UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MASTER

Design and Development of a Causal Inference Machine Learning System based on Textual Data

Daniel Vera Nieto
2021

TRABAJO DE FIN DE MASTER

Título:	Diseño y desarrollo de un sistema de Inferencia Causal de Machine Learning basado en texto
Título (inglés):	Design and Development of a Causal Inference Machine Learning System based on Textual Data
Autor:	Daniel Vera Nieto
Tutor:	Carlos Ángel Iglesias Fernández
Ponente:	PONENTE
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE MASTER

Design and Development of a Causal Inference Machine Learning System based on Textual Data

Junio 2021

Resumen

La comunidad del aprendizaje automático es cada vez más consciente de las limitaciones del paradigma actual de la inteligencia artificial, fuertemente basado en el reconocimiento de patrones, donde los enfoques de aprendizaje profundo han dominado la mayoría de los problemas de aprendizaje automático en la última década. Así, los investigadores están cada vez más interesados en aprender más sobre causas y efectos e incluir este conocimiento causal en los modelos que apoyan la toma de decisiones en la actualidad. El Procesamiento del Lenguaje Natural (PLN) es uno de los campos del Aprendizaje Automático que es cada vez más consciente de la importancia de la causalidad para la interpretabilidad de los modelos y para que éstos sean escalables y robustos. Sin embargo, la infancia de esta tendencia en la intersección de la causalidad y el PLN produce una falta de herramientas para realizar estudios de inferencia causal que incorporen tecnologías innovadoras que faciliten el procesamiento y análisis de grandes cantidades de datos en forma de texto.

El objetivo principal de este proyecto es la integración de la tecnología como herramienta habilitadora para encontrar relaciones causales en presencia de texto. Esto aportará un valor importante a la hora de acelerar la extracción de conocimiento causal en dominios en los que predomina el texto. Para ello, implementamos modelos del estado del arte actual para inferir efectos causales en presencia de datos textuales. Además, desarrollamos diferentes módulos de extracción de características lingüísticas para fomentar y facilitar el estudio del efecto de las propiedades lingüísticas. De hecho el trabajo realizado para el módulo de análisis de emociones culminó con el diseño y desarrollo de un modelo basado en *transformers* que presentamos a la competición EmoEvalEs, enmarcada en la Conferencia IberLef 2021. Hemos conseguido la primera posición en dicha competición, lo que nos ha llevado a publicar el paper ***GSI-UPM in IberLEF2021: Emotion Analysis of Spanish Tweets by Fine-tuning the XLM-RoBERTa Language Model*** en las actas del congreso. Por último, hemos evaluado nuestro sistema en diferentes casos de uso para estudiar el efecto del sentimiento, la emoción u otras propiedades lingüísticas en redes sociales y plataformas en las que predominan los datos de texto generados de forma colaborativa.

Palabras clave: Inferencia Causal, Aprendizaje Automático, Procesamiento de Lenguaje Natural, Transformers, Análsis de Sentimientos, Análisis de Emociones

Abstract

The machine learning community is increasingly aware of the limitations of the current paradigm of artificial intelligence, strongly based on pattern recognition, where deep learning approaches have mastered most machine learning tasks in the last decade. Thus, researchers are more and more interested in shifting to learn more about cause and effect and include this causal knowledge in the models that support decision-making nowadays. Natural Language Processing (NLP) is one of the Machine Learning fields that is becoming increasingly aware of the importance of causality for model interpretability and scalable and robust models. However, the infancy of this trend in the intersection of causality and NLP produces a lack of tools to carry out causal inference studies which incorporate innovative technologies that facilitate the processing and analysis of great amounts of textual data.

The main objective of this project is the integration of technology as an enabling tool to find causal relations in the presence of text. This will provide an important value in speeding up causal knowledge extraction in domains where the text is predominant. To this purpose, we implemented state-of-the-art models to infer causal effects in the presence of textual data. In addition, we developed different linguistic feature extraction modules to foster and facilitate the study of the effect of linguistic properties. These modules have been proven to be useful for sentiment and emotion analysis. In fact, the work done for the emotion analysis module culminated in the design and development of a transformer-based model that we submitted to the EmoEvalEs competition framed in the IberLef 2021 Conference. We have achieved the first position in the EmoEvalEs competition, which led us to publish the paper ***GSI-UPM at IberLEF2021: Emotion Analysis of Spanish Tweets by Fine-tuning the XLM-RoBERTa Language Model*** in the proceedings of the conference. Finally, we have evaluated our system in different use cases to study the effect of sentiment, emotion, or other linguistic properties in social media and platforms where collaborative generated text data is predominant.

Keywords: Causal Inference, Machine Learning, Natural Language Processing, Transformers, Sentiment Analysis, Emotion Analysis

Agradecimientos

Este trabajo no habría sido posible sin todas las personas que han hecho de mi paso por la universidad la experiencia vital tan positiva que ha sido. Desde el profesorado (tanto de las asignaturas que he disfrutado como las que he sufrido) a todo el personal que hace que la escuela funcione cada día a pesar de los tiempos extraños que nos ha tocado vivir últimamente. También agradecer a mis compañeros y compañeras de clase con los que he coincidido, en especial a mis amigos que me han acompañado durante todos estos años.

Mi gratitud al personal que forma el Grupo de Sistemas Inteligentes (GSI), por abrirme las puertas y apoyar mi trabajo. En especial, a mi tutor, Carlos Ángel Iglesias Fernández, cuya orientación y apoyo me ha permitido acabar este trabajo de la mejor forma posible. También me gustaría agradecer a la cátedra Cabify la oportunidad de desarrollar este proyecto y a Eduardo Matallanas las charlas para rematar el trabajo.

La entrega de este proyecto marca el fin una etapa de mi vida y el comienzo de otra nueva. Y no habría podido llegar a este punto si no fuera por el apoyo de mi familia, muy especialmente el de mi madre que, siempre ha tenido palabras de ánimo para continuar cuando la vida se hacía cuesta arriba.

Contents

Resumen **VII**

Abstract **IX**

Agradecimientos **XI**

Contents **XIII**

List of Figures **XVII**

1 Introduction **1**

1.1 Context 2

1.2 Project goals 3

1.3 Structure of this document 4

2 Theory **5**

2.1 A Brief Introduction to Causal Inference 6

2.1.1 Correlation does not imply causation 6

2.1.2 Then, what does imply causation? 8

2.1.3 Causation in observational studies? 9

2.2 Transformers 10

2.2.1 From Bag of Words to Transformers 10

2.2.2 Transformers 13

2.2.2.1 Transformer architecture 14

2.2.2.2 Transformers and transfer learning 15

3 State of Art **17**

3.1 Causal inference and machine learning 18

3.2 Causal inference: Text-based use cases 19

3.3 Causal Inference in social media 21

3.4 TextCause algorithm 21

3.4.1 Adapting Text Embeddings for Causal Inference - Veitch et al. . . . 21

3.4.2 Causal Effects of Linguistic Properties - Pryzant et al. 22

4 Enabling Technologies **25**

4.1 Machine Learning Technologies 26
4.1.1 Scikit-learn 26
4.1.2 Numpy 26
4.1.3 Pytorch 26
4.1.4 HuggingFace Libraries 27
4.1.5 DistilBERT 27
4.2 Data technologies 27
4.2.1 Pandas 27
4.2.2 Matplotlib 27
4.2.3 Twint 28
4.3 Sentiment analysis 28
4.3.1 Senpy 28
4.3.2 LIWC 28
4.3.3 Lexicons 29

5 Architecture **31**

5.1 Framework 32
5.2 TextCause module 34
5.2.1 Reproducing paper 34
5.2.1.1 Amazon reviews experiment 34
5.2.1.2 Consumer complaints experiment 37
5.2.2 TextCause algorithm modifications 40
5.2.3 Simulations for causal inference evaluation 41
5.2.3.1 Datasets 41
5.2.3.2 Evaluation 42
5.2.4 Conclusions 47
5.3 Model distillation: Is distilBETO really worth it? 48
5.4 Transfer learning and fine-tuning 49
5.4.1 Sentiment analysis module 50
5.4.1.1 Datasets 50
5.4.1.2 Evaluation 51
5.4.2 Emotion analysis module 53
5.4.2.1 Dataset 53
5.4.2.2 Fine-tuning XLM-T 54

5.4.2.3 XLM-T evaluation . . . 56
5.4.2.4 Conclusions . . . 58
5.5 LIWC module . . . 59

6 Case studies **61**
6.1 Twitter interactions of ride-hailing companies . . . 62
6.1.1 Dataset . . . 62
6.1.2 Processing . . . 64
6.1.3 Framework . . . 64
6.1.4 Results . . . 65
6.1.5 Conclusions . . . 68
6.2 Case Study: Airbnb . . . 69
6.2.1 Dataset . . . 69
6.2.2 Framework . . . 70
6.2.3 Results . . . 71
6.2.4 Conclusions . . . 73

7 Conclusions **75**
7.1 Conclusion . . . 76
7.2 Future work . . . 77

A Project impact **79**
A.1 Social impact . . . 79
A.2 Economic impact . . . 80
A.3 Environmental impact . . . 80
A.4 Ethical implications . . . 80

B Project budget **81**
B.1 Project structure . . . 81
B.2 Costs evaluation . . . 83
B.2.1 Material resources . . . 83
B.2.1.1 Software . . . 83
B.2.1.2 Hardware . . . 83
B.2.2 Human resources . . . 84
B.2.3 Taxes . . . 84
B.3 Conclusion . . . 84

Bibliography **85**

List of Figures

2.1 Yearly per capita cheese consumption in the U.S. correlates with the yearly number of people who died tangled in their bedsheets . . . 7
2.2 Randomized Controlled Trials eliminate the association between the confounder and treatment . . . 9
2.3 Backdoor adjustment . . . 10
2.4 Unfolded RNN . . . 11
2.5 Machine Traslation with a Encoder-Decoder architecture . . . 12
2.6 Comparison of the number of parameters of each Transformer model until Turing-NLG . . . 13
2.7 Transformer diagram of two stacked encoder-decoders . . . 14

3.1 Text as a confounder . . . 22

5.1 System architecture . . . 33
5.2 Standard deviation for each of the 8 experiments over the 100 random seeds 37
5.3 Average difference of TextCause results with different number of seeds . . . 38
5.4 Reviews by word number . . . 41
5.5 Spanish reviews word cloud . . . 42
5.6 ATE estimates of TextCause algorithm for each language . . . 47
5.7 Fine-tuning process . . . 49
5.8 Where does the model focus? Transformer attension on a tweet by @IbaiLlanos 50
5.9 EmoEvalEs dataset . . . 54
5.10 Where does the model focus? Transformer attention on a tweet by @IbaiLlanos. Green highlighted words contributes positively towards the predicted class . . . 55
5.11 Confusion matrix on test set produced by XLM-T Multi-label classifier . . . 57
5.12 Senpy architecture . . . 59

6.1 Spanish dataset . . . 63
6.2 English dataset . . . 63
6.3 Processing steps . . . 64

6.4 Causal model of Twitter use case 65
6.5 Results in the English dataset for the negative sentiment proxy treatment . 66
6.6 Results in the English dataset for the angry emotion proxy treatment . . . 67
6.7 Results in the Spanish dataset for the negative sentiment proxy treatment . 67
6.8 Results in the Spanish dataset for the angry emotion proxy treatment . . . 68
6.9 Airbnb listings in Madrid. Orange: Entire homes. Green: Rooms. 69
6.10 Causal models for the Airbnb study 71
6.11 Results on Spanish dataset: reviews effect on score 71
6.12 Results on English dataset: reviews effect on score 71
6.13 Effect of LIWC fatures on rating score 72
6.14 Effect of LIWC features on reviews per month 72

CHAPTER 1

Introduction

This chapter is going to introduce the context of the project, including a brief overview of all the different parts that will be discussed in the project. It will also break down a series of objectives to be carried out during the realization of the project. Moreover, it will introduce the structure of the document with an overview of each chapter.

1.1 Context

"If we compare what machine learning can do to what animals accomplish, we observe that the former is rather limited at some crucial feats where natural intelligence excels" [63]. The machine learning community is increasingly aware of the limitations of the current paradigm of artificial intelligence. Thus, it is more and more interested in shifting to learn more about cause and effect instead of pattern recognition, where deep learning approaches have mastered most machine learning tasks in the last decade.

Causal inference is the process of determining the independent, actual effect of a particular phenomenon that is a component of a larger system. Causality, with its focus on representing structural knowledge about the data generating process that allows interventions and changes, can contribute towards understanding and resolving some limitations of current machine learning methods [63].

When it comes to learning causality with data, we need to be aware of the differences between statistical associations and causations. For example, when the car's braking system tell-tale lights up, the driver may observe it is harder to hit the brakes and slow down the car. Accordingly, she would try to disconnect the tell-tale to be able to easily use the brakes again, but, surprisingly, doing this does not work since the tell-tale was not causing the brakes' malfunction. In this case, usually, a low level of brake fluid is the common cause of both the tell-tale warning and the car's malfunction, and we say that it is a confounder of the causality of the tell-tale warning on the car's malfunction.

These new ideas are slowly landing and becoming more relevant in all machine learning fields, such as in Natural Language Processing (NLP). However, the infancy of this trend in the intersection of causality and NLP produces a lack of tools to reproduce current works and experiment with new ideas.

Additionally, the digital transformation and the integration in our lives of internet-based services in recent years have caused an explosion of data that only can be processed automatically, creating new technical challenges in data-intensive applications. For this reason, it is necessary to incorporate innovative technologies that facilitate the processing and analysis of such data dimensions.

Therefore, the main objective of this project is the integration of technology as an enabling tool to find causal relations in presence of text in a reproducible and easy way. This will provide an important value in speeding up causal knowledge extraction in domains where the text is predominant. For this purpose, we implemented the state-of-the-art to infer causal effects in presence of text, and different linguistic feature extraction modules. One of them, the emotion analysis module was used to participate in the EmoEvalEs competition framed in the IberLef 2021 Conference. With this model, we have achieved the first

position in the EmoEvalEs competition, which led us to publish the paper ***GSI-UPM at IberLEF2021: Emotion Analysis of Spanish Tweets by Fine-tuning the XLM-RoBERTa Language Model*** in the proceedings of the competition.

1.2 Project goals

The main objective of this project is to develop a tool capable of discovering causal relations in settings where text is predominant. For this purpose, we have proven the possibility of using current state-of-the-art models in different language settings. We have made these models modular to gain flexibility in the inputs. Finally, we have built a system to extract text characteristics using recent NLP model architectures the transformers. We have tested our system on different use cases: First, messages are taken from the social network Twitter directed towards ride-hailing companies such as Uber and Cabify, selecting both Spanish and English-speaking users. The results obtained from these objectives will help us develop a system that allows us to obtain the characteristics of the tweets that causally influence the studied output, in this case, the time these companies take to answer the social network users. The second use case is to estimate the effect of sentiment, emotion, and other linguistic properties on the rating score and reviews per month of the Airbnb platform.

To achieve these objectives explained in the previous section, the following tasks have been carried out during the project:

1. T1: Study the state of the art about Natural Language Processing and machine learning technologies. Specifically, its intersection with causal inference.

2. T2: Reproduction of current state-of-the-art works on causal inference for Natural Language Processing.

3. T3: Development of a machine learning system to estimate causal effects following the next steps:

   (a) Development of a simulated dataset to test the capabilities of the SOTA model in different language settings.

   (b) Extend the SOTA model to be able to use it with models in different languages.

   (c) Create and implement different feature extracting modules to estimate different linguistic features in the text. More specifically, modules for sentiment and emotion analysis.

   (d) Analysis of the results of the different machine learning modules.

4. T4: Evaluation of the causal inference system on different case studies:

(a) Effect of sentiment and emotion on reply time in the social network Twitter.

(b) Effect of different linguistic attributes on the rating of a product (apartments in Airbnb)

1.3 Structure of this document

The remaining of this document is structured as follows:

- Introduction ***(Chapter 1).*** This chapter introduces the reader to the context where the project is developed, giving a quick overview of the intersection of causal inference and natural language processing, and how we pretend to develop a tool to estimate causal effects in text easier. In addition, the goals of the project are presented along with the specific tasks we will perform.
- Theory introduction ***(Chapter 2).*** This chapter describes briefly the main conceptual foundations behind this work: what is causal inference and what are transformers.
- State of Art ***(Chapter 3).*** This chapter offers the reader a review of all previous knowledge and works that have inspired ours. It explains the current approaches to apply causal inference in the natural language processing domain.
- Enabling Technologies ***(Chapter 4).*** This chapter briefly reviews the main technologies that have made possible this project.
- Architecture ***(Chapter 5).*** This chapter describes the architecture of the project, from the *TextCause module*, which is the core of the system, to the different modules developed to extract accurate linguistic attributes of the text.
- Case studies ***(Chapter 6).*** This chapter describes the system used in two particular use cases, where its validity is discussed and insights are extracted as a Proof-of-Concept.
- Conclusions and future work ***(Chapter 7).*** This chapter details the achieved goals and outcomes. In addition, future work lines are described.
- Project impact ***(Appendix A).*** This appendix shows the social, economic, and environmental impact as well as ethical implications
- Project budget ***(Appendix B).*** This appendix describes the necessary project budget regarding the material and human resources needed, as well as the taxes involved.

CHAPTER 2

Theory

Tens of thousands of years ago, humans began to realize that certain things cause other things and that altering the former can alter the latter. No other species has understood this; certainly not to the same extent that we have. This discovery gave rise to organized societies, towns, and cities, and eventually, to the scientifically and technologically rooted civilization we enjoy today. And all for asking a simple question: Why?

- Judea Pearl, The Book of Why

This chapter offers a theoretical introduction to the science and technology this work is built. First, we will provide a brief introduction to causal inference to let the reader grasp the basic ideas behind this science. Secondly, we will describe the main technology used in this project, the Transformers. Finally, we will discuss the two main works that have lead to the development of the CausalText algorithm and are the starting point for this project.

2.1 A Brief Introduction to Causal Inference

The science of Causal inference addresses questions regarding the cause-effect relations of different events of everyday life. Examples of this kind of questions are:

- the effectiveness of a treatment to prevent or treat a disease
- what is the cost of obesity to the health system
- what is the effect of social media on mental health
- the increase in sales was caused by the new tax law or by our marketing campaign?
- does this HR employee have a gender/race/religion bias when hiring new staff according to his history of hiring?

All these questions try to infer the effects of a treatment/policy/intervention on a determined result and are addressed via different causal inference methods. Thus, causal inference is essential for rigorous decision-making. In this project, we will analyze how to answer this kind of question when some of the variables we are using are linguistic properties contained in the text, and we implement a system to ease the solution to this problem under certain conditions. This section will discuss the main concepts of the science of causal inference to facilitate the understanding of the objectives of the project. It has been written based on the course "Introduction to Causal Inference" by Brady Neal [44].

2.1.1 Correlation does not imply causation

There is a good number of associations in the real world that we easily identify as spurious due to the evident lack of causal relationships among them. For example, it seems that the yearly number of people who died tangled in their bedsheets has a high degree of correlation with the yearly per capita cheese consumption in the U.S. (see Figure 2.1). Does this mean that cheese produces an uncommon side effect on certain parts of the population that makes them uncontrollably move during their sleep until they die tangled? Do the cheese companies advertise more their products when a person is found dead tangled in her sheets? Or is there any other explanation?

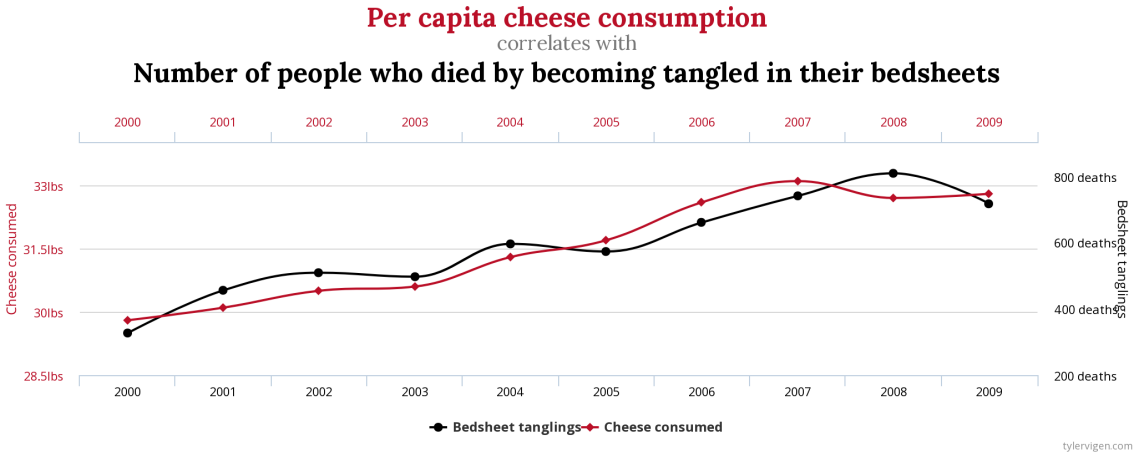

Figure 2.1: Yearly per capita cheese consumption in the U.S. correlates with the yearly number of people who died tangled in their bedsheets[1]

Although in this case, it is obvious for most of us that there is no causal relation between the variables, when working in studies using real-world observational data, it can become harder to distinguish the spurious correlations from the actual causal associations. Moreover, most of the associations are to some extent causal associations. This means that for a given amount of association, a part of it might be a causal association and the other part not.

For example, when the car's braking system tell-tale lights up, the driver may observe it is harder to hit the brakes and slow down the car. Accordingly, she would think that this tell-tale caused the malfunctioning of the breaks and try to disconnect the tell-tale to be able to easily use the brakes again. But, surprisingly, doing this does not work since the tell-tale was not causing the brakes' malfunction. In this case, usually, a low level of brake fluid is the common cause of both the tell-tale warning and the car's malfunction, and we say that it is a confounder of the causality of the tell-tale warning on the car's malfunction. We will call this kind of association confounding association since the association is facilitated by a confounder. Then the total association observed can be made up of both confounding association and causal association.

