NLP Project Report

Thomas Chen, Thomas Garnier and Jedrzej Miczke

Introduction

The aim of the project is to develop an entity classification system that can discriminate items into three categories: cultural agnostic, cultural representative, and cultural exclusive, based on the information from the Wikipedia and Wikidata pages of the entity. To address this task, we have created two models: the state-of-the-art approach – Transformer-based, and the second 'classical' model not utilizing the Transformer architectures.

1 Non-LM-based Classifier

1.1 Methodology - The Overview

After a cleaning of the label (see Section A.1), we started creating the Non-Transformer-based classificiton system. As it is undoubtedly a more demanding endeavor in terms of the required creativity, we had to resort to exploring additional – not given to us in the datasets – features from the Wikipedia and Wikidata pages. After numerous experiments and thorough analysis, we have decided to create 3 different models based on the various extracted features: the data-features-based Neural Network classifier, the description-based Neural Network classifier, and lastly the Wikipedia-photos-based Convolutional Neural Network classifier. Finally, we merged the knowledge from all these 'weak' classifiers by creating a meta-learner ensemble trained on the logits of the predictions of the aforementioned models, hoping we can combine information from thoses different modalities (numerical, text,and photos).

1.2 The Features-Based Neural Network

1.2.1 Feature Mining and Selection

In this model, we wanted to take advantage of the information available on the Wikipedia and Wikidata pages as well as the metadata available about these pages. Apart from our own invention, we also had the variables that were originally given to us, which we have analyzed and decided to: use the entity variable and category variable (after modification), and get rid of the subcategory, due to high sparsity (108 subcategories, where each has less than 1.5% support, see figure 27) and potential sampling bias, assuming that the broader category variable provides sufficient information. For the thorough description, see Section A.2. The other, "mined" by us, variables amount to 14, the description of all of them is available in the appendix (see Section A.3). We choosed these variables as they cover a wide range of features and metadata, allowing the model to learn the classification based on a rich and diverse set of predictors. Furthermore, we have created many different plots regarding these variables (see Section A.11) as the visualization of them helped us to better understand their characteristics.

1.2.2 Feature Cleaning and Selection

After that, we had to analyze the relevance of the results. Firstly, we have checked the occurrences of the NAN values. We have dealt with it in different ways, depending on the variable type and the fraction of NANs(Section A.5.1). Later, we conducted the sanity check(Section A.5.2) – based on common sense – to potentially catch erroneous records or simply API errors. Then we have analyzed the distribution of the variables, to catch extreme outliers and clip the values(Section A.5.3). Finally, we have plotted the distribution of the labels with respect to the mined variables(Section A.12) both in the training and dev set, We aimed to:

- Check if there is any variance in the distribution of labels grouped by these variables in the training set.
- Assess if there are any extreme discrepancies between the training and dev set distributions of labels grouped by these variables.

1.2.3 Feature-Based Models and Results

After applying many specific machine learning models on the features (neural networks, random forest, XGBoost, logistic regression and SVM), in order to classify our items, we retained the neural networks model that presents the best accuracy, precision, recall and F1-Score. The accuracy of the Feature-Based NN model: dev set: 76.67% Train set: 77.38%. (For more thorough analysis of results see Section A.8)

1.3 Description Model

In parallel with the previous model, we developed a model based on the embeddings of the descriptions of the entities. To do so, we utilized spacy, for it's spead and ease, but also FastText embeddings as it can deal with out-of-the vocabulary words such as proper names in a meaningful way. Additionally, for words not present in the embedding vocabulary, we count their occurences in each description and include this count as an extra input feature– we anticipated it could be a strong cultural non-agnosticity marker. The input for the classifier is therefore the normalized average of the words from the description in the embedding space (300 dimensions), concatenated with the OOV count (301 dimensions in total). As models, we tried KNN and neural network methods, for their non parametric and parametric aspects.

1.3.1 Description Model Results

The best performing model is FastText with a a neural network with following accuracies: 68% Train set: 90.15%. It seems overfitting, even with the use of early stop method. See Section A.8

1.4 Convolutionnal Neural Network (CNN)

The last classifier is by far the most unconventional approach. It is a CNN trained on the images extracted from the Wikipedia pages of the entities. Surely, by itself, it is deemed not to be a robust predictor for several reasons (see Section A.6.2). On the other hand, the lack of a picture on the Wikipedia page could potentially be correlated with the label, and more interestingly, the network could hypothetically learn to recognize, for example, human images – which indicate a non-agnostic class. The more in-depth description of the CNN is to be found in the appendix (see Section A.6).

1.4.1 CNN Results

The accuracy of the Photos-Based CNN model: dev set: 49.33% Train set: 65.97%. See Section A.8

1.5 The meta-learner model

We created a meta-learner Neural Network learning the cultural classification task from the logits (predictions before normalization) of the 3 aforementioned classifiers. Indeed, we have observed that each model does not commit the same mistakes (see Section A.7), and none of the models 'dominates' any other. Therefore, merging them, with the help of the meta-learner NN, could be a way to boost the classification performance even more.

1.5.1 Meta-learner Results

The ensemble did not exceed the features model accuracy, reaching only 69.67%. However, it has achieved by far the highest accuracy on the training set: 94.13%, which could suggest strong overfitting, even though we used early stop condition on the model. Our conclusion is that the description and the Wikipedia page photos simply do not allow for such good generalization as the information from the Wikipedia and Wikidata pages. Therefore, they are not as good predictors as the features.

2 Transformers-based classifier

2.1 Methodology

The transformers-based approach involved the fine-tuning of pre-trained transformers for sequence classification. Several models were evaluated including DistilBERT, BERT, RoBERTa, DeBERTa and GPT2. Among them, DistrilBERT presented the best performance. As a result, different input format variations were explored using DistilBERT such as Description, concatenation of name and description, explicit concatenation ("name": <name>, "description": <description>), and an enriched input by adding the 1000 first characters of the Wikipedia article.

2.2 Results

The final selected configuration is DistilBERT model with the simple name-description concatenation as input, due to its highest performance, over the accuracy, precision, recall and F1-score metrics. This setup achieved an accuracy of 77%. Detailed results and additional evaluation metrics are presented in Section A.9

A Appendix

A.1 label cleaning

Firstly, before conducting any classification, we had to clean the train dataset we were given. It turned out there were 9 different labels in the training dataset, whereas in fact there should be 3. The nine different labels on the training set are: cult, cultural, cultural ag, cultural agn, cultural agnostic, cultural ex, cultural exclusive, cultural represent and cultural representative.

For the non-ambigous labels, we directly merged them. Therefore we have cultulral ag, cultulral agn with cultural agnostic; cultural ex with cultural exclusive and cultural represent with cultural representative.

For the unclear label (cult, cultural) and the missing values, we decided to apply the same rule defined to built the dataset : chose the classification given by ChatGPT-o3.

A.2 Original Features Analysis

Apart from the Wikipedia and Wikidata links to the entity pages, we were also given in our datasets: description of the entity, type of entity, category of entity, and subcategory of entity. All these variables could potentially be used as predictors which we could train our feature-based Neural Network on (see Section 1.2).

A.2.1 Subcategories

In the case of the subcategories, we initially wanted to one-hot encode them. However, we found out that there are 112 of them, which would cause a huge explosion of dimensionality in the Neural Network input. What's even worse, due to their high number, these 112 variables had an extremely sparse distribution, as only one of them had support greater than 1.5% in the dev set. Lastly, we tried clustering the subcategories into groups to alleviate the aforementioned issues. We attempted this using the FastText embedding representations of the subcategories. However, we did not achieve good intra-cluster similarity and inter-cluster dissimilarity (only 0.18 silhouette score, see Figure 1), so we decided to drop this variable.

