Democratic and Popular Republic of Algeria
الجمهورية الجزائرية الديمقراطية الشعبية
Ministry of Higher Education and Scientific Research
وزارة التعليم العالي و البحث العلمي

ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE

المدرسة الوطنية العليا للإعلام الآلي
(المعهد الوطني للتكوين في الإعلام الآلي سابقا)
École nationale Supérieure d'Informatique
ex. INI (Institut National de formation en Informatique)

Graduation Thesis

For the attainment of the State Engineer's diploma in Computer Science

Option : Intelligent Systems and Data (SID)

Detection and dissemination of problematic content such as propaganda on social media

Authors :
Mr. MAHMAHI Anis
Mr. HATTABI Ilyes

Supervisors :
Ms. OANA Goga (LIX)
Ms. OANA Balalau (Inria)
Ms. DAKICHE Narimene (ESI)

Promotion : 2023/2024

Dedication

“

It is with great pleasure that I dedicate this work

To my parents, who have always been by my side and supported me throughout my long years of study.

To all my family,To all my friends,

To all those who are dear to me, to all of you

Thank you all.

”

- Ilyes

“

I dedicate this work

To my dear parents, and my family, who have filled my journey with support and inspiration, without you, I would never be the person I am today.

To my friends, to you who have woven the threads of a strong and sincere friendship.

To all those who are dear to me, to all of you.

”

- Anis

Aknowledgements

First of all, I thank Allah, the Almighty, for giving us the courage and patience needed to complete this work.

We would like to extend our special thanks to our supervisor, **Dr. Narimene Dakiche**, for the competent assistance she provided, for her patience and encouragement. Her critical eye has been invaluable in structuring the work and improving the quality of the various sections.

We would also like to thank our mentor, **Dr. Oana Goga** and **Dr. Oana Balalau**, for their immense help, the quality of her supervision, and for all the advice and information she provided us with an unparalleled degree of patience and professionalism.

We also thank **Pr. Ioana Manolescu, Mr. Salim Chouaki** for their valuable help, their encouragement, and for making our internship a very enriching experience.

We take this opportunity to express our sincere gratitude to the teaching and administrative staff of the École nationale Supérieure d'Informatique for their ongoing commitment to high-level education.

We wish to express our deepest gratitude to the members of the jury for graciously dedicating their precious time to evaluating this work.

We are deeply grateful to our parents and family members for their constant support throughout our journey. Their boundless love, trust, and unwavering encouragement.

Finally, we would like to thank everyone who contributed, directly or indirectly, to the completion of this work.

Abstract

In an era where information flows freely and rapidly, the rise of propaganda and disinformation has become a significant concern, especially on social media platforms. These platforms, such as YouTube, are often utilized to spread misleading information **that is not necessarily false but does the way it is communicated** can shape public perception and influence decision-making processes. The unchecked dissemination of such content is particularly problematic in politically charged environments, where propaganda is strategically used to manipulate opinions and control narratives. The pervasive nature of these techniques necessitates effective tools for detecting and analyzing the spread of propaganda.

The challenge addressed in this work lies in developing a robust machine learning model capable of identifying and classifying propaganda within political texts. Our approach aims to not only detect the presence of propaganda but also to pinpoint the specific techniques used to influence audiences. A critical aspect of this study involves the collection and analysis of a new dataset focused on U.S. Senate elections and other political events, offering a fresh perspective on the dissemination of propaganda during election periods.

Our contribution includes the design and implementation of a machine learning model tailored to classify propaganda in political texts and gives where propaganda appeared in that text and which technique was employed. We also provide an extensive analysis of the model's predictions across various political datasets. This analysis uncovers key insights into the prevalence of different propaganda techniques and their distribution across platforms.

Keywords : Propaganda, Disinformation, Social media, Machine learning, dataset collection, Classification.

Résumé

Dans une époque où l'information circule librement et rapidement, l'essor de la propagande et de la désinformation est devenu une préoccupation majeure, notamment sur les plateformes de médias sociaux. Ces plateformes, telles que YouTube, sont souvent utilisées pour diffuser des informations trompeuses **qui ne sont pas nécessairement fausses, mais la manière dont elles sont communiquées** peut influencer la perception publique et les processus décisionnels. La diffusion incontrôlée de ce type de contenu est particulièrement problématique dans des environnements politiquement chargés, où la propagande est stratégiquement utilisée pour manipuler les opinions et contrôler les récits. La nature omniprésente de ces techniques nécessite des outils efficaces pour détecter et analyser la propagation de la propagande.

Le défi abordé dans ce travail réside dans le développement d'un modèle d'apprentissage automatique robuste capable d'identifier et de classifier la propagande dans les textes politiques. Notre approche vise non seulement à détecter la présence de propagande, mais aussi à identifier les techniques spécifiques utilisées pour influencer les audiences. Un aspect critique de cette étude implique aussi la collecte et l'analyse d'un nouveau jeu de données axé sur les élections sénatoriales américaines et d'autres événements politiques, offrant une perspective nouvelle sur la diffusion de la propagande pendant les périodes électorales.

Nous fournissons également une analyse approfondie des prédictions du modèle à travers divers ensembles de données politiques. Cette analyse révèle des informations clés sur la prévalence des différentes techniques de propagande et leur distribution à travers les plateformes.

—

Mots clés : Propagande, Désinformation, Réseaux sociaux, Apprentissage automatique, Collecte de données, Classification.

ملخص

في عصر تتدفق فيه المعلومات بحرية وسرعة، أصبح صعود الدعاية والمعلومات المضللة مصدر قلق كبير، خاصة على منصات وسائل التواصل الاجتماعي. تُستخدم هذه المنصات، مثل يوتيوب، في كثير من الأحيان لنشر معلومات مضللة **التي ليست بالضرورة زائفة، ولكن الطريقة التي يتم بها توصيلها** يمكن أن تؤثر على التصور العام وتؤثر في عمليات اتخاذ القرار. تعتبر عملية نشر هذا النوع من المحتوى دون رقابة مشكلة كبيرة بشكل خاص في بيئات سياسية مشحونة، حيث تُستخدم الدعاية استراتيجيًا لتوجيه الآراء والتحكم في السرد. تتطلب الطبيعة الشاملة لهذه التقنيات أدوات فعالة لاكتشاف وتحليل انتشار الدعاية.

التحدي الذي يتناول هذا العمل هو تطوير نموذج تعلم آلي قادر على تحديد وتصنيف الدعاية داخل النصوص السياسية. تهدف طريقتنا إلى اكتشاف وجود الدعاية فحسب، بل أيضًا إلى تحديد التقنيات المحددة المستخدمة للتأثير على الجمهور. يتضمن جانب حاسم من هذه الدراسة جمع وتحليل مجموعة بيانات جديدة تركز على الانتخابات الأمريكية لمجلس الشيوخ وغيرها من الأحداث السياسية، مما يوفر منظورًا جديدًا حول انتشار الدعاية خلال فترات الانتخابات.

تتضمن مساهمتنا تصميم وتنفيذ نموذج تعلم آلي مخصص لتصنيف الدعاية في النصوص السياسية، مع تحديد أماكن ظهور الدعاية في النص والتقنيات المستخدمة. كما نقدم تحليلًا شاملاً لتنبؤات النموذج عبر مجموعات بيانات سياسية متنوعة. يكشف هذا التحليل عن رؤى رئيسية حول انتشار تقنيات الدعاية المختلفة وتوزيعها عبر المنصات.

كلمات مفتاحية :

الدعاية، التضليل، وسائل التواصل الاجتماعي، تعلم الآلة، جمع البيانات، التصنيف.

Contents

List of Figures

List of Tables

Abbreviations

BERT	*Bidirectional Encoder Representations from Transformers*
DeBERTa	*Decoding-enhanced BERT with Disentangled Attention*
RoBERTa	*Robustly Optimized BERT Pretraining Approach*
LSTM	*Long Short-Term Memory*
LLM	*Large Language Model*
API	*Application Programming Interface*
HQP	*Base Transceiver Station*
SLC	*Sentence-Level Classification*
FLC	*Fragment Level Classification*
CoT	*Chain of Thoughts*
CRF	*Conditional Random Fields*
NLL	*Negative Log Likelihood*
QCRI	*Qatar Computing Research Institute*
LCCS	*Longest Common Consecutive Sequence of words*
NLTK	*Natural Language Toolkit*
MIML	*Multi-instance Multi-label learning*
MLM	*Masked Language Modeling*

Introduction

Problematic

Humans are exposed daily to millions of pieces of information, often without verifying their accuracy, which can significantly impact their subconscious. Moreover, not everyone checks the authenticity of the information they receive before accepting it as fact, influencing their future decisions. This impact can spread to a larger scale since false information tends to reach more people and spreads more rapidly. This is particularly easy nowadays through existing communities on social networks like YouTube. This is problematic because in some countries such as united state sponsored propaganda is utilized to control the flow of information and shape the public's perception of reality using multiple technique from spreading fear and doubt into exaggerate or minimize a situation.

This internship aims to propose a machine learning method to detect and study the dissemination of propaganda and disinformation on social media and more specifically on political news.

Objectives

The first objective is to build a machine learning model capable of classifying political texts to determine whether they contain propaganda and, if so, identify the specific propaganda technique used. The internship description initially mentioned that this model would be run on a dataset of 7.5 million collected Facebook posts. However, due to the numerous elections happening in 2024, our focus has shifted to applying our model to a different dataset centered on election-related content.

This shift introduces us to the second objective: collecting this new dataset. We have chosen YouTube as our target platform, compiling a dataset of video transcriptions related to US Senate elections and other US political events.

The third objective involves performing data analysis on the predictions generated by our model. We will apply the model to both a set of 58,000 Facebook posts (including both sponsored and organic content) and our newly collected YouTube video dataset. The analysis aims to address several questions, such as: Which propaganda technique is most commonly used? Which YouTube channel spreads the most propaganda content? ..ect by achieving these objectives, we aim to gain insights into the prevalence and nature of propaganda in political texts across different platforms.

Report's Plan

To achieve our objectives, we organized our report into several chapters. The first chapter, **"Literature Review"**, where we define propaganda, outlines the criteria for identifying a text as propaganda, and describes some common propaganda techniques. and introduces various machine learning models that process text data and their learning approaches. The second chapter, **"Existing Work"**, surveys recent research and methods developed to detect propaganda texts and techniques, including a comparative study of these approaches. The third chapter, **"Propaganda Classification Model"**, details our first contribution: a comprehensive study of decoder-based large language models, including improvements on previous work by Sprenkamp et al. [2023] and a discussion of its limitations. This chapter also presents our second contribution—our own method for detecting propaganda and its techniques at the token level—along with a thorough series of tests. The fourth chapter, **"Political Data Collection and Analysis"**, discusses the political dataset of U.S. election news that we collected. and presents an analysis of the propaganda found in our collected dataset (U.S. election news) and another dataset comprising sponsored vs. non-sponsored (organic) posts. The final chapter, **"Conclusion"**, summarizes the report and suggests directions for future research.

Part I

Bibliographic Review

Chapter 1

Literature review

1.1 Propaganda and disinformation

1.1.1 Introduction

The way people are consuming information is changing. Traditionally, users had to actively search for online content and had an active role in selecting what they saw and read. This started to change with the popularization of social media when content started to come from friends passively, and worries about filter bubbles started. Today, we arrived at the other end of the spectrum, where online platforms and advertisers push content to users through recommendations and targeted advertising. This is problematic because users have little control over what content gets pushed on their screens, and there is little oversight over what content platforms and advertisers promote and to whom. Disinformation has been recognized by the European Commission as a principal threat to our democracies and encouraged work on commenting on it (Code of Practice on Disinformation) [1].

More recently [Edelson et al., 2021], propaganda has been utilized in the context of political elections, where misinformation and disinformation campaigns can sway voters and undermine democratic processes. In countries such as North Korea and China, state-sponsored propaganda is utilized to control the flow of information and shape the public's perception of reality.

1.1.2 Basic Concepts and Definitions

In this section, we begin by defining key concepts that establish the context of our work. These concepts include the definitions of propaganda, disinformation, and misinformation while highlighting the differences between them.

[1] https://disinfocode.eu

Propaganda

According to the paper written by Balalau and Horincar [2021], propaganda, derived from the Latin term meaning "things that must be disseminated," involves information designed to persuade an audience to embrace a particular idea or cause through specific tactics or by bringing out strong emotions. To further clarify this concept, we examine the work of Martino et al. [2020], who highlight two fundamental aspects of propaganda: (i) the use of information to sway opinions, and (ii) doing so intentionally.

Propaganda Techniques

According to Balalau and Horincar [2021], while there is a general agreement on the definition of propaganda among researchers, but the full list of propagandist techniques remains debated, with Wikipedia listing 68 different techniques.

Some argue that propaganda is a communication method that does not depend on the document's topic and its topic-specific vocabulary. Representations based on writing style, readability, and stylistic features might generalize better than those based solely on word-level representations. Based on this notion, we can compile a list of eighteen (18) propaganda techniques found in research articles that can be evaluated intrinsically, without the need to reference external information. The complete list is provided in the paper by Balalau and Horincar [2021]; however, for this report, we will highlight only a few.

1. **Appeal to fear or prejudice** : this technique seeks to garner support for an idea by inciting anxiety, worry, or fear among individuals about another option, often relying on existing biases.

   - Example 1: "Either we adopt this policy immediately, or our economy will collapse"
   - Example 2: "If we don't bail out the big automakers, the US economy will collapse. Therefore, we need to bail out the automakers."

2. **Black and white fallacy** : This involves giving out two choices as the only possibilities, ignoring other potential alternatives. In extreme cases, it directs the audience to take specific actions, ruling out any other choices.

   - Example 1: Either we ban all cars to stop pollution, or we do nothing and let the planet die

   - Example 2: If you're not with us, you're against us.
   Explanation: This statement presents only two extreme options (supporting or opposing a particular group) and ignores the possibility of neutral or alternative positions.

3. **Causal oversimplification** : An oversimplification fallacy involves attributing a single cause/reason to an issue when, in reality, there are many other contributing factors. This can involve placing blame on one individual or group without considering the complexities involved.

   - Example 1: "President Trump has been in office for a month and gas prices have been skyrocketing. The rise in gas prices is because of President Trump."

- Example 2: "The unemployment rate is high because immigrants are taking all the jobs."

4. **Reductio ad Hitlerum** : it is a rhetorical tactic that attempts to convince the listeners to reject an action or idea by linking it with groups or individuals that are hated by the listeners.

   - Example 1: "You want to implement a national healthcare system? That's exactly what Hitler did in Nazi Germany"

   - Example 2: "Do you know who used to do this? Hitler."

5. **Appeal to authority** : This fallacy states a belief or a claim is true just because an authority or expert made it, without providing additional supporting evidence. A special case of this technique occurs when the referenced person is not an authority or expert in the relevant field, known as an irrelevant authority in literature.

   - Example: This diet plan must be effective because the runner Usain Bolt recommends it. Therefore, it's true. Explanation: While Usain Bolt is a very well-known athlete and can confidently assert the diet's validity, what truly supports the effectiveness of this diet is the overwhelming evidence, not just his authority.

6. **Whataboutism** : This technique attempts to make the opponent seems false or unreliable accusing them of hypocrisy instead of directly addressing their argument.

   - Example 1: How can you condemn our government for corruption when your politicians have been caught in scandals too?

   - Example 2: "Why are you criticizing our country's human rights record? Look at how poorly your own country treats its minority groups"

Many of these methods are fallacies since propagandists often use arguments that appear convincing but lack validity. A fallacy occurs when the evidence fails to support the claim being made. Additionally, these techniques often rely on emotional language or employ disinformation tactics to present an idea. Disinformation, another term that requires further explanation, refers to intentionally false or misleading information that is spread deliberately to deceive.

Disinformation

After reviewing the study conducted by Alam et al. [2022], it becomes apparent that the term 'fake news' is frequently utilized, though in a rather general manner, which tends to steer focus solely towards accuracy. Consequently, international bodies such as the United Nations, World Health Organization, European Union, and NATO exhibit a preference for the term 'disinformation.' This term encompasses information that is both falsified and intentionally disseminated to deceive and inflict harm upon others. The latter aspect, the potential for harm, often goes overlooked but is of equal significance

Misinformation

In a similar survey, we encountered another term related to the topic of false information, known as misinformation. This term refers to the dissemination of false content, but unlike disinformation, it does not carry the intention to cause harm. Misinformation can arise from various sources, including honest mistakes, misinterpretations, or the dissemination of outdated or incomplete information. It is particularly prevalent in the digital age, where information can be shared rapidly across social media platforms, often without thorough verification. In the context of political events, misinformation can influence public opinion, voter behavior, and election outcomes. False information about candidates, voting procedures, or election results can undermine confidence in the electoral process and lead to unrest or violence.

Difference between Disinformation and Misinformation

The distinction between these two concepts is explained by First Draft [Ireton and Posetti, 2018], which provides definitions for each. Misinformation is characterized as "errors such as incorrect photo captions, dates, statistics, translations, or situations where satire is misunderstood," while disinformation is defined as "made-up or intentionally altered text, speech, or visual content, as well as deliberately created conspiracy theories or rumors."

1.1.3 Fact-Checking

Fact-checking has emerged as a critical tool in the fight against propaganda, misinformation, and disinformation. Fact-checking organizations, both independent and affiliated with major news outlets, work to verify the accuracy of statements made in the media, by politicians, and on social platforms. These organizations aim to provide the public with truthful information, thereby counteracting the effects of false or misleading content.

The process of fact-checking involves several steps. First, fact-checkers identify claims that are widely circulated or that have the potential to influence public opinion. These claims are then thoroughly researched using credible sources, such as official documents, expert interviews, and data from reputable organizations. Once the research is complete, the fact-checkers assess the claim's accuracy, often rating it on a scale that ranges from "true" to "false," with various degrees of partial truth in between.

Despite its importance, fact-checking faces significant challenges. One major issue is the sheer volume of information that needs to be verified. The rapid spread of information on social media means that fact-checkers are often playing catch-up, trying to debunk falsehoods after they have already reached a large audience. Additionally, the effectiveness of fact-checking is sometimes undermined by cognitive biases, where individuals continue to believe false information even after it has been debunked.

In response to these challenges, some platforms have integrated fact-checking into their systems, using algorithms and human moderators to flag potentially false information. However, the success of these efforts depends on the cooperation of users, who must be willing to engage with fact-checked content and reconsider their beliefs in light of new

information.

1.1.4 Conclusion

In conclusion, propaganda represents a significant threat to the integrity of information in the digital age. These tactics are used to manipulate public opinion, distort facts, and undermine democratic processes. While fact-checking and other countermeasures play a vital role in combating these threats, the challenges they face are substantial.

Ultimately, the fight against propaganda and disinformation requires a multifaceted approach, involving not only fact-checking but also media literacy education, regulatory frameworks, and the active participation of individuals in critically evaluating the information they consume.

1.2 Language Models

1.2.1 Introduction

Language models (LMs) are essential in NLP, designed to learn the structure and patterns of a language from extensive text data. Traditional models, such as n-gram models, use basic statistical methods to predict a word's likelihood based on its surrounding context. However, these models struggled with capturing long-range dependencies and understanding semantic subtleties. This led to the development of more advanced models like recurrent neural networks (RNNs), LSTM, and GRU, each offering improvements over their predecessors. The introduction of Transformers [Vaswani et al., 2023] marked a significant breakthrough, giving rise to powerful models like BERT, RoBERTa, and XLNet.

Initially, these models were considered large language models (LLMs) and set new benchmarks in NLP. However, they have since been outpaced by even more scalable and larger models like GPT-3.5/4, LLama, and Gemini. These advanced models show exceptional capabilities in many NLP problems and consistently achieve state-of-the-art performance, thanks to their self-supervised pre-training on big amounts of text data.

Despite their success, the application of LLMs to non-standard problems, such as propaganda detection which is a multi-label classification problem, remains relatively unexplored. Additionally, the unique potential and challenges of LLMs in addressing such complex tasks have not been thoroughly examined.

1.2.2 Transformers

Transformers, a revolutionary architecture in natural language processing, were first introduced in the seminal article "Attention is all you need" [Vaswani et al., 2023]. They offered a novel perspective on sequence data and surpassed many established state-of-the-art approaches. Unlike recurrent neural networks (RNNs), which use sequential data processing, transformers provide a more efficient way of handling the input. While RNNs

have to process the input word by word, transformers use attention which enables them to process the whole input at once. This leads to better parallelization, which is why transformers can train much faster. Moreover, RNNs were not so good at detecting dependencies between words far away from each other, while attention in transformers deals with this problem very nicely.

Architecture

This section describes the purpose of the main elements of Transformer model architecture, which can be seen in Figure 1.1.

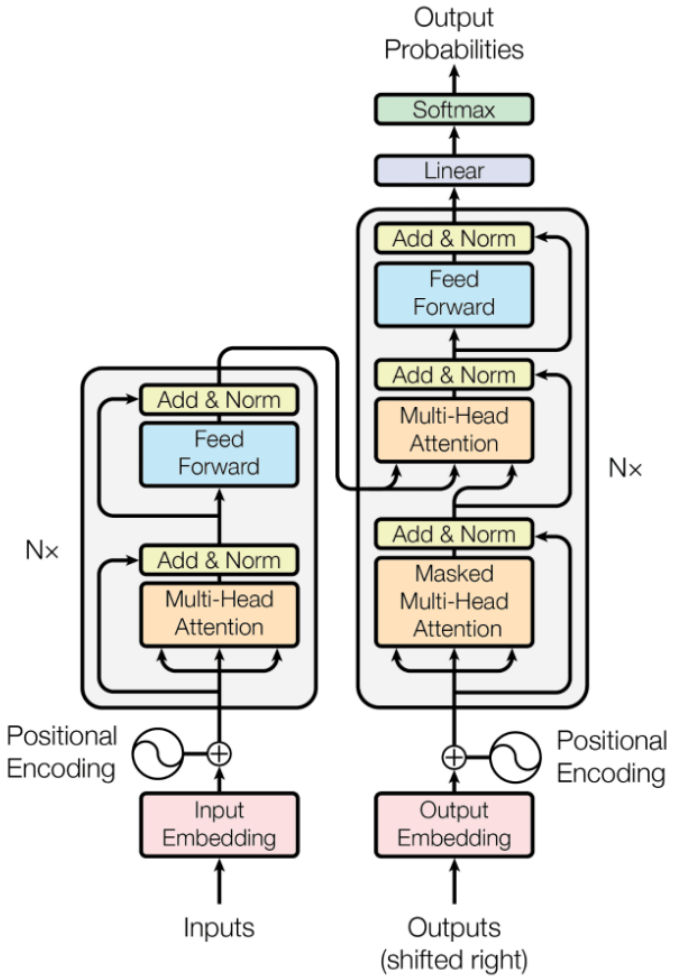


Figure 1.1: Transformers Architecture [Vaswani et al., 2023]

- **Input** data is processed through the *Embedding* and *Position Encoding* layers before being fed into both the Encoder and Decoder layers. These preprocessing layers enhance the declarative value of the data, making it more informative and useful for the Encoder and Decoder to analyze. The embedding layer converts each input word into a vector representation, known as an embedding vector. It captures the meaning of that word in a more detailed way. To ensure accurate parallel processing of inputs, the Position Encoding layer is utilized to calculate fixed values for each

token based on its position in the sequence. This enables the model to maintain an understanding of the original position of the token it is currently processing.

- **Output** component receives data from the final decoder layer and is tasked with determining the final output format. The data is first fed through a Linear layer, which generates word scores. For example, when working with the English language, which has a vocabulary of n words, the output component generates the scores for each word position in the final sentence, indicating the likelihood of a specific word from the vocabulary appearing in that position. As a result, for a sentence of five words, we would receive n score values for each of the five words. The logits generated by the Linear layer are passed to a *Softmax* layer, which converts the scores into probabilities. The final output sentence is then constructed by selecting the word with the highest probability for each position in the sentence.

- **Encoder stack** consists of several Encoder layers. The first encoder layer receives the product of the Embedding and Position Encoding layer. The other Encoder layers are connected sequentially, and each receives the data from the previous one. Each Encoder is just a Multi-head attention (MHSA) layer and a linear layer (feed-forward), followed by a normalization layer. The embedding vectors are enhanced with contextual information from the rest of the sentence through the use of MHSA. This approach utilizes multiple heads to provide the model with multiple versions of dependencies between words, thereby ensuring a more robust understanding of the context. Additional information on the concept of self-attention can be found in the following section. The output from MHSA is later passed to the position-wise Feed-Forward Network for further transformation.

- **Decoder stack** consists of several Decoder layers, all of them connected the same way as the Encoder layers. The first stage, the Masked MHSA layer, puts masks on the inputs to the multi-head attention mechanism so that attention is not applied to hidden (masked) positions. The next stage is a basic MHSA layer as described in the Encoder layer but with extra input from the Encoder. This MHSA layer computes attention between the partially masked output sentence and the input sentence. Then comes the Feed-forward layer, whose function is the same as in the Encoder layer.

In a transformer model, the encoder and decoder are two fundamental components. The encoder processes the input sentence and transforms it into a set of context-rich representations. It applies self-attention mechanisms to capture relationships between all words in the input sentence simultaneously. On the other hand, the decoder outputs a sentence based on the encoder's representations. It uses both self-attention and cross-attention mechanisms to ensure that the output is contextually relevant to the input, its architecture makes it perfect for tasks like machine translation and text generation.

1.2.3 Encoder-only models

This section presents the most common models in the literature that are based only on the encoder component of Transformers, along with their components and the methods

used for their training.

BERT [*Devlinet al.*, 2019]

Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al., 2019] is a groundbreaking machine learning model for natural language processing (NLP) tasks. Developed by researchers at Google, BERT was introduced in 2018 and has since revolutionized the field of NLP.

Unlike traditional language models that process text in a unidirectional manner (left-to-right or right-to-left), BERT, thanks to its encoder architecture, allows the model to consider the context of a word from both the preceding and succeeding words, enabling a more comprehensive understanding of the text, its called bidirectional representation learning.

BERT was pre-trained on two tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). The NSP task aimed to enhance the model's ability to capture long-range dependencies across sentences, a crucial aspect for many downstream NLP applications. The MLM task addressed the limitation of unidirectionality by randomly masking words in the input sentence and training the model to predict the original words based on their bidirectional context.

The model was pre-trained on two datasets, together holding 3,300M words - BooksCorpus and English Wikipedia. It was constructed and released in two versions: $BERT_{BASE}$ and $BERT_{LARGE}$ differing in their architectural configurations. $BERT_{BASE}$ has 12 layers, 768 hidden units, and 12 self-attention heads, with a total of 110 million parameters. $BERT_{LARGE}$, on the other hand, has 24 layers, 1024 hidden units, 16 self-attention heads, and 340 million parameters.

Roberta [*Liuet al.*, 2019]

The RoBERTa (Robustly optimized BERT approach) model was introduced by Liu et al. [2019]. This model builds upon Google's BERT, which was released in 2018. RoBERTa enhances BERT through several key modifications:

Training Corpus: RoBERTa uses a much larger and more diverse training corpus compared to BERT. While BERT was trained on English Wikipedia (2.5B words) and BookCorpus (800M words), RoBERTa is trained on a dataset that includes the aforementioned corpora along with additional sources such as the Common Crawl News dataset (63 million articles), WebText (from OpenAI, 45 million links), and the Stories dataset (a subset of Common Crawl data). This results in a total of over 160GB of uncompressed text data, significantly increasing the amount of data RoBERTa is exposed to during training.

Dynamic Masking : BERT uses a static masking strategy, where the same tokens are masked in each epoch during training. RoBERTa introduces dynamic masking, where the tokens chosen for masking are selected randomly for each training instance on the fly. This ensures that the model sees a variety of masked positions and can better generalize to different patterns of masked tokens.

