Human Value Detection NLP Course Project

Ana Slovic, Tony Chahoud, Davide Tarabelloni and **Shadi Farzankia**
Master's Degree in Artificial Intelligence, University of Bologna
{ ana.slovic, tony.chahoud, davide.tarabelloni, shadi.farzankia }@studio.unibo.it

Abstract

In this report, we address the challenging task of multilabel classification for human value detection, leveraging state-of-the-art natural language processing (NLP) techniques. We explore the effectiveness of three prominent models named: XLNet, SVM, and BERT, and the evaluation was conducted using robust metrics such as precision, recall, F1-score, and accuracy. In this multilabel classification task, XLNet emerged as the standout performer, consistently achieving higher scores compared to other models. Notably, SVM exhibited comparatively poorer performance. Specifically, XLNet achieved an F1 score of 0.44, while SVM and BERT attained F1 scores of 0.41 and 0.21, respectively.

1 Introduction

Human values refer to those that are at the core of being human, and they bring out the fundamental goodness of humans and society at large. Understanding and recognizing these values is crucial since they provide an understanding of attitudes, motivations, and behaviors, as well as influencing our perception of the world around us.
There are various ways that people define human values, but these definitions can change based on the situation or society. Additionally, it's quite challenging to keep track of all these definitions due to their sheer number. Many value schemes are offered by different researchers. Our project is strongly based on the (Schwartz, 2012) 's theory of basic human values. The theory discusses the nature of values and spells out the features that are common to all values and what distinguishes one value from another.
The objective of our work is to classify, given a textual argument and a human value category, whether or not the argument can be drawn from that category. This task uses a set of 20 value categories compiled from the social science literature and described in paper (Schwartz, 2012), which is the first paper focusing on this task computationally.
In our attempt to tackle the intricate challenge of multiclassification, we harnessed the power of three distinct models: Support Vector Machines (SVM), BERT, and XLNET. SVM is a traditional machine learning algorithm focusing on separating classes using hyperplanes. BERT (Devlin et al., 2019) and XLNet (Yang et al., 2020) are cutting-edge language models designed for natural language tasks, with XLNet addressing BERT's limitations by enhancing context understanding. While SVM requires explicit feature engineering, BERT and XLNet learn features automatically. BERT and XLNet generally outperform SVM on complex language tasks due to their advanced architectures and training methods, with XLNet potentially surpassing BERT in certain scenarios due to its unique approach. Each of these models brought unique strengths to the table, offering diverse approaches to understanding and classifying complex textual data. However, we recognized that the efficacy of these models could be further enhanced by fine-tuning their hyperparameters to precisely fit our task requirements.
To achieve this fine-tuning, we integrated the Ray Tune and W&B Sweeps libraries into our workflow. Their optimization capabilities allowed us to systematically explore a wide range of hyperparameter configurations for each model and achieve optimal performance for a range of complex models. We used W&B Sweeps to automate hyperparameter search and visualize rich, interactive experiment tracking for our transformers. Besides, we used ray tune to fine tune our SVM model. Ray Tune is another open-source library specializing in hyperparameter tuning and distributed computation. One of the standout features of Ray Tune is its ability to distribute the optimization process across multiple CPU or GPU resources, enabling parallel and distributed tuning. This results in faster hyperparameter searches, particularly for models

that require extensive computational resources. We strategically employed Ray Tune and W&B Sweeps based on the complexity of the models we were dealing with. For the less advanced models like SVM, we harnessed the power of Ray Tune. This choice made sense because Ray Tune excels in making the hyperparameter tuning process smoother, particularly for models that aren't as complex. It simplified the optimization process for our SVM, letting us quickly find the right settings for optimal performance.

On the flip side, we turned to W&B Sweeps when working with heavy hitters like BERT and XLNet. These models come with intricate architectures that require careful tuning.

