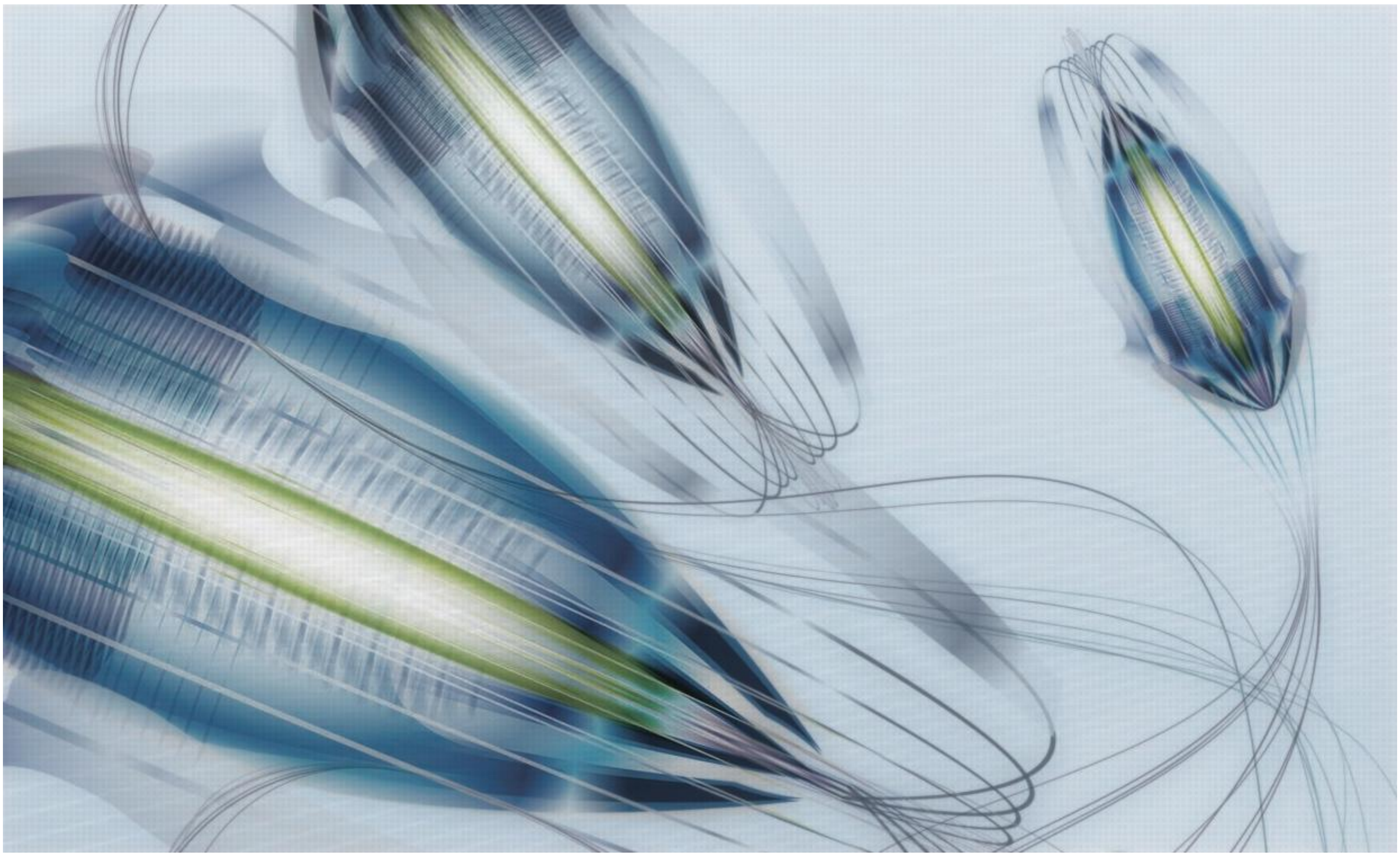
ISSN 2033-3544

CLiPS Technical Report 11

COMPUTATIONAL LINGUISTICS & PSYCHOLINGUISTICS
TECHNICAL REPORT SERIES, CTRS-011, January 2025
www.uantwerpen.be/clips

Multiscope: A User-Friendly Multi-Label Text Classification Dashboard

Jens Van Nooten and Walter Daelemans



CLiPS
Computational Linguistics & Psycholinguistics
University of Antwerp

Multiscope: A User-Friendly Multi-Label Text Classification Dashboard

Computational Linguistics, Psycholinguistics and Sociolinguistics Research Center

CLiPS Technical Report Series (CTRS)

December, 2024

CTRS-011

Authors

Jens Van Nooten
CLiPS Research Center, University of Antwerp (BE)
jens.vannooten@uantwerpen.be

Walter Daelemans
CLiPS Research Center, University of Antwerp (BE)
walter.daelemans@uantwerpen.be

Prinsstraat 13, 2000 Antwerp, Belgium
+32 (0)3 265 52 22

Multiscope: A User-Friendly Multi-Label Text Classification Dashboard

Jens Van Nooten and Walter Daelemans

University of Antwerp (CLiPS)

Abstract. We present Multiscope, a user-friendly tool for training and applying Multi-Label Text Classification (MLTC) models to datasets of choice. MLTC is a complex, yet vital component of analyzing large corpora that aims to assign multiple labels to a single text. However, compared to more traditional classification approaches, training, evaluating and deploying MLTC models can be challenging because of the nature of the task. Multiscope provides a complete pipeline for this classification problem, starting with data stratification, providing insights into the label distribution and interactions between labels. The tool also provides a framework for fine-tuning state-of-the-art transformer models and training classical Machine Learning models. The trained models can be evaluated using multi-label classification metrics.