Additionally, different cognitive biases play a role in the "correlation vs causation" problem, making us predisposed to confound a statistical association with a causal one. For example, the availability heuristic (we find a cause in something available in our minds) and the motivated reasoning (I do not like spending time with my inlaws, then that time I spent with them made me have a headache).

[1]Source: `https://www.tylervigen.com/spurious-correlations`

2.1.2 Then, what does imply causation?

Before answering that question, we must define some concepts: we will use T to denote the random variable for **treatment**, Y to denote the random variable for the **outcome** of interest and C to denote **covariates**. The potential outcome $Y(t)$ denotes what your outcome would be if you were to take treatment t. A potential outcome $Y(t)$ is distinct from the observed outcome Y in that not all potential outcomes are observed. Rather all potential outcomes can potentially be observed. The one that is actually observed depends on the value that the treatment t takes on.

We define the individual causal effect or *Individual Treatment Effect* for a given individual as:

$$ITE = Y_i(do(T = 1)) - Y_i(do(T = 0))$$

where the i subscript refers to that specific individual and the treatment T is considered binary (1 or 0). The *do* notation refers to actually intervening in the treatment to observe the outcome Y. Then, the ITE is the difference between the outcome when the treatment is given ($do(T = 1)$) and the outcome when the treatment is not given ($do(T = 0)$).

Unfortunately, this definition of causal effect runs up against the so-called "Fundamental Problem of Causal Inference": It is impossible to observe all potential outcomes for a given individual. That is, if you observe the outcome given the treatment ($Y_i(do(T = 1))$), you cannot observe what would have happened if you don't give the treatment (($Y_i(do(T = 0))$)). The scenario you do not observe is known as counterfactual. It is a fundamental problem because if you can not observe both $Y_i(do(T = 1))$ and $Y_i(do(T = 0))$, you can not observe the causal effect.

The solution to this problem is to use the Average Treatment Effect (ATE), the average over the ITEs:

$$ATE = E[Y_i(do(T = 1)) - Y_i(do(T = 0))] = E[Y(1) - Y(0)]$$

However, to actually compute the ATE we need a statistical estimand and the first option that comes to mind is the associational difference: $E[Y|T = 1] - E[Y|T = 0]$. Unfortunately, this associational quantity includes also confounding associations and thus, is not equal to the causal quantity of interest. We need to make further assumptions so that the ATE is simply the associational difference: ignorability (assume that the treatment was randomly given) and exchangeability (the treatment groups are exchangeable in the sense that if they were swapped, the new treatment group would observe the same outcomes as the old treatment group).

Sadly, these assumptions are completely unrealistic because there is likely to be confounding in most observational data. A solution to this problem is the use of Randomized

control trials (RCTs), which force the treatment to not be caused by anything (Figure 2.2). When there are no confoundings, the ATE is equal to the difference in conditional expectations.

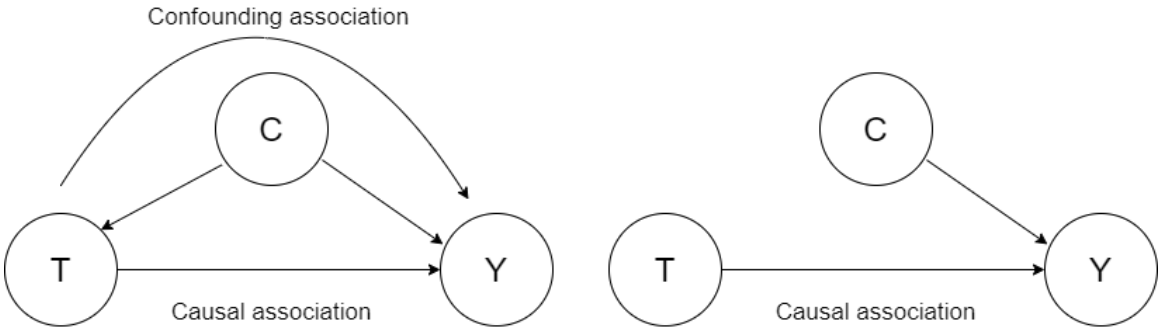

Figure 2.2: Randomized Controlled Trials eliminate the association between the confounder and treatment

2.1.3 Causation in observational studies?

Although Randomized Controlled Trials allow us to compute the causal effect, we can not always randomize treatment due to ethical reasons (it would be unethical to make a group smoke a pack of cigarettes a day to study its effect on lung cancer), infeasibility due to the cost and resources needed to carry out the experiment, or because it is just less convenient.

To be able to use observational data, we need to fulfill the Exchangeability assumption. To achieve that, we can control the relevant variables by conditioning the estimand on them. In other words, we only compare groups with the same level of the confounding variable (all individuals in group A have the confounding $C = 1$ and all in group B have $C = 0$) so that the confounding association between treatment T and outcome Y is blocked. This is known as backdoor adjustment (Figure 2.3).

$$E[Y(1) - Y(0)] = E_C[E[Y|do(T = 1), C] - E[Y|do(T = 0), C]]$$

This project attempts to use recent advances in Natural Language Processing to adjust the text as a confounder and open a new range of causal inference studies based on observation textual data.

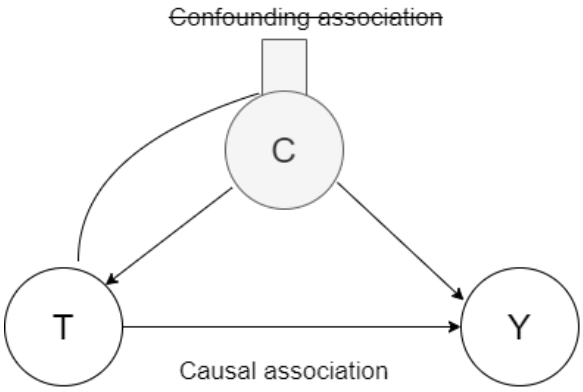

Figure 2.3: Backdoor adjustment

2.2 Transformers

The goal of this section is to provide a brief, not extensive understanding of the evolution of Natural Language Processing (NLP) technologies until the conception of the transformer architecture, and a description of the main foundations of this architecture. This section is based on several previous works [4, 5, 39, 3, 2].

2.2.1 From Bag of Words to Transformers

The first attempts of automatic language processing relied on statistical patterns of human word usage. In this line, techniques such as Bags of words [26] with their origin as early as 1954 provided a statistical representation of the words in a document. However, it was not extensively used until the 2000's when the increase in computing power allowed to processing big amounts of information.

Later, more complex information representations were created based on neural networks, machine learning models based on linear and nonlinear functions. We have witnessed during the last decade the exponential use of this type of model to tackle a wide number of problems in different knowledge fields, proving their great capacity to generate and use meaningful representations of information. This has led to the surge of new applications and solutions, fostering the interest of society in this new Artificial Intelligence spring.

In the field of NLP, this success of neural networks is in great part due to the use of word embeddings, numerical vector representations able to capture the information and associations of words. The better the word representation, the most useful is the information for neural networks to learn. Word2vec [41] is a method to efficiently create word embeddings from large text corpora. It is powered by two different algorithms, Continuous-Bag-of-Words, which goal is to predict a word given its surrounding context; and Skip-gram, which is tasked to predict the context given a word.

The word embeddings produced by this method were successfully used by Recurrent Neural Networks (RNNs), a more complex architecture with backward connections and feedback loops that was able to simulate a "memory". Since they used the information from previous outputs, they could "remember" previous words, improving the retrieval of information from the context. RNNs are combined in more complex architectures to create encoder-decoder and attention-based models

Figure 2.4 depicts the architecture of a basic RNN. The input layers transform the words into their vector representation, then, the hidden layers generate the hidden states that are propagated to the next layers and themselves through a feedback loop, allowing the hidden layers to use information from past time steps. Finally, the output layer provides a probabilistic distribution using different activation functions.

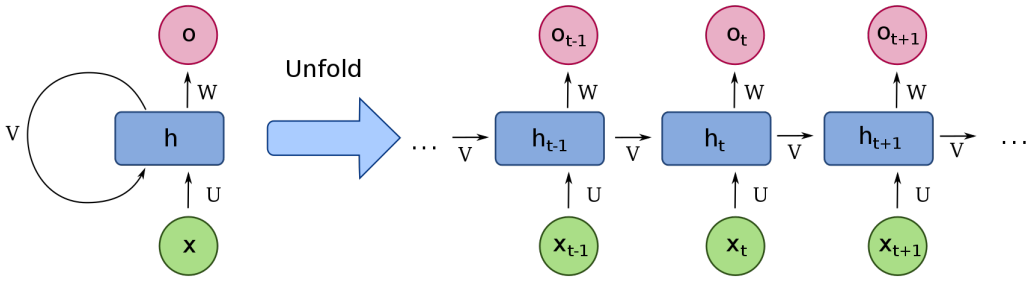

Figure 2.4: Unfolded RNN[2]

During the training process, the network weights are updated, including a new weight matrix with the information from past time steps that are used in the recurrent connections, which allows having short-term memory.

Allowing new feedback loops from the surroundings of a neuron increases the number of connections and the capacity of these architectures to make more complex operations and simulate a longer memory. This creates recurrent neuron cells, which, when stacked, are capable of learning tasks that require long-term memory over long sequences of data.

The most powerful RNNs are those based on Long Short-Term Memory (LSTM) [28] cells, which combines the inputs and various hidden states to simulate both short and long-term memory. The larger number of parameters and operations of these networks allow them to work well even given long delays between significant events and to handle signals that mix low and high-frequency components. However, this comes with an increased cost due to their sequential nature and directly proportional to the exponential growth of the number of parameters and connections, making them computationally expensive during

[2]By fdeloche - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=60109157

both training and inference.

This problem motivated the definition of a simpler yet similar performance to the LSTM cells, the Gated Recurrent Units [11] (GRUs). They have significantly fewer parameters and connections, which makes them faster and more efficient. However, their main disadvantages are their lower capacity to store long-term information and the fact that they are still sequential.

These Recurrent Neural Networks were used to build Encoder-Decoder architectures, which combine neural networks to build multi-input/multi-output models. In the NLP domain, the goal of the encoder, built with an RNN, is to transform a sequence of words into a numeric vector that preserves enough information from the sequence. On the other hand, the decoder, which is also built using RNNs, is tasked to transform the numeric vector into a sequence of words. This idea empowered many applications such as machine translation, question answering, and style transfer.

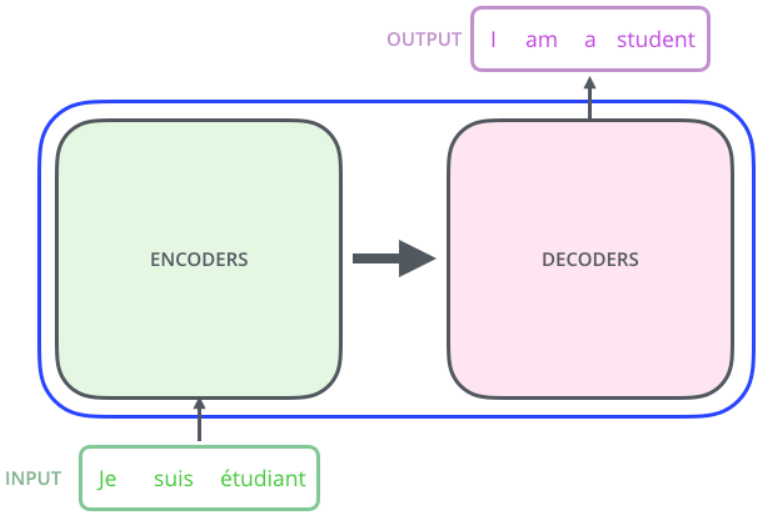

Figure 2.5: Machine Traslation with a Encoder-Decoder architecture[3]

Their main problem was the lack of context information that these models used to generate their predictions. For example, the inability to correctly translate a word that could have different meanings in a determined context. This problem was approached through finding inspiration in the human attention model: new weights are defined to be able to compute numerically the importance of each word, which adds a new complexity layer to these systems. The drawbacks of this solution are the need for ad hoc attention

[3]Alammar, J (2018). The Illustrated Transformer. Retrieved from https://jalammar.github.io/illustrated-transformer/

mechanisms for each architecture and their difficulty to take into account different factors to compute the importance of a word.

2.2.2 Transformers

This problem was solved with the transformer architecture. Transformers [68] is a self-attention-based architecture that allows computing complex representations of information without using Recurrent Neural Networks, which have made it possible to parallelize the training of large language models efficiently. The self-attention mechanism allows to associate different positions of a sequence of words and compute a representation of the information and its importance. The great advantage of this architecture is its simpler design, allowing more efficient training and inference processes, achieving extraordinary results not only in all NLP problems but in other domains such as Computer Vision or Reinforcement Learning.

Introduced in 2017 by researchers at Google and the University of Toronto, unlike RNNs they could be very efficiently parallelized allowing the training of large models. *Really* large models as Figure 2.6 shows, where the latest transformer language models are not included due to having a number of parameters out of scale: GPT-3 with 175 billion parameters (2020) and Wu Dao 2.0 with 1.75 trillion parameters(2021).

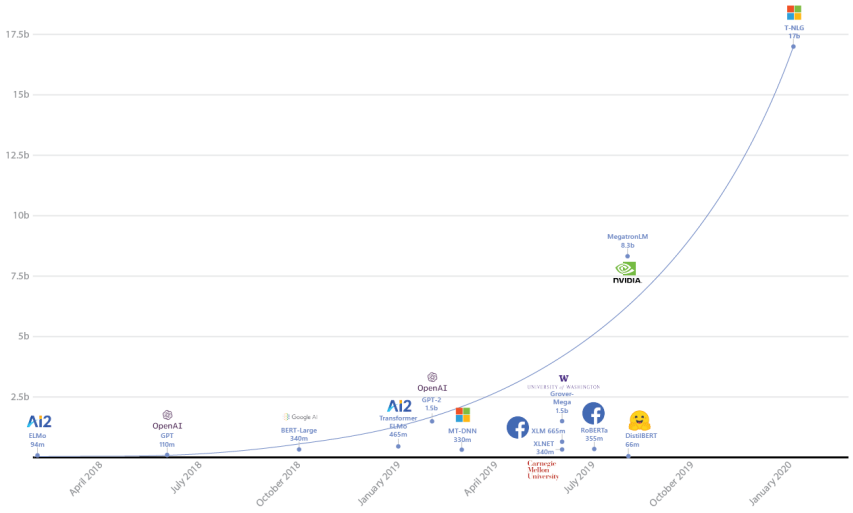

Figure 2.6: Comparison of the parameters of each Transformer model until Turing-NLG[4]

The innovation behind Transformers boils down to three main concepts: Positional Encoding, Attention, and Self-Attention.

[4]Corby Rosset (2020). Turing-NLG: A 17-billion-parameter language model by Microsoft. Retrieved from https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/

The main idea of positional encoding is to append to each word a number with its order in the sentence. Storing word order as data, not structure, makes it easier for the model to learn the position of each word or the distance between different words in the sequence.

Attention is a mechanism that was already used in RNNs since it allows a text model to "look at" the context of every single word in the input sequence when predicting the output sequence. By seeing thousands of examples, the model learns what types of words are interdependent. It learns how to respect gender, plurality, and other rules of grammar.

Self-attention allows a neural network to understand a word in the context of the words around it. As the model processes each word (each position in the input sequence), self-attention allows it to look at other positions in the input sequence for clues that can help lead to a better encoding for this word. This self-attention mechanism is further refined by adding a "multi-headed" attention mechanism that allows the model to focus on different positions and gives the attention layer multiple representation subspaces, i.e., different attention representations for each input embedding.

2.2.2.1 Transformer architecture

A transformer has an encoder-decoder architecture. The encoding component is a stack of N encoders, while the decoding component is a stack of decoders of the same number.

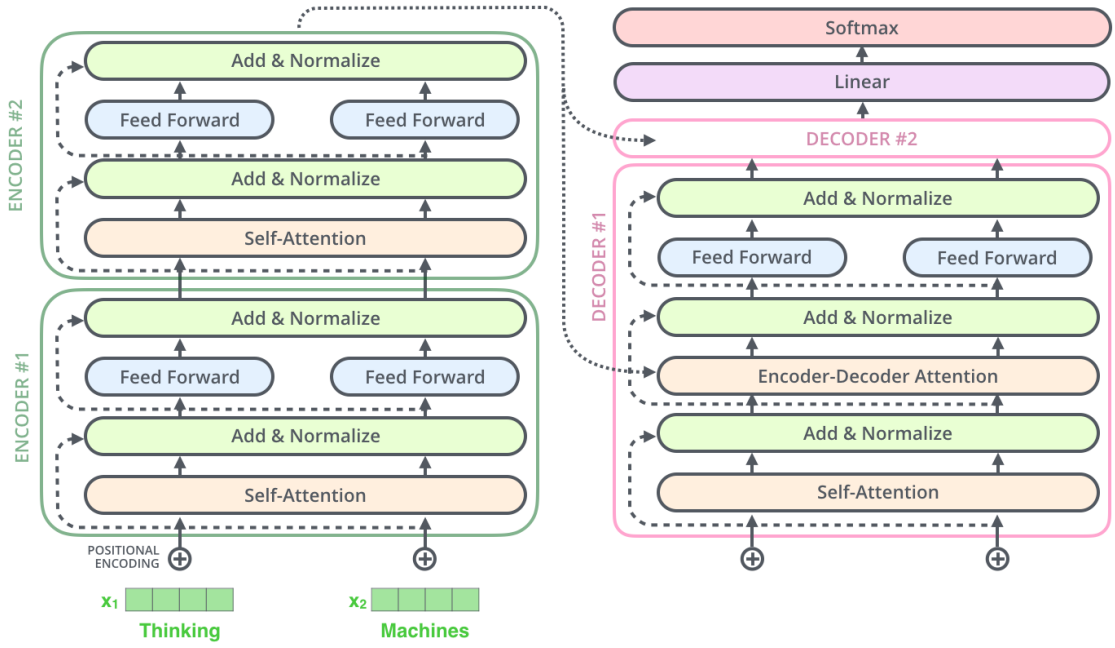

Figure 2.7: Transformer diagram of two stacked encoder-decoders[5]

[5]Alammar, J (2018). The Illustrated Transformer. Retrieved from https://jalammar.github.io/illustrated-transformer/

The word is transformed to an embedding vector and a positional encoding is added to it before entering into the first encoder. The encoders are divided into two sublayers: The encoder's inputs first flow through a self-attention layer, which helps the encoder look at other relevant words in the input sentence as it encodes a specific word. The outputs of the self-attention layer are fed to a feed-forward neural network, independent for each position and thus the various paths can be executed in parallel while flowing through the feed-forward layer. In addition, each sublayer has a residual connection followed by a layer-normalization step, a technique to optimize the training process of neural network architectures.

The decoder has also both those layers, but between them, there is an encoder-decoder attention layer that helps the decoder focus on appropriate places in the input sequence. The output of the decoder is fed into a fully connected neural network that projects the output into a logits vector of vocabulary size length. Finally, the softmax layer outputs the predicted probability for each word of the model vocabulary, selecting the word associated with the logit with the highest probability.

2.2.2.2 Transformers and transfer learning

One of the most popular Transformer-based models is called BERT, short for "Bidirectional Encoder Representations from Transformers" [16]. It was trained by Google researchers on a massive text corpus and open-sourced to be used as a readily available component for any NLP pipeline.

BERT proved that you could build very good language models trained on unlabeled data like text scraped from Wikipedia and Reddit and that these large "base" models could then be adapted with domain-specific data to lots of different use cases. BERT is basically a trained Transformer Encoder stack, that takes a sequence of words as input which keeps flowing up the stack. At the output layer, and embedding that abstracts meaningful information of the input sequence is generated and it can be used to feed a linear neural network layer to classify the sequence among different classes. This process is known as finetuning and basically provides a method to leverage large pre-trained language models to build a model for a downstream task. BERT allowed researchers to smash multiple benchmarks with minimal task-specific fine-tuning and provided the rest of the NLP community with pre-trained models that could easily (with fewer data and less compute time) be fine-tuned and implemented to produce state of the art results.

Since BERT, there have been several improvements in the architecture and training process to create optimized pre-trained models (RoBERTa), more efficient architectures (DistilBERT), or multilingual models (XLM- RoBERTa).

CHAPTER 3

State of Art

Causal inference from observational data is well-studied and has become a topic of growing interest in many machine learning fields to enhance the predictions with causal knowledge. This chapter offers the reader a review of all the previous knowledge and works that have inspired ours. It explains the current approaches to apply causal inference in the natural language processing domain.

3.1 Causal inference and machine learning

The revolution of neural networks in the last decade has produced advances in almost all fields of Artificial Intelligence and has been successfully applied in a wide range of problems in all scientific domains. Yet the limitations of these methods have become obvious when we try to solve more complex problems where pattern recognition is not enough. For this reason, more and more researchers are showing interest in including causal knowledge in new methods that go beyond pattern recognition, building new algorithms to advance the field of causal inference leveraging the power of neural networks.

A recent publication by Luo et al. [37] discusses the main advances in causal relations discovery using neural network algorithms. One of the most popular causal models is the Bayesian Networks, which encode the conditional independencies between variables using directed acyclic graphs. This model has shown its utility in several machine learning challenges, such as interpretable learning, which is a key factor to support decision making. In this line, Kim and Bastani [33], propose a framework for learning causal interpretable models from observational data outperforming or at least being equal to baseline models. However, methods like this one are still computationally expensive due to their combinatorial nature and inefficient search strategies. To solve this problem, a breakthrough work by Zheng et al. [80] established an elegant mathematical connection between the classical combinatorial optimization of the searching space of possible DAG solutions and the continuous optimization of machine learning. They re-formulate the problem of estimating the structure of DAGs as a continuous optimization problem over matrices, avoiding the combinatorial constraints.

