BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

MACHINE COMPREHENSION USING COMMONSENSE KNOWLEDGE

STROJOVÉ POROZUMĚNÍ S POUŽITÍM ZNALOSTNÍ BÁZE

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **TOMÁŠ DANIŠ**
AUTOR PRÁCE

SUPERVISOR **Ing. MARTIN FAJČÍK**
VEDOUCÍ PRÁCE

BRNO 2018

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2018/2019

Bachelor's Thesis Specification

Student: **Daniš Tomáš**
Programme: Information Technology
Title: **Machine Comprehension Using Commonsense Knowledge**
Category: Speech and Natural Language Processing

Assignment:

1. Describe the machine comprehension problem.
2. Research current machine comprehension methods, that are able to incorporate commonsense knowledge information.
3. Choose and describe suitable method, describe how method differs from other related work.
4. Describe your evaluation process for the chosen method.
5. Describe available datasets for the problem.
6. Implement the method.
7. Evaluate the method.
8. Do an ablation study.
9. Compare achieved results with state-of-the-art.

Recommended literature:

- Ostermann, S. et al., 2018. SemEval-2018 Task 11: Machine Comprehension Using Commonsense Knowledge. In *Proceedings of The 12th International Workshop on Semantic Evaluation*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 747-757. Available at: http://aclweb.org/anthology/S18-1119

Requirements for the first semester:

- Complete items 1 to 4 of the assignment

Detailed formal requirements can be found at http://www.fit.vutbr.cz/info/szz/

Supervisor: **Fajčík Martin, Ing.**
Head of Department: Černocký Jan, doc. Dr. Ing.
Beginning of work: November 1, 2018
Submission deadline: May 15, 2019
Approval date: November 1, 2018

Abstract

In this thesis, the commonsense reasoning ability of modern neural systems is explored. The goal is to provide insight into the current state of research in this area and identify promising research directions. A state-of-the-art question-answering model has been implemented and experimented with in various scenarios. Unlike in older approaches, the model achieved comparable results with best available models for the target task without using any task-specific architecture. Furthermore, unintended statistical biases are discovered in a popular commonsense reasoning dataset which allow models to compute the correct answer even when it does not have sufficient information to do so. Based on these findings, recommendations and possible future research areas are suggested.

Abstrakt

V tejto práci je skumaná schopnosť používať zdravý rozum v moderných systémoch založených na neurónových sieťach. Zdravým rozumom je myslená schopnosť extrahovať z textu fakty, ktoré nie sú priamo spomenuté, ale implikuje ich situácia v texte. Cieľom práce je poskytnúť náhľad na súčasný stav výskumu v tejto oblasti a nájsť sľubné výskumné smery do budúcnosti. V práci je implementovaný jeden z najmodernejších modelov na odpovedanie na otázky a je ďalej použitý na experimenty v rôznych situáciách. Narozdiel od starších prístupov, tento model dosahuje porovnateľné výsledky s najlepšími známymi modelmi aj keď jeho architektúra neobsahuje žiadne prvky zamerané konkrétne na zlepšenie schopnosti zdravo uvažovať. Taktiež boli nájdené štatistické artefakty v populárnej sade dát s otázkami vyžadujúcimi zdravé uvažovanie. Tieto artefakty môžu byť použité štatistickými modelmi na nájdenie správnej odpovede aj v prípadoch, kedy by to nemalo byť možné. Na základe týchto zistení sú v práci poskytnuté odporúčania a návrhy pre výskum do budúcnosti.

Keywords

neural network, commonsense reasoning, commonsense knowledge, machine learning, natural language processing, question answering, knowledge base

Klíčová slova

neurónová sieť, uvažovanie zdravým rozumom, znalostná báza zdravého rozumu, strojové učenie, spracovanie prirodzeného jazyka, odpovedanie na otázky, znalostná báza.

Reference

DANIŠ, Tomáš. *Machine Comprehension Using Commonsense Knowledge.* Brno, 2018. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Martin Fajčík

Rozšířený abstrakt

Schopnosť používať zdravý rozum je považovaná za dôležitý milník v ceste za umelou inteligenciou podobnou človeku. Zdravým rozumom je myslená schopnosť extrahovať z textu fakty, ktoré nie sú priamo spomenuté, ale sú implikované situáciou v texte. Táto práca sa preto zameriava na prieskum súčasného stavu výskumu do modelov schopných používať zdravý rozum. Jej cieľ je poskytnúť náhlad na súčasnú situáciu a identifikovať sľubné smery pre výskum do budúcnosti. Práca poskytuje prehľad moderných metód používaných v spracovaní prirodzeného jazyka. Teoretická časť začína popisom základných princípov neurónových sietí a pokračuje prezentáciou moderných architektúr používaných na strojové porozumenie. Taktiež sú popísané verejne prístupné sady dát určené na trénovanie modelov na odpovedanie na otázky, ktoré vyžadujú zdravý rozum na zodpovedanie. Následne sú vysvetlené súčasne používané prístupy a ich spôsoby riešenia sú porovnané. Jedna z týchto architektúr dosahuje výsledky porovnateľné alebo presahujúce najlepšie dosiahnuté aj bez toho aby obsahovala externú bázu znalosti pre zdravý rozum. Namiesto toho bol tento model predtrénovaný všeobecnejšej úlohe predikcie slov na základe kontextu. Tento model bol v práci implementovaný, natrénovaný a experimentovalo sa s ním v rôznych situáciách v snahe poskytnúť náhlad na jeho schopnosti uvažovania zdravým rozumom. Pri experimentoch bola použitá sada dát MCScript, ktorá obsahuje príbehy popisujúce každodenné udalosti z pohľadu prvej osoby a otázky o týchto udalostiach, každá s dvomi kandidátnymi odpoveďami. Táto sada dát sa vyznačuje tým, že nie všetky odpovede na otázky sa nachádzajú v texte. Pri niektorých otázkach je potrebné použiť zdravý rozum na ich zodpovedanie. V práci bolo spravených päť hlavných experimentov. V každom bol natrénovaný model so špecifickými parametrami danými experimentom a vyhodnotený pomocou metriky presnosti. Taktiež sa experimenty vyhodnotili hlbšiou analýzou ich výsledkov. V prvom experimente sa natrénoval model na už spomenutej sade dát a vyhodnotili sa výsledky, hlavne s ohľadom na otázky vyžadujúce zdravý rozum. Účelom tohto experimentu bolo vyhodnotiť schopnosť súčasných modelov zdravo uvažovať. Zistilo sa, že súčasné modely sú schopné zdravého uvažovania aj bez explicitného tréningu tejto schopnosti. Otázky vyžadujúce zdravý rozum síce boli o niečo ťažšie na zodpovedanie, ale hlavným faktorom limitujúcim presnosť modelu bolo celkové porozumenie textu, nie zdravý rozum. Na získanie lepšej predstavy čoho sú dnešné modely schopné, v druhom experimente sa najprv model predtrénoval na príbuznej sade dát SWAG, ktorá taktiež vyžaduje použitie zdravého rozumu na zodpovedanie. Následne bol tento predtrénovaný model natrénovaný na sade MCScript. Účelom tohto experimentu bolo otestovať, či takéto predtrénovanie zlepší schopnost modelu zdravo uvažovať. Ukázalo sa, že sa tak nestalo a teda predtrénovanie na príbuzných úlohách nepomáha pre už predtrénované modely. Taktiež sa dospelo k záveru, že zdravý rozum je jav vznikajúci pri všeobecnej schopnosti porozumenia textu. Pre ďalší výskum zdravého rozumu sa teda odporúča skúmať metódy, ktoré sa sútredia na porozumenie textu na hlbšej úrovni. V posledných troch experimentoch sa trénoval model na sade MCScript, avšak v každom z experimentov chýbala modelu istá časť vstupu. V treťom experimente modelu nebol poskytnutý dokument s informáciami, vo štvrtom otázka a v piatok ani dokument ani otázka. Bolo očakávané, že model bez týchto informácií nebude schopný vybrať správnu odpoveď, avšak opak bol pravdou. Zatiaľ čo v prvom experimente dosiahol model presnosť 88.38%, pri absencií dokumentu stále dosahoval presnosti 78.08%. Bolo zistené, že tento jav je čiastočne vysvetliteľný tým, že model správne odpovedá na otázky vyžadujúce zdravý rozum a absencia dokumentu túto schopnosť nenarušila. Avšak, toto vysvetlenie nedokázalo vysvetliť takú veľkú deviáciu od očakávaných 50%. Ešte viac prekvapujúci výsledok mal piaty experiment. Keď modelu boli na vstupe dané iba 2 kandidátne

odpovede, stále dosahoval presnosti 72.83%. Na základe týchto výsledkov sa dospelo k záveru, že sada dát MCScript obsahuje štatistické artefakty, na základe ktorých sú modely schopné vybrať správnu odpoveď aj v situáciách kedy by to nemalo byť možné. Tieto artefakty zňemožnujú objektívne vyhodnotiť schopnosť modelu vykonávať cieľovú úlohu. Na základe týchto zistení sa odporúča vykonávať analýzu na nájdenie štatistických artefaktov pre všetky sady dát používané v strojovom učení.

Machine Comprehension Using Commonsense Knowledge

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of Ing. Martin Fajčík. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Tomáš Daniš
May 14, 2019

Acknowledgements

I would like to thank my supervisor Ing. Martin Fajčík for his guidance, constructive feedback and help with the thesis.

Contents

Chapter 1

Introduction

In recent years, the rise of artificial intelligence techniques has allowed us to solve problems that were impossible before. These algorithms often outperform even humans. However, despite their strong performance, the algorithms are often easily confused and make a mistake where a human never would. Consider the sentence "The waitress brought my mother the food and she ate it all right away". Who does "she" refer to, the mother or the waitress? It is not possible to decide from the sentence alone without any context. A text processing algorithm would struggle with this sentence. However, it is obvious to any human that "she" in fact refers to the mother. It makes no sense for the waitress to eat the food she just brought to a customer. People would call it common sense. And that is exactly what a lot of modern algorithms lack.

