StFX-NLP at SemEval-2023 Task 4: Unsupervised and supervised approaches to detecting human values in arguments

Ethan Heavey and **Milton King** and **James Hughes**
St. Francis Xavier University
eheavey@stfx.ca mking@stfx.ca jhughes@stfx.ca

Abstract

In this paper, we discuss our models applied to *Task* 4: Human Value Detection of SemEval 2023, which incorporated two different embedding techniques to interpret the data. Preliminary experiments were conducted to observe important word types. Subsequently, we explored an XGBoost model, an unsupervised learning model, and two Ensemble learning models were then explored. The best performing model, an ensemble model employing a soft voting technique, secured the 34^{th} spot out of 39 teams, on a class imbalanced dataset. We explored the inclusion of different parts of the provided knowledge resource and found that considering only specific parts assisted our models.

1 Introduction

SemEval-2023 Task 4: Human Value Detection focuses on classifying a textual argument into one of 20 human value categories, each from Schwartz' value continuum (Kiesel et al., 2023). Examples of categories include, *Self-direction: thought* (it is good to have one's own ideas and interests), *Face* (it is good to maintain one's public image), and *Humility* (it is good to recognize one's own insignificance in the larger scheme of things). Blending computer science and social science, the system can also be applied to a variety of topics outside of academia. Understanding the motives of others allows one to have more insightful conversations as they approach topics from different angles (Crisp and Turner, 2011). By understanding what the other side of argument values, it can help us to reach common ground in discussions and lead to more productive interactions with less friction (Galinsky and Moskowitz, 2000).

We approached this *Task* with models that consider both supervised and unsupervised techniques. Furthermore, we perform an analysis on the data provided but the task to observe trends among different classes. The best performing model ranked 77^{th} out of the 111 submitted. The code is open source and available on GitHub[1].

Our unsupervised learning model employed a cosine similarity comparison between record embeddings and class description embeddings, taken from the knowledge resource[2]. To address some limitations of the system, the methods in which the data was preprocessed and embedded could have impacted the performance of our models. On top of data preprocessing, the dataset was only in English, limiting the reach of our models.

2 Background

The submissions focused only on the ValueEval'23 dataset (Mirzakhmedova et al., 2023), which contains almost $9,000$ records, in English. Roughly 80% of the records came from IBM's argument quality dataset (Gretz et al., 2020), 15% from the *Conference for the Future of Europe* (Barriere et al., 2022), and the remaining 5% from group discussion ideas. These arguments were then split by the *Task* organizers into separate training (60%), validation (20%), and testing (20%) sets. The raw data was broken down into four columns; the ID of the argument (to link the record to its labels), the conclusion, the stance, and the premise. The annotated label file has a record for every argument in the training and validation dataset, along with labels corresponding to the 20 classes.

3 System Overview

The four end-to-end pipelines implemented for this *Task* can be seen in Figure 1. Each row pertains to a specific model, with arrows detailing the data flow for each model. Nodes with identical names represent the same process occurring across different pipelines.

[1] https://github.com/VeiledTee/SemEval-2023

[2] https://touche.webis.de/semeval23/touche23-web/index.html#task

Proceedings of the The 17th International Workshop on Semantic Evaluation (SemEval-2023), pages 205–211

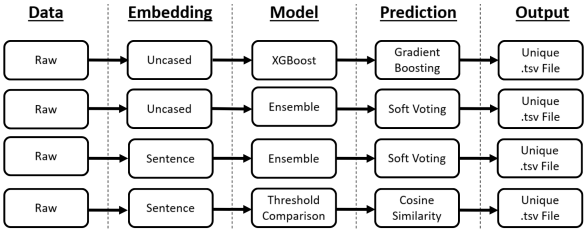

Figure 1: The end-to-end pipeline for the system.

3.1 Encoding

In an effort to test novel approaches to the *Task*, we opted to train and test a wide variety of models. To remedy this, we leveraged the embeddings provided by two different pretrained language models, detailed in the following subsections. We only retrieved the embedding of the casefolded "Premise" column of the dataset, but also kept track of the "Argument ID" for comparison purposes.