Building on this framework, Lachapelle et all [35] proposed a novel score-based approach to learning a directed acyclic graph (DAG) from observational data supporting nonlinear relationships, analyzing the weights of a neural network to reveal the relations between variables and the result. Also, Yu et al [78] proposed a deep generative model and applied a variant of the structural constraint to learn the DAG. Moreover, the previous work by Zheng et al. is further generalized to nonparametric models [81], outperforming Yu et al. work. These advances provide neural architecture search algorithms that could be further explored to achieve both model accuracy and transparency simultaneously. Shi et al [64] address the use of neural networks for the estimation of treatment effects from observational data to find whether it is possible to adapt the design and training of neural networks to improve the quality of the final estimate of the treatment effect. They propose a new neural network architecture that exploits the sufficiency of the propensity score for estimation adjustment, and a regularization procedure that induces a bias towards models that have non-parametrically optimal asymptotic properties.

3.2 Causal inference: Text-based use cases

Causal inference is also a topic of interest in the Natural Language Processing field. The Causal Inference field has taken advantage of the astonishing advances of NLP in recent years, creating new methods to exploit the unstructured nature of the text to estimate the causal associations present in the data.

A work by Egami et al [19] aims to connect the text as data literature with the growing literature on causal inference. For that purpose, they introduced a conceptual framework enabling researchers to discover high-dimensional textual interventions and estimate the ways that observed treatments affect text-based outcomes while avoiding the problems of identification and estimation when using text as the data source. This framework is based on the idea of learning a latent representation of the text, that is sufficient for causal inferences from that text. They identify the problems of identification and estimation when using text data, and propose and formalize a solution: split the text data into a train and test set. While this is not a novel solution, they provide a formal description of how this solution works. Also, they provide two examples of using this framework: when the text is a treatment and when the text is an outcome.

In the pursuit of meaningful and generalizable learning in text, a work by Paul [49] proposes a matching technique for learning causal associations between word features and class labels in document classification, to answer the question "which word features *cause* documents to have the class labels that they do?" Experiments with binary sentiment classification on corpora from different domains show a superior performance of using propensity score matching for document classification, allowing us to learn if a feature has a causal effect on document classes.

A recent work by Veitch et al [71] develops a method to estimate causal effects from observational text data, adjusting for confounding features of the text such as the subject or writing quality. Say that we're interested in finding what writing styles will help posts become popular. Some authors list their genders on Reddit, and a user's gender may also affect popularity through tone, style, or topic choices. How do you decide what kind of language to recommend to any person, regardless of their gender? The challenge is to produce a low-dimensional representation of text that is sufficient for causal adjustment. Thus, they adapt two modern tools for language modeling — BERT and topic modeling — to produce representations that predict the treatment and outcome well. They find that modeling language improves effect estimation and that the proposed causally sufficient representations adjust for confounding as expected, being able to learn document embeddings that can predict both treatment and outcome. Also, they propose a methodology for empirical evaluation that uses real text documents to simulate outcomes with confounding.

Other authors have studied how features in the text influence crowdfunding campaigns or irony detection. Yang et al [76] propose a neural network to quantify how persuasive is a text and identify the persuasive strategies in advocacy requests. The method aims to model persuasion in requests from crowdfunding websites and identify the different strategies at the sentence level. McHardy et al [40] propose a novel method based on an adversarial architecture to control the confounding variable (writing style of publication source) in the automatic detection of satire in news. Using word embeddings to represent the news, the encoder is encouraged to learn representations of the text which are unrelated to the confounds. They divide the problem into satire detection and publication source identification, achieving comparable results to non-adversarial methods but with the correct modeling of satire.

Pryzant has several works pursuing to make NLP models interpretable, which often requires identifying features in the input that predict outcomes while also controlling for potential confounds. In one of his works [56], they hypothesize that textual product descriptions are also important determinants of consumer choice. Thus, they propose a method to identify actionable writing styles and word usages that are highly predictive of consumer purchasing behavior, while controlling for confounders (brand loyalty, pricing strategies). Later, the problem is formalized as a new task [57]: inducing a lexicon that is predictive of a set of target variables yet uncorrelated to a set of confounding variables. Also, two deep learning algorithms are proposed: a bifurcated architecture to separate the explanatory power of the text and confounds, and an adversarial discriminator to force confound-invariant text encodings.

Developing on his previous works, they present two methods for performance attribution [54]: finding the degree to which an outcome can be attributed to parts of a text while controlling for potential confounders. One method uses a CNN to encode the text, an adversarial objective function to control for confounders and projects its weights onto its activations to interpret the importance of each phrase towards each output class. The other method leverages residualization to control for confounds and performs interpretation by aggregating over learned word vectors. Finally, in his last work (last quarter 2020), the problem of estimating the causal effects of linguistic properties on downstream outcomes is considered (i.e., How much will a positive product review increase sales?). They formalize the causal quantity of interest, the effect of the writer's intent, to be able to identify this from observational data. Also, as it is only possible to have access to noisy proxies like lexicons or classifiers, they propose an estimator composed of a model to adjust for the text and distant supervision to improve the quality of noisy proxies. They make experiments on two scenarios: predicting the effect of music reviews sentiment on sales, and complaint politeness on response time.

3.3 Causal Inference in social media

Several researchers have applied causal inference methods using observational data from social networks, incorporating in recent years advances in NLP to also use data present in text form.

Veitch et al [72] explore the use of networks as a proxy of the unobserved confounding, creating embedding models of the most used network in Slovakia to yield inferences about the response of a drug in people connected in the social network. Causal inference methods have demonstrated good results using observational data from social media. A study carried by Pavalanathan & Eisenstein [50] tested whether the adoption of emojis in 2015 in Twitter caused a decrease in the use of emoticons, using a dataset of tweets and obtaining a significant decrease in emoticon use. In a recent study, Gencoglu and Gruber [20] propose a causal inference approach to discover and quantify causal relationships between pandemic characteristics (e.g. number of infections and deaths) and Twitter activity as well as the public sentiment, with text data gathered from Twitter during the COVID-19 pandemic. Their results show the ability to distinguish events that correlate with public attention from events that cause public attention.

3.4 TextCause algorithm

The core of this project is based on a work started by Veitch et al. [71] that was further developed by Pryzant et al. [55], where the TextCause algorithm was proposed. The goal of this algorithm is to estimate causal effects from observational text data, adjusting for confounding features of the text. This section will describe both works so the reader can understand the foundation ideas behind this project.

3.4.1 Adapting Text Embeddings for Causal Inference - Veitch et al.

This work proposes the use of *causally sufficient embeddings* to adjust the confounding features of the text. The underlying idea behind *causally efficient embeddings* is to learn embeddings of text documents that retain only information that is predictive of the treatment and outcome, and also relevant for language understanding.

Figure 3.1 shows the first assumption: The words of the text W_i carry sufficient information to adjust for confounding between treatment T_i and outcome Y_i. The second assumption is that only some properties of the text $z_i = f(w_i)$ suffice for identification. However, we do not observe z_i but the raw text, so the goal is to reduce w_i to a feature z_i with sufficient information to compute the causal effects. This problem is approached by creating an embedding vector $\lambda(w)$ from the words of each document, which would capture

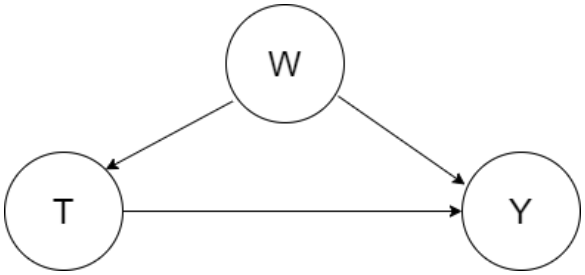

Figure 3.1: Text as a confounder

the confounding aspects of the text. The third assumption is that features of the text that are useful for language understanding are also useful for eliminating confounding, so the authors can rely on the recent advances in language modeling.

To produce these *causally sufficient embeddings*, the authors adapt language understanding models to take in the words w_i and produce the embedding λ_i along with the necessary estimates to compute the causal effect. The authors develop two methods that produce causally sufficient document embeddings: causal BERT and the causal topic model.

Causal BERT is a modification of the BERT transformer model and the method further developed by Pryzant et al. Each input to BERT is the document text w_i and the model is tasked to produce three different outputs:

1. document-level embeddings
2. a map from the embeddings to treatment probability
3. a map from the embeddings to expected outcomes for the treated and untreated.

Then, a fine-tuning approach based on a neural net head on top of BERT is used to predict the estimates of interests to compute the causal effect. In other words, the model objective is designed to predict both the treatment and the outcome and thus adapt the embedding to be useful for the downstream task, i.e., for causal inference. The authors prove that this method is valid for causal inference, since if $\lambda(w)$ carries the relevant. information for the prediction task then it is also causally sufficient. That is, if $\lambda(w)$ preserves the information for both treatment and output prediction, it must also preserve the information for causal effects estimation. They also validate this method in an empirical setup, addressing the problem of unavailable ground causal effect data by using semi-synthetic data where an outcome is simulated from a treatment and a confounder in a real text document dataset.

3.4.2 Causal Effects of Linguistic Properties - Pryzant et al.

Built on the results from the previous work, Pryzant et al. also consider the problem of estimating the causal effect of linguistic properties on downstream outcomes. They approach

two challenges in this problem:

1. Formalize the causal quantity of interest, the writer's intent, and identify it from observational data.

2. Identification: establish the assumptions we need to estimate effects with noisy proxies to be able to build an estimator.

The first challenge is addressed by describing the hypothetical intervention corresponding to the causal effect of interest. This is done by proposing the following causal model:

A writer uses a linguistic property T (the writer intent to be polite or not) and other properties Z (topic, sentiment, etc), which may be correlated, to write text W. From the text, the reader perceives the property of interest, captured by $\widetilde{T}$, and together with other perceived information $\widetilde{Z}$, produces the outcome Y (a faster response). The proxy label of the property obtained via a classifier or lexicon is captured by $\hat{T}$ (prediction made using a model for politeness). This model makes clear that adjusting on the text W is sufficient to block the backdoor path between $\widetilde{T}$ and Y and thus allows to compute the Average Treatment Effect (ATE) of the treatment adjusting for the confounders.

This is where the first problem appears: we want to evaluate the causal effect of the writer's intent T, but we have only access to the reader's perception of the observational data $\widetilde{T}$. To solve this problem, the authors prove that, assuming that the reader correctly identifies the writer's intent, the effect on the reader is equivalent to the effect on the writer. The second problem is we do not observe the property $\widetilde{T}$ directly, instead, we use a model based on a classifier or domain-specific lexicon to predict the value of this property from the text, producing a proxy label $\hat{T}$. The authors prove that this estimand only attenuates the ATE we want to compute and it does not change the sign. That is, using a proxy label can only decrease the magnitude of the effect due to the bias error and in any case, the effect would be inverted, i.e., estimate a negative causal relation when it is positive in the ground truth casual effect. This result led to the conclusion that constructing the most accurate proxy treatment $\hat{T}$ possible will produce the most accurate estimate of the ATE, as long as we adjust for the text.

To this end, the authors propose an approach for improving the accuracy of proxy labels using distant supervision to improve them. They train a classifier to predict the proxy label $\hat{T}$ and then using that classifier (logistic regression trained with bag-of-words) to relabel examples which were labeled $\hat{T} = 0$ but look like $\hat{T} = 1$. Once we have the improved proxy labels $\hat{T}*$, we can adjust for the confounding parts of the text as Veitch et al. proposed, obtaining an estimator for the causal effect. Pryzant et al. implemented the TextCause algorithm using the DistilBERT language model, a light Transformer model based on BERT

architecture with 40% fewer parameters and 60% faster while preserving 97% of BERT's performance.

The algorithm performing these two steps, treatment boosting and text adjusting, was named TextCause. It allows computing the ATE of treatment on the output in a text setting with additional confounder covariates. The authors validated their proposed algorithm on a semi-synthetic Amazon's review dataset and they also conducted an applied study using real-world complaints to find the effect of politeness on a timely response.

CHAPTER 4

Enabling Technologies

This chapter offers a brief review of the main technologies that have made possible this project, as well as some of the related published works.

4.1 Machine Learning Technologies

4.1.1 Scikit-learn

Scikit-learn[51] is an open-source machine learning Python library built on NumPy, SciPy, and matplotlib that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.

This project was created by David Cournapeau as a Google Summer of Code project. Later, Matthieu Brucher started work on this project as part of his thesis. Finally, INRIA assumed the leadership of the project and made the first distribution in 2010. Since then, they have published several releases with the help of a growing international community.

We have used this library to make some preprocessing steps and to evaluate different models.

4.1.2 Numpy

Numpy [23] is the primary array programming library for the Python language, having an essential role in research analysis pipelines in fields as diverse as physics, chemistry, astronomy, geoscience, biology, psychology, materials science, engineering, finance, and economics. It provides a simple but yet powerful programming paradigm for organizing, exploring, and analyzing scientific data.

Numpy is an open-source project aiming to enable numerical computing with Python. It was created in 2005, building on the early work of the Numeric and Numarray libraries.

We have used this library to perform operations involving huge matrices.

4.1.3 Pytorch

PyTorch [48] is a Python package that provides two high-level features: Tensor computation (like NumPy) with strong GPU acceleration and Deep neural networks built on a tape-based autograd system. It accomplishes two goals, usability, and speed. Pytorch provides an imperative and Pythonic programming style that supports code as a model, makes debugging easy, and is consistent with other popular scientific computing libraries, while remaining efficient and supporting hardware accelerators such as GPUs.

Usually, PyTorch is used either as a replacement for NumPy to use the power of GPUs or as a deep learning research platform that provides maximum flexibility and speed. We have used this library to build and modify the neural network architectures we developed.

4.1.4 HuggingFace Libraries

HuggingFace is a company created from an open-source project to open the recent advances on Natural Language Processing (NLP) to the wider machine learning community. They provide several libraries to easily build, share, and have access to state-of-the-art transformer models, NLP datasets and to enable NLP pipelines with processing tools. We have heavily leveraged Transformers library [73], an open-source library that consists of carefully engineered state-of-the-art Transformer architectures under a unified API. Backing this library is a curated collection of pre-trained models made by and available for the community. Transformers are designed to be extensible by researchers, simple for practitioners, and fast and robust in industrial deployments

4.1.5 DistilBERT

DistilBERT [61] is a transformers model, smaller and faster than BERT, which was pre-trained on the same corpus in a self-supervised fashion, using the BERT base model as a teacher. It proofs that it is possible to reduce the size of a BERT model by 40% while retaining 97% of its language understanding capabilities and being 60% faster.

We have used this transformer architecture as the model powering our system.

4.2 Data technologies

4.2.1 Pandas

Pandas [47] is an open-source Python library that provides high-performance, easy-to-use data structures, and data analysis tools. It offers data structures and operations for manipulating numerical tables and time series. The principal features and functionalities that provide pandas library are data grouping, merging, and querying operations as well as time series analysis, reading and writing tools in different file formats, and data munging.

Development was started in 2008 by Wes McKinney to perform quantitative analysis on financial data. In 2015, pandas became a NumFOCUS sponsored project, a non-profit organization from the United States which promotes open-source projects.

Pandas was used in this work to manage, clean, and manipulate our datasets.

4.2.2 Matplotlib

Matplotlib [31] is a Python library that provides 2D plotting capabilities to produce publication-quality figures such as plots, histograms, power spectra, scatterplots, and error charts with just a few lines of code.

It has been originally developed by John Hunter and has grown with the contribution of a large community, becoming one of the NumFocus sponsored projects.

We have used this library to produce the figures shown in this work.

4.2.3 Twint

Twint [67] is an advanced Twitter scraping tool written in Python that allows for scraping Tweets from Twitter profiles without using Twitter's API. It provides most of the functionality needed to exploit Twitter as a data source. Twint makes it easier to gather data from Twitter without the need for authentication and the limits of the API. It utilizes Twitter's search operators to let you scrape Tweets from specific users, scrape Tweets relating to certain topics, hashtags & trends, or scrape user's information.

We have used this library to easily utilize the Twitter Streaming API. In this work, we used the Streaming API to download Twitter messages in real-time. It is useful for obtaining a high volume of tweets, which suits perfectly our goal of collecting a tweet dataset.

4.3 Sentiment analysis

4.3.1 Senpy

Senpy [66] is a framework for sentiment and emotion analysis services. Its goal is to produce analysis services that are interchangeable and fully interoperable. All services built using Senpy share a common interface, which allows users to use them simply by pointing to a different URL or changing a parameter.

The development started in 2014 and was carried out by the Intelligent Systems Group at ETSIT, UPM, as part of the European Mixed Emotions project.

A Senpy plugin to easily use the LIWC dictionary has been developed. Furthermore, this library has allowed us to perform sentiment and emotion analysis on our datasets to carry out further analysis.

4.3.2 LIWC

Linguistic Inquiry and Word Count (LIWC) [52] is a transparent text analysis program that counts words in psychologically meaningful categories. Empirical results using LIWC demonstrate its ability to detect meaning in a wide variety of experimental settings, including to show attentional focus, emotionality, social relationships, thinking styles, and individual differences.

The heart of the program is a group of dictionaries that allow comparing each word in the text against them. The dictionaries identify which words are associated with which

psychologically relevant categories.

We have used this dictionary to extract meaningful features from the text.

4.3.3 Lexicons

We have made use of different sentiment lexicons. In particular, we have utilized the Spanish lexicon from ML-Senticon [14], a set of sentiment lexicons at the lemma level for English, Spanish, Catalan, Basque, and Galician. For each lemma, it provides an estimation of polarity (from very negative -1.0 to very positive +1.0) and a standard deviation. Also, we used the German positive lexicon from the multilingual sentiment open-source project [34].

CHAPTER 5

Architecture

This chapter presents the methodology used in this work. It describes the overall architecture of the project, with the connections between the different components involved in the development of the project.

5.1 Framework

The main goal of our system is to find how different linguistic attributes produce an effect on a specific output. For example, "how much is the effect of writing an angry message on the response time in social networks?". Our system makes it possible to answer this kind of causal question, leveraging state-of-the-art methods to assess causality on text datasets and different modules to describe the linguistic properties of interest such as the sentiment or emotion of the text. It also allows computing the causal effect of any other treatment on the desired output, making this a perfect tool to answer questions in a wide range of fields where text is an important data source, such as social network analysis, marketing, e-commerce, or other social science studies.

Figure 5.1 shows the architecture of the system. The core of the system is the TextCause algorithm proposed by Pryzant et al., which was described in Section 3.4.2 and evaluated in Section 5.2. This method has been modified to be able to test it in different settings and use it for different languages. Linguistic properties like emotion or sentiment could be computed from the text using any estimator. Our system proposes the use of three different modules to compute different linguistic attributes from the text and evaluate their effect on the output. Two of them are based on pre-trained language models with a transformer architecture, the sentiment classification module (Section 5.4.1) and the emotion classification module (Section 5.4.2). The third one is lexicon-based, the LIWC module (Section 5.5).

The system takes as input a dataset with text W, the output variable Y we want to know how is affected by different variables C_N or linguistic properties. These covariates could also be used as confounder covariates. The different modules of the system compute the proxy treatment for each of the desired linguistic attributes of the study. This is the input information for the TextCause module, which outputs the Average Treatment Effect (ATE) for each of the studied variables.

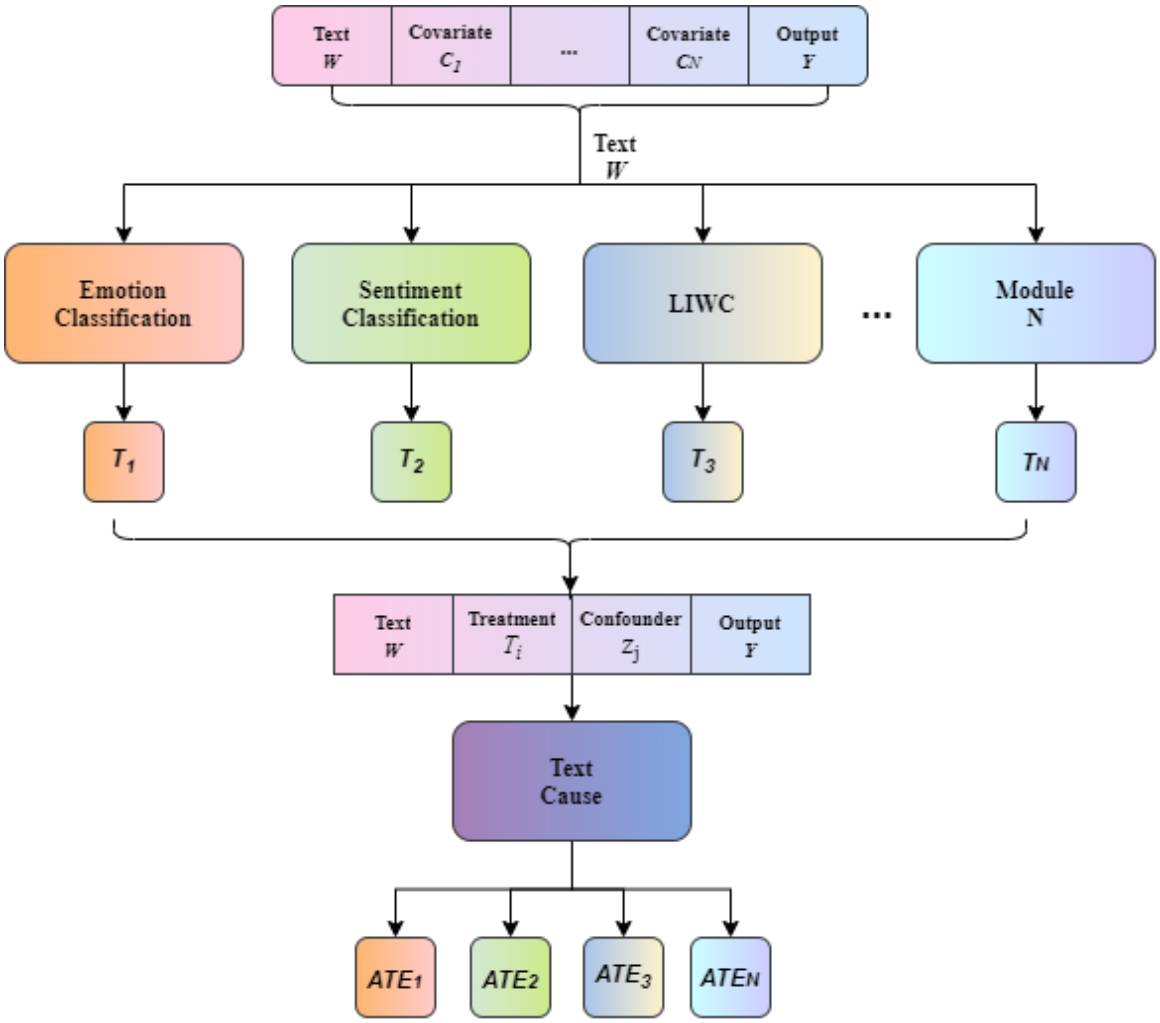

Figure 5.1: System architecture

5.2 TextCause module

This section describes the process followed to implement, validate, and evaluate the TextCause algorithm proposed by Pryzant et al. First, we explain how we reproduced the results claimed by the authors and provide further insights on the robustness of the method. Then, we expose the modification made to the algorithm to ease the experimentation. Finally, we propose a new multilanguage semi-synthetic dataset to test the algorithm in a new setting from a different domain.

