Analyzing Natural Language Inference from a Rigorous Point of View

Felipe de Souza Salvatore

Tese apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Doutor em Ciências

Programa: Ciência da Computação

Orientador: Prof. Dr. Marcelo Finger

Coorientador: Prof. Dr. Roberto Hirata Jr.

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CAPES

São Paulo, Outubro de 2020

Analyzing Natural Language Inference from a Rigorous Point of View

Esta versão da tese contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 14/12/2020. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof. Dr. Marcelo Finger - IME-USP
- Profa. Dra. Valéria de Paiva - PUC-RJ
- Prof. Dr. Thiago Salgueiro Pardo - ICMC-USP
- Prof. Dr. Marcos Lopes - FFLCH-USP
- Prof. Dr. Denis Deratani Mauá - IME-USP

Acknowledgements

I would first like to acknowledge the academic and personal support of my supervisor Marcelo Finger. He was willing to take me on as a student, and he gave me tremendous freedom in pursuing my ideas while ensuring that I was developing productive research. I will be forever grateful for his help.

Roberto Hirata Jr is responsible for a great part of my education in machine learning. His comments and suggestions were very influential for all my research in the field of natural language processing.

I owe special thanks to my colleagues and collaborators in the statistics department Alexandre Patriota and Alexandre Simas. They both have help me to frame my research problem inside the statistical theory in a productive manner.

I am grateful to the faculty and staff from IME. They have provided a great environment in which to pursue my studies. I would also like to thank all my colleagues and friends from LIAMF; especially: Thiago Bueno, Thiago Lira, Fabiano Luz, Paula Moraes, Lucas Moura, and Sandro Preto. Thank you, everyone, for the warm companionship and the meaningful discussions.

I would like to thank my thesis committee: Marcos Lopes, Denis Mauá, Valéria de Paiva, and Thiago Pardo.

Eu gostaria de agraceder meus pais, meu irmão e minha família por todo o amor, apoio e compreensão na minha trajetória acadêmica.

Finally, I would like to thank the CAPES foundation (fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) for the financial support both in Brazil and abroad.

Resumo

Salvatore, F. **Analizando Inferência em Linguagem Natural de um Ponto de Vista Rigoroso**. 2020. 119 f. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2010.

Inferência em linguagem natural (*natural language inference* - NLI) é uma tarefa de classificação de texto baseada em determinar a relação de implicação entre um par de sentenças. Nessa tese estamos interessados em verificar se os modelos de *deep learning* usados em NLI satisfazem algumas propriedades lógicas. Aqui, focamos em duas propriedades: i) a capacidade de resolver problemas de dedução usando algumas formas lógicas (por exemplo, coordenação Booleana, quantificadores, descrição definida e operadores de contagem); e ii) a propriedade de ter a mesma conclusão partindo de premissas equivalentes. Para cada uma dessas propriedades, desenvolvemos um novo procedimento de avaliação. Para i) oferecemos um novo conjunto de dados sintético que podem ser usados tanto para a classificação quanto para a geração de inferência; e para ii) propomos um teste de hipóteses construído para representar as diferentes maneiras que a inclusão de sentenças com o mesmo significado pode afetar o treinamento de um modelo de aprendizado de máquina. Nossos resultados mostram que, embora os modelos de *deep learning* tenham um desempenho excelente na maioria dos problemas de NLI, eles ainda carecem de algumas importantes habilidades de inferência, como lidar com operadores de contagem, prever qual palavra pode formar uma implicação em um contexto específico e apresentar as mesmas deduções para duas entradas de texto que são diferentes mas possuem o mesmo significado. Isso indica que, apesar do grande poder de predição desses novos modelos, eles apresentam alguns viéses de inferência que não podem ser facilmente removidos. Futuras investigações precisam ser feitas para se entender o alcance desse víes. É possível que aumentando o tamanho da amostra de treinamento na fase de *fine-tuning* esse viés seja reduzido.

Palavras-chave: Processamento de Linguagem Natural, Classificação de Texto, Inferência em Linguagem Natural, Víes em Modelos de Aprendizado de Máquina.

Abstract

Salvatore, F. **Analyzing Natural Language Inference from a Rigorous Point of View**. 2020. 119 f. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2020.

Natural language inference (NLI) is the task of determining the entailment relationship between a pair of sentences. We are interested in the problem of verifying whether the deep learning models current used in NLI satisfy some logical properties. In this thesis, we focus on two properties: i) the capacity of solving deduction problems based on some specific logical forms (e.g., Boolean coordination, quantifiers, definite description, and counting operators); and ii) the property of having the same conclusion from equivalent premises. For each one of these properties we develop a new evaluation procedure. For i) we offer a new synthetic dataset that can be used both for inference perception and inference generation; and for ii) we propose a null hypothesis test constructed to represent the different manners that the inclusion of sentences with the same meaning can affect the training of a machine learning model. Our results show that although deep learning models have an outstanding performance on the majority of NLI datasets, they still lack some important inference skills such as dealing with counting operators, predicting which word can form an entailment given an specific context, and presenting the same deductions for two different text inputs with the same meaning. This indicates that despite the high prediction power of these new models, they do present some inference biases that cannot be easily removed. Future investigations are needed in order to understand the scope of this bias. It is possible that by increasing the training sample size in the fine-tuning phase, this bias can be reduced.

Keywords: Natural Language Processing, Text Classification, Natural Language Inference, Bias in Deep Learning.

Contents

List of Abbreviations

ALBERT	A Lite BERT.
BERT	Bidirectional Encoder Representations from Transformers.
BOW	Bag-of-Words.
CBOW	Continuous Bag-of-Words.
CD	Contradiction Detection.
FraCaS	Framework for Computational Semantics.
GLUE	General-Purpose Language Understanding.
GPT	Generative Pre-trained Transformer.
GRU	Gated Recurrent Unit.
IE	Invariance under Equivalence.
iff	if and only if.
IG	Inference Generation.
IID	Independent and Identically Distributed.
IP	Inference Perception.
LSTM	Long Short-Term Memory.
MI	Masked Inference.
MLM	Masked Language Modeling.
NHST	Null Hypothesis Significance Testing.
NLI	Natural Language Inference.
NLP	Natural Language Processing.
NLU	Natural Language Understanding.
NSP	Next Sentence Prediction.
QA	Question Answering.
RNN	Recurrent Neural Network.
RTE	Recognizing Textual Entailment.
ReLU	Rectified Linear Units.
RoBERTa	Robustly optimized BERT approach.
SNR	Signal-to-Noise Ratio.
SOP	Sentence-Order Prediction.

List of Symbols

$\boldsymbol{x}, \boldsymbol{y}, \ldots$	Vectors.
$\boldsymbol{X}, \boldsymbol{Y}, \ldots$	Matrices.
$[\boldsymbol{x}; \boldsymbol{y}]$	Vector concatenation.
$\boldsymbol{X}^\top$	Transpose of matrix $\boldsymbol{X}$.
$\boldsymbol{X} \odot \boldsymbol{Y}$	Element-wise (Hadamard) product of $\boldsymbol{X}$ and $\boldsymbol{Y}$.
$sigm(x)$	The logistic function, $\frac{1}{1+e^{-x}}$.
$tahn(x)$	The hyperbolic tangent, $\frac{e^{2x}-1}{e^{2x}+1}$.
$X, Y, \ldots$	Random variables.
$X \sim P$	The random variable X follows the distribution P.
$\mathbb{P}(X)$	Probability distribution over X.
$\hat{\mathbb{P}}(X)$	Estimated probability distribution over X.
$\mathbb{E}[X]$	Expectation of X.
$\mathbb{V}(X)$	Variance of X.
$X_n \rightsquigarrow X$	X_n converges to X in probability.
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2.
$I(\text{condition})$	Indication function. It returns 1 if the condition is true, 0 otherwise.
$CE(P, Q)$	Cross-entropy between the discrete distributions P and Q.
$\mathcal{V}$	Vocabulary.
$w_1, w_2, \ldots$	Words.
$hot(w)$	One-hot encoding of the word w.
$\lvert A \rvert$	Size of the set A.
$\mathcal{P}(A)$	Power set of the set A.
$A \backslash B$	Set subtraction, i.e., the set containing the elements of A that are not in B.
$\sup_{x \in A} f(x)$	The supremum (the least upper bound) of the set $\{f(x) : x \in A\}$.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Natural Language Inference

Among the *natural language processing* (NLP) tasks, the one centered on deduction is the task known as *natural language inference* (NLI). In this task, a system determines the logical relationship between a pair of sentences P and H (referred to as *premise* and *hypothesis*, respectively). It asserts either that P *entails* H, P and H *are in contradiction*, or P and H are *neutral* (logically independent).

NLI has a long history within the NLP field. The first instance of the NLI problem was initially formulated in the Framework for Computational Semantics (FraCaS) [The Fracas Consortium et al., 1996]. When constructing that framework, the creators of FraCaS also constructed a series of *question answering* (QA) modules, where a system should respond "yes", "no" or "don't know" to a query based on a logical/linguistic puzzle. For example,

$Q =$ `John and his colleagues went to a meeting. They hated it.`
`Did John's colleagues hate the meeting?`

$A =$ *Yes*,

where Q is a question and A is the right answer. Each module on the FraCaS dataset corresponds to a semantic competence (the use of generalized quantifiers, monotonicity, etc.). The idea behind this QA style dataset was to evaluate the proficiency of a NLP system regarding the different semantic competences.

After this initial effort, the introduction of a series of workshops called *the PASCAL recognizing textual entailment (RTE) challenges* helped to transform this initial idea into an active empirical research field [Bar-Haim et al., 2014]. During these contests the field acquired its present format: the use of two text inputs P and H, and three possible prediction classes. At this time the focus of analysis has shifted from formal puzzles to inference problems that can be found in everyday speech. It is worth mentioning that there are two sub-tasks associated with NLI: i) *entailment detection*, where a system should assert whether the pair (P, H) is an entailment or not; and ii) *contradiction detection* (CD), a similar binary classification task centered on the notion of contradiction [de Marneffe et al., 2008].

The modern era of the field started when large crowdsourced datasets were introduced. At this time, the name "NLI" was adopted and the term "RTE" was dropped.[1] The change in size, compared to the old datasets, is striking: the FraCas dataset is composed of 346 examples, some older datasets like RTE-6 [Bentivogli et al., 2009] and SICK [Marelli et al., 2014] have 16K and 10K examples, respectively. On the other hand, the datasets that have become benchmarks for the NLI community: *Stanford natural language inference corpus* (SNLI) [Bowman et al., 2015a] and *Multigenre NLI corpus* (MNLI) [Williams et al., 2018] are composed of 570K and 433K observations, respectively.

[1] It is possible to find recent uses of this term to denote the entailment detection task [Khot et al., 2018].

Recently, some authors formulate the NLI problem as a specific task under the more general problem of *natural language understanding* (NLU) [Wang et al., 2019, 2018]. It is not clear what the precise definition of NLU is. Without entering in the philosophical discussion of what constitutes "understanding" in the computer science domain (or if it is even possible), it is safe to say that NLU is just an aggregation of a variety of text classification tasks including QA, sentiment analysis and NLI. In this line, we say that a system has a "comprehensive text understanding capability" if such system performs with equal competence in all these different tasks.

It is important to highlight that there are two traditions in the NLI field: the *symbolic* and the *machine learning* tradition. The symbolic approach uses a process of translation the natural sentence pair (P, H) into a symbolic representation (e.g. a formula from a formal language [MacCartney and Manning, 2009], or a semantic graph [Kalouli et al., 2018]), after this translation is done, the transformed information is passed to an inference engine. The researchers from the machine learning tradition formulate the NLI task as a text classification problem: they transform the sentence pair (P, H) in a vector of covariates that captures some textual pattern, and after that they fit a statistical model to the textual data.

Since statistical models have dominated the NLI field, in this thesis, we focus exclusively on the machine learning tradition.

1.2 Motivation

1.2.1 Current State of the Art 1: The Success of the Machine Learning Models

The rapid progress in NLP has strongly influenced both NLI and NLU. After the introduction of the recent deep learning methods based on the transformer architecture [Devlin et al., 2019, Radford et al., 2018, Vaswani et al., 2017], the NLI benchmarks *are either solved or close of being solved*. Here, we say that a task is "solved" if such task can be performed by a computer with the same (or better) level of proficiency than humans.

The evolution of the NLI field in the last five years can be visualized in Figure 1.1. The dataset SNLI was created in 2015, and it took approximately *thirty-one months* for the NLI community to solve this task. The MNLI dataset, created in 2018, has two different test sets: i) *matched*, a test set with observations derived from the same sources as those in the training set; ii) *mismatched*, test examples from sources different from the ones used in training. At the time of writing, the matched module is solved and the best model accuracy for the mismatched module is 91.9%; the human performance is estimated in 92.8% (only 0.9% of difference).

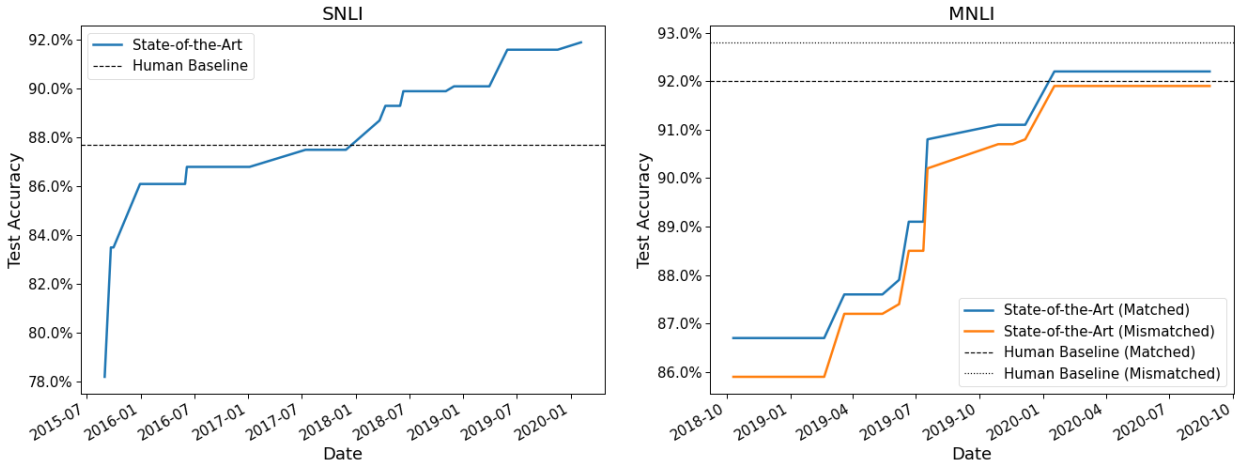

Figure 1.1: *State-of-the-art evolution for two NLI benchmarks. The figures display the best accuracy achieved on the SNLI and MNLI datasets by a machine learning model through time. All model results are compared with human performance. The data from SNLI was obtained from the work of Bowman et al. [2020] and the human performance was estimated by Glockner et al. [2018]. For the MNLI dataset, the data for both model and human performances is by Wang et al. [2020].*

The pace of solving NLP tasks is increasing. It took the community approximately *fifteen months* to solve the matched module on the MNLI dataset. The *general-purpose language understanding* (GLUE) dataset was an attempt of creating a robust NLU dataset [Wang et al., 2018]. The dataset was created in the beginning of 2018 and it was solved in approximately *thirteen months*. This fact have forced the authors to create a new and more challenging NLU dataset called SuperGLUE [Wang et al., 2019]. SuperGLUE was released in May, 2019, the T5 Team at Google has almost reached the human baseline at the score of 89.3% within *eight months*; the human baseline is 89.8% (only 0.5% of difference) [Wang et al., 2020]. In an interview for the *artificial intelligence index report 2019*, Sam Bowman whose group has developed GLUE and SuperGLUE state the following:

> "We know now how to solve an overwhelming majority of the sentence or paragraph level text classification benchmark datasets that we've been able to come up with to date. GLUE and SuperGLUE demonstrate this out nicely, and you can see similar trends across the field of NLP. I don't think we have been in a position even remotely like this before: We're solving hard, AI-oriented challenge tasks just about as fast as we can dream them up" Sam says "I want to emphasize, though, that we haven't solved language understanding yet in any satisfying way". [Perrault et al., 2019, p. 58]

It is evident that the technological improvement on the NLP field have affected the life expectancy of the available text classification tasks. One may argue that the NLI field is not completely solved since the mismatch module of the dataset MNLI is still unsolved, however because the gap between computer and human performance is so low, 0.9%, we believe that in a short period of time this task will be solved (the same is applicable to SuperGLUE).

1.2.2 Current State of the Art 2: The Failure of the Machine Learning Models

Although the notion of "solving a benchmark NLI task" is helpful to summarize the evolution of the field, it can be misleading. One recent finding regarding the machine learning models is that the process of data collection highly influences the quality of the inferences presented in a dataset.

Large datasets were made possible with the use of crowdsource platforms like the Amazon Mechanical Turk [Bowman et al., 2015a, Williams et al., 2018]. The annotation performed by a formal semanticist, like in RTE 1-3 [Giampiccolo et al., 2007], was replaced by the work of an average English speakers. One consequence of this process is the increasing number of undesirable text patterns (*annotation artifacts*) [Gururangan et al., 2018]. For example, there is a high correlation of occurrences of negative words (`no`, `nobody`, `never`, `nothing`) in the premise P of contradiction instances. Similarly, there is a high correlation of generic words (such as `animal`, `instrument`, `outdoors`, etc.) in entailment instances (also in the premise P). Hence, a simple model can achieve a score significantly higher than random guessing by using only P as input [Gururangan et al., 2018, Poliak et al., 2018].

In order to highlight how misleading is the success of the machine learning models in NLI, a new literature emerged focusing on the limitations of those models. This new literature uses the methodology of *adversarial evaluation*. Such methodology can be described as follows: train a machine learning model on a benchmark dataset (or a collection of datasets), and observe how well the model performs on a new test set with some linguistic features. The observations in the new test set are often called *adversarial examples*. In this case, the term "adversarial" refers to the phenomenon where the new observations are easily classified by humans but they force a machine learning model to make a mistake.

In this vein of work, Glockner et al. [2018] have developed a new test set based on different types of lexical knowledge (e.g., hypernymy and hyponymy relations). They shown that machine learning models trained on the datasets SNLI and MNLI perform substantially worse on their new lexical test set.

Nie et al. [2018] have created a new test set where the logical relations do not depend on lexical information alone (for example, it is possible to obtain a new contradiction observation (P, P') from

the pair (P, H), where P' is the result of swapping the subject and object in P). They shown that models trained on SNLI perform poorly on their new adversarial test sets.

Dasgupta et al. [2018] have constructed a test set based on word composition (for example, some entailment examples have the form: $P = X$ `is more cheerful than` Y, and $H = Y$ `is less cheerful than` X). They have observed that different models trained on the SNLI dataset perform badly on their adversarial test set, however they also noted that performance can be corrected when the models are trained with observations similar as the ones from the new test set.

Naik et al. [2018] offered three new adversarial test sets (they have called them "stress tests") based on different linguistic phenomena (e.g., antonymy relation, sentences containing numerals, etc.). After training different models on the MNLI dataset, they have observed that the models show a significant performance drop on their new test sets.

McCoy et al. [2019] have observed three "syntactical heuristics" presented on the benchmark datasets: lexical overlap heuristic (the logical relation can be guessed solely based on word overlap); subsequent heuristic (the logical relation can be guessed solely based on the fact that H is a subsequence of P); constituent heuristic (the logical relation can be guessed from the fact that H is a constituent of P). In order to understand how much a machine learning model rely on such heuristics, McCoy et al. [2019] have constructed an adversarial test set where those heuristic fail. They shown that different models trained on the MNLI dataset perform very poorly on their adversarial test set. Similar to Dasgupta et al. [2018], they also noted that it is possible to obtain good performances on the new test set when similar observations are introduced in the training stage.

Yanaka et al. [2019] have constructed a new test set based on monotonicity inference (this term includes different linguistic phenomena that can cause entailment, for example, the removal of modifiers - `I bought a movie ticket` entails `I bought a ticket`). After training different models on the SNLI and MNLI datasets, they have observed that the performance of the machine learning models on their adversarial test set was unsatisfactory. They have also noted that the performance of the models can be improved when monotonicity inference examples are added when training those models.

1.2.3 A Rigorous Point of View

There is an question in artificial intelligence as old as the field itself, let us call it the *cognition question*:

> Can machines think? [Turing, 1950]

Although the cognition question belongs to the fields of psychology and/or philosophy, a version of this question often appears in NLP and NLI. Because of the recent success of the deep learning models, the updated versions of the cognition question mention this specific model family. For example: Evans et al. [2018] ask themselves "can neural networks understand logical entailment?", and Piotrowski et al. [2019] try to address the question "can neural networks learn symbolic rewriting?". More recently, Bender and Koller [2020] offer an negative answer to the question "can neural networks understand language?" (the answer is somewhat similar to the one given by Searle [1980]).

It seems to us that the starting point for some recent works in NLI is a version of the cognitive question. Such version can be expressed as follows:

> Can neural networks understand X?

where X stands for:

- "inferences that require lexical and world knowledge" [Glockner et al., 2018];
- "lexical and compositional semantics" [Nie et al., 2018];

- "compositional information" [Dasgupta et al., 2018];
- "monotonic reasoning" [Yanaka et al., 2019].

Our basic assumption is that machine learning models (neural networks included) do not perform inference or any kind of reasoning, they perform *pattern recognition*. Hence, we are not interested in answering any version of the cognitive question. Alternatively, our research goal is more humble: we investigate whether machine learning models, when applied to the NLI task, satisfy some *desirable logical properties*. Such properties serve as minimum conditions to the inference task, i.e., the logical properties are a necessary but not sufficient condition for the inference competence.

Furthermore, we believe that formal logic can be an inspiration source for this point of view. The goal is not to reduce NLI to logic, but to use some ideas related to formal systems in order to aid the investigation of NLI models. To be more precise, let g be a NLI classifier trained on a benchmark dataset. For all sentence pairs (P, H), the classifier g defines the entailment relation $\models_g$ as follows:

$$\begin{aligned} P \models_g H &\iff g(P,H) = \textit{entailment}, \\ P, H \models_g \bot &\iff g(P,H) = \textit{contradiction}, \\ P \not\models_g H \text{ and } H \not\models_g P &\iff g(P,H) = \textit{neutral}, \end{aligned} \tag{1.1}$$

where "P, H" denotes the concatenation of sentences P and H (separated by the comma symbol), $\bot$ denotes an absurd statement (e.g., "$1 = 0$"), and $P \not\models_g H$ stands for "$P \not\models_g H$ and $H \not\models_g P$". From the logical point of view, it is expected that $\models_g$ satisfies some properties such as:

(Reflexive Entailment) $P \models_g P$.

(Anti-symmetric Entailment) not always, $P \models_g H \Rightarrow H \models_g P$.

(Symmetric Contradiction) $P, H \models_g \bot \Rightarrow H, P \models_g \bot$.

(Symmetric Neutral) $P \not\models_g H \Rightarrow H \not\models_g P$.

(Monotonic Entailment) $P \models_g H \Rightarrow P, P' \models_g H$.

(Monotonic Contradiction) $P, H \models_g \bot \Rightarrow P, H, P' \models_g \bot$.

(Explosion) $P, H \models_g \bot \Rightarrow P, H \models_g P'$.

These are just a few illustrative examples. Since NLI and formal inference are governed by different rules, it is the task of the researcher to decide which properties inspired by formal logic are suitable for NLI.

1.3 Objectives

Based on these considerations, we can describe the central workflow in this thesis: we start from properties present in logical systems, we carefully consider which ones are applicable in the natural language context, and then we analyse whether the NLI models satisfy these properties or not.

As the guiding objectives of our work, we have selected two main properties to be analyzed:

- The property of correctly performing deduction based on logical connectives (structural inference).
- The property of having the same deductions from equivalent text inputs (invariance under equivalence).

1.4 Contributions

After investigating the two properties mentioned above, we have discovered five new facts about NLI models:

1) *Counting operators are the most challenging logical connectives to NLI models.* We have created a new CD synthetic dataset divided into different modules. Each module is based on a type of logical connective. We have observed that the module associated with counting inference is the most demanding module for all NLI models (Chapter 3).

2) *When the entailment relation is based on structural information, cross-lingual transfer learning can be successfully applied to NLI.* We have shown that for entailment examples based on structural features (the co-occurrence of words in P and H, the position of some connectives in the text input, and the number of occurrences of some specific words) it is possible to use a pre-trained model in one language, and obtain reasonable results in a NLI task defined in other language (Chapter 3).

3) *The task of predicting which word can form an entailment, given a context, is significantly harder than the usual NLI task.* It is possible to formulate a harder inference task: we ask the NLI model to complete a sentence in order to create an entailment observation. When we formulate this kind of "inference generation" task we can observe a significant drop in performance of the current NLI models (Chapter 3).

4) *Current deep learning models show two different inference outputs for sentences with the same meaning.* We propose a new test to measure the invariance under equivalence property. After applying such test using both the SNLI and MNLI datasets, we have observed that the deep learning models fail the test in the vast majority of cases (Chapter 4).

5) *Some NLI models are clearly more robust than others.* By measuring each model's performance on the test set when equivalent examples are present in training, we have observed that while some deep learning models are quite robust, others perform worse than the baseline (Chapter 4).

These contributions were presented as the following papers:

- 1) and 2) are the results from a paper presented in the *Workshop on Deep Learning Approaches for Low-Resource NLP* at the conference *Empirical Methods in Natural Language Processing* (EMNLP) in 2019 [Salvatore et al., 2019a].
- 3) is the contribution from a paper presented in the *Workshop on Neural-Symbolic Learning and Reasoning* at the conference *International Joint Conferences on Artificial Intelligence* (IJCAI) in 2019 [Salvatore et al., 2019b].
- 4) and 5) are the main results of a paper submitted to a relevant NLP journal, currently under review.

1.5 Organization

The thesis is organized as follows: in Chapter 2 we define all the theoretical background; in Chapter 3 we present the analyses centered on logical connectives; in Chapter 4 we present the investigation based on equivalent transformations; and, finally, in Chapter 5 we address open issues and future steps.

Chapter 2

Theoretical Background

There is no unified theory behind the NLP field. In different textbooks on the subject [Eisenstein, 2019, Goldberg, 2016, Manning and Schütze, 1999], the theory is presented as an amalgamation of a variety of fields including machine learning, information theory, probability theory and numerical optimization. In this chapter, we present a brief description of the theoretical tools employed throughout this thesis. We use the problem of text classification as a framework to unify all the concepts, definitions and models presented in this chapter.

2.1 Text Classification

2.1.1 Basic Formulation

In many practical application, we want to assign classes or categories to text according to its content. One of the most famous example is *spam detection*: given a new email in the inbox, it is a spam of a meaningful email?

Since the categorization is based on the text content, many text classification tasks rely on *subjective judgments*. Hence, when we ask "does the text T belongs to the class c?", we are implicitly asking "does the majority of speakers categorize the text T as c?".

This distinction is especially important for the NLI task. Differently from formal logic where we offer a precise definition of the terms *entailment*, *contradiction*, and *neutral*; in NLI these categories are more fluid. Thus, one research group can judge the logical categorization of other research group as erroneous [Kalouli et al., 2017]. The subjective grounding of the NLI task also explains why the human performance for the benchmark datasets is lower than 100% (Figure 1.1).

2.1.2 Technical Formulation

The task of *text classification* (*text categorization*) consists in categorizing *texts* (*documents*) into different classes. We can formulate the categorization process as a *statistical classification* task (a *supervised learning* task). The text data $\mathcal{D}$ is of the form:

$$\mathcal{D} = \{(T_1, Y_1), \ldots, (T_n, Y_n)\}, \tag{2.1}$$

where T_i is a text input (either a word, a sentence, a collection of sentences, or a document) and Y_i is a discrete random variable that takes values in some finite set $\mathcal{Y}$. The NLI task is a version of text classification. In this task, the input text is a tuple of sentences $T_i = (P_i, H_i)$ and the target Y_i is either *entailment*, *contradiction* or *neutral*. For example, one NLI observation is of the form:

$P =$ `A man with a gray shirt holds a young infant in his hands.`

$H =$ `A man is wearing a shirt.`

$Y =$ *entailment*.

Different methods are used to transform a text input into a vector of *covariates* (*features*). We call such methods *representation functions*.

2.1.3 Bag-of-Words

The most simple form to represent a text into a collection of measurements is by using word frequency. For a text data $\mathcal{D}$, by *corpus* we mean the set of all text inputs presented in $\mathcal{D}$. Let $\mathcal{V} = (w_1, \ldots, w_V)$ be the sequence of all unique word types appearing in the corpus ordered by their frequency (we call this sequence the *vocabulary*). The *bag-of-words* (BOW) representation of the text T_i is defined as:

$$X_i = (X_{i,1}, \ldots, X_{i,V}), \tag{2.2}$$

where $X_{i,j}$ captures some frequency information about the word w_j in the text T_i. Usually we choose one of the following types of representation:

- $X_{i,j} \in \{0, 1\}$ indicates the absence or presence of the word w_j in the text T_i.
- $X_{i,j} \in \mathbb{N}$ indicates the number of occurrences of the word w_j in the text T_i.
- $X_{i,j} \in \mathbb{R}$ indicates the frequency of the word w_j in the text T_i rescaled by how often this word appear in all documents (this metric is know as *the term frequency - inverse document frequency*; tf-idf).

Since the vocabulary size can be large, this process can generate a substantial amount of features. Hence, it is normal to combine the BOW method with dimensionality reduction techniques (e.g., principal component analysis) in order to simplify the input data.

2.1.4 Heuristics Based Representations

Instead of using a general approach like BOW, we can apply a heuristic (i.e., a strategy based on domain knowledge) to construct an useful representation function. This sort of strategy is known in the machine learning literature as "feature engineering".

For example, sentiment analysis is the task of classifying text according to subjective judgements. Normally such judgments are expressed as three sentiments related to a text: *positive*, *negative* and *neutral*. For a specific version of this task, say sentiment analysis focused on political discourse, we can simplify the feature creation process by defining only a couple variables related to the occurrence of positive and negative words in the text. A concrete example can illustrate this point. The data by Eight [2016] is a collection of tweets about the early August republican party debate predating the 2016 United States presidential election. In this data we can find observations such as:

> $T =$ `Before the #GOPDebate, 14 focus groupers said they had favorable view of Trump.`
>
> $Y =$ *positive*.

And

> $T =$ `Fox News trying to convince us young Black Americans are more worried about ISIS than police terrorism.`
>
> $Y =$ *negative*.

Let $Pos = \{\texttt{good}, \texttt{favorable}, \ldots\}$ and $Neg = \{\texttt{evil}, \texttt{terrorism}, \ldots\}$ be previous selected sets of positive and negative words, respectively. If our domain knowledge indicates that usually the tweet sentiment is associated with the words on these sets, we can define simple covariates like:

$$X_{i,j} = \frac{|vocab(T_i) \cap Sen|}{|vocab(T_i)|}, \tag{2.3}$$

where either $j = 1$ and $Sen = Pos$, or $j = 2$ and $Sen = Neg$, and $vocab$ is a function associating a text with the set of words concurring on it. If we have an adequate prior knowledge on the subject, the engineered features can be more useful than the ones created by the BOW method.

2.1.5 Classifiers and Performance Metrics

After the representation function is selected, the text data in (2.1) is transformed in the usual set of input-output pairs from a supervised learning task:

$$\mathcal{D} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}, \tag{2.4}$$

where $X_i = (X_{i,1}, \ldots, X_{i,d}) \in \mathcal{X} \subset \mathbb{R}^d$, for some $d \in \mathbb{N}$. At this point, the text classification problem becomes a machine learning problem. Thus, we can use a variety of well-known models from this field to solve the initial task: nearest neighbors, logistic regression, random forest, neural networks, etc. In this context, a *text classifier* is just a machine learning model combined with a representation function.

We asses the generalization performance of those classifiers using the usual metrics from the field. Given a classifier g, the *true accuracy* of g is defined as:

$$\begin{aligned} acc(g) &= \mathbb{E}[I(g(X) = Y)] \\ &= \mathbb{P}(\{g(X) = Y\}), \end{aligned} \tag{2.5}$$

where I is the indication function. In other words, the true accuracy is the probability of the model correctly predicting a new observation. Since we do not have access to the join population distribution for X and Y, we estimate the true accuracy using an independent test data $\mathcal{D}_{Te} = \{(X_i, Y_i) : i = 1, \ldots, m\}$. The estimate is called the *test accuracy*:

$$\widehat{acc}(g) = \frac{1}{m} \sum_{i=1}^{m} I(g(X_i) = Y_i). \tag{2.6}$$

2.2 Word Embeddings

One approach that has been achieving large success in text and image classification is known as *deep learning* (*feature learning*, *representation learning*) [Bengio et al., 2013]. In this perspective both the representation function and the classifier are parametric models whose parameters are estimated in a classification task. We introduce this approach by detailing one of its first contributions to NLP: word embedding, i.e., the process of representing a word as a vector in a multidimensional space such that an aspect of the word meaning is captured by the representation.

2.2.1 Probabilistic Language Modeling

Before we comment on the word embedding, it is worthwhile to describe how such representation can be obtained. For this reason, we describe one central NLP task: *language modeling*. The language modeling task is focuses on estimating the probability of a sentence appearing on a language; more precisely, in this task we are interested to estimate the probability:

$$\mathbb{P}(w_1, \ldots, w_n), \tag{2.7}$$

for every sequence of words $w_1, \ldots, w_n$ obtained from a vocabulary $\mathcal{V}$. Since this is a very demanding task, language models made some simplifying assumptions. Usually, it is assumed the kth order

Markov property for the language generation process, i.e., the next word in a sequence depends only on the last k words. Thus, for every $i \in \mathbb{N}$ we assume that

$$\mathbb{P}(w_i \,|\, w_1, \ldots, w_{i-1}) \approx \mathbb{P}(w_i \,|\, w_{i-k}, \ldots, w_{i-1}). \tag{2.8}$$

Using this assumption and the chain-rule of probability, (2.7) can be approximated as

$$\begin{aligned}\mathbb{P}(w_1, \ldots, w_n) &= \mathbb{P}(w_1)\,\mathbb{P}(w_2 \,|\, w_1)\,\mathbb{P}(w_3 \,|\, w_1, w_2) \ldots \mathbb{P}(w_n \,|\, w_1, \ldots, w_{n-1}) \\ &\approx \prod_{i=1}^{n} \mathbb{P}(w_i \,|\, w_{i-k}, \ldots, w_{i-1}),\end{aligned} \tag{2.9}$$

where $w_{1-k}, \ldots, w_0$ are special padding tokens. For example, by setting $k = 1$, the probability of the sentence `let's go dancing` can be approximated as

$$\mathbb{P}(\texttt{let's go dancing}) \approx \mathbb{P}(\texttt{let's} \,|\, \texttt{[S]})\,\mathbb{P}(\texttt{go} \,|\, \texttt{let's})\,\mathbb{P}(\texttt{dancing} \,|\, \texttt{go}), \tag{2.10}$$

where `[S]` is the start-of-sequence token.[1] With these simplifications, the problem of correctly estimating $\mathbb{P}(w_i \,|\, w_{i-k}, \ldots, w_{i-1})$ becomes the main focus of the language modeling task.

Although there is a long history of different techniques used to estimate these conditional probabilities [Chen and Goodman, 1999], we concentrate on the recent approach where the language modeling task is treated as a machine learning problem. Given $k \in \mathbb{N}$, and a collection of documents - a corpus with the associated vocabulary $\mathcal{V}$ - we can obtain multiple k sequences of consecutive words from the corpus and create a text classification dataset as in (2.1) where each observation (T_i, Y_i) is of the form

$$(T_i, Y_i) = ((w_1, \ldots, w_{k-1}), w_k). \tag{2.11}$$

In essence, we are trying to classify a text compose of $k-1$ words according to the next word w_k. The objective of this classification task is to obtain a model of the probability distribution $\mathbb{P}(W_k \,|\, w_1, \ldots, w_{k-1})$ for each sequence $w_1, \ldots, w_{k-1} \in \mathcal{V}$.[2] In other words, a language model is the function $g : \mathcal{V}^{k-1} \to [0,1]^{|\mathcal{V}|}$ such that

$$\begin{aligned} g(w_1, \ldots, w_{k-1}) &= \hat{\boldsymbol{y}} \\ \hat{\boldsymbol{y}}_i &= \hat{\mathbb{P}}(w \,|\, w_1, \ldots, w_{k-1}),\end{aligned} \tag{2.12}$$

where $i \in \{1, \ldots, |\mathcal{V}|\}$ and w is the ith word in $\mathcal{V}$.

