Human Value Detection (Project work for NLP course)

Mohammadreza Hosseini, Parvaneh Soleimanybaraijany
Master's Degree in Artificial Intelligence, University of Bologna
{ mohammadrez.hossein2, p.soleimanybaraijny}@studio.unibo.it

Abstract

The purpose of this report is to demonstrate the methodologies used to tackle a new NLP task named Human Value Detection, which is a multi-label classification task based on human arguments. We aimed to investigate the extent to which computational models can be utilized to deduce individuals' values from the text they write. To accomplish this task, we designed and implemented three distinct architectures, including SVM, BERT, and XLNet, alongside with a baseline. The results showed that XLNet performed the best among the three models, while SVM performed the worst. The models were evaluated using the F1-score measure, which is a suitable metric for multi-label classification tasks. Our results revealed a micro F1 score of 0.39, 0.47, and 0.48 for SVM, BERT, and XLNet, respectively. In addition we did all the experiments on the 10 most frequent labels and achived F1 score equal to 0.44, 0.51 and 0.52 for SVM, BERT, and XLNet.

1 Introduction

Human values play a crucial role in shaping a person's thoughts, opinions, and behavior, which can often be reflected in their words and actions. In the field of psychology, **values** are commonly defined as a set of beliefs and principles that an individual deems to be desirable and important (ROKEACH, 1973). For instance, someone who places a high value on "honesty" would likely have a strong negative reaction towards politicians who act in an unethical manner, and thus would be less likely to support them in future elections.

Values can be deeply ingrained in a person's language that is used on a day-to-day basis (Lepley, 1957). It acts not only as a tool of communication, but also as a reflection of their experiences and understandings. Through language, people express their beliefs, desires, and priorities unconsciously, making it imperative to identify patterns in language that are associated with certain values or belief systems. The significance of human value detection lies in its ability to influence decision-making and behavior across various fields, including healthcare, education, marketing, and others. In this current study, we aim to analyze the human values behind natural language discussions, which is categorized under argument mining task. This includes identifying the main claims and premises in an argument, even if they are implicit.

Within the research community, there have been numerous frameworks proposed that outline the set of core human values and their relationships with one another. One of the most widely discussed approaches is studying human values in social sciences, with one of the most popular frameworks being the ten value model developed by (Schwartz, 1994). Schwartz's ten value model has seen great success in psychological research as well as other fields. Schwartz and others also proposed a refined theory of basic individual values intended to provide greater heuristic and explanatory power than the original theory of 10 values (Schwartz et al., 2012), making it the baseline for many psychological and computational research in the field of value detection. Another approach involves using computational models to automatically identify, extract, and analyze arguments from text data. To the best of our knowledge, there has been limited research conducted on extracting values for argument mining using language models, making this study a pioneering effort in the field. The paper (Kiesel et al., 2022) is the first that aimed at identifying the values behind arguments computationally.

Our goal in this experiment is to classify arguments based on 20 provided classes, and given a textual argument and a human value category, the challenge is to determine whether or not the argument draws on that category. To achieve this goal, we implemented four different models containing a baseline model, a Support Vector Machine (SVM), a fine-tuned version of BERT, and a fine-tuned ver-

sion of XLNet. As mentioned in (Yang et al., 2019), XLNet is a generalized autoregressive pretraining method that enables the learning of bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order. It overcomes the limitations of BERT and is seen as a significant improvement, thanks to its autoregressive formulation. As compared to the SVM, which is based on a linear model, it may not be suitable for complex, non-linear text data. This highlights the potential of XLNet in the field of value detection and its ability to deliver more accurate results.

The SVM model performed the worst with an F1-score of 0.26, while the fine-tuned version of BERT had a score of 0.34. However, our implementation of XLNet performed the best, with an impressive F1-score of 0.55. Additionally, we also carried out the task for the ten most common categories using the same models and found that XLNet large had an F1-score of 0.60, BERT had a score of 0.50, and SVM had a score of 0.45.

2 Background

The task at hand falls under the subject of argument mining, as categorized by (Kiesel et al., 2022). Argument mining involves the automatic extraction of arguments from generic text corpora for the purpose of providing structured data for computational models and reasoning engines (Lippi and Torroni, 2016). The aim of argument mining is to understand the structure and components of an argument, as well as the relationships between them, by identifying claims, premises, and determining the strength or credibility of the argument. Human value detection, on the other hand, is concerned with identifying and analyzing values, beliefs, and attitudes expressed in a text. The goal of this process is to understand the perspective or sentiment of the author or speaker, as well as the values being communicated. Considering these two definition we can find the relation between argument mining and human value detection which both can be used to extract insights from text data.