2 Background

In our study, we addressed the Touché Human Value Detection Challenge 2023, employing Natural Language Processing to delve into values within argument mining. The Human Value Detection task is structured as a classification challenge, involving a premise, stance, and conclusion as inputs for a multilabel classification scenario. These labels encompass prioritized attributes that commonly influence people's decisions, such as tolerance, objectivity, and humility. While certain values may seem incompatible or opposing, they generally coexist rather than being mutually exclusive, forming the basis for the task's multi-label nature. To explore this, we employed three distinct architectures, each progressively more intricate which are SVM, BERT, and XLNet.
Our first selection was SVM. Support Vector Machine (SVM) is a versatile machine-learning algorithm designed primarily for classification tasks. It excels in distinguishing between different classes by identifying an optimal decision boundary, often referred to as a hyperplane, that maximizes the margin between classes while minimizing classification errors. SVM's strength lies in its ability to handle both linear and non-linear classification challenges, making it an essential tool in various domains.
To refine and optimize our SVM model, we employed the Ray Tune library, a powerful tool for hyperparameter tuning and distributed experimentation. Ray Tune simplifies the process of finding the best set of hyperparameters by automating the exploration of different parameter configurations and efficiently leveraging parallel computing resources.
To address the limitations of traditional methods, we employed deep learning models like BERT and XLNet. BERT captures contextualized word representations, whereas XLNet leverages permutation-based training to capture bidirectional context. To use these models, we got help from the Simple Transformers library which simplifies their implementation. Using Simple Transformers, training BERT and XLNet is streamlined, enabling easy adaptation to the Human Value Detection task.
Afterward, to select the best hyperparameters for our model, we got help from the sweep library. It is a tool provided by Weights & Biases (wandb) that simplifies the process of hyperparameter optimization and fine-tuning for machine learning models. It automates the search for optimal hyperparameters by exploring different combinations and values in a systematic manner. The Sweep library is particularly useful for deep learning models, such as neural networks, where tuning hyperparameters can significantly impact performance.

3 System description

3.1 SVM

In our SVM approach, we initiated the process by conducting comprehensive data preprocessing. To streamline the data and enhance the efficiency of subsequent steps, we exclusively utilized the content from the Premise and Stance columns, as the information in the Conclusion column proved to be redundant. The initial phase encompassed various cleaning procedures, such as converting all words to lowercase, removing punctuation marks, hyperlinks, and stop words(excluding stop words having negative meanings such as 'not', 'no', 'against', etc.), and excessive white spaces. The resulting text was then tokenized to facilitate further analysis.
Having prepared the text data, we harnessed the power of the TF-IDF library to transform the textual content into a meaningful numerical representation. Following this, we employed the StandardScaler to normalize the features, ensuring they share a standard scale and distribution.
Subsequently, we recognized the need to reduce the dimensionality of the datasets to achieve optimization. To accomplish this, we turned to the TruncatedSVD library, a powerful tool for dimensionality reduction. To determine the ideal number of components for dimensionality reduction, we selected values that resulted in a cumulative explained variance of 95

In handling the labels, we utilized the MultiLabelBinarizer library. This facilitated the efficient transformation of our multilabel dataset into a suitable format for our model. We continued building our model by using Scikit-learn's Support Vector Classification (SVC). The Support Vector Machine (SVM) is a well-known machine learning algorithm primarily designed for binary classification tasks. However, to employ it as a multi-label classifier, where the model predicts multiple classes for a single input sample, we employed a multi-output classifier approach. This method entails encapsulating the SVM classifier and training it on several distinct outputs simultaneously. To find the best hyperparameters, we relied on the Ray Tune library, which helped make the optimization process smoother. In our journey to optimize the model, we experimented with training SVM kernels using different degrees. We separated polynomial kernels from sigmoid and radial basis function (RBF) kernels. Surprisingly, our experiments revealed that sigmoid kernels performed better for our unique model compared to both polynomial and RBF kernels.

3.2 BERT and XLNET

In our project, we've taken a well-organized approach to train and evaluate BERT and XLNet models. To simplify the implementation of these models, we've relied on the Simple Transformers library, a fantastic tool for Natural Language Processing (NLP) tasks. It's worth noting that Simple Transformers builds upon the impressive work done by Hugging Face and their Transformers library.
One important decision we made was to skip extensive data preprocessing. We found that BERT and XLNet models are highly capable of handling raw text data efficiently. This choice has streamlined our workflow, allowing us to direct our attention to other critical aspects of model development and assessment.
Due to computational limitations in our system, we made the practical decision to concentrate on training and assessing the performance of two specific models: bert-base-cased and bert-large-based. We employed a similar approach for XLNet models as well. This focused approach helped us make the most of our available computational resources.
For optimizing model performance, we've used the power of the sweep library. This tool has empowered us to explore a range of hyperparameter configurations efficiently. Our systematic exploration of hyperparameters has played a crucial role in fine-tuning our models.
To ensure a fair and meaningful comparison among different model variations, we've applied the best-performing hyperparameters to both the base and large versions of each model. This uniform evaluation setup enables us to accurately assess which model variant excels in our specific task. In terms of evaluating model performance, we've chosen the F1 score as our primary metric. The F1 score provides a balanced assessment of precision and recall, making it particularly suitable for tasks involving imbalanced datasets. Additionally, we've employed the classification_report function to generate a comprehensive evaluation report. This report includes various F1 score averages, such as micro and macro, offering a more nuanced analysis of results.