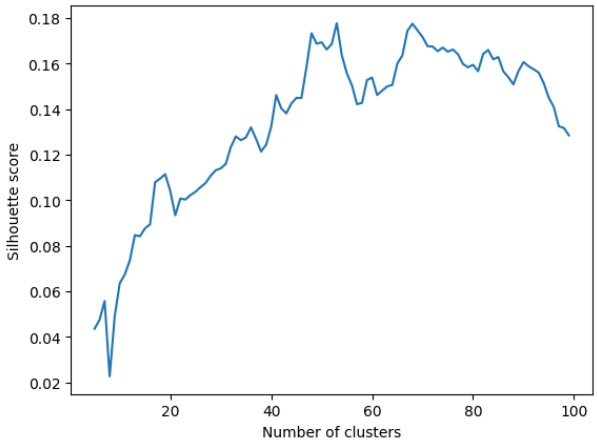

Figure 1: the silhouette score for different number of clusters -subcategories clustering in the FastText embedding space

A.2.2 Categories

The category variable has 19 distinct values, which makes it more appropriate for one-hot encoding. However, we additionally wanted to ensure that it was a valuable predictor. To do so, we conducted a chi-squared test between the distribution of labels with respect to categories in the train and dev sets, to see whether there were significant differences in these distributions. The underlying idea of this action is that, if the distributions are significantly different (at the standard significance level $p = 0.05$), then the correlation between the category in the train set and the labels could just be an inherent bias of the sampling policy. It turned out that the distributions were not significantly different in 14 out of the 19 categories, thus we one-hot encoded only these 14 ones.

A.2.3 Type

The type variable was encoded in binary and retained as a predictor in the features NN, as it did not display any notable issues.

A.2.4 Description

The description was also kept; however, it was utilized in the Description NN model, not in the feature-based one.

A.3 Feature Mining – Feature-based NN Model

The features that we have mined/retrieved from the Wikipedia and Wikidata pages of the entities include various structural and semantic properties. Below, we describe each in detail:

A.3.1 Date

This feature is the existence of a date-related property in Wikidata (e.g. foundation date, year of birth, etc.). It is encoded as a binary variable.

A.3.2 Number of Languages

This refers to the number of language versions in which the Wikipedia page of the entity exists. It provides a reference for the popularity of the term all over the world.

A.3.3 Country Information Existence

A binary feature that denotes if the Wikidata item contains any country-related information. The presence of this feature could potentially indicate the geographical embedding of the term or specific origin.

A.4 Wikidata Hierarchical Graph-Based Features

The data for the subsequent features in this subsection pertains entirely to the Wikidata hierarchical graph relations between the entities. It was retrieved from the Wikidata pages of the entities using the RDF database query language – SPARQL.

A.4.1 Number of Instances Upwards

This variable indicates how many entities the given item is an instance of (i.e., how many more general classes it belongs to). It could provide information about the degree of generality/specificity of the item, as well as the popularity of the given item in terms of demarcation between cultural exclusive and representative.

A.4.2 Number of Instances Downwards

Similar to the previous feature, but in the opposite direction of the relation. It indicates how many Wikidata entities are instantiations of the given item. It could reflect both the specificity and the popularity of the given entity in terms of the demarcation between cultural exclusive and representative.

A.4.3 Items that are Instances of the Given Item (Downward)

This binary variable denotes the existence of at least one entity that is an instance of the given item. It indicates whether the item is a superclass for any other item.

A.4.4 Number of Superclasses (Upward)

This is the number of higher-level superclass links that the given item has. Again, a potentially useful feature for representing the level of generality/specificity of the item.

A.4.5 Superclass Depth

This refers to the number of items in the shortest path from the entity to a root in the subclass hierarchy graph. The shortest path is found with a breadth-first search algorithm. This number is potentially good for assessing the specificity of the item.

A.4.6 Description-Based Features

In addition to the aforementioned features, we wanted to enrich the NN input with additional information derived from the analysis of the Wikidata entity description using the pre-trained `en_core_web_md` model based on FastText embeddings. These features include:

- Number of nouns in the description
- Number of verbs in the description
- Number of adjectives in the description
- Presence of an ethnic group mention in the description
- Presence of an event mention in the description

A.5 Features Cleaning

A.5.1 Dealing with NANs

Nan values could be caused either by the unavailability of the information for certain entities or simply API request issues. For the cases of only a tiny fraction of NANs for a variable, we resorted to filling the value with the median. For the Number of Languages, we had 1 NAN value that we filled with the median. For the other cases, we often assumed the NAN variables are tantamount to the non-existence of the relation/feature and explicitly used them as binary variables — e.g., in the country information variable: existence of the country information vs. non-existence of the country information; date — existence of the date information (e.g., foundation, birth of entity) vs. non-existence of such information.

A.5.2 Sanity Check

We have conducted a sanity check on the mined variables to ensure that there are no obviously clear errors (resulting from faulty data or API request faults), which could thus hinder the training of the neural network. We found that in two cases, the date variable associated with an entity was after 2025. Therefore, we decided to set this value to NAN — non-existence of the relation.

A.5.3 Variable Clipping

Finally, we analyzed the distribution of the values in the mined variables in search of extreme outliers, which could again be caused by API errors or faulty information. Even if such records were not erroneous, it would be beneficial to clip the extreme values for better generalization between the training and the dev set (see the distribution change as a result of clipping in Figure 2).

A.6 CNN model

A.6.1 NANs Handling

As mentioned below in (Section A.6.2), for both the train and dev set we had images in less than 50% of the records. Therefore, we had to somehow deal with the missing values. We decided to fill the NANs with completely white images, trying in this way to prompt the model to learn the relation between the non-existence of the image and the label.

A.6.2 CNN Limitations

There are certain insurmountable limitations to the CNN-based classification approach:

- **Unavailability of records** – The photos could either be not available on the Wikipedia pages or protected against scraping despite their existence on the Wikipedia pages. Therefore, we have only 42% of the records from the training set and only 47% in the dev set.
- **Computational and memory complexity** – We had to compress the photos to a very small size (28x28), thus losing the vast majority of the information contained in them, due to issues concerning memory. Both the loading of the data to Google Colab was taking a lot of time with higher resolution (128x128), and the overload of the RAM during training forced us to substantially decrease the batch size, which resulted in an increase in the training time of the NN.
- **Information content** – Lastly, we do not really see a way that the CNN could learn in any meaningful way the distinction between the cultural exclusive and cultural representative class, as this information simply cannot be conveyed in the photography. This is confirmed by Figure 4, as it is visible there that both the precision and recall(by far) are the best for the agnostic class

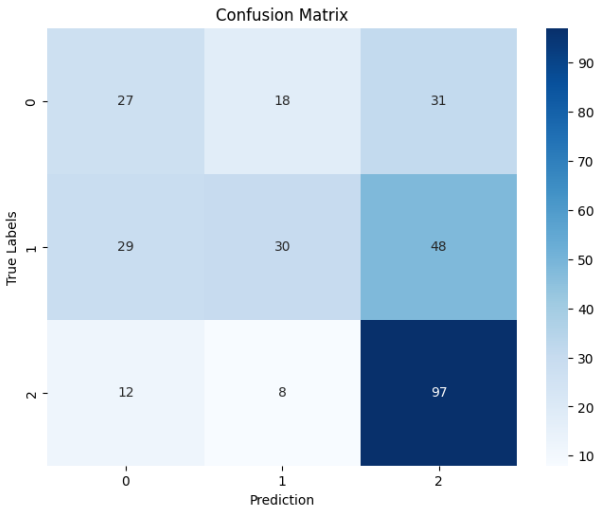

Figure 4: CNN Confusion matrix on the dev set.
0: cultural exclusive
1: cultural representative.
2: cultural agnostic.