XLNet is a transformer-based architecture designed to address the limitations of the popular BERT model and improve its performance in natural language understanding tasks. Both XLNet and BERT are transformer-based language models that have achieved state-of-the-art performance on a variety of natural language understanding tasks. However, XLNet has been shown to outperform BERT on specific tasks such as language understanding, question answering, and text classification. This is due to XLNet's permutation-based training approach, which allows it to better capture the context of words in a sentence compared to BERT's masked language modeling approach.

3 System description

The current project aims to tackle the problem at hand by implementing three architectures. The selection of these NNs was based on their performance on the given task and their suitability for the data and problem domain. In order to have a more comprehensive understanding of the performance of the NN models, all the architectures were run with three different seeds: 42, 1337, and 2022.

We considered a baseline which is simply a matrice of ones with the shape of y_true. The first architecture that was implemented is Support Vector Machines (SVM), a popular machine learning algorithm for text classification. SVM is often combined with the term frequency-inverse document frequency (TF-IDF) representation of text data. We chose to implement SVM to consider a simple supervised algorithm used for classification in order to compare it with powerful NLP algorithms. To use the SVM algorithm, the text data needs to be converted into a numerical representation that can be used as input. This is where TF-IDF comes into play. TF-IDF is a statistical measure that represents the importance of a word in a document.

The next two NN architectures that were implemented are the BERT and XLNet models, both fine-tuned for the task of sequence classification. The implementation involved loading the pre-trained models 'BertForSequenceClassification' and 'XLNetForSequenceClassification' from the transformers library, which are PyTorch implementations. These models come with a linear layer on top of the pooled output, designed for binary or multi-label classification tasks. The linear layer takes the final hidden state of the model (the representation of the input sequence) as input and outputs the logits (prediction scores) for the classification task. The BERT and XLNet models were chosen for this project because of their state-of-the-art performance on various NLP tasks such as language understanding, question answering, and text classification.

4 Data

In this experiment, we utilized data provided by (Kiesel et al., 2022) as the dataset for our research. The dataset has already been divided into separate sets, which include the training set with 5393 samples and the validation set with 1896 samples. However, since the labels for the test set were not provided by the data provider, we decided to devide the validation set into two parts, one for validation and one for test purposes, each one with 948 number of samples. The task we undertook involves a set of 20 values categories that have been compiled from the social science literature. Figure 1 illustrates the frequency of each value in the training and validation sets. It can be observed that the most frequent value in the training set is "Universalism: concern," while in the validation set it is "Security: personal." On the other hand, the least frequent values in the training and validation sets are "Hedonism" and "Conformity: Interpersonal," respectively.

Each argument in the training or validation set includes a unique argument ID, a premise, a conclusion, and a stance attribute. The stance attribute indicates whether the premise is in favor (pro) or against (con) the conclusion. The corresponding values in labels sets contain its argument ID and one column for each of the 20 value categories with a 1 meaning that the argument resorts to the value category and a 0 that not.

To prepare our data to be ready for feeding to SVM model we did some preprocessing on it. Here you can find a list of preprocessing we have done on the data:

- Transforming premise text to lower case.
- Replacing special characters, such as paranthesis, with spacing character
- Removing digits
- Replacing 'br' characters
- Removing any special character that is not in the good symbols list (check regular expression)
- Removing stopwords
- Removing any left or right spacing
- correcting spelling

For the other two pre-trained language representation models, we did not do preprocessing as they uses all of the information in a sentence. Since the inputs in the data have varying lengths, it is necessary to pad short inputs and truncate long ones. In our experiments, the maximum length was set to 92, which was determined based on the length of 99 percent of the inputs. The padding and truncation of the inputs were performed by two different tokenizers, including the BERT tokenizer and the XLNet tokenizer. The BERT tokenizer splits the input text into subwords and adds special tokens, such as [CLS] and [SEP], to indicate the beginning and end of a sentence. It uses WordPiece tokenization, where it segments words into smaller subwords based on their frequency of occurrence in the training corpus. On the other hand, the XLNet tokenizer uses a permutation-based tokenization process, where the input sequence is randomly permuted before being fed into the model. The XLNet tokenizer does not add special tokens, but instead uses the permuted order of tokens as a form of self-attention mechanism.