No Next Sentence Prediction (NSP) : RoBERTa removes this task, as it was found to be not beneficial and potentially detrimental to performance. Instead, RoBERTa focuses solely on the masked language modeling (MLM) task, which predicts randomly masked tokens in a sentence.

These strategies include also using longer sequences, larger batch sizes, and more training iterations, which collectively help in achieving better performance. Additionally, careful hyperparameter tuning is conducted to optimize the model for each specific task.

BerTweet [*Nguyenet al.*, 2020]

BERTweet is a large language model [Nguyen et al., 2020] specifically designed for understanding the nuances of communication on Twitter. It leverages the RoBERTa pre-training procedure and is trained on a massive dataset of 850 million English Tweets.

BERTweet is a specialized variant of the BERT model, designed specifically for processing and understanding text in the context of social media, particularly Twitter. This specialization allows BERTweet to perform well on tasks dealing with informal language, abbreviations, emojis, and hashtags commonly found in social media conversations which makes Twitter text a unique challenge.

1.2.4 Decoder-only models

This section presents the most common models in the literature that are based only on the decoder component of Transformers, along with their components and the methods used for their training.

GPT-Family models

GPT-1: The Pioneer GPT-1, the initial model in the GPT series, pioneered the idea of pre-training on extensive text corpora. It employed a stack of decoder transformer architectures. During training, GPT-1 learned to predict the next token in a sentence based on the preceding context. This unsupervised pre-training enabled it to capture intricate language patterns and semantics, which could then be fine-tuned for specific NLP tasks. One issue with fine-tuning is that it requires a substantial amount of labeled data, which differs from how humans learn.

GPT-2: Scaling Up One solution to address the issues observed in GPT-1 is the use of Meta-Learning. This approach was introduced in the next generation of GPT, namely GPT-2. GPT-2 shares similarities with GPT-1 in that it still includes the same pre-training phase (predicting the next token). However, instead of using the fine-tuning approach, they adopted zero-shot learning. Zero-shot learning involves providing a prompt alongside the input and instructing the model on what actions to perform with the input. However, zero-shot learning posed a challenge for the model, so they scaled up the architecture to capture a broader range of patterns during pre-training.

GPT-2 represented a breakthrough in terms of scale, featuring 1.5 billion parameters, significantly more than its predecessor. The key innovation was its ability to generate

coherent and contextually relevant text over longer sequences. It demonstrated proficiency in tasks such as text completion, translation, and summarization, achieving remarkable fluency.

GPT-3: Unleashing Massive Scale The approach used in GPT-2 did not perform as well as fine-tuning in several benchmarks; however, scaling the architecture did indeed help improve performance in some ways. Continuing in this line of thought, they employed the same strategy, which involves Meta-Learning and further scaling, leading to the third generation of GPT, GPT-3, the largest model with 175 billion parameters. It was trained in the same way as its predecessors, predicting the next token, and then incorporating meta-learning. Unlike GPT-2, it could perform meta-learning using multiple techniques (zero-shot, one-shot, few-shot).

Meta-learning and fine-tuning each have their advantages and disadvantages. It's worth noting that meta-learning did not completely replace fine-tuning in multiple benchmarks. After all, the first version of ChatGPT, released by OpenAI in December 2022, utilizes a fine-tuned GPT-3 model at its core.

LLaMA [*Touvronet al.*, 2023]

Accordingly, to Touvron et al. [2023], the LLaMA models consist of foundational language models with parameters ranging from 7B to 65B. These models were trained on trillions of tokens, demonstrating the feasibility of achieving state-of-the-art performance using only publicly available datasets, without relying on proprietary and inaccessible datasets like GPT. Notably, LLaMA-13B surpasses GPT-3 (175B) in most benchmarks. Compared to GPT models, LLaMA offers advantages in terms of GPU requirements for fine-tuning, as it comes in various parameter sizes (7B requiring 14GB of RAM). Additionally, LLaMA boasts faster inference times than GPT-3 due to its smaller architecture, resulting in fewer operations in the forward pass.

1.2.5 Encoder-Decoder Models

This section presents the most common models in the literature that consist of both encoder and decoder, along with their components and the methods used for their training.

BART [*Lewiset al.*, 2019]

The BART model was introduced by Lewis et al. [2019]. BART was developed to address certain limitations in BERT, which is encoder-only and often performs poorly in text generation tasks. BART utilizes a standard Transformer architecture consisting of two components: a bidirectional encoder (similar to BERT) and a left-to-right decoder (like GPT). BART distinguishes itself from BERT in its pre-training objectives, where a combination of the following transformations is applied to the training datasets:

- Permute the order of sentences and try to find the correct order
- Mask random tokens (just as BERT)

- Rotate the document to make it start at a specific token
- Mask a sequence of k tokens with a single mask token

1.2.6 Language Models Learning Approaches

The effectiveness of large language models (LLMs) largely depends on their Learning Approaches during their development. LLMs come pre-trained on vast amounts of data, and to adapt them for specific tasks (propaganda classification for example), there are two primary methods: fine-tuning and prompt learning as shown in Figure 1.2. Fine-tuning involves further training the pre-trained model on task-specific data, while prompt learning, which includes both hard and soft prompts, modifies the input format to guide the model's responses. Soft prompts include additional training and Hard ones do not.

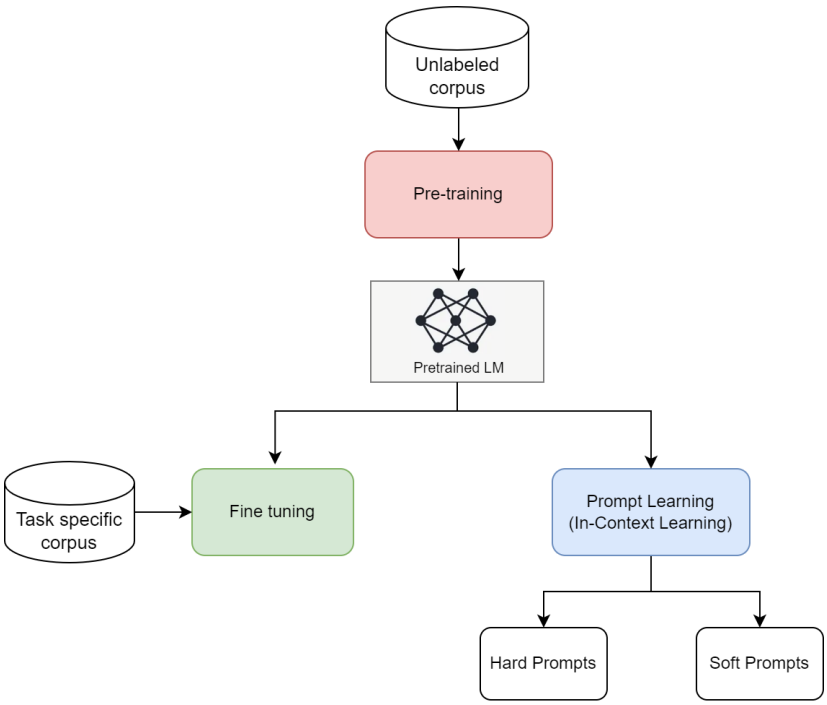


Figure 1.2: Language Models Learning phases

Pre-training

Pre-training is the very first phase where a Language Model is trained on a huge corpus of text data. This stage aims to develop a generalized understanding of language by exposing the model to diverse linguistic patterns, grammar, and knowledge across various domains. During pre-training, the model tries to learn how to predict the next word in a sequence, understand the context, and capture the semantic relationships between words. This process equips the model with a foundational understanding of language, which is crucial for its adaptability and performance in downstream tasks.

The pre-training process involves training the model on a large dataset to learn general patterns before fine-tuning it on downstream tasks, for instance, the datasets that were used to pre-train LLaMA 1 [Touvron et al., 2023] are :

Dataset	Sampling propotion.	Epochs	Size (GB)
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1.1: Pre-training data [Touvron et al., 2023]

It is a resource-intensive process that requires significant computational power, often involving the use of GPUs or TPUs to handle massive datasets and complex model architectures.

Fine-tuning

Fine-tuning refers to the process of taking a pre-trained language model and training it further on a task-specific dataset (propaganda detection in our case). This stage is crucial for tailoring the model's responses to be more relevant and accurate for particular use cases. Fine-tuning involves continuing the training process with a dataset that is more relevant to the desired application, such as legal documents for a legal LM or medical journals for a healthcare-focused LM. During fine-tuning, all model parameters are updated using supervised learning to ensure that the model precisely adapts to the specific dataset or task.

This process requires additional computational resources and can increase training times, especially for larger models.

Fine-tuning can vary between models depending on their architecture. If it is an encoder-only model, a classification head needs to be added on top of the transformer model. The final output embeddings of the transformer serve as the input for this classification head. The classification head typically consists of a fully connected layer with the number of neurons corresponding to the task's requirements, such as binary or multiclass classification. It is crucial to consider the appropriate activation function for this layer. Once the classification head is added, predictions can be generated for the specified number of classes. One significant advantage of fine-tuning encoder-only models for classification tasks is that we can adjust the threshold as needed, thanks to the appended classification head.

If the model contains a decoder part of the transformers, then the prediction is going to be a text. A common problem here is that in classification tasks, we cannot adjust the threshold of the predicted label because the probability given by the model represents the likelihood of the next token (i.e., the probability of the next word). Suppose our label was tokenized into multiple tokens. In that case, the probability of that label loses its meaning in decoder models. This is why in literature, it's often noted that decoder models excel at text generation.

Prompt Learning (In-Context Learning)

There are two categories of prompting methods:

a. Hard prompts (Prompt Design)

Hard prompts, also known as manually handcrafted text prompts, involve the creation of a discrete input sequence of text that gets combined with the input text of that task as shown in 1.3. Both the model and the embeddings of the hard prompt are frozen and no backpropagation updates are happening. Zero-shot learning and few-shot learning can be considered as forms of prompt design. A Zero-shot prompt in our case would be something like this:

You are an expert in detecting and classifying propaganda, here's a text and classify it as to whether it contains propaganda or not: {TEXT}

The downside is shown in Table 1.2 from Liu et al. [2023], where they asked a model each time slightly different prompts by changing [A] position but overall keeping the same meaning, and the model answer is [B], but this minimal change in the prompt would lead to a completely different set of results. It requires a lot of effort and many attempts to craft a good prompt.

Prompt	Presicion
[A] is situated in [B]	31.29
Which country or state is [A] situated in ? [B].	19.78

Table 1.2: Small change in the prompt, a huge performance difference [Liu et al., 2023]

b. Soft prompts (Prompt Tuning)

As shown in 1.3, soft prompts are learnable embeddings that are combined with input embeddings and can be optimized for a specific dataset through backpropagation. However, a drawback is that these "virtual tokens" are not human-readable, as they cannot be directly associated with words.

The central concept of prompt tuning is that the prompt tokens possess their parameters, which are updated separately. This allows the pre-trained model's parameters to remain unchanged, with only the gradients of the prompt token embeddings being adjusted. The performance of prompt tuning is comparable to traditional methods that involve training the entire model, and it improves as the model size increases.

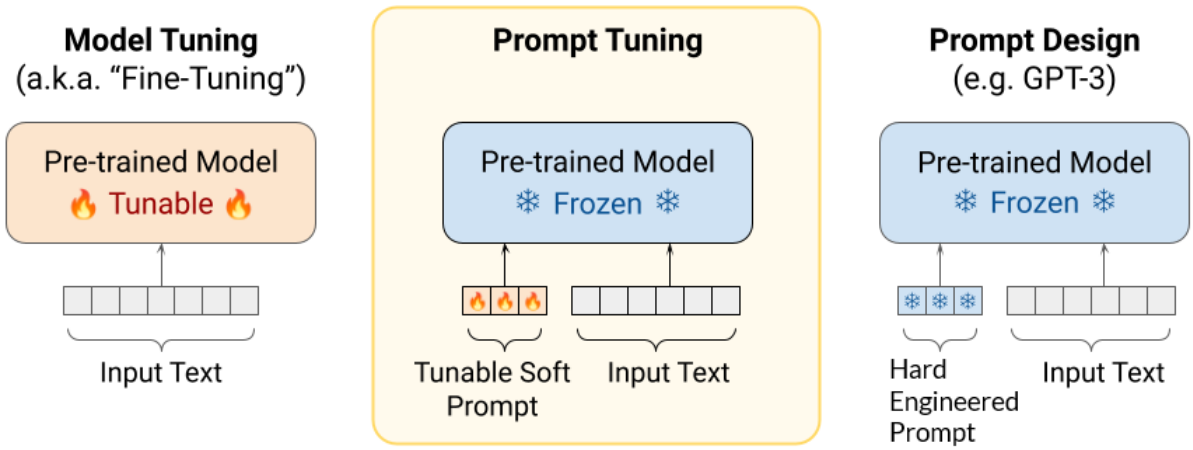


Figure 1.3: Fine-Tuning vs Prompt Tuning vs Hard Prompt Design (Google Research http://goo.gle/3Bch2lL)

1.2.7 Conclusion

In this section, we presented the Transformer architecture, which serves as the foundation for most modern language models, either through its encoder or decoder components. We also discussed the advantages and use cases associated with each category of models. Additionally, we detailed the learning approaches used to train these architectures to become effective language models.

Chapter 2

Existing Work

2.1 Introduction

Propaganda detection is a relatively new challenge in the field, first emerging in the literature around 2019. Despite its recent inception, it has quickly garnered significant research attention. In this section, we will explore key contributions from researchers in this area, including the creation of datasets and the development of machine learning approaches to automatically detect propaganda.

2.2 Propaganda classification in the literature

In the literature, three essential methods are used for annotating and classifying text as either propagandist or not:

- Manual Annotation: Experts serve as annotators, relying on their linguistic expertise to classify the text.
- Weak Labeling: This method involves using noisy or imprecise labels generated from heuristic rules. While not as accurate as manual annotation, weak labeling provides large-scale training data for machine-learning models.
- Machine Learning-Based Methods: Machine learning has revolutionized natural language processing (NLP), leading to significant breakthroughs in the detection of persuasion and misinformation in writing styles.

In the following, we will present some works in each of the 3 categories above.

2.2.1 Manual Annotation

To the best of our knowledge, there are only 2 papers where they detailed how their datasets were annotated manually by experts, which are :

HQP: A Human-Annotated Dataset for Detecting Online Propaganda *Maaroufet al.* [2023]

Human-annotated dataset for detecting online propaganda by Maarouf et al. [2023]. The authors describe the manual annotation of their high-quality propaganda dataset in a detailed process. Here's a summary:

1. **Annotation Guidelines:** They developed detailed guidelines to ensure consistency and clarity in the annotation process. These guidelines included definitions of propaganda techniques, examples of usage, and instructions on how to annotate ambiguous cases. The guideline is provided in Appendix A.

2. **Annotator Training:** The authors trained annotators extensively to familiarize them with the guidelines and the specific tasks. This training involved practice sessions where annotators worked on a subset of data, received feedback, and adjusted their understanding of the guidelines accordingly.

3. **Annotation Process:** The annotation itself was carried out manually by a team of annotators. Each instance in the dataset was carefully reviewed, and the relevant propaganda techniques were identified and labeled according to the guidelines.

4. **Quality Control:** To ensure high quality, the annotations were subject to multiple rounds of review. Initially, each instance was annotated by two independent annotators. Discrepancies between their annotations were then resolved by a third annotator or through discussion among the annotators to reach a consensus.

PROPAGANDA *Martinoet al.* [2019]

In the paper "Fine-Grained Analysis of Propaganda in News Articles", Martino et al. [2019] introduced a dataset that has been used as a benchmark since then and it was manually annotated by experts. The objective was to annotate fragments of text with 18 propaganda techniques. Due to the considerable time investment required to grasp and memorize all these propaganda techniques, relying on crowd-sourcing for annotation is not feasible. Therefore, they collaborated with a professional annotation company called A Data Pro.

They used the γ agreement measure, which handles overlapping annotations and evaluates both span identification and labeling accuracy.

There were two phases to the annotation process, and each team of annotators worked on 220 papers. The same texts were individually annotated in the first step by annotators $a1$ and $a2$. They worked together with a consolidator, $c1$, in the second stage to go over every instance and come to a final agreement on annotation. A similar process was used by consolidator $c2$, annotators $a3$, and $a4$. The entire corpus needed 395 man-hours to annotate. As can be seen in Table 2.1, both teams' agreement ratings were originally somewhat low for span selection (0.30 and 0.34), with slightly lower scores when labeling was taken into account. But after annotators talked to the consolidator about the disparities, the (γ) values improved dramatically, going up to 0.74 and 0.76 for each team.

Annotation		spans	+labels
A_3	A_4	0.34	0.28
A_1	A_2	0.30	0.24
A_1	C_1	0.58	0.54
A_2	C_1	0.74	0.72
A_3	C_2	0.76	0.74
A_4	C_2	0.42	0.39

Table 2.1: (γ_s) agreement between annotators when identifying just spans (span) and when identifying spans including labeling (+labels) [Martino et al., 2019]

2.2.2 Weak-Labeling

Weak labeling is a technique used to annotate a text dataset when we have limited or noisy supervision or maybe a huge amount of data, making it challenging to obtain accurate and complete annotations. This approach leverages available, imperfect, or less reliable sources of information to generate annotations for the dataset. Keywords-based method are one of the most used in weak labeling.

This approach is especially useful when we have access to a set of keywords that are indicative of classes within the text data. In the paper written by Maarouf et al. [2023], authors share their method for collecting a dataset for propaganda classification in which they weak labeled it and human annotated it to show the difference in model performance.

1- Candidate search for positive class (D+):

Similar to a previous study done by Vijayaraghavan and Vosoughi [2022], they conduct a keyword-based search using the Twitter Historical API. They specifically search for quotes and replies that contain terms (keywords) like "propaganda" ∧ "Russian" or "propaganda" ∧ "war" that might be used to accuse the original tweet of being propaganda. See Table 2.2 for the complete list. there were almost 2.5 million D+ applicants.

2- Candidates for negative class (D-):

They randomly crawl a sample of 2.5 million tweets that describe the Russian-Ukrainian conflict but that haven't necessarily been labeled as propaganda in order to gather candidates for the negative class. Two examples of keywords are "war" ∧ "Russia" or "war" ∧ "Ukraine." A comprehensive list of search terms can be found in Table 2.3.

3- Labeling:

Contrary to the work done by Vijayaraghavan and Vosoughi [2022], which labels the tweets based solely on whether they belong to D+ or D- (weak labeling is employed). Maarouf et al. [2023] selects N = 30K tweets from the union of D+ and D- for further manual annotation by hand.

D+ Keywords
propaganda ∧ war
propaganda ∧ putin
propagandist ∧ russia(n)
propaganda ∧
kremlin propagandist ∧ kremlin
putinist(s)
putinism
russia(n) ∧ lie(s)
putin ∧ propagandist
fake news ∧ russia(n)
propaganda ∧ russia(n)
lie(s) ∧ war

Table 2.2: List of keywords used to get propaganda accusations [Maarouf et al., 2023]

Keywords (D-)
#zakharova
russia ∧ war
#russia
#russiaukraine
#nato
putin ∧ war
ukraine ∧ war
#standwithrussia
#standwithputin
#ukraine
#istandwithrussia
#donbass
#lavrov
#istandwithputin
#russianukrainianwar
#putin
mariupol
#ukrainerussiawar

Table 2.3: A list of keywords for retrieving tweets about war [Maarouf et al., 2023]

Limitations : authors of the dataset *HQP* [Maarouf et al., 2023], showed that training SOTA models on the Human Annotated dataset improves AUC by 44%, compared to models trained on weakly labeled datasets.

- **Noise and Inaccuracy** : weak labeling often relies on imperfect sources of information, such as heuristics, rules, or noisy data.
- **False Positive** : weak labeling leads to classifying many tweets as propaganda, even those that don't qualify as propagandist tweets.
- **Quality Control** : human reviews should be in place to verify the accuracy of the generated annotations.

2.2.3 Machine Learning based Methods

In this section, we will present recent works (2019-2023) on machine learning model architectures and techniques used by researchers to address propaganda as both a binary and multi-class classification problem.

San Martino et al, 2019 [*Da San Martinoet al.*, 2019] **:**

Approach :

The system introduced in the paper written by Da San Martino et al. [2019] tackles propaganda detection in two stages: First, it classifies sentences to identify those containing propaganda (Sentence-Level Classification SLC). Second, it pinpoints the specific propaganda technique used and its location (span) within the text (Fragment-Level Classification FLC). It's important to note that these tasks analyze text at granularities g_1 and g_2, i.e., SLC operates on sentences, while FLC focuses on smaller units, like individual words or tokens.

Instead of directly using basic sentence-level (low-granularity) information, the authors propose a model that leverages this data to guide the identification of specific techniques and their locations (higher-granularity task). Figure 2.2 illustrates this model's construction.

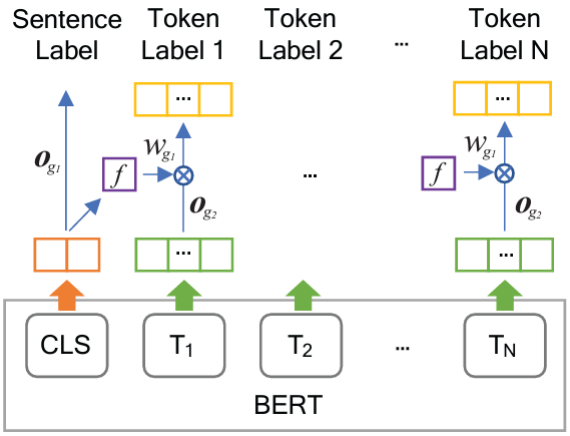


Figure 2.1: Multi-granularity Model [Da San Martino et al., 2019]

Each task (sentence-level and fragment-level) has a dedicated classification layer, denoted as L_{gk} (L_{g2} for fragment-level and L_{g1} for sentence-level) that produces O_{gk}. These layers receive a feature representation (embeddings) based on the corresponding granularity level (g_k). The number of classes for each task defines the output dimension. For example, the sentence-level task uses binary classification (2 outputs: propaganda or not propaganda) while the fragment-level task has 19 outputs due to 18 propaganda techniques and an additional class for "no technique found."

The dimension of the feature representation is determined by the embedding layer (e.g., BERT uses an embedding size of 768). A trainable gate function (f) takes the output (O_{g1}) from the sentence-level layer and generates a weight (w_{gk}) used by the next fragment-level task. This weight essentially modulates the influence of the sentence-level

information on the fragment-level classification.

$$w_{gk} = f(O_{gk})$$

The gate function (f) consists of an activation function and a projection layer that condenses its output to a single value (weight). This weight influences the prediction of the fragment-level task (L_{g2}). Imagine a scenario where the weight (w_{g1}) from the sentence-level task is zero. This signifies high confidence that the sentence doesn't contain propaganda. Consequently, the fragment-level task's output (O_{g2}) will also be driven towards zero, indicating no anticipated propaganda techniques within that sentence.

Similarly, during backpropagation, if w_{g1} is zero for a specific instance, the final error (entropy loss) associated with that instance becomes negligible. As a result, the model prioritizes learning from cases where the sentence-level classifier is less certain (weight closer to 1), leading to more efficient training on identifying propaganda techniques.

Results :

Models	Span			Full Tasks		
	Prec	Rec	F1	Prec	Rec	F1
BERT uncased	39.57	36.42	37.9	21.48	21.39	21.39
Multi-Granularity with activation function:						
*ReLU	43.29	34.74	38.28	23.98	20.33	21.82
*Sigmoid	44.12	35.01	38.98	24.42	21.05	22.58

Table 2.4: (1) **Span** checks if the fragment spans have been identified. **Full tasks** evaluate the correct spans and assign the appropriate propaganda technique to those spans. [Martino et al., 2019]

The multi-granularity model performs better than all baselines. This demonstrates the impact of the model not taking into account sentences that it found to be non-propagandistic when classifying tokens at the token level. However, with an F1 of up to 60.98%, binary classification (propaganda vs Not propaganda) performance is far from ideal.

Models	Prec	Rec	F1 Score
All-Propaganda	23.92	100.0	38.61
BERT	**63.20**	53.16	57.74
Multi-Granularity ReLU	60.41	**61.58**	**60.98**
Multi-Granularity Sigmoid	62.27	59.56	60.71

Table 2.5: SLC task performance [Martino et al., 2019]

The authors first evaluated the model's performance on sentence-level classification (SLC) - the results are shown in Table 2.5. Interestingly, training the model with the

higher-granularity task (fragment-level classification) significantly improved its performance on the lower-granularity task (SLC) compared to a baseline BERT model. This multi-granularity approach achieved an impressive 8.42% increase in recall and a 3.24% increase in F1 score. It's worth noting that, unlike token-level classification, the ReLU activation function yielded better results than the Sigmoid function for this sentence-level task.

Anni Chen et al, 2023 [*ChenandDhingra*, 2023] :

Approach :

The authors were inspired by the decision tree annotation guide used by human experts in previous work by Da San Martino et al. [2019]. This guide (included in the Appendix A) helped them design a "light version" where each decision node is replaced by a machine learning classifier.

As for the pre-processing, unlike prior studies (e.g., San Martino et al., 2019) that focused solely on the target sentence, the authors implemented padding with 256 tokens on both sides of the span to provide additional context for the model. Special tokens, $< bop >$ (beginning of propaganda) and $< eop >$ (end of propaganda), were introduced to identify overlapping or nested spans within the text. This allows the model to differentiate between multiple propaganda techniques within a sentence. The text is divided into overlapping windows of 512 tokens with a stride of 256 tokens. This ensures efficient processing while capturing context. If a span extends beyond a window boundary, it's truncated, and the corresponding $< eop >$ token is inserted at the window's end to maintain balance with the $< bop >$ tokens.

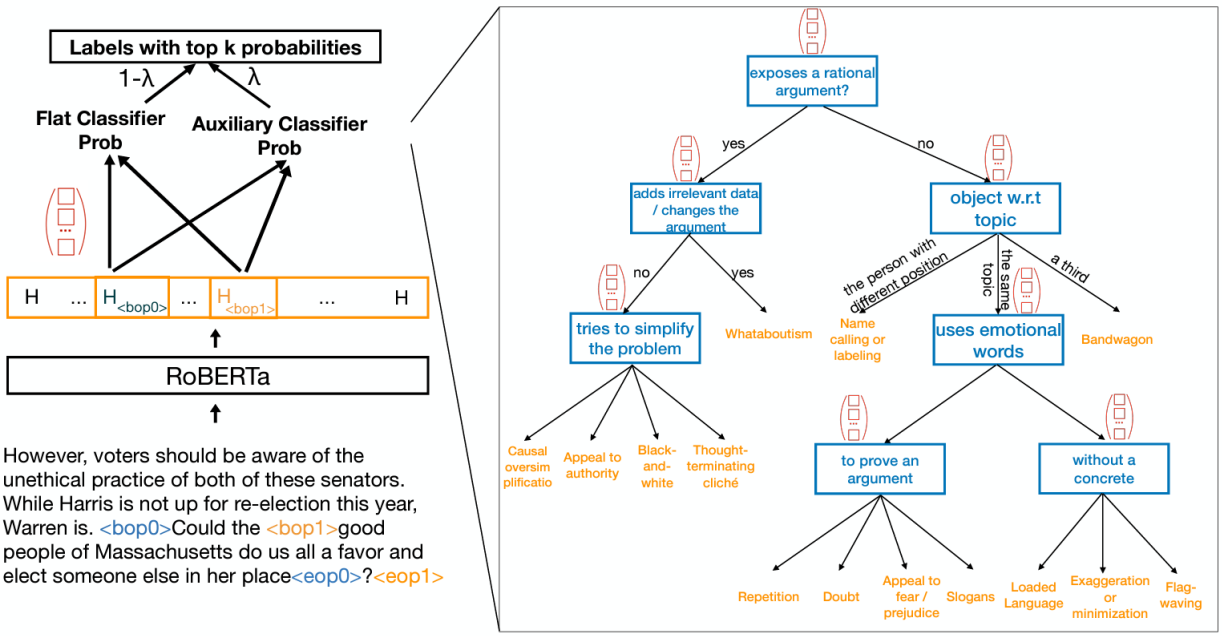


Figure 2.2: Overview of the hier MIML$_{RoBERTa}$ [Chen and Dhingra, 2023]

For the Flat classifier is just a neural network trained on multi-label classification using the cross-entropy loss l_{flat} where classes $c = 0, \ldots, 13$ are propaganda techniques.

$$l_{\text{flat}} = \sum_{c=1}^{C_k} -1_{y=c} \log p_{flat}(c).$$

Let K be the number of intermediate nodes (auxiliary classifiers) in the network, and let C_k represent the number of outgoing edges (labels) for each classifier k. Given a

span representation (h_s) obtained from the RoBERTa model, the system calculates the probability ($p_k(i)$) of following edge i from any node k. This probability is proportional to the exponential of the dot product between h_s and a weight vector $w_{k,i}$ associated with that specific edge. The softmax function is then applied to normalize these probabilities across the C_k labels for each node k. $p_k(i) \propto \exp\left(h_s \cdot w_{k,i}\right), \quad \forall i = 0, \ldots, C_k$

Furthermore, each leaf node in the network represents a specific propaganda technique label (c). The path leading from the root node to this leaf node can be denoted by I_c, which is a set of tuples [(k_1, i_1), (k_2, i_2), ...]. Each tuple in this set represents a pair consisting of a classifier (k) and its corresponding label (i) along the path to the leaf node labeled c.

