Master Thesis

How Different is Stereotypical Bias in Different Languages? Analysis of Multilingual Language Models

---

Author

Ibrahim Tolga Öztürk

Supervisors

Dr. Matthias Aßenmacher
Prof. Dr. Christian Heumann
Department of Statistics

External Supervisor

Msc. Rostislav Nedelchev
Alexander Thamm GmbH

Department of Statistics
Ludwig-Maximilians-Universität München

Munich, 28th of September 2022

Abstract

Warning: *This paper contains examples of offensive stereotypes and may be upsetting.*

Recent studies have demonstrated how to assess the stereotypical bias in pre-trained mono-lingual language models for the English language. In this master thesis, we extend this approach to multi-lingual models and multiple languages to show the stereotypical bias containment between different languages using the *StereoSet* data set (Nadeem et al., 2020). Furthermore, this thesis is not only to compare different languages but also to compare different models in the same language to interpret on an individual model's performance and its potential for a biased response. We perform this evaluation by translating the data set to German by offering guidelines to further extend the work to other languages. Although the translations created with Amazon Web Services (AWS) are adequately satisfying, we fix some issues with semi-automatical methodologies.
The stereotypical bias is assessed based on two tasks in this thesis: mask-filling and next sentence prediction. The models that do not have mask-filling support are inferred using their generative structure, and the models that do not have next sentence prediction support are fine-tuned for this task. The results fundamentally show that the created German data set is adequately valid to perform bias estimation, and German and English languages have similar stereotypical biases. All the code and model checkpoints used in this thesis will be publicly available[1].

[1] https://gitlab.lrz.de/statistics/summer22/ma_ozturk/

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
NLP	Natural Language Processing
AWS	Amazon Web Services
ICAT	Idealized Context Association Test
LMS	Language Modeling Score
SS	Stereotype Score
RNN	Recurrent Neural Network
BiLSTM	Bidirectional Long Short-term Memory
mBERT	Multilingual Bidirectional Encoder Representations from Transformers
mGPT	Multilingual Generative Pre-trained Transformer
mT5	Multilingual Text-To-Text Transfer Transformer
ReLU	Rectified Linear Unit
MLM	Masked Language Modeling
NSP	Next Sentence Prediction
SOP	Sentence Order Prediction
BPE	Byte Pair Encoding
Hidden Dims	Hidden Dimensions
Att. Heads	Attention Heads
Vocab Size	Vocabulary Size
LASER	Language-Agnostic SEntence Representations
LaBSE	Language-Agnostic BERT Sentence Embedding
mC4	Multilingual Colossal Cleaned Common Crawl
NLTK	Natural Language Toolkit
CPU	Central Processing Unit
GPU	Graphical Processing Unit
RAM	Random-Access Memory
FP16	Half-precision
lr	Learning Rate

Acknowledgements

We acknowledge the support of the following projects and organizations: OpenGPT-X (BMWK 68GX21007C), Alexander Thamm GmbH, and Ludwig-Maximilians-Universität München.

1. Introduction

With the rise of deep learning in the industry and research, many controversial actions emerged from machines and programs. One of these controversial behaviors of Artificial Intelligence (AI) is the stereotypically biased results that deep learning models tend to produce. In fact, models learn their expected actions from the data set that they are programmed to learn from, and these data sets are primarily from the pages and websites open to the public, so they might include people's stereotypes. Nonetheless, a model can be programmed to ignore these biases as much as possible, which is the ideal anticipated demeanor. On the other hand, some models are over-focusing on these stereotypes and evolving more unfairly. These stereotypical decisions driven by the deep learning models cause the companies or the engineers to be liable for the stereotypical bias. On one side, the data set can not be inspected one by one to ensure it does not possess any stereotypes due to its typical large size; on the other side, the data set can not be considerably downsized since it would limit the performance of the machine learning model. Hence, the likelihood of producing stereotypical outputs must be minimized, and before that, a generic methodology to measure and evaluate the stereotypical bias in the models is essential. To this day, various approaches to dealing with stereotypical bias measurement exist in the literature. A crucial and significantly related approach that measures stereotypical bias in the pre-trained language models is Nadeem et al. (2020). They fundamentally create an English data set and a methodology to measure the stereotypical bias in English language models. However, this methodology is significantly limited since it supports only one language, whereas the current state-of-the-art multi-lingual models support more than 90 languages (Doddapaneni et al., 2021).
In this thesis, we evaluate the stereotypical bias in multi-lingual models by creating a new data set by translating the StereoSet (Nadeem et al., 2020) data set to German. In other words, we input English and German examples to the pre-trained multi-lingual language models to observe whether the model would show a stereotypical bias or not in different languages for investigating different stereotype containment in different languages. The evaluation is conducted in English and German, and the results for these languages are

compared; however, the code published with this thesis allows for the application of the evaluation technique in different languages remarkably effortlessly. The models are evaluated with the data set and technique discussed above (Nadeem et al., 2020) and the data set's translation to German after applying pre-processing techniques. The code they published[1] is extended, and a more generic and efficient version of it is implemented to apply in different languages. This thesis provides more efficient methodologies because of extending the Nadeem et al. (2020) work with batch-computing and more optimized code. In addition, many errors in the paper from Nadeem et al. (2020) and their code are noticed, and proposed corrections are published and discussed.

Furthermore, this thesis has more variety of models to evaluate: three variants of monolingual and three variants of multi-lingual models. The recently developed and enhanced BERT (Devlin et al., 2018), GPT-2 (Radford et al., 2019; Tan et al., 2021), and T5 (Raffel et al., 2019; Xue et al., 2020) are specifically picked because of their diverse architectures: encoder-based transformer, decoder-based transformer, and encoder-decoder-based transformer. Thus, a generic strategy to apply in different languages and various model types is propounded in this thesis. One of the primary purposes of this thesis is to compare the model's stereotypical biases in English and German languages. One can expect their results to be similar since these languages and the native people of these languages have many things in common.

The thesis is outlined by first discussing the term *Stereotypical Bias* and its examples and discussions in daily life in Chapter 2. In addition, Chapter 2 discusses the emergence of state-of-the-art language models and the detailed specialties of these pre-trained language models. This section clarifies how the selected models work and why these models are selected. Then, the significant research related to this thesis is discussed in Section 2.4. Next, Chapter 3 elucidates the materials of this thesis and applied methodologies consisting of the data set, the pre-processing phase, the prediction phase, and the evaluation phase in detail. Chapter 4 discusses the experiments done using the techniques described in Chapter 3 and the final results achieved from the evaluations.

[1] https://github.com/moinnadeem/StereoSet

2. Stereotypical Bias in Language Models

2.1. Pre-trained Language Models

2.1.1. Evolution of Language Models

Language modeling is a learning process of the joint probability of the words, sub-words, or characters occurring in a sequence. Before the deep learning era in language models and Natural Language Processing (NLP), there were statistical models and perspectives on text analysis. Fundamentally, the main goal was to find the joint probability of the words occurring together in the sentence. The formula of this probability calculation is $p(w_1, w_2, ..., w_N) = \prod_{i=1}^{N} p(w_i|w_1, w_2, ..., w_{i-1})$ for a sentence that includes the words $w_1, w_2, ..., w_N$. Then, the formula for only the word w_1 is $w_1 = p(w_1|w_0)$ as a conditional probability. Due to having too many dependencies in the above probability calculations, especially in a long sequence, n-gram models come out by assuming only $n-1$ previous words are affecting the word instead of the whole sequence. It can be thought of as an $n-1$ order Markov-chain, too.

With the beginning of the 2000s, neural language models started to gain popularity and owned more satisfactory performance than statistical approaches. In 2003, Bengio et al. (2003) published the first neural network-based language model. After some period, about 2013, Recurrent Neural Networks (RNNs) and Long Short Term Memory (LSTMs by Hochreiter and Schmidhuber (1997)) networks evolved as the leading models to operate in language modeling (Li, 2022; Louis, 2020). The formula of Recurrent Neural Network (RNN) is simply defined as,

$$h_{(j)} = f(Wx_{(j)} + Vh_{(j-1)} + b), \tag{2.1}$$

where $f(x)$ is typically $tanh(x)$, $h_{(0)} = \{0\}^d$ and W, V and b are the model parameters. As seen from the formula, LSTM and RNNs work as states like Markov models, and

each state depends on the previous states. These models accept inputs word-by-word or token-by-token in the sequences (from left to right), and the model stores the knowledge of the previous tokens for each token. Therefore, for the last token, it theoretically considers all tokens before that, and the effect of the token diminishes as the length between the two tokens grows.

Indeed, this working procedure of RNNs and LSTMs highly restrains the effect of the first words on the last words. It might be especially significant in the German language as the meaning of the separable verbs ("trennbare Verben") varies according to the last word, which is usually the prefix of the verb, in the sentence. For instance, in the sentence "Ich ziehe nächste Woche in ein Altersheim um." ("I am moving to a retirement home next week."), the verb is indeed "umziehen" ("to move") due to the second and the last word, so the model should connect the second and the last word sufficiently since the meaning entirely alters by modifying the last word. Furthermore, these models enormously suffer from this problem in the long sequences. First, there would be enormously many words affecting the last words, so the effect of a word at the beginning would be nearly zero. Secondly, the model might not even function at all due to vanishing gradients. The loss function of RNN operates as "backpropagation through time," which implies that the loss is calculated backward by computing the gradient with respect to the h variable, so the gradient is estimated backward for each token. Therefore, the gradient might converge to zero, and this would result in reaching zero in the final result, which is called vanishing gradients, or a long time-taking calculation process. Thus, in general, RNNs and LSTMs are known to not function pleasingly in long sequences. In addition, reading the input sequence word-by-word instead of the whole sequence at once prohibits these models from performing in parallel since the process of one state can be accomplished only after terminating the operations of the previous states. This special input processing restricts the training application in parallel; hence, it forbids shortening the training time given the increasing computation and fails to employ efficient computing.

2.1.2. Transformers

Due to the inefficiencies and limitations of RNNs, attention-based models are commonly used. Attention in this context means giving attention to the particular words that should significantly affect the predicted word. Consequently, it allows the model to assign different weights to each token in the sequence with respect to their significance over the word to be predicted. It enables the models to connect words that strongly influence each other and gain a better understanding of the sequence. As the name suggests, the

attention mechanism is inspired by the "cognitive attention" in neuroscience (Hassanin et al., 2022).

In recurrent neural networks, the output of states (h) is a sequence of state values instead of a function of each state. As mentioned before, there is a weight (α) for each state in the attention mechanism, and states are typically represented with the $\mathbf{V}$ vector, so the output becomes $\alpha_1 * \mathbf{v}_1 + \alpha_2 * \mathbf{v}_2 + ... + \alpha_j * \mathbf{v}_j$, where α is the weight for each state and it is measured as the softmax of the scoring function (f) of the query ($\mathbf{Q}$) and key ($\mathbf{K}$) vectors as written in the formula,

$$\alpha_j = \frac{\exp(e_j)}{\sum_{j'=1}^{J} \exp(e_{j'})}, \tag{2.2}$$

where $e_j = f(\mathbf{q}_{j''}, \mathbf{k}_j; \theta_f)$ and f is scoring function which maps the query-key vector to a scalar ("score"), $\mathbf{Q}$ is query vector, $\mathbf{K}$ is key vector and θ is an optional parameter of the scoring function. The query, key, and states typically include the model parameters ($\mathbf{W}$). The softmax operation is formulated in order to transfer the scores to probability space where their sum must be one. An example scoring function, scaled dot-product attention, formulated by Vaswani et al. (2017) is written in the Formula 2.3. The crucial point in this formula is that it does not depend on the parameter θ, which causes computational efficiency.

$$e_{j',j} = f(\mathbf{q}_{j'}, \mathbf{k}_j) = \frac{q_{j'}^T k_j}{\sqrt{d_k}}, \tag{2.3}$$

where d_k is the number of dimensions in the key vector, it is used here for the purpose of normalization. It is noteworthy to mention that estimation with this scoring function requires the number of dimensions of query and key vectors to be equal ($d_q = d_k$).

Indeed, the formula and description mentioned above are the "basic recipe" of the attention mechanism, and the real formula might differ according to attention types, such as self-attention, cross-attention, and multi-head attention. In cross-attention, it is deemed that there are two different vectors; one for the source ($\mathbf{X}$) and one for the target sequence ($\mathbf{Y}$). The target vector forms the query vector as $\mathbf{Q} = \mathbf{Y}\mathbf{W}^{(q)}$ and the source vector transforms the key and state vectors as $\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$ and $\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$. In another typical attention mechanism type, self-attention, there are not two, but only one sequence, $\mathbf{X}$. Self-attention, in fact, has the identical formula, but instead of employing two separate vectors as $\mathbf{X}$ and $\mathbf{Y}$, self-attention utilizes only one vector. The formulas of it are represented by assuming the target sequence to be equal to the source sequence as $\mathbf{X} = \mathbf{Y}$. Moreover, another attention mechanism called "multi-head attention" qualifies

the attention mechanism to be run several times in parallel. It is used in numerous crucial and well-known transformers (e.g., BERT) due to its advantage in parallel computing, shortening the training process time.

Attention mechanisms have a broad application area in terms of model types, and one of the most significant application areas of attention mechanisms is *transformer* models. *Transformer* models are complex deep learning models that adopt the aforementioned attention mechanisms (Vaswani et al., 2017), enabling them to focus on particular tokens for each prediction and make use of parallel computing by providing the input sequence as a whole rather than word-by-word. Transformers are popularly used in different industries, and the usage areas can be extremely divergent. They make the transfer learning (Pratt et al., 1991) concept remarkably advantageous and convenient in the NLP field. Transfer learning is simply applying knowledge gained from a problem to a different problem where the problems are similar. For instance, in work from Pratt et al. (1991), they transfer the knowledge through the weights of one small neural network to the other large neural network. They achieve substantial speed-up in the large neural network and promising results in the speech recognition problem. On the other hand, the pre-trained model approach we use in this thesis is slightly different. Researchers are continuously training giant and efficient models with enormously large data sets and opening these models to public use, and the others transfer the knowledge gained by these training processes to different tasks of their interest or learn knowledge from different data sets. The most common pre-trained language models are based on transformer architectures. A typical depiction of the transformer model architecture is given in Figure 2.1. As seen from the figure, there is no more recurrent feature; it is indeed a simple feed-forward network, and Transformers are originally designed as encoder-decoder-based architecture. In other words, from one side, the model gets the input sequence and encodes it, and from the other side, it gets the previously predicted output sequence and decodes it; then, by combining these, it predicts the new output sequence. The left side of the diagram is the encoder part, and the right side is the decoder part. Nevertheless, in many state-of-the-art transformer models, only the encoder part (e.g., BERT) or only the decoder part (e.g., GPT-2) is employed, and they have also been named transformers. Furthermore, since the input sequence is given as a whole in the attention mechanism, the position of a word is missed out and loses its importance. On the contrary, the position of a word is, indeed, significant and can reveal a variety of specialties about it. For instance, the subjects are mainly used as the first word in a sentence in the English language, and this knowledge would be worthwhile for detecting subjects. To

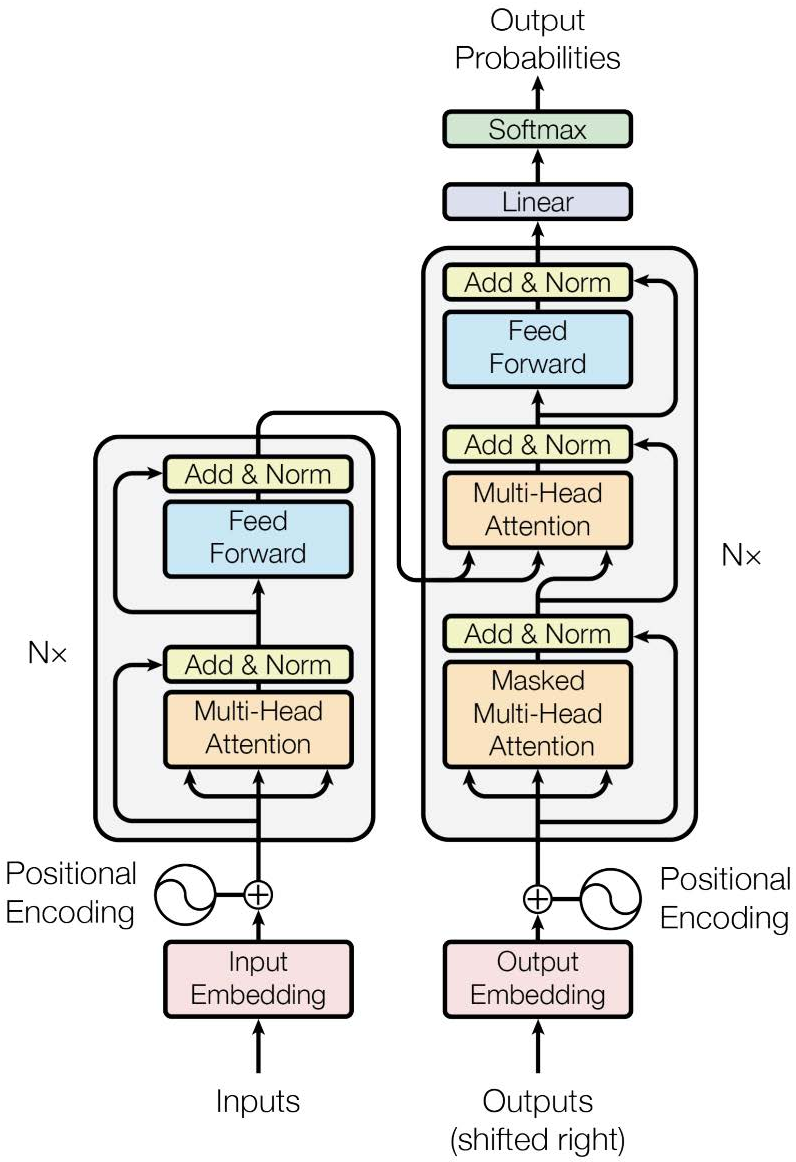

Figure 2.1.: Transformer model architecture (Vaswani et al., 2017)

overcome this problem and benefit the words' positions, transformers have a "positional encoding" representation; it simply encodes the word's position in the sequence added to the embedding of the tokens. By employing this technique, the position of the input is also considered as an input, similar to its embedding vector. After completing the embedding vectors by adding the positional encodings, the multi-head attention layers process the embeddings, where the tokens use each other for the first time. However, one output of the attention layer might be extraordinarily different, affecting or dominating

the whole learning procedure completely. This might cause serious performance declines, especially at the beginning of the learning process. Therefore, the input embeddings are added to regularize any harsh effect from the attention layer and aid the model to avoid over-fitting on some outlier input sequences. This process can be seen in the figure at the "Add & Norm" layer. "Add" fundamentally stands for adding the embeddings to the output of the attention layer as $\mathcal{F}'(\boldsymbol{X};\theta) = \mathcal{F}(\boldsymbol{X};\theta) + \boldsymbol{X}$ (He et al., 2015) where $\mathcal{F}$ is the layer output and $\mathbf{X}$ is the input embeddings. In addition, the "Norm" stands for layer normalization (Ba et al., 2016) which avoids exploding activations on a forward pass (Lubana et al., 2021). The Formula 2.4 is for normalizing a layer where its mean is measured by $\mu = \frac{1}{d}\sum_{i=1}^{d} h_i$ and the standard deviation $\sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d} h_i}$ is computed over the dimensions of $\boldsymbol{h}$. Next, a simple feed-forward layer is applied; it basically consists of two linear transformations with a Rectified Linear Unit (ReLU) activation (Fukushima, 1975) in between. The role of this layer is to transform attention vectors to be accepted by the next layers in the encoder-decoder model. This layer can tackle the input sequence in parallel as opposed to the RNN-based models (Ankit, 2020).

$$\gamma \odot \frac{\boldsymbol{h} - \mu}{\sigma} + \beta \tag{2.4}$$

2.1.3. BERT

As we stated in the previous section, transformer models are mainly divided into three categories. One category is only encoder-based models such as BERT, short for Bidirectional Encoder Representations from Transformers (Devlin et al., 2018). BERT is one of the first and most common transformer models, which came out in 2018. As its name states, BERT is a deep bi-directional model meaning the model does not only learn from left to right but also from right to left. The BERT model is trained on 3.3 billion English words from Wikipedia and Google's BooksCorpus (Zhu et al., 2015), where the data set takes up to 13 GB of memory. BERT does not require labeling in the data set due to its intelligent training objectives. The tokenizer of BERT uses the WordPiece (Wu et al., 2016) technique to create its vocabulary. In the WordPiece technique, the model initially adds all the characters in the data set to the "word unit inventory". Then, it generates a new word-unit by combining two word-units (e.g., subwords) from the inventory, which would increase the likelihood on the training data the most when added to the model and then adds this new word unit to the inventory. This process is accomplished with many iterations until the initially specified vocabulary size is reached, which is 28996 for

BERT models. BERT has the same vocabulary size in all its versions, but its versions vary according to the model size as the base and large BERT models, which we compare in Table 2.1.

	Layers	Hidden Dims	Att. Heads	Parameters
base	12	768	12	110M
large	24	1024	16	336M

Table 2.1.: Comparison of BERT models

Furthermore, one of the most significant differences between different transformers is their training objectives. BERT is trained with two different objectives, namely Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In Masked Language Modelling, the input and target sequences are the same in BERT; basically, it accepts the same sequence but with modifications due to its nature of bi-directionality. Therefore, BERT is not allowed to cheat since some tokens are removed and "masked". The model randomly samples 15% of all tokens and replaces 80% of them by "[MASK]", 10% by a random token, and leaves 10% unchanged, so around 12% of the data set is masked and left to be predicted. An example modified sentence to be used in BERT is "The chess player was [MASK]". The purpose of the second objective of BERT, Next Sentence Prediction (NSP), is to predict whether the second sentence follows the first sentence. The training for this objective also works as self-supervised learning since it does not require labels; half of the data set has the actual next sentence, and the other half has a random sentence as the next sentence; as a result, it has 50% positive and 50% negative examples. "[SEP]" token is used between these sentences to indicate the boundary between sentences. The original final model achieved 97%-98% of accuracy on the NSP objective.