On a side note, since we are using a raw text to construct a text classification dataset, there is no need to manually assign a label Y_i to each observation from the data. Hence, many authors call the language modeling task an "unsupervised" or a "self-supervised" task.

2.2.2 Neural Language Models

In the current stage of the NLP field, there are many possibilities of using a neural network to construct a language model – the result of this construction is called a *neural language model*. Here, we present the model described by Bengio et al. [2003]. It is a very simple solution to the language modeling problem that can help us understand how to obtain word representations as vectors in a multidimensional space.

[1] A token is a sequence of characters that comprises a semantic unit. We can use words or sub-words (like morphemes) as tokens. Often, we add special tokens to the vocabulary in order to make some text information explicit.

[2] W_k denotes a random variable for a word in the k position. Hence, $\mathbb{P}(w \,|\, w_1, \ldots, w_{k-1})$ is just the abbreviation of $\mathbb{P}(W_k = w \,|\, W_1 = w_1, \ldots, W_{k-1} = w_{k-1})$.)

Before describing the model, some notation is needed. Given a vocabulary $\mathcal{V}$ of size V, let w be the ith word on $\mathcal{V}$. We can represent w as a vector $hot(w) \in \{0,1\}^V$. This vector is called the *one-hot vector* (*one-hot encoding*) associated with the word w. It indicates the position of w in $\mathcal{V}$, i.e., for all $j \in \{1,\ldots,V\}$, $hot(w)_j = 1$ iff $j = i$. Given a matrix $\boldsymbol{E} \in \mathbb{R}^{d\times V}$, let $v_{\boldsymbol{E}}(w) = \boldsymbol{E}hot(w) \in \mathbb{R}^d$. By the construction of $hot(w)$, $v_{\boldsymbol{E}}(w)$ is the ith column in $\boldsymbol{E}$. The mapping $v_{\boldsymbol{E}} : \mathcal{V} \to \mathbb{R}^d$ is referred to as an *embedding*, and $v_{\boldsymbol{E}}(w)$ an *embedding vector* (*word embedding*).

Let $((w_1,\ldots,w_{k-1}),w_k)$ be an observation as defined in (2.11). A simple neural language model is defined by the following set of equations

$$\begin{aligned}\hat{\boldsymbol{y}} &= \text{softmax}(\boldsymbol{W_2 h} + \boldsymbol{b_2})\\ \boldsymbol{h} &= g(\boldsymbol{W_1 x} + \boldsymbol{b_1})\\ \boldsymbol{x} &= [v_{\boldsymbol{E}}(w_1);\ldots;v_{\boldsymbol{E}}(w_{k-1})],\end{aligned} \tag{2.13}$$

where

- $\boldsymbol{\theta} = (\boldsymbol{E}, \boldsymbol{W_1}, \boldsymbol{b_1}, \boldsymbol{W_2}, \boldsymbol{b_2})$ is the set of model's parameters such that $\boldsymbol{E} \in \mathbb{R}^{d1\times V}, \boldsymbol{W_1} \in \mathbb{R}^{d_2\times(k-1)d_1}, \boldsymbol{b_1} \in \mathbb{R}^{d_2}, \boldsymbol{W_2} \in \mathbb{R}^{V\times d_2}, \boldsymbol{b_2} \in \mathbb{R}^V$. The dimensions $d_1, d_2 \in \mathbb{N}$ are hyperparameters for this model.
- The context vector $\boldsymbol{x} = [v_{\boldsymbol{E}}(w_1);\ldots;v_{\boldsymbol{E}}(w_{k-1})] \in \mathbb{R}^{(k-1)d_1}$ is the result of concatenating the word vectors $v_{\boldsymbol{E}}(w_1),\ldots,v_{\boldsymbol{E}}(w_{k-1}) \in \mathbb{R}^{d_1}$.
- $g : \mathbb{R}^{d_2} \to \mathbb{R}^{d_2}$ is a non-linear function (e.g., the logistic function $sigm$).
- softmax : $\mathbb{R}^V \to [0,1]^V$ is the mapping that forces the values in $\hat{\boldsymbol{y}}$ to be positive and sum to one, i.e., for $\boldsymbol{z} \in \mathbb{R}^V$:
$$\text{softmax}(\boldsymbol{z})_j = \frac{e^{\boldsymbol{z}_j}}{\sum_{l=1}^V e^{\boldsymbol{z}_l}}.$$
- $\hat{\boldsymbol{y}} \in [0,1]^V$ stands for the estimated probability distribution given the context words $w_1,\ldots,w_{k-1}$, i.e., for a word w in the ith position of the vocabulary, $\hat{\boldsymbol{y}}_i = \hat{\mathbb{P}}_{\boldsymbol{\theta}}(w \mid w_1,\ldots,w_{k-1})$.

We evaluate the model by measuring the dissimilarity between the estimation and the observed word distribution. Usually, we use the *categorical cross-entropy loss* for measurement:

$$CE(hot(w_k),\hat{\boldsymbol{y}}) = -\sum_{i=1}^{V} hot(w_k)_i \log(\hat{\boldsymbol{y}}_i). \tag{2.14}$$

The model is trained using the cross-entropy loss for all observations in the dataset. What is important for our discussion is a byproduct of this model.

Similar to the function $v_{\boldsymbol{E}}$, we can define the mapping $v_{\boldsymbol{W_2}} : \mathcal{V} \to \mathbb{R}^{d_2}$ such that $v_{\boldsymbol{W_2}}(w) = hot(w)\boldsymbol{W_2}$. Since the word representation in $\boldsymbol{W_2}$ is affected by the context vector $\boldsymbol{h}$, one observed side effect is that words that have a similar context have a similar representation in $\boldsymbol{W_2}$ (vector correspondence is often measured using the cosine similarity). For example, words like `king` and `queen` tend to have a more similar word representation compared to the words `king` and `dog`. However, it should be clear that the complete meaning of a word is not captured by this method. For example, since words that are opposite of each others (e.g., `good` and `bad`, `hot` and `cold`) appear on similar contexts, their respective word vectors tend to be similar.

Both $v_{\boldsymbol{E}}$ and $v_{\boldsymbol{W_2}}$ are word embeddings. After the neural language model is trained we can combine these functions into a single mapping (by taking the mean, or just by vector concatenation) and use the resulting function as a word representation for other NLP tasks.

2.2.3 Continuous Bag-of-Words Representation

What we have described is just one possibility of using a language model to obtain a word representation function. The NLP field have developed a family of similar algorithms to obtain word embeddings from non-annotated data. Popular examples are given by Joulin et al. [2017], Mikolov et al. [2013], Pennington et al. [2014].

Going back to the problem of text classification, given an embedding $v : \mathcal{V} \to \mathbb{R}^d$, we can define a representation function as follows: let (T_i, Y_i) be an observation of (2.1) such that T_i is a text of length k_i, i.e., $T_i = (w_{i,1}, \ldots, w_{i,k_i})$ for $w_{i,1}, \ldots, w_{i,k_i} \in \mathcal{V}$. We define the feature vector associated to this text as

$$X_i = \frac{1}{k_i} \sum_{j=1}^{k_i} v(w_{i,j}). \tag{2.15}$$

This approach is called *continuous bag-of-words* (CBOW) representation.[3] It is similar to the BOW method in the sense that it generates a representation function that can be applied on any task without the need of domain knowledge. There are two main advantages of the CBOW representation compared to the BOW method: the former generates less features (d is usually significantly smaller than $V = |\mathcal{V}|$); and since the CBOW representation is usually obtained using a large amount of raw data, this method can offer an useful representation for words not appearing in the training data. For example, if the word `king` appears on the training data and the word `queen` is absent, a new observation containing the word `queen` will present similar features to the ones in the training dataset related to the word `king`.

2.3 Sequence Modeling

The representations presented in (2.2), (2.3), and (2.15) are different forms of transforming a text input into a feature vector. One element in common among all representations presented so far is the lack of any order information. In this section, we present one family of deep learning models widely used in NLP to capture regularities in sequential data.

2.3.1 Recurrent Models

Recurrent neural network (RNN) is a family of non-linear autoregressive models. It is based on the idea that the computation of an input $\boldsymbol{x}_t \in \mathbb{R}^d$ depend on past values $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1} \in \mathbb{R}^d$. Using the neural network terminology, a function applied to previous inputs is called a *recurrent* computation and the product of that function is a *hidden state* (*hidden layer*) that *encodes* past inputs.

An example is in order. Consider the language modeling problem where we use the word sequence $(w_1, \ldots, w_{k-1})$ to predict the next word w_k. Instead of concatenating the $k-1$ word vectors into a single input, we can use a RNN model to generate an output for each sub-sequence. For example, we can modify the neural network (2.13) in the following way:

$$\begin{aligned} \hat{\boldsymbol{y}}_t &= \text{softmax}(\boldsymbol{W_2 h}_t + \boldsymbol{b_2}) \\ \boldsymbol{h}_t &= g(\boldsymbol{W_1 x}_t + \boldsymbol{U h}_{t-1} + \boldsymbol{b_1}) \\ \boldsymbol{x}_t &= \boldsymbol{E} hot(w_t), \end{aligned} \tag{2.16}$$

where

- $t \in \{1, \ldots, k-1\}$.

[3] Here, we follow Goldberg [2016, p. 93] in calling the averaging of embedding vectors a "CBOW representation". The name CBOW is also used to define one specific neural architecture presented by Mikolov et al. [2013].

- The models parameters $\boldsymbol{\theta} = (\boldsymbol{E}, \boldsymbol{W_1}, \boldsymbol{b_1}, \boldsymbol{U}, \boldsymbol{W_2}, \boldsymbol{b_2})$ are defined as in (2.13) with the following additions: $\boldsymbol{W_1} \in \mathbb{R}^{d_2 \times d_1}$ and $\boldsymbol{U} \in \mathbb{R}^{d_2 \times d_3}$, for $d_3 \in \mathbb{N}$.
- $\boldsymbol{h}_0 \in \mathbb{R}^{d_3}$, the initial hidden state, is the zero vector.

One of the main advantages of using a recurrent model is that we are estimating a conditional probability distribution for different context sizes, i.e., given $t \in \{1, \ldots, k-1\}$ and the subsequence of words $(w_1, \ldots, w_t)$, $\hat{\boldsymbol{y}}_t = \hat{\mathbb{P}}_{\boldsymbol{\theta}}(W_{t+1} \mid w_1, \ldots, w_t)$. This change allow us, at least in theory, to choose large values for k in order to model long-range dependencies.

As is usual in neural network construction, we can define a variety of complex models based on the idea of recurrence. The model (2.16) can be augmented by adding more hidden states or by allowing information related to the future words within the sequence. This latter augmentation is called *bidirectional*-RNN. For example, we can modify (2.16) by adding a hidden state $\boldsymbol{s}_t$ formed by the future inputs $\boldsymbol{s}_{t+1}, \ldots, \boldsymbol{s}_{k-1}$:

$$\begin{aligned} \hat{\boldsymbol{y}}_t &= \text{softmax}(\boldsymbol{W_3}[\boldsymbol{h}_t; \boldsymbol{s}_t] + \boldsymbol{b_3}) \\ \boldsymbol{s}_t &= g_2(\boldsymbol{W_2}\boldsymbol{x}_t + \boldsymbol{U_2}\boldsymbol{s}_{t+1} + \boldsymbol{b_2}) \\ \boldsymbol{h}_t &= g_1(\boldsymbol{W_1}\boldsymbol{x}_t + \boldsymbol{U_1}\boldsymbol{h}_{t-1} + \boldsymbol{b_1}) \\ \boldsymbol{x}_t &= \boldsymbol{E}hot(w_t), \end{aligned} \tag{2.17}$$

where both $\boldsymbol{h}_0$ and $\boldsymbol{s}_k$ are the zero vector, and both g_1 and g_2 are non-linear functions.[4] It is not theoretically clear the benefits of using a complex model compared to simple RNNs like (2.16) or (2.17), but it is reported that architectures with several layers obtain better results on different NLP tasks [Goldberg, 2016, p. 172].

The RNN model is trained using the same cross entropy loss (2.14) – in this case, we compare the dissimilarity between $hot(w_k)$ and $\hat{\boldsymbol{y}}_{k-1}$. It should be clear that we are using the language modeling task only as an example. The transition to the text classification task is straightforward: for an observation (T_i, Y_i) of (2.1) such that $T_i = (w_{i,1}, \ldots, w_{i,k_i})$ we obtain from models like (2.16) or (2.17) the vector $\hat{\boldsymbol{y}}_{k_i} = \hat{\mathbb{P}}_{\boldsymbol{\theta}}(Y \mid w_{i,1}, \ldots, w_{i,k_i}) \in [0,1]^{|\mathcal{Y}|}$. The final classification is obtained by taking $\arg\max_y \hat{\mathbb{P}}_{\boldsymbol{\theta}}(Y = y \mid w_{i,1}, \ldots, w_{i,k_i})$. Thus, in the text classification task, we can use the same cross-entropy loss to compare the model's estimated distribution with the dummy variable associated with Y_i.

When training recurrent neural networks, we obtain the models gradient with respect to the loss function using the backpropagation algorithm (the application of this automatic differentiation algorithm to a RNN model is often called *backpropagation through time*). Although this kind of deep learning model is very useful, it presents a severe flaw. When computing the gradients there is a lot of repeated matrix multiplication using the recurrent weight matrix (for example, in (2.16), the matrix $\boldsymbol{U}$). Depending on some configurations of this matrix *the gradients may vanish or explode exponentially with respect to* k. In order to correct this problem, specialized architectures were designed to handing long-term dependencies.

2.3.2 Gated Recurrent Unit

A new architecture called *gated recurrent unit* (GRU) was proposed by Chung et al. [2014]. This model was constructed so that each hidden state $\boldsymbol{h}_t$ can adaptively capture dependencies of different steps. In order to give a concrete formulation of this model, let us continue to use the language modeling task. Again, for $t \in \{1, \ldots, k-1\}$, the GRU model is composed of the following equations:

[4] For the sake of brevity, when a model is composed of multiple parameters matrices, we omit the full detailed description.

$$
\begin{aligned}
\hat{\boldsymbol{y}}_t &= \text{softmax}(\boldsymbol{W_2 h}_t + \boldsymbol{b_2}) \\
\boldsymbol{h}_t &= \boldsymbol{u}_t \odot \widetilde{\boldsymbol{h}}_t + (1 - \boldsymbol{u}_t) \odot \boldsymbol{h}_{t-1} \\
\widetilde{\boldsymbol{h}}_t &= tahn(\boldsymbol{W_1 x}_t + \boldsymbol{U}(\boldsymbol{h}_t \odot \boldsymbol{r}_t) + \boldsymbol{b_1}) \\
\boldsymbol{r}_t &= sigm(\boldsymbol{W_r x}_t + \boldsymbol{U_r h}_{t-1} + \boldsymbol{b_r}) \\
\boldsymbol{u}_t &= sigm(\boldsymbol{W_u x}_t + \boldsymbol{U_u h}_{t-1} + \boldsymbol{b_u}) \\
\boldsymbol{x}_t &= \boldsymbol{E} hot(w_t),
\end{aligned} \tag{2.18}
$$

where *sigm* is the logistic function, *tahn* is the hyperbolic tangent function, $\odot$ is the Hadamard product (the element-wise product), and $\boldsymbol{h}_0$ is the zero vector. There is some intuition behind the construction of this model: $\boldsymbol{r}_t$ is a vector with values in $[0, 1]$ called a *reset gate*, i.e., a vector that at each entry outputs the probability of resetting the corresponding entry in the previous hidden state $\boldsymbol{h}_{t-1}$. Together with $\boldsymbol{r}_t$ we define an *update gate*, $\boldsymbol{u}_t$. It is also a vector with values in $[0, 1]$. Intuitively, we can say that this vector decides how much information on each dimension of the candidate update $\widetilde{\boldsymbol{h}}_t$ we use to form $\boldsymbol{h}_t$. Both $\boldsymbol{r}_t$ and $\boldsymbol{u}_t$ are defined by $\boldsymbol{h}_{t-1}$ and $\boldsymbol{x}_t$. The new hidden state $\boldsymbol{h}_t$ combines the candidate hidden state $\widetilde{\boldsymbol{h}}_t$ with the past hidden state $\boldsymbol{h}_{t-1}$ using both $\boldsymbol{r}_t$ and $\boldsymbol{u}_t$ to adaptively copy and forget information.

2.3.3 Long Short-Term Memory

Long short-term memory (LSTM) is one of the most applied versions of the RNN family of models [Hochreiter and Schmidhuber, 1997]. Historically it was developed before the GRU model, but conceptually we can think in the LSTM as an expansion of the model presented in (2.18). First, let us define the LSTM model using the following equations:

$$
\begin{aligned}
\hat{\boldsymbol{y}}_t &= \text{softmax}(\boldsymbol{W_2 h}_t + \boldsymbol{b_2}) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot tanh(\boldsymbol{c}_t) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \widetilde{\boldsymbol{c}}_t \\
\widetilde{\boldsymbol{c}}_t &= tahn(\boldsymbol{W_1 x}_t + \boldsymbol{U h}_{t-1} + \boldsymbol{b_1}) \\
\boldsymbol{f}_t &= sigm(\boldsymbol{W_f x}_t + \boldsymbol{U_f h}_{t-1} + \boldsymbol{b_f}) \\
\boldsymbol{i}_t &= sigm(\boldsymbol{W_i x}_t + \boldsymbol{U_i h}_{t-1} + \boldsymbol{b_i}) \\
\boldsymbol{o}_t &= sigm(\boldsymbol{W_o x}_t + \boldsymbol{U_o h}_{t-1} + \boldsymbol{b_o}) \\
\boldsymbol{x}_t &= \boldsymbol{E} hot(w_t),
\end{aligned} \tag{2.19}
$$

where both $\boldsymbol{c}_0$ and $\boldsymbol{h}_0$ are the zero vector. The idea behind the LSTM model is to combined different recurrent computation (called *gates*): the *forget gate* $\boldsymbol{f}_t$ controls how much informative is discarded, the *input gate*, $\boldsymbol{i}_t$ controls how much information is updated, and the *output gate* $\boldsymbol{o}_t$ controls how munch each component is outputted. A candidate cell, $\widetilde{\boldsymbol{c}}_t$ is formed as a normal RNN hidden state, and a new cell $\boldsymbol{c}_t$ is formed by forgetting some information of the previous cell $\widetilde{\boldsymbol{c}}_{t-1}$ and by adding new values from $\widetilde{\boldsymbol{c}}_t$ (scaled by the input gate). The new hidden state $\boldsymbol{h}_t$ is formed by filtering $\boldsymbol{c}_t$ using the output gate.

A criticism of the LSTM and GRU architectures is that both seem ad hoc solutions for the vanishing gradient problem. Although the authors of both models claim that the construction of those gates has some grounding, the purpose of each construction decision is not immediately apparent. This type of criticism seems reasonable, studies have shown that by a process of systematic trial and error it is possible to find a set of equation defining a competitive RNN model without the input of human intuition [Rawal and Miikkulainen, 2018, Schrimpf et al., 2018].

2.4 Combining Recurrent Models

In order to facilitate the description of the most recent deep learning model used in NLP, we need to present two concepts built on top of the RNN model: the *encoder-decoder architecture* and the notion of *attention*.

2.4.1 Neural Machine Translation

RNN based language models are successfully used in the machine translation task. In this task we try to map sentences from a *source language* to a *target language*. A dataset for this task is composed of sentence pairs (T_{so}, T_{ta}) such that one sentence is the translated version of the other. For example, by taking English and German as the source and target language, respectively; we have observations of the form:

$T_{so} =$ `Gentleman, today you see me washing glasses.`

$T_{ta} =$ `Meine Herren, heute sehen Sie mich Gläser abwaschen.`

One translation model based on RNNs is known as the *encoder-decoder* architecture (or *sequence-to-sequence model*) [Sustskever et al., 2014]. The main idea is to use one neural language model to encode the sentences from the source language (the *encoder*), and use another neural network to predict the next word in the target language conditioned to previously selected words from the source and target languages (the *decoder*).

For instance, let $T_{so} = (w_1, \ldots, w_k)$, $T_{ta} = (w'_1, \ldots, w'_m)$ and let $\mathcal{V}_{so}$ and $\mathcal{V}_{ta}$ be the vocabularies for the source and target languages, respectively, such that $|\mathcal{V}_{so}| = V_1$ and $|\mathcal{V}_{ta}| = V_2$. We define the encoder, $f_{enc} : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \to \mathbb{R}^{d_2}$ and the decoder, $f_{dec} : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \to \mathbb{R}^{d_2}$ as two different realizations of the same RNN architecture (either a simple RNN, a GRU or a LSTM). We also define a parameterized function $g : \mathbb{R}^{d_2} \to [0,1]^{V_2}$ to map hidden states to a distribution over words from the target language. The computation of the encoder-decoder model can be described in two steps. First, we use f_{enc} to obtain all the hidden states associated with the subsequences from $(w_1, \ldots, w_k)$:

$$\begin{aligned} \boldsymbol{h}_t &= f_{enc}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) \\ \boldsymbol{x}_t &= \boldsymbol{E_{so}} hot_{so}(w_t), \end{aligned} \tag{2.20}$$

where

- $t \in \{1, \ldots, k\}$.
- $\boldsymbol{E_{so}} \in \mathbb{R}^{d_1 \times V_1}$ is the source language embedding matrix and $hot_{so} : \mathcal{V}_{so} \to \{0,1\}^{V_1}$ is the associated one-hot encoding.
- $\boldsymbol{h}_0 \in \mathbb{R}^{d_2}$ is the zero vector.

We interpret $\boldsymbol{h}_k$ as a context vector that encodes the source sentence T_{so}. At a second moment, we use $\boldsymbol{h}_k$ to define a prediction for every word from T_{ta}:

$$\begin{aligned} \hat{\boldsymbol{y}}_t &= g(\boldsymbol{h}'_t) = \text{softmax}(\boldsymbol{W}\boldsymbol{h}'_t + \boldsymbol{b}) \\ \boldsymbol{h}'_t &= f_{dec}(\boldsymbol{x}'_t, \boldsymbol{h}'_{t-1}) \\ \boldsymbol{x}'_t &= \boldsymbol{E_{ta}} hot_{ta}(w'_t), \end{aligned} \tag{2.21}$$

where

- $t \in \{0, \dots, m\}$.
- $\boldsymbol{E_{ta}} \in \mathbb{R}^{d_1 \times V_2}$ and $hot_{ta} : \mathcal{V}_{ta} \to \{0,1\}^{V_2}$.
- $w'_0 =$ `[S]` and $\boldsymbol{h'}_{-1} = \boldsymbol{h}_k$.
- $\hat{\boldsymbol{y}}_t = \hat{\mathbb{P}}_{\boldsymbol{\theta}}(W'_{t+1} \,|\, w_1, \dots, w_k, w'_1, \dots, w'_t)$ is the estimated conditional distribution over the words from the target language.

The most relevant modification in this setup is the use of the encoded sentence $\boldsymbol{h}_k$ as the initial hidden state $\boldsymbol{h'}_{-1}$. For each word w'_t appearing in T_{ta} we can obtain the pointwise loss

$$l_t = CE(hot_{ta}(w'_t), \hat{\boldsymbol{y}}_t). \tag{2.22}$$

The loss associated to the observation (T_{so}, T_{ta}) is the sum of all partial losses $l_1, \dots, l_m$. Both encoder and decoder are trained jointly, thus the representation for both the source and target languages are learned in the training phase.

2.4.2 Attention

The encoder-decoder model uses all encoded words from the source sentence to predict the next target word. However, for some cases only parts of the source sentence are relevant. For example, it is easier to predict the German word `Herren` using only the English word `Gentleman` and the previous German word `Meine`. *Encoder-decoder with attention* is a variant of the model described in (2.20) and (2.21) such that it "learns" which parts of the encoding sentence are relevant for each prediction $\hat{\boldsymbol{y}}_t$ [Bahdanau et al., 2015].

In order to describe this model, let us continue in the same setup as described in subsection 2.4.1, where $\boldsymbol{h}_1, \dots, \boldsymbol{h}_k$ are all hidden states generated by the encoder. The attention model differs from (2.21) in the way that for each $t \in \{0, \dots, m\}$, the prediction $\hat{\boldsymbol{y}}_t$ is based on a context vector $\boldsymbol{c}_t$ that selectively "pays attention" to some elements from the sequence $(\boldsymbol{h}_1, \dots, \boldsymbol{h}_k)$. This model is defined by the following equations:

$$\begin{aligned}
\hat{\boldsymbol{y}}_t &= \text{softmax}(\boldsymbol{W}\widetilde{\boldsymbol{h}}_t + \boldsymbol{b}) \\
\widetilde{\boldsymbol{h}}_t &= tahn(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h'}_t] + \boldsymbol{b_c}) \\
\boldsymbol{c}_t &= attend(\boldsymbol{h}_1, \dots, \boldsymbol{h}_k, \boldsymbol{h'}_t) \\
\boldsymbol{h'}_t &= f_{dec}(\boldsymbol{x'}_t, \boldsymbol{h'}_{t-1}) \\
\boldsymbol{x'}_t &= \boldsymbol{E_{ta}} hot_{ta}(w'_t),
\end{aligned} \tag{2.23}$$

where $\boldsymbol{h'}_{-1} \in \mathbb{R}^{d_2}$ is the zero vector. The core of this technique is the definition of the context vector $\boldsymbol{c}_t$ given by the attention mechanism *attend*. There are many variants of the attention mechanism, here we present the one described by Luong et al. [2015] as *global attention*. The intuition behind the attention mechanism is that we assign a score to the hidden vectors $\boldsymbol{h}_1, \dots, \boldsymbol{h}_k$ according to its relevance for $\boldsymbol{h'}_t$. We normalize these scores using the softmax function and then we define the context vector $\boldsymbol{c}_t$ as the weighted average over all the source hidden states. Formally, we define the attention mechanism as follows:

$$\begin{aligned}
attend(\boldsymbol{h}_1, \dots, \boldsymbol{h}_k, \boldsymbol{h'}_t) &= \sum_{i=1}^{k} \boldsymbol{\alpha}_{t,i} \boldsymbol{h}_i, \\
\boldsymbol{\alpha}_t &= softmax\left(\begin{bmatrix} score(\boldsymbol{h'}_t, \boldsymbol{h}_1) \\ \vdots \\ score(\boldsymbol{h'}_t, \boldsymbol{h}_k) \end{bmatrix} \right),
\end{aligned} \tag{2.24}$$

where *score* is a parametererized function. Luong et al. [2015] list three possible alternatives to this function:

$$score(\boldsymbol{h}'_t, \boldsymbol{h}_i) = \begin{cases} \boldsymbol{h}'^{\top}_t \boldsymbol{h}_i, \\ \boldsymbol{h}'^{\top}_t \boldsymbol{W}_a \boldsymbol{h}_i, \\ \boldsymbol{v}_a^{\top} tahn(\boldsymbol{W}_a[\boldsymbol{h}'_t; \boldsymbol{h}_i]), \end{cases} \tag{2.25}$$

where both $\boldsymbol{v}_a$ and $\boldsymbol{W}_a$ are specific parameters for the score function. In this model, the encoder, decoder and the attention mechanism are learned in the training phase. One of the main advantages of this method is that it introduces some interpretability to the deep learning model. In the decoding stage, we can observe the attention weights $\boldsymbol{\alpha}_{t,i} \in [0, 1]$ (for $i = 1, \ldots, k$) and check which parts of the source sentence are relevant for the prediction on the step t (Figure 2.1 shows an output of this method).

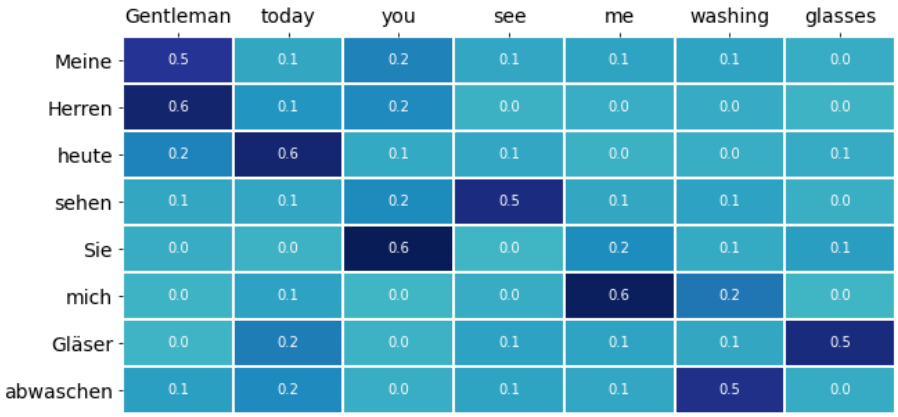

Figure 2.1: *Attention matrix $\boldsymbol{\alpha}$ for the English/German translation example. For $t = 1, \ldots, 8$, the row $\boldsymbol{\alpha}_t$ displays the attention weights $\boldsymbol{\alpha}_{t,i}$ relating the target word w'_t and the source words w_i (where $i \in \{1, \ldots, 7\}$).*

2.5 The Transformer

In the highly influential work by Vaswani et al. [2017], it is proposed a deep learning model based solely on the idea of attention. They have called such model the *transformer*. Since this is a complex model, we first present its main components: *self-attention* and *positional-encoding*. Then, we describe the full model and the associated training routine.

2.5.1 Self-Attention

As the name indicates, self-attention is the process of computing attentions weights by comparing a sentence with itself. Before explain this process, it is worthwhile to follow Vaswani et al. [2017] and formulate the attention method in an abstract way. Roughly speaking, the attention function can be described as a process of taking a *query* $\boldsymbol{q}$, computing the compatibility score of $\boldsymbol{q}$ with each *key* $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_n$; and using such score to construct the weighted sum of the *values* $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$.

More formally, for $d_k, d_v \in \mathbb{N}$, let $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n \in \mathbb{R}^{d_k}$ be a sequence of queries, $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_n \in \mathbb{R}^{d_k}$ be a sequence of keys, and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{R}^{d_v}$ be a sequence of values. For $t = 1, \ldots, n$, we define the attention vector $\boldsymbol{z}_t$ based on the query $\boldsymbol{q}_t$ as follows:

$$
\begin{aligned}
\boldsymbol{z}_t &= \sum_{i=1}^{n} \boldsymbol{\alpha}_{t,i} \boldsymbol{v}_i, \\
\boldsymbol{\alpha}_t &= softmax\left(\begin{bmatrix} score(\boldsymbol{q}_t, \boldsymbol{k}_1) \\ \vdots \\ score(\boldsymbol{q}_t, \boldsymbol{k}_n) \end{bmatrix}\right) \\
score(\boldsymbol{q}_t, \boldsymbol{k}_i) &= \frac{\boldsymbol{q}_t{}^{\top} \boldsymbol{k}_i}{\sqrt{d_k}}.
\end{aligned} \tag{2.26}
$$

Clearly, (2.26) is just a reformulation of the computations described in (2.24) and (2.25) without referring to the hidden states of a recurrent model and selecting an specific score function. Using matrix notation, we can further simplify the attention function. Let $\boldsymbol{Q} \in \mathbb{R}^{n \times d_k}, \boldsymbol{K} \in \mathbb{R}^{n \times d_k}, \boldsymbol{V} \in \mathbb{R}^{n \times d_v}$ be the matrices of queries, keys and values, respectively. The matrix of outputs of the attention process can be computed as:

$$
Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{d_k}}\right)\boldsymbol{V}. \tag{2.27}
$$

Instead of defining a single attention weight to connect a query with a key, Vaswani et al. [2017] define a function that associates multiple attentions weights for different linear projection of these objects. They have called such function a *multi-head attention*:

$$
\begin{aligned}
MultiHead(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= [head_1; \ldots; head_h]\boldsymbol{W^O} \\
head_i &= Attention(\boldsymbol{Q}\boldsymbol{W_i^Q}, \boldsymbol{K}\boldsymbol{W_i^K}, \boldsymbol{V}\boldsymbol{W_i^V}),
\end{aligned} \tag{2.28}
$$

where

- $i \in \{1, \ldots, h\}$.
- $\boldsymbol{W_i^Q} \in \mathbb{R}^{d_1 \times d_k}$, $\boldsymbol{W_i^K} \in \mathbb{R}^{d_1 \times d_k}$, $\boldsymbol{W_i^V} \in \mathbb{R}^{d_1 \times d_v}$, and $\boldsymbol{W^O} \in \mathbb{R}^{hd_v \times d_1}$.
- $d_1, h \in \mathbb{N}$ are hyperparameters of the model. Such that d_1 is the output size and h is the number of heads.

By setting $d_k = d_v$, the *self-attention function* (*self-attention layer*) is the mapping defined as

$$
\text{self-attend}(\boldsymbol{X}) = MultiHead(\boldsymbol{X}, \boldsymbol{X}, \boldsymbol{X}), \tag{2.29}
$$

where $\boldsymbol{X} \in \mathbb{R}^{n \times d_k}$ is a sequence of n vectors in matrix format. For example, $\boldsymbol{X}$ can be a matrix composed of n word embeddings.

Self-attention is one method of capturing relations among a sequence of vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. It is an alternative to the RNN style models. One of the advantages of self-attention is the property of connecting all positions in a sequence with a constant number of sequentially executed operations [Vaswani et al., 2017]. One of the side-effects associated to this technique is the production of interpretable models. After training the model it is expected that the attention weights exhibit a behavior related to some syntactic and semantic phenomenon. However, as usual in deep learning, the interpretation is not straightforward (as illustrated in Figure 2.2).

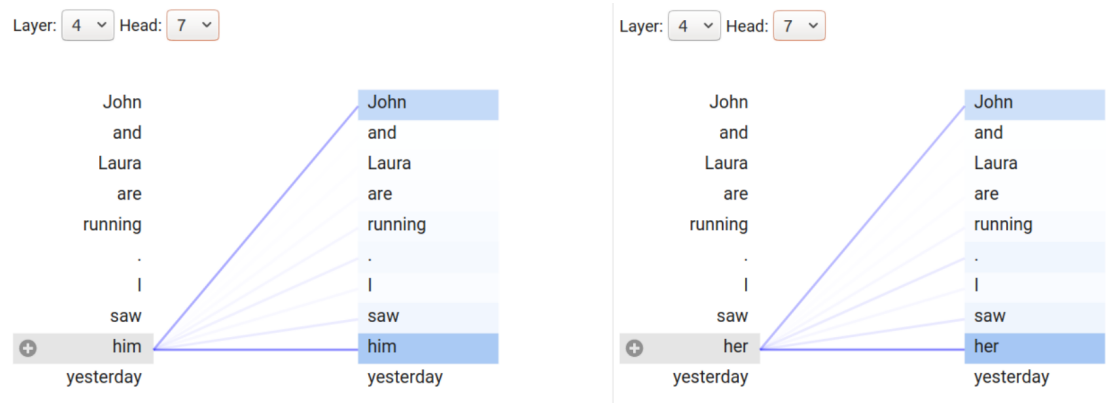

Figure 2.2: *Self-attention output generated by the visualization tool introduced by Vig [2019]. Using a trained model based on self-attention (GPT-2 [Radford et al., 2019]), the figure displays the attention weights related to the attention layer 4 and head 7. The weights are represented as lines connecting the query words (left) with the value words (right). In the left figure, the weights may indicate that this attention head is involved in anaphora resolution. However, by changing the pronoun in the sentence, the right figure, we loose this possible interpretation.*

2.5.2 Positional Encoding

Self-attention can only capture relation between words. In order to add word ordering in the model, Vaswani et al. [2017] have introduced the notion of *positional encoding*. The core idea of this method is to use a multidimensional vector to represent a position t in a sequence of n elements. This is done by defining a vector of sinusoidal functions, a positional vector. Then, such vector is summed to the regular word embeddings.