For the support vector machine (SVM), we selected the premise as the input since the conclusion simply repeats the same words as the premise without adding any new information to the context and just makes the input length longer and accordingly increases the training time. Additionally, the SVM does not consider the semantics of sentences and repeated words over the samples can negatively impact the performance of the model. In contrast, the transformer models took the conclusion, stance, and premise as input.

5 Experimental setup and results

Support Vector Machine (SVM) is a popular machine learning algorithm that was initially developed for binary classification problems. However, in order to use it as a multi-label classifier, which means that the model can predict multiple classes for a single input sample, we utilized a multi-output classifier. This technique involves wrapping the SVM classifier and training it on multiple independent outputs. Before the input data can be fed into the network, it is necessary to represent it in a numerical format. The choice of representation method is critical to the success of the SVM model and should be based on the specific problem being solved, taking into account the trade-off between accuracy, computational efficiency, and interpretability. In our experiment, we used the Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer as it offers a good compromise for many cases.

One of the key steps in preparing the data for the SVM model is to reduce the dimensionality

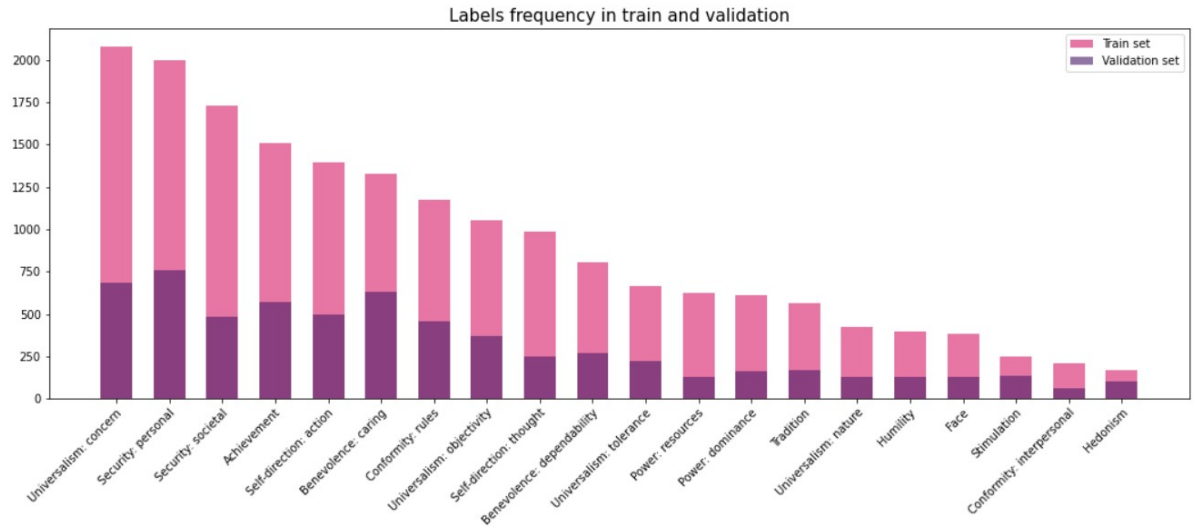

Figure 1: Data frequency

of the data. This is particularly important in our case as the data provided by TF-IDF is of high dimensional. To address this challenge, we utilized Principal Component Analysis (PCA) to reduce the dimensionality of the data. The use of PCA with SVM is a common practice that has been shown to improve the performance of the SVM algorithm, and in our case, we were able to reduce the dimension by almost half.

Finding the optimal model architecture is a crucial step in constructing a successful neural network model. In this experiment we decided to automate the process of exploring a range of possibilities to select the most promising architecture by using Optuna which is a hyperparameter optimization framework. The reason behind choosing it for tunning part is its advantages over other hyperparameter tuning techniques, including: using an efficient sampling algorithm called Tree-structured Parzen Estimator (TPE) that balances exploration and exploitation to quickly find good combinations of hyperparameters. Also Optuna allows to stop unpromising trials early to save resources and allows to perform distributed optimization using distributed optimization libraries. The result of tuning phase for SVM is to find the best kernel, the optimal regularisation (misclassification or error term) which tells the SVM optimisation how much error is bearable and the optimized Gamma parameter. We tried it in 10 trials and found the best hyper-parameters for seed 2022 as follows: 'Kernel': 'rbf', 'C': 1.0984200081270612, 'Gamma': 'auto'. The best parameters found for other seeds

Labels	Random Seed	Train	Val
20	42	0.93	0.39
	1337	0.96	0.37
	2022	0.93	0.39
10	42	0.93	0.44
	1337	0.95	0.43
	2022	0.93	0.44

Table 1: F1-Score over 20 and 10 human values for SVM

are presented in the notebook. In Figure 2 you can see one example of optimization history.