BERT utilizes cross-entropy loss to predict the original token for these masked tokens, trained by employing the Adam (Kingma and Ba, 2014) optimizer. Adam's main difference from the classical stochastic gradient descent algorithms is that it maintains a learning rate for each network weight rather than employing a single learning rate for all weights. In addition, instead of storing a constant learning rate value, Adam adapts the learning rates according to the first (mean) and second moments (uncentered variance) of the gradients; therefore, Adam is indeed derived from "Adaptive Moment Estimation". Indeed, the algorithm is a combination of AdaGrad (Duchi et al., 2011), which is efficient in sparse gradients, and RMSProp (Tieleman et al., 2012), which is efficient in online and non-stationary problems (e.g., noisy). Hence, Adam has two beta values

where one derived from the AdaGrad and the other from RMSProp.
A limitation of BERT is the number of tokens that can be given as input. BERT can consume sequences of up to 512 tokens meaning the sum of tokens in the two sentences of NSP should be less than 512 (Devlin et al., 2018). To overcome the limitations of BERT and to even improve it, many models are proposed with a similar architecture under the research of BERTology (Rogers et al., 2020a). Some of the significant post-BERT models are RoBERTa (Liu et al., 2019), ALBERT (Rogers et al., 2020b) and DistilBERT (Sanh et al., 2019). In RoBERTa, short for the Robust BERT Approach, the training corpus is excessively increased (around ten times) by using the identical model architecture. RoBERTa uses Dynamic masking in the MLM objective to mask the input while pretraining instead of masking and preparing the data set before pretraining, and the model renounced the NSP objective entirely. Moreover, another strong model ALBERT uses a different objective than BERT's NSP as Sentence Order Prediction (SOP). Fundamentally, the positive examples are similar to NSP, but in the negative examples, the ordering of sentences is swapped and left the model to predict the correct order. Due to its cross-layer parameter sharing, ALBERT has fewer parameters. In the comparable amount of model size, ALBERT strongly outperforms BERT. Another important post-BERT model is DistilBERT which is the compressed version of BERT. DistilBERT retains 95% of the performance by halving the model size of BERT. The model's objective is similar to RoBERTa, where it waives NSP and employs a dynamic masking model but utilizes the same corpus as BERT.

2.1.4. GPT-2

GPT-2, short for Generative Pre-trained Transformer 2, is an only-decoder-based transformer model developed by Open-AI (Radford et al., 2019). GPT-2 was released in 2019 as a successor to GPT released in 2018. GPT-2 is basically the direct scale-up model of GPT and is trained with around ten times larger data sets. GPT-2 is trained with a vast English data set of about 40 GB from Reddit, a social media platform, and the pages of the web links in the Reddit posts. Wikipedia pages are removed from these web links because of their repetitions in many other data sets. Hence, they created a unique data set for the model by scraping different web pages and pre-processing; however, the data set is still not publicly shared. Since the model does not depend on any labeling task, it functions in a self-supervised manner.
GPT-2 is trained uni-directional manner as it is a generative model; therefore, it is fully auto-regressive. The objective of GPT-2 is Causal Language Modelling (CLM),

simply meaning predicting the next word given the left context. GPT-2 has multiple types, small, medium, large and extra large, where the small type has the same size as the previous GPT as a baseline, and the medium type has a similar size to the largest version of BERT. The extra large type, which the authors simply call GPT-2, has 1.5 billion parameters which are enormously higher than the BERT base model. The authors stated that the model is not trained on any downstream task specifically, but it performs competently in many due to its diverse and large training data set, and they called this behavior "zero-shot task transfer".
GPT-2 Tokenizer utilises Byte Pair Encoding (BPE) (Gage, 1994; Sennrich et al., 2015) to create its vocabulary. BPE fundamentally transforms the most occurring byte pairs with a byte recursively. For instance, if the vocabulary is created from the text "aaaabbdaabba", the most occurring byte pair is "aa", so it will be replaced with a new byte that is not used in the data, "T". Then, the text becomes "TTbbdTbba, where the most occurring byte pair is "bb", then the text becomes "TTOdTOa" when it is replaced with "O". These steps are recursively processed until there is no byte pair that occurs more than one or a desired vocabulary size is achieved. Because of using an extreme amount of data in GPT-2, the vocabulary size of GPT-2 models, which is 50257, is almost double the BERT vocabulary size by the BPE algorithm. The end of sentence token of GPT-2 is "<|endoftext|>" and will be utilized in further computations.

2.1.5. T5

T5 is the abbreviation of "Text-to-Text Transfer Transformer", and the model is trained by only text input and text output as the name suggests (Raffel et al., 2019). Google introduced T5 in 2020 as an encoder-decoder model, which is an equivalent architecture to the original transformer. Since it receives and outputs text sequences, the model is known as a sequence-to-sequence model.
The training of T5 is accomplished by using many tasks at once, known as "multi-task learning"; hence, the model can operate on various downstream tasks. A particular input format is used to feed the task's name to the model; it is added before the input text and tokenized togetherly with the input text sequence. This multi-task learning technique makes the fine-tuning of the model with different tasks simpler by only appending the task name in front of the inputs, known as "prefix task". The downstream pre-trained tasks of T5 include question answering, summarizing, sentence similarity, sentence completions, and so on.
The model is trained on two types of data sets, one is unlabelled data sets for unsuper-

vised learning for its primary language model, and the second is labeled data sets for supervised learning of its downstream tasks. The primary data set used in T5 is "Colossal Clean Crawled Corpus (C4)", specifically for its unsupervised learning, introduced by the authors and made publicly accessible. The data set is mainly a part of Common Crawl[1] web scrapes but a well-cleaned version, and only this cleaned part form 750 GB which is exceptionally huge compared to the models mentioned above. Furthermore, there are still more than ten other different data sets utilized for the model's supervised downstream task learning.

There are three objectives concerned in T5 training; the first one is language modeling, which is predicting the next word as a comparable objective is done in GPT-2, the second one is masking some tokens as a similar version of BERT's Masked Language Modelling, and the third objective of T5 is deshuffling the input sequence randomly for the model to predict the correct word order. The last objective varies from ALBERT's Sentence Order Prediction (SOP) since T5 predicts the order of the tokens in the intra-sentence context and ALBERT tries to predict the order in the inter-sentence context. Moreover, the second objective also slightly varies from BERT's Masked Language Modelling objective with two different modifications mainly due to T5's encoder-decoder structure. The first modification is masking consecutive tokens as one unique mask instead of masking them separately, so the target sequence becomes a concatenation of the corrupted tokens. The second modification to the BERT's MLM objective is dropping the corrupted tokens altogether and tasking the model to reconstruct dropped tokens.

Furthermore, a practical methodology utilized during the training of T5 is the teacher-forcing method. Due to the model's architecture, it needs target sequences for the decoder part, and a teacher-forcing method is an efficient training method if the model has both source and target inputs. In teacher-forcing methods, instead of allowing the wrong prediction output to be used as an input in the next prediction, the model supplies the correct prediction output, which it gets from the decoder, as an input to the next prediction. This dynamic and internal guidance in the model aids in having a fast and stable convergence (Raffel et al., 2019). As T5 is trained in a generative objective as well, it can also be utilized to generate new words. Inference to the model to generate new words can be achieved by utilizing the "generate" function of Huggingface models. The authors of T5 provided the model with different versions, simply changing the model's size. The base model has a similar size to the BERT base model and the large model with the BERT large model for comparability reasons, as seen in Table 2.2.

[1] https://commoncrawl.org/

	Layers	Hidden Dims	Att. Heads	Parameters
small	6	512	8	60M
base	12	768	12	220M
large	24	1024	16	770M
3B	24	1024	32	3B
11B	24	1024	128	11B

Table 2.2.: Comparison of T5 models

Moreover, the T5 tokenizer also employs a WordPiece approach like BERT with 32000 pre-defined vocabulary size to tokenize the input sequences. A crucial difference between the T5 tokenizer and BERT tokenizer is that the T5 tokenizer's masked token ids are differently encoded. For instance, for the first mask, the text is "<extra_id_0>" and the second is "<extra_id_1>" and this goes on until "<extra_id_99>", so there are 100 different possible masks to be used in a sequence. In addition, the end of the sentence token of the T5 tokenizer is "</s>".

2.1.6. Multilingual Models

The models described above mainly work in English or in only one language (e.g., mono-lingual models), and these models constrain cross-lingual analysis. There are many motivations behind having cross-lingual analysis. One motivation for the cross-lingual analysis is that the expected input might be from different languages in many down-stream tasks. For instance, an analysis of comments in a forum should not be restricted to English since it is open to everyone from different language-speaking countries. Another significant motivation for the cross-lingual analysis is that some languages have few resources and can not have a well-performed mono-lingual language model (Lample and Conneau, 2019).

One of the most crucial parts of the multi-lingual models is the sentence embeddings that they use (Artetxe and Schwenk, 2019). Sentence embeddings are the vector embeddings of the sentences from different languages, and the ones used in multi-lingual models are primarily known as "language agnostic sentence embeddings". The primary specialty of these sentence embeddings is having a similar vector representation for the sentences in different languages with the same meaning, as seen in Figure 2.2. A commonly used embedding is Language-Agnostic SEntence Representations (LASER) (Artetxe and Schwenk, 2019) which covers 93 different languages in 1024 dimensional vector space. LASER uses a Bidirectional Long Short-term Memory (BiLSTM) encoder with a shared

BPE which is coupled with an auxiliary decoder. The encoder is language agnostic (Zhao et al., 2020) meaning that it is shared by all languages, and there is no direct input to the encoder about which language the sentence belongs to since the vocabulary is built by the concatenation of all training corpora. Nonetheless, the input of the decoder is supplied with the language ID that specifies the language to which the sentence belongs. Contrary to the earlier approaches, LASER is trained on only two target languages, English and Spanish, as they are observed to have a sufficiently satisfactory alignment. The main training corpora include 93 input languages and 223 million parallel sentences.

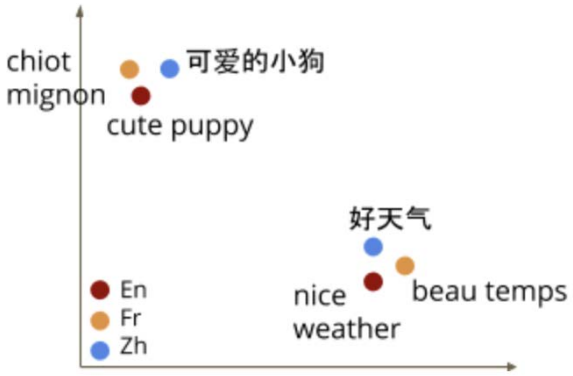

Figure 2.2.: Multi-lingual embedding space. (Yang and Feng, 2020)

Another crucial and state-of-the-art multi-lingual sentence representing approach is Language-Agnostic BERT Sentence Embedding, known as LaBSE (Feng et al., 2020). The model uses the pre-trained English BERT model but with two bidirectional encoders: one is for the text from the source language, and the other is a translation of the text in the target language. In addition to monolingual BERT, Language-Agnostic BERT Sentence Embedding (LaBSE) has a significant objective, Translation Language Modeling (TLM) (Lample and Conneau, 2019; Moberg, 2020). As seen in Figure 2.3, the objective helps the model learn similar words in different languages by providing the source text in Masked Language Modelling format and the target text by masking the words which are not masked in the source language. Then, after the encoders and simple transformations, the model compares the two outputs in its loss function to maximize the similarity of translation pairs in the shared embedding space. The model is trained with mono-lingual and multi-lingual corpus, consisting of 17 billion sentences and 6 billion translation pairs, respectively.
In this thesis, the multi-lingual versions of the chosen mono-lingual models are utilized to be able to compare the results. For BERT, Multilingual Bidirectional Encoder Represen-

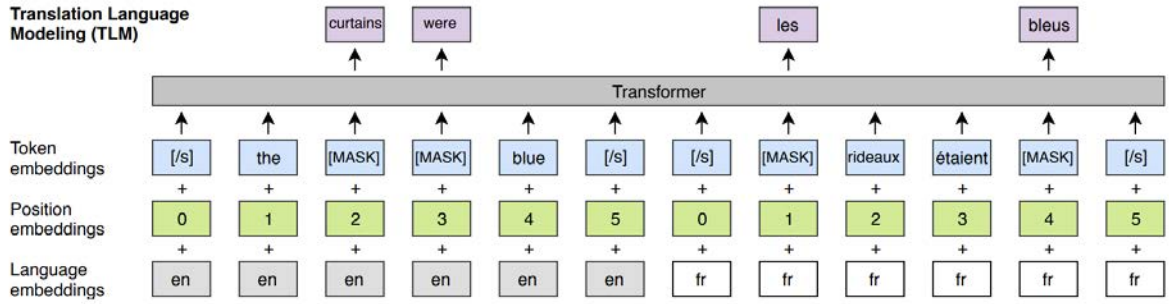

Figure 2.3.: Translation Language Modelling (TLM) objective in English and French languages (Lample and Conneau, 2019)

tations from Transformers (mBERT) (Devlin et al., 2018; Pires et al., 2019) is employed as an encoder model. Multi-lingual BERT has multiple official model types, which vary in size, but we used the base model as it has similar size to the models above. A similar way applies to the Multilingual Text-To-Text Transfer Transformer (mT5) model (Xue et al., 2020). There are official supports for multi-lingual T5 with different model sizes, and the base model is used in this thesis for comparability concerns. Since the authors of GPT-2 did not provide an official multi-lingual model, the model of Tan et al. (2021) is utilized in this work to assess the stereotypical bias in multi-lingual GPT-2 models. The model is chosen due to its popularity among the multi-lingual GPT-2 models, professional documentation, and satisfactory results. Another model from the HuggingFace platform which can be compared is a multi-lingual GPT-2 model from the user "miguelvictor". However, the model does not have any theoretical paper support and even a model card on the platform. Moreover, it is an enormously giant model, which would create difficulties in comparing with other chosen models.

2.1.7. Model Comparisons

The models used in this thesis are fundamentally BERT, GPT-2, and T5 based. Since their language modeling scores are evaluated and compared, it is significant to provide a general comparison for the models to better interpret their final results. Table 2.3 shows that BERT is the most miniature model trained with the smallest data set. BERT is trained with Wikipedia pages and a book corpus. Since these sources have more official requirements and audits for authors than ordinary web pages, the pre-training data set of BERT is expected to include fewer stereotypes. The exact data size of multilingual BERT is not published, but it is basically the Wikipedia web pages for the 104 most common languages. If it is assumed that there is more source in English than any other

	Data Size	Data Source	Layers	Hidden Dims	Att. Heads	Parameters	Vocab Size
bert-base-cased	13GB	Wikipedia & BooksCorpus	12	768	12	110M	28996
t5-base	750GB	C4 (Webpages)	12	768	12	220M	32000
gpt-2	40GB	Reddit and Webpages	24	1024	16	1.5B	50257
bert-base-multilingual-cased	104 languages	Wikipedia	12	768	12	110M	110000
google/mt5-base	26TB (101 languages)	mC4 (Webpages)	12	768	12	220M	250000
THUMT/ mGPT	26TB (101 languages)	mC4 (Webpages)	24	1024	16	560M	250000

Table 2.3.: General Comparison of the Models Used

language in Wikipedia, the maximum data size of multilingual BERT can be 1.3 TB, which is extremely small compared to other multilingual models. The vocabulary size of multilingual BERT obviously increased because of containing many tokens from different languages.

T5 is pre-trained with an enormously large data set compared to other SoTA models; nevertheless, the model size of its base model is precisely the same as BERT's base model. The parameter size of T5 is double the comparable BERT model, although it has an identical model size to BERT, and this is due to T5's encoder-decoder architecture. As stated in Section 2.1.5, the vocabulary size of T5 is prefixed to a similar number with BERT due to achieving comparability. Multilingual T5 is pre-trained with a similar data set from 101 different languages; consequently, the data set became even much more extensive. The vocabulary size of multilingual T5 is expected to be bigger than BERT since the data size is much bigger, and the tokenizer might encounter different tokens in the data set.

GPT-2 is trained with a larger dataset than BERT, but it is a small dataset compared to T5's. The data source of GPT-2 is similar to T5; they both use web scrawling but from different websites. GPT-2 model has almost double the size of BERT and T5. GPT-2 has

a broader vocabulary compared to the others, which also poses a considerable amount (more than ten times BERT) of parameter space for the model. Thus, GPT-2 model is enormously larger than BERT and T5 models. The Multilingual Generative Pre-trained Transformer (mGPT) 2 model has the same size as the mono-lingual GPT-2 since it is directly based on that model. Thus, it is a larger model compared to multilingual BERT and multilingual T5. It should be noted that the training data of the model is different from GPT-2, but it is precisely the same as multilingual T5. This would change the bias consistency of the model, so it is crucial to be considered while interpreting the final results. Furthermore, the parameter size is also not the same with GPT-2 due to using Multi-Stage Prompting (Tan et al., 2021) approach. The vocabulary size of the model is the same as the multilingual T5 tokenizer because it uses the same dataset and tokenizer as the multilingual T5 model.

2.2. Definition of Stereotypical Bias

Stereotype and *Bias* have different meanings, although commonly used in the same context. The stereotype is a generalized belief that attributes certain characteristics to all members of a particular category or class of people (Cardwell and Marcouse, 1996). On the other hand, bias is supporting or opposing an idea, a particular person, or a thing in a way that is prejudicial or unfair due to allowing personal views to influence your judgment (McIntosh, 2013). Therefore, bias is a broader term and can happen without a stereotype. For instance, "overconfidence bias" is a type of bias when a person overestimates their skills and believes they can perform an action that is indeed infeasible without sufficiently thinking (Moore and Healy, 2008). Furthermore, bias can also exist in a positive direction; an intentionally anti-stereotypical action is also a biased behavior. The combination of two terms, "stereotypical bias", is a type of bias that occurs when the reason for the bias is a stereotype. The difference between a stereotype and a stereotypical bias is that a stereotype is only a belief previously defined, and a stereotypical bias is an action, behavior, or idea sourced from a stereotype. For instance, if a hiring manager thinks that all Asians are hardworking, then it is a stereotype; however, if the manager hires an Asian when there is a better-qualified European only because of the stereotype mentioned earlier, it becomes a stereotypical bias. Like people, the machines or programs might also have a stable stereotype in themselves, and a stereotypical bias occurs when the program outputs an unfair result that is sourced from a stereotype.

2.3. Real-life Examples

Before the deep learning and artificial intelligence era, the machines were explicitly programmed for each case and to output an expected result. However, deep learning is a black box methodology, and the machines' results can be unexpected. Unintended results sometimes occur in daily life from artificially intelligent programs (Larson et al., 2016) such as in the healthcare systems (Obermeyer et al., 2019) or simply in Google Search results.

Figure 2.4 shows the search results of "terrorist religion" in Google. The first result that it outputs is the Wikipedia page for "Islamic terrorism", although there are also Wikipedia pages for "Religious Terrorism", "Christian terrorism", "Jewish religious terrorism" and so on. The same experiment is done in Yandex Search, and the results look fairer, as seen in Appendix A.1. Consequently, a simple religion-based stereotypical bias can also occur in the world's most common search engine (Graph 2.5), and the engine, indeed, belongs to one of the most technologically advanced companies in the world.

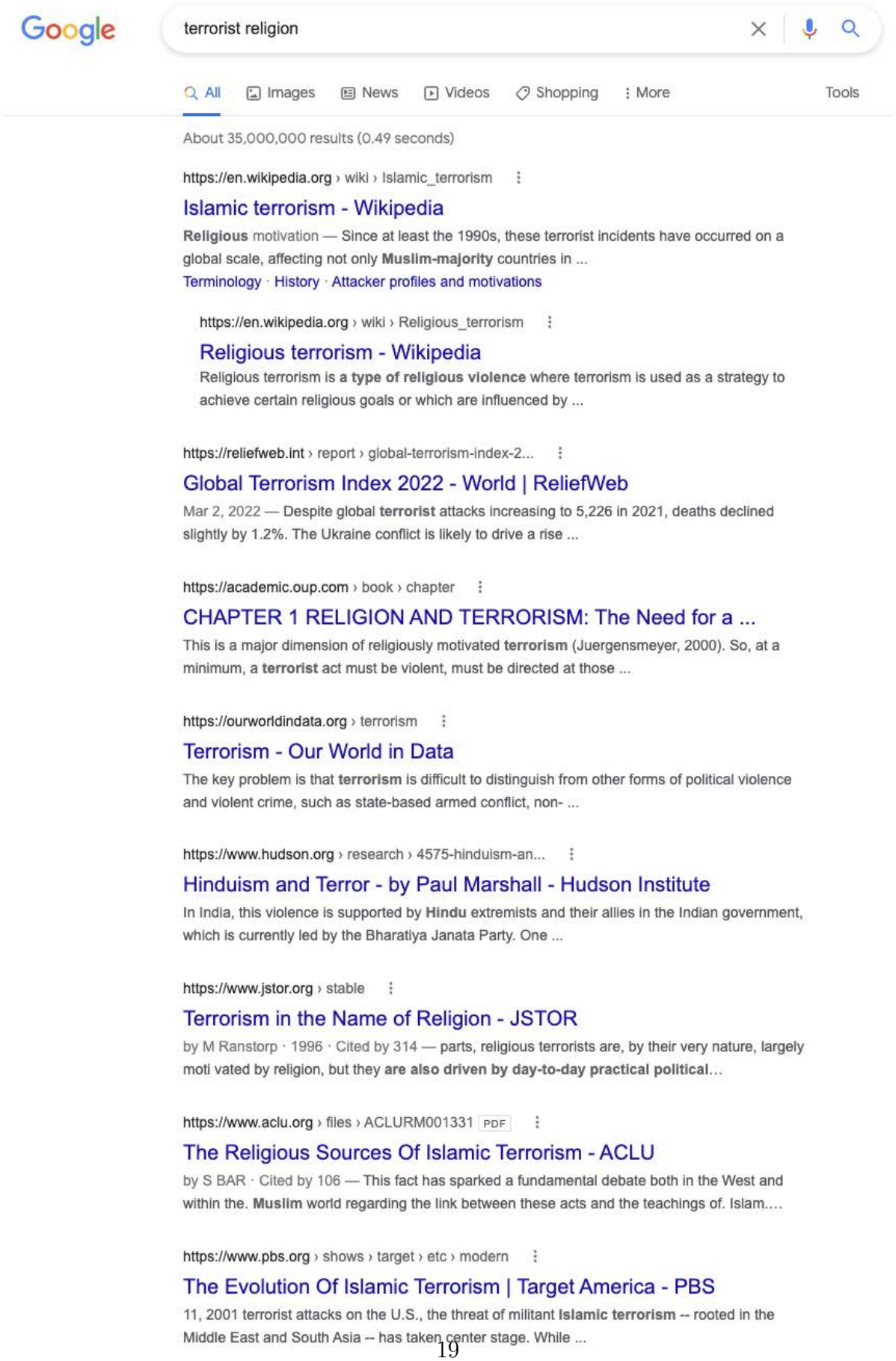

Figure 2.4.: Google Search results for "terrorist religion".