5.2.1 Reproducing paper

The first goal of this project was to implement and reproduce the results claimed by the authors of the original paper. To this end, an exhaustive study of the work foundations, proposal, and implementation was carried out aiming to understand and effectively implement the proposed algorithm. This has also allowed us to further study the robustness and variability of the algorithm and extend it to fit our objectives.

5.2.1.1 Amazon reviews experiment

The first step was to reproduce the results obtained in the semi-synthetic Amazon reviews dataset, a dataset with 200K examples of reviews scraped from Amazon containing the product, the text of the review, and the rating score among other characteristics. We used the script provided by the authors to reproduce the results with the parameters described in their paper. The experiment was developed as follows:

1. The dataset was built by the authors from a corpus of Amazon reviews of music products. They excluded reviews from products worth more than $100 or fewer than 5 words. The confounder covariate is a binary indicator for whether the associated review is a CD or not. The treatment $\widetilde{T}$ is whether the review is positive (5 stars) or not (1 or 2 stars). This variable is used to simulate the outcomes and to calculate ground truth causal effects for evaluation.

2. The proxy treatment $\hat{T}$ is computed via a random noised version of T or with a binary indicator for whether any words overlap with a positive sentiment lexicon. The preprocessed dataset consists of 17,000 examples.

3. The language model used by the algorithm is DistilBERT, which was trained for 3 epochs and a batch size of 32.

4. We reproduced the 8 different experiments with all possible combinations of noise (0 or 1), simulated confounding (-0.4 or 0.4), and simulated treatment (0.4 or 0.8).

These variables were used to simulate the outcomes. Each experiment was replicated using 100 different random seeds for robustness.

Each experiment for each seed took an average of 20 minutes on an NVIDIA GeForce RTX 2080 Ti GPU, adding a total of 266 hours ($8 combinations * 100 seeds * 20 minutes$). Hence, it took us 5.5 days to reproduce the experiment on two GPUs. The environmental impact and the cost of running these kinds of experiments are discussed in Section A.3 and Chapter B.

Table 5.1 shows the ATE estimates obtained when reproducing the claimed results of Pryzant et al. This effort to validate the claimed results led to finding small errors in the published paper which were offset by the authors. The estimated ATE is the expected change in click probability if one were to manipulate the sentiment of a review from negative to positive. The true ATE is given in the top row ("oracle"), estimates closer to the oracle are better. The last column gives the average difference between the estimated and true ATEs; the lower, the better. Now we discuss each of the results:

- Although the semi-oracle method [75] obtains a slightly better result, it assumes additional access to the ground truth measurement model, i.e., the model that associates T and $\hat{T}$, which is not a feasible assumption on observational data settings. We can not know the exact relation between the real value of the treatment (how polite is a text) and the measured value (the predicted politeness of a text).

- The unadjusted method is the expected difference in outcomes conditioned on $\hat{T}$, so it does not take into account any confounding factor.

- The proxy baselines (*proxy-lex* and *proxy-noised*) perform backdoor adjustment for the observed covariate (if the review is from a CD or not), using a random noised version of T (to simulate a classifier with a 93% of accuracy) and a lexicon-based proxy.

- T-boost is the result of the proposed method to improve the proxy labels and W-adjust is the method to adjust for the text as we described previously.

- TextCause is the proposed algorithm that combines the previous two methods. We observe that this algorithm significantly outperforms all previous methods (except for the semi-oracle) with robust estimates to various levels of noise and treatment/confound strength.

These results show that adjusting for the text is crucial since the estimator which adjusted for the covariate C and not the text (proxy-lex) achieved poor results compared to

Noise:	**Low**				**High**				
Treatment:	**Low**		**High**		**Low**		**High**		**Mean delta**
Confounding:	**Low**	**High**	**Low**	**High**	**Low**	**High**	**Low**	**High**	**from oracle**
oracle	9.93	10.03	18.99	19.31	8.28	8.27	16.04	16.16	0.00
semi-oracle	9.73	9.82	18.78	19.08	8.25	8.26	16.02	16.19	0.12
unadjusted	6.85	7.67	13.53	14.50	5.79	6.42	11.51	12.26	3.56
proxy-lex	6.67	6.73	12.88	13.09	5.66	5.66	10.99	11.10	4.28
proxy-noised	8.26	8.27	15.90	16.12	6.69	6.72	13.22	13.34	2.31
+T-boost	7.73	8.60	14.97	16.12	6.62	7.23	12.94	13.73	2.38
+W-Adjust	7.73	8.60	14.97	16.12	6.62	7.23	12.94	13.73	2.38
+TextCause	**9.44**	**10.28**	**18.19**	**19.32**	**7.85**	**8.50**	**15.47**	**16.32**	**0.37**

Table 5.1: Reproduced results on Amazon review dataset from Pryzant et al.

the estimator which also adjusted for the text (W-adjust), even worse than the unadjusted estimator. In addition, the better the accuracy of the proxy for the treatment, the better the results. The proxy-noised estimator (which assumes a proxy accuracy of 93%) significantly improves over the unadjusted estimator, and the TextCause algorithm (proxy label boosting + text adjusting) achieves almost semi-oracle results. Indeed, the authors perform a sensitivity analysis that demonstrates how the accuracy of the proxy for the treatment significantly affects the ATE estimation.

We conducted small research about the variability of the results provided by the algorithm and to discover to which end it is possible to use less random seeds and still have robust results. Figure 5.2 shows the standard deviation of the proposed estimators. The simulated True ATE (oracle) has low variability as expected, while the semi-oracle method slightly increases it. The text adjusting method maintains a constant variability across the different experiments, while those estimators using proxy label improvement increase the variability. In any case, the variability remains almost constant for all estimators among the different experiment settings.

One important drawback of the TextCause algorithm is the high number of random seeds the authors used to provide robust results. We carried out a study to assess to what extent this is necessary, comparing the differences in the results with less number of seeds. We randomly sampled the results for several seeds from 10 to 100 in steps of 10,

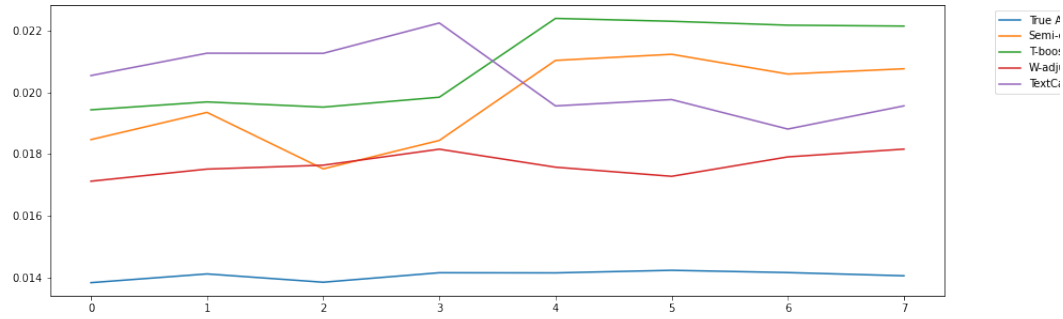

Figure 5.2: Standard deviation for each of the 8 experiments over the 100 random seeds

and we averaged this random sampling over 100 trials. Figure 5.3 shows the results of this experiment with the number of seeds represented on axis X and the difference to the results using 100 random seeds on-axis Y. We can observe how the larger the number of seeds we use, the less difference we found to the experiment using 100 seeds (where the difference becomes zero for evident reasons). We also observe a little difference of approximately 0.0006 between using 100 seeds or 10 seeds, so we assumed for the next experiments that using 10 seeds is robust enough.

5.2.1.2 Consumer complaints experiment

We have also reproduced the other experiment described by the authors, an applied pilot study that seeks to answer, "how does the perceived politeness of a complaint affect the time it takes for that complaint to be addressed?". As the code was not provided, we implemented our own script following the steps described in the paper:

1. A corpus of complaints filed with the Consumer Financial Protection Bureau (CFPB)[1] was used. This is a governmental agency that manages the complaints filed about financial products, and it records the time a company takes to process the complaint. We gathered a corpus of a total of 1.9M complaints arranging from dates from 2011-12-01 to 2021-02-11. We do not have the exact dataset that was used in the original experiment. This dataset has the financial product the complaint is about, the text of the complaint, and the dates the complaint was registered and processed, among other features.

2. The threshold for a quick response is in 15 days, which will become our outcome Y.

3. A confound covariate C is also considered to adjust for the type of financial product

[1] https://www.consumer-action.org/

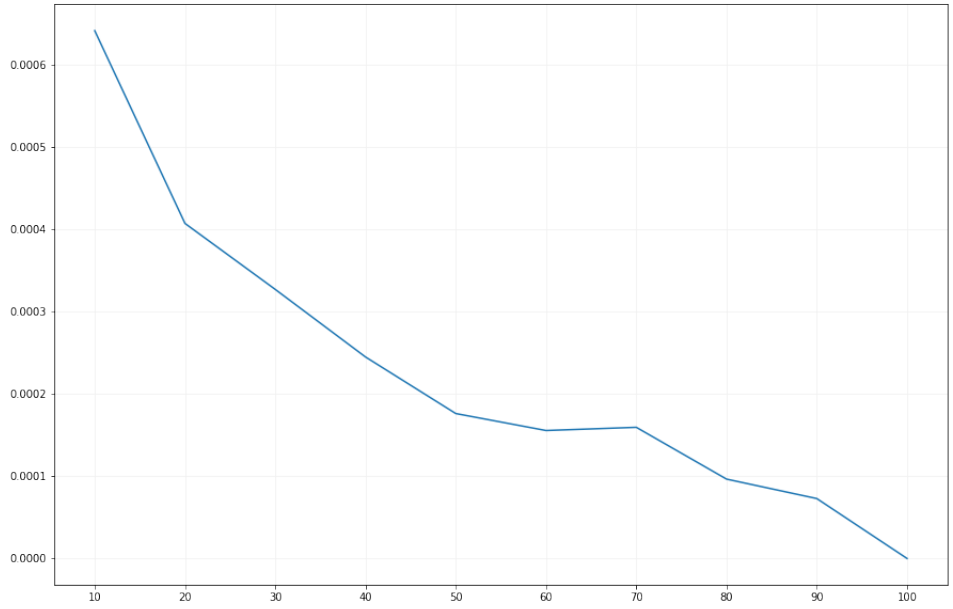

Figure 5.3: Average difference of TextCause results with different number of seeds

(whether the complaint is about *Debt collection* or not). This is another difference in our experiment setting since the proposed confound adjusting (whether a complaint was about mortgage or bank account) was not the most representative type of product in the dataset.

4. To reduce other potentially confounding effects, we pair each Y = 1 complaint with the most similar Y = 0 complaints according to the cosine similarity of TF-IDF vectors. This has been a complex computing operation due to the huge size of the dataset. After filtering the examples without the consumer complaint narrative, we ended up with a dataset of 667K examples. This dataset is heavily unbalanced since only 15K are from the "late response" class. Then, we compute the TF-IDF vectors of the complaints. This vectorizer transform operation has linear time complexity O(N) and generates a matrix of almost 150K features.

5. Now we want to compute the cosine similarity matrix between the *quick-reply class* matrix of dimension $650k * 150k$ and *late-reply class* matrix of dimension $15.5K * 150k$, so the memory requirements we need are: $650K * 15.5K * 64bits/(8bits/B) = 80GB$. This becomes a huge RAM requirement, so we implemented the cosine similarity algorithms making use of the hard drive storage to compute the matrix in chunks.

The computation of the matrix takes 25% more time but now we can take advantage of I/O.

6. Finally, the 4,000 most similar pairs are selected for a total of 8,000 complaints.

7. For the proxy treatment (politeness), we used the same politeness prediction package [77] than the authors. This package reports a score from a trained classifier using expert features of politeness and a hand-labeled dataset. We took the top 25% most polite ($\hat{T} = 1$) and the top 25% least polite ($\hat{T} = 0$) complaints of our dataset, reducing it to a total number of 4,000 complaints with their proxy label, the binary confound value and the outcome.

8. As the authors, the TextCause algorithm was used with a training of 9 epochs and a cross-validation fold of size 2000.

Table 5.2 shows the results obtained by the original work and ours. As we can observe, there is a significant difference between the results of the two experiments, not only in the actual magnitude of the effect but in the trends among the different estimators. While in the original one the smallest ATE was produced by the unadjusted approach, ours is produced by the *proxy-lex* approach. The W-Adjust and TextCause methods, which adjust for covariates and text, produced the largest ATE estimates. We identified different possible causes:

- **Dataset**: As we gathered our own version of the dataset in a wide timeframe, the possible effects may have changed if the original authors used a shorter timeframe or made any preprocess not described in the paper.

- **Covariate C**: We used a different product to create the binary confounder covariate C. This might affect the total effect if the original covariate C was a strong confounding factor because now it would heavily disturb the computed causal effect.

- **Other confounding covariates**: This experiment is performed in a complex scenario, where might be several confounding factors not present in the dataset such as the company's situation when the complaint was registered (maybe there was a severe incident that made several customers fill their complaints), the dates, those complaints were made (there might be significant changes during holidays) or the economic situation (the situation of most companies were not the same back in 2011 just after the global financial crisis than in 2021 after a year of a pandemic). For this reason, the unconfoundness assumption does not hold and the results provided by the algorithm should be taken with awareness, since the choice of the estimator can yield highly varying conclusions.

Estimator	ATE	
	Pryzan et al.	**Ours**
unadjusted	3.01	1.70
proxy-lex	4.03	0.88
+T-boost	9.64	1.46
+W-Adjust	6.30	2.88
+TextCause	10.30	2.45

Table 5.2: Comparative of results in the complaints experiment

5.2.2 TextCause algorithm modifications

This section describes the modifications we performed to the original algorithm implementation, which are mainly three:

1. **Simulation algorithm generalization**: originally, the simulation was performed ad-hoc for the characteristics of the Amazon review dataset. To be able to generate simulations in new settings using different datasets, we modified how the output was generated depending on how the dataset was balanced.

2. **Parametrization**: To foster faster experimentation, we modified all the algorithm to be able to use different distilBERT-based architectures, different number of hyperparameters such as number of epochs or batch size, whether the model is cased or not, the language of the dataset, the GPU id where the experiment should be deployed and the root name for several output files we generate for analyzing and debugging purposes.

3. **Multi-language TextCause**: Motivated by the significantly less research in the NLP community for languages other than English, we evaluated the TextCause algorithm in new languages: Spanish and German. To this end, we had to modify the algorithm to allow the selection of different language models, lexicons, and preprocessing steps to generalize the algorithm to different language settings. This is further discussed in Section 5.2.3

5.2.3 Simulations for causal inference evaluation

Since ground-truth causal effects are unavailable without randomized controlled trials, we produce a semisynthetic dataset for different languages following the same approach Pryzant et al. used in their work. Our goal is to prove the effectiveness of the proposed method in a new context but with similar experiment settings, and for different languages. To this end, we used a PlayStore app review dataset we gathered for this purpose from *Cabify*, *Uber* and *Lyft* smartphone applications both in Spanish and English. Additionally, we studied the validity of the algorithm on an Amazon general products review dataset in German.

5.2.3.1 Datasets

We collected the App Review dataset both in English and Spanish. We wanted to test how well the algorithm performed in a different context (app reviews from ride-hailing companies). This dataset contains the content of the review, the score given by the user, and the application the user reviewed, among other features we have not used.

The English dataset has a total of 1.3M examples from almost 1M different users who submitted reviews for the Uber and Lyft applications at the PlayStore. There are 1.2M examples from Uber and 100K examples from Lyft. As more the 50% of the examples had 5 words or less, we filtered reviews under 5 words obtaining a dataset of 560K examples with an average of 27 words per review. Figure 5.4 shows a histogram with the distribution of reviews per number of words.

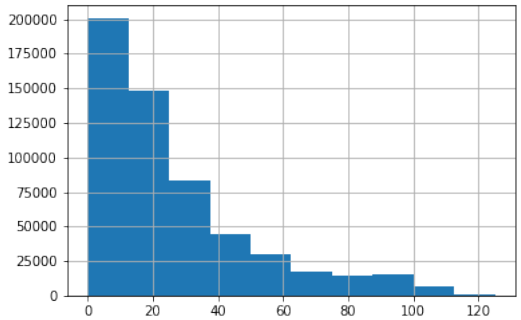

Figure 5.4: Reviews by word number

The confounder covariate C is a binary indicator for whether the review belongs to *Uber* or *Lyft*. The treatment $\widetilde{T}$ is whether the review is positive (5 stars) or not (1 or 2 stars). This variable is used to simulate with our adapted algorithm the outcomes Y and to calculate ground truth causal effects for evaluation. The proxy treatment $\hat{T}$ is computed via a random noised version of T or with a binary indicator for whether any words overlap

with a positive sentiment lexicon.

The Spanish dataset has a total of 750K examples from almost 525K different users who submitted reviews for the Uber and Cabify applications at the PlayStore. There are 670K examples from Uber and 78KK examples from Cabify. We also filtered reviews under 5 words, obtaining a dataset of 387K examples with an average of 25 words per review. Figure 5.5 shows a word cloud with the most frequent words in the dataset, where we observe words related to the app operations and sentiments.

Figure 5.5: Spanish reviews word cloud

The dataset is prepared the same as the English version, but now the confounder covariate C indicates if the app belongs to *Cabify* or *Uber*.

Finally, we filtered the German dataset to take only two different product categories to use them as our confounder covariate C, *home* and *wireless*. After filtering the reviews with less than 5 words, we obtained a dataset of 45K reviews, where 25K were from *home* products and 19K from *wireless products*.

5.2.3.2 Evaluation

The language model used in the English app review dataset is DistilBERT[2], trained for 3 epochs with a batch size of 32; and it is compared against the multilingual version, DistilMBERT[3], which was trained for 3 epochs with a batch size of 16 (a smaller batch size is needed to fit DistilMBERT in the GPU). The goal is to discover to what extent it is possible to use the multilingual model for experiments using text from one of the 104 different languages DistilMBERT was trained on. Following the experiment done by Pryzant et al., we evaluate the method in 8 settings with different combinations of noise,

[2]https://huggingface.co/distilbert-base-uncased
[3]https://huggingface.co/distilbert-base-multilingual-cased

treatment, and confounding levels and averaged over 10 random seeds for robustness, which suffices as we explained in Section 5.2.1.1.

Table 5.3 shows the results obtained in the English App Review dataset, where the column *Mean delta from oracle* shows the average difference to the true ATE across the 8 experiments. As expected, the semi-oracle method achieves the best performance although it is not a feasible method in real-world settings. The unadjusted estimator, the expected difference in outcomes conditioned on $\hat{T}$ that does not take into account any confounder, is the worst-performing estimator. Hence, the proxy baselines which perform backdoor adjustment improve the previous estimator results. The T-boost estimator enhances the effect predicted by the proxy based on the lexicon, although it does not achieve the efficacy of the random noised proxy which simulates a proxy treatment of a 93% accuracy. This led us to think that the proxy treatment based on lexicon may not be the most accurate solution.

The four final rows show the methods using the language models. For the English-specific model, W-adjust achieves a better result than the proxy baseline without text adjustment. However, the multilingual model obtains a worse result than the baseline. This situation is resolved using the full TextCause algorithm, where it achieves the best score using the English-specific language model. Moreover, the TextCause algorithm using the multilingual model improves significantly the W-adjust result. We observe how the difference in the results of using the specific language model for English or multilingual one is an approximate attenuation of 1% to the total effect. **Hence, we conclude that the TextCause algorithm could be easily implemented in different language settings changing the language model and the lexicon used as a proxy.**

Similarly, we carried out the same experiment with a Spanish-specific language model extracted from the DIstilMBERT[4], and the complete multilingual DIstilMBERT model. Table 5.4 shows the results obtained in the Spanish App Review dataset, where the column *Mean delta from oracle* shows the average difference to the true ATE across the 8 experiments. The little improvement of the proxy treatment based on the lexicon over the unadjusted estimator suggests a poor proxy accuracy. However, the treatment label boosting technique enhances the proxy treatment and allows the TextCause algorithm to achieve the best results for the Spanish-specific language model. Although the Spanish DistilBERT achieves a pretty good result, the multilingual DistilBERT performs the best, achieving the most accurate estimate of the ATE.