The first type of embeddings were "BERT" embeddings, created through the use of a BERT model, specifically "`bert-base-uncased`" (Devlin et al., 2019). Receiving premises as input, which were prepended with `[CLS]` and appended with `[SEP]` tokens, the structure of the text passed to the BERT model is as follows: $[CLS] + premise + [SEP]$. The BERT model then yielded a 768-dimension embedding for each token in the sentence. We used the embedding for the `[CLS]` token as it represented a single embedding for the entire text.

The second type of embeddings that were considered were embeddings that represent the entire sentence, generated by the `all-MiniLM-L6-v2` model (Reimers and Gurevych, 2019) from Hugging Face[3].This method yielded a 384-dimension embedding for each sentence, which is half the number of dimensions the previously mentioned BERT model outputs.

3.2 XGBoost

The XGBoost Classifier was the first of three supervised models employed to tackle the *Task*, and operates as follows; the model creates decision trees sequentially, allowing each tree to correct the output of the previous one. The classifier also utilizes weights assigned to each independent input variable, and adjusts the value of the weights assigned to incorrectly classified variables. Step one of a two-step application process was to train XGBoost on the `[CLS]` token embedding yielded by `bert-base-uncased` (Devlin et al., 2019) for each training premise, and the premise's respective labels.

Step two incorporated The TF-IDF score, and came after tuning the hyperparameters of the model. Calculated using the frequency of each type and the number of classes that they appear in, the score of each type from the training dataset was employed in the prediction process. Iterating over each class, if a test premise contained a word type with a score that landed the type in the top $1,000$ (approximately 14%) of all word types associated with the current class, the test premise is assigned the label for said class. The label assigned by the TF-IDF score overwrites the label assigned by the XGBoost model. The threshold of the top $1,000$ types per class was found to yield the best F_1-score, where $[10, 2000]$ represents the range of values tested. The TF-IDF scores for each class were sorted in descending order.

3.3 Ensemble

The ensemble method was implemented with the `VotingClassifier`[4] from the scikit-learn (Pedregosa et al., 2011) library. It consisted of three different models - a logistic regression model[5], a random forest classifier[6], and a Gaussian Naive Bayes[7] model - that were combined into a single ensemble system, and were only evaluated as a collective.

Two versions of model were created, trained on either BERT embeddings or sentence embeddings.

The BERT ensemble method was tailored to interpret the `[CLS]` token embeddings generated by the `bert-base-uncased` model. The model would predict the class of test premises based on each `[CLS]` token embedding, and output said predictions to a submission file. This method will be referred as "BERT ensemble" in this paper.

The sentence ensemble classifier, as it will henceforth be referred to, puts the full-sentence embeddings to use.

[3] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

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

[5] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html\/#sklearn.linear_model.LogisticRegression

[6] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[7] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

3.4 Threshold Comparison - An Unsupervised Approach

Using the definitions of each class from the *Task* knowledge resource, this model encodes them using the sentence embedding model (Reimers and Gurevych, 2019). Figure 2 breaks down the description of the `Self-direction: thought` class, with all remaining classes adhering to the same format. The phrase labeled 1 in red is the class's name, phrase 2 (blue) is a general summation of the values the class represents, and the third section of the description, the bullet points below phrase 2, describe in-depth the values the class represents. Of each bullet point in the third section, there exists two halves; $3a$, consisting of all tokens the the left of the semicolon, and $3b$, comprised of the tokens that appear after the semicolon.

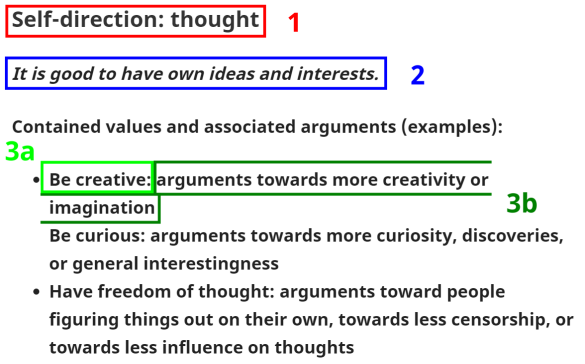

Figure 2: A breakdown of a class's description.