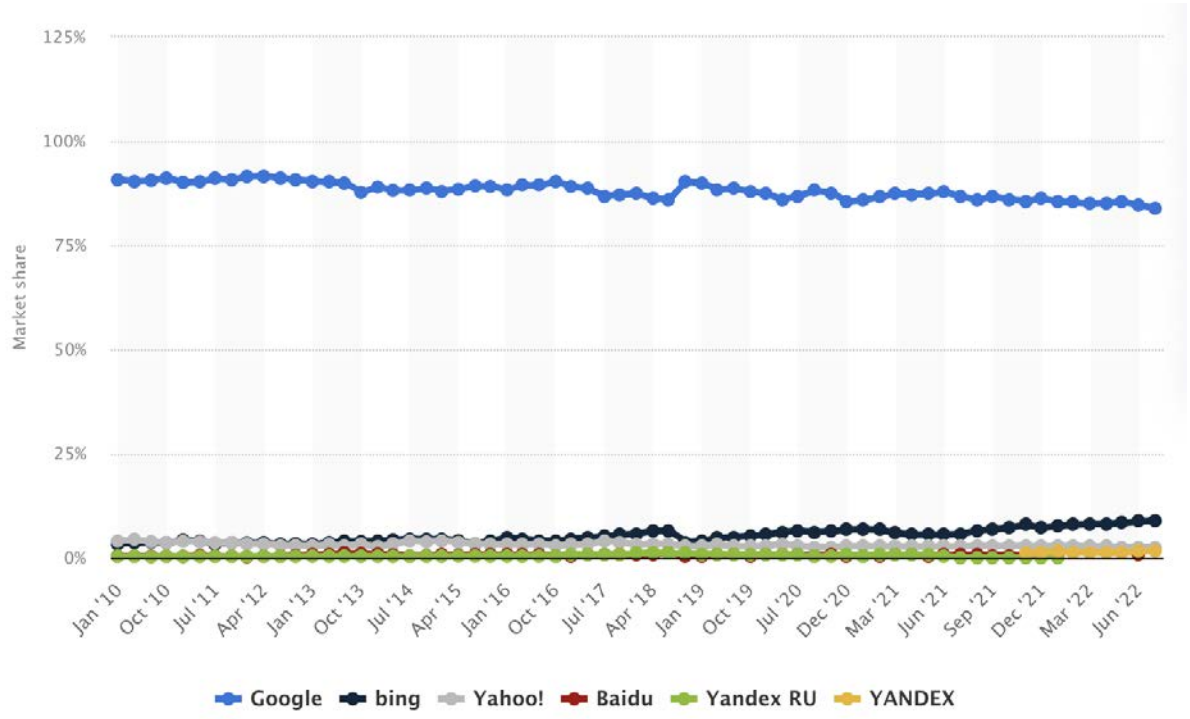

Figure 2.5.: Search engine usage statistics.

In daily life, translation services are one of the most popular usage areas of language models, especially multi-lingual language models. Google, the most widespread translation service, has troubles in its translation services regarding stereotypical bias, too. The Figure 2.6 demonstrates a translation from Turkish to English. Turkish is a gender-neutral language in terms of its pronouns, so there is no difference between *he* and *she*; they are both represented as *"o"*. When we input "cooking" and "being an engineer" terms in the translation, Google matches "cooking" with a female and "being an engineer" with a male, although there is no gender specified in the source language.

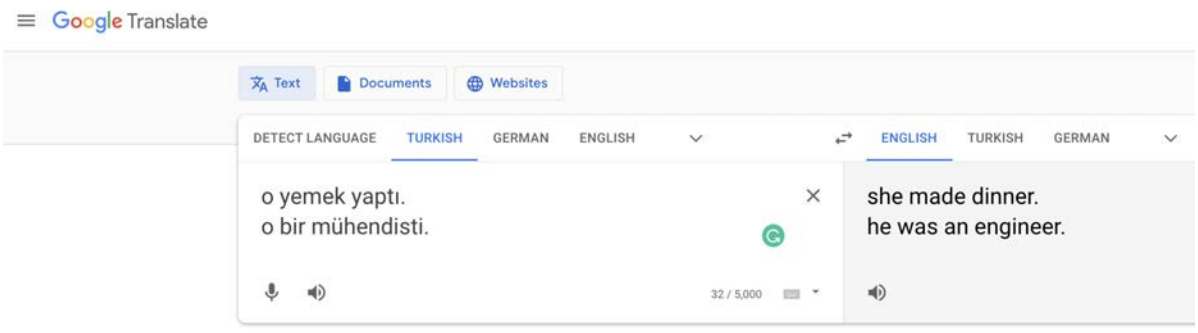

Figure 2.6.: Gender bias in Google translate.

Many examples and experiments can be carried out to prove the stereotypical bias of language models. Because of these examples, some people believe that Google or some big companies are having these stereotypes intentionally (Trielli and Diakopoulos, 2019; Bryan, 2019; O'Neil, 2016). These incidents, especially on popular platforms, might hurt billions of people who are even unrelated to these results. Some people might even blame a country (e.g., the USA) since a company is from that country, which might even result in political conflicts.

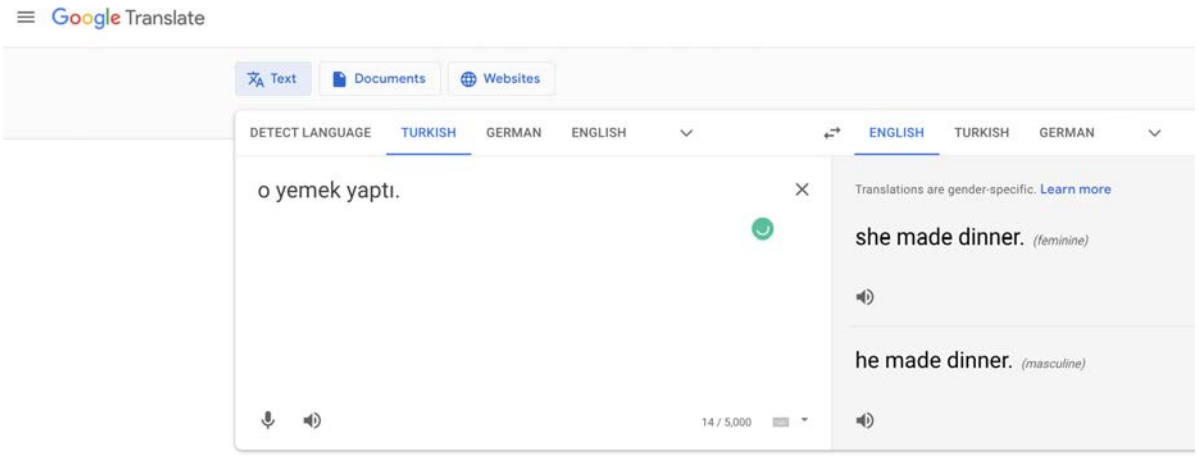

Figure 2.7.: Gender bias reduction in Google translate.

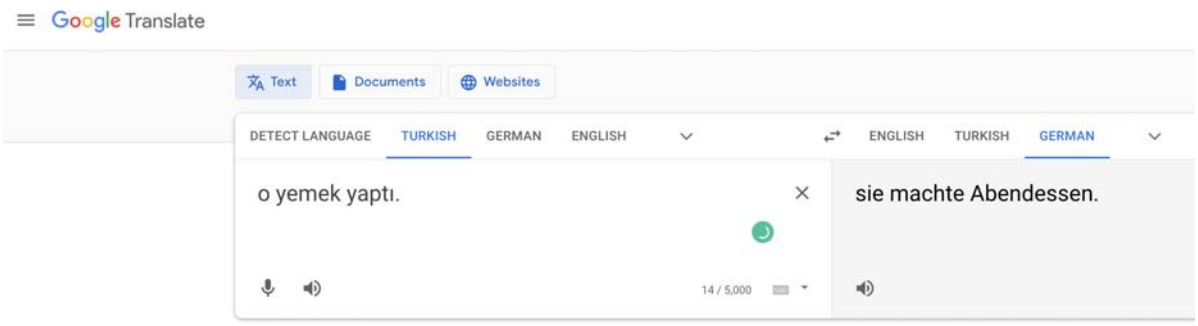

Figure 2.8.: No bias reduction in German language translations of Google translate.

Furthermore, these stereotypes also bring the question of "Who is responsible for these actions?". Are the engineers who designed a stereotyped program, the company providing the service, the people who create the program's data source, or the judiciaries who are not prohibiting these actions responsible for it? Or, from a different perspective, can the AI program be made responsible for its actions? There are many discussions about these questions, and there is no concluding decision yet. For instance, Maruyama

(2022) claims that the responsibility can not be given to the AI unless there are extremely minimal or no regulations on the AI since the responsibility can be considered only after giving its freedom to act. Nonetheless, at least there are ways to reduce or measure the bias and inform the programs' users beforehand (SKEEM and Lowenkamp, 2016; Ghallab, 2019). Google is absolutely aware that some of its products are producing harmful stereotypes and trying to reduce these biases (Johnson, 2020). When one inputs only the "cooking" in the Turkish example mentioned before, Google provides both masculine and feminine forms in English, as seen from Figure 2.7. On the contrary, this is not the case in German, as seen in Figure 2.8, even though German is one of the richest languages in the area of AI. Moreover, an interesting but meaningful experiment is also conducted with the Arabic language. "Muslims are terrorists." text is written in Google Translate as choosing Arabic as the target language, then we translated the result back to English as is seen in the Figure 2.9. The resulting translation became "Muslims are not terrorists.", which is wrong but a fairer translation. This example indicates that Google tries to reduce its bias, but sometimes it yields wrong and mistaken results that might create prominent disfavors or issues for the user. Therefore, there is still a huge necessity to research the stereotypical bias in language models to overcome these problems, especially in the multi-lingual context.

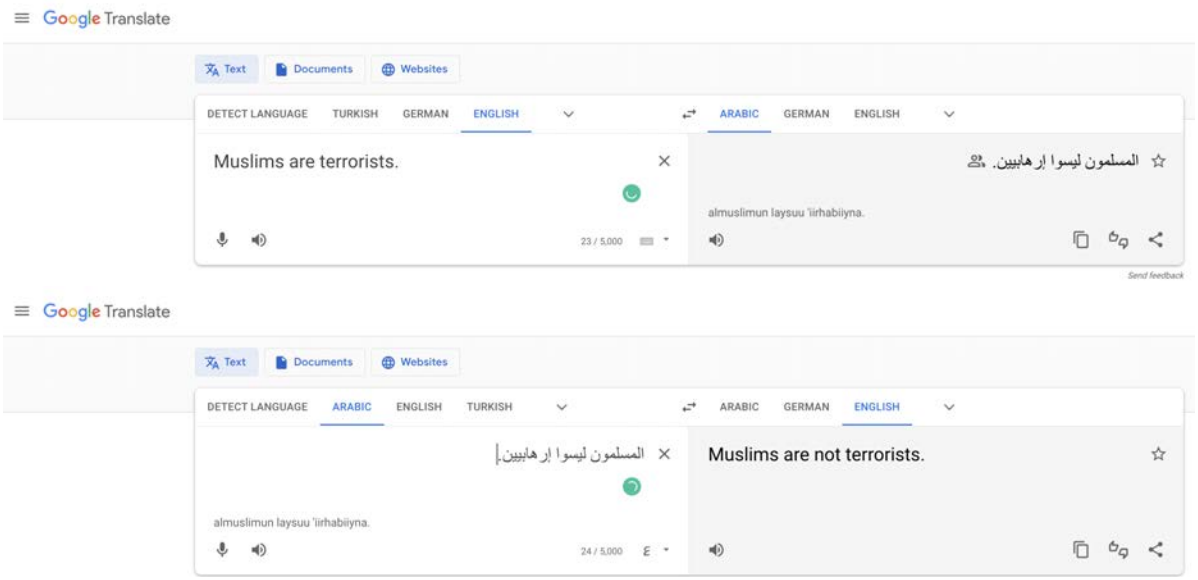

Figure 2.9.: Google Translate's bias reduction causes a wrong translation.

2.4. Related Work

Bias and stereotypes in AI, especially in NLP applications, are a broad research field since these stereotypes might hurt many people. Measuring the bias in a pre-trained model, researching the reasons for these biases, and mitigating the biases have become a common practice in recent years and attracted many researchers from other fields too. In this respect, Caliskan et al. (2017); Bolukbasi et al. (2016) are one of the first impressive works about detecting and proving the occurrence of stereotypical bias in AI applications, especially in NLP. For instance, their word embedding association tests (WEAT) show that European-American names have more positive valence than African-American names in state-of-the-art sentiment analysis applications. They claim this issue is a much broader context than having intentional bias among people since it is more challenging to analyze and research the reasons. Nadeem et al. (2020) measures the stereotypical bias mathematically in the English language by creating their own data set. Their inspiration for the term "Context Association Test (CAT)" comes from WEAT. Although these and most works are conducted in English, there is also admirable research about multi-lingual analysis. For instance, Stanovsky et al. (2019) conducts a practical experiment comparing gender bias in some of the widespread Translation Services. They discovered that Amazon Translate performs second best in German language translations among the picked systems. Moreover, three out of four systems attain the most satisfactory performance in German among eight different languages. A rationale for that might be German's similarity to the English source language. Lauscher and Glavaš (2019) measures different types of cross-lingual biases in seven languages from various language families. They reached an unanticipated outcome that states the Wikipedia corpus is more biased than the corpus of tweets and their results indicate that FastText is the most biased method among the four embedding models. Névéol et al. (2022) extends the CrowS-Pairs data set (Nangia et al., 2020) to the French language and measures the bias while providing the possibility to extend in different languages.
Other than detecting and measuring the biases, there are also incredibly inspiring works about the sources of bias and mitigating these biases (e.g., debiasing). Fundamentally, the source of bias is divided into two categories, one coming from the data and the second from the model (Mehrabi et al., 2019). The model sometimes overfocuses on these biases in data and applies these biases in its predictions more often than their occurrence in the data; this issue is called bias amplification of the model (Zhao et al., 2017; Hall et al., 2022). Hall et al. (2022) discovers a critical correlation between the

strength of bias amplification and the measures such as model accuracy, model capacity, model overconfidence, and amount of training data. Moreover, they imply that bias amplification is more substantial when it is less complicated to recognize group membership than class membership where the groups are like a gender group, an age group, or an ethnic group. The debiasing techniques typically try to overcome harmful bias amplification. Bolukbasi et al. (2016) is not only assessing the bias but also proposing debiasing techniques. Another state-of-the-art work is from Bartl et al. (2020), where they applied Counterfactual Data Substitution (CDS) to GAP corpus and finetuned BERT to mitigate gender bias in the initial pre-trained model. They achieved promising results in the English language; however, the same method performed unsatisfactorily to mitigate bias in the German language. Therefore, this proves that bias mitigation methods do not necessarily perform similarly in different languages, even if they belong to the same language family (e.g., Germanic). One reason for that different languages has different grammar, especially in terms of gender specifications. For instance, German is a gender-marking language and this might cause to encounter more debiasing difficulties in the German language compared to English. Furthermore, Meade et al. (2021) applies debiasing techniques for various kinds of bias types such as racial bias and religious bias instead of only gender bias. They use different benchmark data sets to show the debiasing performance, such as the StereoSet data set (Nadeem et al., 2020).

3. Material & Methods

3.1. Data sets and Data Preprocessing

3.1.1. StereoSet Data Set

We use the StereoSet data set (Nadeem et al., 2020), which is created by employing crowd-workers from Amazon Mechanical Turk. They also validated each example in the data set by five validators, and only the examples where 3 of the validators agree on the classification labels are retained.

The data set is designed to have two association tests to evaluate the models, the so-called intra-sentence test and the inter-sentence test. The intra-sentence test is used for the model to predict the probability of occurrence of specific words in the sentences. We essentially measure how the model would fill a blank in the sentences as shown in Table 3.1. We supply three candidates to the model for each sentence and inspect its prediction of that word's occurrence in the blank part of the sentence. For instance, "The Muslim was extremely BLANK in his practice." is the main sentence and is called a "context sentence" over the paper. In addition, "violent", "nonviolent" and "orange" are the words that are placed into the "BLANK" part. These words are called "candidate words" over the paper since they are the candidates to be substituted with the word "BLANK". The model should compute the probability of the candidate word being in the "BLANK" part of the context sentence. These three words are indeed in three different categories: stereotypical candidate, anti-stereotypical candidate, and unrelated candidate, respectively.

In inter-sentence tests, the probability of a sentence followed by another sentence is inspected. Fundamentally, how the model would predict the next sentence after reading a sentence is analyzed. For each sentence, there are again three candidates for the model to predict where each of them is from different categories: stereotypical, anti-stereotypical, and unrelated, as seen in Table 3.2. For example, "My professor is a Hispanic man" is the main sentence and will also be called a "context sentence" over

Fill in the blank with the appropriate candidate word.
Context Sentence: The Muslim was extremely _ _ _ _ in his practice.
Candidate Word 1 (stereotype): violent
Candidate Word 2 (anti-stereotype): non-violent
Candidate Word 3 (unrelated): orange
Target: Muslim Domain: Religion

Table 3.1.: Example for an intra-sentence test.

the paper. In addition, "He came here illegally", "He is a legal citizen" and "The knee was bruised" are sentences in stereotypical, anti-stereotypical, and unrelated categories, respectively, and they will follow the context sentence. These sentences will be called "candidate sentences" over the paper since they are the candidates for the next sentence of the context sentence. The model should compute the probability of the context sentence to be followed by the candidate sentence for each candidate sentence.

Find the appropriate sentence among candidate sentences to follow the context sentence.
Context Sentence: My professor is a Hispanic man
Candidate Sentence 1 (stereotype): He came here illegally
Candidate Sentence 2 (anti-stereotype): He is a legal citizen
Candidate Sentence 3 (unrelated): The knee was bruised
Target: Hispanic Domain: Race

Table 3.2.: Example for an inter-sentence test.

As stated above, one of the three categories is "unrelated". This candidate is an utterly unrelated sentence/word from the context sentence. This category exists to measure the performance of the language model and whether it predicts a meaningful option (e.g., stereotypical or anti-stereotypical candidate). Therefore, we measure not only the bias but also the performance of the model since when the bias decreases, the model usually performs worse, so it is worthwhile to measure the model's performance.
The targeted word(s) as a stereotype is known in each context sentence example. In other words, in each example, there is information for the "target word" which indicates to which group of people the stereotype is being done. For instance, in the above example intra-sentence test, the target word is "Muslim"; in the inter-sentence example, the

target word is "Hispanic". Hence, we are able to measure the bias for a specific target group as well. These target words are created by Nadeem et al. (2020) using Wikidata relation triples (<subject, relation, object>). The "relation" in these triples gives us the bias type (e.g., gender). There are four different bias types that the data set has; gender, profession, race, and religion. For instance, in the intra-sentence example above, the bias type is religion, as the stereotype is done on the religion of Islam. In addition, The "bias type" of the inter-sentence example above is race since the stereotype is done on the Hispanic race. It is also important to point out that some target terms contain multiple words, such as software developer and civil servant.

3.1.2. Descriptives

Nadeem et al. (2020) provide only the development version of the data set to the public and reserve the test set for themselves. Thus, only the development set is used in this work, and all the descriptions and the work will be about the development set.
There are 2123 sentence examples in the inter-sentence and 2106 in the intra-sentence data set. Of these 2123 inter-sentence examples, 2115 have unique context sentences, so there are some duplicate context sentences. For instance, "Spain is in Europe." occurs two times as the context sentence; however, the candidate sentences are all different in every duplicate context sentence; consequently, these few duplicate sentences do not cause any problem. Likewise, a few candidate sentences are also repeated in some cases, but their context sentence is always different. Therefore, there is no duplicate inter-sentence example. The same situation occurs in intra-sentence examples as well. Even though there are a few repetitions in the context sentences, there is no duplicate example in the data set.
From 79 unique target terms in the inter-sentence data set, the most common target term, "Ethiopia", has 33 occurrences, and the least has 20. For the intra-sentence data set, the number of occurrences of target terms is between 21 and 32 where the most common is "football player". There are also 79 target groups for the intra-sentence tests. Therefore, the data set is well-balanced with respect to the target terms. As seen from the Figure 3.1, "race" bias type occurs most commonly in both data sets. It is significant to mention that there are only around 80 examples for the "religion" bias type category.

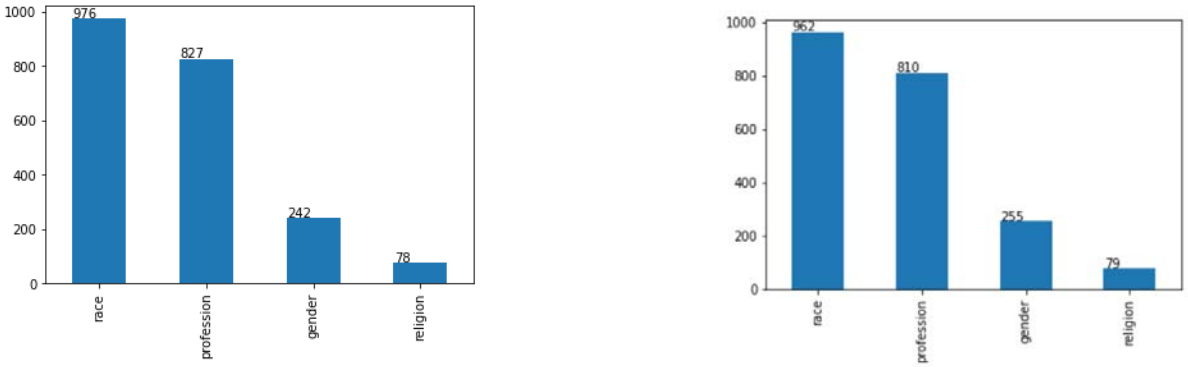

Figure 3.1.: Bias type distribution. (Left:inter-sentence data set, right:intra-sentence data set)

3.1.3. Data Set Creation

The data set is read as two pandas data frame objects (pandas development team, 2020), one for intra-sentence and one for inter-sentence, to have a more manageable and straightforward analysis. Each row in the data frame represents one example in the data set, with 16 columns. These columns include information about the example, such as example id, target term, bias type, and context sentence. Moreover, they contain the information for each candidate sentence, too, such as the sentence itself, gold label, id, and its labels.

For intra-sentences, there is only a masked word that differs between candidate sentences and the context sentence since it is a mask-filling task. Since the masked word is the only important part in the candidate sentences, they are separately stored in a different column to make the subsequent analysis more manageable and understandable. To extract the masked word from the candidate sentence, we follow the same algorithm with the authors of the data set (Nadeem et al., 2020). First, the index of the "BLANK" word is found in the context sentence, and then the corresponding word in this index for the candidate sentence is extracted. After that, we clean the masked word from the punctuations and add it to the new column in the name of "candidate word". This separation of the candidate word allows us to run the translations explained in the following sections.

Translating to German

The data set is translated to German using Amazon Web Service translation services in Python (boto3). The translation cost of AWS is measured according to the number

of characters in the input text. Therefore, we calculate the number of characters to be translated as a reference to have a rough idea about AWS pricing. There are around 130000 characters in the intra-sentence and 380000 characters in the inter-sentence data set to be translated.
A crucial point in the translation process is translating the "BLANK" word of the context sentences in the intra-sentence data set. This word must be kept the same in the outputs, and it must have cared a special word. If it is left as a standard word, it is translated as *"LEER"* (means blank/empty in German) or its different synonyms, and sometimes it is kept the same as "BLANK" automatically by AWS. These different context-based translations for the "BLANK" word would cause difficulties to be found in the following analysis. AWS's "custom terminology" approach is designed specifically for these cases when a user does not want to translate a word. Consequently, a custom terminology must be created for the "BLANK" words for the masks. The custom terminology can be achieved by giving a byte code of "en,de [endline] BLANK,BLANK" in Python.