A.7 Meta-Learner Model

To justify using the meta-learner trained on the logits of the base models, we will present the differences in the predictions of all our base models on the dev set. It turns out that they do not commit the same mistakes, and none of the models is strictly dominated by any other (the set of properly predicted examples of any model is not a subset of properly predicted examples of another). Therefore, the performance of any of the base models could potentially be boosted with the information from the other models. See the proof in the plotted disagreements between the base models with regards to properly predicted classes, Figure 5.

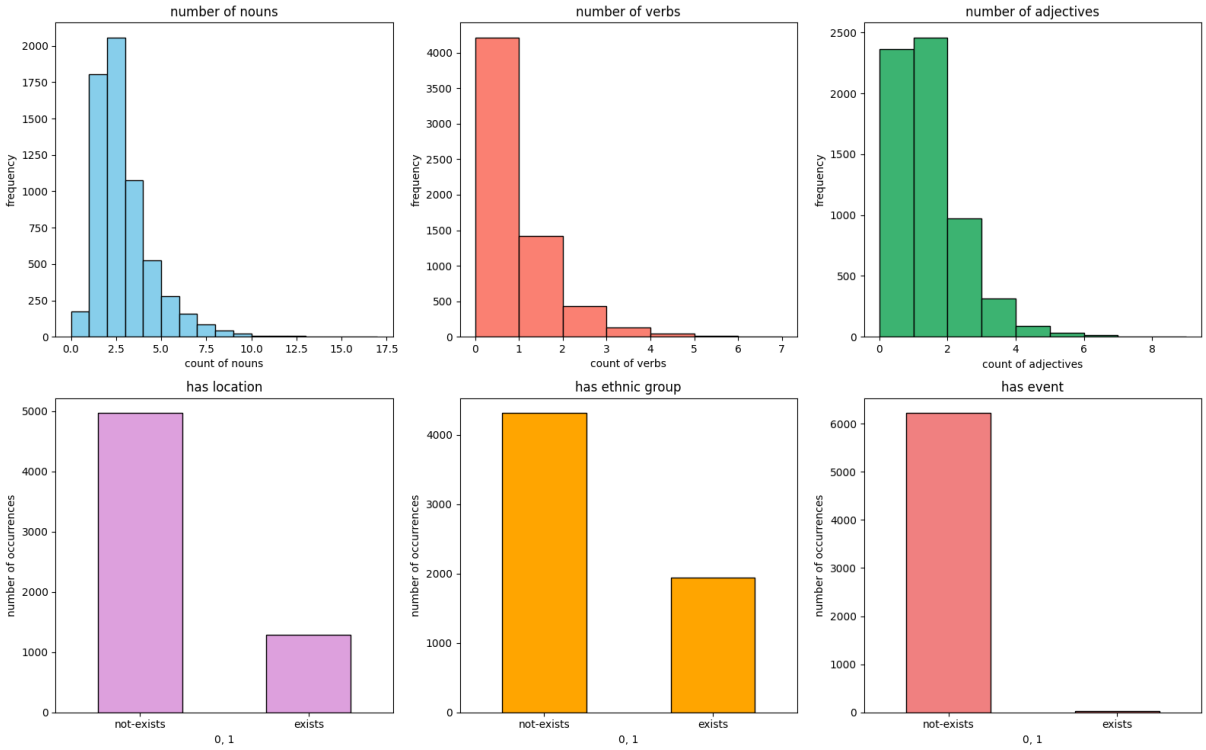

Figure 2: Description-based features before clipping in the training set.

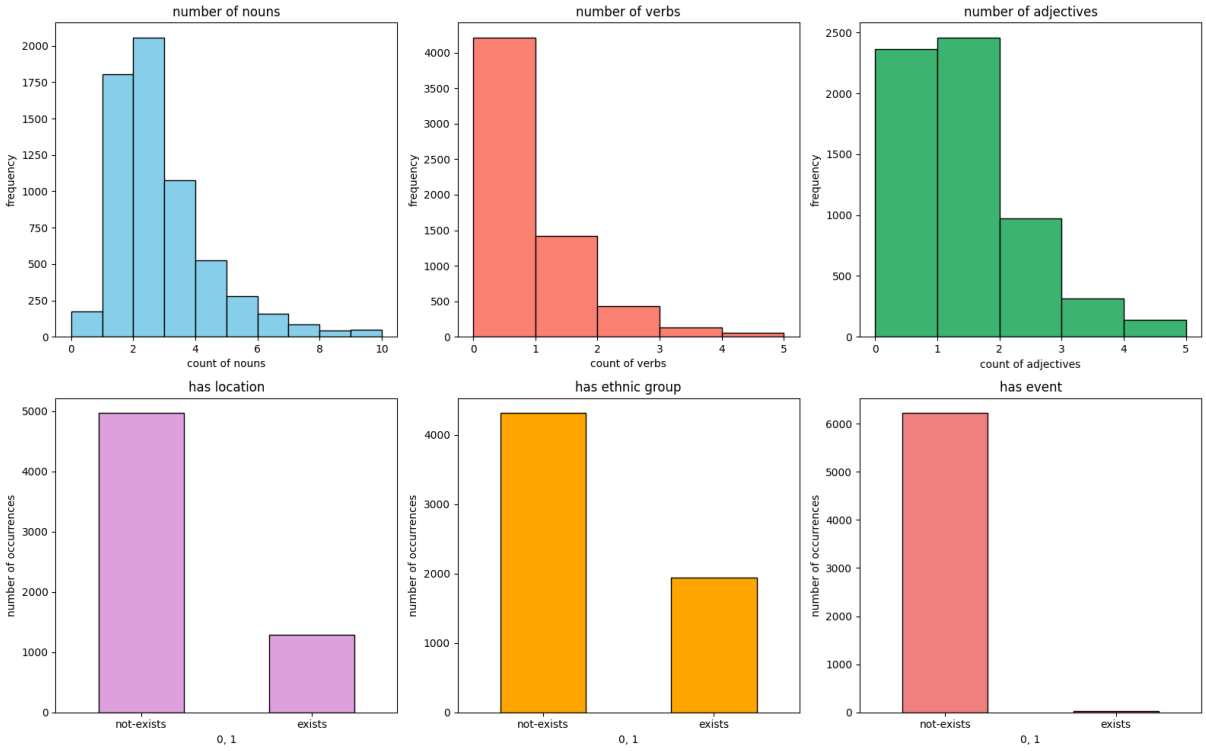

Figure 3: Description-based features after clipping in the training set.

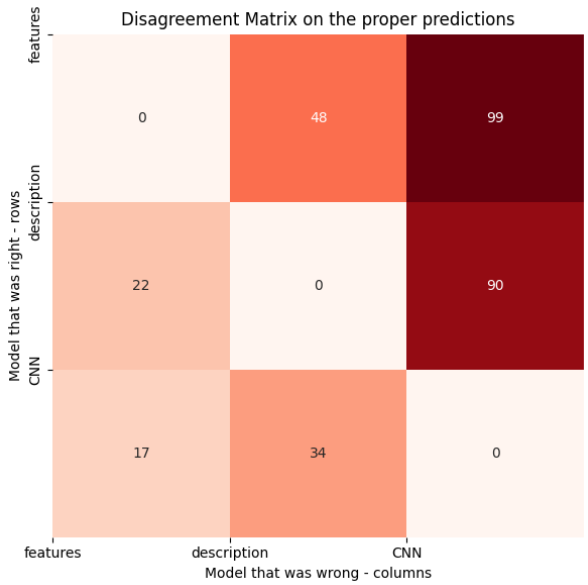

Figure 5: The disagreement matrix for the correct predictions of the models.
Rows: model that predicted correctly.
Columns: model that predicted incorrectly.
Cells: count of such instances.

