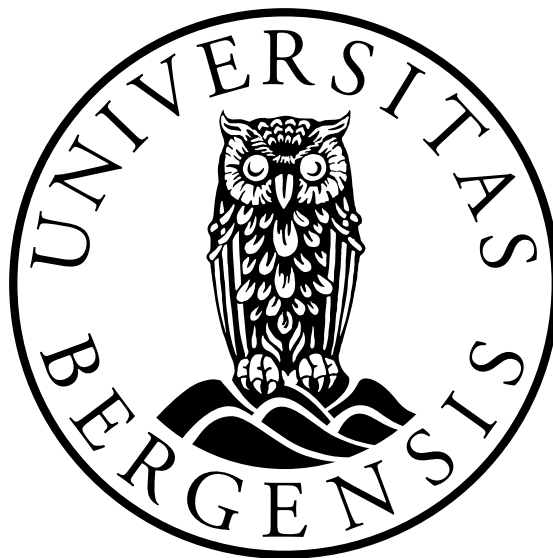


Automated Moderation: Detecting Irony in a Norwegian Facebook Comment Section using a Longformer Transformer Model with a Context Encoded Dataset

Torstein Hatlebakk



Master's Thesis
Department of Information Science and Media Studies
University of Bergen

May 31, 2022

Scientific environment

This study is carried out at the Institute of information- and media science, University of Bergen. The work is supported by the MediaFutures research centre.



Acknowledgements

I want to thank Bjørnar Tessem, TV2 and Samia Touileb for their help and support with the project.

Abstract

Irony is a complex phenomenon of human communication and due to its contextual nature has been notoriously difficult for machine learning algorithms to detect. With an established practical definition of irony based in the environment of Facebook comment sections. Used together with a Norwegian language pre-trained BERT model converted to a long version that supports longer text inputs, and a Norwegian Facebook comment dataset with contextual article and reply comment text included. It was found that the long BERT model trained on the context included inputs dataset outperformed the short BERT models trained on datasets of the same and more comments, but without the contextual information encoded.

Contents

Scientific environment	i
Acknowledgements	iii
Abstract	v
1 Introduction	1
1.1 Motivation	1
1.2 Problem Statement	3
1.3 Objectives	3
1.4 Contribution	4
1.5 Thesis outline	4
2 Background	7
2.1 Definitions	7
2.1.1 Irony	7
2.2 Machine Learning Models	10
2.2.1 Transformer Models	11
2.2.2 Norwegian Transformer Models	13
2.2.3 Longformer	14
3 Methodology	17
3.1 Irony Construct	17
3.2 Comment Linguistics	19
3.3 Facebook Post Constructs	19
3.4 Data Topic Distribution	21
3.5 Context Encoding	22
3.6 Threats to Validity	23
4 Methods	25
4.1 Converting Model to Long	25
4.2 Dataset and Labelling	31
4.2.1 Long and short dataset	31
4.3 Implementation	34
4.4 Evaluation	38

5	Results and Discussion	41
5.1	Results and Analysis	41
5.1.1	Models' Metrics	41
5.2	Discussion	49
6	Conclusions and Future Work	51
6.1	Conclusions	51
6.2	Future Work	51
6.2.1	Larger Datasets	51
6.2.2	Individual Detection	52
6.2.3	Multi-Label	52
6.2.4	Multi-Modal	52
6.2.5	Dialect	52
6.2.6	Other Sentiments and Contexts	52
	Model and Python Files	57

List of Figures

2.1	Examples of verbal irony from [22]	8
2.2	Definition of irony from [10]	9
2.3	Definition of irony from [11]	9
2.4	Inference diagram from [15]	10
2.5	F1 scores for listed models on various tasks [5]	11
2.6	Accuracy for pre-training on various datasets [5]	11
2.7	The transformer model architecture from [20]	12
2.8	RCNN-RoBERTa compared to other models on irony detection SemVal task 2018 [16]	13
2.9	The Norwegian language trained models [?]	14
2.10	The different types of attention mechanisms from [2]	15
3.1	The design science process [21]	17
3.2	A model of the relevant constructs included in a Facebook post model.	21
3.3	The distribution of topics and the amount of comments per topic in the TV2 dataset.	22
3.4	Performance metrics of differently trained longformer models and RoBERTa	23
4.1	Example of the input ids, attention mask and converted tokens generated by using norBERT 2's tokenizer, which was used for the short models, on the Norwegian example phrase: "Dette er et eksempel på en tokenifisert setning ved bruk av norBERT2 sin tokenifiserer."	33
4.2	Example of the input ids, attention mask and converted tokens generated by using LongNorBERT 2's tokenizer, which was used for the long model, on the Norwegian example phrase with artilce and reply context texts: "Eksempel artikkel tekst. [SEP] Dette er et eksempel på en tokenifisert setning ved bruk av LongNorBERT2 sin tokenifiserer. [SEP] Eksempel svar til kommentar."	34
4.3	Formula for accuracy metric	38
4.4	Formula for precision metric	38
4.5	Formula for recall metric	38
4.6	Formula for F1 score metric	39
4.7	ROC by comparing TPR and FPR values	39

5.1	Graph of the number of true positive predictions given classification threshold values by the three models on the validation set	42
5.2	Graph of the number of false positive predictions given classification threshold values by the three models on the validation set	43
5.3	Graph of the numerical difference between the TP and FP number predictions given classification threshold values by the three models on the validation set	44
5.4	Graph of the accuracy of predictions given classification threshold values by the three models on the validation set	47
5.5	Graph of the F1 scores the of predictions given classification threshold values by the three models on the validation set	47
5.6	Graph of the precision of predictions given classification threshold values by the three models on the validation set	48
5.7	Graph of the recall of predictions given classification threshold values by the three models on the validation set	48
5.8	Graph of the relation between the true postive rate and false positive rate of predictions given classification threshold values by the three models on the validation set	49

List of Tables

- 3.1 Table showing examples of how different comment texts can be recognized as ironic with different methods 18
- 3.2 Table showing how article text, comment text and reply text are encoded in the dataset 22

- 4.1 The datasets' basic metrics 32
- 4.2 Example of a few short dataset inputs 32
- 4.3 Example of a few long dataset inputs 33
- 4.4 Training parameters for the TV2FacebookCoronaPoliticsIrony-LongNorBERT2 model. 36
- 4.5 Training parameters for the TV2FacebookCoronaPoliticsIronyShort-NorBERT2 model. 37
- 4.6 Training parameters for the all topic TV2FacebookIronyShortNorBERT2 model. 37

- 5.1 The validation set's metrics 41
- 5.3 Caption 49

Chapter 1

Introduction

Online comment sections such as those on the website Facebook are a great place for people to express and process their thoughts and opinions with friends, strangers, and various kinds of institutions. However, it is also a prime environment for harassment. This kind of behaviour tends to devolve serious conversations about important topics, causes mental harm and in some cases pose legal issues for the institutions that host those comments. For instance, the Norwegian news and entertainment company TV2, have closed their comment sections due to these issues.

In response to the difficulty of moderating online human communication, the field of automated moderation in which machine learning based natural language processing is used to prevent or moderate the comments, has grown significantly. However, human language and social behaviour is deeply complex and difficult to accurately moderate without stifling free speech. One such aspect of human communication which is difficult to precisely detect is irony. And as ironic statements are part of free speech it is necessary to not censor them, by miscategorising them as illegal or unwanted language, such as harassment.

1.1 Motivation

On the 1st of July 2020, the Norwegian government's ministry of culture and equality enacted a law titled *Act relating to the editorial independence and liability of editor-controlled journalistic media*, abbreviated as the *Media Liability Act*. Section 3d of the act, which concerns definitions, specifies that statements made by users not affiliated with the media editor is considered user generated: "*d. user-generated content: a statement published by a media user outside the editor's management and control*" [12]. Furthermore, when it comes to the liability of hosting illegal user-generated content, in which comments that contain an illegal degree of discriminatory or hateful content would fall under. The act details that the editor must allow reporting it and must remove it with opportunity for appeal: "*The editor must facilitate the reporting of illegal user-generated content. If user-generated content is removed or access to the content is barred because it is considered illegal, the editor must, to the extent possible, notify the author of*

the content and inform him/her of opportunities to appeal” [12].

However, the act adds that they are only held liable and penalised if they act with gross negligence, meaning they do not remove the illegal user-generated content without undue delay: *”The editor, or any person acting on behalf of the editor, may only be penalised for illegal user generated content in the medium if he or she has acted with intent. To be held liable for compensation, he or she must have acted with gross negligence. The editor, or any person acting on behalf of the editor, cannot be held liable pursuant to the first subsection if he or she without undue delay, once the conditions for liability were present, implements the necessary measures to remove or bar access to the illegal statement” [12].*

If an institution such as TV2 desired to create or re-open a comment section on their website, they would have to comply with these stipulations. However, moderation companies can be expensive. Especially in regard to moderation of Norwegian language content, which is far more niche than English language content. And seeing as a company like TV2 can host the comments on their articles on external sites which carry the burden of moderation instead, such as the website Facebook, or integrate the Facebook comment section into their own site, there is little economic incentive to create their own. However, by outsourcing the comments and responsibility for moderation, the control over freedom of expression in the sense of what is considered illegal content is surrendered to an external institution whose values may not align with TV2’s values or Norwegian policy.

One way to not rely on external actors, while reducing expenses, manual labour and complying with the law, is to remove all reported comments. However, this would likely cause comments that do not contain illegal content to be removed as well, reducing the amount of legal user-generated content and by extension freedom of expression. The Norwegian government’s official website Government.no states the following about their laws and attitude towards freedom of expression: *”Freedom of expression is not only a prerequisite for democracy, it is also vital for the realisation of other fundamental human rights, such as freedom of assembly and freedom of religion or belief. Promoting freedom of speech is therefore an important part of Norway’s foreign policy and human rights priorities” [13].* However, comments that contain illegal content still have to be removed, as it further elaborates by specifying: *”This does not mean that all manifestations of freedom of expression are allowed. It is in contravention of Norwegian law and Norway’s international human rights obligations to publicly make discriminatory or hateful statements” [13].*

If an automatic system would be used to moderate a non-external comment section in compliance with Norwegian law, it would have to be 100 percent accurate as any reported illegal content comments that it does not remove would make for instance TV2 liable for penalization. As no NLP model operating on the complexity of a comment section has achieved 100 percent accuracy in classification, the system would require manual input in the form of a human in the loop, to

verify decisions and prevent legal liability. However, the higher accuracy the classification system performs with, the less manual labour would be required to verify it. Machine learning algorithms that can assist a manual moderator by classifying the comments in advance, only pending his final verification, would maximize efficiency to incentivize self-governance, while simultaneously maximizing freedom of expression.

1.2 Problem Statement

State of the art language phenomenon detection technology, such as those which detect irony, use standard transformer type machine learning natural language processing algorithms [16]. These algorithms encode the relations between words in a way that allows it to represent the correlations between those words better than any other existing models. However, statements in comments made as a reply to its respective post or another comment, can be interpreted as ironic or harassing depending on the content of the post or comment they are replying to. And the standard algorithms have input length limitations that prevent them from including contextual elements such as the post's text, article and previous comments.

Because of this, a standard transformer type algorithm such as BERT, trained on a labeled dataset of ironic comments encode the word combination, word order, special characters, grammar and capitalization, which can be referred to as linguistic patterns in the ironic comments, but not the context in which the comments were made or how the context affects a comment's ironic qualities. And as irony is a highly contextual language phenomenon, its ability to predict irony in complex scenarios with a lot of contextual information, such a Facebook comment section, suffers. And as a human must be kept in the loop for a moderation system, the worse the automated detection system performs, the more work and people are needed.

1.3 Objectives

The hypothesis is that by using longer versions of the transformer NLP methods, both the contextual language patterns and the linguistic patterns can be understood by the trained model, leading to more accurate detection of irony in complex environments such as a *TV2 Nyheter* Facebook post's comment section [19].

Research Question 1: *To what degree better or worse than a model without, does including a comment's contextual text information in an NLP model's dataset improve its for sequence classification accuracy in predicting a comment's irony value?*

Research Question 2: *To what degree does fine-tuning on comments from topics different from the validation dataset affect prediction accuracy of the vali-*

dation dataset?

1.4 Contribution

Due to TV2 privacy concerns and article 5.1 of the European Union’s General Data Protection Act regarding the use of personal data, despite not containing names, the data allows for finding the comments’ authors using the article and comment text. Therefore the dataset contains too much personal information to publicly release. However, the produced models, non-fine-tuned converted long model and encoding technique can [?].

List of contributions produced by this master thesis:

- **C1** A long non-pre-trained version of the Norwegian language transformer model norBERT 2, called 1024-LongNorBERT 2.
- **C2** A TV2 Facebook corona politics comments irony fine-tuned 1024-LongNorBERT 2 model.
- **C3** A TV2 Facebook corona politics comments irony fine-tuned norBERT 2 model.
- **C4** A TV2 Facebook multi-topic comments irony fine-tuned norBERT 2 model.
- **C5** A technique of article, comment and reply context encoding for text classification datasets.

1.5 Thesis outline

Chapters:

- **Chapter 2** Background
 - **Definitions**, elaborates on which definition of irony is used, why that definition is specifically chosen and how it fits the Facebook comment section environment.
 - **Machine Learning Models**, describes the state of the art models used in irony detection, how they work, and why specific models have to be used for detecting irony in the Norwegian language, as well as how to bypass text length limitations to include context data.
- **Chapter 3** Methodology
 - **Irony Construct**, explains how the definition is applied to the Facebook comment section environment.

