenizers, because there are different needs for different applications and different natural languages. Also, due to structural ambiguity, it is not always clear, what tokenization is valid for a given text. Depending on the tokenizer, the string 'A. Einstein' could be recognized as:

- ['A', '.', 'Einstein'] (sentence ends with 'A', next sentence starts with 'Einstein')
- ['A.', 'Einstein'] (tokenizer has rules for detecting abbreviations)

Tokenized text can be represented in various forms. A common representation that is easy to read, is tokens delimited by space characters. In many cases this is just inserting spaces before punctuation characters.

For our example this would be:

A piece of a tree , or to puncture with something pointed .

Given such a representation, it is possible, to enumerate tokens for later reference:

1. A
2. piece
3. of
4. ...

3.1.2 Part of speech tagging

Detecting the *part of speech (POS)* of a word in its context is called *POS-tagging*. Therefor a *set of POS-tags (tagset)* has to be defined (Simple example: "N" for nouns, "V" for verbs, "J" for adjectives, ...). The tagset is usually language specific. Common methods for POS-tagging are based on supervised machine learning with large sets of manually labeled data.

```
A/DT piece/NN of/IN a/DT tree/NN ,/, or/CC to/TO
puncture/VB with/IN something/NN pointed/VBD ./.
```

Figure 3: POS-tagging for first example clue.

3.1.3 Lemmatization

Most natural languages have rules for *inflection*. For example the english verb "to go" can be inflected as:

- goes
- gone
- went
- ...

All this word forms share the same basic concept (something is transitioning to somewhere), but are represented by different word types. The *lemma or base form* of these words is the verb 'go'. It has been shown that transforming words into their stem or lemma can drastically improve recall in information retrieval [4]. While the term *lemmatization* refers to a method that aims to find the

linguistically correct lemma of a word, *stemming* refers to a more simple approach that transforms words into pseudo-stems that are sufficient for most applications in IR, but don't have to be linguistically correct.

Methods to retrieve the base forms of words are usually language specific. Methods for lemmatization or stemming can be rule-based (mostly early approaches), list-based (not feasible for most languages), or based on supervised or unsupervised machine learning. Combinations of these approaches are also possible. [7, 2]

The Watson team chose to represent linguistic annotations as *Prolog* facts.[1] for our previous enumeration of tokens, lemmatization would look like:

- lemma(1, 'a')
- lemma(2, 'piece')
- lemma(3, 'of')
- ...

3.1.4 Question decomposition

Given the previously gathered tokenization , Watson should be able to *decompose* a clue into *subclues* if needed. This decomposition can be achieved by applying a set of Prolog rules. In our example the *grammatical conjunction* "or" indicates a boundary between subclues.

Algorithm 1 Prolog rule for subclue decomposition by detecting conjunctions

```
subClues(subclue1, subclue2) :-
  subclue(subclue1)
  conjunction()
  subclue(subclue2)
```

3.1.5 Parsing

In comparison to programming languages, it is usually not feasible, to parse any given natural language text unambiguously, due to the ambiguous nature of natural languages. However, Watson might generate separate *parse trees* for a clue and its subclues.

For this work Stanford Parser [19] has been used for illustration.

Algorithm 2 Partial parse-tree of clue

```
(ROOT
  (NP
    (NP (DT A) (NN piece))
    (PP (IN of)
      (NP (DT a) (NN tree)))))
...
```

[19] Stanford Parser (2013)

3.1.6 Named Entity Recognition

Named entities are parts of texts that represent locations, persons, organizations, times, amounts and other categories of entities. *Named entity recognition (NER)* is the process of automatic annotation of such occurrences, which can carry important semantic information about the clue. Such annotations can be useful for later use in the pipeline. A famous system for NER, that can be tested online, is Stanford NER. [20].

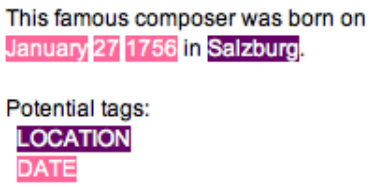

Figure 4: Named entity recognition example output of Stanford NER

3.1.7 Subject and Object Detection

In order to allocate a clue a semantically, it is helpful to differentiate between *subject and object in a verb phrase*. For the first subclue of our first example, Stanford Parser provided following dependencies:

Algorithm 3 Type dependencies represented as Prolog facts

```
det(piece-2, A-1)
root(ROOT-0, piece-2)
prep(piece-2, of-3)
det(tree-5, a-4)
pobj(of-3, tree-5)
```

3.1.8 Focus detection

In most cases, the *focus* of a clue in Jeopardy can be replaced with the answer without altering the semantics of the clue. The focus of our second example question is "This famous composer". For our main example both subclues are question foci. Generally the focus is a fragment of a question that indicates what is actually asked for. Watson detects such foci by applying various Prolog detection rules. [1]

The focus of our second example is "This famous composer", as it could replaced by "Wolfgang Amadeus Mozart", without altering the meaning of the sentence.

