Adam-Smith at SemEval-2023 Task 4: Discovering Human Values in Arguments with Ensembles of Transformer-based Models

Daniel Schroter, Daryna Dementieva and **Georg Groh**
TU Munich, Department of Informatics, Germany
daniel@schroter.biz, daryna.dementieva@tum.de, grohg@in.tum.de

Abstract

This paper presents the best-performing approach alias "Adam Smith" for the SemEval-2023 Task 4: "Identification of Human Values behind Arguments". The goal of the task was to create systems that automatically identify the values within textual arguments. We train transformer-based models until they reach their loss minimum or f1-score maximum. Ensembling the models by selecting one global decision threshold that maximizes the f1-score leads to the best-performing system in the competition. Ensembling based on stacking with logistic regressions shows the best performance on an additional dataset provided to evaluate the robustness ("Nahj al-Balagha"). Apart from outlining the submitted system, we demonstrate that the use of the large ensemble model is not necessary and that the system size can be significantly reduced.

1 Introduction

"We should ban whaling, whaling is wiping out species for little in return." say some, "We shouldn't, it is part of a great number of cultures." say others. Both arguments support their claim, but why are some arguments more convincing to us than others? This might relate to the underlying values they address. Whereas the first argument appeals to the value of "universalism: nature", the second one addresses the value of "tradition". Whether an argument is in agreement or disagreement with our values influences its ability to persuade. The task organizers (Kiesel et al., 2023) are the first who extend the field of argument mining by this "value" dimension. As part of the SemEval-2023 workshop, they organize the task of automatically detecting human values behind arguments. They decided to add two additional test datasets to evaluate the robustness of the developed systems (Mirzakhmedova et al., 2023).

Our system uses an ensemble of transformer-based models that are either trained until they reach their loss minimum or the f1-score maximum. To ensemble the models we average the individual predictions and calculate a decision threshold for the final system on a separate "Leave-out-Dataset". **This model achieves the best performance in the competition** ("Main" dataset). Each team was allowed to submit up to four systems. Ensembling the predictions by using stacking with logistic regressions leads to the system with the best performance on the additional "Nahj al-Balagha" dataset. In this paper, we describe the best-performing system on the "Main" dataset and briefly outline the ideas behind the other submitted systems.

The system can be accessed through a web-based interface that is available online[1]. Furthermore, a docker container, models, and code are open source and publicly accessible (Appendix A).

2 Background

Mirzakhmedova et al. (2023) created a labeled dataset of 9324 arguments from 6 different sources. The arguments are structured as follows:

Premise	whaling is part of a great number of cultures
Conclusion	We should ban whaling
Stance	against
Labels	['Tradition', 'Conformity: interpersonal']

Table 1: Example argument about whaling

The arguments are in English and in total there are 20 different value categories to predict. Each argument is labeled with one or multiple values. Hence the task at hand can be characterized as a multi-labeling problem. The systems can be tested against two additional datasets to evaluate their robustness on unseen data from different domains. The "Nahj al-Balagha" dataset contains arguments from Islamic religious texts. The "New York Times" dataset contains arguments from texts about

[1] https://values.args.me/

COVID-19. A detailed introduction to the datasets can be found in Mirzakhmedova et al. (2023). The task organizers (Kiesel et al., 2023) created two baseline models: 1-baseline and a BERT-based system. The system we propose builds upon the transformer architecture by Vaswani et al. (2017) and in particular the BERT model (Devlin et al., 2018). In fact, we use two improved versions of the original BERT model called RoBERTa (Liu et al., 2019) and DeBERTa (He et al., 2021). Further, we apply well-known techniques from the field of practical AI such as ensembling (Zhou, 2012), cross-validation and early-stopping (Goodfellow et al., 2016). By presenting the best-performing system in the task and outperforming the baselines by a large margin, this paper delivers valuable insights into how values can be automatically detected within text.

3 System Overview

This section is dedicated to precisely describing the best-performing submission as a baseline for future research and further development. The proposed system is an ensemble of 12 individual models. Figure 1 provides an overview of the inference pipeline of the final system. We briefly outline the process of making a prediction and subsequently describe each of the individual steps in detail. To make a prediction the following steps are performed:

1. We take an input argument and concatenate the premise, stance, and conclusion.

2. The input is then fed into the neural networks. The output of each neural network is a vector containing 20 values with the "confidence" (values between 0 and 1) whether the sample has the corresponding label. The final system consists of 12 models, so we get 12 of these vectors.