For $d \in \mathbb{N}$ (where d is even), the positional encoding is the mapping $PE : \mathbb{N}\backslash\{0\} \to \mathbb{R}^d$ such that for any position t, the positional encoding vector $PE(t)$ is defined as:

$$PE(t)_i = \begin{cases} \sin\left(\omega_{\kappa(i)} t\right), & \text{if } i \text{ is odd} \\ \cos\left(\omega_{\kappa(i)} t\right), & \text{otherwise,} \end{cases} \tag{2.30}$$

where

- $i \in \{1, \ldots, d\}$.
- $\kappa : \mathbb{N} \to \mathbb{N}$ is the mapping:
$$\kappa(i) = \begin{cases} \frac{i+1}{2}, & \text{if } i \text{ is odd} \\ \frac{i}{2}, & \text{otherwise.} \end{cases}$$
- for $m \in \mathbb{N}$, ω_m is the frequency of the each sinusoidal curve defined as
$$\omega_m = \frac{1}{10000^{\frac{2m}{d}}}.$$

To put it another way, the positional encoding is just a vector containing pairs of sines and cosines with different frequencies. All sinusoidal functions are applied to t:

$$PE(t) = \begin{bmatrix} \sin(\omega_1 t) \\ \cos(\omega_1 t) \\ \sin(\omega_2 t) \\ \cos(\omega_2 t) \\ \vdots \\ \sin\left(\omega_{\frac{d}{2}} t\right) \\ \cos\left(\omega_{\frac{d}{2}} t\right) \end{bmatrix}$$

For example by setting $d = 80$, we can visualize how each encoding vector represents one position in a sequence of length 200 (the positional embeddings are the rows in matrix displayed in Figure 2.3).

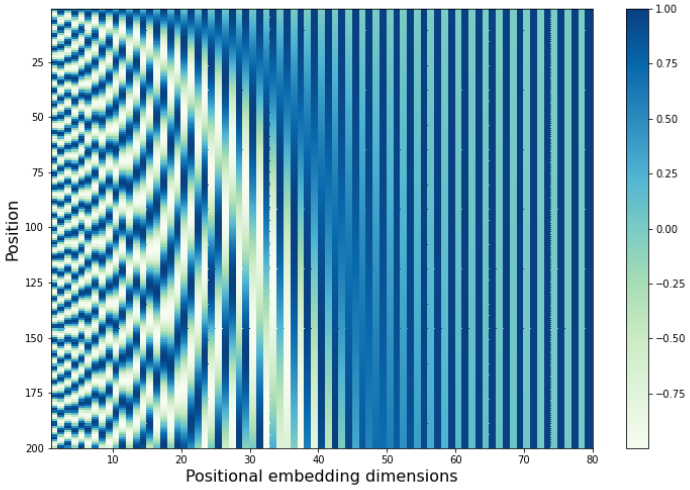

Figure 2.3: *Positional encoding of dimension* 80. *Each row* t *represents the positional vector* $PE(t)$.

2.5.3 The Transformer Architecture

Self-attention and positional encoding are the essential ideas behind the transformer model. Before writing all the equations to define the model it is helpful to establish some auxiliary concepts. As usual in deep learning, the transformer architecture also uses affine transformations combined with a *rectified linear units* (ReLU) function. This computation can be summarized on a single function called *feed-forward network* FFN:

$$FFN(\boldsymbol{X}) = max(0, \boldsymbol{X}\boldsymbol{W_1} + \boldsymbol{b_1})\boldsymbol{W_2} + \boldsymbol{b_2}. \tag{2.31}$$

In order to stabilize the computations of the model, Vaswani et al. [2017] use the *layer normalization* method described by Ba et al. [2016]. This stabilization process is denoted as $LayerNorm$. With all theses notions at hand, we can easily describe the transformer network. This model can be seen as a variation of the encoder-decoder architecture. The notable difference is the substitution of the notion of recurrence by self-attention.

We explain the transformer model using the automatic translation task. Hence, we start with the source and target sentences $T_{so} = (w_1, \ldots, w_k)$ and $T_{ta} = (w'_1, \ldots, w'_m)$ as mentioned previously. Let $\boldsymbol{X} \in \mathbb{R}^{k \times d}$ and $\boldsymbol{Y} \in \mathbb{R}^{m \times d}$ be the rearranging of the source and target word embeddings in matrix format, respectively. As in the case when using the encoder-decoder model, we first apply the encoder model to the input:

$$\begin{aligned} \boldsymbol{e} &= F_{enc}(\boldsymbol{X'}) \\ \boldsymbol{X'} &= \boldsymbol{X} + \begin{bmatrix} PE(1)^\top \\ \vdots \\ PE(k)^\top \end{bmatrix}, \end{aligned} \tag{2.32}$$

where F_{enc} is the *transformer encoder* (*encoder layer*) defined by the equations:

$$\begin{aligned} F_{enc}(\boldsymbol{X'}) &= LayerNorm(\boldsymbol{e^3} + \boldsymbol{e^2}) \\ \boldsymbol{e^3} &= FFN(\boldsymbol{e^2}) \\ \boldsymbol{e^2} &= LayerNorm(\boldsymbol{X'} + \boldsymbol{e^1}) \\ \boldsymbol{e^1} &= \text{self-attend}(\boldsymbol{X'}). \end{aligned} \tag{2.33}$$

After the encoded matrix $\boldsymbol{e} \in \mathbb{R}^{k \times H}$ is computed (H is the output size of the encoder layer), we use the decoder model to predict the probability of the next target word. We need to perform this computation for each $t \in \{1, \ldots, m\}$. Since the decoder model uses the target sentence $\boldsymbol{Y}$ as input, we need to apply a mask in order to block the attention mechanism to store information on future words during training.

$$\begin{aligned} \hat{\boldsymbol{y}}_t &= \text{softmax}(\boldsymbol{W}\boldsymbol{d}_t + \boldsymbol{b}) \\ \boldsymbol{d}_t &= F_{dec}(\boldsymbol{O'_t}, \boldsymbol{e}) \\ \boldsymbol{O'_t} &= mask(\boldsymbol{O}, t) \\ \boldsymbol{O} &= \boldsymbol{Y} + \begin{bmatrix} PE(1)^\top \\ \vdots \\ PE(m)^\top \end{bmatrix}, \end{aligned} \tag{2.34}$$

where

- $t \in \{1, \ldots, m\}$.
- $mask(\boldsymbol{O'}, t)$ is the result of filling with zeros every row $j \geq t$ of the matrix $\boldsymbol{O'}$.
- F_{dec} is the *Transformer decoder* (*decoder layer*) defined as follows:

$$\begin{aligned} F_{dec}(\boldsymbol{O'_t}, \boldsymbol{e}) &= LayerNorm(\boldsymbol{d_t^4} + \boldsymbol{d_t^5}) \\ \boldsymbol{d_t^5} &= FFN(\boldsymbol{d_t^4}) \\ \boldsymbol{d_t^4} &= LayerNorm(\boldsymbol{d_t^2} + \boldsymbol{d_t^3}) \\ \boldsymbol{d_t^3} &= MultiHead(\boldsymbol{e}, \boldsymbol{e}, \boldsymbol{d_t^2}) \\ \boldsymbol{d_t^2} &= LayerNorm(\boldsymbol{O'_t} + \boldsymbol{d_t^1}) \\ \boldsymbol{d_t^1} &= \text{self-attend}(\boldsymbol{O'_t}). \end{aligned} \tag{2.35}$$

As in the case of neural machine translation, the loss function is defined by using the categorical cross-entropy loss to compare $\hat{\boldsymbol{y}}_t$ with $hot(w'_t)$. For presentation purposes, in equations (2.32) and (2.34) we have defined the Transformer model using only one single pair of encoder and decoder layers (Figure 2.4 helps us visualize the model architecture using a single diagram). In practice, we define more complex models by combining multiple instances of the same type of layer. For example,

in the original paper [Vaswani et al., 2017], the architecture used was composed of six encoders and six decoders layers.

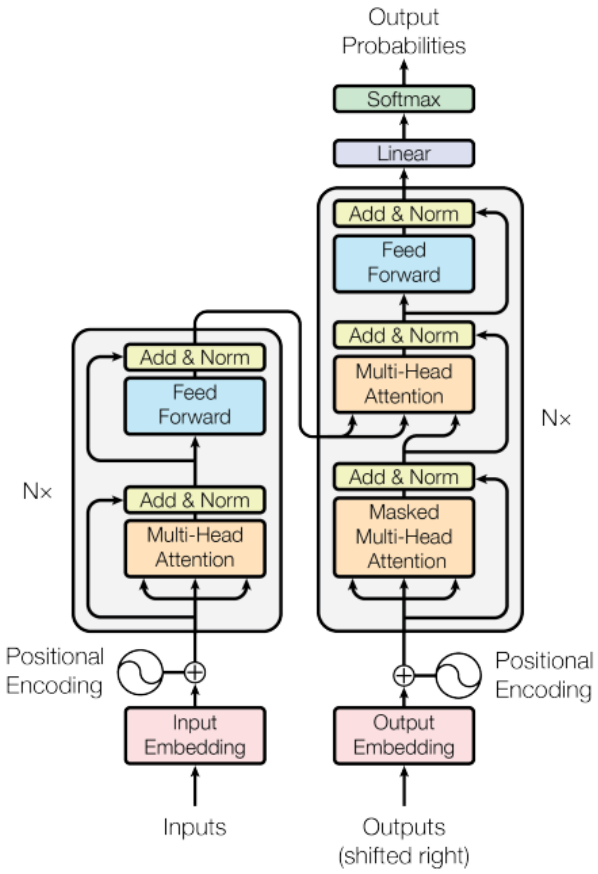

Figure 2.4: *Transformer model represented as a diagram (Figure by Vaswani et al. [2017]) The figure displays a visual representation of equations (2.32), (2.33), (2.34), and (2.35).*

2.6 Transformer Based Models

One of the recent breakthroughs in the NLP field is the combined use of the transformer model and generative pre-training [Howard and Ruder, 2018, Radford et al., 2018]. The main idea is to first train a transformer network in the language modeling task, then use all learned attention weights and adapt the parameters for a new text classification task. This technique is known as *transfer learning*. In this section, we briefly describe the application of transfer leaning for NLP, we also explain some models based on this training strategy.

2.6.1 Transfer Learning

Transfer learning is the procedure of training a model in two stages. We first train a model on a *source task*, then we take the estimated model and train it again on a *target task*. These two phases are refereed to as *pre-training* and *fine-tunning*, respectively. Such technique is well established in the computer vision community [Yosinski et al., 2014]. Often in computer vision, the source task is an image classification task defined in a large dataset of different images. And the target task is another image classification task composed of non-overlapping images. It is expected that the features learned in the source task can be general enough in order to be useful for different target tasks:

> Many visual categories *share* low-level notions of edges and visual shapes, the effects of geometric changes, changes in lighting and so on. In general, transfer learning, multi-

> task learning and domain adaptation can be achieved via representation learning when there exist features that are useful for the different setting or tasks, corresponding to underlying factors that appear in more than one setting. [Goodfellow et al., 2017, p. 527]

As we have seen in Section 2.2, word embedding is a simplified form of transfer learning where only one part of the neural network model was first pre-trained and then fine-tuned (the embedding matrix). A more general approach is proposed by Howard and Ruder [2018]: train a model on the language model task, then fine-tune some version of the model in a new text classification task. By choosing language modeling as the source task we can make use of the large unlabeled corpora public available (e.g., all texts present in the Wikipedia). This is especially beneficial when training deep learning models in target tasks with low-resources dataset (the NLP community also uses the term *downstream task* to refer to a target task). As in the case of computer vision, it is expected that some general properties of language are captured when we pre-train the model in a large corpus.

2.6.2 BERT

The network named BERT (bidirectional encoder representations from transformers) is a remarkable achievement due to its simplicity and prediction power. This model is based on the core idea by Radford et al. [2018]: apply the transfer learning technique from Howard and Ruder [2018] using the transformer architecture. However, the model by Radford et al. [2018] (called GPT - generative pre-trained transformer) was based on a *transformer decoder*. As we have seen in (2.34), the decoder only uses the past words from the target sentence to generate the prediction; in other words, it is not a bidirectional model. One of the main ideas by Devlin et al. [2019] is to construct a model based on a *transformer encoder* to allow bidirectional predictions.

Using the notation from Devlin et al. [2019], we can describe BERT as a stack of L encoder layers, the size of the vector output after passing through those layers is H and the number of self-attention heads in each layer is A. There are two versions of the BERT model:

- $\text{BERT}_{\text{BASE}}$: $L = 12$, $H = 768$ and $A = 12$.
- $\text{BERT}_{\text{LARGE}}$: $L = 24$, $H = 1024$ and $A = 16$.

Since BERT is designed to be a model adaptable to different tasks (text classification using one or two text inputs, language modeling and QA), the input text used for this model is arranged to be as general as possible. There are two new strategies to organize the text input: i) the token `[CLS]` is added in the beginning of text input (the associated embedding for this token encodes the class in a classification task); ii) in order to differentiate two sentences inside the input text, the separate token `[SEP]` is introduced between the sentences; on top of that, an especial embedding for each sentence type is added to each word.

Thus, the text input for this model is of the form `[CLS]` $w_1 \dots w_k$ `[SEP]` $w'_1 \dots w'_m$ `[SEP]`. In this case, $w_1 \dots w_k$ denotes the first sentence and $w'_1 \dots w'_m$ denotes the second sentence. Tasks like NLI and QA use two sentences as the input. For other tasks that do not require such type of input, the input text is of the form `[CLS]` $w_1 \dots w_k$ `[SEP]`. All inputs are constrained such that $m + k \leq n$, where n is the maximum input size. In [Devlin et al., 2019], n is set to 512. Figure 2.5 shows a simple sentence pair organized as the input for this model.

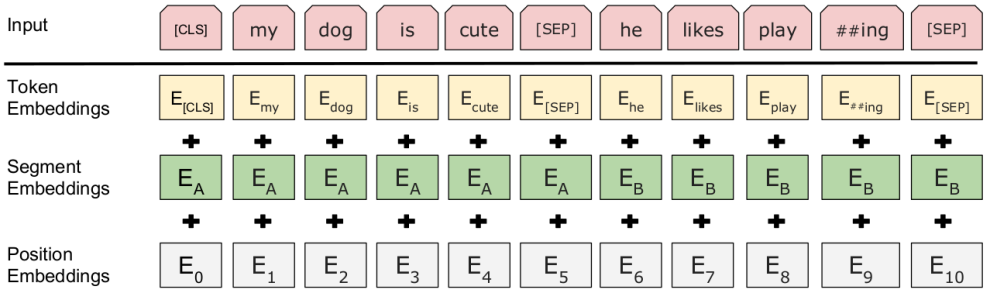

Figure 2.5: *Input format for the BERT model (Figure by Devlin et al. [2019]).*

As mentioned before, BERT differentiates itself from the model presented by Radford et al. [2018] by using the encoder layer as the basic building block for its architecture. However, by doing so, it was observed that the language modeling task should be altered. It is not possible to use the same training routine with the transformer encoder, because the self-attention mechanism forces that all words in the input sentence become connected among themselves. In order to circumvent this problem, Devlin et al. [2019] propose two new language modeling tasks.

Masked language modeling (MLM) is the task of predicting an specific token that was omitted from a sentence. The omitted word is replaced by the `[MASK]` token. For example, one possible input is the phrase `this place is [MASK] quiet` obtained from the sentence `this place is awful quiet`; the goal is to predict the omitted word `awful`. The dataset for this task is constructed by taking a large corpus and masking randomly some words from the sentences that compose the corpus. In the work by Devlin et al. [2019], only 15% of the words in each sequence are masked. The output of the BERT model is a matrix of size $512 \times H$. Thus, in the MLM task only the vectors corresponding to the masked token are used to create the prediction. It should be noted that in the MLM task we do not perform explicit density estimation as in the original language modeling task. The aim here is to reconstruct the original data from a corrupted input, hence the MLM task allows the model to use a bidirectional context for reconstruction.

Next sentence prediction (NSP) is the text classification task of deciding whether a tuple of sentences (A, B) follow each other in the original text. To create a dataset for this task, it is used both real sequence pairs obtained from the corpus and synthetic examples of non-consecutive sentences. One positive example is the pair ("`to be, or not to be?`", "`that is the question`"); and one negative example is ("`in the beginning God created the heaven and the earth`", "`there was no possibility of taking a walk that day`").

The complete pre-training for BERT is based on both tasks, i.e., the training loss is the sum of the mean masked language modeling and mean next sentence prediction likelihood. In order to perform the fine-tuning for text classification we take the pre-trained model and use only the vector corresponding to the `[CLS]` embedding (the first output of the BERT model) denoted $\boldsymbol{c} \in \mathbb{R}^H$. Hence, the BERT model for classification can be defined as follows:

$$\begin{aligned} \hat{\boldsymbol{y}} &= \operatorname{softmax}(\boldsymbol{W}\boldsymbol{c} + \boldsymbol{b}) \\ \boldsymbol{c} &= BERT^*(\boldsymbol{X'}) \end{aligned}$$

$$\boldsymbol{X'} = \boldsymbol{X} + \begin{bmatrix} PE(1)^\top \\ \vdots \\ PE(512)^\top \end{bmatrix} + \begin{bmatrix} SE(A)^\top \\ \vdots \\ SE(B)^\top \end{bmatrix}, \tag{2.36}$$

where

- $\hat{\boldsymbol{y}}$ is the estimated label probabilities.

- $BERT^*$ is the pre-trained version of either $\text{BERT}_{\text{BASE}}$ or $\text{BERT}_{\text{LARGE}}$.
- $\boldsymbol{X}$ is the matrix of word embeddings associated to the input text; PE is the positional encoding vector and SE in the segment embedding vector.

The model is trained as usual in the text classification task, i.e., all parameters are modified in order to minimize the cross-entropy error. Figure 2.6 shows a diagram representing the acquisition of the vector $\boldsymbol{c}$ in a text classification task that uses a pair of sentences as the input.

$\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$ differ in complexity, the former is composed of 110M parameters and the latter has 340M parameters. Empirically it seems that complex models yield a better performance. In the work by Devlin et al. [2019], it is reported that the average accuracy for these models on different text classification tasks are 79.6% and 81.9%, respectively.

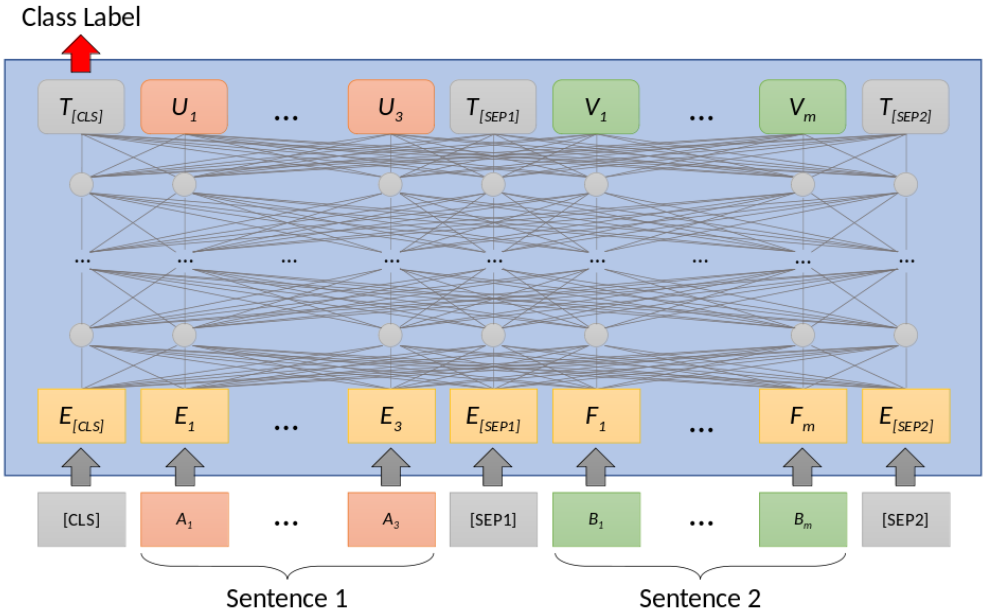

Figure 2.6: *BERT performing text classification. In this example the input text is composed of two sentences (Figure by Lin [2020]).*

2.6.3 RoBERTa

ROBERTA (robustly optimized BERT approach) [Liu et al., 2019b] is a variation of the BERT model based on different design decisions when pre-training BERT models. They improve the training in BERT using four new ideas:

- *Dynamic masking.* When creating a masked observations for the MLM task, the original training in BERT was based on defining a single mask for each observation in the pre-processing stage (*static masking*). In RoBERTA, it is used a *dynamic masking*, i.e., a training strategy where a new mask is generated each time a training observation is fed to the model.
- *Full-sentences without NSP.* The input for the MLM task is constructed from full sentences sampled contiguously from one or more documents. An addition separator token is introduced to separate sentences from different documents. The loss term associated to the NSP task is removed.
- *Large mini-batches.* It is possible to improve training when using large mini-batches by taking an appropriate learning rate. The original BERT model was trained using a batch size of 256 sequences, in RoBERTa it is used batches of 8K sentences.

- *Larger byte-level byte-pair encoding.* Byte-pair encoding is a type of encoding that uses sub-words units instead of full words. The construction of the sub-words is based on a statistical analysis of the training corpus. In order to store the resulting large vocabulary created by this encoding (the vocabulary size range from 10K-100K subword units), each subword unit is implemented using bytes instead of Unicode characters.

2.6.4 ALBERT

The model known as ALBERT (a lite BERT) [Lan et al., 2020] is a variation of the BERT architecture aimed to produce a model with fewer parameters, but maintaining state-of-the-art performance. Instead of using 100M-330M parameters for the base and large configuration as in BERT, ALBERT is able to produce models composed of 12M-18M parameters for the base and large configurations. This makes both training and inference faster without hurting performance. Three main techniques are used to perform parameter reduction:

- *Factorized embedding parameterization.* Language modeling usually deal with a large vocabulary $\mathcal{V}$. The vocabulary size affects the embedding matrix used to transform each one-hot representation to one word embedding. And since the size of the embedding is related to the hidden size of the model, the size of the vocabulary becomes one limitation in choosing large hidden size for the model. ALBERT solution is to use a factorization of the embedding matrix, decomposing them into two smaller matrices.

- *Cross-layer parameter sharing.* To stabilize the model, one design decision in ALBERT is to share all parameters across layers.

- *Inter-sentence coherence loss.* As observed by Liu et al. [2019b], removing the NSP loss improves model performance on different downstream tasks. In ALBERT, the NSP loss is substituted by the loss of other pre-training task: *the sentence-order prediction* (SOP). SOP is similar to the NSP task, the only difference is that the negative examples are two consecutive sentences but with their order swapped. One negative example in this task is the pair (“`that is the question`”, “`to be, or not to be?`”).

2.6.5 XLNet

For the sake of brevity, we are not going to review in detail the model XLNET. It is the most complex model among the ones present in this section. Here we will only comment on the motivations for this architecture.

XLNet is a model based on the transformer-XL model: an architecture that combines both the transformer network and the autoregressive nature of the RNN model [Dai et al., 2019]. The authors that proposed the XLNet model [Yang et al., 2019] bring the notion of recurrence back to the transformer model in order to add dependency in the MLM taks. One problem observed in this task, as formulated in the original BERT paper, is that the prediction of each masked token is independent even when there are a clear dependency between words. For example, one possible masking of the sentence `It is taller than the Empire State` is `It is taller than the [MASK] [MASK]`. Thus, in training the BERT model, it tries to predict the masked token for each one of the words `Empire` and `State` independently.

To solve this issue, the authors of XLNet propose to change the LM task by allowing different permutations of the sentence’s words in order to model multiple possible dependencies. Yang et al. [2019] also add an additional attention layer in the model in order to allow the model to use the position of the masked token (without revealing its content) to perform the prediction.

2.7 Hypothesis Testing

Often, we want to investigate whether there is a difference between two procedures. For example, we want to compare the average effects of the two medical treatments over the population. In the particular case of this thesis, we desire to compare the predictions of a text classifier in two different contexts. *Null hypothesis significance testing* (NHST) offers one theoretical framework to perform such comparisons.

2.7.1 Basic Formulation

Hypothesis testing is one inference method that help us decide which of two complementary statements about a population parameter are true. Such statements are often called the *null hypothesis* (H_0) and the alternative hypothesis (H_1). This inference method is used in science to help decide which one of two actions is going to be taken.

For example, we want to decide whether or not to use hydroxychloroquine (HCQ) in patients with COVID-19. Since this medicine can cause some undesirable side effects, we want to have some evidence to aid our decision in adopting such drug. One way of obtaining such verification is by experimentation. We start with an universal statement about the drug not having an effect: "For every possible scenario, HCQ has no effect in the treatment of patients with COVID-19". To show that this statement is false (and so we can conclude that for some scenarios, HCQ does have an effect) we try to find some empirical data to contradict it. One crucial step in this process is the translation of the universal statement into a precise statement about probability distributions. For example, one naive translation is:

> "Let A be a binary variable indicating the death of a COVID-19 patient after the 10th day, and let B be a similar variable associated with a patient treated with HCQ. Then, $A \sim Bernoulli(p_1)$, $B \sim Bernoulli(p_2)$, and $p_1 = p_2$" (I)

The null hypothesis (H_0) is just the part "$p_1 = p_2$" in (I) (the term "null" is associated with some treatment not having an effect). After we define such precise statement we collect some data about the subject and evaluate the likelihood of the claim (I) being true based on the observable data. If we detect a large discrepancy between (I) and the observable data, we say that we "Reject H_0", i.e., given the empirical evidence we can say that (I) is false.

There are two interpretation of the falsehood of (I): i) the whole theoretical formulation described in (I) is false, or (ii) the theoretical formulation is true and only the part "$p_1 = p_2$" is false. In i) we can affirm that there is evidence about the treatment having an effect for some scenarios, however we have no information about the correct model that governs the data generation process. In ii) we conclude that (I) becomes true when we substitute "$p_1 = p_2$" by its negation "$p_1 \neq p_2$" (the alternative hypothesis - H_1). The positions i) and ii) represent, respectively, the Fisher and Neyman-Pearson approaches to testing statistical hypothesis. In i) hypothesis testing is a tool to draw provisional conclusions about an experiment situation; and in ii) hypothesis testing is interpreted as a mechanical decision rule [Lehmann, 1993]. The current NHST theory comprehend both perspectives, because the main difference between the approaches lies on how to interpret a test result. It is up to the scientific worker to choose the best approach for each situation.

2.7.2 Technical Formulation

We define *an statistical model* as the triple $(\Omega, \mathcal{A}, \mathcal{P})$ where:

- Ω is the *sample space*, i.e., the set of possible outcomes of an experiment.
- $\mathcal{A} \subseteq \mathcal{P}(\Omega)$ is a σ-algebra, i.e., $\emptyset \in \mathcal{A}$, $\mathcal{A}$ is closed under complement, and is closed under countable unions.
- $\mathcal{P}$ is a set of probability distributions defined on $\mathcal{A}$.

We say that a statistical model is a *parametric model* if $\mathcal{P}$ can be written as

$$\mathcal{P} = \{\mathbb{P}_\theta \,:\, \theta \in \Theta\},$$

where $\Theta \subset \mathbb{R}^d$ for some $d \in \mathbb{N}$.

In hypothesis testing we represent one experiment scenario using a parametric statistical model $(\Omega, \mathcal{A}, \mathcal{P})$ whose parameter space is Θ. We assume that underlying the statistical model lies the true probability model $(\Omega, \mathcal{A}, \mathbb{P}_\theta)$, we divide the parameter space into two disjoint sets Θ_0 and Θ_1 and decide in which one lies the population parameter θ. The general format of the null and alternative hypothesis is:

$$H_0 : \theta \in \Theta_0 \text{ versus } H_1 : \theta \in \Theta_1.$$

Both H_0 and H_1 are claims (*hypotheses*) about the statistical model $(\Omega, \mathcal{A}, \mathcal{P})$. We say that a hypothesis H of the form $H : \theta = \theta_0$ is a *simple* hypothesis. Similarly, for $|\Theta'| > 1$, we say that a hypothesis H of the form $H : \theta \in \Theta'$ is a *composite hypothesis*.

Let X be a random variable defined in the probability space $(\Omega, \mathcal{A}, \mathbb{P}_\theta)$, and $X_1, \ldots, X_n$ a sample. We test the hypothesis H_0 by defining a *test statistic* $T(X_1, \ldots, X_n)$ and a *rejection region* R of the form $R = \{(x_1, \ldots, x_n) : T(x_1, \ldots, x_n) > c\}$ for $c \in \mathbb{R}$. A *test* is a function defined on the sample space of the form:

$$te(x_1, \ldots, x_n) = \begin{cases} \textbf{decision 0}, \ \textbf{if } (x_1, \ldots, x_n) \in R, \\ \textbf{decision 1}, \ \textbf{otherwise} \ , \end{cases} \tag{2.37}$$

where "decision 0" and "decision 1" stand for: i) "Reject H_0" and "Do not reject H_0", respectively (the Fisher approach); and ii) "Reject H_0 / Accept H_1" and "Reject H_1 / Accept H_0", respectively (the Neyman-Pearson approach).

Example 1. Let $X_1, \ldots, X_n \sim \mathcal{N}(\mu, \sigma^2)$ where σ is known. We want to test:

$$H_0 : \mu \leq 0 \text{ versus } H_1 : \mu > 0.$$

In this case, $\Theta_0 = (-\infty, 0]$ and $\Theta_1 = (0, \infty)$. □

It should be noted that, by design, we know that the underlying process follows a normal distribution, hence it is not problematic to adopt the Neyman-Pearson approach for this example. Thus, one family of tests suitable for this scenario can be defined using the sample mean as the test statistic:

$$te_c(x_1, \ldots, x_n) = \begin{cases} \textbf{Reject } H_0 \text{ / } \textbf{Accept } H_1, \ \textbf{if } \bar{x} > c, \\ \textbf{Reject } H_1 \text{ / } \textbf{Accept } H_0, \ \textbf{otherwise} \ , \end{cases} \tag{2.38}$$

where $\bar{x}$ is the sample mean and $c \in \mathbb{R}$. Even in this simple scenario, we can define a infinite number of tests for the same null hypothesis ($te_{0.022}, test_{0.132}$, etc.). In order to simplify the decision process, we need to establish some method of test selection.

2.7.3 Power and Size

Consider the same scenario as in **Example 1** and a test te_c with rejection region $R_c = \{(x_1, \ldots, x_n) : \bar{x} > c\}$. For each $\mu \in \Theta$ we can calculate the probability of a random sample falling in the reject region R_c:

$$\begin{aligned} \mathbb{P}_\mu((X_1, \ldots, X_n) \in R_c) &= \mathbb{P}_\mu(\bar{X} > c) \\ &= \mathbb{P}_\mu\left(Z_n > \frac{\sqrt{n}(c - \mu)}{\sigma}\right), \end{aligned} \tag{2.39}$$

where $\bar{X} = n^{-1}\sum_{i=1}^{n} X_i$ and $Z_n = \sqrt{n}(\bar{X} - \mu)/\sigma$. By the Central Limit Theorem [Wasserman, 2010, Theorem 5.8], Z_n converges to Z in distribution ($Z_n \rightsquigarrow Z$) where $Z \sim \mathcal{N}(0,1)$.[5] Thus, let ϕ be the cumulative distribution function for the standard normal, we have

$$\begin{aligned}\mathbb{P}_\mu\left(Z_n > \frac{\sqrt{n}(c-\mu)}{\sigma}\right) &\approx \mathbb{P}\left(Z > \frac{\sqrt{n}(c-\mu)}{\sigma}\right)\\ &= 1 - \phi\left(\frac{\sqrt{n}(c-\mu)}{\sigma}\right).\end{aligned} \tag{2.40}$$

With this result we can approximate the probability of a random observation being in the reject region for any test when we assume that the population parameter is $\mu \in \Theta$. For example, consider the test $te_{0.022}$, and assume $n = 100$, $\sigma = 1$ and $\mu = 0$:

$$\begin{aligned}\mathbb{P}_\mu(\bar{X} > 0.022) &\approx \mathbb{P}\left(Z > \frac{\sqrt{100}(0.022-0)}{1}\right)\\ &\approx \mathbb{P}\left(Z > 0.22\right)\\ &= 1 - \phi(0.22)\\ &= 1 - 0.59\\ &= 0.41.\end{aligned} \tag{2.41}$$

This means that when the true parameter is $\mu = 0$ and we use the test $te_{0.022}$ to verify the claim H_0 by taking a sample of size 100, the probability of affirming that H_0 is false when H_0 is indeed the case is 0.41. This type of mistake is refereed to *type I error*.

In the general case, we calculate the probability of making this mistake in order to understand the limitations of each test. Hence, we define the *power function* of a hypothesis test with reject region R as the function $\beta : \Theta \to [0,1]$ such that

$$\beta(\theta) = \mathbb{P}_\theta(X \in R). \tag{2.42}$$

With the power function, we can compare different tests from the point of view of the type I error. Going back to **Example 1**, in Figure 2.7, we see that for each point in Θ_0 the test $te_{0.022}$ has a larger probability of making the type I error than $te_{0.132}$. Hence, *based only on this type of error*, we say that $te_{0.132}$ is a better test than $te_{0.022}$. This informal consideration can be better understood when we define the notion of *size*.

[5] This notion of convergence is defined as follows: let F_n and F denote the cumulative distribution functions of Z_n and Z respectively. By $Z_n \rightsquigarrow Z$ we mean that

$$\lim_{n\to\infty} F_n(u) = F(u),$$

for every $u \in \mathbb{R}$ for which F is continuous.

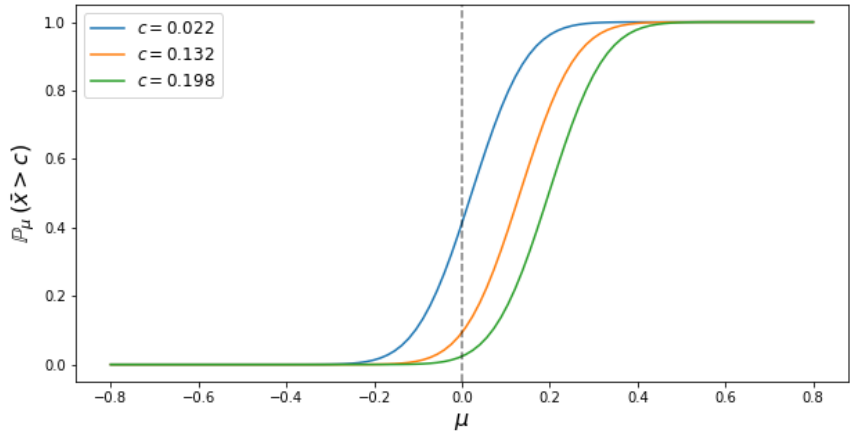

Figure 2.7: *Power function for the tests* $te_{0.022}$, $te_{0.132}$, *and* $te_{0.198}$. *In all cases,* $n = 100$ *and* $\sigma = 1$.

For $0 \leq \alpha \leq 1$ a test with power function $\beta(\theta)$ is *size* α if

$$\sup_{\theta \in \Theta_0} \beta(\theta) = \alpha. \tag{2.43}$$

For example, since $\sup_{\mu \leq 0} \beta(\mu) = \beta(0)$, the sizes of the tests $te_{0.022}$ and $te_{0.132}$ are 0.41 and 0.09, respectively.

In order to control the type I error, experimenters commonly choose the test based on its size α (also known as *significance level*). Common choices are 0.01 and 0.05, however the appropriate choice for a significance level depend on the application. In **Example 1**, for $\alpha \in [0, 1]$ we can define the test $te_{c(\alpha)}$ where

$$c(\alpha) = \frac{\sigma \phi^{-1}(1-\alpha)}{\sqrt{n}},$$

and $\phi^{-1}(1-\alpha)$ is the $1-\alpha$ quantile of the standard normal. Hence, the size of the test $te_{c(\alpha)}$ is given by

$$\begin{aligned}
\beta(0) &= \mathbb{P}_\mu(\bar{X} > c(\alpha)) \\
&= \mathbb{P}_\mu\left(Z_n > \frac{\sqrt{n}(c(\alpha) - \mu)}{\sigma}\right) \\
&= \mathbb{P}_\mu\left(Z_n > \phi^{-1}(1-\alpha)\right) \\
&\approx \mathbb{P}\left(Z > \phi^{-1}(1-\alpha)\right) \\
&= 1 - \phi(\phi^{-1}(1-\alpha)) \\
&= \alpha.
\end{aligned}$$

There is another type of test mistake refereed to *type II error*. This mistake happens when the test do not reject H_0 and H_1 is true. For $\theta \in \Theta_1$ the probability of type II error is given by

$$\mathbb{P}_\theta(X \notin R) = 1 - \beta(\theta). \tag{2.44}$$

In the Neyman-Pearson perspective, we first select one specific level of type I error α; then, we consider the class of size α tests and select the test that yields the smallest type II error probability. The Neyman-Pearson Lemma [Casella and Berger, 2002, Theorem 8.3.12] gives the conditions to find such test. On the other hand, in the Fisher perspective, we only control the probability of the type I error.