We represented a class by concatenating a combination of sentence embeddings (Reimers and Gurevych, 2019), representing phrases 1, 2, $3a$, and $3b$. The average of these embeddings was taken to yield a single embedding for each class. Once the embeddings for each of the 20 classes were generated, they were all compared to each other via the cosine similarity to get a sense of what classes were described in a semantically similar manner. The permutation that most significantly distinguished the classes was the inclusion of phrase 1, the exclusion of phrase 2 and $3a$, and the inclusion of phrase $3b$. With phrase 2 being nearly identical across all classes, it brought the cosine score of each class closer together, and thus was excluded. The $3a$ phrases are similar across all classes have a very similar structure, summarizing its paired $3b$ phrase, but not elaborating upon it. Excluding them thus increases the uniqueness of each class's description, in turn maximizing the differences between classes.

In the beginning of the training phase, all premises in the training data were embedded and fed into the model. By looping through each class (and the class's description embedding), all premise embeddings were compared to each class, one at a time. If the cosine similarity between the embedding for a given premise and a class description was above a set threshold, it was assigned the label 1, indicating the premise belongs to that class. If the embedding of the data does not reach the set threshold, the class is skipped and a label of 0 is assigned. These values were saved in a list that represented the labels of all premises, in order, of a particular class. This process is walked through in Algorithm 1. The similarity threshold was determined by examining the performance of the model on the range of values $[0.01, 0.5]$. The highest F_1-score was consistently generated by a similarity threshold value of 0.2.

4 Experiments

In this section, we discuss the design of our experiments and our hyperparameter tuning. Our team name for this Task was "Jesus of Nazareth".

4.1 Experimental Setup

The data (Mirzakhmedova et al., 2023) provided by the *Task* organizers (Kiesel et al., 2023) was already split into three sets, "training", "validation", and "test". The supervised learning models detailed in Sections 3.2 and 3.3 used only the training set. The hyperparameters for these models were tuned on the validation set, and used the best-performing settings for each modelwere applied to the test set.

For the unsupervised learning model (Section 3.4), both the training and validation data were used to tune the model. Since unsupervised learning does not require labels, this approach aimed to solve the *Task* in a novel manner. Finally, the test dataset was used to evaluate the threshold comparison model using the threshold that yielded the best results during tuning.

4.2 Hyperparameters

This subsection discusses the different hyperparameters of the four models reviewed in Section 3.

The "Max" and "Min" rows in Table 1 detail the range at which the hyperparameters for the XGBoost model were evaluated. The "Best" row shows the hyperparameters that yielded the highest F_1-score on the validation data, which were then employed in the final version of the model.

Working with the BERT ensemble model, the logistic regression model used all the default pa-

	eta	max_depth	subsample	colsample_bytree	lambda
Min	0.01	6	0	0	0
Max	0.75	25	1	1	1
Best	0.5	15	0.25	1	0

Table 1: The ranges used when tuning the hyperparameters of the XGBoost model. It also displays the best performing hyperparameters within said range.

rameters, bar one; `max_iter`. As shown in Table 2, the parameter was subject to a range of values from 100, to 100,000 with the best-performing value being 1,000. The `max_depth` and `n_estimators` hyperparameters of the random forest model were also tuned, with the ranges also displayed in Table 2, along with the "Best" value for each hyperparameter found within said range (50 and 200, respectively). The Gaussian Naive Bayes model was not tuned, but utilized out of the box, as we wished to observe if a raw statistical model would have a positive influence on the classification *Task* in comparison to the XGBoost model outlined in Section 3.2.

The `voting` parameter of the `VotingClassifier` model can use either "hard" or "soft" voting. "Hard" voting predicts a class based on the majority vote from each ensemble method (Seni and Elder, 2010), whereas "soft" voting places emphasis on the probability of each model's prediction and predicts the class with the highest average predicted probability across all models (James et al., 2013). Due to the class imbalance in the dataset, "hard" voting performs poorly and may predict more commonly occurring classes ("Face", "Universamism: concern") over uncommon ones ("Conformity: interpersonal", "Hedonism"). In comparison, the emphasis "soft" voting places on the predicted probabilities allows uncommon classes to be much more prevalent in the final prediction. "Soft" voting provided on average 10.44% and 21.93% higher F_1-scores with BERT and sentence embedding data, respectively.

Integrating "soft" voting into a multi label problem, required the class prediction to be framed as a binary classification problem by iterating over all classes, with "soft" voting predicting merely if a test premise was or wasn't the current class. The "BERT" and "Sentence" rows in Table 2 reflect how the imbalanced distribution of classes affected the best performing voting technique. The imbalance of classes led to "soft" voting models performing better than "hard" voting ones. With more balanced classes, the models could have performed better.

