KULeuven

Department of
Computer Science

ADVANCED CAPITA SELECTA ARTIFICIAL INTELLIGENCE (H02A8A)

Contemporary AI topics
Report

Jeroen Craps (r0292642)
Jorik De Waen (r0303087)

Academic year 2016–2017

I. Introduction

February 1996, **Deep Blue**[1] wins the first ever game of Chess as a computer against the world champion at the time Garry Kasparov. March 2016, **AlphaGo** defeats 18-time world champion Lee Sedol in a five-game Go match with a score of 4-1.

Originally the biggest test for an artificial intelligence was the Turing test. The test requires that a human being is unable to distinguish the machine from another human being during a conversation where it has to answer to questions. A limitation of this test is that there is no objective way to measure the progress towards the goals of AI [3].

Computers have been able to perform tasks better than humans like path planning, finding patterns and playing games. After 20 years of progress, computers are now able to defeat the human mind at very complex games, but can it solve the same questions as asked in a 4th grade exam?

Even though 4th grade exams are trivial to solve for humans, they present an enormous challenge for our current AI systems. Researchers are actively working on this problem as this is seen as a key component of any new measurement of artificial intelligence [3].

There are several reasons behind this. It has all the requirements of a test [3]: Accessible, Comprehensible, Measurable and offer a graduated progression for simple everyday things to deeper understanding of subjects. Also, to answer these questions, a significant improvement in language understanding and the modelling of the world are be required. In this report we will be discussing some of the current methods that are being used to improve the performance of AI on these kind of tests.

[1] The Chess playing computer designed by IBM.

II. Overview

To learn the required knowledge, the artificial intelligence needs a source of data to work with. These data will almost exclusively consist of scientific papers and elementary school books. Currently two separate directions are being pursued for the Aristo system: image recognition [1] and text based knowledge extraction [4, 5].

Both streams will be explained in this report, but the focus will be put on the latter. The first will try to retrieve images from scientific papers and correctly label it by using the caption that is included with the table or figure. In the second method the goal is to extract logical statements from the text. When taking the test, the questions are translated into logical queries. The questions all use a multiple-choice format. This means that the system can focus on determining the most likely answer from the given possibilities. By matching every possibility to the concepts and relations extracted from the text, the support for each answer can be quantified by measuring how well it fits within the knowledge that's available. The answer with the best support is considered to be the best answer for this question.

III. Image retrieval

In current academic documents figures and tables are key sources of information, i.e. taking a look at a table in a paper can quickly summarize the work that has been done. However, these are not currently used in academic search engines. To better answer any kind of question the use of figures and tables is encouraged.

When there is a focus on scientific papers, it has been shown that **captions** are the key elements to indentify figures and tables. The difference between body text and captions has been proven to relatively easy to detect [1]. The search is done by looking for keywords that are likely to start a caption. Afterwards false positives are removed by applying a filter to

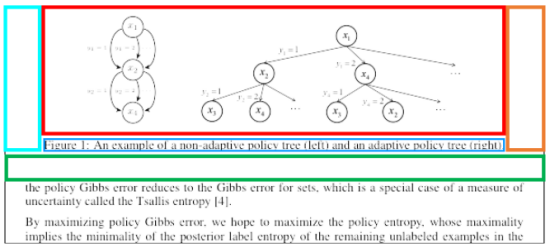

Figure 1: *An example of seperation into regions*

”Mechanical energy is produced when two objects move together.”

⇓

(“two objects”/?x “produce” “Mechanical energy”) “when” /CONDITION
(“two objects” /?x “move” “” [“together”])

Figure 2: *An example of an extraction*

them. This filter is focused on a particular format convention. This process of these filters is repeated until no false positives remain.

The second step is to classify every part of the paper as a certain region: caption, body text, graphical element or figure text. Caption regions are made from the captions and the following lines of text (if there are any). Poppler's algorithm is used to define body text or figure text. Additional heuristics are applied for improved accuracy. To find the graphical elements, the pdf is directly parsed. Internally PDF's make use of various “operators” that draw elements on the page. The algorithm uses the bounding boxes defined by the operators that the PDF would use to draw. The boxes can be merged if required to form the complete graphical region. An example of this can be seen in Figure 1.

To assign captions/titles to figures or tables clustering is utilized. These clusters would be pruned with additional rules for increased performance. Questions that would match up with elements in a caption can now be linked to a corresponding image which might contain useful information to answer the question.