3.1.9 Detection of Lexical Answer Type

The LAT of a question is not always helpful for semantically interpreting questions. For some questions it is, but for other questions it even can be misleading. [3] However, *LAT-detection* can contribute to some correct answers. The Watson team trained a supervised classifier with a set of question-answer-pairs and by employing the previously discussed linguistic information to build a LAT-detector. However, for some of the questions no meaningful LAT can be detected. In such cases a *general purpose answer type* applies. [3][5]

Algorithm 4 LAT-detection rule for second example clue

```
<FOCUS_PERSON> was born on <DATE>* in <LOCATION>.
=> PERSON_BY_BIRTHDATE_AND_BIRTHPLACE
```

[20] Stanford NER

3.2 Step 2 - Finding Possible Answers: Hypothesis Generation

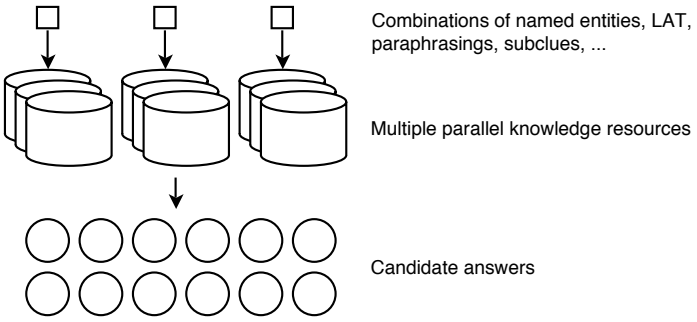

Figure 5: Answer candidate generation

At this point, maybe one second after Watson received the question, the system ideally "understands" the clue and "knows" what type of answer is expected. The next step in the previously sketched intuition is finding the best answer. This implies, that any possible answers have been found at all, and Watson's next task is exactly that: *finding answer candidates*.

3.2.1 Primary Search

In the previous steps, depending on the clue, Watson gathered various informational fragments, such as subclues, lemmata, named entities, LAT and dependent words to the focus. Based on this features, Watson can apply rules to generate queries for its knowledge resources, such as *index based search engines*, *triplestores*[21], *arithmetic expression evaluators, dictionaries and more*.

For our second example clue, a SPARQL query can retrieve the correct answer.

Algorithm 5 SPARQL query that finds answer for the second example clue

```
SELECT ?name ?birth ?person WHERE {
  ?resource dbpedia2:composer ?person .
  ?person dbo:birthPlace :Salzburg .
  ?person dbo:birthDate ?birth .
  ?person foaf:name ?name .
  FILTER (?birth < "1757"^^xsd:date) .
  FILTER (?birth >"1756"^^xsd:date) .
}
```

[22]

However, *Watson would not rely on a single search operation*, but also searches for different combinations of informational fragments, or in other resources. For example a lookup for a search query that consists of "composer", "born", "Salzburg" and "January 27 1756", is likely to retrieve documents that contain the correct answer as well. This can be illustrated with a search engine lookup. [23]

[21] Triplestores are databases for triples, consisting of subject, predicate and object. Example triple: (Mozart, bornIn, Salzburg)

[22] SPARQL lookup for second example clue (2013)

[23] Search engine lookup for example clue 2 (2013)

The correct answer to the other example question can be found by search engine lookups as well. To emulate a search in an inverted index of wordnet pages, a search query that is restricted to pages of the online version of wordnet does a fairly good job.[24]

Since the category of the first clue is "Alternate Meanings", dictionary and thesaurus lookups can be successful too. [25]

In order to increase recall of search operations, queries can be *paraphrased*. A naive approach to paraphrasing would be substituting words by synonyms that are retrieved from a thesaurus.

3.2.2 Candidate Answer Generation

From the results of the previous step, it is only a small step to a list of candidate answers. Depending on the type of knowledge resource, *a number of answer sized snippets is extracted* from every result set. According to the Watson team, in about 85% of all cases, the correct answer is included in the top 250 candidates. If the correct answer is not included in the list of candidates, there is no chance, that Watson ever answers the question correctly. [3]

3.3 Step 3 - Finding the best Answers: Filtering, Merging, Confidence Estimation and Ranking

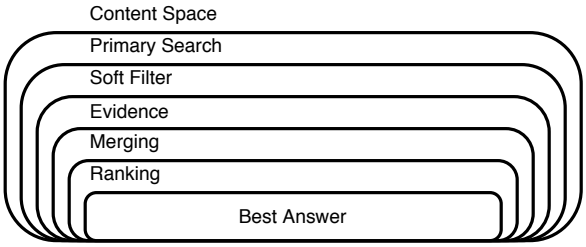

Figure 6: Watson filtering everything but the final answer from its knowledge

At this point, maybe two or three seconds after Watson retrieved the question, it ideally has a few dozen or even hundreds of candidate answers. Now the system needs to figure out the candidates that are most likely to be the correct answer, *rank* them by *confidence* and finally decide, if it is confident enough about the top ranking answer.