- **Comment Linguistics**, illustrates some potential problems that might lower model performance, concerning the form of the language used in the Norwegian comments.
- **Facebook Post Constructs**, lists constructs for the various parts of a Facebook post and why only specific constructs are used for the model.
- **Data Topic Distribution**, shows the topics for the text in the dataset and explains which parts of the dataset were used for different purposes.
- **Context Encoding**, presents a technique for encoding context that can be used for fine-tuning of text classification.
- **Threats to Validity**, argues for several factors that likely cause inaccuracies in the trained models.
- **Chapter 4 Methods**
 - **Converting Model to Long**, details the process of converting a 512 token limited BERT model to a 1024 token limit long BERT model.
 - **Dataset and Labelling**, introduces the format, reasoning, and filtering process for pre-processing and labelling the datasets.
 - **Implementation**, clarifies how the models were trained and with which training parameters.
 - **Evaluation**, discloses how the models were evaluated by listing the metrics used on their predictions.
- **Chapter 5 Results and Discussion**
 - **Results and Analysis**, shows and comments on the graphed results of the evaluation metrics.
 - **Discussions**, argues about the models' compared performance and the likely reasons behind their specific performances.
- **Chapter 6 Conclusions and Future Work**
 - **Conclusions**, a summary of the insights deduced from the results of the models' performance.
 - **Future Work**, a listing of greater dataset scales and various other pre-existing technologies that can be combined with the thesis's methodologies and methods, to likely produce greater results.

Chapter 2

Background

2.1 Definitions

2.1.1 Irony

In order to detect irony, a definition of irony is required to precisely define what language qualifies as ironic. In her book titled *Irony* Claire Colebrook states: *"Despite its unwieldy complexity, irony has a frequent and common definition: saying what is contrary to what is meant..., a definition that is usually attributed to the first-century Roman orator Quintilian who was already looking back to Socrates and Ancient Greek literature. But this definition is so simple that it covers everything from simple figures of speech to entire historical epochs. Irony can mean as little as saying, 'Another day in paradise', when the weather is appalling. It can also refer to the huge problems of postmodernity; our very historical context is ironic because today nothing really means what it says. We live in a world of quotation, pastiche, simulation and cynicism: a general and all-encompassing irony. Irony, then, by the very simplicity of its definition becomes curiously undefinable"* [3]. As Colebrook states, irony can be considered undefinable, however a piece of sentiment detection technology intended to detect irony, only needs to operate within the practical context of its environment. Hence, in the event the irony in a comment is subtle, requires inaccessible knowledge or contextual understanding that the general public does not have, then most people exposed to the comment would interpret it literally, and whether the author intended it ironically or not makes no practical difference. This in turn entails that what classifies as ironic under the labelling process of the dataset is exposed to the subjective bias of what the labeler thinks the general public interprets as ironic.

The paper *Irony, The Many Types: Irony vs. Satire and Paradox, Linguistic Irony vs. Situational Irony, Stable vs. Observable Irony, Dramatic Irony, Tragic Irony, Dark Irony, and Visual Irony*, describes the many forms irony can take. However, all of these types of irony are exclusive to specific situations, most of the being literary tools for storytelling, as they describe how an audience or reader perceives irony in a specific situation, and not the phenomena of communication between people in a natural social environment, such as a Facebook comment section [14]. According to the paper *Explaining Irony*, the traditional type of

irony is called *verbal irony*. It defines verbal irony as: *"In classical rhetoric, verbal irony is analysed as a trope: an utterance with a figurative meaning that departs from its literal meaning in one of several standard ways... irony... it is the contrary or contradictory of the literal meaning"* [22].

- (1) *Mary (after a boring party): That was fun.*
- (2) *I left my bag in the restaurant, and someone kindly walked off with it.*
- (3) *Sue (to someone who has done her a disservice): I can't thank you enough.*

Figure 2.1: Examples of verbal irony from [22]

In the state of the art irony and sarcasm detect paper *A transformer-based approach to irony and sarcasm detection*, irony and sarcasm is also categorised as figurative under the term *figurative language* or FL for short: *"The main FL expression forms are sarcasm, irony and metaphor."* And that: *"The linguistic phenomenon of figurative language (FL) refers to the contradiction between the literal and the nonliteral meaning of an utterance"* [16]. It further elaborates by explaining a computational approach to defining irony: *"Indeed, this is the case of pragmatic FL phenomena like irony and sarcasm that main intention of in most of the cases, are characterized by an oppositeness to the literal language context. It is crucial to distinguish between the literal meaning of an expression considered as a whole from its constituents' words and phrases. As literal meaning is assumed to be invariant in all context at least in its classical conceptualization..., it is exactly this separation of an expression from its context that permits and opens the road to computational approaches in detecting and characterizing FL utterance. We may identify three common FL expression forms, namely irony, sarcasm and metaphor. In this paper, figurative expressions, and especially ironic or sarcastic ones, are considered as a way of indirect denial. From this point of view, the interpretation and ultimately identification of the indirect meaning involved in a passage does not entail the cancellation of the indirectly rejected message and its replacement with the intentionally implied message... On the contrary, ironic/sarcastic expressions presuppose the processing of both the indirectly rejected and the implied message so that the difference between them can be identified"* [16].

The paper *Muting the Meaning A Social Function of Irony*, clarifies the distinction between irony and sarcasm: *"Ironic insults, where the positive literal meaning is subverted by the negative intended meaning, will be perceived to be more positive than direct insults, where the literal meaning is negative"* [?]. Additionally, in the Marriam Webster dictionary definitions in figures 2.2 and 2.3, sarcasm is defined as negative ironic utterances or language. Because of this relation, sarcasm can be defined to be a subcategory of the larger category of irony. This is because all sarcastic utterances are ironic, but not all ironic utterances are negative and therefore not all ironic utterances are sarcastic [10] [11].

irony noun

 Save Word

iro·ny | \ ˈɪ-rə-nē  also ˈɪ(-ə)r-nē  \

plural ironies

Definition of *irony*

- 1 **a** : the use of words to express something other than and especially the opposite of the literal meaning
 - b** : a usually humorous or sardonic literary style or form characterized by irony
 - c** : an ironic expression or utterance
- 2 **a** **(1)** : incongruity between the actual result of a sequence of events and the normal or expected result

(2) : an event or result marked by such incongruity

b : incongruity between a situation developed in a drama and the accompanying words or actions that is understood by the audience but not by the characters in the play


— called also *dramatic irony*
- 3 : a pretense of ignorance and of willingness to learn from another assumed in order to make the other's false conceptions conspicuous by adroit questioning

— called also *Socratic irony*

Figure 2.2: Definition of irony from [10]

sarcasm noun

 Save Word

sar·casm | \ ˈsär-,ka-zəm  \

Definition of *sarcasm*

- 1 : a sharp and often satirical or ironic utterance designed to cut or give pain
- 2 **a** : a mode of satirical wit depending for its effect on bitter, caustic, and often ironic language that is usually directed against an individual
 - b** : the use or language of sarcasm

Figure 2.3: Definition of irony from [11]

With all these explanations and definitions in mind, the definition of irony the comments are analyzed and label with is: *An utterance whose total semantic expression clarifies a significant difference in the its indirectly rejected literal expression and implied expression, which is often in some form of opposition with its literal expression.*

2.2 Machine Learning Models

The detection of language phenomenon such as irony and or sarcasm is typically classified under the larger category of sentiment analysis, and has been done by training or fine-tuning various kinds of machine learning NLP models on irony labelled datasets. The paper *Comparative Study of Machine Learning Models and BERT on SQuAD*, compares the accuracy of the top performing model architectures. We can see from figure 2.4 that the BERT model architecture outperforms the other architectures by at least 7% or more [15]. The BERT model's superior performance, is further supported by the state of the art irony and sarcasm detector RCNN-RoBERTa being based on the BERT model architecture.

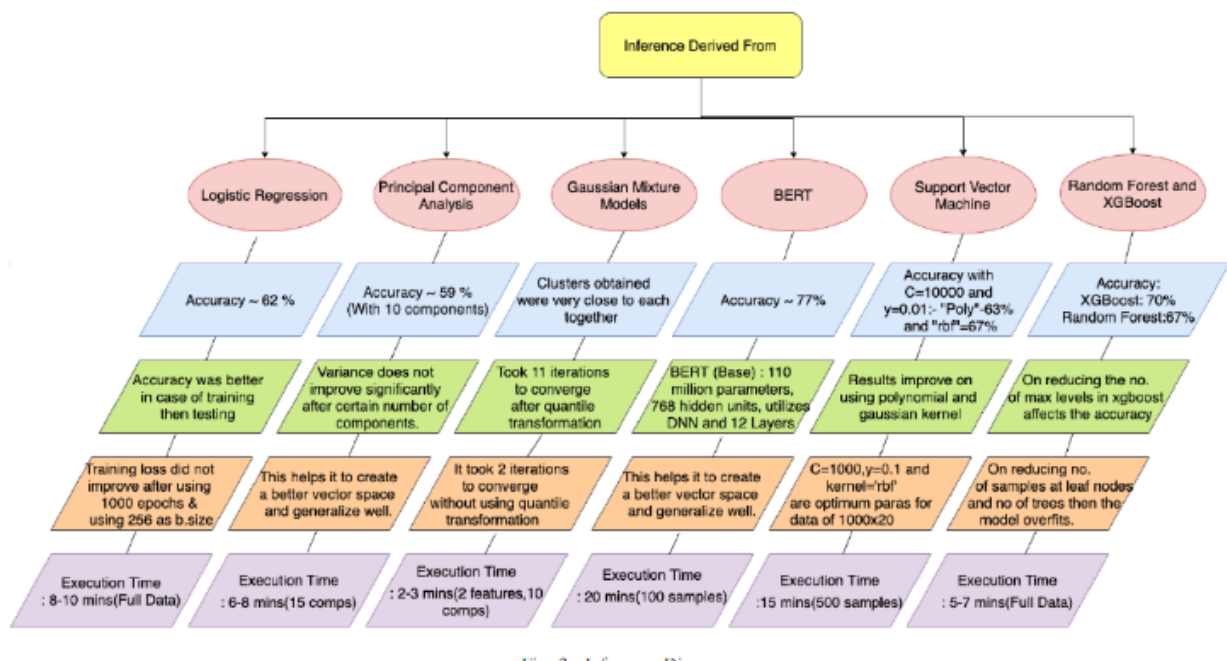


Figure 2.4: Inference diagram from [15]

The model architecture that was the state of the art before BERT, was *Long Short-Term Memory* models or LSTM for short. However, as the original BERT paper shows in figures 2.5 and 2.6, that the BERT models outperforms the LSTM models. This is because unlike BERT, LSTM networks pass words through its network sequentially and generate words sequentially, making it not capture the whole context of how words in a text relate to one another [6]. This even applies to bidirectional LSTM networks as they encode the relations between words right and left separately before concatenating them [18]. BERT models on the other hand are faster and more word context preserving, as they can learn the re-

lations between words in both directions simultaneously and for multiple words simultaneously [5].

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

Figure 2.5: F1 scores for listed models on various tasks [5]

Tasks	Dev Set				
	MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
BERT _{BASE}	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Figure 2.6: Accuracy for pre-training on various datasets [5]

2.2.1 Transformer Models

Bidirectional Encoder Representations from Transformers or BERT for short, is a technique of NLP model pre-training developed by Google in 2018 [5]. It is already used by Google’s search engine for language queries and has become the baseline model for state of the art NLP [17]. It is based on the transformer deep learning model also originally developed by Google in 2017 [20]. As seen in figure 2.7, the transformer architecture consists of an encoder and decoder. The encoder takes every word in for instance a sentence and generates vector embeddings for each word simultaneously. The decoder takes vector embeddings and generates words based on their values. In essence, the decoder learns the context of the text it is provided, such as grammar. While the decoder learns how vector embeddings relate to one another, such as how words in one language relate to words in another [20]. The BERT architecture is a structure consisting of multiple transformer encoders. This structure allows the transformers to be used for more tasks, which includes sentiment analysis. This is done by giving BERT a general understanding of a language, by pre-training it on a large corpus of text using masked language modelling or MLM, and next sentence prediction or NSP as training tasks. The pre-trained model can then be fine-tuned to do a specific task with a smaller labelled dataset that creates an input layer for the specific task [5].

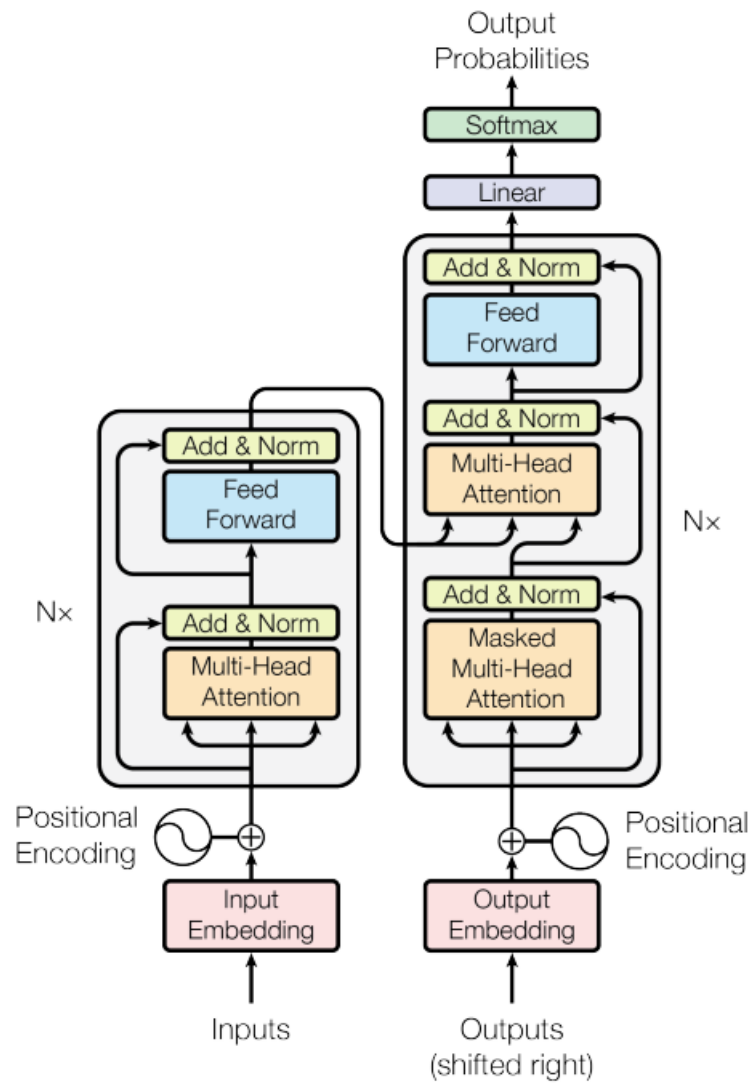


Figure 2.7: The transformer model architecture from [20]

In the state of the art irony detection paper *A transformer-based approach to irony and sarcasm detection*, the proposed method is a RCNN-RoBERTa model with both RoBERTa layers and BiLSTM layers [16]. RoBERTa is short for A Robustly Optimized BERT Pretraining Approach [?]. As shown in figure 2.8, it outperforms all previous methods on the SemEval-2018 task. And as listed in the paper, it also outperforms the other models on the Reddit SARC 2.0 politics and Riloff sarcastic dataset [16]. However, even though RoBERTa is the state of the art in irony detection, it is not pre-trained on the Norwegian language and can therefore not be used to fine-tune for irony prediction in Norwegian comments.