3. We ensemble the opinions of the models by taking the average of the 12 vectors per label.

4. As we now have the averaged values for each of the 20 labels we must decide which labels to assign. Therefore we use a threshold. For the values in the vector that are above the threshold, the corresponding label is assigned.

3.1 Data Preprocessing

Transformer-based models are trained on large corpora of natural text. Hence, it seems reasonable to transform the input in a format that is most similar to a human-like formulation. Therefore we concatenate the premise, stance and conclusion into a single text string. This transforms the above example (Table 1) into **whaling is part of a great number of cultures against We should ban whaling**.

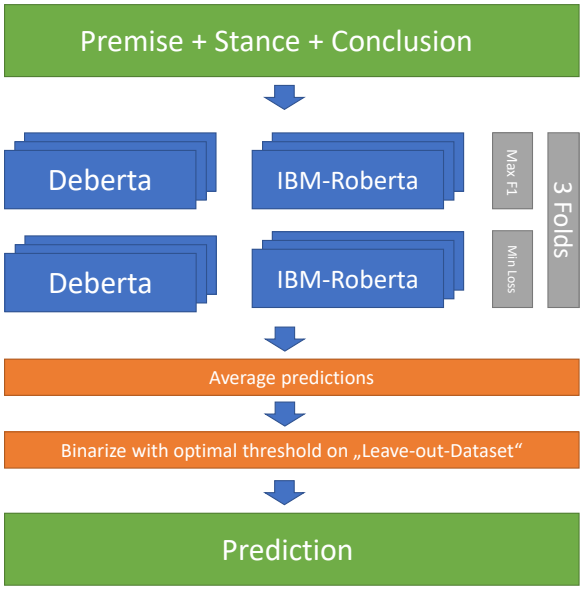

Figure 1: Inference System

3.2 The Models

The final system is an ensemble of 12 individual models that are based on the transformer architecture. Each model has the same structure: It consists of a transformer-based language model with one additional fully connected linear layer on top. We use the CLS token as input for the additional layer as described by Devlin et al. (2018). The sigmoid function maps the values for each label to the (0,1) interval.

We use two deviations of BERT (Devlin et al., 2018), namely "microsoft/deberta-large" ("Deberta") and a pretrained roberta-large model ("IBM-Roberta") as base models. They are then optimized for loss minimization or f1-score maximization leading to four different configurations as depicted in blue in Figure 1. Each of the four configurations is trained on 3 folds, leading to the 12 models in the final system (Table 9 in the Appendix). In the following sections, we describe the different configurations in more detail.

3.2.1 Pretraining

According to Mirzakhmedova et al. (2023) 83% of the arguments are retrieved from the IBM-ArgQ-Rank-30kArgs dataset (Gretz et al., 2019). We pretrain our models on the IBM-dataset using masked language modeling (Devlin et al., 2018) to shift the language understanding capabilities of our model

Pretraining	F1 Validation
IBM-Deberta-Large	.516
Microsoft Deberta Large	**.523**
IBM-Roberta-large	**.529**
Roberta-large	.519

Table 2: Pretraining: Values are calculated without tuned hyperparameters and before the training data update by the organizers during the competition.

towards the specific language in the arguments. Table 2 shows that pretraining only improves the roberta model but not the deberta model. Subsequently, we proceeded with the pre-trained roberta model (IBM-Roberta) and used the deberta model in its base version.

3.2.2 Optimizing for Loss and f1-score

A traditional training procedure requires a train-validation split. To prevent models from overfitting, the training is stopped as soon as the validation loss reaches its minimum. Within the task, the models are evaluated against the f1-score. So instead of training until the minimum loss is reached we train the models until they reach the maximum f1-score.

$$f1 = \frac{2 * recall_{macro} * precision_{macro}}{recall_{macro} + precision_{macro}} \quad (1)$$

The used f1-score (1) differs slightly from the f1-scores as defined in most software packages (e.g macro average f1-score in sklearn[2]). Instead of calculating the f1-score for each label and then taking the average, the "macro average recall"[3] and "macro average precision"[4] are calculated first and are then used in the f1-score (formula 1).

