Human Values Detection for Touché at SemEval 2023
NLP Course Project

First Student, Second Student, Third Student and **Fourth Student**
Master's Degree in Artificial Intelligence, University of Bologna
{ alessio.bonacchi, marco.guerra20, alessio.cocchieri2, riccard.grassi5 }@studio.unibo.it

Abstract

The aim of this project is to provide an efficient solution to *Human Value Detection* problem, achieved by implementing several models in order to exploit their peculiarities and find the best approach. The complete list of the model we used is: XLNet-large, XLNet-base,BERT-large, BERT-base, roBERTa-large, roBERTa-base, distilBERT and SVM.

XLNet-large and roBERTa-large allowed us to obtain the best results compared to the other models, both obtaining 0.5 as macro-averaged F1-Score. The other models perform a little bit worse though. The worst one turned out to be SVM which reached a score of 0.37. Nonetheless, all our models perform way better than the ones defined in the original paper[1]

1 Introduction

According to the literature[1], "people have different beliefs and priorities of what is generally worth striving for (e.g., personal achievements vs. humility) and how to do so (e.g., being self-directed vs. respecting traditions), often referred to as *human values*" (see Figure 1).

In our research we dealt with *Human Values Detection* task, which is a multi-label classification task, that given a textual argument it is expected to assign to it one or more of the 20 values categories. The arguments are composed of a *premise*, a *conclusion* and a *stance*, which stated if the premise is *in favor of* or *against* the conclusion.

The literature[1] proposes two approaches to solve the problem; the first one, based on a Support Vector Machine (SVM), achieved a 0.30 macro-avg F1 score on the 20 labels classification task, while the second one, a fine-tuned multi-label bert-base-uncased model, managed to obtain a 0.34 macro-avg F1. With our approach we achieved results that increases the ones obtained in the paper[1] by more than 20% for both the models, with our SVM and

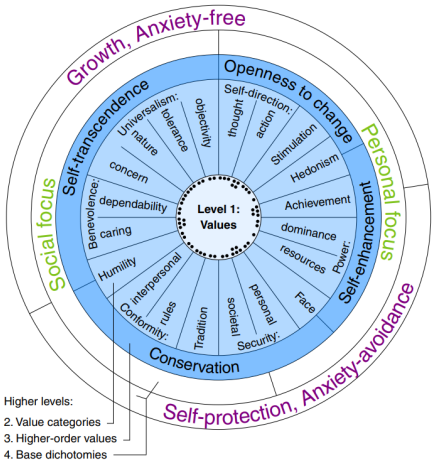

Figure 1: Human values

BERT-base getting 0.37 and 0.42 respectively. In addition to this, we also trained the bigger version BERT-large, which got a score of 0.44, improving the score of its 'base' model. Alternatively to BERT, we have also decided to use roBERTa, which manages to score 0.47 and 0.5 in macro-avg F1, for the base and large version respectively. Since it offers faster inference speed we also decided to test distilBERT: even if it obtained a macro-avg F1 score of 0.43, it showed off good performances with respect to the others, considering the fact that it is a much simpler network. We gained the best result (0.5 macro-avg F1) with opportunely fine-tuned XLNet-large and roBERTa large models, improving of more than 47% the best result achieved in the paper[1]. For completion sake, also the base version of XLNet was implemented and tested, scoring a solid 0.44 on macro-avg F1. All the scores refer to the evaluation made assuming the validation data (*validation.tsv*) as test set. Multiple run of training with different seeds (42, 2022, 1337) have been done in order to check the stability of the models. Since the results are stable across all seeds, we display only the results corresponding to seed 42 for readability sake.

2 Background

All social sciences are concerned one way or another with human values. A value[3] is described as a belief pertaining to desirable end states or modes of conduct, effectively resulting in a modality of selective orientation. Such a definition allows a systematic analysis by attributing values to persons instead of objects. As a result we can try to unveil personal values supporting arguments, even implicitly.

In the Touché Human Value Detection Challenge 2023 [4], Natural Language Processing is applied to values for argument mining. Human Value Detection is formulated as a classification task: a premise, a stance and a conclusion are the input to a multi-label classification problem. Labels account for the priorities that have been identified to generally guide people choices (tolerance, objectivity, humility and others). Although some values may be incompatible or opposite, they are generally not mutually exclusive, this is the reason why the task is multi label. We decided to test three different architectures, of increasing complexity.