As we are dealing with imbalanced data, for evaluation of the models we used F1-score both micro and macro measure to provide a comprehensive evaluation of the model's performance. Table 1 summarizes the output obtained by SVM on train and validation sets. The SVM model has several limitations that render it less ideal for text classification. Two of the most significant drawbacks that impacted our issue are its inability to effectively capture semantic significance and lack of context sensitivity. These limitations are evident in the results produced.

Unfortunately, the hyperparameter tuning phase for BERT and XLNet could not be performed using the Optuna optimizer due to memory constraints, both on our devices and in Colab. The hyperparameters that are set for the transformers are as follows:

- **Number of epoches**: We experimented with the different number of epochs consisting 10,

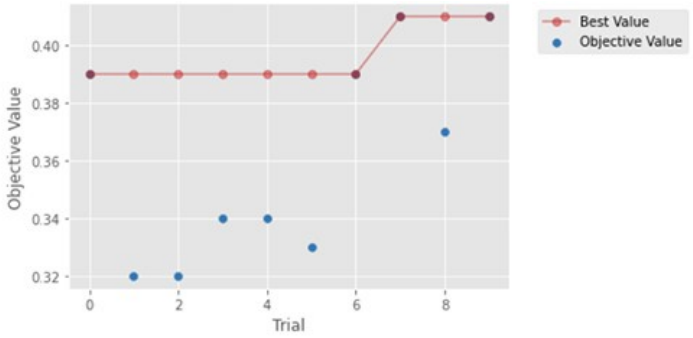

Figure 2: Optimization history plot for SVM classifier with seed 2022

15 and 20 for some models and found that 14 was the optimal number. This number is a good choice as it prevents overfitting while still allowing the model to learn effectively.

- **Optimizer**: We set AdamW as the model optimizer. Based on what presented in (Loshchilov and Hutter, 2019), AdamW provides a more principled approach to weight decay regularization compared to Adam, and is generally considered to be a better choice for deep learning models.
- **Weight-decay**: L2 regularization parameter of the model was set to 0.01 in order to prevent overfitting by reducing the magnitude of the model's weights.
- **Learning-rate**: The learning rate was set to 2e-5
- **Warmup-steps**: It was set to 500 which shows the number of steps during which the learning rate is gradually increased to its maximum value of 2e-5 at the end of the warm-up period.
- **Resume-from-checkpoint**: It was set to True, to force resuming training from the last saved checkpoint if available.

The models were trained twice (for each seed), once using all the labels and once using only the 10 most frequent labels. This behaviour is also proposed in the main paper (Kiesel et al., 2022). The evaluatioan results on all categories and 10 categories on the validation data for the best models are shown in Figure 3 and Figure 4 respectively.

6 Discussion

Table 2 Summarizes Transformers model's evaluation on validation data. It can be seen that among the models trained on all labels, XLNet with seed 2022 outperforms the others with F1 micro score of 0.48. Among the models trained on 10 most frequent labels, XLNet with seed 2022 performs the best with F1 score equal to 0.52.

Model	Labels	Seed	F1 Macro	F1 Micro
BERT	20	42	0.29	0.45
		1337	0.30	0.47
		2022	0.36	0.47
	10	42	0.44	0.50
		1337	0.42	0.51
		2022	0.43	0.51
XLNet	20	42	0.33	0.48
		1337	0.32	0.47
		2022	0.35	0.48
	10	42	0.44	0.51
		1337	0.47	0.52
		2022	0.47	0.52

Table 2: F1-score over 20 and 10 Human Values Categories for the BERT and XLNet models

The purpose of training the models on three different seeds was to ensure that variations in the seed did not significantly impact the f1 score. After comparing the effects of the different seeds, we then compared the three architectures using only one seed (since the results from the other seeds were nearly identical) in order to focus on comparing the architecture rather than the seed. The results obtained during this comparison were significantly different.

