Commonsense Reasoning Using Theorem Proving and Machine Learning

Sophie Siebert, Claudia Schon, Frieder Stolzenburg

▶ **To cite this version:**

Sophie Siebert, Claudia Schon, Frieder Stolzenburg. Commonsense Reasoning Using Theorem Proving and Machine Learning. 3rd International Cross-Domain Conference for Machine Learning and Knowledge Extraction (CD-MAKE), Aug 2019, Canterbury, United Kingdom. pp.395-413, 10.1007/978-3-030-29726-8_25 . hal-02520044

HAL Id: hal-02520044

https://inria.hal.science/hal-02520044

Submitted on 26 Mar 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Commonsense Reasoning using Theorem Proving and Machine Learning*

Sophie Siebert[1] [0000−0002−6812−796X], Claudia Schon[2] [0000−0003−2455−0974], and Frieder Stolzenburg[1] [0000−0002−4037−2445]

[1] Harz University of Applied Sciences, Automation and Computer Sciences Department
Friedrichstr. 57–59, 38855 Wernigerode, Germany
{ssiebert,fstolzenburg}@hs-harz.de
http://artint.hs-harz.de/

[2] Universität Koblenz-Landau, Institute for Web Science and Technologies
Universitätsstr. 1, 56070 Koblenz, Germany
schon@uni-koblenz.de
http://www.uni-koblenz.de/

Abstract Commonsense reasoning is a difficult task for a computer to handle. Current algorithms score around 80% on benchmarks. Usually these approaches use machine learning which lacks explainability, however. Therefore, we propose a combination with automated theorem proving here. Automated theorem proving allows us to derive new knowledge in an explainable way, but suffers from the inevitable incompleteness of existing background knowledge. We alleviate this problem by using machine learning. In this paper, we present our approach which uses an automatic theorem prover, large existing ontologies with background knowledge, and machine learning. We present first experimental results and identify an insufficient amount of training data and lack of background knowledge as causes for our system not to stand out much from the baseline.

Keywords: commonsense reasoning · causal reasoning · machine learning · theorem proving · large background knowledge.

1 Introduction

Commonsense reasoning is the sort of everyday reasoning humans typically perform about the world [22]. It allows to derive knowledge about continuity and object permanence, e.g., if a person enters a room, then afterwards, the person normally will be in the room, if she has not left the room. People have knowledge about objects, events, space, time, and mental states and may use that knowledge. All this implicit background knowledge is part of everyday human reasoning and must be added to a cognitively adequate automated reasoning system.

* The authors gratefully acknowledge the support of the German Research Foundation (DFG) under the grants SCHO 1789/1-1 and STO 421/8-1 *CoRg – Cognitive Reasoning*. A short and preliminary version of this paper appeared in [29].

Commonsense reasoning seems to be a task solved easily by humans. However, for a computer it is a rather difficult task, as the knowledge needed for this kind of problems often is huge and complex. Rather than relying on explicitly given facts, like e.g. geography, commonsense reasoning needs a broad understanding of the world on a very general level. This includes among others knowledge about physics, social interaction, cultural nuances, and basic objects present in the world. This kind of knowledge is so vast that it is difficult to explicitly formulate it in knowledge bases for computers. Problems resulting from this are incompleteness and inconsistency of the knowledge among others.

Error prone and inconsistent data are often addressed with machine learning techniques, e.g., in question-answering tasks. These algorithms perform very well on various tasks and are rather easily adaptable to multiple scenarios. However, they often lack explainability. Especially neural networks are more or less only a black box. While the behavior of a single neuron unit can be described easily, the sheer amount of computational entities leads to a complex overall behavior which is difficult to retrace and to understand. In addition, designing a neural network is often a process of guessing and trying out the right parameters. Also it is difficult to determine which inputs where important for certain processes. To maintain explainability one idea may be to make use of deductive reasoning techniques with theorem provers. They work in a deterministic way to derive facts from given statements, i.e., starting from a situation and a knowledge base, one can conduct all valid conclusions. However approaches relying solely on an automated theorem prover are facing problems with incomplete knowledge bases [5].

In this paper, we want to address commonsense reasoning by combining theorem proving with machine learning. Given a question-answering task in natural language (in English), we search in large knowledge bases for relevant information and feed both the task and the selected knowledge into an automated theorem prover which infers a logical model. This model contains additional facts that the theorem prover has been able to derive. However, it cannot be expected that the theorem prover alone is able to derive the answer to the question at hand. The derived facts can be fed into a machine learning algorithm, closing the gap to answer the commonsense reasoning task.

2 Related Works and Explainable AI

Explainable AI is artificial intelligence programmed to describe its purpose, rationale and decision-making process in a way that can be understood by the average person.[3] It is often discussed in relation to deep learning for applications like text analysis or object recognition in medical diagnosis systems or for autonomous cars.[4] In this context, *post-hoc* and *ante-hoc* analysis can be distinguished [17]: Models of the former type explain the given answer afterwards, e.g., by inspecting a learned neural network, while in the other case the model itself is explanatory, e.g., a decision tree.

How can explainable AI be achieved for commonsense reasoning? For this, a hybrid approach that combines a logic-based method with machine learning seems to be the

[3] http://whatis.techtarget.com/definition/explainable-AI-XAI, accessed: 14-June-2019
[4] http://heatmapping.org/slides/2018_MICCAI.pdf, accessed: 14-June-2019

right way [9]. Logical reasoning alone does not seem to be sufficient to handle commonsense reasoning tasks. Here, recurrent networks with LSTM (long short-term memory) [15] are a promising strategy and often used. They achieve a success rate of up to 84% on commonsense reasoning benchmarks, e.g., SemEval [24] which comprises 1,000 questions with two possible answers that require commonsense knowledge for finding the correct answers. Most of the teams participating in this competition use neural network approaches in combination with ontological knowledge like ConceptNet [19,30].

A general problem of machine learning approaches is often that they need big data to learn the desired behavior. This problem can be addressed by unsupervised pretraining, also for improving natural language understanding. [7] and [25] report recent encouraging results on a variety of benchmarks, e.g., question answering, based on this procedure. So, from a behavioral point of view, pure machine learning approaches can solve commonsense reasoning tasks. However, there is no representation of a reasoning process and hence usually no explanatory component in these systems.