We started with a SVM approach. Support Vector Machines (SVM) are still a powerful tool to initially tackle a classification problem, also thanks to their resistance to the overfit phenomenon. The problem is formulated as a series of binary classification problems: this approach is called one-vs-rest strategy and it consists in fitting one classifier per class against all others with the addition of a binary relevance method to perform multi-label classification, resulting in a methodology that is both computationally efficient and interpretable.

SVMs are not specialized for NLP: for this reason, we tried to increase the complexity of the approach by testing out different BERT pre-trained models, which consist of Transformers models specifically pre-trained on NLP tasks. For our specific purpose, a classification head composed of a dropout layer and a linear layer allows us to fine-tune the models on the multilabel classification problem. We compare two different Transfomers models: BERT, which performs masking once during data preprocessing, and RoBERTa, which instead performs masking at training time. BERT relies in fact on randomly masking and predicting tokens. The BERT strategy results in a single static mask, that is outperformed by the use of a dynamic masking strategy where we generate the masking pattern every time we feed a sequence to the model [5]. We compare two variants of each of these models, the standard ($\sim 110/120$ million parameters) and the large versions (~ 350 million parameters). We also compare these with a smaller version of BERT: distilBERT, which has around 40% less parameters and runs 60% faster while preserving 95% of the performance.[6]).

Finally, we fine-tuned an implementation of XL-Net, which tends to perform better than BERT in a number of different tasks, since it overcomes some of the disadvantages that come with the use of mask tokens in the BERT pre-training process. Mask tokens, which allow BERT models to use the whole forward and backward context of the sentence, are not present in the fine-tuning phase: this creates a discrepancy between the pre-training and fine-tuning in BERT. XLNet manages not to use mask tokens, constraining the context word to two directions, either forward or backward, but is still able to use the whole context of the sentence through Permutation Language Modeling.

3 System description

3.1 SVM

For the SVM model, pre-processing has been carried out using Natural Language ToolKit (NLTK). Out of the *premise*, *conclusion* and *stance* columns, we are only interested in processing the *premise*, since the other two don't add any information with respect to the premise, given the SVM preprocessing and training. The pre-processing phase transforms all words in lowercase, removes all characters that are not letters, then deletes all punctuation symbols and stop-words from the premise. The stop-words are retrieved from NLTK's *corpus* module. After this first stage, WordNet lemmatizer is applied, resulting in essentially a list of words (that are not stop-words) in the order that they appeared, separated by a space, for every record.

The pre-processing output is then fed into a Support Vector Classification Scikit Learn Pipeline: we apply TF-IDF vectorization from Scikit Learn as a first stage, followed by an intermediate step of dimensionality reduction involving Truncated Singular Value Decomposition (*TruncatedSVD*). At the end of the pipeline, a OneVsRestClassifier wraps the SVC module to allow for multilabel classification. By setting the *class weight* SVC parameter, we try to overcome the fact that the classes in the dataset are heavily unbalanced, using weights

that are inversely proportional to class frequencies. We evaluate the model using Classification Report from Scikit Learn.

3.2 BERT models

The different BERT models are all loaded through Huggingface Transformers library using PyTorch. All our processes use CUDA. For each model, we used the corresponding tokenizer provided by Huggingface. For the pre-processing, the *premise*, *stance* and *conclusion* columns are concatenated (separated by a white space), since redundancy may improve the result of the Transformers model, especially through the use of attention. The resulting dataframe is used to instantiate a *CustomDataset* class that we defined, which generates tokenized outputs and tags used during training. The tokenizer uses the *encode plus* method to perform tokenization and generate the necessary outputs: *ids*, *attention mask* and *token type ids* (except for Distilbert, which does not use the token type). The class can generate a Training Dataset used to fine tune the model and a Validation Dataset to evaluate the performance of the model.