One method of reporting the results of a hypothesis test is to report the size α. The size of the test allow us to judge the importance of the test. If α is small, the decision to reject H_0 is convincing, but if α is large, the decision to reject H_0 is not very convincing because the test has a large probability of incorrectly making that decision. Another way of reporting the results of a hypothesis test is to report the value of a certain kind of test statistic called the *p-value*.

2.7.4 p-Value

A *p-value* $p(X_1, \ldots, X_n)$ is a test statistic satisfying $0 \leq p(x_1, \ldots, x_n) \leq 1$ for every sample point $(x_1, \ldots, x_n)$. We say that a p-value *is valid* if, for every $\theta \in \Theta_0$ and every $0 \leq \alpha \leq 1$,

$$\mathbb{P}_\theta(p(X_1, \ldots, X_n) \leq \alpha) \leq \alpha. \tag{2.45}$$

It should be noted that, if p is a valid p-value then the test

$$te(x_1, \ldots, x_n) = \begin{cases} \textbf{decision 0}, \ \textbf{if } p(x_1, \ldots, x_n) \leq \alpha, \\ \textbf{decision 1}, \ \textbf{otherwise} \ , \end{cases} \tag{2.46}$$

is a test of size α. Given a test of the form "decision 0 iff $T(X_1, \ldots, X_n) \geq c$", it is common to define the p-value as the probability (under H_0) of observing a value of the test statistic the same as or more extreme than what was observed. More formally, for every sample point $(x_1, \ldots, x_n)$ we define

$$p(x_1, \ldots, x_n) = \sup_{\theta \in \Theta_0} \mathbb{P}_\theta(T(X_1, \ldots, X_n) \geq T(x_1, \ldots, x_n)). \tag{2.47}$$

It can be proved that the definition (2.47) yields a valid p-value [Casella and Berger, 2002, Theorem 8.3.27]. For example, for the family of tests (2.38) we define the p-value for a sample point $(x_1, \ldots, x_n)$ as follows:

$$\begin{aligned} p(x_1, \ldots, x_n) &= \sup_{\mu \in \Theta_0} \mathbb{P}_\mu(\bar{X} > \bar{x}) \\ &= \mathbb{P}_0(\bar{X} > \bar{x}) \\ &= \mathbb{P}_0\left(Z_n > \frac{\sqrt{n}(\bar{x} - 0)}{\sigma}\right) \\ &\approx \mathbb{P}\left(Z > \frac{\sqrt{n}\bar{x}}{\sigma}\right) \\ &= 1 - \phi\left(\frac{\sqrt{n}\bar{x}}{\sigma}\right). \end{aligned} \tag{2.48}$$

2.7.5 Paired t-Test

In different occasions, *paired data* (*matched data*) appears when conducting an experiment. For example, we can measure the blood pressure of a person before and after an exercise session. If we repeat this measurement n times (using n different subjects), we obtain two samples: $A_1, \ldots, A_n$ and $B_1, \ldots, B_n$. Although the As are independent among themselves (the same goes for the Bs), there is a dependency in each pair (A_i, B_i) because both measurements are obtained from the *same person* in two distinct moments. *Paired test* (*paired difference test*) is a type of hypothesis testing that take this dependency into account when comparing the A and B samples.

Let $A \sim P_1$ and $B \sim P_2$ be two random variables where there is some dependency between them. We test that the expectation of the distributions P_1 and P_2 are the same, in other words, we test:

$$H_0 : \delta = 0 \text{ versus } H_1 : \delta \neq 0,$$

where

$$\delta = \mathbb{E}[A] - \mathbb{E}[B]. \tag{2.49}$$

For the samples $A_1, \ldots, A_n$ and $B_1, \ldots, B_n$, let $\hat{\delta}$ be an estimate of δ defined as:

$$\hat{\delta} = \frac{1}{n}\sum_{i=1}^{n} \hat{\delta}_i = \bar{A} - \bar{B}, \tag{2.50}$$

where $\hat{\delta}_i = A_i - B_i$ and $\bar{A}, \bar{B}$ are the mean of each sample. The standard deviation estimate of $\hat{\delta}$ (the estimated standard error) is given by:

$$\hat{se}(\hat{\delta}) = \frac{S}{\sqrt{n}}, \tag{2.51}$$

where

$$S = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{\delta}_i - \hat{\delta})^2}. \tag{2.52}$$

The test statistic is $|t|$ where

$$t \;=\; \frac{\hat{\delta} - 0}{\hat{se}(\hat{\delta})} \;=\; \frac{\sqrt{n}(\bar{A} - \bar{B})}{S}. \tag{2.53}$$

The *paired t-test* is the family of tests of the form "decision 0 iff $|t| > c$" for $c \in \mathbb{R}$. Given a significance level α, we can select a size α test in this family using p-values. As stated in (2.47), the p-value is defined for each observable test statistic $|\hat{t}|$ as follows:

$$p(\hat{t}) = \mathbb{P}_0(|t| > |\hat{t}|). \tag{2.54}$$

It should be noted that despite the dependency between each A_i and B_i, $\hat{\delta}_1, \ldots, \hat{\delta}_n$ are n independent and identically distributed (IID) data points from some distribution F. Thus, if we assume the null hypothesis (H_0), we have that

$$\begin{aligned}\mathbb{E}[\hat{\delta}] &= \mathbb{E}[\bar{A}] - \mathbb{E}[\bar{B}] \\ &= \mathbb{E}[A] - \mathbb{E}[B] \\ &= 0.\end{aligned} \tag{2.55}$$

By a version of the Central Limit Theorem [Wasserman, 2010, Theorem 5.10]:

$$t = \frac{\hat{\delta} - \mathbb{E}[\hat{\delta}]}{\hat{se}(\hat{\delta})} \rightsquigarrow Z, \tag{2.56}$$

where $Z \sim \mathcal{N}(0, 1)$, in other words, the estimator $\hat{\delta}$ is *asymptotically Normal*. This means that we can approximate the p-value (2.54) as follows:

$$\begin{aligned}p(\hat{t}) &= \mathbb{P}_0(|t| > |\hat{t}|) \\ &\approx \mathbb{P}(|Z| > |\hat{t}|) \\ &= \mathbb{P}(-|\hat{t}| < Z) + \mathbb{P}(Z > |\hat{t}|) \\ &= 2\phi(-|\hat{t}|).\end{aligned} \tag{2.57}$$

Hence, the following decision procedure:

$$te(\hat{t}) = \begin{cases} \textbf{decision 0}, \ \textbf{if } 2\phi(-|\hat{t}|) \leq \alpha, \\ \textbf{decision 1}, \ \textbf{otherwise} \ , \end{cases} \tag{2.58}$$

is a test of size α.

2.7.6 Bootstrap Hypothesis Test

Bootstrap is a technique to calculate measures of uncertainty. The main idea behind this method is based on creating new simulated data through resampling the data at hand, and using these new samples (*bootstrap samples*) to infer about the population characteristics.

In hypothesis testing we can use the bootstrap method to estimate the distribution of a test statistic under the null hypothesis. We use this technique instead of asymptotic theory (as we have been doing so far) because in certain scenarios, the presuppositions of this theory are not satisfied [Fisher and Hall, 1990].

In the case of the paired t-test, we use the available paired data $\{(A_i, B_i)\}$ to create $\mathcal{S} \in \mathbb{N}$ bootstrap samples. Each bootstrap sample is created in a way that reflects the null hypothesis. One strategy for doing so is the following: we draw n observations with replacement from $\{(A_i, B_i)\}$ such that each simulated observation is either (A_j, B_j) or (B_j, A_j), with probability $1/2$, for some $j \in \{1, \ldots, n\}$ [Konietschke and Pauly, 2014].

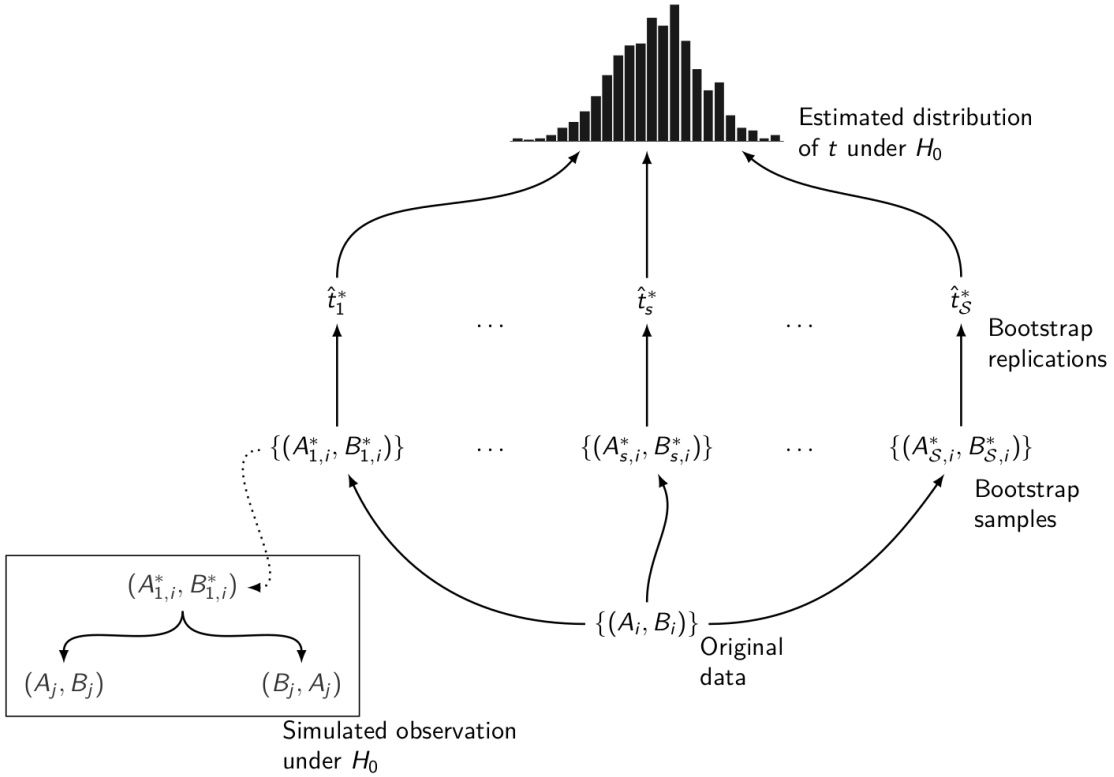

Figure 2.8: *Schematic of the bootstrap process for the paired t-test. We generate $\mathcal{S}$ bootstrap samples forcing them to satisfy the null hypothesis, then we calculate the test statistic for each sample and obtain the empirical distribution of this statistic.*

At the end of the resampling process, we obtain the new samples $\{(A^*_{1,i}, B^*_{1,i})\}, \ldots, \{(A^*_{\mathcal{S},i}, B^*_{\mathcal{S},i})\}$ and for each one we compute the *bootstrap replication* of the test statistic: $\hat{t}^*_1, \ldots, \hat{t}^*_{\mathcal{S}}$. With these new values we can estimate the distribution of t under H_0 (Figure 2.8). Based on the empirical distribution of the bootstrap replications, for each observable test statistic $\hat{t}$ we calculate the probability of obtaining a test statistic at least as extreme as $\hat{t}$ (the p-value). If we assume that t is symmetrically distributed around zero, we define the p-value as the symmetric bootstrap p-value:

$$p(\hat{t}) = \frac{1}{\mathcal{S}} \sum_{s=1}^{\mathcal{S}} I(|t_s^*| > |\hat{t}|). \tag{2.59}$$

If we are not willing to make the symmetry assumption, we can instead use the equal-tail bootstrap p-value:

$$p(\hat{t}) = 2 \min \left(\frac{1}{\mathcal{S}} \sum_{s=1}^{\mathcal{S}} I(t_s^* \leq \hat{t}),\ \frac{1}{\mathcal{S}} \sum_{s=1}^{\mathcal{S}} I(t_s^* > \hat{t}) \right). \tag{2.60}$$

The two possibilities are valid methods of computing bootstrap p-values [MacKinnon, 2009]. Using these new p-values we define the bootstrap test:

$$te(\hat{t}) = \begin{cases} \textbf{decision 0}, \ \textbf{if } p_{boot}(\hat{t}) \leq \alpha, \\ \textbf{decision 1}, \ \textbf{otherwise} \ , \end{cases} \tag{2.61}$$

where p_{boot} is either (2.59) or (2.60).

Chapter 3

Structural Inference

At first glance, the NLI task appears to be a simple text classification exercise: we need to predict a label $Y \in \{entailment, neutral, contradiction\}$ from a pair of text inputs. However, when we investigate what facts are responsible in determining the value of Y, we see that complex linguistic phenomena are at play in this task. Some entailment circumstances are the following:

- *Hyperonym Relation*

  `a woman won the lottery` entails `a person won the lottery`.

- *Common Knowledge*

  `a woman from Kyoto won the lottery` entails `a woman from Japan won the lottery`.

- *Relative Clauses*

  `a woman who became the youngster winner of the lottery has spoken out against him` entails `a woman won the lottery`.

Our first investigation will be focused on NLI observations that can be gathered under the label *structural inference*. By structural inference we refer to any inference phenomenon caused by the use of logical connectives in natural language. These types of connectives depend only on the form of the sentences and not on the meaning of the specific verbs, nouns and adjectives being used. For example, take the sentence pair:

$P =$ `Jenny and Sally play with my daughter`.

$H =$ `Jenny plays with my daughter`.

Clearly, P entails H. Moreover, P still entails H if we substitute in both sentences `Jenny` by `Carl` or even substitute the verbs `play` / `plays` by `sing` / `sings`. However, if we substitute `and` by `or`, the entailment relation is disrupted. What causes the entailment in this example is the position of the arguments together with the connective `and`.

This chapter is centered on the notion of structural inference. In Sections 3.1, 3.2 and 3.3 we explain how one can use formal logic to construct a synthetic NLI dataset based on logical connectives. In Section 3.4 we offer a new contradiction detection dataset where the contradiction examples do not depend on the presence of negative words, but they arise by the use of specific connectives. Together with such dataset, we evaluate two kinds of deep learning models that implicitly exploit language structure: recurrent models and the transformer network BERT. We also show a successful case of cross-lingual transfer learning between English and Portuguese. In Section 3.5 we define a new task based on inference generation and we compare the performance of the machine learning models in the usual NLI task and in the proposed generation task. In conclusion, in Section 3.6, we discuss the advantages and disadvantages of synthetic datasets.

3.1 Template Language

The *template language* is a formal language used to generate NLI instances. This language is composed of two basic sets: Pe, Pl and three binary relations $V(x, y)$, $x > y$, and $x \geq y$. The semantic motivation behind this choice is that we are trying to model a simplistic universe. The sets Pe and Pl stand for a generic set of people and places, respectively. Similarly, the intended meaning for the binary relations $V(x, y)$, $x > y$, $x \geq y$ are x `has visited` y, x `is taller than` y and x `is as tall as` y, respectively.

In order to increase the expressivity of this language we add logical connectives to it. Since the goal is the correspondence between the template language and natural language, we add only logical operators that correspond to a linguistic expression in most languages. Hence, for each new connective we indicate a possible translation in English.

Boolean Connectives. The most basic logical operators are the Boolean connectives. Although there is a family of different kinds of connectives in the field of formal logic, they do not appear in natural language with the same frequency. The logical operators most used in the English language (and many others) are $\neg$ (`not`), $\wedge$ (`and`) and $\vee$ (`or`) [van Wijk, 2006]. These operators allow us to combine the simple propositions based on binary relations in a variety of ways. For example, the formula $V(x_1, y_1) \wedge \neg V(x_2, y_2)$ describe the domain where x_1 is V-related to y_1 and the same relation does not hold between x_2 and y_2. By assigning proper names to the variables, this formula can be translated to English as the sentence `Felix has visited Bolivia and Bruce didn't visit Ecuador`.

Quantifiers. In order to express properties about individuals, we add the quantifiers $\forall$ and $\exists$ (`every` and `some`, respectively). With this addition we can express either properties that apply for all entities in the domain or properties that hold only for non-specific entities in the universe. Sentences of the form $\forall x(x > y)$ (`Everyone is taller than John`) and $\exists x(x \geq y)$ (`Someone is as tall as Henry`). We can also add restricted quantifiers, i.e., quantifiers that range over an specific sub-domain (the sets Pe and Pl). Hence, $\forall x \in Pe$ should be interpreted as `every person`, and $\forall x \in Pl$ should be read as `every place`. A similar interpretation holds for $\exists$. With this addition, we have formulas of the form $(\exists x \in Pe)(\forall y \in Pl)V(x, y)$ (`Some person has visited every place`).

Equality and Definite Description. By *definite description* we mean expression of the form `the queen of England`, `the best soccer player in 2019`, `the first hedge fund`, etc. These kind of expressions can be seen as a type of quantification. In formal languages, we add this quantification by including the operator ι to perform description and the equality relation $=$. This inclusion allow us to write formulas that create different references for the same individual. For example, $x_1 = \iota y V(x_2, y)$ (`John is the person visited by Henry`).

Counting Operators. The standard quantifiers $\forall$ and $\exists$ are just two instances of a broader family of operators called *generalised quantifiers*. Inside this family there are set of quantifiers specialized in expressing properties about a precise realization of the universe. Instead of the broader notion of existence conveyed by $\exists$, we can introduce a class of quantifiers that denote a specific quantity of individuals that have some property. These types of operators are called *counting quantifiers*. The counting operators can describe an exact number $\exists_{=n} x$ (`there are exact` n x`'s`), or they can given an lower bound $\exists_{\geq n} x$ (`there are at least` n x`'s`). Similar to $\exists$, we can also restrict these new quantifiers to some sub-domain; for example, $(\exists_{=3} y \in Pl)V(x_1, y)$ (`Philip has visited only three places`).

The combination of all logical operators yields a language complex enough to describe a variety of facts even when the universe (the set $Pe \cup Pl$) is a small one. The possible formulas range from basic concatenation of facts:

$$V(x_1, x_2) \wedge V(x_3, x_2) \wedge V(x_4, x_2)$$
`Henry,Felix and Bruce have visited John,`

to complex description of the entities in the universe:

$$x_1 = \iota x\,((\forall y \in Pe)(\neg((\exists_{=2} z \in Pl)V(y, z)) \vee x \geq y))$$
`John is the tallest person that has visited exactly two places.`

It should be noted that the template language is a fragment of *first-order logic*. For this reason, we have offered only an informal presentation of the formation rules of this language. A rigorous formulation of those rules can be found in any logic textbook [da Silva et al., 2006, Shoenfield, 1967].

3.2 Logical Rules and Templates

In order to make this language useful for creating NLI datasets, we should stipulate a method of obtaining examples of entailment, contradiction and neutral pairs. One simple strategy is the one based on *inference rules*: principles that describe how to introduce and eliminate a connective according to the correct reasoning. Since these rules govern the use of the logical operators, we can even say that they define the *meaning* of those connectives [Martin-Löf, 1996].

A logical rule is written as a pair: a set of premises an a conclusion. For example, the rules for the introduction of the connectives $\wedge$ and $\exists$ can be written as follows:

$$\frac{V(x_1, y_1) \quad V(x_2, y_2)}{V(x_1, y_1) \wedge V(x_2, y_2)}\ (\wedge_I), \tag{3.1}$$

$$\frac{V(x_1, x_2)}{\exists x V(x, x_2)}\ (\exists_I). \tag{3.2}$$

These rules are the formal expression of very reasonable principles: $\wedge_I$ states that if both premises are true, then the concatenation of them by the $\wedge$ operator forms a true formula; $\exists_I$ states that if $V(x_1, x_2)$ is true for a particular individual x_1, then it is the case that exists some individual x in the domain that has the property $V(x, x_2)$. These are only a couple of rules, the complete set of rules can be found in the work by Troelstra and Schwichtenberg [1996].

Logical rules are *truth preserving operations*, i.e., the set of premises *entails* the conclusion. Hence, we can translate the formulas that occur in a logical rule to English and obtain an example of entailment in natural language (we take the concatenation of premises as a singe sentence P and the conclusion as the sentence H). For example, using one specific translation of the variables, the rule $(\wedge_I)$ generates the pair:

$P =$ `Felix has visited Paris, John has visited Paris.`

$H =$ `Felix and John have visited Paris.`

And the rule $(\exists_I)$ produces:

$P =$ `Felix has visited John.`

$H =$ `Someone has visited John.`

It is also possible to use the same set of rules to produce contradiction examples. Since the premises entail the conclusion, if we negate the conclusion appearing in a logical rule we can produce contradictions. For example, by using the $\wedge_I$ rule we get:

$P =$ `Felix has visited John, Gerald has visited John.`

$H =$ `Felix didn't visit John or Gerald didn't visit John.`

The generation of neutral examples can be done by employing the *Interpolation Theorem* [Shoenfield, 1967, p. 80]. Let $\boldsymbol{p}, \boldsymbol{h}$ and $\boldsymbol{q}$ be variables for formulas, and let $L(\boldsymbol{p})$ be the set of non-logical symbols occurring in $\boldsymbol{p}$. This theorem states that if $\boldsymbol{p}$ entails $\boldsymbol{h}$, then there is some formula $\boldsymbol{q}$ such that $L(\boldsymbol{q}) \subset L(\boldsymbol{p}) \cap L(\boldsymbol{h})$, $\boldsymbol{p}$ entails $\boldsymbol{q}$ and $\boldsymbol{q}$ entails $\boldsymbol{h}$. It is easy to see that for the particular case where $\boldsymbol{p}$ and $\boldsymbol{h}$ are formulas that the true value depends only on the interpretation[1], if $L(\boldsymbol{p}) \cap L(\boldsymbol{h}) = \emptyset$, then neither $\boldsymbol{p}$ entails $\boldsymbol{h}$ nor $\boldsymbol{h}$ entails $\boldsymbol{p}$. Thus, if we modify a logical rule such that there is no variable intersection between premise and conclusion we can produce neutral sentence pairs (as long as the truth value of the formulas depends only on the interpretation). For example, by altering the rule $(\exists_I)$ we can obtain:

$P =$ `Felix has visited Gerald.`

$H =$ `Someone has visited John.`

We call a *template* any pair of formulas that when translated to a natural language form a NLI observation. Using the example above, the following template is a modification of the rule $(\exists_I)$ that can generate neutral pairs (P, H):

$P = V(x_1, x_2)$

$H = \exists x V(x, x_3)$

$Y =$ *neutral*.

In other words, templates are just schemas for constructing different types of NLI observations. Since each template is associated with a logical rule, we have total control on the grounding for the logical relationship of each NLI observation created by this process.

3.3 Translation

One additional advantage in using a template language is the possibility of translating it to different natural languages. The linguistic requirements to perform a translation are few: it is needed a translation of the binary relations, a name assignment for the elements in $Pe \cup Pl$ and an interpretation of the logical connectives. Ideally this translation process should be as subtle as possible. For example, instead of translating the formula $V(x_1, x_2) \wedge V(x_1, x_3)$ as `Felix has visited Gerald and Felix has visited Marty` we can use the condensed version `Felix has visited Gerald and Marty`.

More formally, we can think about translations as a *realisation* of the template language, i.e., a function r mapping Pe and Pl to nouns such that $r(Pe) \cap r(Pl) = \emptyset$; it also maps the relation symbols and logic operators to corresponding forms in some natural language. It should be noted that since the requirements for translating the template language to a natural language is low, we can construct the same type of dataset in different languages. This is useful to compare model performance in multiple linguistic scenarios; and, at the same time, this approach can contribute to dataset creation for low-resource languages. For example, the rule $(\exists_I)$ can be used to obtain entailment pairs in Spanish:

$P =$ `Felix visitó a Juan.`

$H =$ `Alguien visitó a Juan.`

[1] Satisfiable formulas that are not valid, to use the technical term.

Or in Portuguese:

$P =$ `Felix visitou João.`

$H =$ `Alguém visitou João.`

We utilize this translation function to divide a NLI dataset into different portions. After fixing one target language, say English, we take three different realizations that map the template language into the target language r_1, r_2 and r_3 such that

$$\bigcup_{\substack{i,j\in\{1,2,3\}\\ i\neq j}} r_i(Pe) \cap r_j(Pe) \;=\; \bigcup_{\substack{i,j\in\{1,2,3\}\\ i\neq j}} r_i(Pl) \cap r_j(Pl) \;=\; \emptyset. \tag{3.3}$$

Thus, the image of each realization can be defined as the training, validation and test portions of the data. This strategy of splitting the dataset is a way to guarantee that the noun assignment does not affect the performance of a text classifier. In order to highlight this splitting procedure, we use r_{train}, $r_{validation}$ and r_{test} to denote the realizations r_1, r_2 and r_3.

3.4 Analysis I: Contradiction Detection

In this section we describe the analysis published as a paper by Salvatore et al. [2019a]. It is our first work based on synthetic data.

3.4.1 A Logical-Based Corpus for Cross-Lingual Evaluation

In this analysis, we present a collection of small datasets designed to measure the competence of detecting contradictions in structural inferences. To perform a cross-lingual comparison, we have decided to perform the analysis both in English and in Portuguese. The choice to focus on the *contradiction detection* (CD) task was done because it is harder for an average crowdsourcing annotator to create examples of contradictions without excessively relying on the same patterns [Gururangan et al., 2018]. At the same time, CD has practical importance since it can be used to improve consistency in real case applications, such as chatbots [Welleck et al., 2019].

We choose to focus on CD based on structural inference because we have detected that the current datasets are not appropriately addressing this particular problem. In an experiment, we have verified that *structural information* is not needed to achieve an accuracy higher than random guess (50%) on the benchmark datasets. To check this fact we have first transformed the SNLI and MNLI datasets into a CD task and we have evaluated the new version of this data using a simple BOW classifier (a type of model that uses only the minimum structural information from a text input).[2] The accuracy was significantly higher than the random classifier, 63.9% and 61.9% for SNLI and MNLI, respectively[3]. Even the recent dataset focusing on contradiction, Dialog NLI [Welleck et al., 2019], presents the same issue. The same BOW model achieves an accuracy of 76.2% on this dataset.

In this analysis, we propose a CD dataset divided by tasks. What defines the difference from one task to another is the introduction of logical operators, and the underlining logical rules that govern those operators. Using this new dataset, we evaluate two kinds of deep learning models that implicitly exploit language structure: recurrent models and the transformer network BERT [Devlin et al., 2019].

[2]Clearly, any NLI dataset can be transformed in a CD task: the transformation is done by converting all instances of *entailment* and *neutral* into *non-contradiction*, and by balancing the classes in the new dataset.

[3]For the MNLI it was used the combination of the matched and miss-matched development data.

3.4.2 A Dataset of Contradictions

For the sake of brevity, we give only a brief overview of each task. The detailed list of all templates used to produce each task is displayed in the Appendix A (and the full dataset in both languages, together with the code to generate it can be found online, [Salvatore, 2019a]).

Task 1: Simple Negation. The premise P is a collection of facts about some agents visiting different places. The hypothesis H can be either a negation of one fact stated in P, or a new fact not related to P. The number of facts occurring on P vary from two to twelve. For example,

$P =$ `Charles has visited Chile, Joe has visited Japan, Henry didn't visit France.`

$H =$ `Charles didn't visit Japan.`

$Y =$ *non-contradiction.*

It should be noted, that in this task what causes the contradiction is not the occurrence of the negation (`didn't`), it is the role of the agents and places that appear in the premise and the hypothesis.

Task 2: Boolean Coordination. In this task, the premise P describes a group of agents traveling to a place (or a single agent traveling to multiple places). The new information H can state that one of the agents did not travel to a mentioned place. For example,

$P =$ `Felix, Ronnie, and Tyler have visited Bolivia.`

$H =$ `Tyler didn't visit Bolivia.`

$Y =$ *contradiction.*

A non-contradictory observation in this task is formed by a hypothesis H that contains a new fact (`Bruce didn't visit Bolivia`).

Task 3: Quantification. Here, we construct contradictory examples that explicitly exploit the difference between the two basic entities, people and places. The premise P states a general fact about all people, and the hypothesis H can be the negation of one particular instance of P, or a fact that does not violate P. For example,

$P =$ `Everyone has visited every place.`

$H =$ `Timothy didn't visit Anthony.`

$Y =$ *non-contradiction.*

Task 4: Definite Description. In this task, we use the premise to describe one agent using an specific property, and we formulate the hypothesis in a way that it can violate the description. For example,

$P =$ `Carlos is the person that has visited every place, Carlos has visited John.`

$H =$ `Carlos didn't visit Germany.`

$Y =$ *contradiction.*

To formulate an example of non-contradiction, the hypothesis can be a fact that do not speak about the agent being described in the premise; for example, `John did not visit Germany`.

Task 5: Comparatives. By adding the comparative relation $>$ in the language, we are interested in investigating if the model can recognise a basic property related to comparison: transitivity. The premise P is composed of a collection of simple facts $x_1 > x_2, x_2 > x_3$ (`Francis is taller than Joe, Joe is taller than Ryan`). Assuming the transitivity of $>$, the hypothesis can be either a consequence of P, $x_1 > x_3$ (`Francis is taller than Ryan`), or a fact that violates the transitivity property, $x_3 > x_1$ (`Ryan is taller than Francis`). The number of facts that appear on P varies from four to ten. It should be noted that the contradictions in this task do not use any negative words.

Task 6: Counting. To analyze the counting competence of models we create contradictions based only on the counting operators. The premise P is composed of facts that enumerate an agent visiting an specific number of people and places. The hypothesis H can be either a fact consistent with the number appearing in the premise or a conjunction of facts that contradicts P. For example,

$P =$ `Philip has visited only three places and only two people.`

$H =$ `Philip has visited John, Carla, and Bruce.`

$Y =$ *contradiction.*

We have added counting quantifiers corresponding to numbers from one to thirty.

Task 7: Mixed. In order to guarantee variability, we have created a dataset composed of different samples from the previous tasks.

Basic statistics for the English and Portuguese realisations of all tasks can be found in Table 3.1.

Task	Vocabulary size	Vocabulary intersection	Mean input length	Max input length
1 (Eng)	3561	77	230.6	459
2 (Eng)	4117	128	151.4	343
3 (Eng)	3117	70	101.5	329
4 (Eng)	1878	62	100.81	134
5 (Eng)	1311	25	208.8	377
6 (Eng)	3900	150	168.4	468
7 (Eng)	3775	162	160.6	466
1 (Pt)	7762	254	209.4	445
2 (Pt)	9990	393	148.5	388
3 (Pt)	5930	212	102.7	395
4 (Pt)	5540	135	91.8	140
5 (Pt)	5970	114	235.2	462
6 (Pt)	9535	386	87.8	531
7 (Pt)	8880	391	159.9	487

Table 3.1: *Task description. Column 1 presents two realizations of the described tasks - one in English (Eng) and the other in Portuguese (Pt). Column 2 presents the vocabulary size for the task. Column 3 presents the number of words that occurs both in the training and test data. Column 4 presents the average length in words of the input text (the concatenation of P and H). Column 5 presents the maximum length of the input text.*

Since we are using a large number of facts in the premise, the input text is longer than the ones presented in average NLI datasets (for comparison sake, the mean input length of the SNLI and MNLI datasets is 20.3 and 30, respectively).

3.4.3 Models

Baseline The baseline model is a random forest classifier that models the input text, the concatenation of P and H, using the BOW representation. Since we have constructed the dataset centered on the notion of structure-based contradictions, we believe that the baseline should perform slightly better than random. At the same time, by using such baseline, we can certify whether the proposed tasks are indeed requiring structural knowledge.

Recurrent Models. One popular family of models in the NLP field specialised in modelling sequential data is the recurrent neural networks and its variations, LSTM and GRU. We consider both the standard and the bidirectional variants of this family of models. As input for these models, we use the concatenation of the premise and hypothesis as a single sentence.

Traditional multilayer recurrent models are not the best choice to improve the benchmark on NLI [Glockner et al., 2018]. However, in recent works, it has been reported that recurrent models achieve a better performance than Transformer-based models to capture structural patterns for logical inference [Evans et al., 2018, Tran et al., 2018]. We want to investigate if the same result can be achieved using our dataset.

Transformer-based Models. A recent non-recurrent family of neural models known as transformer networks was introduced by Vaswani et al. [2017]. Different from the recurrent models that recursively summarizes all previous input into a single representation, the transformer network employes a self-attention mechanism to directly attend to all previous inputs. Although, by performing regular training using this architecture alone we do not see surprising results in inference prediction [Evans et al., 2018, Tran et al., 2018], when we pre-trained a transformer network in the language modeling task and fine-tuned afterwards on an inference task we see a significant improvement [Devlin et al., 2019].

Among the different transformer-based models we will focus our analysis on the multilayer bidirectional architecture known as BERT [Devlin et al., 2019]. This bidirectional model, pre-trained as a masked language model and as a next sentence predictor, has two versions: $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$. The difference lies in the size of each architecture, the number of layers and self-attention heads. Since $\text{BERT}_{\text{LARGE}}$ is unstable on small datasets [Devlin et al., 2019] we have used only the $\text{BERT}_{\text{BASE}}$ model. Thus, throughout this section "BERT" is used to denote "$\text{BERT}_{\text{BASE}}$".

The strategy to perform NLI classification using BERT is the same as the one presented in Section 2.6.2: together with the pair (P, H) we add new special tokens `[CLS]` (classification token) and `[SEP]` (sentence separator). Hence, the textual input is the result of the concatenation: `[CLS]` P `[SEP]` H `[SEP]`. After we obtain the vector representation of the `[CLS]` token, we pass it through a classification layer to obtain the prediction class (*contradiction* / *non-contradiction*). We fine-tune the model for the CD task in a standard way, the original weights are co-trained with the weights from the new layer.

By comparing BERT with other models we are not only comparing different architectures but different techniques of training. The baseline model uses no source of additional information. The recurrent models use only a soft version of transfer learning with fine-tuning of pre-trained embeddings (the fine-tuning of one layer only). On the other hand, BERT is pre-trained on a large corpus as a language model. It is expected that this pre-training step helps the model to capture some general properties of language [Howard and Ruder, 2018]. Since the tasks that we propose are basic and cover very specific aspects of reasoning, we can use it to evaluate which properties are being learned in the pre-training phase.

3.4.4 Evaluation

Model evaluation is performed as usual: for each CD task and model, we estimate the model's parameters on the training portion of the data and evaluate the model based on its classification accuracy on the test section of the dataset.

However, the simplicity of the tasks has motivated us to use transfer-learning in an additional

manner: instead of simply using the multilingual version of BERT[4] and fine-tune it on the Portuguese version of the tasks, *we have decided to perform a supplementary evaluation by checking the possibility of transferring structural knowledge from high-resource languages (English / Chinese) to Portuguese.*

Hence, we use the Portuguese corpus and the different pre-trained weights of the BERT model to perform a cross-lingual evaluation. This can be done because for each pre-trained model there is a tokenizer that transforms the Portuguese input into a collection of tokens that the model can process. We have decided to use the regular version of BERT trained on an English corpus (BERT_{eng}), the already mentioned Multilingual BERT (BERT_{mult}), and the version of the BERT model trained on a Chinese corpus (BERT_{chi}).

We hypothesize that *most structural patterns learned by the model in English can be transferred to Portuguese*. By the same reasoning, we believe that BERT_{chi} should perform poorly. Not only the tokenizer associated to BERT_{chi} will add noise to the input text, but also Portuguese and Chinese are grammatically different; for example, the latter is overwhelmingly right-branching while the former is more mixed [Levy and Manning, 2003].