3.1.4. Data Preprocessing

Fixing Issues in Translations

Initially, the custom terminology feature surprisingly did not work well with some examples. These few examples were also tested alone, but we noticed that this feature did not function on these specific examples and translated the custom word to the target language even though it was stated to not translate. However, a few weeks after, the same code produced correct results (no translations for the custom words), probably due to some fixes in AWS libraries. Therefore, it is crucial to check if there is an inter-sentence example that does not contain a "BLANK" word after the translations.
After the translations, all data set is checked for punctuations and special characters. Only some of the translated candidate words in the intra-sentence data set have some unnecessary characters, such as "-". We clean these to obtain more accurate estimations in the following parts. Another potential problem is the translated target terms for the "football player". One can check the sentences and notice that this profession is defined as American football players instead of soccer players in the original data set. In the German language, the translation for that should be "American-football-spieler" instead of "Fußballspieler", and this has to be fixed too. Furthermore, another issue in the translations is including a dash near the "BLANK" word. Dashes are commonly used after adjectives in German, so when the translator predicts that the "BLANK"

	Target Term	Context Sentence
English	Russian	The BLANK Russian sat on the couch.
German	Russisch	Der BLANK Russe saß auf der Couch.

Table 3.3.: An example of problematic translations

word is an adjective or defined by an adjective, it places a dash near it. These dashes are also cleaned from the "BLANK" word.

Moreover, a primary problem in the translations is that some of the target terms do not occur in the context sentences anymore after the translations. The target terms are translated independently of the context; however, in the sentences, they are in a context and are translated according to that context. Hence, the translations of the same word do not match with each other for some examples. The most common translation error occurs in the country or language names due to having a more comprehensive vocabulary of them in German. While in English, there is usually one common word for a person from a country and the country's language, on the other hand, in German, it is typically two separate words. For instance, "Turkish" can mean both the language and a person in English (e.g., Turkish girl and Turkish language). However, in German, the person would be called "Türke" or "türkische Person" and the language would be "Türkisch". An example from the data set is "Russian" as the target term and "The BLANK Russian sat on the couch." as the context sentence, seen in Table 3.3. Many target term examples have the same problem, such as Hispanic, Iranian, Ethiopian, etc. This common issue poses severe problems for the mismatch between the translation of the target term and the translation of the context sentence. Hence, the non-occurrence of target terms in the context sentences is due to the context-independent translations of target terms.

To detect these issues, we check whether the target term occurs in the context sentence or not for each example. We discover that there are 562 examples in the intra-sentences and 535 in the inter-sentences suffering from the same problem. This problem would get even more significant if the candidate words were not separated from the candidate sentences in the intra-sentence data set. Therefore, there is a need to change some of the target terms to adapt them to their context sentence. Since the target terms will be modified, their original English version is kept as a new column "target_original" for comparability, and they will be used in the further evaluation parts. We fix this problem with semi-automatic methodologies.

The targets which do not occur in their corresponding context sentence are analyzed, and Figure 3.2 displays how many problematic examples each target term has. The

Figure 3.2.: Number of problematic examples per target term. (Left: intra-sentence data set, Right: inter-sentence data set)

most problematic target term in intra-sentence is "Häftling", translated from "prisoner". "Prisoner" has different translations in German such as "Gefangene" and "Häftling". Since the "Häftling" word is gender-neutral and the plural version is also the same, we decide to use "Häftling" instead of its synonyms. When modifying the translations, one should be careful about the articles of these nouns (e.g., der/die/das, etc.). The articles are carefully modified in this work by working with native speakers. The second most problematic target term is "schuljunge", which is the translation of "schoolboy" in the original data. As the original version already reveals the gender of the word, the target term can be modified to "Schüler", as this word mainly occurs in context sentences. Another problematic target term is "Möbelpacker", the person who transports and moves possessions (e.g., mover). This term has multiple synonyms in German (e.g., Umzugsunternehmen, Beweger). However, the most appropriate word for the context sentences is "Möbelpacker" since it can be a person instead of a company and is gender-neutral. Consequently, all other translations in the context sentences are replaced with the word "Möbelpacker". Another target term that needs a fix is "Sierraleon". This term has only one correct usage in German, "Sierra Leone", and all other usages are converted to the correct version. Another important target term that needs to be investigated is "Lieferant", the equivalent of "delivery man" in the original data set. This word also has many translations in the German language, such as "Lieferbote" and "Zusteller". "Zusteller" is picked as the final word for this target term due to its gender-neutral structure. Another target term posing problems to the data set is "darstellender Künstler" which means "performing artist". The problem of not match for the target word for this case is primarily due to the grammatical cases ("Kasus") in German. Therefore, the correct version changes in every sentence and requires an example-by-example investigation. The last target term that can be fixed in this way is "Norweigan". Indeed, translations are wrong for this case due to the mistakes in the English StereoSet data set. The correct word for this target term should be "Norwegian", instead of "Norweigan".

The "Norweigan" word is converted to "Norweg", and the purpose of changing it to "Norweg" is to be fixed automatically in the subsequent processes.

Figure 3.3.: Number of problematic examples per target term after the initial semi-automatic fix. (Left:intra-sentence data set, right:inter-sentence data set)

After having these modifications, we reduce the number of problematic examples to 378 from 562 in the intra-sentence data set and 352 from 535 in the inter-sentence data set. The distribution of the remaining problematic target terms is plotted in Figure 3.3. These remaining problematic cases are fixed by another method by finding the closest word to the target word in the context sentence. The target word is compared with each word (words are found by splitting the sentence by white space character) in the context sentence by employing the "get_close_match()" function of the "difflib[1]" library. The cutoff variable is fixed by 0.4 as it can be interpreted as detecting the words with 40 percent similarity to the target word. A tricky point here is that the target terms have more than one word; however, there are only a few in the data set (e.g., "darstellender Künstler"), and they are excluded from this method. After finding the most similar word in the context sentence, the target term is replaced with that word. While in most cases, the unmatching of the target term is due to the different endings of the word (e.g., Russich and Russe), there are some cases where the sentence has a synonym (e.g., Herren and Gentlemen) of the target term. In these cases, the similarity rate is lower than 0.4, so the target term does not change. After this methodology, indeed, most of the problematic target terms are fixed, and it is reduced from 378 to 57 in the intra-sentence data set and from 352 to 41 in the inter-sentence data set. By this algorithm, we also fix the problem with the "Norweigan" word since the different word types in sentences are all similar to the word "Norweg".
Next, the rest of the problematic cases are plotted in Figure 3.4 and investigated one by one. "Ihr selber" and "sich selbst" (himself and herself in English) have a specific case since they have significantly different grammatical usage in German compared to English. Since there are approximately 50 problematic examples with these target terms, they

[1] https://docs.python.org/3/library/difflib.html

Figure 3.4.: Number of problematic examples per target term after the automatic fix by finding the closest word. (Left:intra-sentence data set, right:inter-sentence data set)

are not easily negligible. Thus, we manually fix the problematic examples with these target terms, and they are modified sentence by sentence by German native speakers. "männlich" is the third biggest problematic target term in this phase; however, it is easily fixable since all of its translations in the candidate sentences are the same word "Mann". Hence, the target word "männlich" in these examples is replaced by "Mann". Similar with "männlich", the target term "herren" is replaced with "Gentlement", and "Befehlshaber" with "Kommandant". All target terms are fixed at the end of all these changes except for four examples in the intra-sentence data set and five in the inter-sentence data set.

Figure 3.5.: Number of problematic examples per target term after all semi-automatic fixes. (Left:intra-sentence data set, right:inter-sentence data set)

3.2. Methods for Probability Predictions

3.2.1. Intra-sentence Predictions

The intra-sentence tests can be conducted by benefiting BERT's Masked Language Modelling (MLM) objective. In addition, as explained in Section 2.1.5, T5 has a highly similar objective to BERT's MLM. This BERT-style objective makes T5 extremely straightforward to infer for intra-sentence tests. However, GPT-2 does not have any objective related to Masked Language Modeling, and it is fundamentally a pure generative model. Hence, the inference for the probabilities will be made in a discriminative approach for BERT and T5-based models and a generative approach for GPT-2 based models. It is significant to note that this inference can be made directly by using the "fill-mask" pipeline, but the pipeline does not support T5 and GPT-2 based models, thereby lacking the benefit for the scope of this thesis.

Discriminative Approach

In intra-sentence tests, the main task is predicting the probability of a candidate word being placed in the masked part of the context sentence. However, the candidate word is usually multi-token, and the probability of the whole word cannot be calculated directly, so there is a multiple mask tokens case. Therefore, the candidate word is divided into tokens, and each token is unmasked every time from left to right. The algorithm to solve the problem is inspired by Nadeem et al. (2020). After manipulating the data set to multiple token masked words, as shown in Figure 3.6, more than three sentences for each context sentence might arise. Nevertheless, due to efficient object-oriented handling, the inference can be accomplished batch by batch and with multiprocessing.

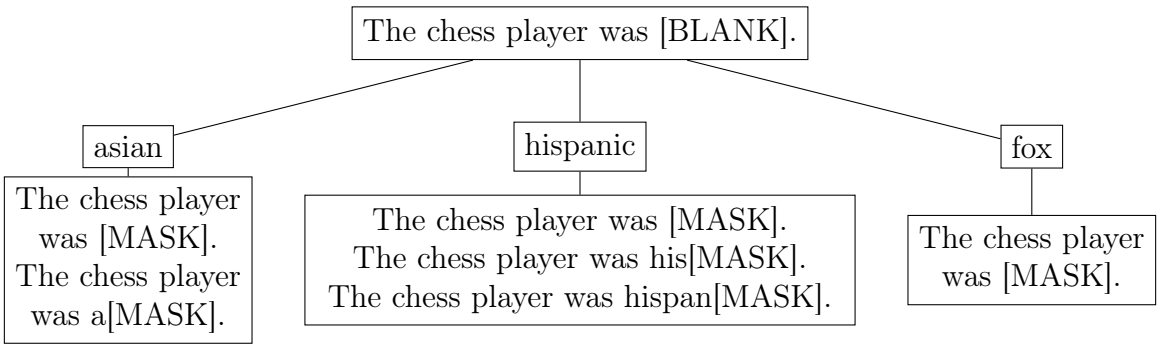

Figure 3.6.: An example for multiple mask tokens. There are six different sentences to be processed for only one example in this case.

We accomplish tokenizing the sentences as a batch by padding the sequences to the size of the longest sentence in the batch. This modification is implemented by extending the Nadeem et al. (2020) code where the padding was done to a fixed length (e.g., 256), with the aim of reducing memory consumption. After replacing the masked token with the word "BLANK" in the context sentences, these sentences are passed to the model with their corresponding input ids and attention masks. Then, a softmax operation is applied to the results' likelihood scores in its last dimension, which possesses all tokens in the vocabulary. The softmax function enables us to convert the output scores to a probability distribution by the Formula 3.1

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{3.1}$$

After acquiring the probability of the masked token, these probabilities have to be stored in a specific format. A dictionary of lists is created to store multiple probabilities (each masked token has a different probability) in a list format for each candidate word. Therefore, the keys in the dictionary are the candidate IDs, and the values are the list of probabilities for each candidate. Then, these probabilities for each candidate sentence are accumulated by averaging its corresponding list of probabilities. Hence, the resulting probability for a candidate sentence is the average of its tokens probabilities to be replaced with "BLANK" in the context sentence.

Moreover, one should be careful about the masked token of the model's corresponding tokenizers. Although all masked tokens are coded equivalently as "[MASK]" in BERT, one mask token might be coded differently than the other in T5. In BERT, the masked token is automatically gathered from its tokenizer. On the other hand, in T5, a sentence can contain multiple masks, and each of them is numbered as "<extra_id_0>", "<extra_id_1>", and so on according to the number of different masks in a sentence. Since there will be only one mask in our approach, "<extra_id_0>" is the only necessary string to be replaced with the masked part of the sentences. Thus, one assumption is that this thesis only has one mask for intra-sentence context sentences.

Another significant point that T5 has different from BERT is its encoder-decoder architecture. Due to its architecture, it requires some input to its decoder part, which should be labels of the task. Hence, due to this additional requirement of T5, the mask filling is done using the "generate" function (Wolf et al., 2019) in the state-of-the-art methodologies. The length of the generated output depends on its hyper-parameter "max_length". To get the likelihood scores for the vocabulary, one must set "output_scores" and "re-

turn_dict_in_generate" hyper-parameters to "True". The generator output of T5 always generates first the beginning token, then the "<extra_id_0>" and finally the token for the masked part. Therefore, if one wants to generate only one token for the mask, then the "max_length" parameter must be set to 3, and for two tokens, it must be set to 4. Furthermore, the likelihood scores of the generator function for T5 always include probability distribution for the "<extra_id_0>" as a first element and then for the "masked_word", which is the only important consideration for intra-sentence tests. Therefore, one must investigate the probability distribution given in the second element of the scores output.

Generative Approach

After initializing the GPT-2 tokenizer and the GPT-2 model, the pad token is added to the tokenizer since it does not have a pad token in its pre-trained form. The size of token embedding of the model is, of course, updated according to. After loading the tokenizer and the model, the "pandas" data frame is loaded using a specific data loader class. The new data instances include the candidate sentences and their corresponding IDs. By utilizing the special object-oriented programming technique to load the dataset, it is achieved to make the model learn in batch mode and with multiprocessing which was not existing in the Nadeem et al. (2020). The tokenizer encodes the candidate sentences batch by batch by padding the sentences to have the same size as the longest sentence in the batch. Obviously, the attention mask is returned from the tokenizer and utilized in the model to specify which parts are padding and should be ignored in the likelihood computations.

GPT-2 is a uni-directional model and approaches the sequence from left to right. Therefore, if one tries to generate the probability distribution for the masked part, the right context of the masked part would be ignored as the model considers only the left context. This behavior is not intended since the stereotype might be occurring in the right part of the mask. Thus, a higher-level calculation is necessary for GPT-2 sentences to consider all parts of the sentence. We handle this by generating the probability distribution for every token by providing their respective left context to the model; this can also be acquired by supplying the whole sentence in one to the model instead of inferring it separately for each token. In other words, the generation is executed for every token instead of only the masked part. In this case, the masked part obviously does not affect only one token on its right but the whole context on its right because of the native left-to-right behavior of the model. The output of this operation produces a separate

distribution for each token where each distribution expresses the likelihood distribution for the corresponding next token. Hence, the likelihood of generating a specific token is obtained by examining the previous token's likelihood distribution output.
The first token does not have any left context, and the probability of it cannot be calculated directly. Consequently, the probability distribution of the first token is computed individually from the other tokens. The start token of the GPT-2, which can be considered as the token before the first token (e.g., BOS token), is "<|endoftext|>", so the likelihood distribution of the first token is measured by providing the start token to the model as a left-context of the first token.
After calculating these likelihood distributions both for the first token and for the whole sentence, the softmax operation is performed separately over the vocabulary dimension to flatten the results into a probability space where each of the results is between zero and one. To merge these probabilities from each token, the following formula inspired by Nadeem et al. (2020) is used,

$$2^{\frac{\sum_{i=1}^{N} \log_2(P(x_i|x_0, x_1, ..., x_{i-1}))}{N}}, \tag{3.2}$$

where N is the number of tokens in the sentence.

3.2.2. Inter-sentence Predictions

Inter-sentence tests can be conducted by taking advantage of the BERT's NSP objective. In the NSP objective, the model is pre-trained to predict whether the sentences are consecutive or random. One can extract the likelihood of being the next sentence and compare it with other candidates. This straightforward algorithm can be applied in BERT and Multilingual BERT due to their native NSP objective. However, T5 and GPT-2 based models were not pre-trained in the NSP objective, in consequence, different ways to test inter-sentence examples in these models are considered. An effective approach is fine-tuning T5 and GPT-2 models for the Next Sentence Prediction downstream task and inferring the predictions from these fine-tuned models. This approach and the details of the fine-tuning process are explained in the next part. Another approach is predicting the probability of each word in the next sentence by using the generative characteristics of these models. The second auto-regressive approach is not specifically for Next Sentence Prediction but more about the model's language modeling abilities and predictions depending on all words in the candidate sentence individually

instead of a general prediction. The results of T5 and GPT-2 using the generative approach are expected to perform poorer than BERT due to their incapability of NSP task and a non-specific approach for NSP.

NSP Fine-tuning for GPT-2 and T5 Based Models

The main task of this process is teaching the model to predict whether a sentence is really the next sentence of a sentence or just a random sentence. Therefore, the data set used here does not need any labels, the positive examples (actual next sentences) can be created from paragraphs on the internet, and the random sentences can be inputted as negative examples. Thus, the learning process of this downstream task is completely self-supervised learning, similar to BERT's NSP pre-training. The crucial part is where the sentences would be collected and how many of them are needed.

We inspire this process mainly from Nadeem et al. (2020); however, many changes have been made, such as the data type of the data set. Different than Nadeem et al. (2020), the data set is collected from Wikipedia using the Huggingface "datasets" library (Foundation, 2022). The library provides prepared versions of English and German Wikipedia dumps extracted on May 1, 2022, in the Huggingface Dataset format. Using this format decreased the memory consumption of the process due to its unique on-demand reading feature. Huggingface Datasets save the data set in the disk when loaded instead of holding it all the time in the memory. 9.5 Million sentences take around 20 GBs of memory, posing a significant obstacle to the training process. By using this data format in the whole training process to hold the data, the memory consumption for the same data set decreased to 2.5 GB from 20 GB, which left many spaces for the model training.

The articles on every page of Wikipedia dumps include many sentences in paragraphs without splitting them. Hence, the articles are tokenized at the sentence level using Natural Language Toolkit (NLTK) (Loper and Bird, 2002) with multiprocessing. Then, after shuffling the data set, it is "exploded" such that each row would include a sentence and the ID of the article that the sentence belongs to. For each row in the data set, the sentence and the ID of the next row are added as a new column; this enables to create the consecutive sentence tuples. The main idea of creating the data set is to assign its next sentence, which will be called "related", and one random sentence, which will be called "unrelated", for each sentence. For the next sentence, this is achieved by filtering the newly created data set where the ID of the sentence and ID of the next sentence are equal. One might initially consider that they would always be equal; however, when the

sentence is the last sentence of the corresponding article, the next sentence would be from even a different article instead of the following sentence. The unrelated (random) examples are created by drawing a random sentence for each sentence. It is crucial to note that the random sentence should be taken from a different article than the article to which the original sentence belongs to. This strategy would help the model to have more accurate predictions by being able to differentiate between articles. Both assigning the related sentence and the unrelated sentence can be achieved by benefiting multiprocessing. Finally, these two data sets, where one of them includes the related examples and the other unrelated, are concatenated and shuffled.

It should be noted that the final data set would have around double the size of the original one (after sentence tokenization) as there becomes two rows, one for related and another for unrelated, for each sentence. Due to efficient data format usage and multiprocessing, creating a data set consisting of 9.5 Million sentences takes only a few minutes using eight Central Processing Unit (CPU) cores. In addition, the process does not exceed even five GB of Random-Access Memory (RAM) usage. As an example to decide the number of articles to use, 220000 articles on average form 9.5 million sentences in the final data set. This whole algorithm is done for both English and German languages, and then they are concatenated and shuffled. There is the same number of articles used for both languages to ensure a balanced data set. In conclusion, half of the whole data set is for German, and the other half is for English; for each language, half of the corresponding data set has positive examples (actual next sentence), and the other half has negative examples (random sentence).

The monolingual English GPT-2 is already fine-tuned by Nadeem et al. (2020) with 9.5 Million examples from Wikipedia articles, and the authors provide the model. Therefore, there is no need to re-do the same process due to the environmental and time costs of fine-tuning the giant model on such a large data set. However, since there was no work on multilingual GPT-2 models and for T5, there is a strong necessity to fine-tune these models for the NSP downstream task. The employed GPT-2 based models are "GPT2Model" from the HuggingFace transformers (Wolf et al., 2019). It is a bare model without any head and is ready to be fine-tuned by adding a new head. Inspired by Nadeem et al. (2020), the head of the model is designed as three linear layers sequentially. It outputs two results: one would be the likelihood of having the real next sentence, and the other would be the likelihood of having a random sentence. Loss function and many specifications of the training process are kept the same with the work of Nadeem et al. (2020) to achieve the comparability of the results. Hence, the cross-entropy loss (Zhang

and Sabuncu, 2018) is used to measure the loss of each step as its formula is given at 3.3. Gradient accumulation is also provided to be used optionally to increase training speed. The gradient accumulation saves gradient values in every batch, proceeds to the next batch, and adds up the new gradients; then, it does weight updates after several batches have been processed by the model, instead of updating the weights in every batch (Kozodoi, 2021). As a consequence of skipping the weight update in some batches, the training process accelerates. The gradient accumulation can be seen as a pseudo batch mode as it allows the model to combine more examples and does the computation for all. Accumulation steps are the number of batches to combine for updating weights. Furthermore, Graphical Processing Unit (GPU) machines are supported in all parts of the training process and suggested to be utilized by tuning appropriate hyper-parameters in the large models to save an enormous amount of time.

$$-\left(y\log(p)+(1-y)\log(1-p)\right) \tag{3.3}$$

Since GPT-2 was not trained with a padding token, a padding token is added to its tokenizer, and the token embeddings of the model are resized according to. Adam algorithm (Kingma and Ba, 2014) with weight decay fix (Loshchilov and Hutter, 2017) is used as the optimization for the training process. Two different learning rates are used where one is for the core model, and the other is used for the head part of the model. For the former, 5e-6 is used as a learning rate; for the latter, it is 1e-3. Yet, the learning rates are not fixed; a scheduler with a warm-up period is used. The scheduler has hard restarts, and it decreases the values of the learning rate by a cosine function (Loshchilov and Hutter, 2016). Utilizing warm-up steps means keeping smaller learning rate values in the beginning and slowly reaching the specified learning rate value with the step size defined by warm-up steps. The warm-up period is operated to avoid significant shifts in the gradient vector at the beginning to diminish the impact of over-fitting to the initial estimates. An image that depicts the general flow of the learning rate under this scheduler is given in Figure 3.7. The parameter for the number of warm-up steps is a fixed number (250) in their work; however, the warm-up should change according to the data size, batch size, and accumulation steps. Otherwise, with a small data set, the whole training might be completed in the warm-up part without reaching the central learning part. Thus, the warm-up steps are chosen as $(total\ number\ of\ training\ steps)\ //\ 100\ +\ 1$ where the total number of training steps is measured by $((number\ of\ batches)\ //(accumulation\ steps)) * (number\ of\ epochs)$, so the warm-up steps is chosen to be around 1 percent of the training steps. Plus, one is

added to ensure the existence of a warm-up procedure since the division might get 0 as it is an integer division. Most of the other hyper-parameters are kept the same with the Nadeem et al. (2020) to compare the results with the GPT-2 model that they trained.

