UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

A Study of Commonsense Reasoning with Language Models

Ruben Miguel Rosa Branco

Mestrado em Ciência de Dados

Dissertação orientada por:

Professor Doutor António Manuel Horta Branco

2021

“If a machine is expected to be infallible, it cannot also be intelligent.”

— Alan Turing, *Lecture to the London Mathematical Society on 20 February 1947*

Agradecimentos

O trabalho apresentado nesta dissertação é o resultado de cerca de cinco anos no Grupo NLX e na Faculdade de Ciências da Universidade de Lisboa.

Ao meu orientador, Professor Doutor António Branco, agradeço o voto de confiança e a oportunidade de integrar o grupo, os conselhos e a ajuda. É uma fonte de inspiração que ao longo destes anos foi essencial para o meu desenvolvimento como jovem investigador. Agradeço-lhe a orientação, disponibilidade para discussão e o grande esforço de revisão investido durante esta dissertação.

Agradeço a João Silva e a João Rodrigues por toda a disponibilidade para discutir vários problemas que surgiram durante a produção desta dissertação. Agradeço também a ajuda para ultrapassar as dificuldades e melhorar o resultado final.

Ao Grupo NLX, que desde o primeiro dia me receberam de forma aberta. Obrigado por toda a ajuda, as conversas, diversão e ensinamentos que me fizeram crescer em todos os aspetos como pessoa. Agradeço também pelos recursos disponibilizados, sem os quais seria impossível realizar o trabalho descrito nesta dissertação.

À FCT - Fundação para a Ciência e Tecnologia, agradeço o financiamento através da infraestrutura PORTULAN CLARIN (PINFRA/22117/2016).

Aos amigos, agradeço por todas as aventuras, conversas e apoio ao longo dos anos. Foram particularmente importantes para me manterem motivado e concentrado para alcançar o objetivo de concluir esta dissertação, produzida na sua totalidade durante uma pandemia global.

Agradeço aos meus pais e à minha família pelo apoio incondicional, paciência, sacrifício e motivação durante estes anos. Sem sombra de dúvida que sem esse apoio, nunca teria tido as condições necessárias para chegar aqui.

Abstract

Artificial Intelligence (AI) has gone through an increasing growth in the past decades, which in the present day translates to its usage in almost every sector of society. From its inception, AI pursues the reproduction of human intelligence. Currently, AI-equipped devices are capable of solving particular problems within specific domains with varying degrees of success. The goal and hope is that the combination of these systems will eventually approximate human intelligence. This dissertation addresses a problem in Natural Language Processing (NLP), a central subfield of AI that aims to produce devices capable of handling human language for problems such as translation, parsing, commonsense reasoning, and others.

Deep learning has fueled state-of-the-art NLP research. The current most prominent methodology leverages large scale neural networks and large amounts of data to achieve outstanding performances. Recent research has started to uncover how these neural networks obtain state-of-the-art results. In some cases the models appear to latch on to so called data artifacts, whether they sustain valid generalizations or not, which happen to minimize loss w.r.t. the training dataset distribution. Although this is generally the rationale behind a machine learning approach, it can be error inducing, as models can fail miserably when the distribution of the input data differs from the training data.

Our work reported in this dissertation investigates whether models learn to perform commonsense reasoning, a cognitively demanding task inherent to the human experience, by resorting to such shortcuts. Five state-of-the-art models of different major types are trained to perform four most prominent commonsense reasoning tasks. Models undergo stress testing with five additional tasks devised to provide hints of possible shortcut learning and of memorization.

The results indicate that the models seem to be resorting to shortcut learning in three of the four commonsense reasoning tasks; they seem to be learning a different task from the one the data is meant to convey by relying on spurious patterns present in the dataset. For example, the trained models can pick the answer from a set of options without even being supplied with the question they are meant to answer. Further experimentation confirmed that this behavior could not be attributed to memorization. This behavior is worrisome, as the field measures progress by the capabilities of these models to perform these tasks, and show that their cognitive abilities are disappointingly still low, susceptible to simple deceptions in spite of the overwhelming good scores obtained under mainstream performance metrics.

Parts of this work have passed peer review and were accepted for publication (Branco et al., 2021a,b).

Keywords: Artificial Intelligence, Natural Language Processing, Deep Learning, Commonsense Reasoning, Shortcut Learning

Resumo Alargado

A Inteligência Artificial (IA) teve um enorme crescimento nas últimas décadas, que se traduziu hoje em dia na sua utilização em quase todos os setores da sociedade. Por exemplo, está presente no sector financeiro, onde modelos neuronais são utilizados para fazer previsões em mercados financeiros; está presente na nossa vida social através das redes sociais, que utilizam modelos de IA para todo o tipo de tarefas e análises; esta dissertação aborda um problema de Processamento de Linguagem Natural (PLN), uma subárea da IA que visa produzir dispositivos capazes de usar e compreender a linguagem humana.

Desde o início, a IA visa reproduzir a inteligência humana. Atualmente, produzimos dispositivos capazes de resolver problemas específicos, em domínios específicos, com algum grau de sucesso. A esperança para o futuro é que, através da combinação desses sistemas, as suas capacidades cognitivas conjuntas se aproximem da inteligência humana. Em PLN, os modelos são aplicados a vários problemas, como tradução, análise sintática, argumentação, raciocínio de senso comum, entre outros.

Esta dissertação apresenta um estudo sobre consequências negativas da metodologia mais proeminente em PLN na sua aplicação ao raciocínio de senso comum, um desafio/tarefa central em IA. Essa metodologia consiste em utilizar redes neuronais de grande escala, geralmente modelos Transformer, e pré treiná-los com grandes quantidades de texto através de modelação de linguagem. Dado este pré-treino, onde as redes aprendem as nuances da linguagem natural, os modelos quando aplicados a tarefas específicas obtêm desempenhos excecionais, que podem em alguns casos rivalizar e até superar as capacidades humanas.

O raciocínio de senso comum é uma tarefa clássica em IA, tendo sido objeto de estudo de um dos pioneiros da IA, John McCarthy. É uma capacidade humana essencial, que está em constante utilização, pois o conhecimento de senso comum emerge naturalmente da experiência humana: observar e atuar no nosso ambiente. É necessário raciocinar com este conhecimento de base para tomar decisões, por muito imediatas que sejam. Em PLN, as tarefas deste género geralmente são de pergunta & resposta que necessitam de raciocínio de senso comum para serem respondidas. Ensinar uma máquina, que por enquanto não consegue facilmente interagir com o ambiente e aprender dele, continua a ser um desafio central.

A investigação recente começa a descobrir como as redes neuronais obtêm resultados que constituem o estado da arte. Por meio de aprendizagem por atalhos, os modelos prendem-se aos chamados artefactos presentes nos dados, quer estes produzam generalizações válidas ou não, os quais procuram minimizar perdas relativamente à distribuição do conjunto de dados. Um exemplo deste fenômeno foi descoberto

numa tarefa de SemEval 2018, Argument Reasoning Comprehension Task, onde os modelos classificavam texto através de palavras-chave como *"not"*, *"is"*, *"do"* e *"are"*, que estavam altamente correlacionadas com o resultado desejado. Embora minimizar as perdas com base em padrões nos dados seja a abordagem subjecente à aprendizagem automática, pode acabar por ser detrimental fazê-lo, pois os padrões podem não refletir uma generalização sobre a tarefa em questão, mas podem resultar fortuitamente do processo de construção dos dados. Quando a distribuição dos dados muda, o que pode acontecer quando, por exemplo, utilizamos dados de entrada que podem ser consideravelmente diferentes dos dados de treino, os modelos exibem falhas aparatosas.

Este trabalho investiga se os modelos realmente aprendem raciocínio de senso comum, uma tarefa cognitivamente exigente e inerentemente de cariz humano. Cinco modelos de Transformer de estado da arte são aplicados a quatro tarefas diferentes de raciocínio de senso comum, de modo a perceber a sua aptidão na tarefa e estabelecer dados comparativos. Dois modelos são escolhidos para serem submetidos a um teste de pressão, com cinco tarefas concebidas para obter indícios de aprendizagem por atalhos e memorização: (i) Treino com dados de entrada parciais (*Partial Input Training*), onde segmentos dos dados de entrada, essenciais para completar a tarefa, são retirados, e o efeito nos modelos é observado. Se os modelos forem capazes de cumprir a tarefa igualmente bem, então é um indício que estarão a usar artefactos nos dados. (ii) Ataque adversarial (*Adversarial Attack*), que consiste na utilização de algoritmos que modificam a frase de entrada, de forma que a semântica é conservada, e que levam o modelo a mudar a sua decisão para uma classificação errada. Se a degradação dos resultados for significativa, pode ser um indício de uma aprendizagem superficial, potenciada por atalhos nos dados. (iii) Contaminação de dados (*Data Contamination*), que procura descobrir se existe uma sobreposição entre os dados de teste de uma tarefa com os dados de pré-treino. Como previamente referido, a metodologia mais atual utiliza grandes volumes de dados de texto para pré-treinar modelos, que podem ser obtidos das mesmas fontes utilizadas para construir dados para outras tarefas. Os modelos têm capacidade de reter informação, portanto, podem utilizar mais tarde durante a avaliação, quebrando princípios de senso comum de testes de modelos: modelos devem ser testado em dados que não teram sido vistos previamente. (iv) Avaliação cruzada de tarefas (*Cross-Task Evaluation*), que consiste em pegar num modelo treinado numa certa tarefa e avaliar noutra, sem que o modelo tivesse aprendendo-a. Isto permite observar se há transferência de conhecimento, que seria possível pois as tarefas têm o mesmo conceito comum subjacente, que é raciocínio de senso comum. Caso haja degradação forte nos resultados, isto é indicativo que os modelos aprenderam atalhos que não foram transferidos para as outras tarefas, pois eram específicos aos dados onde treinou. (v) Exploração de atalhos (*Shortcut Exploration*), que investiga dois tipos de atalhos: desiquilíbrio de classes e "sinais" (*cues*) lexicais, que são palavras que fornecem indícios da classe pertencente a cada examplo. Modelos que são treinados com um conjunto de dados que tenha desiquilíbrio de classes conseguem obter melhores resultados ao tirar proveito desse desquilíbrio, enquanto que "sinais" lexicais providenciam um sinal útil para os modelos obterem uma boa prestação.

As experiências mostram que os modelos parecem recorrer a aprendizagem por atalho em três das quatro tarefas. Na experiência (i), em três das quatro tarefas de raciocínio de senso comum, é possível chegar perto dos resultados impressionantes retirando segmentos dos dados fundamentais, no ponto de vista do raciocínio humano, para resolver a tarefa. Como exemplo, os modelos conseguem escolher

respostas corretas a perguntas que não são fornecidas. Na experiência (ii), as mesmas tarefas sofreram uma degradação superior. No geral, a degradação é alta, mostrando que os modelos ainda são frágeis perante ataques adversários. Com a experiência (iii) observa-se que embora existe diferentes níveis de contaminação dos dados das tarefas, estes não conseguem explicar os resultados obtido nas experiências anteriores, e, portanto, memorização não poderá ser o fenômeno conducente aos resultados obtidos. Na experiência (iv), verifica-se que os modelos na sua maioria consegue transferir o seu conhecimento para outras tarefas, sem serem treinados nelas. Finalmente, na experiência (v), descarta-se desiquilíbrio de classes como um possível atalho e identifica-se alguns "sinais" lexicais presentes nos dados, embora que não são abrangentes o suficiente para explicar os resultados obtidos nas experiências (i), (ii) e (iv).

Estes indícios mostram que os modelos não estarão a realizar a tarefa pretendida, em vez disso, estão a aprender e realizar tarefas diferentes que acontece que maximizam as métricas da tarefa pretendida, através de padrões encontrados nos dados. O facto de estes fenômenos se verificarem é preocupante por vários motivos. A área (PLN) consegue medir o progresso através da capacidade destes modelos realizarem tarefas, como as utilizadas nesta dissertação. Mas se os modelos conseguem obter bons resultados não através da tarefa pretendida, mas uma derivada, o progresso pode ser inflacionado. Outra preocupação refere-se ao grande objetivo traçado desde o começo da área, a reprodução de inteligência humana. Dado que os modelos não aprendem as tarefas supostas, talvez por falta de especificação, e são suscetíveis a simples enganos como mudar apenas uma palavra para um sinónimo, é difícil de argumentar a capacidade cognitiva que eles possuem, por muito impressionante que seja o desempenho e tamanho. Investigação futura é necessária, através de uma revisão cuidadosa e comparação entre os métodos e procedimentos usados no desenvolvimento de dados, modelos e metodologia de treino.

Partes deste trabalho foram alvo de revisão por pares e aceites para publicação (Branco et al., 2021a,b).

Palavras Chave: Inteligência Artificial, Processamento de Linguagem Natural, Aprendizagem Profunda, Raciocínio de Senso Comum, Aprendizagem por Atalhos

Contents

1 Introduction **1**

1.1 Context and Motivation . . . 1
1.2 Objectives and Contributions . . . 3
1.3 Dissertation Outline . . . 4

2 Related Work **5**

2.1 Commonsense Knowledge & Reasoning . . . 7
2.2 Recurrent Neural Networks & Transformer . . . 10
2.2.1 Encoder-Decoder Model . . . 12
2.2.2 The Attention Mechanism . . . 13
2.2.3 Sequence Encoding . . . 15
2.2.4 The Transformer . . . 16
2.3 Pre-Trained Language Models . . . 17
2.3.1 BERT . . . 17
2.3.2 Other Variants . . . 18
2.4 Commonsense Reasoning with Transformers . . . 19
2.5 Shortcut Learning . . . 20
2.6 Adversarial Attacks . . . 23
2.7 Neuro-Symbolic Systems . . . 24
2.8 Summary . . . 26

3 Tasks **29**

3.1 Argument Reasoning Comprehension Task . . . 29
3.2 AI2 Reasoning Challenge . . . 31
3.3 Physical Interaction: Question Answering . . . 33
3.4 CommonsenseQA . . . 34
3.5 Summary . . . 36

4 Implementation **39**

4.1 Models . . . 39
4.1.1 RoBERTa . . . 40
4.1.2 GPT-2 . . . 43

4.1.3 T5 . . . 44
4.1.4 COMET(BART) . . . 47
4.2 Adversarial Attack . . . 50
4.3 Training Methodology . . . 52
4.4 Data Contamination Study . . . 54
4.5 Shortcut Exploration . . . 56
4.6 Summary . . . 57

5 Results **59**
5.1 Neural Models . . . 59
5.1.1 Evaluation on Commonsense Reasoning Tasks . . . 59
5.1.2 Retraining and Evaluation with Partial Input . . . 61
5.2 Neuro-Symbolic Models . . . 62
5.2.1 Evaluation on Commonsense Reasoning Tasks . . . 62
5.2.2 Retraining and Evaluation with Partial Input . . . 64
5.3 Adversarial Attack . . . 64
5.4 Cross-Task Evaluation . . . 65
5.5 Data Contamination . . . 67
5.6 Shortcut Exploration . . . 69
5.6.1 Class Balance . . . 69
5.6.2 Lexical Cues . . . 69
5.7 Summary . . . 74

6 Conclusion **75**
6.1 Summary . . . 75
6.2 Contributions . . . 77
6.3 Future Work . . . 78

References **81**

Appendix A Training Hyper-Parameters **99**

List of Figures

2.1 The Transformer architecture. The input embeddings are passed through N encoder blocks (left section), the output of the encoder blocks is then used by the Decoder (right section) to generate text. Figure is based on the architecture diagram presented in the original paper (Vaswani et al., 2017). . . . 12
2.2 Seq2Seq Encoder-Decoder architectures. . . . 13
2.3 BERT performing masked language modeling. The corrupted tokens are replaced with a `[MASK]` token, and the model must guess the correct words. . . . 17
2.4 Reproduced from (Jia and Liang, 2017). The addition of an unrelated fact in the paragraph made the model change its original prediction, indicating a shallow understanding. . . . 22
2.5 Reproduced from (Szegedy et al., 2013). Left column: original image that was correctly classified. Right column: adversarial example produced with small perturbations. Middle column: difference between the two images. All the images on the right were wrongly classified as an ostrich. . . . 23
2.6 An adversarial example produced with TextFooler (Jin et al., 2020) on the ARC dataset (Clark et al., 2018) (Section 3.2), targeting a fine-tuned RoBERTa on the task. Simply changing *ability* to *capacity* in option A is enough to make the model predict A instead of its original prediction B. . . . 24

3.1 Three examples from the ARCT dataset. . . . 30
3.2 Three examples from the ARC dataset. . . . 32
3.3 Three examples from the PIQA dataset. . . . 33
3.4 Reproduced from (Bisk et al., 2020b). Instructions provided to turkers on Amazon Mechanical Turk. . . . 34
3.5 Three examples from the CSQA dataset. . . . 35
3.6 Adapted from (Talmor et al., 2019). Generation process for CommonsenseQA. Crowdworkers receive a subgraph from ConceptNet, and must produce questions using the concepts and the relations between them. . . . 36

4.1 Two types of fine-tuning schemes for RoBERTa. . . . 41
4.2 Layout of a classification head. In-between linear projections, dropout (Srivastava et al., 2014) is employed. The final linear layer projects to an output space, whose dimension equals the number of output classes. It is common to perform a softmax after. . . . 42

4.3 The first part in the example is the name of the task (e.g. arct). For each section of the input, a textual description is prepended (e.g. question: (...)). The output must also be generated (text), which in this example, is choice 2. 45
4.4 Formatted example for the ARCT task input to a T5 model. The expected value in the form of binary value (0 or 1) is converted to True or False. 45
4.5 Formatted example for the ARC Task input to a T5 model. 46
4.6 Formatted example for the PIQA task input to a T5 model. 46
4.7 Formatted example for the CSQA task input to a T5 model. 46
4.8 Example of the tail-prediction task. PersonX and PersonY corresponds to two different persons. The example features different types of relations, and their definitions are the following. xWant: as a result, PersonX wants; xAttr: X is seen as; oEffect: as a result, PersonY or others will. 47
4.9 Reproduced from (Hwang et al., 2020). Examples for each relation type and their respective size in ATOMIC2020. 48
4.10 An example of how to use BART for classification problems. 49
4.11 Conceptual implementation of a multi-choice BART model. In this example, the input contains two examples (batch size of two), each example having two candidate answers (binary classification). 51

List of Tables

3.1 Number of examples in each dataset partition. 31
3.2 Number of examples in each dataset partition. 32
3.3 Number of examples in each dataset partition. 35
3.4 Number of examples in each dataset partition. 35

5.1 Accuracy of models (rows) on the selected tasks (columns). Scores displayed are the mean of the accuracy scores for 5 runs. A bold figure indicates the best result in that task. Human benchmarks and state of the art (SOTA) for CSQA were taken from their public leaderboard; for ARCT, human benchmark from (Habernal et al., 2018a) and SOTA from (Zhou et al., 2020); and for PIQA, human benchmark from (Bisk et al., 2020b) and SOTA from their public leaderboard. '*' indicates results that are statistically significant with $\alpha = 0.05$. 60
5.2 Partial input training results (accuracy). Scores above random choice are in bold. 61
5.3 Accuracy of models (rows) on the selected tasks (columns). Scores displayed are the mean from the scores obtained for 5 runs. A bold figure indicates the best result in the task. Human benchmarks and state of the art (SOTA) for CSQA were taken from their public leaderboard; for ARCT, human benchmark from (Habernal et al., 2018a) and SOTA from (Zhou et al., 2020); and for PIQA, human benchmark from (Bisk et al., 2020b) and SOTA from their public leaderboard.'*' indicates results that are statistically significant with $\alpha = 0.05$. 63
5.4 Partial input training results (accuracy). Scores above random choice are in bold. 64
5.5 Results of the adversarial attack on RoBERTa-Large and COMET(BART), on each task. 65
5.6 Cross-task results for RoBERTa (in accuracy). The values in the diagonal are from Table 5.1. 66
5.7 Cross-task results for COMET(BART) (in accuracy). The values in the diagonal are from Table 5.3. 66
5.8 Data contamination statistics for each task. An example is considered dirty if it has at least a single N-gram (N Value in 3rd column) collision with any of the pre-training datasets. . 67
5.9 RoBERTa's accuracy when tested on the full testset (Original Accuracy Score), on the Dirty Set (contains only dirty examples) and Clean Set (contains only clean examples). . 68
5.10 COMET(BART)'s accuracy when tested on the full testset (Original Accuracy Score), on the Dirty Set (contains only dirty examples) and Clean Set (contains only clean examples). 68

5.11 Class balance for each task dataset split. Relative frequency in bold indicates a frequency above random chance. 71
5.12 Top 10 unigram and bigram cues with regards to coverage, in descending order, for the ARCT dataset. 72
5.13 Top 10 unigram and bigram cues with regards to coverage, in descending order, for the ARC dataset. In bold are cues whose productivity $\pi_k > 1/4$, indicating a useful cue. . . 72
5.14 Top 10 unigram and bigram cues with regards to coverage, in descending order, for the PIQA dataset. In bold are cues whose productivity $\pi_k > 1/2$, indicating a useful cue. . . 73
5.15 Top 10 unigram and bigram cues with regards to coverage, in descending order, for the CSQA dataset. In bold are cues whose productivity $\pi_k > 1/5$, indicating a useful cue. . 73

A.1 Hyper-parameters found through a search used in each experiment. 99

Acronyms

ABox	Assertional Box
AI	Artificial Intelligence
ARCT	Argument Reasoning Comprehension Task
ARC	AI2 Reasoning Challenge
CNN	Convolutional Neural Network
CSQA	CommonsenseQA
DL	Deep Learning
DNN	Deep Neural Network
GRU	Gated Recurrent Units
KG	Knowledge Base
LM	Language Model
LSTM	Long Short-Term Memory
MLM	Masked Language Modeling
ML	Machine Learning
MRC	Machine Reading Comprehension
NLI	Natural Language Inference
NLP	Natural Language Processing
NLU	Natural Language Understanding
NMT	Neural Machine Translation
NSP	Next Sentence Prediction
OWL	Web Ontology Language
PIQA	Physical Interaction Question Answering
PLL	Pseudo-log-likelihood
PMI	Pointwise Mutual Information
RNN	Recurrent Neural Network
TBox	Terminological Box

Chapter 1

Introduction

Natural Language Processing (NLP) is a sub-field of Artificial Intelligence (AI), whose overall goal is to endow devices with the capability to understand and use human language. Both AI and NLP have received increased interest in academia, evidenced by the ever increasing rate of scientific publications and the industry uptake, as the global market cap increases for such technologies. However, in parallel to this interest and applicability, the uncovering of underlying pervasive problems related to the deep neural models and datasets have raised justified concerns about their robustness and generalization capabilities. In this dissertation, an investigation into these problems is performed when state-of-the-art deep learning models are applied to a classic NLP task: commonsense reasoning.

The present chapter introduces the contents of this dissertation. Context and motivation for this body of work is presented below in Section 1.1. The scope, through objectives and contributions, is introduced in Section 1.2. To close off the chapter, the outline of the structure of this dissertation is provided in Section 1.3.

1.1 Context and Motivation

As a scientific field, Artificial Intelligence (AI) is considered to have its roots in a workshop organized by the "Founding Fathers" of the field, famously known as the Dartmouth workshop, where a study was to be carried out under a conjecture (McCarthy et al., 2006):

> The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make use of language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves.

The grand challenge of AI, outlined in this seminal workshop, is to reproduce human intelligence in computational devices. Furthermore, much like with humans, to a large extent, abstractions and concepts,

reasoning processes, and collaboration should be made resorting to (human) language, making NLP a core subfield of AI.

Language is an outstanding communication channel through which humans can share thoughts, wishes, instructions and perform the most varied array of complex mental actions. It can thus be a distinctive lens through which to evaluate the level of cognitive capabilities of AI models.

NLP has evolved into producing systems/models, usually neural networks, applied to specific tasks. This produces task-specific, highly capable models, that have even begun to be widely adopted for commercial exploitation by companies. These models are designed to maximize task-specific performance evaluation metrics, which can create a tunnel vision in research by focusing on a hill-climb approach for metric performance. This can be counter-productive to the exercise itself and to one of the long-term goals of AI: producing generally intelligent machines of the likes of human intelligence.

Recent research is beginning to uncover the downside of this hill-climbing race, as models are starting to be analyzed more broadly regarding their behaviour and knowledge. Models are brittle and form wrong generalizations by fixating on particular combinations of patterns from the input that happens to maximize performance on that particular dataset but eventually do not reflect the nature of the task meant to be conveyed or induced by that dataset.

More research into the drawbacks of state-of-the-art NLP methodology is essential, as blindly pursuing evaluation metrics does not seem to track the actual progress of the field with regards to 1) truly intelligent machines and 2) task prowess. This originates from the fact that models, having learned wrong generalizations induced by a greedy strategy, can fail when deployed in the real world and faced with an inference regime with input data from a distribution different from the distribution of the data where they were trained. This difference with the data from the real world appears to be far too large, as the strategies learned to solve the task no longer apply (Geirhos et al., 2020; Geva et al., 2019; Gururangan et al., 2018; Kaushik and Lipton, 2018; Niven and Kao, 2019; Poliak et al., 2018; Zech et al., 2018). As such, the model cannot be considered to have mastered the task if such brittleness occurs.

The motivation for this work is not to produce a state-of-the-art model that improves the capabilities of solving a particular task. Instead, it is to implement and play with already existing state-of-the-art solutions and seek to understand to what extent they might be learning the task we meant for them to. In this dissertation, to assess and understand whether models are generalizing well toward the effective learning of the task, it required establishing a methodology to stress test the models in regards to their learning, seeking to uncover possible hints of model brittleness caused by the model becoming attached to spurious patterns in the data. The choice of task the kind of task is an important decision for our aim in this dissertation, which seeks to show that despite the models' capability to solve highly complex tasks, such seems not to be a demonstration of its cognitive abilities. As such, a cognitively demanding task is preferable.

One of the founding fathers of AI, John McCarthy, was interested in and worked on commonsense reasoning. Commonsense reasoning is a quintessential human capacity as commonsense knowledge encompasses intrinsic human experience: our values and needs. Moreover, reasoning with it makes up a

significant portion of our higher-level cognition skills. Commonsense reasoning is thus an appropriate task to perform experiments with in this dissertation, given that the model should have to acquire highest level cognitive skills essential for humans. Commonsense reasoning, being an essential and innate human capacity, can be performed with any natural language. In this dissertation, however, the tasks chosen to evaluate the models with are in the English language.

This work is done in the hopes of contributing to an assessment of the current state of methodology in NLP.

1.2 Objectives and Contributions

The objective of this dissertation is to pursue a deeper understanding of how well state-of-the-art solutions generalize towards commonsense reasoning tasks. Achieving this requires performing exploratory experiments to assess the capabilities of such solutions for commonsense reasoning, which offer insights beyond what can be learned from mainstream performance metrics.

The extensive exploration for the assessment of the models provided the following contributions, present in this dissertation:

1. Implementation of mainstream state of the art deep learning models of different major types with the purpose of learning commonsense reasoning;
2. Application of the implemented models to four most prominent commonsense reasoning tasks;
3. Defining a set of stress tests that support the assessment of state of the art commonsense reasoning deep learning models;
4. Application of the set of stress tests to the implemented deep learning models;
5. Implementation and release of an open source library that supports large-scale, task-agnostic data contamination assessment tests;
6. Performing a data contamination assessment test between the datasets of four commonsense reasoning tasks and RoBERTa's pre-training datasets;
7. Empirically-based determination of the extent to which state-of-the-art models for commonsense reasoning benefit from inflated scores through shortcut learning;
8. Empirically-based determination of some factors that can be excluded from being responsible for such shortcut learning of commonsense reasoning by state-of-the-art deep learning models.