4 Data

To execute the project, we leveraged the datasets represented by (Kiesel et al). This dataset is structured into three distinct segments: the training set, the validation set, and the test set. Each set is comprised of two separate files: one for arguments and another for labels. Specifically, the training dataset encompasses 5,393 instances, the validation dataset contains 1,896 instances, and the test dataset comprises 1,578 instances. Each data entry possesses a unique argument ID, facilitating the straightforward differentiation of its corresponding label.
Our argument dataset includes the 'Conclusion', 'Stance', and ' Premise' columns. The Stance indicates whether the Conclusion is in favor of or against the Premise. In our training procedure, we omitted the 'conclusion' column as it does not convey any extra information about the labels and it is mostly repeated in the 'features' column.
Within our argument dataset, there are 'Conclusion,' 'Stance,' and 'Premise' columns. The 'Stance' column indicates whether the 'Conclusion' aligns favorably or unfavorably with the 'Premise.' In our training process, we opted to exclude the 'Conclusion' column. This decision was based on the observation that it doesn't provide supplementary insights into the labels and is frequently redundant within the 'features' column.
While all models benefit from preprocessing, the nature of preprocessing varies. For SVM,

it's more about feature engineering, scaling, and outlier handling. For BERT and XLNet, it involves tokenization and masking. Proper preprocessing is essential for obtaining meaningful results and optimizing the performance of both approaches. For the reason of optimality, we merged the columns 'Stance' and 'Premise' into another feature called 'Context'.

4.1 SVM

For the Support Vector Machine algorithm, the preprocessing steps for argument data are the following procedures:

- Removal of all stop words, excluding those devoid of negative connotations
- Conversion of uppercase words to lowercase for uniformity
- Elimination of hyperlinks, punctuation marks, excessive white spaces, and digits
- Tokenization of the remaining words

Following the aforementioned transformations, the arguments are then subjected to vectorization using the TF-IDF library. The TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer is a well-established technique in natural language processing and text analysis. It transforms a collection of text documents into numerical vectors, with each vector representing a document's content through TF-IDF values assigned to its words. Subsequently, the obtained dataset from TF-IDF undergoes normalization using the 'StandardScaler.' However, due to the predominantly sparse nature of the resulting TF-IDF matrices, the application of TruncatedSVD for dimensionality reduction is advisable. This keeps things running smoothly and tackles the problems that pop up when dealing with lots of different factors. We figured out how many components to use for TruncatedSVD based on how much of the total variance they explained. We picked the number of dimensions that added up to a big chunk of the variance, like 95% in our case. The result was that we settled on using 710 components.

4.2 BERT and XLNet

Regarding the two remaining pre-trained language representation models, no preprocessing was undertaken due to their inherent ability to leverage the entirety of sentence information. During our experimental phase, we set the maximum sequence length to 100, a parameter established after assessing the lengths of 99 percent of the input instances. This a a pivotal processing step that involves padding inputs of shorter length and truncating those that are excessively long. These truncation and padding steps are acquired with two distinct tokenizers: the BERT tokenizer and the XLNet tokenizer. The primary distinction between the BERT tokenizer and the XLNet tokenizer lies in their underlying tokenization strategies and how they handle special tokens and self-attention. BERT uses WordPiece subword tokenization and masked self-attention with special tokens, while XLNet employs whole-word tokenization and permutation-based self-attention without special tokens.