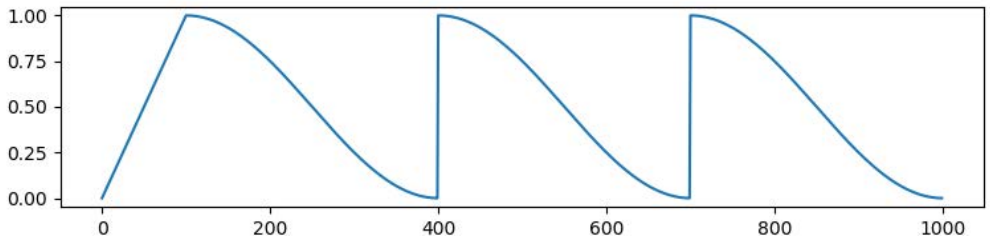

Figure 3.7.: Cosine learning rate scheduling with hard restarts after a warm-up period (Wolf et al., 2019).

The tokenization of the sentences is done batch by batch and tuples of sentences (e.g., sentence pairs) where the first is the main sentence, and the next is the (real or fake) next sentence. The padding is done for each batch as a size of the longest sentence pair length in the respective batch. The maximum sequence length is also supplied as 256, not to exceed the memory size during the training. Token type IDs and attention masks are significant to return in this step due to having batched training and providing pairs of sentences.

Moreover, it is incredibly crucial to state that the string of the sentence texts is given in a different version for T5 models. As described in Section 2.1.5, T5 models are pre-trained on different downstream tasks by adding a "prefix task" name to the beginning of the input sequence, so they are taught to operate one task by adding the name of the task to the start of the sequence. Therefore, the text "binary classification: ", a unique wording in the T5 tokenizer, is added to the start of every input sequence. This prefix task name is not required in T5 fine-tuning, but it is recommended to have it to speed up the convergence process. On the contrary, Multilingual T5 models (Xue et al., 2020) were not pre-trained with downstream tasks, unlike the original T5 model. Multilingual T5 models are only pre-trained with the unsupervised language modeling objective, thereby withdrawing the advantage of providing prefix tasks for these models' fine-tuning processes. It is only helpful if there is multi-task learning, and it is not in the scope of this work.

Furthermore, T5 and mT5 models differ from others by being a sequence to sequence (Seq2Seq) model, so the model outputs a string instead of an integer or a float number. As a consequence of that, the labels to the model must be supplied as tokenized string

sequences instead of binary variables as 0 and 1. Hence, the labels are converted to their string form "0" and "1" for 0 and 1, respectively, and then tokenized using the T5 tokenizer. The need for label transformation is also the case for multilingual T5 models, so this transformation for the labels is also applied in multilingual T5 fine-tuning.
For GPT-2 and BERT-based models, the forward function of the model is straightforward. The input IDs and attention masks are provided to their corresponding model to obtain the likelihoods, then to the added head layers. The output of the head layers is the likelihood for the binary classification as it has only two output channels. Then, the loss is calculated using the cross-entropy loss measured by these likelihoods and the ground-truth labels. Then, the loss is divided by the accumulation steps because its the loss for that accumulation of the batches and added to the total loss (e.g., running loss). Prediction of the model is acquired by applying softmax operation on the likelihoods and choosing the element with a higher probability. In fact, the softmax operation is unnecessary in this case since the only interest is the comparison of the values to identify the greater value. Then, the accuracy is computed by comparing these final predictions of the model with the ground-truth labels.
For T5 and MT5 models, calculating loss and accuracy also slightly differ from the other models due to its encoder-decoder architecture and teacher forcing methodology, expounded in Section 2.1.5. The loss is computed from the forward function of the T5 models and returned in its output. As a consequence, a separate calculation for the loss is unnecessary, and the loss function used in T5 is also the cross-entropy loss we used for the other models. (Raffel et al., 2019). The forward function of the model is called by passing the encoded input, attention mask (due to batch training), and the transformed labels. We do not compute the model's accuracy by the same process in T5 since the accuracy of the model's predictions tends to overfit and becomes overestimated due to additionally providing the labels to the model. In most cases, the accuracy scores, even in the first batches, are over 90% if measured from the predictions of its forward function. Therefore, the accuracy should be measured by calling a method without providing the labels. Calling the "generate" function is the most appropriate way to measure it because it is based on generative prediction and applies it without teacher forcing. Hence, the accuracy calculations are processed by applying the "generate" function to remove the bias in the predictions. Since the expected output has one token, the max length hyper-parameter is given as three as elucidated in Section 3.2.1. After running the generate function, the likelihoods to obtain "0" and "1" are compared, and the greater one is chosen for the model's final prediction. Then, the total loss and the

accuracy are calculated in the same way with GPT-2 models.
After calculating the loss, the gradients of the loss with respect to the weights are calculated by backward propagation, also known as "backpropagation". After the backward propagation, the gradient norm of model parameters is clipped for each accumulated batch, the scheduler and optimizer are updated, and gradients are zeroed. Clipping norms are used to overcome exploding gradients. The derivative values often become too large compared to the others and have an over-effect on the weights in the weight updating step. Sometimes they even become infinitely large and manipulate the weights; this situation is called "exploding gradients". Some methods to overcome the problem include regularization or clipping the gradients. Gradient clipping is fundamentally clipping the gradient vectors to a threshold as it is depicted in Figure 3.8. In this thesis, the threshold is considered as 1.0 (unit length) to have comparability with Nadeem et al. (2020). Moreover, PyTorch accumulates all of the previous gradients in every step, and to overcome it, gradients must be zeroed in every step.

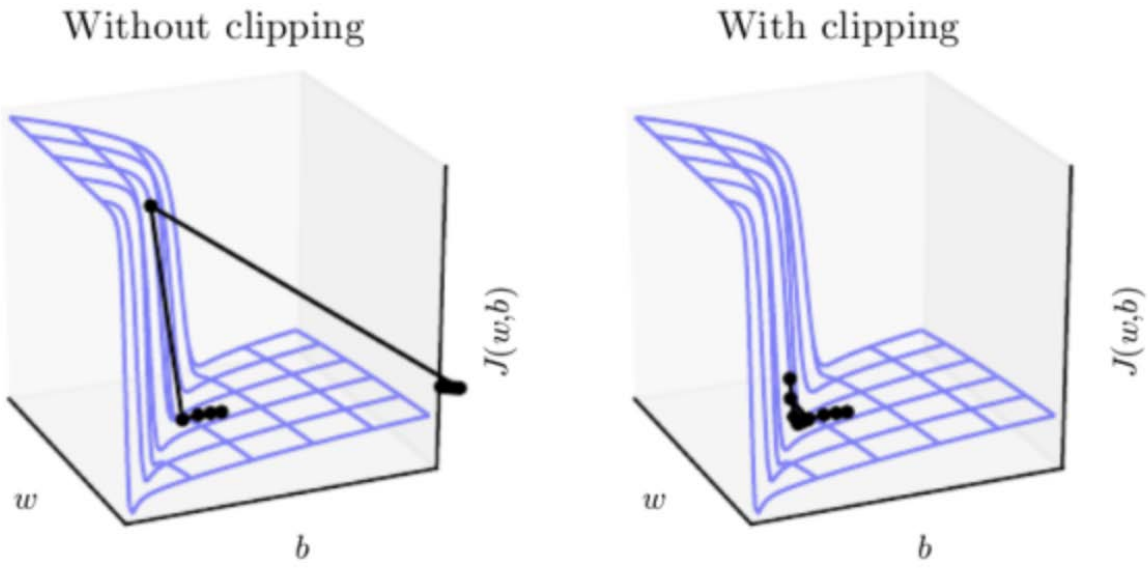

Figure 3.8.: (Left) Gradient descent without gradient clipping receives a huge gradient. (Right) Gradient descent with gradient clipping has a more moderate reaction (Bajaj, 2022).

The training process results are saved in every 500 examples for monitoring purposes. The current number of batches processed in this monitoring window is counted in a variable called "ticks", and the total accuracy and total loss are divided by this number to obtain the current accuracy and the loss. Indeed, the current loss is multiplied by

accumulation steps due to the optimization done by gradient accumulation. These results are logged in the Weights and Biases platform (Biewald, 2020) to track all the progress in an appropriate environment. After each monitoring, the total loss, total accuracy, and the ticks are reset.

Furthermore, to speed up the training process, support for Half-precision (FP16) floating point format in native PyTorch (Paszke et al., 2019) is added. PyTorch and most machine learning frameworks use a 32-bit floating point format in their computations, yet using only 16-bit is enough for many processes, and 16-bit enables to have faster computation. Authors of PyTorch claim that their mixed-precision support improves performance while maintaining accuracy. However, the performance increase does not always occur, and the increase is not always the same even if it occurs (Platen, 2022). The performance change mainly depends on the model and the machine that it runs on.

Checkpointing is crucial in training deep learning models due to longer training times. One can stop the training process, save the model and then continue the training back again by loading the model. The details of the present parameters of the model are also saved with its weights to be able to find out when the model quit the training process. Saving the model regularly is also significant in case of unexpected crashes (e.g., out of battery). The program saves the model weights and other training parameters for that moment in every 20% of the total training length with a name that includes the current number of batches processed.

After training the model using the same program, the testing can be done by providing appropriate hyper-parameters. Creating the data set in the testing process is the same approach as training; the labels must be created again to calculate the evaluation metrics (e.g., accuracy). After turning on the evaluation mode for the model, we load the model and move it to the Cuda device if necessary. All process is the same with training, but without updating the weights. Therefore, the gradient clipping, zeroing, and optimizing are skipped since they are unnecessary for testing purposes. The testing accuracy is measured by taking the mean of the accuracy values from each step.

Discriminative Approach

The data set is loaded with the same data loader class for the intra-sentence predictions explicated in Section 3.2.1. Due to this class, multiprocessing and batch inference are made possible in the predictions of the next sentences. Each candidate sentence is added to its corresponding context sentence by creating a tuple variable as a sentence pair. This sentence pair and the candidate's ID are only needed data for the inter-sentence tests

in discriminative approaches. Since there are two different sentences for each candidate, these sentence pairs are inputted to the tokenizer as a sentence pair tuple. However, for T5, as we state in the explanation of T5 fine-tuning, we fine-tune it by adding "binary classification: " text to the beginning of the sentence as a prefix task. Therefore, the input must be manipulated with the same format in the inference time.
After obtaining the input IDs, the corresponding model is called to produce the likelihoods. Since BERT is pre-trained with NSP objective and its "BertNextSentence" model is called, it produces only two outputs, either 0 or 1. In the BERT's pre-training, the positive output (1) is encoded for the unrelated sentence pairs, and the negative output (0) is encoded for the actual next sentences. On the other hand, in our fine-tuning process explained in the previous part, the positive output indicates an actual next sentence, and the negative output indicates a random sentence, which is the opposite of encoding in BERT. This difference results from the labeling in the dataset creation of the fine-tuning part. In T5 and multilingual T5 models, the label encoding is quite different from the other models due to their sequence-to-sequence model structure. The output of the results is directly compared with the tokenized "0" and "1" string texts. The softmax function is operated to convert the likelihood predictions to a probability distribution. Then, for each candidate sentence, the probability of predicting 0 for BERT and 1 for others is saved in a separate predictions file with their corresponding ID values.

Generative Approach

To have a clear separation between context and candidate sentences, the last character of the context sentence is checked and added a dot (".") if it does not already include any punctuation character. After loading the dataset by utilizing the standard dataset class for all approaches, the whole inference process is made compatible with batch processing and multiprocessing, where they were not supported by Nadeem et al. (2020).
The rest of the inference process, including tokenization, differs substantially between T5 and GPT-2 based models. In T5 models, the candidate sentence is fully masked; this creates difficulty in predicting the whole next sentence for the model though. A general form of the input sentence to the model's encoder layers is "(context sentence) <extra_id_0>". A specific example is "My professor is a Hispanic. <extra_id_0>". However, to handle this cumbersome prediction, the "teacher forcing" methodology explained in Section 2.1.5 is utilized for the model to learn from labels while predicting. By default, forward call for the T5 models, the teacher forcing method is utilized, and input for the encoder and another input for its decoder layers must be provided. The

input for its decoders is designated to have a form of "<pad> <extra_id_0> (candidate sentence)" and a specific example can be "<extra_id_0> He is a legal citizen.". When a user provides the input for the decoder as a label, T5 models shift the tokens to one position right, prepending the start token to the beginning of the sequence, before providing it to the decoder. Thus, a padding token "<pad>" must be added manually if the sequence is inputted directly to the decoder. By utilizing the teacher forcing technique and providing the labels, the model does not predict a token only by relying on the context sentence; instead, it benefits from the other tokens in the candidate sentence as well.

After tokenizing the sentences by padding them to the longest sequence for both encoder input and decoder input separately, the outputs of tokenizers are moved to the GPU machine if the inference is being made on it. Then, the T5-based model is called by supplying input IDs and attention masks of the encoder and decoder. Softmax operation is applied after that to obtain a probability distribution. Next, the output predictions (it is only for candidate sentence) is looped over for each token, and the probability prediction for each token is stored in a list. In the model's output, the last index is for the probability after the end of the candidate sentence. It usually predicts the end of the sentence token ("</s>") since the last token is sentence-ending punctuation; hence, the last index should not be considered in the probability calculations. After obtaining the probabilities of each token, they are combined by applying the Formula 3.2 given in the generative approach of intra-sentence tests. Fundamentally, the logarithm in the base two of the probability of each token is computed. Then the power of two is applied to the mean of these previously calculated logarithm values.

The inference of GPT-2 for the generative approach in inter-sentence tests is quite different from T5; nevertheless, it has many similarities with the generative approach in intra-sentence tests. The context and candidate sentences are directly merged by separating them with a whitespace character and can be shown as "(context sentence) (candidate sentence)". This version of the sentence will be called a "full sentence", and a specific example can be given as "My professor is a Hispanic. He is a legal citizen.". The candidate sentence and the full sentence are tokenized separately, and the tokenization for them is applied by padding them to the longest sequence in the batch. Then, the tokenized inputs are moved to the GPU machine if necessary.

Since GPT-2 is a uni-directional model, the probability of the first token ("My" in the example above) cannot be measured directly from the sentence itself. Thus, we input the model with the start token of the GPT-2 tokenizer ("<|endoftext|>") and analyze

the result of it for the first token of the full sentence, as it is also accomplished in the intra-sentence tests. Then, the probability of the first token is gathered after applying the Softmax operator.

There are different ways to measure the final score of a candidate sentence; the Nadeem et al. (2020) is measuring it by calculating the probability of the candidate sentence by providing the model only the tokens of the candidate sentence and dividing it by the probability of the context sentence. However, this ratio of candidate sentence over the context sentence does not evaluate any dependence between the context and candidate sentence; it measures the probability of them entirely separately. The context sentence is given to the model on its own and the candidate model also, thereby unsatisfying this approach for the purposes of inter-sentence tests since the test fundamentally tries to measure the dependence between context and candidate sentences. Nadeem et al. (2020) states that the results of this generative approach are not satisfying; we find out that this is probably due to using a wrong ratio since the results found in this work with this approach are incredibly satisfying.

We propose six different final score calculation cases, and all these cases are compared, and three final methodologies are selected to be the best. The first option is the one utilized in Nadeem et al. (2020), and it will be named "orig" as an abbreviation of the original over the paper. The probability achieved by providing only the candidate sentence to the model will be called $P(only\_candidate)$, and $P(context)$ would mean the probability obtained by providing only the context sentence to the model. Note that the computation of a score of a sentence is designed by combining each token's probability by taking their logarithm in base two, averaging it, and applying the power of two as more details are given in the previous parts of this thesis. $P(full\_sentence)$ is formed by considering all tokens in the full sentence after providing a full sentence to the model. $P(candidate)$ is measured by considering only the tokens in the candidate sentence after providing a full sentence to the model, thereby including the effect of the context sentence on the candidate probability in this case. Six different final score calculation options, their corresponding mathematical depictions, and their scores on GPT-2 using English data are given in the Table 3.4.

The main interest of the inter-sentence tests can be mathematically formed as $P(candidate \mid context)$, which can be formulated as,

$$P(candidate \mid context) = \frac{P(candidate) \cap P(context)}{P(context)} \tag{3.4}$$

where $P(context)$, which is the probability of context sentence to exist, is, in fact, iden-

Case No	Formula	LM Score	SS Score	ICAT
orig	$\frac{P(only\_candidate)}{P(context)}$	58	46.4	55.6
b	$\frac{P(candidate)}{P(full\_sentence)}$	73.9	52.7	69.9
c	$P(full\_sentence)$	76.8	51.1	75
d	$P(context)$	76.3	52.1	73
e	$\frac{P(context}{P(only\_candidate)}$	85.5	60.7	67.2
f	$\frac{P(full\_sentence)}{P(only\_candidate)}$	62.6	57.7	53

Table 3.4.: Various options of calculating scores with generative methodology in inter-sentence tests. The ICAT column indicates the achieved ICAT scores of each option on the GPT-2 model.

tical for all candidates since their context sentence is the same. Hence, the denominator is ignorable in this formula; the primary focus should be $P(candidate) \cap P(context)$, the existence of both candidate and context sentence together. This intersection can be interpreted as inputting the context and candidate sentence togetherly into the model, and that cases are actually "c" and "d" in Table 3.4. Hence, the options "c" and "d", which attain the highest Idealized Context Association Test (ICAT) Scores, make the most sense in terms of mathematical interpretation. Due to case d's similarity to the T5's generative calculations, we pick case "d" as our preferred technique to obtain comparability with T5 results. However, case "e" acquires the highest language modeling score; thus, we also report the results from cases "c" and "e", too.

3.3. Bias Evaluation Metrics

The predictions calculated by the methods expounded in Section 3.2 must be evaluated to derive a model's final Stereotype Score (SS). However, as it is elucidated in Section 3.1.1, only scrutinizing the amount of stereotypical bias in a model is insufficient. A random model that always outputs random candidates would also be non-stereotypical, but it would not have any language modeling ability. The ideal model should excel in language modeling ability by maintaining fairness regarding stereotypical bias. Therefore, the language modeling score and the ICAT score measurements from Nadeem et al. (2020) are also used in this work.

Although the main inspiration of the work is Nadeem et al. (2020), there are an extreme amount of differences between the codes. There are also mistakes identified in the code shared by Nadeem et al. (2020). For instance, the code that the authors shared calculates the "count" variable mistakenly due to an indentation mistake in line 49 of the evaluation code for the repository that is valid at the time of this work. They add the same example to the dataset every time for each sentence, as a consequence, the created data set duplicates each example three times. Consequently, the count number, which is part of the output, multiplies by three for each score and the target term. Hence, the results for the count number, which indicates the number of examples, must be considered as one-third of the resulting value. This mistake poses a higher complexity to the program and causes it to be longer since the size of the dataset is tripled. However, this mistake does not affect the scores published since the number "3" cancels in the ratios. The proposed correct version of the code and the results will be given with this work.

3.3.1. Stereotype Score (SS)

A stereotype score is designated to assess the potential number of stereotypes in language models by comparing the model's preference over the stereotypical and anti-stereotypical candidates. Always preferring an anti-stereotypical candidate is also appraised as discriminatory behavior since it would also create unfairness for the stereotypical group. Thus, a model must prefer neither stereotypical nor anti-stereotypical to be considered unbiased. In other words, the ideal model would prefer stereotypical and anti-stereotypical candidates in equal numbers. Mathematically, the number of examples with a higher probability for a stereotypical candidate than an anti-stereotypical candidate is divided by the total number of examples as formulated in Formula 3.5. This

score should be 50% for an unbiased model as it would pick the stereotypical candidate half the time.

$$SS = \frac{1}{N}\sum_{i=1}^{N} g(x_i) * 100,\ g(x) = \begin{cases} 1, & (x_{stereotype} > x_{antistereotype}) \\ 0, & (x_{stereotype} < x_{antistereotype}) \end{cases} \tag{3.5}$$

3.3.2. Language Modelling Score (LMS)

As stated before, being an unbiased model is not enough; the model should also perform satisfactorily in language modeling. Language modeling is assessed by measuring the number of examples that the model prefers the stereotypical and anti-stereotypical candidate over the unrelated candidate. For every example, the probability of a stereotypical and anti-stereotypical candidate is compared with the unrelated candidate. It is counted towards the "related" example if these meaningful candidates have a higher probability than the unrelated candidate. Then, the number of related examples is divided by two since it is counted for both anti-stereotypical and stereotypical. Then it is divided by the total number of examples to acquire the relative ratio as the Formula 3.6 states. The ideal model should always prefer both stereotypical and anti-stereotypical candidates over unrelated candidates; thus, it would have a 100% language modeling score.

We inspire the calculation of the score by Nadeem et al. (2020); however, their code and the explanation in the paper contradict with each other at this point. In the paper, the calculation is written to be counted towards the meaningful example for "either stereotypical or anti-stereotypical" candidate's superiority; indeed, it is counted towards "both stereotypical and anti-stereotypical" candidate's superiority. The difference would be apparent in an example where the stereotypical candidate's probability is higher than the unrelated candidate's, which is higher than the anti-stereotypical candidate's. In this example, the score would be 100% according to the paper; however, it would be 50% according to the code that the authors published. Our approach is based on the code that they published since the same results with their publication are, in fact, reached by the code.

$$
\begin{aligned}
LMS &= \frac{1}{2N}\sum_{i=1}^{N} g(x_i) * 100, \\
\text{where } g(x) &= \begin{cases}
2, & (x_{stereotype} > x_{unrelated}) \wedge (x_{antistereotype} > x_{unrelated}) \\
1, & (x_{stereotype} > x_{unrelated}) \wedge (x_{antistereotype} < x_{unrelated}) \\
1, & (x_{stereotype} < x_{unrelated}) \wedge (x_{i_{antistereotype}} > x_{unrelated}) \\
0, & (x_{stereotype} < x_{unrelated}) \wedge (x_{antistereotype} < x_{unrelated})
\end{cases}
\end{aligned} \tag{3.6}
$$

3.3.3. Idealized CAT Score (ICAT)

The language modeling score and the stereotype score usually contradict each other, as an improvement of one usually deteriorates the other. The ICAT score, which is inspired by Nadeem et al. (2020), combines both scores to overcome this trade-off between the two scores and show a final evaluation of a model. The formula of the score is written as,

$$
ICAT = LMS * \frac{min(SS, 100 - SS)}{50} \tag{3.7}
$$

The formula satisfies three primary axioms that need to exist in the final evaluation: an axiom for an ideal model, a random model, and an entirely biased model. The first axiom is that a completely unbiased model which always prefers meaningful candidates (e.g., 100 LMS, 50 SS) should score 100. The second axiom states that an entirely random model (e.g., 0 LMS, 50 SS) should score 0. The third axiom guarantees obtaining a 0 ICAT score for a model that always picks the stereotypical or anti-stereotypical candidates over the other (e.g., 0 or 100 SS).