Contributions 1-8 were compiled into a research paper which has passed peer review and was accepted for publication (Branco et al., 2021a).

Contributions 1 and 2 were compiled into a separate research paper, which has also passed peer review and was accepted for publication (Branco et al., 2021b).

Contributions 5 and 6 are being compiled into a research paper, unpublished at the time of writing this dissertation.

The code and some of the data used for the experiments presented in this dissertation is provided here: https://github.com/nlx-group/study-of-commonsense-reasoning.

1.3 Dissertation Outline

This dissertation is organized into six chapters (including the current chapter).

Chapter 2 extensively covers the related work done on NLP methodology, including when applied to commonsense reasoning. In addition, research into interpretation and probing of NLP models is also covered.

Chapter 3 presents the commonsense reasoning tasks to be handled by the models.

Chapter 4 describes the implementation of the models, the adversarial attack, the data contamination package and the shortcut exploration methods.

Chapter 5 provides an analysis of the results obtained for the proposed experiments.

Chapter 6 offers concluding remarks of the previous chapters, and lays out future work.

Lastly, Appendix A provides additional information regarding training methodology.

Chapter 2

Related Work

Reasoning helps humans to cope with the world, whether when performing a complex task such as devising a plan to solve a pandemic, or simple, intuitive inferences like what will happen to a coffee mug when dropped to the ground. It helps us to infer new facts and beliefs and justify them to others. Reasoning with knowledge shared by humans, most commonly labeled as commonsense, makes up a significant portion of our experience.

Commonsense knowledge captures human values and needs. By reasoning with it, we can make sensible arguments and actions: a chef would not want to serve spoiled food, despite still having precious proteins that we require to survive, as spoiled food contains bacteria and fungi that make humans sick, and being sick is not desirable.

Endowing machines with this knowledge and reasoning capabilities allows them to understand the worldview of humans, their needs, capabilities, beliefs, and act according to that. This exercise, as exemplified before, can be verbalized. Knowledge can be written, the inference chain of a reasoning process can also be expressed in natural language. Thus, commonsense reasoning becomes an object of study for AI, and more concretely in this dissertation, of natural language processing.

Automated commonsense reasoning tasks usually involve answering questions that require different types of commonsense reasoning to be answered. In order for the model to obtain good performance and generalization, ideally, it would have to acquire the knowledge and reasoning skills.

In the early days of AI, more specifically in the Knowledge Representation field, the formalization power that logics had developed to formalize mathematical knowledge through formal languages, was seen as a candidate to represent world knowledge and reason with it. Logic programming languages, as they are called, can represent concepts and their relationships.

A popular formal language used is propositional logic, a subset of first-order logic. Knowledge is based on propositions, which evaluate to a boolean (true or false). We can have simple propositions, such as "Water is wet", which only have one symbol, called atomic propositions, but we can also create complex propositions using logical connectives (`AND`, `OR`, `IMPLIES`, `NEGATION`): "Water is wet **`AND`** water is a liquid".

Another popular family of representation languages are description logics, which offer more expressive power than propositional logic. Description logics systems are separated into two parts, the terminological box (TBox) where the concepts and roles are stated, along with the relationships between them, and the assertional box (ABox), where concrete instances/examples are created.

A nice example of a TBox descriptor with typical constructors is given in (Van Harmelen et al., 2008): the concept of a "Happy Man" – "A man that is married to a doctor, and all of whose children are either doctors or professors.".

$$\mathsf{HappyHuman} \equiv \mathsf{Human} \sqcap \neg\, \mathsf{Female} \sqcap (\exists\, \mathsf{married}.\, \mathsf{Doctor}) \sqcap (\forall\, \mathsf{hasChild}.(\mathsf{Doctor} \sqcup \mathsf{Professor}))$$

Here we have used three binary/boolean operators that have been introduced before in propositional logic. $\sqcap$ is a conjunction of concepts, previously denoted as `AND` for propositional logic, where concept A and B must be true. $\sqcup$ is the disjunction of concepts, denoted as `OR` for propositional logic, which returns true unless both concepts are false. $\neg$ is the negation of concepts (`NEGATION`), not A – every other concept but A. Quantification was also used, $\exists$ indicates an existential quantification, by means of which at least one married instance is said to exist in the denotation of Doctor. $\forall$ is a universal quantifier, every successor of hasChild must either be a Doctor or a Professor.

ABox allows us to create the data part of the system. Let us imagine we want to say we definitely know a man named John that fits our concept of a happy man, we can do it like so:

$$\mathsf{HappyMan}(\mathsf{John})$$

Web Ontology Language (OWL),[1] a language that powers the Semantic Web, is based on description logics, one of the many impactful applications that are powered by description logics systems, whose power of formalism enables robust representation and reasoning.

Declaring knowledge is a crucial step, and in order to extract conclusions, one must reason with it. Solvers are algorithms that perform a search over the declared knowledge in the search for a solution to a given query, which can be as simple as "What color is the sky".[2] Inference time using solvers increases due to the expressiveness of the language, size of the domain and the variables/restrictions involved in the query (Brachman and Levesque, 1984; Mitchell et al., 1992), which can quickly become unmanageable. SAT solvers can, at their worst, have an exponential runtime. With optimizations performed, however, runtimes become tractable for real-world use. Nonetheless, this is one of the problems associated with this chapter of AI research.

The other main problem with logical systems is the declarative effort necessary. Knowledge must be explicit, built carefully by humans. An undertaking to represent every domain of commonsense quickly

[1] https://www.w3.org/OWL/

[2] Query is written in natural language. The actual query would have to be written as a logical proposition in order for a solver to infer.

becomes a difficult task, as such would require an insurmountable amount of experts. An ongoing project, Cyc (Lenat et al., 1985), is attempting just that. There are mixed reactions regarding Cyc. It is seen as a valiant effort that goes against the statistical learning status-quo. Marvin Minsky is a supporter of Cyc and criticizes modern methods, calling them "brain-dead" (Baard, 2018). Cyc also has its fair share of skepticism and criticism regarding its usefulness. It is unclear what advances for AI has been made due to lack of public evaluation (Davis and Marcus, 2015). Pedro Domingos (Domingos, 2015) considers the project "the most notorious failure in the history of AI", stating that AI is not just a knowledge engineering problem, it requires more, and the effort is nullified by the unending amount of data needed to represent – furthermore with a system incapable of learning by itself.

Deep learning is a promising solution for the problems that plague the previous era of AI. It has brought significant advancements to NLP across the board, and for commonsense reasoning, the learning capabilities of deep learning systems means that the necessary knowledge and reasoning capabilities can be implicitly learned from natural raw text, available in abundance on the web. Deep learning is already able to outperform symbolic systems in some tasks, and a hybrid of the two, so-called neuro-symbolic systems, are also revealing to be very capable reasoners (D. Hwang et al., 2021; Bosselut et al., 2019; Kapanipathi et al., 2020; Riegel et al., 2020).

In the remaining of this chapter, commonsense reasoning is characterized, along with the methodology that has been developed in AI to endow the machines with the capabilities to perform it. Section 2.1 presents the concept of commonsense knowledge & reasoning, along with methodology to computationally represent such knowledge, and how to reason with it. Section 2.2 and Section 2.3 introduce the deep learning architectures and training methodologies that are used in this dissertation. Section 2.5 establishes the notion of Shortcut Learning, a problem that can occur when using large neural networks. In Section 2.6 methods to uncover the side effects of shortcut learning in models are presented. At last, in Section 2.7 a brief overview of neuro-symbolic systems is presented.

2.1 Commonsense Knowledge & Reasoning

Commonsense has been an object of study for almost three millennia. Philosophers in ancient Greece contemplated how humans acquire knowledge about the world, through the use of senses, and their interpretation of the perceptions. In *Meno*, the socratic dialogue by Plato (Grube et al., 1980), the process of learning is described as recollection, where knowledge is present at birth, common to all humans, and one has to search for it to rescue it from oblivion. To illustrate this, Socrates draws geometrical figures and questions a young slave, who is ignorant of formal geometry. Despite his ignorance, the slave is able to understand some geometric problems related to areas of squares, in view of arguing that it happened by recollecting knowledge already present in his mind from birth. There is two distinct concepts at work here, one is the knowledge shared by humans, which we can entitle commonsense, the other is the reasoning process. Reasoning over what we perceive of the world or our current knowledge in order to

justify actions or build new knowledge – this is what is called commonsense reasoning. Throughout time, different philosophers provide different interpretations of commonsense, further developing the concept, which became a powerful tool in philosophy.

Commonsense knowledge arises from naturally observing and thinking about the world, and to share and discuss it with other humans. It is a set of beliefs, intuitions, principles, becoming an integral part of the human experience. It covers different aspects of the world and the society, such as spatial, physical, social, temporal, psychological and many more dimensions (Davis, 2014; Liu and Singh, 2004).

Commonsense knowledge can be shared by a large group of humans (most would agree that the sky is blue) or by a more restricted group of individuals, as is the case of cultural knowledge. It can be passed on from generation to generation, and can eventually change, making it temporary.

As previously mentioned, commonsense knowledge covers different aspects of the human experience and environment. Physical knowledge such as knowing water is a liquid and unsupported things fall back to Earth. Social knowledge such as knowing that if you bump into someone on the street, that person is likely to be mad at you. Temporal knowledge such as knowing that going for a vacation takes longer than going for a walk (Zhou et al., 2019). These non-exhaustive examples demonstrate the empirical nature of the manner in which we develop commonsense knowledge, by interacting with the environment and others.

We use this knowledge to reach new conclusions (new knowledge can be built through foundational commonsense premises), to justify our actions and the actions of others (Mercier and Sperber, 2017), to understand why and how things happen (Kintsch and Van Dijk, 1978), through reasoning processes. It is essential then, for (general) intelligent machine that is to be inserted in our society to perform tasks that require real-world knowledge (on the various complex domains), to be able to reason using commonsense knowledge to make sensible inferences and decisions.

It is not enough for models to be capable of learning commonsense knowledge, they must reason with them. It is not enough to know that *placing your hands in a fire will burn you*, but also to realize that *placing your hands in a fire will burn you* ***because*** *fire is hot* ***and*** *extreme heat burns you*. This enables the model to make decisions using facts that may not be explicitly present but that can be "computed" or "reasoned" by combining known facts.

In Artificial Intelligence, commonsense reasoning has been identified as one of the greatest challenges. Marvin Minsky, one of the greatest minds in AI, recalls that in 1959, when the MIT Artificial Intelligence Project was founded along with John McCarthy, who was immensely important for AI and commonsense reasoning as well, both agreed that the most critical problem was in fact how minds perform commonsense reasoning (Minsky, 1988). Later that year McCarthy would release what is considered the first paper that laid a theoretical framework, based on logic, to perform commonsense reasoning, a program that is named "Advice Taker" (McCarthy et al., 1960). Met with strong opposition at first, McCarthy and others would follow down the line in the coming decades by formalizing logical frameworks (e.g. situational calculus) to design commonsense reasoning capable agents.

How we might represent this knowledge in a way that allows us to build models that are able to use it

to reason remains a real challenge. There are different types of reasoning that naturally require different types of representation. In an extensive overview of Commonsense Reasoning in AI (Davis and Marcus, 2015), different types of reasoning were identified as ongoing challenges:

1. **Taxonomic reasoning**, performed on taxonomies that cover different domains (e.g. WordNet (Miller, 1995)). It is useful to know that Dog is a subset of Mammal which in turn are subsets of Animals, and if we know that Scooby-Do is a dog, we can infer that it is an animal (which might sound trivial to humans, but how might machines do such inferences?);

2. **Temporal reasoning**, consisting of being able to correctly establish a timeline of events (and their relations) from textual descriptions. This is non-trivial as the relations between events are oftentimes implicit (this problem is not exclusive to this type of reasoning, but a universal problem in commonsense reasoning), making inferences difficult;

3. **Action and change**, or reasoning over the state of the world. We perform actions (events), and these change the state of the world (bumping into someone on the street will result in that person being annoyed). This can be applied in a multitude of scenarios and considering that these events can happen simultaneously, they can be probabilistic, and that they can be made by multiple actors makes this type of reasoning difficult;

4. **Qualitative reasoning**, comprising of reasoning about the direction of change in interrelated quantities, for example, one might expect that in an environment consisting of predators and prey, if the number of predators goes down due to some event, then the death rate of the prey will go down (and their population goes up). It can be even more difficult when doing qualitative reasoning with physical processes, where representation (and thus reasoning) becomes very difficult due to the complexity of the environment.

Strategies to build resources that represent this knowledge (that can later be used to reason with) can be categorized into two groups: manual and automatic. Manually building these resources is a costly effort, in terms of human resources, as experts are needed to build the resource. One such project that has been active since 1985 is Cyc (Lenat et al., 1985).

Another strategy is to automatically build these resources, removing the need for human experts. Algorithms to extract facts and relations from raw text or even graphs, such as NELL (Mitchell et al., 2015) and TransOMCS (Zhang et al., 2020), enable large scale resources to be built, albeit at the cost of lower resource quality.

Methods to perform reasoning can be arranged in three groups. 1) We can develop algorithms to directly manipulate the symbolic structures, such as inference engines. 2) Other modern methods completely disregard the use of these resources and rely instead on the hope that commonsense knowledge can be acquired by using powerful neural models to consume large amounts of raw text in an automatic manner (Section 2.3). The model can then reason with the acquired knowledge. 3) We can merge the two

approaches and create models (neuronal) that can learn not only from raw text, but also from the knowledge contained in these symbolic structures (Section 2.7). These symbolic structures bring additional information, that may be otherwise difficult to grasp from raw text alone, enriching the model overall.

In this dissertation, methods belonging to the second and third groups will be explored. In the remaining sections of this chapter, an introduction is given to the models to be used in this dissertation, along with training methodology and examples of their application.

2.2 Recurrent Neural Networks & Transformer

In commonsense reasoning, or any other task in NLP, the objective is to present a model with a piece of text (e.g. a sentence or a paragraph), which the model analyzes and makes a certain prediction. This prediction depends on the task: in translation the model must predict the translated version of the original piece of text, while for Question & Answer (Q&A) the model must pick the correct answer. The piece of text (input to the model), can be thought of as a sequence of words:

$$x = [w_0, w_1, ..., w_n]$$

In the neural age of NLP, the models are, as suggested by the name, neural networks. The input to neural networks must have a numerical representation, as opposed to the natural representation of words as strings. The most common practice is to represent a word as a continuous N-dimensional vector, called a word embedding. The first layer of the network is thus, usually, an Embedding Layer, which maps the words of the sequence to a sequence of vectors representing them. As the network learns to perform the task, the embedding layer is also adjusted (learned).

The input sequence, with the introduction of embeddings, becomes a matrix, where row vectors (size m) represent each word in the sequence (x_i):

$$X = \begin{bmatrix} w_{0,0} & w_{0,1} & ... & w_{0,m} \\ w_{1,0} & w_{1,1} & ... & w_{1,m} \\ ... & ... & ... & ... \\ w_{n,0} & w_{n,1} & ... & w_{n,m} \end{bmatrix}$$

To obtain a representation for the sequence, we need to resort to a method that can process a matrix of variable size as input, as sequences can come in various sizes. One popular neural network architecture capable of doing it are the family of Recurrent Neural Networks (RNN). RNNs, such as long short-term memory networks (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRU) (Chung et al., 2014) can handle inputs of variable sizes by processing one word at a time, updating its internal state (which functions like an internal memory) at each time step. The memory unit present in RNNs can capture long-distance dependencies in sentences. Dependencies/relationships between words in a sentence or even a document is important in NLP, and as the distance between words becomes larger

(long-distance), the harder it is to capture the dependencies. An example of the need of long-distance dependencies are anaphoras. Given a text that introduces an entity in the first sentence, a model should be able to relate a pronoun, even if used much later on in the text, with that entity.

RNNs were the favored neural networks in NLP for some time, producing fantastical results in different topics, such as in machine translation (Bahdanau et al., 2014; Sutskever et al., 2014; Wu et al., 2016; Zhang et al., 2016; Zhou et al., 2016), text classification such as machine reading (Cheng et al., 2016), sentiment analysis (Liu et al., 2016) and many other tasks (Minaee et al., 2020). RNNs pose two practical problems that would motivate the emergence of new architectures.

Computationally Expensive. RNNs are computationally expensive, as they cannot be parallelized. As previously noted, in order to compute the internal state at any given time step, it needs to have access to the internal state of the previous step. If the sequence is small enough, training can be tolerable, however, as sequence sizes grow, training becomes difficult. The sequential nature means the forward propagation time increases as the sequence length increases, hurting inference time. During training, the network activations must be kept in order to compute gradients, and with large sequence sizes, the computation graph severely increases the time and memory requirements for training. Furthermore, as the computation graph grows, vanishing gradients can become a problem even with LSTMs, requiring more training steps to converge, possibly even diverging. Training algorithms for RNNs have been devised to ease these problems (Pascanu et al., 2013; Williams and Peng, 1998; Williams and Zipser, 1989), however, they still remain a difficult and demanding type of network to train.

Long-distance dependencies. Despite the theoretical capabilities of RNNs being able to learn long-distance dependencies, such is extremely difficult in practice. Due to the recurrent nature of the architecture, vanishing and exploding gradients can cause the network to become "myopic" (forgetting the past easily) (Bengio et al., 1993, 1994). LSTM's and GRU's themselves solve this issue, however, it still falls short of optimal and may require different types of training (Martens and Sutskever, 2011).

The Transformer (Vaswani et al., 2017) (Figure 2.1) is a feed-forward model that attempts to address the shortcomings of RNNs. It can encode a variable-sized input considerably more efficiently than RNNs, and its architecture is designed so that scaling the model is straightforward. The Transformer showed strong promise from the start, introducing new state-of-the-art results in neural machine translation (NMT) in its original paper (Vaswani et al., 2017), and on later work that expanded its capabilities (Edunov et al., 2018; Liu et al., 2020; Mehta et al., 2021). New methods using the architecture also displayed brilliant performance in other Natural Language Understanding (NLU) tasks, such as Q&A, sentiment analysis, linguistic acceptability, natural language inference, and many more (Brown et al., 2020; Devlin et al., 2019; Khashabi et al., 2020; Liu et al., 2019b; Radford et al., 2018; Raffel et al., 2020), becoming the mainstream NLP architecture.

Transformer is an extremely general architecture. The only requirement is that the input is represented as continuous vectors. It has been applied to computer vision and produced state-of-the-art results (Dosovitskiy et al., 2020).

In the following subsections, key concepts necessary to understand the Transformer architecture are

introduced, namely (i) Transformer as an Encoder-Decoder model; (ii) the multi-head self-attention mechanism and how it replaces recurrence; (iii) Positional encoding; and how an overview of how it all comes together.

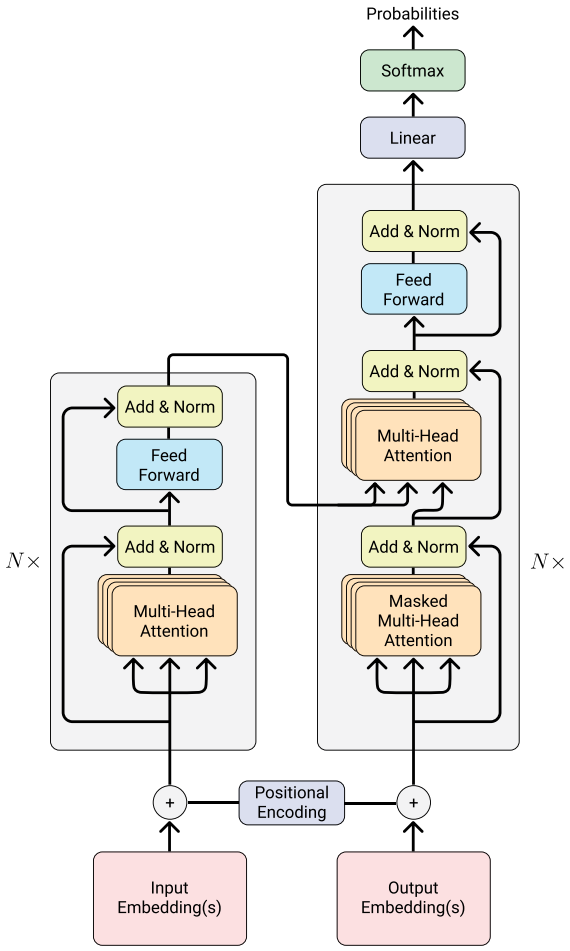


Figure 2.1: The Transformer architecture. The input embeddings are passed through N encoder blocks (left section), the output of the encoder blocks is then used by the Decoder (right section) to generate text. Figure is based on the architecture diagram presented in the original paper (Vaswani et al., 2017).

2.2.1 Encoder-Decoder Model

The concept of Encoder-Decoder models was introduced to handle the problem of variable-sized outputs, which can happen, for example, in any text generation task, such as translation, where the translated sentence size depends on the size of the input sentence and the target language.

The method was introduced to tackle machine translation using RNNs, and it was labelled as Encoder-

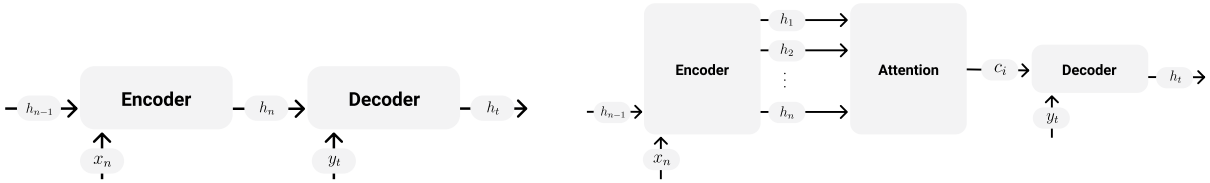
Decoder Sequence-to-Sequence (seq2seq) (Sutskever et al., 2014). The reasoning behind the name lies in the way it functions: the input sequence is encoded into a single representation using an RNN, called the Encoder; the representation is then given to a separate RNN (the decoder), which will decode an output sequence. The model thus maps an input sequence to an output sequence (sequence-to-sequence).

Figure 2.1 displays Transformers' architecture, which is divided into two subsections: the encoder blocks (on the left) and the decoder blocks (on the right). The input sequence is encoded through a succession of encoder blocks, with the representation then feeding the decoder blocks, which will produce an output prediction.

2.2.2 The Attention Mechanism

In an Seq2Seq Encoder-Decoder architecture, as pictured in Figure 2.2a, the decoder receives the encoded input sequence after the encoder has processed it, through a hidden state. This is rather demanding for the Encoder, as it has to store the relevant information of the sequence in a single vector. The longer the sequence of text, such as when encoding a whole document, the more difficult it becomes for the information to be effectively stored and the easier it becomes to forget information from the past. This problem is named the information compression problem, which plagued seq2seq models.

The attention mechanism (Bahdanau et al., 2014; Graves et al., 2014; Luong et al., 2015) was proposed as a way to alleviate the problem of information compression. The attention mechanism relies on the Encoder making available all its intermediate hidden states, representing from the first token of the sequence all the way to the last. A new vector named context vector (c_i) is computed through the weighted sum of all the hidden states. This allows the network to choose relevant information from different parts of the sequence. This is pictured in Figure 2.2b.



(a) Encoder-Decoder architecture. The Decoder receives the encoded input as the last hidden state of the Encoder (h_n).

(b) Encoder-Decoder architecture with Attention Mechanism. A context vector c_i is formed through all the Encoder hidden states, received by the Decoder.

Figure 2.2: Seq2Seq Encoder-Decoder architectures.

As previously mentioned, the idea was to decide which parts of the input should be given more attention at that precise moment through a weighted sum of all the encoder states, forming the context vector c_i (at time step i). This is performed at each decoding time step.

Given a sequence of encoder states,

$$\{h_1, h_2, \dots, h_n\}$$

The context vector is computed as the following,

$$c_i = \sum_{j=1}^{n} \alpha_{ij} h_j \tag{2.1}$$

where the weight α_{ij} is the attention weight given to h_j. The attention weight is computed by,

$$\alpha_{ij} = \frac{\mathsf{exp}(e_{ij})}{\sum_{k=1}^{n} \mathsf{exp}(e_{ik})} \tag{2.2}$$

where e_{ij} is the alignment score given by,

$$\mathsf{g}(s_{i-1}, h_j) \tag{2.3}$$

g is an alignment function, which can be learned through an MLP or can be a static function such as dot product or cosine. s_{i-1} is the decoder hidden state from the previous step. In other words, attention will compute the relevance of each encoder hidden state with respect to the previous decoder hidden state.

The Attention box in Figure 2.2b thus calculates the alignment scores between the Encoder hidden states and the previous decoder hidden state, then using the alignment scores to build a better contextualized representation of the input, for that precise decoding time step.

In the Transformer, the authors identify attention as a way to replace recurrence (RNNs). The recurrent nature of RNNs enabled the representation of a sequence of word embeddings into a single, informative representation. Attention is ultimately a weighted sum, meaning it can equally compact a sequence of word embeddings, and it can be done in parallel, unlike recurrence, which is a sequential method.

Self-attention (see explanation below), in simple terms, changes the representation of each word by computing how important the other words in the sequence are to it, altering that representation proportionally. This produces contextualized representations and allows for sequence representation without the need for recurrence. In the attention mechanism defined in Equation 2.1, the alignment is done between encoder and decoder states. Self-attention is a special case of attention, comprising an alignment between any two sequences of vectors, and the sequences can be the same, hence the term "self".

The attention for the Transformer receives as input three sequences of vectors: Queries (Q), Keys (K) and Values (V). Queries and keys are used to compute the attention weights, and values are the sequence of vectors to be combined. For example, in the case of the attention mechanism previously presented, the queries would be the decoder states, and the keys and values would be the encoder states (same sequence). The attention function is defined as follows,

$$\mathsf{Attention}(Q, K, V) = \mathsf{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{2.4}$$

where the attention weights are given by,

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{2.5}$$

and the alignment scores are computed using a static alignment function, the dot product between the queries and keys QK^T, and are scaled with a scaling factor of $\frac{1}{\sqrt{d_k}}$, where d_k is the dimension of the key vectors.

The alignment scores, much like in Equation 2.2, are normalized with a softmax.

Even after integrating the attention mechanism into the architecture, the burden of producing a contextualized and informative representation for a sequence is now mainly on the shoulders of the attention function, which can still be sub-optimal. It would be beneficial if we could perform several attention functions to represent a sequence, so as to relieve the "pressure" when using a single attention function.

A further innovation was introduced with the Transformer to attempt just that, which was the Multi-Head Attention. Instead of performing a single attention function in each attention layer, the authors opt to compose an attention layer that performs an arbitrary amount of attention functions (each function is called a Head). However, an adaptation must be done at the input level for each function, otherwise the end product of each function will be the same, as the functions do not have any parameters which could distinguish them. To solve this issue, first the queries, keys and values are linearly projected to a learned subspace, different for each attention head. Each head will thus attend to information in different representations, capturing different aspects of the sequence.

Multi-Head Attention is formally defined as,

$$\begin{aligned} \text{MultiHeadAttention}(Q, K, V) &= \text{Concat}(head_1, head_2, ..., head_n) \\ head_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.6}$$

which concatenates the output of each attention head $head_i$. The linear projection is performed by the learned matrices W_i^Q, W_i^K and W_i^V.