For the training phase, a Dataloader is employed. The model to train is wrapped by a custom class that adds a classification head to the pre-trained model, composed by a Dropout layer with *p* parameter set to 0.3 and a Linear layer with 20 output channels. They respectively account for regularization and multilabel classification. The final Linear layer is what is used to calculate the loss and to determine the accuracy of model prediction. As loss function, we use *BCEWithLogitsLoss* from PyTorch, a loss function that combines a Sigmoid layer and the BCELoss in one single class. This version is more numerically stable than using a plain Sigmoid followed by a BCELoss. Before the evaluation of the model, we iterate a number of different thresholds to apply to the model output to select the classes to assign to the input, and select the one giving the highest F1 macro score. We evaluate the model using Classification Report from Scikit Learn.

3.3 XLNet

The two XLNet models are all loaded through Huggingface Transformers library using PyTorch. For each model, we used the corresponding tokenizer provided by Huggingface. For the pre-processing, the *premise*, *stance* and *conclusion* columns are concatenated, since redundancy may improve the result of the model, especially through the use of attention. A tokenizing function was produced in order to tokenize the text, truncate to the desired length and convert into padded numeric sequences of ids. A special function was produced to create attention masks that ignore padding tokens.

A special class was also produced to add a Linear layer with output dimensionality 20 to serve as classification head. We have to specify that in XLNet Large we also re-initialized the last three layers (which correspond to the pooling layers and the last Transformer layer). We created training and test Dataloaders wrapping a SequentialSampler. As loss function, we use *BCEWithLogitsLoss* from PyTorch. Like in the BERT models, we iterate different thresholds to find the best one and complete evaluation using Classification Report from Scikit Learn.

4 Data

The datasets we used are those provided by SemEval competition, available in Zenodo (link in Section 8). In particular we used *arguments-training.tsv* and *labels-training.tsv* for training, and *arguments-validation* and *labels-validation.tsv* for testing. Here we provide a brief architecture of the above datasets:

- *arguments-<split>.tsv* :
  - Argument ID: The unique identifier for the argument
  - Conclusion: Conclusion text of the argument
  - Stance: Stance of the Premise towards the Conclusion; with values "in favor of" or "against"
  - Premise: Premise text of the argument
- *labels-<split>.tsv* :
  - Argument ID: The unique identifier for the argument
  - Other: Each other column corresponds to one value category (among 20 available), with a 1 meaning that the argument resorts to the value category and a 0 that not

Since the categories for the test splits were not provided we only considered train and validation splits to conduct our research. The train set has 5393 arguments, while the test set only has 1896 records.