Finally, we evaluated the experiment for the German-specific language model [5] against the multilingual DIstilMBERT model. Table 5.5 shows the results obtained in the German

[4]https://huggingface.co/Recognai/distilbert-base-es-multilingual-cased

[5]https://huggingface.co/distilbert-base-german-cased

Noise	**Low**				**High**				**Mean delta from oracle**
Treatment	**Low**		**High**		**Low**		**High**		
Confounding	**Low**	**High**	**Low**	**High**	**Low**	**High**	**Low**	**High**	
oracle	10.36	10.66	19.52	19.99	8.67	9.00	16.43	16.88	0.00
semi-oracle	9.95	10.05	19.10	19.26	8.38	8.45	15.99	16.09	0.53
unadjusted	7.37	7.97	14.26	14.92	6.06	6.56	11.84	12.35	3.77
proxy-lex	7.91	8.17	14.65	15.07	6.18	6.38	12.04	12.28	3.60
proxy-noised	7.80	8.03	15.25	15.75	6.40	6.54	12.73	12.82	3.27
+T-boost	7.73	7.83	14.79	14.97	6.75	6.85	12.79	13.01	3.35
+W-adjust	8.65	9.02	16.23	16.68	6.90	7.28	13.28	13.73	2.46
+TextCause	**9.45**	**9.72**	**17.68**	**18.01**	**7.69**	**7.98**	**14.58**	**14.93**	**1.43**
+W-Adjust (multi)	7.39	7.94	14.26	14.87	6.13	6.69	12.00	12.51	3.71
+TextCause (multi)	**8.11**	**8.64**	**15.77**	**16.28**	**7.27**	**7.83**	**13.64**	**14.24**	**2.46**

Table 5.3: Estimators comparison in semi-synthetic App Review dataset using specific English distilbert and multilingual distilbert model. Lower mean delta from oracle is better.

Amazon Review dataset, where the column *Mean delta from oracle* shows the average difference to the true ATE across the 8 experiments. The worse result of the proxy treatment based on the lexicon over the unadjusted estimator suggests a really poor proxy accuracy worsening the prediction. This problem is propagated to the T-boost, W-adjust, and TextCause algorithms, although the former accomplish better results. In this case, although text adjusting with the multilingual model gets worse results than the German model, the TextCause algorithm with the multilingual version achieves the best score for the algorithm. However, the best result is obtained using the proxy-noised estimator, which adjusts for the covariate and approximates the proxy with a 93% accuracy. This outcome confirms that ATE estimates can lose fidelity when the proxy is less than 80% accurate.

Noise	**Low**				**High**				**Mean delta from oracle**
Treatment	**Low**		**High**		**Low**		**High**		
Confounding	**Low**	**High**	**Low**	**High**	**Low**	**High**	**Low**	**High**	
oracle	9.83	9.90	18.88	19.12	8.11	8.21	15.90	16.10	0.00
semi-oracle	9.91	9.92	19.09	19.15	8.04	8.07	15.94	16.00	0.09
unadjusted	6.87	7.39	13.44	13.99	5.61	6.07	11.26	11.74	3.71
proxy-lex	7.18	7.20	13.77	13.92	5.93	6.01	11.63	11.77	3.58
proxy-noised	8.33	8.34	15.88	16.12	7.08	7.04	13.55	13.65	2.01
+T-boost	8.23	8.19	15.60	15.74	6.83	6.81	13.14	13.17	2.29
+W-adjust	6.53	7.07	12.56	13.09	5.27	5.73	10.23	10.66	4.36
+TextCause	**8.61**	**9.30**	**16.71**	**17.38**	**6.88**	**7.37**	**13.62**	**14.11**	**1.51**
+W-adjust (multi)	7.48	8.09	14.32	14.94	6.74	7.26	12.24	12.80	2.84
+TextCause (multi)	**8.84**	**9.73**	**16.88**	**17.72**	**7.24**	**7.93**	**13.96**	**14.63**	**1.21**

Table 5.4: Estimators comparison in semi-synthetic App Review dataset using specific Spanish distilbert and multilingual distilbert model. Lower mean delta from oracle is better.

Noise	**Low**				**High**				**Mean delta from oracle**
Treatment	**Low**		**High**		**Low**		**High**		
Confounding	**Low**	**High**	**Low**	**High**	**Low**	**High**	**Low**	**High**	
oracle	9.18	9.27	18.30	18.54	7.00	7.03	14.84	14.88	0.00
semi-oracle	7.43	7.48	16.21	16.68	4.50	4.37	13.19	13.15	2.01
unadjusted	4.78	5.40	9.45	10.34	3.73	4.13	8.25	8.72	5.53
proxy-lex	3.65	3.69	7.79	8.02	2.27	2.21	6.41	6.40	7.33
proxy-noised	**7.76**	**7.79**	**15.47**	**15.67**	**6.33**	**6.35**	**12.91**	**13.02**	**1.72**
+T-boost	3.81	3.84	9.63	9.97	2.12	2.13	7.46	7.43	6.58
+W-adjust	5.59	5.95	10.19	10.84	4.75	4.90	9.05	9.42	4.79
+TextCause	5.68	5.65	11.70	12.14	4.97	5.03	10.47	10.47	4.12
+W-adjust (multi)	5.29	5.58	10.29	10.99	3.89	4.24	8.67	9.00	5.14
+TextCause (multi)	**6.18**	**6.42**	**12.75**	**13.50**	**5.56**	**5.80**	**11.60**	**11.76**	**3.19**

Table 5.5: Estimators comparison in semi-synthetic Amazon Review dataset using specific German distilbert and multilingual distilbert model. Lower mean delta from oracle is better.

5.2.4 Conclusions

This section has shown how we reproduced the experiment to validate the results obtained by Pryzant et al., and we proposed further experiments in similar experiments yet in different contexts and different languages. We found out that this algorithm is valid for different languages, potentially any of those where DistilMBERT was trained on. Additionally, we confirmed the importance of an accurate proxy treatment to achieve the best performance when estimating the ATE. Finally, we discovered that the English language model trained only on an English corpus performs better than the multilingual language model, although the difference is not that relevant. However, for Spanish and German, the multilingual model estimates better the ATE. This could be due to differences during the training of the base model. Figure 5.6 shows the estimates of the different estimators compared to the True ATE for the analyzed languages.

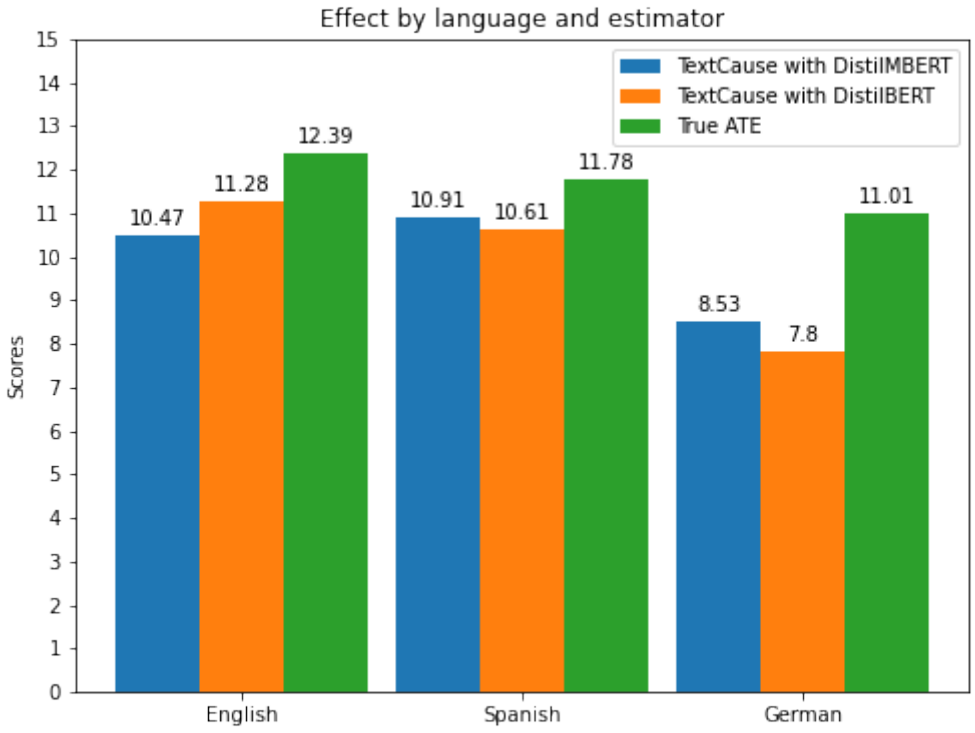

Figure 5.6: ATE estimates of TextCause algorithm for each language

5.3 Model distillation: Is distilBETO really worth it?

Motivated by the empirical results we obtained in Section 5.2.3, where the performance of the TextCause algorithm is greater when we use a language model for a specific language rather than a multilingual model, we studied and implemented the necessary code to train distilBETO, a distilled version of BETO [10], the Spanish version of BERT language model.

BETO is a BERT model trained on a big Spanish corpus [9] composed of different text sources such as the web, Wikipedia, books, official documents, or subtitles that have been curated. BETO is of a size similar to the BERT-Base model and was trained with the Whole Word Masking technique. The benchmarks provided by their authors show a slight improvement over the Multilingual BERT version.

Distillation is the method used to compress a large model, called teacher, into a smaller model, called student. The idea is to train a student network to mimic the full output distribution of the teacher network. This technique has been successfully applied to different language models based on the transformer architecture [30]. The distilling process is possible by using the Kullback-Leiber Loss, a measure of how one probability distribution is different from a second reference probability distribution.

$$KL(p||q) = \sum_i p_i log p_i - \sum_i p_i log q_i$$

The training loss of the distilled model is a linear combination of the distillation loss and the original masked modeling loss. This is a computationally expensive process, for example, DistilBERT was trained using 8 Nvidia V100 16GB GPUs during 3.5 days.

We followed the implementation described in HuggingFace documentation [30] to prepare the code to train distilBETO. First, we downloaded the 20GB of Spanish text data. Then, we modified the provided scripts to use the BETO model and its corresponding tokenizer, so we can process our corpus into a suitable input for the model. As starting distilled training with good initialization of the model weights is crucial to reach decent performance, we also initialized a few layers of the teacher BETO model modifying the corresponding script. At this point we would be ready to train our distilled version of BETO, however, we decided not to carry out this process due to the large computing and time resources that are needed since we only counted with 2 Nvidia GeForce RTX 2080 Ti 12GB GPUs. In addition, the small expected improvement over the multilingual distilBERT model would make the process not worth it and out of the scope of this project.

5.4 Transfer learning and fine-tuning

The idea of transfer learning is to get the advantage of the knowledge obtained by a model trained with lots of data to train a smaller model for another task. In the NLP domain, and more specifically when using large pre-trained language models, this process is known as **fine-tuning**.

This technique provides a means to not only train models faster, but to obtain pre-trained models that yield better results. This is because pre-trained models have a statistical understanding of the language used during pre-training. In practice, fine-tuning is applied on a given model by throwing away its head, that is, its last layers focused on the training objective, and replacing it with a new one that is randomly initialized and suitable for the downstream task of interest. We will focus on building fine-tuned models for the Spanish language, where we find fewer pre-trained models compared to English.

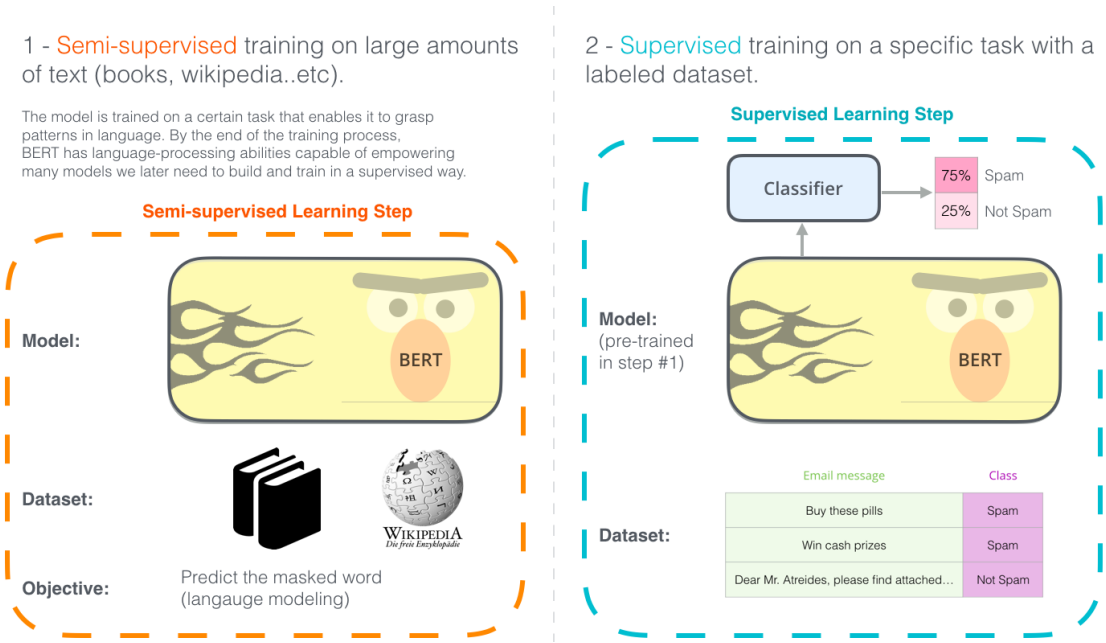

Figure 5.7: Fine-tuning process[6]

We followed the transformers library documentation[7] to fine-tune our models for sentiment and emotion classification. Once we had the dataset for our desired supervised downstream task, we prepared it using the tokenizer and data loader classes, which encode the text to the input embeddings for the model and eases how the data is fed into the model respectively. We used the Trainer class for the training process, which is optimized

[6] Alammar, J (2018). The Illustrated Transformer. Retrieved from https://jalammar.github.io/illustrated-transformer/

[7] https://huggingface.co/transformers/training.html

for transformer finetuning and provides a wide range of training options. Once the fine-tuning process is finished, we can use our finetuned model leveraging on the power of the pre-trained models.

5.4.1 Sentiment analysis module

This module is based on a finetuned model using the Spanish DistilBERT[8], a model extracted from the multilingual version of DistilBERT. These fine-tuned models abstract the input information to classify the text into the most probable class. Figure 5.8 shows the words the model focuses on to predict the sentiment polarity. Here, it focuses on words like "ben ##dicion" or "gracias" to predict a positive polarity.

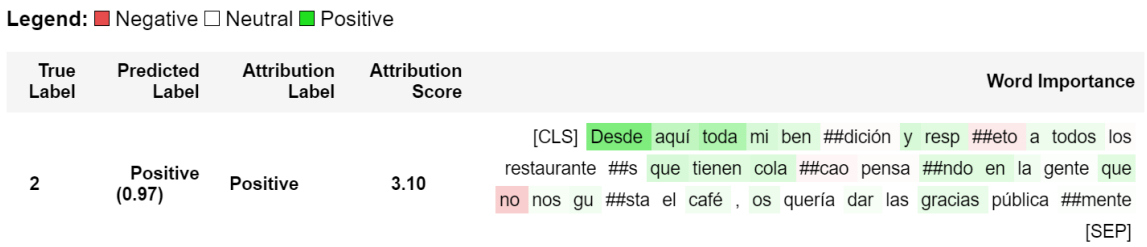

Figure 5.8: Where does the model focus? Transformer attension on a tweet by @IbaiLlanos

5.4.1.1 Datasets

As a first experiment, we have trained on an Amazon product reviews dataset [32] with 200,000, 5,000, and 5,000 reviews in the training, development, and test sets respectively. Each record in the dataset contains the review text, the review title, and the star rating, among other information.

We considered reviews with 5 or 4 stars to be positive and those with 1 or 2 stars to be negative, leaving the 3 stars' reviews as neutral. As the star rating was uniformly distributed in the dataset, we ended up with 80K, 2K, and 2K reviews per positive and negative classes and 40K, 1K, and 1K reviews for the neutral class in the training, development, and test sets respectively.

As a second experiment, we have finetuned the model using three different datasets:: the Amazon review dataset [32], Spanish Movie Reviews from Muchocine website [38], and InterTASS Spanish Twitter Sentiment corpus [18].

The Muchocine reviews dataset contains 3,872 long-form movie reviews in the Spanish language, each with a shorter summary review, and a rating on a 1-5 scale. We processed the labels of the Spanish Movie Reviews dataset the same way we did with the Amazon

[8]https://huggingface.co/Recognai/distilbert-base-es-multilingual-cased

reviews. Finally, we split the dataset into train, test, and development sets of 2478, 774, and 620 examples, respectively.

The International TASS Corpus (InterTASS) is a corpus released in TASS 2017 for the polarity classification at tweet level task. The final collection has 3,413 tweets separated into 1839 training examples, 324 development examples, and 870 test examples. The dataset is annotated with the labels positive (P), negative (N), or neutral (NEU).

5.4.1.2 Evaluation

The metrics we have used to evaluate the performance are Recall (Rc), Precision (Pr), Accuracy (Ac), and F1 Score (i.e., F1).

- Precision. Measures the percentage of true results among the total number of positive predictions.
$$Precision = \frac{TP}{TP + FP}$$
- Accuracy. Proportion of correct predictions among the total number of cases examined.
$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$
- Recall. Measures the percentage of actual positives that is correctly classified
$$Recall = \frac{TP}{TP + FN}$$
- F1 score. Harmonic mean of precision and recall. It will be low if either precision or recall are low.
$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Once we have defined the metrics we can evaluate the performance of each model.

We performed the finetuning process for 3 epochs using a batch size of 16 for the first experiments, and for 3 epochs and the same batch size number for the second experiment. Then, we compared our model with other finetuned models for sentiment analysis in Spanish. These baselines have shown good performance in their English version. In particular:

- beto-sentiment-analysis [58]: Model trained TASS 2020 corpus[9] (around 5k tweets) using the Spanish BERT, BETO [10] as the base model. Uses POS, NEG, NEU labels.
- bert-base-multilingual-uncased-sentiment [45]: This is a model finetuned for sentiment analysis on product reviews in six languages: English, Dutch, German, French, Spanish and Italian. It predicts the sentiment of the review as a number of stars (between 1 and 5).

[9]Dataset: `http://tass.sepln.org/2020/?page_id=74`

- Twitter-xlm-roberta-base-sentiment (XLM-T) [7]: This is a XLM-roBERTa-base model trained on 198M tweets and finetuned for sentiment analysis. The sentiment finetuning was done in 8 languages (Ar, En, Fr, De, Hi, It, Sp, Pt).

Additionally, we also compared the predictions of our model with a commercial solution, MeaningCloud. MeaningCloud offers a professional sentiment analysis service that can be accessed via a web API [10]. We extract the sentiment estimations for all examples using this service.

The evaluation of the first experiment has been made on the three datasets: the Amazon review dataset (only the test set of 5000 examples since we used this dataset for model training), all examples from Spanish Movie Reviews from the Muchocine website, and all examples from InterTASS Spanish Twitter Sentiment corpus. The second experiment has been evaluated using only the test set of each dataset.

Table 5.6 shows the results obtained in the first experiment for each sentiment predictor in each dataset. We observe how our model performs best on Amazon and Muchocine datasets, although it does not perform well on the former. However, those models finetuned on Twitter datasets overperform our model. This fact highlights the importance of finetuning a model on a dataset similar to the downstream task where is going to be evaluated. Both MeaningCloud and M-BERT got the worst results, a step behind the rest of the models.