Keywords: Multi-label text classification, NLP for the humanities

1 Introduction

Natural Language Processing has seen rapid developments in the past few years with the introduction of the transformers model architecture and Large Language Models (LLMs). These developments lead to improved results for a multitude of NLP tasks, including text classification, which is a valuable asset to analyzing large text corpora. A complex subproblem in text classification is *Multi-Label Text Classification* (MLTC), which aims to assign multiple labels to a single text. This differs from *single-label classification* problems (*binary* and *multi-class classification*), which assign one to each text. Use cases of MLTC include topic classification and emotion classification.

The aforementioned developments in NLP inspired applications for Digital Humanities research, which is stimulated and facilitated by the CLARIAH-Flanders project. Multiscope aims to bridge the gap between NLP and Digital Humanities research by introducing a user-friendly GUI to aid researchers with developing multi-label text classifiers. The tool was developed in Python and can be used from a user interface developed with Gradio (Abid, 2019). The code and installation instructions can be found on GitHub: https://github.com/clips/multiscope.

The remainder of this report details general user guidelines for Multiscope. Demonstrations are provided through screenshots from the application.

2 Multi-Label Text Classification

Setting up an environment for developing and evaluating MLTC models raises a few challenges. The first challenge arises when creating train, validation and test splits for the models to be trained on. Since multiple labels are assigned to a single text at once, one cannot rely on traditional stratification methods. The second challenge pertains to the training procedure of MLTC models. Traditional Support Vector Machines (SVMs) and Transformers -or neural networks in general- have to be adapted to accommodate the complex label space during training and inference by transforming the classification problem or by adjusting the model's architecture. The final challenge is related to the evaluation of such models, since traditional metrics like precision, recall and F1 are interpreted differently. Moreover, such multi-label models also have to be evaluated with specific metrics that deal with the complexity of multiple predicted labels. Additionally, the notion of a confusion matrix becomes more complex in this setting, since it is unclear which labels are confused when incorrect labels are predicted or not all correct labels are predicted. Solutions to these challenges are addressed in the following sections.

Data stratification Data stratification is essential to developing well-calibrated classifiers when a dataset is imbalanced: It ensures that models are evaluated on data samples that are representative of a source dataset by approximating its label distribution when creating a validation or test dataset. This is especially important for *cross-validation* experiments. While ensuring an equal label distribution is straightforward for single-label classification, it becomes more complicated for MLTC. Sechidis et al. (2011) propose an iterative stratification method that not only takes the occurrences of labels in isolation into account, but also co-occurrences between labels. This provides a more reliable approximation of the original dataset's distribution when creating sub-samples.

Classifier Adaptation In traditional Machine Learning (ML) research, MLTC is often reinterpreted as multiple binary classification problems (*binary relevance*). Neural networks, especially transformers, tend to leverage a sigmoid activation layer on top of output logits to accommodate multiple predictions for a single instance. This yields a probability for each label independently.

Model Evaluation Traditional evaluation metrics such as precision, recall and F1-scores must be reinterpreted for MLTC. In essence, F1-scores per class are calculated by taking the harmonic mean of precision and recall of the positive class, i.e. the cases where a label is assigned to a text. While F1-scores provide an accurate estimation of a model's performance, other metrics catered to MLTC are required in an evaluation setup. Examples of such metrics are the following:

- **Exact Match Ratio (EMR)**: the proportion of instances where *all* labels are predicted correctly.
- **Hamming Loss (HL)**: the average of labels that are predicted incorrectly per instance.
- **Normalized Discounted Cumulative Gain at K (nDCG@k)**: measures the ranking quality of predictions by comparing the ranking of predictions with the 'ideal' ranking.

Heydarian et al. (2022) introduced a multi-label confusion matrix, which allows for confusion between classes to be visualized more easily. The novelty of this approach lies in the addition of an extra row (No True Label; NTL) and an extra column (No Predicted Label; NPL), which take missing labels from prediction sets and superfluous predictions into account (cf. Figure 8).

3 Multiscope

I. Loading data

Users will first need to determine whether they want to upload their data from a local file (as a JSON, CSV or Excel file) (1) or use a dataset that is publicly available on the Huggingface hub (Wolf et al. 2020) (2). When using Huggingface, the user must specify the dataset identifier, text column name (defaults to 'text'), label column name (defaults to 'labels') and, if applicable, the relevant subset of the data. Then, the dashboard leverages the *datasets* library (Lhoest et al. 2021) to load the data. It should be noted that these datasets can vary in structure. It is recommended that a local version of the data is made that fits the structure criteria that are described below.