3.3.1 Soft Filtering

A first step that tries to minimize the amount of candidates is *soft filtering*. A set of *fast computing methods,* which filter out candidates by simple measures, such as the likelihood of a candidate being an instance of the previously detected answer type. [3]

For our second example clue, a candidate from a search engine result could be "Innsbruck". Since Watson already detected, that the expected answer is a person, it can easily filter out this candidate.

[24] Search engine lookup for example clue 1 (2013)
[25] Thesaurus lookup for example clue 1 (2013)

3.3.2 Evidence Retrieval and Scoring

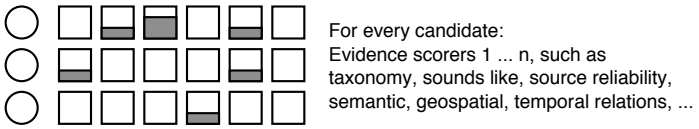

Figure 7: Evidence Scoring

Every candidate that passed soft filtering, still has a chance to become the final answer. In order to *rank candidates*, Watson employs an *ensemble of independent scorers*. Each scorer measures a specific aspect, such as *different types of relationships*.

For our first example it might detect a short path in a graph based thesaurus, or it might calculate a high amount of coreferences that the terms "puncture pointed" and the candidate "stick" share.

For our second example clue, it might detect that "Wolfgang Amadeus Mozart" can be classified as composer, based on taxonomy lookups.

Watson has many different of such evidence scorers, to cover a large range of different domains. For example it contains a method that calculates the scrabble score of an answer candidate, or general purpose methods like calculating the TF-IDF of an answer in combination with the question. [3]

3.3.3 Answer Merging

By *merging semantically equivalent answers*, Watson has a better chance to rank the best answer on top of the list, because equivalent answers would compete against each other during ranking. [3] Depending on the answer type this can be done by synonymy, arithmetic equality, geospatial distance or other measures, such as *semantic relatedness*. Ideally Watson would merge the answers "Mozart" and "Wolfgang Amadeus Mozart" and combine their results from evidence scoring. To achieve merging of semantically equivalent answers, DeepQA can employ methods like normalization, matching algorithms or calculation of semantic relatedness. [3] A measure that would indicate the semantic relatedness of two answer terms would be *Pointwise Mutual Information (PMI)*, a measure of statistical dependence of two terms [8]:

$$PMI(A,\, B) = log \frac{P(A \mid B)}{P(A)P(B)}$$

Where $P(A)$ and $P(B)$ are be the probabilities of the terms A and B in a given *text corpus* and $P(A \mid B)$ is the probability of *cooccurrence* of both terms in documents or smaller units of the corpus. The higher the value, the stronger is the considered semantic connection between both phrases.

3.3.4 Ranking and Confidence Estimation

Taking the output vector of the evidence scoring as an input parameter, Watson can now calculate an *overall ranking* for the candidates. Again, Watson does not have to trust a single *ranking algorithm*. Instead it can employ an ensemble of algorithms and calculate the arithmetic mean of the ranking scores for each candidate. For the first example clue, the answer "stick" gets a much higher rank than the answers "lumber" and "bark", since the latter two answers only correlate with the first subclue, while evidence scoring is likely to return no matches for the second subclue. The more evidence for an answer was found by evidence scorers, the higher is the confidence of that answer. Watsons confidence level for the answer "stick" was above a certain threshold and so Watson decided to buzzer.

4 Conclusion and Outlook Possible Applications

There is no question, that the Watson research team did a remarkable job at improving the current state of QA technology and their current plans to apply their DeepQA method to other domains such as healthcare and business decision processes seem reasonable.

This work tried to follow the path of Watson finding an answer to a question by example and illustrated the example by employing publicly available online tools wherever possible. Hopefully this encourages the readers to reproduce the results.

References

[1] P Fodor A Lally. Natural Language Processing With Prolog in the IBM Watson System. 2011.

[2] Mathias Creutz, Krista Lagus, K. Lind'en, and Sami Virpioja. Morfessor and hutmegs: Unsupervised morpheme segmentation for highly-inflecting and compounding languages. 2005.

[3] David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, and John M. Prager. Building Watson: An overview of the DeepQA project. 31:59–79, 2010.

[4] Wessel Kraaij and Renée Pohlmann. Viewing stemming as recall enhancement. In *SIGIR*, pages 40–48, 1996.

[5] Xin Li and Dan Roth. Learning question classifiers. In *International Conference on Computational Linguistics*, 2002.

[6] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[7] M. F. Porter. The Porter Stemming Algorithm. 2003.

[8] Peter D. Turney. *Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL.* 2001.