	Amazon				**Muchocine**				**Twitter**				
	acc	**f1**	**recall**	**precision**	**acc**	**f1**	**recall**	**precision**	**acc**	**f1**	**recall**	**precision**	**Weighted**
M-BERT	0.627	0.559	0.627	0.583	0.395	0.280	0.395	0.365	0.471	0.375	0.471	0.313	0.405
XLM-T	0.713	0.682	0.713	0.671	0.460	0.425	0.460	0.439	0.773	0.770	0.773	0.771	0.626
BETO	0.696	0.703	0.696	0.712	0.472	0.452	0.472	0.454	**0.815**	**0.816**	**0.815**	**0.817**	**0.657**
MeaningCloud	0.642	0.627	0.642	0.635	0.425	0.423	0.425	0.444	0.557	0.547	0.557	0.558	0.532
DistilBERT (Ours)	**0.793**	**0.789**	**0.793**	**0.786**	**0.477**	**0.466**	**0.477**	**0.475**	0.495	0.471	0.495	0.472	0.576

Table 5.6: Sentiment analysis comparison of our model finetuned on Amazon review dataset. Higher is better (bolded)

Table 5.7 shows the results obtained in the second experiment for each sentiment predictor in each dataset. Base DistilBERT is the model we used in the first experiment, finetuned only on the Amazon dataset. The rest of the models are finetuned on the three datasets and are named by the order the model was finetuned on these datasets. For example, *amt* stands for the model finetuned first on the Amazon dataset, secondly on the Muchocine dataset, and finally, on the Twitter dataset. Shuffled is the model where all datasets were concate-

[10] https://www.meaningcloud.com/products/sentiment-analysis

	Amazon				Muchocine				Twitter				
	acc	f1	recall	precision	acc	f1	recall	precision	acc	f1	recall	precision	Weighted
M-BERT	0.627	0.559	0.627	0.583	0.373	0.258	0.373	0.333	0.452	0.360	0.452	0.300	0.393
XLM-T	0.713	0.682	0.713	0.671	0.444	0.409	0.444	0.428	0.697	0.694	0.697	0.694	0.595
BETO	0.696	0.703	0.696	0.712	0.452	0.434	0.452	0.441	**0.817**	**0.817**	**0.817**	**0.818**	**0.651**
MeaningCloud	0.642	0.627	0.642	0.635	0.405	0.399	0.405	0.424	0.585	0.576	0.585	0.588	0.534
Base DistilBERT (ours)	0.793	**0.789**	0.793	0.786	0.471	0.460	0.471	0.475	0.488	0.464	0.488	0.466	0.571
shuffled (ours)	0.795	0.783	0.795	0.778	0.508	0.494	0.508	0.512	0.530	0.520	0.530	0.519	0.599
amt (ours)	0.790	0.782	0.790	0.778	0.507	0.490	0.507	0.506	0.537	0.537	0.537	0.539	0.603
atm (ours)	0.791	0.784	0.791	0.781	0.506	0.479	0.506	0.511	0.551	0.546	0.551	0.545	0.603
mat (ours)	0.787	0.787	0.787	**0.788**	0.499	0.479	0.499	**0.543**	0.532	0.528	0.532	0.536	0.598
mta (ours)	0.795	0.786	0.795	0.782	0.494	0.490	0.494	0.488	0.532	0.531	0.532	0.531	0.602
tam (ours)	**0.797**	**0.789**	**0.797**	0.784	**0.525**	**0.519**	**0.525**	0.524	0.537	0.539	0.537	0.543	0.616
tma (ours)	0.796	0.788	0.796	0.784	0.514	0.495	0.514	0.510	0.526	0.521	0.526	0.527	0.601

Table 5.7: Sentiment analysis comparison of our model finetuned on the three datasets. Higher is better (bolded)

nated and randomly shuffled. Those models trained on the three datasets achieve higher performance on Muchocine and Twitter's datasets as could be expected and on the Amazon dataset although in this case is a slight increase. On Muchocine and Twitter datasets we observe an enhancement of nearly 5 with

5.4.2 Emotion analysis module

The development of this module led to the participation of the Intelligent Systems Group (GSI) at Universidad Politécnica de Madrid (UPM) in the Emotion Analysis competition EmoEvalEs, part of the IberLEF 2021 Conference. The addressed challenge proposes an emotion classification task of Spanish tweets, categorizing each message into seven emotions. We propose the design and development of a fine-tuned language model based on XLM-RoBERTa to tackle this challenge. We have obtained excellent results with this approach, obtaining first place in the competition with a macro-averaged F1 score of 71.70%.

5.4.2.1 Dataset

The dataset for the EmoEvalEs task consists of Spanish tweets related to events that occurred in April 2019 related to different domains: entertainment, catastrophe, political, global commemoration, and global strike. The task presents an emotion classification challenge in the form of a multilabel classification task, where the emotions considered are

anger, *disgust*, *fear*, *joy*, *sadness*, *surprise*, and *others*. The dataset is provided by the competition organizers divided into three different sets: training, development, and test sets. The latter's labels are provided only after the end of the competition. The training set has 5723 examples, the development set 844, and the test set 1656 examples. The distribution of the emotion labels is shown in Figure 5.9, where we can observe is highly unbalanced.

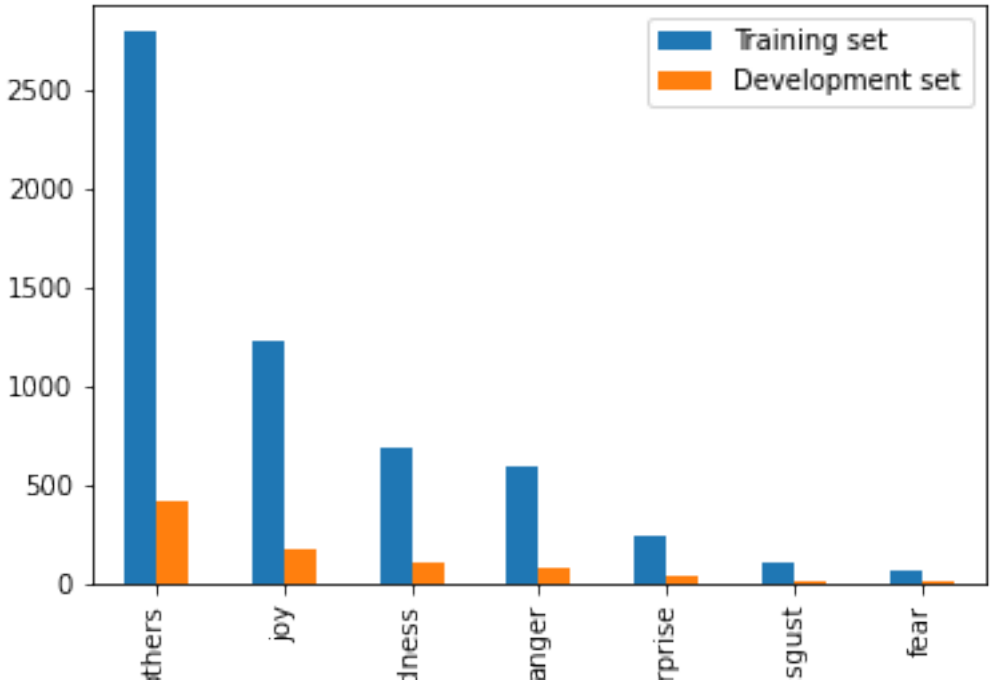

Figure 5.9: EmoEvalEs dataset

5.4.2.2 Fine-tuning XLM-T

The language model XLM-RoBERTa [12] stands out as a model pre-trained in 100 different languages, achieving state-of-the-art performance on cross-lingual classification, sequence labeling, and question answering. The lack of pre-trained language models in languages different than English has geared researchers' interest towards multilingual models that have demonstrated that it is possible to have a single large model for all languages without sacrificing too much performance for each language. However, previous research shows that multilingual models tend to underperform monolingual models in language-specific tasks [60]. This context framed the pre-trained language model we have used for this work, XLM-T [7], a version of XLM-R that achieves better results in the Twitter domain than its XLM-R baseline since it has been pre-trained on millions of tweets in over 30 different

languages.

We have fine-tuned the Twitter-specific pre-trained language model in the downstream task of emotion classification following parameter-efficient transfer learning techniques [29].

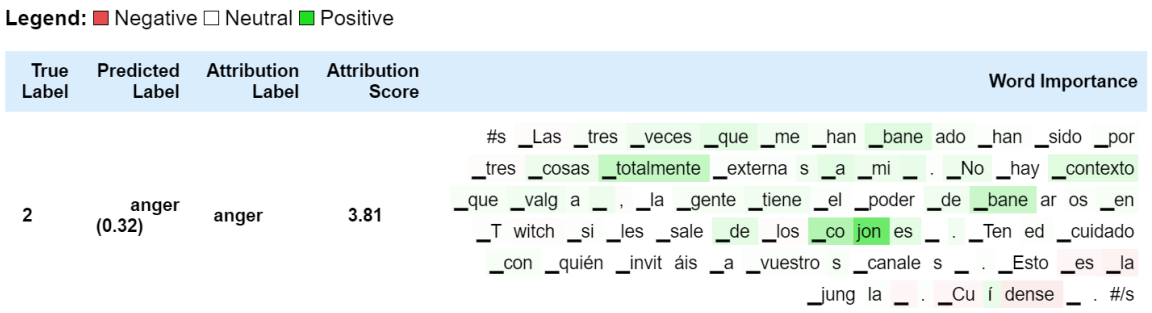

Figure 5.10: Where does the model focus? Transformer attention on a tweet by @IbaiLlanos. Green highlighted words contributes positively towards the predicted class

We have implemented this architecture and run the training process using the modules and the Trainer API from the HuggingFace Transformer library [73], which is optimized and provides a wide range of training options and built-in features. We have tested three different approaches to solve the problem:

- **Multi-label classification problem**: We have trained the model to predict the class with higher probability among the seven possibilities.
- **Binary classification problem**: Frame the multilabel problem as a one-vs-all problem where seven different models are trained. To solve ties, the output from the model with the highest confidence score is selected.
- **Additional Features**: We extend the classification head to use the additional features, *event* and *offensive*, available in the dataset as new inputs encoded as one-hot vectors.

The classification head consists of a dense layer at the output of the language model, followed by a dropout layer with the default dropout probability of the language model, and a final projection layer with the number of labels. For the Additional Features model, we have added new inputs to the first dense layer.

We have made available our code for reproducibility at `https://github.com/gsi-upm/emoevales-iberlef2021`. Also, the finetuned model is available at the HuggingFace model hub as *daveni/twitter-xlm-roberta-emotion-es*.

The hyper-parameters used are the batch size of 16 per GPU, max length (tokenizer) of 200, and training for 5 epochs. The rest of the parameters are the default in the Trainer

	Accuracy	Weighted F1 score
XLM-T Multi-label classification	73.10	71.10
XLM-T Binary one-vs-all classification	72.39	70.01
XLM-T with Additional Features	71.80	69.89

Table 5.8: Evaluation on dev test of the finetuned models

	Accuracy	Weighted F1 score
XLM-T Multi-label classification	72.77	71.70
XLM-T Binary one-vs-all classification	71.43	68.93
XLM-T with Additional Features	71.67	69.66

Table 5.9: Evaluation on test test of the finetuned models

API. The trainer API also saves the checkpoint of the best epoch that usually occurs at epoch 3. We run the training process for 1 hour on two 2 Nvidia GeForce RTX 2080 Ti 12GB GPUs.

Before tokenizing, we have slightly preprocessed the tweets with the Twitter preprocessing module of the GSITK library [6]. We have found this helpful to achieve slightly better results.

5.4.2.3 XLM-T evaluation

This section describes the performance of the different approaches we have followed during the finetuning of the pre-trained model. Table 5.8 shows the accuracy and weighted F1 scores of the different approaches on the development set, where the model finetuned in the multilabel classification problem has achieved the best results. Table 5.9 shows the same information for the test set, where the best model is the multilabel estimator again.

These results show the superior performance of the multilabel classifier over the combination of various binary classifiers, although better combination strategies could improve the results of the latter. Moreover, including additional features without any preprocessing and at the same level as the output produced by the pre-trained language model decreases the classifier performance.

Figure 5.11 depicts the confusion matrix produced by the XLM-T Multi-label classifier

on the test set. We observe the evident unbalancing of the dataset, where almost half of the records belong to the *others* class. Moreover, this class is usually confounded with the *joy* class. Additionally, this matrix shows the difficulty of distinguishing between emotions that share similar features, such as *anger* and *disgust*. Finally, the low number of records in some classes (*anger*, *disgust*, and *surprise*) is an additional challenge since the models tend to fail in those classes.

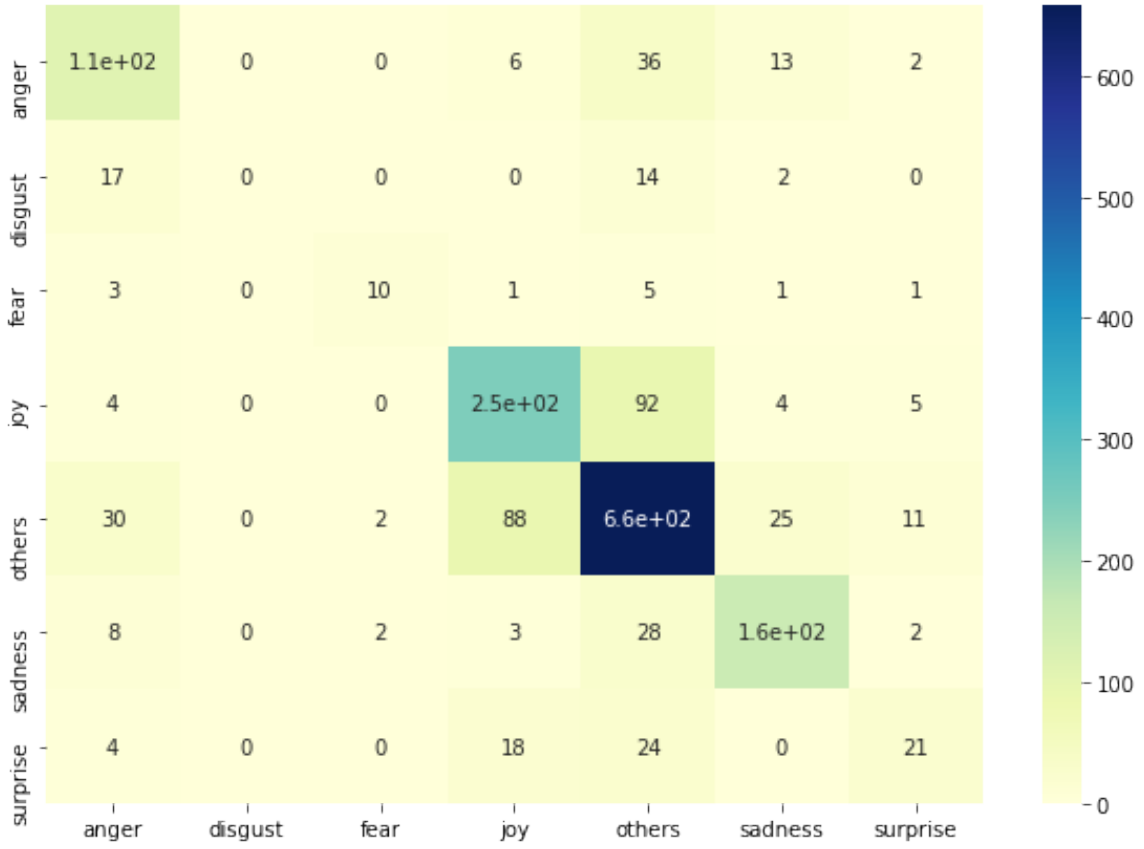

Figure 5.11: Confusion matrix on test set produced by XLM-T Multi-label classifier

5.4.2.4 Conclusions

This work led us to publish the paper ***GSI-UPM at IberLEF2021: Emotion Analysis of Spanish Tweets by Fine-tuning the XLM-RoBERTa Language Model***, where we describe our participation in the EmoEvalEs competition framed in the IberLef 2021 Conference. Our proposal relies on the use of large pre-trained language models, outperforming previous methods with little effort using the HuggingFace library, which provides an easy implementation of these pre-trained language models. The pre-trained model we have used is a RoBERTa transformer trained on a multilingual corpus of tweets, XLM-T. We have evaluated different strategies to approach the problem, finding that the finetuned model for a multilabel classification task obtains better results than the combination of various binary classifiers and the model with additional features. We have achieved the first position in the EmoEvalEs competition with this model, obtaining a macro-averaged F1 score of 71.70% on the test set.

5.5 LIWC module

This module has been built as a plugin for Senpy using the LIWC dictionary (see Section 4.3). This way, we are able to provide the use of the dictionary as a service with linked data features, a web interface, and a common API. We make it easier to use the dictionary through an API call to the Senpy framework and compute different linguistic properties from the variables of the dictionary. Selected variables[11] are shown in Table 5.10. The module computes the features extracted from the LIWC dictionary for the different linguistic variables present in the dictionary.

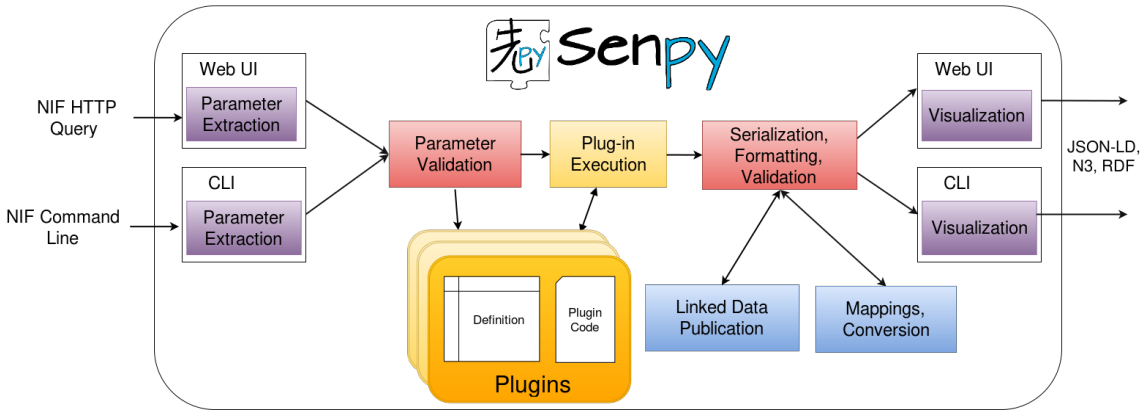

Figure 5.12: Senpy architecture

Summary Variable	Affect Words	Social Words	Time Orientation	Cognitive Processes	Informal Speech	Personal Concerns
Analytical Thinking	Positive emotion	Family	Past focus	Insight	Swear words	Work
Clout	Negative emotion	Friends	Present focus	Cause	Netspeak	Leisure
Authentic		Female referents	Future focus	Discrepancies	Assent	Home
Emotional Tone		Male referents		Tentativeness	Nonfluencies	Money
				Certainty	Fillers	Religion
				Differentiation	Differentiation	Death

Table 5.10: Selected LIWC dictionary variables

[11] More information about the variables at: https://liwc.wpengine.com/interpreting-liwc-output/

CHAPTER 6

Case studies

In this chapter, we are going to describe two different use cases reflecting how sentiment and emotion attributes of text influence a specific outcome. We will use the text generated by users of the social network Twitter towards ride-hailing companies and users of the hospitality company Airbnb to find out to what extent it affects different aspects of their operations. We use our system to perform sentiment and emotion analysis and compute the Average Treatment Effect (ATE) of these attributes.

6.1 Twitter interactions of ride-hailing companies

Several works have explored the role of social media on customer engagement and relationships [21, 79], and more specifically, the role of the social network Twitter [8, 42, 46]. Nowadays, social networks are a new channel for customer service and interaction with potential clients, so they have become a tool of paramount importance for the marketing goals of every company. Hence, new ways of obtaining insightful information on this new channel are extremely useful to optimize marketing campaigns and enhance the companies reputation.

To this end, being able to answer causal questions such as *How much impact does humorous posts have on social media interactions?* or *Does the writing style affect the number of responses? To what extent?* may help to improve a company's reputation or boost engagement with its customers. In this use case, we will explore the question ***How does it affect the sentiment or emotion of a message on the reply time?***. Our first hypothesis is that negative or angry messages, probably related to complaints, get a quicker reply from the official Twitter accounts. The answer to this question attempts to uncover any possible bias of the community managers towards messages which display a specific sentiment or emotion; or the effectiveness of ticket priority systems implemented on the social media domain. We will focus on the ride-hailing companies *Cabify*, *Uber*, and *Lyft*; both in English and Spanish language using the causal inference system we have implemented to compute the effect of sentiment or emotion features on the reply time of the official accounts.

6.1.1 Dataset

This section describes the dataset gathering process. We used Twint, a library that provides several search options, to capture tweets matching our searching criteria. To create the tweets dataset, the tweets had to meet the following criteria:

- The tweets language must be Spanish (Spanish dataset) or English (English dataset)
- They have been dated between January 1st, 2011 and April 23rd, 2021
- They can not be re-tweets (a re-post of a tweet originally posted by a different user). Therefore we focus only on original tweets.
- They have to be directed to one of the ride-hailing companies' accounts or written by one of them (see Table 6.1). In short, we listed all Twitter accounts engaging with the customers of the ride-hailing companies in the Spanish and English Twitter community.

Dataset	Target Twitter usernames
Spanish	@cabify_espana, @cabify, @Cabify_Chile, @cabify_arg, @cabify_ecuador, @cabify_colombia, @Cabify_Mexico, @Cabify_Peru, @cabify_uruguay, @Uber_ES, @Uber_MEX, @Uber_ARG, @Uber_CR
English	@uber_support, @asklyft, @lyft, @uber, @uber_canada, @uberuk, @uber_nz

Table 6.1: Target usernames to create dataset

The collected sample dataset matching this criterium contained a total number of 865,487 English tweets and 722,669 Spanish tweets. The distribution of tweets from the different Twitter accounts is shown in Figure 6.1 and Figure 6.2 for the Spanish and English datasets, respectively. Is worth mentioning that tweets from other users directed to the official accounts are not described in those figures. As we observe, the predominant username in the Spanish dataset is *uber_mex* and *uber_support* in the English dataset.

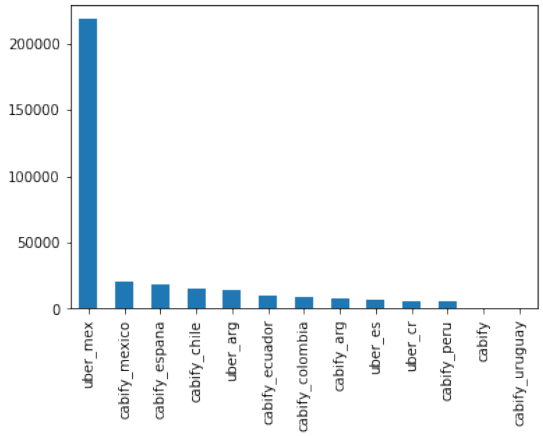

Figure 6.1: Spanish dataset

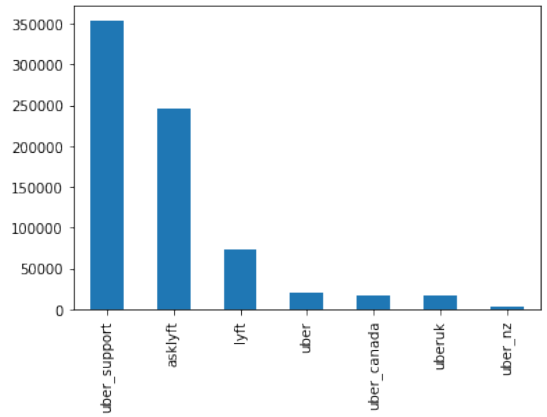

Figure 6.2: English dataset

6.1.2 Processing

This section describes how we processed the datasets to produce the covariates needed to answer our question. The tweets, collected in JSON format, are parsed to extract the attributes we are interested in and then grouped by $conversation_id$ to generate conversations between the companies and the customers.