3.4.5 Experimental Settings

Given the above considerations, four research questions arose:

(i) *How the different models perform on the proposed tasks?*

(ii) *How much each model relies on the occurrence of non-logical words?*

(iii) *Can cross-lingual transfer learning be successfully used for the Portuguese realization of the tasks?*

(iv) *Is the dataset biased? Are the models learning some unexpected text pattern?*

To answer those questions, we evaluated the models performance in four different ways:

(i) Each model was trained and evaluated on different proportions of the dataset, such that there is no noun intersection. i.e., $r_{train}(Pe) \cap r_{test}(Pe) = \emptyset$ and $r_{train}(Pl) \cap r_{test}(Pl) = \emptyset$.

(ii) We have trained the models on a version of the dataset where we allow full intersection of the train and test vocabulary, i.e., $r_{train}(Pe) = r_{test}(Pe)$ and $r_{train}(Pl) = r_{test}(Pl)$.

(iii) For the Portuguese corpus, we have fine-tuned the pre-trained models mentioned previously: BERT_{eng}, BERT_{mult}, and BERT_{chi}.

(iv) We have trained the best model from (i) on the following modified versions of the dataset:

  (a) *Noise label* - each pair (P, H) is unchanged but we randomly labeled the pair as contradiction or non-contradiction.

  (b) *Premise only* - we keep the labels the same and omit the hypothesis H.

  (c) *Hypothesis only* - the premise P is removed, but the labels remain intact.

3.4.6 Implementation

All deep learning architectures were implemented using the Pytorch library [Paszke et al., 2017]. To make use of the pre-trained version of BERT we have based our implementation on the HuggingFace transformer library [Wolf et al., 2019]. All the code for the experiments is public available [Salvatore, 2019a].

[4]Multilingual BERT is a model trained on the concatenation of the entire Wikipedia from 100 languages, Portuguese included. https://github.com/google-research/bert/blob/master/multilingual.md

The different recurrent architectures were optimized with Adam [Kingma and Ba, 2015]. We have used pre-trained word embedding from Glove [Pennington et al., 2014] and Fasttext [Joulin et al., 2017], we also used random initialized embeddings. We randomly searched across embedding dimensions in $\{10, \ldots, 500\}$, hidden layer size of the recurrent model in $\{10, \ldots, 500\}$, number of recurrent layer in $\{1, \ldots, 6\}$, learning rate in $[0, 1]$, dropout in $[0, 1]$ and batch sizes in $\{32, \ldots, 128\}$.

The hyperparameter search for BERT follows the one presented by Devlin et al. [2019] that uses Adam with learning rate warmup and linear decay. We randomly searched the learning rate in $[2 \cdot 10^{-5}, 5 \cdot 10^{-5}]$, batch sizes in $\{16, \ldots, 32\}$ and number of epochs in $\{3, 4\}$.

3.4.7 Results

(i) How the different models perform on the proposed tasks?

In most tasks, BERT$_{eng}$ presents a clear advantage when compared to all other models. Tasks 3 and 6 are the only ones where the difference in accuracy between BERT$_{eng}$ and the recurrent models is small, as can be seen in Table 3.2. Even when we look at BERT$_{eng}$'s results on the Portuguese corpus, which are slightly worse when compared to the English results, we still see a similar pattern.

Task	Baseline	RNN	GRU	LSTM	BERT
1 (Eng)	52.1	50.1	50.6	50.4	**99.8**
2 (Eng)	50.7	50.2	50.2	50.8	**100**
3 (Eng)	63.5	50.3	66.1	63.5	**90.5**
4 (Eng)	51.0	51.7	52.7	51.6	**100**
5 (Eng)	50.6	50.1	50.2	50.2	**100**
6 (Eng)	55.5	84.4	82.7	75.1	**87.5**
7 (Eng)	54.1	50.9	53.7	50.0	**94.6**
Average (Eng)	53.9	55.4	58.0	56.2	**96.1**
1 (Pt)	53.9	50.1	50.2	50.0	**99.9**
2 (Pt)	49.8	50.0	50.0	50.0	**99.9**
3 (Pt)	61.7	50.0	70.6	50.1	**78.7**
4 (Pt)	50.9	50.0	50.4	50.0	**100**
5 (Pt)	49.9	50.1	50.8	50.0	**99.8**
6 (Pt)	58.9	66.4	**79.7**	67.2	79.1
7 (Pt)	55.4	51.1	51.6	51.1	**82.7**
Average (Pt)	54.4	52.6	57.6	52.6	**91.4**

Table 3.2: *Results of the experiment (i). Test accuracy (%) for all models in the English (Eng) and Portuguese (Pt) corpora.*

Figure 3.1 shows that BERT$_{eng}$ is the only model improved by training on more data. All other models remain close to random independently of the amount of training data.

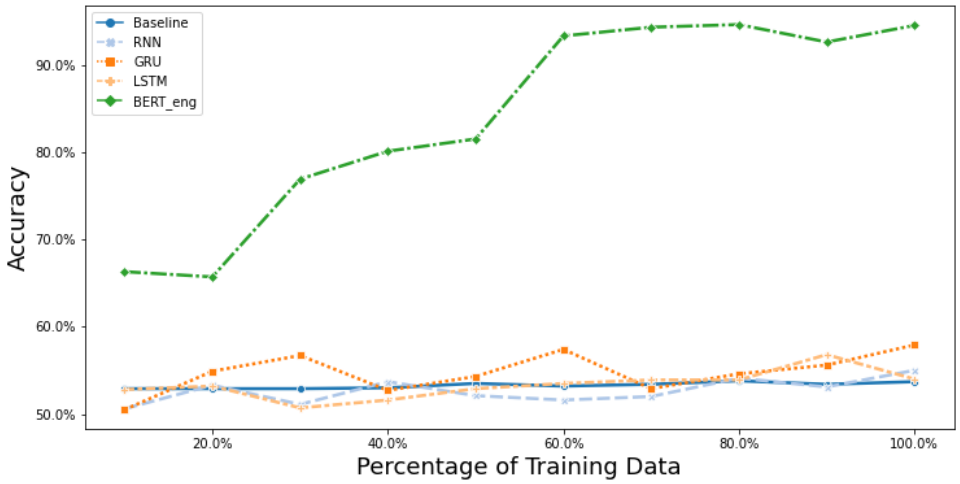

Figure 3.1: *Results of the experiment (i). In the x-axis we have different proportions of the training data used in training. The y-axis displays the average accuracy among all tasks (English corpus).*

Accuracy improvement over training size indicates the difference in difficulty of each task. On the one hand, Tasks 1, 2 and 4 are practically solved by BERT using only 40% examples of the training data (99.5%, 99.7%, 97.6% of test accuracy, respectively). On the other hand, the results for Tasks 3 and 6 remain below all tasks average, as seen in Figure 3.2.

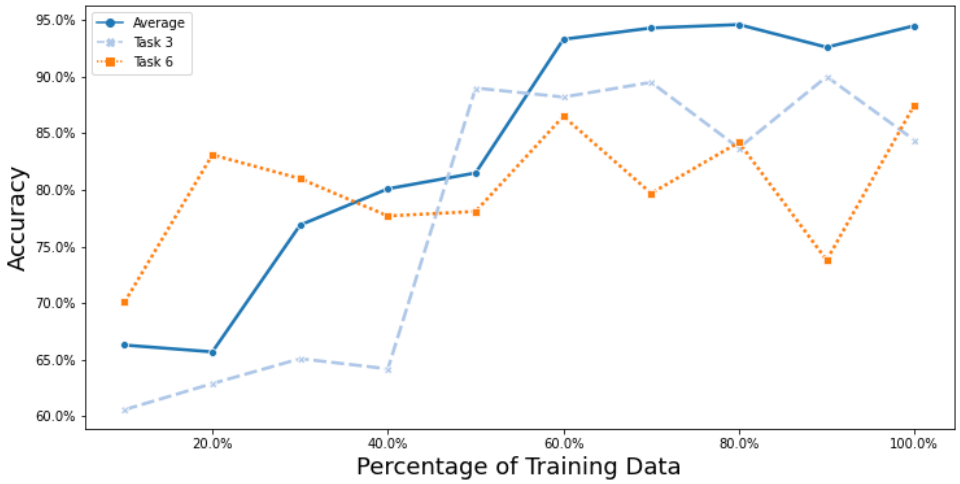

Figure 3.2: *Results of the experiment (i). In the x-axis we have different proportions of the training data used in training. The y-axis displays $BERT_{eng}$'s accuracy on different tasks (English corpus).*

(ii) How much each model relies on the occurrence of non-logical words?

With the full intersection of the vocabulary, experiment (ii), we have observed that the average accuracy improvement differs from model to model: baseline, GRU, $BERT_{eng}$, LSTM and RNN present an average improvement of 17.6%, 9.6%, 5.3%, 4.25%, 1.3%, respectively. The baseline is relying more on noun phrases than the neural models. However, it is hard to form any conclusive judgment about BERT and the recurrent models, more investigation is required.

(iii) Can cross-lingual transfer learning be successfully used for the Portuguese realization of the tasks?

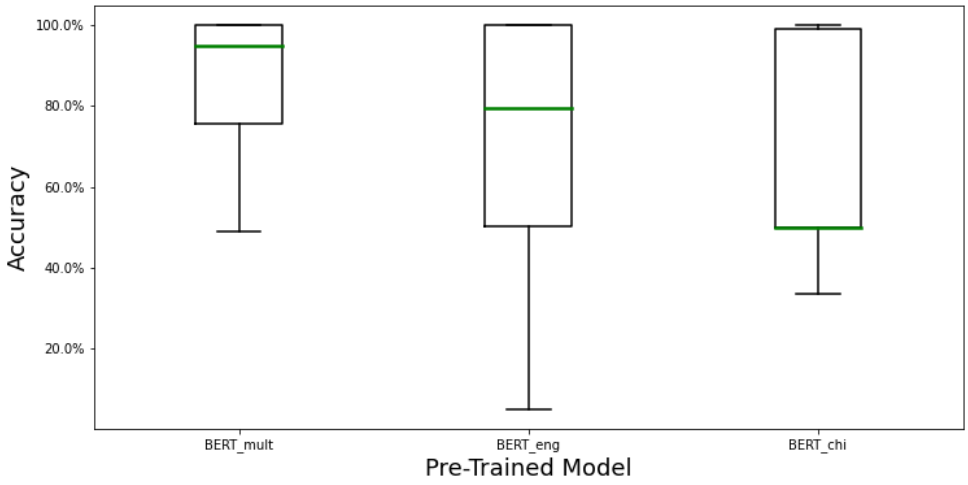

Figure 3.3: *Results of the experiment (iii). Accuracy distribution on all tasks for different pre-trained versions of the model BERT (Portuguese corpus).*

As expected, when we fine-tuned BERT_{multi} to the Portuguese version of the dataset we have observed an overall improvement. And, among the pre-trained models, BERT_{chi} was the weakest one (Figure 3.3).

The most notable improvements reside in Tasks 6 and 7, where we achieve a new accuracy of 87.4% and 92.3% respectively. Surprisingly, BERT_{chi} is able to solve some simple tasks, namely Tasks 1, 2 and 4. However, when this pre-trained model was trained on the mixed version of the dataset, Task 7, it repeatedly presented a random performance.

One of the most important features observed by evaluating the different pre-training models is that although BERT_{eng} and BERT_{mult} show a similar result on the Portuguese corpus, BERT_{eng} needs more data to improve its performance, as seen in Figure 3.4.

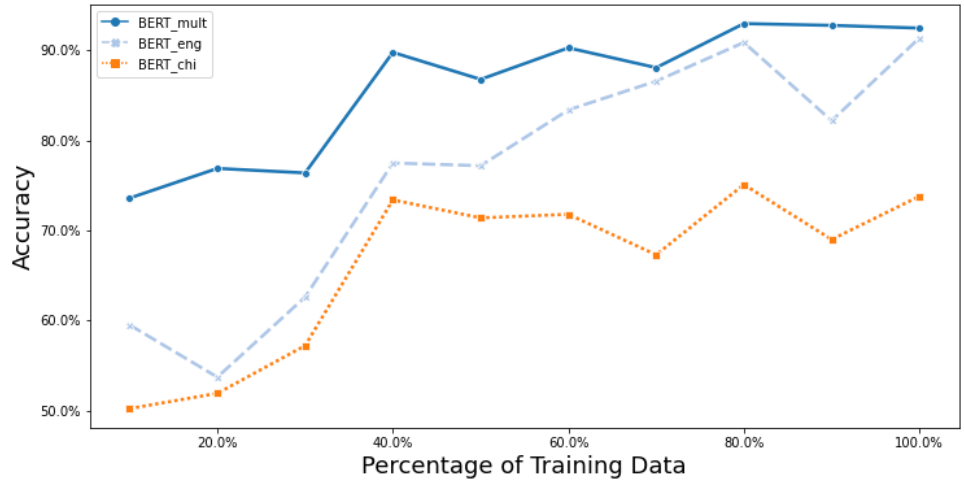

Figure 3.4: *Results of the experiment (iii). In the x-axis we have different proportions of the training data used in training. The y-axis displays the average accuracy among all tasks (Portuguese corpus).*

(iv) Is the dataset biased? Are the models learning some unexpected text pattern?

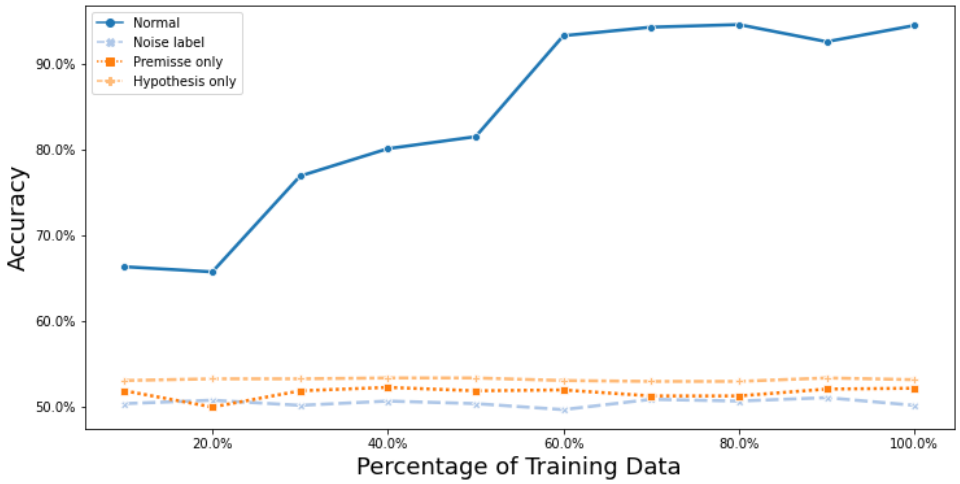

Figure 3.5: *Results of the experiment (iv). In the x-axis we have different proportions of the training data used in training. The y-axis displays the average accuracy among all tasks. All results presented in the figure are associated with the performance of the model BERT$_{eng}$' on different versions of the data (English corpus).*

By taking BERT$_{eng}$ as the best classifier, we have repeated the training using all the listed data modification techniques. The results, as shown in Figure 3.5, indicate that BERT$_{eng}$ is not memorizing random textual patterns, neither excessively relying on information that appears only in the premise P or the hypothesis H. When we applied it on these versions of the data, BERT$_{eng}$ behaves as a random classifier.

3.4.8 Discussion

The results presented above are similar to the ones reported by Goldberg [2019] : *transformer-based models like BERT can successfully capture syntactic regularities and logical patterns.*

These findings do not contradict the results reported by Evans et al. [2018] and Tran et al. [2018], because in both papers, the transformer models are trained from scratch, while here we have used models that were pre-trained on large datasets with the language model objective.

The results presented both in Table 3.2 and in Figures 3.3 and 3.4 *seem to confirm our initial hypothesis on the effectiveness of transfer learning in a cross-lingual fashion.* What has surprised us was the excellent results regarding Tasks 1, 2 and 4 when transferring structural knowledge from Chinese to Portuguese. We offer the following explanation for these results. Take for example this contradiction pair defined in Task 4:

$P = x_1$ `is the person that has visited everybody,` x_1 `has visited` x_3

$H = x_1$ `didn't visit` x_4.

$Y =$ *contradiction.*

By taking one possible Portuguese realization of the pair above and applying the different tokenizers we have the following strings:

1. Original sentence:

   $P =$ `Gabrielle é a pessoa que visitou todo mundo, Gabrielle visitou Luís.`

   $H =$ `Gabrielle não visitou Ianesis.`

2. Applying the multilingual tokenizer:

   $P =$ `gabrielle a pessoa que visito ##u todo mundo gabrielle visito ##u lu ##s`

   $H =$ `gabrielle no visito ##u ian ##esis`

3. Applying the English tokenizer:

   $P =$ `gabrielle a pe ##sso ##a que visit ##ou tod ##o mundo gabrielle visit ##ou lu ##s`

   $H =$ `gabrielle no visit ##ou ian ##esis`

4. Applying the Chinese tokenizer:

   $P =$ `ga ##b ##rie ##lle a pe ##ss ##oa q ##ue vi ##sit ##ou to ##do mu ##nd ##o ga ##b ##rie ##lle vi ##sit ##ou lu ##s`

   $H =$ `ga ##b ##rie ##lle no vi ##sit ##ou ian ##es ##is`

Although the Portuguese words are destroyed by the English and Chinese tokenizers, the model is still able to learn in the fine-tuning phase the *simple* structural pattern between the tokens. To find the contradiction in the example above, it is needed to check the occurrence of the same individual (Gabrielle) in the premise and hypothesis. Both in the multilingual and English version, this person is denoted by the token (`gabrielle`), and in the Chinese version the same entity can be found by observing the presence of four consecutive tokens (`ga ##b ##rie ##lle`).

This may explain why the counting task (Task 4) presents the highest difficulty for BERT. There is some structural grounding for finding contradictions in counting expressions, but to detect contradiction in all cases one must fully grasp the *meaning* of the multiple counting operators.

3.4.9 Analysis Conclusion

With the possibility of using pre-trained models we can successfully craft small datasets ($\sim$ 10K sentences) to perform fine grained analysis on machine learning models. In this analysis, we have presented a new dataset that is able to isolate a few competence issues regarding structural inference. It also allows us to bring to the surface some interesting comparisons between recurrent neural networks and pre-trained transformer-based models. As our results show, *compared to the recurrent models, BERT presents a considerable advantage in learning structural inference. The same result appears even when fine-tuned one version of the model that was not pre-trained on the target language.*

By the stratified nature of our dataset, we can pinpoint BERT's inference difficulties: *there is space for improving the model's counting understanding.* Hence, we can either craft a more realistic NLI dataset centered on the notion of counting or modify BERT's training to achieve better results in the counting task.

The results on cross-lingual transfer learning are stimulating. One possible area for future research is to check if the same results can be attainable using simple structural inferences that occur within complexes sentences. This can be done by carefully selecting sentence pairs in a cross-lingual NLI corpus (the corpus by Conneau et al. [2018] is a good candidate for this task).

3.5 Analysis II: Inference Generation

This section shows a subsequent analysis published in a paper by Salvatore et al. [2019b]. It uses the same building blocks as the previous analysis (the template language, inference rules and translation functions), however in this work we have decided to create an alternative NLI task. As seen in Section 3.2, the use of a particular logical connective yields the implication relation within

the pair (P, H). Hence, instead of classifying the logical relation of a given pair, we can imagine a new kind of text classification task *where the model should choose the correct connective to appear in a sentence in order to generate the entailment relation.* In this section, we present the proposal of this new task together with a brief analysis showing the difficulties of the deep-learning models in solving this new challenge.

3.5.1 A New Type of NLI Task

Since the proposal of the FRACAS project [The Fracas Consortium et al., 1996], we are used to formulating natural language inference as a classification problem. Given a pair of textual inputs P and H, we need to determine what is the logical relationship between them. It is fair to say that the NLI community has overemphasized the *inference perception* (IP) capability of machine learning models. Although IP has a lot of central applications on NLP such as automatic summarization and QA, it fails to connect with an important stream in the tradition of formal logic and proof theory. These fields do not concentrate on finding a function that can determine the logical relationship between P and H. Instead, the focus lies on how to generate a conclusion H from a premise P according to some established rules.

Here, we are proposing to shift the attention of the NLI community from inference perception to *inference generation* (IG). Instead of using the machine learning models to classify a pair of sentences, in this analysis, we propose a new task to evaluate how a neural language model can generate sentences according to logical rules. Together with this proposal we offer a synthetic dataset; and we also perform a first evaluation of this task using state-of-the-art models.

3.5.2 From Perception to Generation

It is the right moment to change from IP to IG: the new wave of pre-trained models [Devlin et al., 2019, Liu et al., 2019b] are challenging the NLI field. The large-scale popular IP datasets are close to being solved (the state-of-the-art results for SNLI, MNLI, and SciTail are 91.9%, 92.2%/91.9%, and 94.1%, respectively, as reported by Bowman et al. [2020], Liu et al. [2019b], Wang et al. [2020]). This may suggest that the inference problem in natural language has become trivial, but when we change how the current state-of-the-art models perform inference, we see entirely different results.

One example is the recent work from DeepMind by Saxton et al. [2019] where the authors developed a dataset for solving mathematical problems through symbolical manipulation. For example, a model receives as input a mathematical formula like $(x + 1)(2x + 3)$, and it is asked to generate the expansion of the polynomial. They have divided the mathematical reasoning task in 56 modules that involve different formal capacities; they also have trained a transformer and a LSTM model on those modules. Some mathematical skills, like polynomial expansion, were easy to master; others like factoring numbers into primes present a substantial difficulty for these models. When the best model was tested with real math exams for 16-year-old schoolchildren, it got only a score of 14/40 — indicating that there is a lot of space for improvement.

We are proposing a similar set of modules but centered on natural language deduction. Our focus is on informal logical reasoning: *the ability to use basic logical forms in everyday speech.* The different forms are defined by the use of some operators like Boolean coordination, quantifiers, definite description, and counting operators. We already have seen that using those operators to construct simple IP tasks and track the logical competence of different neural models is feasible. Here we want to investigate if the same kind of results can be carried over to the IG task and explore what kind of new models are necessary to accomplish this task.

3.5.3 Masked Inference

To test the generative capability of a transformer based model like BERT we developed the task of *masked inference* (MI) in analogy to the MLM task described by Devlin et al. [2019]. The MI task is defined as follows: in the training phase, we chose some tokens at random to be replaced

by the [MASK] token; the model is then trained to predict the missing token given the context; in the test phase, *we masked only the word that corresponds to a logical operator*; the model is then asked to predict the correct operator. For example, in the training phase we have the following observation:

$P =$ Joshua [MASK] visited Frank, Ricky [MASK] visited Jim.

$H =$ Joshua [MASK] visited Frank and Ricky [MASK] visited Jim.

$Y =$ has.

And in the test phase we have examples like:

$P =$ Lindsey has visited Jamaica, Patricia has visited Serbia.

$H =$ Lindsey has visited Jamaica [MASK] Patricia has visited Serbia.

$Y =$ and.

To generate the different pairs (P, H) we apply some well-known inference rules such that H is inferred from P. We divided the corpora into three modules: **Boolean coordination**, **Quantifiers** and **Counting**. it should be noted that by using the same logical rules we can create observations for both inference tasks: the standard form of NLI classification and the MI task. Hence, for each module we have two types of datasets.

The process used here to create a standard NLI dataset from logical rules is the same one described in Section 3.2. Thus, in what follows, we comment only on how to use templates to create observations from the MI task. For readability sake, we only offer an informal description of each template. The full description of the dataset in terms of the template language can be found in Appendix B.

3.5.4 Boolean Coordination

In this module the inference is centered on the connectives and and or ($\wedge$ and $\vee$, respectively) the model should be able to predict when to use each connective. There are only two types of rules in this module:

- Introduction of the conjunction

  $P =$ Ida didn't visit Senegal, Ethel has visited Clara.
  $H =$ Ida didn't visit Senegal [MASK] Ethel has visited Clara.
  $Y =$ and.

- Introduction of the disjuction

  $P =$ Neil has visited Kittery, Adam didn't visit Vincent.
  $H =$ Neil has visited Kittery [MASK] Adam has visited Daryl.
  $Y =$ or.

The difference between the two rules lies on the repetition of two facts that can occur both in P and H. The variability of the examples created by these rules reside not only on the nouns being used, the number of facts occurring in P can change, and the same facts can appear negated or not.

3.5.5 Quantifier Reasoning

For quantifier reasoning, we want the models to infer `everyone` and `someone` (∀ and ∃, respectively) according to the following rules:

- Introduction of the existential

  $P =$ `Patricia has visited Uruguay, Natasha didn't visit Bolivia.`
  $H =$ `[MASK] has visited Uruguay.`
  $Y =$ `someone.`

- Negation of the existential

  $P =$ `Leona has visited Vietnam, everyone has visited Joanne.`
  $H =$ `it's false that [MASK] didn't visit Joanne.`
  $Y =$ `someone.`

- Negation of the universal

  $P =$ `Christina has visited Philippines, Dawn didn't visit Frances.`
  $H =$ `it's false that [MASK] has visited Frances.`
  $Y =$ `everyone.`

3.5.6 Counting

The type of inference related to counting can be defined by the use of counting quantifiers $\exists_{\geq n} x$ — `at least` n x`'s`. These quantifiers are related to numerals (one, two, etc). In this module, we work with the numerals from one to twenty. We use a single rule that can create examples using different counting quantifiers. In all cases the model should infer the numeral related to the number of places visited by an specific agent. For variability, we use different numbers of facts in the premise.

- Introduction of the counting quantifier

  $P =$ `Billy didn't visit Socorro,Billy has visited Harlingen,`
  `Brandon has visited Wyandotte, Billy has visited Huntington.`
  $H =$ `Billy has visited at least [MASK] places.`
  $Y =$ `two.`

For each module, we provide training and test data with 10K and 1K examples, respectively. All data is balanced; and, as usual, the model's accuracy is evaluated on the test data. The full data and the code for the experiments are available online [Salvatore, 2019b].

3.5.7 Experiments

In order to understand how difficult is the MI task, we have performed a preliminary evaluation using only the pre-trained model $\text{BERT}_{\text{BASE}}$ [Devlin et al., 2019]. We use the same model in three different ways:

- **BERT-IP**. First, we fine-tune BERT on the standard NLI task using the IP version of the synthetic dataset (remember, by "IP version" we mean the usual text classification task well established in the NLI field). Then we obtain the text accuracy for each module according to its IP version.

- **BERT-IG-pre-trained-only**. We evaluate the pre-trained version of the model BERT in the MI task *without* performing the fine-tuning.
- **BERT-IG-fine-tuned**. We fine-tune BERT for the MI task and check the performance of the model on each module.

The results for the experiments can be seen in Table 3.3.

Model	Module		
	Boolean Coordination	Quantifier Reasoning	Counting
BERT-IP	100	90.5	87.5
BERT-IG-pre-trained-only	0	18.5	4.7
BERT-IG-fine-tuned	55.3	100	13.2

Table 3.3: *Test accuracy (%) for the two types of inference task (all modules).*

The effects of changing from IP to IG is not the same across modules. On the one hand, in the quantifiers module there is an increase in performance when we change the type of inference task. On the other hand, there is a significant drop in performance when we move from IP to IG (both in the Boolean coordination and counting modules). It seems that the basic quantifiers inference is solvable by pre-trained models like BERT; however *Boolean coordination and counting quantifiers pose a challenge to pre-trained models.*

As an attempt to check how much logical deduction is learned in the MLM task that determines the pre-training stage of BERT, we can compare the performance between BERT-IG-pre-trained-only and BERT-IG-fine-tuned. Figures 3.6, 3.7 and 3.8 show the performance of these modules regarding each logical connective on the different modules.

In Figure 3.6, we can observe that the simple skill of Boolean coordination introduction cannot be learned on the pre-training stage. On the other hand, in Figure 3.7, we see that the model BERT-IG-pre-trained-only is able to correctly use the quantifiers `someone` and `everyone` for some examples.

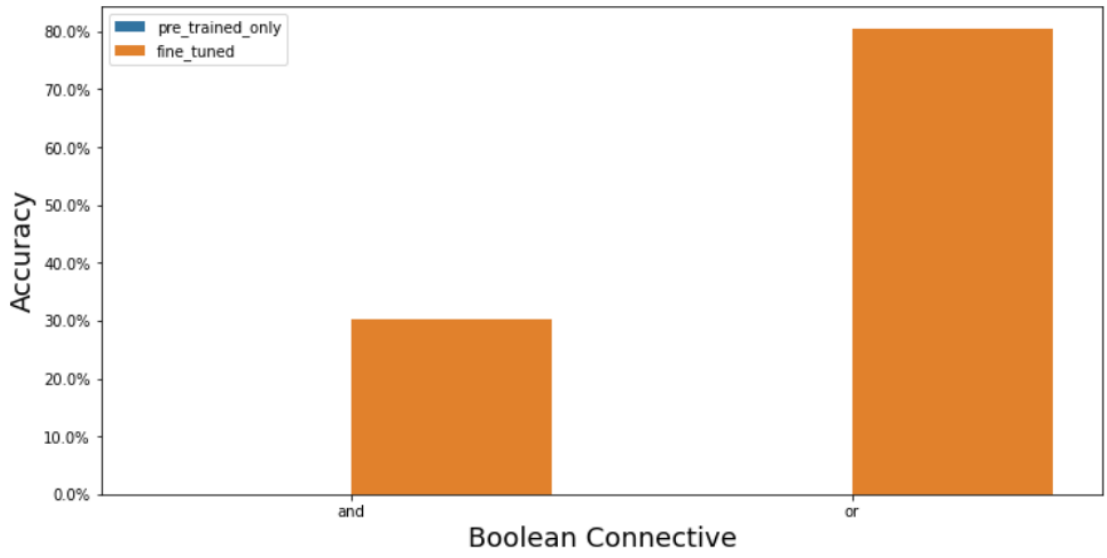

Figure 3.6: *Test accuracy for each connective inside the Boolean coordination module in the MI task. The figure shows BERT's accuracy with and without fine-tuning.*

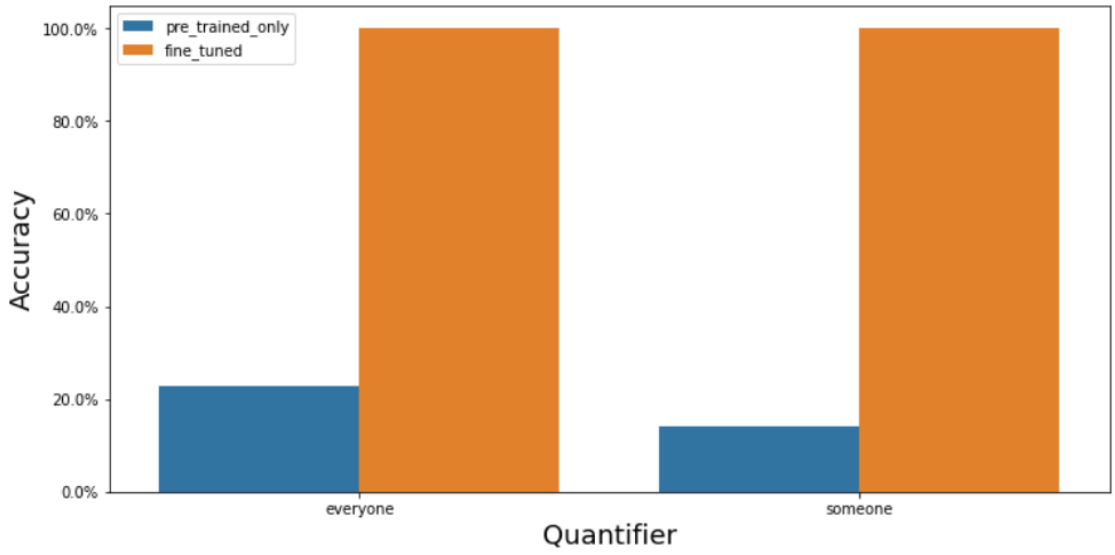

Figure 3.7: *Test accuracy for each quantifier inside the quantifier reasoning module in the MI task. The figure shows BERT's accuracy with and without fine-tuning.*

After a close inspection on the results associated with the masked inference task for the counting module (Figure 3.8), we observe that the model BERT-IG-pre-trained-only is only able to correctly predict examples for the numerals `two` and `three` (perhaps this is one bias from the pre-training stage, i.e., maybe there is an over-representation of these two numerals in the MLM task). Regarding the BERT-IG-fine-tuned model, we see that although this models shows promising results for the numerals `three` and `seventeen`, *for the majority of cases the accuracy of the model is zero*. Indicating that even when we fine-tune BERT for this module we are unable to solve deductions problems based on simple numerals. This outcome is aligned with the one described in Section 3.4: *the competence related to counting is the most demanding for BERT*.

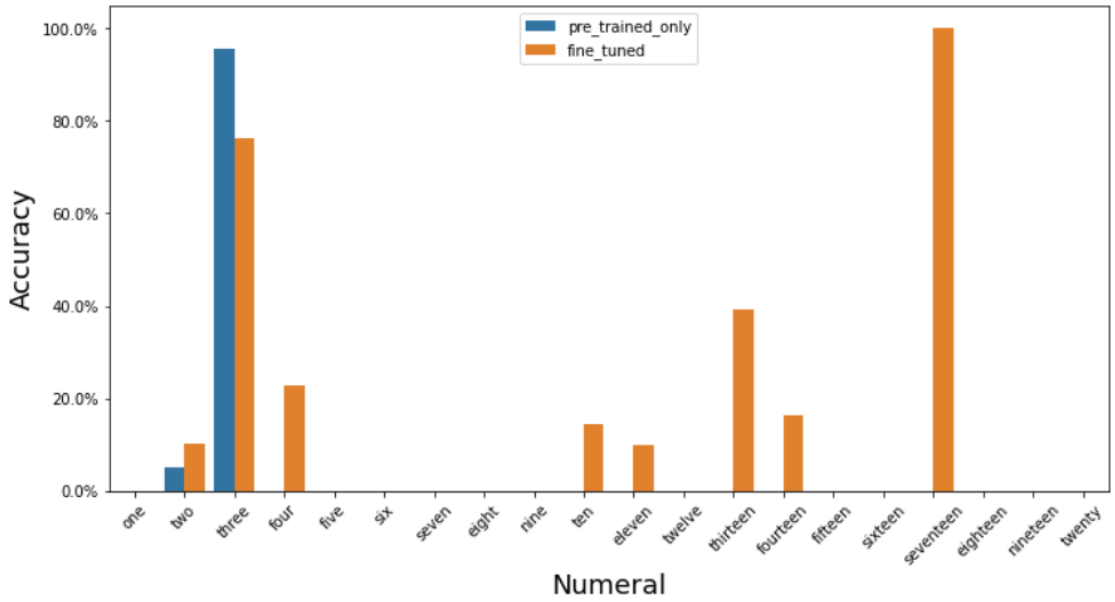

Figure 3.8: *Test accuracy for each numeral inside the counting module in the MI task. The figure shows BERT's accuracy with and without fine-tuning.*

3.5.8 Analysis Conclusion

The experiment performed in this analysis were just a first exploration into the MI task. We have successfully shown that this new tasks offers enough difficulty to the NLI models so that in future works we can expand on the ideas presented here and create a complete dataset centered on inference generation.

3.6 Benefits and Limitations of Synthetic Corpora

The use of synthetic data is not a new strategy in NLI. This procedure can be traced back to the creation of the Fracas dataset [The Fracas Consortium et al., 1996]. Through the years we have seen the creation of synthetic data to analyze the different competences of neural models [Bowman et al., 2015b, Evans et al., 2018, Tran et al., 2018, Weston et al., 2016]. The novelty of our approach lies on the use of logical rules as a source of dataset creation and cross-lingual analysis. This approach has allowed us to simultaneously create standard NLI datasets in multiple languages and, at the same time, explore a new type of inference generation task.

In the analyses done in Sections 3.4 and 3.5 we have obtained the following results:

- The counting quantifiers are the most challenging operator for deep learning models (both in English and Portuguese).
- There is evidence in favor of using cross-lingual transfer learning for NLI.
- By adopting a new type of evaluating procedure (masked inference), we can observe that the logical competence that on the surface seem solved by the current models is, in fact, a hard competence to master.

These three results show the main benefit of using an synthetic data in NLI: *the creation of a controlled environment to measure logical competence*. Since we have used a collection of datasets where no other inference factors are present (protecting the resulting datasets from biases that can be introduced by crowdsource annotators [Gururangan et al., 2018]), we were able to identify exactly that the current text classification models still struggle to perform inferences based on logical connectives.