As already mentioned, the reason is that machine learning with deep learning neural networks works as a black box. Without further components, drawn conclusions can neither be explained nor corrected if necessary. In the worst case, the computed answers are biased and may be discriminating. A famous example for this was Amazon's recruiting engine that was not rating candidates in a gender-neutral way. The computer models were trained to vet applicants by observing patterns in résumés. But since as a matter of fact of statistics most came from men, male candidates were preferred.[5]

Thus, the benefit of explainable AI may be diverse: First, systems with an explicit (symbolic) knowledge representation may help to find the correct answer. An example for this is the reasoning capacity of theorem provers which employs inference rules on given facts and rules. Second, it allows us to understand and hence to evaluate and possibly to revise answers. Furthermore, artificial intelligence systems should not only provide explanations but should also be advisable by explanations to guide the search for answers and to avoid biases or discrimination.

Commonsense reasoning tasks require a vast amount of knowledge data. Hence reasoning techniques from the fields of deduction, logics, and nonmonotonic reasoning should be employed and combined with machine learning. There are two ways of combination: Machine learning can be used as a subsystem to improve the reasoning process of theorem provers [10]. But it is also possible to do it the other way round. This means, we learn also the argument leading to the conclusion and thus provide explanations only *a posteriori*. In the context of big data, both procedures (with deductive reasoning and with machine learning) may be problematic, because possibly the resulting longish explanations, e.g., a complete proof of an argument, may also not be helpful. In this paper, we attempt to combine theorem proving with machine learning on top of it and try to tackle commonsense reasoning problems.

There are already several approaches for extracting rules from neural networks in general [8]. Special approaches combine inductive logic programming and machine learning [11,12]. These neural-symbolic learning systems start with a set of logical rules, encoded in neural networks. Then in addition, more knowledge is incorporated from examples into the system. Finally, a modified rule set can be extracted from the

[5] www.theguardian.com/technology/2018/oct/10/amazon-hiring-ai-gender-bias-recruiting-engine

```
Premise: The man broke his toe. What was the CAUSE of this?
Alternative 1: He got a hole in his sock.
Alternative 2: He dropped a hammer on his foot.

Premise: The pond froze over for the winter. What happened as a RESULT?
Alternative 1: People skated on the pond.
Alternative 2: People brought boats to the pond.
```

Figure 1. Problems 273 and 13 from the Choice of Plausible Alternative challenge.

improved learned network. But in this context it is assumed that the input is given as logic-based representation. For general natural-language question answering or text comprehension, which we want to address here, this does not hold, however.

3 Basic Methods and System Architecture

The objective of our project is to answer commonsense reasoning question-answering tasks like COPA [27] or the Story Cloze Test [21]. In general, each of these commonsense reasoning tasks consists of several parts of textual input: the premise describing a situation, a question about the situation together with n sentences describing an alternative answer from which the solution has to be selected. In our project, we currently focus on the COPA challenge.

The task is to determine the answer candidate which has a stronger (causal) relationship to the premise. Our approach to tackle these benchmarks is based on a combination of symbolic and subsymbolic methods: Knowledge represented in ontologies shall be used as background knowledge to perform inferences with the help of an automated theorem prover. The result of these inferences is then evaluated using machine learning, more precisely neural networks, to find answers.

3.1 Benchmarks

To get a better understanding of the project, we first describe the commonsense reasoning benchmarks used to evaluate our implementation. They form the input and consist of a question or situation and a set of possible answers. Currently we focus on COPA (Choice of Plausible Alternatives) [27] and the Story Cloze Test [21] which uses the ROCStories Corpora.

Problems in the COPA challenge (see Figure 1) consist of a premise and two alternative answers, each given in natural language. The corpus is equally divided into two categories, marked by the question: *cause* and *result*. The *cause* category requires backward causal reasoning, while the *result* category requires forward causal reasoning. All together there are 1,000 tasks which are split into 500 training and 500 test tasks.

The Story Cloze Test has a similar structure. It also has two answers per task, however, the situation part is longer there, see Figure 2. It is based on the ROCStories Corpora of 98,159 five-sentence stories. 3,744 of these stories were crafted into the Story Cloze Test, by taking the first four sentences of a ROCStory to describe a situation, and

```
Premise: Karen was assigned a roommate her first year of college. Her
roommate asked her to go to a nearbycity for a concert. Karen agreed
happily. The show was absolutely exhilarating.
Alternative 1: Karen became good friends with her roommate.
Alternative 2: Karen hated her roommate.
```

Figure 2. A Story Cloze Test example.

the last sentence for the correct answer, i.e., the most plausible continuation of the story. The wrong answer is a new element to complete the test.

3.2 The System

Based on the example of the COPA challenge, we now describe the structure of our system and provide details of our approach which is implemented in the system CoRg – Cognitive Reasoning (see Figure 3). As mentioned before, each problem in the COPA challenge consists of a premise and two alternatives. Since all three are given in natural language and our aim is to use an automated theorem prover, the first step of our systems transforms the input into first-order logic formulae. This is achieved using KNEWS [2], a tool that performs semantic parsing, word sense disambiguation, and entity linking. Predicate symbols used in the formulae created by KNEWS in most cases correspond to words (e.g., nouns, verbs and adjectives) of the original text. Word sense disambiguation in our case considers the predicate names of the formulae consisting of the so-called synset IDs of WordNet [20], a lexical-semantic network of the English language. Synset IDs group words with similar meaning for which short definitions, examples and also relations to other synsets are given. They correspond to the predicate symbols occurring in the first-order logic formulae and are used to determine the synonyms, hyponyms and hypernyms of the symbols from WordNet, providing us with a small lexical knowledge base related to the original text. Figure 4 presents the first-order logic formula for the premise of the COPA example 13 in Figure 1 created by KNEWS.

The symbols as well as the related gathered lexical information from WordNet are used to select relevant information from large first-order logic knowledge bases like SUMO [23], Adimen-SUMO [1], ResearchCyc [18] and YAGO [31]. Our system currently only uses Adimen-SUMO but we plan to integrate other ontologies and knowledge graphs such as ConceptNet [19,30]. Section 3.3 provides details on the selection of background knowledge. All gathered background knowledge together with the logical representation of the natural language is fed into the automated theorem prover Hyper [3] which performs inferences resulting in a (possibly partial) model. This is done separately for each answer and the premise of a problem resulting in $n+1$ (partial) models for each task, with n being the number of answer candidates of the benchmarks. In the case of problems from the COPA challenge, this process leads to the construction of three (partial) models.