Irony/SemVal-2018-Task 3.A [35]

System	Acc	Pre	Rec	F1	AUC
ELMo	0.66	0.66	0.67	0.66	0.72
USE	0.69	0.67	0.67	0.67	0.74
NBSVM	0.69	0.70	0.69	0.69	0.73
FastText	0.69	0.71	0.69	0.69	0.73
XLnet	0.71	0.71	0.72	0.70	0.80
BERT-Cased	0.70	0.69	0.70	0.69	0.77
BERT-Uncased	0.69	0.68	0.69	0.68	0.77
RoBERTa	0.79	0.78	0.79	0.78	0.89
Wu et al. [97]	0.74	0.63	0.80	0.71	–
Ilić et al. [40]	0.71	0.70	0.70	0.70	–
THU_NGN [97]	0.73	0.63	0.80	0.71	–
NTUA-SLP [5]	0.73	0.65	0.69	0.67	–
Zhang et al. [102]	–	–	–	0.71	–
DESC [76]	0.74	0.73	0.73	0.73	0.78
Proposed	0.82	0.81	0.80	0.80	0.89

Bold figures indicate superior performance

Figure 2.8: RCNN-RoBERTa compared to other models on irony detection SemVal task 2018 [16]

2.2.2 Norwegian Transformer Models

In order to fine-tune a model to detect irony in Norwegian comments, a transformer model with an understanding of the Norwegian language is required. As a part of the NorLM initiative to create large Norwegian language models, the SANT project, NLPL (the Nordic Language Processing Laboratory) and EOSC-Nordic (European Open Science Cloud) have created a series of Norwegian pre-trained NLP models. Among them is a Norwegian pre-trained BERT model called

norBERT 2. It was trained on over 15 billion Norwegian words, which is equivalent to about 1 billion Norwegian sentences in both Bokmål and Nynorsk. As shown in figure 2.9, the model NB-BERT-Base outperforms norBERT 2 in almost all tasks except binary sentiment analysis, which irony detection falls under [?].

Model/task	mBERT	XLM-R	NorBERT	NorBERT 2	NB-BERT-Base
Part-of-Speech tagging Bokmål (accuracy) code	98.0		98.5	98.3	98.7
Part-of-Speech tagging Nynorsk (accuracy) code	97.9		98.0	97.7	98.3
Fine-grained sentiment analysis (Mean Targeted F1 across 5 runs) code	34.9	33.9	35.0	33.8	34.4
Binary sentiment analysis (F1 score) code	67.7		77.1	84.2	83.9
Named entity recognition Bokmål (F1 score) code	78.8		85.5	88.2	90.2
Named entity recognition Nynorsk (F1 score) code	81.7		82.8	84.5	88.6

Figure 2.9: The Norwegian language trained models [?]

2.2.3 Longformer

The problem with using a base BERT model such as norBERT 2, is its 512 token limit. A token is an encoded part of a sentence or larger text. Tokenizing text generates inputs ids for the corresponding words to a vocabulary the tokenizer has generated during its pre-training, an attention mask which indicates which parts of the input have words and lastly the input ids can be converted to their representative tokens to see how the tokenizer separates words and characters into tokens [5].

The problem with the 512 token limit, is that it prevents contextual data, such as an article from being used, as they consistently exceed 512 tokens. Methods for compressing the text could be utilized, but would be counter intuitive to the usage of norBERT 2 as the correlation between Norwegian words in sentences are pre-trained, not compressed representations of text. However, there is a type of BERT model that supports token lengths larger than 512. These models are called longformer models or long models [2],

Attention or self-attention is a mechanism of the transformer architecture that relates positions of a token sequence. Meaning attention is what dictates which parts of a text the model spends most of its attention, because it thinks its important [20]. The traditional transformer models, such as BERT have a self-attention that scales quadratically with the lengths of its input sequences. This means that size and training time becomes exponentially larger and longer the longer the input sequence is. Longformer solves this problem by using an attention mechanism called sliding window attention with global attention, causing self-attention to scale linearly with sequence length. Windowed and global attention are ways restricting attention access to lower computation time and memory size, while still allowing representations to be built across whole inputs [2].

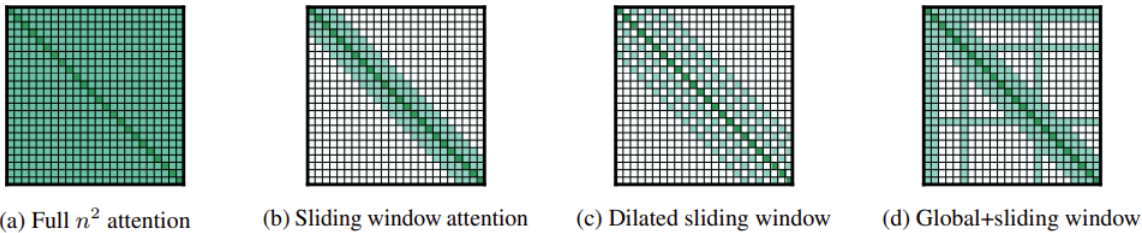


Figure 2.10: The different types of attention mechanisms from [2]

Chapter 3

Methodology

In order to test the hypothesis, an irony recognizer artefact needs to be created and evaluated using the principles of design science. Before the artefact can be instantiated, constructs of irony, social interaction and the Facebook social ecosystem are needed to form a model in which the artefact can be instantiated from. Such a model specifies the requirements of the artefact using a construct of what comment properties constitute irony and a construct of the Facebook data structure that serves as the context to distinguishing the irony from non-irony. Using the resulting model, the artefact can then be instantiated though and implementation process [21].

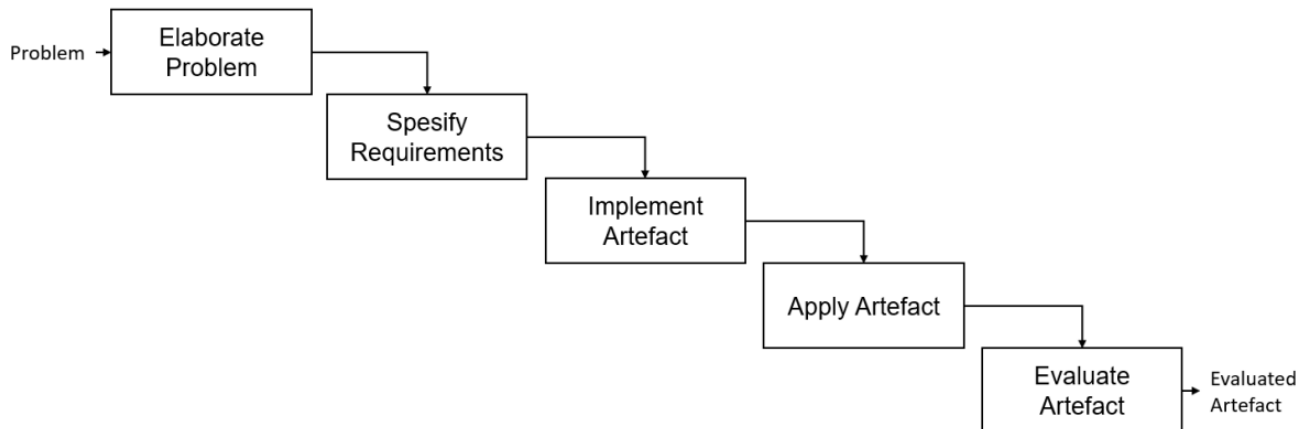


Figure 3.1: The design science process [21]

3.1 Irony Construct

In the paper *The Anatomy of a Design Theory* Shirley Gregor and David Jones describe a construct as: *"Constructs provide representations of 'entities of interest'. These entities may correspond to phenomena in the real world (e.g., a software fault) or components of the artefact (e.g., a relational table). In some cases, constructs can be decomposed into sub-constructs. Such constructs might be part of another design theory"* [21]. The communication phenomena of irony fits

this description.

Because in some cases, only some parts of a comment are be ironic, such as one of multiple sentences, unless the total interpretation of the comment’s meaning is ironic, the comment is not considered ironic. This is because if certain parts of a statement are harassing, and no another part of the same comment transforms the meaning of the harassing part into a meaning that is not harassing, for instance via irony, then whether or not it contains irony does not alter the comment’s harassing qualities. Hence the total semantic meaning of comment must be ironic for the comment to be considered ironic. Using the established definition of irony from 2.1.1, the construct of irony in the context of Facebook comments is an interpretation of the sentiment of the whole comment that is significantly different from the literal interpretation of the words written or the inverse in some capacity of the literal interpretation of the words written, to a degree where an average TV2 Nyheter Facebook reader would detect.

Separate from the definition of what irony is, is how it is recognized. Sentiment analysis text classification models that detect irony, but do not have context encoded, such as RCNN-RoBERTa, only have the information of the provided statements and their training data [16]. Therefore they can only recognize irony from the structure of a statement, and its pre-trained knowledge of the structure of ironic and non-ironic statements. Recognizing irony by a statement’s structure requires comparing which words and characters, in what order, with a trained understanding how often those specific words and characters are in that order, in ironic and non-ironic statements. I will be referring to this method of irony recognition as linguistic irony recognition.

The other method of recognizing irony is using the statement’s context. By comparing the text of preceding and or succeeding texts, the relation between the content of contextual text and a statement’s ironic value can be understood by a model during training. This allows statements that are almost impossible to detect as ironic without their context, such a single words, or statements that only express a positive or negative attitude towards something in its context, to be accurately classified. Additionally, the training of contextual recognition will also include the linguistic recognition of irony, as all the data of the statement’s text is still in the input, but also the relation between linguistic irony and the statement’s context.

Irony Recognition Examples	
Comment Example from Dataset	Recognition Method
Du verden, går det virkelig an å spise utan alkohol???	Linguistic
Fantastisk	Contextual

Table 3.1: Table showing examples of how different comment texts can be recognized as ironic with different methods

3.2 Comment Linguistics

The language used in the dataset’s comments is Norwegian with a few occasional exceptions in the form of English, Icelandic, Swedish and Danish. These were filtered out as their contribution to the accuracy of the classifier would likely only be negative. However, comments that contain a few English words, but are mainly Norwegian are kept as technical terms, slang and other references are often in English. Especially in posts that concern international topics.

The Norwegian language used in these TV2 Facebook comments is often dialect in written form with no conventions for how words within the same dialect are spelled. As norBERT 2 is trained on Bokmål and Nynorsk, it likely has little or no exposure to written dialects and its pre-training likely has little effect if any on classification of these words. Additionally some written dialect words might be confused for other words, worsening its effect.

Another facet of the language is that the grammar in these comments is quite divergent from the general grammatical rules of the Norwegian language. Out of the 2000+ comments labelled, very few did not contain any grammatical mistakes or typographical irregularities. This would make the classifier worse at detecting irony if other ways of writing similar statements are used instead of the ones it was trained on. It would also weaken any of norBERT 2 Norwegian language pattern understanding as it was mainly trained on Norwegian literature and wiki data, which likely has a high degree of grammatical rule compliance.

3.3 Facebook Post Constructs

There is a lot of data in one Facebook post and not necessarily all of it is relevant in its correlation with irony. One Facebook post contains data such as reactions, likes, shares, the post text, the post article if there is one, the comments, sub-comments of other comments, likes on comments, reactions on comments, timestamps for the actions, and lastly which people did the corresponding actions.

For an implementation of this artefact, only the text of the comment will be analysed to determine whether a comment should be labelled non-harassment ironic or not. This mean excluding all the other aforementioned Facebook post properties as well as GIFs, videos, images and any other non-text data that can be included in a comment. There are machine learning models and techniques that allow for multi-modal machine learning, which allows the combination of different data types when training a model, such as text, numbers, audio and images [9]. However, as norBERT 2 has no pre-training for images or reaction numbers, it would mean effectively training the TV2 image irony detection aspects of the hypothetical fine-tuned multi-modal norBERT from scratch. Additionally, GIFs and videos are sequences of images, and require extensive memory and training time. There may also arise complications between the longformer structure and multi-modal structure, though it appears possible as it is used in *Multi-Scale Vi-*

sion Longformer [?].

Facebook commands and other text-based functions included in Facebook, for example the function of using @ to link and notify another person also have to be removed. This is because this data often contains names or either is suspected to have little correlation with irony. While names could potentially correlate with irony, even in small dataset, as the TV2 posts have a lot of reoccurring users. However, article 5.1.c of the European Union’s General Data Protection Act regarding the use of personal data, states: *“adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed (‘data minimisation’)”* [?]. As names are personal data and have potential for misuses and at best would only have a marginal if any positive effect on the performance of the trained models, they are omitted.

Links to external information such as Wikipedia or other news articles also rarely occur. These and all other links are removed as the comment that includes use them as information to supplement or extend the meaning they express in their comment. However, these articles can be very long such as Wikipedia articles with thousands of words, to short news articles with only a few hundred. These articles do not consistently fit in either the 512 token limit or 1024 token limit model and are therefore removed. As the comment with them is often predicated upon them, the entire comment is removed if it contains a link.