It is worth mentioning that the subsequent paper by Richardson et al. [2020] was influenced by the analysis presented in Section 3.4. In that work, the authors have created a set of different NLI tasks that they call *semantic fragments*: logic fragments (an extension of the tasks presented in Subsection 3.4.2), and a fragment based on monotonicity reasoning.

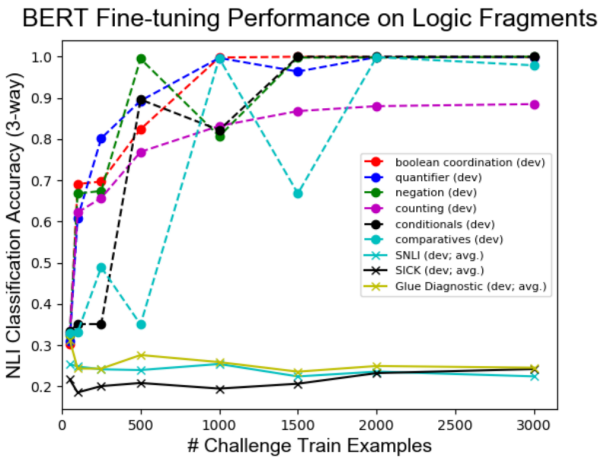

Figure 3.9: *Accuracy results for BERT on the different logic fragments (Figure by Richardson et al. [2020]).*

The findings from that work are consistent with ours: it is possible to train the BERT model to obtain an accuracy close to 100% for all logic tasks (counting is the only exception, as seen in Figure 3.9). Moreover, they have also reported that by training BERT on benchmark datasets and evaluating it on logical fragments we obtain a poor performance. Let BERT_{SNLI} and $\text{BERT}_{SNLI+MNLI}$ denote the versions of the BERT model trained on the SNLI and the combination of the SNLI and MNLI datasets, respectively. The authors have reported that the average test accuracy for BERT_{SNLI} and $\text{BERT}_{SNLI+MNLI}$ on the logic fragments is 46.1% and 47.3%, respectively. Showing that basic logic inference cannot be learned using the benchmark datasets.

Although there is a confluence in results, Richardson et al. [2020] formulate a fair criticism of our work:

> In nearly all cases, it is possible to train a model to master a fragment (with counting being the hardest fragment to learn). In other studies on learning fragments [Geiger et al., 2018, Salvatore et al., 2019a], this is the main result reported, however, we also show that the resulting models perform below random chance on benchmark tasks, meaning that these models are not by themselves very useful for general NLI. This even holds for results on the GLUE diagnosis test [Wang et al., 2018], which was hand-created and designed to model many of the logical phenomena captured in our fragments.

They have highlighted the main weakness of our synthetic data: *the sentences produced by the template language do not adequately represent the linguistic variety of everyday speech.* This problem can be verified in different ways. The average vocabulary size for a dataset presented in Section 3.4 is approximately 3K (Table 3.1). A number notably lower when compared to the vocabulary size of the SNLI and MNLI datasets (42K and 100K, respectively). Other way to observe the specificity of our dataset is by looking at the sentence complexity. For example, if we select from the benchmark datasets the observations related to counting, we can find a linguistic diversity that our templates are unable to represent. Take this observation from the SNLI:

> $P =$ `Two men are on scaffolding as they paint above a storefront while a man on the sidewalk stands next to them talking on the phone.`
>
> $H =$ `Three men are outside.`
>
> $Y =$ *entailment.*

Or this example in the MNLI dataset:

> $P =$ `Five forks guarantee real comfort, but the food will not necessarily be better than in a two-or three-fork establishment, just more expensive.`
>
> $H =$ `Five fork restaurants are rated higher because they are more popular and have celebrity appearances.`
>
> $Y =$ *neutral.*

It is possible to obtain a more general understanding on this issue when we compare the datasets using a readability metric. For example, the Flesch score is a metric applied to texts for the objective of measuring reading difficulty [Flesch, 1948]. The scores ranges from $(-\infty, 122]$, high scores imply an easy to read text, and low scores are interpreted as an indication of an extremely difficulty text.[5]

[5]This score should be interpreted as a proxy for reading complexity, since it measures only some superficial text features. The Flesch score of a document is given by:

$$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right).$$

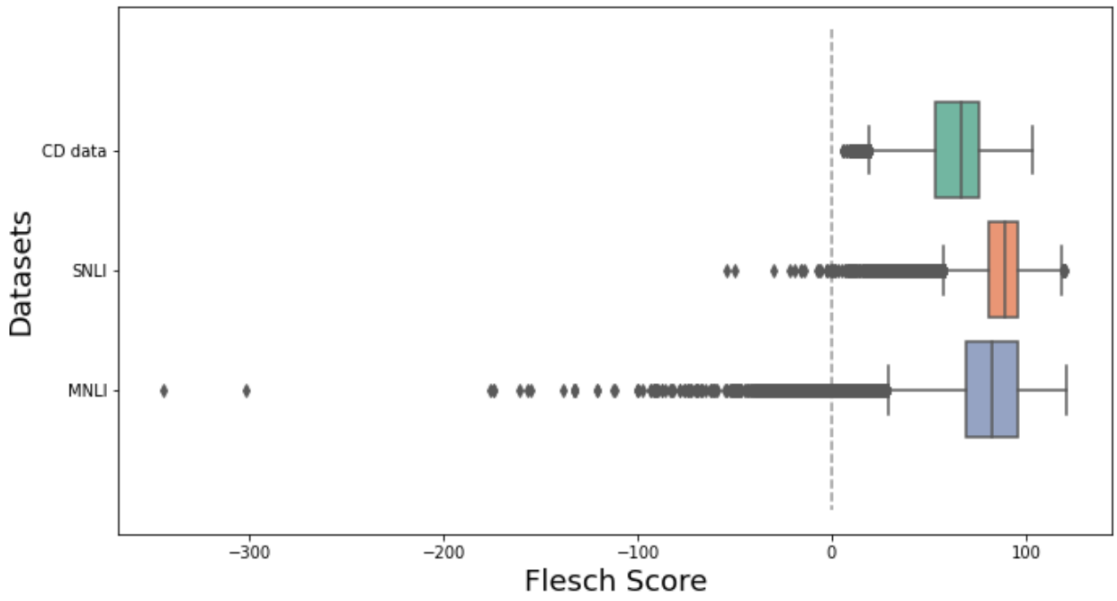

Figure 3.10: *Flesch score distribution for the datasets CD data (the dataset presented in Section 3.4.2), SNLI and MNLI.*

Figure 3.10 shows the distribution of the Flesch score for three datasets: the dataset presented in Section 3.4.2 (all modules), and the benchmark datasets SNLI and MNLI. The score was calculated using as the input document the concatenation of the premise and the hypothesis. As can be seen in the figure, the score range for the synthetic data is the smallest one. The CD dataset has no observation with a score less than zero, this is one indication that we were not able to create high demanding texts using the template language.

This representation problem does not invalidated the results presented in Sections 3.4 and 3.5, however, it raises the question of the generalization power of the same results. After considering the points made by Richardson et al. [2020], we believe that a valid research path lies in defining better representations, i.e., the project of combining logical rules with the help of crowdsourcing annotators with the goal of constructing a faithful dataset that is centered on an specific logical property.

Chapter 4

Equivalences

This chapter marks a shift in direction. Instead of creating new synthetic corpora, we focus on modifying existing datasets. By doing so, we are not only trying to avoid the generalization issues mentioned on the previous chapter, but by using dataset altering functions, we can improve how the field measures the limitations of NLI models. The latter point needs further explanation. As mentioned in Chapter 1, the recent progress of deep learning techniques based on pre-trained models has produced models capable of achieving high scores on benchmark NLI datasets [Devlin et al., 2019, Howard and Ruder, 2018, Liu et al., 2019b, Radford et al., 2019]. In order to understand how these powerful models generalize to new inference examples, different approaches have used a similar evaluation method: create a new NLI test set that comprises of sentences with known logic and semantic properties, train a model on a benchmark NLI dataset, and then evaluate such model on the new set [Glockner et al., 2018, Richardson et al., 2020, Yanaka et al., 2019]. This type of evaluation produces an analysis problem, namely, if a model fails to generalize to a new test set, we cannot be sure whether we are observing a real model limitation or just a consequence of the properties of the training data. In order to address this problem, we propose an alternative evaluation method.[1]

4.1 A New Resampling-Based Method to Evaluate NLI Models

Our goal here is to investigate whether the NLI models perform the same type of inference for different text inputs with the same *intended meaning*. For this purpose, we define a class of text transformations that can change a NLI input without altering the underlying logical relationship. Based on such transformations and a benchmark dataset, we construct an experimental design where a percentage of the training data is substituted by its transformed version. We also define different versions of the test set: the original one obtained from the benchmark dataset, and the one were all observations are transformed. Then, we propose an adaptation of the paired t-test to compare the model's performance on the two versions of the test set. We call the whole procedure the *Invariance under Equivalence* (IE) test. This approach provides two direct advantages: we substitute the expensive endeavour of dataset creation by the simpler task of constructing an adequate transformation function; and since the proposed hypothesis test is carefully crafted to account for the variety of ways that a transformation can affect the training of a machine learning model, we offer an evaluation procedure that is both meaningful and statistically sound.

As a case study, we study the sensibility of different state-of-the art models using benchmark datasets *Stanford Natural Language Inference Corpus* (SNLI) [Bowman et al., 2015a] and *Multi-Genre NLI Corpus* (MNLI) [Williams et al., 2018] under a small perturbation based on *synonym substitution*. We focus on this specific transformation as a way to understand how much lexical knowledge can be learned when fine-tuning deep learning models. Two main results have been obtained:

[1] This chapter is based on a paper, it was submitted to a relevant NLP Journal, and now it is under review.

- *Current deep learning models show two different inference outputs for sentences with the same meaning.* After applying the IE test using both datasets and different percentages of transformation in the training data, we have observed that the deep learning models fail the IE test in the vast majority of cases. This result indicates that by just adding transformed examples in the fine-tuning phase we are not able to remove some *biases* originated from the pre-training stage.
- *Some NLI models are clearly more robust than others.* By measuring each model's performance on the non-transformed test set when altered examples are present in training, we have observed that BERT [Devlin et al., 2019] and RoBERTa [Liu et al., 2019b] are significantly more robust than XLNet [Yang et al., 2019] and ALBERT [Lan et al., 2020].

The chapter is organized as follows: in Section 4.2 we show how to define logical preserving transformations using the notion of equivalence; in Section 4.3 we introduce the IE test; in Sections 4.4 and 4.5 we present one application of the IE test for the case of synonym substitution and comment on the experimental results; and in Section 4.6 we discuss the related literature.

4.2 Equivalence

The concept of equivalence, which is formally defined in logic, can also be employed in natural language with some adjustments to take into account its complex semantics. Once we establish an equivalent relation, we define a function that maps sentences to their equivalent counterpart and extend this function to any NLI dataset.

4.2.1 Equivalence in Formal and Natural Languages

There are two complementary perspectives on inference in a formal language, namely, the one focused on the truth value of the formulas and the other centered on the notion of a deductive system. These two approaches coincide under a complete deductive system, i.e., the theorems proved on the deductive system are exactly the ones that are true. One property related to completeness is based on the notion of equivalence. We say that two formulas are *equivalent* if both have the same truth value. For example, let $\boldsymbol{p}$ denote a propositional variable, $\wedge$ the conjunction operator, and $\top$ a tautology (i.e., a sentence that is always true, e.g., $0 = 0$). The truth value of the formula $\boldsymbol{p} \wedge \top$ depends only on $\boldsymbol{p}$ (in general, any formula of the form $\boldsymbol{p} \wedge \top \wedge \cdots \wedge \top$ has the same truth value as $\boldsymbol{p}$). Hence, we say that $\boldsymbol{p} \wedge \top$ and $\boldsymbol{p}$ are equivalent formulas.

If a deductive system is complete, then any equivalence in the formal language can be proved inside the system. Hence, using the fact that the inference rules are equivalence preserving operations [Shoenfield, 1967, p. 34], in a complete system, we can substitute one formula for any of its equivalent versions without disrupting the deductions from the system. For example, let $\boldsymbol{q}$ be a propositional variable, $\rightarrow$ the implication connective and $\vdash$ the consequence relation in a complete deductive system. It follows that:

$$\begin{gathered}
\boldsymbol{p}, \boldsymbol{p} \rightarrow \boldsymbol{q} \vdash \boldsymbol{q}, \\
\boldsymbol{p} \wedge \top, \boldsymbol{p} \rightarrow \boldsymbol{q} \vdash \boldsymbol{q} \wedge \top, \\
\vdots \\
\boldsymbol{p} \wedge \top \wedge \cdots \wedge \top, \boldsymbol{p} \rightarrow \boldsymbol{q} \vdash \boldsymbol{q} \wedge \top \wedge \cdots \wedge \top.
\end{gathered}$$

In other words, under a complete system, if we assume $\boldsymbol{p} \rightarrow \boldsymbol{q}$ and any formula equivalent to $\boldsymbol{p}$ we always can deduce a formula equivalent to $\boldsymbol{q}$. This result offers one simple way to verify that a system is incomplete: we take an arbitrary pair of equivalent formulas and check whether by substituting one for the other the system's deductions diverge.

We propose to incorporate this verification procedure to the NLI field. This is a feasible approach because the concept of equivalence can be understood in natural language as *meaning identity* [Shieber, 1993]. Thus, we formulate the property associated with a complete deductive system as a linguistic competence:

> If two sentences have the same meaning, it is expected that any consequence based on them should not be disrupted when we substitute one sentence for the other.

We call this competence the *invariance under equivalence* (IE) property. Similar to formal logic, we can investigate the limitations of NLI models by attesting if they fail to satisfy the IE property.

Moving from logical equivalence to meaning identity is not straightforward. Some phenomena that have some grounding in logic become negligible in the linguistic context. For example, the relationship mentioned above based on tautology addition hardly corresponds to meaning identity in natural language. Hence, we need to frame the IE property considering a variety of equivalent forms that emerge from language use. We list a few examples here.

Synonym Substitution. The basic case of equivalence can be found in sentences composed by constituents with the same denotation. For example, take the sentences: `a man is fishing` , `a guy is fishing`. This instance shows the case where one sentence can be obtained from the other by substituting one or more words by their respective synonyms while denoting the same fact.

Voice Transformation. One stylistic transformation that is usually done in writing is the change in grammatical voice. It is possible to write different sentences both in the active and passive voice: `the crusaders captured the holy city` can be modified to `the holy city was captured by the crusaders`, and vice-versa.

Constituents Permutation. Since many relations in natural language are symmetric, we can permute the relations' constituents without causing meaning disruption. This can be done using either definite descriptions or relative clauses. In the case of definite descriptions, we can freely permute the entity being described and the description. For example, `Iggy Pop was the lead singer of the Stooges` is equivalent to `The lead singer of the Stooges was Iggy Pop`. When using relative clauses, the phrases connected can be rearranged in any order. For example, `John threw a red ball that is large` is interchangeable with `John threw a large ball that is red`.

4.2.2 The IE Property for the NLI Task

In order to simplify the analysis, we have decided to formulate all the discussion on linguistic equivalences in terms of *transformation functions*. Such functions are designed to transform a sentence into a specific equivalent version of itself. For instance, take the mapping that substitutes all occurrences of the word `man` in a sentence by the word `guy`; this mapping is one particular representative of the synonym substitution equivalence relation.[2]

Due to its meaning preserving property, we apply a transformation function to all sentences in a NLI dataset without affecting the original logical relationship. More formally, given a NLI data $\mathcal{D} = \{(P_i, H_i, Y_i) \ : \ i = 1, \ldots, n\}$, where P_i , H_i, Y_i are the premise, hypothesis and target, respectively; and given a transformation φ, we define $({P_i}^{\varphi}, {H_i}^{\varphi}, Y_i)$ as the result of applying φ to both the premise and hypothesis of the i-th observation of this set. We also define $\mathcal{D}^{\varphi}$ as the entire set transformation, i.e., $\mathcal{D}^{\varphi} = \{({P_i}^{\varphi}, {H_i}^{\varphi}, Y_i) \ : \ i = 1, \ldots, n\}$.

Let $\mathcal{D}_T, \mathcal{D}_V, \mathcal{D}_{Te}$ be variables for the training, validation, and test portions of a NLI dataset, respectively; and let φ be a transformation function. It is well know that when we train a machine learning model in a sample with no transformed observations, such model will likely perform poorly on a test set only containing transformed sentences. Hence, in order to assert that a NLI model fails the IE property in a meaningful way, we allow some fraction of the training data to be altered by φ. For this reason, we formulate the IE property for the NLI task as follows:

[2] Throughout this chapter, we use "transformation" to denote an alteration function associated to an equivalent relation.

Given a machine learning model trained on a dataset containing a sufficient amount of observations transformed by φ, it is expected that the model's performances on the sets $\mathcal{D}_{Te}$ and $\mathcal{D}_{Te}^{\varphi}$ are not significantly different.

4.3 Testing for Invariance

In this section, we propose an experimental design to measure the IE property for the NLI task: the IE test. Broadly speaking, the IE test is composed of three main steps: i) we resample an altered version of the training data and obtain a classifier by estimating the model's parameters on the transformed sample; ii) we perform a paired t-test to compare the classifier's performance on the two versions of the test set; iii) we repeat steps i) and ii) M times and employ the Bonferroni method to combine the multiple paired t-tests into a single decision procedure. In what follows, we describe in details steps i), ii) and iii). After establishing all definitions, we present the IE test as an algorithm and comment on some alternatives.

4.3.1 Training on a Transformed Sample

First, let us define a generation process to model the different effects caused by the presence of a transformation function on the training stage. Since, we are assuming that any training observation can be altered, the generation method is constructed as a stochastic process.

Given a transformation φ and a *transformation probability* $\rho \in [0, 1]$ we define the (φ, ρ) *data generating process*, $\text{DGP}_{\varphi,\rho}(\mathcal{D}_T, \mathcal{D}_V)$, as the process of obtaining a modified version of the train and validation datasets where the probability of each observation being altered by φ is ρ. More precisely, let $\mathcal{D} \in \{\mathcal{D}_T, \mathcal{D}_V\}$, $|\mathcal{D}| = n$ and $L_1, \ldots, L_n \sim \text{Bernoulli}(\rho)$. An altered version of $\mathcal{D}$ is the set composed of the observations of the form $({P_i}^{new}, {H_i}^{new}, Y_i)$, where:

$$({P_i}^{new}, {H_i}^{new}, Y_i) = \begin{cases} ({P_i}^{\varphi}, {H_i}^{\varphi}, Y_i) & \text{if } L_i = 1, \\ (P_i, H_i, Y_i) & \text{otherwise} \end{cases} \tag{4.1}$$

This process is applied independently to $\mathcal{D}_T$ and $\mathcal{D}_V$. Hence, if $|\mathcal{D}_T| = n_1$ and $|\mathcal{D}_V| = n_2$, then there are $2^{(n_1+n_2)}$ distinct pairs of transformed sets $(\mathcal{D}_T{}', \mathcal{D}_V{}')$ that can be sampled. We write

$$\mathcal{D}_T{}', \mathcal{D}_V{}' \sim \text{DGP}_{\varphi,\rho}(\mathcal{D}_T, \mathcal{D}_V), \tag{4.2}$$

to denote the process of sampling a transformed version of the datasets $\mathcal{D}_T$ and $\mathcal{D}_V$ according to φ and ρ.

Second, to represent the whole training procedure we need to define the underlying NLI model and the hyperparameter space. For $d, s \in \mathbb{N}$, let $\mathcal{M} = \{f(x; \theta) \ : \ \theta \in \Theta \subseteq \mathbb{R}^d\}$ be a parametric model, and let $\mathcal{H}_{\mathcal{M}} \subseteq \mathbb{R}^s$ be the associated *hyperparameters space* required by the model. By *search* we denote any algorithm of hyperparameter selection (e.g., random search [Bergstra and Bengio, 2012]). Thus, given a number of maximum search $\mathcal{B}$, a *budget*, this algorithm chooses a specific *hyperparameter value* $h \in \mathcal{H}_{\mathcal{M}}$:

$$h = \text{search}(\mathcal{D}_T, \mathcal{D}_V, \mathcal{M}, \mathcal{H}_{\mathcal{M}}, \mathcal{B}). \tag{4.3}$$

A classifier g is attained by fitting the model $\mathcal{M}$ on the training data $(\mathcal{D}_T{}', \mathcal{D}_V{}')$ based on a hyperparameter value h and a stochastic approximation algorithm (*train*):

$$g = \text{train}(\mathcal{M}, \mathcal{D}_T{}', \mathcal{D}_V{}', h). \tag{4.4}$$

The function g is an usual NLI classifier: its input is the pair of sentences (P, H), and its output is either -1 (contradiction), 0 (neutral), or 1 (entailment).

4.3.2 A Bootstrap Version of the Paired t-Test

The IE test is based on the comparison of the classifier's accuracies in two paired samples: $\mathcal{D}_{Te}$ and $\mathcal{D}_{Te}^{\varphi}$. Pairing occurs because each member of a sample is matched with an equivalent member in the other sample. To account for this dependency, we perform a paired t-test. Since we cannot guarantee that the presuppositions of asymptotic theory are preserved in this context, we formulate the paired t-test as a bootstrap hypothesis test [Fisher and Hall, 1990, Konietschke and Pauly, 2014].

Given a classifier g, let A and B be the variables indicating the correct classification of the two types of random NLI observation:

$$A = I(g(P, H) = Y), \qquad B = I(g(P^{\varphi}, H^{\varphi}) = Y). \tag{4.5}$$

The *true accuracy* of g for both version of the text input is given by

$$\mathbb{E}[A] = \mathbb{P}(g(P, H) = Y), \qquad \mathbb{E}[B] = \mathbb{P}(g(P^{\varphi}, H^{\varphi}) = Y). \tag{4.6}$$

We approximate theses quantities by using the estimators $\bar{A}$ and $\bar{B}$ defined on the test data $\mathcal{D}_{Te} = \{(P_i, H_i, Y_i) \ : \ i = 1, \ldots, n\}$:

$$\bar{A} = \frac{1}{n}\sum_{i=1}^{n} A_i, \qquad \bar{B} = \frac{1}{n}\sum_{i=1}^{n} B_i, \tag{4.7}$$

where A_i and B_i indicate the classifier's correct prediction on the original and altered version of the i-th observation, respectively. Let *match* be the function that returns the vector of matched observations related to the performance of g on the datasets $\mathcal{D}_{Te}$ and $\mathcal{D}_{Te}^{\varphi}$:

$$\text{match}(g, \mathcal{D}_{Te}, \mathcal{D}_{Te}^{\varphi}) = ((A_1, B_1), \ldots, (A_n, B_n)). \qquad \text{(matched sample)} \tag{4.8}$$

In the matched sample (4.8) we have information about the classifier's behavior for each observation of the test data *before* and *after* applying the transformation φ. Let δ be defined as the difference between probabilities:

$$\delta = \mathbb{E}[A] - \mathbb{E}[B]. \tag{4.9}$$

We test hypothesis H_0 that the probabilities are equal against hypothesis H_1 that they are different:

$$H_0 : \delta = 0 \text{ versus } H_1 : \delta \neq 0. \tag{4.10}$$

Let $\hat{\delta}_i = A_i - B_i$ and $\hat{\delta} = \bar{A} - \bar{B}$. We test H_0 by using the paired t-test statistic:

$$t \ = \ \frac{\hat{\delta} - 0}{\hat{se}(\hat{\delta})} \ = \ \frac{\sqrt{n}(\bar{A} - \bar{B})}{S}, \tag{4.11}$$

such that $\hat{se}(\hat{\delta}) = S/\sqrt{n}$ is the estimated standard error of $\hat{\delta}$, where

$$S = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{\delta}_i - \hat{\delta})^2}. \tag{4.12}$$

In order to formulate the IE test in a suitable manner, we write $X = (X_1, \ldots, X_n)$ to denote the vector of paired variables (4.8), i.e., $X_i = (A_i, B_i)$ for $i \in \{1, \ldots, n\}$. We also use $t = f_{\text{paired t-test}}(X)$ to refer to the process of obtaining the test statistic (4.11) based on the matched data X. The observable test statistic is denoted by $\hat{t}$.

The test statistic t is a standardized version of the accuracy difference $\bar{A} - \bar{B}$. A positive value for t implies that $\bar{A} > \bar{B}$ (the classifier is performing better on the original data compared to the transformed data). Similarly, when t takes negative values we have that $\bar{B} > \bar{A}$ (the performance on the modified test data surpasses the performance on the original test set). If the null hypothesis (H_0) is true, then it is more likely that the observed value $\hat{t}$ takes values closer to zero. In order to formulate probability judgments about $\hat{t}$, we need to obtain the distribution of the test statistic that would follow if the null hypothesis were true.

Following the bootstrap method, we estimate the distribution of t under the null hypothesis through resampling the matched data (4.8). It is worth noting that we need to generate observations under H_0 from the observed sample, even when the observed sample is drawn from a population that does not satisfy H_0. In the case of the paired t-test, we employ the resampling strategy mentioned by Konietschke and Pauly [2014]: a resample $X^* = (X_1^*, \ldots, X_n^*)$ is draw from the original sample with replacement such that each X_i^* is a random permutation on the variables A_j and B_j within the pair (A_j, B_j) for $j \in \{1, \ldots, n\}$. In other words, X^* is a normal bootstrap sample with the addition that each simulated variable X_i^* is either (A_j, B_j) or (B_j, A_j), with probability 1/2, for some $j \in \{1, \ldots, n\}$. This is done to force that the average values related to the first and second components are the same, following the null hypothesis (in this case, $\mathbb{E}[A] = \mathbb{E}[B]$).

We use the simulated sample X^* to calculate the bootstrap replication of t, $t^* = f_{\text{paired t-test}}(X^*)$. By repeating this process $\mathcal{S}$ times, we obtain a collection of bootstrap replications $t_1^*, \ldots, t_{\mathcal{S}}^*$. Let $\hat{F}^*$ be the empirical distribution of t_s^*. We compute the *equal-tail bootstrap p-value* as follows:

$$\begin{aligned} \text{p-value} &= 2\min(\hat{F}^*(\hat{t}), 1 - \hat{F}^*(\hat{t})) \\ &= 2\min\left(\frac{1}{\mathcal{S}}\sum_{s=1}^{\mathcal{S}} I(t_s^* \leq \hat{t}),\ \frac{1}{\mathcal{S}}\sum_{s=1}^{\mathcal{S}} I(t_s^* > \hat{t})\right). \end{aligned} \tag{4.13}$$

In (4.13), we are simultaneously performing a left-tailed and a right tailed test. The p-value is the probability of observing a bootstrap replication, in absolute value $|t^*|$, larger than the actual observed statistic, in absolute value $|\hat{t}|$, under the null hypothesis.[3]

4.3.3 Multiple Testing

We make use of the (φ, ρ) data generating process to produce different effects caused by the presence of φ in the training stage. This process results in a variety of classifiers influenced by φ in some capacity. Using the paired t-test, we compare the performance of all these classifiers on the sets $\mathcal{D}_{Te}$ and $\mathcal{D}_{Te}^{\varphi}$ (as illustrated in Figure 4.1).

To assert that a model fails to satisfy the IE property we check whether at least one classifier based on this model presents a significantly different performance on the two versions of the test set. There is a methodological caveat here. By repeating the same test multiple times the likelihood of incorrectly rejecting the null hypothesis (i.e., the type I error) increases. One widely used correction for this problem is the Bonferroni method [Wasserman, 2010, p. 166]. The method's application is simple: given a significance level α, after testing M times and acquiring the p-values $p_1, \ldots, p_M$, we reject the null hypothesis if $p_m < \alpha/M$ for at least one $m \in \{1, \ldots, M\}$.

4.3.4 Invariance Under Equivalence Test

We call *invariance under equivalence test* the whole evaluation procedure of resampling multiple versions of the training data, acquiring different p-values associated with the classifiers' performance,

[3] Since we do not assume that t is symmetrically distributed around zero,we use this equation to calculate the p-value instead of the *symmetric bootstrap p-value*: $\frac{1}{\mathcal{S}}\sum_{s=1}^{\mathcal{S}} I(|t_s^*| > |\hat{t}|)$.

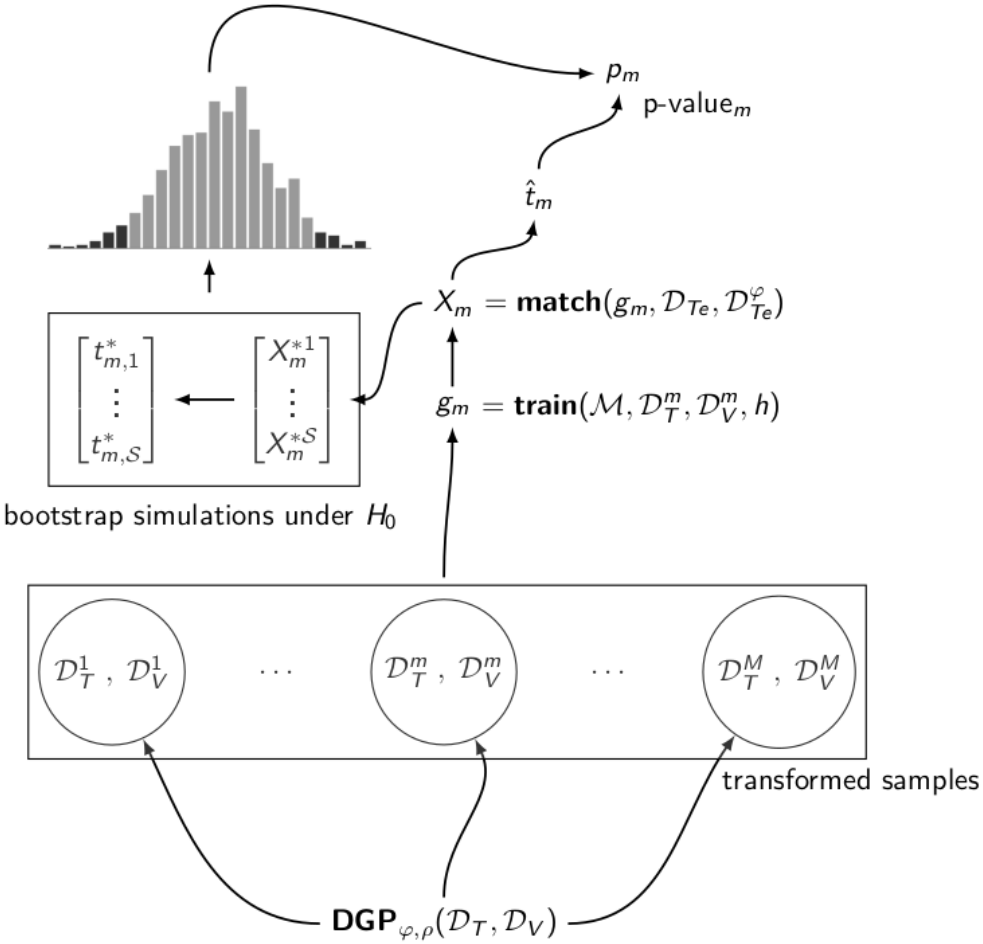

Figure 4.1: *The bootstrap version of the paired t-test applied multiple times. For $m = 1, \ldots, M$, g_m is a classifier trained on the transformed sample $(\mathcal{D}^m_T, \mathcal{D}^m_V)$. The p-value p_m is obtained by comparing the observable test statistic associated with g_m, $\hat{t}_m$, with the bootstrap distribution of t under the null hypothesis.*

and, based on these p-values, deciding on the significance of difference between accuracies. The complete description of the test can be found in Algorithm 1.

Many variations of the proposed method are possible. We comment on some options:

Alternative 1 As an alternative to the paired t-test, one can employ the McNemar's test (a simplified version of the Cochran's Q test) [Cochram, 1950, McNemar, 1947]. The McNemar statistic measures the symmetry between the changes in samples. The null hypothesis for this test states that the expected number of observations changed from $A_i = 1$ to $B_i = 0$ is the same as the ones changed from $A_i = 0$ to $B_i = 1$. Thus, the described strategy to resample the matched data (4.8) can also be used in this case. The only difference is in the calculation of the p-value, the McNemar's test is an one-tailed test.

Alternative 2. By the stochastic nature of the training algorithm used in the neural network field, there can be performance variation caused only by this algorithm. This is particularly true for deep learning models used in text classification [Dodge et al., 2020]. The training variation can be accommodated in our method by estimating multiple classifiers using the same transformed sample and hyperparameter value. After training all those classifiers, one can take the majority vote classifier as the single model g_m.

Alternative 3. Since we have defined the hyperparameter selection stage before the resampling process, one single hyperparameter value can influence the training on difference M samples. Another option is to restrict a hyperparameter value to a single sample. Thus, one can first obtain a modified sample and then perform the hyperparameter search.

All alternatives are valid versions to the method we are proposing. However, it should be noted

Algorithm 1 Invariance under equivalence test (IE test)

1. Select all basic variables: $\mathcal{D}_T, \mathcal{D}_V, \mathcal{D}_{Te}, \mathcal{M}, \mathcal{H}_{\mathcal{M}}, \mathcal{B}, \varphi, \rho, M, \mathcal{S}$ and α.
2. Obtain a hyperparameter value
$$h = \text{search}(\mathcal{D}_T, \mathcal{D}_V, \mathcal{M}, \mathcal{H}_{\mathcal{M}}, \mathcal{B}).$$
3. For m $= 1, \ldots, M$:
   (a) Generate a transformed training and validation sets
   $$\mathcal{D}_T^m, \mathcal{D}_V^m \sim \text{DGP}_{\varphi,\rho}(\mathcal{D}_T, \mathcal{D}_V).$$
   (b) Train a classifier on the new pair of sets using the selected hyperparameters
   $$g_m = \text{train}(\mathcal{M}, \mathcal{D}_T^m, \mathcal{D}_V^m, h).$$
   (c) Evaluate g_m on the two versions of the test data to obtain the matched sample X_m
   $$X_m = \text{match}(g_m, \mathcal{D}_{Te}, \mathcal{D}_{Te}^{\varphi}).$$
   (d) Obtain the observable value for the test statistic
   $$\hat{t}_m = f_{\text{paired t-test}}(X_m).$$
   (e) For $s = 1, \ldots, \mathcal{S}$ obtain the bootstrap sample generated under the null hypothesis X_m^{*s}, and compute the bootstrap replication of t, $t_{m,s}^* = f_{\text{paired t-test}}(X_m^{*s})$.
   (f) Using the empirical distribution of the simulated test statistics $t_{m,s}^*$ and the observable value $\hat{t}_m$, compute the bootstrap p-value p_m as described in (4.13).
4. Reject the null hypothesis if $p_m < \alpha/M$ for at least one $m \in \{1, \ldots, M\}$.

that the combination of alternatives 2 and 3 with large deep learning models, as the ones used to obtain state-of-the art results in NLI, yields a high computational cost.

4.4 Case Study: Verifying Invariance under Synonym Substitution

As a starting point to understand the effects of equivalent modifications on a NLI task, we have decided to concentrate our focus on transformations based on *synonym substitution*, i.e., any text manipulation function that substitutes an occurrence of a word by one of its synonyms.

4.4.1 Defining a Transformation Function

Among the myriad of synonym substitution functions, we have decide to work only with the ones based on the WordNet database [Fellbaum, 1998]. One of the principles behind our analysis is that an equivalent alteration should yield the smallest perturbation possible, hence we have constructed a transformation procedure based on the word frequency of each corpus. We proceed as follows: we utilize the spaCy library [Explosion, 2020] to select all nouns in the corpus, then for all nouns we use the WordNet database to list all synonyms and choose the one with the highest frequency. If no synonym appears in the corpus we take the one with the lower edit distance. Figure 4.2 shows a simple transformation example.