These models represent the inferences performed by the theorem prover and are fed into a neural network to come to a decision. Each COPA task is split into two training examples such that the model of each answer is paired with the model of the respective

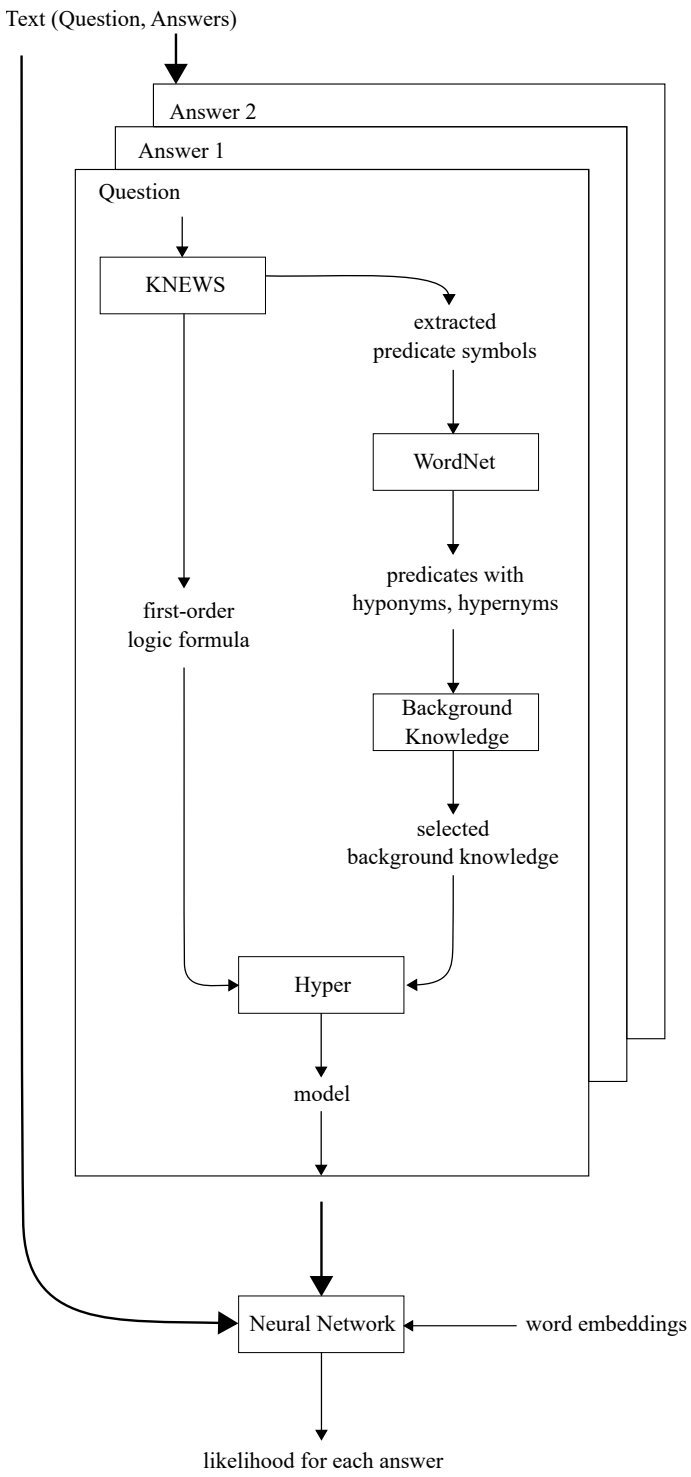

Figure 3. The CoRg system.

$$\exists x\big(pond(x) \wedge \exists y,z,v(r1to(y,x) \wedge r1Theme(y,z) \wedge r1Actor(y,v) \wedge bring(y) \wedge boat(z) \wedge people(v))\big)$$

Figure 4. First-order logic formula for the first alternative of the COPA example 13 presented in Figure 1 by KNEWS. To increase readability, the WordNet synset IDs determined by KNEWS for the words *pond* (09420266-n), *bring* (01441539-v), *boat* (02861626-n) and *people* (07958392-n) were replaced by their natural language identifier.

answer candidate. The neural network calculates a likelihood that the presented answer indeed fits to the situation it is paired with. The alternative with the higher likelihood is assumed to be the answer of the system. In case of multiple choice questions with n alternatives, the answer with the highest likelihood of being the right answer is selected.

3.3 Background Knowledge

Solving a reasoning task, humans naturally use their broad knowledge about the world. This background knowledge contains knowledge about physical relationships (e.g., a vehicle can overtake another vehicle only if it is faster) but also general knowledge (like the fact that dogs like bones). Since this kind of background knowledge goes far beyond statistical correlations on texts, we aim to use background knowledge which is represented in ontologies and knowledge graphs. Furthermore, we only use already existing knowledge bases and refrain from manually creating knowledge bases for the commonsense reasoning tasks. Currently WordNet and Adimen-SUMO are used as background knowledge.

As described in the previous section, the natural language text of a COPA problem is translated into first-order logic formulae using the KNEWS system, which furthermore performs word sense disambiguation by providing a WordNet synset ID for each noun, verb or adjective occurring in the text. As mentioned before, WordNet contains relations between synsets. Relations that are particularly interesting for our purposes are the hyper- and hyponym relation between synsets. From these relations we generate background knowledge in the form of first-order logic formulae. For example, the facts that *lake* is a hypernym of *pond*, *fishpond* is a hyponym of *pond*, and *pond* and *pool* are synonymous are translated into the following formulae:

$$\forall x\ (pond(x) \rightarrow lake(x))$$
$$\forall x\ (fishpond(x) \rightarrow pond(x))$$
$$\forall x\ (pond(x) \leftrightarrow pool(x))$$

Since this knowledge generated from WordNet has only a taxonomic character, these formulae are supplemented by parts of Adimen-SUMO. Adimen-SUMO is a large ontology consisting of axioms and individuals.

The problem that the Adimen-SUMO symbols do not coincide with the symbols of the KNEWS output in general is solved using the Adimen-SUMO WordNet mapping. This mapping specifies for each WordNet synset Adimen-SUMO symbols that are equivalent or belong to a subclass of the synset. From this information, we generate

bridging formulae. Since the generation of these formulae corresponds to the generation of formulae from hypernyms and synonyms described above, we refrain from a more precise description.