2.2.3 Sequence Encoding

The first step for sequence encoding is usually to tokenize the text at a word-level, such that a sequence of words is obtained. The sequence can then be represented numerically by obtaining the representation for each word from the embedding layer. The embedding layer, as previously explained, contains a matrix, where each row represents a word.

A common problem with this approach are out-of-vocabulary words. The vocabulary represented in the embedding layer is usually pre-determined by choosing the most frequent words in large corpora. Due to memory restraints, the vocabulary size is limited, usually in the low range of hundreds of thousands of

words. Inevitably during the learning process (or after), the input will contain a word which is not found in the vocabulary. A usual solution for this was to reserve a row in the matrix for an "unknown" token.

To solve the out-of-vocabulary words problem, input text can be tokenized into sub-words, instead of words. Words are decomposed into smaller tokens, following certain rules that preserves original form, depending on the technique (Kudo and Richardson, 2018; Sennrich et al., 2016; Wu et al., 2016). The decomposition can go as far as the character-level, decomposing a word into letters, meaning out-of-vocabulary words become very unlikely. In the paper that introduces the Transformer, text is tokenized using sub-word tokenization, meaning the architecture leverages embeddings which represent sub-words.

An additional component of the sequence encoding in the Transformer is positional encoding. Replacing recurrence with self-attention has the downside that positional encoding is lost, as the position of the embedding in the weighted sum is irrelevant. A positional embedding is added to the token embedding to produce the final representation to account for token position. The positional encoding method in the Transformers' original paper is absolute position static embeddings, calculated using the sine and cosine functions. These embeddings can also be learned (Gehring et al., 2017) and relative instead of absolute (now the most common) (Shaw et al., 2018).

2.2.4 The Transformer

Having introduced the concept of Encoder-Decoder (Section 2.2.1), Self-Attention (Section 2.2.2) and how sequence encoding works in the Transformer (Section 2.2.3), the inner workings of the Transformer are almost all described.

The Transformer (Figure 2.1) is comprised of stacked encoder Transformer blocks that feed representations of the input to stacked decoder Transformer blocks that can then decode the output. The difference between the encoder and decoder blocks is minimal. The Transformer block computes self-attention, with a Multi-Head Attention layer, receiving the input sequence and producing contextualized representations. These representations are then given onto a feed-forward layer (FFN) that will perform a linear transformation, which is passed along to the next block.

The difference between an encoder and decoder Transformer block is an additional Multi-Head Attention layer on the decoder block. The decoder first attends over the output that has been decoded so far, and the resulting representation is then used in an Encoder-Decoder Multi-Head Attention layer, where the queries are the decoder representations and the keys and values are the encoder representations.

Scaling with the Transformer means adjusting the number of Transformer blocks on each side, and scaling up is less expensive as it is a feed-forward model, unlike RNNs, due to their recurrent nature.

2.3 Pre-Trained Language Models

2.3.1 BERT

As Transformers continued to cement a strong position in NLP, further advances were made in a series of papers that became known as "Sesame Street" papers (Devlin et al., 2019; Liu et al., 2019b). The first in this series was the model entitled "**B**idirectional **E**ncoder **R**epresentations from **T**ransformers" (Devlin et al., 2019), or BERT. BERT makes use of the Transformer architecture, however, drops the decoder stack, being left with the stack of encoders. The goal is to create a model that specifically excels at classification tasks.

The training methodology used is named as the fine-tuning paradigm (Radford et al., 2018), which comprises two stages: (i) pre-training stage; and (ii) fine-tuning stage.

Pre-training stage. In this stage, the model undergoes a number of tasks that serve the purpose of endowing the model with some capacity to suitably process natural language (or at least a good amount of linguistic phenomena). Usually these tasks are called denoising tasks, where the model is given a corrupted input and it must recover the originally correct version. A commonly used denoising task is known as Masked Language Modeling (MLM), where a number of tokens from the input are masked, and the model must guess the masked tokens, relying on the information of the context (Figure 2.3). To make BERT learn about inter-sentence dependencies, the second pre-training task is Next Sentence Prediction (NSP), where given two sentences, BERT must predict whether the second follows from the first. Sentences are separated by a special token, the separation token (`[SEP]`), and a segment embedding (for each different segment of the input, in this case, segment A and B) is added to the representations of tokens in their respective segments.

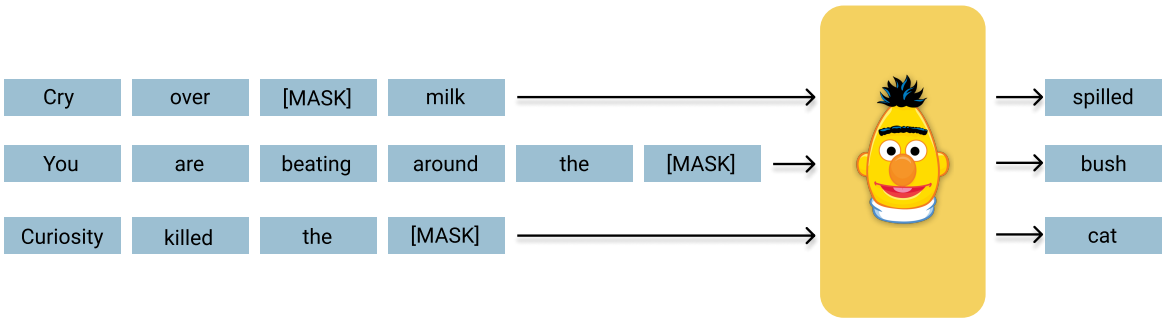


Figure 2.3: BERT performing masked language modeling. The corrupted tokens are replaced with a `[MASK]` token, and the model must guess the correct words.

MLM is a token classification task and NSP is a sentence classification task. For token classification tasks, the hidden state representing the token is fed into a classification or MLM head (which is just a classification head with the number of classes equaling the length of the vocabulary). A classification

head is composed of a dropout at the input, a linear layer and finally a softmax over the classes. For sentence classification tasks, BERT prefixes the sentence with a special token, the classification (`[CLS]`) token, whose state is fed to a classification head to classify the given sentence(s).

Fine-tuning stage. After the pre-training procedure, the model is then used to learn a specific task, in what is called the fine-tune stage. Here, the model already possesses a vast amount of knowledge, learning the downstream task more effectively when compared to a random initialization with no pre-training. At the time of its release, BERT was the state-of-the-art model for text classification.

BERT models, and other variants, are pre-trained with large amounts of raw text, making it computationally costly. Pre-trained models are usually openly available, making them a tool that can be reused by the community. The parameters in the model retain knowledge from pre-training (Roberts et al., 2020) that is later used during fine-tuning and inference.

2.3.2 Other Variants

Transformers can be grouped as follows: encoder-only, decoder-only and encoder-decoder. BERT (Section 2.3.1) is an encoder-only architecture. GPT (Radford et al., 2018), and its subsequent iterations, are decoder-only architectures. Pre-trained also with large amounts of text, the excellence of GPT lies on its generational power, able to generate human-like text. Its most recent version, GPT-3 (Brown et al., 2020), is one of the *known* largest neural networks trained, with an impressive 175B parameters. It produces inspiring results in a multitude of tasks with no fine-tuning (strictly in-context learning). Its text generation capabilities also improved.

The impressive size and performance of the GPT "series" has motivated research on the scaling power of the Transformer. Recent studies suggest that the scaling power follows a power law (Kaplan et al., 2020; Henighan et al., 2020) (optimum size with respect to compute power), with no asymptote in sight, motivating larger scale models and the use of larger computer networks.

Another encoder-only Transformer, derivative of BERT, is the RoBERTa (**R**obustly **o**ptimized **BERT** **a**pproach) (Liu et al., 2019b) model, that was conceptualized from a study of optimization of BERT models. RoBERTa is pre-trained with an MLM task only, removing the NSP task from the pre-training phase. Other optimizations are implemented, such as removing the segment embeddings from the input embeddings composition. RoBERTa is pre-trained on five raw text corpora, totaling around 160GB of text. The authors evaluate both RoBERTa and BERT on all GLUE (Wang et al., 2018) (General Language Understanding Evaluation) tasks, and found that RoBERTa surpasses BERT in all of them. The changes performed resulted in a much more capable classification model compared to BERT. A more complete description of RoBERTa is given in Section 4.1.1.

The remaining category is the encoder-decoder, in which T5 (Raffel et al., 2020) (**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer) belongs to. Given that it is a text-to-text framework, it requires no special tokens. The input should be as natural as possible, and the output is also generated text, possible due to the decoder. This flexibility, due to both the input and output being text, facilitates its usability, as T5 can

accommodate different tasks with no additional changes to the network itself. In fact, in its introductory paper, T5 performs multi-task learning of all GLUE tasks, obtaining state-of-the-art results in each.

T5 is pre-trained with a denoising task called span prediction. Instead of masking single tokens, contiguous spans are masked, and the model must predict the tokens that were masked. It is shown that this leads to a better performing model on downstream tasks (Raffel et al., 2020). The model was pre-trained with C4 (Colossal Clean Crawled Corpus), which after undergoing filtering (for quality reasons) totals around 745GB of raw text. More details about T5 are given in Section 4.1.3.

2.4 Commonsense Reasoning with Transformers

In a previous study (Zhou et al., 2020), the knowledge and reasoning capabilities of different pre-trained Transformer-based language models (LM), including RoBERTa, were studied on different commonsense reasoning tasks, some of them adopted in the present dissertation. The tasks are framed as sentence scoring tasks. So, for a given example (e.g. a question and several possible answers), the perplexity of the LM when presented a pair of `(question, answer)` is measured, and the pair with the lowest perplexity (thus being the one that makes the most sense) is the response of the model. This allows the authors to "probe" the knowledge contained in the model without fine-tuning on the task.

This can be realized by summing the log-likelihood of each word in the context (what can be called the pseudo-log-likelihood (PLL) of the input). To do this, each word is iteratively masked and the probability for the original word as perceived by the model is retrieved. The authors define this procedure as the sentence score:

$$Score(S) = \frac{\sum_{k=1}^{n} log(P_\theta(w_k|context_k))}{n} \tag{2.7}$$

The sum of log-likelihoods is normalized by the sequence length, denoted as n.

They found that the models consistently performed better than random (despite being close to random in some tasks), however, well below human performance. After devising a robustness test, they found that the models are brittle and show no consistency, likely relying on spurious correlations to perform these tasks instead of performing the actual task.

The evaluation methodology with sentence scoring can be extended and used as an alternative to the conventional method of fine-tuning. (Tamborrino et al., 2020) frame the problem as a plausibility ranking problem, where from one premise and several hypotheses, we aim to pick the most plausible premise-hypothesis pair (this might also be described as an entailment task). Here the score of a given `(premise, hypothesis)` is given by the following:

$$S_i^P = \sum_{k=1}^{L_p} log\left[P(p^{(k)}|s_i^{p^{(k)}})\right] \tag{2.8}$$

Where $s_i^{p^{(k)}}$ is the sentence with the token $p^{(k)}$ masked, and L_p is the sequence length of the premise. There is no need for normalization as the sequence length will be constant for all hypotheses. In this method, only the tokens belonging to the premise are masked, to lessen the effect of statistical cues in the data, as it was observed by the authors.

This method is computationally expensive, as the computation graph must be stored for the forward passes of each example, which may have M hypotheses, and each hypothesis may have N forward passes. There are other ways to approximate the PLL score, which can prove to be effective and cheaper alternatives (Salazar et al., 2020).

Ranking task formulation has also been found to help in tasks such as natural language inference. In (Liu et al., 2019a), each `(question, answer)` pair is separately passed through the model. The state of each `[CLS]` token is passed through a linear layer, producing a relevance score for each pair. The pair with the maximum relevance score is the candidate answer. This is a cheaper alternative to the previous ranking methods and produces near state-of-the-art results with RoBERTa. This method is denoted as a pairwise ranking task.

2.5 Shortcut Learning

For as much as the newest NLP techniques, such as the ones shown in Section 2.3, bring about better performance on downstream tasks unlike any others that we had previously seen, the manner in which they do has been called into question. This questioning trend has spanned across different topics in deep learning, such as Computer Vision, Decision-Making Algorithms, Medical Imaging, Clinical Predictions, and many more (Geirhos et al., 2020; D'Amour et al., 2020).

Complex neural networks that rule deep learning, and NLP, are difficult to interpret, appearing as a blackbox. This has motivated research into their interpretability (Rogers et al., 2020; Chakraborty et al., 2017). For example, establishing a relationship between specific input features and specific output is not attainable. There are techniques that rely on gradient information (Han et al., 2020; Simonyan et al., 2013; Smilkov et al., 2017; Sundararajan et al., 2017), attention information (Clark et al., 2019; Lee et al., 2017; Vig and Belinkov, 2019; Wu et al., 2020; Xu et al., 2015) to infer some connection, but the topic still remains in its infancy.

This opacity leaves researchers unsatisfied, but also curious. Some hints have started to appear that sheds light on some (unwanted) learning patterns, which has been regarded as Shortcut Learning (Geirhos et al., 2020). Loss functions (cross-entropy), DNN architectures and optimizers are biased towards simple functions (De Palma et al., 2018; Jacobsen et al., 2018; Sun and Nielsen, 2019; Valle-Pérez et al., 2018; Wu et al., 2017), and those functions can rely on features and patterns that are superficial or even spurious, which fails spectacularly when tested on out-of-distribution (o.o.d) data.

Examples of shortcut learning can be found across modalities (vision, language, ...). A study (Zech et al., 2018) found that in a Convolutional Neural Network (CNN) trained to identify pneumonia in pa-

tients' chest radiographs, such spurious patterns were used to achieve high performance. The shortcut taken was the following: pneumonia had more incidence in certain hospitals, so the model learned to identify hospitals from metal tokens present in the radiography, instead of learning patterns associated with the disease itself. By identifying hospitals, the model can then abuse the unbalance to obtain better performance. When tested on o.o.d data, the model performed **bad**. More recently, CNN's were applied to radiographs of patients to aid the detection of COVID-19 (Wang et al., 2020; Hemdan et al., 2020; Ozturk et al., 2020), achieving high performance. A new study found that these models are only required to look at the borders of the radiography to achieve high performance, completely disregarding the pathology of COVID-19 (DeGrave et al., 2021).

In NLP, studies have been conducted to probe for this phenomenon. The Argument Reasoning Comprehension Task (ARCT) (Habernal et al., 2018a) is one such task where pre-trained language models shined, obtaining 77% test accuracy, slightly below untrained humans, with 80% test accuracy. This task was one of the tasks in SemEval 2018. A subsequent study (Niven and Kao, 2019) found that the presence of "not" and other high frequency words such as "is", "do" and "are" was highly correlated with the output, obtaining above random performance with just "not". By adding adversarial examples in a way that balanced this correlation, the strong signal disappeared and the performance dropped to random chance (50%).

In another paper, an adversarial attack (Section 2.6) is done in a machine reading comprehension task by introducing unrelated sentences in the input, such that it confuses the models but do not contradict the correct answer (Figure 2.4). Through this exercise, a new adversarial test set is generated. The accuracy on the adversarial test set was about a half that of the accuracy on the original test set.

Machine reading comprehension (MRC) models appear not to do much reading (Kaushik and Lipton, 2018), as models can perform reasonably well when given only a passage (without the question) or a passage with a randomly assigned question. This demonstrates that the models are not performing the task intended to be conveyed by the datasets; instead, they pick up on some signal present in the data to optimize performance.

Large-scale natural language inference (NLI) datasets also exhibit these problems. NLI is a classic NLP task, where given a premise, the model must determine whether a hypothesis is true (entailment), false (contradiction), or undetermined (neutral). NLI datasets possess linguistic phenomena that correlate well with certain classes, whereby even simple classifier models can perform well by only looking at the hypothesis (Gururangan et al., 2018; Poliak et al., 2018).

A study (Geva et al., 2019) on the annotator bias on natural language understanding datasets (including CommonsenseQA – Section 3.4) has found that annotator bias is evident and models did not generalize well across annotators. Moreover, a model can exploit those biases to inflate its performance if the same annotator is present in the training and testset.

The use of non-robust features (or shortcuts) naturally arise from the training methodology of neural networks, which are optimized to maximize the performance in a given dataset distribution. A good description of this is given in (Ilyas et al., 2019) (more on adversarial perturbations in Section 2.6):

Article: Super Bowl 50
Paragraph: "*Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*"
Question: "*What is the name of the quarterback who was 38 in Super Bowl XXXIII?*"
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

Figure 2.4: Reproduced from (Jia and Liang, 2017). The addition of an unrelated fact in the paragraph made the model change its original prediction, indicating a shallow understanding.

> Recall that we usually train classifiers to solely maximize (distributional) accuracy. Consequently, classifiers tend to use any available signal to do so, even those that look incomprehensible to humans. After all, the presence of "a tail" or "ears" is no more natural to a classifier than any other equally predictive feature. In fact, we find that standard ML datasets do admit highly predictive yet imperceptible features. We posit that our models learn to rely on these "non-robust" features, leading to adversarial perturbations that exploit this dependence.

Despite the theoretical ability to learn any arbitrary function, in practice the concepts that determine a correct label are just as valid to the model as any other pattern present in the data.

Note that humans are not immune to using shortcuts and shallow understanding to perform tasks. Students oftentimes try to memorize (fully or partially) the contents of a class ahead of exams, which can fail when faced with questions that require a deeper understanding or have different framing from the ones they saw during practice (NAP, 1999).

Addressing this issue, not only in this dissertation but in the field overall, is important in two aspects. On a more practical note, when applying NLP methods to solve real world problems, the potential shift in the distribution of inputs may cause the model to fail, reducing its usefulness. On a more scientific aspect, more relevant for the goal of this dissertation, models that rely on non-robust features and spurious generalizations defeat the long-term purpose of AI – which is to produce a general intelligence machine, rational enough to avoid making such surface level mistakes.

2.6 Adversarial Attacks

In Section 2.5, the problem of shortcut learning is introduced, but also the notion that perturbation in the network (e.g. in the form of hiding or altering features) can significantly diminish the capabilities of these models. The practice of minimally perturbing the input examples, while preserving their "semantics", to lead models to change their prediction is called adversarial attacks, which produce adversarial examples.

Early work (Goodfellow et al., 2015; Szegedy et al., 2013) in adversarial attacks against DNN's focused on computer vision. They found that tiny non-random (and semantic preserving) perturbations in the image, imperceptible to the human eye, could alter the predictions of the network (Figure 2.5).

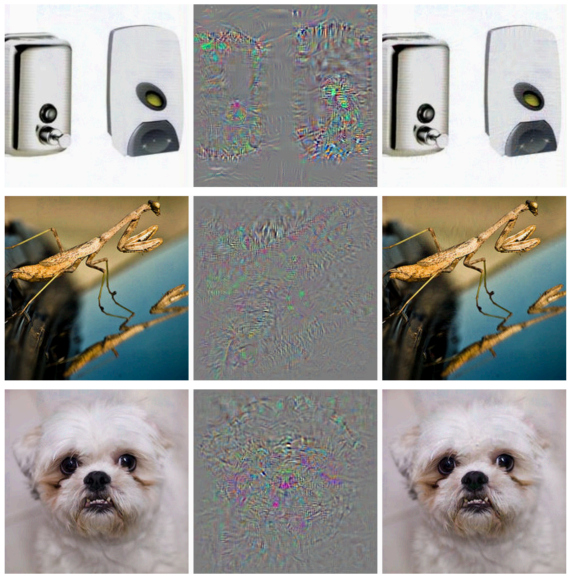


Figure 2.5: Reproduced from (Szegedy et al., 2013). Left column: original image that was correctly classified. Right column: adversarial example produced with small perturbations. Middle column: difference between the two images. All the images on the right were wrongly classified as an ostrich.

Adversarial attacks (with adversarial examples) can be shown to be possible due to the presence of non-robust features, directly targeting them (Ilyas et al., 2019). As such, they can be a valuable tool to show brittleness in models, which can (or not) be attributed to shortcuts found during learning. While the decrease in performance is evidence of frailty, it is not definitive proof of shortcut learning.

The input perturbation approach applied to images can also be transferred to NLP. Inserting, deleting or replacing spans of text in the input, in a manner that preserves the semantics of the original input, is the ultimate goal of any algorithm that performs adversarial attacks.

It is difficult to devise automatic methods that produce semantic preserving adversarial examples with low error rates. Even state-of-the-art techniques struggle to preserve semantics (Morris et al., 2020a). Most algorithms focus on replacing important words with synonyms, each differing in the way that they

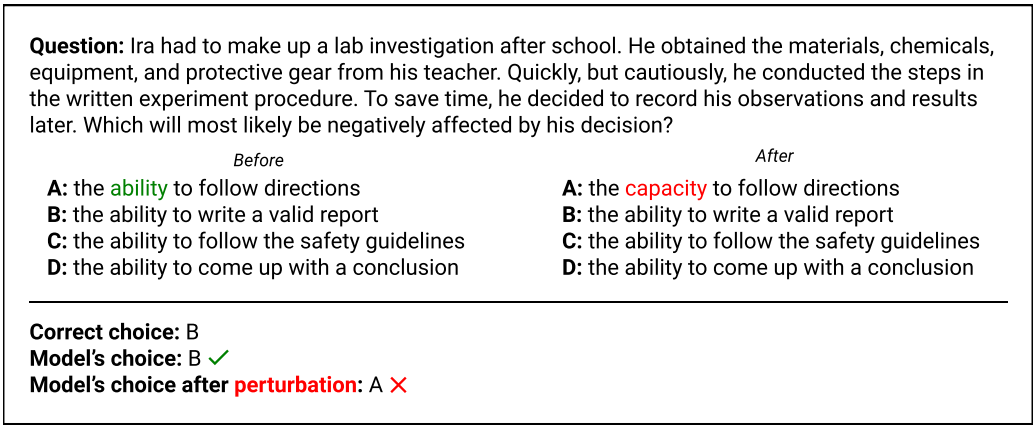


Figure 2.6: An adversarial example produced with TextFooler (Jin et al., 2020) on the ARC dataset (Clark et al., 2018) (Section 3.2), targeting a fine-tuned RoBERTa on the task. Simply changing *ability* to *capacity* in option A is enough to make the model predict A instead of its original prediction B.

select candidate words/spans.

Use of lexical ontologies such as WordNet (Ren et al., 2019) and HowNet (Zang et al., 2020) have been succesfully used to find synonym words to produce adversarial examples. Other popular methods (Alzantot et al., 2018; Kuleshov et al., 2018; Li et al., 2018) utilize the semantics encoded in pre-trained word embeddings to find substitute words (e.g. the words with above a certain threshold for cosine similarity). Both of these methods can fail to find words that fit in the context, as the comparisons are done at word level. To address this, some of these methods introduce a threshold on sentence semantic similarity between the original sentence and the adversarial sentence, calculated using a sentence encoder, such as Universal Sentence Encoder (Cer et al., 2018). One such method that uses word embeddings and sentence encoders is TextFooler (Jin et al., 2020), which will later be used in this dissertation. An adversarial example produced by this system is given in Figure 2.6.

To produce more context-aware adversarial examples, methods have been devised (Garg and Ramakrishnan, 2020; Li et al., 2020a,b) to take advantage of the contextual power of pre-trained language models, such as BERT and RoBERTa, to find word replacements.

2.7 Neuro-Symbolic Systems

Neuro-symbolic systems, alluded to Section 2.1, seek to combine neural networks with symbolic systems, with the latter containing rich knowledge represented in knowledge bases (KB).

Interest in neuro-symbolic systems for NLU has been slow but steady. In the neural era, early attempts at transferring this knowledge made use of Graph Embedding techniques to represent concepts from these knowledge bases as vectors, which could then be used to initialize the embedding layer of a neural

network. Techniques such as Node2Vec (Grover and Leskovec, 2016), TransE (Bordes et al., 2013) and WordNet Embeddings (Branco et al., 2019; Saedi et al., 2018) can leverage the nature of graph structures to produce embeddings for the nodes through the strength of the relations between them. Rich lexical knowledge coming from ontologies such as WordNet can then be used as a way to enrich neural networks in a neuro-symbolic system (Salawa et al., 2019).

Earlier attempts to integrate KB's with the Transformer made use of the self-attention system to merge embeddings from external KB's with the internal state from the Transformer block (Peters et al., 2019; Yang et al., 2019; Zhang et al., 2019). This usually requires the addition of a module that selects relevant entities, fetches their embeddings and includes them in a self-attention operation with the sentence word states.

These approaches are limited in three ways. For one, we have additional parameters to be learned in order to merge the KG entity embeddings, making the search space for optimum performance rather expensive.

Second, after training the model, if we want to insert an additional entity because the KG has been extended, re-training may prove costly.

Lastly, holding the embedding matrix of entities quickly becomes a problem, especially as some popular and extensive KB's can encode millions of entities, being prohibitively expensive to hold in memory.

In an interesting new wave of commonsense reasoning (inspired by GOFAI[3]), more complex knowledge bases and methods are being developed to tackle the problem. Open resources for commonsense reasoning, whether manually built or automatically retrieved, tend to encode taxonomic knowledge, which is a subset of the types of commonsense knowledge.

In ATOMIC (Hwang et al., 2020; Sap et al., 2019), we find an approach that seeks to improve reasoning by encoding causal and inferential knowledge. It is important for an agent/model to reason about what might be the causes for a certain event to happen, and given that it did, what we can infer from it. In its most recent update, ATOMIC encodes knowledge of social-interactions, physical-entity relations, and event-centered relations. ATOMIC is then used as a resource to build a dataset to enrich pre-trained language models by fine-tuning them on a tail generation task (Bosselut et al., 2019; Hwang et al., 2020). Tail generation task is designed to enable models to learn the knowledge contained in a KB. The task consists in presenting the model with a concept A and a relation, and the model must generate a concept B that has such relationship with concept A. This neuro-symbolic approach, which enriches models through a generation task, is named Commonsense Transformers (COMET). COMET is described in more detail in Section 4.1.4.

The reasoning behind the generation task lies in the fact that the universe of commonsense knowledge is so vast that we cannot hope to build a resource that is complete from training on piles of raw text (in pre-training) or from smaller, specifically labeled datasets. Thus, we need our models to be able to integrate

[3]Good old-Fashioned AI

more and broader knowledge other than the knowledge that incidentally must be contained in (labeled) text datasets. In the paper that introduces ATOMIC's latest version, ATOMIC2020 (Hwang et al., 2020), pre-trained language models were tasked with generating tail concepts by being prompted with a head concept and a relation, without any fine-tuning. The pre-trained models struggled to generate the correct tail entities, as they may not possess the capabilities required to express such knowledge. COMET models were then evaluated on the same task and could successfully generate the correct tail entities.

2.8 Summary

This chapter laid out the related work necessary to grasp the work conducted in this dissertation.

Commonsense knowledge, which captures human values and needs, is an essential aspect of the human experience and one which is helpful for machines. Endowing them with this knowledge allows them to understand our worldview, our needs, capabilities and beliefs. Furthermore, reasoning with such knowledge allows humans, and hopefully the machines, to infer new knowledge and to act when facing problems unseen before.

Commonsense reasoning has been an object of study in AI since its inception. Automated commonsense reasoning was first realized using logic programming, in what is labeled the symbolic paradigm of AI. Knowledge is represented as propositions, which can be connected to form complex propositions. This knowledge can then be used to answer questions, which are formulated as queries, according to the syntax of the logic programming language. A search algorithm, named solver, will search through the encoded knowledge to find the query's suitable answer(s).

Logic programming has two inherent problems which motivate the adoption of deep learning for automated commonsense reasoning: (i) complex queries are demanding for solvers and can become intractable in terms of time, being unable to be resolved in a timely manner; (ii) Knowledge representation is accomplished through the efforts of humans, commonly experts in the domain. Representing every domain of commonsense becomes unfeasible, as it requires an overwhelming amount of experts. Deep learning offers the promise of learning such knowledge from natural raw text, which can then be used to reason with.