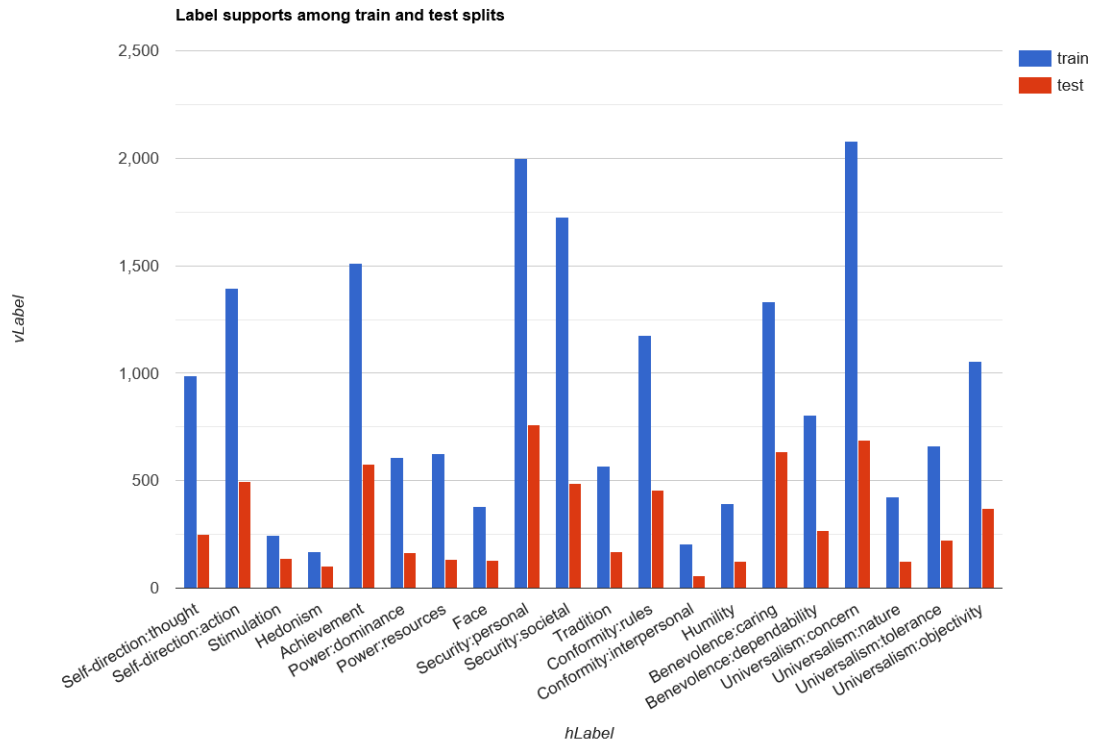

Figure 2: Supports for the different labels

We also noticed that the number of instances for each label is not uniform 2, resulting in classes having a very high support and classes having a very low one. *Conclusion*, *premise* and *stance* arguments are merged into one sentence labelled as *text*.

We statistically analysed the length of *text* in order to not truncate the input in most of the cases, having the maximum length at 170; By using percentiles we discovered that with 97% and 99% percentile we have maximum length of 75 and 100 correspondingly (Figure 3), so we decided to use them as *MAX_LEN* parameter for our models. In particular we noticed that XLNET works better with longer sentences, then we decided to use 100 as maximum length for it. No preprocessing was done for transformers since they take advantage of sentence semantics and even stopwords and punctuation come in handy in order to leverage the multi head self attention mechanism. For SVM we do not encounter this problem since we only select the *premise*: this because *conclusion* does not add any additional information as it repeats the same words in *premise* and we know that SVM does not take

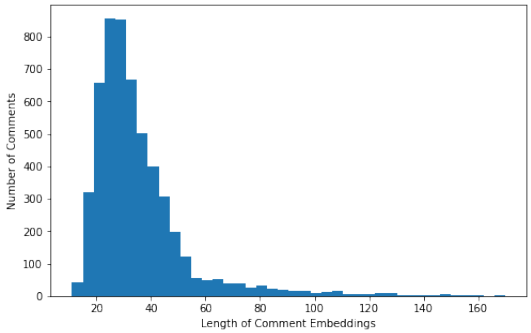

Figure 3: Distribution of the input lengths

any benefits from semantic and then adding more useless information it could results in worst performances. Other than this, in order to prevent bad results, we also preprocessed the text by removing punctuation, stop words and bad symbols, along with a lemmatisation step.

5 Experimental setup and results

All the evaluation metrics are provided by Scikit Learn Classification Report, which returns model precision, recall and F1 calculated in average and across the classes.