Now the question arises of how to optimize for the f1-score. During one validation step, we get the predictions for the validation set. Instead of calculating the loss, we could binarize the predictions according to a threshold and use them to calculate the f1-score. We could then train the model until no further improvement in the f1-score is observed. This procedure would require that we select the threshold at each validation step that maximizes the f1-score.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html

[4] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html

Algorithm 1 Threshold selection with f1-score maximization

```
Input: yTrue, yPred
Output: optimal threshold optT

threshold ← 0
maxF1 ← 0
optT ← 0
while threshold ≤ 1 do
    yPredBin ← binarize yPred with threshold
    recall ← recall(yTrue, yPredBin, average="macro")
    precision ← precision(yTrue, yPredBin,
                          average="macro")
    if precision + recall ≠ 0 then
        f1 ← 2 * recall * precision / (recall + precision)
        if f1 ≥ maxF1 then
            optT ← threshold
            maxF1 ← f1
        end if
    end if
    threshold ← threshold + 0.01
end while
return optT
```

Consequently, we define Algorithm 1 to determine the threshold that maximizes the f1-score. As input, it takes a set of true labels and a set of predictions with values between 0 and 1. We iterate over all possible thresholds in small steps (0.01). In each iteration, we binarize the predictions according to the threshold and calculate the corresponding f1-score. As output, we return the threshold that maximizes the f1-score.

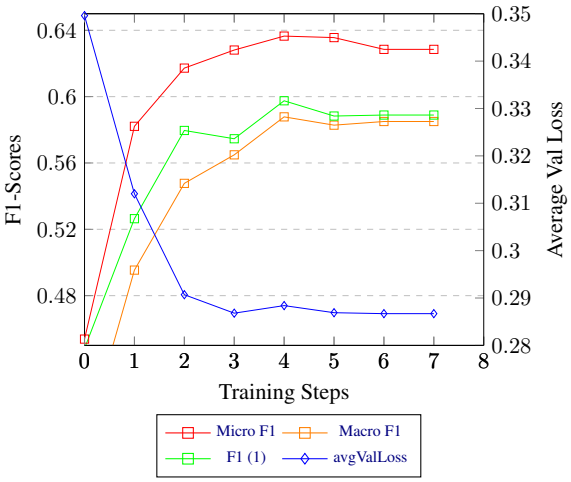

Figure 2: F1-scores and average validation loss in each training step in a single run of a Deberta model

In Figure 2 we can clearly see, how the model improves its performance during training. It further shows that the training step that minimizes the average validation loss and the training step maximizing the f1-score can differ. The average

validation loss (blue) has its minimum at step 3, whereas the f1-score (green) has its maximum at training step 4. Subsequently, we decide to optimize each model with respect to the f1-score and with respect to the loss. This leads to the two blue rows in Figure 1. The first blue row in Figure 1 indicates the training with the goal of f1-score maximization, whereas the second row indicates the validation loss minimization.

3.2.3 Cross-Validation

To make use of as much training data as possible we apply a variant of 3-fold-cross-validation. Each of the four model configurations is trained three times, with a validation set of 500 samples. Each validation set is created by taking a different random split from the training dataset. Figure 1 reflects these 3 versions of each model with the blue boxes in the background.

3.3 Ensembling

During the training process, various models (section 3.2) are developed. Hence, the question arises of how to ensemble them to create the final labels. Our final submissions contain two different approaches.

The best-performing system uses Algorithm 1 to select an optimal threshold that can be used for the test dataset. In order to do so, the models are not trained on the whole dataset but instead, we split a "Leave-Out-Dataset" of 300 samples apart. These 300 samples are not seen by any of the models before and are used to determine the optimal threshold.

Recipe I: 1. Get the predictions on the "Leave-Out-Dataset" for all single models. 2. Average the individual predictions. 3. Select the optimal threshold for the "Leave-Out-Dataset" that maximizes the f1-score with Algorithm 1 4. Repeat steps 1 and 2 for the test dataset and use the optimal threshold for the final prediction.

We also submitted the best-performing system on the "Nahj al-Balagha"-dataset. This system has the same architecture but uses stacking (Wolpert, 1992) as an ensemble method. Instead of defining one "global" threshold for all labels, we train logistic regressions for each label to decide whether a label should be 0 or 1. The models are trained on the entire dataset.

Recipe II: 1. Get the predictions for 3000 samples of the training dataset for all single models. 2. Train multiple logistic regressions[5] (input: predictions, output: true labels) to predict the labels based on the predictions. 3. Get the predictions for the test dataset and use the trained logistic regressions to predict the final labels.

4 Experimental Setup

The final system was trained on the data provided by the task organizers (training + validation set) except for a "Leave-Out-Dataset" of 300 samples. During the different cross-validation runs a validation set of 500 samples is taken from the training data. We use a linear learning rate schedule and early stopping. We stop the training process if the validation loss or f1-score does not improve in 3 consecutive evaluation steps. Especially the parameter for the learning rate schedule (total_training_steps) was manually optimized because it defines the speed at which the learning rate decays and is therefore crucial for the learning process. The hyperparameters for pretraining and finetuning, used model versions as well as links to the code and a docker container reproducing the results can be found in Appendix A.