To calculate the overall probability of selecting a specific propaganda technique label (c) based on the path probabilities ($p_k(i)$) we can use the formula:

$$p_{\text{aux}}\left(c\right) = \prod_{(k,i) \in I_c} p_k(i).$$

When training is conducted, an auxiliary loss is computed for each span that aims to minimize the negative log-likelihood of selecting the correct label y along the path.

$$l_{\text{aux}} = \frac{1}{|I_y|} \sum_{(k,i) \in I_y} \sum_{c=1}^{C_k} -1_{k=c} \log p_k(i).$$

The final loss function is made up of two parts: the flat loss and the auxiliary loss mentioned above.

$$l_{\text{ovr}} = (\lambda - 1) \cdot (-l_{\text{flat}}) + \lambda \cdot l_{\text{aux}}$$

Results :

In the results of their paper, the authors did not provide a profound analysis and only focused on the micro-F1 score, without presenting the Precision/Recall for each propaganda technique.

The non-hierarchical model is obtained by setting $\lambda = 0$, meaning they trained only the flat model, while the Hierarchical model is trained using both losses (flat and auxiliary).

Table 2.6 shows that the MIML$_{RoBERTa}$ hierarchical model performed the best among single models.

System	Test F1
non-hierarchical MIML RoBERTa	62.179
hierarchical MIML RoBERTa	**62.793**

Table 2.6: the performance of the models on the test set of PROPAGANDA dataset measured by F1 score [Chen and Dhingra, 2023]

Sprenkamp et al, 2023 [*Sprenkampet al.*, 2023] :

Approach :

The dataset was divided into 75 articles for testing and 371 training (with 20% retained for validation). To identify propaganda techniques in the news articles, they used **five** variants of fine-tuned GPT-3 [Brown et al., 2020] and standard GPT-4 [OpenAI, 2023].

In their work with GPT-4, the authors utilized two distinct prompting strategies. The first strategy involved prompting the model solely to generate labels for a given article. This approach is referred to as the "baseline" prompt. To complement this, they employed a "chain-of-thought" prompt that instructed the model to reason and give explanations about the predicted labels which proved to give better results overall. Notably, both prompting strategies utilized a "few-shot" learning approach, providing a single illustrative example for each propaganda technique within the prompt itself.

They used two different kinds of prompts for GPT-4. In one method, labeled as **'baseline'** prompt, the model's only responsibility is to output the labels for a given article. They also use a **'chain-of-thought'** prompt that, in a complementary manner, tells the model to reason about the expected labels. They use the **'few-shot'** pattern for both kinds of prompts by providing a **single** example for every propaganda tactic mentioned in the prompt.

They also used the provided dataset to fine-tune GPT-3 Davinci and used both types of prompts (**'baseline'** and **'chain-of-thought'**) at the **prediction** level.

When the authors compared these five models to Abdullah et al.'s state-of-the-art method [Abdullah et al., 2022], they found that RoBERTa received the highest F1 score Using each model's micro average F1 score, precision, recall, and F1 scores per label (i.e., propaganda strategy).

Results :

The results of the proposed experiments are presented in Tables 2.7 and with more details on each propaganda technique in table 2.8. As can be seen, both GPT-4 versions demonstrably outperform fine-tuned GPT-3 models. Notably, the GPT-4 "base" model achieves the highest F1 score among all developed models.

Models	Prec	Rec	F1
GPT-3 baseline	44.35%	44.00%	44.18%
GPT-3 chain-of-thought	48.62%	28.16%	35.66%
GPT-4 baseline	52.86%	64.52%	58.11%
GPT-4 chain-of-thought	56.86%	57.82%	57.34%
Baseline (Abdullah et al., 2022)	NA	NA	63.40%

Table 2.7: Performance of the 4 models. fine-tuned GPT-3 and out-of-the-box GPT-4 with the two prompt types [Sprenkamp et al., 2023]

Propaganda Techniques	GPT-4 Base	GPT-4 COT	GPT-3 base	GPT-3 COT	Baseline
Appeal_to_Authority	30.29%	47.05%	11.11%	15.38%	47.36%
Bandwagon_ad_hitlerum,Reductio	19.35%	25.00%	0.00%	0.00%	4.87%
Appeal_to_fear-prejudice	34.28%	64.17%	57.14%	59.52%	43.6%
Exaggeration,Minimisation	62.29%	64.28%	31.03%	44.15%	33.03%
Black-and-White_Fallacy	44.44%	22.22%	14.81%	13.33%	24.09%
Causal_Oversimplification	27.58%	41.46%	13.63%	30.30%	19.44%
Doubt	45.09%	50.79%	13.63%	38.70%	61.36%
Flag-Waving	29.17%	30.43%	10.52%	10.52%	61.49%
Repetition	11.76%	39.43%	58.11%	66.17%	31.14%
Loaded_Language	89.61%	91.01%	86.79%	91.80%	75.71%
Thought-terminating_Cliches	12.50%	0.00%	31.57%	36.36%	25%
Name_Calling,Labeling	76.26%	60.55%	48.42%	64.34%	67.49%
Whataboutism,Straw_Men,Red_Herring	16.66%	30.76%	23.52%	14.28%	20.83%
Slogans	0.00%	0.00%	0.00%	0.00%	54.90%

Table 2.8: A Table that presents F1 Scores for each Propaganda Technique. COT stands for chain of thought [Sprenkamp et al., 2023]

Further analysis of the best 2 models (GPT-4 Base and COT):

Propaganda Technique	GPT-4 Base		GPT-4 COT	
	PR	RC	PR	RC
Appeal to Authority	50.00%	21.73%	42.85%	52.17%
Causal Oversimplification	57.14%	18.18%	28.33%	77.27%
Bandwagon and Reductio ad hitlerum	16.66%	23.07%	17.14%	46.15%
Appeal to fear-prejudice	48.83%	93.33%	48.31%	95.55%
Black and White Fallacy	27.27%	46.15%	15.62%	38.46%
Loaded Language	84.53%	95.34%	93.82%	88.37%
Exaggeration,Minimisation	49.35%	84.44%	53.73%	80.00%
Name Calling,Labeling	72.60%	80.30%	76.74%	50.00%
Doubt	33.33%	69.69%	34.40%	96.96%
Flag-Waving	53.84%	20.00%	63.63%	20.00%
Thought-terminating Cliches	25.00%	8.33%	0.00%	0.00%
Slogans	0.00%	0.00%	0.00%	0.00%
Repetition	100%	6.25%	60.86%	29.16%
Whataboutism,Straw Man and Red Herring	25.00%	12.50%	40.00%	25.00%

Table 2.9: Precision and Recall of GPT4-base and Chain of thought. [Sprenkamp et al., 2023]

Khosla et al, 2020 [Khoslaet al., 2020] :

Approach :

BERT-BiLSTM with Multi-granularity

The authors implemented a model that takes input sentences to then predict if it contain propaganda or not, and this helps the second phase of identifying propaganda spans. and each step has its own classification layer denoted L_{sent} and L_{tok}, where the span feature representation H_{tok} is influenced by H_{sent}. the first step consists of mean-pooling the representations taken from the bert model output to use it for sentence classification, which is then concatenated with the document-level (f_{doc}) features and sentence-level (f_{sent}) features, and passed the result of this concatenation through a fully-connected layer to get p_{sent}:

$$p_{sent} = \text{Fully-connected}(H_{sent})$$

where:

$$H_{sent} = [\text{mean-pool}(\text{bert-base}([\text{CLS}],\ s_1, s_2, \ldots, s_n,\ [\text{SEP}])); f_{sent}; f_{doc}]$$

For span level predictions, the authors used representations from the bert output, concatenated with span features for each word (F_{word}), and passed them through a Bidirectional LSTM:

$$H_{tok} = \text{Bidirectional-LSTM}(H^*_{tok})$$

where:

$$H^*_{tok} = [\text{bert-base}([\text{CLS}],\ s_1, s_2, \ldots, s_n,\ [\text{SEP}]) \oplus f_{word}],$$

Here, $\oplus$ denotes the element-wise concatenation operator, and H_{tok} represents the contextual token representations output by the BiLSTM.

The sentence representation is then masked so that p_{sent} is in a single dimension, then passing it to a sigmoid function. The span representation (H^{sp}_i) is multiplied by the gate to produce G_{sp}, which biases the span model to overlook samples that are strongly classified as negative at the sentence level. This approach enables the model to focus on learning additional information that can enhance the prediction of individual tokens:

$$G^{sp}_i = g_{sent} * H^{sp}_i, \quad p^{sp}_i = \text{FC}(G^{sp}_i)$$

where :

$$g_{sent} = \sigma(W_g p_{sent} + b_g),$$

The cross-entropy loss was used for both sentence and span level classification, with the sentence classification having a sigmoid activation function and the span classification using softmax, they also used a hyper-paramter α to balance the contributions of the two loss functions as follows:

$$L = \alpha L_{sent} + (1 - \alpha) L_{tok}$$

Additional Features

- **Affective and Semantic Features:** They add emotional lexicon elements, such as the LIWC dictionary to the word embeddings because propaganda is frequently characterized by affective and emotional language. Additionally, they give each token a score based on how frequently it appears in propaganda spans. They also integrate Empath's finer-grained thematic categories and named entities, which are semantic class features.

  The token-level BERT representations are concatenated to these word-level features (F_{word}).

- **Sentence-level Features F_{sent}:** To create a 1024-dimensional vector representation of the sentence, it is passed to a BERT-large-case model that has been refined using news items from different sources.

- **Documentlevel Features F_{doc}:** In a similar manner, they average the BERT embeddings for every sentence in the document to get a 1024dimensional embedding of the text.

  These characteristics provide the model with information on the article's general thematic content, which is particularly useful in identifying "repetition" propaganda—that is, propaganda that mentions the same events repeatedly.

Unsupervised Fine-tuning

They pre-train BERT on news articles from both propaganda and trusted sources to capture the persuasion techniques and metaphorical language used in propaganda. They scraped about 20k articles. Then using the Huggingface transformers library, they trained BERT on next sentence prediction and masked language modeling losses.

Class-Imbalance

They give samples of the minority class more weights for losses in order to alleviate class imbalance. These weights are computed as the inverse of the class samples' normalized frequency, and they are then passed to the cross-entropy function:

$$L = -\frac{1}{N} \sum_{n=1}^{N} w_n * \text{loss}_n$$

where the weight for each sample's loss is denoted by w_n.

Results :

For the results, the multi-granularity BERT and BERT-BiLSTM performance averaged on 5 runs is presented in table 3.14, whereas table 3.15 presents the importance of each granularity level, either word or sentence level, and finally, table 3.16 presents the impact of the various additional features.

Models	Span F1 score
BERT-BU (base-uncased)	41.38
BERT-BC (base-cased)	42.12
BERT-LU (large-uncased)	43.12
BERT-LC (large-cased)	**43.56**
BERT-LU Bidirectional LSTM	43.41
BERT-LC Bidirectional LSTM	**43.86**

Table 2.10: Analysis of model performance on different granularity levels [Khosla et al., 2020]

Feature	F1 score
Word	
MGU LSTM	43.41
+ Aff + LIWC	43.69
+ Aff + LIWC + Syn (A)	**44.11**
+ Aff + LIWC + Syn + NER + Emp	43.81
MGC LSTM	43.86
+ Aff + LIWC	43.87
+ Aff + LIWC + Syn	44.10
+ Aff + LIWC + Syn + NER + Emp (B)	**44.28**
Document	
MGU-LSTM-A + Doc	44.27
MGC-LSTM-B + Doc	44.35
Sentence	
MGU-LSTM-A + Sent	44.07
MGC-LSTM-B + Sent	**44.38**

Table 2.11: Incremental study of the models with different features (word, document, and sentence) [Khosla et al., 2020]

Models	Span F1 score
MGC-LSTM-AXN-S	44.38
+ Fine-tune	44.56
+ Weight	45.04
+ Weight + Fine-tune	**45.45**

Table 2.12: Results for adding weights to the loss function and unsupervised fine-tuning [Khosla et al., 2020]

Dimov et al, 2020 *Dimovet al.* [2020] :

Approach :

Span Identification

The authors of the paper approach this task as a binary sequence labeling problem: 0 for non-propagandistic spans and 1 for propagandistic ones. And to make it more robust to text overlapping, they labeled the text at the sentence level but this makes it loses the connection between sentences.

To tackle this challenge, the researchers leveraged LaserTagger, a cutting-edge text-editing model from Google. This model excels in tasks like simplification, paraphrasing, and correcting grammatical errors. Notably, LaserTagger utilizes encoder activations directly, bypassing the need to learn encoder-decoder attention weights. This approach led to a strong baseline performance, with LaserTagger achieving an F1 score of 42% on the development set. Given its effectiveness, the researchers chose this model for further fine-tuning in subsequent experiments.

To enhance model performance, the authors implemented two techniques:

- This approach improves the performance of autoregressive models by mitigating error propagation. During training, the model is fed its own predictions (which might be inaccurate) alongside the ground truth labels. This strategy helps the model become more robust when making predictions during inference (when it only receives its own predictions). The teacher forcing rate is gradually decreased from 1 (using only correct labels) to 0 (using only the model's output) throughout training. However, this technique can significantly slow down the training process because it doesn't fully utilize the transformer's capability of processing entire sequences at once.

- This technique was implemented to address two issues: reducing the likelihood of false-positive predictions and mitigating minor inconsistencies in the data that might arise from preprocessing imperfections.

The final model comprised a single LaserTagger instance with a BERTBASE encoder, a single decoder layer with a hidden size of 128, and 4 attention heads. To optimize training efficiency, a batch size of 32 was effectively utilized by accumulating gradients every 2 steps. The Adam optimizer was employed with a learning rate of 2e-5, incorporating a learning rate warmup for the initial 10% of training steps. Additionally, teacher forcing was progressively reduced from 1 to 0 over the training process. This configuration achieved an F1 score of 46.1% on the development set and 44.6% on the test set.

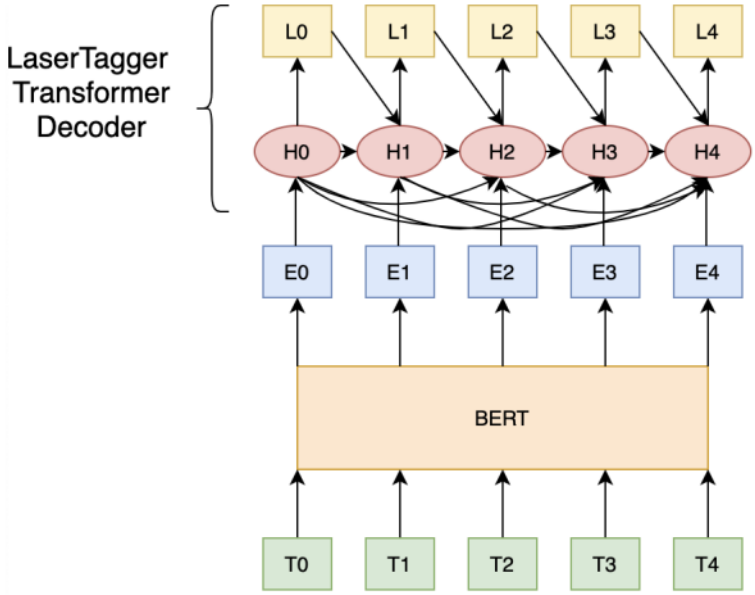


Figure 2.3: Architecture of the Laser Tagger model [Dimov et al., 2020]

- T represents Bert's embeddings, E are the outputs, H are hidden states and L are the final labels.

Technique Classification

To account for potential overlaps across sentences, all overlapping sentences were utilized to ensure accurate classification, thereby treating the problem as a multilabel classification task.

The authors utilized the ^ token to encapsulate propaganda spans. By employing either averaging or weighted sums, they aggregated the BERT representations of these marked tokens. In particular, they calculated the weighted sum for BERT representations $e\_i, \ldots, e\_i + k$ of propaganda tokens in the current span using learnable parameters. The weighted sum was determined as $\sum\_j = i^{i+k} \alpha\_j e\_j$, where α_j was obtained from the softmax function over the scalar product.

$$\alpha_j = \frac{\exp(b + (w, e_j))}{\sum_{j=i}^{i+k} \exp(b + (w, e_j))}$$

The α_j vector is combined with the [CLS] token output using bert to make final predictions. These methods are further enhanced by integrating BERT in-task fine-tuning, which has proven effective for this task due to its similarity to text classification.

Furthermore, they implemented a streamlined BERT model for technique classification by solely leveraging the embedding of the [CLS] token. All models were trained with a linear warmup constituting 10% of the training iterations, employing the Adam optimizer with a learning rate of 2e-5.

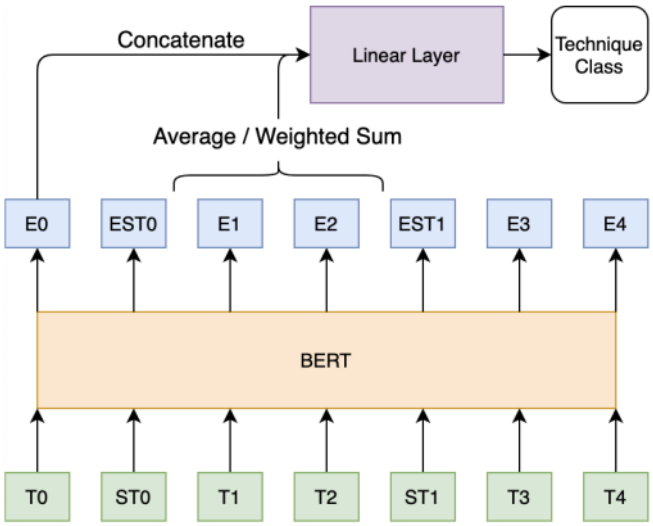


Figure 2.4: The architecture of the technique labeling model is based on R-BERT [Dimov et al., 2020]

- The token embeddings for BERT are represented by T_i, while ST_0 and ST_1 are special tokens used to highlight the propaganda span. The BERT outputs for E_i and EST_i are also included in the model.

Results :

Span extraction results

Models	F1 Score
biLSTM (GLOVE + charLSTM + postprocessing)	0.349
biLSTM (ELMO + postprocessing)	0.345
BERT (LINEAR)	0.408
BERT + LSTM + CRF	0.414
LaserTagger + teacher forcing	0.451
LaserTagger + teacher forcing + label smoothing	**0.461**
LaserTagger + teacher forcing + label smoothing (test)	0.446

Table 2.13: The results of the span extraction model on the development and test sets [Dimov et al., 2020]

Technique labeling results

Models	F1 Score (micro-averaged)
BERT (CLS)	0.573
R-BERT	0.592
R-BERT (intask finetuning)	0.584
R-BERT (weighted sum)	0.590
R-BERT (weighted sum + intask finetuning)	0.590
All models	**0.606**
All models (test)	0.582

Table 2.14: Propaganda technique labeling model results on development and test sets [Dimov et al., 2020]

2.3 Comparative study

In this section, we build on the previous discussion of machine learning-based methods, comparing these approaches from a broader perspective. We will examine the benefits of each technique and how they contribute to improving overall performance.

- **Granularity and Hierarchical Approaches**: Both the multi-granularity approach of Da San Martino et al. [2019] and the hierarchical model of Chen and Dhingra [2023] emphasize the importance of leveraging different levels of granularity in text analysis. This demonstrates that addressing the propaganda task at a finer level of granularity (token level) is more effective than considering the entire document (or sentence) at once.
- **Model Architecture**: Almost all the existing works use **encoder** pre-trained language models like BERT and RoBERTa which is common across these studies. The encoder models seem to better capture the nuances of propaganda techniques, particularly in complex texts.
- **Performance and Limitations**: Despite the advancements, each model exhibits limitations. For instance, while the multi-granularity model improves SLC (sentence level classification) performance, its overall F1 score remains modest. Similarly, the hierarchical MIMLRoBERTa model achieves higher accuracy but lacks detailed performance metrics across different propaganda techniques. GPT-4, despite being the most advanced language model, struggles with certain techniques, suggesting that there is space for improvement.
- **Ensemble Methods and Post-Processing**: The use of ensemble methods and careful post-processing, as seen in Dimov et al. [2020]'s work, highlights an effective strategy to improve model robustness and accuracy. These steps, often overlooked, can significantly enhance the final model's performance.

2.4 Conclusion

In summary, while significant progress has been made in propaganda detection through the use of different models and approaches, challenges remain in achieving consistently high performance across all propaganda techniques. For example, Decoder models were unexploited and not used sufficiently (only one work by Sprenkamp et al. [2023] to this day), especially with the rise of GPT-4 and Gemini where they can leverage their generative ability to explain even more the propaganda found.

Our future work could benefit from integrating these diverse approaches and filling out where the literature did not exploit sufficiently, perhaps combining the granularity and hierarchical strategies with more advanced language.

Part II

Contribution

Chapter 3

Propaganda Classification Model

3.1 Reminder of our objectives

As a reminder of our objectives, the primary objective is to build a machine learning model capable of classifying political texts to determine whether they contain propaganda and, if so, identify the specific propaganda technique used. The internship description initially mentioned that this model would be run on a dataset of 7.5 million collected Facebook posts. However, due to the numerous elections happening in 2024, our focus has shifted to applying our model to a different dataset centered on election-related content.

This shift introduces us to the second objective: collecting this new dataset. We have chosen YouTube as our target platform, compiling a dataset of video transcriptions related to US Senate elections and other US political events.

The third objective involves performing data analysis on the predictions generated by our model. We will apply the model to both a set of 58,000 Facebook posts (including both sponsored and organic content) and our newly collected YouTube video dataset. The analysis aims to address several questions, such as: Which propaganda technique is most commonly used? Which YouTube channel spreads the most propaganda content? ... etc. By achieving these objectives, we aim to gain insights into the prevalence and nature of propaganda in political texts across different platforms.

3.2 Datasets

To train our model effectively, we leveraged four datasets encompassing a wide range of propaganda techniques across various domains (e.g., COVID-19, education, politics) and content types (news articles, social media). A summary of these datasets is provided in Table 3.1, while Table 3.2 details the distribution of propaganda techniques within each dataset split. To improve the model's generalizability and ability to detect a broader range of techniques, we combined all the datasets, significantly increasing the sample size for the target labels (propaganda techniques).

Data	#N	Domain
ARGOTARIO	880	Dialogue
PROPAGANDA	5.1k	News
LOGIC	621	SocMed/News
COVID-19	477	News

Table 3.1: Summary of all datasets

3.2.1 Data sources

PROPAGANDA [*Da San Martinoet al.*, 2019]

Our primary dataset, PROPAGANDA introduced by Da San Martino et al. [2019], provides annotations for 18 propaganda techniques within news articles at both the sentence and token levels. To ensure sufficient training data for each technique, we focused only on 13 techniques that are both fallacious and frequently encountered in the dataset. These techniques include Loaded Language, Name Calling/Labeling, Exaggeration/Minimization, Doubt, Appeal to Fear/Prejudice, Flag-Waving, Causal Oversimplification, Appeal to Authority, and Black-and-White Fallacy.

ARGOTARIO [*Habernalet al.*, 2017]

The second dataset, ARGOTARIO, was introduced by Habernal et al. [2017]. It is designed for fallacy detection, where the task is to identify fallacies in given QA pairs. Their scheme includes five fallacy types: Red Herring, Ad Hominem, Irrelevant Authority, Appeal to Emotion and Hasty Generalization. For our propaganda technique classification, we only used the Irrelevant Authority samples as they correspond to the Appeal to Authority technique.

LOGIC [*Jinet al.*, 2022]

The third dataset, LOGIC, recently released by Jin et al. [2022], includes 13 logical fallacies found in educational websites. These fallacies are: , Ad Hominem, Fallacy of Extension, Ad Populum, Faulty Generalization, Appeal to Emotion, Equivocation, Fallacy of Relevance, False Dilemma, Fallacy of Credibility, Intentional Fallacy, Circular Claim, False Causality and Deductive Fallacy. The dataset features diverse text types such as dialogue and short statements.

COVID-19 [*Musiet al.*, 2022]

The final dataset, COVID-19 [Musi et al., 2022], focuses on fact-checked content related to COVID-19. To identify misinformation, the authors analyzed fact-checked social media posts and news articles, specifically looking for the presence of ten common fallacies.

These fallacies include Cherry Picking, Strawman, Red Herring, False Authority, Evading the Burden of Proof, Hasty Generalization, Post Hoc, False Cause, False Analogy, and Vagueness. By identifying these manipulative tactics, the authors aimed to expose misleading information circulating around COVID-19.

Since some fallacies overlap partially or fully across the four datasets, we merge these types to enhance our target labels (15 propaganda techniques) by unifying the names of the labels (e.g., False Cause and False Causality are renamed into → Causal Oversimplification, and Irrelevant Authority is renamed → Appeal to Authority) and we end up with this table 3.2 .

Fallacy	ARGOTARIO train/dev/test	PROPAGANDA train/dev/test	LOGIC train/dev/test	COVID-19 train/dev/test	Total train/dev/test	Total All
Causal Ov.simp.	-/-/-	111/28/34	303/49/36	36/10/10	450/87/80	617
Doubt	-/-/-	263/66/82	-/-/-	-/-/-	263/66/82	411
Exag/Mini	-/-/-	304/76/94	-/-/-	-/-/-	304/76/94	474
B&W Fallacy	-/-/-	60/16/19	192/40/25	-/-/-	252/56/44	352
Flag-Waving	-/-/-	145/37/45	-/-/-	-/-/-	145/37/45	227
Loaded Lang.	-/-/-	1,331/333/416	-/-/-	-/-/-	1,331/333/416	2,080
Appeal to Auth	92/24/29	57/15/17	96/18/33	26/8/8	271/65/87	423
Prejudice/Fear	-/-/-	131/33/41	-/-/-	-/-/-	131/33/41	205
Slogans	-/-/-	84/22/26	-/-/-	-/-/-	84/22/26	132
Reductio AH.	-/-/-	33/9/10	-/-/-	-/-/-	33/9/10	52
Name Calling	-/-/-	685/172/214	-/-/-	-/-/-	685/172/214	1,071
Red Herring	115/29/35	16/4/10	214/43/46	28/8/8	373/84/99	625
Strawman	-/-/-	4/1/6	-/-/-	28/8/8	32/9/14	91
Thought-Term.	-/-/-	48/12/14	-/-/-	-/-/-	48/12/14	74
Whataboutism	-/-/-	33/9/10	-/-/-	-/-/-	33/9/10	52
Total (splits)	207/53/64	3,305/833/ 1,065	805/150/ 140	118/34/ 34	4,435/1,070/1,303	
Total (All)	324	5203	1095	186	6,808	

Table 3.2: Count of propaganda techniques in each split across all datasets.