Researchers have tried to instil common sense into algorithms before, but as the concept of common sense is hard to grasp, these efforts had not been met with much success. Only recently has there been some progress in this field. However, in order to build machines that can do human tasks independently without supervision, it is of utmost importance that they have common sense. The author believes it is a major limiting factor in advancing artificial intelligence and thus was motivated to choose this topic for the thesis.

The goal of this work is to explore the current state of research into common sense reasoning techniques, identify promising approaches and provide suggestions for future research. The rest of this thesis is organised as follows: Chapter 2 contains the overview of techniques used in natural language processing and common sense reasoning. It starts off with an explanation of fundamental techniques common to a lot of machine learning approaches and then continues to describe concepts specific to natural language processing. Concepts used in machine comprehension are described in more detail as well. Chapter 3 describes publicly available datasets for machine comprehension tasks which require some form of common sense reasoning. These datasets are used for training later in the thesis. Chapter 4 contains a portfolio of current models capable of common sense reasoning. Architectures with various approaches to the problem were chosen to get an overview of the current state of the field. Chapter 5 describes models used in the experiments in this thesis and the nature of the experiments performed. The experiments are intended to provide insight into the model's common sense reasoning capability and find its weaknesses or strengths. Chapter 6 presents the results of the experiments and provides a deeper analysis of the results of each experiment along with theories explaining them. Chapter 7 draws conclusions from the results reported in the previous chapter and fulfils the goal of the thesis by providing recommendations on future research in common sense reasoning and

natural language processing. These recommendations are based on the results found in the thesis.

Chapter 2

Natural language processing

Natural language processing (NLP) is a branch of artificial intelligence that studies how to process and analyse natural language using computers. It is concerned with analysing both the syntax and the semantics of the given natural language. Main goals of NLP include using computers to understand and speak natural languages at the human level.

In this chapter, the reader is introduced to the techniques currently used in NLP which are relevant to this thesis. First, the fundamental concepts on which more advanced methods are built on are described. Afterwards, the techniques specifically used in statistical question answering which are applied in this thesis are explained.

2.1 Neural networks

The purpose of this section is to introduce the reader to the basics of neural networks. We describe what they are, what is their purpose and how they work. We start with an explanation of the principle of a single artificial neuron and then move on to networks with multiple neurons. Finally, an overview of how neural networks learn is provided.

2.1.1 Artificial neuron

Information in this section were taken from [2].

The initial motivation for building neural networks was modelling biological neural systems like the brain. A neuron is a basic computational unit of the brain. It is a cell which has a number of connections called dendrites through which it receives input signals and a single connection called an axon through which it produces output signals. The axon branches and connects to dendrites of other neurons. In the biological model, the dendrites carry the signal to the cell body where they all get summed. If the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon.

An artificial neuron is modeled after the description above. It is a computational unit with a set of real-valued inputs $x_1, x_2, x_3 ... x_n$ and an output y. Each input x_i also has a corresponding weight w_i which determines the influence of that input on the output. The output of a neuron is the weighted sum of its inputs to which an activation function f is applied. An activation function models the threshold for neuron firing. Figure 2.1 and 2.2 illustrates both models.

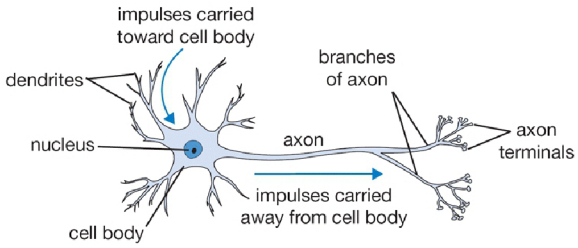

Figure 2.1: Biological neuron [2]

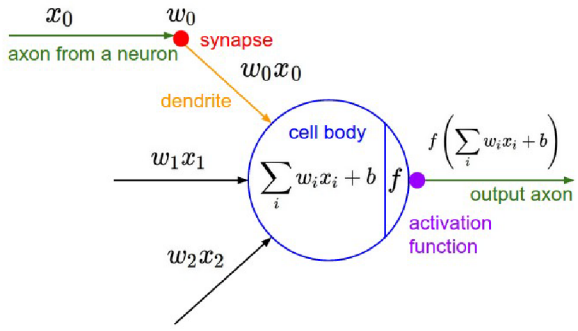

Figure 2.2: Artificial neuron [2]

There are multiple possibilities for an activation function. A common choice in the past was the sigmoid function $\sigma = \frac{1}{1+e^{-x}}$ (figure 2.3), but it is not used as often anymore due to its limitations. Other possibilities are the *tanh* function (Figure 2.4) or Rectified Linear Unit $ReLU = max(0, x)$ (Figure 2.5).

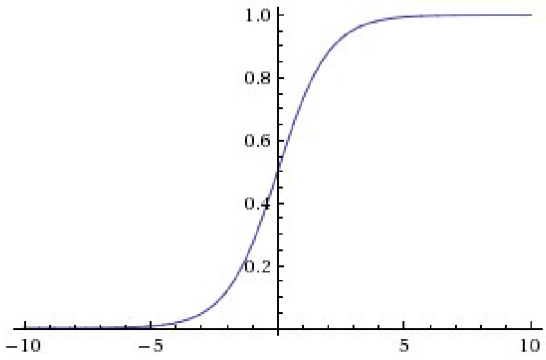

Figure 2.3: Sigmoid function [2]

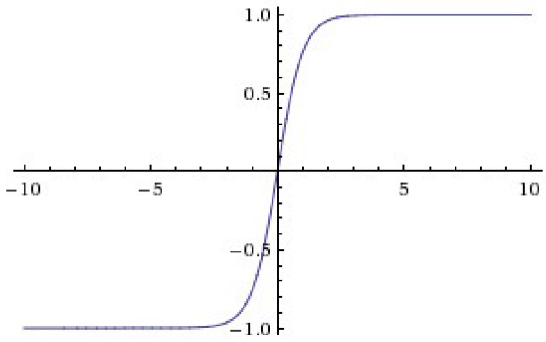

Figure 2.4: Tanh function [2]

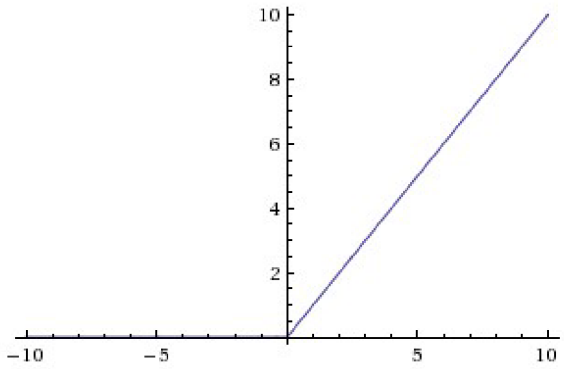

Figure 2.5: ReLU function [2]

The principle of an artificial neuron makes it appropriate for use as a binary classifier. Depending on its weights, it outputs large values for some inputs and small values for others. By thresholding the output, we can classify the inputs into 2 classes. For example, the output of a neuron with a sigmoid activation function $\sigma(\sum_i x_i w_i + b)$ can be interpreted as the probability of belonging to a class y $P(y = 1|x_1; x_2; ..; x_n; w_1; w_2; ..)$. In this case, as the sigmoid function is restricted to the interval $(0, 1)$, the threshold for classification would be 0.5.

2.1.2 Neural networks

Information in this section were taken from [2].

By connecting multiple neurons (binary classifiers), we can detect more patterns in the input data and solve more complex classification problems. In summary, neural networks are acyclic directed graphs of neurons. Outputs of some neurons are connected to inputs of other neurons and the calculation is propagated through the network. By convention, neural networks are organised into layers where generally the input of a layer is the output of a previous layer. However, there are layers which are interconnected in other ways. The most common type of layer is the fully-connected layer where all neurons between adjacent layers are connected with each other, but neurons within a single layer have no connections between themselves.

Figure 2.6 illustrates examples of fully connected layers.

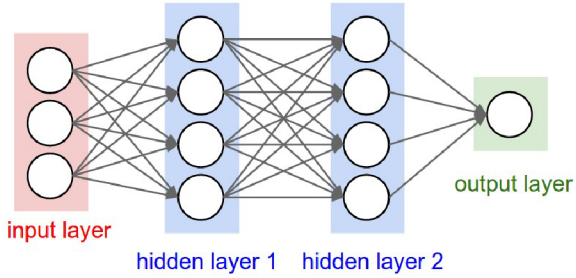

Figure 2.6: 3-layer fully connected network [2]

2.1.3 Training neural networks for classification

In order for a neural network to perform the desired classification task, it needs to be trained to do so. Training takes place using annotated data. Annotated data is a set of inputs with a corresponding correct class. By feeding this data to the network, it can adjust the weights on its neurons so that it predicts correct classes on the training data. To facilitate training, a function which measures the difference between the correct classes and the predicted classes based on the networks weights is defined. This function $L(W)$ is called a loss function with parameter W being a vector of all weights of all neurons in the network. The goal of training is to minimise $L(W)$ as that will yield the most correctly predicted classes. The training procedure is therefore a minimalisation problem. However, with a large number of weights in the network, it is not feasible to solve it analytically. For this reason, numerical methods such as gradient descent are used. However, gradient descent requires the gradients of the loss function with respect to all its parameters as its input. Computing these gradients is not a trivial task. To compute them, an efficient algorithm called backpropagation has been developed. However, its description is not necessary for understanding the contents of this thesis and therefore, it is not discussed here.

2.2 Language representation in neural networks

In this section, the reader is introduced to the concept of word embeddings. The concept of word embeddings, the motivation for their usage and their properties are explained. In the following subsection, word embedding models relevant to this thesis are described in greater detail.

In order to process words using a neural network, they need to be converted into an appropriate representation. As neural network input is a vector, the words need to be transformed into vectors. A naive solution would be constructing a vocalbulary out of all the words we want to process and using one-hot encoding to convert them into vector representations. Each word would be an L-dimensional vector which is all zeros except a single element which would be one. The non-zero element's position would depend on the position of the word in the vocalbulary. L is the size of the vocabulary. Figure 2.7 illustrates an example of this representation.

This approach has several problems. Firstly, the vectors are very sparse. This makes it hard for the network the learn as only one neuron is activated with each word. Secondly, this representation doesn't capture any relations between the words. The vectors of all the words are perpendicular to each other, making their dot product always 0. This would tell

us that words „good“, „better“ and „ship“ are totally different from each other, which is obviously not true.