JSON files should adhere to the following structure:

```
{
    data:{
            'train':    [{'id': ID_1, 'text': TEXT_1, 'labels': [LABELS]}, ..., {'id': ID_N, 'text': TEXT_N, 'labels': [LABELS]} ],
            'val':      [{'id': ID_1, 'text': TEXT_1, 'labels': [LABELS]}, ..., {'id': ID_N, 'text': TEXT_N, 'labels': [LABELS]} ],
            (if present)
            'test':     [{'id': ID_1, 'text': TEXT_1, 'labels': [LABELS]}, ..., {'id': ID_N, 'text': TEXT_N, 'labels': [LABELS]} ]
            (remove 'labels' if not present in test set)
       }
  }
```

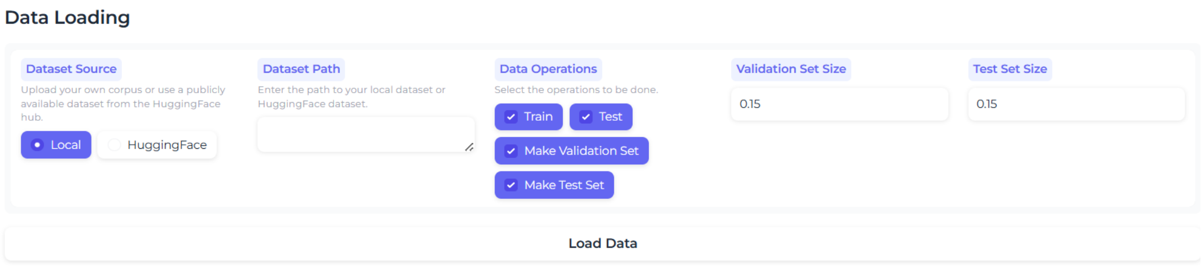
Excel and CSV files should adhere to the following structure. The “text” column should contain strings of texts. “Labels” should contain lists of strings or integers and “split” should contain strings (“train”, “val” or “test”) indicating to which split of the dataset the corresponding text belongs.

text	labels	split
“Text 1”	[L1, L2]	“train”
“Text 2”	[L1, L2, L3]	“train”
“Text 3”	[L3, L4]	“val”
“Text 4”	[L1, L2, L4…]	“test”
…	…	…

Below are a few examples taken from the SemEval2018 dataset (Mohammad et al. 2018):

text	labels	split
“Worry is a down payment on a problem you may never have'. Joyce Meyer. #motivation #leadership #worry	[anticipation, optimism, trust]	“train”
“Whatever you decide to do make sure it makes you #happy.”	[joy, love ,optimism]	“train”
I blew that opportunity -__- #mad	[anger, disgust, sadness]	“test”
…	…	…

(1):



Data Loading

Dataset Source
Upload your own corpus or use a publicly available dataset from the HuggingFace hub.
● Local ○ HuggingFace

Dataset Path
Enter the path to your local dataset or HuggingFace dataset.

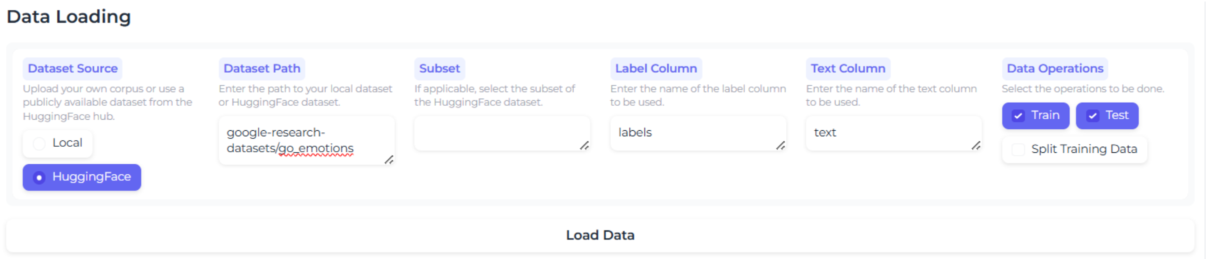
Data Operations
Select the operations to be done.
☑ Train ☑ Test
☑ Make Validation Set
☑ Make Test Set

Validation Set Size
0.15

Test Set Size
0.15

Load Data

(2):



Data Loading

Dataset Source
Upload your own corpus or use a publicly available dataset from the HuggingFace hub.
○ Local ● HuggingFace

Dataset Path
Enter the path to your local dataset or HuggingFace dataset.
google-research-datasets/go_emotions

Subset
If applicable, select the subset of the HuggingFace dataset.

Label Column
Enter the name of the label column to be used.
labels

Text Column
Enter the name of the text column to be used.
text

Data Operations
Select the operations to be done.
☑ Train ☑ Test
☐ Split Training Data