Because of the huge size of Adimen-SUMO it cannot be used by a theorem prover as a whole. Therefore we use selection techniques to select a subset of the knowledge provided by Adimen-SUMO that is relevant for the respective COPA problem under consideration. The selection technique we currently use is a relevance-based selection called SInE (Sumo INference Engine) [16] and is broadly used by first-order logic theorem provers.

In a preprocessing step, SInE computes some information about the knowledge base: For each symbol s, the number of occurrences in the whole knowledge base, denoted by $occ(s)$, is determined. Next, a *triggers* relation between symbols and formulae is defined. For each formula F and symbol s occurring in F, $triggers(s,F)$ is true iff for all symbols s' occurring in F $occ(s) < t \cdot occ(s')$ for some $t \in \mathbb{R}$. For $t = 1$, this means that each formula is triggered by the symbol with the fewest occurrences. For $t > 1$, the formula F is triggered by exactly those symbols from F that occur at most t times more frequently than the rarest symbol in F. The parameter t is called *tolerance*.

The selection of formulae suitable for a problem is performed by computing the so-called d-relevance as follows: Every symbol occurring in the problem and the formulae created using WordNet is set be 0-relevant. For a symbol s, which is d-relevant, all formulae F with $triggers(s,F)$ are d-relevant. Furthermore, all symbols occurring in a d-relevant formula become $d+1$-relevant. Given a problem, a knowledge base and $d \in \mathbb{N}$, this selection extracts the subset of the knowledge base consisting of all formulae which are d-relevant for the problem. More or less background knowledge is selected depending on the selection of the parameter d. In the following, the parameter d is referred to as *recursion depth*.

Recursion depth and tolerance are two parameters that we can vary when generating background knowledge. In addition, we can generate WordNet formulae for the text under consideration or not. Table 1 gives an overview of the different parameters for the selection of background knowledge. In future work, we also plan to add further sources for background knowledge like ResearchCyc, Yago and ConceptNet.

Parameter	Value
Integrate WordNet	true or false
Recursion Depth (SInE)	1-5
Tolerance (SInE)	1-5

Table 1. Parameters for Background Knowledge.

3.4 Using an Automated Theorem Prover

After background knowledge for the premise and both alternatives has been gathered, the automated theorem prover is called. We use the theorem prover Hyper [3] as it is not only able to provide proofs of unsatisfiability but also constructs models for satisfiable

problems. These models consist of a set of facts that can be inferred from the knowledge base (see Figure 5 for an example). Since this inferred knowledge is an important input for the following machine learning step, it is essential for us that the theorem prover is able to deliver this output. In addition, Hyper outputs the formulae used for the performed inferences which we want to use in future work to generate explanations.

The application of Hyper in our system has limitations: Even if multiple sources are used for background knowledge, the background knowledge still does not contain all important information for all problems. Therefore, we cannot expect the theorem prover to construct a complete inference chain to one of the alternatives but only a few inferences useful for the problem. Therefore, these are used as an input for our machine learning component.

4 Employing Machine Learning

Our system as described until now deduces a logical model for each premise and both of the alternative answer candidates. In those models, in addition to the original natural language text, facts are derived and can be used for further processing. As the overall goal is to answer the commonsense reasoning benchmark, we want to determine the likelihood for each of the answers belonging to the respective premise. For this, we use neural networks, as they proved their suitability in various other commonsense reasoning and text processing tasks (cf. Section 2). Thus, in this section, we will present the machine learning part of our system. This includes preprocessing the models and the explanation of the neural network architecture.

4.1 Preprocessing

Given a commonsense reasoning task, the natural language input usually needs to be encoded in a suitable manner. For neural networks, the input consists of the natural language text itself, as well as additional information like word embeddings, part-of-speech embeddings, and other features.

In contrast to other approaches, in our system we additionally face the challenge to process a logical model. While the natural language text in our context mostly is rather short and its meaning depends on the word order, this does not hold for a logical model derived by the theorem prover (Hyper). The logical model can contain up to thousands of lines. Furthermore, the derived facts are order-independent, as they do not form sentences but rather single statements on their own. Due to the size it is not reasonable to fed the whole logical model into the network. Therefore we apply numerous preprocessing steps to it which we shall explain now. To start, we depict representative parts of a logical model in Figure 5.

While preprocessing, we dismiss all structural information of the model and extract only the predicate and function symbols, like *winter*, *pond*, or *SeasonOfYear*. First, we replace the special characters $(), .$ with spaces, so that we can process the single elements. For each of those elements we apply some normalization, as they are later linked to word embeddings. This means: We drop all skolem constants (like *sK*0) and skolem functions (like *sK*5). We further drop the prefixes from the background knowledge base

$$winter(sK0).$$
$$pond(sK1).$$
$$n1froze(sK2).$$
$$p\_\_d\_\_subclass(c\_\_Freezing, c\_\_StateChange).$$
$$p\_\_d\_\_instance(sK2, c\_\_Freezing).$$
$$p\_\_d\_\_instance(sK5(sK2), c\_\_Cooling).$$
$$p\_\_d\_\_instance(sK0, c\_\_WinterSeason).$$
$$p\_\_d\_\_instance(sK0, c\_\_SeasonOfYear).$$

Figure 5. Excerpt of a model produced by the automated theorem prover Hyper for the formula representing the sentence *The pond froze over for the winter.* (cf. Figure 1) together with the gathered background knowledge.

which in case of Adimen-SUMO is $p\_\_d\_\_$ and $c\_\_$ as well as other underscore character variants. Often this preprocessing step causes two lines of the logical knowledge to become identical, e.g., both $(sk3(sk8(bird)))$ and $sk4(bird)$ are transformed into $bird$. In this case, we delete one of the duplicates. The remaining elements are usually common words and are looked up in the word embedding. If there is no corresponding entry, they are either not a real word or we cannot map the word to a numerical representation. This makes them useless for further processing. Thus we delete them.

This procedure results in a sequence of words which can be interpreted as text, although it is not a grammatically correct sentence in any form. The size of this text is greatly reduced in comparison to the original logical model and mostly does not exceed 100 words which is a reasonable size to put into the network. Optionally, we can transform the remaining words into a set, getting rid of duplicates. As the model consists of derived facts, ordering or duplicate words should not influence the outcome. This again reduces the input size. In the future we could also get rid of the meta-predicates from the background knowledge, like *subclass* or *instance*. As long as we do not integrate structure information, these words do not help deciding for an answer candidate, as they appear in both alternative models.