IV. Text Based Knowledge Extraction

Texts from books and other sources often contain a large amount of information. Natural languages are flexible enough to communicate complex concepts, intricate relations and more. However, AI systems currently have a hard time accessing and reasoning with the knowledge contained in texts.
Even though interpreting natural language is, so far, an unsolved problem, the limited scope of solving science tests brings a solution within reach. A significant part of the knowledge can be expressed with relations like “X causes Y”, “X is part of Y”, “X is an example of Y” or “X [verb] Y”. The algorithm by Clark et al. [2] uses a hand-crafted set of extraction rules to generate a set of expressions from the text, where relations of interest are expressed. Figure 2 shows an example of this. Extractions are a semi-formal data structure, but they can easily be translated into formal representation. Many of these rules combined form a knowledge base.

V. Answering Questions

Answering a question is a two-fold problem: The first part is actually understanding the question. A question is a query for a piece of knowledge, so the text of the question needs to be translated to a formal language. The second part is actually answering the question. Once the query is expressed in the right format, it needs to be executed on the knowledge base. Not every answer is stated explicitly in the knowledge base, so logical inference is required to combine multiple rules into a single answer.

i. Lexical analysis of questions

The approach chosen by Krishnamurthy [4] to parse the question is to actually reconstruct it

from logical rules. The model is trained with manually translated questions. This allows the model to come up with possible rules and the likelihood that words and combinations of words translate into those rules. When the model is executed with a new question, it can look up the words and combine their corresponding rules from the bottom up. The probabilities calculated during training are used to decide at each step which of the options is the most likely.

i.1 Generating rules

Training the model comes down to constructing a probabilistic context free grammar which constructs questions. This kind of grammar consists of unary rules, nonterminal rules and terminal rules. A question can be built from a grammer by repeatedly replacing unary or nonterminal rules with other, matching, rules untill only terminal rules are left. A training example consists of a question (w)and a set of logical rules(L) which match the question. The goal is to learn rules which form a grammar that can construct the question. When faced with a new question, those same rules are used to build the new question. The algorithm that generates rules from a training example works as follows:

1. Add all rules from the training example: Add all L to the grammar as a nonterminal rule (it can be seen as the starting rule), and add a unary rule $L \rightarrow \ell$ to the grammar for every $\ell \in L$.

2. Enumerate all logical form splits: Many logical forms are a combination two independent logical rules which can be split into separate rules. For all $\ell \in L$, do a depth-first search starting at ℓ. This involves splitting logical form f into a set of g, h pairs, adding the rules $f \rightarrow g\ h$ and $f \rightarrow h\ g$ to the grammar and adding g and h to the queue to be explored later on. Note that these rules are binary, leading to a tree structure when applied.

3. Create lexicon entries: Add a terminal rule $f \rightarrow w$ for every combination between a word in the question and logical forms f encountered during the search above

4. Allow word skipping: Add nonterminal rules which allow allow for word skipping. Add $f \rightarrow f\ \ SKIP$ and $f \rightarrow SKIP\ \ f$ for all logical forms and $SKIP \rightarrow w$ for all words in the question.

Training Example:

$\mathbf{w}$ = What is the predator of bass ?

L = $\{\lambda x.\text{EATS}(x, \text{BASS}),$
$\lambda x.\text{CAUSE}(\text{INCREASE}(\text{BASS}), \text{INCREASE}(x)),$
$...\}$

Generated Grammar:

Unary rules:

$L \rightarrow \lambda x.\text{EATS}(x, \text{BASS})$

$L \rightarrow \lambda x.\text{CAUSE}(\text{INCREASE}(\text{BASS}), \text{INCREASE}(x))$

Nonterminal rules:

$\lambda x.\text{EATS}(x, \text{BASS}) \rightarrow \text{BASS}\ \ \lambda f.\lambda x.\text{EATS}(x, f)$

$\lambda x.\text{EATS}(x, \text{BASS}) \rightarrow \lambda f.\lambda x.\text{EATS}(x, f)\ \ \text{BASS}$

$\text{BASS} \rightarrow \text{SKIP}\ \ \text{BASS}$

...

Terminal rules:

$\lambda x.\text{EATS}(x, \text{BASS}) \rightarrow$ what

$\lambda x.\text{EATS}(x, \text{BASS}) \rightarrow$ is

$\text{SKIP} \rightarrow$ what

...

Figure 3: *The grammar generated by the algorithm from a training example*

i.2 Modelling probabilities

The generated rules from many different training examples form a grammar G. The replacement rules in G can build a large amount of parse trees. To determine the most likely interpretation for a question, we need to know the probability of generating each unique parse tree. $P(t|L;\theta)$ denotes the probability of generating tree t from G with as root a given label L, and parameters θ. This can in turn be broken down into a product of the probabilities of applying each rule in t. $P(f \rightarrow g\ h;\theta)$ and $P(f \rightarrow w;\theta)$ represent the probability of picking a replacement rule given the nonterminal f.

$$P(t|L;\theta) = \prod_{(f \to g\ h) \in t} P(f \to g\ h;\theta) \times \prod_{(f \to w) \in t} P(f \to w;\theta)$$

Not all of these trees build the same question, so for each specific question we're interested in all the trees with root L that build a certain question w. This can be denoted as $P(w, t|L;\theta)$. Finally, the probability of building a question given L is:

$$P(\mathbf{w}|L;\theta) = \sum_{t} P(\mathbf{w}, t|L;\theta)$$

To train the model, expectation maximisation can be used. This will tweak the probabilities of applying each replacement rule in such a way that the probabilities of generating the training questions is are maximised when their labels are given. With the probabilities of applying the specific rules from grammar optimised, parser can determine the most likely parse tree by going through the building process in reverse.