The sentence-embedding-based ensemble model's `max_iter` and `max_depth` values (depicted in the "Sentence" row of Table 2) are the only elements that differ in comparison to the tuned "BERT" model parameters. With a shallower random forest and less maximum iterations, the sentence ensemble model performed 7.67% better than its BERT counterpart during validation.

	max_iter	max_depth	n_estimators	voting
Min	100	10	50	soft
Max	100,000	1,000	100,000	hard
BERT	1,000	50	200	soft
Sentence	100	100	200	soft

Table 2: The ranges used when tuning the hyperparameters of the ensemble model. The "BERT" row shows the best hyperparameter values for the BERT ensemble model. "Sentence" shows the ideal hyperparameter settings for the sentence ensemble model.

The threshold comparison model was tuned by analyzing the definition of each class and the similarity threshold used in comparing the cosine similarity of each premise embedding to each class's embedding. The description embeddings for each class were broken up into phrases 1, 2, 3a, and 3b and the permutation of phrase 1 and phrase 3b yielded the best results.

The same combination of phrases was found to be the top performer, which reinforces the initial intuition that a better-defined separation of classes leads to better model performance. The similarity threshold was tuned on a range of values $[0.01, 0.5]$, with 0.2 yielding the highest F_1-score. The final iteration of the threshold comparison model utilizes a cosine similarity threshold of 0.2 and makes use of phrases 1 and 3b when defining a class.

4.3 Libraries used

Table 4 shows the Python libraries and their versions used for this *Task*. Python version 3.10.6 was used. The full `requirements.txt` file is available in the GitHub repository[8] for the project.

4.4 Evaluation Measures

As stated on the *Task* website[9], runs submitted to the TIRA (Fröbe et al., 2023) platform are evaluated primarily on F_1 score, precision, and recall "average over all value categories and for each category individually."

[8] https://github.com/VeiledTee/SemEval-2023

[9] https://touche.webis.de/semeval23/touche23-web/index.html

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
Ensemble (Sentence)	.40	.40	.52	.07	.17	.47	.30	.40	.05	.71	.55	.41	.48	.07	.07	.44	.17	.63	.69	.36	.39
Ensemble (BERT)	.33	.29	.51	.14	.20	.46	.26	.34	.01	.55	.44	.30	.45	.19	.07	.40	.15	.57	.34	.21	.40
Threshold Comparison (Sentence)	.30	.13	.42	.17	.11	.38	.13	.23	.12	.49	.46	.32	.34	.08	.08	.42	.18	.57	.39	.33	.26
XGBoost (BERT)	.26	.16	.40	.09	.04	.41	.14	.13	.11	.51	.39	.19	.30	.06	.09	.33	.19	.54	.17	.22	.45

Table 3: Achieved F_1-score of the team on the test dataset, from macro-precision and macro-recall (All) and for each of the 20 value categories. 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 Results

With our final results depicted in Table 3, the best performing model on the test dataset (placing $77^{th}/111$ was the Ensemble-Sentence model, followed by Ensemble-BERT model (placing $90^{th}/111$), with the unsupervised Threshold Comparison model closely tailing (placing $94^{th}/111$). The worst performing model was the XGBoost-BERT model (Section 3.2), placing $98^{th}/111$, which unfortunately didn't surpass the 1-Baseline model's performance. Our overall performance was 34^{th} out of 39 teams.

All models performed poorly on classes with very little representation in the training and validation datasets. The average F_1-score for the Stimulation, Hedonism, and Conformity: interpersonal classes (all classes with $< 2\%$ representation in the training and validation data) being 0.1175, 0.13, and 0.095, respectively. These less represented classes in the dataset may not have had sufficient data for our models to learn when an embedding represents this class. Conversely, when analysing the F_1-score of classes with $> 11\%$ representation (Security: personal, Universalism: concern), the average F_1-scores were significantly higher (0.565 and 0.5775, respectively). This led us to the conclusion that, with the class imbalance in the dataset, our models struggle to truly generalize, tending to predict more commonly occurring classes over uncommonly occurring ones. Despite the limitation outlined previously, our models outperformed the organizer's BERT implementation on some classes with less representation in the dataset.