4.2 The Neural Network Architecture

In the neural network, we implement an attentive reader approach [32] making use of the framework Keras.[6] The input of the neural network consists of question-answer pairs, i.e., each task consisting of n answers is transformed into n training examples. We use a multi-input neural network, where the question and the answer are first separately encoded in a single network, and later on merged into a calculation core. The output is the likelihood that the given answer fits to the corresponding question. The general structure is shown in Figure 6 and explained now.

Each training example input consists of a premise P, an alternative A and a classification $y \in \{(0,1),(1,0)\}$. P and A are a list of word indices. The first step of the picture

[6] www.keras.io, accessed: 22-April-2019

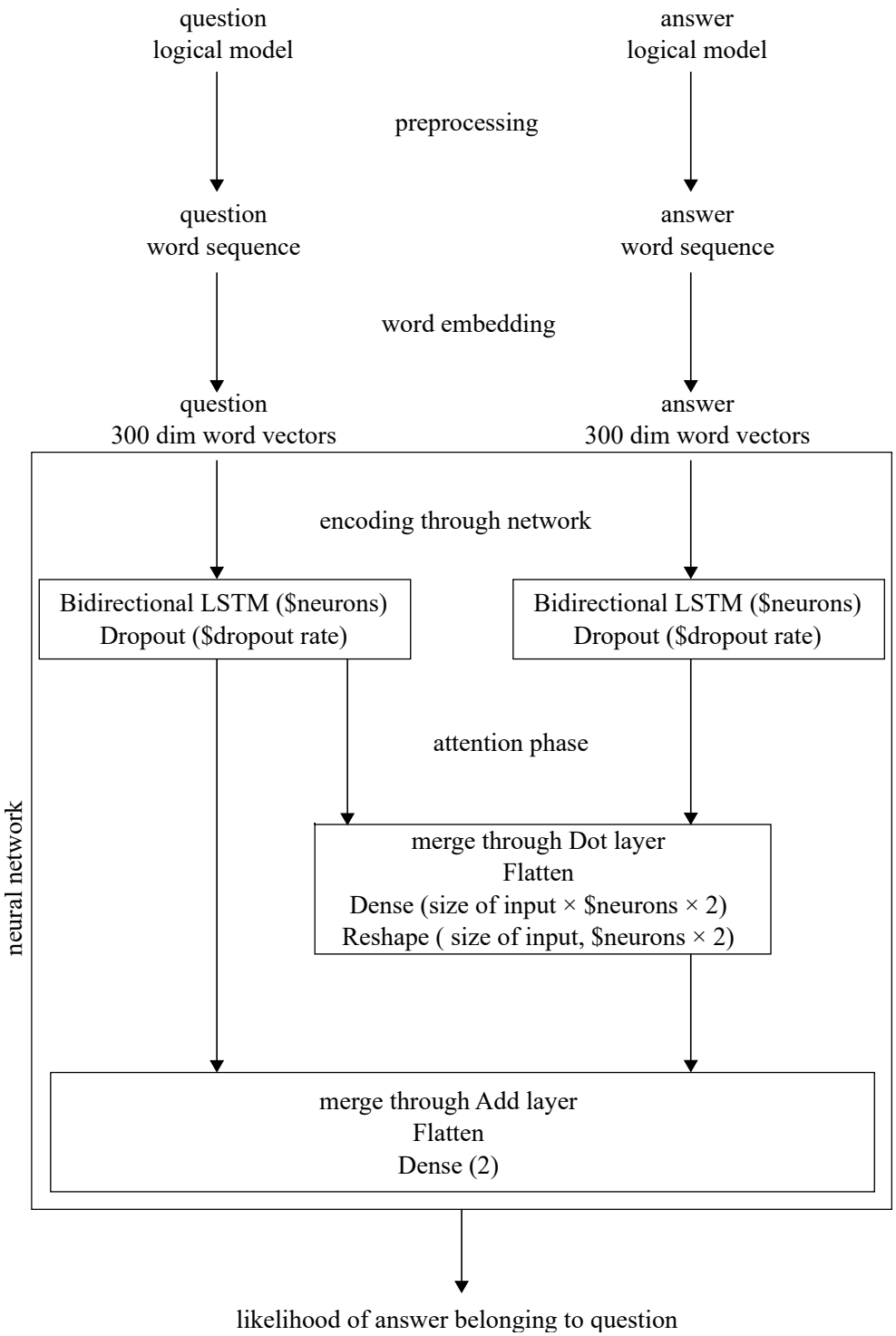

Figure 6. The neural network. The terms marked with $ are hyperparameters, which we use to tune the network. They are varied, such that the resulting network is optimized.

covers the preprocessing as explained in the previous section. The second step is the integration of word embeddings to enable similarity calculations among identifiers. For each word in P we use a word embedding vector from ConceptNet Numberbatch. Word embeddings map words into a high-dimensional numerical space, so that similar words have a short distance. In this project we use the state-of-the-art word embedding ConceptNet Numberbatch [30]. It outperforms other word embeddings like word2vec[7] and gloVe[8] in several benchmarks and reduce the bias of prejudices.[9] In our work, we reimplemented the built-in Keras embedding layer, so we can adjust the weight of the single words. We can choose between a constant weighting and a frequency weighting. In the future we may consider TF-IDF weighting [26] as well.

The third step is the encoding of the word embeddings information using a BiLSTM (bidirectional long short term memory) layer h^P. Analogously, this is done for the words in the answer candidate A. An LSTM can remember and forget previous information, providing a context of a current word into its previous text. In a bidirectional LSTM, the input is read both forward and backwards. Analogous to the forward-read input, the backward-read input provides a context into its following text, i.e., a look-ahead. Bidirectional LSTMs therefore embed a word into its context for previous and succeeding information.