The remaining data used from the Facebook posts, the linked article text if there is one, the comments and the sub-comments, as in comments that reply to other comments. The text in the comments will be labelled ironic or not using the construct of irony detailed in 2.1.1. In the case of TV2, the post text is almost always very similar to the article title it links to, or semantically superfluous, and therefore not included due to token length constraints.

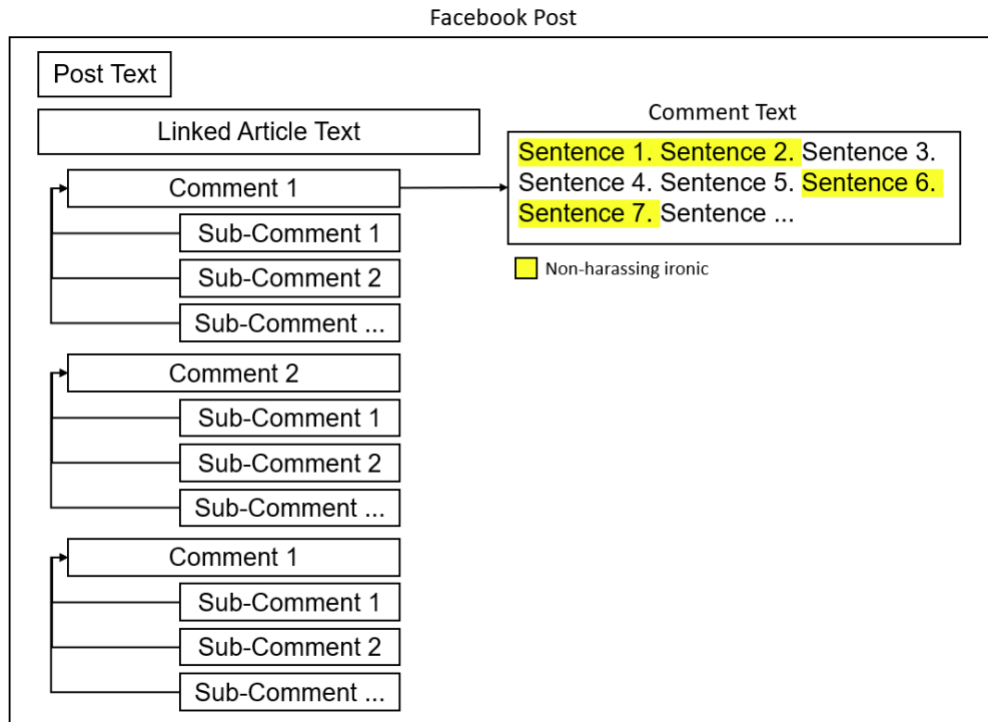


Figure 3.2: A model of the relevant constructs included in a Facebook post model.

3.4 Data Topic Distribution

In order to test the difference in performance between a model trained on comments from posts of varying topics and a model trained on comments from posts of the same or very similar topics, two datasets for the fine-tuning of the non-BERT 2 model were created. One dataset based around comments from posts about corona politics and one dataset encompassing all of the comments from the posts in the dataset, referred to as the all topic dataset. Due to the large size disparity, the corona politics dataset was supplemented with additional manually written off TV2 Nyheter Facebook post comments about corona politics. As the labelling process for creating a long supported version of a dataset is very time consuming, only the corona politics dataset was copied and converted into a long version.

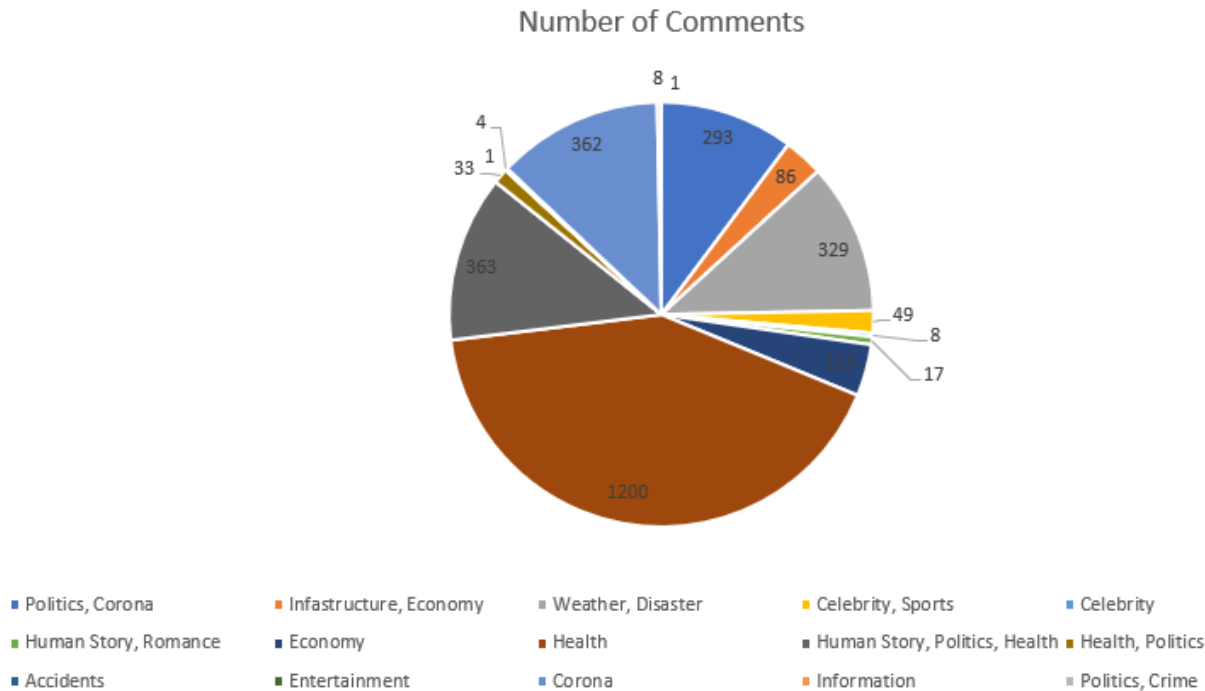


Figure 3.3: The distribution of topics and the amount of comments per topic in the TV2 dataset.

3.5 Context Encoding

Unlike the 512 token limit norBERT 2 model which only contains the comments not exceeding 512 tokens and exclusively learns linguistic irony, the LongNorBERT 2 model which contains the article and reply comments, needs to encode these in its dataset inputs in order to learn their correlation. However, there does not appear to be a public precedent for how to encode comment context in dataset inputs for sequence text classification.

Comment Context Encoding			
Article Text	Comment Text	Reply Comment Text	Irony Label
Article text about corona politics from post	Comment to post/article text		0.0
Article text about corona politics from post	Comment to post/article text	Comment replying to comment text	1.0

Table 3.2: Table showing how article text, comment text and reply text are encoded in the dataset

There is currently no public longformer model trained exclusively on Norwegian language content. However, the multilingual BERT model roBERTa, which is trained on a long list of languages. And there is a version of roBERTa con-

verted to a long version. However, the pre-training part of this conversion process was exclusively done on English text using the wiki103 corpus. Additionally, the model's size exceeded the thesis's available GTX 980 GPU's memory, even when using two of them. However, one of the creators of the longformer Iz Beltagy wrote a python script on his github page that can convert a 512 token limit transformer model into a higher token limit version of the model[8]. Using this script norBERT 2 was converted from a token limit of 512 into a custom long version named LongNorBERT2 with a token limit of 1024. norBERT 2's tokenizer was also converted to a long version.

While the new model and tokenizer allows processing of tokenized texts larger than 512 tokens, it is not pre-trained on a large corpus of longer Norwegian text inputs. According to figure 3.4, no MLM pre-training has a notable, but not catastrophic effect on performance. This step was skipped due to access to large Norwegian long text input corpus being limited, only two GTX 980 GPUs are available for training, and time constraints. As also seen in figure 3.4, the token limit can be extended higher than 1024 tokens, and the standard limit for longformer models is 4096 tokens, however it increases model size and only 5 of long context encoded inputs in the dataset exceeded 1024 tokens.

Model	Accuracy / Δ
Longformer (seqLen: 4,096)	73.8
RoBERTa-base (seqLen: 512)	72.4 / -1.4
Longformer (seqLen: 4,096, 15 epochs)	75.0 / +1.2
Longformer (seqLen: 512, attention: n^2)	71.7 / -2.1
Longformer (seqLen: 2,048)	73.1 / -0.7
Longformer (no MLM pretraining)	73.2 / -0.6
Longformer (no linear proj.)	72.2 / -1.6
Longformer (no linear proj. no global atten.)	65.5 / -8.3
Longformer (pretrain extra position embed. only)	73.5 / -0.3

Figure 3.4: Performance metrics of differently trained longformer models and RoBERTa

3.6 Threats to Validity

Most models trained to perform sentiment analysis, such as those created for kaggle competitions are usually trained on datasets that contain at least tens of thousands of inputs and often as high as multiple hundred thousands of inputs. Comparatively the dataset used for this model is quite small with only 1000 inputs. This disparity already indicated problems during the training of the corona politics 512 token limit model, as validation loss fluctuated wildly, and quickly began increasing after only one or two epochs, as the model becomes overfitted.

This indicates noise in the dataset, and as the data and model complexity is equal or lower than the long version, which did not have the same issues, it is likely due to the dataset's small size. The all topic 512 token limit model, did not have this issue and its dataset is more than twice as large.

Even though the long and all topic models did not show signs of overfitting, they still have comparatively small datasets and whether the results and behaviours exhibited by these models remain consistent when trained on a larger dataset, across multiple domains and or topics, is also uncertain.

Additionally, the pre-train MLM step skipped, as mentioned in 3.5, and the language problems described in 3.2 also have a low, but inconclusive effect on the models' results.

Because the aforementioned problems could have a indeterminable low or high impact on the models' performances, the models' accuracy could be random luck and therefore jeopardize the validity of their results and entailing conclusions.

Chapter 4

Methods

4.1 Converting Model to Long

The first step to training a norBERT 2 model for detecting irony with context included inputs, is to convert the original norBERT 2 into a long version that supports longer token lengths. While there are other models that support longer token input lengths, such as the BigBird model [23], the conversion of the 512 tokens limit to longer input lengths process has publicly only been done with longformer. In order to make the norBERT 2 model support token input lengths longer than 512, we have to change its self-attention from its default BertSelfAttention to a long version of self-attention that extends LongformerSelfAttention, but with a few modifications. The code used is a modified version of the updated `convert_model_to_long.py` by Adam Wawrzynski [1], updated from the original script from the github of one of the authors of the longformer paper Iz Beltagy [8][2]. The python class for the modified self-attention class:

```
[breaklines]
class norBERTLongSelfAttention(LongformerSelfAttention):
    def forward(
        self,
        hidden_states,
        attention_mask=None,
        head_mask=None,
        encoder_hidden_states=None,
        encoder_attention_mask=None,
        past_key_value=None,
        output_attentions=False,
    ):

        hidden_states = hidden_states.transpose(0, 1)

        # project hidden states
        query_vectors = self.query(hidden_states)
        key_vectors = self.key(hidden_states)
        value_vectors = self.value(hidden_states)
```

```

## Lines below has been changed

attention_mask = attention_mask.squeeze(dim=2).squeeze(dim=1)

# is index masked or global attention
is_index_masked = attention_mask < 0
is_index_global_attn = attention_mask > 0
is_global_attn = any(is_index_global_attn.flatten())

## End of modification

seq_len, batch_size, embed_dim = hidden_states.size()
assert (
    embed_dim == self.embed_dim
), f"hidden_states should have embed_dim = {self.embed_dim},
but has {embed_dim}"

# normalize query
query_vectors /= math.sqrt(self.head_dim)

query_vectors = query_vectors.view(seq_len, batch_size, self.num_heads,
self.head_dim).transpose(0, 1)
key_vectors = key_vectors.view(seq_len, batch_size, self.num_heads,
self.head_dim).transpose(0, 1)

attn_scores = self._sliding_chunks_query_key_matmul(
    query_vectors, key_vectors, self.one_sided_attn_window_size
)

# values to pad for attention probs

## Lines below has been changed

remove_from_windowed_attention_mask = (attention_mask != 0).unsqueeze(0)
1).unsqueeze(dim=-1)

## End of modification

# cast to fp32/fp16 then replace 1's with -inf
float_mask = remove_from_windowed_attention_mask.type_as(query_vectors)
    remove_from_windowed_attention_mask, -10000.0
)
# diagonal mask with zeros everywhere and -inf inplace of padding
diagonal_mask = self._sliding_chunks_query_key_matmul(
    float_mask.new_ones(size=float_mask.size()), float_mask,
self.one_sided_attn_window_size

```



```

)

# pad local attention probs
attn_scores += diagonal_mask

assert list(attn_scores.size()) == [
    batch_size,
    seq_len,
    self.num_heads,
    self.one_sided_attn_window_size * 2 + 1,
], f"local_attn_probs should be of size ({batch_size}, {seq_len},
{self.num_heads}, {self.one_sided_attn_window_size * 2 + 1}), but is
of size {attn_scores.size()}"

# compute local attention probs from global attention keys and
contact over window dim
if is_global_attn:
    # compute global attn indices required through out forward
fn
    (
        max_num_global_attn_indices,
        is_index_global_attn_nonzero,
        is_local_index_global_attn_nonzero,
        is_local_index_no_global_attn_nonzero,
    ) = self._get_global_attn_indices(is_index_global_attn)
    # calculate global attn probs from global key

    global_key_attn_scores = self._concat_with_global_key_attn_probs(
        query_vectors=query_vectors,
        key_vectors=key_vectors,
        max_num_global_attn_indices=max_num_global_attn_indices,
        is_index_global_attn_nonzero=is_index_global_attn_nonzero,
        is_local_index_global_attn_nonzero=is_local_index_global_attn_
        is_local_index_no_global_attn_nonzero=is_local_index_no_global
    )
    # concat to local_attn_probs
    # (batch_size, seq_len, num_heads, extra attention count
+ 2*window+1)
    attn_scores = torch.cat((global_key_attn_scores, attn_scores),
dim=-1)

    # free memory
    del global_key_attn_scores

    attn_probs = F.softmax(attn_scores, dim=-1, dtype=torch.float32)
    # use fp32 for numerical stability