In the obtained results it is clearly noticeable that the F1 macro score is lower than the F1 micro score for models with 20 labels. It means the models are performing well on the majority classes, but not so well on the minority classes. This is known as the class imbalance problem, where some classes have very few examples and are therefore difficult to predict accurately. In such cases, it might be beneficial to focus on improving the performance of the minority classes, either by oversampling or

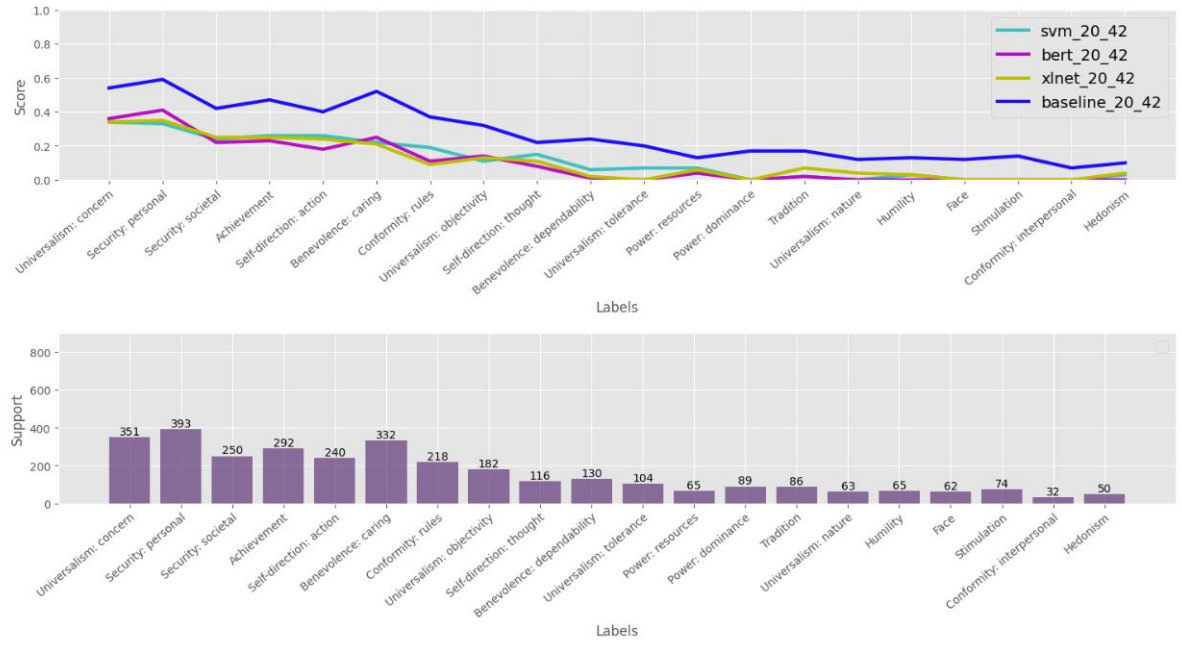

Figure 3: F1-score over all values for four best implemented architectures

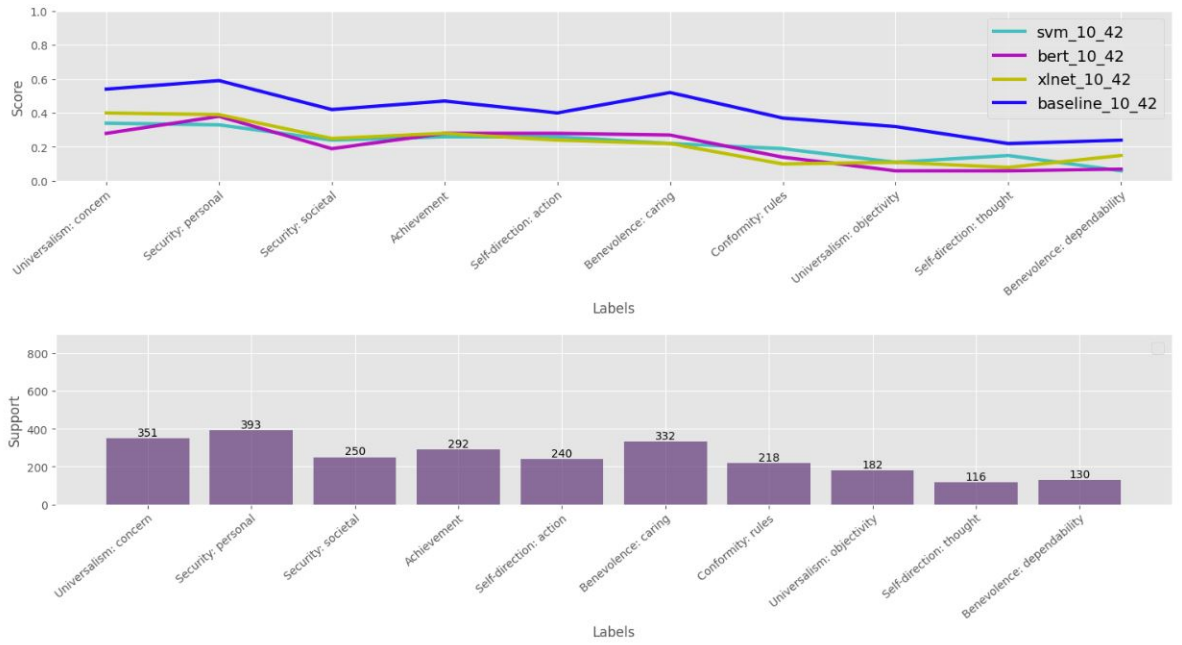

Figure 4: F1-score over 10 values for four best implemented architectures

undersampling the majority class, or by using more advanced techniques like cost-sensitive learning. The crucial aspect is that the models trained on the 10 labels do not experience the class imbalance issue, as the chosen labels guarantee a balanced distribution of classes and as a result there is no meaningful difference between the two mentioned measures for these models.

While considering only the n most frequent labels can reduce the number of classes in the problem and thus make the model training and prediction faster, it can also lead to the loss of information and reduce the diversity of the training data. Sometimes, rare labels can still be important for our prediction task, and ignoring them can result in biased or incomplete predictions. However, if the aim is to focus on the most important classes, or if the data is highly imbalanced with a very large number of classes, considering only the n most frequent labels can be a reasonable strategy, like in our case.

During training phase of the BERT model, we noticed that the F1-score started out very low and reached 0 within the next 2 to 3 epochs, but then increased in subsequent epochs. This is due to a phenomenon known as "catastrophic forgetting". In the early epochs of fine-tuning, the model updates its weights to better perform the task it was fine-tuned for, but in doing so, it may forget some of the information it had learned from its pre-training on large amounts of text data. As fine-tuning continues, the model is