3.3 Decoder-LLM as Propaganda Classifier

3.3.1 Introduction

With the rise of Generative Large Language Models (LLMs) which are decoder-only models, and the lack of their use on a complex task such as propaganda techniques classification, we delved into this matter in this part of the report and we used Gemini, GPT-3.5, and GPT-4 as multi-label classifiers and evaluated them.

In Chapter 2, we discussed the Language model learning methods, which include fine-tuning and prompt learning (hard and soft prompts). When considering Gemini and GPT-3.5/4, we can exclude the fine-tuning option due to lack of resources and the soft-tuning option because we do not have access to the LLMs' code. Therefore, we relied solely on hard prompt design.

3.3.2 Conception

The main idea is to create a detailed prompt that guides the model toward understanding and correctly categorizing the input text into one of the propaganda techniques. Our approach is based on the annotation guide provided by experts for annotating the PROPAGANDA dataset [Da San Martino et al., 2019], as shown in Figure 3.1.

This guide consists of a schema to follow, which involves answering a series of questions about the input text. Each answer directs you further along the schema until you reach a leaf node that represents the identified propaganda technique in the text, or a node indicating that no propaganda technique is present.

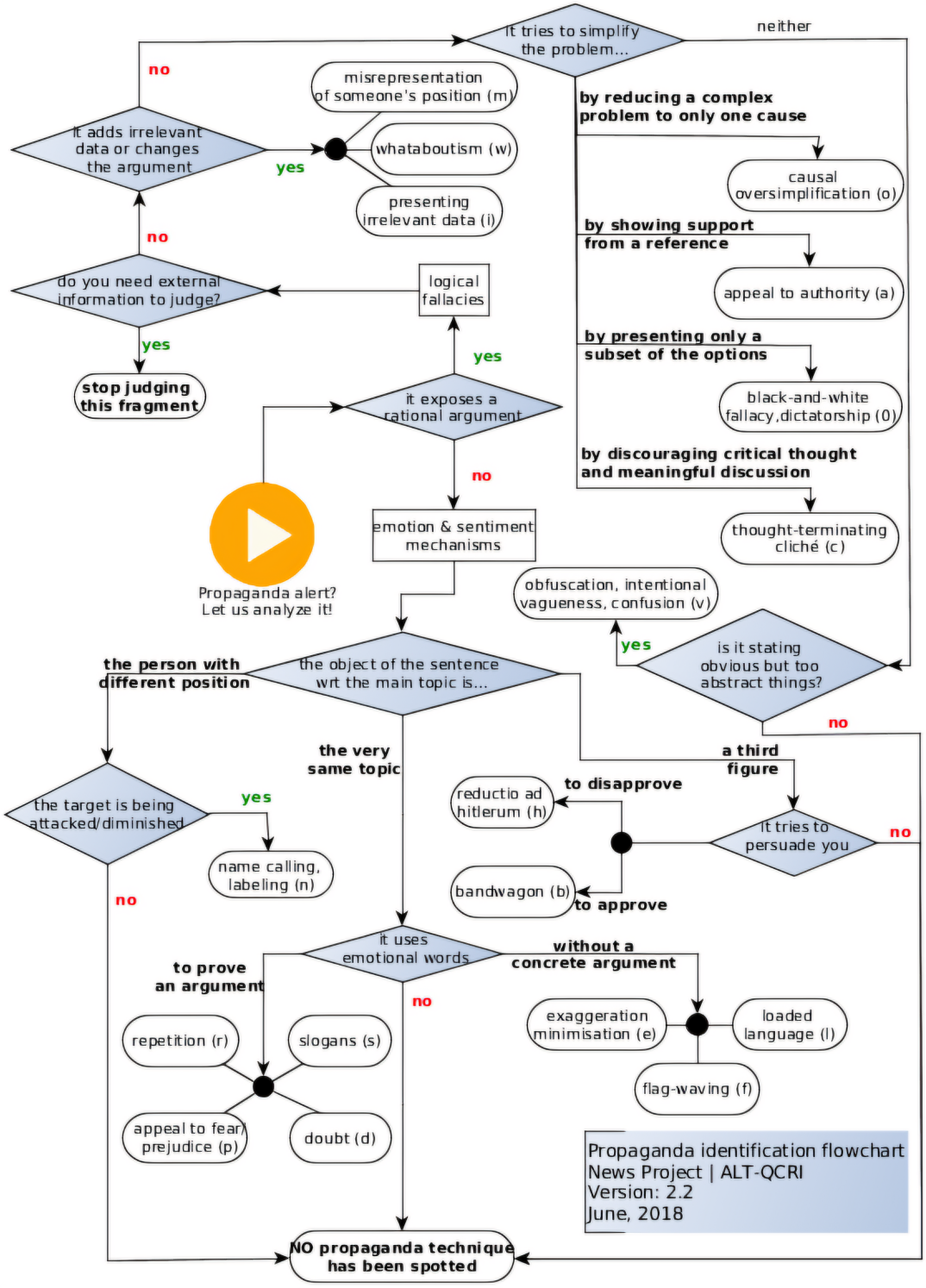


Figure 3.1: Annotation guide [Da San Martino et al., 2019]

We turned the annotation schema 3.1 into text and passed it to the LLM along with the input text to be classified, as shown in Figure 3.2.

```
prompt_instruction = f"""
You are a multi-label text classifier indetifying 14 propaganda techniques within news paper articles. You must apply this decision tree to determine the techniques used.
Decision tree :
Does it contain a rational argument?
  * Yes:
    * Does the argument include irrelevant data or is it otherwise changed?
      * Yes: then Whataboutism_Straw_Men_Red_Herring
      * No:
        * Does the text try to oversimplify a problem by any of the following:
    * Reducing the problem to only one cause : then Causal_Oversimplification
          * Show support from a reference : then Appeal_to_Authority
          * Present only a subset of options : then Black-and-White_Fallacy
          * Discourage critical thought : then Thought-terminating_Cliches
          * If yes to any of the above, then stop and return the techniques found.
          * If no to all of the above, then: No propaganda detected
  * No:
    * What is the object of the sentence in relation to the main topic?
      * The same topic:
        * Does the argument use emotional words?
          * Yes:
            * to prove an argument: choose the most suitables techniques based on their definitions, multiple techniques can be chosen:
          * Repetition : if it repeats the message over and over in the article so that the audience will accept it.
          * Slogans : if it is a brief and striking phrase that contains labeling and stereotyping.
          * Appeal_to_fear-prejudice : if it builds support for an idea by instilling anxiety and/or panic in the audience towards an alternative.
          * Doubt : if it questions the credibility of someone or something.
        * without a concrete example : choose the most suitables techniques based on their definitions, multiple techniques can be chosen:
          * Exaggeration_Minimisation : if either representing something in an excessive manner or making something seem less important than it actually.
          * Flag-Waving : if playing on strong national feeling (or with respect to a group, e.g., race, gender, political preference) to justify or promote an action or idea.
          * Loaded_Language : if uses specific phrases and words that carry strong emotional impact to affect the audience.
            * No: then No propaganda detected
        * A person with a different position:
          * Is the target being attacked or diminished?
            * Yes: then Name_Calling_Labeling
            * No: then No propaganda detected
        * A third figure:
          * Is the argument trying to persuade you of something?
            * Yes: then Bandwagon_Reductio_ad_hitlerum
            * No: then No propaganda detected
        """
```

Figure 3.2: Annotation guide as a prompt

We dropped four propaganda techniques, because they were very infrequent. Referring back to the work of Sprenkamp et al. [2023], some techniques were almost never detected by GPT-4 or GPT-3.5. These techniques are: Bandwagon, Reductio ad Hitlerum, Whataboutism, Straw Man, Red Herring, and Thought-Terminating Clichés. Additionally, we excluded Slogans because it was never detected (0.0% F1 score) by GPT-4 / GPT-3.5, regardless of whether the base or CoT prompt was used. So we settled on 10 propaganda techniques to be detected and the most frequent on news articles.

As shown in Figure 3.2, we start by assigning the LLM a new role as a multi-label classifier tasked with identifying 14 propaganda techniques within news articles. The LLM is instructed to follow the decision tree (annotation guide) to determine the propaganda techniques present in the text. The annotation guide is presented as a series of questions. Each question is designed to narrow down the specific propaganda technique. At the end of the prompt, we provide the text to be annotated and allow the model to generate the appropriate labels.

One problem we faced while doing so was the output text format. The output was a text and it was very hard to extract the labels from that text automatically and just looking for the name of that label in that text is not a proper solution because the text can contain at the same time “The text presents an Appeal to authority” and also can contain “Appeal to authority is not present in this text” so just searching the exact name is not a solution, and doing it manually is time-consuming.

The solution is to add how the output shape should be in the prompt like this :

```
prompt_tree = f"""
For the given article please state which of the 10 propaganda techniques are present. If no propaganda technique was identified return 'no propaganda detected'.
An example output would list the propaganda techniques with each technique in a new line e.g.:
*Loaded_Language
*Name_Calling,Labeling

Here is the article:
"""
```

Figure 3.3: Instruct the output shape.

3.3.3 Realisation

Building applications on top of ChatGPT will likely require us to make several parallel calls. Unfortunately, often the API will return a "too many requests" error. So we need a good way to deal with such errors while making several parallel calls.

In this section, we will cover these two important topics to efficiently perform calls to ChatGPT API:

1. Perform several calls in parallel
2. Retry calls in case they fail

1. Performing several calls in parallel

The code leverages Python's **asyncio** library along with **aiohttp** to perform multiple API calls concurrently. This approach significantly improves efficiency and reduces the total execution time when dealing with numerous requests.

- **'asyncio' :** A Python library used to write concurrent code using the async/await syntax.
- **'aiohttp' :** An asynchronous HTTP client/server framework that works seamlessly with asyncio to perform non-blocking HTTP requests.

Semaphore : a semaphore is used to control the number of concurrent tasks, preventing the system from being overwhelmed by too many simultaneous requests. If the max amount is reached it will block until some of the calls finish.

```
async def get_completion(content, session, semaphore, progress_log):
    async with semaphore:
        async with session.post("https://api.openai.com/v1/chat/completions
   ", headers=headers, json={
            "model": "gpt-4-turbo-preview",
            "messages": [{"role": "user", "content": content}],
            "temperature": 0
        }) as resp:

            response_json = await resp.json()
```

```
            progress_log.increment()
            print(response_json)

            return response_json["choices"][0]['message']["content"]
```

Listing 3.1: get_completion Function

- **Purpose :** Sends a POST request to the OpenAI API with the provided content and returns the generated completion.
- **Concurrency Control :** The function uses async with semaphore to ensure that only a specified number of requests are processed concurrently.
- **Session Management:** Utilizes an aiohttp session for making HTTP requests, which is more efficient than creating a new session for each request.

```
async def get_completion_list(content_list, max_parallel_calls, timeout):
    semaphore = asyncio.Semaphore(value=max_parallel_calls)
    progress_log = ProgressLog(len(content_list))

    async with aiohttp.ClientSession(timeout=aiohttp.ClientTimeout(timeout)
    ) as session:
        return await asyncio.gather(*[get_completion(content, session,
    semaphore, progress_log) for content in content_list])
```

Listing 3.2: get_completion_list Function

- **Purpose:** Manages multiple get_completion calls and aggregates their results.
- **Parameters:**
  - **content_list :** A list of news articles to be annotated.
  - **max_parallel_calls:** Maximum number of concurrent API calls allowed.
  - **timeout:** Timeout value for the HTTP requests.
- **Concurrency Execution:** Uses ***asyncio.gather*** to run multiple instances of get_completion concurrently while respecting the semaphore limit.
- **Progress Tracking:** An instance of ***ProgressLog*** is used to monitor the progress of the requests.

Execution Flow

1. **Semaphore Initialization :** A semaphore is created with a specified maximum number of parallel calls.
2. **Session Creation :** An aiohttp client session is established with the defined timeout.
3. **Concurrent Execution :** The get_completion function is called concurrently for each item in content_list using asyncio.gather.
4. **Result Aggregation :** The results from all concurrent calls are collected into a single list.

2. Retry calls in case they fail

To enhance robustness, the code implements a retry mechanism using the tenacity library. This ensures that transient errors do not cause the entire operation to fail and provides resilience against temporary issues such as network glitches or rate limiting by the API provider.

- **'tenacity' :** A Python library for retrying code when exceptions occur, offering flexible configurations for wait times, stop conditions, and more.

```
from tenacity import (
    retry,
    stop_after_attempt,
    wait_random_exponential,
)

@retry(wait=wait_random_exponential(min=60, max=80), stop=
    stop_after_attempt(2), before_sleep=print, retry_error_callback=lambda _
    : None)
async def get_completion(content, session, semaphore, progress_log):
    ...
```

Listing 3.3: Retry Decorator

- **@retry Decorator :** Wraps the get_completion function to automatically retry in case of exceptions.
- **Parameters:**
  - **wait=wait_random_exponential(min=60, max=80)** : Specifies an exponential backoff strategy with random wait times between 60 and 80 seconds before each retry.
  - **stop=stop_after_attempt(2) :** Limits the number of retry attempts to 2.
  - **before_sleep=print :** Prints a message before each retry sleep period, useful for logging and monitoring.
  - **retry_error_callback=lambda _: None :** Returns None if all retry attempts fail, allowing the program to continue execution.

Conclusion

The provided code efficiently performs multiple asynchronous API calls while incorporating a robust retry mechanism to handle potential failures. By leveraging ***asyncio***, ***aiohttp***, and ***tenacity***, the implementation achieves concurrency, efficiency, and resilience. Proper configuration and monitoring of the concurrency limits, timeout values, and retry parameters are crucial for optimizing performance and reliability in different operational environments.

3.3.4 Tests and Results

To test our prompt against the prompts of Sprenkamp et al. [2023], we used the same dataset "PROPAGANDA" by Da San Martino et al. [2019], detailed above in section 3.2.

Propaganda Technique	Gemini Base			Gemini COT			Gemini Tree		
	PR	RC	F1	PR	RC	F1	PR	RC	F1
Appeal_to_Authority	16.66%	4.34%	6.89%	40.00%	17.39%	24.24%	27.58%	34.78%	**30.76%**
Appeal_to_fear-prejudice	60.41%	64.44%	**62.36%**	61%	51.11%	55.42%	69.44%	55.55%	61.72%
Black-and-White_Fallacy	28.12%	69.23%	39.99%	17.85%	38.46%	24.38%	33.33%	58.84%	**42.55%**
Causal_Oversimplification	54.54%	27.27%	**36.36%**	18.18%	9.09%	12.12%	45.45%	22.72%	30.30%
Doubt	47.82%	33.33%	39.28%	44.44%	24.24%	31.37%	50.00%	54.54%	**52.17%**
Exaggeration,Minimisation	71%	11.11%	19.23%	44.44%	8.88%	14.80%	55.22%	22.22%	**31.69%**
Flag-Waving	56.25%	25.71%	35.29%	55.55%	14.28%	22.72%	69.23%	25.71%	**37.50%**
Loaded_Language	100.00%	31.39%	47.78%	100.00%	25.58%	40.74%	96.87%	36.04%	**52.53%**
Name_Calling,Labeling	81.08%	45.45%	58.25%	72.00%	27.27%	39.56%	72.22%	59.09%	**65.00%**
Repetition	100.00%	6.25%	11.76%	43.70%	14.58%	21.86%	93.33%	29.16%	**44.44%**

Table 3.3: F1 score, Precision, Recall for each Propaganda Technique using Gemini with 3 prompts (Base, Chain of thoughts and Tree)

Propaganda Technique	GPT 4 Base			GPT 4 COT			GPT 4 Tree		
	PR	RC	F1	PR	RC	F1	PR	RC	F1
Appeal_to_Authority	**50.00%**	21.73%	30.29%	42.85%	52.17%	**47.05%**	21.05%	17.39%	19.05%
Appeal_to_fear-prejudice	48.83%	93.33%	64.12%	48.31%	95.55%	64.17%	50.56%	100%	**67.16%**
Black-and-White_Fallacy	27.27%	46.15%	34.28%	15.62%	38.46%	22.22%	34.78%	61.53%	**44.44%**
Causal_Oversimplification	57.14%	18.18%	27.58%	28.33%	77.27%	41.46%	28.81%	77.27%	**41.97%**
Doubt	33.33%	69.69%	45.09%	34.40%	96.96%	50.78%	37.93%	100%	**55.00%**
Exaggeration,Minimisation	49.35%	84.44%	62.29%	53.73%	80.00%	**64.28%**	48.48%	71.11%	57.65%
Flag-Waving	53.84%	20.00%	29.17%	63.63%	20.00%	30.43%	78.94%	42.85%	**55.55%**
Loaded_Language	84.53%	95.34%	89.61%	93.82%	88.37%	91.01%	95.06%	89.53%	**92.21%**
Name_Calling,Labeling	72.60%	80.30%	**76.26%**	76.74%	50.00%	60.55%	78.00%	59.09%	67.24%
Repetition	100%	6.25%	11.76%	60.86%	29.16%	39.43%	67.85%	39.58%	**50.00%**

Table 3.4: F1 score, Precision, Recall for each Propaganda Technique using GPT-4 with 3 prompts (Base, Chain of thoughts and Tree)

From the Tables 3.3 and 3.3 above, we can conclude :

- Tree Prompt Superiority : Our tree prompt inspired by the annotation guide is generally superior to both the COT and Base prompts in both models (GPT-4 and Gemini). It enhances the precision and F1 scores for most propaganda techniques, indicating higher confidence and accuracy in predictions.

- For GPT-4, precision is notably higher using Tree prompt in several techniques which makes the model more confident and accurate in its predictions.
- For Gemini, Tree prompt often matches or surpasses the precision of Base and COT prompts, proving its effectiveness.

- Gemini vs GPT-4 : While GPT-4 with the Tree prompt shows overall better performance (higher F1 scores) compared to Gemini-Tree, the Gemini model often provides higher precision for several techniques.

- Precision is more important than F1 score : Given the importance of precision for our use case, where higher precision indicates more confident predictions, because a higher precision indicates that the model is more confident in its predictions, which is desirable because these labels will be used for analysis later. Conducting analysis on inaccurate data is not good idea. It is worth noting that focusing on precision won't capture all true labels, but only some of them. Therefore, if we accurately annotate only 5% of X number of articles, is way important than annotating all the correct ones but including much more False Positive. In summary, although we may sacrifice output size, this is not a concern given the vast amount of data available. Therefore, annotating just 1% of this data accurately is significant. The Tree prompt is particularly valuable in this matter.

3.3.5 Limitations

The limitations we faced while using these LLMs as multi-label classifiers are :

1. Output Format:

The output text sometimes does not only show the propaganda technique but also provides an explanation. While explanations can be helpful, they add unnecessary verbosity to the output, making it more challenging to extract the relevant classification labels directly.

2. Hallucinations:

The explanation provided by the LLM (limitation 1) are sometimes appears plausible but are incorrect. This phenomenon is called Hallucinations which refer to instances where the LLM generates content that appear reasonable but are factually incorrect. For example, the LLM might confidently classify a text snippet with a specific propaganda technique and even provide its own explanation for why this technique is used. However, this explanation could be misleading and cause users to believe the propaganda technique is present in the text when it isn't.

3. Span Extraction:

When the LLM is instructed to give the span of the text where propaganda is present, it often returns the entire text instead of the specific segment. This lack of precision in identifying the exact span reduces the usefulness of the output for tasks requiring detailed analysis.

4. Inability to Control the Threshold:

We do not have access to the probabilities of the propaganda techniques, which means we cannot control the precision and recall effectively. Because these models are decoder-only, we lack the ability to set thresholds that would balance false positives and false negatives according to the needs of the task.

5. Non-deterministic behaviour :

Given the same prompt and text, we observed that the identified propaganda techniques varied across different executions. Both OpenAI (GPT) and Google (Gemini) have promised to address this issue in the near future.

To measure this indeterminacy, we ran Gemini 1.5 and GPT-3.5 & 4 on 110 news articles from the PROPAGANDA dataset [Da San Martino et al., 2019] using the same prompt over five executions. We then calculated the intersection between the output results. The findings are presented in Table 3.5. For example, if the LLM identifies a technique during the first execution, we mark it as 1_{t_1} at time $t1$. If the technique is not found in the second execution, we mark it as 0_{t_2} at time t_2, and so on until t_5. The formula used to calculate the similarity is:

$$S = \frac{\max(\text{Count of 0s}, \text{Count of 1s})}{\text{Number of executions}}$$

Propaganda Techniques	Gemini 1.5	GPT-4	GPT-3.5
Appeal_to_Authority	80.00%	80.00%	60.00%
Bandwagon_ad_hitlerum, Reductio	80.00%	80.00%	60.00%
Appeal_to_fear-prejudice	100.00%	80.00%	60.00%
Exaggeration, Minimisation	80.00%	100.00%	80.00%
Black-and-White_Fallacy	100.00%	80.00%	80.00%
Causal_Oversimplification	80.00%	60.00%	40.00%
Doubt	60.00%	80.00%	40.00%
Flag-Waving	100.00%	60.00%	80.00%
Repetition	80.00%	60.00%	40.00%
Loaded_Language	100.00%	100.00%	100.00%
Thought-terminating_Cliches	80.00%	20.00%	40.00%
Name_Calling, Labeling	80.00%	60.00%	60.00%
Whataboutism, Straw_Men, Red_Herring	60.00%	40.00%	40.00%
Slogans	80.00%	60.00%	40.00%

Table 3.5: Deterministic score over 5 executions. **Note** : Bigger score does not reflect any better performance, maybe the LLM did not even find that technique at all during the 5 runs which lead to 100% score

The above table 3.5, lists the consistency percentages for each propaganda technique across the three models.

- **High Consistency**:

  - *Loaded Language :* Achieves 100% consistency across all three models, indicating strong reliability in identifying this technique.

  - *Appeal to fear-prejudice :* High consistency in Gemini 1.5 (100%) and reasonably high in GPT-4 (80%), but only 60% in GPT-3.5.

- **Variation in Consistency :**

  - Significant variation is observed in techniques like ***Causal Oversimplification*** and ***Doubt*** where consistency scores range from 40% to 80% across different models.

- *Thought-terminating Cliches* exhibited a wide range, with GPT-4 showing only 20% consistency compared to 80% for Gemini 1.5.

- **Model-Specific Trends :**

  - *Gemini 1.5 :* displays high consistency in most techniques, with several techniques identified with 80% or higher consistency. It shows the highest scores in ***Appeal to fear-prejudice***, ***Black-and-White Fallacy***, ***Flag-Waving*** .

  - *GPT-4 :* also shows high consistency for several techniques but slightly less so than Gemini 1.5.

  - *GPT-3.5 :* generally exhibits lower consistency across the board, with the highest score being 100% for ***Loaded Language*** only.

5. Limited number of API calls and Slow:

Even when performing asynchronous API calls, the prompts are still large and take a considerable amount of time to process compared to running a model locally. Additionally, all LLMs have a limited number of API calls. For instance, Gemini 1.5 limits requests to 120 per minute and recommends not exceeding 1 request per second. GPT operates slightly differently by counting the number of tokens per minute (TPM), capping them at 40,000 TPM. If we exceed this limit, additional usage incurs costs, which can be quite expensive.

6. Expensive :

GPT-4 and GPT-3.5 can be very expensive; in our experiments, we quickly reached 100 euros by annotating just 300 articles.

3.3.6 Conclusion

We explored the Decoder family of large language models (LLMs), which have not been extensively studied in the literature. We improved the work of Sprenkamp et al. [2023] by providing a more effective prompt (a tree-based prompt) and conducted a comprehensive study on the advantages, disadvantages, and limitations of using decoder-only LLMs.

In conclusion, prompt engineering is a time-consuming process to identify the optimal prompt, and fine-tuning is generally preferred for better adaptation to specific needs but we were constrained by insufficient computational power and the unsuitable format of the available data. Additionally, our experimentation showed that we had less flexibility with decoder-based models compared to encoder-based models. And this will make the subject of the next section where we build our Encoder-LLM Propaganda Classifier to overcome all the decoder-only limitations mentioned above.

3.4 Encoder-LLM as Propaganda Classifier

3.4.1 Why Encoder-LLM as a solution ?

All the limitations mentioned above, especially the inability to control the threshold, the limited number of API calls compared to our dataset size, and the considerable amount of time required to process data compared to running a model locally, made us reconsider the usage of such LLMs. Therefore, we opted to train our own Language Model to address these limitations:

- The architecture is an encoder-only model, allowing us to control the threshold. This means we can manage the precision, which is crucial for our use case since precision is more important than recall or the F1 score due to the need for accurate labels for later analysis.

- It will be significantly faster because we will run the model locally, eliminating the need for API calls.

- Precise span extraction compared to what we got with Gemini and GPT-4.

- Deterministic behavior.

- An output format that is very easy to manipulate.

3.4.2 Pre-processing the data

Datasets referred in section 3.2 , in its raw format, consisted of multiple .txt files named article_id.txt. Each article file was accompanied by an associated label file named labels_id.txt, which contained the labels.

Input Articles

Input Articles are in 3 folders, train-articles, dev-articles and test-articles. The title is on the first row, followed by an empty row. The content of the article starts from the third row, one sentence per line. Each article has been retrieved with the newspaper3k library and sentence splitting has been performed automatically with NLTK sentence splitter. Here is an example article :

[0]Manchin says Democrats acted like [34]babies[40] at the SOTU (video) Personal Liberty Poll Exercise your right to vote.

..... *Empty line* .....

Democrat West Virginia Sen. Joe Manchin says his colleagues' refusal to stand or applaud during President Donald Trump's State of the Union speech was disrespectful and a signal that [299]the party is more concerned with obstruction than it is with progress[368].

In a glaring sign of just how [400]stupid and petty[416] things have become in Washington these days, Manchin was invited on Fox News Tuesday morning to discuss how he was one of the only Democrats in the chamber for the State of the Union speech [607]not looking as though Trump [635]killed his grandma[653].

When others in his party declined to applaud even for the most uncontroversial of the president's remarks, Manchin did.

He even stood for the president when Trump entered the room, a customary show of respect for the office in which his colleagues declined to participate.