```

```

        # softmax sometimes inserts NaN if all positions are masked,
        # replace them with 0
        attn_probs = torch.masked_fill(attn_probs, is_index_masked[:,
        :, None, None], 0.0)
        attn_probs = attn_probs.type_as(attn_scores)

        # free memory
        del attn_scores

        # apply dropout
        attn_probs = F.dropout(attn_probs, p=self.dropout, training=self.training)

        value_vectors = value_vectors.view(seq_len, batch_size, self.num_heads,
        self.head_dim).transpose(0, 1)

        # compute local attention output with global attention value
        # and add
        if is_global_attn:
            # compute sum of global and local attn
            attn_output = self._compute_attn_output_with_global_indices(
                value_vectors=value_vectors,
                attn_probs=attn_probs,
                max_num_global_attn_indices=max_num_global_attn_indices,
                is_index_global_attn_nonzero=is_index_global_attn_nonzero,
                is_local_index_global_attn_nonzero=is_local_index_global_attn_n
            )
        else:
            # compute local attn only
            attn_output = self._sliding_chunks_matmul_attn_probs_value(
                attn_probs, value_vectors, self.one_sided_attn_window_size
            )

        assert attn_output.size() == (batch_size, seq_len, self.num_heads,
        self.head_dim), "Unexpected size"
        attn_output = attn_output.transpose(0, 1).reshape(seq_len, batch_size,
        embed_dim).contiguous()

        # compute value for global attention and overwrite to attention
        # output
        # TODO: remove the redundant computation
        if is_global_attn:
            global_attn_output, global_attn_probs = self._compute_global_attn_o
            hidden_states=hidden_states,
            max_num_global_attn_indices=max_num_global_attn_indices,
            is_local_index_global_attn_nonzero=is_local_index_global_attn_n
            is_index_global_attn_nonzero=is_index_global_attn_nonzero,
            is_local_index_no_global_attn_nonzero=is_local_index_no_global_

```

```

        is_index_masked=is_index_masked,
    )

    # get only non zero global attn output
    nonzero_global_attn_output = global_attn_output[
        is_local_index_global_attn_nonzero[0], :, is_local_index_global_a
    ]

    # overwrite values with global attention
    attn_output[is_index_global_attn_nonzero[::-1]] = nonzero_global_a
        len(is_local_index_global_attn_nonzero[0]), -1
    )
    # The attention weights for tokens with global attention
are
    # just filler values, they were never used to compute the
output.
    # Fill with 0 now, the correct values are in 'global_attn_probs'.
    attn_probs[is_index_global_attn_nonzero] = 0

    outputs = (attn_output.transpose(0, 1),)

    if output attentions:
        outputs += (attn_probs,)

    return outputs + (global_attn_probs,) if (is_global_attn and
output attentions) else outputs

```

Using two custom classes that modify the default BertForSequenceClassification and BertModel, the classes can replace their default self-attention class with the new norBERTLongSelfAttention class.

```
[breaklines]
class norBERTLongForSequenceClassification(BertForSequenceClassification):
    def __init__(self, config):
        super().__init__(config)
        self.bert.encoder.layer = nn.ModuleList([norBERTLongSelfAttention(config,
layer_id=index) for index in range(config.num_hidden_layers)])

class norBERTLongModel(BertModel):
    def __init__(self, config):
        super().__init__(config)
        self.encoder.layer = nn.ModuleList([norBERTLongSelfAttention(config,
layer_id=index) for index in range(config.num_hidden_layers)])
```

The last part of the process is a function that loads the new long self-attention model and its respective tokenizer, before altering the model's embeddings to their new max positions, which in the case of LongNorBERT2 is 1024, before saving the new altered model and its tokenizer. The attention window of the model is not modified and remains at 512.

```
[breaklines]
def create_long_model(
    initialization_model,
    initialization_tokenizer,
    save_model_to,
    attention_window,
    max_pos
):
    model = BertForSequenceClassification.from_pretrained(initialization_model,
num_labels=1)
    tokenizer = BertTokenizerFast.from_pretrained(initialization_tokenizer,
model_max_length=max_pos)
    config = model.config

    # extend position embeddings
    tokenizer.model_max_length = max_pos
    tokenizer.init_kwargs['model_max_length'] = max_pos
    current_max_pos, embed_size = model.bert.embeddings.position_embeddings.weight.size()
    config.max_position_embeddings = max_pos
    assert max_pos > current_max_pos

    # allocate a larger position embedding matrix
    new_pos_embed = model.bert.embeddings.position_embeddings.weight.new_empty(
embed_size)
```

```

# copy position embeddings over and over to initialize the new position
embeddings
k = 0
step = current_max_pos
while k < max_pos - 1:
    new_pos_embed[k:(k + step)] = model.bert.embeddings.position_embedding
    k += step
model.bert.embeddings.position_embeddings.weight.data = new_pos_embed
model.bert.embeddings.position_ids.data = torch.tensor([i for i in
range(max_pos)]).reshape(1, max_pos) # v4.0.0+

# replace the `modeling_bert.BertSelfAttention` object with `norBERTLongSe
config.attention_window = [attention_window] * config.num_hidden_layers
for i, layer in enumerate(model.bert.encoder.layer):
    longformer_self_attn = norBERTLongSelfAttention(config, layer_id=i)
    longformer_self_attn.query = copy.deepcopy(layer.attention.self.query)
    longformer_self_attn.key = copy.deepcopy(layer.attention.self.key)
    longformer_self_attn.value = copy.deepcopy(layer.attention.self.value)

    longformer_self_attn.query_global = copy.deepcopy(layer.attention.self
longformer_self_attn.key_global = copy.deepcopy(layer.attention.self.k
longformer_self_attn.value_global = copy.deepcopy(layer.attention.self

    layer.attention.self = longformer_self_attn

```

4.2 Dataset and Labelling

The dataset provided by TV2 consists of 1000 comments from the TV2 Nyheter Facebook page spread across 23 different Facebook posts, all from around the 9th of January 2022. However, as they were gathered around the 9th of January, most of the posts received more comments after having being gathered into the dataset. These additional comments were manually written into the dataset in order to have a holistic collection of the complete comment ecosystem for each post.

4.2.1 Long and short dataset

Datasets		
Dataset Name	Comment Amount	Irony Percentage
Long TV2 Corona Politics	826	9.01%
Short TV2 Corona Politics	816	9.01
Short TV2 All Topic	2064	5.78
Validation TV2 Corona Politics	290	6.79

Table 4.1: The datasets' basic metrics

For the short dataset, each comment is individually separated, while in the long dataset, the article, comment and reply are included as separate columns. In order to process the data into a usable format and clean some data entries, all special characters such as commas, punctuation marks and dashes before the first word are removed. Additionally all newlines are converted to spaces. Names were manually removed, and in the cases where the statement no longer made grammatical sense without them, the name was replaced by a fitting pronoun. Comments that only included names were removed entirely. All additional spaces at the end of statements were also removed as they are not visible to a human reader, but add inconsistency to the dataset. The long dataset contains more entries than its short counterpart due to their being some comments that alone exceed the 512 token limit, but where the article, comment and potential reply together tokenized did not exceed 1024 tokens. Five entries were removed from the long dataset due to them exceeding the 1024 token limit.

Short Dataset	
Comment Text	Irony Label
Hun skriver på dialekten sin! Og det er vel lov??	0.0
bedre å stole på youtube	1.0
Sånn var det ikke med den forrige regjeringen.	0.0

Table 4.2: Example of a few short dataset inputs


Long Dataset			
Article Text	Comment Text	Reply Comment Text	Irony Label
Raymond Johansen: – Må se om smittetalene...	Av og til tar Raymond liksom litt av. Hvordan skal...		0.0
Raymond Johansen: – Må se om smittetalene...	Av og til tar Raymond liksom litt av. Hvordan skal...	les nå hva han faktisk skriver.	0.0
Finansminister Trygve Slagsvold Vedum (Sp) vil...	Du verden, går det virkelig an å spise utan alkohol???		1.0
Finansminister Trygve Slagsvold Vedum (Sp) vil...	Du verden, går det virkelig an å spise utan alkohol???	Yup fungerer på samme måte som med alkohol du åpner kjæften stapper innpå lukker munnen tygger og svelger 	1.0

Table 4.3: Example of a few long dataset inputs

The inputs for the datasets are converted into tokens in the form of input ids and attention masks. The models are then trained with these tokens as the input.

```
input ids: [102, 614, 146, 204, 1563, 161, 154, 1822, 106, 10998, 755,
21017, 320, 682, 149, 426, 21985, 50073, 50065, 659, 1822, 106, 10998,
3036, 737 103]\newline
```

```
attention mask: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1]
```

```
[[CLS], Dette, er, et, eksempel, på, en, tok, ##en, ##ifi, ##sert,
setning, ved, bruk, av, nor, ##BER, ##T, ##2, sin, tok, ##en, ##ifi,
##serer, ., [SEP]]
```

Figure 4.1: Example of the input ids, attention mask and converted tokens generated by using *norBERT 2's* tokenizer, which was used for the short models, on the Norwegian example phrase: "Dette er et eksempel på en tokenifisert setning ved bruk av *norBERT2* sin tokenifiserer."

The hashtag character means that the token will be considered as part of the token before it. Special tokens, such as [CLS] and [SEP] are tokens that signify the beginning, separation and end of texts. In order to encode the context texts into inputs, [SEP] tokens are used between the article text, comment text and reply text if there is one, to signify that the texts are to be treated as separate texts.

```
input ids: [102, 13083, 4092, 4213, 737, 103, 614, 146, 204, 1563, 161,
154, 1822, 106, 10998, 755, 21017, 320, 682, 149, 9847, 12462, 21985,
50073, 50065, 659, 1822, 106, 10998, 3036, 737, 103, 13083, 1144, 143,
2649, 737, 103]\newline
```

```
attention mask: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

```
['[CLS]', 'Eksempel', 'artikkel', 'tekst', '.', '[SEP]', 'Dette',
'er', 'et', 'eksempel', 'på', 'en', 'tok', '##en', '##ifi', '##sert',
'setning', 'ved', 'bruk', 'av', 'Long', '##Nor', '##BER', '##T', '##2',
'sin', 'tok', '##en', '##ifi', '##serer', '.', '[SEP]', 'Eksempel',
'svar', 'til', 'kommentar', '.', '[SEP]']
```

Figure 4.2: Example of the input ids, attention mask and converted tokens generated by using LongNorBERT 2’s tokenizer, which was used for the long model, on the Norwegian example phrase with article and reply context texts: ”Eksempel artikkel tekst. [SEP] Dette er et eksempel på en tokenifisert setning ved bruk av LongNorBERT2 sin tokenifiserer. [SEP] Eksempel svar til kommentar.”

To prevent a difference in shape between each input. All inputs are padded to their maximum allowed length, which is 512 for the 512 token length limit short models and 1024 for the long model. The padding adds 0 at the end to both the input ids and attention mask until their lists reach their maximum length.

4.3 Implementation

Using only the comments and replies as separate inputs, the corona politics dataset was used to train one 512 token limit model fine-tuned from norBERT 2, called TV2FacebookCoronaPoliticsIronyShortNorBERT2, and with the articles, comments and replies each in one input, one 1024 token limit model, called TV2FacebookCoronaPoliticsIronyLongNorBERT2. Lastly, using the dataset containing data from the all the posts, another 512 token limit fine-tuned norBERT 2 model was trained called TV2FacebookIronyShortNorBERT2. All models were trained with BCE loss as their criterion.

The training was done using Pytorch and Pytorch Lightning, due to Pytorch Lightning’s speed and multi-GPU support, using code modified from Curiously’s article *Multi-label Text Classification with BERT and PyTorch Lightning* [4]. The norBERTLongForSequenceClassification class is replaced by the BertModel class for the short models.

```
[breaklines]
```

```
class IronyLongCommentTagger(pl.LightningModule):
```

```
    def __init__(self, n_classes: int, n_training_steps=None, n_warmup_steps=None):
        super().__init__()
```



```
self.bert = torch.nn.Lstm(attention_window=512, num_labels=1, return_dict=True)
self.classifier = nn.Linear(self.bert.config.hidden_size, n_classes)
self.n_training_steps = n_training_steps
self.n_warmup_steps = n_warmup_steps
self.criterion = nn.BCELoss()

def forward(self, input_ids, attention_mask, label=None):

    output = self.bert(input_ids, attention_mask=attention_mask)
    output = torch.sigmoid(output[0])
    loss = 0
    if label is not None:
        loss = self.criterion(output, label)

    return loss, output

def training_step(self, batch, batch_idx):
    input_ids = batch["input_ids"]
    attention_mask = batch["attention_mask"]
    label = batch["label"]
    loss, outputs = self(input_ids, attention_mask, label)
    self.log("train_loss", loss, prog_bar=True, logger=True)
    return {"loss": loss, "predictions": outputs.detach(), "label": label}

def validation_step(self, batch, batch_idx):
    input_ids = batch["input_ids"]
    attention_mask = batch["attention_mask"]
    label = batch["label"]
    loss, outputs = self(input_ids, attention_mask, label)
    self.log("val_loss", loss, prog_bar=True, logger=True)
    return loss

def test_step(self, batch, batch_idx):
    input_ids = batch["input_ids"]
    attention_mask = batch["attention_mask"]
    label = batch["label"]
    loss, outputs = self(input_ids, attention_mask, label)
    self.log("test_loss", loss, prog_bar=True, logger=True)
    return loss

def training_epoch_end(self, outputs):
    label = []
    predictions = []
    for output in outputs:
        for out_label in output["label"].detach().cpu():
```