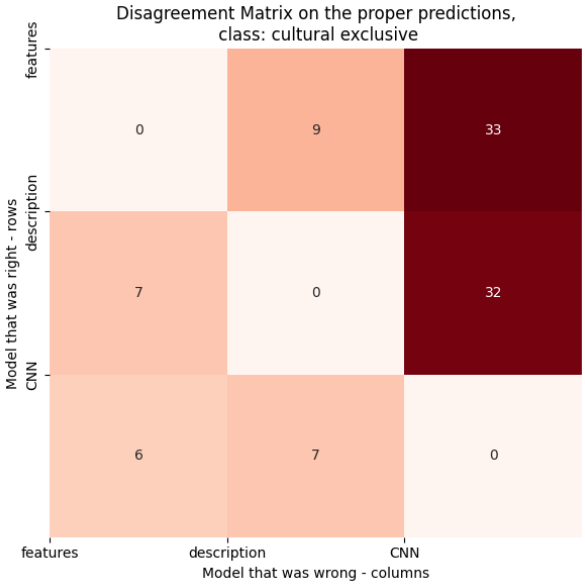

Figure 7: The disagreement matrix for the correct predictions of the models - cultural exclusive class.
Rows: model that predicted correctly.
Columns: model that predicted incorrectly.
Cells: count of such instances.

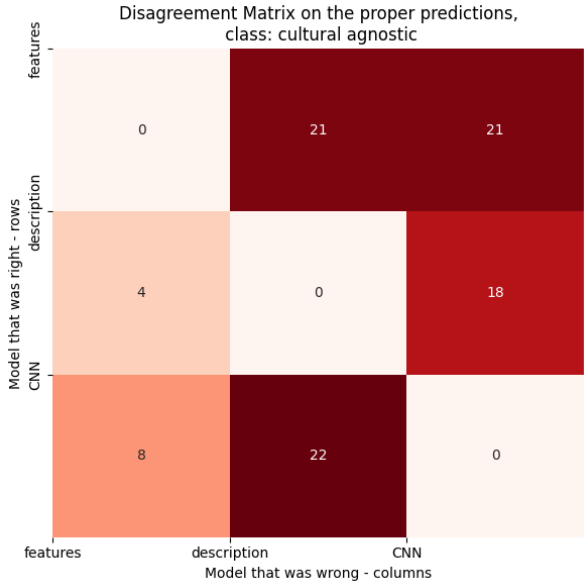

Figure 6: The disagreement matrix for the correct predictions of the models - cultural agnostic class.
Rows: model that predicted correctly.
Columns: model that predicted incorrectly.
Cells: count of such instances.

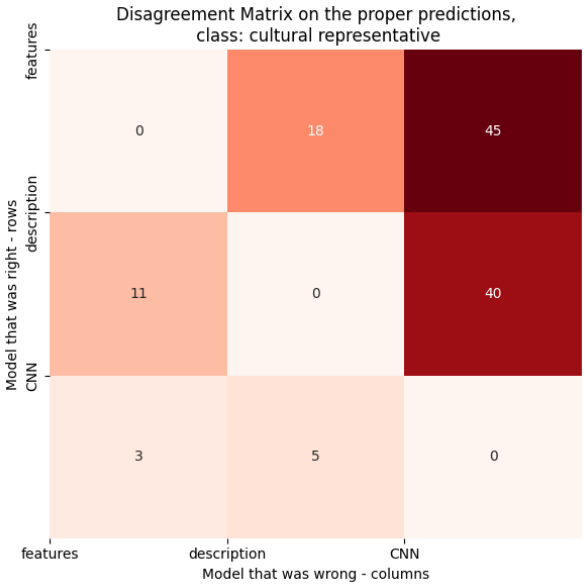

Figure 8: The disagreement matrix for the correct predictions of the models - cultural representative class.
Rows: model that predicted correctly.
Columns: model that predicted incorrectly.
Cells: count of such instances.

A.8 Summary of the results for the non transformer models

Model	Accuracy	Recall	Precision	F1-Score
CNN Model				
White Image Filling	**51.33**	**48.82**	49.46	**46.84**
Drop NaN	48.95	46.17	**50.07**	45.15
Description Model				
SpaCy + KNN	64.00	64.71	63.75	60.64
FastText + KNN	66.33	67.01	66.73	65.02
FastText + Neural Network	**68.00**	**68.00**	**68.13**	**67.44**
Features Model				
Neural Network	**75.67**	**75.67**	**75.50**	**75.06**
Random Forest	74.67	74.67	74.32	74.39
XGBoost	74.33	74.33	73.91	74.03
Logistic Regression	76.33	76.33	76.44	75.51
SVM	73.00	73.00	72.79	72.00

Ensemble Method	Accuracy	Recall	Precision	F1-Score
Neural Network	**69.67**	**69.67**	**71.26**	**69.52**
XGBoost	65.00	65.00	66.92	65.19
Logistic Regression	63.33	63.33	65.63	63.80
SVM	64.67	64.67	66.82	65.08

A.9 Transformer Model - Results

Table 1: Performance of various Transformer models on the development set

Model	Input Type	Accuracy	Recall	Precision	F1-Score
DistilBERT	Description only	69.33	69.33	69.15	68.90
DistilBERT	**Name + Description**	**77.00**	**77.00**	**76.63**	**76.62**
DistilBERT	Name + Explicit Description	76.67	76.67	76.48	76.36
DistilBERT	Name + Description + Wikipedia	74.33	74.33	74.28	74.23
BERT	Name + Description	70.03	70.03	68.60	68.42
RoBERTa	Name + Description	68.69	68.69	66.76	66.78
DeBERTa	Name + Description	74.33	74.33	73.84	73.84
GPT	Name + Description	70.67	70.67	71.14	70.42

As we see in Table 1, the Transformer model based on DistilBERT and the "Name + Description" format performed the best across all metrics, outperforming also the classical model. It is worth keeping in mind that a certain threshold of accuracy may not be possible to surpass, when generalizing from training set on the dev set, due to the inherent disagreements between the training and development sets. The labeling policy for both datasets was very different—the training set was labeled by GPT O3, whereas the development set was labeled by humans. Therefore, each of these labeling policies reflects a different kind of bias.

A.10 Additional Ideas for the Project

We have also had other ideas for boosting the performance of our models in the cultural classification task. We explored the idea of round-translation of the descriptions, which means translating the description to a set of pivot languages and then back to English. We wanted to measure whether the back-translated version of the description is somehow different/deformed, as it could suggest a lack of the term in the pivot language. The similarity measure that we wanted to utilize was either the cosine distance in the embedding space (with some ready embeddings like FastText), or simply the Levenshtein distance between the translation and back-translation weighted by the inverse of the length of the description (however, this might not be optimal due to possible replacements with synonyms in agnostic terms). Unfortunately, in this case, the practical limitations turned out to be insurmountable, as the API request to the translator endpoint took too much time — which we realized after running the approach on a subset of 50 items from the train dataset, which took approximately 10 minutes.

A.11 Plots of the features distributions

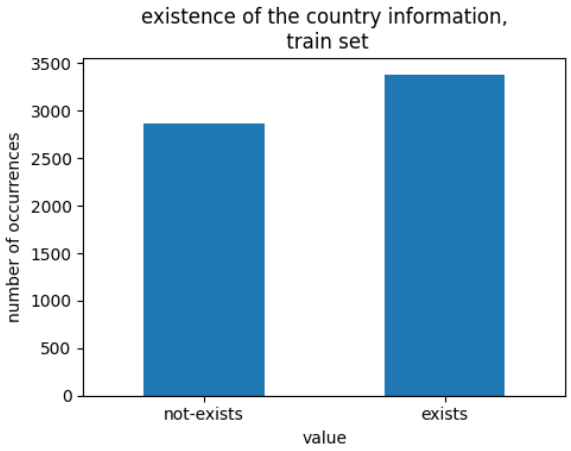

Figure 9: Presence of country information in training data.