During the competition, we used an internal test dataset of 500 samples to create an internal "leaderboard" and choose the final systems to submit in the competition.

The models are implemented with PyTorch-Lightning and are trained on an NVIDIA Tesla T4 GPU.

5 Results

Each team was allowed to make up to four submissions (Table 3).

EN-Thres-LoD: The system described in this paper and the best-performing system in the competition. The ensemble threshold is calculated on a "Leave-out-Dataset" ("EN-Thres-LoD"), and the model achieves an f1-score of 0.56 (Table 3).

EN-Log-Reg: A system that uses stacking with logistic regressions as an ensemble method (Section 3.3). It has an f1-score of 0.54 on the "Main" dataset and was the best-performing system with an f1-score of 0.40 on the "Nahj al-Balagha" dataset.

EN-Thres-Train: The same system architecture as EN-Thres-LoD. Instead of calculating the threshold on the "Leave-Out-Dataset", the entire dataset

[5] https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.MultiOutputClassifier.html#sklearn.multioutput.MultiOutputClassifier

was used for training and the optimal threshold was calculated on the training data.

EN-Silver-Labels: The system uses a self-training approach (Appendix B.1). A first version of a system creates additional training data ("silver labels") for arguments from the IBM-30k dataset. The model architecture is the same as "EN-Thres-LoD". The size of the training data is then increased to 140% by adding the "silver-label" data. We followed a similar approach as in Jurkiewicz et al. (2020). The performance during training seems to slightly improve (Figure B.1), but when ensembled the system was outperformed by the other architectures.

All of our four submitted systems rank high in the competition on the "Main" dataset and the "Nahj al-Balagha". On these datasets, our systems outperform the BERT baseline by a large margin. However, there is a different picture for the "New York Times" dataset. With a maximum score of 0.27, the systems are only slightly better than the BERT baseline and are outperformed by other systems by a large margin (0.34 as "best approach" in Table 3).

The scores reported in Table 3 refer to the overall scores across all labels. Clearly, the model delivers better results for some labels than for others. A table with the scores for each label can be found in Table B in the appendix. The system delivers the best performances for the labels "Universalism: nature" and "Security: personal" with f1-scores of 0.82 and 0.76 respectively. The weakest performances are seen for the labels "Hedonism" (0.25) and "Stimulation" (0.32). The dataset is imbalanced and there seems to be a correlation between the frequencies of labels in the training data and the performance of the system (Figure 4).

5.1 Ablation study

The above-presented approach turned out to be sub-optimal. Based on our own "internal leaderboard" it seemed like the suggested ensemble of all 12 models has slightly superior performance (F1 inter. in Table 8 in Appendix B.3). After submission, we evaluated the individual components of the final ensemble on the official test dataset. Table 3 shows that ensembling the 3 deberta models, which are optimized for the f1-score (EN-Deberta-F1) leads to slightly better performance (0.57) on the "Main" dataset and significantly better performance on the "New York Times" dataset (0.34). A single Deberta model, optimized for f1-score (Single-Deberta-F1) leads to slightly worse performance on the test dataset (0.55) but significantly better performance on the "New York Times" dataset (0.37). The model selection for these ensembles can be found in Table 9 in the Appendix.

Approach / Test Dataset	**Main**	**Nahj al-Balagha**	**New York Times**	Threshold	# models
Main					
Best per category	.59	.48	.47		
Best approach	.56	.40	.34		
BERT	.42	.28	.24		
1-Baseline	.26	.13	.15		
Submitted Models					
EN-Thres-Train	.56	.36	.26	0.32	12
EN-Log-Reg	.54	**.40**	**.27**	-	12
EN-Thres-LoD (1st)	**.56**	.34	.25	0.26	12
EN-Silver-Labels	.54	.34	.24	0.29	12
Ablation Studies					
EN-Deberta-F1*	.57	.33	.34	0.27	3
Single-Deberta-F1*	.55	.35	.37	0.25	1

Table 3: Achieved f1-score of team adam-smith per test dataset. Approaches marked with * were not part of the official evaluation. Approaches in gray are shown for comparison: an ensemble using the best participant approach for each individual category; the best participant approach; and the organizer's BERT and 1-Baseline.