A better approach is using word embeddings. Word embeddings are dense vectors representing words. They are shorter when compared to one-hot vectors, but every element carries information instead of just one. Individual elements are in most cases not human interpretable, but the vector as a whole captures 2 important pieces of semantic information: semantic similarity between words, which is a measure of how similar is the meaning of different words, and semantic relatedness, which is a measure of how related different words are. Words are said to be related if their meaning ties to similar concepts, even if the words themselves are not synonymous. For example, „bus“ and „car“ are semantically similar, but both are also semantically related to „driving“. Figure 2.8 shows an example of word embeddings.

Thanks to these properties, word embeddings help neural networks learn more complex relationships and patterns in the text, allowing them to be used for more complicated tasks. At the time of writing, there is a great variety of word embeddings for the English language. Method of obtaining them is different for each variant. Examples include word2vec [13] or GLoVe [17].

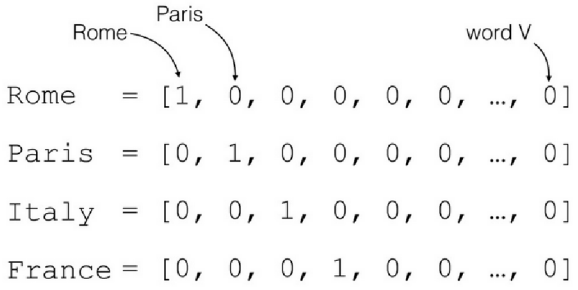

Figure 2.7: Illustration of one-hot vectors for several words. [4]

	0	1	2	3	4	5	6	7	8	9	...	290	291	292	
fox	-0.348680	-0.077720	0.177750	-0.094953	-0.452890	0.237790	0.209440	0.037886	0.035064	0.899010	...	-0.283050	0.270240	-0.654800	0.10
ham	-0.773320	-0.282540	0.580760	0.841480	0.258540	0.585210	-0.021890	-0.463680	0.139070	0.658720	...	0.464470	0.481400	-0.829200	0.35
brown	-0.374120	-0.076264	0.109260	0.186620	0.029943	0.182700	-0.631980	0.133060	-0.128980	0.603430	...	-0.015404	0.392890	-0.034826	-0.72
beautiful	0.171200	0.534390	-0.348540	-0.097234	0.101800	-0.170860	0.295650	-0.041816	-0.516550	2.117200	...	-0.285540	0.104670	0.126310	0.12
jumps	-0.334840	0.215990	-0.350440	-0.260020	0.411070	0.154010	-0.386110	0.206380	0.386700	1.460500	...	-0.107030	-0.279480	-0.186200	-0.54
eggs	-0.417810	-0.035192	-0.126150	-0.215930	-0.669740	0.513250	-0.797090	-0.068611	0.634660	1.256300	...	-0.232860	-0.139740	-0.681080	-0.37
beans	-0.423290	-0.264500	0.200870	0.082187	0.066944	1.027600	-0.989140	-0.259950	0.145960	0.766450	...	0.048760	0.351680	-0.786260	-0.36
sky	0.312550	-0.303080	0.019587	-0.354940	0.100180	-0.141530	-0.514270	0.886110	-0.530540	1.556600	...	-0.667050	0.279110	0.500970	-0.27
bacon	-0.430730	-0.016025	0.484620	0.101390	-0.299200	0.761820	-0.353130	-0.325290	0.156730	0.873210	...	0.304240	0.413440	-0.540730	-0.03
breakfast	0.073378	0.227670	0.208420	-0.456790	-0.078219	0.601960	-0.024494	-0.467980	0.054627	2.283700	...	0.647710	0.373820	0.019931	-0.03
toast	0.130740	-0.193730	0.253270	0.090102	-0.272580	-0.030571	0.096945	-0.115060	0.484000	0.848380	...	0.142080	0.481910	0.045167	0.05
today	-0.156570	0.594890	-0.031445	-0.077586	0.278630	-0.509210	-0.066350	-0.081890	-0.047986	2.803600	...	-0.326580	-0.413380	0.367910	-0.26
blue	0.129450	0.036518	0.032298	-0.060034	0.399840	-0.103020	-0.507880	0.076630	-0.422920	0.815730	...	-0.501280	0.169010	0.548250	-0.31
green	-0.072368	0.233200	0.137260	-0.156630	0.248440	0.349870	-0.241700	-0.091426	-0.530150	1.341300	...	-0.405170	0.243570	0.437300	-0.46
kings	0.259230	-0.854690	0.360010	-0.642000	0.568530	-0.321420	0.173250	0.133030	-0.089720	1.528600	...	-0.470090	0.063743	-0.545210	-0.19
dog	-0.057120	0.052685	0.003026	-0.048517	0.007043	0.041856	-0.024704	-0.039783	0.009614	0.308416	...	0.003257	-0.036864	-0.043878	0.00
sausages	-0.174290	-0.064869	-0.046976	0.287420	-0.128150	0.647630	0.056315	-0.240440	-0.025094	0.502220	...	0.302240	0.195470	-0.653980	-0.29
lazy	-0.353320	-0.299710	-0.176230	-0.321940	-0.385640	0.586110	0.411160	-0.418680	0.073093	1.486500	...	0.402310	-0.038554	-0.288670	-0.24
love	0.139490	0.534530	-0.252470	-0.125650	0.048748	0.152440	0.199060	-0.065970	0.128830	2.055900	...	-0.124380	0.178440	-0.099469	0.00
quick	-0.445630	0.191510	-0.249210	0.465900	0.161950	0.212780	-0.046480	0.021170	0.417660	1.686900	...	-0.329460	0.421860	-0.039543	0.15

20 rows × 300 columns

Figure 2.8: Example of word embeddings for several words. [5]

2.2.1 WordPiece

Information in this section was taken from [24].

An approach to representing text as meaningful vectors similar to word embeddings is using sub-word embeddings. In this case, instead of having a vector for each word in the vocabulary, words are decomposed into their constituent parts. Embeddings are then found for these parts and in a neural network, a word is represented by the vectors of its constituent parts. In other words, some kind of sub-word units serve as the vocabulary for the neural network. How exactly to split the words and what vectors to assign them are problems each sub-word embedding methods solves differently. The advantage of these approaches is that they can construct embeddings for new and unknown words. These representations might also mirror the structure of real words more closely. For example, in a lot of languages, words have prefixes and suffixes that slightly alter the meaning of the word, but do not change it completely. Having individual embeddings for these word modifiers might allow the vectors to capture these relationships. WordPiece is an approach falling into this category that is described in this section as it is used later in the thesis.

To determine how to split words into their constituent parts (further referred to as tokens), the WordPiece algorithm needs the desired number of tokens D to keep, e.g the size of the vocabulary. With this, the objective of the algorithm is to find such D tokens, that given a text training corpus, the number of tokens representing the corpus when all words in it are split is minimal. This objective forces the algorithm to find the most most common tokens, appearing in the largest number of words possible. Such tokens are more likely to be meaningful.

The second task of the algorithm is to find vector representations for the found tokens. This is accomplished through a language modelling task. In this task, the objective is to predict a word in a text, given the sequence of words that precede it. In this case, tokens are

used in the task instead of words. Thanks to this, representations are found which capture relationships between the tokens such as semantic similarity and semantic relatedness.

2.3 Recurrent neural networks

In this section, recurrent neural networks (RNNs) are described. RNNs are a special type of a neural network used in natural language processing. First, the general concept is introduced. Subsequently, RNNs used in this thesis are looked at in more detail.

2.3.1 General model

Recurrent neural networks share the same structure as neural networks described in section 2.1, except each layer also has an internal state, which captures information about previous inputs of the layer. This allows the network to take past inputs into consideration when processing current inputs. This internal state is often called a hidden state and effectively serves as the network's memory. Recurrent neural networks are widely used in NLP as they allow temporal depedencies in the data, like context, to be captured. An RNN is illustrated in figure 2.9.

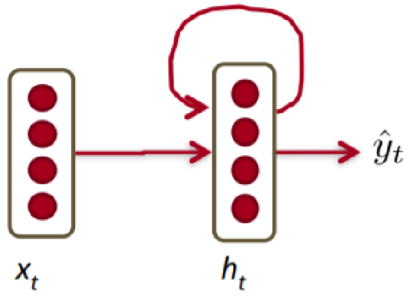

Figure 2.9: A simple reccurent neural network. [3]

The hidden state is realized as a vector. At each input, a new hidden state is calculated from the input and the previous hidden state. Output of the network is then calculated from this new hidden state. The equations 2.1 and 2.2 show the formulas for a general RNN. [3]

$$h_t = \sigma(W^{(hh)}h_{t-1} + W^{(hx)}h_t) \tag{2.1}$$

$$y_t = softmax(W^{(S)}h_t) \tag{2.2}$$

However, RNNs suffer from a range of problems. The hidden state cannot capture all the information about previous inputs due to its limited size. This leads to the RNN forgetting information from inputs further in the past, which makes processing of longer texts infeasible. This issue has motivated researchers in development of more advanced RNNs like the Gated Recurrent Unit [9], which we will now introduce.

2.3.2 Gated Recurrent Unit (GRU)

Gated recurrent unit (GRU) [9] is an advanced RNN that adds another layer of transformations between the input and the output. Instead of computing the new hidden state directly from the input and the previous states, it first computes so-called gates. Gates help the network determine the importance of current and past information so that it can save the most relevant information in the new hidden state.

Specifically, GRU computes a reset gate r and an update gate z. The reset gate controls how much information from the past is retained for future use. The update gate controls how much information from the past should be used to compute the current output. The equations 2.3 and 2.4 describe how to calculate the gate values.

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \tag{2.3}$$

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \tag{2.4}$$

The gate vectors are used in the hidden state computation as shown in equations 2.5 and 2.6.

$$h'_t = tanh(Wx_t + r_t \cdot U^h_{t-1}) \tag{2.5}$$

$$h_t = z_t \cdot h_{t-1} + (1 - z) \cdot h'_t \tag{2.6}$$

As can be seen from the formulas, when reset gate values are close to 0, the network drops past information and mostly saves information from the current input. Similarly, when the update gate values are close to 1, information from the past has a big part in computing the new hidden state.