The mainstream deep learning approach for natural language processing tasks currently is the Transformer. The Transformer is a sequence-to-sequence neural network model, meaning its input and output are sequences of text. It comprises two parts: the Encoder, which encodes the input sequence, and the Decoder, which decodes the output sequence based on the input. The Transformer uses a self-attention mechanism to relate each token of a given sequence to provide contextual information.

It has become standard practice to divide the training of these networks into two stages. In the first stage, named the pre-training, the model is trained on language modelling tasks using a large corpus of raw text, endowing it with some capacity to suitably process natural language (or at least a good amount of linguistic phenomena). In a second stage, named the fine-tuning, the model, which now possesses

enhanced priors, is tasked with learning to solve a specific natural language problem.

Different variants of the Transformer exist, trained with this pre-train-then-fine-tune methodology. BERT and RoBERTa are Encoder-only Transformers that excel at classification tasks. GPT is a Decoder-only transformer that excels at generation tasks. T5 is an Encoder-Decoder Transformer (standard) which frames all tasks in a textual manner, named the text-to-text methodology.

These models can be tasked with learning commonsense reasoning tasks. Previous work has demonstrated that they acquire some commonsense knowledge during the pre-training stage, which can be reasoned with to solve tasks. When fine-tuned on commonsense reasoning tasks, these models can achieve good results and are starting to approximate human ability.

Despite the success of these models, the manner in which they achieve them is beginning to be unveiled in recent research. Similarly to other neural networks, the Transformer seems to be picking up on spurious signals in the data, so-called shortcuts, which may help solve the dataset, but not the task in a general fashion. For example, in computer vision, some models tasked with identifying pneumonia in chest radiographs learned to identify hospitals instead through tokens present in the radiographs. For COVID-19 identification, some models looked at the borders of the radiography to achieve high performance. In NLP, the way in which certain words are distributed in the dataset can provide the models with hints that help resolve the task.

A class of algorithms that can expose the negative influence of these shortcuts are “Adversarial Attacks”. Adversarial attack algorithms perturb the input to the network, e.g. by replacing words in the input with their synonym, in a way that preserves the semantics of the original input and is grammatically correct. These perturbations can lead models astray, considerably dropping their performance. This is a potential indication that the models are using shortcuts.

Lastly, a promising technique, which aims to “merge” the symbolic with the neural paradigm, to build more robust and knowledgeable models. Common between most neuro-symbolic methods is the idea that knowledge contained in knowledge bases, which became a valuable tool during the symbolic paradigm, are transferred to or accessed by neural networks, affording them with systematic knowledge. The resulting models should have more refined priors than regular neural networks, which could help avoid forming wrong generalizations by utilizing shortcuts.

Chapter 3

Tasks

Tasks that require the model to perform some commonsense reasoning are tasks of the type that involve commonsense reasoning when performed by humans. Namely tasks required to reach new conclusions/-facts and to piece knowledge together to explain events.

In NLP literature, these types of tasks commonly fall under two categories: multiple-choice question answering (Q&A) and machine reading comprehension. In this dissertation, to simplify the implementation and focus on one type of task for the extensive experimentation phase, I select four, Q&A only, commonsense reasoning tasks, covering different topics and different reasoning types, setting a demanding environment to probe different language models' knowledge and ability.

3.1 Argument Reasoning Comprehension Task

The Argument Reasoning Comprehension Task (ARCT) (Habernal et al., 2018a) is a task developed to test a model's ability to reason through arguments. ARCT requires comprehension of an argument structure and how its parts relate with each other, which to humans, requires language understanding, logic skills and also commonsense.

The underlying structure of an argument, whose uncovering dates back to Aristotle and his study of argumentation (especially important with the definition of syllogisms), is defined as a series of premises that support a respective claim/conclusion. In the example below (E1), a major premise (*all humans are mortal*) and a minor premise (*Socrates is human*) allow for the conclusion that *Socrates is mortal*.

(E1)
All humans are mortal
Socrates is human.

Therefore, Socrates is mortal

Toulmin decomposes an argument further, indicating a structure composed of six components (Toul-

min, 1958). There are three fundamental components in his approach: grounds (also known as reasons or premises), warrants and the claim. A warrant is itself a (implicit) premise, and plays the role of linking the explicit premises and the claim, such that the claim must logically follow from the grounds, creating a reasoning chain: *Reason* $\rightarrow$ *Warrant* $\rightarrow$ *Claim*.

Warrants are, however, primarily implicit in arguments (Freeman, 2011). They are left to the addressee to be figured out, as it is presupposed that it is common knowledge shared between the addressee and the speaker. The addressee relies on common sense to identify them, requiring commonsense reasoning to perform warrant identification. An example of a warrant can be given by simplifying example E1 in the following manner:

(E2)
Socrates is human.
———————————
Therefore, Socrates is mortal

Notice how the first premise was not provided, yet the argument still makes sense. The missing warrant is commonsense as we know that all humans are mortal.

ARCT is a conceptually simple task: given a reason (R) and a claim (C), select the correct warrant from two options such that the correct warrant supports the deductive inference $R \rightarrow C$. Figure 3.1 provides three examples from the ARCT dataset.

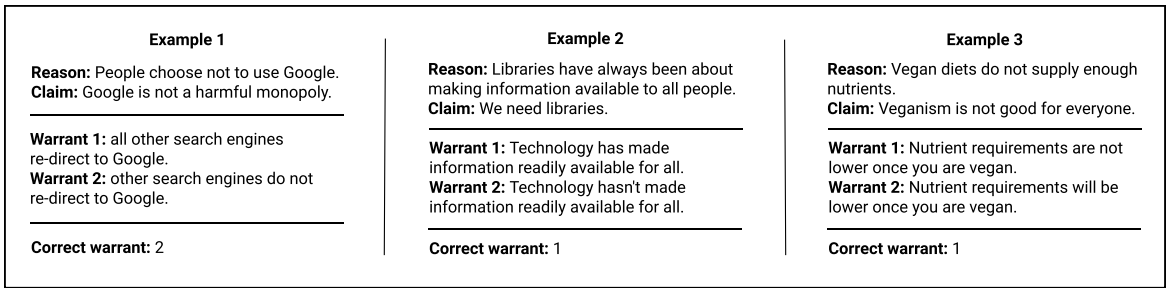


Figure 3.1: Three examples from the ARCT dataset.

The dataset was constructed with data from the *Room for Debate*, a section in the New York Times,[1] where knowledgeable contributors participate in debates regarding contemporary issues. The section has an editorial board and moderation, guaranteeing a good quality of data compared to open forums such as Create Debate.[2] The authors select 188 debates of controversial issues and used crowdworkers (referred as turkers) from Amazon Mechanical Turk to perform several annotation, summarization and validation tasks, comprising an eight step pipeline. The crowdsourcing produced 1970 dataset instances.

The dataset was used to create a shared task (Habernal et al., 2018b) at SemEval 2018,[3] where re-

[1] https://www.nytimes.com/roomfordebate
[2] https://www.createdebate.com/
[3] https://alt.qcri.org/semeval2018/

searchers and practitioners can participate in a competition by devising their own system to tackle the task.

The original dataset used in SemEval 2018 has been flagged with data artifacts enabling spurious correlation in (Niven and Kao, 2019) (as described in Section 2.5). The authors identified key words (such as "not") that were being successfully used by the models as shortcuts to determine the target class.

After expunging the dataset from these spurious cues, BERT's performance dropped to random chance. A reproduction work (Rodrigues et al., 2020) empirically confirmed the drop in test scores when using the revised dataset for the top scoring models participating in SemEval 2018.

For this dissertation, the revised dataset will be used instead of the original. The revised dataset size is described in Table 3.1.

Train	Dev	Test	Total
2420	632	888	3940

Table 3.1: Number of examples in each dataset partition.

3.2 AI2 Reasoning Challenge

The AI2 Reasoning Challenge (ARC) (Clark et al., 2018) is a multiple-choice natural science question answering task made from a collection of questions from 3rd to 9th-grade science exams, mostly in the US.

The motivation behind this reasoning challenge comes from the fact that most previous Q&A datasets do not require deep comprehension, instead relying primarily on surface-level cues to be resolved. To this end, the authors collected natural science exams, which in theory require different types of reasoning and knowledge (a good portion attributed to world knowledge & commonsense).

SAT, the standardized test used for college admissions in the US, has been used as a Grand Challenge in AI (Bayer et al., 2005), as the tests require many of the capabilities desired in a cognitive system. Standardized tests are thus a valuable resource to build robust tasks.

The dataset was compiled from various sources, with most of the questions coming from an undisclosed partner. Other sources are mostly programs and education departments from different states. Three examples of the dataset are provided in Figure 3.2.

After analyzing a sample of 100 examples, the authors consider that models are required to learn different types of knowledge: Definitions, facts & properties, structure, processes & causal, teleology/purpose, algebraic, experiments, and spatial/kinematic. It also requires different types of reasoning: question logic, linguistic matching, multihop, comparison, algebraic, hypothetical/counterfactual, explanation/meta-reasoning, spatial/kinematic and analogy.

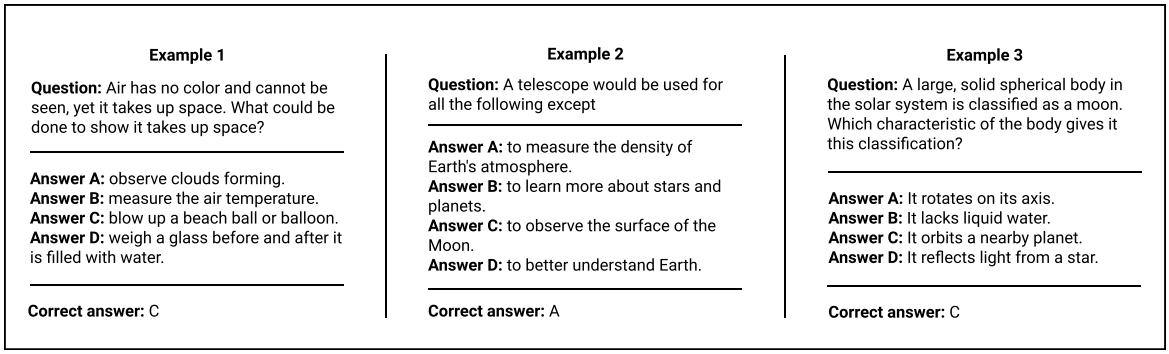


Figure 3.2: Three examples from the ARC dataset.

To guarantee that the task is challenging for models, the dataset is divided into two sets: the "Easy" and "Challenge" sets. To partition the dataset, the authors implement two trivial solvers that act as filters, whereby if an example can be solved with these solvers, it will be placed in the Easy set; otherwise, it will be placed in the Challenge set.

The solvers used to determine hard questions were the following:

Information Retrieval Solver. Identifying a question and answer option in a large web-based text corpus, with a certain confidence, using a query engine. The sentence with the highest overlapping score is chosen as the prediction.

Pointwise Mutual Information (PMI) Solver. PMI (Church and Hanks, 1990) establishes associative strength between two n-grams in a corpus. This solver uses PMI to establish associative strength between the question and each answer option, picking the option with the largest average PMI (lexical cues).

The final filtered dataset size is described in Table 3.2.

	Challenge	Easy	Total
Train	1119	2251	3370
Dev	299	570	869
Test	1172	2376	3548
Total	2590	5197	7787

Table 3.2: Number of examples in each dataset partition.

3.3 Physical Interaction: Question Answering

Physical commonsense knowledge is a critical dimension of commonsense that is in constant use in our lives. Learning about the properties of objects starts at an early age of exploration. A study conducted with 5-month-old infants (Hespos and Spelke, 2004) found that infants develop a conceptual representation of tight vs. loose-fitting of one object to another, indicating that object mechanics can be learned before language acquisition (although physical knowledge is eventually enhanced with language).

For humans, acquiring physical commonsense knowledge is part of the human experience. We can interact with the world, manipulate objects and figure out how we might use them to solve problems. This process is called grounding (Bisk et al., 2020a). As of yet, models cannot interact with the world to learn these properties, like humans do. It stands as a real challenge for models to acquire physical knowledge from raw text only.

Physical Interaction Question Answering (Bisk et al., 2020b) (PIQA) tests the physical commonsense knowledge of models. The task presents the model a goal, mostly an everyday situation that a human might want to accomplish, and two possible solutions to attain the goal. The model has to pick the solution that makes more sense. Three dataset examples are presented in Figure 3.3.

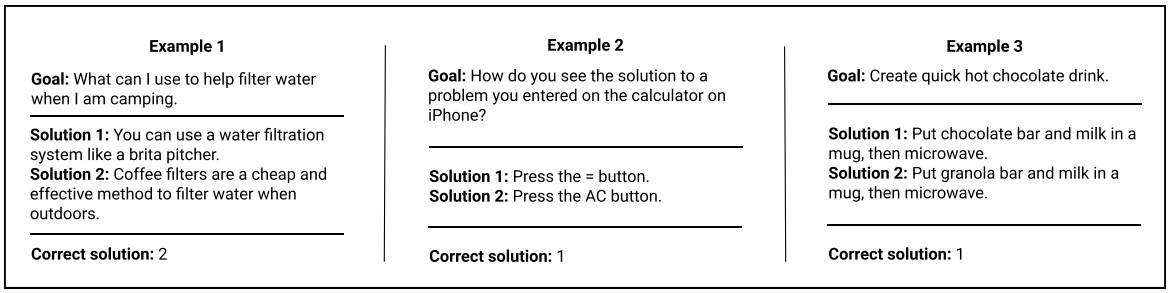


Figure 3.3: Three examples from the PIQA dataset.

The inspiration (and data source) for the task came from instructables.com, a website that provides instructions to build, craft or manipulate objects (including cooking). The contents of each instructable can be used to extract valuable physical knowledge regarding various objects. The authors made use of the crowdsource platform Amazon Mechanical Turk to produce the dataset examples (Figure 3.4).

The turkers receive an instructable as inspiration. From the instructable contents, they are asked to produce a goal, a solution, and a trick, which is similar to the proposed solution but with a trick that makes it wrong. Ideally, the goal requires physical knowledge to answer. After collecting the examples, the authors performed a pipeline of artifact removal from examples that are biased towards a particular target. The dataset size is described in Table 3.3.

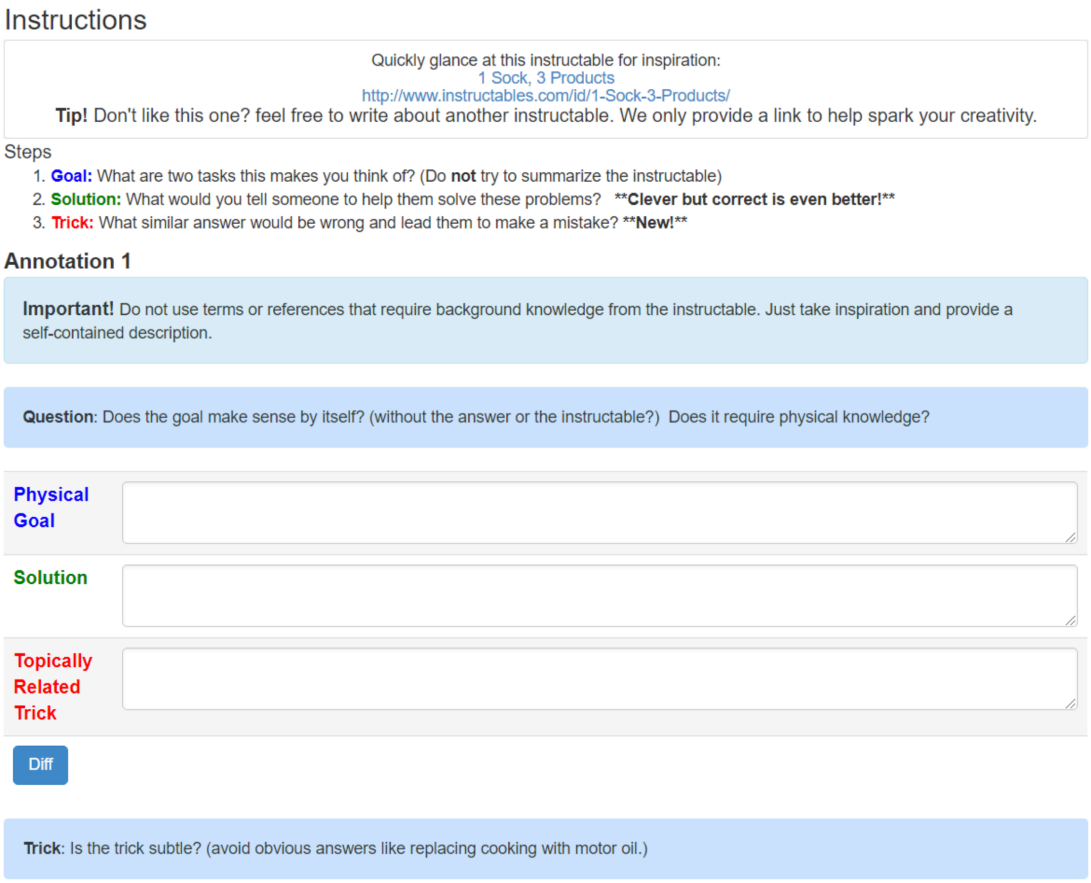


Figure 3.4: Reproduced from (Bisk et al., 2020b). Instructions provided to turkers on Amazon Mechanical Turk.

3.4 CommonsenseQA

The tasks introduced in the previous sections had particular domains of knowledge, from arguments to science to physical knowledge. As the fourth and final task, a general domain task is convenient, maximizing the domains covered in this study.

CommonsenseQA (Talmor et al., 2019) (CSQA) is a multi-choice question answering dataset that targets different types of commonsense knowledge, across several domains. To generate the dataset, the authors resort to ConceptNet (Liu and Singh, 2004), a large knowledge base, that contains commonsense knowledge. ConceptNet is organized as a graph, which can be decomposed in triples (c_1, r, c_2), where c_1 and c_2 are concepts and r is the relation between them. Figure 3.6 provides a general pipeline of the generation process.

Each crowdworker receives a question set composed of three triples with the same source concept (blue subgraph in Figure 3.6). For each triple, the crowdworker must formulate a question containing the

Train	Dev	Test	Total
16113	1838	3084	21035

Table 3.3: Number of examples in each dataset partition.

source concept, such that the answer is the target concept, e.g. for `(river, atLocation, waterfall)` a possible question is *"Where on a river can you hold a cup upright to catch water on a sunny day?"*. The target concepts of the other triples are included as candidate answers.

The crowdworker chooses the additional two distractors following two rules. The first distractor is chosen by the crowdworker from a set of target concepts with the same relation to the source concept (red subgraph in Figure 3.6). The worker formulates the other distractor with the restraint that it must be a plausible answer. The authors train workers to classify each example as "unanswerable" or selecting the correct answer to verify question quality. Three examples from the resulting dataset are shown in Figure 3.5.

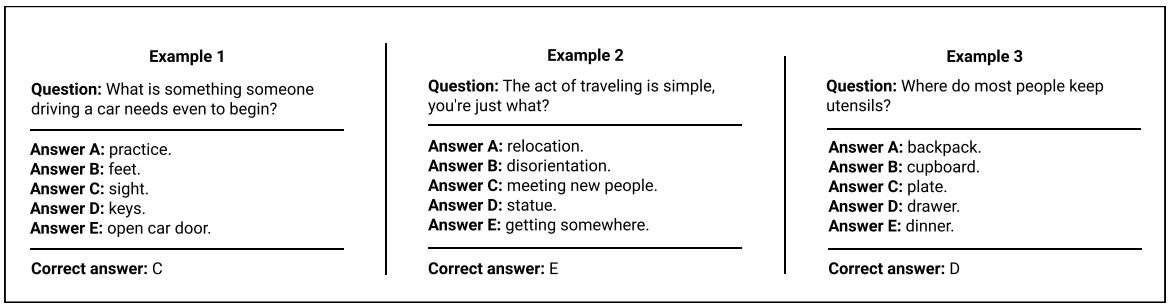


Figure 3.5: Three examples from the CSQA dataset.

A sample of 100 examples from the dataset was analyzed, verifying that they require a vast amount of knowledge types, as defined by the authors: spatial, cause & effect, has parts, is member of, purpose, social, activity, definition, and preconditions.

The dataset size is described in Table 3.4.

Train	Dev	Test	Total
9741	1221	1140	12102

Table 3.4: Number of examples in each dataset partition.

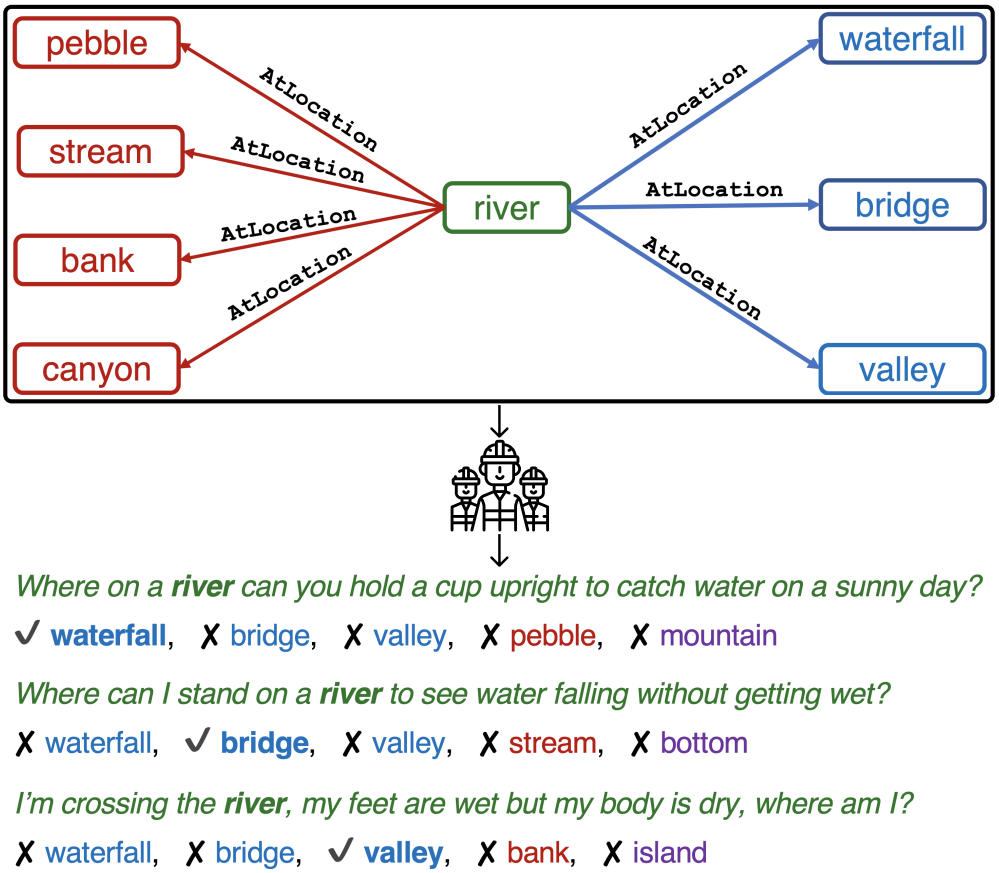


Figure 3.6: Adapted from (Talmor et al., 2019). Generation process for CommonsenseQA. Crowdworkers receive a subgraph from ConceptNet, and must produce questions using the concepts and the relations between them.

3.5 Summary

In this chapter, the commonsense reasoning tasks proposed for this dissertation were presented.

Argument Reasoning Comprehension Task (ARCT) tests the argument reasoning ability of a model, requiring not only language and logic skills but also commonsense knowledge. An argument is composed by three major segments: the claim, which is the assertion trying to be made; the premises / reasons, which are facts or evidences which support the claim; and the warrant, which is an implicit premise that allows the claim to logically follow from the premises. The warrant is left implicit, on the supposition that it is knowledge shared between the addresser and the addressee easily recovered by the latter. In ARCT, the model is given a reason, a claim and two possible warrants, having to pick the warrant which supports the connection between the reason and claim.

The second task is AI2 Reasoning Challenge (ARC), which is a multiple-choice question answering

task focused on natural sciences. It was made from a collection of questions from 3rd to 9th-grade science exams. It requires different types of reasoning and commonsense knowledge. The model is provided with a question and up to five candidate answers, and it must choose the correct answer to the question.

The third task, Physical Interaction Question Answering (PIQA), tests the capabilities of models to answer commonsense questions regarding the physical world. Models are presented with a goal, mostly an everyday situation that a human might want to accomplish, and two possible solutions to attain the goal. Models will need to learn, from raw text only, physical commonsense knowledge.

The final task is CommonsenseQA (CSQA), a multiple-choice question answering task, whose questions belong to a wide array of knowledge types. CSQA covers many different types of scenarios which one might encounter in daily life. It was built resorting to a knowledge base named ConceptNet.

Chapter 4

Implementation

This chapter addresses the implementation of the models (Section 4.1) along with their training methodology (Section 4.3), the adversarial attack (Section 4.2), data contamination studies (Section 4.4) and shortcut exploration (Section 4.5).

The implementation of the models and some of the data used for the experiments can be found in the following repository: `https://github.com/nlx-group/study-of-commonsense-reasoning`.

Section 2.3.2, Section 2.6 and Section 2.7 provided a brief introduction and background of the models and methods that are described in this chapter.

4.1 Models

To study the commonsense reasoning capabilities of NLP models, models are grouped according to similar characteristics that reflect different paradigms in neural NLP.

One group is the Encoder-only Transformers, of which the chosen instance is RoBERTa (Liu et al., 2019b) (Section 4.1.1). Encoder-only Transformers were first introduced with BERT (Devlin et al., 2019), starting a research line on classification models.

Another group comprises the Decoder-only Transformers. The GPT series of models have gained notoriety, and I selected the most recent *computationally affordable* version, GPT-2 (Radford et al., 2019) (Section 4.1.2).

Yet another group is the Encoder-Decoder Transformers. T5 (Raffel et al., 2020) (Section 4.1.3) is chosen as the representative due to the promising capabilities displayed through its excellent scores in the GLUE benchmark. By including this type of architecture, the objective is to determine whether (i) having both the encoder and decoder helps; and (ii) the text-to-text framework helps with reasoning.

The final group is the Neuro-Symbolic Transformers. COMET (Bosselut et al., 2019; Hwang et al., 2020) (Section 4.1.4) is the selected candidate due to the simplicity of the training objective to inject the knowledge from a Knowledge Base (KB) into the model. The goal is to determine whether the quality

of data coming from a KB can boost the reasoning abilities through finer priors.

All models were implemented using PyTorch (Paszke et al., 2019) and Huggingface's Transformers library (Wolf et al., 2020a). PyTorch allows to perform tensor computation using GPU's, and features an automatic differentiation system. Huggingface's Transformers is built on top of PyTorch. Its goal is to provide implementations of state-of-the-art Transformer architectures for NLP. It offers the possibility of re-using pre-trained weights, remove the need to pre-train the language models, which is computationally expensive.

4.1.1 RoBERTa

RoBERTa (Liu et al., 2019b) is a derivative of BERT (Devlin et al., 2019), resulting from the optimization of BERT models. The authors sought to optimize BERT by building a profile of the impact of each model attribute and pre-training task, akin to an ablation test. As a consequence, the authors propose a series of changes to the original BERT architecture, the resulting model being named RoBERTa. This is a summary of the changes,

- **Masking.** For the MLM task, BERT uses a static mask, meaning the words to be masked in each sentence are selected in pre-processing and do not change over the training epochs. RoBERTa uses a dynamic mask, where for each sentence, which can be seen many times during training, a new mask is calculated for each appearance.