5.2 Other approaches

Besides the different approaches submitted for the competition, we have experimented with the generative T5 transformer model. The T5 model is trained on multiple downstream tasks including natural language inference. We assumed that this multi-tasking ability might have a beneficial influence on our task. Therefore we fine-tuned a T5-large model to predict the values. With an f1-score of 0.493 on the validation set it beats the baseline BERT model but is far from the performance of the other approaches. We further weighted the loss function according to the class distribution in order to account for the class imbalance problem. This was ruled out in the early stages of the system development because it showed slightly lower performance. Comparing several approaches can be complicated due to the randomness in training procedures and the need for different hyperparameter optimizations. These modeling decisions are based on the performances captured in Table 7 in the appendix.

6 Conclusion

We have presented the best-performing system for the task of automatically detecting human values in arguments. Our system ensembles 12 models that have been either optimized for loss minimization or f1-score maximization. As an ensemble method, we choose one global decision threshold for all labels. The threshold maximizes the f1-score for a "Leave-out-Dataset". This approach outperforms stacking as an ensemble method, where for each label a logistic regression is trained. Even though our systems show the best performances on the "Main" and "Nahj al-Balagha" datasets, they are outperformed by other approaches on the "New York Times" dataset. In the ablation study, we show that such a large ensemble is not necessary. In fact, an ensemble of only 3 models shows better performance and robustness while decreasing the system's memory requirements significantly.

For future work, an analysis of this phenomenon could be considered. Does the reduction of ensemble size lead to a more robust system and what are the counter-productive elements in the larger ensemble? Furthermore, we have only manually combined a few different models into an ensemble. Hence, it would be interesting to see whether a systematic selection of different approaches within an ensemble could further boost performance.

7 Limitations

The proposed system is trained on a very specific argument structure taken from the IBM-30k dataset. The system's performance noticeably declines when tested on additional datasets, which raises questions about their ability to handle new, unprepared datasets and textual arguments with robustness. Furthermore, the best performing system consists of 12 individual models leading to a high resource requirements.

8 Acknowledgments

We would like to thank Nicolas Handke and Johannes Kiesel for creating the docker container and the web application and thereby making the system easier accessible for future research.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press, Cambridge, MA, USA. http://www.deeplearningbook.org.

Shai Gretz, Roni Friedman, Edo Cohen-Karlik, Assaf Toledo, Dan Lahav, Ranit Aharonov, and Noam Slonim. 2019. A large-scale dataset for argument quality ranking: Construction and analysis.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention.

Dawid Jurkiewicz, Lukasz Borchmann, Izabela Kosmala, and Filip Gralinski. 2020. Applicaai at semeval-2020 task 11: On roberta-crf, span CLS and whether self-training helps them. *CoRR*, abs/2005.07934.

Johannes Kiesel, Milad Alshomary, Nailia Mirzakhmedova, Maximilian Heinrich, Nicolas Handke, Henning Wachsmuth, and Benno Stein. 2023. Semeval-2023 task 4: Valueeval: Identification of human values behind arguments. In *Proceedings of the 17th International Workshop on Semantic Evaluation*, Toronto, Canada. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Nailia Mirzakhmedova, Johannes Kiesel, Milad Alshomary, Maximilian Heinrich, Nicolas Handke, Xiaoni Cai, Barriere Valentin, Doratossadat Dastgheib, Omid Ghahroodi, Mohammad Ali Sadraei, Ehsaneddin Asgari, Lea Kawaletz, Henning Wachsmuth, and Benno Stein. 2023. The Touché23-ValueEval Dataset for Identifying Human Values behind Arguments. *CoRR*, abs/2301.13771.

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

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2):241–259.

Z.H. Zhou. 2012. *Ensemble Methods: Foundations and Algorithms*. CHAPMAN & HALL/CRC MACHINE LEA. Taylor & Francis.

A Experimental Setup

This section includes the appendices for the experimental setup such as links to relevant resources and hyperparameters.

A.1 Code, Docker Container and Models

The Code[6] and Docker Container[7] for the final system are available online. We pushed the Single Deberta Model[8] from the ablation studies to Huggingface for simple usage. Furthermore, all models are publicly available.[9]

A.2 Language Models

We used the microsoft/deberta-large[10] model from huggingface and the pretrained roberta-large.[11]

A.3 Hyperparameters

Parameters	Value
General	
batch_size	8
epochs*	3* (see caption)
Optimizer	
Optimzier	AdamW
learning_rate_schedule	linear
Learning Rate	2e-5
total_training_steps	2502
n_warmup_steps	500
Early_Stopping	
validation_inverval	300
epochs	early_stopping*
patience	3

Table 4: Hyperparameters: The epochs are not actually the number of trained epochs. Instead they are used to calculate the linear learning rate schedule by calculating the Total_training_steps = (len(training_data)//Batch_Size)*Epochs. The models are then trained with early stopping