Next, the encoded text passages are merged together using a dot product $Dot_P^A(h^P, h^A) = \sum_{i=1}^{N} h_i^P \cdot h_i^A$. This process is the attention phase of the network [32], calculating the shared features between the texts. It is followed by a fully connected so-called *dense* layer, to generate an answer embedding in the context to the question $Att_P^A(Dot_P^A) = \sum_{i=1}^{N} Dot_P^{A_i} \cdot w_i$. In the last step, both the merged core network as well as the input question network is again merged using addition $Add_P^A(h^P, Att_P^A) = \sum_{i=1}^{N} h_i^P + Att_P^A$. A two-neuron dense layer with softmax activation $y(Add_P^A) = \sigma(\sum_{i=1}^{N} Add_P^{A_i} \cdot w_i)$ assigns an output in the shape of $y^* \in [0,1]^2$, with $|y^*| = 1$ and describes a likelihood.

Throughout the network we use various layers to reshape our tensors as well as dropout and kernel constraint measures to avoid overfitting. We also changed the number of neurons of the LSTMs and dense layers. We use categorical cross entropy as loss, adamax as optimizer and accuracy as metric [13]. Currently, we can feed both the logical model and the natural language as is into the neural network. For future experiments, we want to integrate both approaches into one network.

During experiments we varied multiple parameters, see Table 2. First, one can change the general network structure. We propose an attentive reader approach [32] with LSTMs. In our first attempts, we also did experiments with a simple feed-forward net and an LSTM approach without the attentive reader model. However, they did not score well on our task. Second, we can choose between natural language and a logical model as input. Third, we can choose the embedding weights. In our current experiments we choose the input to be a set as it kept the input small and thus saves computation time. The following parameters are standard neural network hyperparameters [4] and were evaluated in the current experiments.

[7] https://github.com/tmikolov/word2vec, accessed: 21-June-2019
[8] https://nlp.stanford.edu/projects/glove/, accessed: 21-June-2019
[9] github.com/commonsense/conceptnet-numberbatch, accessed: 22-April-2019

Parameter	Value
Network structure	feed-forward, LSTM, attentive reader
Data type	natural language, logical model
Embedding weights	sequence, set, frequency
Neurons	2 – 100 in first layer
Dropout rate	0.1 – 0.5
Optimizer	adam, adamax
Learning rate	0.0005 – 0.002
Kernel constraint	1.0 – 10.00

Table 2. Parameters for the neural network.

5 First Ideas towards Explainability

The architecture of our system allows us to generate explanations for the decisions made. Since this has not yet been implemented, we will illustrate our idea briefly with an example. Let us take another look at COPA example 13 from Figure 1. The theorem prover Hyper derives the facts given in Figure 5 from the formula representation of the premise of this problem together with the selected background knowledge. Machine learning finds out that the facts in this model point more in the direction of model generated for alternative one than the model belonging to alternative two and therefore makes the decision for alternative one. With the help of a word embedding we determine the symbols in the model of the premise that point most in the direction of alternative one. Because of the similarity of *skate* to *winter*, *freezing* and *cooling* these symbols are recognized as relevant for the decision. In the computed model, the following facts (among others) contain these symbols:

$$n1froze(sK2).$$
$$p\_\_d\_\_instance(sK2, c\_\_Freezing).$$
$$p\_\_d\_\_instance(sK5(sK2), c\_\_Cooling).$$

To derive these facts, Hyper used (among others) the following two formulae:

$$\forall x((n1froze(x)) \rightarrow (p\_\_d\_\_instance(x, c\_\_Freezing))) \quad (1)$$
$$\forall x(p\_\_d\_\_instance(x, c\_\_Freezing) \rightarrow \exists y(p\_\_d\_\_instance(y, c\_\_Cooling) \wedge p\_\_subProcess(y, x))) \quad (2)$$

From the second formula, it is possible to generate an explanation that freezing involves a subprocess of cooling. We are aware that this is not yet an explanation for the correctness of alternative one. The reason for this is the fact that the background knowledge we currently use contains mainly taxonomic knowledge and does not adequately represent commonsense knowledge and reasoning in the strict sense. Therefore, we do not find any formulae in it that associates a frozen lake with winter sports. We hope to solve this problem by adding more sources of background knowledge such as ConceptNet.

However, the basic idea for generating explanations remains the same even after the background knowledge has been extended.

By neural-symbolic learning systems [11] (cf. Section 2) it is possible to encode the information including background knowledge more explicitly. After the learning phase it is possible to extract the actual rules from the neural network. They may yield the basis for human-understandable explanations.

6 Evaluation

In this section, we describe our evaluation and the achieved results on the COPA challenge as well as first experiments with the Story Cloze Test. Concerning the latter, the logical models are not yet conducted, thus we have only experiments with the natural language and machine learning part without using the deduction part of the system.

6.1 Cross Evaluation

The COPA challenge specifies 1,000 problems, 500 for training and 500 for testing. We evaluated our system using stratified 10-fold cross-evaluation, splitting the training set ten times into 450 training and 50 validation examples. As our training examples for the neural network are crafted pairing the premise with each of the answers, we have 900 training and 100 validation examples. After processing both training examples of a pair through the network, we got a likelihood for each answer belonging to the respective premise. The answer of a pair which got the higher likelihood is assumed to be the answer and chosen by the system.

The goal of the cross evaluation is to identify the parameters which might lead to a good performance on the test set. In Figure 7 we show the results on the cross-evaluation for different selected parameters. The green plots refer to the training set, while the blue plots show the respective validation set. Each data point refers to a cross-evaluation such that each boxplot represents a 10-fold cross-evaluation. The y-axis corresponds to the accuracy of our predictions and the x-axis gives the value of the respective parameter. The left plot describes different selection methods for the background knowledge, resulting in a different amount of knowledge available for the theorem prover. This in turn leads to bigger or smaller conducted logical models. *no bgk* refers to not using knowledge from Adimen-SUMO, but still from WordNet. *rec* refers to the recursion depth, while *tol* stands for the tolerance of the SInE selection. The right plot presents the amount of neurons used in the first starting layer in the neural network. The amount of neurons in the following layers are dependent on that number, as seen in Figure 6.

The results on the training sets are very good. They range from 82.6% to 100% with a mean of 98.4% and a low derivation of 3.6%. The accuracy on the validation sets ranges from 30% to 72% with a mean of 50.2% and a derivation of 7.7%. The experiments on the parameters we do not present here behave similarly. The accuracy of our system on the test set is not good, with an average of only 50%. Nevertheless we can observe a few indicators. For instance, concerning the background knowledge integrating more information does not improve the performance in our setting, however it lowers the variance of the performance.