able to recover some of this information and improve its performance. This is why the F1 score here starts low and then increases in later epochs. It is a normal behavior when fine-tuning BERT for sequence classification. In fact, it is expected and a sign that the pre-trained BERT model is being fine-tuned effectively for the new task. The model is able to learn task-specific information while retaining its knowledge from pre-training, which helps improve its performance over time.

7 Conclusion

In conclusion, the study aimed to compare the performance of three different models for addressing the task of Human value detection: SVM, BERT, and XLNet. The results indicated that the XLNet model outperformed the other two models, achieving the highest F1-score on the evaluation metrics. This highlights the potential of XLNet as a powerful tool for text classification tasks. In summary, this study provides insights into the current state of the art in human value detection and highlights the importance of selecting the appropriate model for specific tasks.

References

Johannes Kiesel, Milad Alshomary, Nicolas Handke, Xiaoni Cai, Henning Wachsmuth, and Benno Steinón-Cedeño. 2022. Identifying the human values behind arguments. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 4459–4471.

R Lepley. 1957. In *The Language Value*. Columbia University Press, New York.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology*, 16(2):10:1–10:25.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. URL: https://arxiv.org/abs/1711.05101.

M ROKEACH. 1973. In *The Nature of Human Values*, volume 70. New York Free Press.

Shalom H Schwartz. 1994. Are there universal aspects in the structure and contents of human values? *Journal of Social Issues*, 50:19–45.

Shalom H Schwartz, Jan Cieciuch, Michele Vecchione, Eldad Davidov, Ronald Fischer, Constanze Beierlein, Alice Ramos, Markku Verkasalo, Jan-Erik Lönnqvist, and Kursad Demirutku ón-Cedeño. 2012. Refining the theory of basic individual values. In *Journal of personality and social psychology*, volume 103.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. In *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. 33rd Conference on Neural Information Processing Systems.