2.4 Attention mechanism

In this section, the attention mechanism and its uses are described. First, the general definition of attention is provided and then its variants relevant to the thesis are explored.

2.4.1 General attention

According to [22], an attention function takes as input a set of key-value pairs and a query, all of which are vectors. A compatibility function is computed between the query and all given keys. This compability function outputs a score which signifies the relevance of the value corresponding to the key with regards to the query. The output of the function is a weighted sum of all the values with the scores computed by the compability function as weights. [22] define attention as:

To provide some intuition on the definition, attention can be used to identify relevant data with respect to some query. When the compatibility function of the query with each key is computed, keys corresponding to important values with respect to the query will get high scores while unimportant values will get low scores. When the output is computed as a sum of the values weighted by the scores, the more important values will be a bigger part of the output. We say that we are paying attention to these values.

The general advantage of attention mechanism is that it can look at the entire input at once and choose the important passages. This is in contrast with RNNs which can only work with the current input and the hidden state. In practice, the hidden state is unable

to capture past information about inputs deep in the past, which leads to loss of context. Attention helps alleviate this issue. Attention is illustrated in figure 2.10.

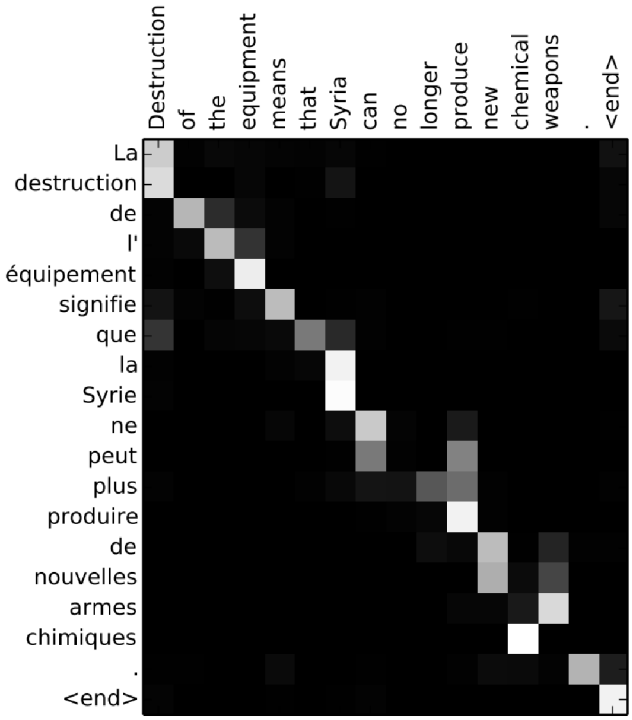

Figure 2.10: An illustration of attention on machine translation task. Matrix shows which in french (rows) were paid attention when generating the English sentence (columns). [1]

2.4.2 Self-attention

Self-attention is an attention mechanism where the all the keys, values and the query are from the same sequence. It is used to compute a new representation of this sequence. The query is always a single element from the sequence while the keys and values are the entire sequence. Output of this attention is a new representation of the element that was the query. By progressively choosing every element as the query, a new representation of the whole sequence is obtained.

2.5 Transformer encoder

Transformer encoder [22] is a neural architecture which is a combination of attention mechanisms and fully connected feed-forward network. The attention mechanism allows it to learn dependencies in arbitrarily distant positions. Its architecture also allows for easy parallelization during training.The remainder of this chapter describes the architecture of a single transformer encoder unit which is illustrated in Figure 2.11.

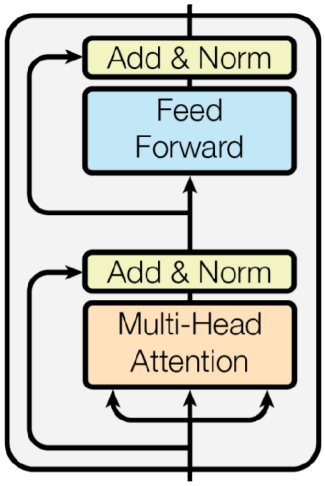

Figure 2.11: An architecture of a transformer encoder unit. [22]

Each transformer encoder unit is composed of two sub-layers. The first is a multi-head self-attention layer, described later in this section, with the second one being a fully-connected feed-forward neural network. Furthemore, there are also residual connections around both sub-layers, followed by layer normalization. Residual connections connect the unprocessed input to the output of the sub-layer and add them. Therefore, the output of each sub-layer is defined by $LayerNorm(x + Sublayer(x))$ where x is the input and $Sublayer(x)$ is the function implemented by the sub-layer.

The multi-head attention sublayer is a modification of the self-attention mechanism as described in section 2.4.2. The keys, values and query are all transformed using three different linear layers. Afterwards, self-attention is applied to the sequence. The used key-query compabatibility function is a simple dot product. A modification to the mechanism lies in the fact that the attention scores are divided by $\sqrt{d}$ where d is the dimension of the input vectors. This is done to decrease variance of the resultant dot product, which helps with training as for some activation functions, gradients are biggest around values close to zero. The described transformation is applied to the same input sequence multiple times in parallel, but with different linear layer weights each time. The rationale is that thanks to different transformations, the attention mechanism is able to attend to multiple aspects of the input sequence, capturing its characteristics more robustly. Before applying the softmax function to get the attention scores, it is possible to use a mask to mask out some of the query-key scores, essentially assigning them the score of zero. This can be useful for some tasks, but it is primarily used in other variations of the transformer architecture. The output from each of the attentions is concatenated and transformed using a linear layer. 2.12 along with the equations 2.7, 2.8 and 2.9 illustrate this sublayer.

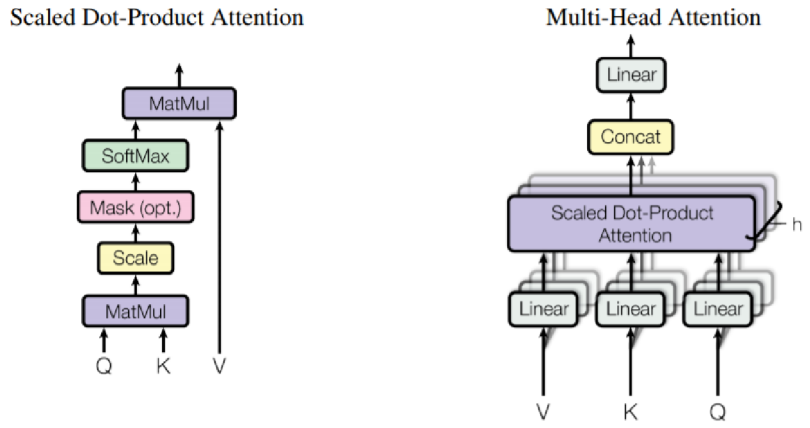

Figure 2.12: An illustration of multi-head attention. [22]

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d}})V \tag{2.7}$$

$$MultiHead(Q, K, V) = Concat(head_1, head_2, ..)W^O \tag{2.8}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.9}$$

The feed forward sublayer is a simple fully-connected layer network with 2 layers as described in section 2.1. The output of the sublayer is described by the equation 2.10.

$$FFN(x) = max(0, xW_1 + b1)W_2 + b_2 \tag{2.10}$$

2.6 Bidirectional Encoder Representations from Transformers (BERT)

BERT [11] is a neural architecture achieving state-of-the-art results in a wide range of NLP problems. Structure-wise, it consists of L transformer encoder blocks stacked on top of each other, with the result of the final block being the output of BERT.

The input of BERT consists of the following:

- A „[CLS]“ token. Every input sequence of BERT begins with this token. Its purpose is to have the corresponding output token serve as an aggregate input sequence representation and use it for classification tasks.
- The sequence itself. In the case of sequence pairs, for example [Question, Answer], they are packed together with a „[SEP]“ token between them.Figure 2.14 illustrates the high-level architecture of BERT and its input.

BERT uses the WordPiece embeddings described in section 2.2.1. However, to get the final input representation, two other kinds embeddings are added to the WordPiece vectors:

- Segment embeddings. When working with sequence pairs, a learned segment embedding A is added to every token of the first sequence and a learned segment embedding B is added to every token of the second sequence. When the input consists of only one sequence, only A embeddings are used.

- Positional embeddings as required due to using the transformer architecture.

An illustration of an example input can be found in Figure 2.13

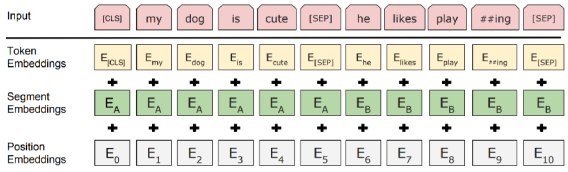

Figure 2.13: An illustration of BERT input format. [11]

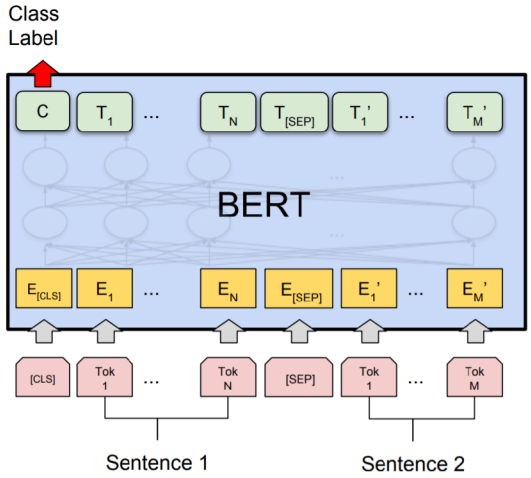

Figure 2.14: BERT architecture and input. [11]

However, the main advantage of BERT does not lie in the architecture itself, but in using pre-trained BERT models. Pretraining is a process where a model is trained on a different task from the target task. The parameters learned from training are then used as initial parameters for the model of the target task. The rationale behind this approach is that pre-training allows the model to learn task-independent language features which can be leveraged by the target task model. This technique has been shown to improve performance in a wide range of NLP problems. [19]. BERT is pre-trained on 2 tasks:

- Masked language modelling. The task of the model in language modelling is to predict a given word based on its surrounding words. BERT pre-trains a bidirectional language model, taking into account words before and after the given word. However, in a multi-layer bidirectional language model, it is possible a layer asked to predict a word could receive information about it from the representations of the surrounding words outputted by the previous layer. To avoid this problem, 15% of words are replaced with a „[MASK]“ token and the model is only asked to predict those. Pre-training on this task allows BERT to learn general features of the language which can aid in any NLP task.