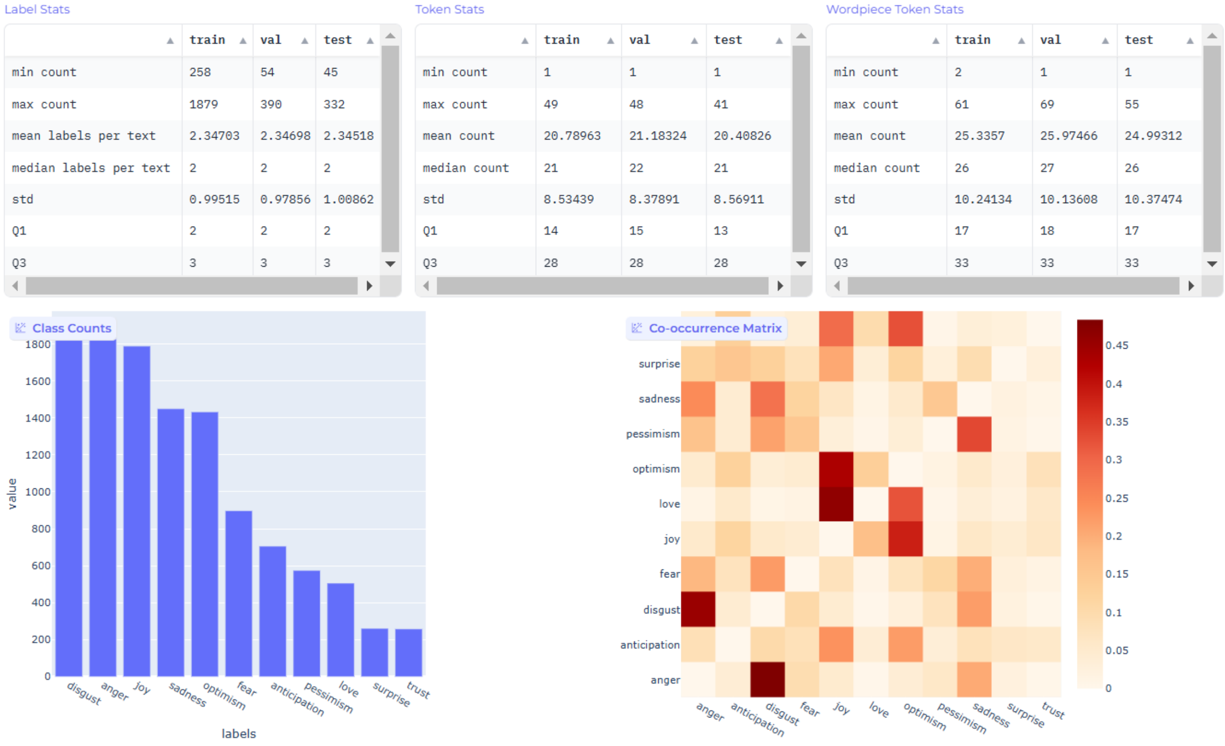
Load Data

The second step is to select the operations that are to be performed on the data. The user has four options, which can all be used in conjunction, or separately:

- ‘Train”: Trains a model on the training split of the provided dataset. If selected on its own, it will only train a model and not perform inference on a test set.
- “Test”: If used in conjunction with “Train”, the dashboard automatically evaluates the trained model on the provided test split. If specified on its own, it only performs inference on the provided test split. If no labels are present in the test split, only inference is performed and no performance metrics are calculated.
- “Make Validation Set”: Creates a stratified validation split (if selected or if a validation split is not provided). The use can also define the portion of the training data that should be used to create this split.
- “Make Test Set”: Creates a stratified test split. The use can also define the portion of the training data that should be used to create this split.

Once the dataset has been loaded, Multiscope provides statistics related to token counts, wordpiece token counts (these are relevant for BERT-like models) and the label distribution for all provided splits of the data (3). In terms of label-related information, the dashboard provides a bar chart showing the label distribution and a co-occurrence matrix, thereby providing insight into which labels tend to co-occur in the training data.

(3):



Label Stats

	train	val	test
min count	258	54	45
max count	1879	390	332
mean labels per text	2.34703	2.34698	2.34518
median labels per text	2	2	2
std	0.99515	0.97856	1.00862
Q1	2	2	2
Q3	3	3	3

Token Stats

	train	val	test
min count	1	1	1
max count	49	48	41
mean count	20.78963	21.18324	20.40826
median count	21	22	21
std	8.53439	8.37891	8.56911
Q1	14	15	13
Q3	28	28	28

Wordpiece Token Stats

	train	val	test
min count	2	1	1
max count	61	69	55
mean count	25.3357	25.97466	24.99312
median count	26	27	26
std	10.24134	10.13608	10.37474
Q1	17	18	17
Q3	33	33	33

II. Training a model

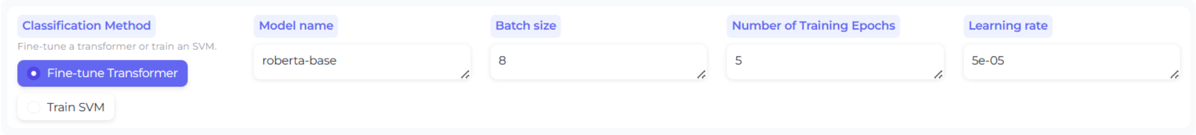
Once the data has been loaded in, the user can then choose between two classification frameworks, namely **fine-tuning a BERT-like transformer** (Devlin et al. 2019) or **training a Support Vector Machine (SVM)**.

A. Fine-tuning Transformers

For fine-tuning BERT-like models, the dashboard relies on the transformers library, developed by HuggingFace. The user has the option to choose the base model, which can be any fine-tunable model available on the HuggingFace hub[1] that is compatible with the *AutoModelForSequenceClassification* class. If a model has been chosen, the name of the model as it is on the HuggingFace hub needs to be pasted in the relevant text box. In the case of only performing inference on the test set, a fine-tuned local (or remote) model can also be loaded by entering the path to the model *directory*.

Additionally, the user has the option to determine the value of selected hyperparameters (4), namely batch size, max sequence length, the number of training epochs and the learning rate. The recommended values should generally work for most models. Further experimentation with these hyperparameters, especially number of epochs and learning rate, is recommended. It should be noted that the batch size is heavily dependent on the available GPU memory: A smaller batch size equals lower memory usage.

(4):



Classification Method

Fine-tune a transformer or train an SVM.

● Fine-tune Transformer

○ Train SVM

Model name: roberta-base

Batch size: 8

Number of Training Epochs: 5

Learning rate: 5e-05

The best performing model, which is selected based on the performance on the validation set (macro F1) during training, is automatically saved after training. This model can be loaded later to perform inference on a held-out test set, or be used for other applications.