- **Pre-training objective.** BERT defines its pre-training objective as the combined loss of MLM and NSP tasks. It was found with RoBERTa that NSP does not improve downstream task performance, opting to remove it from the pre-training objective, maintaining MLM only.

- **Large batch training.** BERT was trained for 1M steps and a batch size of 256. More recent literature (You et al., 2020) suggests trading the number of steps for large batch sizes, resulting in faster training and better performance. RoBERTa implements an 8K batch size and experiments with 100K, 300K and 500K training steps, with the latter (500K) creating a better model.

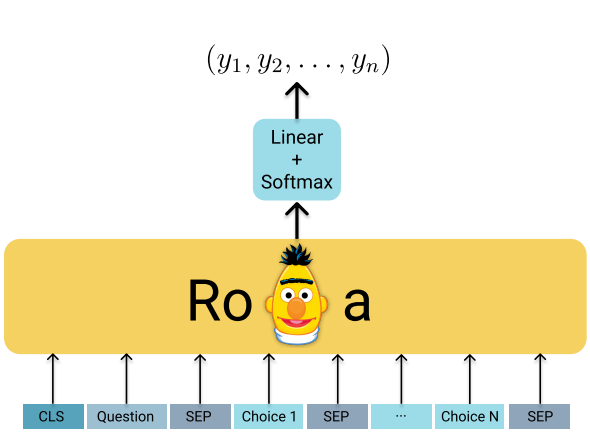
- **Text Encoding.** Byte-Pair Encoding (Sennrich et al., 2016) is the subword tokenization algorithm used by BERT. In BERT, the vocabulary size was set to 30K. In RoBERTa, the vocabulary size was increased to 50K. An additional change was the manner in which input embeddings are composed. In BERT, input embeddings are the sum of the token embedding, positional embedding and segment embedding. In RoBERTa, a simplification is made and segment embeddings are removed.

One of the authors' assumptions regarding the pre-training phase was that BERT was under-trained, so they compiled a new training set that includes more data for the pre-training phase. The data was compiled from six different sources, the BookCorpus (Zhu et al., 2015) & English Wikipedia (original BERT

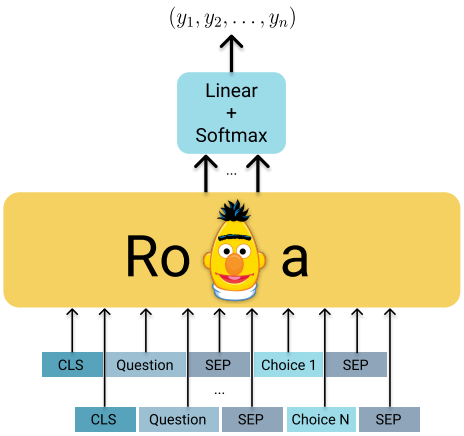
training set), CC-News (Nagel, 2016), OpenWebText (Gokaslan and Cohen, 2019) and Stories (Trinh and Le, 2018). Overall, RoBERTa is trained with 160GB of raw text, while BERT was trained with 16GB.

Regarding the fine-tuning stage, RoBERTa behaves just like BERT. In a text classification task, the input is a single sequence of tokens, starting with a `CLS` token. Each segment of the input is separated by a `SEP` token. This is exemplified in Figure 4.1a. The hidden state (representation) corresponding to the `CLS` token in the final layer is passed on to a classification head (Figure 4.2) to make a prediction.

During fine-tuning, the authors deploy a linear learning rate scheduler with warm-up, meaning the learning rate increases from 0 to the learning rate set by the user, over the course of a number of pre-determined steps, called warm-up steps. After reaching the learning rate value set by the user, it linearly decreases to 0. This scheduler is used in this dissertation during RoBERTa's training.



(a) General input example for classification tasks. The token embeddings are the input to the network, and a final hidden state is calculated after N Encoder blocks. The final hidden state corresponding to the `CLS` token is handed to a classification layer which will produce a prediction.



(b) Multiple-Choice RoBERTa scheme. Input is sectioned into N inputs, one for each candidate answer. The inputs are passed separately, and the classification head outputs one value per input, which is shaped into a probability distribution vector. The maximum value corresponds to the most relevant candidate answer, ultimately being the prediction of the network.

Figure 4.1: Two types of fine-tuning schemes for RoBERTa.

The authors additionally made task-specific modification for NLI tasks, employing a method previously mentioned at the end of Section 2.4. For tasks that require the model to choose between several candidate answers, given a premise (e.g. multiple-choice Q&A), providing the whole input simultaneously (as pictured in Figure 4.1a) can introduce errors and bias, as each candidate answer can attend to other answers. By sectioning the input into pairs of `<question, answer>`, the model can use the linguistic knowledge it possesses to choose the pair that it deems more "natural", or more correct.

The modification frames problems under a pairwise ranking scheme. For example, let us imagine a multiple-choice task comprised of a question and multiple (N) possible answers. Under this scheme,

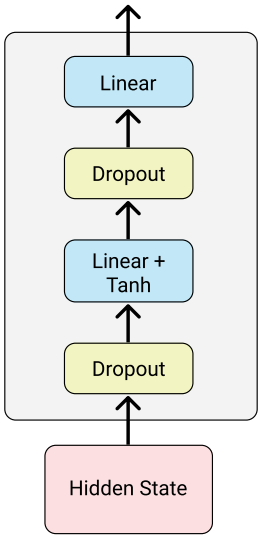


Figure 4.2: Layout of a classification head. In-between linear projections, dropout (Srivastava et al., 2014) is employed. The final linear layer projects to an output space, whose dimension equals the number of output classes. It is common to perform a softmax after.

the input is sectioned into N inputs, each containing the question and a candidate answer. Each input is passed through the model, from which the hidden state corresponding to the `CLS` token is used for prediction. The classification layer contains just one output neuron, which calculates the relevance of the `<question, answer>` pair. The pair with the maximum relevance score is chosen as the predicted answer. This is shown in Figure 4.1b.

In this dissertation, the model produced from this modification is called "Multi-Choice RoBERTa". Multi-Choice RoBERTa is implemented and made available from Huggingface.[1] The implementation of this kind of model is described in Section 4.1.4.

All the proposed tasks (Section 3) for this dissertation can be converted into a multiple-choice format. Three tasks are compatible with a standard Multi-Choice RoBERTa implementation. ARCT and PIQA are binary classification tasks, thus have two candidate answers. CSQA has five categories, which translates to five candidate answers. ARC's Challenge Set differs slightly from the previous, as it does not have a fixed number of candidate answers. Most of the examples have four candidate answers; however, a small number of examples have five, which poses a problem due to batch training.

In order to have a shorter and more stable training, computations are usually done in batch. This can be a challenge when learning ARC, as batches can contain examples with different numbers of candidate answers, meaning tensors would have to vary in shape, which is not possible. There are two options to address this issue. One option is to produce batches with a fixed length of candidates; the other is to introduce padding candidates. I have chosen the latter to preserve pseudo-random batches as much

[1] https://huggingface.co/transformers/model_doc/roberta.html#robertaformultiplechoice

as possible. The padding candidates have their relevance score set to zero before computing the loss to avoid them being chosen as the network's prediction.

The input to the network then becomes a 3D Tensor, $N \times M \times L$, where N is the batch size, M is the number of candidate answers, L is the sequence length, in subwords.

4.1.2 GPT-2

GPT-2 (Radford et al., 2019) is the second version of the GPT model series, developed by OpenAI.[2] It differs from the other models used in this dissertation due to its strictly left-to-right generative nature, as opposed to architectures that use the Transformer encoder, which performs bidirectional attention.

GPT-2 leverages the Transformer decoder (on the left in Figure 2.1) to produce a general language model, that can be applied to most NLP tasks with little adjustments.

Similarly to RoBERTa (Section 4.1.1), GPT-2 is first pre-trained with large corpora on a language modelling task to learn the intricacies of the human language. Unlike RoBERTa, however, the language modeling task is performed in a more canonical formulation, in a left-to-right manner: given a context $(w_0, w_1, \ldots, w_{-1})$, predict the next word w_i. The authors constructed their pre-training corpus, named WebText, by scraping all outbound links from Reddit with at least three karma score (three "upvotes"). After de-duplication and heuristic-based cleaning, the final dataset contains just over 8 million documents and totals 40GB of text.

The fine-tuning procedure is also similar to RoBERTa's. Stacked Transformer decoders encode the sequence, and the hidden states are passed on to a classification head. The classification token (`CLS`) is placed at the end of the sequence, as opposed to the beginning (as in RoBERTa), in order to encode the meaning of the entire sequence in a single representation. As the architecture is inherently left-to-right, only the last token can access and encode information of the whole sequence.

The resulting model, GPT-2, was applied to several tasks to evaluate its performance. When applied in a zero-shot manner (without fine-tuning), it could still achieve great results, surpassing the state of the art in some tasks. This shows that unsupervised pre-training provides useful generalizations for other tasks.

In GPT-2's paper, the problem of text memorization from the pre-training phase was explored. An experiment was devised to verify whether the results could be due to memorization. They create bloom filters[3] from WebText and calculate an upper bound for text collisions between the downstream tasks datasets and WebText. The existence of text collision between the pre-training dataset (WebText) and fine-tuning datasets (downstream tasks) is called data contamination. It was found that due to text overlap, the model gains small but consistent benefits, arguably due to memorization. This problem motivates further experimentation for this dissertation (Section 4.4).

[2] `https://openai.com/`

[3] Probabilistic data structure that can efficiently definitely determine whether an element is not in a set, but cannot guarantee that it is in the set, only that it may be in it.

For this dissertation, a list of changes made to the GPT-2 implementation provided by HuggingFace is indicated below:

- The pre-training phase does not use a padding (`PAD`) token and classification (`[CLS]`) token. Embedding layer is resized to include these two tokens, so they can be used and trained.
- Pairwise ranking scheme is used to train GPT-2. The available architecture is adapted so as to follow this methodology, by passing each `(question, answer)` pair separately through a classification head.

A cosine learning rate scheduler is used during the fine-tuning process, as described in the original GPT paper (Radford et al., 2018). The learning rate increases linearly to the learning rate value set by the user in N warm-up steps, and decays to 0 following a cosine wave.

4.1.3 T5

T5 is a text-to-text framework leveraging a standard Transformer architecture. The input to the network should be kept as natural as possible, and unlike RoBERTa, it requires no special tokens (e.g. the `CLS` token). To help the network understand what task it is performing, and to identify the different segments of the input, textual prefixes are added to the input. The task is prepended to the input, and a prefix is prepended to each segment of the input, as can be seen in Figure 4.3. The output of T5 is textual, even for regression tasks.

This flexibility, due to both the input and output being textual, facilitates its usability, as T5 can accommodate different tasks with no additional changes to the network itself. In fact, in its introductory work, T5 performs multi-task learning of all GLUE (Wang et al., 2018) (General Language Understanding Evaluation) tasks, at the same time, obtaining state-of-the-art results in each.

T5 is pre-trained with a denoising task called span prediction. Instead of masking single tokens, contiguous spans of tokens are masked, and the model must predict the tokens that were masked, restoring the original input. It is shown in its paper that models pre-trained with span prediction perform better on downstream tasks than models pre-trained with single token prediction. The model was pre-trained with the "Colossal Clean Crawled Corpus" (C4), which after undergoing filtering (for quality reasons) totals around 745GB of raw text.

In order to generate text, T5 has a Transformer decoder stack available, unlike BERT and RoBERTa. The decoder block (Figure 2.1) comprises a Masked Multi-Head Attention layer, an Encoder-Decoder Multi-Head Attention layer, and a feed-forward layer for the output. The Masked Multi-Head Attention block can only access information from the past at each timestep, hence why it receives a partial (masked) input at each timestep. The Multi-Head Attention layer that follows it is slightly different from the one present in the Encoder. Here, attention will be calculated using the previous layer's output (in the decoder) and the Encoder's output, which contains valuable information from the input.

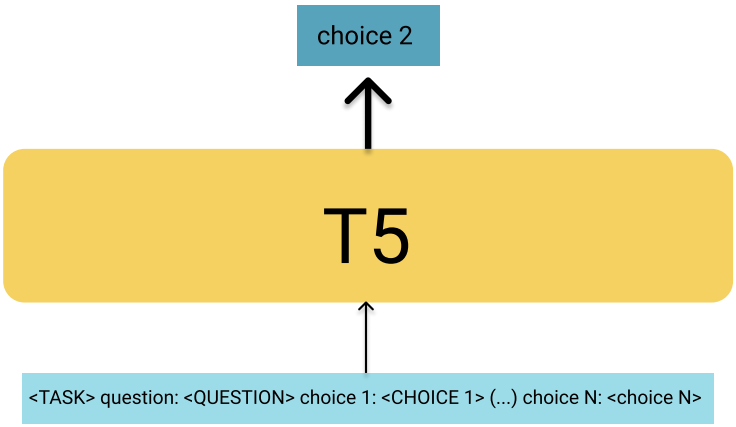


Figure 4.3: The first part in the example is the name of the task (e.g. arct). For each section of the input, a textual description is prepended (e.g. question: (...)). The output must also be generated (text), which in this example, is choice 2.

During training, the partial outputs (masked outputs) are fed to the decoder all at once for performance reasons. To begin generation during inference time, the decoder is *primed* with a begin-of-sentence token and performs generation in an auto-regressive manner until a maximum number of tokens is reached, or the decoder itself emits an end-of-sentence token.

Since the model's output is generated, hallucination might occur. Hallucination happens when the model generates a label that does not belong to the set of labels for the task. Despite the authors not having observed this behavior in their experiments, it can still happen when using T5, and naturally, hallucinated responses would be marked as incorrect answers.

A linear learning rate scheduler is used during the training of T5 models. Due to T5's flexibility, the architecture needs no changes across tasks, requiring only an input adjusted to the task. Examples of the formatting followed for each task are found below (Figure 4.4, Figure 4.5, Figure 4.6 and Figure 4.7).

Input →	`arct claim: Google is not a harmful monopoly reason: People can choose not to use Google. warrant0: all other search engines re-direct to google warrant1: other search engines do not re-direct to google`
Expected Output →	`True`

Figure 4.4: Formatted example for the ARCT task input to a T5 model. The expected value in the form of binary value (0 or 1) is converted to True or False.

Input →	arc question: Air has no color and cannot be seen, yet it takes up space. What could be done to show that air takes up space? A: observe clouds forming B: measure the air temperature C: other search engines do not re-direct to google D: weigh a glass before and after it is filled with water
Expected Output →	C

Figure 4.5: Formatted example for the ARC Task input to a T5 model.

Input →	piqa goal: What can i use to help filter water when I am camping. sol0: You can use a water filtration system like a brita pitcher. sol1: Coffee filters are a cheap and effective method to filter water when outdoors.
Expected Output →	True

Figure 4.6: Formatted example for the PIQA task input to a T5 model.

Input →	csqa question: What is something someone driving a car needs even to begin? A: practice B: feet C: sight D: keys E: open car door
Expected Output →	C

Figure 4.7: Formatted example for the CSQA task input to a T5 model.

4.1.4 COMET(BART)

COMET, short for Commonsense Transformers, is a method that aims to enrich a pre-trained transformer with commonsense knowledge coming from knowledge bases. To incorporate the knowledge, COMET resorts to a fine-tune task named tail-prediction. Given a triple, `(head, relation, tail)`, the model is presented with the head and relation, and it must (learn to) generate the tail (Figure 4.8).

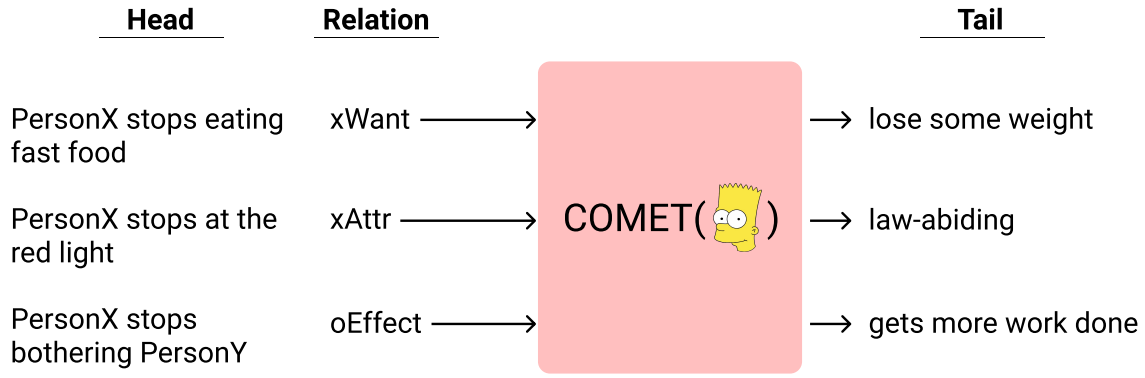


Figure 4.8: Example of the tail-prediction task. PersonX and PersonY corresponds to two different persons. The example features different types of relations, and their definitions are the following. xWant: as a result, PersonX wants; xAttr: X is seen as; oEffect: as a result, PersonY or others will.

The authors fine-tune Transformers on this task for one epoch and observe that the models can in fact learn the commonsense knowledge from the knowledge base and, additionally, generalize and create new, unseen knowledge.

Knowledge bases that can be represented as triples can be leveraged by COMET models. In the paper, the authors apply GPT-2 (Radford et al., 2019) and BART (Lewis et al., 2020) to learn the tail-generation task, using ATOMIC2020 (Hwang et al., 2020; Sap et al., 2019) triples as a source of knowledge. ATOMIC (Sap et al., 2019) is a Commonsense Knowledge Base that encodes commonsense knowledge regarding the physical and social aspects of the human experience. Knowledge is grouped into three categories: **Physical-Entity**, which encodes knowledge about entities and objects, e.g. What can be done using eggs; **Social-Interaction**, concerning the reaction, intentions, purposes and other aspects of socially-triggered events; **Event-Centered**, which gives an intuition as to how common events related to each other. Its second version, ATOMIC2020 (Hwang et al., 2020), was expanded with more knowledge using the same approach as used in version one, and also through the integration of knowledge coming from ConceptNet (Liu and Singh, 2004). Figure 4.9 provides examples of each type of knowledge encoded in ATOMIC2020.

In this dissertation, I will use the BART variant of COMET, named COMET(BART). BART is an Encoder-Decoder Transformer model with some slight modifications, such as changing the activation function from ReLU to GeLU (Hendrycks and Gimpel, 2016)). BART's pre-training phase comprises

	Head	Relation	Tail	Size
PHYSICAL-ENTITY	bread	ObjectUse	make french toast	165,590
		AtLocation	basket; pantry	20,221
		MadeUpOf	dough; wheat	3,345
		HasProperty	cooked; nice to eat	5,617
	baker	CapableOf	coat cake with icing	7,968
		Desires	quality ingredients	2,737
		Not Desires	bad yeast	2,838
EVENT-CENTERED	X runs out of steam	IsAfter	X exercises in the gym	22,453
		HasSubEvent	become tired	12,845
		IsBefore	X hits the showers	23,208
		HinderedBy	drinks too much coffee	106,658
		Causes	takes a break	376
		xReason	did not eat breakfast	334
	X watches ___ anyway	isFilledBy	the game; the TV	33,266
SOCIAL-INTERACTION	X runs out of steam	xNeed	do something tiring	128,955
		xAttr	old; lazy; lethargic	148,194
		xEffect	drinks some water	115,124
		xReact	tired	81,397
		xWant	to get some energy	135,360
	X votes for Y	xIntent	to give support	72,677
		oEffect	receives praise	80,166
		oReact	grateful; confident	67,236
		oWant	thank X; celebrate	94,548

Figure 4.9: Reproduced from (Hwang et al., 2020). Examples for each relation type and their respective size in ATOMIC2020.

several denoising tasks, combining the loss of each one. It follows a list of the pre-training tasks:

- **Token Masking.** The same task as in BERT's pre-training phase. Single random tokens are masked, and the model must predict the original input.
- **Token Deletion.** Instead of masking, delete the token. The model must identify where the missing token is and predict it.
- **Text Infilling.** Similar to the span prediction task, mask a span of tokens (or insert a mask that should not be there). The model must generate the original text.
- **Sentence Permutation.** Permute the sentences in the input text. The model must generate the original sequence.

- **Document Rotation.** Random token is selected, and input is rotated so that it starts with the selected token. The model must generate the original text.

The pre-training dataset is the same as RoBERTa's, comprising around 160Gb of raw text.

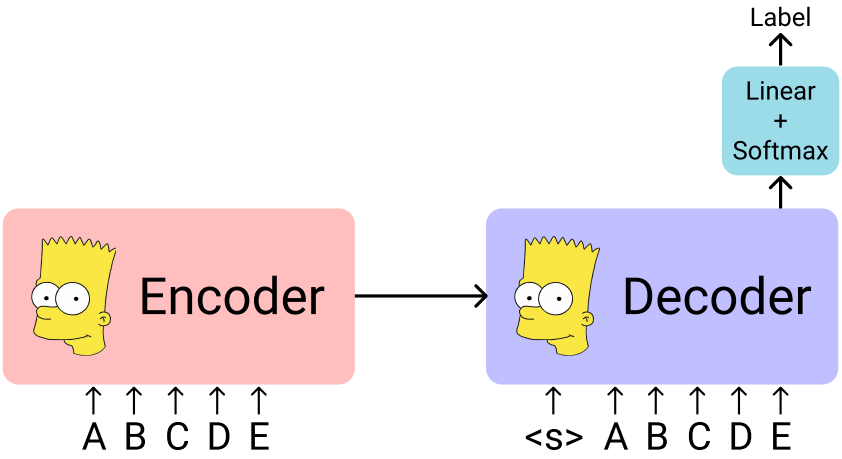


Figure 4.10: An example of how to use BART for classification problems.

To fine-tune a trained COMET model on an additional task, we follow the best practices for BART, as described in the respective paper. For multiple-choice tasks, the methodology used is the same as described in Multi-Choice RoBERTa's implementation (Section 4.1.1). The major difference lies in the fact that BART has a decoder stack, so instead of the Encoder's final state being fed into a classification head, it is fed into the Encoder-Decoder Multi-Head Attention instead. On the decoder side, we introduce the same sequence that we introduced to the Encoder. After the decoder has processed the whole sequence, the hidden states corresponding to the last token is given to a classification head. This process is depicted in Figure 4.10.

The pre-trained BART(COMET) model used in the original paper was released by the authors.[4] In this dissertation, it is used to initialize a BART model before fine-tuning on the proposed tasks. COMET(BART)'s fine-tuning phase is identical to Multi-Choice RoBERTa's (Section 4.1.1): the input is the 3D tensor, where each example is decomposed into multiple inputs, one for each candidate answer.

Unlike what happened for RoBERTa, Huggingface does not feature an implementation of a multi-choice BART model, so it was implemented by me. The implementation required only two minor changes (i) in the input shape; and (ii) classification head. The multiple stages of the pipeline are depicted in Figure 4.11. It follows a description of the series of operations that comprises the implementation.

Input Reshape. The required input tensor is the same as described in Section 4.1.1, a 3D Tensor, where the first axis contains each batch example, the second axis contains each `(question, answer)`

[4] https://github.com/allenai/comet-atomic-2020#model-comet-atomic-2020

pair, where each pair is a matrix containing subword embeddings. The size of the second axis tells us the number of candidate answers.

We will follow along the stages in Figure 4.11. The first operation is a reshape of the input tensor, going from a 3D tensor to a 2D tensor. Here, the `(question, answer)` pairs are all arranged sequentially. This is necessary to perform a batch forward pass with all the pairs from each example at the same time, saving compute time.

Feeding BART(COMET). The second operation is feeding the pairs to the COMET(BART) model, which represents each token in the sequences as continuous vectors, of which we keep the last token of each pair, denoted here as hidden states $h_{(q_i,c_i)}$.

Classification. In the third operation, we pass the hidden states through a classification head that is slightly different from the standard (Figure 4.2). Dropout is applied to each input hidden state. The hidden states are then passed through a linear layer with only one neuron, producing one value per pair, which is the relevancy score attributed to the pair. This produces a vector of relevancy scores ($r_{(q_i,c_i)}$), one for each pair.

Output Reshape. In the fourth and final operation, we reshape the vector by grouping every two scores. We group with a step size of two because this is an example for a binary task, where the model has to choose from two choices. This produces a two by two matrix (two examples, two candidates). Given the matrix of scores, the prediction of the network is the argmax[5] of each vector.

BART's original implementation[6] uses a polynomial learning rate scheduler, as such, in this dissertation, COMET(BART) is trained with the same scheduler.

4.2 Adversarial Attack

Adversarial attacks, in this dissertation, serve the purpose of hinting at possible brittle, surface-level comprehension, vulnerable to simple changes in the input that preserve the semantics of the example. They only provide hints, and not definite proof, because these are automated methods, and as such, not all perturbations will preserve semantics and grammaticality. A good portion of them will have these characteristics, however, which means it is still a useful approach.

Three adversarial attack algorithms are considered for this dissertation: BERTAttack (Li et al., 2020b), BAE (Garg and Ramakrishnan, 2020) and TextFooler (Jin et al., 2020). BERTAttack and BAE use a BERT model to select candidate word replacements, while TextFooler uses counter-fitted GLOVE Word Embeddings (Mrkšić et al., 2016; Pennington et al., 2014). These three algorithms are implemented in a Python package named TextAttack (Morris et al., 2020b), which is used in this dissertation.

TextAttack abstracts and automates the attacking process, along with the implementation of several attack algorithms. The user of the library is required to implement two classes: Dataset and Model.

[5] argmax is an operation that retrieves the index of the maximum value of all the elements in a vector.

[6] https://github.com/pytorch/fairseq/blob/master/examples/bart/README.md

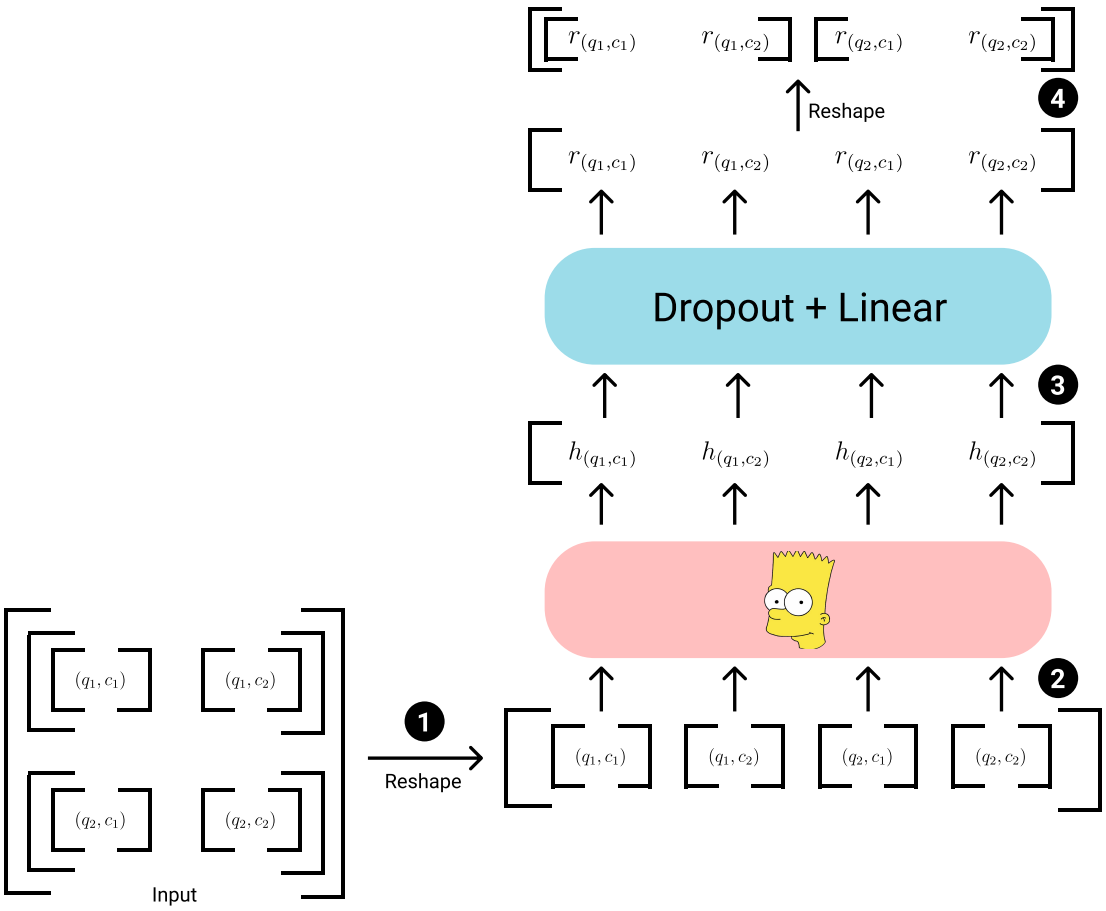


Figure 4.11: Conceptual implementation of a multi-choice BART model. In this example, the input contains two examples (batch size of two), each example having two candidate answers (binary classification).