- Next sentence prediction. The input consists of 2 sentences and the task of the model is to say whether the second sentence follows the first in a text. Pre-training on this

task allows the model to learn relationships between sentences, helping on sentence-level tasks such as question answering.

2.7 Hyper-parameter optimization algorithms

Neural architecture have various hyperparameters such as the number of neurons in a layer, training time or learning rate. It has been shown that varying these parameters can have huge effects on the performance of the model, ranging from chance to state-of-the-art results. [10]. Therefore, it is vital optimize these parameters use a method to its full potential. In this chapter, techniques for hyperparameter optimization are described.

2.7.1 Grid search

The simplest technique for hyperparameter optimization is grid search. When using grid search, a list of possible values for each hyperparameter is supplied by the user. The algorithm tries every valid combination of hyperparameters and returns the set which performs the best. This approach has several disadvantages. Firstly, the number of all combinations can be extremely huge, making it computationally impossible to do an exhaustive search. Secondly, the algorithm cannot deal with parameters which are expected to be in a continuous range of values. These drawbacks have led to the development of alternatives techniques, some of which are described below.

2.7.2 Random search

In random search, the hyperparameter space is once again given by the user. However, compared to grid search, the space can be continuous and the algorithm will sample the continous range when running trials. Furthermore, a probability distribution which to sample can also be specified, allowing more refined control of which hyperparameter values to try. Random search runs for a given number of iterations and in each, it samples each hyperparameter from the distribution it has been given for it and tries those values as the hyperparameter values. If the number of iterations is big enough, random search can find hyperparameter values which are close to the optimum. However, this method does not utilize any heuristics from previous runs, making the search inefficient and trials often redundant. It is also possible to get unlucky and not find a good set of hyperparameters.

2.7.3 Hyperopt

Hyperopt [7] is a tool for optimizing over search spaces with real-valued, discrete, and conditional dimensions. It uses algorithms with heuristics which consider previously tried hyperparameter values and the corresponding results to how the individual parameters affect the performance. With this information, it is possible to suggest a hyperparameter values which are more likely to yield greater performance. One of these algorithms, the one used for hyperparameter optimization of models in this thesis, is Tree of Parzen Estimators [6]. However, its description is beyond the scope of this thesis and the curious reader is referred to the original paper cited.

Chapter 3

Datasets for statistical question answering

In this section, we describe publicly available datasets for question answering which are of interest to us. The chosen datasets contain questions which often cannot be answered from the text alone, but also require some commonsense knowledge about the world.

3.1 MCScript

MCScript [14] is a large dataset of narrative texts and question about these texts. These texts describe every day events and activities such as going to the restaurant or taking a shower from a first person perspective in chronological order. The reason for this format is that the dataset is intended to test script knowledge. Script knowledge is knowledge most modern humans have about how certain everyday scenarios, like going to the restaurant, usually play out. In the restaurant example, the expected scenario is entering the restaurant, getting seated, ordering food, waiting, eating, paying and leaving. As this is considered common knowledge, a lot of information in the texts describing these scenarios is not explicitly stated, but is implied by the situation. Therefore, answers to some question do not have to be stated in the text, but a human could answer them correctly anyway. This format makes it possible to what degree do machine comprehension system possess script knowledge, which is the dataset's purpose.

Each text has several question associated with it, each with 2 answer candidates and exactly one correct answer. An example can be seen in Figure 3.1. The dataset was published in 2018 and contains 13939 questions in total. 27.46% of these are marked as requiring commonsense knowledge to answer correctly. 50.2% of the questions have the first answer as the correct one, meaning the dataset is balanced class-wise. The dataset comes split into training, development and test sets whose proportions are respectively 70%, 10% and 20% of the dataset.

An analysis of the type of questions in the dataset has been performed. The motivation behind this is to better understand the kind of reasoning necessary to answer them and to evaluate performance of the models based on question type later in the thesis. A pie chart depicting question type distribution can be be seen in Figure 3.2.

T I wanted to plant a tree. I went to the home and garden store and picked a nice oak. Afterwards, I planted it in my garden.

Q1 What was used to dig the hole?
a. a shovel b. his bare hands

Q2 When did he plant the tree?
a. after watering it b. after taking it home

Figure 3.1: An example from the MCScript dataset [14]

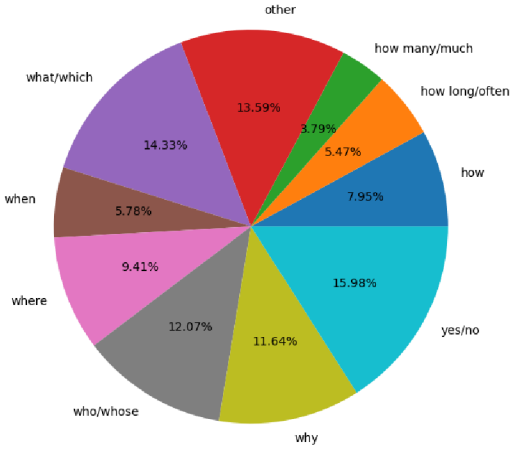

Figure 3.2: Distribution of question types in the MCScript dataset

3.1.1 Test set

This section provides a separate analysis of the test set of the MCScript dataset. This dataset will be used for evaluation of the models later in the thesis which is why its contents are of particular interest.

The test sets contains 2797 questions of which 25.85% require commonsense knowledge to answer. 50.77% of the questions have the first answer as the correct one. A subject analysis can be found in Figure 3.3. It can be observed that both the proportion of commonsense question and the question types characteristics are similar to the ones in the whole dataset. This is a good sign as it implies the test set is a representative sample from the dataset.

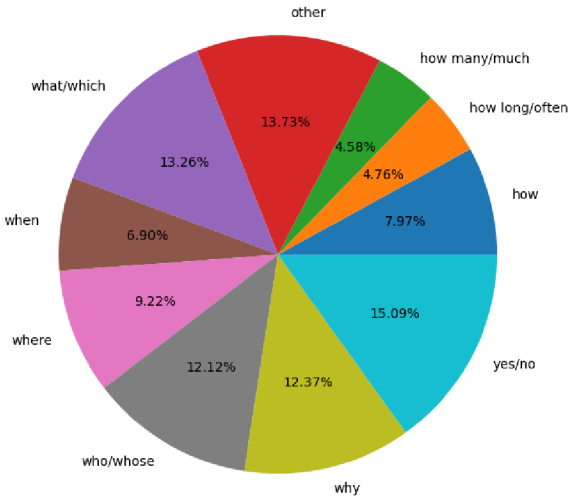

Figure 3.3: Distribution of question types in the test set of the MCScript dataset

3.2 SWAG

SWAG [25] is a dataset containing descriptions of events in a video. Each item in the dataset consists of a sentence describing the situation. The beginning of the next sentence is also provided, along with 4 possible continuations. However, only one of these is correct. The situations described are simple enough so that humans do not require to see the video to be able to choose the correct continuation as it is usually natural given the setting. Therefore, this dataset is suitable for testing machine comprehension system's commonsense reasoning ability. An example of an item from the dataset can be seen in Figure 3.4.

SWAG was published in 2018 and contains 113k multiple-choice question split into training, development and test sets with respective sizes of 73k, 20k and 20k. To choose incorrect candidate answers for each question, the authors of the dataset used a technique called Adversarial Filtering to ensure that all 4 options are equally likely to be selected by a system which considers only stylistic features of the answers such as matching or similar words in the answer and the question. This ensures that to answer a question correctly, the system must perform deep understanding of the text.

On stage, a woman takes a seat at the piano. She a) sits on a bench as her sister plays with the doll. b) smiles with someone as the music plays. c) is in the crowd, watching the dancers. **d) nervously sets her fingers on the keys.**
A girl is going across a set of monkey bars. She a) jumps up across the monkey bars. b) struggles onto the monkey bars to grab her head. **c) gets to the end and stands on a wooden plank.** d) jumps up and does a back flip.
The woman is now blow drying the dog. The dog **a) is placed in the kennel next to a woman's feet.** b) washes her face with the shampoo. c) walks into frame and walks towards the dog. d) tried to cut her face, so she is trying to do something very close to her face.

Figure 3.4: An example from the SWAG dataset [25]. The bolded answers are the correct options.

Chapter 4

Existing work on commonsense reasoning

This section presents recent approaches to commonsense reasoning and their results.

4.1 Three-way attentive networks (TriAN)

TriAN [23] is a recurrent neural network based model intended for the MCScript dataset that uses attention to model interactions between the document, question and the answers. It uses ConceptNet [20] as an external form of commonsense knowledge. ConceptNet is a knowledge graph of words and phrases in natural language connected with labeled edges. The label edges describe the relations between the connected items. Some examples of edge labels are „MadeOf“, „IsA“, „HasProperty“. TriAN has been submitted to Task 11 of the SemEval 2018 competition [15] and placed 1st with at the time state-of-the-art result of 83.95% on the test set of the MCScript dataset.

The overall architecture of TriAN can be found in Figure 4.1. As input, they use GloVe [17] embeddings. To these they concatenate binary vectors specifying the part-of-speech role of the word, whether it is a named entity such as a location or a name and handcrafted features. For words from the document, they query ConceptNet and determine whether it shares an edge with some word from the question and the answer and include this information as well. The next component in the architecture is an attention layer defined by equations 4.1 and 4.2.

$$Att_{seq}(u, \{v_i\}_{i=1}^{n}) = \sum_{i=1}^{n} \alpha_i v_i \tag{4.1}$$

$$\alpha_i = softmax_i(f(W_1 u)^\top f(W_1 v_i)) \tag{4.2}$$

Vector u serves as the query with the vectors v_i being the keys and the values simultaneously. f is an activation function set to ReLU (Figure 2.5). This layer is used to model interactions between the input. They calculate question-aware passage representation where each word of the passage is used as the query for attention with keys and values being the entire question. Each such attention produces a new representation of the query word, eventually giving a new representation for the whole document. Similarly, they calculate passage-aware answer representation and question-aware answer representation.