[1] Consult https://huggingface.co/models for a complete list of models.

B. *Training an SVM*

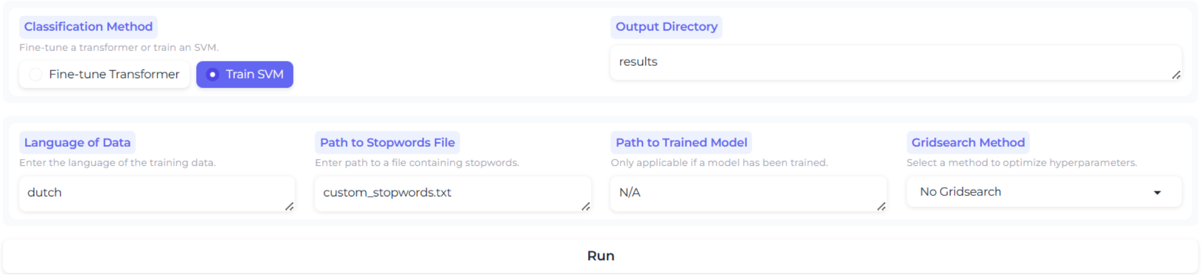
Multi-scope also offers the user the possibility to train an SVM. For this, the dashboard relies on the scikit-learn package (Pedregosa et al. 2011). Using the binary relevance approach for MLTC (*OneVsRestClassifier*), it trains a binary classifier for each label separately and concatenates predictions. As a data-preprocessing step, stop words are removed from the data. This can be done by (optionally) providing the language of the training data or providing a custom list of stop words in a TXT file (one word per line). To remove language-specific stop words, the dashboard loads the provided lists from the NLTK package (Bird et al. 2009). If a language nor custom stop words file are provided, the language defaults to English and loads NLTK stop words.

The user is provided three options for experiments (5), including options to perform *Grid Search* experiments to find the optimal hyperparameter settings for an SVM. The three options are the following: An option where no Grid Search is performed and two options where five-fold cross-validation Grid Search experiments are performed with stratified splits:

- **No Grid Search**: The dashboard will not perform Grid Search experiments, but instead uses a pipeline with standard hyperparameters.
- **Standard**: Performs a basic Grid Search experiment with stratified five-fold cross validation. The options for the hyperparameters are determined beforehand.
- **Custom**: Allows the user to manually select the hyperparameters and the number of options to be considered during the Grid Search experiments (6). In the case highlighted below, the *N-gram range*, *max_df* and *C* parameters will be Grid Searched, with four, three and six options for each respectively. The more options per hyperparameter are Grid Searched, the longer it takes for the model to finish the training procedure.

If a Grid Search experiment is performed, the dashboard automatically saves the best performing model (based on the -averaged F1 score), the best performing parameters and all scores per parameter combination.

(5):



Classification Method
Fine-tune a transformer or train an SVM.
○ Fine-tune Transformer ● Train SVM

Output Directory
results

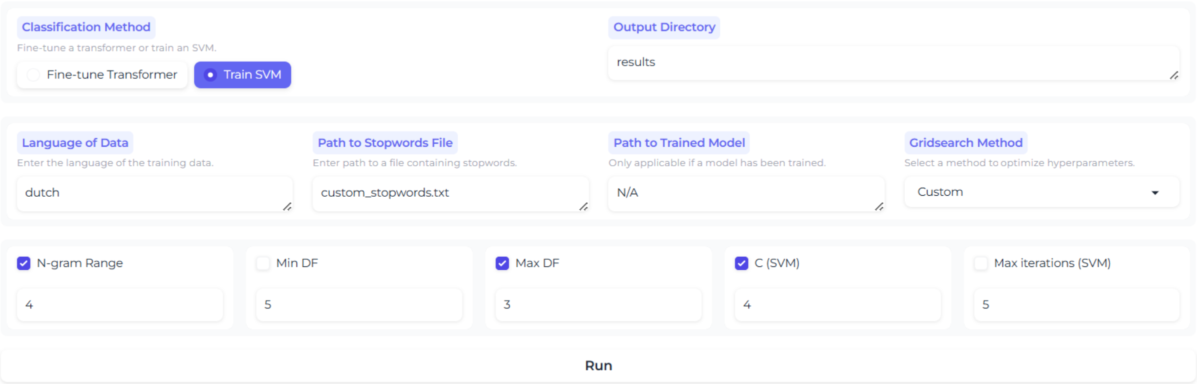
Language of Data
Enter the language of the training data.
dutch

Path to Stopwords File
Enter path to a file containing stopwords.
custom_stopwords.txt

Path to Trained Model
Only applicable if a model has been trained.
N/A

Gridsearch Method
Select a method to optimize hyperparameters.
No Gridsearch

(6):



Classification Method

Fine-tune a transformer or train an SVM.