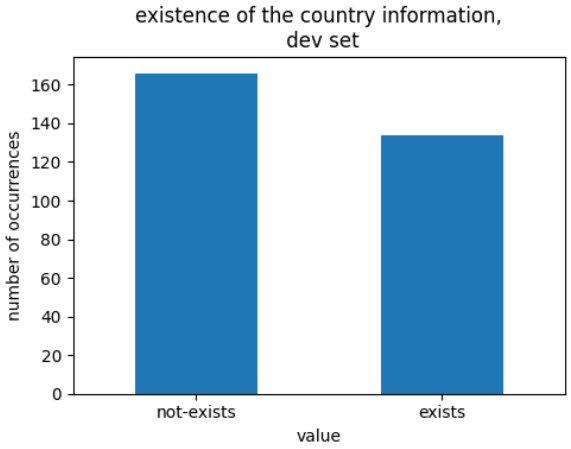

Figure 10: Presence of country information in development data.

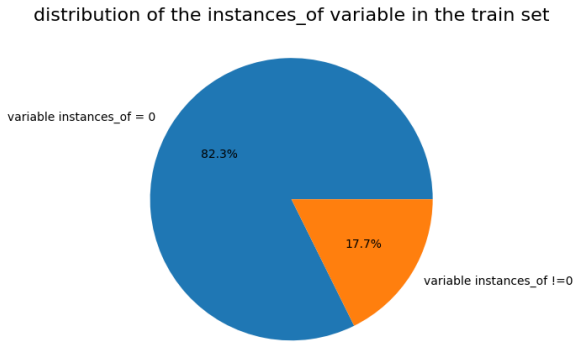

Figure 11: Pie chart of instances_of attribute in training set.

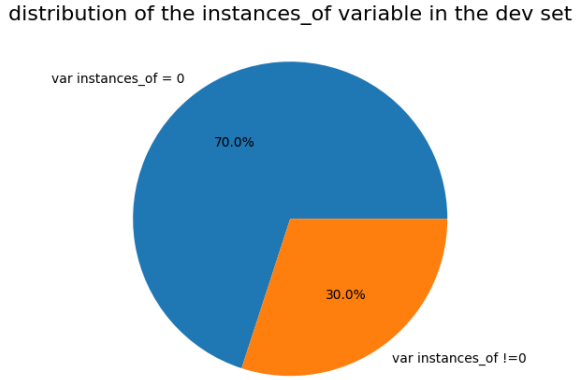

Figure 12: Pie chart of instances_of attribute in dev set.

A.12 Plots of labels distribution with respect to features

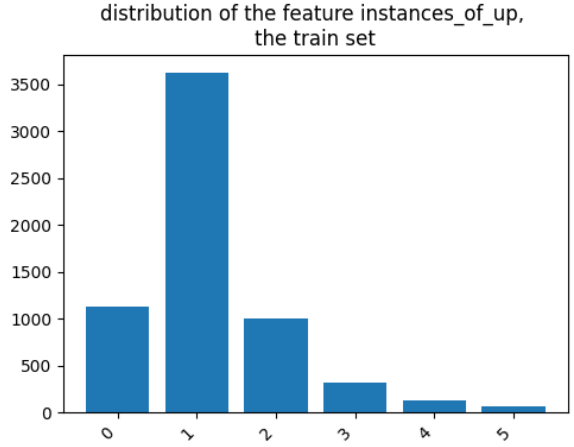

Figure 13: Distribution of upward instance links in training set.

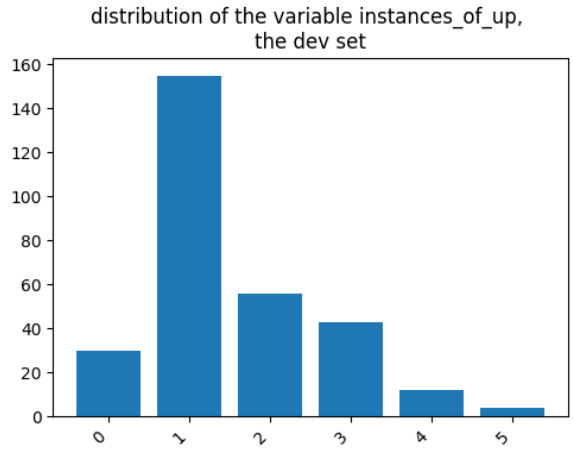

Figure 14: Distribution of upward instance links in dev set.

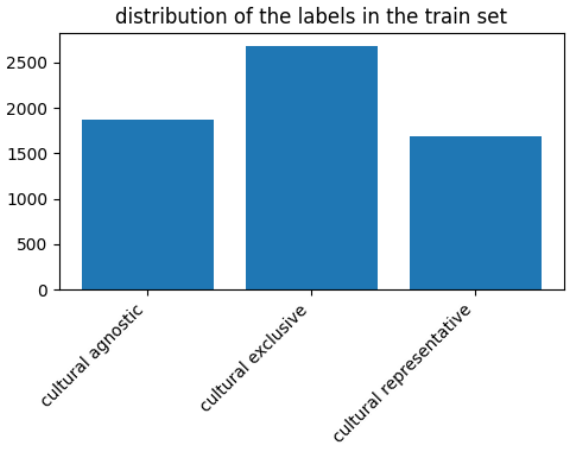

Figure 15: Label distribution in the training set.

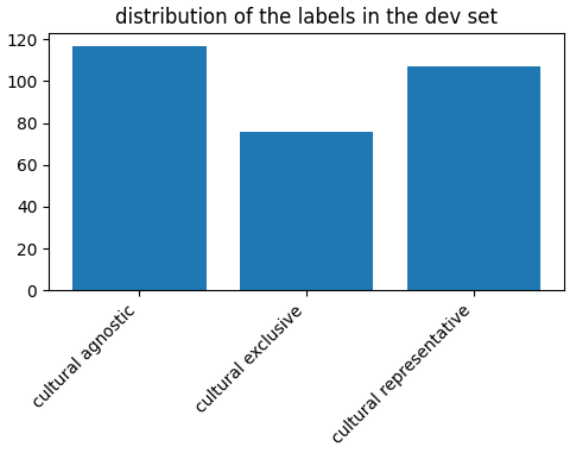

Figure 16: Label distribution in the development set.

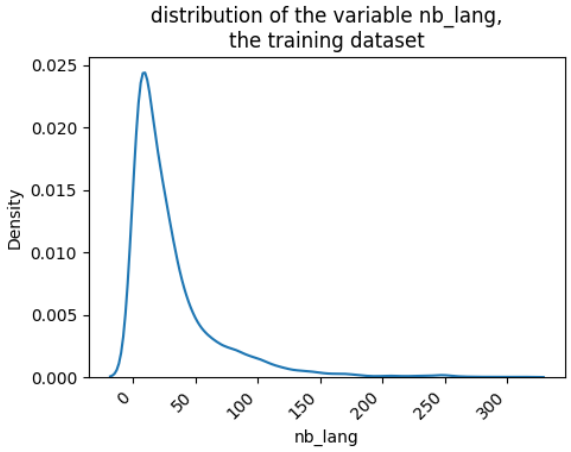

Figure 17: Distribution of the number of language entries in Wikidata.

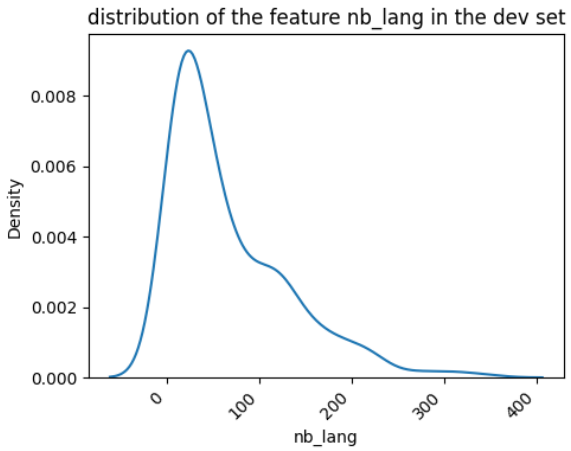

Figure 18: Language count distribution in the development set.