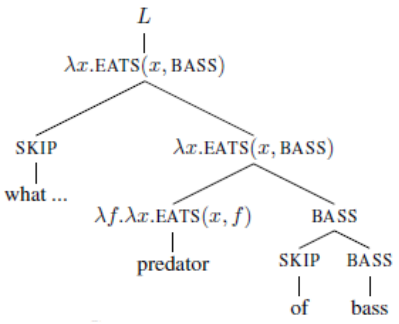

Figure 4: *The parse tree for the question "What is the predator of bass?" During training, the algorithm works from the top down to build the replacement rules. When the model is actually applied, these replacement rules are applied from the bottom up to come up with the final interpretation.*

ii. Querying the knowledge graph

Once the meaning of the question is clear, an answer needs to be found in the knowledge base. The algorithm by Li and Clark [5] is one way to achieve that goal. After parsing the question, they extract keywords. Background knowledge is added, which results in a knowledge graph of concepts connected by the relations found in the knowledge base. These concepts are the keywords themselves, but also other related words which provide context. The questions are multiple choice, so each option is placed within that graph and connected to the other concepts. The answer that gets the highest confidence score in the knowledge graph is chosen as the solution.

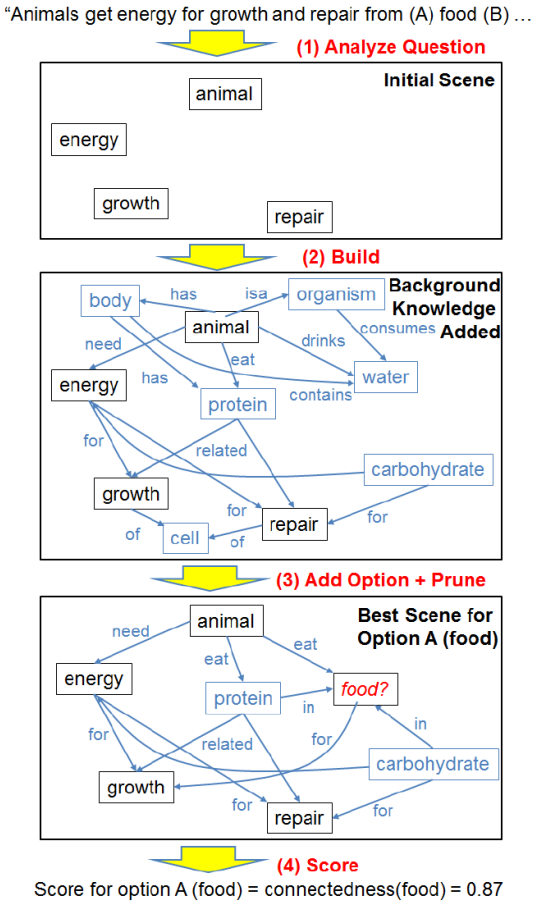

Figure 5: *An example of a question being answered using a knowledge graph.*

VI. Conclusion

To conclude this report, we can say that the current scientific progress is hopeful and improving rapidly. Texts can be interpreted to reasonable extend and converted to logical rules. Images can be extracted with the corresponding title and caption. With every iteration the scores that the artificial intelligence is able to achieve increases.

References

[1] Christopher Clark and Santosh Divvala. Pdffigures 2.0: Mining figures from research papers. In *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*, JCDL '16, pages 143–152, New York, NY, USA, 2016. ACM.

[2] Peter Clark, Niranjan Balasubramanian, Sumithra Bhakthavatsalam, Kevin Humphreys, and Kinkead. Automatic construction of inference-supporting knowledge bases. In *4th Workshop on Automated Knowledge Base Construction (AKBC)*, 2014.

[3] Peter Clark and Oren Etzioni. My computer is an honor student - but how intelligent is it? standardized tests as a measure of ai. *AI Magazine*, 2016.

[4] Jayant Krishnamurthy. Probabilistic models for learning a semantic parser lexicon. In *Proceedings of NAACL-HLT*, pages 606–616, 2016.

[5] Yang Li and Peter Clark. Answering elementary science questions by constructing coherent scenes using background knowledge. In *EMNLP*, pages 2007–2012, 2015.