Ground-truth Labels

The format of a tab-separated like this :

id	technique	begin_offset	end_offset

where *id* is the identifier of the article, technique is one out of the 14 techniques, *begin_offset* is the character where the covered span begins (included) and *end_offset* is the character where the covered span ends (not included). Therefore, a span ranges from *begin_offset* to *end_offset-1*. The first character of an article has index 0. The number of lines in the file corresponds to the number of techniques spotted (for this task overlapping techniques are not merged). This is the ground truth file for the article above :

id	technique	begin_offset	end_offset
123456	Name_Calling,Labeling	34	40
123456	Black-and-White_Fallacy	299	368
123456	Loaded_Language	400	416
123456	Exaggeration,Minimization	607	653
123456	Loaded_Language	635	653

In our work, we didn't limit our focus solely to the span where the propaganda technique appeared, as is common in many literature works. Instead, we adopted the same

approach as Chen and Dhingra [2023], considering extra context both to the right and left of the target sentence. This is crucial because a simple data analysis revealed that in the training dataset, there are quite a few (8.71%) small text segments (less than 5 words) that are unrecognizable without their context, as depicted in Figure 3.4.

```
len(train[(train["sentence_length"]<5)&(train["Technique"]!='Not propaganda')]) /len(train[train["Technique"]!='Not propaganda'])*100
8.717494089834515
```

	text	label	Technique	sentence_length
95	Not from the masses.	P	Repetition	4
262	Hillary is still walking free.	P	Repetition	5
703	Powell would swiftly undo that.	P	Repetition	5
721	I do not know.	P	Repetition	4
758	They wrote that!	P	Repetition	3
814	We don't know.	P	Repetition	3
846	They said that.	P	Repetition	3

Figure 3.4: Non-comprehended target sentences without their context

Training a model solely on these sentences wouldn't make sense, as 'I do not know' isn't a form of propaganda. Therefore, our solution involves splitting every text article into chunks of 512 tokens with a rolling window of 256 tokens until the end of the text. This way, we start each new chunk with 256 tokens from the previous chunk. If two or more techniques are present within the same chunk, they are processed in the following manner (Figure 3.5) where the propaganda techniques are one-hot encoded and the problem is considered as multi-label classification. We also deleted links from the text.

```
multi[(multi['Appeal_to_Authority']==1)&(multi['Repetition']==1)].head(1)
```

	text	Appeal_to_Authority	Appeal_to_fear-prejudice	Black-and-White_Fallacy	Causal_Oversimplification	Doubt	Exaggeration,Minimisation	Flag-Waving	Loaded_Language	Name_Calling,Labeling	Obfuscation,Intentional_Vagueness,Confusion	Red_Herring
560	Hungarian Prime Minister Hits Nail On The Head...	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0

Figure 3.5: Multiple techniques within the same snippet

3.4.3 Conception

The idea here is to do the inverse of what is typically done in the literature, as shown in Figure 3.6. Traditionally, the whole text is first classified as propaganda or not, and then the specific technique used is identified. We want to reverse this process by initially treating the problem as a multi-label classification problem. Our approach involves identifying the propaganda techniques in the given text first; if no propaganda technique is found, then we can conclude that the text does not contain propaganda.

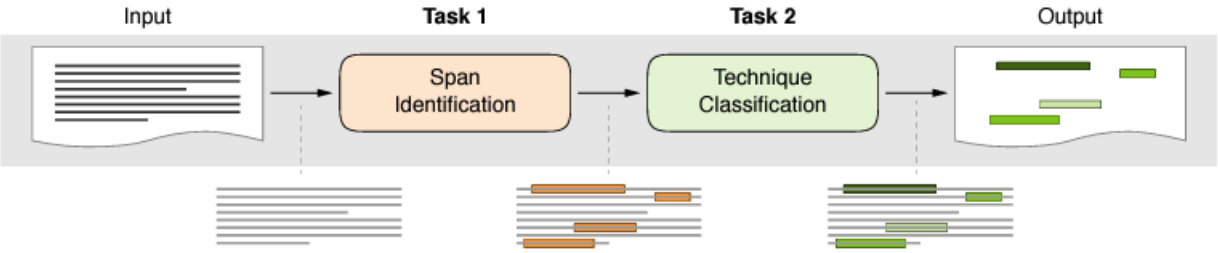


Figure 3.6: Literature existing workflow [Dimov et al., 2020]

Our work was influenced by a competition called Diplomats 2023 competition "Automatic Detection and Characterization of Propaganda Techniques from Diplomats of Major Powers" [1] . In this competition, researchers built systems to identify and categorize propaganda techniques used by diplomats from Europe, Russia, US and China in their English and Spanish tweets.

A unique aspect of their dataset was its organization. Tweets were grouped into four categories based on Aristotle's three principles of persuasion: ethos (appealing to credibility), pathos (appealing to emotions), and logos (appealing to logic).

The competition organizers proposed that digital propaganda can have two goals: "constructive" (influencing an opponent to make favorable decisions) and "destructive" (disrupting an opponent's decision-making). Based on this idea, they classified propaganda techniques as either "constructive" or "destructive" rhetoric.

Groups of techniques

Group 1: Appeal to Commonality

- Flag Waving.
- Causal Oversimplification.

Group 2: Discrediting the Opponent

- Appeal to fear-prejudice.
- Whataboutism.
- Name Calling or Labeling.
- Doubt.

[1] https://sites.google.com/view/dipromats2023/

Group 3: Loaded Language	Group 4: Appeal to Authority
• Loaded Language. • Black and White Fallacy. • Exaggeration Minimisation.	• Appeal to Authority. • Bandwagon.

Table 3.6: Groups of Propaganda Techniques

The overall solution is illustrated in Figure 3.7.Four specialized models are constructed, each serving as a multi-label classifier that outputs 1 or 0 for each technique it is designed to detect. If any model identifies the text as containing a specific propaganda technique, the final output will be 1, indicating that the text is propaganda.

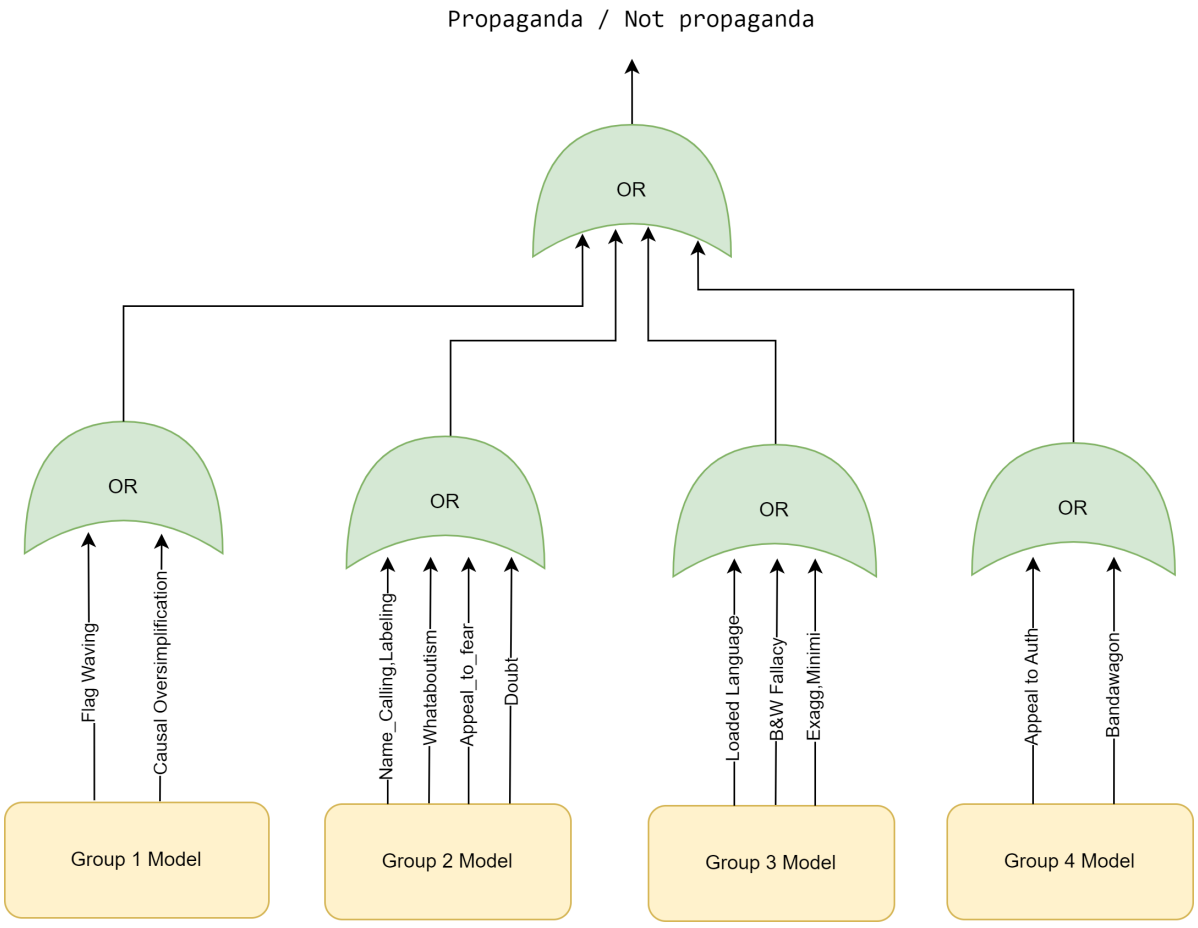


Figure 3.7: Overall Solution Architecture

Because we want a model that can identify the start and end points of propaganda, as shown in Figure 3.8, this task is called token classification in the NLP domain.

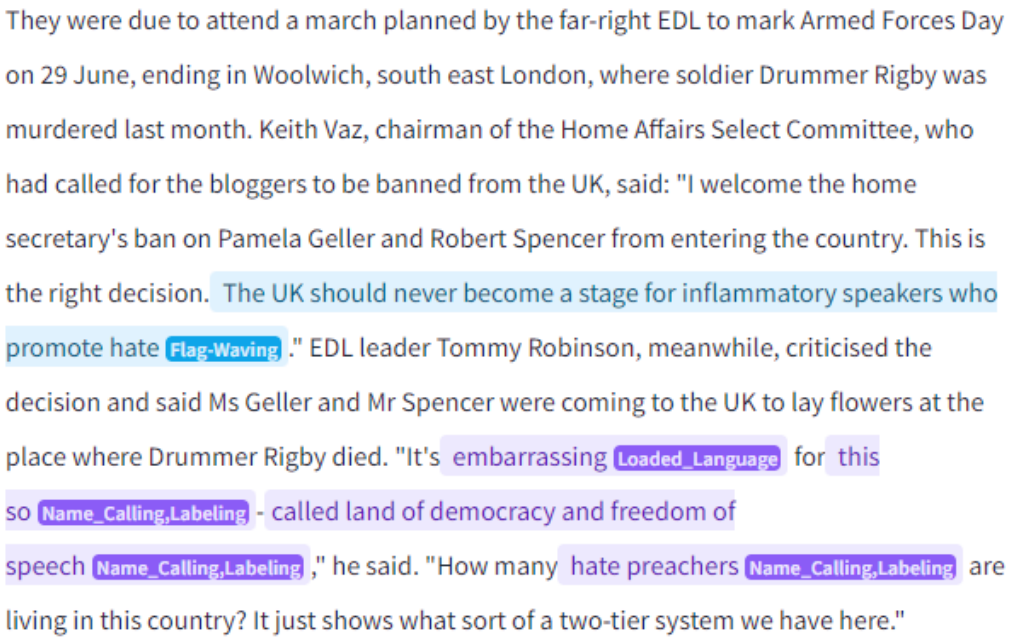


Figure 3.8: Output of the model

The model design is the same for all the 4 groups and only the data that is changed.

Backbone Model

The backbone model we utilized is DeBERTa-V3 from He et al. [2023] because, as stated in their paper, it outperforms RoBERTa and BERT on a majority of NLU tasks. The DeBERTa-V3 base model consists of 12 layers and a hidden size of 768. It comprises only 86M backbone parameters and a vocabulary containing 128K tokens, introducing 98M parameters in the embedding layer.

The choice of this model is driven by two main reasons: (i) We opted for an encoder-only model because we require access to the logits of each label to set a threshold. This capability is not available in models that have decoders because the probability given by the model represents the likelihood of the next token (i.e., the probability of the next word). If our label were tokenized into multiple tokens, the probability of that label loses its meaning in decoder models. This is why it is often noted in the literature that decoder models excel at text generation. (ii) Additionally, after experimenting with RoBERTa, BERT, and BERTweet, we found that DeBERTa-V3 exhibited superior performance compared to those models.

Classification Head

We used a neural network as a classification head with the number of outputs equal to the number of classes and the number of inputs set to 768 neurons (embedding size).

CRF Layer

One problem with the above setup is that each token is classified independently of the surrounding tokens: while these surrounding tokens are taken into account in the contextualized embeddings that DeBERTa-V3 produces, there is no modeling of the dependency between the predicted labels: for example, logically I-Loaded_Language cannot follow O, but DeBERTa does not model it.

Thus, we further added a linear-chain Conditional Random Field (CRF) as an additional layer, in order to model the dependency between the labels predicted for the individual tokens; this model can observe that the sequence "O → I-Loaded_Language" is never present in the training data [2], and thus it can assign a very low probability to the transition from an O tag to an I-Loaded_Language tag.

Loss Function : Negative Log-Likelihood

The CRF layer calculates the log-likelihood of the correct sequence of labels given the input sequence. This log-likelihood represents how likely the correct sequence is according to the model. During training, the model aims to maximize the log-likelihood of the correct label sequences. However, since most optimization algorithms (like gradient descent) are designed to minimize a loss function, the negative of the log-likelihood is used as the loss function.

Mathematical Formulation

Given an input sentence $X = (x_1, x_2, \ldots, x_T)$ of length T, and the corresponding ground-truth sequence of labels $Y = (y_1, y_2, \ldots, y_T)$, the CRF computes the score $s(X, Y)$ for a sequence of labels as :

$$s(X, Y) = \sum_{t=1}^{T} \text{emission}(x_t, y_t) + \sum_{t=1}^{T-1} \text{transition}(y_t, y_{t+1})$$

Where :

- $\text{emission}(x_t, y_t)$ is the logit of label y_t at position t for the token x_t .
- $\text{transition}(y_t, y_{t+1})$ is the transition score for moving from label y_t to label y_{t+1}.

The model then normalizes these scores over all possible label sequences Y' to compute the log-likelihood:

$$\log P(Y|X) = s(X, Y) - \log \sum_{Y'} \exp(s(X, Y'))$$

The **negative log-likelihood** (NLL) is then:

[2] This is due to the BIO labeling scheme, where 'O' denotes a token outside of any entity, 'B-' indicates the beginning of an entity, and 'I-' marks the inside of an entity. The transition from an 'O' tag to an 'I-' tag is not valid because an 'I-' tag must be preceded by an 'I-' or 'B-' tag.

$$\mathrm{NLL}(X, Y) = -\log P(Y|X)$$

This NLL is what is minimized during training.

In-depth Model Architecture

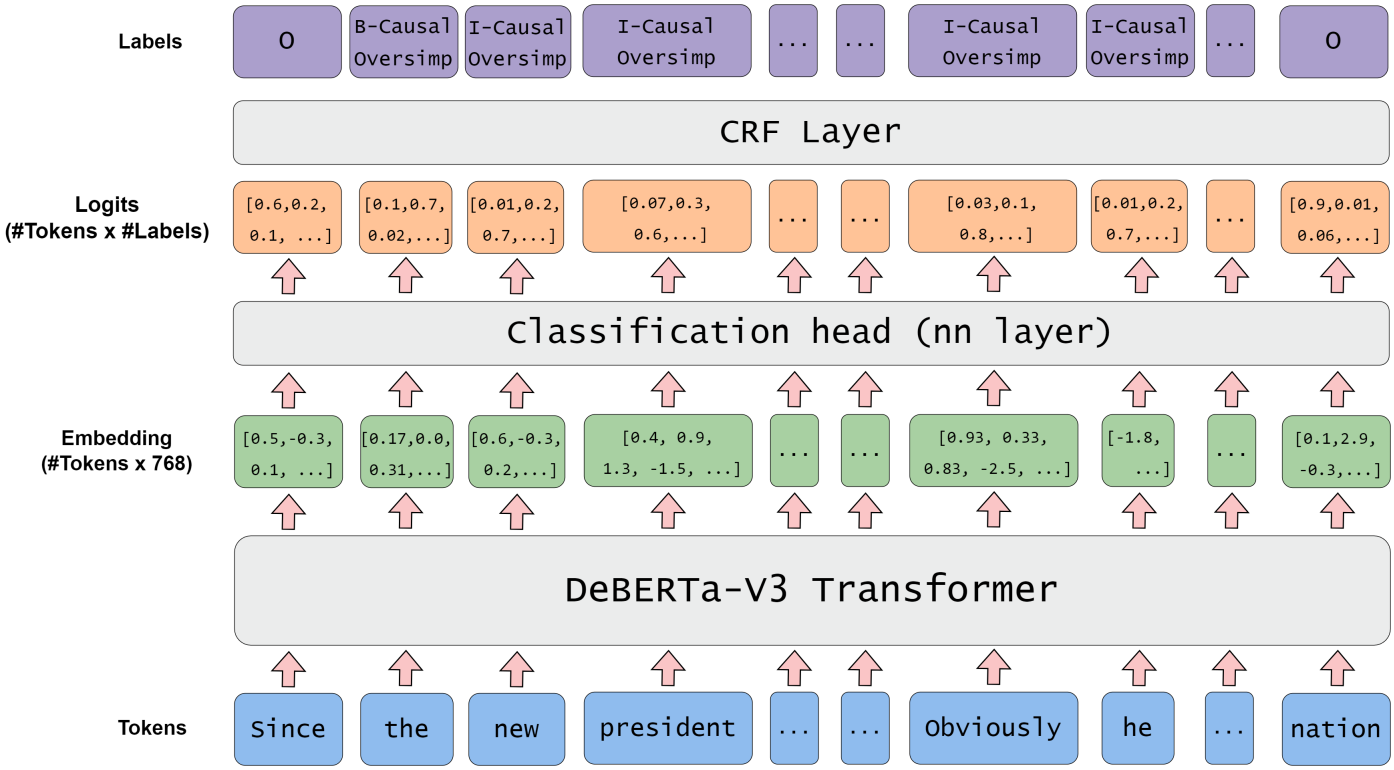


Figure 3.9: In-depth Model Architecture

3.4.4 Realisation

1. Load Dataset

We start by loading any group's dataset from our hugging face account [3]:

```
>>> data["train"][1]
{'id': '1',
 'tags': [0, 0, 0, 0, 0, 0, 0, 5, 6, 6, 6, 0, 0, 0, 0],
 'tokens': ["These", "two", "are", "among", "some", "of", "the", "most", "
   extreme", "anti-Muslim", "activists", "in", "the", "world", "."]
}
```

Listing 3.4: Sample Data from Group 3 Dataset

Instead of using numbers, we can replace them with actual labels describing the propaganda techniques. This makes it easier to understand the specific types of persuasion being used in the text.

```
>>> propaganda_list = data["train"].features[f"tags"].feature.names
>>> propaganda_list
[
"O",
"B-Loaded_Language",
"I-Loaded_Language",
"B-Black-and-White_Fallacy",
"I-Black-and-White_Fallacy",
"B-Exaggeration,Minimisation",
"I-Exaggeration,Minimisation"
]
```

Listing 3.5: Target labels of Group 3 Dataset

The letter that prefixes each **tags** indicates the token position of the propaganda technique:

- B- stands for "Begins Propaganda Technique." This tag identifies the first word or phrase that marks the start of a propaganda technique.
- I- stands for "Inside Propaganda Technique." This tag is assigned to any word or phrase that continues an existing propaganda technique but isn't the first word.
- 0 indicates "Neutral." This tag is used for any word or phrase that isn't part of a propaganda technique.

2. Tokenize the data

We'll use a Deberta-V3 tokenizer to prepare the text for processing. Even though the text appears to be broken down into words, it actually has not been. To ensure the tokenizer treats each word as a collection of smaller units (subwords), we will enable the is_split_into_words=True option. This prevents the tokenizer from mistakenly considering words as entirely new entities.

[3] https://huggingface.co/anismahmahi

```
>>> snippet = data['train'][1]
>>> processed_text = tokenizer(snippet["text"], is_split_into_words=True)
>>> list_tokens = tokenizer.convert_ids_to_tokens(processed_text["input_ids
    "])
>>> list_tokens
['[CLS]', 'These', 'two', 'are', 'among','some', 'of', 'the', 'most', '
    extreme', 'anti', '-', 'Muslim', 'activ, '##ists', 'in', 'the', 'world',
     '[SEP]'
]
```

Listing 3.6: Tokenization of an example

While tokenization breaks down text into smaller units, it can introduce some challenges. Specifically:

- Special Tokens: The tokenizer adds special tokens like ”[CLS]” and ”[SEP]” which aren't part of the original text. These tokens need to be ignored during training (assigned an id of -100) to avoid affecting the model's learning.
- Subword Tokenization: Sometimes, words are split into even smaller units called subwords. This can cause a mismatch with the original labels, where a single label might have been assigned to a whole word, but now applies to multiple subwords.

To address these issues, we need to realign the labels with the processed tokens:

1. We use the word_ids method to identify which subwords belong to the same original word.
2. We assign a label of -100 to the special tokens to prevent them from influencing the model's loss function (cross entropy loss in pytorch uses -100 to skip items).
3. We only assign the original label to the first subword representing a word. All other subwords from the same word are assigned -100.

This realignment process ensures the labels accurately reflect the meaning of the original text, even after tokenization.

```
def process_and_align_labels(text_data):
  processed_inputs = tokenizer(text_data["text"], truncation=True,
    is_split_into_words=True)

  word_labels = []
  for i, label in enumerate(text_data["label_tags"]):
    word_ids = processed_inputs.word_ids(batch_index=i)
    previous_word_index = None
    aligned_labels = []
    for word_index in word_ids:
      if word_index is None:
        aligned_labels.append(-100)
      elif word_index != previous_word_index:
        aligned_labels.append(label[word_index])
      else:
        aligned_labels.append(-100)
```

```
      previous_word_index = word_index
    word_labels.append(aligned_labels)

  processed_inputs["word_labels"] = word_labels
  return processed_inputs

processed_data = data.map(process_and_align_labels, batched=True)
```

Listing 3.7: Tokenize and align labels

After tokenizing and aligning the labels, now we need to pad the sentences to have them all with the same size and this is done simply with **DataCollatorForTokenClassification** from hugging face library which will add [PAD] token to the end of each sentence that did not make it to 512 tokens.

So by the end of tokenization, the final dataset would look like this :

```
>>> preprocessed_data["train"][0]
{'id': '0',
 'tags': [0, 0, 0, 0, 0, 0, 0, 5, 6, 6, 6, 0, 0, 0, 0, 0, 0, 0, ... ],
 'tokens': [ '[CLS]', 'These', 'two', 'are', 'among','some', 'of', 'the', '
   most', 'extreme', 'anti', '-', 'Muslim', 'activ, '##ists', 'in', 'the',
   'world', '[SEP]', [PAD],[PAD],[PAD],[PAD] ... x493 times
]
}
```

Listing 3.8: Final preprocessed Dataset

3. Model Training

After combining all described datasets in section 5.1 our hypothesis is that training a model on fallacy types from various datasets will enable it to identify general patterns that define these fallacies. This would be superior to the model simply memorizing characteristics specific to a single dataset.

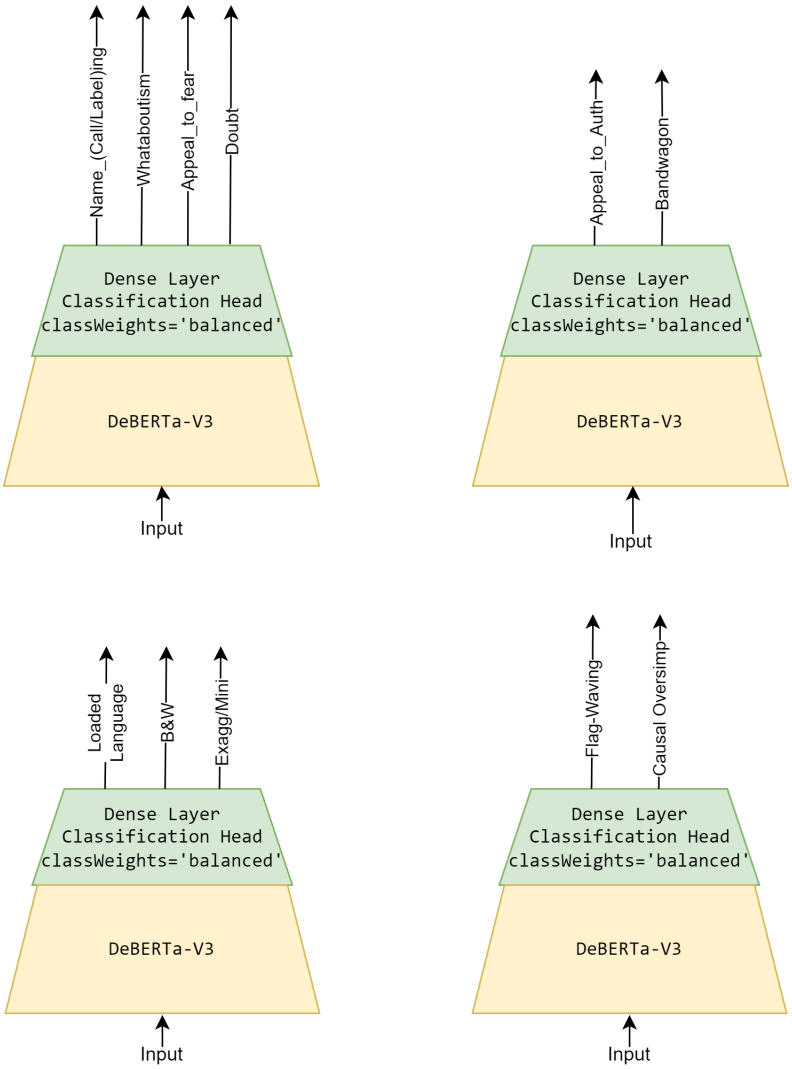


Figure 3.10: four groups models

Just to remember that our models are multi-label classifiers which means that a single text can contain multiple propaganda techniques, so when training each one of our models we make sure to set each token, of the other propaganda techniques that are not concerned to be detected by that model, to the token ”O”.

- **Sequence Length** : the maximum length of an input sequence is set to 512 tokens.
- **Batch Size** : training happens in batches of 16 examples.
- **Learning Rate & Epochs** : learning rate controls how much the model updates its weights during training, is set to 4e -5 for 13 epochs.
- **Freezing Layers** : the first 16 layers of the pre-trained model are frozen. This prevents the model from forgetting the general language knowledge it learned and focuses training on the final layers for the specific task.

3.5 Tests and results

In this section, we will present the metrics and evaluation of multiple state-of-the-art models against our own across three tasks: (1) propaganda techniques classification, referred to as multi-label classification; (2) propaganda vs. non-propaganda classification, referred to as binary classification; and (3) span extraction, which identifies from where to where the propaganda techniques appear in the text.

3.5.1 Multi-label classification results

In this section, we compare the precision, recall, and F1 score of each propaganda technique. We did not introduce the model by Chen and Dhingra [2023] here because it is designed only for binary classification. Instead, we introduced it in the next section (Binary Classification Results).

Propaganda Technique	GPT 4 Tree			Gemini Tree			QCRI			Our Model		
	PR	RC	F1	PR	RC	F1	PR	RC	F1	PR	RC	F1
Appeal_to_Authority	21.05%	17.39%	19.05%	**27.58%**	**34.78%**	**30.76%**	23.07%	13.04%	16.66%	16.98%	14.40%	15.58%
Appeal_to_fear-prejudice	50.56%	100%	67.16%	69.44%	55.55%	61.72%	61.27%	57.77%	59.47%	**73.20%**	**67.30%**	**70.13%**
Black-and-White_Fallacy	34.78%	61.53%	44.44%	33.33%	58.84%	42.55%	57.14%	30.76%	39.99%	**69.12%**	**33.00%**	**44.67%**
Causal_Oversimplification	28.81%	77.27%	41.97%	45.45%	22.72%	30.30%	37.50%	13.63%	19.99%	24.20%	18%	20.64%
Doubt	**37.93%**	**100%**	**55.00%**	50.00%	54.54%	52.17%	54.16%	39.39%	45.61%	**79.82%**	**36.40%**	**50.00%**
Exaggeration,Minimisation	**48.48%**	**71.11%**	**57.65%**	55.22%	22.22%	31.69%	73.55%	22.23%	34.14%	**70.00%**	**44.00%**	**54.04%**
Flag-Waving	78.94%	42.85%	55.55%	69.23%	25.71%	37.50%	**76.13%**	**82.28%**	**79.09%**	**78.40%**	**71.00%**	**74.52%**
Loaded_Language	**95.06%**	**89.53%**	**92.21%**	96.87%	36.04%	52.53%	**92.50%**	**86.04%**	**89.15%**	**99.20%**	**80.10%**	**88.63%**
Name_Calling,Labeling	78.00%	59.09%	67.24%	72.22%	59.09%	65.00%	74.18%	65.15%	69.37%	**86.10%**	**65.30%**	**74.27%**
Repetition	36.66%	63.07%	46.37%	**56.21%**	**57.42%**	**56.81%**	23.02%	15.72%	18.68%	21.78%	19.16%	20.39%

Table 3.7: F1 score, Precision, Recall for each Propaganda Technique using Gemini with 3 prompts (Base, Chain of thoughts and Tree)

	Gemini Tree		GPT-4 Tree		QCRI		Our Model	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
Precision	61.26%	57.56%	53.14%	51.02%	60.41%	57.25%	**70.13**%	**61.88**%
Recall	32.38%	42.69%	**76.28**%	**68.18**%	61.58%	42.60%	69.73 %	54.87%
F1-Score	42.36%	46.10%	62.64%	54.66%	60.98%	47.22%	**69.92**%	**58.16**%

Table 3.8: Macro and Micro Metrics (Precision, Recall and F1-score)

Table 3.8 compares the four models Gemini Tree, GPT-4 Tree, QCRI, and "Our Model" based on their Micro and Macro metrics for Precision, Recall, and F1-Score.