Parameters	Value
Batch Size	16
accumulated gradients	2
epochs	8
Learning Rate	2e-5

Table 5: Hyperparameters Pretraining with Masked Language Modelling

[6] https://github.com/danielschroter/human_value_detector

[7] https://github.com/touche-webis-de/team-adam-smith23

[8] https://huggingface.co/tum-nlp/Deberta_Human_Value_Detector

[9] https://zenodo.org/record/7656534#.Y_yKdyaZP30

[10] https://huggingface.co/microsoft/deberta-large

[11] https://huggingface.co/danschr/roberta-large-BS_16-EPOCHS_8-LR_5e-05-ACC_GRAD_2-MAX_LENGTH_165/tree/main?doi=true

B Results

B.1 Self-training and Silver Labels

We further applied a self-training procedure: 1) First we trained the IBM-Roberta and Deberta Model for f1-score-maximization. 2) We then ensembled the models and defined the optimal threshold based on a small Leave_out_Dataset. 3) We used the ensemble to created additional silver labels (additional training data) from the IBM-30k dataset. Thereby we ensured to not include samples, that are in our internal test dataset or the test dataset of the competition. 4) We retrained the models on an extended dataset including the silver labels in the training data. (During Cross-Validation, it is ensured that silver-labels are not added to the validation set). Figure 3 shows how pretraining might have a positive impact on the performance of the model. The values are the averages from cross-validating with 3 folds. However, the submitted system (ensemble of 12) containing the self-trained models was outperformed by the other models.

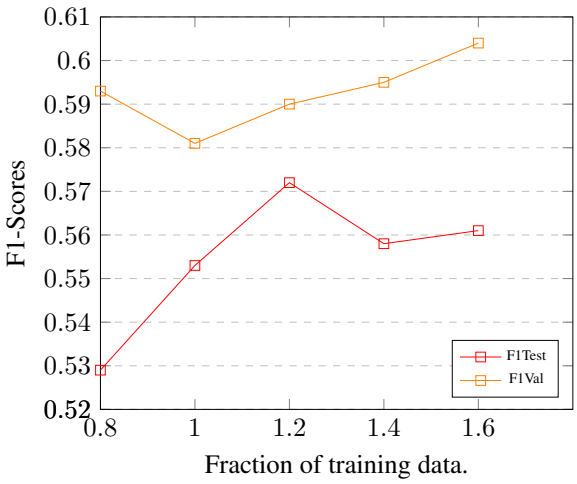

Figure 3: F1 Scores on average Validation loss during Cross-Validation (F1 Val) and the internal Test-Dataset (F1 Test). The X Axis represents the amount of training data used. Values above 1 indicate that additional silver-labels are included in the training.

B.2 Results per category

Table 6 contains the performances of the systems submitted for the competition and developed in the ablation studies. It shows the performance for each label individually.