Subsequently, for each input, they concatenate all its available representations and serve them as an input into a bidirectional LSTM recurrent neural network [12] to obtain contextualized representations of the inputs. In the next layer, the answers and the question are summarized into a fixed-length representation a and q by self-attention over the contextualized tokens output by the LSTM layer. The used attention function is described by equations 4.3 and 4.4.

$$Att_{self}(\{u_i\}_{i=1}^{n}) = \sum_{i=1}^{n} \alpha_i u_i \tag{4.3}$$

$$\alpha_i = softmax_i(W_2^\top u_i) \tag{4.4}$$

A fixed length passage p representation is computed by the Att_{seq} attention function defined by equations 4.1 and 4.2. The attention uses q as the query and the contextualized passage tokens from the LSTM are used as keys and values simultanously. Finally, the probability of the answer a being correct for question q given passage p is calculated as defined by equation 4.5.

$$y = \sigma(p^\top W_3 a + q^\top W_4 a) \tag{4.5}$$

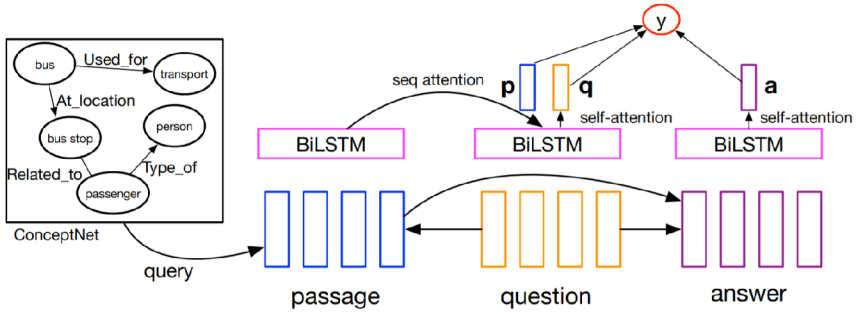

Figure 4.1: An illustration of TriAN architecture. [23]

4.2 General reading strategies

In 2019, a machine reading comprehension technique inspired by cognitive science was introduced. [21] The authors propose the implementation of general reading strategies for humans into a neural network. They used the OpenAI GPT model [18] similar to BERT (section 2.6) that uses a variation of the transformer architecture and is pre-trained as a unidirectional language model. They complemented this model with the implemented general reading strategies to achieve a new state-of-the-art of 89.5% on the MCScript dataset.

The authors of the technique proposed the following three reading strategies:

- *Back and forth reading* - This strategy imitates human reading process where they go back and forth in the text to understand it better. The strategy is implemented by considering both the original and reverse order of the input sequence. In the basic design of the model, the input consists of the document concatenated with the question, a delimiter token followed by the answer. With this strategy, they also

train a model where the input consists of the answer, a delimiter, the question and the document. The two models are then ensembled.

- *Highlighting* - This reading strategy is inspired by people highlighting important information in a text to help them remember it. It is implemented as a set of 2 trainable embeddings which are added to each word in the document. One of the embeddings is used for important words and the other for the rest. Which embedding to use e.g the importance of a word is determined by its part-of-speech tag and whether it also appears in the question or the answer.

- *Self-assesment* - Self-asssesment is the process of asking oneself questions about a just read text and checking if one can answer them. They implement this strategy by taking the input document and generating questions with answer candidates to be used for further pre-training. The questions are generated by selecting a few sentences from the document. From these, find a span of text that is removed. The removed text is used as an answer candidate along with other incorrect answer candidates generated from different spans of text. The selected sentences after the answer removal is used as a question and the model is asked to choose the correct text span to put into the the now empty space.

4.3 BERT

BERT has already been described in Section 2.6. The reason why it is once again mentioned is because even if there are no attempts to instill commonsense knowledge into the model at training time, it still achieves state-of-the-art performance on commonsense reasoning tasks. Specifically, it set the 86.3% state-of-the-art on the SWAG dataset (Section 3.2). Previous approaches have tried to explicitly train commonsense reasoning and work with commonsense knowledge. BERT sets an interesting precedent, indicating that commonsense is not an ability on its own, but rather an emergent phenomenon. This is part of the reason why BERT was chosen as the model the experiment with in this thesis.

Chapter 5

Experiments

This chapter presents the efforts of this thesis in exploration of the nature of commonsense reasoning current models are able to perform. First, a baseline model was implemented and trained where there was no attempt to instill any commonsense knowledge into it except by training on the target dataset. This makes it possible to evaluate the efficiency of various strategies that attempt to teach commonsense knowledge. The second part of this chapter describes experiments with a state-of-the-art model which provide insight into the model's commonsense reasoning capabilities. In both cases, the PyTorch [16] framework was used for implementation.

5.1 Stanford Attentive Reader baseline

The baseline has been inspired by the Stanford Attentive Reader model [8], but in this thesis the author implemented it with a few modifications.

The model is trained on the MCScript dataset, described in section 3.1. As input, GloVe embeddings [17] are used. The document, question, and both answers are all indepedently contextualized using a GRU unit, described in section 2.3.2. To get a question-aware document representation, document-to-question is performed. In more detail, for each document token $t_1, t_2..t_n$ output by the GRU unit, attention score s_j is computed using the attention function. The query of the attention is the final hidden state of the GRU unit for the question q. These scores are then used to perform a weighted average of all the document tokens, resulting in a final, question-aware document representation. As the attention function, we use a bilinear interaction function with learnable parameters W_a and bias b. Equations 5.1 and 5.2 describe these calculations.

$$s_j = softmax(t_j^\top W_a q + b) \tag{5.1}$$

$$t = \sum_j s_j t_j \tag{5.2}$$

To obtain a score specifying how likely an answer is correct, another bilinear interaction function with learnable parameters W_s between the document representation and the final hidden state of the GRU unit for the answer a is used. Finally, a softmax function is performed over the scores to transform them into probabilities and an answer is chosen based on those. Equation 5.3 describes this process.

$$p(a|t,q) = softmax(t^\top W_s a) \tag{5.3}$$

An illustration of the architecture can be found in Figure 5.1.

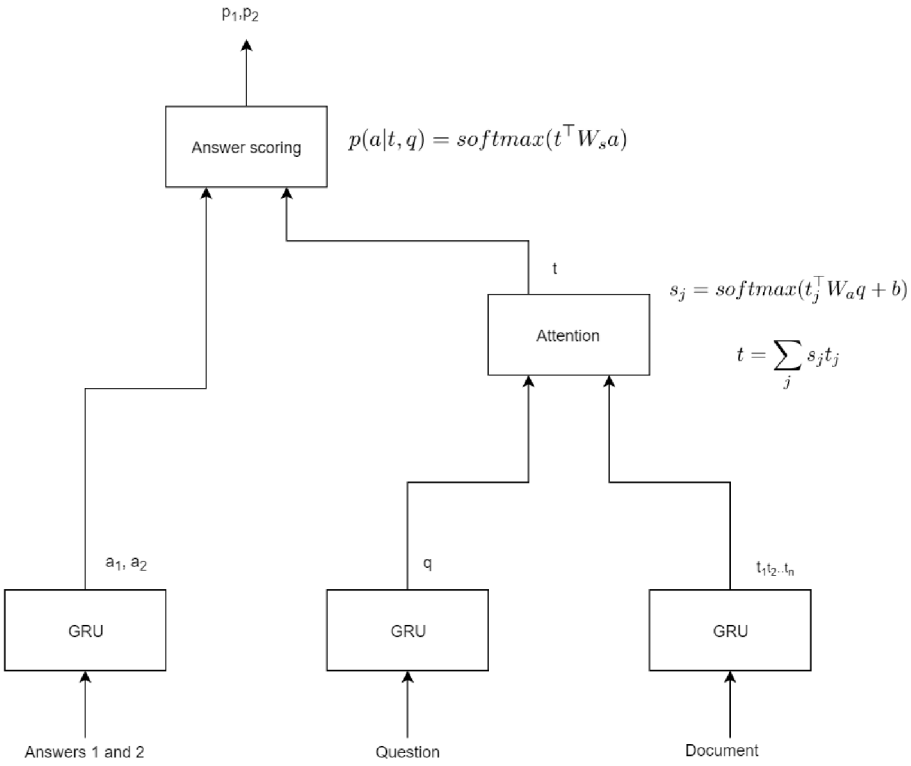

Figure 5.1: Modified Stanford Attentive Reader architecture used as a baseline in this thesis.

5.2 BERT experiments

To explore current model's commonsense reasoning capabilities, the BERT model, described in section 2.6 was chosen. This model is interesting because it achieves state-of-the-art results on difficult commonsense reasoning tasks like SWAG (Section 3.2) even though there was no attempt to instill commonsense knowledge into the model at training time. It also does not work with any external commonsense knowledge sources, indicating that the knowledge is encoded within the parameters themselves. The author of this thesis sees this approach to commonsense reasoning as promising and that is why this thesis investigates it further.

Specifically, the following experiments are performed on the BERT model:

- BERT is trained on the MCScript dataset. The purpose of this experiment is to find how much commonsense knowledge does BERT contain pretrained on out-of-the-box. Hyperopt, described in section 2.7.3, is used to optimize the hyperparameters of the model.

- BERT is trained on the SWAG dataset. This model is further trained on the MCScript dataset. The purpose of this experiment is to see whether it is possible to instill

more commonsense knowledge into the model by pretraining on related tasks when compared to out-of-the-box pretrained BERT.

- BERT is trained on the MCScript dataset, but not all the information is about the question is given. Specifically, a model is trained which only gets the question and the answers as the input, without the document. Furthermore, a model which only gets the document and the answers (without the question) is also trained. Finally, an experiment is performed where only the answers are given on the input. The purpose of these experiments is to ascertain that the question cannot be answered using stylistic features without deep understanding. In case some of the question are answerable anyway, these experiments will provide insight into their nature and how the model makes its decisions.

For all the experiments, $BERT_{large}$ architecture from the BERT paper [11] is used. This model consists of 24 transformer blocks, each having 16 attention heads, hidden size 1024 and feed-forward size of 4096. This model contains approximately 340 million parameters.