3.3.4. Multi-Class Evaluations

In the inspired work (Nadeem et al., 2020), the predictions were evaluated by considering each target term as one class, and they defined it as a multi-class classification problem. In this thesis, we not only used multi-class ideas but also considered all examples together without separating them into different classes for each target term.

There are some logical mistakes in the multi-class evaluations in work from Nadeem et al. (2020). The macro score is the average of the main scores (e.g., F1 Score) for each class, so the main score should be computed for each class first. Therefore, it first calculates the main score for each class and then averages them. On the other hand, the micro

score has a different order of these two steps; first, it takes the average of all sub-scores (e.g., True Positive, False Negative) for each class and then calculates the main score (e.g., F1 Score) from these average sub-scores. In this work, the main score is considered ICAT Score, and the sub-scores are considered Language Modeling Score (LMS) and SS scores as ICAT Score is formed by them.

However, in work from Nadeem et al. (2020), the macro-ICAT and micro-ICAT are named in the opposite way of their definition, which results in misnaming and false reporting in the publications. The current authors of that work were contacted, and they also confirmed the misnaming.

4. Experiments & Results

4.1. Fine-Tuning

Since GPT-2 and T5-based models do not have direct support for Next Sentence Prediction objective, these models are fine-tuned, and the NSP head is added as described in Section 3.2.2. The process is not done for the mono-lingual English GPT-2 model because it is already trained in a similar way by Nadeem et al. (2020) and the repetition of the process would have both environmental and financial costs, as a result, the same model with NSP head is downloaded and utilized. Moreover, all the training processes are carried out on the Tesla V100-SXM2-16GB GPU machine, so the machine has 16 GB of memory.

4.1.1. Multilingual GPT-2

The multilingual GPT-2 model developed by Tan et al. (2021) is chosen with the reasons expounded in Section 2.1.6. In fact, another multilingual GPT-2 model, which is from the user "miguelvictor" in Huggingface, was initially chosen to carry the training out. However, the model was too large to train on the machine arranged and to compare with other models.

Multilingual T5 tokenizer is utilized instead of GPT-2 tokenizer due to its usage in the model's pre-training process (Tan et al., 2021). 110000 articles are considered for each language, constituting around 9.5 Million sentences, the same number of sentences used in the experiments done by Nadeem et al. (2020). The batch size is given as four which was the maximum that could be given not to exceed the memory limit of the machine. However, due to not being able to supply many examples in one batch, the accumulation steps are chosen to be large as 16. Thus, the weights are updated in every 64 examples. The core learning rate used in the learning process for the core model is given as 5e-6, and the learning rate for the Next Sentence Prediction head is given as 1e-3 because these rates are the ones that Nadeem et al. (2020) used in their training by default. The model is shortly trained with both half-precision and single-precision

floating points separately, and it is discerned that it achieved better results in a shorter time with half-precision floating points. Therefore, the training is carried out with a half-precision (FP16) floating point. In addition, there are 4 CPU cores provided to the program, but that is related to creating the data set instead of the training process.
The accuracy started from 50% as a random prediction and jumped to 90% only with 10% of the data set, which is around 1 million examples. The loss function decreased from 0.7 to around 0.2 only with 10 percent of the data set too. The graph for the accuracy and loss functions is given in Figure 4.1. Since these results are already sufficiently satisfactory, the training is stopped at that point for environmental and financial considerations.

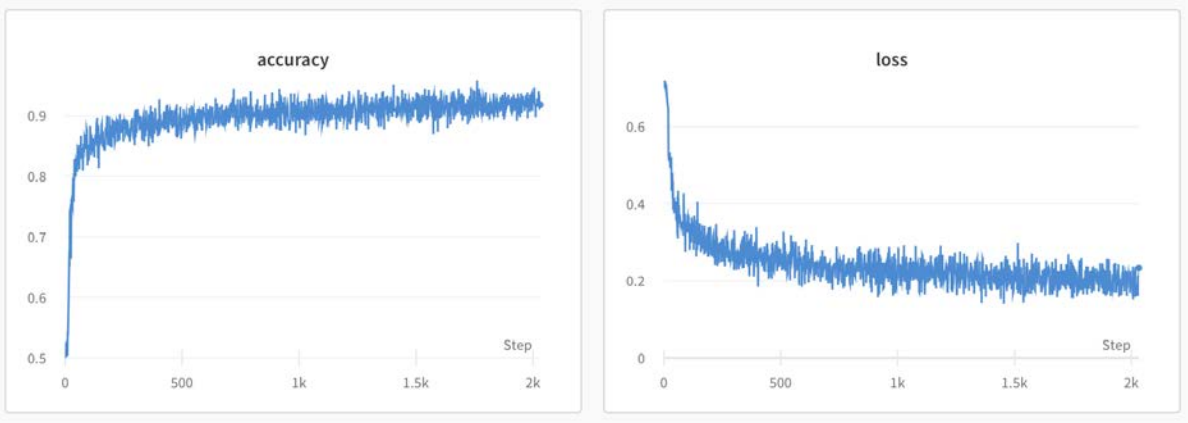

Figure 4.1.: Graphs for the accuracy and loss function for multilingual GPT-2 training.

Furthermore, the loss and accuracy functions still don't look fully converged, and they still have room to grow, as is seen from the graphs in Figure 4.2 where their original graphs' are smoothed with a smoothing parameter of 0.92. The smoothing for this graph and the rest of the paper is done with the "Exponential Moving Average" smoothing option of the Weights & Biases framework (Biewald, 2020). The fact that the convergence is not finished can also be noticed from Graph 4.3 where the change in the learning rates for the core and head models is provided. Only 10% of the entire training process takes 7 hours 40 minutes in the specified configurations on Tesla V100-SXM2-16GB; more details for the GPU usage and system information are given in Appendix A.2.1. A longer or fully completed training is left as future work.
Since the scheduler uses the warm-up, the learning rates first increase and then decrease after reaching the specified hyper-parameter values, which are 5e-6 and 1e-3. It is obvious that the warm-up steps are completed as seen in the learning rate scheduler Graph 4.3, but they do not fully converge to 0 in this work as it is left as future work.

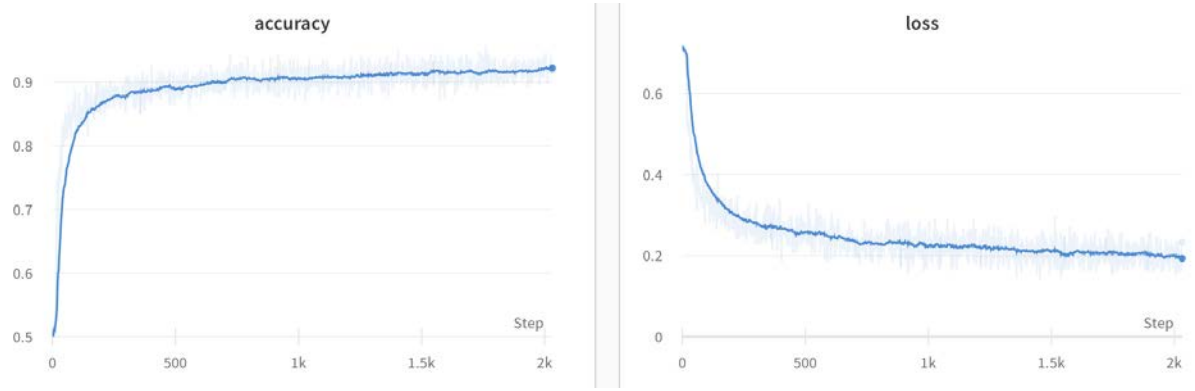

Figure 4.2.: Smoothed graphs for the accuracy and loss function for multilingual GPT-2 training.

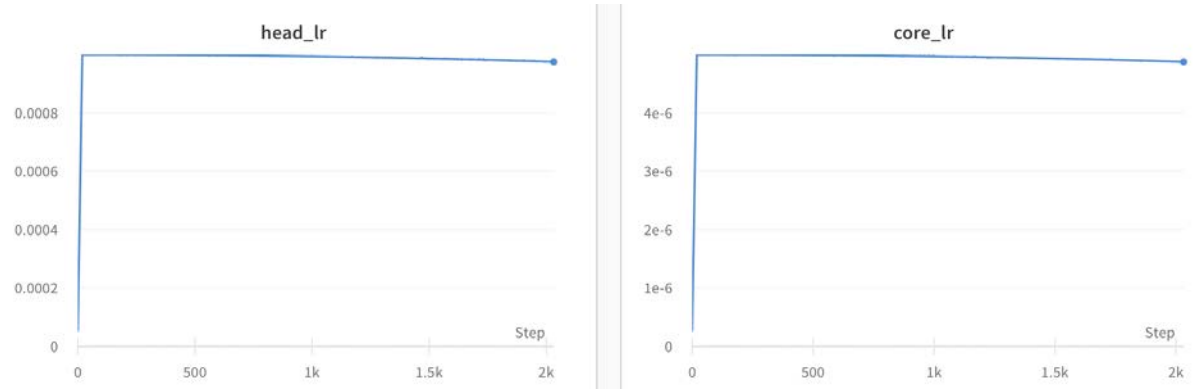

Figure 4.3.: Learning rate scheduler graph in multilingual GPT-2 training.

Moreover, the model is saved every 5% of the data set since, initially, this model is expected to grow with a small amount of data. After 10% of the data set, the saved model is tested with a new 300 thousand examples, 30% of the training data set, consisting of English and German, and the model attains 91% test accuracy.

4.1.2. T5

As stated in Section 3.2.2, the base model of T5 is used because it has comparability with the BERT model. As mentioned earlier, the multilingual GPT-2 model is trained only with 10% of the entire data set, which is the sentences from 110 thousand articles per language. Therefore, it is indeed trained with 22 thousand articles, 11 thousand in English and the rest in German. Since the T5 model is mono-lingual, this work aims to train it with an English data set to perform Next Sentence Prediction in English to observe similar behavior in mono-lingual BERT. Thus, the sentences from 22 thousand

English articles are fed to the model in this training process which constitutes around 1 million examples, instead of providing complete data and abandoning the training process at half. Stopping in the middle strategy is not chosen since the learning rate would never converge, and the learning rate can be too high to find the optimal points; it might even not pass the warm-up steps. Therefore, one should be careful about how much data is fed at the beginning of the training. The original tokenizer of the T5 model is utilized in this process. T5 models and tokenizers are much smaller than GPT-2 models and tokenizers as they have been compared in Table 2.3; consequently, more data can fit into the GPU in each training step with batched training. In this fine-tuning, batch size is used as 24, the maximum number of examples that could fit into the machine in one step. The step for gradient accumulation is chosen as 3; it is lower than the mGPT-2 training process because of utilizing larger batches. As a result, the weight updating is accomplished in every 72 examples, and it is 64 in the previously mentioned training process, which is comparable. The training is conducted shortly with a half-precision floating point (FP16) and without a half-precision floating point, and the two results are compared. Contrary to the expectations, the training takes longer with half-precision, and this is mentioned by other researchers in some forums too (Platen, 2022), thereby conducting the primary training without a half-precision floating point. Moreover, 8 CPU cores are provided for the data set to be created with multi-processes. The accuracy begins at 50% and achieves around 90% at the end of the full-training process. The latest saved model has also experimented with test data consisting of 70 thousand unseen English articles, and the model performs 89% test accuracy. The loss function converged from 10 to 0 during the fine-tuning, as seen in Figure 4.4. However, the loss between 0 and 1 can not be analyzed well from the graph because the loss decreases sharply from 10 to 1 only in a few steps and stays between 0 and 1 for many steps. The graph 4.5 is the zoomed version of the original loss function to investigate the behavior of the loss function between 0 and 1.

The accuracy does not seem to converge fully, and apparently, it can even be enriched by supplying more data, although the loss function looks converged enough to zero. These can be seen from the smoothed graphs in Figure 4.6, which are smoothed with the 0.32 smoothing parameter of Weights and Biases. As seen from the smoothed accuracy graph, one might even grow the current outcome by providing more data, which is left as future work. The complete data set is processed in 5 hours with the machine Tesla V100-SXM2-16GB where the utilization details of the GPU machine are provided in Appendix A.2.2.

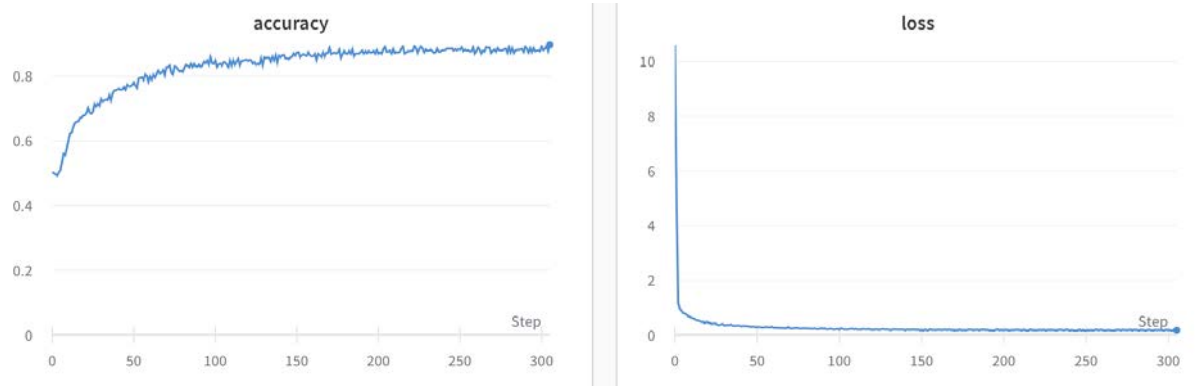

Figure 4.4.: Graphs for the accuracy and loss function for T5 Next Sentence Prediction Fine-tuning.

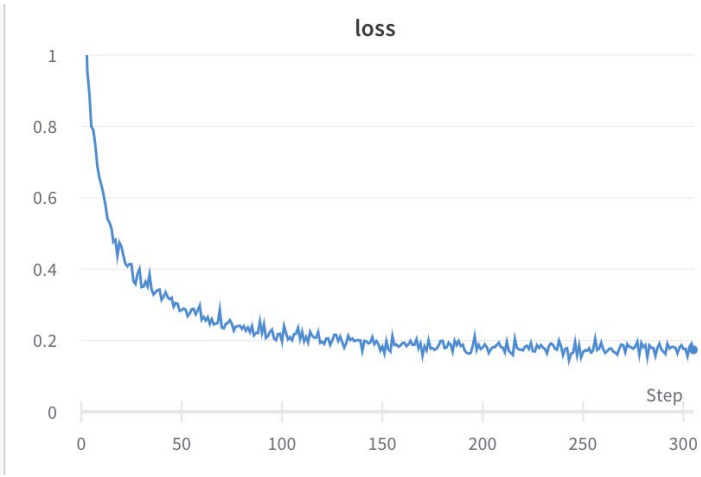

Figure 4.5.: Zoomed graph for the loss function for T5 Next Sentence Prediction fine-tuning.

Contrary to the mGPT-2 training process, the learning rate scheduler ultimately finished its convergence since the model is trained with the total amount of data that is initially given instead of being stopped in the middle. Since there is no separate head for the model in T5 fine-tuning, as is explained in Section 3.2.2, the learning rate is only for the core model, as is seen from the Graph 4.7. The learning rate first commences from smaller values than the configured hyper-parameter to warm the training process up. Then it rises until it reaches the configured hyper-parameter 5e-6. After reaching the user-specified value, the scheduler gradually diminishes the learning rate to zero.

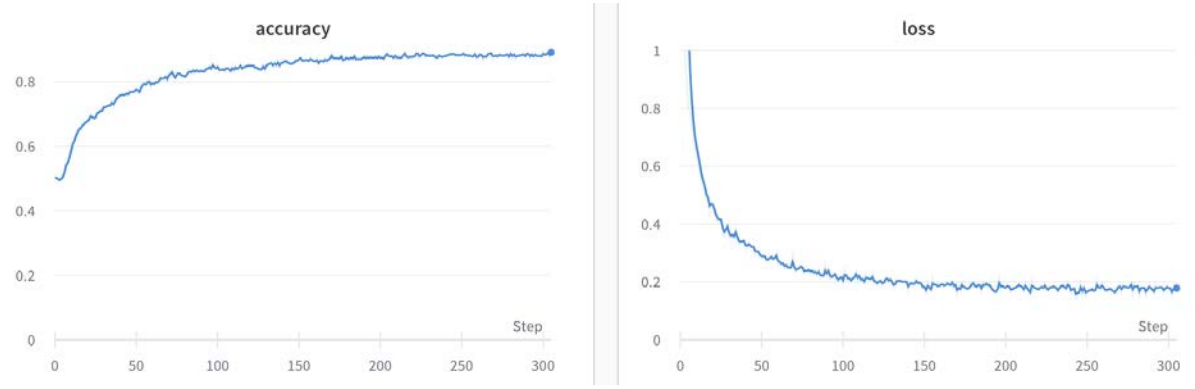

Figure 4.6.: Smoothed graphs for the accuracy and loss functions for T5 Next Sentence Prediction fine-tuning.

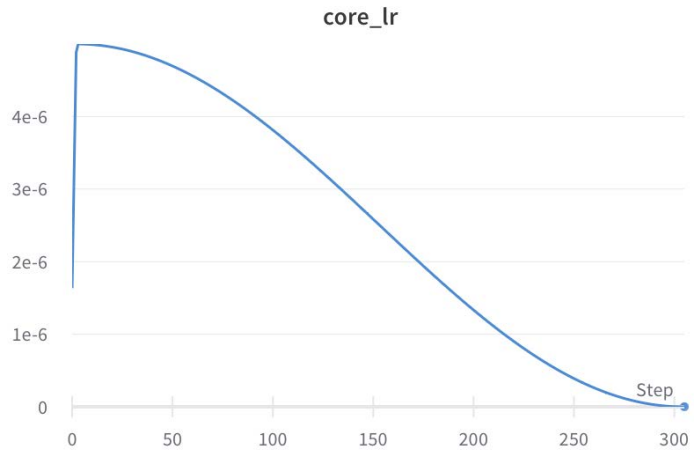

Figure 4.7.: Learning rate scheduler graph for T5 Next Sentence Prediction fine-tuning.

4.1.3. Multilingual T5

The base form of the multilingual T5 model from Xue et al. (2020) is employed since it has an officially published model in the Huggingface platform and is comparable with BERT and T5 models used in the previous experiments. The multilingual T5 is not pre-trained with downstream tasks, and it is expected to learn slower than the mono-lingual T5 model; therefore, a double data set is initially provided to the model by being able to stop the process in the middle. The training process of T5 is executed with 22 thousand English articles from Wikipedia; for multilingual T5, there are 22 thousand English and 22 thousand German articles fed. The maximum number of examples in a batch that could fit into the machine is 8; thus, the batch size is given as 8. To have comparable weight updates with the previously mentioned experiments, the step size for

the gradient accumulation is inputted as 8, too. Hence, the weight updates are done in every 64 examples, which is identical to multilingual GPT-2 training and slightly lower than the T5 training. Adequate experiments are done using half-precision floating point format; however, it is not preferred in the final training since it does not add any value to the process and even decelerates the process. It was an expected behavior from the multilingual T5 model because the same situation has been observed in the T5 model, which has an identical architecture. Moreover, 8 CPUs are used to work parallel, but they are effective in the data set creation process instead of the training process.
The accuracy in the multilingual T5 training starts around 52%, then it always volatiles around 52% and never exceeds even 60% for 4 hours in the Tesla V100-SXM2-16GB machine as it is seen from the Graph 4.8. In 4 hours, the model processed half of the data set, which mono-lingual T5 would already achieve more than 90% accuracy. Since the model does not improve the outcome even after providing 1 million examples and running the GPU machine for 4 hours, the training process is halted in the middle due to not cost any unnecessary environmental and financial damage. Nevertheless, the loss function decreased substantially from 22 to 0.44, so there is still a chance for the model to increase its accuracy by conducting a much longer training experiment. Additionally, the continuing decrease in the loss function can be observed from Graph 4.9 where the graph of the loss function is zoomed to track its movement between the range 0 and 2. The graphs given in Figure 4.10 are the smoothed version of the graphs in Figure 4.8 with the smoothing parameter 0.8. From these graphs, it is more observable that the accuracy does not have any considerable increase and the loss function still has room to decrease. However, many complaints exist about the multilingual T5 model's difficulties in learning new tasks in various online forums (Jsrozner et al., 2020; Mohd-Ali-Ansari, 2020; SarraCode, 2020; tomhosking, 2020). A significant reason for its slow learning characteristics might be the fact that it is not pre-trained with any downstream task contrary to the T5 model because the model has an identical architecture with its mono-lingual conjugate, and their main difference is downstream tasks in the pre-training phase. Furthermore, different configurations are tried since the learning cannot be achieved with the aforementioned hyper-parameter configurations. For instance, "Adafactor", Adaptive Learning Rates with Sublinear Memory Cost from Shazeer and Stern (2018), optimizer is employed, instead of Adam algorithm with weight decay. Moreover, we also tried larger and fixed learning rates such as 1e-3, but there was not any considerable increase in the accuracy. Thus, Next Sentence Prediction could not be performed with multilingual T5 in this work as its limitation, so the improvement of

this training process to evaluate multilingual T5's bias using Next Sentence Prediction methodology is left for researchers as future work.

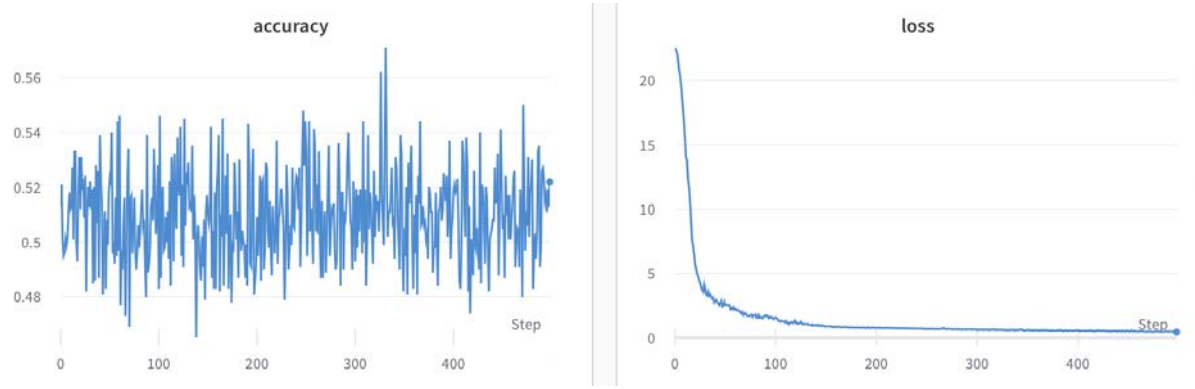

Figure 4.8.: Graphs for the accuracy and loss function for Multilingual T5 Next Sentence Prediction Fine-tuning.