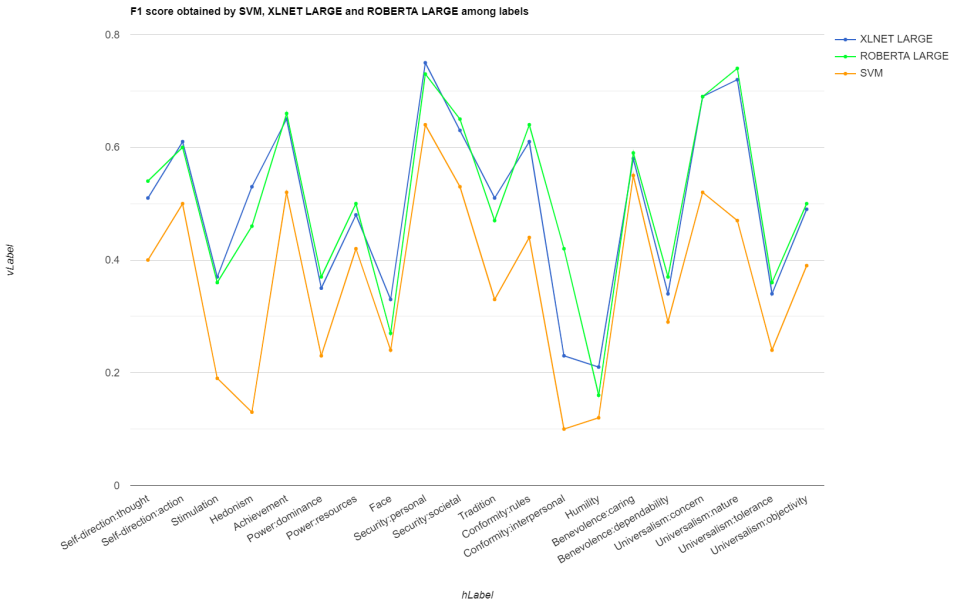

Figure 4: Comparison between different models across the 20 labels

Model	Test F1	Training Iterations	Batch Size
bert-base	0.42	4 epochs	8
bert-large	0.44	4 epochs	8
roBERTa-base	0.47	8 epochs	8
roBERTa-large	0.5	5 epochs	8
distilBERT	0.43	12 epochs	16
xlnet-base	0.44	3 epochs	16
xlnet-large	0.5	3 epochs	16
SVM	0.37	10000 max iters	None

Table 1: Macro-avg F1 score on test set, all models

Regarding the input vectorization for the SVM model, the vocabulary considers only the top 5000 features ordered by term frequency across the corpus. This is done by setting the *max features* parameter. For *TruncatedSVD* we set the output data dimensionality to 300. The SVC classifier has been trained with a regularization parameter of 18 and a Radial Basis Function Kernel of coefficient 0.01 (both the kernel and the coefficients have been selected on the basis of performance). The maximum limit for iterations within solver is 10000.

The input tokenizers for the BERT models all use a model maximum length of 75, which stands at around the 97th percentile of the input lengths distribution. The Dataloader batch size used is 8, for the only exception of distilBERT that uses a batch size of 16. We train BERT-base and BERT-large models for 4 epochs, as recommended by BERT authors[7], while we used different epochs for roBERTa and distilBERT since we saw that increasing the number of epochs would not fall in overfitting, as shown in Table 1. The training process runs an Adam optimizer with an initial learning rate of 2^{-5}. We also used a linear scheduler with warmup (*get_linear_schedule_with_warmup*) for roBERTa and distilBERT models since we managed to obtain higher scores than without it. Not all models benefit from the addition of the scheduler.

About XLNet models input tokenization, the tokenizer model maximum length is set to 100. This value stands at around the 99th percentile of the input lengths distribution. Dataloader batch size used is 16, and the models are trained for 3 epochs, using AdamW as optimizer with initial learning rate 2^{-5} and weight decay 0.01 with no bias correction. We decided to not perform more than 3 epochs since, as shown in figure 5 for XLNet-Large, after 2 epochs the model starts to overfit considerably.

By increasing the approach of the complexity, we get better results accordingly: XLNet-large and roBERTa-large are the models that perform better, followed by the other BERT models and SVM. Size

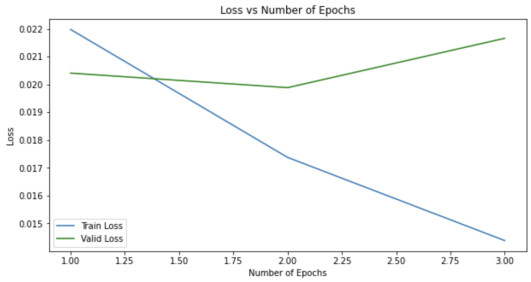

Figure 5: XLNet: Train loss vs Validation loss

of the model also contributes to the performance: all the large versions perform better than their corresponding base versions, and distilBERT performs better than the base version of BERT.

6 Discussion

At the end of the experimentation, we noticed that XLNet-large and roBERTa large, with a macro-avg F1 score of 0.5, gave us the best performances over all other models, while SVM performed the worst with a score of 0.37. We expected that SVM's results to be lower than the other model, since SVM do not consider semantic, unlike *Transformers* models. Nevertheless, even being the simplest model, it almost reached the performances of BERT-base model, decreasing its result by 12%. Among the *Transformers*, surprisingly BERT base obtained the poorest scores. Even distilBERT, managed to score better than BERT base, being a less complex model: maybe we can explain this behaviour with the fact that being simpler and given the fact that we do not have a huge amount of data it helps to generalise well, in the hope of preventing overfitting. It was used in order to see if a less complex model could manage the task, and with respect to the other performances it establishes a solid score from which starting to improve.