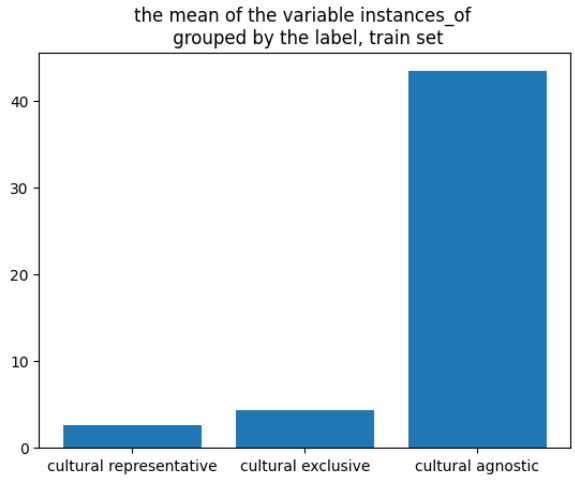

Figure 19: Mean number of downward instances in training set.

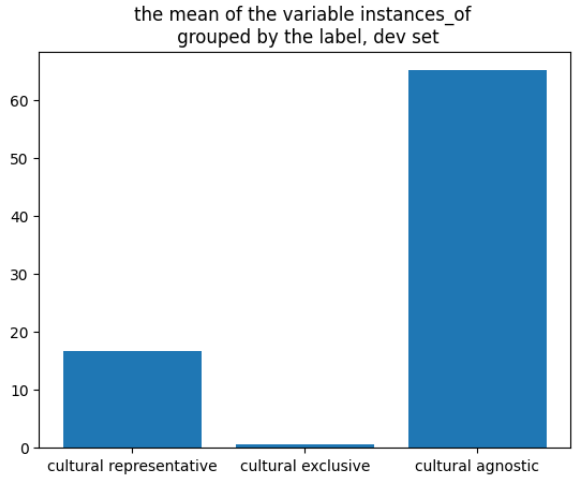

Figure 20: Mean number of downward instances in dev set.

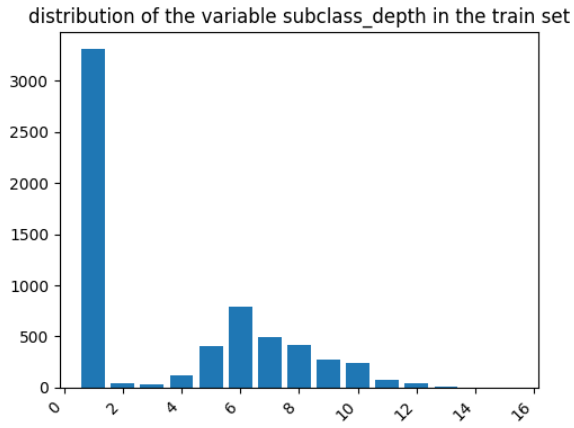

Figure 21: Distribution of subclass depth in training set.

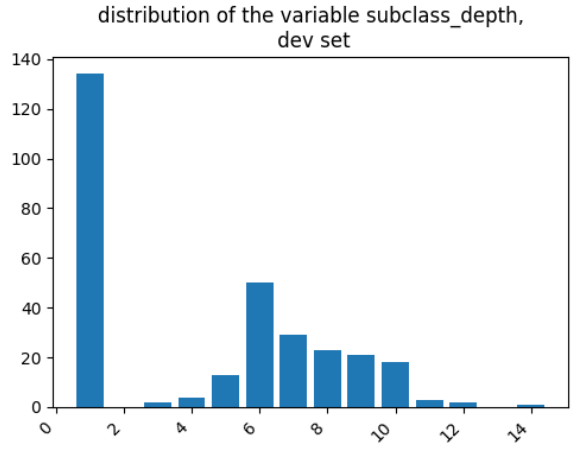

Figure 22: Distribution of subclass depth in the dev set.

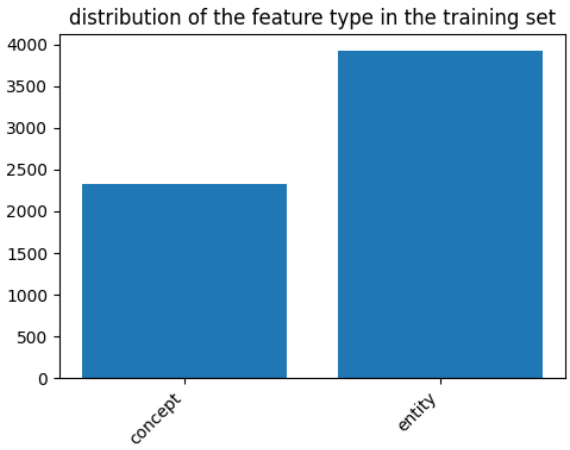

Figure 23: Entity type distribution in training set.

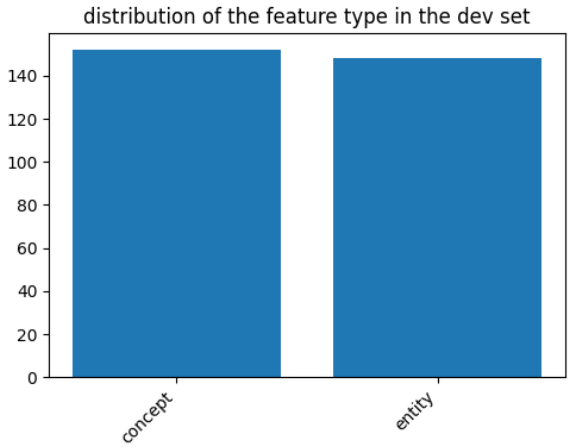

Figure 24: Entity type distribution in development set.

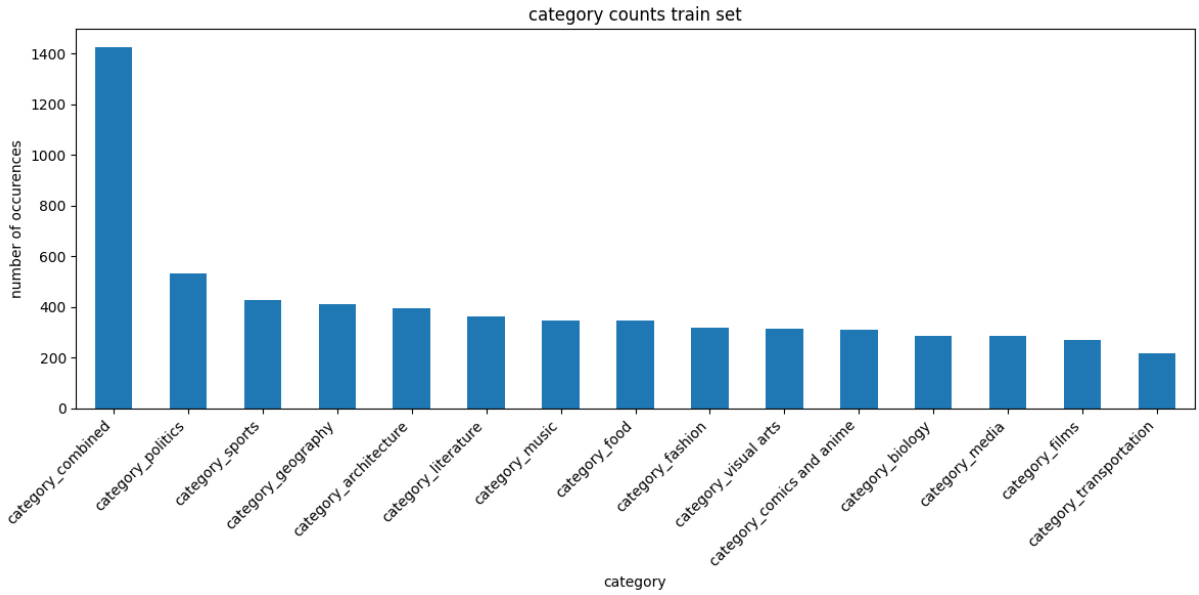

Figure 25: Distribution of categories in the training set.