```

        label.append(out_label)
    for out_predictions in output["predictions"].detach().cpu():
        predictions.append(out_predictions)

label = torch.stack(label).int()
predictions = torch.stack(predictions)

for i, name in enumerate(LABEL_COLUMNS):
    class_roc_auc = auROC(predictions[:, i], label[:, i])
    self.logger.experiment.add_scalar(f"{name}_roc_auc/Train", class_roc_auc,
self.current_epoch)

def configure_optimizers(self):
    optimizer = AdamW(self.parameters(), lr=2e-5)
    scheduler = get_linear_schedule_with_warmup(
        optimizer,
        num_warmup_steps=self.n_warmup_steps,
        num_training_steps=self.n_training_steps
    )
    return dict(
        optimizer=optimizer,
        lr_scheduler=dict(scheduler=scheduler, interval='step'),
    )

```

These are the training parameters of the three final models and the training step checkpoints with the minimum validation loss that were used for evaluation:

TV2FacebookCoronaPoliticsIronyLongNorBERT2	
Training Parameter	Value
Strategy	ddp sharded GPU
Batch Size	1
Accumulate Grad Batches	1
Epochs	30
Learning Rate	5e-5
Total Training Steps	24840
Warm-up Steps	4968
Training Time	8 Hours
Lowest Validation Loss	0.194
Lowest Val Loss Train Step	12,450

Table 4.4: Training parameters for the TV2FacebookCoronaPoliticsIronyLongNorBERT2 model.

TV2FacebookCoronaPoliticsIronyShortNorBERT2	
Training Parameter	Value
Strategy	GPU
Batch Size	1
Accumulate Grad Batches	1
Epochs	10
Learning Rate	5e-5
Total Training Steps	8150
Warm-up Steps	1630
Training Time	2 Hours
Lowest Validation Loss	0.216
Lowest Val Loss Train Step	815

Table 4.5: Training parameters for the TV2FacebookCoronaPoliticsIronyShortNorBERT2 model.

TV2FacebookIronyShortNorBERT2	
Training Parameter	Value
Strategy	GPU
Batch Size	1
Accumulate Grad Batches	1
Epochs	10
Learning Rate	5e-5
Total Training Steps	18320
Warm-up Steps	3664
Training Time	4 Hours
Lowest Validation Loss	0.184
Lowest Val Loss Train Step	3664

Table 4.6: Training parameters for the all topic TV2FacebookIronyShortNorBERT2 model.

4.4 Evaluation

Evaluation of the artefact was conducted by applying the artefact to tokenized text inputs from the validation dataset to receive irony value predictions for each text input. The long model text inputs also included the article and potential reply context texts from the validation set. Various metrics commonly used to gauge the performance of classification models were then used on the resulting predictions. In addition to the raw true positive, true negative, false positive and false positive metrics, the metrics used were accuracy, precision, recall, F1 score and ROC.

Accuracy shows how many of all the predictions made are correct predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure 4.3: Formula for accuracy metric

Precision indicates how many of the positive predictions actually were correct predictions.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Figure 4.4: Formula for precision metric

Recall denotes how many of the true positives were correctly predicted.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Figure 4.5: Formula for recall metric

F1 otherwise known as F-score, is a harmonious combination of precision and recall, which measures accuracy of predictions by displaying the relation between precision and recall.

$$F_1 = \frac{2}{\frac{1}{\text{recall}} \times \frac{1}{\text{precision}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Figure 4.6: Formula for F1 score metric

The ROC curve, which is short for the receiver operating characteristic curve, is a way to graph the performance of classification by plotting the true positive rate against the false positive rate at different classification threshold values.

$$TPR = \frac{TP}{TP + FN} \quad FPR = \frac{FP}{FP + TN}$$

Figure 4.7: ROC by comparing TPR and FPR values

Chapter 5

Results and Discussion

5.1 Results and Analysis

When giving the trained model a list of tokenized input ids and an attention mask of a tokenized text it returns a value between 0.0 and 1.0, signifying its predicted irony value for the tokenized text. However, what threshold or cutoff value between 0.0 and 1.0 that determines whether the input is classified as ironic or not needs to be determined and is likely unique for each of the three models. In order to determine which threshold value leads to the best model classification performance, all values between 0.0 and 1.0 in 0.01 increments are tested. There was no difference between increments of 0.01 and 0.001 for values in terms of classification results.

5.1.1 Models' Metrics

The validation set which consists of comments and the article from a corona politics topic post, has a percentage of irony at 6.79 percent, which is more than 2 percent lower than the corona politics dataset, but 1 percent higher than the all topic dataset. Its high comment uniqueness and complexity compared to the datasets makes it a rather difficult dataset to accurately classify, and all of the models will therefore likely perform badly. With more, varied and complex data in the datasets for the models to train on, they would likely perform better.

Validation Dataset	
Distribution	Amount
Total Comments	283
Non-Ironic Comments	265
Ironic Comments	18
Ironic Percentage	6.792

Table 5.1: The validation set's metrics

In the graph of accurately classified ironic comments, otherwise referred to as true positives, at value 0.0, all 18 comments are classified as ironic, but so are all non-ironic, as shown in the graph of false positives. While the all topic and long models immediately wrongly classify a few ironic comments, they do so in slower increments as the threshold values increases. The corona politics short however, retains almost all of the true positives, until about 0.24, where it reaches 0 true positives. The same happens to the false positives graph for the same model, showing that the corona politics short model predicts no comment's ironic value to be above 0.24, and that its volatile nature indicates that it struggles to separate ironic comments from non-ironic comments. Its unique results compared to the other models is likely due to its smaller dataset and non-contextual encoding.

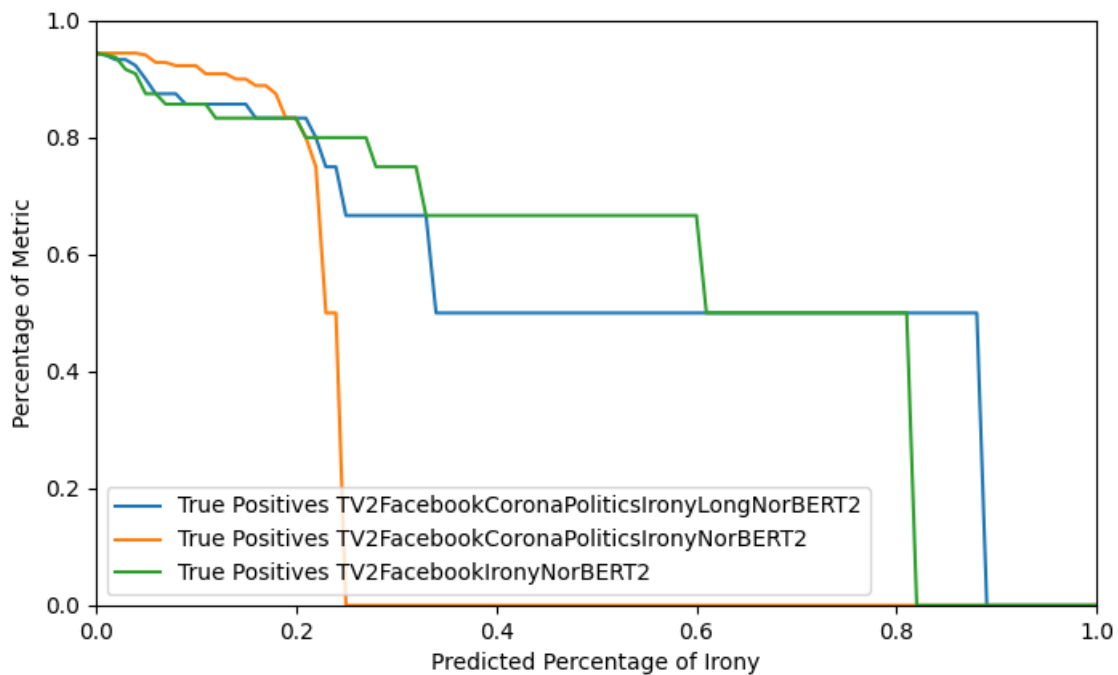


Figure 5.1: Graph of the number of true positive predictions given classification threshold values by the three models on the validation set

As assumed, all of the models have a higher false positive number than true positives for all except very high value thresholds. Notably, the false positive classification number for the long model immediately drops significantly below the two other models until significantly higher values. It seems that the long model outperforms the other models at low threshold values, but is beaten the by non-topic short model at threshold values above 0.5.

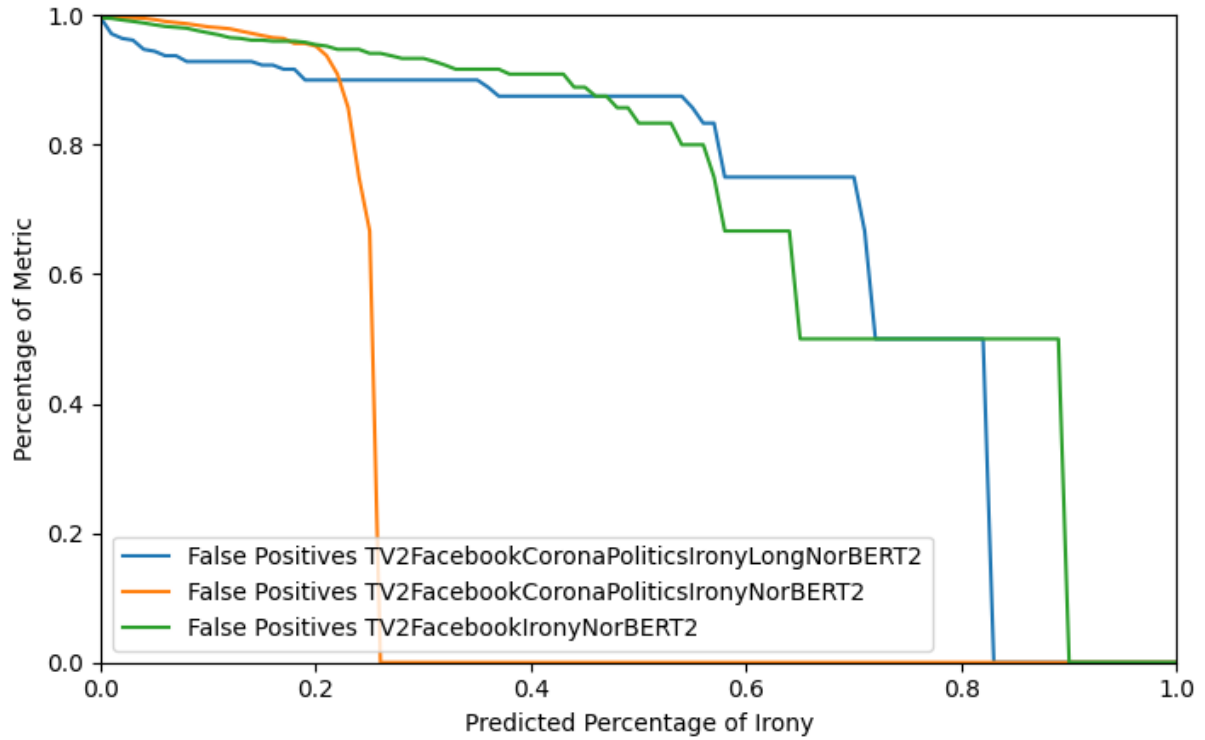


Figure 5.2: Graph of the number of false positive predictions given classification threshold values by the three models on the validation set

This is even more apparent when looking at the distance between true positives and false positives for each threshold value. Calculated as the absolute value of true positives minus false positives. In this graph it is shown that the long model has comparatively a significantly smaller distance between wrong and correct classifications compared to the non-topic short model, until about threshold value 0.5.

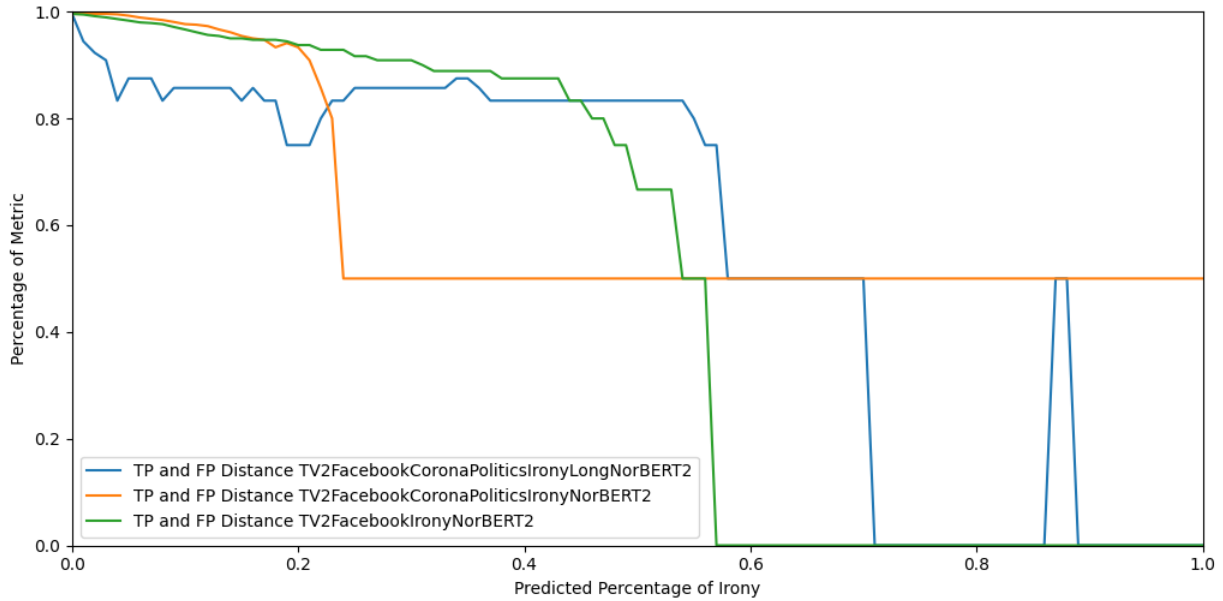


Figure 5.3: Graph of the numerical difference between the TP and FP number predictions given classification threshold values by the three models on the validation set