Chapter 6

Results and discussion

In this chapter, results of the experiments described in the previous section are presented and their implications are discussed. For quantitative evaluation, the models are evaluated on the test set of the MCScript dataset and the accuracy metric is used. Accuracy in this case is defined as the number of correctly answered questions divided by the total number of questions. There is no need for more sophisticated evaluation metrics, such as the F1-score, as the dataset is class-wise almost perfectly balanced. Table 6.1 contains results from all the experiments.

Model	Accuracy
Baseline	68.54%
State of the art	89.50%
BERT	88.38%
BERT pre-trained on SWAG	87.69%
BERT without input document	78.08%
BERT without input question	85.52%
BERT input answers only	72.83%

Table 6.1: Results of the experiments.

The results of the experiment seem very surprising for several reasons. For example, it seems that pre-training an already pre-trained model does not seem to help, even if the tasks are related. Furthermore, BERT is able to achieve high accuracies even when crucial inputs, such as the document, are missing. In order to better understand these results, the rest of this chapters examines the predictions made by the model in each experiment more deeply.

6.1 Experiment #1: Training BERT on MCScript

In this experiment, the trained BERT model achieves accuracy of 88.38% on the MCScript dataset. This is only 1.12% than the current state of the art [21]. However, unlike in [21], BERT contains no task-specific architecture for the dataset or even machine comprehension in general. This suggest that the main benefitting factor is the superior way of pre-training. This is further supported by comparing the performance of BERT to the baseline. Neither

of the models are using any commonsense knowledge bases and BERT has over 20% increase in performance.

When examining questions incorrectly answered by the model, it was found that 32.61% are questions requiring commonsense reasoning to answer. This constitutes 14.66% out of all commonsense questions. The proportion of commonsense questions in the set of incorrectly answered question increased from the 25.85% present in entire test set, implying that these questions are harder to answer when compared to questions having an answer in the text. This is not surprising, but it shows the model is still struggling with commonsense reasoning. However, there is only a difference of 6.76%, indicating that the main bottleneck of the model does not come from lack of commonsense reasoning ability, but inability to understand more complex relations in a text.

This is further supported by taking a look at the question type distribution of the incorrectly answered questions. This can be found in Figure 6.1. When compared to the overall distribution of question in the test dataset (Figure 3.3), it can be seen that the proportion of „yes/no“ and „other“ increased the most, indicating the model struggles with them. A possible explanation is that these types of questions might sometimes require complex reasoning to answer. On the other hand, more fact-based types of questions such as „who/whose“ or „what/which“ decreased in proportion the most, possibly because they do not require as much reasoning.

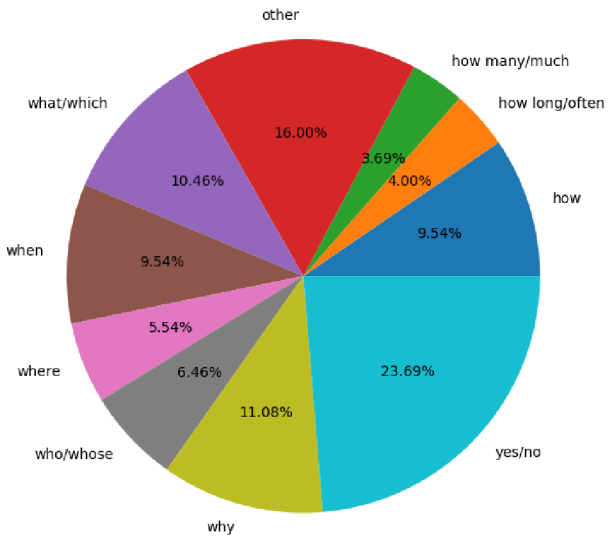

Figure 6.1: Question distribution of incorrectly answered questions in experiment # 1.

6.2 Experiment #2: Pre-training BERT on SWAG

Examining the incorrectly answered questions in the second experiment, it was found that 30.20% of them require commonsense reasoning. While this is less than 32.61% reported in the first experiment (Section 6.1), these questions constitute 14.24% of all commonsense questions. This is less than 14.66% reported previously, but it is not a statistically significant difference. Furthermore, as reported in Table 6.1, the overall performance was worse. Therefore, it seems that the pre-training has hurt the model's ability to extract information from text. This may be partly explained by the differences between the SWAG and the MCScript dataset. Examples in SWAG always consist of 2 sentences. To answer them, the model does not need to extract information from a longer text, unlike in the MCScript dataset, where the documents are mostly 10+ sentences long. It is therefore reasonable to conclude that language model pre-training provides a sufficient foundation for downstream tasks to benefit from and task-specific pre-training is no longer necessary.

Examining the distribution of question types of incorrectly answered questions (Figure 6.2) yields similar results as in Section 6.1. Explanation for these results is therefore also similar.

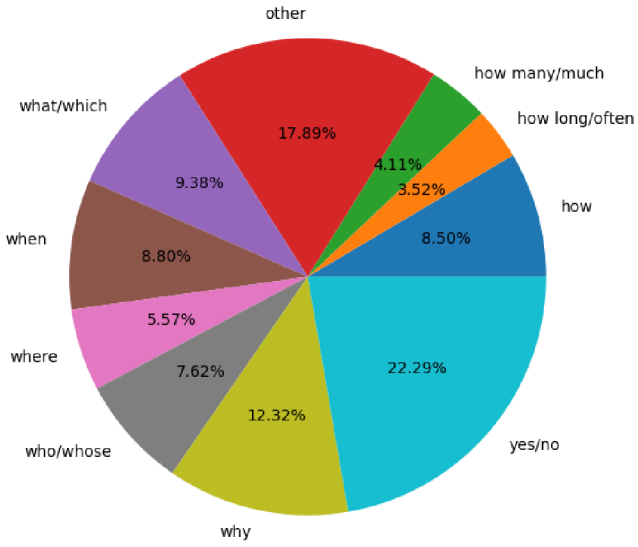

Figure 6.2: Question distribution of incorrectly answered questions in experiment #2.

6.3 Experiment #3: Question answering without the information document

The high performance of the model in this experiment is surprising. Furthermore, the proportion of commonsense questions in the set of incorrectly answered set is only 17.45%.

In the set of all commonsense questions, these make up 14.79%, value very similar to the one in previous experiments. In other words, the removal of the document did not harm the model's ability to answer commonsense questions at all. Given that the answers to the commonsense questions are not found in the document, this result is intuitive. However, the model was still able to answer text-based questions surprisingly well. Given current performance on commonsense questions, if the model was answering text-based questions randomly, the overall expected accuracy 59.10%. The difference of 18.98% between the actual and expected performance indicates that the dataset contains biases and/or stylistic features which the model was able to pick up on. Further work is necessary to pinpoint the exact cause of this disrecipancy.

Examining the question type distribution of the incorrectly answered questions (Figure 6.3), it can be seen that it is mostly similar to the overall distribution in the test set (Figure 3.3) except for types „who/whose“, „how long/often“ and „how many/much“, which are considerably less represented. This is surprising as these questions inquire about factual information. However, this confirms the presence of systematic biases in the dataset.

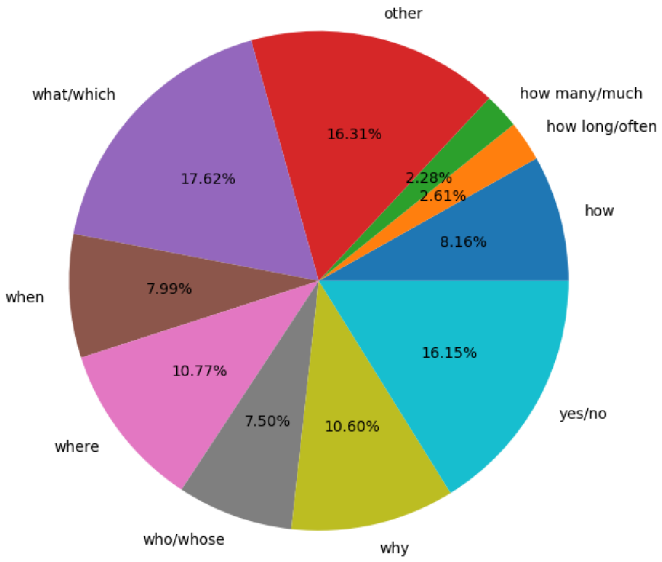

Figure 6.3: Question distribution of incorrectly answered questions in experiment #3.

6.4 Experiment #4: Question answering without the question

There were no note-worthy results about commonsense questions. 33.50% of all incorrectly answered questions were commonsense, constituting 18.53% of all commonsense questions. These results are understandably slightly worse than in the first experiment (Section 6.1),

but are still comparable. However, this experiment showed the unimportance of the question in contextualizing the answers in this dataset. The performance of 85.52% is only 2.85% less than the best performance achieved. It seems that without the question, the model is choosing answers which seem more relevant to the document. To confirm this, the results were manually examined. It was found that in a lot of examples, the correct answer is present in the text while the information in the incorrect one is not mentioned in the text even in a different context, which could make it harder to choose without a question. This way, it is easy for the model to learn to choose the answer more relevant to the text. This argument is further supported by looking at the question type distribution in Figure 6.4. By far the most prominent type is the „yes/no“ question type. In these questions, the answers are sometimes just a single word indicating the answer. In this case, it is impossible to make any reasonable guess about the correct answer without knowing the question, thus explaining the poor performance on this question type.

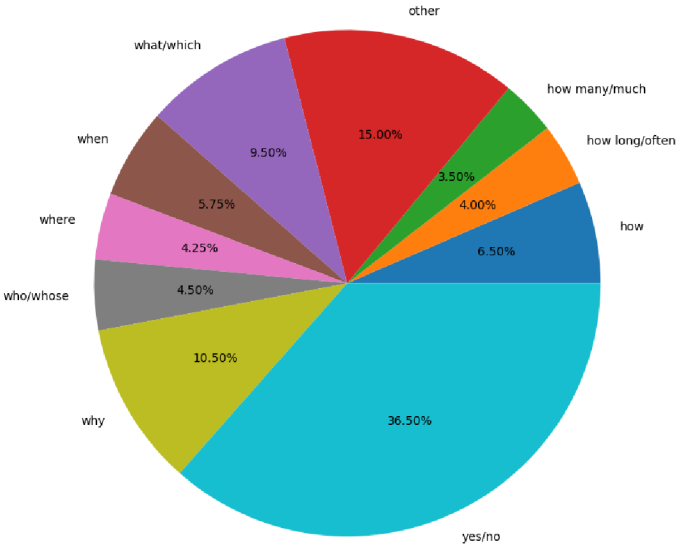

Figure 6.4: Question distribution of incorrectly answered questions in experiment #4.