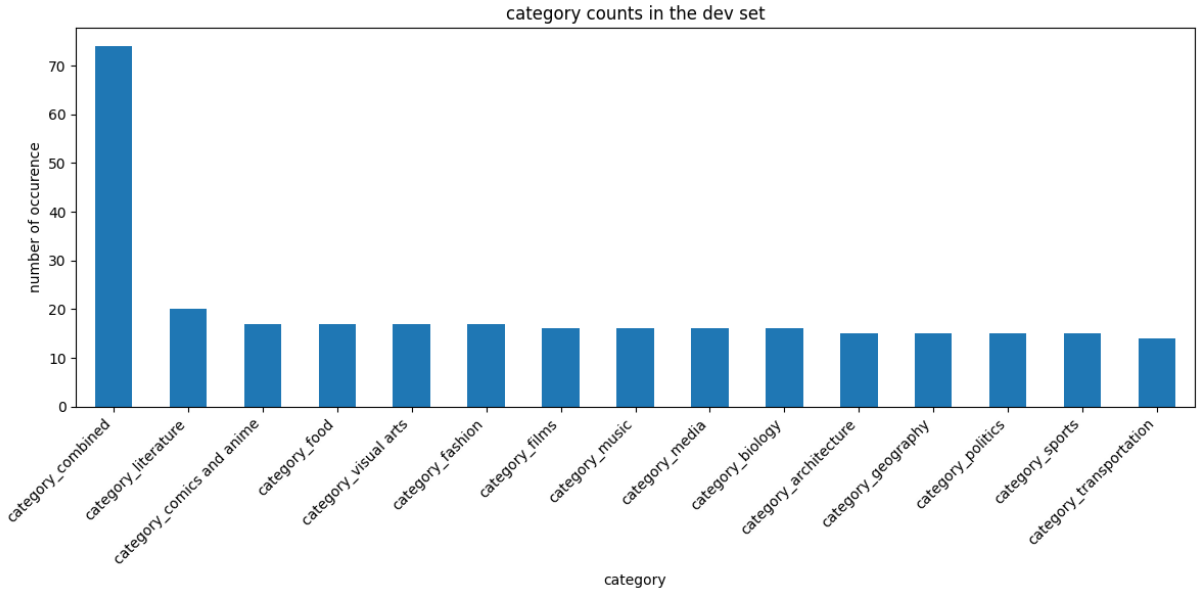

Figure 26: Distribution of categories in the development set.

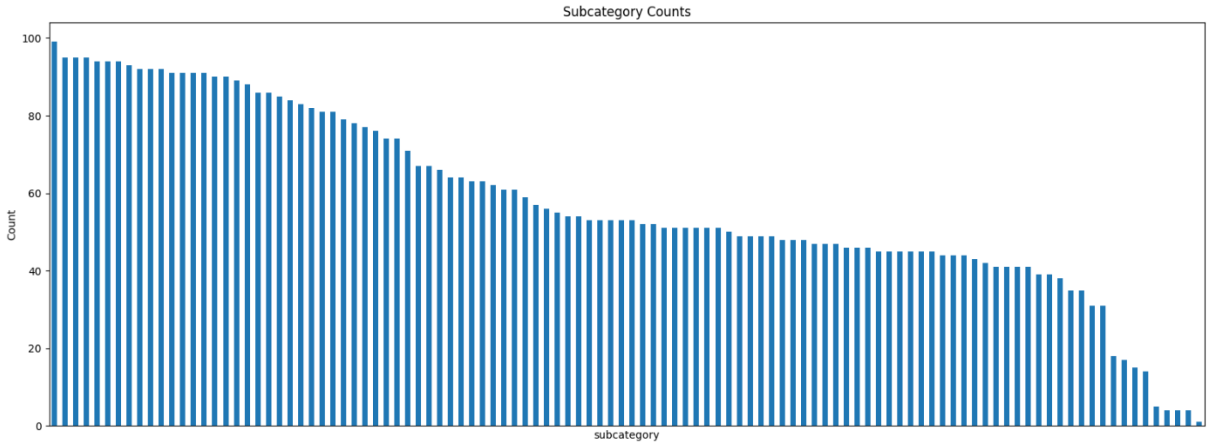

Figure 27: Distribution of subcategories in the training set.

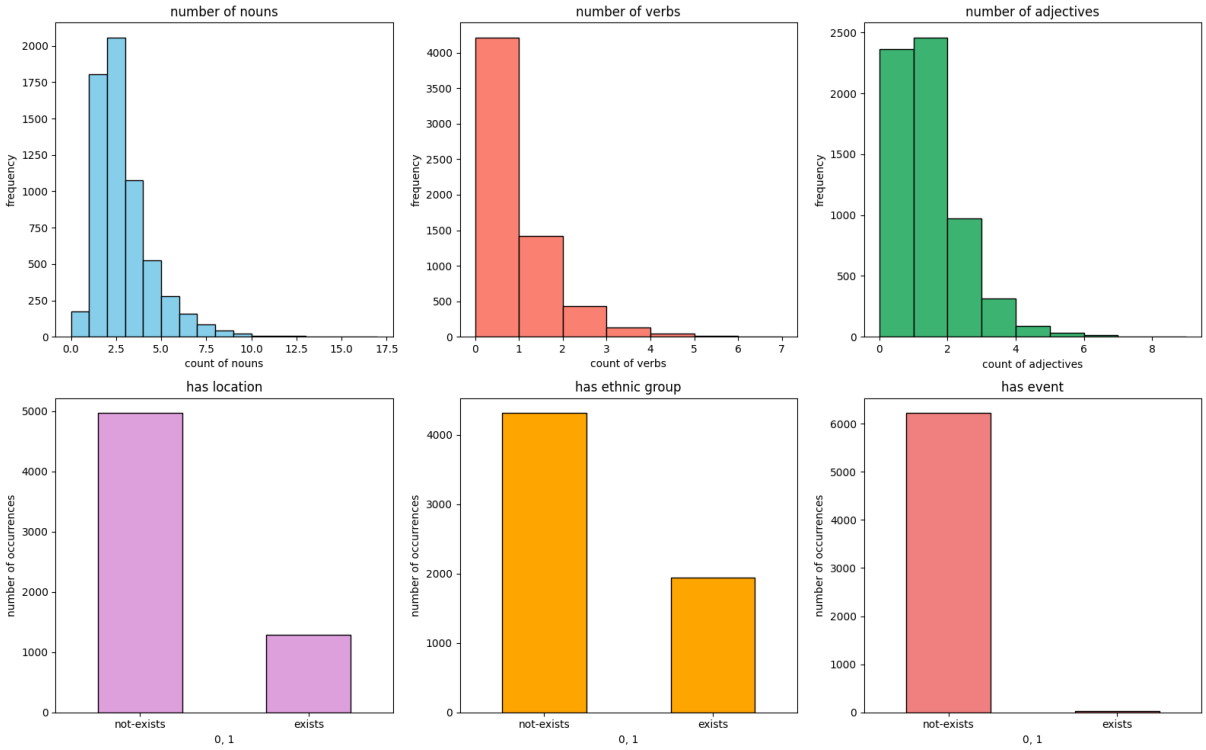

Figure 28: Description-based feature distribution in the training set.