Comparing the specific numbers for true positives and false positives for each value we can see the extreme drop of 18/265 to 17/35 with only a 0.01 value difference for the long model, while its short counterpart remains at 18/265 and the non-topic short model only drops to 17/207.

Specific TP and FP Results			
Irony Threshold	TV2FacebookCoronaPoliticsIrony-LongNorBERT2	TV2FacebookCoronaPoliticsIronyNorBERT2	TV2FacebookIronyNorBERT2
Value	TP/FP	TP/FP	TP/FP
0.0	18/265	18/265	18/265
0.01	17/35	18/265	17/207
0.02	15/28	18/265	16/136
0.03	15/26	18/265	12/106
0.04	13/19	18/220	11/84
0.05	10/18	17/152	8/68
0.06	8/16	14/106	8/58
0.07	8/16	14/89	7/54
0.08	8/14	13/77	7/50
0.09	7/14	13/65	7/42
0.1	7/14	13/56	7/37
0.11	7/14	11/52	7/33
0.12	7/14	11/48	6/29
0.13	7/14	11/41	6/28
0.14	7/14	10/36	6/26
0.15	7/13	10/32	6/26
0.16	6/13	9/29	6/25
0.17	6/12	9/28	6/25

0.18	6/12	8/23	6/25
0.19	6/10	6/23	6/24
0.2	6/10	6/21	6/22
0.21	6/10	5/16	5/21
0.22	5/10	4/11	5/19
0.23	4/10	2/7	5/19
0.24	4/10	2/4	5/19
0.25	3/10	1/3	5/17
0.26	3/10	0/0	5/17
0.27	3/10	0/0	5/16
0.28	3/10	0/0	4/15
0.29	3/10	0/0	4/15
0.3	3/10	0/0	4/15
0.31	3/10	0/0	4/14
0.32	3/10	0/0	4/13
0.33	3/10	0/0	3/12
0.34	2/10	0/0	3/12
0.35	2/10	0/0	3/12
0.36	2/9	0/0	3/12
0.37	2/8	0/0	3/12
0.38	2/8	0/0	3/11
0.39	2/8	0/0	3/11
0.4	2/8	0/0	3/11
0.41	2/8	0/0	3/11
0.42	2/8	0/0	3/11
0.43	2/8	0/0	3/11
0.44	2/8	0/0	3/9
0.45	2/8	0/0	3/9
0.46	2/8	0/0	3/8
0.47	2/8	0/0	3/8
0.48	2/8	0/0	3/7
0.49	2/8	0/0	3/7
0.5	2/8	0/0	3/6
0.51	2/8	0/0	3/6
0.52	2/8	0/0	3/6
0.53	2/8	0/0	3/6
0.54	2/8	0/0	3/5
0.55	2/7	0/0	3/5
0.56	2/6	0/0	3/5
0.57	2/6	0/0	3/4
0.58	2/4	0/0	3/3
0.59	2/4	0/0	3/3
0.6	2/4	0/0	3/3
0.61	2/4	0/0	2/3
0.62	2/4	0/0	2/3
0.63	2/4	0/0	2/3
0.64	2/4	0/0	2/3

0.65	2/4	0/0	2/2
0.66	2/4	0/0	2/2
0.67	2/4	0/0	2/2
0.68	2/4	0/0	2/2
0.69	2/4	0/0	2/2
0.7	2/4	0/0	2/2
0.71	2/3	0/0	2/2
0.72	2/2	0/0	2/2
0.73	2/2	0/0	2/2
0.74	2/2	0/0	2/2
0.75	2/2	0/0	2/2
0.76	2/2	0/0	2/2
0.77	2/2	0/0	2/2
0.78	2/2	0/0	2/2
0.79	2/2	0/0	2/2
0.8	2/2	0/0	2/2
0.81	2/2	0/0	2/2
0.82	2/2	0/0	1/2
0.83	2/1	0/0	1/2
0.84	2/1	0/0	1/2
0.85	2/1	0/0	1/2
0.86	2/1	0/0	1/2
0.87	2/0	0/0	1/2
0.88	2/0	0/0	1/2
0.89	1/0	0/0	1/2
0.9	1/0	0/0	1/1
0.91	1/0	0/0	1/1
0.92	1/0	0/0	1/1
0.93	1/0	0/0	1/1
0.94	1/0	0/0	1/1
0.95	1/0	0/0	1/1
0.96	1/0	0/0	1/1
0.97	1/0	0/0	0/1
0.98	1/0	0/0	0/0
0.99	1/0	0/0	0/0
1.0	0/0	0/0	0/0

The observation of the long model's superior early performance is further corroborated by looking at the graphs for the other evaluation metrics; F1 score, precision, recall, accuracy and the ROC curve.

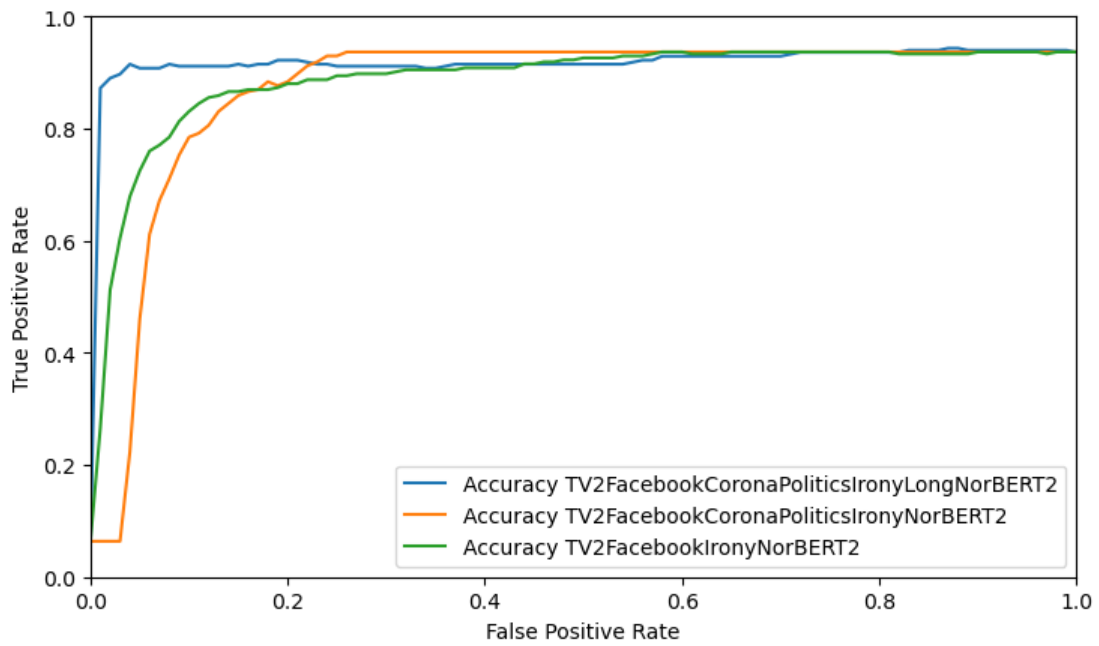


Figure 5.4: Graph of the accuracy of predictions given classification threshold values by the three models on the validation set

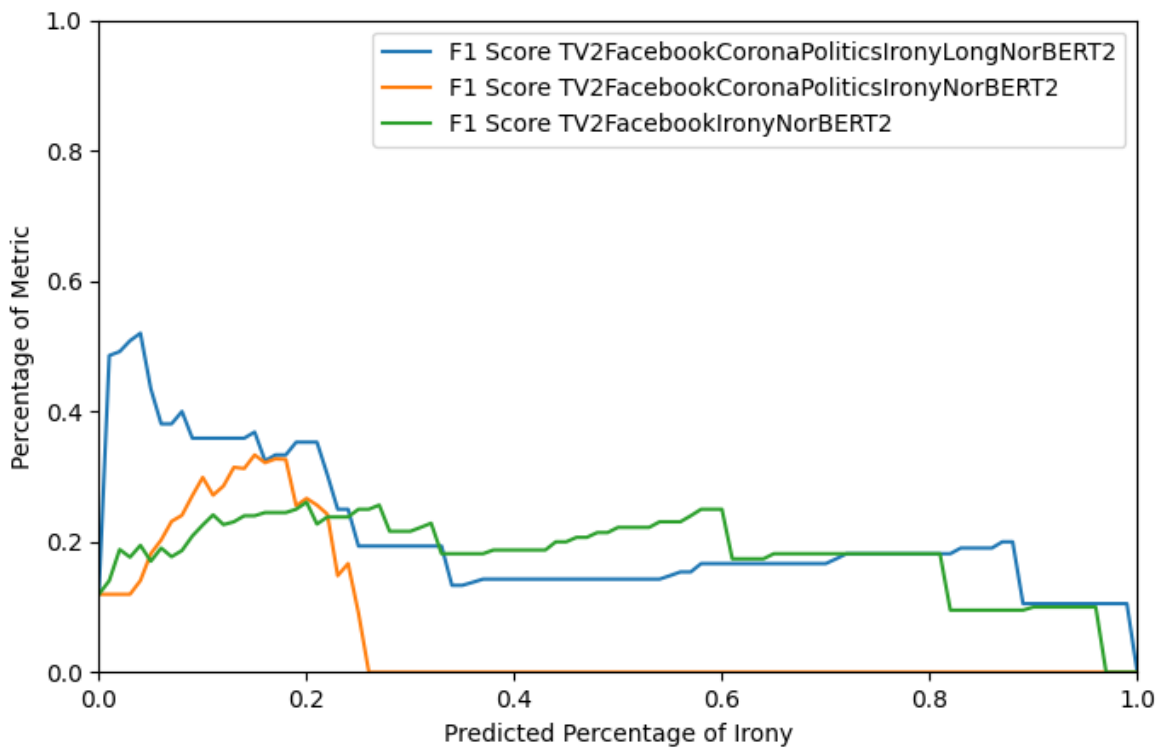


Figure 5.5: Graph of the F1 scores the of predictions given classification threshold values by the three models on the validation set

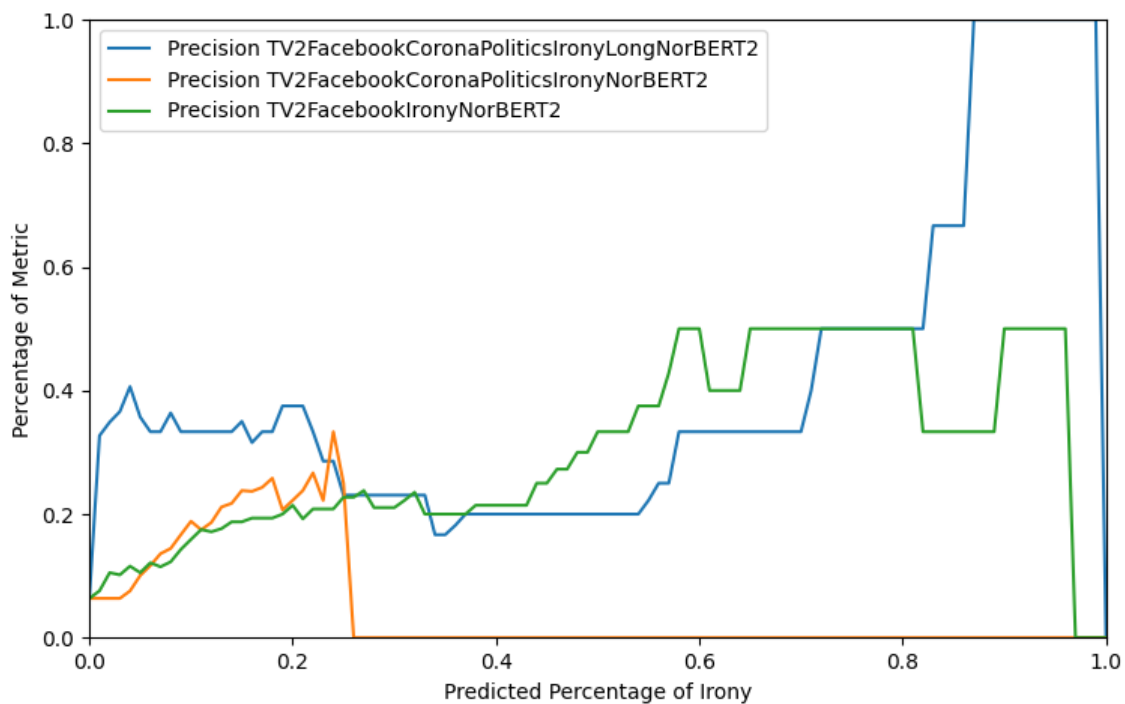


Figure 5.6: Graph of the precision of predictions given classification threshold values by the three models on the validation set

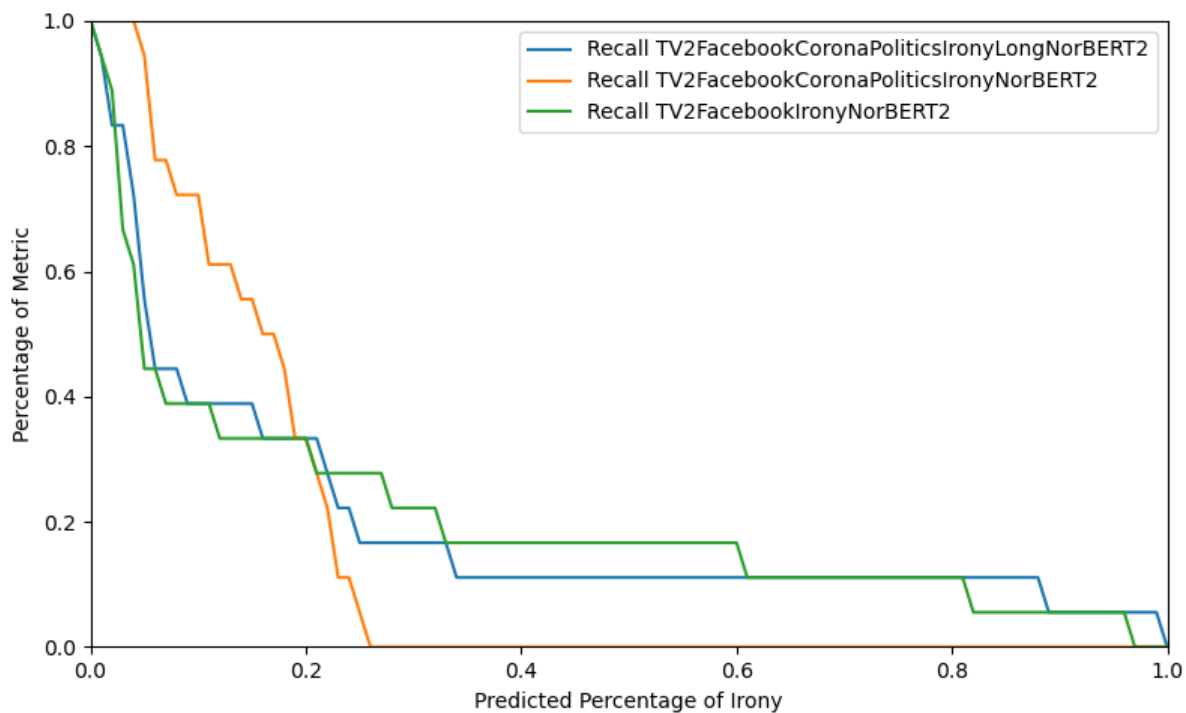


Figure 5.7: Graph of the recall of predictions given classification threshold values by the three models on the validation set

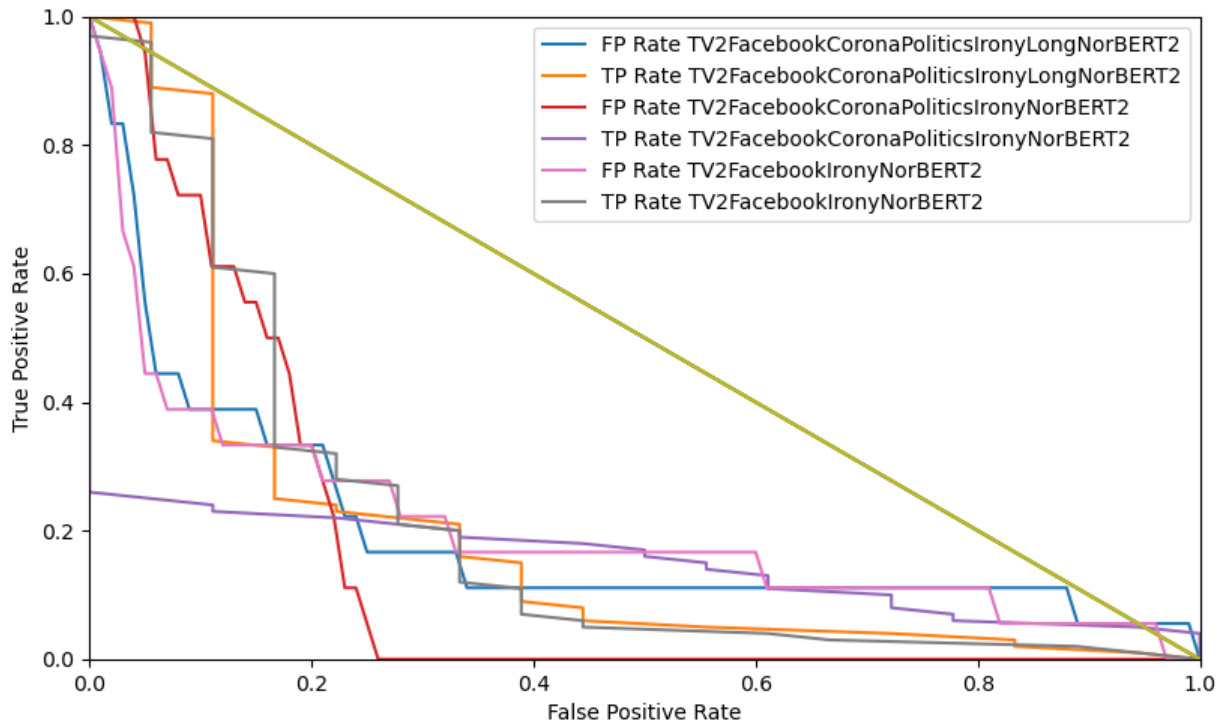


Figure 5.8: Graph of the relation between the true positive rate and false positive rate of predictions given classification threshold values by the three models on the validation set

5.2 Discussion

When comparing the long model to the all topic short model to determine which performs the best, the ratio of how much a true positive is worth compared to a false positive becomes a critical question. The all topic model notably has about 3 percent lower ironic comments in its dataset, which may have a large affect with such small datasets. Though, it outperforms at higher values, but at these values it only predicts 3 or less out of the 18 ironic comments. These comments as seen in the figure below are also some of the shorter, easier and more linguistically ironic comments among the 18 ironic comments in the validation dataset.

TV2 All Topic Short Model	
Comment Text	Prediction
Du skriver som du har vett til.	0.965
Ja, helt sikkert 😊	0.813
J-vla effektive vaksiner de har gitt oss!! 😡	0.603

Table 5.3: Caption

A recommendation tool that only accurately recommends ironic comments 3/18 of the time, but only wrongly recommends non-ironic comments 3/265 of the time, is less useful than one that wrongly predicts 35/265 of the time but also accurately predicts 17/18 of the time ironic comments. As deduced from the labelling

process, it is far often clearer that a comment is non-ironic than vice versa. If a person has to analyze almost all recommendations to see if they contain irony as they know the model will wrongly recommend 15/18 of the ironic comments, it will take more focus and time then knowing almost all ironic-comments will be accurately recommended and only wrongly predict non-ironic comments needs to quickly analyzed to determine they are non-ironic.

With this in mind, the artefacts intended environment and purpose of saving a manual moderator time and mental processing capacity, it could be argued that the long model significantly outperforms the all topic model. Due to the long model's dataset being smaller than the all topic model's dataset, its superior classification ability could be argued to be an indication that context encoding, which also encompasses the linguistic patterns, is a greater method of creating datasets for the training of irony detection models.

Revisiting the research questions from 1.3, while the results taken at face value show that context encoding and topic diversity increase accuracy to a significant degree, the overfitting of the model they are compared to make this conclusion less reliable. Especially in the measurement of degree of better or worse performance. In order to have a more solid answer to these research questions, larger datasets and more trained models are required to compare, which was not possible under the circumstances of this thesis.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The datasets used for the models compared to contemporary state of the art model datasets, are too small and the range of topics included compared to the total amount of topics *TV2 Nyheter* regularly posts articles about, are too narrow. And there are several threats to the results' validity. However, I think the results are sufficient enough to suggest that a context encoded fine-tuned long model performs better than a non-context encoded short fine-tuned model in at least the aspect of irony text classification. And that there is not much that suggests that this behaviour will not be present at larger scales. However, to what degree a context-encoded model outperforms a non-context-encoded model is uncertain, though longformer models in the original longformer paper only had a slightly better performance on for instance the wikipop dataset than RoBERTa 512 token limit model, as seen in 3.4 [2]. And whether the exponentially longer training times required to train them are worth the higher performance is also trade-off to consider.

6.2 Future Work

There are lot of potential improvements and extensions to this technique, and combinations of the cutting edge NLP machine learning technology that remains to be explored.

6.2.1 Larger Datasets