6.5 Experiment #5: Classifying stand-alone answers

In this experiment, 24.67% of incorrectly commonsense question required commonsense knowledge. This is approximately equal with the distribution of questions in the test set, indicating that the model has equal performance on both commonsense and non-commonsense questions. However, the main finding in this experiment is that the answers contain enough information to be distinguishable by themselves to some degree. Choosing the answers at random, the expected accuracy would be 50%, 22.83% less than the observed performance.

Intuitively, the learned classification process must be meaningless in the context of actually answering the underlying questions. This confirms the presence of an unintended stylistic bias in the dataset. Further evidence can be found in Figure 6.5, depicting the question type distribution of incorrectly answered questions. The types „who/whose“, „how many/much“ and „how long/often“ are underrepresented when compared to the distribution in the test set. The same phenomenon could be observed in experiment 3 (Section 6.3), indicating that both models are picking up on similar biases in the answers.

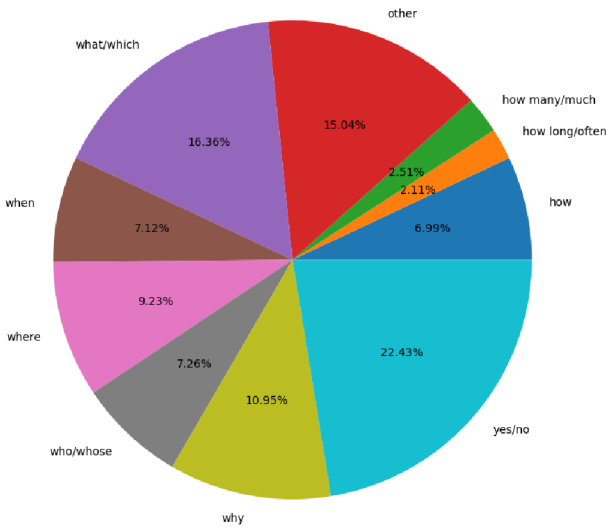

Figure 6.5: Question distribution of incorrectly answered questions in experiment #5.

6.6 Summary

The experiments described above yielded the following main findings:

- Pre-training on a language modelling task seems to instil enough commonsense knowledge into the model for it not to have significantly more difficulty with commonsense question than with text-based ones. In fact, while commonsense questions were a bit harder to answer, the main bottleneck of performance is general text comprehension ability, not commonsense reasoning.
- Pre-training models on related tasks to the target tasks no longer helps when the model has already been pre-trained on a more general task, for example language modelling. This further supports the argument that commonsense reasoning is an ability arising from a solid language understanding, not a skill to be learned on its own.

- The answer candidates in the MCScript dataset stylistic biases which allow them to be distinguished even without their corresponding documents and questions. Furthermore, the information in the incorrect answer can often not be found in the document in any context, making it possible to reliably choose the correct answer without even knowing the question. These artifacts highlight the difficulty of collecting representative, unbiased machine learning datasets and call for more sophisticated techniques to be developed.

Chapter 7

Conclusion

In this thesis, the commonsense reasoning abilities of machine comprehension models were examined. A state-of-the-art model was implemented, trained and evaluted in various scenarios through several experiments. It was discovered that this model was capable of using commonsense knowledge even without an external source of it or having it purposefully instilled during training. Instead, commonsense reasoning ability seems to arise as a result of general text understanding. Therefore, instead of trying to make models learn a high-level reasoning ability, the authors suggests improving the fundamentals of language understanding through techniques such as transfer learning or multi-task learning. Furthermore, it was also discovered that a popular commonsense reasoning dataset, MCScript, contains statistical biases which help models bypass the need for language understanding to answer the question. In one of the experiments, a model was able to correctly choose the correct answer to a question 72% percent of the time even when the question and an information document were not provided. Biases like these make it hard to evaluate model's true ability to deal with the target task in real-life scenarios as it is hard to check what factors influence the final decision. This discovery highlights the difficulty of collecting a truly representative dataset and raises a call for the development of more sophisticated dataset creation techniques which would detect and remove any unintended statistical biases. Perhaps even more so however, it highlights the need for the development of better training methodologies. It is unreasonable to expect a model to perform a high-level complex task with reliability without being able to do any other similar tasks. Unlike a human, the model has no way to tell which features are meaningful in the context of the target task and which are just incidental correlations. Once again, the author sees transfer learning and multi-task learning as a potential remedy for these problems if more progress in these areas is made. Until these problems are solved, it is recommended to carry out an analysis looking for statisticial biases in any dataset used for training neural models. A potential continuation of this work could collect more datasets and perform this analysis. Another possible contiuation would lie in exploration of other commonsense reasoning tasks and identifying weaknesses of the current model.

Bibliography

[1] Attention and memory in deep learning and NLP. [Online; retrieved on 15.01.2019]. Retrieved from: http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/

[2] CS231N Convolutional Neural Networks for Visual Recognition. [Online; retrieved on 15.01.2019]. Retrieved from: http://cs231n.github.io/neural-networks-1/

[3] Natural Language Processing with Deep Learning CS224N, Recurrent Neural Networks and Language Models. [Online; retrieved on 20.01.2019]. Retrieved from: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/lectures/cs224n-2017-lecture8.pdf

[4] One hot encoding of text. [Online; retrieved on 15.01.2019]. Retrieved from: https://medium.com/@athif.shaffy/one-hot-encoding-of-text-b69124bef0a7

[5] Understanding feature engineering part 4: Deep learning methods for text data. [Online; retrieved on 15.01.2019]. Retrieved from: https://towardsdatascience.com/understanding-feature-engineering-part-4-deep-learning-methods-for-text-data-96c44370bbfa

[6] Bergstra, J.; Bardenet, R.; Bengio, Y.; et al.: Algorithms for Hyper-Parameter Optimization. In *NIPS*. 2011.

[7] Bergstra, J.; Yamins, D.; Cox, D. D.: Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In *ICML*. 2013.

[8] Chen, D.; Bolton, J.; Manning, C. D.: A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. In *Association for Computational Linguistics (ACL)*. 2016.

[9] Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; et al.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*. vol. abs/1406.1078. 2014. 1406.1078. Retrieved from: http://arxiv.org/abs/1406.1078

[10] Cox, D. D.; Pinto, N.: Beyond simple features: A large-scale feature search approach to unconstrained face recognition. *Face and Gesture 2011*. 2011: pp. 8–15.

[11] Devlin, J.; Chang, M.; Lee, K.; et al.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*. vol. abs/1810.04805. 2018. 1810.04805.
Retrieved from: http://arxiv.org/abs/1810.04805

[12] Hochreiter, S.; Schmidhuber, J.: Long Short-term Memory. *Neural computation*. vol. 9. 12 1997: pp. 1735–80. doi:10.1162/neco.1997.9.8.1735.

[13] Mikolov, T.; Chen, K.; Corrado, G.; et al.: .

[14] Ostermann, S.; Modi, A.; Roth, M.; et al.: MCScript: A Novel Dataset for Assessing Machine Comprehension Using Script Knowledge. *CoRR*. vol. abs/1803.05223. 2018. 1803.05223.
Retrieved from: http://arxiv.org/abs/1803.05223

[15] Ostermann, S.; Roth, M.; Modi, A.; et al.: SemEval-2018 Task 11: Machine Comprehension Using Commonsense Knowledge. In *Proceedings of The 12th International Workshop on Semantic Evaluation*. New Orleans, Louisiana: Association for Computational Linguistics. June 2018. pp. 747–757. doi:10.18653/v1/S18-1119.
Retrieved from: https://www.aclweb.org/anthology/S18-1119

[16] Paszke, A.; Gross, S.; Chintala, S.; et al.: Automatic differentiation in PyTorch. 2017.

[17] Pennington, J.; Socher, R.; Manning, C. D.: GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 2014. pp. 1532–1543.
Retrieved from: http://www.aclweb.org/anthology/D14-1162

[18] Radford, A.: Improving Language Understanding by Generative Pre-Training. 2018.

[19] Razavian, A. S.; Azizpour, H.; Sullivan, J.; et al.: CNN Features off-the-shelf: an Astounding Baseline for Recognition. *CoRR*. vol. abs/1403.6382. 2014. 1403.6382.
Retrieved from: http://arxiv.org/abs/1403.6382

[20] Speer, R.; Chin, J.; Havasi, C.: ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. *CoRR*. vol. abs/1612.03975. 2016. 1612.03975.
Retrieved from: http://arxiv.org/abs/1612.03975

[21] Sun, K.; Yu, D.; Yu, D.; et al.: Improving Machine Reading Comprehension with General Reading Strategies. *CoRR*. vol. abs/1810.13441. 2018. 1810.13441.
Retrieved from: http://arxiv.org/abs/1810.13441

[22] Vaswani, A.; Shazeer, N.; Parmar, N.; et al.: Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, edited by I. Guyon; U. V. Luxburg; S. Bengio; H. Wallach; R. Fergus; S. Vishwanathan; R. Garnett. Curran Associates, Inc.. 2017. pp. 5998–6008.
Retrieved from:
http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[23] Wang, L.: Yuanfudao at SemEval-2018 Task 11: Three-way Attention and Relational Knowledge for Commonsense Machine Comprehension. *CoRR*. vol. abs/1803.00191.

2018. 1803.00191.
Retrieved from: http://arxiv.org/abs/1803.00191

[24] Wu, Y.; Schuster, M.; Chen, Z.; et al.: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*. vol. abs/1609.08144. 2016. 1609.08144.
Retrieved from: http://arxiv.org/abs/1609.08144

[25] Zellers, R.; Bisk, Y.; Schwartz, R.; et al.: SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. *CoRR*. vol. abs/1808.05326. 2018. 1808.05326.
Retrieved from: http://arxiv.org/abs/1808.05326