Figure 3.11: Micro Metrics Comparison

Understanding Micro vs. Macro Metrics

- Micro Metrics aggregate the contributions of all classes to compute the metric. It is typically more useful when we have class imbalance and want to ensure that every instance counts equally.
- Macro Metrics calculate the metric independently for each class and then take the average, treating all classes equally. This is useful when we want to assess performance across all classes without class-size bias.

Key Takeaways

1. Our Model has the highest **Micro Precision** (70.13%) and **Macro Precision** (61.88%), indicating that it is the most accurate model in terms of predicting correct positive instances across all classes and this is due to our ability to control the threshold which made us prioritizing precision over recall as explain previously.
2. Our Model consistently performs the best across most metrics, particularly in Precision and F1-Score, making it the most balanced and reliable model. Its strong performance in both Micro and Macro metrics suggests that it handles both individual classes and overall predictions well.
3. GPT-4 Tree might be more suitable in contexts where high recall is crucial, though it comes with a trade-off in precision.

3.5.2 Binary classification results

The given table 3.9, presents the results of identifying propaganda versus non-propaganda task. The performance of each model is evaluated using three key metrics, **NA** indicates that these metrics were not reported in the original papers by the authors.

Classifier	Precision	Recall	F1
Hier MIML$_{RoBERTa}$ [Chen and Dhingra, 2023]	NA	NA	62.79%
QCRI [Da San Martino et al., 2019]	60.41%	61.58%	60.98%
RoBERTa [Abdullah et al., 2022]	NA	NA	63.40%
GPT-4 Tree	53.14%	**76.28**%	62.64%
Gemini Tree	61.26%	32.38%	42.36%
Our Model	**70.13**%	69.73%	**69.92**%

Table 3.9: F1 score, Precision, Recall of propaganda vs non-propaganda task

- GPT-4 showed a strong Recall at 76.28%, the highest among all models, indicating it correctly identifies most positive cases. However, its Precision is relatively low.

- Our model stands out with the highest Precision (70.13%) and F1 score (69.92%) among all models, as well as a high Recall (69.73%). The results suggest that this model offers a good balance between precision and recall, making it the most effective model in this table for the given binary classification task.

3.5.3 Span Identification results

Our model is a token classification model, meaning we need to evaluate the quality of the extracted spans. A span is a piece of text that indicates the beginning and end of the propaganda. Evaluating spans can be challenging because strict string comparison is not always fair. For example, if the actual span and the span extracted by our model differ by only one word, it would be unfair to mark it as incorrect. To address this, we developed a function based on finding the **longest common consecutive sequence of words (LCCS)**, which is a more robust method than strict string comparison. We consider the extracted span and the actual span to be the same if the **length** of the longest common consecutive sequence is three words or more.

Example :

```
---------Appeal_to_fear-prejudice-----------
LCCS = ['Muslims', 'are', 'ready', 'to', 'seek', 'total']
Extracted = The Sharia is being implemented in the United Kingdom and the Muslims are ready to seek total control.
Actual = Muslims are ready to seek total control
-----------------------------------
```

Figure 3.12: Example of LCCS more than 3 words

if **LCCS has 2** words then : either the extracted or the actual span should be 2 words to consider it as True Positive (TP).

Example :

```
---------Loaded_Language------------
LCCS = ['sharia', 'compliance']
Extracted = It’s sharia compliance in New Mexico.
Actual = sharia compliance
------------------------------------
```

Figure 3.13: Example of LCCS equals to 2 words

if **LCCS has 1** word then : either the extracted or the actual span should be 1 word to consider it as True Positive (TP).

Example :

```
---------Loaded_Language------------
LCCS = ['betrayed']
Extracted = betrayed the British people
Actual = betrayed
------------------------------------
```

Figure 3.14: Example of ground-truth Span consisted of 1 word.

The most 2 important metrics in our study are the Precision and Recall.

- **Precision** : response the to question : from what the model extract, are those spans really correct ? how accurate our model is getting from the true spans.

- **Recall** : response the to question : from all the true spans, are the extracted spans really correct ? means how many true spans our model is getting true.

Special Observations :

- High recall but low precision : means the model is returning the whole text as that technique.

- High precision and low recall : means that model is just extracting spans that it is so confident they are true spans, or small snippets from the true span.

We exclude in this test, models by Abdullah et al. [2022] and Chen and Dhingra [2023] as they are sentence classifiers and not token classifiers which means they can only classify the whole text and not piece of text.

Propaganda Technique	GPT 4 Tree		Gemini Tree		QCRI		Our Model	
	PR	RC	PR	RC	PR	RC	PR	RC
Appeal_to_Authority	25.00%	6.00%	14.00%	2.00%	37.00%	28.00%	34.00%	20.00%
Appeal_to_fear-prejudice	32.00%	18.00%	17.00%	14.00%	35.00%	20.00%	70.00%	30.00%
Black-and-White_Fallacy	26.00%	20.00%	10.00%	8.00%	28.00%	22.00%	56.00%	35.00%
Causal_Oversimplification	38.00%	16.00%	45.45%	10.00%	68.00%	83.00%	60.00%	34.00%
Doubt	13.00%	11.00%	24.00%	14.00%	20.00%	21.00%	50.00%	25.00%
Exaggeration,Minimisation	17.00%	11.00%	50.00%	1.00%	25.00%	10.00%	65.00%	35.00%
Flag-Waving	21.00%	7.00%	0.00%	0.00%	75.00%	49.00%	82.00%	48.00%
Loaded_Language	33.00%	5.00%	18.00%	2.00%	65.00%	80.00%	70.00%	88.00%
Name_Calling,Labeling	25.00%	7.00%	16.00%	7.00%	58.00%	10.00%	60.00%	20.00%
Repetition	23.00%	5.00%	22.00%	1.00%	35.00%	17.00%	24.00%	15.00%

Table 3.10: Evaluating the quality of extracted spans for 4 models (Precision and Recall)

Key Takeaways

1. In this section, it is evident that encoder models (QCRI and Our model) outperform decoder models (GPT-4 and Geminir Tree), especially in cases where the decoder models are not fine-tuned on extracted spans. Unlike encoder models, where each token is classified individually.

2. The Gemini Tree model, in particular, exhibits very low recall rates across almost all techniques, often identifying only a small fraction of the correct spans.

3. Some techniques are more challenging to detect, with certain spans being harder to capture than others. For example:

   - **Appeal to Authority** : as expected, all models performed poorly on this technique.
   - **Repetition** : this technique also yielded poor results, likely due to the limited context window (512 tokens), which prevents the entire text from being processed, making it difficult to detect repetition across the entire document.

4. Although our model generally outperforms the others, we believe it is not yet ready for practical applications.

3.6 Conclusion

To summarize, we conducted an exploratory study of decoder-based Language Models (LLMs). We identified a significant gap in their application to complex tasks like propaganda technique classification. While we achieved improvements over existing work, some limitations rendered our solution impractical. We then transitioned to our second contribution, where we developed a token-level classification model using an encoder-based LLM that overcame the limitations of decoder-based LLMs. We detailed the implementation phase, including the technical aspects of our architecture, the tools and libraries used, data preprocessing steps, and model development.

Subsequently, we conducted extensive testing and evaluation of our system, comparing its performance with state-of-the-art models and our decoder models from the first contribution, and we were able to outperform them.

Chapter 4

Political Data Collection and Analysis

4.1 Introduction

With over 2 billion users worldwide, YouTube has become the essential platform for sharing video content. Whether for individuals, brands, content creators, or news outlets, the data on this platform can be extremely valuable for better understanding various patterns, which in our case can be propaganda in news. YouTube's massive reach and diverse user base make it an unparalleled medium for the dissemination and consumption of information. The platform's algorithms, which prioritize engagement, can significantly influence public opinion and the spread of news.

This chapter provides an overview of the techniques used to scrape data from YouTube, the difficulties encountered, and the initial analysis of the data that was collected. Scraping YouTube data involves extracting metadata, comments, view counts and other relevant information from videos. The process can be challenging due to YouTube's frequent updates to its API, rate limits, and the sheer volume of data. By analyzing this data, we aim to uncover propaganda dissemination, engagement metrics, and audience interaction.

In terms of what videos to scrape, we thought it would be best to collect content from American news networks because the senate elections are set for November 5, 2024. It is anticipated that the elections would be intensely contested and well followed, with multiple candidates from diverse political parties competing for seats in different states. The significance of these elections is heightened by their potential to shape the legislative direction of the country. Due to the increased level of political engagement and public interest, YouTube has developed into a vital forum for the regular sharing, debating, and dissemination of news about these elections.

4.2 Data Collection Conception

The primary objective of our study is to investigate and study the trends of propaganda in YouTube videos pertaining to the 2024 United States Senate elections. To accomplish this goal, we have outlined a three-step process to collect the necessary dataset.

1. The first step involves scraping the names of the candidates.
2. The second step is using the YouTube API to search for each candidate name as a keyword and retrieving the videos posted by the channels in the last year.
3. The final step is to obtain each channel from the dataset that has posted content about a senate candidate and then scrape all the videos that they have posted in the last year.

This will enable us to analyze the trends of propaganda across videos about senate candidates and videos about other topics more **effectively**.

4.2.1 List of candidates for the U.S. Senate election

Collecting the names of all the candidates running for the 2024 United States Senate elections can be done by accessing official election websites, databases, or other reliable sources that list the candidates. Ensuring the accuracy and completeness of this list is crucial, as it forms the foundation of our subsequent data collection efforts. Other features can be collected also such as the party that each candidates belong to and the state they elected from which can be really helpful in extracing more insights.

After searching, we found a website called Ballotpedia [1], which contains all the information we need to advance in our objective.

About Ballotpedia

According to their about us page, Ballotpedia is a non-profit online encyclopedia that specializes in American politics and is the leading source of unbiased information on elections, political candidates, and policy decisions. Their goal is to offer readers trustworthy, reputable, and easily navigable content that addresses every facet of American politics. The policy materials on Ballotpedia are also helpful in educating individuals about public policy since they offer succinct and understandable information on significant policy topics. They are committed to keeping all of their content impartial.

On the 2024 Senate elections page, we can see the element we need to get the data which is a table that contains the candidates per state and their party

[1] https://ballotpedia.org/United_States_Senate_elections_2024

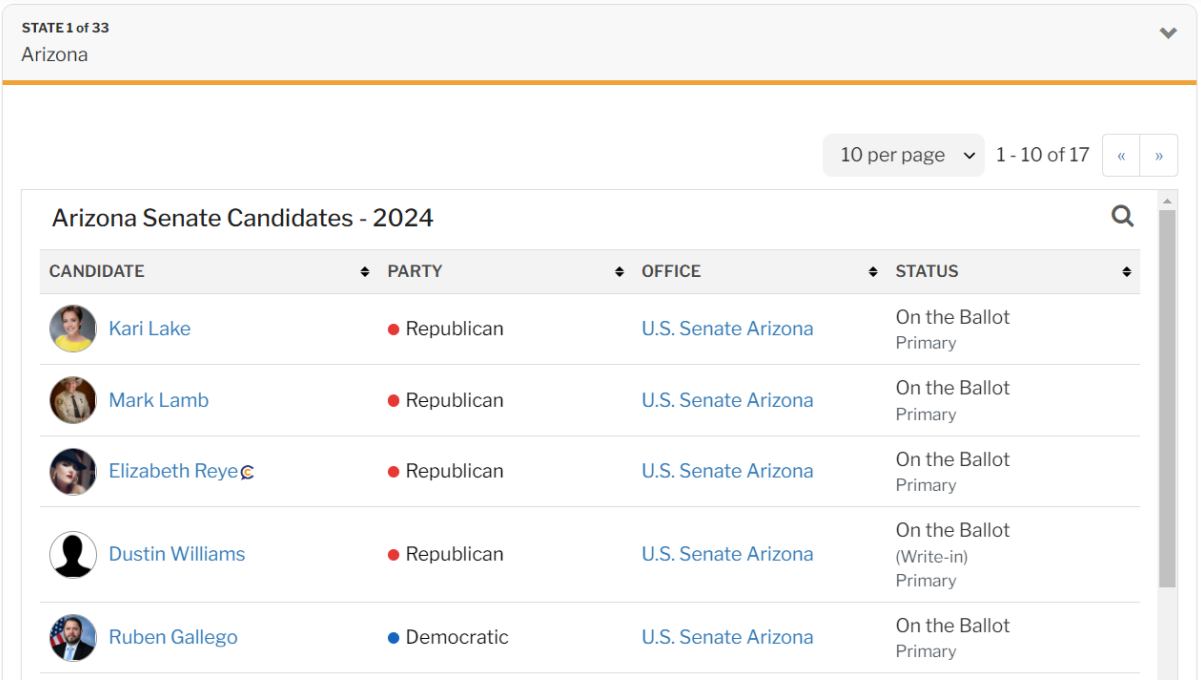


Figure 4.1: The table element containing data about senate candidates for 2024 (Ballotpedia https://shorturl.at/dzPVw)

To scrape this table, we needed the ID and class of the relevant HTML element. After inspecting the page's HTML code, we identified the element for the Arizona state candidates, which had the ID SenateArizona4375 and the class table. To collect the names of all candidates, we would need to iterate through each state. The tool we used to scrape this data will be explained in the next section (Realisation).

```
<table id="SenateArizona4375" class="table bp-table widget-table
table-responsive table-scrollable table-hover sortable F400 table
-responsive sortable jquery-tablesorter" data-enable-pagination=
"true" data-page-size="10" data-enable-search="true">
```

Figure 4.2: The HTML element containing the the senate candidates table

Collected data at step 1 :

After writing the script required to scrape the table shown above 4.2 (the names of the candidates, state and party). We ended up with a list of 450 candidates under this form:

Candidate	Political party	State
Kari Lake	Republican	Arizona
Mark Lamb	Republican	Arizona
	⋮	
Joe Manchin	Democratic	West Virginia

Table 4.1: Senate Candidates dataset

Among the total of 450 lines, there are 23 political parties and 33 states. The Republican Party leads with 211 candidates, followed by the Democrats with 142 candidates, and then 40 independent candidates. The remaining parties have fewer than 13 candidates each.

Regarding the states, California has the most candidates with 55, followed by Florida with 31, then 29 in Maryland and so on. With this data we can go to the second step of collecting the subtitles of the YouTube videos.

4.2.2 Collecting YT videos using candidates names as keywords

The second step is to use the YouTube to search for each candidate's name as a keyword. This involves querying the YouTube API or a scraping library to find videos that mention these candidates. We will focus on retrieving videos posted by various channels within the last year. This time-frame is chosen to capture the most relevant and recent content related to the upcoming elections. The retrieved data will include video metadata such as titles, descriptions, upload dates, view counts, and any other relevant information that can help in our analysis. And to do that we found a python library that suits our needs called YouTube-search-python.

Collected data at step 2 :

The YouTube-search-python library enables us to scrape the subtitles of YouTube videos by specifying these arguments in the script:

- ***Query :*** The search query to find relevant YouTube videos.
- ***region :*** Target country (default: 'US') - The region to filter videos from.
- ***language :*** Video's language (default: 'en') - The language of the videos.
- ***min_duration :*** Minimum duration (default: None) - Get videos with a minimum duration (in seconds).
- ***max_duration :*** Maximum duration (default: None) - Get videos with a maximum duration (in seconds).

- ***min_date :*** Minimum upload date (default: None) - The minimum upload date of videos (format: YYYY-MM-DD).
- ***max_date :*** Maximum upload date (default: None) - The maximum upload date of videos (format: YYYY-MM-DD).

After collecting the data by iterating through the list of candidates and adding 'senate elections' to each name then passing it as the query, next we set the region to 'US', language to 'en' (english) and min_date to (2023-05-01) to get all the videos over the span of one year. and then store all the metadata in a pandas dataframe (a dataframe is the main data structure in the pandas library constructed with rows and columns, similar to an Excel spreadsheet) which will make it easier to clean this data and store it in a csv. To clean the data, we have to remove the duplicates, remove the rows that contain empty values and if the name of the candidate is present in the subtitle of the video. Here is a model of the data collected:

title	**duration**	**link**	**channel**	**category**	**upload**
The Divison did ...	205	link_1	CBS Boston	News & Politics	2024-04-27
This year's Elections ...	455	link_2	12 News	News & Politics	2024-01-11

views	**transcript**	**candidate**	**state**	**party**	**propaganda**
393	Greater Boston starts ...	Kari Lake	Arizona	Republican	['Loaded-Language', 'Flag-Waving']
4522	Senate moves ...	Joe Manchin	California	Democratic	['Name-Calling']

Table 4.2: Youtube videos subtitles dataset

Where each column from left to right presents the title of the video, duration in seconds, the link to the video, the channel that posted it, the category of the video, the upload date, the number of views, the subtitles in the transcript column, the candidate mentioned in the video and the state and political party he belongs to.

Finally, we passed the transcript of each video to our Encoder-LLM model to extract the propaganda techniques used in that video. We mapped those predictions to a column called propaganda that is empty if no propaganda is found, or contains one or more elements from a list of 8 propaganda techniques ('Appeal_to_fear-prejudice','Black-and-White_Fallacy', 'Causal_Oversimplification', 'Doubt', 'Exaggeration,Minimisation', 'Flag-Waving', 'Loaded_Language', 'Name_Calling,Labeling').

The dataset contains 119.024 rows, the videos were posted by 223 different channels and they were labeled with 13 categories, News & Politics being the most frequent with 114.450 videos, followed by Entertainment with 2490 videos and then Sports with 1115 videos. And they were all uploaded between 02-05-2023 and 14-05-2024 and have a duration of 13 minutes and a view count of 36000 on average. After passing this data through

our model, 40.482 out of 119.024 videos (around 34%) have at least one propaganda technique used, the distribution of these techniques is as follows:

Propaganda technique	Number of videos	Percentage
Loaded Language	32828	27.58%
Name Calling,Labeling	21369	17.95%
Exaggeration,Minimisation	15940	13.39%
Appeal to fear-prejudice	12782	10.71%
Flag-Waving	12351	10.37%
Doubt	3050	2.56%
Black-and-White Fallacy	2655	2.23%
Causal Oversimplification	1999	1.67%

Table 4.3: Distribution of propaganda techniques in The Youtube Senate videos dataset

4.2.3 Collecting Videos from extracted Channels

In the final step, we will identify all the channels that have posted content about any Senate candidate from the previous dataset. For each of these channels, we will scrape all the videos they have uploaded in the past year, not just those directly related to the Senate candidates. This will provide a broader dataset, allowing us to compare and contrast the nature and frequency of propaganda in videos about Senate candidates versus other topics. By analyzing this comprehensive dataset, we aim to identify patterns, themes, and trends in how propaganda is disseminated and its potential impact on public perception. For the implementation we used the same Youtube-Search-python library.

Collected data at step 3 :

This dataset consists of 314,831 videos that cover a diverse range of topics, unlike the previous one which only contained fewer videos and focused on the senate elections. The videos are labeled with 13 categories, with 303,930 videos labeled as News & politics, followed by 5,528 labeled as Entertainment, and 2,554 as Sports, and so on. The videos were uploaded from 02-05-2023 to 14-05-2024 by 223 YouTube channels. After passing the subtitles of each video to the model presented in the first section, the distribution of the predicted propaganda techniques is presented in the table below.

Propaganda technique	count of videos	Percentage
Loaded Language	59268	18.82%
Name Calling,Labeling	34242	10.87%
Exaggeration,Minimisation	24938	7.92%
Appeal to fear-prejudice	20208	6.41%
Flag-Waving	19128	6.07%
Doubt	3999	1.27%
Black-and-White Fallacy	3493	1.10%
Causal Oversimplification	2673	0.84%

Table 4.4: Distribution of propaganda techniques in The Youtube videos by channels dataset

The data provided shows that the proportion of each propaganda method in the new 300k dataset, which includes 100k videos about senate candidates, has decreased significantly compared to the first 100k videos about senate candidates. This suggests that videos about senate candidates contain more propaganda than videos about other topics. To emphasize this distinction, we removed the 100k videos from the dataset and calculated the percentages of propaganda techniques. The results are presented in the two tables below (note that VAS stands for videos about senate candidates and VAOT signifies videos about other topics)

Propaganda technique	count of VAS	count of VAOT
Loaded Language	32828	26440
Name Calling,Labeling	21369	12873
Exaggeration,Minimisation	15940	8998
Appeal to fear-prejudice	12782	7426
Flag-Waving	12351	6777
Doubt	3050	949
Black-and-White Fallacy	2655	838
Causal Oversimplification	1999	674

Table 4.5: comparaison of propaganda techniques distribution between subtitles in videos about senates and videos about other topics

Propaganda technique	percentage in VAS	percentage in VAOT
Loaded Language	27.58%	13.50%
Name Calling,Labeling	17.95%	6.57%
Exaggeration,Minimisation	13.39%	4.59%
Appeal to fear-prejudice	10.71%	3.79%
Flag-Waving	10.37%	3.46%
Doubt	2.56%	0.48%
Black-and-White Fallacy	2.23%	0.42%
Causal Oversimplification	1.67%	0.34%

Table 4.6: Comparaison between the percentages of propaganda techniques in both VAS and VAOT

As the data shows, the proportion of each propaganda technique in videos concerning the senates is almost three to four times greater than in videos addressing other subjects, despite the latter having only half the number of videos. This presents a promising prospect for the next section, where we will delve deeper into the dataset and insights at our disposal.

4.3 Tools and environment

BeautifulSoup

Python is widely used for web scraping due to its capacity to handle most processes effortlessly and its extensive range of libraries designed specifically for this purpose. One such popular open-source framework for web scraping is Scrapy, written in Python. It is an ideal choice for web scraping and data extraction through APIs. Another Python library, Beautiful Soup, which we will be utilizing, is highly suitable for web scraping. It creates a parse tree that can be used to extract data from HTML on a website. Additionally, Beautiful Soup offers numerous features for navigation, searching, and modifying these parse trees.

Pandas

Pandas is a powerful and versatile open-source data manipulation and analysis library for Python, designed to facilitate data analysis tasks. It offers data structures like DataFrames and Series, which are ideal for handling and analyzing structured data. With Pandas, users can efficiently handle large datasets, perform complex data transformations, and manipulate data in a way that is both intuitive and highly flexible. This library excels in tasks such as data cleaning, merging, reshaping, and aggregating, making it an essential tool for data scientists, analysts, and engineers.

YouTube-search-python Library

YouTube-search-Python is a library developed to simplify the process of searching for YouTube videos and obtaining relevant information without the need for direct interaction with the YouTube Data API. This tool allows developers to integrate YouTube search functionality into their applications, making it easier to find YouTube content with code. The key features of this library are:

- **Easy Search:** Perform YouTube searches and retrieve video results with minimal code without limitations on number of pages.
- **Metadata Retrieval:** Access essential video information such as titles, URLs, view counts, subtitles and more.
- **Simplified API:** Avoid the complexities of handling API keys and quotas associated with the YouTube Data API.

In this final chapter, we work with two main datasets. The first dataset includes politically sponsored and non-sponsored (organic) Facebook posts, while the second consists of the collected YouTube subtitles, which were introduced in the previous chapter. After applying our model to these datasets, we will extract insights to better understand how propaganda influences various features, such as video duration, view count, and more.

4.4 Dashboard

To effectively present our analysis, we will use a Streamlit dashboard. This interactive interface will group all insights with accompanying graphs and explanations, making our work accessible to anyone, even without a background in computer science or NLP.

The web application's user interface is simple, featuring a sidebar on the left with a dropdown menu for selecting datasets. Below the menu, each dataset has multiple buttons that allow users to navigate through the different analysis done on that dataset.

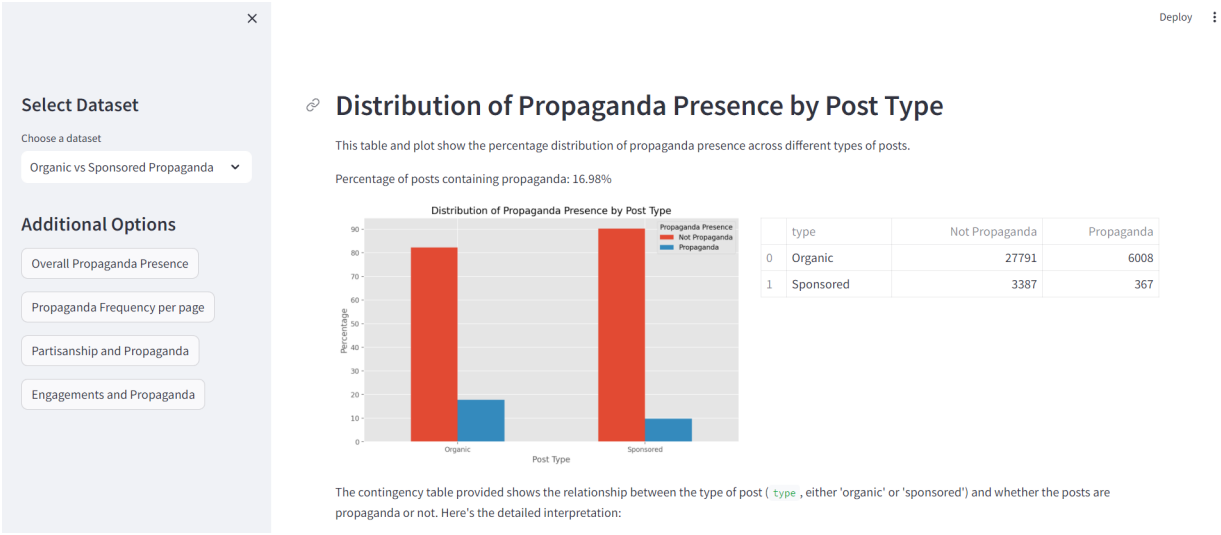


Figure 4.3: The user interface of the dashboard

4.4.1 Development Environment

Matplotlib
Matplotlib is a popular plotting library in Python used for creating static, interactive, and animated visualizations. It can generate a wide range of plots that integrate well with other Python libraries. We chose Matplotlib because it supports interactive plots that can be embedded in various applications and environments, like Jupyter notebooks and Streamlit, which we will use to make our dashboard.

SciPy
SciPy is an open-source Python library used for scientific and technical computing. It builds on top of NumPy by adding a large collection of algorithms and functions for various mathematical, scientific, and engineering disciplines. It includes functions for Linear Algebra, Optimization, and Signal and Image Processing. Most importantly, it has functions for statistical analysis, including probability distributions, hypothesis testing, and descriptive statistics, which we will be using in our study.

Seaborn
Seaborn is a Python data visualization library built on top of Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn offers a wide range of plot types. Seaborn makes it easy to create visually appealing and informative plots with minimal code. It also provides features for customizing plots. It is also easy to use and integrates well with Pandas, making it a good choice for statistical plotting, which makes it perfect for our study.