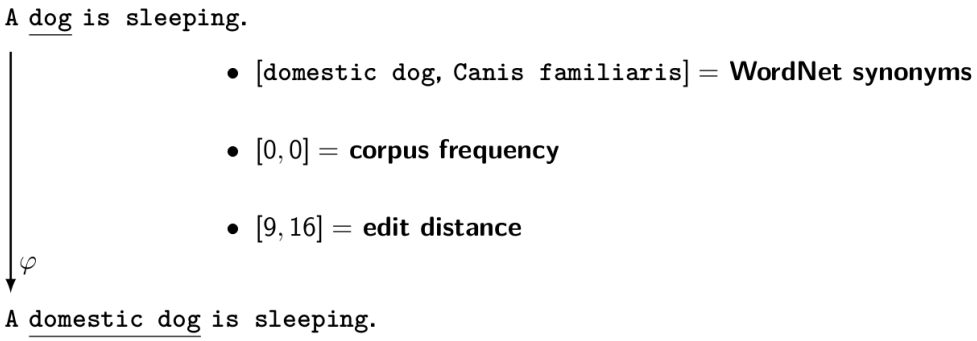

Figure 4.2: *Example of sentence transformation. In this case, there are two synonyms associated with the only noun appearing in the source sentence (`dog`). Since both synonyms have the same frequency in the corpus (zero), the selected synonym is the one with the lower edit distance (`domestic dog`).*

We expand this function to a NLI dataset applying the transformation to both the premise and the hypothesis. In all cases, the target Y remains unchanged.

4.4.2 Datasets

We have used the benchmark datasets *Stanford Natural Language Inference Corpus* (SNLI) [Bowman et al., 2015a] and *MultiGenre NLI Corpus* (MNLI) [Williams et al., 2018] in our analysis. The SNLI and MNLI datasets are composed of 570K and 433K sentence pairs, respectively. Since the transformation process described above is automatic (allowing us to modify such large datasets), it inevitably causes some odd transformations. Although, we have carefully reviewed the transformation routine, we have found some altered sentence pairs that are either ungrammatical or just unusual. For example, take this observation from the SNLI dataset (the relevant words are underlined):

P = `An old man in a baseball hat and an old woman in a jean jacket are standing outside but are covered mostly in shadow.`

H = `An old woman has a light jean jacket.`

Using our procedure, it is transformed in the following pair:

P^{φ} = `An old adult male in a baseball game hat and an old adult female in a denim jacket are standing outside but are covered mostly in shadow.`

H^{φ} = `An old adult female has a visible light denim jacket.`

As one can see, the transformation is far from perfect. It does not differentiate the word `light` from adjective and noun roles. However, unusual expressions as `visible light denim jacket` form a small part in the altered dataset and the majority of them are sound. To minimize the occurrence of any defective substitutions we have created a black list, i.e., a list of words that remain unchanged after the transformation. To grasp how much distortion we have added in the process, we estimate the *sound percentage* for each NLI dataset (Table 4.1). This quantity is defined as the number of sound transformations in a sample divide by the sample size. In Appendix A, we display some examples of what we call sound and unsound transformations for each dataset.

	95% Confidence Interval		
Dataset	Lower Bound	Upper Bound	Observable Value
SNLI	75.4%	89%	82.2%
MNLI	77.4%	91%	84.2%

Table 4.1: *Sound percentages for the transformation function based on the WordNet database. The values were estimated using a random sample of 400 sentence pairs.*

4.4.3 Methodology

The parameter ρ is a key factor in the IE test because it determines what is a "sufficient amount" of transformation in the training phase. Our initial intuition is that any machine learning model will not satisfy the IE property when we select extreme values of ρ. We believe that the samples generated by those values are *biased samples*: by choosing low values for ρ we do not offer enough examples of transformed sentences for the machine learning model in training; similarly when we use high values for ρ there is an over-representation of the modified data in the training phase. Hence, in order to find meaningful values for the transformation probability, *we utilize a baseline model to select values for ρ where it is harder to refute the null hypothesis.* As the baseline, we employ the gradient boosting classifier together with the bag-of-words representation.

The main experiment consists in applying the IE test to the recent deep learning models used in NLI: BERT [Devlin et al., 2019], XLNet [Yang et al., 2019], RoBERTa [Liu et al., 2019b], and ALBERT [Lan et al., 2020]. In order to repeat the test for different transformation probabilities and altered samples in a feasible time, we utilize only the pre-trained weights associated with the base version of these models. The only exception is for the model RoBERTa. Since this model has a large version fine-tuned on the MNLI dataset, we consider that it is relevant for our investigation to include a version of this model specialized in the NLI task. We use "RoBERTa$_{LARGE}$" to denote this specific version of the RoBERTa model. For the same reason, we work with a smaller version of each training dataset. Hence, for both SNLI and MNLI datasets we use a random sample of 50K observations for training (this means we are using only 8.78% and 11.54% of the training data of the SNLI and MNLI, respectively). Although this reduction is done to perform the testing, the transformation function is always defined using the whole corpus of each dataset. It should be noted that since the MNLI dataset has no labeled test set publicly available, we use the concatenation of the matched and mismatched development sets as the test portion for this dataset.

Because the change in transformation probabilities does not affect the hyperparameter selection stage, we perform a single search for each model and dataset with a budget to train 10 models ($\mathcal{B} = 10$). In Appendix B, we detail the hyperparameter spaces and the selected hyperparameter values for each model. For each value of ρ, we obtain 5 p-values and perform 1K bootstrap simulations ($M = 5, \mathcal{S} = 10^3$). We set the significance level to 5% ($\alpha = 0.05$); hence, the adjusted significant level is 1% ($\alpha/M = 0.01$). All the deep learning models were implemented using the HuggingFace transformer library [Wolf et al., 2019]. The code and data used for the experiments can be found online [Salvatore, 2020].

4.5 Results

In this section, we present the results and findings of the experiments with the synonym substitution function on SNLI and MNLI datasets. First, we describe how changing the transformation probability ρ affects the test for the baseline model. Second, we apply the IE test for the deep learning models using the most interesting choices for ρ. We comment on the test results and observe how to utilize the experiment outcome to measure the robustness of the NLI models.

4.5.1 Baseline Exploration

To mitigate the cost of training deep learning models, we have used the baseline to find the intervals between 0 and 1 where rejecting the null hypothesis is not a trivial exercise. Figure 4.3 shows the test results associated with the baseline for each dataset using 101 different choices of ρ (values selected from the set $\{0, 0.01, 0.02, \ldots, 0.98, 0.99, 1\}$).

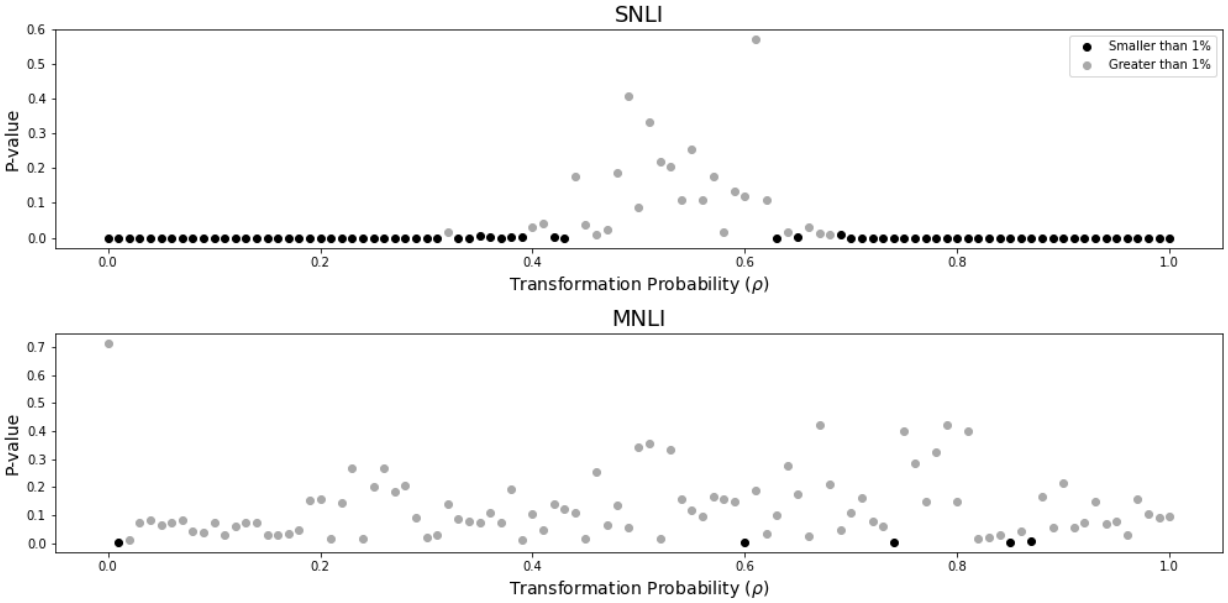

Figure 4.3: *Baseline results. In the x-axis we have different choices of transformation probabilities used in training. The y-axis displays the minimum value for the p-values acquired in five paired t-tests. We reject the null hypothesis if the minimum p-value is smaller than* 1%.

The results for the SNLI data are in agreement with our initial intuition: on the one hand, choosing extremes values for ρ (values from the intervals $[0, 0.2]$ and $[0.8, 1.0]$) yields p-values concentrated closer to zero, and so rejecting the null hypothesis at 5% significance level. On the other hand, when choosing a transformation probability in the interval $[0.4, 0.6]$, we are adding enough transformed examples for training, and so we were not able the reject the null hypothesis. The same phenomenon cannot be replicated in the MNLI dataset. It seems that for this dataset the introduction of transformed examples does not change the baseline performance - independently of the choice of ρ. Although we are able to obtain p-values smaller than 1% in five scenarios (namely for $\rho \in \{0.01, 0.6, 0.74, 0.85, 0.87\}$), the SNLI pattern does not repeat in the MNLI dataset.

4.5.2 Testing Deep Learning Models

The baseline has helped us to identify the interval of transformation probabilities where the performances on the two versions of the test set might be similar: the interval $[0.4, 0.6]$. Based on that information, we have chosen three values from this interval for the new tests, namely, $0.4, 0.5, 0.6$. To obtain a broader representation, we have also selected two values for ρ in both extremes. Hence, we have tested the deep learning models using seven values for ρ: $0, 0.2, 0.4, 0.5, 0.6, 0.8, 1$.

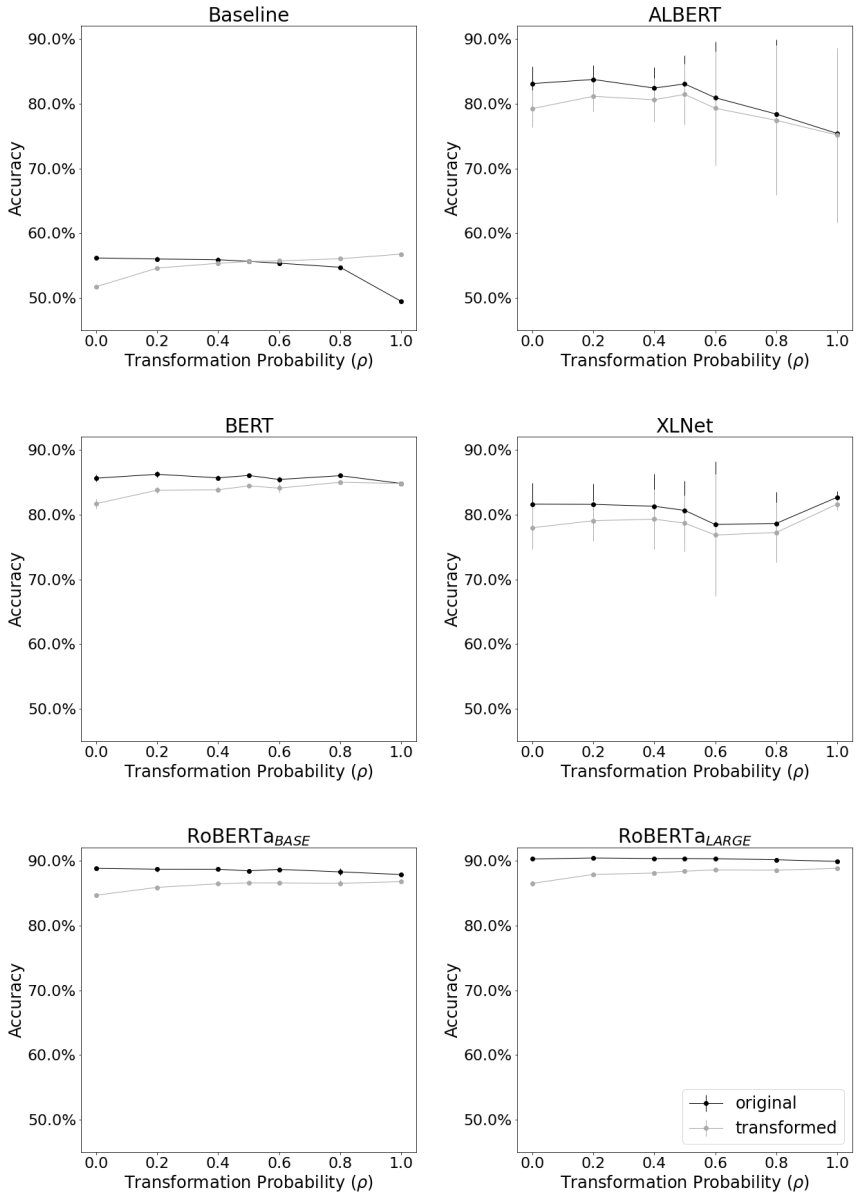

Figure 4.4: *SNLI results. In the x-axis we have different choices of transformation probabilities in training. The y-axis displays the accuracy. Each point represents the average accuracy in five runs. The vertical lines display the associated standard deviation. The black and grey lines represent the values for the original and transformed test sets, respectively.*

According to the test accuracies (Figures 4.4 and 4.5), we observe that ROBERTA$_{LARGE}$ is the best model. This is an expected result. ROBERTA$_{LARGE}$ is the larger version of the ROBERTA model with an architecture composed of more layers and attention heads. Not only ROBERTA$_{LARGE}$ outperforms ROBERTA$_{BASE}$ in different language understanding tasks [Liu et al., 2019b], but also the specific version of the ROBERTA$_{LARGE}$ model used in our experiments was fine-tuned on the MNLI dataset.

Each model is affected differently by the change in ρ. On the SNLI dataset (Figure 4.4), all models, except for ALBERT, continue to show a high accuracy even when we use a fully transformed training dataset. We have a similar picture on the MNLI dataset, (Figure 4.5). However, in this case, we notice a higher dispersion in the accuracies for the models ALBERT, XLNet, and ROBERTA$_{BASE}$.

In the majority of cases, we also observe that *the performance on the original test set is superior compared to the transformed version.* As seen in the Figures 4.4 and 4.5, in almost all choices of ρ and for all deep learning models, the black line (the accuracy on the original test set) dominates the

grey line (the accuracy on the transformed version of the test set). This difference becomes more evident for the test statistic (Figure 4.6). For almost every choice of ρ, all deep learning models have generated test statistics with extremely positive values. When we compare these statistics with the empirical distribution generated under the null hypothesis we obtain p-values smaller than 10^{-4} for the majority of cases. The exceptions are related to the models ALBERT and BERT on the SNLI dataset using $\rho = 1$. In these cases, the minimal p-values are 0.008 and 0.156, respectively. Hence, for all IE tests associated with the deep learning models, we reject the null hypothesis in 69 tests out of 70.

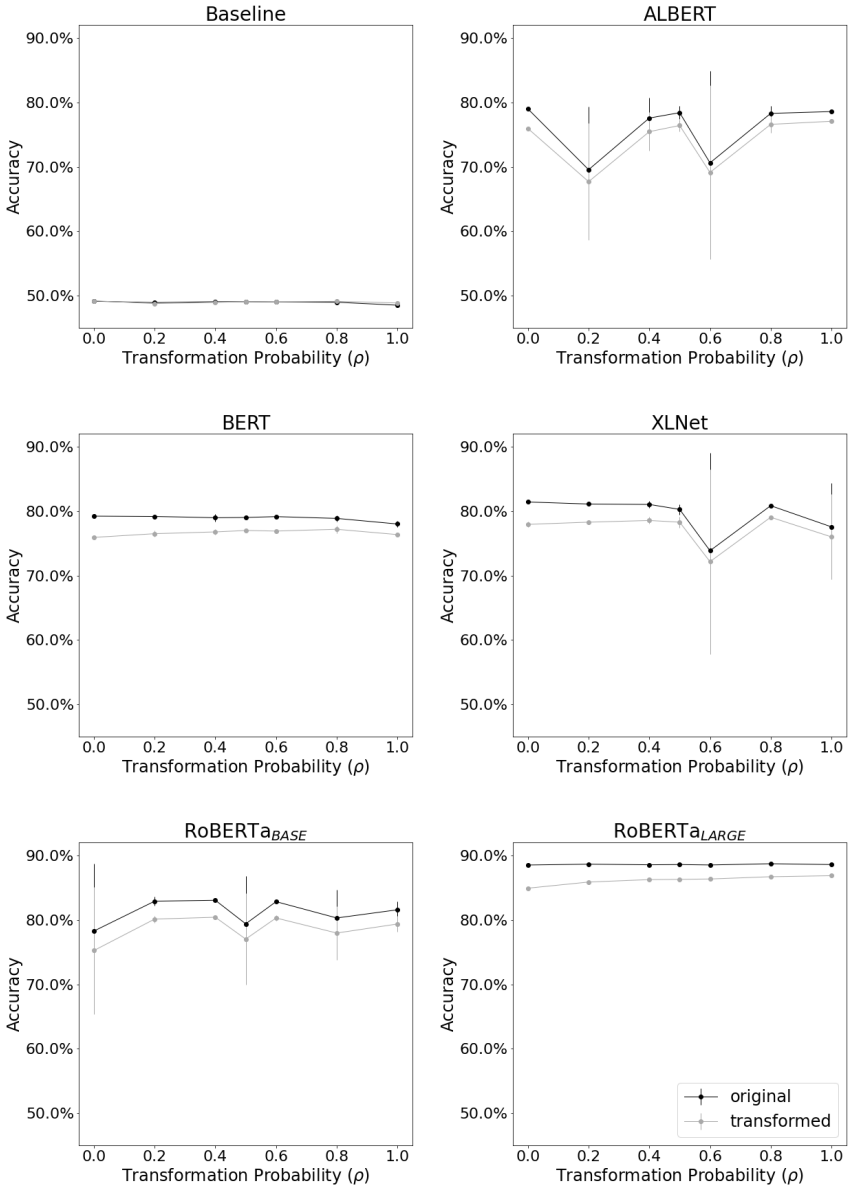

Figure 4.5: *MNLI results. In the x-axis we have different choices of transformation probabilities in training. The y-axis displays the accuracy. Each point represents the average accuracy in five runs. The vertical lines display the associated standard deviation. The black and grey lines represent the values for the original and transformed test sets, respectively.*

The empirical evidence shows that the deep learning models are not invariant under equivalence. This indicates that although these models present an impressive inference capability, they still lack the skill of producing the same deduction based on different sentences with the same meaning. After rejecting the null hypothesis when using different transformation probabilities, we are convinced that this is not a simple data acquisition problem. Since we are seeing the same pattern for almost

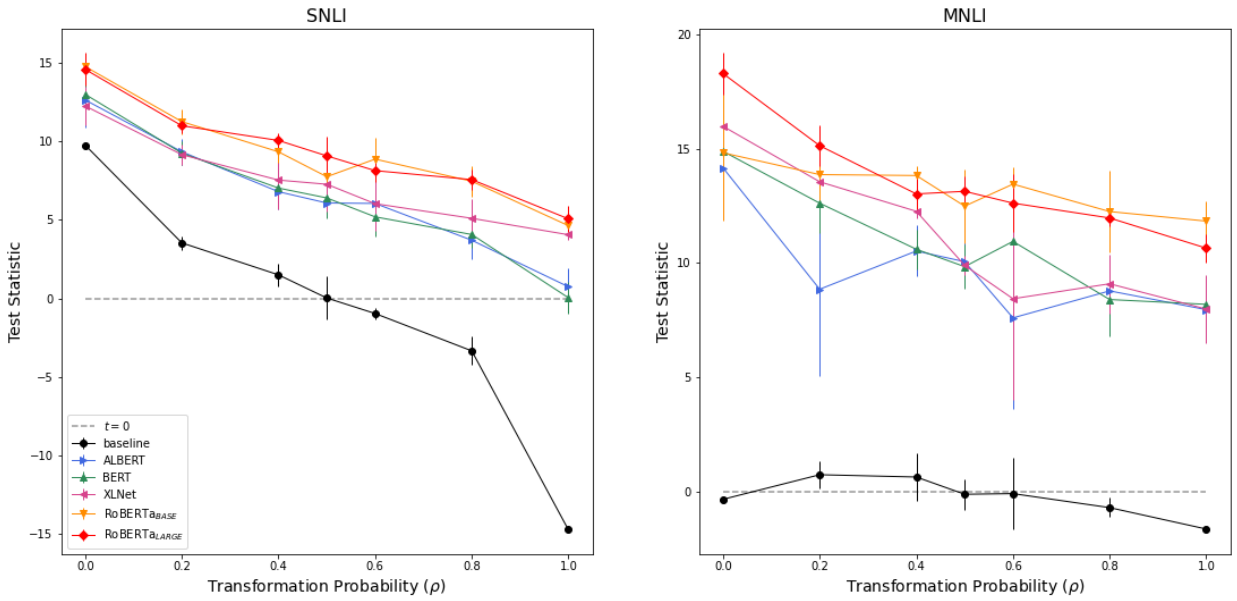

Figure 4.6: *Test statistics from the IE test for all models. In the x-axis we have different choices of transformation probabilities used in training. The y-axis displays the values for the test statistic. Each point represents the average test statistics in five paired t-tests. The vertical lines display the associated standard deviation.*

all models in both datasets, it seems that the absence of the invariance under equivalence propriety is a feature in the transformer based models.

4.5.3 Experimental Finding: Model Robustness

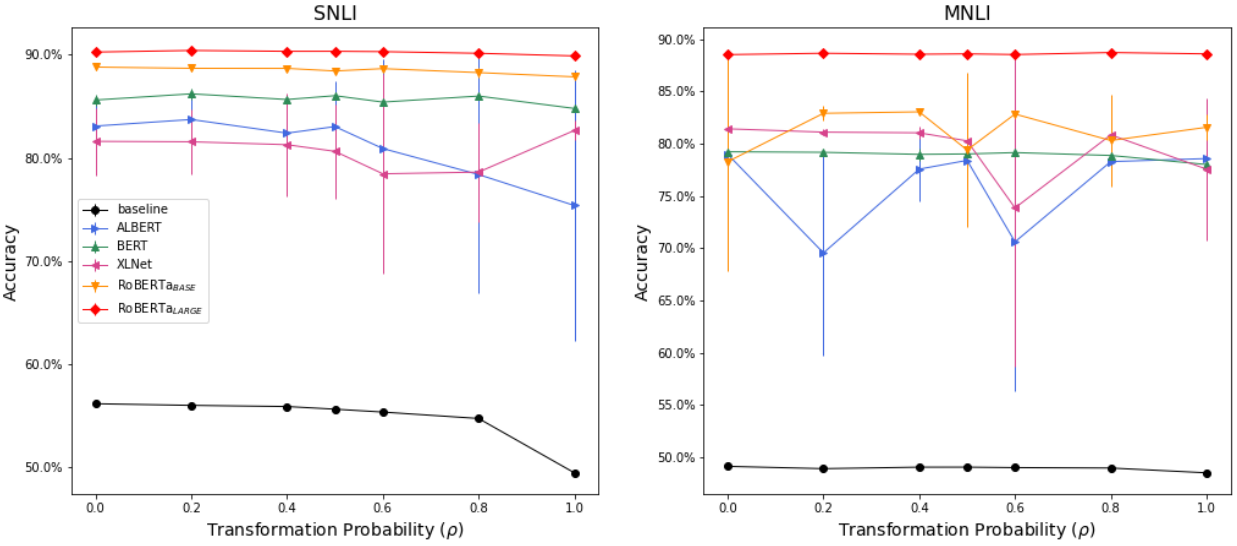

Figure 4.7: *Models' accuracy on the original test set. In the x-axis we have different choices of transformation probabilities used in training. The y-axis displays the accuracy. Each point represents the average accuracy in five runs. The vertical lines display the associated standard deviation.*

One possible interpretation of the transformation function is that this alteration is a noise that can be added to the training data. Although this type of noise is imperceptible for humans, it can force the machine learning model to make false predictions. By this interpretation, the transformation function is an "attack", a "challenge", or an "adversary" to the model [Dasgupta et al., 2018, Liu et al., 2019a, McCoy et al., 2019, Naik et al., 2018, Nie et al., 2018, Richardson et al.,

2020, Yanaka et al., 2019, Zhu et al., 2018]. Along these lines, a robust model is one that consistently produces high test accuracy even when we add different proportions of noised observations in training; in other words, a robust model should combine higher prediction power and low accuracy variation. Given a model $\mathcal{M}$ and a dataset, we train the model using the (φ, ρ) data generation process for different values of ρ and obtain a sample of test accuracies (accuracies associated with the original test set). Here, we use the *signal-to-noise ratio* (SNR) as a measure of robustness. Let $\hat{\mu}_{\mathcal{M}}$ and $\hat{\sigma}_{\mathcal{M}}$ be the sample mean and standard deviation, respectively, we define:

$$\text{SNR}_{\mathcal{M}} = \frac{\hat{\mu}_{\mathcal{M}}}{\hat{\sigma}_{\mathcal{M}}}. \tag{4.14}$$

This statistical measure has an intuitive interpretation: the numerator represents the model's overall performance when noise is added; and the denominator indicates how much the model's predictive power changes when different levels of noise are present. Since this score can be given for each model and dataset, we rank the models by their robustness under a transformation function by averaging the model's SNR on different datasets (Table 4.2).

	Signal-to-Noise Ratio		
Model	SNLI	MNLI	Average
RoBERTa$_{LARGE}$	393.1	569.3	481.2
BERT	151.1	151	151.1
RoBERTa$_{BASE}$	222.6	16.5	119.5
baseline	**24.8**	**212.6**	**118.7**
XLNet	16.7	12.8	14.8
ALBERT	10.8	10.7	10.8

Table 4.2: *Ranked models according to the SNR metric. In this case, the noise is the synonym substitution transformation.*

From the perspective of robustness - based only on the SNR metric - the networks XLNet and ALBERT are *worse* than the baseline. Although both deep learning models present, on average, a higher accuracy compared to the baseline; they are not able to maintain high accuracy when different quantities of altered sentences are included in the training data. In contrast, the baseline produces a low yet consistent test accuracy (Figure 4.7). Among all models, BERT and ROBERTA appear to be the most robust ones. As seen in Table 4.2, BERT shows consistent performance in both datasets; and ROBERTA$_{BASE}$ shows an almost unchanged behavior on the SNLI dataset. Clearly, ROBERTA$_{LARGE}$ stands out when compared to the rest. This model was not only able to obtain a higher accuracy on both datasets but also it had maintained a high performance regardless of the choice of ρ.

It has been reported that increasing model size leads to large improvement in different natural language processing tasks [Devlin et al., 2019, Radford et al., 2019]. We conduct a similar investigation regarding robustness by checking whether SNR improves as the model size increases. Since our experiments were not initially designed to measure robustness, the output of our tests can only provide a glimpse of the relationship between model size and robustness. As seen in Figure 4.8, there is a positive tendency between these factors. However, this tendency is just a speculative result. Due to the fact that the sample size is too small (only six models are analyzed), the tendency heavily depends on some individual points. For example, the correlation between SNR and model size is 0.88 when all points are present. When we remove the point associated with the ROBERTA$_{LARGE}$ model, the correlation drops to 0.20. To reach a decisive conclusion regarding this tendency more investigation is needed. We should either add the larger version of each deep learning model or add models composed of 150-300M parameters.

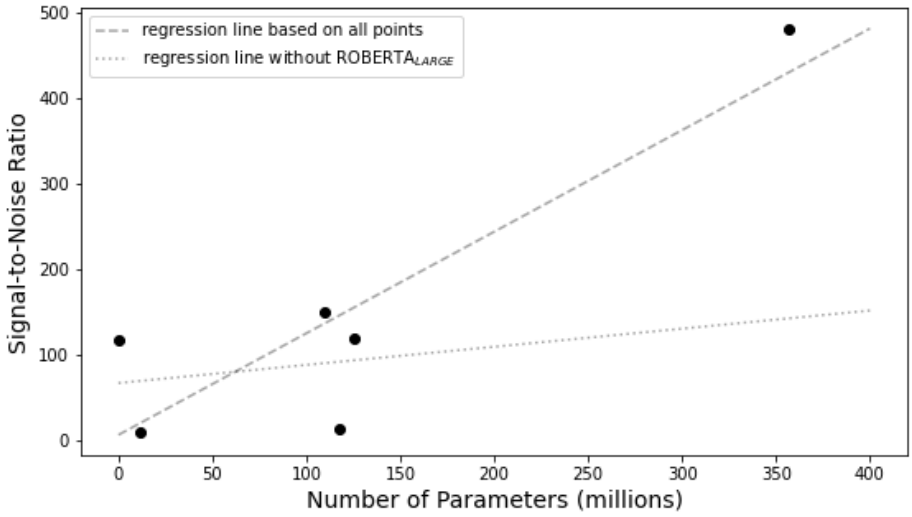

Figure 4.8: *Robustness as a function of model size. Robustness is measured as the average SNR on the datasets SNLI and MNLI. Although we observe a linear relationship between SNR and model size, this relationship is heavily influenced by the results related to the model ROBERTA$_{LARGE}$.*

4.5.4 Discussion and Limitations

The result of the IE tests applied to the deep learning models shows that these models can have different deductions for sentences with the same meaning. What is surprising about the deep learning models is that they perform very differently in the test sets $\mathcal{D}_{Te}$ and $\mathcal{D}^{\varphi}_{Te}$ *even when sufficient amount of transformed observations are added in training.* We offer a possible explanation for this phenomenon. All transformer based models are trained in two stages: they are pre-trained using unlabeled text and fine-tuned for the NLI task. *The* (φ, ρ) *data generation process only affects the fine-tuning phase.* Hence, we believe that it is possible to reduce the accuracy difference between the two versions of the test data by allowing the addition of transformed sentences in the pre-training stage. However, since pre-training these large models is a computationally expensive endeavor, the results presented here are relevant. It seems that we cannot correct how these models perform inference by just adding examples in the fine-tuning phase.

There is a limitation in the present analysis. As stated in the methodological considerations (Subsection 4.4.3), we have used a small random sample of the training data (small compared to the original size of the SNLI and MNLI dataset). Hence, one can argue that the results associated with the deep learning models are restricted to small NLI datasets. Further research is needed to verify this claim. We can take bigger samples (samples with more than 50K observations) and apply the IE test to verify if there is a *minimum training size* needed to correct the biases of the deep learning models in the fine-tuning phase. Even if such minimum size exists, our results expose some limitations of the current NLI models. A more intensive investigation is needed in order to understand the full picture of those limits.

4.6 Related Work

There is a growing literature highlighting the inadequacy of models trained on the benchmark NLI datasets. In this line, Glockner et al. [2018] have shown that the models trained on SNLI and MNLI perform poorly on a test set composed of observations that rely only on lexical knowledge. In a similar fashion, Yanaka et al. [2019] have shown that models trained on SNLI and MNLI perform poorly on a test set composed of specific examples based on monotonic reasoning. And Richardson et al. [2020] have exposed that models trained on the benchmark datasets are weak in solving some simple logical tasks. The clear limitation of this type of analysis is that when we observe a poor performance in a new test set, we cannot be sure if this is a limitation of the model

or just a sampling problem. Geiger et al. [2019] have highlighted this methodological problem by means of the notion of *fairness*. They argued that a generalization evaluation method is not fair if the model was not trained on a sample that does not support the required generalization. The IE test is an alternative tool to approach the fairness problem. Instead of defining "fair datasets" as done by Geiger et al. [2019], we define the (φ, ρ) data generation process and leave to each researcher to choose a fair value for ρ.

One work that presents an analysis similar to ours is the one presented by Liu et al. [2019a]. In that paper, the authors have proposed an analysis of the limitation of datasets and models from the NLI literature by defining a collection of transformation functions ("challenges to benchmarks") and a training procedure that includes transformed observations ("inoculation by fine-tuning"). The IE test can be seen as a generalization of this type of analysis. In the process of inoculation, the authors fix a "small number" of transformed examples for training and compare model performance on two test sets ignoring any statistical significance test. By contrast, our method allows any portion of the training data to be altered and it also measures the performance difference in a statistically sound manner.

Chapter 5

Conclusions

In this thesis, we have explored different logic based properties that the NLI models should have. This exploration was done by employing different methods: first, the novel contradiction detection dataset based on logical operators (Chapter 3), then the new task of inference generation called *masked inference* (Chapter 3); and finally, the *invariance under equivalent test* (Chapter 4). After the introduction of such methods what have we learned about the properties and limitations of the NLI models? What are the promising research tracks for this field? We address these question in this final chapter.

5.1 Synthetic Data: Lessons Learned and New Paths

In Chapter 3 we have described a general method for using a formal language to generate NLI observations. This approach allowed us to create datasets for the NLI task and the related sub tasks. One of the main advantages of this method is the possibility of using the same template to create datasets in different natural languages, and, at the same time, being able to control what causes the logical relation between sentences.

We have observed that, regarding the standard NLI task, almost all types of inference based on logical connectives are solved by a machine learning model (counting is the only exception). On the other hand, when we move to the proposed inference generation task, the same type of deduction becomes harder to master. As stated in Section 3.6, *in order to strengthen the results obtained from a synthetic data, we should, in some capacity, associate the synthetic sentences with examples from everyday speech.*

One interesting idea in this direction was recently suggested by Joshi et al. [2020]. Influenced by our work, the authors of that paper have created a taxonomy of the various reasoning tasks covered by NLI datasets. They have used some logical tasks displayed on Section 3.4.2 as the base for the following taxonomic categories: "negation" (simple negation), "Boolean" (Boolean coordination), "quantifier" and "comparative". With this new taxonomy, they have hired crowdsource annotators to label 10K random observations from the MNLI dataset. For example, the following observation was marked as *quantifier*:

$P =$ `Some travelers add Molokai and Lanai to their itineraries.`

$H =$ `No one decides to go to Molokai and Lanai.`

This is one way to use the inference rules as an inspirational source of evaluation. In this case, Joshi et al. [2020] offer a procedure of labeling the observations from a benchmark dataset that most resembles the synthetic examples. The main disadvantage of this approach is the cost of reproducing the labelling process for a new dataset. For each new data, we need to pay a new group of crowdsource annotators, check the quality of the labels, check the labelling agreement between annotators, etc.

An alternative research path is the formulation of some type of *everyday speech inference rules*. In order to reflect more truthfully some deductions that appear in natural language, we can define a NLI counterpart of the rules displayed in Section 3.2. For example, we can modify the inference rule (3.1) as follows:

$$\frac{P \models_g H_1 \quad P \models_g H_2}{P \models_g H_1 \texttt{ and } H_2} \ (\wedge_I^*), \tag{5.1}$$

where $\models_g$ is the entailment relation derived from a classifier g (1.1), and P, H_1, H_2 are natural language sentences. Given a test set $\mathcal{D}_{Te}$, we can score a classifier based on its adequacy to each inference rule. For example, for the rule (5.1) we define the following score function

$$s(\wedge_I^*, g, \mathcal{D}_{Te}) = \frac{1}{|S|} \sum_{(P,H_1,H_2)\in S} I(P \models_g H_1 \texttt{ and } H_2)$$

$$S = \{(P, H_1, H_2) \in \mathcal{D}_{Te}^p \times \mathcal{D}_{Te}^h \times \mathcal{D}_{Te}^h : P \models_g H_1, P \models_g H_2\}, \tag{5.2}$$

where $\mathcal{D}_{Te}^p$ and $\mathcal{D}_{Te}^h$ are the sets of premises and hypothesis from $\mathcal{D}_{Te}$, respectively.

More generally, by selecting n inference rules $r_1, \ldots, r_n$ and defining a score function for each one, it is possible to construct a *structural inference score*:

$$score(g, \mathcal{D}_{Te}) = \beta_0 \, \widehat{acc}(g) + \beta_1 \, s(r_1, g, \mathcal{D}_{Te}) + \cdots + \beta_n \, s(r_n, g, \mathcal{D}_{Te}), \tag{5.3}$$

where $\beta_0, \beta_1, \ldots \beta_n \in [0, 1]$ sum up to one. With such score function, we can create an experimental setting to answer some relevant questions like:

- Which inference rules are mastered by the current deep learning models? The same ones as those described here? It is easier to obtain a high score on the rules solved in the synthetic dataset?
- Can we define a sequence of inference rules that yields a low score for undesirable classifiers (e.g., the classifier that relies only on the premise, the classifier that returns the same output for every sentence pair, etc.)?
- How the properties of the training dataset affects the structural inference score (number of observations, input length, vocabulary size)?
- Does a classifier have a consistent score across different NLI datasets?
- There is a positive correlation between model complexity and structural inference score?