6 Conclusion

In working on this *Task*, a wide range of machine learning models, both supervised and unsupervised, were implemented and tuned to maximize the primary metric used to evaluate models submitted, the F_1 score. Supervised learning techniques performed best when employing "soft" voting and having a somewhat-shallow depth. When tuning the unsupervised learning model, using only phrases 1 and $3a$ outlined in Figure 2, taken from the knowledge resource, yielded the best results. Other combinations of phrases led to unclear distinctions between classes, leading to the unsupervised model having a more difficult time predicting labels, and dragging down performance. The models were able to surpass the organizer's BERT implementation's F_1 score on classes with lots of representation in the training data, but unable to surpass the BERT model's overall F_1 score on the test data.

If we were to revisit this project in the future, we would like to delve further into the unsupervised learning approach by interpreting the classes and data in different ways. When updating the supervised learning models, employing more of the data, for example, embedding the "Conclusion" and "Stance" columns, could be beneficial in more accurately predict the label of each premise. It would also be interesting to employ other state-of-the-art embedding methods and compare the results.

References

Valentin Barriere, Guillaume Guillaume Jacquet, and Leo Hemamou. 2022. Cofe: A new dataset of intra-multilingual multi-target stance classification from an online european participatory democracy platform. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, pages 418–422.

Richard J Crisp and Rhiannon N Turner. 2011. Cognitive adaptation to the experience of social and cultural diversity. *Psychological bulletin*, 137(2):242.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Maik Fröbe, Matti Wiegmann, Nikolay Kolyada, Bastian Grahm, Theresa Elstner, Frank Loebe, Matthias Hagen, Benno Stein, and Martin Potthast. 2023. Continuous Integration for Reproducible Shared Tasks with TIRA.io. In *Advances in Information Retrieval. 45th European Conference on IR Research (ECIR 2023)*, Lecture Notes in Computer Science, Berlin Heidelberg New York. Springer.

Adam D Galinsky and Gordon B Moskowitz. 2000. Perspective-taking: decreasing stereotype expression, stereotype accessibility, and in-group favoritism. *Journal of personality and social psychology*, 78(4):708.

Shai Gretz, Roni Friedman, Edo Cohen-Karlik, Assaf Toledo, Dan Lahav, Ranit Aharonov, and Noam Slonim. 2020. A large-scale dataset for argument quality ranking: Construction and analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7805–7813.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An introduction to statistical learning*, volume 112. Springer.

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.

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.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Giovanni Seni and John Elder. 2010. *Ensemble methods in data mining: improving accuracy through combining predictions*. Morgan & Claypool Publishers.

A Appendix

Library	Version
`pandas`	1.5.0[10]
`numpy`	1.23.3[11]
`scikit-learn`	1.2.1[12]
`scipy`	1.9.3[13]
`sentence-transformers`	2.2.2[14]
`tokenizers`	0.13.1[15]
`torch`	1.13.1[16]
`transformers`	4.23.1[17]
`xgboost`	1.7.2[18]

Table 4: Table of major Python libraries (and their versions) employed while working to solve the *Task*.

[10] https://pandas.pydata.org/docs/whatsnew/v1.5.0.html

[11] https://numpy.org/devdocs/release/1.23.3-notes.html

[12] https://scikit-learn.org/stable/

[13] https://docs.scipy.org/doc/scipy-1.9.3/release.1.9.3.html

[14] https://www.sbert.net/

[15] https://github.com/huggingface/tokenizers

[16] https://pytorch.org/

[17] https://huggingface.co/docs/transformers/index

[18] https://xgboost.readthedocs.io/en/stable/

Algorithm 1 Threshold Comparison Algorithm

$T \leftarrow 0.2$
$c \leftarrow$ embeddings of all class descriptions
$d \leftarrow$ embeddings of all premises from data
$output \leftarrow [\ ]$
for $description\_embedding$ in c **do**
    $predictions \leftarrow [\ ]$
    for $premise\_embedding$ in d **do**
        if cosine_similarity($premise\_embedding$, $description\_embedding$) $\geq T$ **then**
            append 1 to $predictions$
        else
            append 0 to $predictions$
        end if
    end for
    append $predictions$ to $output$
end for

save $output$ to submission file