Also roBERTa-large gave optimal results improving over the performances of both XLNet-base and BERT-large by 13% with a macro-avg F1 score of 0.5. As we expected "large" versions of the models outperformed the correspective "base" ones, being simply bigger in size; also we observed that both XLNet and roBERTa models obtain results better than BERT, according to the fact that they were designed to improve BERT performances over multiple NLP tasks. Talking about results obtained in Figure 6, it can be observed that roBERTa-large generally outperformed the other BERT models on almost every label, and it's results appear much more stable: while other models show a wider variability in results (which contribute to lowering the macro-avg F1 score).

In figure 4 we reported all scores obtained by SVM, XLNet-large and roBERTa-large. We choose the last two ones because they represent the 2 *Transformer* models that performs better on this task. As we can see, roBERTa-large and XLNet-large outdo the performances of SVM for every label. The performances of the first 2 are pretty similar, with roBERTa giving better results on some labels (i.e. *Conformity: interpersonal*) and XLNet on some others (i.e. *Hedonism*) .Talking about the trend of the results we can see that a great role is played by the number of supports for each class. In general, comparing 2 and 4, is pretty straightforward that the F1 increases with the increase of the support and it decreases once the support gets lower. As an example we can just look at *Security:personal* and *Universalism:concern*: they both are the labels that have the highest support and also two of the highest F1 scores. On the contrary, labels like *Conformity:interpersonal* show low support and thus got low F1 scores. Although we found some anomalies: *Universalism:nature* gets pretty high F1 scores even if it has low support and *Universalism:tolerance*, which has good support, gives poor results instead.

In order to answer this doubts we decided to analyse the TF-IDF scores among all classes for the 15 most frequent words of a single class(see 7), with the function *TfidfComparison* (in SVM notebook). Taken as example *Universalism:nature*, we can see that the TF-IDF values for its 15 most frequent words are high and such scores can not be found in any other class, given the same word. This means that this words make the model able to easily discern this label from the others. For the same reason, with *Universalism:tolerance* we are experiencing the opposite thing: the reason for which the F1 score is low beside the high support could be addressed by the fact that its 15 most frequent words have high TF-IDF scores also in other classes, then making it less separable. For completion sake we also reported the TF-IDF scores for *Security: personal* in order to prove that high TF-IDF scores usually coincide with high F1 scores.

We also conducted another error analysis study using *multi label confusion matrices* provided in *mlcm* library. As stated in the paper[2], "*it creates a confusion matrix that, similar to the multi-class*