The Dataset class is responsible for reading and producing a structured format of each example of the dataset. It creates a list of dictionaries, each dictionary containing an example, where the keys are the labels for each segment of the input and the values hold the respective text.

The Model class encapsulates the model under attack. The three algorithms (BERTAttack, BAE and TextFooler) assume the target is a black-box model, and in order to find important words in the input, they compare the difference in the logit value[7] for the correct class with and without the word in the input. The model class thus receives several examples, passes them through the model, and returns the logit values. The framework calculates the difference in values and picks the most important words. Starting with the most important word and successively progressing towards the least, replacement candidates for each word are determined until the prediction of the model is changed. Determining the replacement

[7]Logits are the non-normalized predictions made by the model.

candidates depends on the algorithm. BERTAttack and BAE use a BERT model to do so, while TextFooler uses word embeddings.

With the intention of learning about the quality of the adversarial examples produced by each algorithm, I performed preliminary experimentation with them. Upon inspection of small samples of each algorithm, despite BAE and BERTAttack using complex and powerful language models, they appeared to yield worse quality adversarial examples than TextFooler, which uses a simpler method with off-the-shelf word embeddings. BERTAttack and BAE produced ungrammatical and non-semantic preserving examples more often than TextFooler did. As a result, TextFooler is chosen as the algorithm to be used in the experimentation.

Although TextFooler fared better than the others, it was still somewhat inconsistent using the out-of-the-box implementation. An evaluation study (Morris et al., 2020a) carried out by the authors of TextAttack, found that even state-of-the-art adversarial attack algorithms, such as TextFooler, often do not preserve semantics, and 38% introduce grammatical errors. The authors of the paper make a suggestion regarding TextFooler's hyper-parameters, by finding more appropriate threshold values through human evaluation.

The hyper-parameters relate to filtering out candidate words and candidate adversarial examples. TextFooler selects candidate replacement words by calculating the target word's cosine similarity with every other word in the GLOVE's vocabulary. All the words above a certain threshold are selected as candidate replacements. This threshold is one of the two hyper-parameters. In TextFooler's original implementation this threshold was set to 0.5. The evaluation study mentioned above finds a 0.9 cosine similarity threshold more suitable. In this dissertation, 0.9 is used as the threshold.

The remaining hyper-parameter is the sentence similarity threshold. A generated adversarial example should be semantically equivalent to the original example. TextFooler addresses this issue by calculating the cosine similarity between the sentence embeddings of the original example and the adversarial examples, only selecting the adversarial examples above the sentence similarity threshold. Sentence representation is obtained using the Universal Sentence Encoder (Cer et al., 2018). In the original implementation of TextFooler, this threshold was set at ≈ 0.841. The evaluation study finds a suitable threshold of 0.98.

In this dissertation, after performing some preliminary tests on the proposed tasks (Chapter 3), 0.98 was far too strict and could barely produce any adversarial examples. A compromise was found at 0.9 threshold value, which produced a considerable number of adversarial examples, and from the inspection of a small sample, the examples had reasonable levels of semantic preservation and grammaticality.

4.3 Training Methodology

BERT and its derivative models are unstable across random runs (Devlin et al., 2019; Mosbach et al., 2021), meaning that with the same setup but using different seeds, the results can vary significantly. The currently accepted hypothesis set forth by a study (Mosbach et al., 2021) suggests that despite catastrophic

forgetting (Kirkpatrick et al., 2017; McCloskey and Cohen, 1989) and small datasets being correlated with the instability, the problem is inherently an optimization one. The authors notice that the gradients are several orders of magnitude smaller during failed runs than successful runs. It is thus advisable to follow the empirically determined fine-tune practices established in the models' respective papers.

In this dissertation, the hyper-parameters used are kept to their default, presented in the models' papers; however, a sequential hyper-parameter search is performed for two hyper-parameters: learning rate and batch size. First, learning rate is optimized by searching the model with the highest dev accuracy score, after fine-tuning for 10 epochs, from the following range of rates: { 1e-3, 1e-4, 1e-5, 2e-3, 2e-4, 2e-5, 3e-3, 3e-4, 3e-5 }. Using the learning rate determined in the previous step, an appropriate batch size is determined in the the same way, from the range: { 4, 8, 16, 32 }. The learning rate and batch size found for each task are described in Appendix A.

After the hyper-parameter search, models in each run are trained up to 30 epochs, with epoch-wise checkpointing. The checkpoint that yields the highest dev accuracy is chosen as the model to test with. In order to have a significant result, the reported results are the mean of five runs, each with different seeds: 42, 1128, 1143, 1385, 1415.

Datasets for NLP are usually partitioned into three parts: train, dev, and test split. A lifecycle of a model consists of training on the train data, optimizing hyper-parameters or checking for overfitting by intermittently testing on the dev split, and finally, after all that, testing on previously unseen data with the test split. NLP has seen a rise in task competitions, where participants have access to training data and dev data and can create models from the data. Instead of having access to the reference output of the test split, they only have access to the input of the test split. They use the models to make predictions on the testset inputs and send them to the competition organizer, that calculates some performance metric by comparing the predictions with the testset reference output, and the metric score is displayed on a leaderboard online.

Two of the proposed tasks, PIQA and CSQA, are active competitions, and as such, I do not have access to the testset reference. In order to have three splits of the data, I consider the dev set as testing data and report on that data in Section 5. From the training data, I produce two splits: 90% of the training data is kept as training data, and 10% is set aside as dev data in a random fashion. In order to keep a balance of classes in the train and dev splits, I use stratified splitting,[8] which guarantees that the random split performed in the data preserves the distribution of the classes in the original data.

To verify statistical significance between models, a Student t-test is performed with $\alpha = 0.05$.

All experimentation was done on a single NVIDIA Titan RTX 24Gb VRAM over the course of five months (with some stoppage time in between experiments).

[8] https://scikit-learn.org/stable/modules/cross_validation.html#stratified-shuffle-split

4.4 Data Contamination Study

Data contamination is becoming an extremely relevant phenomena in NLP, as the pre-train & fine-tune procedure becomes more widespread. As models become larger, they can benefit from more pre-training data. The web provides a colossal amount of text, and as such, researchers usually resort to scraping the web to collect more pre-training corpora from which to train models with. Projects like CommonCrawl[9] are perpetually scraping the web and making available large amounts of text from it, having collected petabytes of data since 2008.

A problem arises when researchers build datasets for tasks from the same webpages that have been used as pre-training data. Models can potentially memorize spans of text from those resources, and as such, the basic principle of testing on unseen (o.o.d) data is broken.

The authors of GPT-2 (Radford et al., 2019), as mentioned in Section 4.1.2, were concerned that their model could be benefitting from data contamination, by memorizing text from the pre-training corpus. They create bloom filters[10] from WebText (GPT-2's pre-training corpus), producing a set of 8-Grams present in the corpus. They then go through the tasks testsets and decompose it into sets of 8-Grams, from which they can calculate the overlap with WebText using the bloom filters. As bloom filters are probabilistic structures, this gives them an upper-bound on the overlap, with a calculated false positive rate upper-bound of $\frac{1}{10^8}$. It was found that due to text overlap, the model gains small but consistent benefits, arguably due to memorization.

In the next version, GPT-3 (Brown et al., 2020), the authors decided not to use bloom filters, due to the trade off of certainty for time not being deemed as worthy. In GPT-3's paper, they calculate direct overlap of N-Gram sets, to obtain deterministic, trustworthy overlap statistics. The methodology follows a three part algorithm:

1. **Determining the size of the N-Grams.** In the previous study, the authors use a static size of n-grams, $N = 8$. In this refined methodology, the size of the N-Grams changes depending on the downstream task testset. The authors define N as the fifth percentile of the distribution of the size (in tokens) of the examples in the testset. In order to avoid spurious collisions and time-intensive computations, they set minimum and maximum values for N, 8 and 13, respectively.

2. **Creating the set of N-Grams.** After determining the size of the N-Gram for the task testset, they create a set of unique N-Grams from the testset examples.

3. **Computing collisions.** Using the same size of N-Grams, iterate over the pre-training corpus, producing N-Grams and verifying if they are present in the set of N-Grams from the task testset. All

[9] https://commoncrawl.org/

[10] Probabilistic data structure that can efficiently definitely determine whether an element is not in a set, but cannot guarantee that it is in the set, only that it may be in it.

> N-Gram collisions are saved with counters. The authors speed this computation by parallelizing the process and using an Apache Spark cluster.[11]

Popular machine reading datasets such as QuAC (Choi et al., 2018), SQuAD 2 (Rajpurkar et al., 2018) or DROP (Dua et al., 2019) are flagged for >90% contamination. PIQA (Bisk et al., 2020b) is flagged with 29% contamination. However, removing the contaminated text only decreases the performance by 3%, regardless of model size. The authors see this as a sign that memorization may not be at play but rather statistical cues in the data, though they did not offer empirically based support to that conjecture.

To perform the study in this dissertation, I implemented this algorithm in Python. NLTK (Loper and Bird, 2002) was used to compute N-Grams from text. Step one and two can be implemented resorting to basic data structures in Python's standard library, such as the Set and Dictionary. Step three would require an Apache Cluster to parallelize. Instead, multiprocessing[12] is used to distribute the workload to different processes. The pre-training corpus is evenly split according to the number of workers defined. Each worker is responsible for calculating overlaps between their section of the corpus and the computed N-Grams from the task testset. After all workers have finished, the results are merged. The computation was done in a server with two Intel®Xeon®Gold 6152, with 44 cores & 88 threads available. The computation was thus carried out with 88 workers.

The initial algorithm implementation used to obtain the results discussed in Section 5.5 was later refined to devise an open source library, with the aim of allowing researchers to analyze textual overlaps between pre-training datasets and any given task testset.

Obtaining the pre-training RoBERTa's/BART's pre-training corpora proved to be an arduous task. As described in Section 4.1.1 and Section 4.1.4, RoBERTa and BART were trained on five datasets: BookCorpus (Zhu et al., 2015) & English Wikipedia, CC-News (Nagel, 2016), OpenWebText (Gokaslan and Cohen, 2019) and Stories (Trinh and Le, 2018). COMET(BART) was additionally trained with ATOMIC2020 (D. Hwang et al., 2021).

BookCorpus has recently stopped being openly distributed, and several initiatives to obtain the corpus by crawling the original source have been made. The Pile (Gao et al., 2020), a large language modelling dataset, made by merging several corpus together, has made available a re-crawled BookCorpus. In this dissertation, The Pile's version[13] of BookCorpus is used.

OpenWebText and English Wikipedia can be obtained using HuggingFace's Datasets (Wolf et al., 2020b) package. They can be programmatically downloaded and manipulated from Python. Information regarding both datasets can be read through their documentation webpages.[14]

Stories is a corpus built from CommonCrawl. Unfortunately, due to data loss, the original distribution

[11] https://spark.apache.org/

[12] https://docs.python.org/3/library/multiprocessing.html

[13] Download URL: https://the-eye.eu/public/AI/pile_preliminary_components/books1.tar.gz

[14] OpenWebText: https://huggingface.co/datasets/openwebtext; Wikipedia: https://huggingface.co/datasets/wikipedia#20200501en

website no longer exists. After contact with the author, he made openly available a backup of the dataset.[15]

ATOMIC2020 is a Commonsense Reasoning Knowledge Graph, distributed as a triple file, with a head, relation and tail. The authors made the data openly available.[16]

The last dataset, and also more difficult to obtain, is CC-News, a specific section of CommonCrawl that contains news from different websites and different languages. RoBERTa used a version of CC-News, with 63 million news articles created between September 2016 and February 2019. I used News-Please (Hamborg et al., 2017) to extract, parse and save the news articles as json files, by adapting an existing script from the news-please repository.[17] As CommonCrawl joins news together from the various sources and languages, all news articles from that specific time range had to be processed and filtered, such that (i) news are within the correct range of dates; (ii) news are in the English language. Download and filtering had an execution time of 28 days.

4.5 Shortcut Exploration

An exploration is carried out in the search of shortcuts/cues that might be guiding the models' learning. In this exploration, two experiments are implemented: class balance and ngram lexical cues.

The implementation for the first experiment, where class balance is examined, does not require an extensive description as it is done with basic arithmetic with Python. The relative frequency of occurrences for each candidate answer is calculated, for each split of the dataset, and from there one can observe whether the relative frequency is balanced, or not, over all the candidate answers.

For the second experiment, the metrics developed in (Niven and Kao, 2019) are adopted. These metrics aim to provide a quantitative measure of how beneficial ngrams present in the dataset are as cues, if used by the model. Three metrics are used: applicability (α_k), productivity (π_k) and coverage (ξ_k). For the experiments, unigrams and bigrams are considered.

The proposed tasks (Chapter 3) can be framed as a multiple choice problem, whereby the model must choose the correct answer from a set of candidate answers. As such, each dataset example has two major segments: the question (or in the case of ARCT, the reason and claim) and the candidate answers (in ARCT, the warrants). The calculation of the metrics take into account tokens present in the candidate answers.

The applicability α_k of a cue k (Equation 4.1) measures the number of examples where the cue k occurs in one candidate answer, but not in any of the others:

$$\alpha_k = \sum_{i=1}^{n} \mathbb{1}\left[\exists j, k \in \mathbb{T}_j^{(i)} \wedge k \notin \mathbb{T}_{\neg j}^{(i)}\right] \tag{4.1}$$

[15] https://drive.google.com/drive/u/1/folders/1yZzwaV8LO1hK8ChIm0sxazXF8BSIZ683

[16] https://allenai.org/data/atomic-2020

[17] https://github.com/fhamborg/news-please/blob/cc0be8e5e238a4743c04c304acb759970ab9ef17/newsplease/examples/commoncrawl.py

where $\mathbb{1}$ is the indicator function (yields 1 if the input is true, and 0 if not) and $\mathbb{T}_j^{(i)}$ is the set of tokens for candidate answer j in example i.

Applicability tells us the number of examples where a cue provides a direct signal to one of the candidate answers, but it does not inform us on whether that signal yields a correct answer. Productivity π_k of a cue k (Equation 4.2) provides us with a measure of the proportion of applicable examples where the cue predicts the correct answer,

$$\pi_k = \frac{\sum_{i=1}^{n} \mathbb{1}\left[\exists j, k \in \mathbb{T}_j^{(i)} \wedge k \notin \mathbb{T}_{\neg j}^{(i)} \wedge y_i = j\right]}{\alpha_k} \tag{4.2}$$

where $y_i = j$ means the correct answer for example i is answer j, which cue k belongs to. A cue provides a useful signal if its productivity is above random chance: $\pi_k > \frac{1}{m}$, where m is the number of candidate answers in each example.

The previous two metrics encompasses applicable examples only, which are examples where a cue k is present in one candidate answer and not the other. It is useful, to understand how broad cue k's presence in the dataset is, to define its coverage ξ_k,

$$\xi_k = \frac{\alpha_k}{n} \tag{4.3}$$

where n is the total number of examples.

Using the previously defined equations, the metrics were calculated in Python.

4.6 Summary

This chapter described the implementation of four models (RoBERTa, GPT-2, T5 and BART), the adversarial attack algorithm TextFooler, the data contamination analyses and the shortcut detection. It also describes the training methodology used for fine-tuning tasks.

All of the models were implemented using HuggingFace's Transformer library. For RoBERTa, GPT-2 and BART, a pairwise ranking fine-tuning method was used. This method is implemented for RoBERTa in the HuggingFace library; however, such is not the case for GPT-2 and BART. Therefore, I extended the library to implement those methods.

The data contamination study followed an algorithm described in the paper introducing the models GPT-2 and GPT-3. It was implemented in Python, and parallelized using its multiprocessing library. The code was later improved to produce a new open-source package that helps researchers check their pre-training and fine-tuning datasets for textual overlaps.

The pre-training datasets used to train RoBERTa and BART, which were this study's target, proved challenging to obtain, as their distribution either was not possible due to licensing issues or the repositories had expired.

Lastly, the implementation of the two experiments in the shortcut exploration was done in Python. The first experiment, which checks the class balance of each dataset, required only the calculation of relative frequencies of each candidate answer. The second experiment, which searches for lexical cues, was achieved by implementing metrics specifically designed in the literature for that purpose.

Chapter 5

Results

This chapter presents the results and findings from the different experiments performed. In Section 5.1, the experiments were conducted using regular language models, pre-trained with raw text, and fine-tuned on the tasks (Chapter 3). The same tasks were the target of fine-tune using a neuro-symbolic model, pre-trained on raw text and ATOMIC2020 commonsense knowledge base, in Section 5.2. Adversarial attacks are described in Section 5.3. Cross-Task Evaluation is discussed in Section 5.4. Data contamination study is presented in Section 5.5. Finally, the product of the exploration of shortcuts is presented in Section 5.6.

5.1 Neural Models

5.1.1 Evaluation on Commonsense Reasoning Tasks

In order to prepare the neural models to solve the selected tasks, I fine-tuned the models on those four different tasks, previously presented in Chapter 3.

Table 5.1 shows the accuracy for each task. As expected, all the models perform above random choice, showing that the tasks can be performed comfortably by the neural models.

While there is a clear gap between the performance of the models and humans, some tasks are notably more challenging than others, showing mixed results across tasks for each model.

On ARCT, a binary classification task testing argument reasoning, the best scoring model is RoBERTa-Large, with a gap of 0.09 accuracy from the human upper bound. CSQA, a 5-choice Q&A general domain commonsense reasoning task, has a gap of 0.156, similar to PIQA, where the gap is 0.16. PIQA differs from CSQA, being a binary classification task focused on physical commonsense knowledge. These two feature a more significant margin.

Somewhat surprisingly, RoBERTa fares better across tasks compared to T5: this is unexpected as T5

[1] https://www.tau-nlp.org/csqa-leaderboard
[2] https://yonatanbisk.com/piqa/

	ARCT	ARC	PIQA	CSQA	Params
Random	0.5	0.25	0.5	0.2	-
HUMAN	0.909	N/A	0.949	0.889	-
RoBERTa-Large	**0.815 ± 0.011***	0.411 ± 0.022	**0.789 ± 0.006***	**0.733 ± 0.006***	355M
GPT2-Medium	0.540 ± 0.071	0.318 ± 0.009	0.706 ± 0.005	0.551 ± 0.012	345M
T5-Large	0.743 ± 0.006	**0.440 ± 0.008***	0.772 ± 0.005	0.713 ± 0.007	770M
State of the Art	0.599	0.814	0.835	0.833	-

Table 5.1: Accuracy of models (rows) on the selected tasks (columns). Scores displayed are the mean of the accuracy scores for 5 runs. A bold figure indicates the best result in that task. Human benchmarks and state of the art (SOTA) for CSQA were taken from their public leaderboard;[1] for ARCT, human benchmark from (Habernal et al., 2018a) and SOTA from (Zhou et al., 2020); and for PIQA, human benchmark from (Bisk et al., 2020b) and SOTA from their public leaderboard.[2] '*' indicates results that are statistically significant with $\alpha = 0.05$.

consistently tops the charts in comprehension tasks (Raffel et al., 2020),[3] and in this case, has more than the double of parameters. RoBERTa performs better on three of the four tasks, ARCT, PIQA, and CSQA, emerging as the most capable reasoner.

GPT-2 stands out in a negative way, performing worse than RoBERTa and T5 in every task, with a considerable gap to them.

While recalling that our goal in this dissertation is to study the integrity of the tasks that the models are intended to be performing, rather than to seek to beat the state of the art concerning the performance scores for these tasks, it is worth noting though that a new state of the art is actually established for the revised ARCT dataset (0.815), surpassing the previous one (0.599) by a considerable amount. While the SemEval shared-task regarding ARCT gained traction, amassing a considerable amount of participants, and consequently papers reporting on their solutions to the task, the revised version (Niven and Kao, 2019) seems to not have gained equal traction. Upon inspection of its citations, only two papers report a solution using the revised dataset, which may help explain why the previous state of the art is so much lower than the result obtained in this dissertation.

In the remaining tasks, the models fall below the state of the art, while not by much in PIQA (0.789 against 0.835) and CSQA (0.733 against 0.833).

Two patterns seem to arise from the empirical results, which are worthwhile to mention.

Despite ARCT being harder for humans to solve than PIQA, machines are better at solving ARCT than PIQA. Both are binary problems, but PIQA is much more comfortable to humans than ARCT, as it probes for physical knowledge, more natural than analyzing arguments about controversial problems.

[3] https://super.gluebenchmark.com/leaderboard

Reciprocally, it seems natural that PIQA is more difficult to models, as they cannot easily capture the facts of the physical world.

Another pattern regards ARC and CSQA, both multiple-choice problems with up to five possible answers. ARC's accuracy score is almost half of CSQA's, which cannot be discarded based on having more margin for error due to more possible answers. While CSQA has a wider array of commonsense dimensions than ARC, the latter was obtained from science exams, thus probing a more profound knowledge about the physical world, like physics and chemistry laws. It is not inconceivable then that ARC is a more arduous task for machines to solve.

As RoBERTa emerges as the most capable reasoner, the experiments that follow will be carried out resorting to RoBERTa.

5.1.2 Retraining and Evaluation with Partial Input

For human to perform the tasks at stake, every segment of the input is necessary, despite not all of them being equally important. A task that can be solved by models by just looking at the answers and not knowing the respective question it is a clear indication that the models are taking some shortcut.

Table 5.2 contains the results of the partial input training. For each task, the hyper-parameters (including random seed) that produced the best performing RoBERTa-Large model are used to train models that are fed with partial inputs.

Task	Full Inputs	Random Score	Full Input Score	Partial Input	Partial Input Score
ARCT	Claim (C) + Reason (R) Warrant 0 & 1 (W)	0.5	0.831	C+R	0.5
				R+W	0.5
				C+W	**0.785**
ARC	Question (Q) + Candidate Answers (A)	0.25	0.435	Q	0.227
				A	0.245
PIQA	Goal (G) + Solution 1 & 2 (Sol)	0.5	0.795	Goal	0.495
				Sol	**0.735**
CSQA	Question (Q) + Candidate Answers (A)	0.2	0.738	Q	0.196
				A	**0.218**

Table 5.2: Partial input training results (accuracy). Scores above random choice are in bold.

Accuracy above random baseline are in bold. For both ARCT and PIQA, an accuracy score obtained

with partial inputs, close to the score when using full inputs, is achieved. Despite CSQA also having a score above random baseline when only using answers, it is only 0.018 above it, which is not enough to say there are cues present that the models are benefiting from.

ARCT is a "repeated offender" in this setting. In previous work that flagged severe problems in the dataset (Niven and Kao, 2019), the authors noticed that providing just one or two segments of the input is enough for the model to perform above the random baseline, as it is relying on cues to solve the task, breaking it completely. After balancing out the cues, this was not possible anymore, at least when using BERT. We can see that such is still possible when using RoBERTa, as providing only the claim (C) and the warrant (W) is enough for the model to obtain an accuracy score of 0.785, just 0.046 shy of the score using the whole input (0.831). It seems that RoBERTa is better at picking up these cue signals and is taking advantage of them to perform the task, a strong indicator of surface learning.

PIQA shows the same problem as ARCT. Providing just the solution and leaving out the goal yields an accuracy score of 0.735, 0.06 shy of the score when using the full inputs (0.795). The model is providing solutions for a problem/goal it was not provided with, yet it is able to perform way above the random baseline. Without a goal, both solutions should be equally likely, unless one of the solutions seems so ridiculous that it is ruled out just off of commonsense, which would mean that the model is performing another type of commonsense reasoning task. Given the history of cue abuse by these models, this scenario seems unlikely, although not impossible.

Both tasks have gone through pre-processing steps to eliminate statistical lexical cues. It is not impossible that some lexical cues remain, but the cues the models are using are likely highly non-linear, making them likely difficult to detect. I will be back to this in more detail below in Section 5.6.

The remaining tasks, ARC and CSQA, seem to be more resistant to shortcut learning by RoBERTa. Providing just the question or just the answer leaves the models confused, as it should, which is reflected in scores in the vicinity of the random score. One of the most apparent differences between these two tasks and the others is that they offer more candidate answers (4-5) than the others (binary). It could be a hint that Q&A tasks with multiple choices provide better generalization power.

To understand if this is the reason, the tasks were converted to binary tasks by picking one random wrong answer and the correct answer. The models could not still obtain a score above the random baseline.

5.2 Neuro-Symbolic Models

5.2.1 Evaluation on Commonsense Reasoning Tasks

Table 5.3 shows the performance scores of COMET(BART) and BART-Large, the latter being a baseline to which COMET(BART) is compared.

[4] https://www.tau-nlp.org/csqa-leaderboard
[5] https://yonatanbisk.com/piqa/

	ARCT	ARC	PIQA	CSQA	Params
Random	0.5	0.25	0.5	0.2	-
HUMAN	0.909	N/A	0.949	0.889	-
BART-Large	0.655 ± 0.154	0.382 ± 0.027	0.777 ± 0.005	$\mathbf{0.738 \pm 0.005}$*	406M
COMET(BART)	$\mathbf{0.790 \pm 0.005}$	$\mathbf{0.412 \pm 0.011}$	$\mathbf{0.783 \pm 0.008}$	0.718 ± 0.008	406M
State of the Art	0.599	0.814	0.835	0.833	-

Table 5.3: Accuracy of models (rows) on the selected tasks (columns). Scores displayed are the mean from the scores obtained for 5 runs. A bold figure indicates the best result in the task. Human benchmarks and state of the art (SOTA) for CSQA were taken from their public leaderboard;[4] for ARCT, human benchmark from (Habernal et al., 2018a) and SOTA from (Zhou et al., 2020); and for PIQA, human benchmark from (Bisk et al., 2020b) and SOTA from their public leaderboard.[5] '*' indicates results that are statistically significant with $\alpha = 0.05$.