Whether context-encoding works and to what degree it scales with larger datasets is a useful question to answer for large scale company automatic moderation. For instance training long models such as the longformer-base-4096, longformer-large-4096, or a long version of the RoBERTa model, such as xlm-roberta-longformer-base-4096 on a more readily available and larger English dataset with context-encoded inputs, would likely shed some light on the scalability of this dataset encoding technique [2] [?].

6.2.2 Individual Detection

Another method of increasing classification accuracy is either training separate models for separate comment authors, or including some identification in the dataset inputs to encode the author of each input. This would allow a trained model to understand individual's styles of irony. While this method could potentially increase accuracy by a worth-while amount, it is a strict violation of the previously stated GDPR privacy values, and is therefore ethically questionably in its worth of implementation [?].

6.2.3 Multi-Label

Multi-label text classification is a method of classification that allows you to assign an input multiple labels. This could be used to have more graded and nuanced classification of irony with multiple irony sub-categories. Similar to what is done in *Detecting and Grading Hateful Messages in the Norwegian Language* [?].

6.2.4 Multi-Modal

Multiple potentially useful types of data included in a Facebook posts were not used, such as reactions, which can be used to gauge the viewer's general sentiment towards a post or comment, but also images, GIFs and videos. As mentioned in 3.3, using the multi-modal technology, images and by extension GIFs and videos, reaction numbers or other numerical data, can be combined with the text to make more data rich inputs [?]. And there are papers that claim that multi-modal models are significantly better [7].

6.2.5 Dialect

A way to improve the disruptions caused by the language issues described in 3.2, is to pre-train a model on the domain you are attempting to detect language phenomena in, instead of fine-tuning on it. Though, Facebook's terms of service and robot.txt disallow automatic gathering of data which is needed to realistically make a large enough corpus to pre-train a language model [19]. However, other website with comment sections that contain dialect, slang, mixed languages and malformed grammar, which allow large scale data scraping, could pre-train a model specific to their site's comment section.

6.2.6 Other Sentiments and Contexts

The dataset encoding technique of separating chaining elements of text that relate to one another, such as article, comment and reply to comment. Can be applied to other contexts, such as messages that respond to each other or sentences in a book. In the case of automated moderation, it remains to be seen, but would likely have a similar performance to irony if applied to other complex

sentiments, for instance, insults, threats and discrimination. With technology capable of training a model with longer input limits than 1024, data such as the contents of links in comments or entire sequences of an article, its comments and their replies could be encoded as one input, which may lead to higher accuracy.

Bibliography

- [1] adamwawrzynski (2020), longformer github. 4.1
- [2] Beltagy, I., M. E. Peters, and A. Cohan (2020), Longformer: The long-document transformer, *arXiv:2004.05150*. (document), 2.2.3, 2.10, 4.1, 6.1, 6.2.1
- [3] Colebrook, C. (2004), *Irony*, Routledge. 2.1.1
- [4] Curiously (2021), Multi-label text classification with bert and pytorch lightning. 4.3
- [5] Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019), Bert: Pre-training of deep bidirectional transformers for language understanding. (document), 2.2, 2.5, 2.6, 2.2.1, 2.2.3
- [6] Hochreiter, S., and J. Schmidhuber (1997), Long short-term memory, *Neural computation*, 9, 1735–80, doi:10.1162/neco.1997.9.8.1735. 2.2
- [7] Huang, Y., C. Du, Z. Xue, X. Chen, H. Zhao, and L. Huang (2021), What makes multi-modal learning better than single (provably). 6.2.4
- [8] ibeltagy (2020), longformer github. 3.5, 4.1
- [9] Khare, Y., V. Bagal, M. Mathew, A. Devi, U. D. Priyakumar, and C. V. Jawahar (2021), MMBERT: multimodal BERT pretraining for improved medical VQA, *CoRR*, *abs/2104.01394*. 3.3
- [10] Merriam-Webster (2022), Irony. (document), 2.1.1, 2.2
- [11] Merriam-Webster (2022), Sarcasm. (document), 2.1.1, 2.3
- [12] Ministry of Culture and Equality (2020), Act relating to the editorial independence and liability of editor-controlled journalistic media (the media liability act). 1.1
- [13] Ministry of Foreign Affairs (2020), International efforts to promote freedom of expression and independent media). 1.1
- [14] Nilsen, D., and A. Nilsen (2021), Irony, the many types: Irony vs. satire and paradox, linguistic irony vs. situational irony, stable vs. observable irony, dramatic irony, tragic irony, dark irony, and visual irony. 2.1.1

- [15] Patel, D., P. Raval, R. Parikh, and Y. Shastri (2020), Comparative study of machine learning models and bert on squad. (document), 2.2, 2.4
- [16] Potamias, R. A., G. Siolas, and A. G. Stafylopatis (2020), A transformer-based approach to irony and sarcasm detection - neural computing and applications. (document), 1.2, 2.1.1, 2.2.1, 2.8, 3.1
- [17] Rogers, A., O. Kovaleva, and A. Rumshisky (2021), Primer in bertology: What we know about how bert works. 2.2.1
- [18] Schuster, M., and K. Paliwal (1997), Bidirectional recurrent neural networks, *IEEE Transactions on Signal Processing*, 45(11), 2673–2681, doi:10.1109/78.650093. 2.2
- [19] TV2 (2022), Tv 2 nyheter. 1.3, 6.2.5
- [20] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017), Attention is all you need. (document), 2.2.1, 2.7, 2.2.3
- [21] Williamson, K., and G. Johanson (2018), *Research methods: Information, systems, and contexts*, Chandos Publishing. (document), 3, 3.1, 3.1
- [22] Wilson, D., and D. Sperber (2012), *Explaining irony*, pp. 123–145, doi:10.1017/CBO9781139028370.008. (document), 2.1.1, 2.1
- [23] Zaheer, M., G. Guruganesh, A. Dubey, J. Ainslie, C. Alberti, S. Ontañón, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed (2020), Big bird: Transformers for longer sequences, *CoRR*, abs/2007.14062. 4.1

Model and Python Files

As the model files are larger than the allowed upload size for appendix files, they have been compressed and uploaded using Git LFS on a public github repository: <https://github.com/ThorTheStone/Irony-Detection-Master-Thesis-Files>

- **create_long_model.py**, a file that converts a 512 token limit BERT model to 1024 token limit long BERT model.
- **long_text_classification_trainer.py**, a file that fine-tunes a 1024 token limit long BERT model for ForSequenceClassification of irony on a CSV file dataset using sharded distributed data parallel on two GPUs.
- **long_text_predicter.py**, Uses a 1024 token limit long fine-tuned BERT model to make predictions on a CSV validation dataset, and stores the results as a CSV.
- **short_text_classification_trainer.py**, a file that fine-tunes a BERT model for ForSequenceClassification of irony on a CSV file dataset.
- **short_text_predicter.py**, Uses a fine-tuned BERT model to make predictions on a CSV validation dataset, and stores the results as a CSV.
- **calculate_prediction_metrics.py**, a file that contains code for calculating and showing graphs of various metrics of the predictions made by one or multiple models from CSV prediction files.

The files use the Python *transformers* package version 4.18.0 and *torch* version 1.11.0, with small modifications to the *sharded.py* file, as problems with the detection of the Python package *fairscale* occurred.