Test set / Approach	All	Self-direction: thought	Self-direction: action	Stimulation	Hedonism	Achievement	Power: dominance	Power: resources	Face	Security: personal	Security: societal	Tradition	Conformity: rules	Conformity: interpersonal	Humility	Benevolence: caring	Benevolence: dependability	Universalism: concern	Universalism: nature	Universalism: tolerance	Universalism: objectivity
Main																					
Best per category	.59	.61	.71	.39	.39	.66	.50	.57	.39	.80	.68	.65	.61	.69	.39	.60	.43	.78	.87	.46	.58
Best approach	.56	.57	.71	.32	.25	.66	.47	.53	.38	.76	.64	.63	.60	.65	.32	.57	.43	.73	.82	.46	.52
BERT	.42	.44	.55	.05	.20	.56	.29	.44	.13	.74	.59	.43	.47	.23	.07	.46	.14	.67	.71	.32	.33
1-Baseline	.26	.17	.40	.09	.03	.41	.13	.12	.12	.51	.40	.19	.31	.07	.09	.35	.19	.54	.17	.22	.46
EN-Thres-Train	.56	.59	**.71**	.22	.29	**.66**	.48	.52	.30	**.79**	.67	**.65**	.61	.61	.19	**.60**	.36	.74	.84	.41	**.53**
EN-Log-Reg	.54	**.61**	**.71**	.20	.29	.62	.46	.44	.30	.78	**.68**	.64	.59	.61	.20	.59	.36	**.76**	**.85**	.38	.49
EN-Thres-LoD (1st)	.56	.57	**.71**	**.32**	.25	**.66**	.47	.53	**.38**	.76	.64	.63	.60	**.65**	**.32**	.57	**.43**	.73	.82	.46	.52
EN-Silver-Labels	.54	.58	.70	.13	.29	.65	.45	.53	.19	.73	.59	.64	.55	.60	.16	.57	.38	.71	.84	.46	.50
EN-Deberta-F1*	**.57**	.57	**.71**	.30	**.34**	.65	**.50**	**.55**	**.38**	.78	.64	.64	.60	.60	**.32**	.57	**.43**	.75	.83	**.47**	**.53**
Single-Deberta-F1*	.55	.54	.70	.29	.32	.65	.44	.55	.37	.77	.63	.62	**.62**	**.65**	.29	.55	.42	.74	.81	.46	.52
Nahj al-Balagha																					
Best per category	.48	.18	.49	.50	.67	.66	.29	.33	.62	.51	.37	.55	.36	.27	.33	.41	.38	.33	.67	.20	.44
Best approach	.40	.13	.49	.40	.50	.65	.25	.00	.58	.50	.30	.51	.28	.24	.29	.33	.38	.26	.67	.00	.36
BERT	.28	.14	.09	.00	.67	.41	.00	.00	.28	.28	.23	.38	.18	.15	.17	.35	.22	.21	.00	.20	.35
1-Baseline	.13	.04	.09	.01	.03	.41	.04	.03	.23	.38	.06	.18	.13	.06	.13	.17	.12	.12	.01	.04	.14
EN-Thres-Train	.36	.12	.43	**.50**	**.50**	**.66**	.22	.00	.56	.50	.23	.55	.23	.15	**.31**	.30	.27	.26	.40	.00	.35
EN-Log-Reg (1st)	**.40**	**.13**	**.49**	.40	**.50**	.65	.25	.00	.58	.50	**.30**	.51	**.28**	**.24**	.29	**.33**	**.38**	.26	**.67**	.00	**.36**
EN-Thres-LoD	.34	.09	.33	.33	.44	.59	.22	.20	**.62**	.51	.20	**.55**	.23	.12	.24	.26	.24	**.29**	.40	.05	.30
EN-Silver-Labels	.34	.06	.37	.33	.40	.62	.22	.18	.51	.49	.23	.51	.21	.23	.20	.24	.24	.24	.50	.00	.32
EN-Deberta-F1*	.33	**.13**	.34	.25	.31	.64	.21	**.22**	.57	.53	.21	**.55**	.23	.15	.27	.27	.21	.24	.40	**.11**	.30
Single-Deberta-F1*	.35	.10	.35	.15	.29	.65	.22	.15	.55	**.54**	.25	.46	.24	.17	.20	.25	.23	.25	**.67**	.10	.34
New York Times																					
Best per category	.47	.50	.22	-	.03	.54	.40	-	.50	.59	.52	-	.33	1.0	.57	.33	.40	.62	1.0	.03	.46
Best approach	.34	.22	.22	-	.00	.48	.40	-	.00	.53	.44	-	.18	1.0	.20	.12	.29	.55	.33	.00	.36
BERT	.24	.00	.00	-	.00	.29	.00	-	.00	.53	.43	-	.00	.00	.57	.26	.27	.36	.50	.00	.32
1-Baseline	.15	.05	.03	-	.03	.28	.03	-	.05	.51	.20	-	.07	.03	.12	.12	.26	.24	.03	.03	.33
EN-Thres-Train	.26	.29	.14	-	.00	**.54**	.00	-	.00	.56	.42	-	.23	.00	.00	**.33**	**.40**	.58	.33	.00	.40
EN-Log-Reg	.27	**.33**	**.18**	-	.00	.42	.00	-	.00	**.58**	**.52**	-	.18	.00	.00	.21	.31	**.62**	**.50**	.00	**.46**
EN-Thres-LoD	.25	.18	.17	-	.00	.42	.00	-	.00	.57	.38	-	.27	.00	.20	.26	.37	.50	.33	.00	.42
EN-Silver-Labels	.24	.22	.15	-	.00	.42	.00	-	.00	.56	.36	-	**.29**	.00	.00	.26	.34	.50	.40	.00	.44
EN-Deberta-F1*	.34	.22	.15	-	**1.0**	.44	.00	-	.33	.56	.37	-	**.29**	.00	.22	.28	.24	.44	.33	.00	.42
Single-Deberta-F1*	**.37**	.22	.00	-	**1.0**	.42	.00	-	**.40**	.56	.36	-	.22	**.67**	.22	.21	.29	.48	.33	.00	.45

Table 6: Achieved f1-score of team adam-smith per test dataset, from macro-precision and macro-recall (All) and for each of the 20 value categories. Approaches marked with * were not part of the official evaluation. Approaches in gray are shown for comparison: an ensemble using the best participant approach for each individual category; the best participant approach; and the organizer's BERT and 1-Baseline. The bold values highlight the best per category approach.