5 Experimental setup and results

5.1 SVM:

Given that the input for the StandardScaler is a sparse matrix (generated through the TF-IDF process), we set the 'with_mean' parameter to False. This prevented the scaler from centering features by subtracting their mean, as such an operation is unnecessary with sparse data. In addition, to establish the number of components for dimensionality reduction, we opted for the minimum number of components that yielded a cumulative explained variance of 0.95. The specific number we arrived at was 710. As previously mentioned, we employed various hyperparameters to train our model, aiming to identify those that yielded the highest F1 score. In this endeavor, we harnessed the capabilities of the Ray Tune library, a robust tool for hyperparameter tuning and distributed experimentation in the realm of machine learning. Ray Tune brings a multitude of advantages that substantially enhance the efficiency and efficacy of our model training and hyperparameter optimization endeavors. These benefits encompass scalability, parallelization, resource management, and more. When dividing the SVM models into two configurations, the range of hyperparameters for the first configuration, which includes the polynomial kernels, was defined as follows:

- **C:** The regularization parameter C which controls the trade-off between maximizing the margin between classes and minimizing the classification error ranges from **2e-3 to 2e3**.

- **Gamma:** The gamma parameter which determines the influence of individual training samples on the decision boundary ranges from **2e-3 to 2e3**.

It's noteworthy that the same range of parameters (C and gamma) was chosen for the polynomial kernels as well. This symmetrical parameter range allows for a fair and direct comparison of the performance between models using different kernels, ensuring an equitable evaluation of their capabilities. However, we added another parameter called the degree parameter determines the degree of the polynomial kernel function and significantly affects the shape of the decision boundary.

- The range of values we considered for the degree hyperparameter were **2, 3, 4, 5, 6, 7, 8, 9, 10**.

The system configuration was consistently applied to both configurations, encompassing the following settings:

- **num_samples:** A total of **30** samples were utilized for experimentation.
- **max_t:** The maximum allocation of resources (e.g., training epochs) per configuration was capped at **50**.
- **grace_period:** Each configuration enjoyed a grace period of **50** resource units, during which performance assessments were deferred.

5.2 BERT and XLNet

As previously mentioned, we employed the Simple Transformers library to train our XLNet and BERT models. Since we were facing GPU limitations, our hyperparameter exploration was restricted to the following parameters:

- **Number of Training Epochs:** We explored values of **3**, **5**, and **10** for the total training epochs.
- **Training Batch Size:** We experimented with batch sizes of **16** and **32**.

The following parameters were held constant during training:

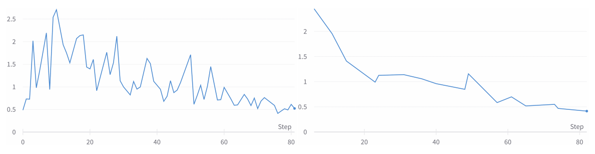
- **Manual Seed:** Set to **42** for reproducibility.
- **Overwrite Output Directory:** Enabled for saving model outputs.
- **Gradient Accumulation Steps:** Set to **8** for gradient accumulation.
- **Learning Rate:** Fixed at **2e-4**.
- **Optimizer:** Utilized the **AdamW** optimizer. Our hyperparameter search was conducted using a grid search approach, resulting in a total of 6 training configurations.

6 Discussion

We conducted fine-tuning for all three of our models and successfully obtained the optimal hyperparameters for each. Specifically, for the SVM model, we identified the best hyperparameters, which consisted of a 'C' value of 0.2747602509809857, a 'gamma' value of 0.020471419760911644, and the selection of the 'sigmoid' kernel.

Upon fine-tuning the BERT base model, we determined that the most effective hyperparameters included 10 training epochs and a training batch size of 32. These values represented the maximum allowable values within their respective parameter ranges. Unfortunately, our system had limitations that prevented us from exploring higher values. Additionally, when dealing with the more intricate BERT large model, we faced challenges due to CUDA limitations. As a result, we opted to train it using the hyperparameters that had been fine-tuned for the base model. The figure denoted as "Figure 1" illustrates the historical evolution of two essential metrics during the training phase of the BERT base model, using the best-performing hyperparameters obtained through fine-tuning. These metrics encompass the training loss and the evaluation loss. As is evident from the figure, the general behavior of the model exhibits a consistent trend where both training and evaluation losses decrease over time. The observed zigzag patterns in the losses can be attributed to the inherent randomness introduced during the training of transformer models. Transformers use techniques like dropout and stochastic gradient descent, which introduce randomness to the optimization process. This randomness can lead to fluctuations in the loss curves during training.