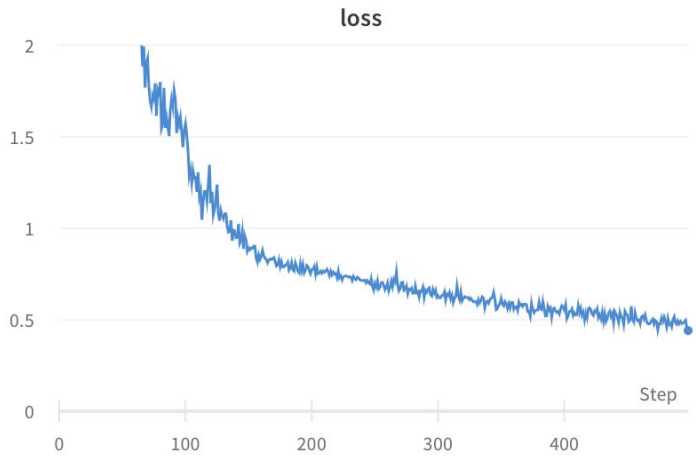

Figure 4.9.: Zoomed graph for the loss function for Multilingual T5 Next Sentence Prediction fine-tuning.

Because the training process is abandoned at half, the scheduler obviously could not converge the learning rate to zero, as is seen from Graph 4.11. Nonetheless, the training is conducted with sufficient steps after the warm-up stage, where its details are delivered in Appendix A.2.3.

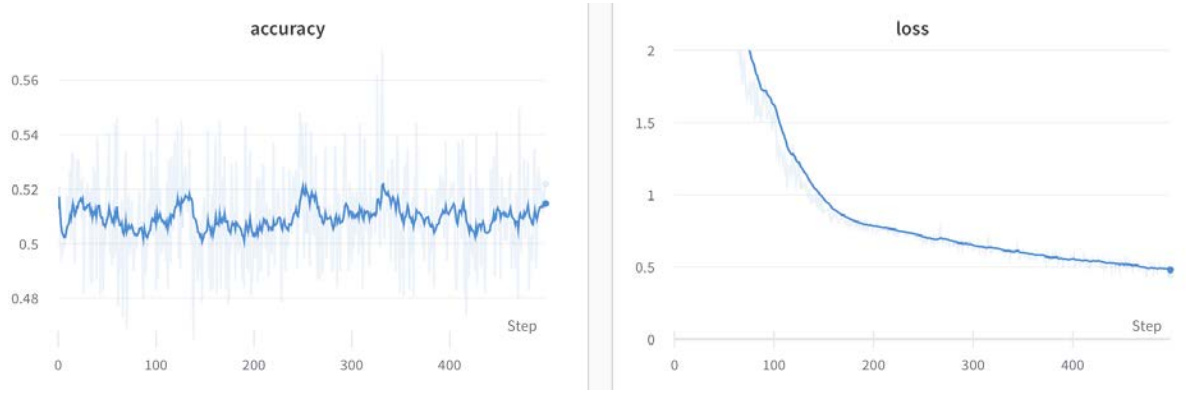

Figure 4.10.: Smoothed graphs for the accuracy and loss functions for Multilingual T5 Next Sentence Prediction fine-tuning.

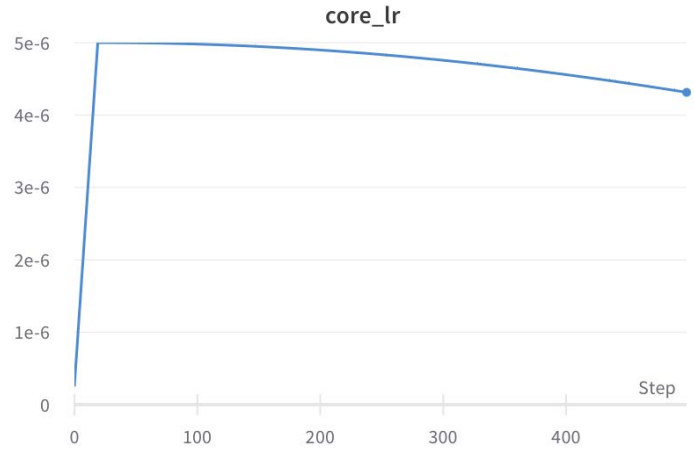

Figure 4.11.: Learning rate scheduler graph for Multilingual T5 Next Sentence Prediction fine-tuning.

4.2. Evaluations

We report the evaluations about BERT's base model (Devlin et al., 2018), multilingual BERT's base model, T5's base model (Raffel et al., 2019), multilingual T5's base model (Xue et al., 2020), GPT-2 model and multilingual GPT-2 from Tan et al. (2021) in this section. As stated in Section 3.3, the evaluations are made in two different techniques; first, we evaluate all examples together, and second, we consider each target term as a class and treat the problem as a multi-class classification case. Although in the paper from Nadeem et al. (2020) only the multi-class evaluations are considered and reported, we give more importance to the general evaluation but still report the multi-class evaluations as a separate section to obtain comparability of the results with Nadeem et al.

(2020).

Table 4.1 provides the evaluation results of all models in the intra-sentence tests. As stated in Section 3.1.4, three examples are removed from the German data set since their mistakes resulting from the translations could not be saved. These "Count" numbers were published wrongly in work from Nadeem et al. (2020) because of a mistake in its corresponding published code that is mentioned in Section 3.3.

The table shows that the mono-lingual GPT-2 model has a better language modeling performance than its comparable BERT and T5 models in the intra-sentence tests and yet it also has more stereotypical bias than the others. Nevertheless, the model performs more acceptable than the others in the ICAT score, which is a combination of language modeling and stereotype scores. The general understanding that having an advanced language model would have more stereotypes is not valid as it would be dis-proven by comparing BERT and T5 models. BERT has both a higher language modeling score and fewer stereotypes than the T5 model, thereby having a higher ICAT score.

The multilingual GPT-2 performs more promisingly than comparable multilingual models in both languages regarding language modeling and ICAT score. Multilingual GPT-2 has a 16% higher language modeling score than mBERT and mT5 in English, and it has a negligible amount (2.5%) of more stereotypes than the others, thereby having a much higher (10%) ICAT score than the others. mBERT and mT5 have almost identical performances in English; all their scores are incredibly similar in the intra-sentence tests. In German, the multilingual GPT-2 model has both the highest language modeling score and the least stereotype in the intra-sentence tests; hence, it performs hugely better than the others, as is seen from its superiority in the ICAT score. Although multilingual T5 and BERT perform remarkably similarly in English intra-sentence tests, mT5 slightly outperforms mBERT in German. The multilingual BERT's stereotype score in the German language is 49.17%, which is less than 50%. In other words, the model behaves anti-stereotypical in German mask-filling tasks. The multilingual GPT-2 and BERT have almost 50% stereotype scores in German intra-sentence tests, which is the ideal case. Thus, it can be stated that they have almost no stereotypes in German mask-filling tasks.

Table 4.2 provides inter-sentence evaluation results for all models. As it is described in Section 3.2.2, there are two different approaches to evaluating GPT-2 and T5 models; one is evaluating them after fine-tuning for the Next Sentence Prediction task, and the second is evaluating them with a generative approach described in Section 3.2.2. The evaluations based on the former approach are written with "NSP", and the latter are

Model Name	Count	LMS	SS	ICAT Score
BERT	2106	83,1	58,74	68,58
GPT-2	2106	91,14	61,97	69,33
T5	2106	79,08	60,02	63,24
mBERT English	2106	69,94	52,37	66,62
mBERT German	2103	65,67	49,17	64,58
mGPT-2 English	2106	86,49	55,08	77,7
mGPT-2 German	2103	77,03	50,21	76,7
mT5 English	2106	69,87	52,52	66,35
mT5 German	2103	73,97	54,3	67,6

Table 4.1.: Evaluation results for intra-sentence tests

written with "Generative". Generative approaches can have multiple cases, which can be seen from Table 3.4. For the sake of simplicity, only case d will be discussed in this section since it has a reasonable mathematical background and a similar approach to T5 generative evaluations.

As seen from the table, BERT and fine-tuned T5 models have higher language modeling scores than fine-tuned GPT-2 models for inter-sentence tests; however, the fact that they are significantly more biased than GPT-2 model lowers their corresponding ICAT Score. Hence, the fine-tuned GPT-2 model (Nadeem et al., 2020) outperforms BERT and fine-tuned T5 models in inter-sentence tests with the NSP approach, and BERT and fine-tuned T5 models have almost identical outcomes. In fact, GPT -2's evaluation result from the generative approach outperforms all other models in inter-sentence tests according to its ICAT Score. Therefore, GPT-2 model actually does not even need to be fine-tuned for the NSP downstream task, contrary to the paper from Nadeem et al. (2020); it can achieve even higher scores with its generative structure. On the other hand, T5's generative approach is not satisfactory for performing a Next Sentence Prediction task since it has an extremely low language modeling (54.78%) and ICAT (50.37%) score.

Fine-tuned multilingual GPT-2 (Tan et al., 2021) outperforms the multilingual BERT model both in English and German languages in the inter-sentence tests. Despite having a higher language modeling score than multilingual BERT in the German language, it is fairer than BERT, resulting in its superiority in the ICAT Score. Since multilingual T5 could not be fine-tuned, as is explained in Section 3.2.2, the results for multilingual T5 for the NSP inter-sentence evaluations are not published. The generative approach in multilingual GPT-2 is not as satisfying as in the mono-lingual GPT-2; it has a significantly

Model Name	LMS	SS	ICAT Score
BERT NSP	88,41	60,24	70,3
GPT-2 NSP	76,17	51,91	73,26
T5 NSP	88,48	60,39	70,1
GPT-2 Generative d	76,57	52	73,5
GPT-2 Generative c	76,9	50,92	75,48
GPT-2 Generative e	85,49	60,67	67,25
GPT-2 Generative Orig	58,27	46,21	53,85
T5 Generative	54,78	54,03	50,37
mBERT NSP English	82,9	57,94	69,74
mBERT NSP German	77,27	58,03	64,86
mGPT-2 NSP English	81,56	54,26	74,61
mGPT-2 NSP German	77,39	53,6	71,81
mGPT-2 Generative d English	69,78	45,6	63,64
mGPT-2 Generative d German	67,57	43,48	58,75
mGPT-2 Generative c English	78,71	45,83	72,15
mGPT-2 Generative c German	74,63	43,81	65,39
mGPT-2 Generative e English	52,54	52,71	49,7
mGPT-2 Generative e German	45,71	54,78	41,34
mGPT-2 Generative Orig English	58,64	43,15	50,61
mGPT-2 Generative Orig German	62,22	42,3	52,64
mT5 Generative English	31,56	52,85	29,76
mT5 Generative German	32,6	53,93	30,03

Table 4.2.: Evaluation results for inter-sentence tests

lower language modeling score compared to the multilingual BERT model. Interestingly, the mGPT-2 model is anti-stereotypical in the generative approach based inter-sentence tests. Multilingual T5 has incredibly poor performance with the generative approach, like its mono-lingual version.

The overall results calculated from the combination of both intra-sentence and inter-sentence examples are presented in Table 4.3. The abbreviations in the "Model Name" column have the same meaning previously mentioned in the inter-sentence test evaluations; the NSP and Generative words hint at which approach is used in the inter-sentence evaluations. For the GPT-2 generative approaches, only case d shows a simple and more understandable display.

In the final results, GPT-2 (both with NSP and the generative approach) demonstrates the best performance in terms of the ICAT Score among the mono-lingual models and has the most negligible stereotypical bias. Then, BERT follows GPT-2 as it has the highest language modeling score and the second highest ICAT score among mono-lingual models.

Model Name	Count	LMS	SS	ICAT Score
BERT NSP	4229	85,76	59,49	69,48
GPT-2 NSP	4229	83,62	56,92	72,06
T5 NSP	4226	83,8	60,18	66,75
GPT-2 Generative d	4229	83,83	56,96	72,15
T5 Generative	4229	66,88	57,01	57,5
mBERT NSP English	4229	76,45	55,17	68,55
mBERT NSP German	4226	71,5	53,62	66,32
mGPT-2 NSP English	4229	84,02	54,67	76,17
mGPT-2 NSP German	4226	77,21	51,92	74,25
mGPT-2 Generative d English	4229	78,1	50,32	77,6
mGPT-2 Generative d German	4226	72,28	46,83	67,7
mT5 Generative English	4229	50,64	52,68	47,92
mT5 Generative German	4226	53,18	54,12	48,8

Table 4.3.: Overall evaluation results

In the multilingual models, the superiority of the mGPT-2 model in both types (NSP and generative) still holds for both languages. Multilingual GPT-2 scores are higher in both language modeling and stereotypical scores, so obviously also in the ICAT scores. As it is observed that the multilingual T5 model performs exceptionally poorly in inter-sentence tests, its overall scores remain behind mBERT models. Thus, GPT-2 models, both the mono-lingual and multilingual versions, significantly outperform BERT and T5 models in English and German by being fairer and attaining higher ICAT Scores. Likewise, BERT models beat T5 models on both the NSP and generative approaches in English and German.

4.2.1. Multi-Class Evaluations

Assuming the target terms to be a separate class results in a multi-class classification problem, where the approach is inspired by Nadeem et al. (2020), and it is kept in this work to compare the results. Most of the interpretations of the results in this evaluation type are almost identical to those mentioned above, so only the differences will be mentioned in this section to avoid repetition. The multi-class description of the problem has two separate ICAT Scores, Macro ICAT, and Micro ICAT, defined in Section 3.3. The results for the intra-sentence tests can be found in Table 4.4, for the inter-sentence tests in Table 4.5 and for the overall results in Table 4.6.
Although the interpretations stated for general evaluation in the intra-sentences hold for multi-class intra-sentence evaluations, there are some differences in the inter-sentence

Model Name	LMS	SS	Macro ICAT Score	Micro ICAT Score
BERT	83,02	58,63	64,57	68,69
GPT-2	91,11	61,93	66,69	69,37
T5	79,04	59,98	60,03	63,26
mBERT English	69,94	52,36	56,58	66,64
mBERT German	65,64	49,13	53,59	64,5
mGPT-2 English	86,52	55,18	69,18	77,56
mGPT-2 German	77,12	50,13	65,06	76,91
mT5 English	69,97	52,56	55,69	66,39
mT5 German	73,99	54,19	59,63	67,8

Table 4.4.: Multi-class evaluation results for intra-sentence tests

tests. For instance, the fine-tuned mono-lingual GPT-2 model outperforms BERT and fine-tuned T5 models in Micro ICAT as it does in the general evaluations, but not in Macro ICAT. These differences between Macro and Micro ICAT occur when the ICAT score between different classes varies enormously. The overall outcomes from the multi-class evaluations are profoundly similar to the general evaluations, and the interpretations expressed in that section still hold for multi-class evaluations.

Model Name	LMS	SS	Macro ICAT Score	Micro ICAT Score
BERT NSP	88,53	60,43	67,13	70,06
GPT-2 NSP	76,26	52,28	64,11	72,79
T5 NSP	88,59	60,71	67,36	69,61
GPT-2 Generative d	76,37	52,17	65,46	73,06
GPT-2 Generative c	76,77	51,12	65,21	75,05
GPT-2 Generative e	85,55	60,72	65,15	67,21
GPT-2 Generative Orig	58,09	46,39	47,94	53,89
T5 Generative	54,79	54,14	46,28	50,25
mBERT NSP English	83,06	58,1	65	69,61
mBERT NSP German	77,4	57,99	59,95	65,02
mGPT-2 NSP English	81,7	54,51	65,73	74,34
mGPT-2 NSP German	77,42	53,76	64,15	71,59
mGPT-2 Generative d English	69,83	45,92	58,17	64,12
mGPT-2 Generative d German	67,43	43,64	54,34	58,85
mGPT-2 Generative c English	78,81	46,12	65,66	72,69
mGPT-2 Generative c German	74,58	44,05	59,09	65,71
mGPT-2 Generative e English	52,44	52,9	43,62	49,4
mGPT-2 Generative e German	45,46	54,73	37,46	41,16
mGPT-2 Generative Orig English	58,45	43,28	46,82	50,6
mGPT-2 Generative Orig German	62,19	42,45	48,88	52,8
mT5 Generative English	31,57	52,8	26,27	29,81
mT5 Generative German	32,59	53,88	26,76	30,06

Table 4.5.: Multi-class evaluation results for inter-sentence tests

Model Name	LMS	SS	Macro ICAT Score	Micro ICAT Score
BERT NSP	85,77	59,53	68,17	69,42
GPT-2 NSP	83,65	57,03	69,28	71,89
T5 NSP	83,8	60,28	65,59	66,57
GPT-2 Generative d	83,76	57	70,22	72,03
T5 Generative	66,88	57,03	55,8	57,48
mBERT NSP English	76,52	55,19	64,64	68,58
mBERT NSP German	71,53	53,63	61,71	66,33
mGPT-2 NSP English	84,06	54,81	71,47	75,97
mGPT-2 NSP German	77,25	51,98	67,65	74,19
mGPT-2 Generative d English	78,12	50,43	68,49	77,44
mGPT-2 Generative d German	72,25	46,85	62,27	67,71
mT5 Generative English	50,68	52,61	43,13	48,04
mT5 Generative German	53,22	54,06	45,34	48,89

Table 4.6.: Overall multi-class evaluation results

5. Discussion

One of the most significant implications of this work is that the stereotype score of all the models in English and German languages are incredibly close to each other, and they have a maximum 5% difference. Overall, multilingual BERT is fairer in German, and multilingual GPT-2 and multilingual T5 are fairer in English. It is crucial to note that the original StereoSet data set is created in English and by people from the USA (Nadeem et al., 2020), in consequence, the stereotypes are expected to be from the perspective of Americans. Despite the stereotypes in the data set being the stereotypes from the American perspective, sometimes there are more stereotypes measured in the German language. This strongly confirms the assumption that English and German languages would have similar stereotypes, and it might be because of similar norms and values shared in the Western countries, the USA and Germany.

Another interesting observation in this thesis is that the multilingual T5 model always has a surprisingly higher language modeling score and ICAT score in German than in English. Normally, a better language modeling performance in English is expected since the data sets that the models pre-trained on usually have more English data than German data. Nevertheless, in most cases, the other models than multilingual T5 perform better in English than German. Furthermore, the similar results in English and German and sometimes even better results in German also prove that the created German data set is adequately valid.

In terms of the model comparisons, as explained in detail in Section 4.2, GPT-2 models have higher ICAT scores than the comparable BERT and T5 models in both intra-sentence and inter-sentence tests and both languages. Furthermore, although GPT-2 models are mostly fairer than the others, they sometimes have higher language modeling scores, especially in the intra-sentence tests. With this and many other examples, this thesis also proves that a model can both have satisfactory language modeling ability and be a fair model. Another crucial point is the significance of fairness in a model. For instance, mono-lingual BERT has a remarkably higher language modeling score than GPT-2 in inter-sentence tests, but its ICAT Score is lower at the end. Therefore,

an outstanding language modeling ability is not enough to be a valuable model for a pre-trained language model; an adequate amount of fairness is also essential.

Moreover, according to the outcomes from the fine-tuned models, the fine-tuning is performed successfully and effectively since some results are even higher than the corresponding BERT models, although BERT is indeed pre-trained with the NSP objective. Furthermore, the data set used in the fine-tuning process is incredibly smaller than the data set used by (Nadeem et al., 2020), so it is proven that similar results could be gained with a much smaller amount of environmental and financial burden. In contrast, multilingual T5 training could not be completed and left as future work. We halted the multilingual T5's training early due to not obtaining sufficient advancement in the training process. We are aware of the environmental cost of machine learning applications and endeavored to obtain the greatest possible results with the lowest amount of carbon emissions. The experiments in this thesis reached a cumulative 16 hours of computation performed on the hardware of type Tesla V100-SXM2-16GB (TDP of 250W), and total emissions are estimated to be 2.44 kgCO_2eq, which is equal to 9.86 km driven by an average ICE car. Estimations were conducted using the MachineLearning Impact calculator presented in Lacoste et al. (2019).

6. Conclusion & Outlook

The main goal of this thesis was to evaluate the potential stereotypical bias containment in the multi-lingual models using German and English data sets. We approached this problem by benefiting the English data set prepared by Nadeem et al. (2020), which evaluated the models with mask-filling and next sentence prediction tasks and translated the data set to German using AWS Translation Services. Although the quality of the translations was observed to be adequately accurate, we corrected some identified issues as explained in Section 3.1.4. Nonetheless, there were also some observed issues in the original English data set and still minor issues remaining in the German translations. After procuring the data sets, they were supplied to each particularly selected model as described in Section 3.2 to ascertain whether the model would choose a stereotypical, anti-stereotypical, or unrelated option. The models were explicitly selected to have different architectures (e.g., encoder, decoder, encoder-decoder) in order to obtain this work's applicability in various pre-trained language models for future usage. However, this brought some difficulties while inputting the data set. Nonetheless, the support for whole selected models was implemented by aiding different inference methodologies (e.g., discriminative, generative, Seq2Seq specific ways). Since some models were not adapted for the next sentence prediction tasks, they were fine-tuned with this task by collecting the data from Wikipedia dumps as explained with visualizations in Section 3.2.2. The choices of the models were evaluated with different evaluation metrics and strategies to identify each model's overall potential stereotypical bias containment. Finally, the outcomes of these evaluations and the fine-tuning experiments are interpreted in Chapter 4.

Bibliography

Ankit, U. (2020). Transformer neural network: Step-by-step breakdown of the beast.

Artetxe, M. and Schwenk, H. (2019). Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization.

Bajaj, A. (2022).

Bartl, M., Nissim, M., and Gatt, A. (2020). Unmasking contextual stereotypes: Measuring and mitigating bert's gender bias.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. 3(null):1137–1155.

Biewald, L. (2020). Experiment tracking with weights and biases. Software available from wandb.com.

Bolukbasi, T., Chang, K.-W., Zou, J., Saligrama, V., and Kalai, A. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings.

Bryan, D. (2019). Are google search results biased?

Caliskan, A., Bryson, J. J., and Narayanan, A. (2017). Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Cardwell, M. and Marcouse, I. (1996). *Dictionary of psychology*. Routledge.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Doddapaneni, S., Ramesh, G., Khapra, M. M., Kunchukuttan, A., and Kumar, P. (2021). A primer on pretrained multilingual language models.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Feng, F., Yang, Y., Cer, D., Arivazhagan, N., and Wang, W. (2020). Language-agnostic bert sentence embedding.

Foundation, W. (2022). Wikimedia downloads.

Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20:121–136.

Gage, P. (1994). A new algorithm for data compression. *The C Users Journal archive*, 12:23–38.

Ghallab, M. (2019). Responsible ai: requirements and challenges. *AI Perspectives*, 1(1).

Hall, M., van der Maaten, L., Gustafson, L., and Adcock, A. (2022). A systematic study of bias amplification.