○ Fine-tune Transformer ● Train SVM

Output Directory: results

Language of Data — Enter the language of the training data.: dutch

Path to Stopwords File — Enter path to a file containing stopwords.: custom_stopwords.txt

Path to Trained Model — Only applicable if a model has been trained.: N/A

Gridsearch Method — Select a method to optimize hyperparameters.: Custom

☑ N-gram Range: 4

☐ Min DF: 5

☑ Max DF: 3

☑ C (SVM): 4

☐ Max iterations (SVM): 5

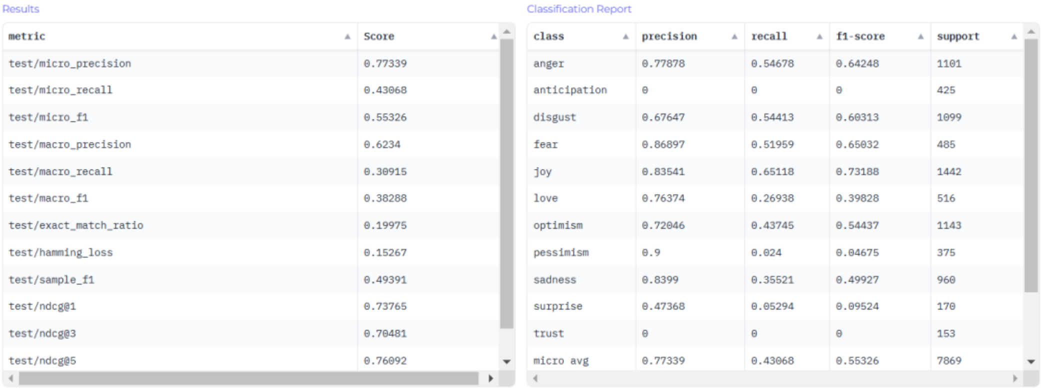
III. Evaluating a model

After training, the selected model is evaluated on the test, if specified (cf. Section 3, I.). Several metrics are calculated:

- **Micro/macro-averaged precision**
- **Micro/macro-averaged recall**
- **Micro/macro-averaged F1 & sample F1**
- **Exact Match Ratio**
- **Hamming Loss**
- **nDCG@k, where *k* = [1, 3, 5, 10]**

These are provided in a table, in addition to a classification report where the performance on each class is shown (precision, recall and F1) (7).

(7):



Results

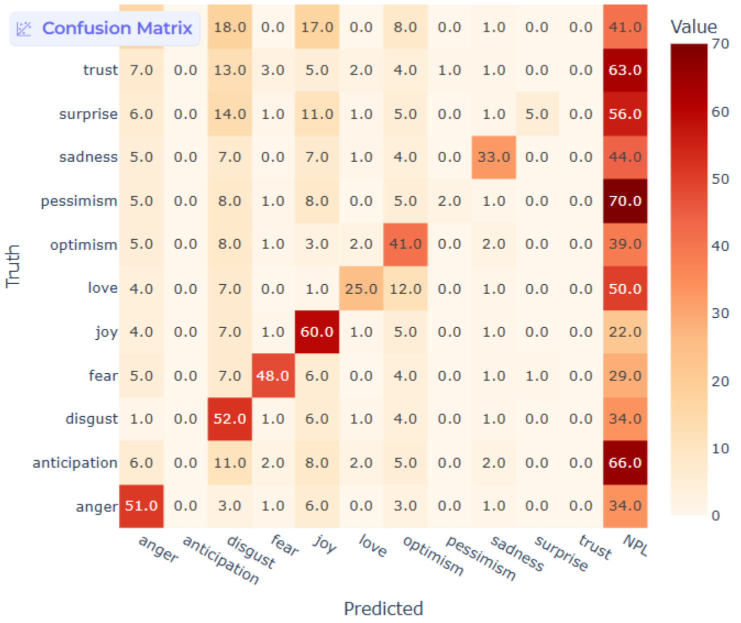
metric	Score
test/micro_precision	0.77339
test/micro_recall	0.43068
test/micro_f1	0.55326
test/macro_precision	0.6234
test/macro_recall	0.30915
test/macro_f1	0.38288
test/exact_match_ratio	0.19975
test/hamming_loss	0.15267
test/sample_f1	0.49391
test/ndcg@1	0.73765
test/ndcg@3	0.70481
test/ndcg@5	0.76092

Classification Report

class	precision	recall	f1-score	support
anger	0.77878	0.54678	0.64248	1101
anticipation	0	0	0	425
disgust	0.67647	0.54413	0.60313	1099
fear	0.86897	0.51959	0.65032	485
joy	0.83541	0.65118	0.73188	1442
love	0.76374	0.26938	0.39828	516
optimism	0.72046	0.43745	0.54437	1143
pessimism	0.9	0.024	0.04675	375
sadness	0.8399	0.35521	0.49927	960
surprise	0.47368	0.05294	0.09524	170
trust	0	0	0	153
micro avg	0.77339	0.43068	0.55326	7869

Moreover, a multi-label confusion matrix (Heydarian et al. 2022) is also provided (8).

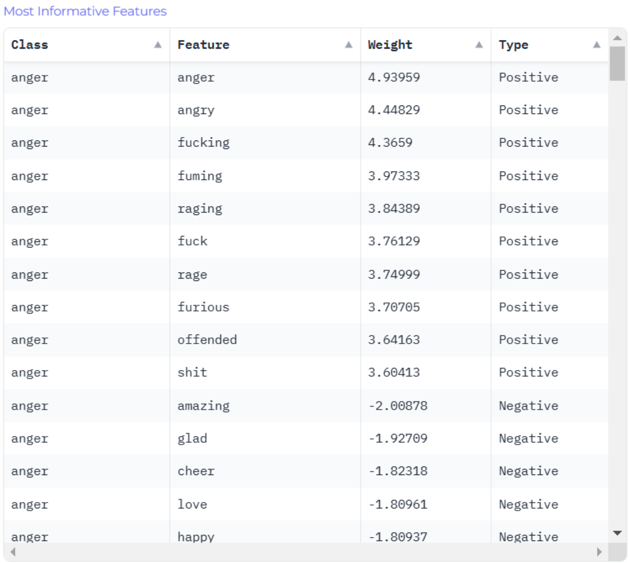
(8):