Streamlit
Streamlit is an open-source Python framework used to create web applications specifically for data science and machine learning projects. It allows you to turn Python scripts into interactive web apps quickly and easily, without requiring any front-end web development skills.

4.5 Sponsored vs Organic Facebook Posts Dataset

This is a Facebook News-Related Posts Dataset, a rich dataset collected from 472 U.S.-based users between November 2020 and February 2021. The data was obtained using the **CheckMyNews** browser extension and focuses on news-related posts on Facebook.

This dataset is particularly valuable for studying the overall presence of propaganda across different types of posts (organic / sponsored), propaganda frequency per page, study the distribution of propaganda techniques by partisanship, which party uses the most sponsoring to spread propaganda. ... etc.

4.5.1 Dataset Composition

The dataset categorizes Facebook news-related posts into four distinct types, based on how they appear in users' feeds:

1. **Targeted News Exposure** :
    - **Description**: Posts shown as part of targeted advertising campaigns based on user characteristics or interests.
    - **Source**: Advertisers use targeting mechanisms to deliver these posts to specific audiences.
    - **Volume**: 11,566 posts.
2. **Incidental News Exposure** :
    - **Description**: Posts received by users without actively following the news source or being specifically targeted by Facebook.
    - **Source**: Includes posts shared by friends, groups, or random pages that the user follows.
    - **Volume**: 60,529 posts.

4.5.2 Data Structure

The dataset is organized into 2 compressed files, each corresponding to one of the categories mentioned above. Each file contains a CSV file, structured with several columns capturing various aspects of the news-related posts. These columns include:

- ***timestamp :*** Millisecond-precise time when the post was received.
- ***fb_advertiser_fb_id :*** Facebook identifier of the page that published the post.
- ***fb_advertiser_fb_page :*** URL of the page that published the post.
- ***text :*** Text content of the post.
- ***has_landing_url :*** Indicates if the post contains an external URL.

- ***landing_domain :*** Domain of the URL included in the post.
- ***time_on_post :*** Time the user spent on the post (in milliseconds).
- ***action_on_post :*** Indicates if the user interacted with the post (liking, commenting, sharing, etc.).
- ***visible_action_on_post :*** Specifies if the interaction was visible (liking, commenting, sharing).
- ***unvisible_action_on_post :*** Indicates non-visible interactions (clicking on an image, visiting a URL).
- ***visit_landing :*** Indicates if the user visited the external URL or not.
- ***shared :*** Indicates if the user shared the post or not.
- ***commented :*** Indicates if the user commented on the post or not.
- ***partisanship :*** An estimated evaluation of the political bias of the post, based on assessments from Media Bias Fact Check and News Guard.

4.5.3 Analysis

1. Distribution of Propaganda Presence by Post Type

The given table 4.7 and plot 4.4 show the percentage distribution of propaganda presence across different types of posts.

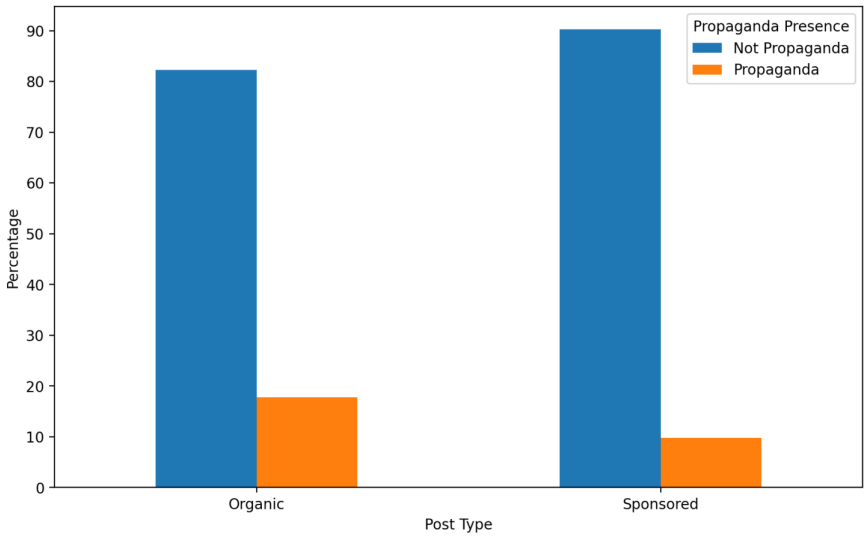


Figure 4.4: Distribution of Propaganda Presence by Post Type (Organic / Sponsored)

Type	Not Propaganda	Propaganda
Organic	27791	6008
Sponsored	3387	367

Table 4.7: Contingency table of the type of post and propaganda presence

The contingency table provided shows the relationship between the type of post ('type', either 'organic' or 'sponsored') and whether the posts are propaganda or not. Here's the detailed interpretation:

- **Organic Posts**:
  - 27,791 posts are not propaganda.
  - 6,008 posts are propaganda.
- **Sponsored Posts** :
  - 3,387 posts are not propaganda.
  - 367 posts are propaganda.

Chi-square Test Results :

- P-value : 4.16e-35, which is far below the significance level (alpha = 0.05) indicating a significant association between whether a post is organic or sponsored and whether it is propaganda.. **Relative Proportions :**

- **Organic Posts**:
  - Percentage of propaganda posts: (3,387 / (27,791, 3,387)) * 100 17.77%.
- **Sponsored Posts** :
  - Percentage of propaganda posts: (367 / (3,387 + 367)) * 100 9.77%.

Conclusion : organic posts have a higher proportion of propaganda (17.77%) compared to sponsored posts (9.77%). This suggests that organic posts are more likely to be propaganda than sponsored posts. This could be due to Facebook ads system and who create that organic and sponsored content.

2. Propaganda Frequency by Facebook Page

To conduct a more in-depth analysis, we focused on individual Facebook pages to determine which ones share the most and the least propaganda, identify those that use sponsorship most frequently to promote propaganda, and examine the content of propagandist posts.

In this study, we only included pages with at least 20 posts. This decision was made because many pages with just a single post were labeled as propaganda by our model,

skewing the results when sorting pages by the percentage of propagandist content. To ensure a more accurate analysis, we limited our focus to more active pages.

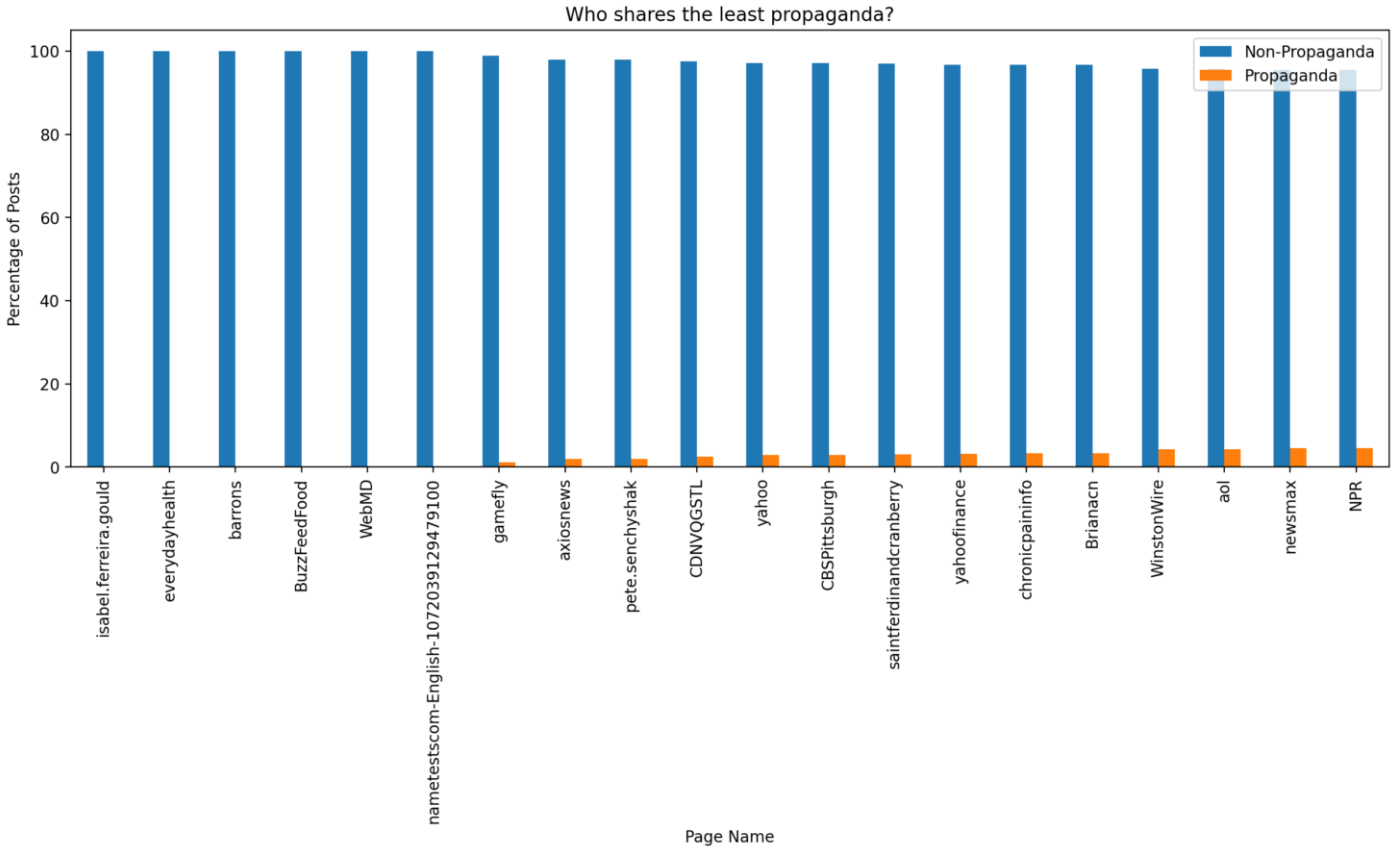


Figure 4.5: Who shares the propaganda the least

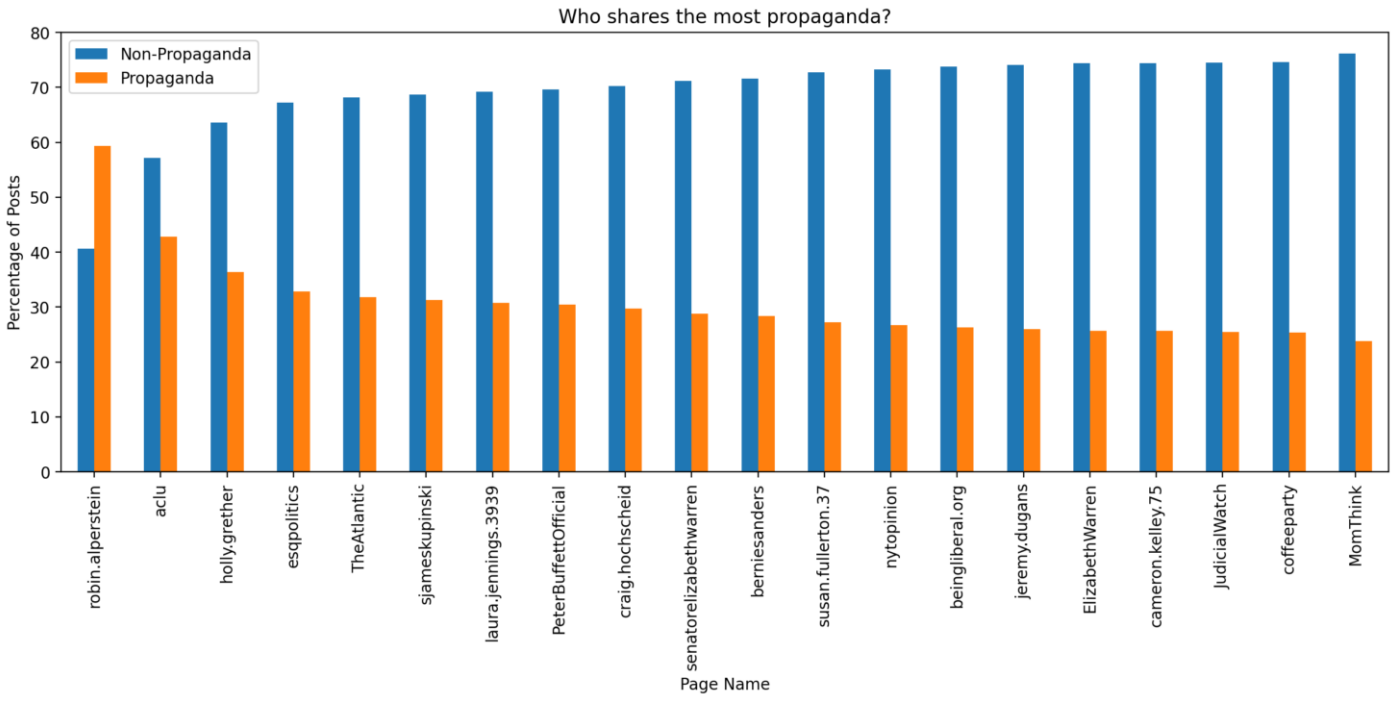


Figure 4.6: Who shares the propaganda the most

The plot 4.6 shows that the page associated to the id robin.alperstein is on top of who shares propaganda the most with almost 60% of her posts contains propaganda.

With a little research on the internet, we discovered that Robin Alperstein is a lawyer and an attorney in private practice in New York City. She concentrates her practice in complex commercial litigation, handling disputes in federal and state courts, and known for her strong opposition to Trump and Republicans. With that being said we decided to do a topic modeling on her facebook posts that were considered as propaganda.

Topic modeling is a technique used in natural language processing (NLP) and machine learning to discover the underlying themes or topics within a collection of text documents. It helps identify patterns, relationships, and common themes among words or phrases in a corpus.

For topic modeling, we used a well-known model called Top2Vec, along with some preprocessing of the post texts. First, we converted all text to lowercase and removed stopwords (using NLTK stopwords) and links. The processed text was then passed to the Top2Vec model.

Topic Modeling results

The topic modeling identified five distinct topics from the Facebook posts, with the following key terms associated with each topic:

- **Topic 1 :** white, corruption, yesterday, just, endless, abuse, power, obstruction, man, nearly.
- **Topic 2 :** country, election, use, republicans, going, headline, democracy, lies
- **Topic 3 :** people, actually, know, court, read, make, discrimination, trump, yesterday
- **Topic 4 :** lies, force, trump, nearly, man, obstruction, gop, life, absolute, democracy
- **Topic 5 :** trump, article, does, abuse, republicans, mcconnell, political, power, going, read

Interpretation :

These topics indicate a strong focus on political corruption, elections, legal battles, and the roles of specific individuals or groups, particularly Republicans and Trump. The loaded language in these posts is used to convey strong opinions against these subjects

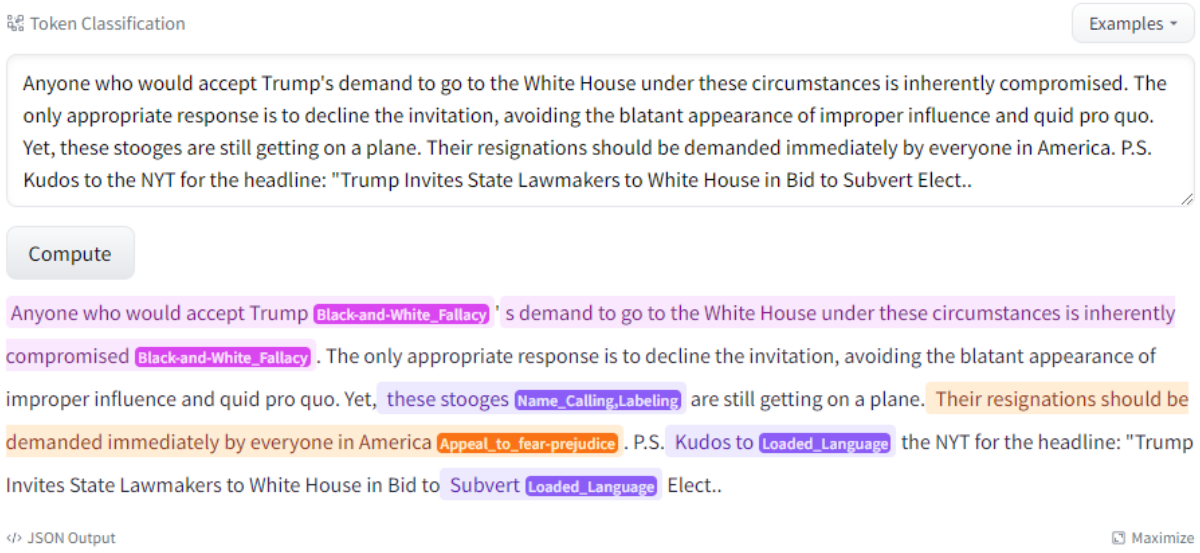


Figure 4.7: Output by our model showing Propaganda Techniques used in Robin Alperstein post

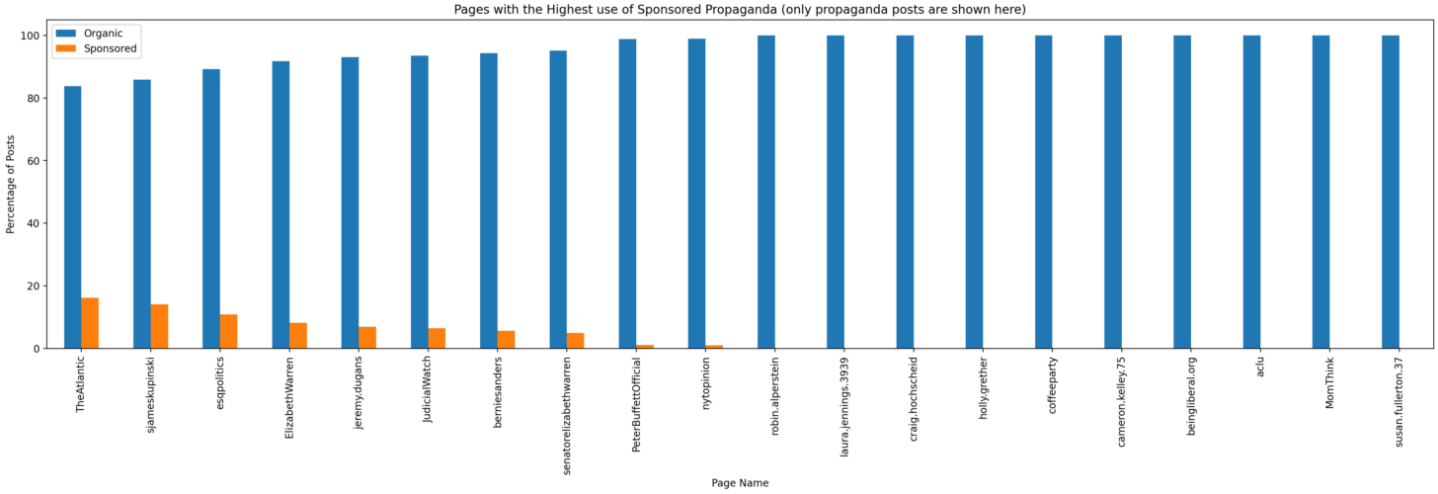


Figure 4.8: Who uses sponsoring the most to publish propaganda ?

3. Partisanship and Propaganda

The below table 4.8 and plot 4.10 show the distribution of propaganda posts by partisanship.

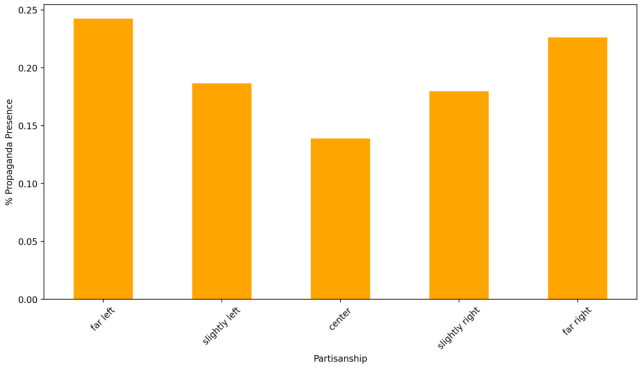


Figure 4.9: Frequency of Propaganda Presence by Partisanship

Party	Not propaganda	Propaganda
center	12309	1985
far left	3303	1057
far right	1235	361
slightly left	9284	2129
slightly right	2053	450

Table 4.8: Count of Posts by Partisanship

To further investigate the relationship between propaganda and political parties, we decided to perform a statistical analysis using the propagandist posts from Facebook pages associated with a specific party (e.g., Far Left). For each page, we calculate the mean engagement of both sponsored and organic posts separately and document these results. This process is repeated for all pages sharing propaganda within that party. Ultimately, we obtain two sets of numbers: one representing the mean engagement of sponsored propaganda posts and the other representing the mean engagement of organic propaganda posts within the same party.

We chose the Wilcoxon signed-rank test for our statistical analysis due to its compatibility with the specific characteristics of our data:

- **Paired Data** : Our dataset consists of paired observations. Each pair of observations within the two series represents engagement metrics from the same page for both organic and sponsored posts. The Wilcoxon signed-rank test is designed to analyze differences within paired samples.

- **Non-normality Assumption** : Engagement data (such as the number of likes and shares) often does not follow a normal distribution, which is the case for our data. The Wilcoxon signed-rank test, similar to the Mann-Whitney U test, is a non-parametric test. This means it does not rely on any assumptions about the underlying data distribution, making it suitable for analyzing engagement metrics.

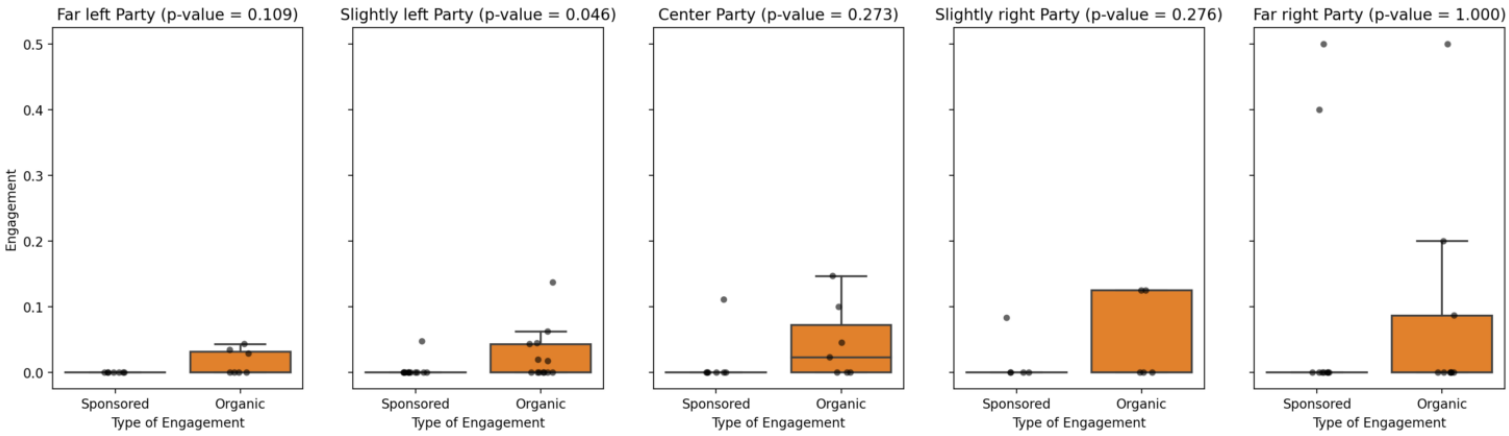


Figure 4.10: Engagement on Organic/Sponsored posts within each Party

In Figure 4.10, we observe that only the slightly left-leaning party has a p-value $<$ 0.05, indicating a significant difference between engagement on propagandist posts and non-propagandist posts. The boxplot further explains this difference by showing that organic posts receive more engagement than sponsored posts. This finding aligns with our initial hypothesis that Facebook's ad system is stringent with content involving hate, false information, or triggering material.

For the other parties, the p-values are $>$ 0.05. This lack of significance is likely due to the small sample size, as there were few observations available for our analysis. Specifically, very few pages shared both sponsored and organic propagandist posts within the same party. Consequently, we cannot draw any definitive conclusions for these parties.

4. Engagements and Propaganda

In this section, we examine the relationship between user engagement and the dissemination of propaganda. We leverage the ***action_on_post*** attribute, which is a boolean indicator of whether a user interacted with a post (e.g., liking, commenting, sharing).

Initially, we used a chi-square statistical test to explore the relationship between each propaganda technique and user engagement. Among the techniques analyzed, only the "loaded language" technique showed a significant dependency with engagement. This result was expected, as loaded language relies on words and phrases with strong connotations that evoke emotional responses, leading to strong positive or negative reactions that often prompt users to engage with the content.

Engagement	Loaded Language Present	Loaded Language Absent
Yes	173	1318
No	3132	29543

Table 4.9: Contingency table of the type of post and propaganda presence

The contingency table 4.9 shows the distribution of user engagement based on whether loaded language was present in a post.

Chi-square Test Results :

The p-value from the chi-square test is **0.0113**, which is less than the typical significance level of 0.05. This suggests that there is a statistically significant relationship between the presence of loaded language and user engagement. This indicates that posts containing loaded language are more likely to drive user engagement compared to posts that do not contain loaded language.

The analysis suggests that loaded language is an effective tool for driving user engagement. This can be particularly relevant for understanding how propaganda techniques influence online behavior. Content creators, advertisers, or propagandists might use such language intentionally to increase engagement, thereby spreading their message more widely.

4.6 YouTube Videos dataset Analysis

4.6.1 Propaganda techniques distribution

The first insight we could find is about the distribution of propaganda techniques, the following plot presents each technique and the percentage of its appearence in the data.

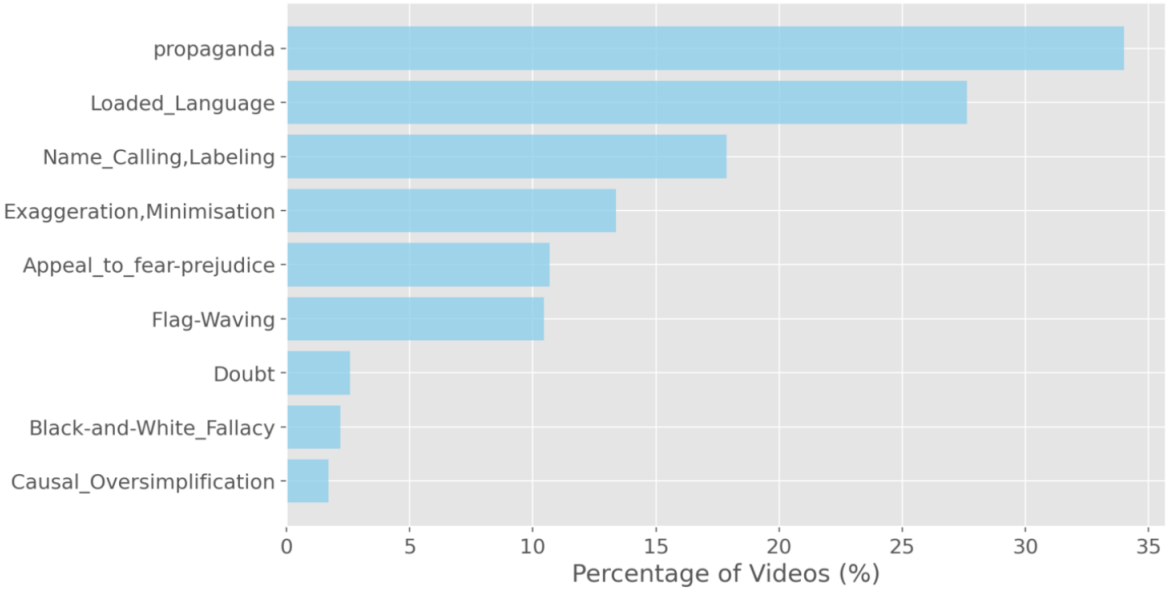


Figure 4.11: percentage of Propaganda techniques in the data

The first thing to note is that the overall percentage of propaganda in the dataset is 34%. We can see that the top three most used propaganda techniques are all syntactic techniques, which means that they are based on the structure of the text rather than the content. On the other hand, the least used propaganda techniques are all semantic techniques.

this suggests that the creators of the videos in the dataset tend to rely more on syntactic techniques by using strong words and labels because they are easier to use and more effective than comming up with arguments.