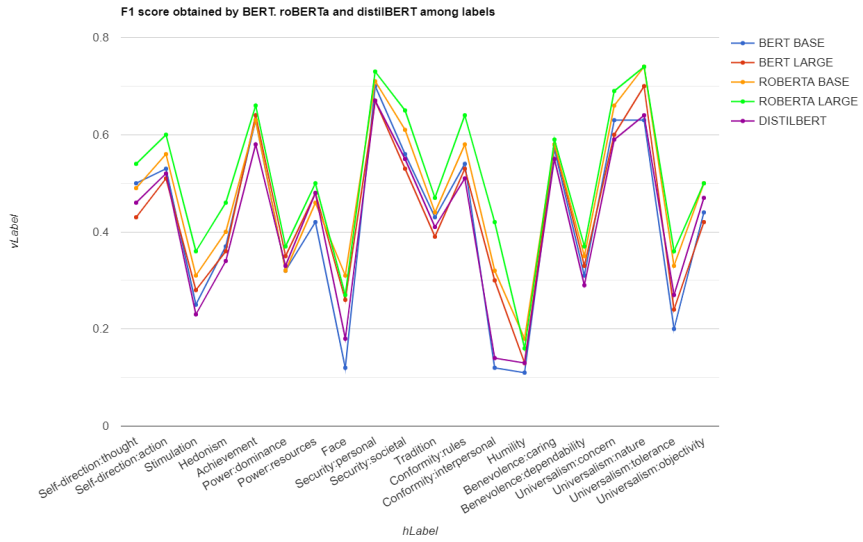

Figure 6: Comparison of different BERT models across the 20 labels

ColumnName	good	harm	brings	people	banned	foster	care	loan	social	child	payday	medium	homeopathy	education
Self-direction: thought	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000227	0.000017	0.000111
Self-direction: action	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000015	0.000418	0.000000	0.000000	0.000383	0.000154	0.000025	0.000205
Stimulation	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000100	0.000000	0.000000	0.000000	0.000000	0.000515	0.000000	0.000301
Hedonism	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000072	0.000000	0.000105
Achievement	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000049	0.000079	0.000000	0.000000	0.000053	0.000117	0.000038	0.000323
Power: dominance	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000033	0.000000	0.000000	0.000000	0.000000	0.000113	0.000055	0.000044
Power: resources	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.002291	0.000000	0.000000	0.002004	0.000000	0.000000	0.000700
Face	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000038	0.000000	0.000000	0.000000	0.000000	0.000465	0.000000	0.000094
Security: personal	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000326	0.002168	0.000000	0.000000	0.001978	0.000524	0.000253	0.000232
Security: societal	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000034	0.000030	0.000000	0.000000	0.000030	0.000174	0.000008	0.000093
Tradition	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000052	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000026
Conformity: rules	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000028	0.000355	0.000000	0.000000	0.000323	0.000382	0.000000	0.000033
Conformity: interpersonal	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.001044	0.000000	0.000000
Humility	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000283	0.000000	0.000000	0.000283	0.000418	0.000041	0.000102
Benevolence: caring	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000156	0.000407	0.000000	0.000000	0.000373	0.000260	0.000019	0.000132
Benevolence: dependability	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000057	0.000532	0.000000	0.000000	0.000532	0.000196	0.000019	0.000201
Universalism: concern	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000046	0.000296	0.000000	0.000000	0.000275	0.000094	0.000006	0.000079
Universalism: nature	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000013
Universalism: tolerance	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000312	0.000043	0.000054
Universalism: objectivity	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000033	0.000065	0.000000	0.000000	0.000065	0.000220	0.000028	0.000112

Figure 7: TF-IDF scores for most frequent terms of "Security: personal"

(single-label) confusion matrix, shows the distribution of FNs from one class over other classes". It is an interesting tool since it provides the distribution of false negatives for each label thus showing with which other classes the class of our interest is usually misclassified. As a result, we could compare the TF-IDF analysis with this one in order to discover if high TF-IDF scores in other classes would result in a misclassification. It is a little bit tricky to compare but you can observe that classes that have high TF-IDF scores for the most frequent terms of the target label would be more prone to be misclassified.

7 Conclusion

For this project we managed to handle the *Human value detection* task, implementing and testing several different approaches in order to obtain the best performances. We are satisfied with the results we obtained since we managed to improve the overall F1 score in the paper[1] by 47% with our best model. It was also interesting to observe that SVM, though being the simplest model, have been able to achieve very promising results. We observed that performances are mostly dictated by the different distribution of support among labels, and the analysis over the TF-IDF scores and multilabel confusion matrix were a key factor for investigating the errors produced by the models. According to this, possible future improvements can be obtained by simply increase the number of instances in the dataset and balancing their distribution among the labels.

8 Links to external resources

- Touché23-ValueEval on Zenodo

References

1 Johannes Kiesel, Milad Alshomary, Nicolas Handke, Xiaoni Cai, Henning Wachsmuth, and Benno Stein. Identifying the human values behind arguments. In Proceedings of the 60th Annual Meeting of the As- sociation for Computational Linguistics (Volume 1: Long Papers), pages 4459–4471, Dublin, Ireland, May 2022. Association for Computational Linguis- tics. doi: 10.18653/v1/2022.acl-long.306. URL https://aclanthology.org/2022.acl-long.306.

2 M. Heydarian, T. E. Doyle and R. Samavi, "MLCM: Multi-Label Confusion Matrix," in IEEE Access, vol. 10, pp. 19083-19095, 2022, doi: 10.1109/ACCESS.2022.3151048.

3 [Rokeach [Milton Rokeach. 1973. The nature of human values. New York, Free Press.]]

4 URL: https://touche.webis.de/semeval23/touche23-web/index.html#synopsis

5 URL: https://arxiv.org/abs/1907.11692

6 URL: https://huggingface.co/docs/transformers/model_doc/distilbert

7 URL: https://github.com/google-research/bert