Figure 6.3: Processing steps

The processing has been made as follows:

1. **Preprocessing**. The raw dataset is preprocessed to drop duplicates and select the columns of interest.
2. **Conversations**. Tweets in the dataset are grouped by conversation id, generating dialogues between the companies and other users. Those conversations and filtered to get only those where:
   (a) A company account is replying to another user.
   (b) There are only two different usernames in the conversation (the company and another user)
3. **Reply time**. The time took for the company account to reply is computed. Additionally, each conversation is tagged with the company account involved.
4. **Costumer tweets selection**. Finally, we filter out the tweets from official accounts since we are only interested in the reply time to customers from the official accounts.

Through this process, we obtained 106,377 tweets from conversations in the Spanish dataset, and 14,635 from the English dataset. Since the size of both datasets was similar, this difference suggests that a wiser preprocessing could produce a larger English dataset with additional information. Each dataset is composed of the tweets from users (costumers), the ride-hailing company involved, and the time it took the company to reply. Only tweets with 5 or more words were used.

6.1.3 Framework

This section describes how we used our system to answer the target causal question: **How does it affect the sentiment or emotion of a message on the reply time?** Thus, we

define the desired outcome Y of study as a late reply if the computed time is greater than a defined threshold. That is, if the reply time to a message t is greater than a threshold th, then the outcome is $Y = 1$.

For the proxy treatment $\hat{T}$ we use the sentiment and emotion modules to perform sentiment and emotion analysis on the dataset. This allows us to compute the effect of a positive or negative message, or the effect of writing an angry, happy, or sad message. We have used negative sentiment and angry emotions as a proxy treatment for our experiment. As a confounder C we use the target company to adjust for the different companies' cultures and processes that may influence the final result. Hence, C will be a binary indicator of the target account of a message: *Cabify* or *Uber* in the Spanish dataset, and *Uber* or *Lyft* in the English dataset. Figure 6.4 shows the causal model for this experimental setting.

To adjust for other potential confounders, we match the most similar pairs with different proxy treatment $\hat{T}$. To this end, we compute the TF-IDF vectors of each document and calculate their cosine similarity matrix. Finally, we select the top 5000 most similar pairs, obtaining a dataset of 10,000 examples. Finally, the TextCause algorithm is trained for 6 epochs and a batch size of 32, using the corresponding language model for each language and averaging over 10 random seeds for robustness.

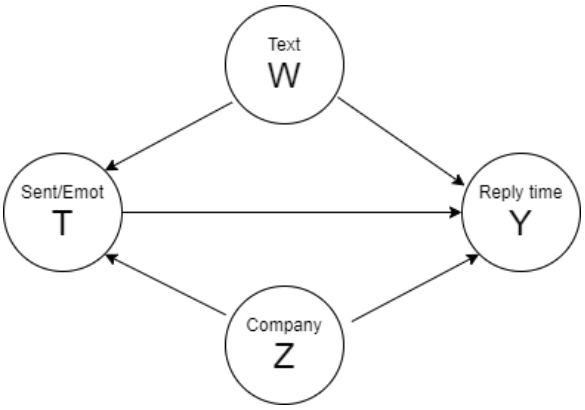

Figure 6.4: Causal model of Twitter use case

6.1.4 Results

This section describes the results obtained in this experiment. We have performed a search of different thresholds for the outcome Y. That is, we have computed the estimated ATE considering late replies, those greater than different thresholds: 5 minutes, 15 minutes, 30 minutes, 45 minutes, 60 minutes, 90 minutes, 120 minutes, 4 hours, 8 hours, 16 hours, 24 hours, and 48 hours. We have also considered different proxy treatments $\hat{T}$: if a tweet is negative or not; and if a tweet displays anger or not.

Figure 6.5 and Figure 6.6 show the results in the English dataset for both negative sentiment and angry emotion, respectively. Each bar represents the expected percent change in the likelihood of getting a late response when the message negative or angry level is hypothetically increased.

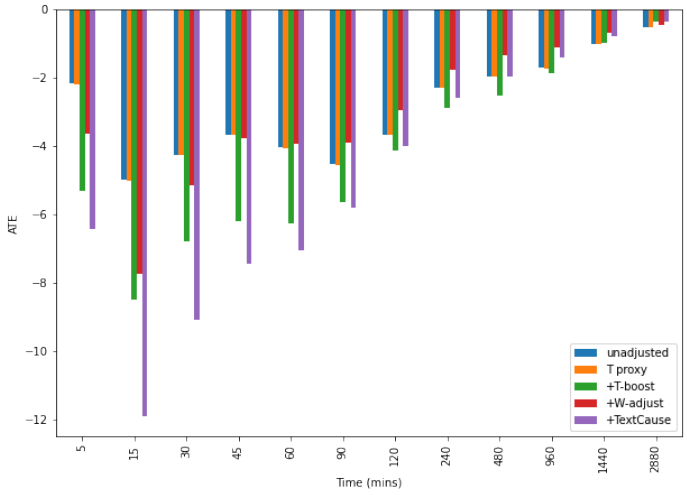

Figure 6.5: Results in the English dataset for the negative sentiment proxy treatment

The first look at these results confirms our initial hypothesis: complaints in social media get a faster response. As we can see, we obtained a negative ATE for all time thresholds for both negative and angry proxy treatments. This means that when a message has negative sentiment or displays anger, it gets a faster response from the ride-hailing official accounts. Looking closer to the results obtained with the negative sentiment, we observe that the TextCause algorithm, which adjusts for text and covariates, provides the largest effect for each time threshold, suggesting that the effect size increases as we adjust for increasing amounts of information. The "unadjusted" approach which does not perform any adjustment produces the smallest ATE. "T-proxy", which only adjusts for covariates, indicated the second-smallest ATE. This effect reaches its maximum at 15 minutes, where **a message is a 12% more likely to get a faster response**. The effect decays with time, being negligible past the 24 hours.

This same trend is observed when the angry emotion proxy treatment is used (Figure 6.6). However, this time the TextCause algorithm attenuates the ATE computed with the T-proxy and T-boost estimators which do not adjust for the text. In addition, we observe how the TextCause algorithm provides a positive effect when estimating the likelihood to get a response later than five minutes. This could be due to the small-time threshold used, which could provide very different results depending on the timeframe of the study.

These results suggest that the text contains additional information that helps to infer the true effect. Furthermore, as negative sentiment and angry emotions are related, we also observe in this case the same decaying trend as we increase the time threshold.

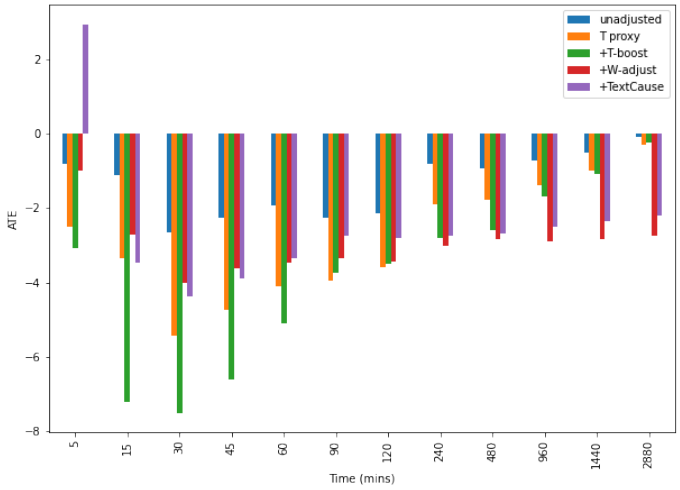

Figure 6.6: Results in the English dataset for the angry emotion proxy treatment

We carried out the same experiment for the Spanish dataset with tweets from *Cabify* and *Uber*. We used the modules we developed for sentiment and emotion analysis in Spanish. Figure 6.7 shows the results obtained with the negative sentiment proxy treatment. Different from the English dataset, we observe a positive ATE for all time thresholds. That is, you are likely to get a late response if you write an angry message. Again, we see how the effect mitigates for higher time thresholds, indicating that the sentiment is less relevant when studying its effect of very late responses.

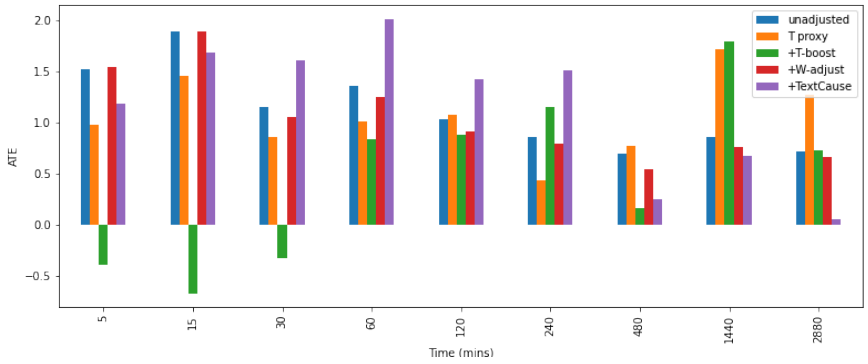

Figure 6.7: Results in the Spanish dataset for the negative sentiment proxy treatment

Figure 6.8 shows the results obtained with the angry emotion proxy treatment. In this case, the TextCause estimate is generally similar to the unadjusted ATE although it suggests a slightly higher effect of the treatment. Those estimators which do not adjust for the text (T-proxy and T-boost) indicate the highest ATE, which points out the importance of taking into account the information contained in the text to infer the effect of a variable in the output. Additionally, the TextCause estimator gives a result more similar to the obtained using the negative sentiment proxy treatment, which would be coherent since negative sentiment and angry emotion are related.

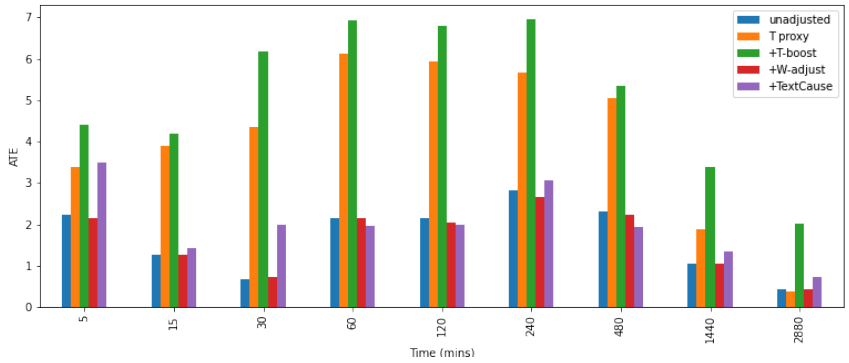

Figure 6.8: Results in the Spanish dataset for the angry emotion proxy treatment

6.1.5 Conclusions

We carried out a causal inference study using observational data from the social network Twitter to estimate the effect of sentiment and emotion on the time taken by ride-hailing companies to respond. Our results in the English dataset suggest that negative or angry messages, usually complaints, get a faster response compared to the rest of the messages. However, this trend is inverted in the Spanish dataset, showing that negative or angry messages get a late answer. This could be due to many factors. First, the accuracy of the proxies is key to the correct estimate of the treatment effect. The models we used for Spanish may lack the necessary accuracy to provide meaningful results. Secondly, we saw in Section 5.2 that the Spanish language model underperforms its English counterpart. Additionally, the different policies and processes implemented in the companies may disturb the effect estimation. Finally, this suggests that there is a significant amount of confounding in real-world studies, and the choice of the estimator can yield highly varying conclusions.

6.2 Case Study: Airbnb

Airbnb is an American company that operates an online marketplace for lodging, primarily homestays for vacation rentals, and tourism activities. Users can use the platform to book apartments for their holidays based on the location, price, description, photos, and reviews of other users, among other available information. This context provides a perfect setting to carry out new experiments using our system to find answers to new causal questions. This time we are interested in three different causal questions:

1. What is the effect of positive reviews on the rating score of the apartment?
2. Which linguistic attributes of the description affect the rating score of the apartment?
3. Do they also affect the number of reviews per month received?

Modeling the effect of different linguistic attributes from the description and reviews would help to optimize the bookings and ratings of an apartment, and would provide new information on the dynamics of the platform.

6.2.1 Dataset

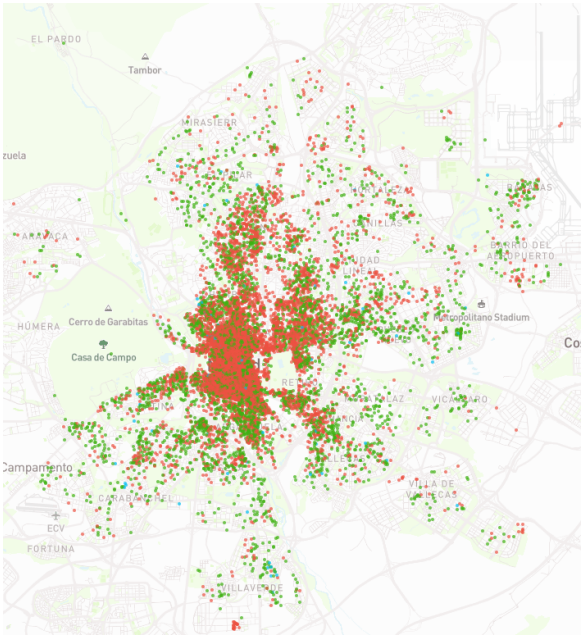

Figure 6.9: Airbnb listings in Madrid. Orange: Entire homes. Green: Rooms.

We used a dataset made freely available under a Creative Commons license by Murray Cox that is dynamically scraped from the Airbnb website[13]. This dataset contains the

listings, calendars, and reviews on the Airbnb platform for every city where they operate. We focus the scope of the study on the city of Madrid (Spain), where there are 19,618 apartments or rooms listed currently. We also collected the reviews associated with those listings, a total of 625,000 reviews. Figure 6.9 shows the distributions of the apartments in the city.

After a first exploratory data analysis, we realized there were different languages in the dataset. As we were only interested in the examples in English or Spanish, we used a language detection library [43] to select only those examples with a description and reviews in the desired languages. As it misclassified some examples with non-ASCII characters (i.e., Japanese, Chinese, Korean) we also used regular expressions to filter out these examples. Additionally, we excluded examples with no description or zero reviews. Finally, we obtained the datasets described in Table 6.2

Dataset	**#Examples**
Spanish descriptions	3,174
English description	3,507
Spanish reviews	215,932
English reviews	240,654

Table 6.2: Airbnb datasets

6.2.2 Framework

This section describes how we used our system to answer the target causal questions. We model each question with the causal models of Figure 6.10. For the first and second questions, we defined the outcome Y as the rating score of the apartment associated with the description or review. Hence, the outcome Y is a binary indicator of whether the rating is 100 (maximum) or not.

For the proxy treatment $\hat{T}$ we use the sentiment and emotion modules to perform sentiment and emotion analysis on the reviews; and the LIWC module to extract other linguistic attributes from the description. We have used positive sentiment and joy emotion as a proxy treatment for our experiment on the reviews dataset. For the experiments involving the descriptions of the listings, we used different variables discussed in the results section. As a confounder C we use a binary indicator of whether the price is higher than the percentile 75% or not, which is around 100€ both for the Spanish and English datasets.

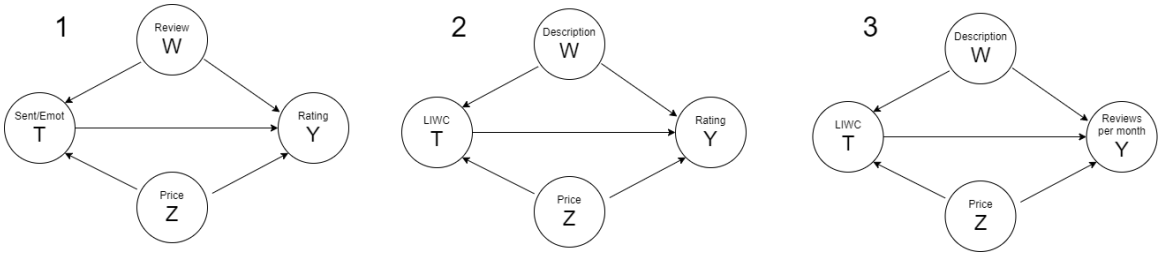

Figure 6.10: Causal models for the Airbnb study

To adjust for other potential confounders, we match the most similar pairs with different proxy treatment $\hat{T}$ for the reviews experiment. To this end, we compute the TF-IDF vectors of each document and calculate their cosine similarity matrix. Finally, we select the top 5000 most similar pairs, obtaining a dataset of 10,000 examples. Finally, the TextCause algorithm is trained for 4 epochs and a batch size of 32, using the corresponding language model for each language and averaging over 10 random seeds for robustness.

6.2.3 Results

This section describes the results obtained in the Airbnb study. First, we want to answer the question *What is the effect of positive reviews on the rating score of the apartment?*. Figure 6.11 and Figure 6.12 show the results obtained in the reviews experiments both for Spanish and English, respectively, for the positive sentiment (left) and joy emotion (right) proxy treatments. We can see how in both cases the TextCause estimate (purple) attenuates the unadjusted ATE (blue) and gives a lower ATE estimate than the rest of the estimators. This suggests that there is important confound information in the text, and not adjusting for it gives an exaggerated ATE. These results point at there is a positive causal relationship between positive reviews and the rating score, as could be expected beforehand.

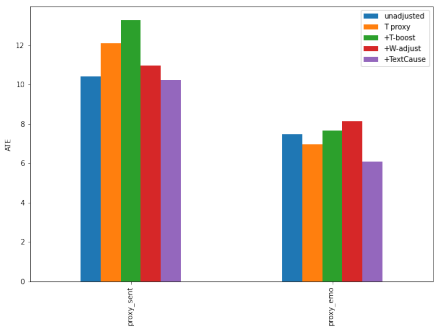

Figure 6.11: Results on Spanish dataset: reviews effect on score

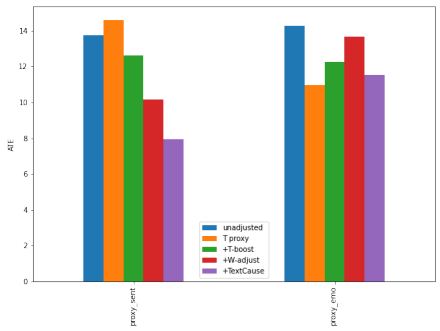

Figure 6.12: Results on English dataset: reviews effect on score

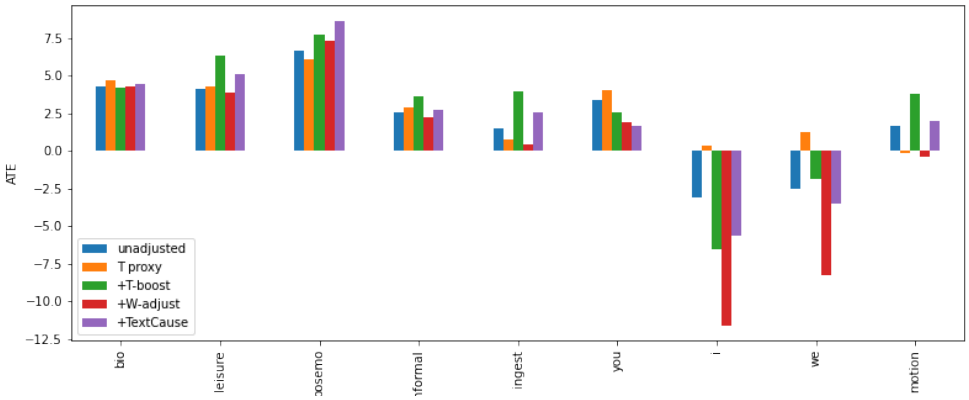

Figure 6.13: Effect of LIWC fatures on rating score

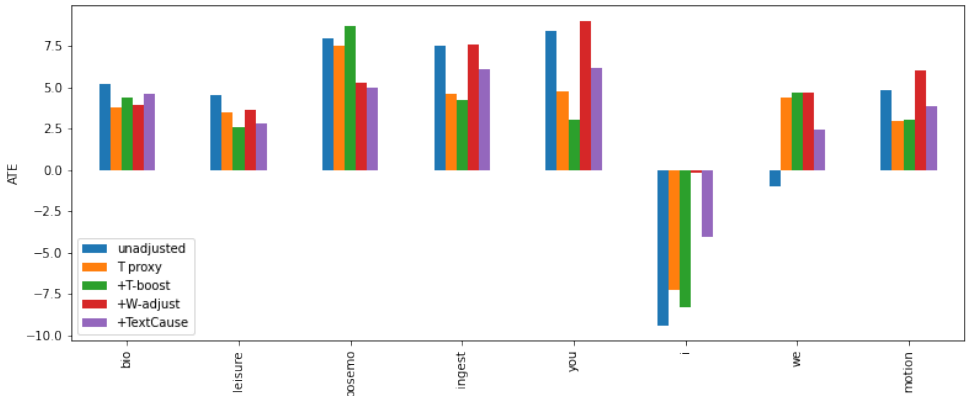

Figure 6.14: Effect of LIWC features on reviews per month

Secondly, we want to answer the question *Which linguistic attributes of the description affect the rating score of the apartment?*. Figure 6.13 displays the ATE estimates of different linguistic features computed using the LIWC module. Generally, the TextCause estimate increases the ATE estimate over the unadjusted estimator. These results indicate that giving information about health-related information such as cleanliness (*bio* feature), leisure activities, culinary information (*ingest* feature), or using and informal language and the second person (*you* feature) makes more likely to get the maximum rating score; while giving information in the first person in singular or plural (*I, we* features) may reduce the effect.