Hassanin, M., Anwar, S., Radwan, I., Khan, F. S., and Mian, A. (2022). Visual attention methods in deep learning: An in-depth survey.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Johnson, M. (2020). A scalable approach to reducing gender bias in google translate.

Jsrozner, Yusukemori, Valhalla, rohankhrn56, dheeraja486, Kirill, PhillBoy, Brando, Zokica, Pierreguillou, and et al. (2020). T5 finetuning tips.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Kozodoi, N. (2021).

Lacoste, A., Luccioni, A., Schmidt, V., and Dandres, T. (2019). Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*.

Lample, G. and Conneau, A. (2019). Cross-lingual language model pretraining.

Larson, J., Mattu, S., Kirchner, L., and Angwin, J. (2016). How we analyzed the compas recidivism algorithm.

Lauscher, A. and Glavaš, G. (2019). Are we consistently biased? multidimensional analysis of biases in distributional word vectors.

Li, H. (2022). Language models: past, present, and future. *Communications of the ACM*, 65:56–63.

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

Loper, E. and Bird, S. (2002). Nltk: The natural language toolkit.

Loshchilov, I. and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization.

Louis, A. (2020). A brief history of natural language processing — part 2.

Lubana, E. S., Dick, R. P., and Tanaka, H. (2021). Beyond batchnorm: Towards a unified understanding of normalization in deep learning.

Maruyama, Y. (2022). Moral philosophy of artificial general intelligence: Agency and responsibility. In Goertzel, B., Iklé, M., and Potapov, A., editors, *Artificial General Intelligence*, pages 139–150, Cham. Springer International Publishing.

McIntosh, C. (2013). *Cambridge Advanced Learner's Dictionary*. Cambridge University Press.

Meade, N., Poole-Dayan, E., and Reddy, S. (2021). An empirical survey of the effectiveness of debiasing techniques for pre-trained language models.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2019). A survey on bias and fairness in machine learning.

Moberg, J. (2020). A deep dive into multilingual nlp models.

Mohd-Ali-Ansari (2020). Mt5-small is taking large amount of ram while preprocessing. · issue #43 · google-research/multilingual-t5.

Moore, D. and Healy, P. (2008). The trouble with overconfidence. *Psychological review*, 115:502–17.

Nadeem, M., Bethke, A., and Reddy, S. (2020). Stereoset: Measuring stereotypical bias in pretrained language models. *CoRR*, abs/2004.09456.

Nangia, N., Vania, C., Bhalerao, R., and Bowman, S. R. (2020). CrowS-pairs: A challenge dataset for measuring social biases in masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online. Association for Computational Linguistics.

Névéol, A., Dupont, Y., Bezançon, J., and Fort, K. (2022). French CrowS-pairs: Extending a challenge dataset for measuring social bias in masked language models to a language other than English. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8521–8531, Dublin, Ireland. Association for Computational Linguistics.

Obermeyer, Z., Powers, B., Vogeli, C., and Mullainathan, S. (2019). Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453.

O'Neil, C. (2016). *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown Random House.

pandas development team, T. (2020). pandas-dev/pandas: Pandas.

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

Pires, T., Schlinger, E., and Garrette, D. (2019). How multilingual is multilingual bert?

Platen, P. v. (2022). Training with fp16 precision gives nan in longt5 · issue #17978 · huggingface/transformers.

Pratt, L. Y., Mostow, J., and Kamm, C. A. (1991). Direct transfer of learned information among neural networks. In *Proceedings of the Ninth National Conference on Artificial Intelligence - Volume 2*, AAAI'91, page 584–589. AAAI Press.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer.

Rogers, A., Kovaleva, O., and Rumshisky, A. (2020a). A primer in bertology: What we know about how bert works.

Rogers, A., Kovaleva, O., and Rumshisky, A. (2020b). A primer in bertology: What we know about how bert works.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

SarraCode (2020). Can't reproduce finetuning · issue #19 · google-research/multilingual-t5.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units.

Shazeer, N. and Stern, M. (2018). Adafactor: Adaptive learning rates with sublinear memory cost. *CoRR*, abs/1804.04235.

SKEEM, J. and Lowenkamp, C. (2016). Risk, race, and recidivism: Predictive bias and disparate impact*: Risk, race, and recidivism. *Criminology*, 54.

Stanovsky, G., Smith, N. A., and Zettlemoyer, L. (2019). Evaluating gender bias in machine translation.

Tan, Z., Zhang, X., Wang, S., and Liu, Y. (2021). Msp: Multi-stage prompting for making pre-trained language models better translators.

Tieleman, T., Hinton, G., et al. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

tomhosking (2020). Generating from mt5 · issue #8704 · huggingface/transformers.

Trielli, D. and Diakopoulos, N. (2019). Search as news curator: The role of google in shaping attention to news information. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–15, New York, NY, USA. Association for Computing Machinery.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2019). Huggingface's transformers: State-of-the-art natural language processing.

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

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. (2020). mt5: A massively multilingual pre-trained text-to-text transformer.

Yang, Y. and Feng, F. (2020). Language-agnostic bert sentence embedding.

Zhang, Z. and Sabuncu, M. R. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels.

Zhao, J., Wang, T., Yatskar, M., Ordonez, V., and Chang, K.-W. (2017). Men also like shopping: Reducing gender bias amplification using corpus-level constraints.

Zhao, W., Eger, S., Bjerva, J., and Augenstein, I. (2020). Inducing language-agnostic multilingual representations.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books.

A. Appendix

A.1. Bias Example Appendix

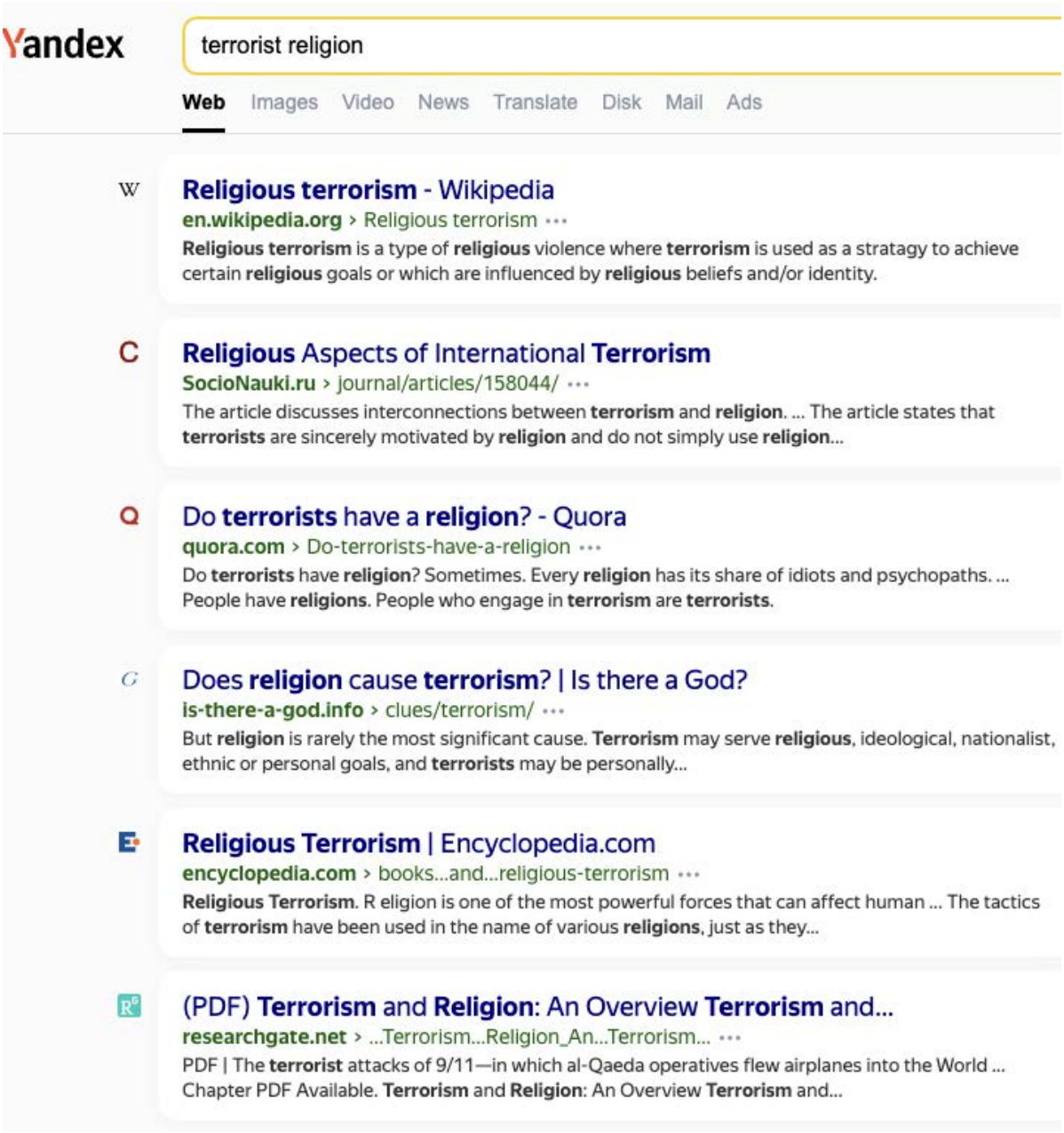

Figure A.1.: Yandex Search results for "terrorist religion".

A.2. GPU Usage Appendix

A.2.1. Multilingual GPT-2 Training

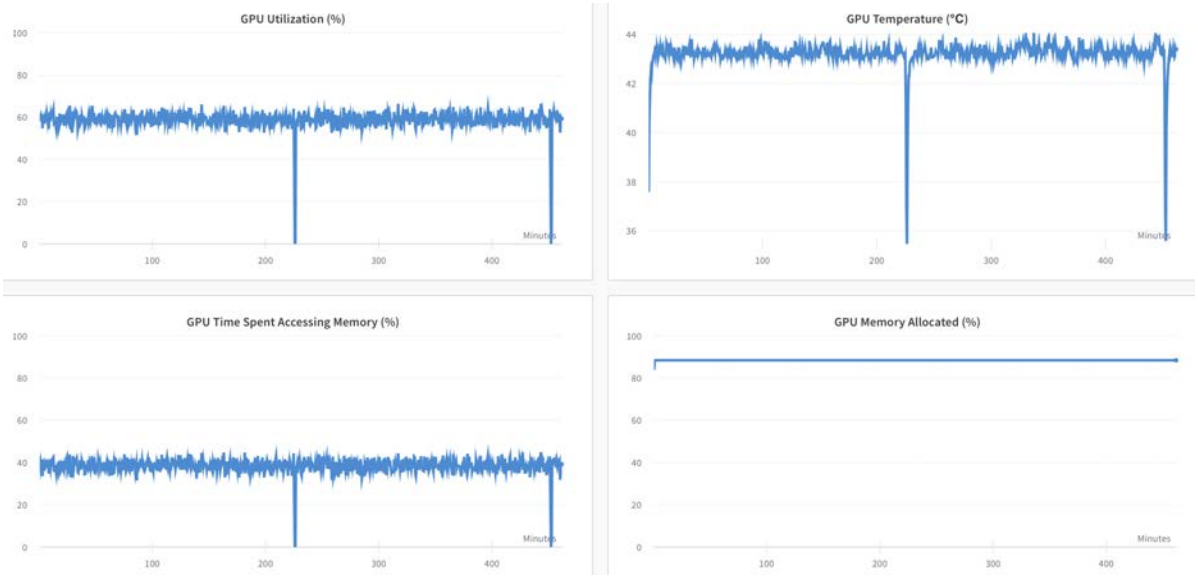

Figure A.2.: GPU usage graphs in multilingual GPT-2 training.

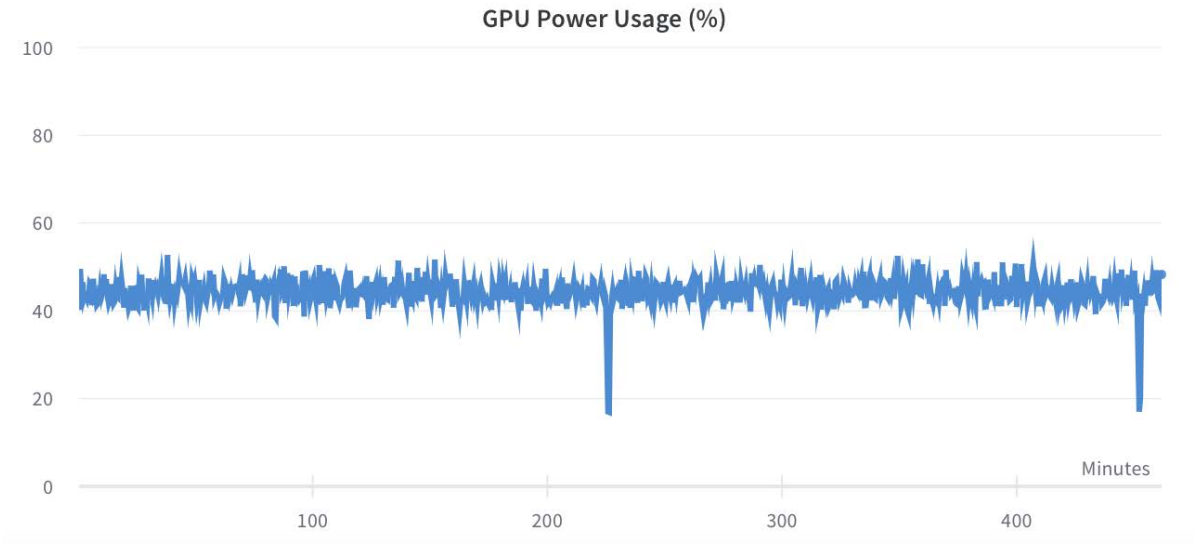

Figure A.3.: GPU power usage (%) graph in multilingual GPT-2 training.

A.2.2. T5 Training

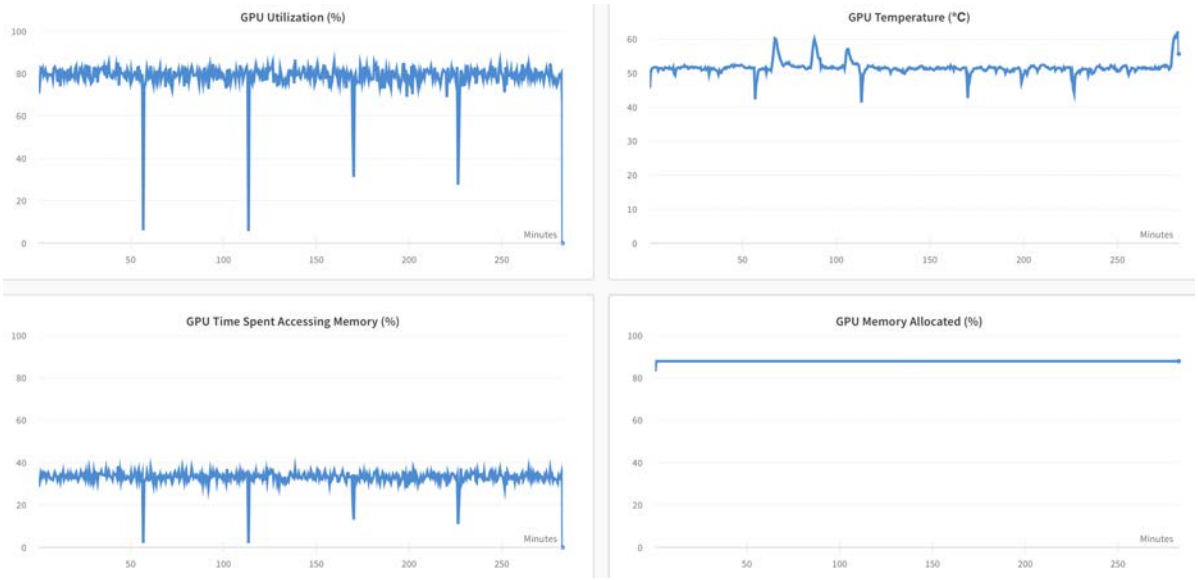

Figure A.4.: GPU usage graphs in T5 training.

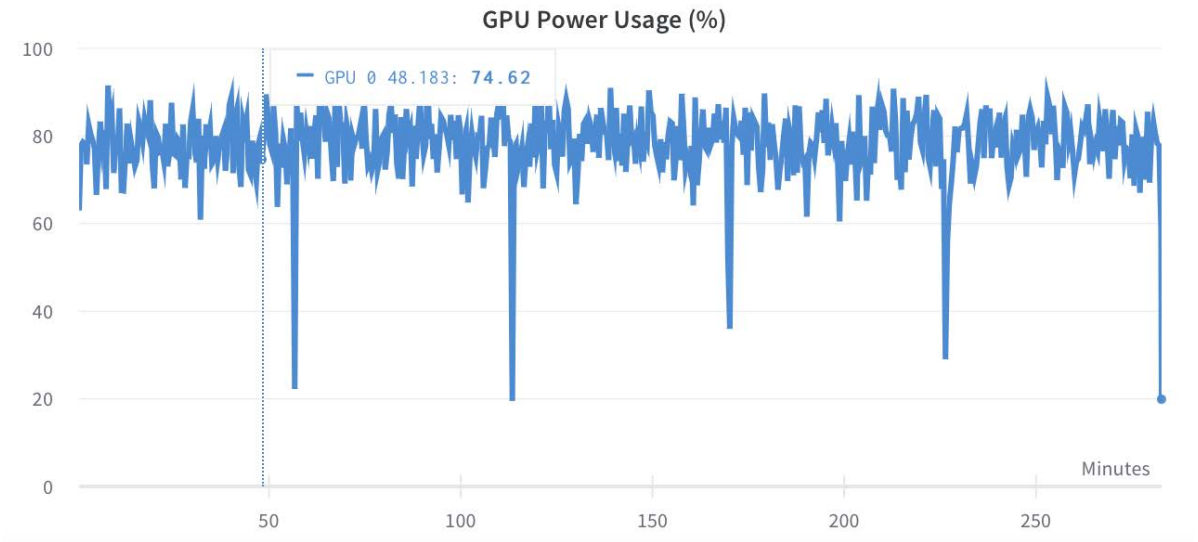

Figure A.5.: GPU power usage (%) graph in T5 training.

A.2.3. Multilingual T5 Training

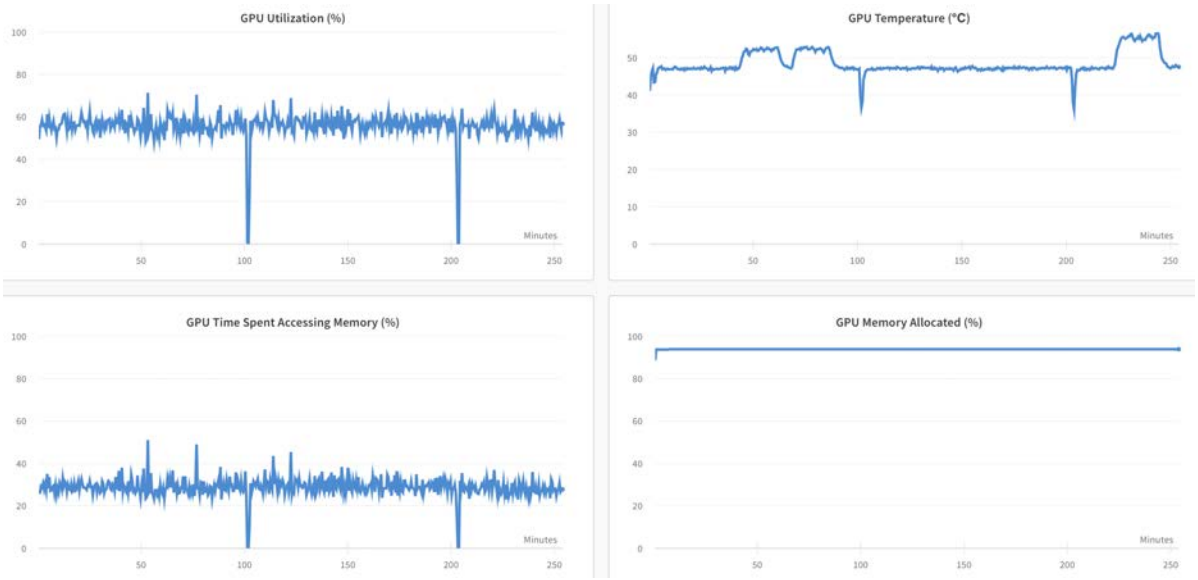

Figure A.6.: GPU usage graphs in Multilingual T5 training.

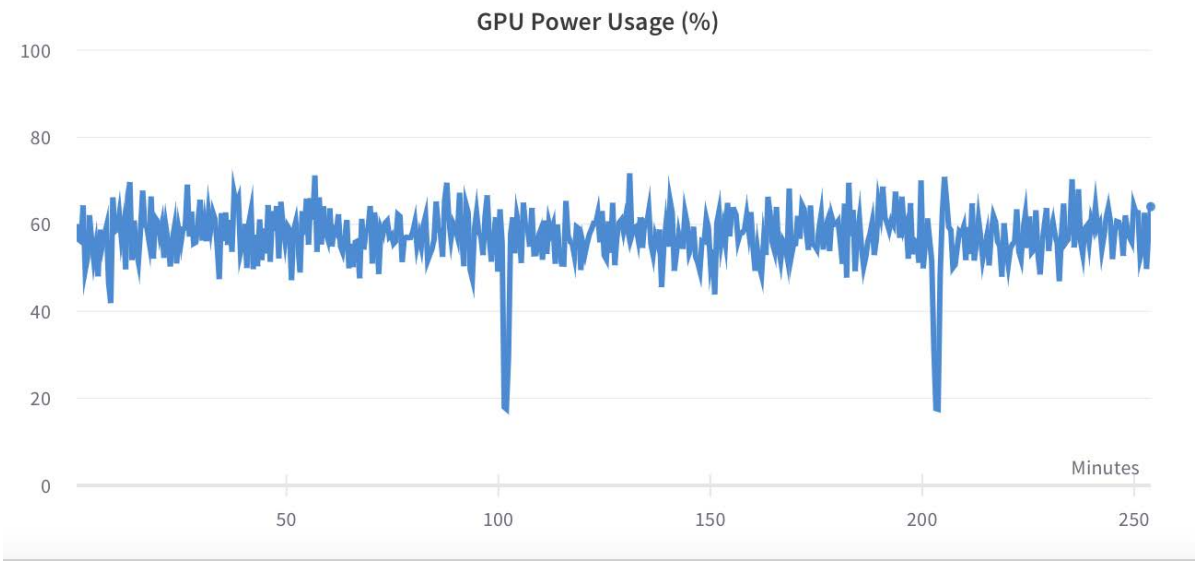

Figure A.7.: GPU power usage (%) graph in Multilingual T5 training.

Declaration of Authenticity

The work contained in this thesis is original and has not been previously submitted for examination which has led to the award of a degree.

To the best of my knowledge and belief, this thesis contains no material previously published or written by another person except where due reference is made.

Ibrahim Tolga Öztürk