Although the main difficulty of this research path lies in the selection of the rules and the construction of the score function for each rule, we believe that this evaluation technique constitutes a reasonable alternative to what Joshi et al. [2020] are proposing. It can be applied to multiple NLI datasets without the cost of crowdsourcing labor and it is a natural continuation of the work presented in this thesis.

5.2 Invariance under Equivalence and Bias

In Chapter 4, we have presented the most important contribution of the thesis: the invariance under Equivalence (IE) test, a method to evaluate whether an NLI model can make the same type of inference for equivalent text inputs. By using an equivalent transformation function based on synonym substitution we have tested the state-of-the-art models and observed that these models

show two different inferences for two sentences with the same meaning. We have also ranked these models by their performance robustness when transformed data is introduced.

The results presented in that chapter show only a partial picture of the limitations of the current NLI models. Our results can be improved using the IE test in a broader study to investigate whether the IE property is violated for the other equivalent relations mentioned in Section 4.2. The data generation process presented in Section 4.3.1 can be used to further analyze how model complexity affects robustness, and, using the IE test, we can investigate whether by changing the size of the training data, we are able to correct any biases present in pre-trained models. Another future venue of investigation consists of exploring whether the recently proposed hybrid NLI model [Kalouli et al., 2020] violates the IE property. Regarding the synonym substitution transformation, we expect a better performance from the hybrid model. This new model has an inference engine with access to the WordNet; hence, at least in theory, it should have an advantage over the deep learning models.

From the theoretical side, there is still space for improvement. The IE test is based on resampling an altered version of the dataset multiple times. When we combine this strategy with models that can have hundreds of millions of parameters we can easily encounter hardware and time limitations. Thus, a natural continuation of this research path is to combine the resampling method with the reduction techniques that can increase training and inference speed.

One of the main advantages of the IE test is that it can be formulated for other reading comprehension tasks that fall under the name NLU. Since equivalence in natural language is based on the general phenomenon of meaning identity, it is reasonable to extend the IE test for any text classification task. This extension should be carefully established because the definition of an adequate transformation function is task-dependent. After selecting a transformation, the IE test can be used to find a discrepancy in model performance. For example, the IE test can be used to check for any biases in the pre-training of the transformer based models. In this way, we hope that the methods developed here can contribute to the general discussion on the limitations of the deep learning models used in NLP.

Appendix A

Templates for the Dataset Presented in Section 3.4

A.1 Simple Negation

Contradiction Templates

$$\begin{aligned} P &= V(x_1, y_1), \ldots, V(x_n, y_n) \\ H &= \neg V(x_i, y_i), \end{aligned} \tag{A.1}$$

where $i \in \{1, \ldots, n\}$.

Non-Contradiction Templates

$$\begin{aligned} P &= V(x_1, y_1), \ldots, V(x_n, y_n) \\ H &= \neg V(x_i, y^*), \end{aligned} \tag{A.2}$$

where $i \in \{1, \ldots, n\}$ and y^* is new.

$$\begin{aligned} P &= V(x_1, y_1), \ldots, V(x_n, y_n) \\ H &= \neg V(x^*, y_i), \end{aligned} \tag{A.3}$$

where $i \in \{1, \ldots, n\}$ and x^* is new.

A.2 Boolean Coordination

Contradiction Templates

$$\begin{aligned} P &= V(x_1, y) \wedge V(x_2, y) \wedge \cdots \wedge V(x_n, y) \\ H &= \neg V(x_i, y), \end{aligned} \tag{A.4}$$

where $i \in \{1, \ldots, n\}$.

$$P = V(x, y_1) \wedge V(x, y_2) \wedge \cdots \wedge V(x, y_n)$$
$$H = \neg V(x, y_i), \tag{A.5}$$

where $i \in \{1, \ldots, n\}$.

Non-Contradiction Templates

$$P = V(x_1, y) \wedge V(x_2, y) \wedge \cdots \wedge V(x_n, y)$$
$$H = \neg V(x^*, y^*), \tag{A.6}$$

where either $x^* \in \{x_1, \ldots, x_n\}$ and y^* is new, or x^* is new and $y^* = y$.

$$P = V(x, y_1) \wedge V(x, y_2) \wedge \cdots \wedge V(x, y_n)$$
$$H = \neg V(x^*, y^*), \tag{A.7}$$

where either x^* is new and $y^* \in \{y_1, \ldots, y_n\}$, or $x^* = x$ and y^* is new.

A.3 Quantification

Contradiction Templates

$$P = (\forall x \in Pe)\ (V(x, y_1) \wedge \cdots \wedge V(x, y_n))$$
$$H = \neg V(x_1, y_i), \tag{A.8}$$

where $x_1 \in Pe$ and $y_i \in \{y_1, \ldots, y_n\}$.

$$P = (\forall x \in Pe)(\forall y \in Pl)\ V(x, y)$$
$$H = \neg V(x_1, y_1), \tag{A.9}$$

where $x_1 \in Pe$ and $y_1 \in Pl$.

$$P = (\forall x \in Pe)(\forall y \in Pe)\ V(x, y)$$
$$H = \neg V(x_1, y_1), \tag{A.10}$$

where $x_1, y_1 \in Pe$.

$$P = (\forall x \in Pe)(\forall y \in Pe)(\forall z \in Pl)\ (V(x, y) \wedge V(x, z))$$
$$H = \neg V(x_1, y_1), \tag{A.11}$$

where $x_1, y_1 \in Pe$.

$$\begin{aligned} P &= (\forall x \in Pe)(\forall y \in Pe)(\forall z \in Pl)\ (V(x,y) \wedge V(x,z)) \\ H &= \neg V(x_1, z_1), \end{aligned} \tag{A.12}$$

where $x_1 \in Pe$ and $z_1 \in Pl$.

Non-Contradiction Templates

$$\begin{aligned} P &= (\forall x \in Pe)\ V(x,y_1) \wedge \cdots \wedge V(x,y_n) \\ H &= \neg V(x_1, y^*), \end{aligned} \tag{A.13}$$

where $x_1 \in Pe$ and y^* is new.

$$\begin{aligned} P &= (\forall x \in Pe)(\forall y \in Pl)\ V(x,y) \\ H &= \neg V(x_1, z_1), \end{aligned} \tag{A.14}$$

where $x_1, z_1 \in Pe$.

$$\begin{aligned} P &= (\forall x \in Pe)(\forall y \in Pe)\ V(x,y) \\ H &= \neg V(x_1, z_1), \end{aligned} \tag{A.15}$$

where $x_1 \in Pe$ and $z_1 \in Pl$.

$$\begin{aligned} P &= (\exists x \in Pe)(\forall y \in Pe)(\forall z \in Pl)\ (V(x,y) \wedge V(x,z)) \\ H &= \neg V(x_1, y_1), \end{aligned} \tag{A.16}$$

where $x_1, y_1 \in Pe$.

$$\begin{aligned} P &= (\exists x \in Pe)(\forall y \in Pe)(\forall z \in Pl)\ (V(x,y) \wedge V(x,z)) \\ H &= \neg V(x_1, z_1), \end{aligned} \tag{A.17}$$

where $x_1 \in Pe$ and $z_1 \in Pl$.

A.4 Definite Description

In order to distinct between the equality relation in the template and meta language, we use $\approx$ to denote the equality symbol in the template language.

Contradiction Templates

$$\begin{aligned} P &= x \approx \iota y(\forall z \in Pl)\ V(y,z) \\ H &= \neg V(x, z_1), \end{aligned} \tag{A.18}$$

where $z_1 \in Pl$.

$$\begin{aligned} P &= x \approx \iota y(\forall z \in Pe)\ V(y,z) \\ H &= \neg V(x, z_1), \end{aligned} \tag{A.19}$$

where $z_1 \in Pe$.

Non-Contradiction Templates

$$\begin{aligned} P &= x \approx \iota y(\forall z \in Pl)\ V(y,z) \\ H &= \neg V(x^*, z_1), \end{aligned} \tag{A.20}$$

where x^* is new and $z_1 \in Pl$.

$$\begin{aligned} P &= x \approx \iota y(\forall z \in Pe)\ V(y,z) \\ H &= \neg V(x^*, z_1), \end{aligned} \tag{A.21}$$

where x^* is new and $z_1 \in Pe$.

A.5 Comparatives

For a set $\{x_1, \ldots, x_n\}$ and a binary relation R, we use $chain(\{x_1, \ldots, x_n\}, R)$ to denote the facts $x_1 R x_2,\ x_2 R x_3,\ \ldots,\ x_{n-1} R x_n$. We also use $yR\{x_1, \ldots, x_n\}$ to denote $yRx_1,\ yRx_2,\ \ldots,\ yRx_n$.

Contradiction Templates

$$\begin{aligned} P &= chain(\{x_1, \ldots, x_n\}, >) \\ H &= x_j > x_i, \end{aligned} \tag{A.22}$$

where $1 \leq i < j \leq n$.

$$\begin{aligned} P &= chain(\{x_1, \ldots, x_n\}, \geq),\ x_n > y \\ H &= y > x_i, \end{aligned} \tag{A.23}$$

where $x_i \in \{x_1, \ldots x_n\}$.

$$\begin{aligned} P &= xR\{x_1, \ldots, x_n\},\ y \geq x \\ H &= x_i > y, \end{aligned} \tag{A.24}$$

where $x_i \in \{x_1, \ldots x_n\}$.

Non-Contradiction Templates

$$\begin{aligned} P &= chain(\{x_1, \ldots, x_n\}, >) \\ H &= x_j > x_i, \end{aligned} \tag{A.25}$$

where $1 \leq j < i \leq n$.

$$\begin{aligned} P &= chain(\{x_1, \ldots, x_n\}, \geq),\ x_n > y \\ H &= x_i > y, \end{aligned} \tag{A.26}$$

where $x_i \in \{x_1, \ldots x_n\}$.

$$\begin{aligned} P &= xR\{x_1, \ldots, x_n\},\ y \geq x \\ H &= y > x_i, \end{aligned} \tag{A.27}$$

where $x_i \in \{x_1, \ldots x_n\}$.

A.6 Counting

Contradiction Templates

$$\begin{aligned} P &= (\exists_{=n} y \in Pe)\ V(x, y) \\ H &= V(x, y_1) \wedge \cdots \wedge V(x, y_{n+1}), \end{aligned} \tag{A.28}$$

where $y_1, \ldots, y_{n+1} \in Pe$.

$$\begin{aligned} P &= (\exists_{=n} y \in Pl)\ V(x, y) \\ H &= V(x, y_1) \wedge \cdots \wedge V(x, y_{n+1}), \end{aligned} \tag{A.29}$$

where $y_1, \ldots, y_{n+1} \in Pl$.

$$\begin{aligned} P &= (\exists_{=n} z \in Pl)(\exists_{=m} y \in Pe)\ (V(x, z) \wedge V(x, y)) \\ H &= V(x, z_1) \wedge \cdots \wedge V(x, z_{n+1}), \end{aligned} \tag{A.30}$$

where $z_1, \ldots, z_{n+1} \in Pl$.

$$\begin{aligned} P &= (\exists_{=n} z \in Pl)(\exists_{=m} y \in Pe)\ (V(x, z) \wedge V(x, y)) \\ H &= V(x, y_1) \wedge \cdots \wedge V(x, y_{m+1}), \end{aligned} \tag{A.31}$$

where $y_1, \ldots, y_{m+1} \in Pe$.

Non-Contradiction Templates

$$\begin{aligned} P &= (\exists_{=n} y \in Pe)\ V(x,y) \\ H &= V(x,y_1) \wedge \cdots \wedge V(x,y_k), \end{aligned} \tag{A.32}$$

where $k < n$ and $y_1, \ldots, y_k \in Pe$.

$$\begin{aligned} P &= (\exists_{=n} y \in Pl)\ V(x,y) \\ H &= V(x,y_1) \wedge \cdots \wedge V(x,y_k), \end{aligned} \tag{A.33}$$

where $k < n$ and $y_1, \ldots, y_k \in Pl$.

$$\begin{aligned} P &= (\exists_{=n} z \in Pl)(\exists_{=m} y \in Pe)\ (V(x,z) \wedge V(x,y)) \\ H &= V(x,z_1) \wedge \cdots \wedge V(x,z_k), \end{aligned} \tag{A.34}$$

where $k < n$ and $z_1, \ldots, z_k \in Pl$.

$$\begin{aligned} P &= (\exists_{=n} z \in Pl)(\exists_{=m} y \in Pe)\ (V(x,z) \wedge V(x,y)) \\ H &= V(x,y_1) \wedge \cdots \wedge V(x,y_k), \end{aligned} \tag{A.35}$$

where $k < m$ and $y_1, \ldots, y_k \in Pe$.

Appendix B

Templates for the Dataset Presented in Section 3.5

Notation

First, some notation: we use $\pm$ to denote the occurrence or not of the negation symbol $\neg$; we also use l_i to indicate the same occurrence (or absence) of the negation symbol in a formula; and Q_i is used to denote a quantifier ($\exists$ or $\forall$).

B.1 Boolean Coordination

Introduction of the Conjunction

$$\begin{aligned} P &= l_1\ V(x_1, y_1), \ldots, l_n\ V(x_n, y_n) \\ H &= l_i\ V(x_i, y_i)\ \texttt{[MASK]}\ l_j\ V(x_j, y_j) \\ Y &= \wedge, \end{aligned} \tag{B.1}$$

where $i, j \in \{1, \ldots, n\}$.

Introduction of the Disjunction

$$\begin{aligned} P &= l_1\ V(x_1, y_1), \ldots, l_n\ V(x_n, y_n) \\ H &= l_i\ V(x_i, y_i)\ \texttt{[MASK]}\ \pm\ V(x^*, y^*) \\ Y &= \vee, \end{aligned} \tag{B.2}$$

where $i \in \{1, \ldots, n\}$, and both x^* and y^* are new.

B.2 Quantifier Reasoning

Introduction of the Existential

$$\begin{aligned} P &= l_1\ V(x_1, y_1), \ldots, l_n\ V(x_n, y_n) \\ H &= \texttt{[MASK]}\ x\ l_i\ V(x, y_i) \\ Y &= \exists, \end{aligned} \tag{B.3}$$

where $i \in \{1, \ldots, n\}$.

Negation of the Existential

$$\begin{aligned} P &= Q_1 x\ V(x, y_1), \ldots, \forall x\ V(x, y_i), \ldots, Q_n x\ V(x, y_n) \\ H &= \neg\ \texttt{[MASK]}\ x\ \neg\ V(x, y_i) \\ Y &= \exists, \end{aligned} \tag{B.4}$$

where $i \in \{1, \ldots, n\}$.

Negation of the Universal

$$\begin{aligned} P &= Q_1 x\ V(x, y_1), \ldots, \neg V(x_i, y_i), \ldots, Q_n x\ V(x, y_n) \\ H &= \neg\ \texttt{[MASK]}\ x\ V(x, y_i) \\ Y &= \forall, \end{aligned} \tag{B.5}$$

where $i \in \{1, \ldots, n\}$.

B.3 Counting

Introduction of the Counting Quantifier

$$\begin{aligned} P &= l_1\ V(x_1, y_1), \ldots, l_n\ V(x_m, y_m), V(x, z_1), \ldots, V(x, z_n) \\ H &= \exists_{\texttt{[MASK]}}\ z\ V(x, z) \\ Y &= n, \end{aligned} \tag{B.6}$$

where $z_1, \ldots, z_n \in Pl$, and $x \notin \{x_1, \ldots, x_m\}$.

Appendix C

Synonym Substitution Examples

The distinction between sound and unsound transformations is based on subjective judgments. What has guided us to determine an alteration as sound is how the transformation affects the associated label. Hence, we have allowed modifications that produce minor grammatical errors (e.g. "a adult male"). When we observe that either the overall logical relationship is disturbed or the sentence is unusual we classify the transformation result as unsound.

Original Pair	Transformed Pair
A man and his son riding bikes down the sidewalk. The man and the boy were in town.	A adult male and his boy riding bikes down the pavement. The adult male and the male child were in town.
A male and female are asleep on a couch with a large black dog as four people sit at a table behind them. The male and female that are asleep on the couch are in a relationship.	A male and female are asleep on a sofa with a large black domestic dog as four people sit at a table behind them. The male and female that are asleep on the sofa are in a human relationship.
A woman in a blue winter jacket is pushing a shopping cart through snow. A homeless woman is eating a hamburger.	A adult female in a blue wintertime jacket is pushing a shopping cart through snowfall. A homeless person adult female is eating a burger.
Dark image of two people inside a fish market. There are fish.	Dark mental image of two people inside a fish marketplace. There are fish.

Table C.1: *Sound transformations for SNLI.*

Original Pair	Transformed Pair
A woman and child are on a boat and the woman is looking out into the ocean through a scope. A lady and a child are on a boat and the lady is looking out into the ocean through a scope.	A adult female and kid are on a boat and the adult female is looking out into the ocean through a range. A lady and a kid are on a boat and the lady is looking out into the ocean through a range.
A man in a white shirt holds a microphone. A band is playing on a stage.	A adult male in a white shirt holds a mike. A band is playing on a phase.
A cattle dog nips the leg of an animal. A dog nips a cow.	A cows domestic dog shot the leg of an creature. A domestic dog shot a moo-cow.
A band of people playing brass instruments is performing outside. A jazz funeral is taking place.	A band of people playing brass instruments is performing outside. A wind funeral is taking place.

Table C.2: *Unsound transformations for SNLI.*

Original Pair	Transformed Pair
Another majestic view of the city is from a charming park Miradouro de Santa Luzia just down the hill from the castle. The castle is on the highest hill in the city.	Another olympian view of the metropolis is from a charming parkland Miradouro de Santa Luzia just down the hill from the palace. The palace is on the highest hill in the metropolis.
The agency cites the clean air act 42 usc. The agency discusses the clean air act in chapter 3 of the book.	The office cites the clean air enactment 42 usc. The office discusses the clean air enactment in chapter 3 of the book.
Renovated in 2000 this full-service resort fronts a tremendous swimming and snorkeling beach with dozens of turtles. The resort was renovated in 2000.	Renovated in 2000 this full-service resort hotel fronts a tremendous swimming and snorkeling beach with lots of turtles. The resort hotel was renovated in 2000.
Bolstered by a new influx of immigrants to meet the rubber and tin booms of the 1920s, non-malays now slightly outnumbered the indigenous population. The population of malays to non-malays was equal and all the work was shared.	Bolstered by a new inflow of immigrants to meet the india rubber and tin booms of the twenties, non-malays now slightly outnumbered the indigenous population. The population of malays to non-malays was equal and all the work was shared.

Table C.3: *Sound transformations for MNLI.*

Original Pair	Transformed Pair
They might as well steal it then they don't have to pay taxes on it. Taxes are entirely irrelevant.	They power as well steal it then they don't have to salary taxes on it. Taxes are entirely irrelevant.
You know you writers are coming you know you're having a hard time here. The writers are having a hard time keeping the show interesting.	You know you author are coming you know you're having a difficult time here. The author are having a difficult time keeping the show interesting.
Pigs are sociable loving and a hell of a lot brighter than dalmatians. Pigs are very smart.	Pigs are sociable loving and a inferno of a lot brighter than dalmatians. Pigs are very smart.
2 billion in benefits to over 13 million recipients. A couple of billion in benefits for the public to do whatever they want with.	Deuce billion in benefits to over 13 million recipients. A duo of billion in benefits for the public to do whatever they want with.

Table C.4: *Unsound transformations for MNLI.*

Appendix D

Hyperparameter Search

Regarding the experiments form Chapter 4, here we present all the hyperparameters used in training, the associated search space, and the selected value for each dataset. The hyperparameter values were selected using the random search algorithm.

Gradient Boosting

Hyperparameter	Search Space	Selected Value for SNLI	Selected Value for MNLI
number of estimators	$\{10, \ldots, 30\}$	26	29
max depth	$\{2, \ldots, 20\}$	15	8
reg alpha	$[0.05, 1.0]$	0.65	0.75
reg gamma	$[0.05, 1.0]$	0.15	0.7
learning rate	$[0.05, 1.0]$	0.55	0.4
subsample	$[0.05, 1.0]$	1.0	1.0
colsample bytree	$[0.05, 1.0]$	0.95	0.9

Table D.1: *Best hyperparameter assignments for the gradient boosting classifier.*

ALBERT

Hyperparameter	Search Space	Selected Value for SNLI	Selected Value for MNLI
number of epochs	$\{1, 2, 3\}$	2	2
max input length	$\{50, 60, \ldots, 200\}$	90	130
learning rate	$[5 \times 10^{-5}, 1 \times 10^{-4}]$	6.7×10^{-5}	6.7×10^{-5}
weight decay	$[0, 0.01]$	1.1×10^{-3}	6.6×10^{-3}
adam epsilon	$[1 \times 10^{-8}, 1 \times 10^{-7}]$	3×10^{-8}	2×10^{-8}
max grad norm	$[0.9, 1.0]$	0.91	0.97

Table D.2: *Best hyperparameter assignments for ALBERT.*

BERT

Hyperparameter	Search Space	Selected Value for SNLI	Selected Value for MNLI
number of epochs	$\{1, 2, 3\}$	3	2
max input length	$\{50, 60, \ldots, 200\}$	130	90
learning rate	$[5 \times 10^{-5}, 1 \times 10^{-4}]$	7.7×10^{-5}	7.2×10^{-5}
weight decay	$[0, 0.01]$	2.2×10^{-3}	3.3×10^{-3}
adam epsilon	$[1 \times 10^{-8}, 1 \times 10^{-7}]$	1×10^{-7}	3×10^{-8}
max grad norm	$[0.9, 1.0]$	0.94	1.0

Table D.3: *Best hyperparameter assignments for BERT.*

XLNet

Hyperparameter	Search Space	Selected Value for SNLI	Selected Value for MNLI
number of epochs	$\{1, 2, 3\}$	1	2
max input length	$\{50, 60, \ldots, 200\}$	100	100
learning rate	$[5 \times 10^{-5}, 1 \times 10^{-4}]$	6.7×10^{-5}	6.1×10^{-5}
weight decay	$[0, 0.01]$	0.01	4.4×10^{-3}
adam epsilon	$[1 \times 10^{-8}, 1 \times 10^{-7}]$	4×10^{-8}	1×10^{-7}
max grad norm	$[0.9, 1.0]$	1.0	0.9

Table D.4: *Best hyperparameter assignments for XLNet.*

RoBERTa$_{BASE}$

Hyperparameter	Search Space	Selected Value for SNLI	Selected Value for MNLI
number of epochs	$\{1, 2, 3\}$	3	1
max input length	$\{50, 60, \ldots, 200\}$	140	150
learning rate	$[5 \times 10^{-5}, 1 \times 10^{-4}]$	3.2×10^{-5}	6.1×10^{-5}
weight decay	$[0, 0.01]$	8.8×10^{-3}	3.3×10^{-3}
adam epsilon	$[1 \times 10^{-8}, 1 \times 10^{-7}]$	2×10^{-8}	1×10^{-7}
max grad norm	$[0.9, 1.0]$	0.9	0.93

Table D.5: *Best hyperparameter assignments for RoBERTa$_{BASE}$.*

RoBERTa$_{LARGE}$

Hyperparameter	Search Space	Selected Value for SNLI	Selected Value for MNLI
number of epochs	$\{1, 2, 3\}$	1	2
max input length	$\{50, 60, \ldots, 200\}$	140	150
learning rate	$[5 \times 10^{-5}, 1 \times 10^{-4}]$	5×10^{-5}	5×10^{-5}
weight decay	$[0, 0.01]$	8.8×10^{-3}	8.8×10^{-3}
adam epsilon	$[1 \times 10^{-8}, 1 \times 10^{-7}]$	4×10^{-8}	5×10^{-7}
max grad norm	$[0.9, 1.0]$	0.93	0.9

Table D.6: *Best hyperparameter assignments for RoBERTa$_{LARGE}$.*

Bibliography

Ba, L. J., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *CoRR*, abs/1607.06450. 20

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*. 16

Bar-Haim, R., Dagan, I., and Szpektor, I. (2014). Benchmarking applied semantic inference: The PASCAL recognising textual entailment challenges. In *Language, Culture, Computation. Computing - Theory and Technology - Essays Dedicated to Yaacov Choueka on the Occasion of His 75th Birthday, Part I*, pages 409–424. 1

Bender, E. M. and Koller, A. (2020). Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198. Association for Computational Linguistics. 4

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828. 9

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155. 10

Bentivogli, L., Clark, P., Dagan, I., and Giampiccolo, D. (2009). The sixth PASCAL recognizing textual entailment challenge. *Theory and Applications of Categories*. 1

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305. 60

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015a). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics. 1, 3, 57, 65

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2020). SNLI site. https://nlp.stanford.edu/projects/snli. xv, 2, 49

Bowman, S. R., Manning, C. D., and Potts, C. (2015b). Tree-structured composition in neural networks without tree-structured architectures. In *Proceedings of the 2015th International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches*, volume 1583 of *COCO'15*, pages 37–42. 54

Casella, G. and Berger, R. (2002). *Statistical Inference*. Duxbury Resource Center, Pacific Grove. 30, 31

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394. 10

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*. 13

Cochram, W. G. (1950). The comparison of percentages in matched samples. *Biometrika*, 37(3/4):256–266. 63

Conneau, A., Rinott, R., Lample, G., Williams, A., Bowman, S. R., Schwenk, H., and Stoyanov, V. (2018). XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485. Association for Computational Linguistics. 48

da Silva, F. S. C., Finger, M., and de Melo, A. C. V. (2006). *Lógica para computação*. Cengage Learning, São Paulo, SP. 37

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988. Association for Computational Linguistics. 26

Dasgupta, I., Guo, D., Stuhlmüller, A., Gershman, S. J., and Goodman, N. D. (2018). Evaluating compositionality in sentence embeddings. *CoRR*, abs/1802.04302. 4, 5, 70

de Marneffe, M., Rafferty, A. N., and Manning, C. D. (2008). Finding contradictions in text. In *Proceedings of ACL-08:HLT*, pages 1039–1047. Association for Computational Linguistics. 1

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). 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. xv, 2, 23, 24, 25, 39, 42, 44, 49, 51, 57, 58, 66, 71

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., and Smith, N. (2020). Fine-tuning pretrained language models: weight initializations, data orders, and early stopping. *CoRR*, abs/2002.06305. 63

Eight, F. (2016). First GOP Debate Twitter Sentiment. https://www.kaggle.com/crowdflower/first-gop-debate-twitter-sentiment#Sentiment.csv. 8

Eisenstein, J. (2019). *Introduction to Natural Language Processing*. MIT Press, Cambridge, MA. 7

Evans, R., Saxton, D., Amos, D., Kohli, P., and Grefenstette, E. (2018). Can neural networks understand logical entailment? In *6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*. 4, 42, 47, 54

Explosion (2020). spaCy: Industrial-strength NLP. https://github.com/explosion/spaCy. 64

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA. 64

Fisher, N. I. and Hall, P. (1990). On bootstrap hypothesis testing. *Australian Journal of Statistics*, 32(2):177–190. 33, 61

Flesch, R. (1948). A new readability yardstick. *Journal of Applied Psychology*, 32(3):221–233. 55

Geiger, A., Cases, I., Karttunen, L., and Potts, C. (2018). Stress-testing neural models of natural language inference with multiply-quantified sentences. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353. Association for Computational Linguistics. 55

Geiger, A., Cases, I., Karttunen, L., and Potts, C. (2019). Posing fair generalization tasks for natural language inference. 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)*. Association for Computational Linguistics. 73

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. (2007). The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL@ACL 2007 Workshop on Textual Entailment and Paraphrasing*, pages 1–9. 3

Glockner, M., Shwartz, V., and Goldberg, Y. (2018). Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. xv, 2, 3, 4, 42, 57, 72

Goldberg, Y. (2016). *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, San Rafael, California. 7, 12, 13

Goldberg, Y. (2019). Assessing BERT's syntactic abilities. *CoRR*, abs/1901.05287. 47

Goodfellow, I., Bengio, Y., and Courville, A. (2017). *Deep Learning*. MIT Press. 23

Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S. R., and Smith, N. A. (2018). 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. Association for Computational Linguistics. 3, 39, 54

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780. 14

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. 22, 23, 42, 57

Joshi, P., Aditya, S., Sathe, A., and Choudhury, M. (2020). TaxiNLI: Taking a ride up the NLU hill. *CoRR*, abs/2009.14505. 75, 76

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume abs/1607.01759, pages 427–431. Association for Computational Linguistics. 12, 44

Kalouli, A.-L., Crouch, R., and de Paiva, V. (2020). Hy-NLI: a hybrid system for natural language inference. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5235–5249, Barcelona, Spain (Online). International Committee on Computational Linguistics. 77

Kalouli, A.-L., Crouch, R., de Paiva, V., and Real, L. (2018). Graph knowledge representations for sick. EasyChair Preprint no. 217. 2

Kalouli, A.-L., Real, L., and de Paiva, V. (2017). Textual inference: getting logic from humans. In *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*. 7

Khot, T., Sabharwal, A., and Clark, P. (2018). Scitail: A textual entailment dataset from science question answering. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 5189–5197. 1

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*. 44

Konietschke, F. and Pauly, M. (2014). Bootstrapping and permuting paired t-test type statistics. *Statistics and Computing*, 24:283–296. 33, 61, 62

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020). Albert: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*. 26, 58, 66

Lehmann, E. L. (1993). The Fisher, Neyman-Pearson theories of testing hypotheses: One theory or two? *Journal of the American Statistical Association*, 88(424):1242–1249. 27

Levy, R. and Manning, C. (2003). Is it harder to parse Chinese, or the Chinese treebank? In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 439–446. Association for Computational Linguistics. 43

Lin, J. (2020). BERT diagrams. https://twitter.com/lintool/status/1285599163024125959. xv, 25

Liu, N. F., Schwartz, R., and Smith, N. A. (2019a). Inoculation by fine-tuning: A method for analyzing challenge datasets. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics. 70, 73

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019b). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692. 25, 26, 49, 57, 58, 66, 68

Luong, M., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics. 16, 17

MacCartney, B. and Manning, C. D. (2009). An extended model of natural logic. In *Proceedings of the Eight International Conference on Computational Semantics*, pages 140–156, Tilburg, The Netherlands. Association for Computational Linguistics. 2

MacKinnon, J. G. (2009). *Bootstrap Hypothesis Testing*, chapter 6, pages 183–213. John Wiley Sons, Ltd. 34

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA. 7

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., and Zamparelli, R. (2014). A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223. European Language Resources Association (ELRA). 1

Martin-Löf, P. (1996). On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1):11–60. 37

McCoy, T., Pavlick, E., and Linzen, T. (2019). Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics. 4, 70

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157. 63

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Workshop Track Proceedings*. 12

Naik, A., Ravichander, A., Sadeh, N., Rose, C., and Neubig, G. (2018). Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics. 4, 70

Nie, Y., Wang, Y., and Bansal, M. (2018). Analyzing compositionality-sensitivity of NLI models. *CoRR*, abs/1811.07033. 3, 4, 70

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch. 43

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *In EMNLP*, pages 1532–1543. Association for Computational Linguistics. 12, 44

Perrault, R., Shoham, Y., Brynjolfsson, E., Clark, J., Etchemendy, J., Grosz, B., Lyons, T., Manyika, J., Mishra, S., and Niebles, J. C. (2019). *The AI Index 2019 Annual Report*. Stanford University, Stanford, CA. 3

Piotrowski, B., Urban, J., Brown, C. E., and Kaliszyk, C. (2019). Can neural networks learn symbolic rewriting? *CoRR*, abs/1911.04873. 4

Poliak, A., Naradowsky, J., Haldar, A., Rudinger, R., and Durme, B. V. (2018). Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191. Association for Computational Linguistics. 3

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. In *OpenAI Blog*. 2, 22, 23, 24

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. In *OpenAI Blog*. xv, 19, 57, 71

Rawal, A. and Miikkulainen, R. (2018). From nodes to networks: Evolving recurrent neural networks. *CoRR*, abs/1803.04439. 14

Richardson, K., Hu, H., Moss, L. S., and Sabharwal, A. (2020). Probing natural language inference models through semantic fragments. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*. AAAI Press. xvi, 54, 55, 56, 57, 70, 72

Salvatore, F. (2019a). ContraBERT. https://github.com/felipessalvatore/ContraBERT. 40, 43

Salvatore, F. (2019b). InferenceLanguageModels. https://github.com/felipessalvatore/InferenceLanguageModels. 51

Salvatore, F. (2020). Looking-for-Equivalences. https://github.com/felipessalvatore/looking-for-equivalences. 66

Salvatore, F., Finger, M., and Hirata Jr, R. (2019a). A logical-based corpus for cross-lingual evaluation. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 22–30. Association for Computational Linguistics. 6, 39, 55

Salvatore, F., Preto, S., Finger, M., and Hirata Jr, R. (2019b). Using neural models to perform inference. In *Proceedings of the 2019 International Workshop on Neural-Symbolic Learning and Reasoning (Nesy 2019)*, volume 1, pages 85–87. 6, 48

Saxton, D., Grefenstette, E., Hill, F., and Kohli, P. (2019). Analysing mathematical reasoning abilities of neural models. In *7th International Conference on Learning Representations, ICLR 2019*. 49

Schrimpf, M., Merity, S., Bradbury, J., and Socher, R. (2018). A flexible approach to automated RNN architecture generation. In *6th International Conference on Learning Representations, ICLR 2018, Workshop Track Proceedings*. 14

Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences*, 3:417–424. 4

Shieber, S. M. (1993). The problem of logical-form equivalence. *Computational Linguistics*, 19(1):179–190. 59

Shoenfield, J. R. (1967). *Mathematical Logic*. Addison-Wesley, Boston, MA. 37, 38, 58

Sustskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112. 15

The Fracas Consortium, Cooper, R., Crouch, D., Eijck, J. V., Fox, C., Genabith, J. V., Jaspars, J., Kamp, H., Milward, D., Pinkal, M., Poesio, M., Pulman, S., Briscoe, T., Maier, H., and Konrad, K. (1996). Using the framework. 1, 49, 54

Tran, K. M., Bisazza, A., and Monz, C. (2018). The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4731–4736. Association for Computational Linguistics. 42, 47, 54

Troelstra, A. S. and Schwichtenberg, H. (1996). *Basic Proof Theory*. Cambridge University Press, New York, NY. 37

Turing, A. (1950). Computing machinery and inteligence. *Mind*, pages 433–460. 4

van Wijk, M. (2006). *Logical connectives in natural language: a cultural-evolutionary approach*. PhD Thesis, Universiteit Leiden. 36

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010. Curran Associates Inc. xv, 2, 17, 18, 19, 20, 22, 42

Vig, J. (2019). A multiscale visualization of attention in the transformer model. *CoRR*, abs/1906.05714. xv, 19

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019). SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *CoRR*, abs/1905.00537. 2, 3

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2018). 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. Association for Computational Linguistics. 2, 3, 55

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2020). GLUE benchmark. https://gluebenchmark.com/leaderboard. xv, 2, 3, 49

Wasserman, L. (2010). *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated, New York, NY. 29, 32, 62

Welleck, S., Weston, J., Szlam, A., and Cho, K. (2019). Dialogue natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3731–3741. Association for Computational Linguistics. 39

Weston, J., Bordes, A., Chopra, S., and Mikolov, T. (2016). Towards AI-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016,Conference Track Proceedings*. 54

Williams, A., Nangia, N., and Bowman, S. R. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics. 1, 3, 57, 65

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771. 43, 66

Yanaka, H., Mineshima, K., Bekki, D., Inui, K., Sekine, S., Abzianidze, L., and Bos, J. (2019). Can neural networks understand monotonicity reasoning? In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, volume abs/1906.06448, pages 31–40. Association for Computational Linguistics. 4, 5, 57, 71, 72

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc. 26, 58, 66

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3320–3328. MIT Press. 22

Zhu, X., Li, T., and de Melo, G. (2018). Exploring semantic properties of sentence embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 632–637, Melbourne, Australia. Association for Computational Linguistics. 71