4.6.2 Propaganda and video duration

The two plots below show the average and median duration of videos for each propaganda technique.

We can see in the plot on the left side that there are 3 techniques that have a considerably higher average duration than the others, and seeing how the average can be affected by outliers, we can confirm whether the deduction is right or wrong by plotting the median duration as well which is not affected by outliers, and we can see that it leads to the same conclusion which suggests that the 3 techniques ('Black and white fallacy',' Causal oversimplification' and 'doubt') are used in longer videos than the others. Looking at these 3 propaganda techniques we can see that they are all semantic-based

techniques which depends on story-telling and arguments to persuade the audiance. this might be the reason why they are found in much longer videos, contrary the other 5 syntactic techniques that depends on using strong words and labels which doesn't require long videos.

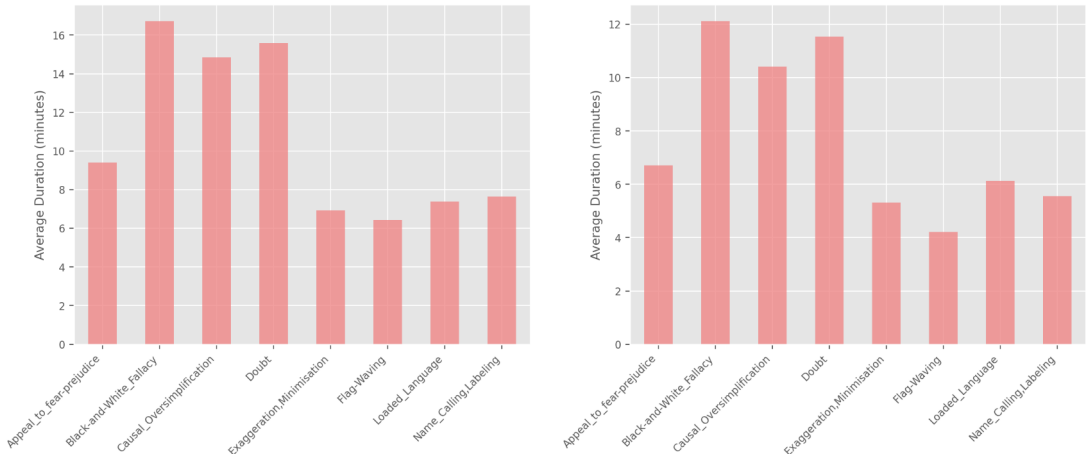


Figure 4.12: Average and median duration by propaganda technique

4.6.3 Propaganda and view count

The bar plot below shows the average views for videos with and without propaganda and based on positive or negative sentiment in the subtitles. But before we can plot the average views, we had the idea to normalize them to make the comparison more accurate by dividing each amount of views a video gets by the number of subscribers of the channel, that way the results won't be affected by videos that are posted by big channels which have a lot more views than small ones.

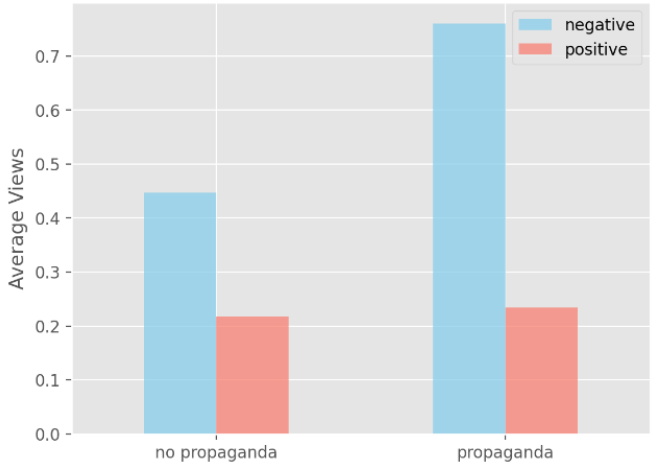


Figure 4.13: Average view count (normalized) per sentiment

The first thing we can notice is that videos with propaganda have a higher average view than videos without propaganda, which suggests that propaganda is more effective in attracting views.

Secondly, we can see that videos with negative sentiment have a higher average views than videos with positive sentiment in both propaganda and non-propaganda videos, one possible cause for this is that negative sentiment videos tend to be more controversial and attract more attention. plus, social media algorithms are programmed to show more controversial content to attract more views. which was the case when Facebook was once sued for promoting negative posts and comments more than the positive ones, which might be the case for YouTube.

finally, we can deduce that using propaganda to positively influence the audience toward a certain idea doesn't seem to have an impact on views, contrary to the negative one.

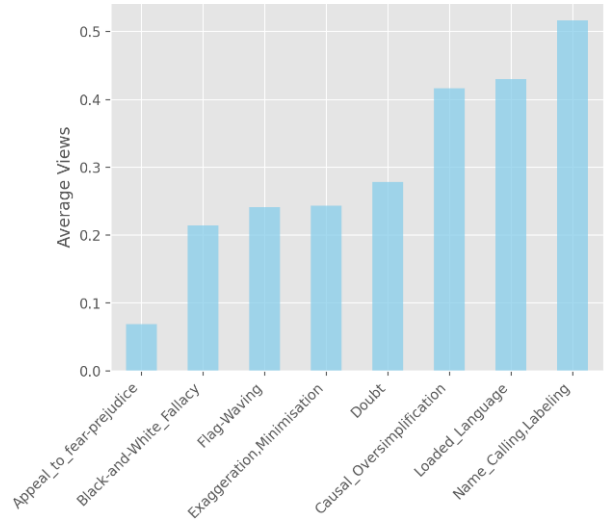


Figure 4.14: Average view count (normalized) per propaganda technique

The bar plot above shows the average views for each propaganda technique. It is apparent that the top 3 propaganda techniques that have higher average views are all syntactic-based techniques, and it can be explained by the fact that strong and explicit wording and messages which is used in syntactic techniques can attract more views. Additionally, We saw in previous analysis that the duration of videos that contain these 3 propaganda techniques is lower than others, which might be another cause for the higher view count, as shorter videos tend to attract more views.

4.6.4 Evolution of propaganda over time

Another interesting analysis we did was to see how the percentage of propaganda in the dataset evolved over time.

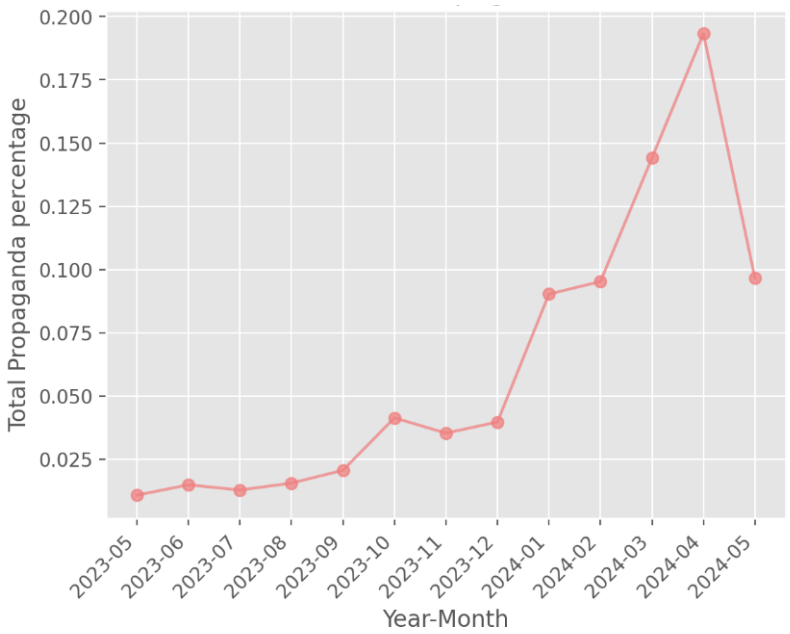


Figure 4.15: A line plot representing the percentage of propaganda over time

The plot above shows the percentage of propaganda in the dataset for each month from the first of May 2023 to the 15th of May 2024. Knowing that the United States Senate elections is scheduled to be held on November 5, 2024. We can see in the plot that the closer the elections are the more the percentage of propaganda increases, the percentage of propaganda in April 2024 is 20 times the percentage of propaganda in May 2023. As for mai 2024, the percentage of propaganda is low because the data was collected on the 14th of March 2024, thus the videos were only collected for half of this month.

4.6.5 Propaganda and Political parties

The following graph represents the percentage of each propaganda technique that targets each political party.

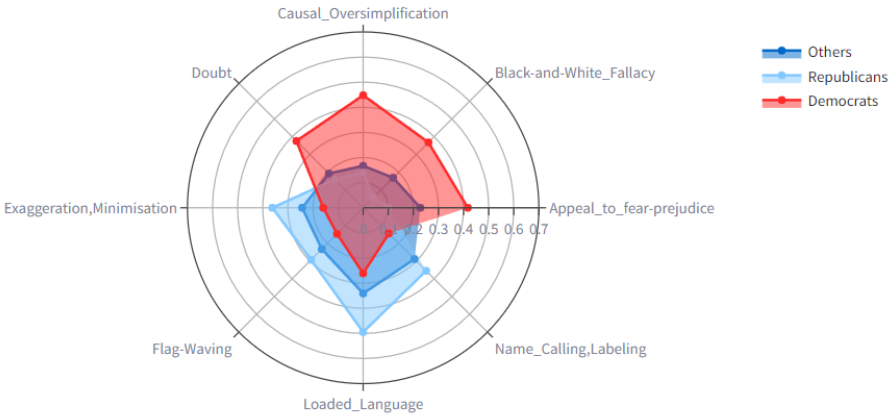


Figure 4.16: percentage of each propaganda technique for each political party

As we can see, there is a pattern that suggests that the propaganda techniques used

against the democrats are more semantic-based techniques, such as 'doubt', 'Causal over-simplification', 'Black and white fallacy', and 'Appeal to fear'. while Republicans are targeted with more syntactic-based techniques such as 'Loaded language', 'Name-calling', 'Flag waving', and 'Exaggeration'.

It's hard to find a reason for this pattern, but one possible explanation we found is that the most used word in the videos about republicans is Trump which is a very controversial figure and could possibly single-handedly cause the high percentage of syntactic techniques used against republicans. other most used words indicate the same thing for example words like Russia, crisis, military, and money are all frequent words in videos about republicans.

As for Democrats, we can find Biden as the most frequent word but he is less mentioned in videos almost half as much as Trump, and less controversial, other frequent words like state, good, news, and case are more neutral words.

4.7 Conclusion

To conclude, we collected a political news dataset from YouTube, consisting of video transcriptions related to the upcoming US Senate elections and other US political events.

Finally, we performed data analysis on the predictions generated by our model on two datasets: a set of 58,000 Facebook posts (both sponsored and organic content) and our newly collected YouTube video dataset. The analysis aimed to address several questions about the dissemination of propaganda on social media.

General Conclusion

General Conclusion

In this graduation report, we explored the advancements and challenges in propaganda detection and the classification of its techniques. We began by defining propaganda, distinguishing it from related concepts such as disinformation, and outlining its various techniques. We also covered the fundamentals of Natural Language Processing (NLP), focusing on Language Models and their primary architecture, Transformers, including their two main components: the Encoder and Decoder. Additionally, we examined well-known and high-performing models like BERT, RoBERTa, and BART, discussing their learning and training approaches, such as pretraining, fine-tuning, and prompt learning.

After conducting a comprehensive literature review of existing methods for propaganda classification, ranging from manual annotation to more advanced machine learning models, we shifted our focus to studying decoder-based Language Models (LLMs). We identified a significant gap in their application to complex tasks like propaganda technique classification. While we achieved improvements over existing work, some limitations rendered our solution impractical.

We then transitioned to our second contribution, where we developed a token-level classification model using an encoder-based LLM that overcame the limitations of decoder-based LLMs. We detailed the implementation phase, including the technical aspects of our architecture, the tools and libraries used, data preprocessing steps, and model development.

Subsequently, we conducted extensive testing and evaluation of our system, comparing its performance with state-of-the-art models and our decoder models from the first contribution. We described our experimental environment, including binary and multi-label classification and span extraction. Our findings demonstrated the effectiveness of our new architecture in improving propaganda detection while surpassing the performance of existing state-of-the-art models.

As a second objective, we collected a political news dataset from YouTube, consisting of video transcriptions related to the upcoming US Senate elections and other US political events.

Finally, we performed data analysis on the predictions generated by our model on two datasets: a set of 58,000 Facebook posts (both sponsored and organic content) and our newly collected YouTube video dataset. The analysis aimed to address several questions about the dissemination of propaganda on social media.

Bibliography

M. Abdullah, O. Altiti, and R. Obiedat. Detecting propaganda techniques in english news articles using pre-trained transformers. In *2022 13th International Conference on Information and Communication Systems (ICICS)*, pages 301–308. IEEE, 2022.

F. Alam, S. Cresci, T. Chakraborty, F. Silvestri, D. Dimitrov, G. D. S. Martino, S. Shaar, H. Firooz, and P. Nakov. A survey on multimodal disinformation detection, Sep 2022. URL https://arxiv.org/pdf/2103.12541.pdf.

O. Balalau and R. Horincar. From the stage to the audience: Propaganda on reddit. *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021. doi: 10.18653/v1/2021.eacl-main.309.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners advances in neural information processing systems 33. 2020.

A. Chen and B. Dhingra. Hierarchical multi-instance multi-label learning for detecting propaganda techniques. In B. Can, M. Mozes, S. Cahyawijaya, N. Saphra, N. Kassner, S. Ravfogel, A. Ravichander, C. Zhao, I. Augenstein, A. Rogers, K. Cho, E. Grefenstette, and L. Voita, editors, *Proceedings of the 8th Workshop on Representation Learning for NLP (RepL4NLP 2023)*, pages 155–163, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.repl4nlp-1.13. URL https://aclanthology.org/2023.repl4nlp-1.13.

G. Da San Martino, S. Yu, A. Barrón-Cedeño, R. Petrov, and P. Nakov. Fine-grained analysis of propaganda in news article. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1565. URL https://aclanthology.org/D19-1565.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

I. Dimov, V. Korzun, and I. Smurov. Nopropaganda at semeval-2020 task 11: A borrowed approach to sequence tagging and text classification. *International Conference on Computational Linguistics*, 12 2020. doi: 10.18653/v1/2020.semeval-1.194.

L. Edelson, M.-K. Nguyen, I. Goldstein, O. Goga, D. McCoy, and T. Lauinger. Understanding engagement with us (mis) information news sources on facebook. In *Proceedings of the 21st ACM internet measurement conference*, pages 444–463, 2021.

I. Habernal, R. Hannemann, C. Pollak, C. Klamm, P. Pauli, and I. Gurevych. Argotario: Computational argumentation meets serious games. In L. Specia, M. Post, and M. Paul, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 7–12, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-2002. URL https://aclanthology.org/D17-2002.

P. He, J. Gao, and W. Chen. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing, 2023.

C. Ireton and J. Posetti. *Journalism, "fake news" amp; disinformation: Handbook for journalism education and training*. United Nations Educational, Science, and Cultural Organization, 2018.

Z. Jin, A. Lalwani, T. Vaidhya, X. Shen, Y. Ding, Z. Lyu, M. Sachan, R. Mihalcea, and B. Schölkopf. Logical fallacy detection, 2022.

S. Khosla, R. Joshi, R. Dutt, A. W. Black, and Y. Tsvetkov. Ltiatcmu at semeval-2020 task 11: Incorporating multi-level features for multi-granular propaganda span identification. *ArXiv (Cornell University)*, 01 2020. doi: 10.18653/v1/2020.semeval-1.230.

M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang. Gpt understands, too. *AI Open*, 2023.

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

A. Maarouf, D. Bär, D. Geissler, and S. Feuerriegel. Hqp: A human-annotated dataset for detecting online propaganda, 2023.

G. D. S. Martino, S. Yu, A. Barrón-Cedeño, R. Petrov, and P. Nakov. Fine-grained analysis of propaganda in news articles, 2019.

G. D. S. Martino, S. Cresci, A. Barron-Cedeno, S. Yu, R. Di Pietro, and P. Nakov. A survey on computational propaganda detection, Jul 2020. URL https://arxiv.org/abs/2007.08024.

E. Musi, M. Aloumpi, E. Carmi, S. Yates, and K. O'Halloran. Developing fake news immunity: fallacies as misinformation triggers during the pandemic. *Online Journal of Communication and Media Technologies*, 12(3), 2022.

D. Q. Nguyen, T. Vu, and A. T. Nguyen. BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, 2020.

OpenAI. Gpt-4 technical report, 2023.

K. Sprenkamp, D. G. Jones, and L. Zavolokina. Large language models for propaganda detection. *arXiv preprint arXiv:2310.06422*, 2023.

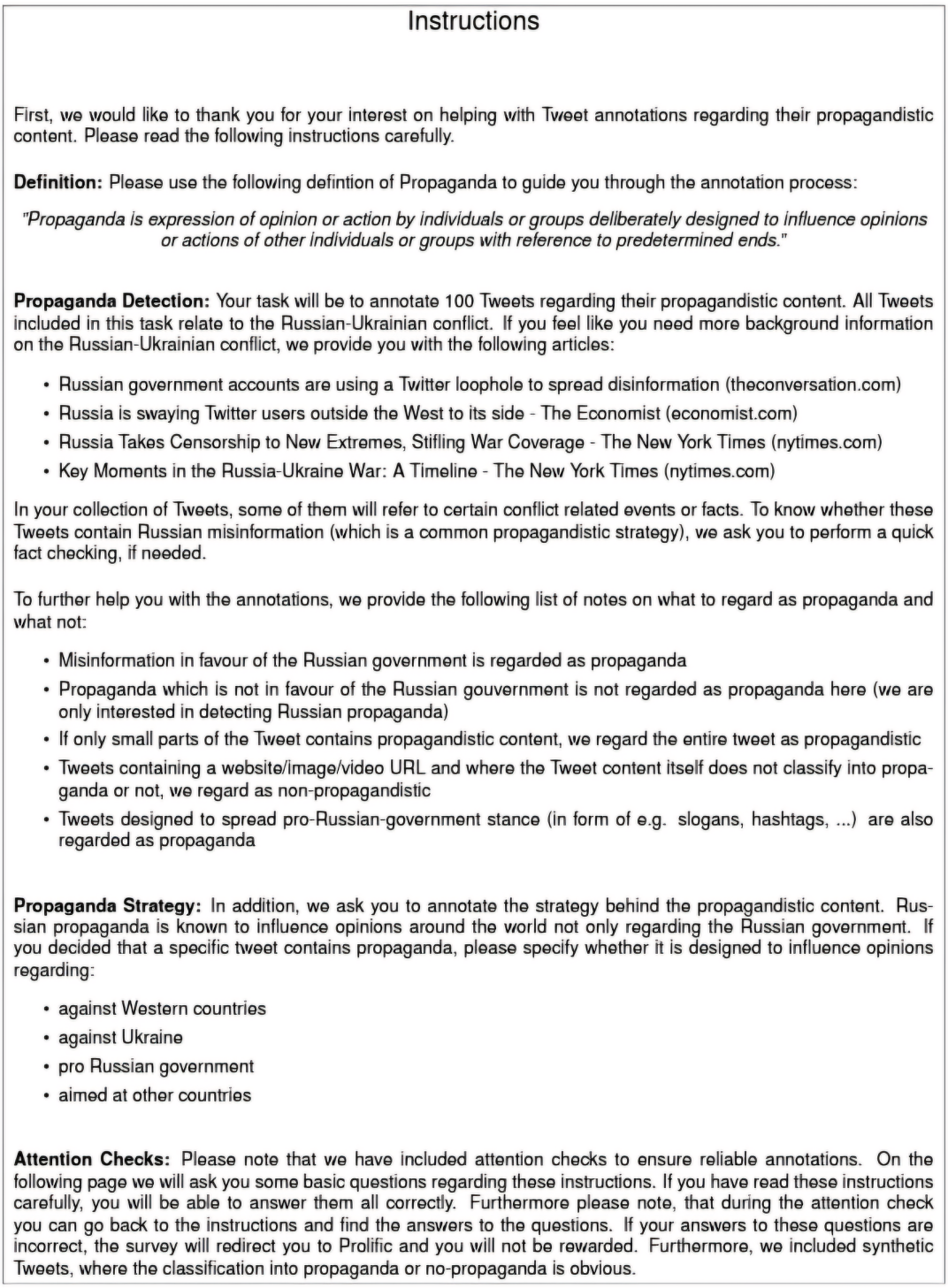
H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models, 2023.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.

P. Vijayaraghavan and S. Vosoughi. TWEETSPIN: Fine-grained propaganda detection in social media using multi-view representations. In M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3433–3448, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.251. URL https://aclanthology.org/2022.naacl-main.251.

Appendices

A Appendix



Instructions

First, we would like to thank you for your interest on helping with Tweet annotations regarding their propagandistic content. Please read the following instructions carefully.

Definition: Please use the following defintion of Propaganda to guide you through the annotation process:

"Propaganda is expression of opinion or action by individuals or groups deliberately designed to influence opinions or actions of other individuals or groups with reference to predetermined ends."

Propaganda Detection: Your task will be to annotate 100 Tweets regarding their propagandistic content. All Tweets included in this task relate to the Russian-Ukrainian conflict. If you feel like you need more background information on the Russian-Ukrainian conflict, we provide you with the following articles:

- Russian government accounts are using a Twitter loophole to spread disinformation (theconversation.com)
- Russia is swaying Twitter users outside the West to its side - The Economist (economist.com)
- Russia Takes Censorship to New Extremes, Stifling War Coverage - The New York Times (nytimes.com)
- Key Moments in the Russia-Ukraine War: A Timeline - The New York Times (nytimes.com)

In your collection of Tweets, some of them will refer to certain conflict related events or facts. To know whether these Tweets contain Russian misinformation (which is a common propagandistic strategy), we ask you to perform a quick fact checking, if needed.

To further help you with the annotations, we provide the following list of notes on what to regard as propaganda and what not:

- Misinformation in favour of the Russian government is regarded as propaganda
- Propaganda which is not in favour of the Russian gouvernment is not regarded as propaganda here (we are only interested in detecting Russian propaganda)
- If only small parts of the Tweet contains propagandistic content, we regard the entire tweet as propagandistic
- Tweets containing a website/image/video URL and where the Tweet content itself does not classify into propaganda or not, we regard as non-propagandistic
- Tweets designed to spread pro-Russian-government stance (in form of e.g. slogans, hashtags, ...) are also regarded as propaganda

Propaganda Strategy: In addition, we ask you to annotate the strategy behind the propagandistic content. Russian propaganda is known to influence opinions around the world not only regarding the Russian government. If you decided that a specific tweet contains propaganda, please specify whether it is designed to influence opinions regarding:

- against Western countries
- against Ukraine
- pro Russian government
- aimed at other countries

Attention Checks: Please note that we have included attention checks to ensure reliable annotations. On the following page we will ask you some basic questions regarding these instructions. If you have read these instructions carefully, you will be able to answer them all correctly. Furthermore please note, that during the attention check you can go back to the instructions and find the answers to the questions. If your answers to these questions are incorrect, the survey will redirect you to Prolific and you will not be rewarded. Furthermore, we included synthetic Tweets, where the classification into propaganda or no-propaganda is obvious.

Figure 4.17: Instructions for annotators (HQP Dataset) [*Maaroufet al.*, 2023].

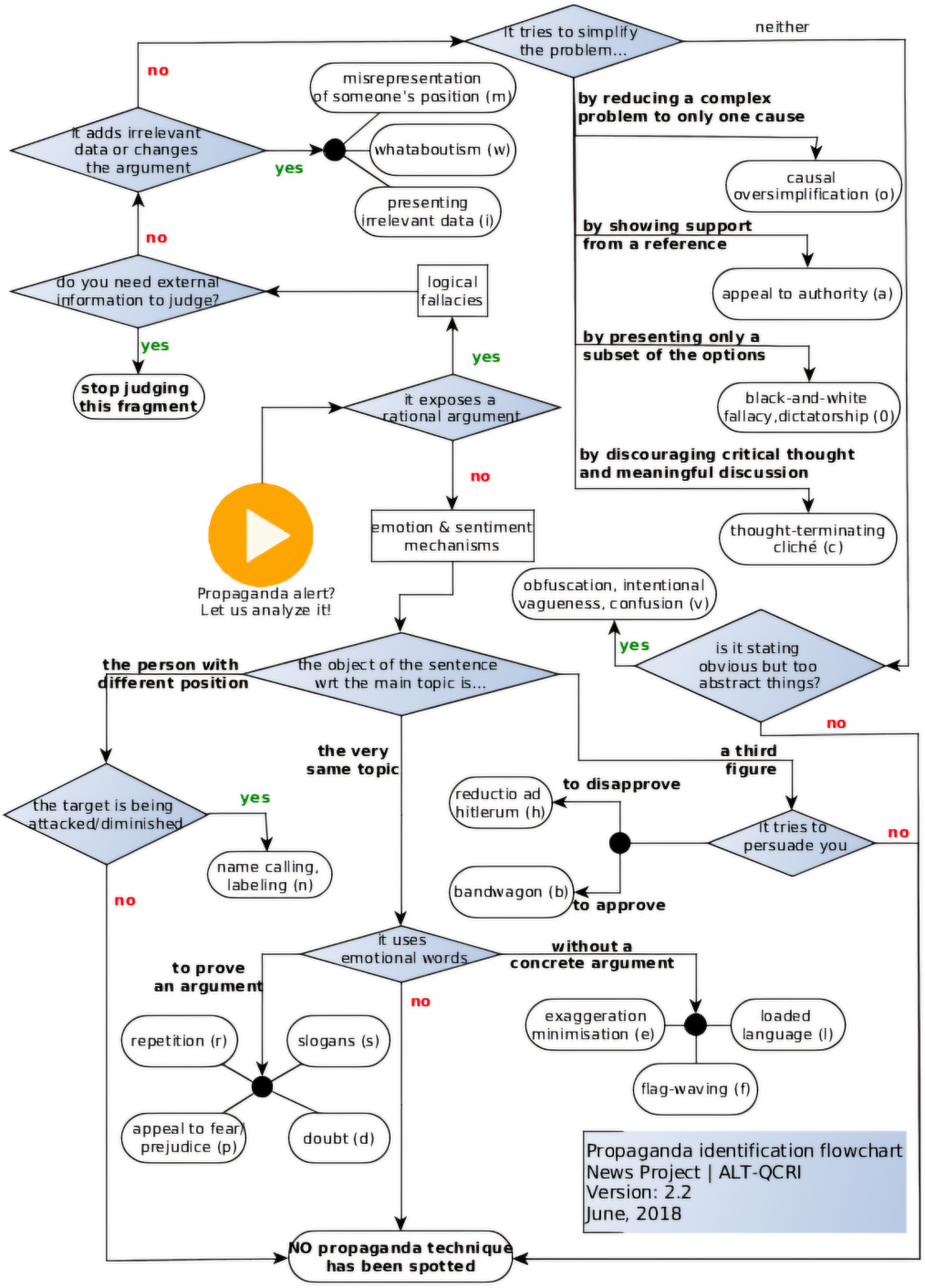


Figure 4.18: Experts Instructions (PROPAGANDA Dataset) [*Da San Martinoet al.*, 2019]