	anger	anticipation	disgust	fear	joy	love	optimism	pessimism	sadness	surprise	trust	NPL
			18.0	0.0	17.0	0.0	8.0	0.0	1.0	0.0	0.0	41.0
trust	7.0	0.0	13.0	3.0	5.0	2.0	4.0	1.0	1.0	0.0	0.0	63.0
surprise	6.0	0.0	14.0	1.0	11.0	1.0	5.0	0.0	1.0	5.0	0.0	56.0
sadness	5.0	0.0	7.0	0.0	7.0	1.0	4.0	0.0	33.0	0.0	0.0	44.0
pessimism	5.0	0.0	8.0	1.0	8.0	0.0	5.0	2.0	1.0	0.0	0.0	70.0
optimism	5.0	0.0	8.0	1.0	3.0	2.0	41.0	0.0	2.0	0.0	0.0	39.0
love	4.0	0.0	7.0	0.0	1.0	25.0	12.0	0.0	1.0	0.0	0.0	50.0
joy	4.0	0.0	7.0	1.0	60.0	1.0	5.0	0.0	1.0	0.0	0.0	22.0
fear	5.0	0.0	7.0	48.0	6.0	0.0	4.0	0.0	1.0	1.0	0.0	29.0
disgust	1.0	0.0	52.0	1.0	6.0	1.0	4.0	0.0	1.0	0.0	0.0	34.0
anticipation	6.0	0.0	11.0	2.0	8.0	2.0	5.0	0.0	2.0	0.0	0.0	66.0
anger	51.0	0.0	3.0	1.0	6.0	0.0	3.0	0.0	1.0	0.0	0.0	34.0

When the user has opted for training an SVM, the most informative features for each class are also provided in a table (9). These are extracted based on the features (n-grams) with the highest positive and negative coefficients for each class.

(9):



Most Informative Features

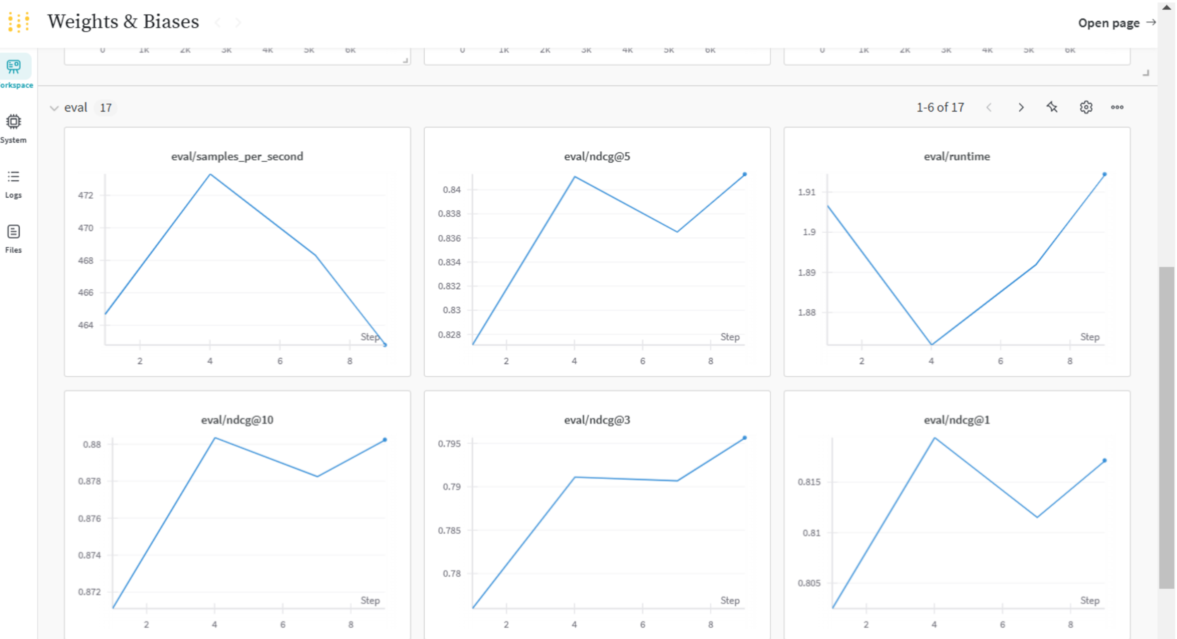
Class	Feature	Weight	Type
anger	anger	4.93959	Positive
anger	angry	4.44829	Positive
anger	fucking	4.3659	Positive
anger	fuming	3.97333	Positive
anger	raging	3.84389	Positive
anger	fuck	3.76129	Positive
anger	rage	3.74999	Positive
anger	furious	3.70705	Positive
anger	offended	3.64163	Positive
anger	shit	3.60413	Positive
anger	amazing	-2.00878	Negative
anger	glad	-1.92709	Negative
anger	cheer	-1.82318	Negative
anger	love	-1.80961	Negative
anger	happy	-1.80937	Negative

Multiscope also makes use of Weights and Biases to track the training process of neural networks and to report classification metrics on the evaluation set and test set, if applicable. Important to note is that the user needs a Weights and Biases account for this[2]. The project name and run name are generated automatically by the script. The dashboard only shows the page for a single run (10). However, the user can browse to the project page (11) by clicking “Open page” in the top right corner