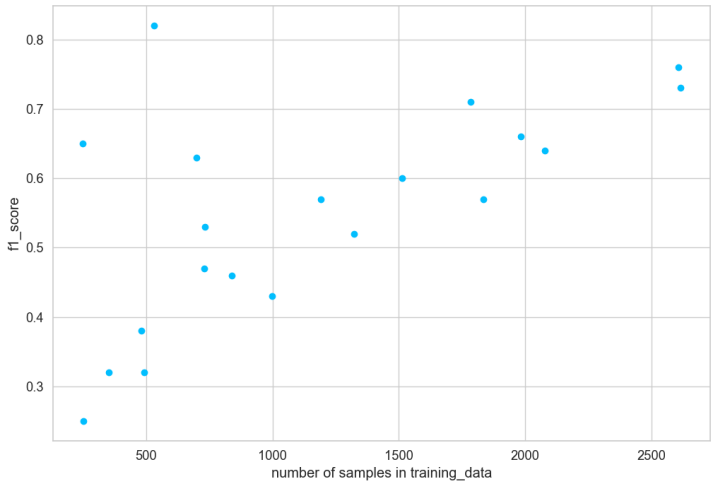

Figure 4: Error Analysis: Label Frequency in training data vs f1-performance of the system.

Pretraining	F1 Validation	pre-training	weighted-loss	epochs*	optimized	LR	batch
IBM-Roberta-large	**.529**	(+)		20	F1	2e-5	8
IBM-Roberta-large	.526	(+)	(+)	20	F1	2e-5	8
Roberta-large	.519			20	F1	2e-5	8
T5-large	.493			35	F1	.001	16

Table 7: Results of other approaches: Values are calculated without tuned hyperparameters. This training was at the beginning of the competition where instead of applying 3-fold-cross validation we took a validation sample of 500 and trained the model with 3 different random seed initialization. For weighting the loss function we used "Inverse Number of Samples"[12] as weights. The epochs are not actually the number of trained epochs. Instead they are used to calculate the linear learning rate schedule by calculating the Total_training_steps = (len(training_data)//Batch_Size)*Epochs. The models are then trained with early stopping.

B.3 Internal Leaderboard for Ensembling

Table 8 compares the performance of the different ensembles. Based on the f1-score performance on an internal test-split we decided to submit the system with 12 models.

Model Selection	#	Thres.	F1 Test	F1 inter.
EN-Max-F1*	6	.26	.555	.596
EN-Thres-LoD (1st)	12	.26	.561	**.599**
EN-Deberta-F1*	3	.27	**.566**	.589
Single-Deberta-F1*	1	.25	.554	.565

Table 8: Ablation Studies: Scores on official test set (F1-Test) and scores for internal test split. For Single-Deberta-F1 the Model with Random Seed = 123 was selected. # represents the number of models in ensemble. The models with their identifier are listed in Table 9 in the Appendix

B.4 Error Analysis

We plotted the frequencies of the labels in the training data against their f1-score performance (Figure 4)

B.5 Other Approaches

We provide the results of some different approaches. They have been calculated on the same validation set of 500 samples, but we initialized the training with three different random seeds. Table 7 contains different methodologies.

B.6 Model Mapping - Ablation Studies

Table 9 shows the identifier of the models in the model repository together with optimization goals and random seeds. The Table also shows which model is included in the Ensembles in the Ablation Studies.

Model Mapping	model	Fold-Seed	Optimized	EN-Max-F1*	EN-Deberta-F1*	Single-F1*
HCV-406	deberta	42	F1	(+)	(+)	
HCV-408	deberta	96	F1	(+)	(+)	
HCV-409	deberta	123	F1	(+)	(+)	(+)
HCV-402	danschr-roberta	42	F1	(+)		
HCV-403	danschr-roberta	96	F1	(+)		
HCV-405	danschr-roberta	123	F1	(+)		
HCV-364	deberta	42	Loss			
HCV-366	deberta	96	Loss			
HCV-368	deberta	123	Loss			
HCV-371	danschr-roberta	42	Loss			
HCV-372	danschr-roberta	96	Loss			
HCV-375	danschr-roberta	123	Loss			

Table 9: The models are publicly available (Appendix A). The danschr-roberta represents the "IBM-Roberta" in the paper. The first column identifies the model in the model repository.