All the same procedures and parameters were also used for training both the XLNET base and XLNET large models. Once again, we found that the optimal hyperparameters for our XLNet base model were 10 epochs and a batch size of 32. As a result, we employed these same hyperparameters



(a) Training Loss (b) Validation Loss

Figure 1: Loss Changes for BERT Base

to train our XLNet large model. This approach allowed us to maintain consistency in the training process and directly compare the performance of both models on our datasets. The f1 score results we got after training the models are shown in Table 1.

Table 1: F1 Scores resulted from training 3 different models

Model	micro	macro
SVM	0.22	0.21
BERT BASE	0.43	0.34
BERT LARGE	0.52	0.41
XLNET BASE	0.52	0.41
XLNET LARGE	0.56	0.44

As shown in Table 1, it is clear that the XLNet model achieved the highest F1 score, indicating the best overall performance among the models. Following XLNet, the BERT model yielded the next-highest F1 score. Finally, the SVM model demonstrated the lowest score among the three models. Hence, it becomes evident that enhancing the intricacy of the models yields improved outcomes. Another clear observation is that the F1 macro score is lower than the F1 micro score for all the models. This discrepancy indicates that the models are performing well on the majority classes but not as effectively on the minority classes. This scenario is commonly referred to as the "class imbalance problem." The class imbalance problem in multiclassification tasks occurs when there is a significant disparity in the number of instances among different classes in the dataset. It is evident in Figure 1 which displays the number of comments in each category. To overcome this problem, one common approach would be using resampling techniques, such as oversampling the minority class, undersampling the majority class, or a combination of both. It is also noticeable when analyzing the classification report that the label 'Universalism: concern' consistently achieves the highest F1 score across all the models we evaluated and it dedicated

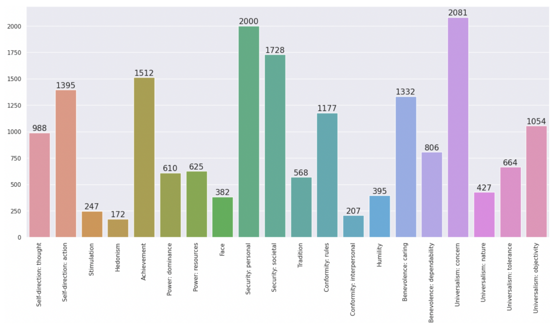


Figure 2: comments in each category

the maximum number of samples among all.

Furthermore, upon examining the training loss depicted in Figure 1, we observed an initial rapid increase in the training loss during the early epochs, followed by a subsequent decrease in subsequent steps. This phenomenon is commonly referred to as "catastrophic forgetting."Catastrophic forgetting is a phenomenon that can occur in neural networks, including models like BERT (Bidirectional Encoder Representations from Transformers) when they are fine-tuned or trained on new tasks or datasets. It refers to the model's tendency to forget previously learned knowledge or representations when it is exposed to new data, leading to a significant drop in performance on the initial steps. As the fine-tuning process continues, the model gradually regains some of this lost information, leading to an improvement in its performance.

7 Conclusion

In our project, we conducted a comparative analysis of three models for a multilabel task classification. Among these models, XLNet demonstrated superior performance, whereas SVM turned out to be the least effective. This outcome was anticipated since SVM lacks the inherent semantic understanding capabilities found in transformer-based models like XLNet. Hence, it is clear that increasing the complexity of the model can yield improved results. Furthermore, the availability of more powerful computational devices would enhance the performance of model training. Besides, an important observation revolved around the pronounced impact of label distribution on model performance. The varying prevalence of labels across instances proved instrumental in determining the effectiveness of the models. To pave the way for potential enhancements in the future, expanding the dataset size and carefully balancing label distribution could potentially lead to improved model performance.

8 Links to external resources

- https://docs.ray.io/en/latest/tune/index.html
- https://docs.wandb.ai/guides/sweeps
- https://touche.webis.de/semeval23/touche23-web/
- https://simpletransformers.ai/

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Shalom H. Schwartz. 2012. An overview of the schwartz theory of basic values. *Online Readings in Psychology and Culture*, 2.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2020. Xlnet: Generalized autoregressive pretraining for language understanding.