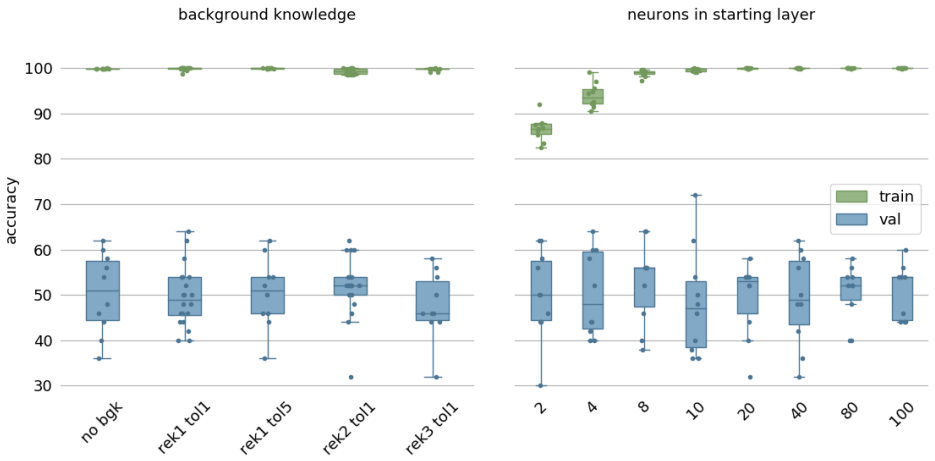

Figure 7. Results on the different parameters *background knowledge* and *neurons in the starting layer*. The green boxplots present the performance on the training set, while the blue boxplots present the performance on the validation set. Note that the total number of neurons of the whole network is several order of magnitudes higher than the number of neurons in the starting layer.

In the right diagram one can see that choosing two or four neurons in the starting layer, the network cannot learn the training data well, with scores of 82% to 98% and not 100%. Simultaneously, the accuracy of the validation data does not decrease in comparison to more neurons. This indicates that fewer neurons are better in generalization, probably because they do not overfit so easily. In addition there are models, which score on up to 70% on the validation data, indicating that our approach is theoretically capable of learning the right parameters.

As the training set in general is almost perfectly learned, we assume to have an overfitting problem. We tried to tackle this problem with constraints on the weights and dropout layers as presented in Table 2, however we did not get better results. We believe that we can tackle our problems with more training data, as the overall number of neurons in the tiniest possible network in the attentive reader approach is still one million. To tackle this size with 1,000 training examples seems to be impossible.

6.2 Performance and Discussion

Using the validation set and a fixed random seed, we identified the most promising parameters for both the background knowledge and the neural network, as described in Section 6.1 to calculate a best model. However, we did not yet conducted an exhaustive search of all parameter combinations. We choose the input to be a set erased of duplicates, a SInE selection recursion depth of 2, a SInE selection tolerance parameter of 1, 20 neurons in the starting layer, 40 epochs, a dropout rate of 0.4, and a learning rate of 0.002. With those parameters and a varying seed, we calculated an ensembled

model out of 7 models. We repeated this 30 times to get stable results ranging from 49.30% to 56.10% with a mean of 52.51%, i.e., only slightly above chance, and a low variance of 1.49%.

The baseline given for the COPA challenge is a PMI approach (Pointwise Mutual Information), as described in [6], scoring 58.8% [28]. When the challenge came out, first approaches scored 65.4% [14] in 2011, while the state-of-the-art approach from OpenAI currently scores 78.6% [25]. The latter uses an approach solely based on neural networks with pretraining techniques and an enormous amount of training data and thus lacks explainability. With the same setup as before we also did experiments using only the natural language as it is as input, neither with background information nor a logical model. On COPA we achieved then results ranging from 53.80% to 57.00% with a mean of 55.57%, and a variance of 0.85%. On the Story Cloze Test we achieved better results ranging from 69.05% to 71.51% with a mean of 70.17%, and a variance of 0.43%. The baseline here is 59.5% [21] which is an improvement of 17.93%. However, OpenAI already scores 86.5% [25].

So far, our results do not reach the performance of state-of-the-art systems. Nevertheless we can identify tendencies. The model approach on the COPA set with 52.51% accuracy works slightly worse in comparison to the natural language only approach with 55.57% accuracy. But this might be not a general trend. Also, the Story Cloze Test using natural language with 70.17% scores better than the COPA set using natural language with 55.57%. This might be either due to the bigger data set of 3,744 problems in comparison to 1,000 problems in COPA, or to the longer description part of four sentences instead of one. Taking a look at the OpenAI results indicate that indeed the Story Cloze Tests are easier to tackle, because they also score 7.9% better on this problems.

7 Summary

In this paper, we present first experiences in combining automated theorem provers with machine learning methods to solve commonsense reasoning problems. This combination is motivated by the fact that approaches based solely on machine learning cannot provide explanations for the decisions made by the system. The use of background knowledge in the form of ontologies suggests that this is achievable in our system. Finally we present an idea of how explanations can be generated.

Unfortunately, our first experiments did not lead to good results: We obtain an accuracy of 52.51% on the COPA test set, thus our approach is hardly better than guessing. Nevertheless, the accuracy on the training set is close to 100%. We believe that this mainly is caused by too few training data. As already said in Section 3.1 and Section 6.2, we are currently integrating the Story Cloze Test. For now, we only use them as natural language, but we are soon processing them into logical models. They consist of 3,744 problems and are therefore three times as many training examples as with the COPA challenge. They are an additional benchmark and can be used as a pretraining set for the COPA tasks, as they are similar in structure and inference. In addition, we consider to make use of unsupervised pretraining on continuous text (cf. [25]).

Another problem is the quality of the background knowledge. The background knowledge we are currently using contains mostly taxonomic knowledge which is pos-

sibly only of little help in the area of commonsense reasoning. Hence, in future work, we plan to integrate further sources of background knowledge like ConceptNet. They provide knowledge graphs representing factual knowledge as triplets of the form (s, p, o) (subject – predicate – object). For this, the machine learning procedure shall also be refined to deal with the structural information in knowledge graphs.