COMET(BART) outperforms BART-Large on all but one task, CSQA, broad-ranging five-choice Q&A task. In CSQA, BART-Large performs even better than the previously best performing model in this dissertation, RoBERTa (Section 5.1.1). Despite the evident differences in the means, the t-test yielded a p-value of > 0.05 for all tasks but CSQA, meaning the differences are not statistically significant in other tasks.

This could be due to the small sample size (5 runs), which may not be enough to detect a meaningful difference in this instance. BART-Large also has a larger standard deviation on ARC and ARCT, especially the latter, possibly due to instability across the random seeded runs.

If disregarding the statistical significance, the neuro-symbolic method does provide an advantage over the baseline (BART-Large), meaning the refined priors enhance the capabilities of the model to perform the commonsense reasoning tasks. However, when comparing with the best performing model from Section 5.1.1, RoBERTa, COMET still falls short.

In comparison with the state of the art for each task, the patterns are similar to the ones observed in Section 5.1.1. Both COMET(BART) and BART-Large comfortably surpass the state of the art, whereas fall short for ARC, PIQA and CSQA. The gap to the state of the art for ARC is the largest.

In tasks other than CSQA, RoBERTa-Large remains as the most capable reasoner, obtaining better results than both COMET(BART) and BART-Large.

The same patterns observed in Section 5.1.1 are present here. In terms of the binary tasks, PIQA (physical commonsense) and ARCT (argument reasoning), PIQA remains harder, but not by much, than ARCT for COMET(BART), despite humans finding ARCT harder than PIQA. ARC, which targets science-related commonsense reasoning, continues to be harder than CSQA, with nearly half the accuracy.

5.2.2 Retraining and Evaluation with Partial Input

Task	Full Inputs	Random Score	Full Input Score	Partial Input	Partial Input Score
ARCT	Claim (C) + Reason (R) Warrant 0 & 1 (W)	0.5	0.795	C+R	0.5
				R+W	0.5
				C+W	**0.782**
ARC	Question (Q) + Candidate Answers (A)	0.25	0.422	Q	0.227
				A	**0.344**
PIQA	Goal (G) + Solution 1 & 2 (Sol)	0.5	0.794	Goal	0.495
				Sol	**0.724**
CSQA	Question (Q) + Candidate Answers (A)	0.2	0.727	Q	0.196
				A	0.184

Table 5.4: Partial input training results (accuracy). Scores above random choice are in bold.

Table 5.4 shows the partial input training results for COMET(BART).

ARCT and PIQA continue to be resolved by just taking into account the answers, as this had been observed in Section 5.1.2. The same question arises from this result: what are the models actually solving?

ARC, which previously showed no signs of being solved with partial inputs, emerges here as another "broken" task, with COMET(BART) achieving 0.344 accuracy, just 0.078 shy of the score obtained when looking at the whole input. The model is answering science-related commonsense reasoning questions without looking at the question itself. The fact that RoBERTa could not take advantage of the signals present in the data, and COMET(BART) did, shows the influence of the pre-training regime on the models' capabilities on downstream tasks. BART itself is pre-trained with the same data as RoBERTa, but with different tasks. These differences allow it to learn different linguistic phenomena, which may be helpful in some ways and not in others. On top of that, through the COMET framework, BART was injected with commonsense knowledge. These fundamental differences generate a different solution (in the optimization landscape), such that the model seems to be able to capture different underlying shortcuts.

5.3 Adversarial Attack

As per the implementation described in Section 4.2, the neuro-symbolic model and the best of the neuro-only models, namely COMET(BART) and RoBERTa, are put to the test, being the target of a TextFooler (Jin

Task	Random	Model	Before	After	Δ	Δ%
ARCT	0.5	RoBERTa-Large	0.831	0.476	-0.355	42.7%
		COMET(BART)	0.795	0.512	-0.283	35.5%
ARC	0.25	RoBERTa-Large	0.435	0.157	-0.278	63.9%
		COMET(BART)	0.422	0.107	-0.315	74.7%
PIQA	0.5	RoBERTa-Large	0.795	0.306	-0.489	61.5%
		COMET(BART)	0.794	0.286	-0.508	64.0%
CSQA	0.2	RoBERTa-Large	0.738	0.536	-0.202	27.4%
		COMET(BART)	0.727	0.500	-0.226	31.3%

Table 5.5: Results of the adversarial attack on RoBERTa-Large and COMET(BART), on each task.

et al., 2020) attack. For both RoBERTa-Large and COMET(BART), the best performing model from the five runs for each task is selected as the attack target. Table 5.5 shows the results of the adversarial attack.

Both RoBERTa-Large and COMET(BART) show brittleness, with the neuro-symbolic model, despite having been exposed to fine-grained commonsense facts, still not being any less susceptible to the attack than RoBERTa-Large.

A sharper drop in performance is observed in ARCT, ARC and PIQA, the same tasks that were flagged in Section 5.1.2 and Section 5.2.2 as being "broken". This aligns with the hypothesis that adversarial attacks target non-robust features, present when models learn the task through shortcuts. Since a sharper drop is observed in these tasks, together with the evidence other experiments uncovered, it hints at the possibility that these models are not learning solely the commonsense reasoning task, but learning also to identify certain spurious signals present in the data.

5.4 Cross-Task Evaluation

Testing on other datasets, without having trained on them, provides an excellent opportunity of evaluating the knowledge retention and how well the models can generalize and reason with it. If a given model happens to have found spurious cues for a given dataset, and thus specific only to that dataset, it would try to apply them to other datasets and fail to do so.

Table 5.6 and Table 5.7 show the results of the cross task evaluation, for RoBERTa-Large and COMET(BART), respectively.

All but two directions, one per each model, perform well above the tasks' random baseline. In RoBERTa-Large, the direction PIQA → ARC falls 0.02 below the random baseline, and for COMET(BART),

		Tested On			
		ARCT	ARC	PIQA	CSQA
Trained With	ARCT	*0.831*	0.310	0.571	0.293
	ARC	0.589	*0.435*	0.627	0.343
	PIQA	0.597	0.230	*0.795*	0.552
	CSQA	0.627	0.384	0.687	*0.738*
	Random	0.5	0.25	0.5	0.2

Table 5.6: Cross-task results for RoBERTa (in accuracy). The values in the diagonal are from Table 5.1.

		Tested On			
		ARCT	ARC	PIQA	CSQA
Trained With	ARCT	*0.795*	0.321	0.614	0.350
	ARC	0.542	*0.422*	0.562	0.355
	PIQA	0.580	0.350	*0.794*	0.579
	CSQA	0.592	0.114	0.660	*0.727*
	Random	0.5	0.25	0.5	0.2

Table 5.7: Cross-task results for COMET(BART) (in accuracy). The values in the diagonal are from Table 5.3.

CSQA → ARC does so by 0.136. As a reminder, CSQA is a 5-choice Q&A commonsense reasoning task with a wide array of domains. ARC is an up-to 5-choices Q&A commonsense reasoning task targeting scientific commonsense. A repeating pattern is the target direction, as both instances are tested on ARC. At first glance, this pattern could further reinforce the findings from Section 5.2.2, where ARC is solved by providing COMET(BART) with just the "answers", as the model abuses shortcuts to obtain a good performance. However, that extrapolation cannot be sustained by these results. Models trained on other tasks can stay well above random on ARC, which can be due to transferring knowledge or shortcuts. These two isolated cases cannot thus sustain the hypothesis that models have learned shortcuts that are not transferable to ARC. A possible explanation for these divergences can be what is known as negative transfer (Ruder, 2017), through gradient conflict (Javaloy and Valera, 2021; Levi and Ullman, 2020; Suteu and Guo, 2019; Yu et al., 2020). In more concrete terms, the optimization landscape is different in both tasks. The local optima found in these two instances happen to be a horrible solution in ARC's optimization landscape, diverging from any local optima.

CSQA appears to provide the best prior knowledge out of all the tasks, making sense as it is a very general domain dataset, covering a broad range of commonsense dimensions and reasoning types. It does

not stray far off from a fine-tuned model's scores when applied to other tasks in a zero-shot manner.

ARCT appears to provide the least out of all the tasks, which is somewhat expected as it is not an ordinary commonsense reasoning task. While requiring commonsense reasoning, the task differs not only in the domain, as it covers controversial social topics, but the task itself is different, not so much a Q&A task as the rest of them, but an argument mining task of warrant identification.

The gap to the fine-tune baseline (diagonal values) is shorter in COMET(BART) than in RoBERTa-Large, hinting at the fact that COMET(BART) could be better maximizing the transferability of knowledge (or shortcuts)

Given that the models generally behave well in a zero-shot manner, an hypothesized performance decrease due to non-transferable shortcuts is not observed, and thus the experiment is inconclusive in that regard.

5.5 Data Contamination

Name	Split	N	Total Examples	Dirty Examples	Dirty Percentage	Clean Examples	Clean Percentage
ARCT	test	13	888	0	0%	888	100%
ARC	test	10	1172	14	1.19%	1158	98.81%
PIQA	dev	8	1838	243	13.22%	1595	86.78%
CSQA	dev	8	1221	62	5.08%	1159	94.92%

Table 5.8: Data contamination statistics for each task. An example is considered dirty if it has at least a single N-gram (N Value in 3rd column) collision with any of the pre-training datasets.

The results obtained in the previous sections motivates an inquiry into a possible contamination of the tasks' respective test sets, whereby the examples might have been observed during the pre-training phase. The methodology followed for this study is described in Section 4.4.

Table 5.8 shows the outcome of such study analysis.

Interestingly, one of the most affected tasks, ARCT, is entirely clean, with none of the examples being flagged for an overlap with any of the pre-training testsets. This effectively eliminates the option of memorization as an explanation for the shortcut learning observed, and since previous work (Niven and Kao, 2019) eliminated trivial lexical spurious cues, the only remaining explanation for the brittleness of the task is highly non-linear shortcuts in the data that models exploit.

The remaining tasks exhibit different levels of contamination. ARC was flagged for 1.19% of its testset, 14 examples out of 1172, the lowest of the contaminated tasks. CSQA follows ARC with 5.08% contamination, 62 examples out of 1221. The most contaminated task is PIQA, with 13.22% of the devset contaminated, 243 examples out of 1838.

An additional experiment was performed to verify the level of data contamination between the tasks testsets. It was found that they share no n-grams between themselves, meaning there is no data contamination beween tasks. This effectively rules out the possibility of memorization being an explanatory factor for the Cross-Task Evaluation experiment (Section 5.4).

Name	Split	Original Accuracy Score	Accuracy Score Dirty Set	Accuracy Score Clean Set
ARC	test	0.435	0.714 (+0.279)	0.432 (-0.003)
PIQA	dev	0.795	0.835 (+0.040)	0.789 (-0.006)
CSQA	dev	0.738	0.726 (-0.012)	0.739 (+0.001)

Table 5.9: RoBERTa's accuracy when tested on the full testset (Original Accuracy Score), on the Dirty Set (contains only dirty examples) and Clean Set (contains only clean examples).

Name	Split	Original Accuracy Score	Accuracy Score Dirty Set	Accuracy Score Clean Set
ARC	test	0.422	0.643 (+0.221)	0.420 (+0.002)
PIQA	dev	0.794	0.819 (+0.025)	0.790 (-0.004)
CSQA	dev	0.727	0.710 (-0.017)	0.727 (+0.000)

Table 5.10: COMET(BART)'s accuracy when tested on the full testset (Original Accuracy Score), on the Dirty Set (contains only dirty examples) and Clean Set (contains only clean examples).

To further understand the impact of the contamination on the accuracy scores obtained, and to what extent memorization may be at play, two sets were created from the testsets/devsets: one set denoted as "Dirty Set", containing only dirty examples, and another denoted as "Clean Set", containing only clean examples.

Table 5.9 and Table 5.10 shows the performance of RoBERTa and COMET(BART), respectively, on these two sets and the original testset/devset.

Performance of ARC increases considerably on the Dirty Set; however, since the set is so tiny (14 examples), the impact on the overall score is marginal, such that in the Clean Set, there is no clear advantage or disadvantage in having the dirty examples present. Nevertheless, both models do find those examples much easier, hinting at the impact that memorization can have on the evaluation process.

PIQA, the most contaminated task, appears to have had its accuracy score slightly inflated through memorization. Performance on the Dirty Set is slightly better than in the Clean Set, which in turn is lower than the original accuracy score, albeit not by much.

CSQA, differently from the previous two tasks, actually shows degradation of performance due to contamination. Performance on the Dirty Set is lower than on the Clean Set, meaning the model must be

conflating concepts, not generalizing well to the clean examples, bringing the performance down.

Overall, models trained on ARC greatly benefit from data contamination, and a slight benefit was also detected for PIQA, but the same does not happen for ARCT and CSQA. Given that data contamination only provided substantial performance gains in one of four tasks, the behavior that leads the models not to learn the task meant to be conveyed by the datasets cannot be fully explained by memorization alone. Thus shortcuts should be present in the data.

5.6 Shortcut Exploration

In this section, the results of two experiments looking to identify shortcuts are presented. The first experiment, which checks class balance of each task dataset, is discussed in Section 5.6.1. Section 5.6.2, in turn, covers the analysis of possible lexical cues present in each task dataset.

5.6.1 Class Balance

Table 5.11 contains the statistics of target distribution for each task dataset split.

ARCT, PIQA and CSQA appear to be quite well balanced. Despite PIQA and CSQA not being perfectly balanced, the gap between the unbalanced classes and random choice is so small that no real advantage are likely to be obtained from it.

ARC is slightly more unbalanced than the other tasks, although not by much either. In the train split, candidate answer in the first position, labeled 0 in the table because it is zero-indexed, has a diminished presence in the split. The remaining three candidate answer positions are relatively well balanced, meaning it would be difficult for a model to take advantage of the slight unbalance. In the development and test splits, the first position is also underrepresented, and two classes have a bigger present than the others, creating a slight unbalance.

In spite of this, the unbalance cannot explain the results for ARC on the previous experiments, as the model learns from the train split, and as such in order for the model to learn to take advantage of this unbalance, it would have to be present in the train split as well. Since the train split is relatively well balanced, the models should not pick up on such signal.

5.6.2 Lexical Cues

In this section, a discussion ensues from the cues detected using two of metrics described in Section 4.5: coverage (ξ_k), which measures the percentage of the total examples in which the cue is present; and productivity (π_k), which indicates the proportion of examples where the cue predicts a correct answer. A useful cue has a productivity above random choice, $\pi_k > 1/m$, where m is the number of candidate answers.

ARCT Cues. Table 5.12 presents the top ten unigram and bigram cues in ARCT's dataset. The most extensive coverage is achieved with "not", which is incidentally the most useful cue in the original dataset, with a coverage of 0.64 and productivity of 0.61. The dataset was balanced in order to nullify the productivity of the cues (Niven and Kao, 2019). In the revised dataset, "not" has a coverage of 0.38 and a productivity of 0.5 (random chance). Since the dataset has been revised with the aid of these metrics, it is expected that there are no high coverage useful unigram and bigram cues lingering, evidenced by the fact that none of the cues detected had productivity greater than 1/2.

ARC Cues. In Table 5.13, the top ten unigram and bigram cues present in ARC's dataset are shown. Three useful bigrams were detected, and five useful unigram cues are present in the top ten: "to", "and", "on", "for", "an". The coverage of these unigram cues are relatively low, with the largest coverage being 0.13 (13% of the dataset) corresponding to cue "to", however, its productivity is very near the random baseline of 0.25. The remaining cues have coverages of 0.06 and 0.05, with the largest productivity being 0.41 for cue "and". This cue is thus a strong signal when found in only one of the candidate answer, but because it is such a common function word, it rarely is found in only one of the candidates, having a low application because of this.

ARC therefore has lexical cues present in the dataset, although their coverage and productivity are so low that it cannot explain the scores obtained.

PIQA Cues. Table 5.14 outlines the top 10 (in coverage) unigram and bigram cues for PIQA's dataset. Four useful unigram and two useful bigram cues are present in the top 10. Three of the unigrams are function words, and the cue "a" has the most coverage with 0.10. The remaining cues hover between 0.07 and 0.01 coverage. The cues that were found have low productivity, nearing the random choice mark (0.5), and when considering their low coverage, it is an indication that the dataset does not have "anchors" from which models can fully take advantage of.

CSQA Cues. Top 10 (in coverage) unigram and bigram cues found for CSQA's dataset are shown in Table 5.15. Similar to what can be observed in ARC's and PIQA's dataset, CSQA contains a few unigram and bigram cues, but their coverage and productivity are so low that the performance cannot be attributed to their presence. Five unigram and six bigram cues are useful, as their productivity is greater than 0.2 (random baseline). The coverage for the bigram cues is just 0.01, while for the unigrams is in the range of 0.07 to 0.03. The productivity for the cues does not stray far from the random baseline, but three bigrams achieved a productivity of 0.27 or greater, which is a considerable gap to the random baseline.

In light of the results, one might be tempted to conjecture that no real useful cues are present in the dataset. This conjecture, however, would not stand as the metrics used allow the detection of contiguous lexical cues, because they function with ngrams. It could be that the dataset contains other lexical cues that are not so linear, e.g. when the word "boat" appears in the third position in the sentence, and "car" in the eight position, the answer is 0.

There are also other types of cues which are not strictly lexical, but instead rely on the features of word embeddings and hidden states. These require other types of analysis and are not trivial to find and interpret. It is an emerging topic of research.

Task	Split	Choice Number	Occurrences	Relative Frequency	Random Chance
ARCT	Train	0	1210	0.500	0.500
		1	1210	0.500	
	Development	0	316	0.500	
		1	316	0.500	
	Test	0	444	0.500	
		1	444	0.500	
ARC	Train	0	239	0.214	0.250
		1	296	**0.265**	
		2	291	**0.260**	
		3	293	**0.262**	
	Development	0	64	0.214	
		1	73	0.244	
		2	78	**0.261**	
		3	83	**0.278**	
		4	1	0.003	
	Test	0	266	0.227	
		1	311	**0.265**	
		2	310	**0.265**	
		3	285	0.243	
PIQA	Train	0	8053	0.500	0.500
		1	8060	0.500	
	Development	0	910	0.495	
		1	928	**0.505**	
CSQA	Train	0	1909	0.196	0.200
		1	1973	**0.203**	
		2	1946	0.200	
		3	1985	**0.204**	
		4	1928	0.198	
	Development	0	239	0.196	
		1	255	**0.209**	
		2	241	0.197	
		3	251	**0.206**	
		4	235	0.192	

Table 5.11: Class balance for each task dataset split. Relative frequency in bold indicates a frequency above random chance.

Unigrams			Bigrams		
Unigram	Coverage (ξ_k)	Productivity (π_k)	Bigram	Coverage (ξ_k)	Productivity (π_k)
(not,)	0.38	0.5	(is, not)	0.09	0.5
(do,)	0.12	0.5	(are, not)	0.07	0.5
(does,)	0.06	0.5	(do, not)	0.04	0.5
(can,)	0.06	0.5	(can, not)	0.03	0.5
(to,)	0.06	0.5	(does, not)	0.03	0.5
(and,)	0.05	0.5	(not, be)	0.03	0.5
(no,)	0.04	0.5	(is, a)	0.03	0.5
(a,)	0.04	0.5	(can, be)	0.02	0.5
(ca,)	0.04	0.5	(will, not)	0.02	0.5
(be,)	0.04	0.5	(not, a)	0.02	0.5
(more,)	0.03	0.5	(to, be)	0.02	0.5

Table 5.12: Top 10 unigram and bigram cues with regards to coverage, in descending order, for the ARCT dataset.

Unigrams			Bigrams		
Unigram	Coverage (ξ_k)	Productivity (π_k)	Bigram	Coverage (ξ_k)	Productivity (π_k)
(to,)	**0.13**	**0.26**	(of, the)	0.07	0.15
(in,)	0.13	0.25	(in, the)	0.06	0.24
(of,)	0.13	0.25	(to, the)	0.04	0.24
(a,)	0.11	0.22	(amount, of)	0.03	0.25
(the,)	0.09	0.25	**(from, the)**	**0.03**	**0.27**
(water,)	0.09	0.15	**(in, a)**	**0.03**	**0.30**
(from,)	0.07	0.23	(on, the)	0.03	0.22
(and,)	**0.06**	**0.41**	**(the, same)**	**0.02**	**0.30**
(on,)	**0.06**	**0.26**	(number, of)	0.02	0.16
(for,)	**0.05**	**0.29**	(the, amount)	0.02	0.24
(an,)	**0.05**	**0.29**	(of, a)	0.02	0.25

Table 5.13: Top 10 unigram and bigram cues with regards to coverage, in descending order, for the ARC dataset. In bold are cues whose productivity $\pi_k > 1/4$, indicating a useful cue.

Unigrams			Bigrams		
Unigram	Coverage (ξ_k)	Productivity (π_k)	Bigram	Coverage (ξ_k)	Productivity (π_k)
(a,)	**0.10**	**0.52**	(in, the)	0.03	0.41
(of,)	0.07	0.50	**(on, the)**	**0.03**	**0.54**
(to,)	0.07	0.49	(of, the)	0.03	0.50
(and,)	**0.07**	**0.52**	(with, a)	0.03	0.47
(in,)	0.06	0.47	**(use, a)**	**0.02**	**0.51**
(on,)	**0.06**	**0.53**	(to, the)	0.02	0.47
(the,)	0.05	0.40	(in, a)	0.02	0.50
(with,)	0.05	0.47	(and, then)	0.02	0.43
(it,)	0.05	0.48	**(into, the)**	**0.01**	**0.52**
(water,)	**0.04**	**0.52**	(top, of)	0.01	0.45
(your,)	0.04	0.45	(the, top)	0.01	0.47

Table 5.14: Top 10 unigram and bigram cues with regards to coverage, in descending order, for the PIQA dataset. In bold are cues whose productivity $\pi_k > 1/2$, indicating a useful cue.

Unigrams			Bigrams		
Unigram	Coverage (ξ_k)	Productivity (π_k)	Bigram	Coverage (ξ_k)	Productivity (π_k)
(store,)	**0.07**	**0.24**	(go, to)	0.01	0.20
(house,)	0.06	0.19	**(new, york)**	**0.01**	**0.21**
(to,)	0.06	0.17	(grocery, store)	0.01	0.20
(of,)	0.06	0.19	**(have, fun)**	**0.01**	**0.25**
(in,)	0.05	0.12	(talk, to)	0.01	0.06
(office,)	**0.03**	**0.23**	**(office, building)**	**0.01**	**0.25**
(city,)	**0.03**	**0.22**	**(friend, house)**	**0.01**	**0.27**
(room,)	**0.03**	**0.23**	(each, other)	0.01	0.10
(school,)	0.03	0.19	**(neighbor, house)**	**0.01**	**0.28**
(get,)	**0.03**	**0.23**	(living, room)	0.01	0.19
(park,)	0.03	0.16	**(music, store)**	**0.01**	**0.34**

Table 5.15: Top 10 unigram and bigram cues with regards to coverage, in descending order, for the CSQA dataset. In bold are cues whose productivity $\pi_k > 1/5$, indicating a useful cue.

5.7 Summary

This chapter presented the results obtained in this dissertation for several experiments. The models are, at face value, very capable reasoners and appear to be catching up to humans, although they face a steep battle still. A new state-of-the-art result is obtained for the revised ARCT dataset.

Over the course of the chapter, different experiments were laid out that aimed to show that the models are not learning the task meant to be conveyed by the datasets. Instead, they appear to be learning different tasks due to greedy learning through shortcuts.

When presented with partial inputs, models can perform just as well as with the whole inputs. They were shown to be brittle also when faced with adversarial attacks.

Models seem able to transfer knowledge from one task to another, which can be attributed either to generalized knowledge or transferable shortcuts, or both.

A data contamination study was performed, finding that 3/4 tasks were contaminated, although with little impact on the accuracy score when examined further.

COMET(BART), despite possessing more refined priors through its neuro-symbolic pre-training, does not behave any different from RoBERTa, showing the same (and even worse, in part) behaviour in terms of shortcut learning and general brittleness.

Lastly, an exploration for possible shortcuts was performed. The datasets were found to be, in different degrees, quite balanced, and as such that cannot be responsible for the behavior observed. Unigram and bigram lexical cues were found in ARC, PIQA and CSQA. However, their coverage and productivity are so low that they cannot explain the shortcutting detected with other experiments.

Chapter 6

Conclusion

This chapter concludes this dissertation. Section 6.1 provides an overall summary of its contents. In Section 6.2, a list of contributions set forth by this dissertation is highlighted, and lastly, Section 6.3 sketches future work elicited by the lingering questions.

6.1 Summary

While aiming at experimenting with deep learning of commonsense reasoning, this dissertation had two connected sub-objectives: to explore how mainstream deep learning-based language models perform on a quintessential AI task, commonsense reasoning, and to tamper with the limits of these techniques, supported by artificial neural networks which are different from the human capacity they seek to approximate.

Five state-of-the-art architectures were leveraged to learn each four prominent commonsense reasoning NLP tasks, in the English language. A new state-of-the-art result was established for the revised ARCT dataset. The results obtained for the remaining tasks fall slightly below the state of the art established in the literature, and exhibit a gap to human performance.

One of the architectures was a neuro-symbolic architecture enriched with commonsense knowledge coming from a commonsense knowledge graph. The conjecture that this could bring extra performance, through a fine-grained prior, eventually was not empirically supported.

The best performing model from Section 5.1.1, RoBERTa, was selected along with COMET(BART), the neuro-symbolic model, to undergo stress testing to cater to possible frailties in the generalization capacity of their architectures.

First, models trained on partial inputs (obtained by removing specific segments, such as the question) obtained scores very close to models trained with the whole input sequence. This phenomenon surfaces in three of the four tasks considered, namely ARCT (Argument Reasoning Comprehension Task: given a reason and a claim, select the correct warrant from two offered options), ARC (AI2 Reasoning Challenge:

multiple-choice natural science Q&A task for questions from 3rd to 9th-grade science exams) and PIQA (Physical Interaction Question Answering: a binary-choice for solutions to attain a given goal in the realm of physical commonsense). This is solid evidence that the models are not learning the task meant to be instantiated by the examples in the dataset, instead leveraging shortcuts in the data to minimize loss, shortcutting the task to be performed.

Second, the adversarial attacks performed showed the brittleness of the models. In all four tasks, severe degradation of performance was observed. An additional interesting pattern emerged from this experiment: the degradation is more significant for the tasks that were flagged in the partial input experiment (ARCT, ARC and PIQA). As adversarial attacks exploit non-robust feature dependence, this further reinforces the point that the models should be exploiting some shortcuts.

Third, the models were cross-evaluated in a zero-shot manner and obtained scores that are overall comfortably above the random baseline. These results can be due to genuine knowledge transfer, but also could be due to the transferring shortcuts detected with the experiments mentioned above, thus leaving open the research to their ultimate explanation.

Fourth, a data contamination study was carried out, revealing different levels of contamination in the testsets. ARCT is entirely clean, while ARC, PIQA and CSQA (Commonsense Question Answering: multiple-choice question answering task regarding multiple commonsense domains) have been flagged with, respectively, 1.19%, 13.22% and 5.08% of overlap between their pre-training and fine-tuning datasets. This supports one of the downsides of the pre-training methodology: downstream task examples may have been already input during the pre-training phase, potentially inflating results. Further experiments were performed to illustrate this point. The models did show a slight advantage, although so insignificant that memorization during pre-training cannot be what explains the inflated results, thus implying that eventual shortcuts should be searched for be searched for elsewhere.

Finally, the datasets were checked for two different types of possible shortcuts. The first type of shortcut is class imbalance and the second is lexical cues. In terms of class balance, the only dataset which exhibited slight unbalance was ARC, although not enough to be considered a problem. Regarding lexical cues, several of them were identified for ARC, PIQA and CSQA, but their coverage and productivity are low, implying that the inflated results cannot be completely due to them.