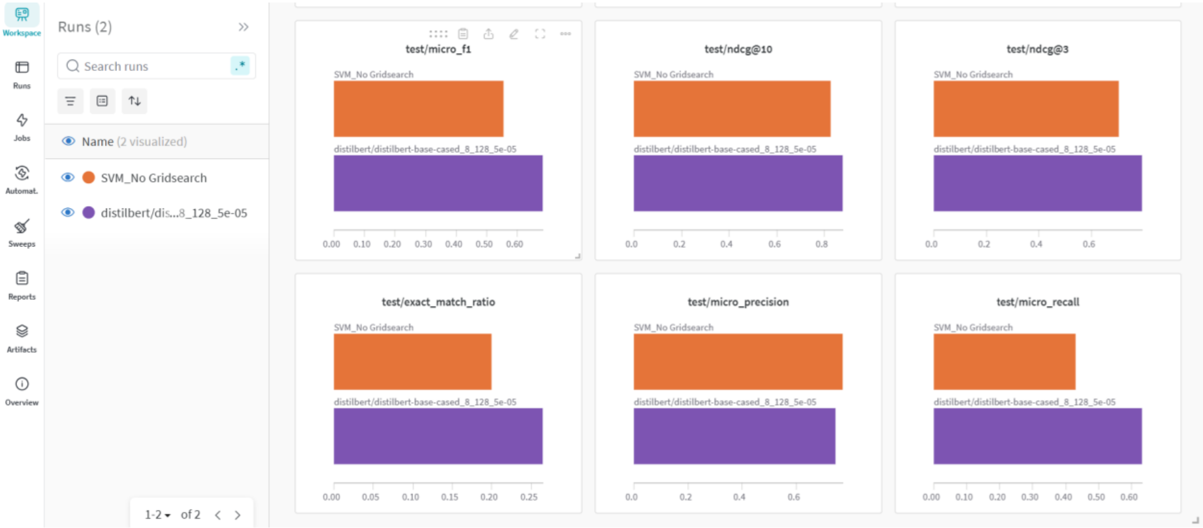
[2] https://wandb.ai/site/

of the embedded report. This page provides an overview of all runs and results thereof. If multiple experiments are conducted, W&B provides visualizations that allow for an easy comparison of models and setups.

(10):



(11):



4 Conclusion

In this report, we introduced Multiscope, a dashboard for training MLTC models. The dashboard provides a complete pipeline for classification, starting with creating a stratified validation split and providing informative statistics about the label distribution and interactions between labels. This is followed up by adjusting a transformers model's/SVM's architecture for MLTC and, finally, evaluating the models with appropriate metrics and a confusion matrix for multi-label classification. The dashboard also provides insights into the SVM's decision-making process by retrieving the most informative features per class.

How to cite

Jens Van Nooten and Walter Daelemans. 2024. Multiscope: A User-Friendly Multi-Label Text Classification Dashboard. *CLiPS Technical Report Series 11 (CTRS 11)*. ISSN 2033-3544. Computational Linguistics, Psycholinguistics, and Sociolinguistics Research Center.

Bibliography

Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan and James Y. Zou. 2019. Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild. arxiv:1906.02569

Steven Bird, Edward Loper and Ewan Klein. 2009. *Natural language processing with Python*. O'Reilly Media Inc.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, & Toutanova, Kristina. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, & Thamar Solorio (Eds.), *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)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Mohammadreza Heydarian, Thomas E. Doyle and Reza Samavi. 2022. MLCM: Multi-label confusion matrix. *IEEE Access, 10*, 19083–19095. https://doi.org/10.1109/ACCESS.2022.3151048

Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, Vanderplas, Jake, Passos, Alexandre, Cournapeau, David, Brucher, Matthieu, Perrot, Matthieu, & Duchesnay, Édouard. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*(85), 2825–2830. Retrieved from http://jmlr.org/papers/v12/pedregosa11a.html

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, et al. 2021. Datasets: A Community Library for Natural Language Processing. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 175-–184, Punta Cana, Dominican Republic. Association for Computational Linguistics (ACL).

Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2011. On the stratification of multi-label data. *Machine Learning and Knowledge Discovery in Databases: ECML PKDD 2011*, volume 6913 of Lecture Notes in Computer Science, pp. 145–158. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-23808-6_10

Mohammad, Saif, Bravo-Marquez, Felipe, Salameh, Mohammad, & Kiritchenko, Svetlana. (2018). SemEval-2018 Task 1: Affect in Tweets. In Marianna Apidianaki, Saif M. Mohammad, Jonathan May, Ekaterina Shutova, Steven Bethard, & Marine Carpuat (Eds.), *Proceedings of the 12th International Workshop on Semantic Evaluation* (pp. 1–17). New Orleans, Louisiana: Association for Computational Linguistics. https://doi.org/10.18653/v1/S18-1001

Wolf, Thomas, Debut, Lysandre, Sanh, Victor, Chaumond, Julien, Delangue, Clement, Moi, Anthony, Cistac, Pierric, Rault, Tim, Louf, Rémi, Funtowicz, Morgan, Davison, Joe, Shleifer, Sam, von Platen, Patrick, Ma, Clara, Jernite, Yacine, Plu, Julien, Xu, Canwen, Le Scao, Teven, Gugger, Sylvain, Drame, Mariama, Lhoest, Quentin, & Rush, Alexander M. (2020). HuggingFace's Transformers: State-of-the-art natural language processing. *arXiv*. https://arxiv.org/abs/1910.03771