Finally, we want to know if these linguistic features also affect the number of reviews per month received. Figure 6.14 shows the answer to this question. All features except the first person (*I* feature) increases the likelihood to get more reviews each month, which could

be directly associated with the number of guests an apartment hosts each month. Thus, these results suggest that the more information provided in the description about activities (*leisure* or *ingest* features), or positive sentiment (*posemo* feature), the more reviews a listing could get.

6.2.4 Conclusions

We carried out this second study using observational data from Airbnb to estimate the effect of different linguistic attributes on the rating score and the reviews per month (this could be used as a proxy for the number of guests per month). We used our system to compute the ATE estimate of those examples with a description and reviews in English or Spanish. Our findings support the hypothesis that the more information and positive sentiment in the description, the higher the score. Also, we confirmed something that could be supposed beforehand: more positive reviews affect positively the rating of a listing. Finally, the differences in the estimates of the different methods suggest that there is a significant amount of confounding in real-world studies, as the rating could be influenced by other confounder variables such as the neighborhood, the size of the apartment, or the facilities provided.

CHAPTER 7

Conclusions

This chapter will describe the achieved goals done by the master thesis following some of the key points developed in the project.

7.1 Conclusion

In this project, we have developed a system to infer the causal effects of linguistic properties. In addition, we have evaluated it on different use cases to demonstrate the applicability of the system. The project has fulfilled the proposed objectives, but we must refer to each one of these objectives to draw a general conclusion. First, we have been introduced to the causal inference field along with its intersection with Natural Language Processing. Secondly, we have replicated the state-of-the-art work that serves as a baseline for this project and we have evaluated it in new datasets, extended it to allow the use of languages different than English, and it has been modularized to be used as the core module of our system. Then, we have developed a system based on different modules to extract linguistic attributes from the text: emotion, sentiment, and other features extracted from the LIWC dictionary. The emotion and sentiment modules have been developed using current state-of-the-art technologies based on the transformer architecture. Finally, we have carried out two different causal inference studies based on observational text data.

We would like to highlight that the development of the model used for the emotion analysis module led us to participate in the EmoEvalEs competition, framed in the IberLef 2021 Conference. With this model, **have achieved the first position in the competition**, which allowed us to describe our model in the paper ***GSI-UPM at IberLEF2021: Emotion Analysis of Spanish Tweets by Fine-tuning the XLM-RoBERTa Language Model***, which will be published in the proceedings of the conference.

In the first case study, we have computed the causal effect of the sentiment and emotion of messages directed to ride-hailing companies on Twitter on the reply time. We have observed how messages with negative sentiment or angry emotions get a faster response in English, although we obtained the inverse effect in Spanish. Then, in the second case study, we analyze the effect of several variables on our outcome of interest. First, we checked something that, although we could have expected it beforehand, we confirmed as true: positive reviews positively influence the score received by an apartment. Then, we calculated the effect of different linguistic features on the score, finding that features related to additional information about cleanliness, leisure activities, or gastronomy affect positively, while those related to the tenant (I, we) affect negatively. Finally, we have studied the effect of these features on the reviews per month (which could be used as a proxy for the number of monthly guests of an apartment). As in the previous case, more information related to these characteristics affects positively. However, this time the feature "we" has a positive influence, which could indicate that there are more guests in the apartments that refer to several hosts (a couple or family) rather than a single individual.

However, this study is subject to several limitations. First of all, the difficulty to take into

account all possible confounders in real-world studies can yield highly varying conclusions when we compute the causal effects using observational data. In addition, current works in the intersection of causal inference and Natural Language Processing lack the complexity of the real world, making it challenging to obtain valid conclusions from studies with multiple causal relations.

Overall, we think this is an innovative work that demonstrates the utility and cost-effectiveness of existing big data sources not produced by this research. It is a very promising tool to evaluate the causal effects of linguistic properties in a variety of settings where text is present and further work should be carried out in this line.

7.2 Future work

This section describes the possible new features or improvements that could be done in this project.

- **Model compatibility**. Modify the TextCause module to leverage the last advances of the transformer library and to make possible the use of different language models not based on DistilBERT.
- **Extension to higher dimensional outcomes and confounders**. Extend the causal effect computation to new scenarios where confounders and/or outcomes are not encoded as binary indicators.
- **Benchmark datasets**. Develop new datasets based on paired randomized controlled trials and observational studies to be able to fairly compare different causal inference models.
- **Emotion and sentiment datasets**. Create comprehensive benchmarks for sentiment and emotion analysis in Spanish, since currently there are no reference benchmarks as the SST-Sentiment analysis benchhmark [65].

APPENDIX A

Project impact

This appendix reflects, quantitatively or qualitatively, the possible social, economic and environmental impact jointly with ethical implications.

A.1 Social impact

This section describes the main social impacts our work might have. As a consequence of the rise of social media in recent years, a huge amount of public-available data is daily shared by people. Projects like this one are able to analyze and exploit this data for different purposes.

In our case, the study and the tool developed will allow researchers to carry out multipurpose causal inference studies involving textual data much more easily benefiting from public available big data. Also, this system could be considered as a starting point for further work on the subject.

Studies using our system would be carried much faster, giving useful insights to estimate the effect of different treatments on business metrics. This would be very useful to apply policies that may benefit the organizations taking data-based decisions.

A.2 Economic impact

In this section, we summarize the main economic impacts of our project. The possibility of carrying out studies to determine the effect of different policies on a desired business metric using using observational data instead of randomized controlled trials directly implies less duration and cost in capital and resources. Our system allows to study this effects in unstructured datasets where text is predominat and, thus, opens the range of causal inference studies that can be performed.

A.3 Environmental impact

This section briefly describes the main environmental impacts of our project.

All systems based on machine learning and big data have an important ecological footprint due to the energy needed for, firstly, producing the huge amount of data these systems need and, secondly, training the machine learning models, which may last several weeks. Fortunately, we leverage on pre-trained models using transfer learning techniques and avoiding to train from scratch each language model, which is a really computationally expensive task.

In particular, we have to mention the energy consumption of the computer and big data cluster needed for this project. Each study carried out using this system takes an average of two days of computing using one GPU, but it depends on the amount of data used in the studied so this time could be increased even to weeks, incrementing accordingly the power consumption.

A.4 Ethical implications

Finally, this section depicts the main ethical considerations.

The main ethical issue that may concern us is related to privacy. Although the project has been developed using publicly available data, most of the population is not concerned about what really implies share their data publicly in social networks like Twitter. However, people have agreed on the use of their data for different purposes, including projects like this. We conclude that more education about the value of privacy is needed to be able to not consider this issue.

APPENDIX B

Project budget

This appendix details an adequate budget to bring about the project. The project structure is described along with the activities undertaken to complete it and costs are evaluated including material and human resources, as well as taxes.

B.1 Project structure

In order of achieving the proposed goals, the project have been divided in the activities shown in Table B.1, where details about its duration and dependencies are provided. All activities require a person with good programming and machine learning background, a telecommunication engineer for example. As the project has been carried by only one person, effort of each task is not considered since it is directly related to the duration of the activity. **The total duration of the project has been 900 hours approximately**, taking into account that one day corresponds to 4 working hours.

Activity	**Description**	**Dependencies with other tasks**	**Duration (days*)**
1.- Causal Inference course	The fundaments of Causal Inference and its relationship with Machine Learning	-	20
2.- State of art research	Research about works related to the goal of our project	-	15
3.- HuggingFace Library	Get to know how HuggingFace transformers library works and create the first fine-tuned models.	-	25
4.- Data capture & Preprocessing	Capture the datasets needed to evaluate SOTA models on new settings	2,3	15
5.- SOTA evaluation	Implement and evaluate SOTA models on new settings.	2,3,4	30
6.- Sentiment analysis module	Implement the sentiment analysis module	5	10
7.- Emotion analysis module	Implement the emotion analysis module	5,6	15
8.- LIWC module	Implement the LIWC module	5,6,7	7
9.- EmoEvalEs competition	Participation in the EmoEValEs competition, improving our models and writing the paper for the conference proceedings.	7	25
10.- Twitter Case Study	Use of the developed system on Twitter use case	5,6,7,8	20
11.- Airbnb Case Study	Use of the developed system on Airbnb case	5,6,7,8	10
12.- Report writing	Writing of the TFM	2,10,11	30

Table B.1: Project structure division by activity.

1

B.2 Costs evaluation

This section summarizes the material resources needed to develop the project, divided in software and hardware resources, as well as the human resources and taxes associated to it.

B.2.1 Material resources

B.2.1.1 Software

All the project has been developed using open-source software that is available on Internet for free. For this reason, there is no cost associated to software.

B.2.1.2 Hardware

To carry out this project a computer and a big data cluster, both provided by the Intelligent Systems Group, has been used.

The specifications of the computer are:

- Intel Core i5 CPU of 3.2GHzx4
- 8 GB of RAM
- Hard disk of 500 GB

On the other hand, the big data cluster specifications are:

- DELL PowerEdge R320
- Intel® Xeon® E5-2430 v2
- 4x32GB RDIMM
- 3x3TB, SATA
- 2x12GB NVIDIA GeForce RTX 2080 Ti GPU

A computer with this specifications costs 700€ approximately, while the approximate cost of the big data cluster is 3600€. Thus, **the hardware cost is 4300€**. Although it should have been taken into account, amortization has not been computed so the final hardware cost should be lower or even 0€ if we consider the hardware have already been amortized.

B.2.2 Human resources

We have needed one telecommunication engineer to accomplish this project. The salary have been based on the Cabify University-Industry Chair scholarship, through which this project has been carried out, stipulated in 550€/month for 80 working hours (20 working hours per week). Thus, the fee per hour would be 6,875€. Since the project duration has been estimated in 900 hours, the total fee for the project ascend to 6187,5€. We have multiplied this number by 1.3 to approximately include Social Security, thus, **the total cost of human resources is around 8045€**.

B.2.3 Taxes

In case the final product is sold to an interested company, taxes related to a software engineering project must be taken into account. The fees paid by the company would be the corresponding VAT established in the local country

B.3 Conclusion

The project had a **duration of 900** hours and the **total cost ascends to 12350€**.

Bibliography

[1] Building layered, multilingual sentiment lexicons at synset and lemma levels. *Expert Systems with Applications*, 41(13):5984–5994, October 2014. Publisher: Pergamon.

[2] Jay Alammar. The illustrated bert, elmo, and co. (how nlp cracked transfer learning) – jay alammar – visualizing machine learning one concept at a time. `http://jalammar.github.io/illustrated-bert/`. (Accessed on 06/23/2021).

[3] Jay Alammar. The illustrated transformer – jay alammar – visualizing machine learning one concept at a time. `https://jalammar.github.io/illustrated-transformer/`. (Accessed on 06/23/2021).

[4] Jay Alammar. The illustrated word2vec – jay alammar – visualizing machine learning one concept at a time. `https://jalammar.github.io/illustrated-word2vec/`. (Accessed on 06/23/2021).

[5] Jay Alammar. Transformers, explained: Understand the model behind gpt-3, bert, and t5. `https://daleonai.com/transformers-explained`. (Accessed on 06/23/2021).

[6] Oscar Araque, Ganggao Zhu, and Carlos A. Iglesias. A semantic similarity-based perspective of affect lexicons for sentiment analysis. *Knowledge-Based Systems*, 165:346 – 359, 2019.

[7] Francesco Barbieri, Luis Espinosa Anke, and José Camacho-Collados. XLM-T: A multilingual language model toolkit for twitter. *CoRR*, abs/2104.12250, 2021.

[8] Ana Isabel Canhoto and Moira Clark. Customer service 140 characters at a time: The users' perspective. *Journal of Marketing Management*, 29(5-6):522–544, April 2013. Publisher: Routledge _eprint: https://doi.org/10.1080/0267257X.2013.777355.

[9] José Cañete. Compilation of large spanish unannotated corpora, May 2019.

[10] José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*, 2020.

[11] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [cs, stat]*, September 2014. arXiv: 1406.1078.

[12] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual*

Meeting of the Association for Computational Linguistics, pages 8440–8451, Online, July 2020. Association for Computational Linguistics.

[13] Murray Cox. Get the data - inside airbnb. adding data to the debate. `http://insideairbnb.com/get-the-data.html`. (Accessed on 06/22/2021).

[14] Fermín L. Cruz, José A. Troyano, Beatriz Pontes, and F. Javier Ortega. Building layered, multilingual sentiment lexicons at synset and lemma levels. *Expert Systems with Applications*, 41(13):5984–5994, 2014.

[15] Fermín L. Cruz, José A. Troyano, Beatriz Pontes, and F. Javier Ortega. Building layered, multilingual sentiment lexicons at synset and lemma levels. *Expert Systems with Applications*, 41(13):5984–5994, October 2014.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. arXiv: 1810.04805.

[18] Díaz-Galiano. The democratization of deep learning in tass 2017 — díaz-galiano — procesamiento del lenguaje natural. `http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/5556`. (Accessed on 06/16/2021).

[19] Naoki Egami, Christian J. Fong, Justin Grimmer, Margaret E. Roberts, and Brandon M. Stewart. How to Make Causal Inferences Using Texts. *arXiv:1802.02163 [cs, stat]*, February 2018. arXiv: 1802.02163 version: 1.

[20] Oguzhan Gencoglu and Mathias Gruber. Causal modeling of twitter activity during COVID-19. *CoRR*, abs/2005.07952, 2020.

[21] Yujuan Guo, Di Fan, and Xiao Zhang. Social media–based customer service and firm reputation. *International Journal of Operations Production Management*, ahead-of-print, 05 2020.

[22] Yujuan Guo, Di Fan, and Xiao Zhang. Social media–based customer service and firm reputation. *International Journal of Operations and Production Management*, 40(5):575–601, January 2020. Publisher: Emerald Publishing Limited.

[23] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[24] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith,

Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[25] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. Number: 7825 Publisher: Nature Publishing Group.

[26] Zellig S. Harris. Distributional structure. *¡i¿WORD¡/i¿*, 10(2-3):146–162, 1954.

[27] Zellig S. Harris. Distributional Structure. *WORD*, 10(2-3):146–162, August 1954. Publisher: Routledge _eprint: https://doi.org/10.1080/00437956.1954.11659520.

[28] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[29] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. *CoRR*, abs/1902.00751, 2019.

[30] HuggingFace. transformers/examples/research_projects/distillation at master · huggingface/-transformers. `https://github.com/huggingface/transformers/tree/master/examples/research_projects/distillation`. (Accessed on 06/15/2021).

[31] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[32] Phillip Keung, Yichao Lu, György Szarvas, and Noah A. Smith. The multilingual amazon reviews corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

[33] Carolyn Kim and Osbert Bastani. Learning Interpretable Models with Causal Guarantees. *arXiv:1901.08576 [cs, stat]*, January 2019. arXiv: 1901.08576.

[34] Stony Brook Data Science Lab. Multilingualsentiment - data science lab. `https://sites.google.com/site/datascienceslab/projects/multilingualsentiment`. (Accessed on 06/23/2021).

[35] Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-Based Neural DAG Learning. April 2020.

[36] Zachary C. Lipton. The Mythos of Model Interpretability. *arXiv:1606.03490 [cs, stat]*, March 2017. arXiv: 1606.03490.

[37] Yunan Luo, Jian Peng, and Jianzhu Ma. When causal inference meets deep learning. *Nature Machine Intelligence*, 2(8):426–427, August 2020.

[38] Fermín Cruz Mata. Datasets - fermín cruz mata homepage. `http://www.lsi.us.es/~fermin/index.php/Datasets`. (Accessed on 06/16/2021).

[39] Chris McCormick. Bert word embeddings tutorial · chris mccormick. `https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/`. (Accessed on 06/23/2021).

[40] Robert McHardy, Heike Adel, and Roman Klinger. Adversarial Training for Satire Detection: Controlling for Confounding Variables. 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 660–665, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[41] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, September 2013. arXiv: 1301.3781.

[42] Fotis Misopoulos, Miljana Mitic, Alexandros Kapoulas, and Christos Karapiperis. Uncovering customer service experiences with Twitter: the case of airline industry. *Management Decision*, 52(4):705–723, January 2014. Publisher: Emerald Group Publishing Limited.

[43] Shuyo Nakatani. Language detection library for java, 2010.

[44] Brady Neal. Introduction to causal inference. `https://www.bradyneal.com/causal-inference-course`. (Accessed on 06/23/2021).

[45] NLPtown. nlptown/bert-base-multilingual-uncased-sentiment · hugging face. `https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment`. (Accessed on 06/17/2021).

[46] Shereen Oraby, Mansurul Bhuiyan, Pritam Gundecha, Jalal Mahmud, and Rama Akkiraju. Modeling and computational characterization of twitter customer service conversations. *ACM Trans. Interact. Intell. Syst.*, 9(2–3), March 2019.

[47] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[49] Michael J. Paul. Feature Selection as Causal Inference: Experiments with Text Classification. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 163–172, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[50] Umashanthi Pavalanathan and Jacob Eisenstein. Emoticons vs. Emojis on Twitter: A Causal Inference Approach. *arXiv:1510.08480 [cs]*, October 2015. arXiv: 1510.08480.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[52] James Pennebaker, Roger Booth, Ryan Boyd, and Martha Francis. Linguistic inquiry and word count: Liwc2015, 09 2015.

[53] Reid Pryzant. Text Feature Selection for Causal Inference.

[54] Reid Pryzant, Sugato Basu, and Kazoo Sone. Interpretable Neural Architectures for Attributing an Ad's Performance to its Writing Style. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 125–135, Brussels, Belgium, 2018. Association for Computational Linguistics.

[55] Reid Pryzant, Dallas Card, Dan Jurafsky, Victor Veitch, and Dhanya Sridhar. Causal Effects of Linguistic Properties. *arXiv:2010.12919 [cs]*, October 2020. arXiv: 2010.12919.

[56] Reid Pryzant, Young-joo Chung, and Dan Jurafsky. Predicting Sales from the Language of Product Descriptions. page 10.

[57] Reid Pryzant, Kelly Shen, Dan Jurafsky, and Stefan Wager. Deconfounded Lexicon Induction for Interpretable Social Science. page 11.

[58] Juan Manuel Pérez. finiteautomata/beto-sentiment-analysis · hugging face. `https://huggingface.co/finiteautomata/beto-sentiment-analysis`. (Accessed on 06/17/2021).

[59] Manuel Romero. mrm8488/electricidad-small-finetuned-restaurant-sentiment-analysis · hugging face. `https://huggingface.co/mrm8488/electricidad-small-finetuned-restaurant-sentiment-analysis`. (Accessed on 06/17/2021).

[60] Phillip Rust, Jonas Pfeiffer, Ivan Vulic, Sebastian Ruder, and Iryna Gurevych. How good is your tokenizer? on the monolingual performance of multilingual language models. *CoRR*, abs/2012.15613, 2020.

[61] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[62] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv:1910.01108 [cs]*, February 2020. arXiv: 1910.01108.

[63] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Towards causal representation learning. *CoRR*, abs/2102.11107, 2021.

[64] Claudia Shi, David M. Blei, and Victor Veitch. Adapting Neural Networks for the Estimation of Treatment Effects. *arXiv:1906.02120 [cs, stat]*, October 2019. arXiv: 1906.02120.

[65] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[66] J. Fernando Sánchez-Rada, Carlos A. Iglesias, Ignacio Corcuera, and Óscar Araque. Senpy: A pragmatic linked sentiment analysis framework. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 735–742, 2016.

[67] OSINT Team. twintproject/twint: An advanced twitter scraping & osint tool written in python that doesn't use twitter's api, allowing you to scrape a user's followers, following, tweets and more while evading most api limitations. `https://github.com/twintproject/twint`. (Accessed on 06/23/2021).

[68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[69] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv:1706.03762 [cs]*, December 2017. arXiv: 1706.03762.

[70] Victor Veitch, Dhanya Sridhar, and David Blei. Adapting Text Embeddings for Causal Inference. In *Conference on Uncertainty in Artificial Intelligence*, pages 919–928. PMLR, August 2020. ISSN: 2640-3498.

[71] Victor Veitch, Dhanya Sridhar, and David M. Blei. Adapting Text Embeddings for Causal Inference. *arXiv:1905.12741 [cs, stat]*, July 2020. arXiv: 1905.12741.

[72] Victor Veitch, Yixin Wang, and David M. Blei. Using Embeddings to Correct for Unobserved Confounding in Networks. *arXiv:1902.04114 [cs, stat]*, May 2019. arXiv: 1902.04114.

[73] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[74] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv:1910.03771 [cs]*, July 2020. arXiv: 1910.03771.

[75] Zach Wood-Doughty, Ilya Shpitser, and Mark Dredze. Challenges of Using Text Classifiers for Causal Inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4586–4598, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[76] Diyi Yang, Jiaao Chen, Zichao Yang, Dan Jurafsky, and Eduard Hovy. Let's Make Your Request More Persuasive: Modeling Persuasive Strategies via Semi-Supervised Neural Nets on Crowdfunding Platforms. In *Proceedings of the 2019 Conference of the North*, pages 3620–3630, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[77] Michael Yeomans, Alejandro Kantor, and Dustin Tingley. The politeness Package: Detecting Politeness in Natural Language. *The R Journal*, 10(2):489, 2019.

[78] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. DAG-GNN: DAG Structure Learning with Graph Neural Networks. In *International Conference on Machine Learning*, pages 7154–7163. PMLR, May 2019. ISSN: 2640-3498.

[79] Mingli Zhang, Lingyun Guo, Mu Hu, and Wenhua Liu. Influence of customer engagement with company social networks on stickiness: Mediating effect of customer value creation. *International Journal of Information Management*, 37(3):229–240, June 2017.

[80] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9472–9483. Curran Associates, Inc., 2018.

[81] Xun Zheng, Chen Dan, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. Learning Sparse Nonparametric DAGs. *arXiv:1909.13189 [cs, stat]*, March 2020. arXiv: 1909.13189.