References

1. Álvez, J., Lucio, P., Rigau, G.: Adimen-SUMO: Reengineering an ontology for first-order reasoning. International Journal on Semantic Web and Information Systems (IJSWIS) **8**(4), 80–116 (2012)
2. Basile, V., Cabrio, E., Schon, C.: KNEWS: Using logical and lexical semantics to extract knowledge from natural language. In: Proceedings of the European Conference on Artificial Intelligence (ECAI) (2016)
3. Bender, M., Pelzer, B., Schon, C.: System description: E-KRHyper 1.4. In: Bonacina, M.P. (ed.) Automated Deduction – CADE-24: Proceedings of 24th International Conference on Automated Deduction. pp. 126–134. LNAI 7898, Springer, Lake Placid, NY, USA (2013)
4. Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. In: Neural networks: Tricks of the trade, pp. 437–478. Springer (2012)
5. Bos, J.: Is there a place for logic in recognizing textual entailment? Perspectives on Semantic Representations for Textual Inference **9** (2013)
6. Church, K.W., Hanks, P.: Word association norms, mutual information, and lexicography. Computational Linguistics **16**(1), 22–29 (1989)
7. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR – Computing Research Repository abs/1810.04805, Cornell University Library (2018), http://arxiv.org/abs/1810.04805
8. Diederich, J., Tickle, A.B., Geva, S.: Quo vadis? Reliable and practical rule extraction from neural networks. In: Koronacki, J., Ras, Z.W., Wierzchon, S.T., Kacprzyk, J. (eds.) Advances in Machine Learning I: Dedicated to the Memory of Professor Ryszard S. Michalski, Studies in Computational Intelligence, vol. 262, pp. 479–490. Springer (2010), http://doi.org/10.1007/978-3-642-05177-7_24
9. Doran, D., Schulz, S., Besold, T.R.: What does explainable AI really mean? A new conceptualization of perspectives. In: Besold, T.R., Kutz, O. (eds.) Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017, co-located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017). CEUR Workshop Proceedings, vol. 2071. CEUR-WS.org, Bari, Italy (2018), http://ceur-ws.org/Vol-2071/CExAIIA_2017_paper_2.pdf
10. Furbach, U., Schon, C., Stolzenburg, F., Weis, K.H., Wirth, C.P.: The RatioLog project: Rational extensions of logical reasoning. KI **29**(3), 271–277 (2015), http://link.springer.com/article/10.1007/s13218-015-0377-9
11. d'Avila Garcez, A.S., Broda, K., Gabbay, D.M.: Symbolic knowledge extraction from trained neural networks: A sound approach. Artificial Intelligence **125**(1-2), 155–207 (2001), http://doi.org/10.1016/S0004-3702(00)00077-1
12. d'Avila Garcez, A.S., Zaverucha, G.: The connectionist inductive learning and logic programming system. Applied Intelligence **11**(1), 59–77 (1999), http://doi.org/10.1023/A:1008328630915
13. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, London (2016), http://www.deeplearningbook.org

14. Gordon, A.S., Bejan, C.A., Sagae, K.: Commonsense causal reasoning using millions of personal stories. In: Twenty-Fifth AAAI Conference on Artificial Intelligence (2011)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997), http://doi.org/10.1162/neco.1997.9.8.1735
16. Hoder, K., Voronkov, A.: Sine qua non for large theory reasoning. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) Automated Deduction – CADE-23, Lecture Notes in Computer Science, vol. 6803, pp. 299–314. Springer Berlin Heidelberg (2011), http://doi.org/10.1007/978-3-642-22438-6_23
17. Holzinger, A.: Explainable AI (ex-AI). Informatik Spektrum **41**(2), 138–143 (2018), http://doi.org/10.1007/s00287-018-1102-5, Aktuelles Schlagwort, in German
18. Lenat, D.B.: CYC: A large-scale investment in knowledge infrastructure. Communications of the ACM **38**(11), 33–38 (1995)
19. Liu, H., Singh, P.: ConceptNet – a practical commonsense reasoning tool-kit. BT technology journal **22**(4), 211–226 (2004)
20. Miller, G.A.: WordNet: a lexical database for English. Communications of the ACM **38**(11), 39–41 (1995)
21. Mostafazadeh, N., Roth, M., Louis, A., Chambers, N., Allen, J.: LSDSem 2017 shared task: The story cloze test. In: Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics. pp. 46–51 (2017)
22. Mueller, E.T.: Commonsense Reasoning. Morgan Kaufmann, San Francisco, 2nd edn. (2014)
23. Niles, I., Pease, A.: Towards a standard upper ontology. In: Proceedings of the International Conference on Formal Ontology in Information Systems. pp. 2–9. ACM (2001)
24. Ostermann, S., Roth, M., Modi, A., Thater, S., Pinkal, M.: SemEval-2018 task 11: Machine comprehension using commonsense knowledge. In: Proceedings of The 12th International Workshop on Semantic Evaluation. pp. 747–757 (2018)
25. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. Tech. rep., Open AI (2018), http://openai.com/blog/language-unsupervised/
26. Ramos, J., et al.: Using TF-IDF to determine word relevance in document queries. In: Proceedings of the first instructional conference on machine learning. vol. 242, pp. 133–142. Piscataway, NJ, USA (2003)
27. Roemmele, M., Bejan, C.A., Gordon, A.S.: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In: AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning. pp. 90–95 (2011)
28. Roemmele, M., Bejan, C.A., Gordon, A.S.: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In: 2011 AAAI Spring Symposium Series (2011)
29. Siebert, S., Stolzenburg, F.: CoRg: Commonsense reasoning using a theorem prover and machine learning. In: Benzmüller, C., Parent, X., Steen, A. (eds.) Selected Student Contributions and Workshop Papers of LuxLogAI 2018. Kalpa Publications in Computing, vol. 10, pp. 20–26. EasyChair (2019), http://doi.org/10.29007/lt5p, Deduktionstreffen 2018, Luxembourg
30. Speer, R., Chin, J., Havasi, C.: ConceptNet 5.5: An open multilingual graph of general knowledge. In: AAAI Conference on Artificial Intelligence. pp. 4444–4451 (2017), http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14972
31. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: A large ontology from Wikipedia and WordNet. Web Semantics **6**(3), 203–217 (Sep 2008), http://doi.org/10.1016/j.websem.2008.06.001
32. Tan, M., Santos, C.d., Xiang, B., Zhou, B.: LSTM-based deep learning models for non-factoid answer selection. CoRR – Computing Research Repository abs/1511.04108, Cornell University Library (2015), http://arxiv.org/abs/1511.04108