Throughout this dissertation, different conjectures emerge in the efforts to find an explanation for the seemingly shortcut learning by the models,

Number of candidate answers. In Section 5.1.2, tasks with more than two candidate answers (non-binary) seemed to be more robust to the partial input training stress. This hinted at the possibility that having more answers to choose from helps to improve the generalization capabilities and avoid the learning of shortcuts. However, such could not be confirmed after further experimentation, as converting the tasks to a binary choice yielded the same results as obtained with their multiple-choice version, not going above the random baseline. In Section 5.2.2, where the same methodology was applied to a neuro-symbolic model, ARC was able to be solved by only providing the answers, further confirming that the number of candidate answers is not a deciding factor.

Data Contamination. The pre-train-then-fine-tune methodology, which has become standard practice, carries with it the danger of data contamination. The pre-training corpus is commonly gathered from mass scraping the web. It is a possibility that when fine-tuning a model on a given task, the testset in which the model is tested on shares the same sources with a portion of the pre-train corpus. If the model has memorized that portion of text, it can potentially use it, breaking the assumption of an independent testset, and potentially inflating the performance. The pre-training corpus and the proposed tasks testsets were checked for textual overlap. The results indicate some overlap, but that overlap did not translate into meaningful performance gains upon further inspection. As such, it cannot be the explanation for the shortcut learning.

Class Balance. Class imbalance can be a powerful shortcut that models can take advantage of. However, the experiments showed that ARCT, PIQA and CSQA are well balanced, and ARC is slightly unbalanced but not to an extent where the model could benefit greatly. Therefore, the results cannot be accounted for with class imbalance.

Lexical Cues. In ARCT, lexical cues, such as the word "not", were powerful indicators of the correct warrant. Experiments in this dissertation were conducted to detect unigram and bigram cues which could plague the testsets. The outcome of the experiments show presence of a few cues in ARC, PIQA and CSQA; however, their coverage and productivity are so low that they should not be able to guide models to the performances.

6.2 Contributions

The major contributions of this dissertation are the following:

- **A study of neural language models applied to Commonsense Reasoning.** An exploratory study was performed, using models from major families of the state-of-the-art Transformer architectures. RoBERTa emerged as the most capable reasoner.
- **Stress study on language models.** The stress experiments performed rendered evidence that the models are, after all, not performing the commonsense reasoning tasks in the way meant for them. Instead, they appear to be funneling on spurious signals present in the data, exploiting them to boost performance.
- **Data contamination library.**[1] Expanding on existing work on data contamination, a library was produced and open sourced to allow researchers to analyze possible overlaps between the datasets used in pre-training and the testsets of the downstream tasks.
- **Three research papers produced.** Three research papers have been produced based on this body of work. Two of them have passed peer review and were accepted for publication (Branco et al.,

[1] https://github.com/nlx-group/overlapy

2021a,b). A third paper is still in preparation and is unpublished at the time of writing this dissertation.

In order to permit the reproduction and replication of our results reported in this dissertation, our code and relevant materials are publicly available at: https://github.com/nlx-group/study-of-commonsense-reasoning.

6.3 Future Work

The results obtained in this dissertation provide hints of the power of deep neural networks to find patterns in data to minimize loss, regardless of their potential to support ample generalization. These hints support the hypothesis that, to a considerable extent, this is the process guiding the learning of these tasks but offer no explanation on two major questions:

1. How may we identify these shortcuts? Trivial shortcuts are identifiable using co-occurrence statistics, for example. However, the uncovering of non-trivial shortcuts, such as a possible non-linear combination of features, are still an unsolved aspect in NLP and deep learning.

2. How may we accurately identify the mechanisms responsible for these shortcuts? This is an important question, which naturally follows from the first one. Even if we are able to identify these shortcuts, how can we design a learning process free of such error inducing procedures?

The first question falls within the scope of interpretability of NLP models. This research line aims to create methods to interpret decisions made by neural networks, which can potentially unveil such data artifacts, such as saliency maps, which highlight the importance of words in the sentence. Saliency maps can be built using gradients (Simonyan et al., 2013; Sundararajan et al., 2017; Smilkov et al., 2017), LIME (Ribeiro et al., 2016) and using attention (Clark et al., 2019; Lee et al., 2017; Vig and Belinkov, 2019; Wu et al., 2020; Xu et al., 2015). Alternative methods are also being developed with the concept of influence functions, which model the influence of dataset examples, and its words, on the performance of the model (Guo et al., 2020; Han et al., 2020; Jacovi et al., 2021; Kobayashi et al., 2020). While these methods are promising, they mostly provide local explanations, making it difficult to identify highly non-linear shortcuts. To the best of our knowledge, no clear methodology has been developed to combat shortcut learning using these methods.

A research line has also been established regarding the second question. More robust methods of learning have been proposed in the literature, such as Adversarial Training (Goodfellow et al., 2015; Huang et al., 2017; Papernot et al., 2016; Shaham et al., 2015), which consists of producing adversarial datasets during the training process, and Meta-Learning (Finn et al., 2017; Santoro et al., 2016; Schmidhuber, 1987), where models learn how to learn. These methods were designed to create more robust

learning techniques to mitigate the symptoms of shortcut learning but were not motivated by any mechanism underlying it. Identifying such mechanisms could allow for more precise methods of learning to counter-act it.

These two non-trivial questions are thus left unanswered by this dissertation and are open research questions in the literature overall. The previous two techniques do provide a starting ground from which research the tackling of these questions could begin. However, the highly non-linear and complex nature of the parameter space of these large deep learning models represents a difficult challenge.

References

How People Learn: Brain, Mind, Experience, and School. The National Academies Press, Washington, DC, 1999. doi: 10.17226/6160. URL https://www.nap.edu/catalog/6160/how-people-learn-brain-mind-experience-and-school. 22

M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1316. URL https://www.aclweb.org/anthology/D18-1316. 24

M. Baard. Ai founder blasts modern research, Mar 2018. URL https://www.wired.com/2003/05/ai-founder-blasts-modern-research/. 7

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 11, 13

S. Bayer, L. Damianos, C. Doran, L. Ferro, R. Fish, L. Hirschman, I. Mani, L. Riek, and B. Oshika. Selected Grand Challenges in Cognitive Science. Technical report, 2005. URL https://www.mitre.org/publications/technical-papers/selected-grand-challenges-in-cognitive-science-search-engine-for-the. 31

Y. Bengio, P. Frasconi, and P. Simard. The problem of learning long-term dependencies in recurrent networks. In *IEEE international conference on neural networks*, pages 1183–1188. IEEE, 1993. 11

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. 11

Y. Bisk, A. Holtzman, J. Thomason, J. Andreas, Y. Bengio, J. Chai, M. Lapata, A. Lazaridou, J. May, A. Nisnevich, N. Pinto, and J. Turian. Experience grounds language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8718–8735, Online, Nov. 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.703. URL https://www.aclweb.org/anthology/2020.emnlp-main.703. 33

Y. Bisk, R. Zellers, R. LeBras, J. Gao, and Y. Choi. Piqa: Reasoning about physical commonsense in natural language. In *AAAI*, pages 7432–7439, 2020b. XIII, XV, 33, 34, 55, 60, 63

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NIPS)*, pages 1–9, 2013. 25

A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, and Y. Choi. Comet: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, 2019. 7, 25, 39

R. J. Brachman and H. J. Levesque. The tractability of subsumption in frame-based description languages. In *AAAI*, volume 84, pages 34–37, 1984. 6

R. Branco, J. Rodrigues, C. Saedi, and A. Branco. Assessing wordnets with WordNet embeddings. In *Proceedings of the 10th Global Wordnet Conference*, pages 253–259, Wroclaw, Poland, July 2019. Global Wordnet Association. URL https://www.aclweb.org/anthology/2019.gwc-1.32. 25

R. Branco, A. Branco, J. Silva, and J. Rodrigues. Shortcutted commonsense: Data spuriousness in deep learning of commonsense reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, Nov. 2021a. Association for Computational Linguistics. To appear. V, IX, 3, 77

R. Branco, A. Branco, J. Silva, and J. Rodrigues. Commonsense reasoning: how do neuro-symbolic and neuro-only approaches compare? In *Proceedings of the CIKM 2021 Workshops*, Online, Nov. 2021b. CEUR-WS. To appear. V, IX, 4, 78

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 11, 18, 54

D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Céspedes, S. Yuan, C. Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018. 24, 52

S. Chakraborty, R. Tomsett, R. Raghavendra, D. Harborne, M. Alzantot, F. Cerutti, M. Srivastava, A. Preece, S. Julier, R. M. Rao, T. D. Kelley, D. Braines, M. Sensoy, C. J. Willis, and P. Gurram. Interpretability of deep learning models: A survey of results. In *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1–6, 2017. doi: 10.1109/UIC-ATC.2017.8397411. 20

J. Cheng, L. Dong, and M. Lapata. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, 2016. 11

E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, and L. Zettlemoyer. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1241. URL https://www.aclweb.org/anthology/D18-1241. 55

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 10

K. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990. 32

K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL https://www.aclweb.org/anthology/W19-4828. 20, 78

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. XIII, 24, 31

J. D. Hwang, C. Bhagavatula, R. Le Bras, J. Da, K. Sakaguchi, A. Bosselut, and Y. Choi. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. In *AAAI'21*, March 2021. URL https://allenai.org/data/atomic-2020. 7, 55

A. D'Amour, K. Heller, D. Moldovan, B. Adlam, B. Alipanahi, A. Beutel, C. Chen, J. Deaton, J. Eisenstein, M. D. Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020. 20

E. Davis. *Representations of commonsense knowledge*. Morgan Kaufmann, 2014. 8

E. Davis and G. Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103, 2015. 7, 9

G. De Palma, B. T. Kiani, and S. Lloyd. Deep neural networks are biased towards simple functions. *arXiv preprint arXiv:1812.10156*, 2018. 20

A. J. DeGrave, J. D. Janizek, and S.-I. Lee. Ai for radiographic covid-19 detection selects shortcuts over signal. *Nature Machine Intelligence*, pages 1–10, 2021. 21

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423. 11, 17, 39, 40, 52

P. Domingos. *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books, 2015. 7

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 11

D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*, 2019. 55

S. Edunov, M. Ott, M. Auli, and D. Grangier. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1045. URL https://www.aclweb.org/anthology/D18-1045. 11

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017. 78

J. B. Freeman. *Argument Structure:: Representation and Theory*, volume 18. Springer Science & Business Media, 2011. 30

L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020. 55

S. Garg and G. Ramakrishnan. Bae: Bert-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, 2020. 24, 50

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252, 2017. 16

R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, Nov 2020. ISSN 2522-5839. doi: 10.1038/s42256-020-00257-z. URL https://doi.org/10.1038/s42256-020-00257-z. 2, 20

M. Geva, Y. Goldberg, and J. Berant. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1161–1166, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1107. URL https://www.aclweb.org/anthology/D19-1107. 2, 21

A. Gokaslan and V. Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019. 41, 55

I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572. 23, 78

A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014. 13

A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016. 25

G. M. A. Grube et al. *Meno*. Hackett Publishing, 1980. 7

H. Guo, N. F. Rajani, P. Hase, M. Bansal, and C. Xiong. Fastif: Scalable influence functions for efficient model interpretation and debugging. *arXiv preprint arXiv:2012.15781*, 2020. 78

S. Gururangan, S. Swayamdipta, O. Levy, R. Schwartz, S. Bowman, and N. A. Smith. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2017. URL https://www.aclweb.org/anthology/N18-2017. 2, 21

I. Habernal, H. Wachsmuth, I. Gurevych, and B. Stein. The argument reasoning comprehension task: Identification and reconstruction of implicit warrants. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1930–1940, New Orleans, Louisiana, June 2018a. Association for Computational Linguistics. doi: 10.18653/v1/N18-1175. URL https://www.aclweb.org/anthology/N18-1175. XV, 21, 29, 60, 63

I. Habernal, H. Wachsmuth, I. Gurevych, and B. Stein. SemEval-2018 task 12: The argument reasoning comprehension task. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 763–772, New Orleans, Louisiana, June 2018b. Association for Computational Linguistics. doi: 10.18653/v1/S18-1121. URL https://aclanthology.org/S18-1121. 30

F. Hamborg, N. Meuschke, C. Breitinger, and B. Gipp. news-please: A generic news crawler and extractor. In *Proceedings of the 15th International Symposium of Information Science*, pages 218–223, March 2017. doi: 10.5281/zenodo.4120316. 56

X. Han, B. C. Wallace, and Y. Tsvetkov. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5553–5563, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.492. URL https://www.aclweb.org/anthology/2020.acl-main.492. 20, 78

E. E.-D. Hemdan, M. A. Shouman, and M. E. Karar. Covidx-net: A framework of deep learning classifiers to diagnose covid-19 in x-ray images. *arXiv preprint arXiv:2003.11055*, 2020. 21

D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 47

T. Henighan, J. Kaplan, M. Katz, M. Chen, C. Hesse, J. Jackson, H. Jun, T. B. Brown, P. Dhariwal, S. Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020. 18

S. J. Hespos and E. S. Spelke. Conceptual precursors to language. *Nature*, 430(6998):453–456, 2004. 33

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 10

S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. Adversarial attacks on neural network policies. *arXiv*, 2017. URL https://arxiv.org/abs/1702.02284. 78

J. D. Hwang, C. Bhagavatula, R. L. Bras, J. Da, K. Sakaguchi, A. Bosselut, and Y. Choi. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. *arXiv preprint arXiv:2010.05953*, 2020. XIV, 25, 26, 39, 47, 48

A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 125–136. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/e2c420d928d4bf8ce0ff2ec19b371514-Paper.pdf. 21, 23

J.-H. Jacobsen, J. Behrmann, R. Zemel, and M. Bethge. Excessive invariance causes adversarial vulnerability. *arXiv preprint arXiv:1811.00401*, 2018. 20

A. Jacovi, S. Swayamdipta, S. Ravfogel, Y. Elazar, Y. Choi, and Y. Goldberg. Contrastive explanations for model interpretability. *arXiv preprint arXiv:2103.01378*, 2021. 78

A. Javaloy and I. Valera. Rotograd: Dynamic gradient homogenization for multi-task learning. *arXiv preprint arXiv:2103.02631*, 2021. 66

R. Jia and P. Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1215. URL https://www.aclweb.org/anthology/D17-1215. XIII, 22

D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8018–8025, Apr. 2020. doi: 10.1609/aaai.v34i05.6311. URL https://ojs.aaai.org/index.php/AAAI/article/view/6311. XIII, 24, 50, 64

P. Kapanipathi, I. Abdelaziz, S. Ravishankar, S. Roukos, A. G. Gray, R. F. Astudillo, M. Chang, C. Cornelio, S. Dana, A. Fokoue, D. Garg, A. Gliozzo, S. Gurajada, H. Karanam, N. Khan, D. Khandelwal, Y.-S. Lee, Y. Li, F. P. S. Luus, N. Makondo, N. Mihindukulasooriya, T. Naseem, S. Neelam, L. Popa, R. G. Reddy, R. Riegel, G. Rossiello, U. Sharma, G. P. S. Bhargav, and M. Yu. Question answering over knowledge bases by leveraging semantic parsing and neuro-symbolic reasoning. *CoRR*, abs/2012.01707, 2020. URL https://arxiv.org/abs/2012.01707. 7

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 18

D. Kaushik and Z. C. Lipton. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1546. URL https://www.aclweb.org/anthology/D18-1546. 2, 21

D. Khashabi, S. Min, T. Khot, A. Sabharwal, O. Tafjord, P. Clark, and H. Hajishirzi. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.171. URL https://www.aclweb.org/anthology/2020.findings-emnlp.171. 11

W. Kintsch and T. A. Van Dijk. Toward a model of text comprehension and production. *Psychological review*, 85(5):363, 1978. 8

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 53

S. Kobayashi, S. Yokoi, J. Suzuki, and K. Inui. Efficient estimation of influence of a training instance. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 41–47, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.sustainlp-1.6. URL https://www.aclweb.org/anthology/2020.sustainlp-1.6. 78

T. Kudo and J. Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://www.aclweb.org/anthology/D18-2012. 16

V. Kuleshov, S. Thakoor, T. Lau, and S. Ermon. Adversarial examples for natural language classification problems. 2018. 24

J. Lee, J.-H. Shin, and J.-S. Kim. Interactive visualization and manipulation of attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 121–126, 2017. 20, 78

D. B. Lenat, M. Prakash, and M. Shepherd. Cyc: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI magazine*, 6(4):65–65, 1985. 7, 9

H. Levi and S. Ullman. Multi-task learning by a top-down control network. *arXiv preprint arXiv:2002.03335*, 2020. 66

M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://www.aclweb.org/anthology/2020.acl-main.703. 47

D. Li, Y. Zhang, H. Peng, L. Chen, C. Brockett, M.-T. Sun, and B. Dolan. Contextualized perturbation for textual adversarial attack. *arXiv preprint arXiv:2009.07502*, 2020a. 24

J. Li, S. Ji, T. Du, B. Li, and T. Wang. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*, 2018. 24

L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu. Bert-attack: Adversarial attack against bert using bert. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, 2020b. 24, 50

H. Liu and P. Singh. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004. 8, 34, 47

P. Liu, X. Qiu, and X. Huang. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2873–2879. AAAI Press, 2016. ISBN 9781577357704. 11

X. Liu, P. He, W. Chen, and J. Gao. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, 2019a. 20

X. Liu, K. Duh, L. Liu, and J. Gao. Very deep transformers for neural machine translation. *arXiv preprint arXiv:2008.07772*, 2020. 11

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019b. 11, 17, 18, 39, 40

E. Loper and S. Bird. Nltk: The natural language toolkit. *CoRR*, cs.CL/0205028, 2002. URL `http://dblp.uni-trier.de/db/journals/corr/corr0205.html#cs-CL-0205028`. 55

M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015. 13

J. Martens and I. Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1033–1040. Citeseer, 2011. 11

J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12–12, 2006. 1

J. McCarthy et al. *Programs with common sense*. RLE and MIT computation center, 1960. 8

M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 53

S. Mehta, M. Ghazvininejad, S. Iyer, L. Zettlemoyer, and H. Hajishirzi. Delight: Deep and light-weight transformer. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=ujmgfuxSLr0. 11

H. Mercier and D. Sperber. *The enigma of reason*. Harvard University Press, 2017. 8

G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 9

S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao. Deep learning based text classification: A comprehensive review. *arXiv preprint arXiv:2004.03705*, 2020. 11

M. Minsky. *Society of mind*. Simon and Schuster, 1988. 8

D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of sat problems. In *AAAI*, volume 92, pages 459–465. Citeseer, 1992. 6

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015. 9

J. Morris, E. Lifland, J. Lanchantin, Y. Ji, and Y. Qi. Reevaluating adversarial examples in natural language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3829–3839, 2020a. 23, 52

J. X. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp, 2020b. 50

M. Mosbach, M. Andriushchenko, and D. Klakow. On the stability of fine-tuning {bert}: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=nzpLWnVAyah. 52

N. Mrkšić, D. Ó Séaghdha, B. Thomson, M. Gašić, L. Rojas-Barahona, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young. Counter-fitting word vectors to linguistic constraints. In *Proceedings of HLT-NAACL*, 2016. 50

S. Nagel. Cc-news, Oct 2016. URL https://commoncrawl.org/2016/10/news-dataset-available/. 41, 55

T. Niven and H.-Y. Kao. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1459. URL https://www.aclweb.org/anthology/P19-1459. 2, 21, 31, 56, 60, 62, 67, 70

T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, and U. R. Acharya. Automated detection of covid-19 cases using deep neural networks with x-ray images. *Computers in biology and medicine*, 121:103792, 2020. 21

N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, et al. Technical report on the cleverhans v2. 1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2016. 78

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013. 11

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. 40

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014. 50

M. E. Peters, M. Neumann, R. L. Logan, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith. Knowledge enhanced contextual word representations. In *EMNLP*, 2019. 25

A. Poliak, J. Naradowsky, A. Haldar, R. Rudinger, and B. Van Durme. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-2023. URL https://www.aclweb.org/anthology/S18-2023. 2, 21

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018. 11, 17, 18, 44

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019. 39, 43, 47, 54

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html. 11, 18, 19, 39, 60

P. Rajpurkar, R. Jia, and P. Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2124. URL https://www.aclweb.org/anthology/P18-2124. 55

S. Ren, Y. Deng, K. He, and W. Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097, 2019. 24

M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 78

R. Riegel, A. G. Gray, F. P. S. Luus, N. Khan, N. Makondo, I. Y. Akhalwaya, H. Qian, R. Fagin, F. Barahona, U. Sharma, S. Ikbal, H. Karanam, S. Neelam, A. Likhyani, and S. K. Srivastava. Logical neural networks. *CoRR*, abs/2006.13155, 2020. URL https://arxiv.org/abs/2006.13155. 7

A. Roberts, C. Raffel, and N. Shazeer. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online, Nov. 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-main.437. 18

J. Rodrigues, R. Branco, J. Silva, and A. Branco. Reproduction and revival of the argument reasoning comprehension task. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5055–5064, 2020. 31

A. Rogers, O. Kovaleva, and A. Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020. 20

S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017. 66

C. Saedi, A. Branco, J. Rodrigues, and J. Silva. Wordnet embeddings. In *Proceedings of the third workshop on representation learning for NLP*, pages 122–131, 2018. 25

M. Salawa, A. Branco, R. Branco, J. António Rodrigues, and C. Saedi. Whom to learn from? graph- vs. text-based word embeddings. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1041–1051, Varna, Bulgaria, Sept. 2019. INCOMA Ltd. doi: 10.26615/978-954-452-056-4_120. URL https://www.aclweb.org/anthology/R19-1120. 25

J. Salazar, D. Liang, T. Q. Nguyen, and K. Kirchhoff. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.240. URL https://www.aclweb.org/anthology/2020.acl-main.240. 20

A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016. 78

M. Sap, R. Le Bras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith, and Y. Choi. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035, 2019. 25, 47

J. Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987. 78

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016. 16, 40

U. Shaham, Y. Yamada, and S. Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*, 2015. 78

P. Shaw, J. Uszkoreit, and A. Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, 2018. 16

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 20, 78

D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017. 20, 78

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html. XIII, 42

K. Sun and F. Nielsen. Lightlike neuromanifolds, occam's razor and deep learning. *arXiv preprint arXiv:1905.11027*, 2019. 20

M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328, 2017. 20, 78

M. Suteu and Y. Guo. Regularizing deep multi-task networks using orthogonal gradients. *arXiv preprint arXiv:1912.06844*, 2019. 66

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27:3104–3112, 2014. 11, 13

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. XIII, 23

A. Talmor, J. Herzig, N. Lourie, and J. Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://www.aclweb.org/anthology/N19-1421. XIII, 34, 36

A. Tamborrino, N. Pellicanò, B. Pannier, P. Voitot, and L. Naudin. Pre-training is (almost) all you need: An application to commonsense reasoning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3878–3887, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.357. URL https://www.aclweb.org/anthology/2020.acl-main.357. 19

S. E. Toulmin. *The Uses of Argument*. Cambridge University Press, 1958. 29

T. H. Trinh and Q. V. Le. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*, 2018. 41, 55

G. Valle-Pérez, C. Q. Camargo, and A. A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint arXiv:1805.08522*, 2018. 20

F. Van Harmelen, V. Lifschitz, and B. Porter. *Handbook of knowledge representation*. Elsevier, 2008. 6

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. XIII, 11, 12

J. Vig and Y. Belinkov. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4808. URL https://www.aclweb.org/anthology/W19-4808. 20, 78

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018. 18, 44

L. Wang, Z. Q. Lin, and A. Wong. Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *Scientific Reports*, 10(1):1–12, 2020. 21

R. Williams and J. Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2, 09 1998. doi: 10.1162/neco.1990.2.4.490. 11

R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. doi: 10.1162/neco.1989.1.2.270. 11

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020a. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6. 40

T. Wolf, Q. Lhoest, P. von Platen, Y. Jernite, M. Drame, J. Plu, J. Chaumond, C. Delangue, C. Ma, A. Thakur, S. Patil, J. Davison, T. L. Scao, V. Sanh, C. Xu, N. Patry, A. McMillan-Major, S. Brandeis, S. Gugger, F. Lagunas, L. Debut, M. Funtowicz, A. Moi, S. Rush, P. Schmidd, P. Cistac, V. Muštar, J. Boudier, and A. Tordjmann. Datasets. *GitHub. Note: https://github.com/huggingface/datasets*, 1, 2020b. 55

L. Wu, Z. Zhu, et al. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017. 20

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144. 11, 16

Z. Wu, T.-S. Nguyen, and D. Ong. Structured self-AttentionWeights encode semantics in sentiment analysis. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 255–264, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.blackboxnlp-1.24. URL https://www.aclweb.org/anthology/2020.blackboxnlp-1.24. 20, 78

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015. 20, 78

A. Yang, Q. Wang, J. Liu, K. Liu, Y. Lyu, H. Wu, Q. She, and S. Li. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2346–2357, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1226. URL https://www.aclweb.org/anthology/P19-1226. 25

Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Syx4wnEtvH. 40

T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33, 2020. 66

Y. Zang, F. Qi, C. Yang, Z. Liu, M. Zhang, Q. Liu, and M. Sun. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.540. URL https://www.aclweb.org/anthology/2020.acl-main.540. 24

J. R. Zech, M. A. Badgeley, M. Liu, A. B. Costa, J. J. Titano, and E. K. Oermann. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross-sectional study. *PLoS medicine*, 15(11):e1002683, 2018. 2, 20

B. Zhang, D. Xiong, J. Su, H. Duan, and M. Zhang. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530, 2016. 11

H. Zhang, D. Khashabi, Y. Song, and D. Roth. Transomcs: From linguistic graphs to commonsense knowledge. In C. Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on*

Artificial Intelligence, IJCAI-20, pages 4004–4010. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/554. URL https://doi.org/10.24963/ijcai.2020/554. Main track. 9

Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, 2019. 25

B. Zhou, D. Khashabi, Q. Ning, and D. Roth. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3363–3369, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1332. URL https://www.aclweb.org/anthology/D19-1332. 8

J. Zhou, Y. Cao, X. Wang, P. Li, and W. Xu. Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4:371–383, 2016. 11

X. Zhou, Y. Zhang, L. Cui, and D. Huang. Evaluating commonsense in pre-trained language models. In *AAAI*, pages 9733–9740, 2020. XV, 19, 60, 63

Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, 2015. doi: 10.1109/ICCV.2015.11. 40, 55

Appendix A

Training Hyper-Parameters

Task	Model	Batch Size	Learning Rate	Epochs
ARCT	RoBERTa-Large	16	1e-5	25
	GPT2-Medium	8	2e-3	18
	T5	8	2e-5	17
	BART-Large	16	2e-4	12
	COMET(BART)	8	1e-4	25
ARC	RoBERTa-Large	8	1e-4	16
	GPT2-Medium	4	1e-3	26
	T5	8	2e-5	12
	BART-Large	8	1e-4	27
	COMET(BART)	8	3e-5	22
PIQA	RoBERTa-Large	16	3e-3	28
	GPT2-Medium	8	1e-3	22
	T5	8	1e-5	9
	BART-Large	4	1e-3	19
	COMET(BART)	32	3e-4	16
CSQA	RoBERTa-Large	8	3e-4	13
	GPT2-Medium	8	1e-3	14
	T5	8	2e-5	5
	BART-Large	8	3e-4	18
	COMET(BART)	8	1e-4	14

Table A.1: Hyper-parameters found through a search used in each experiment.