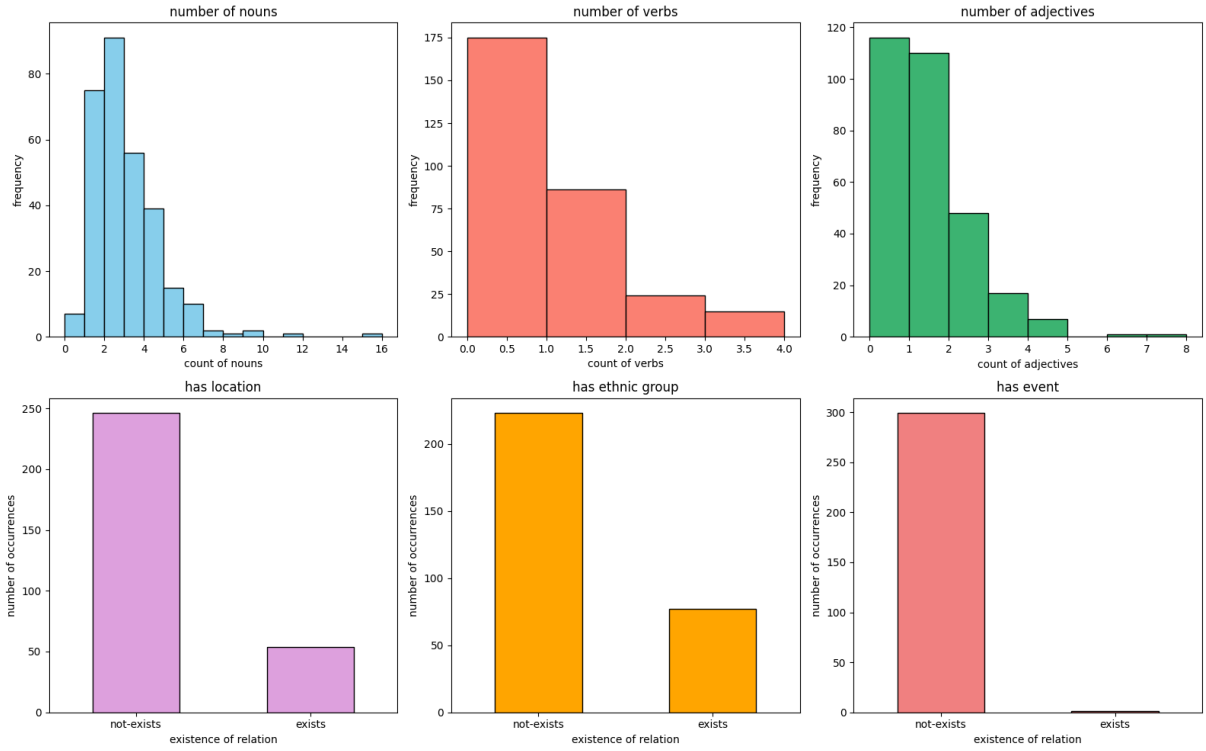

Figure 29: Description-based feature distribution in the development set (clipped).

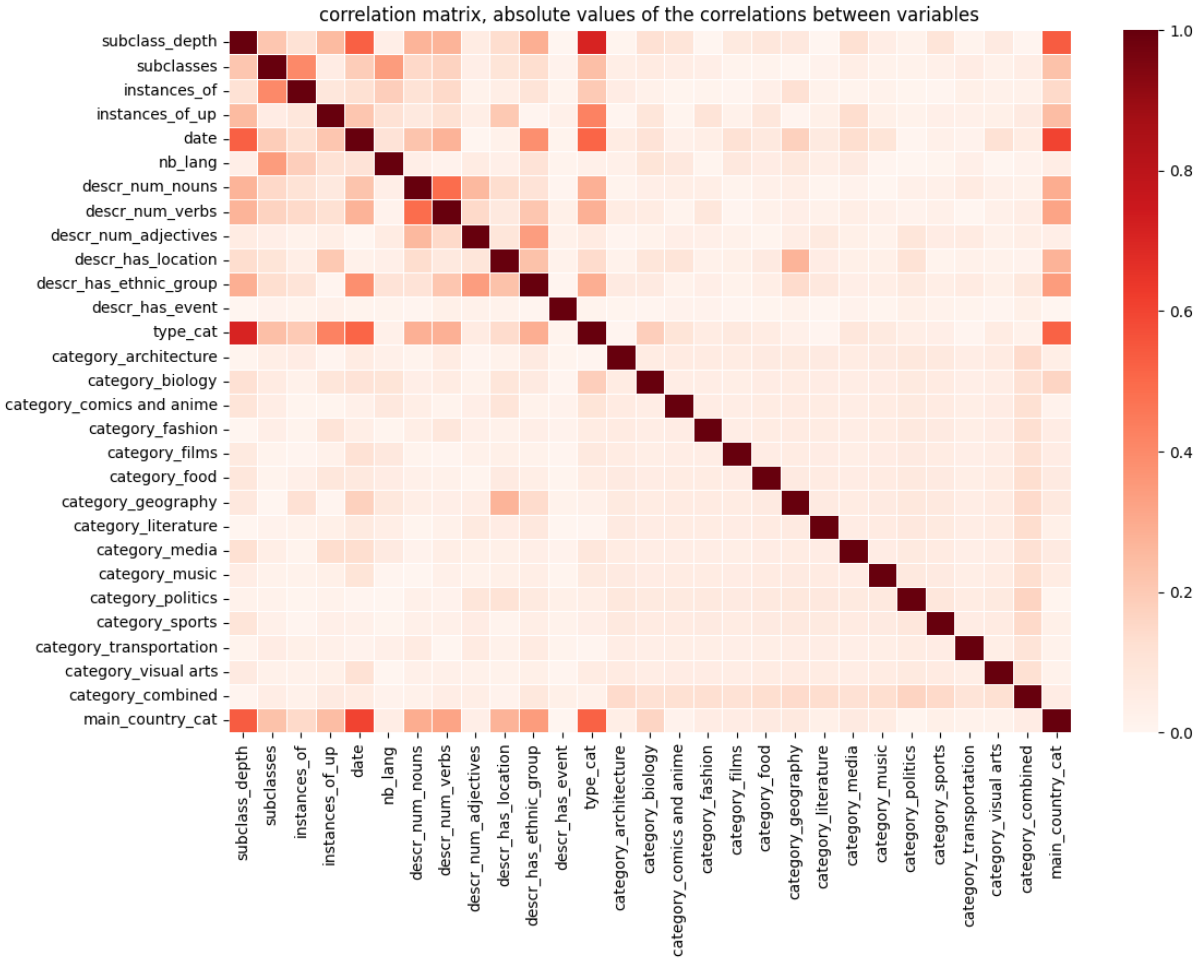

Figure 30: the absolute value of the Correlation matrix of selected features.

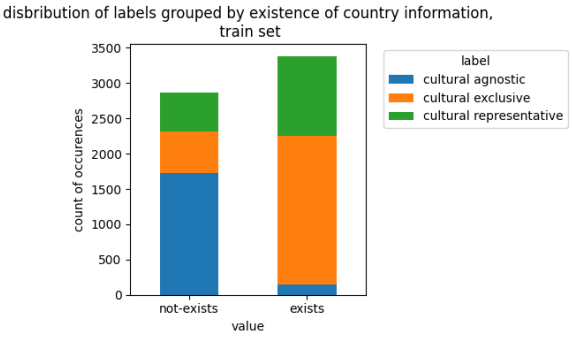

Figure 31: Country label presence in training data.

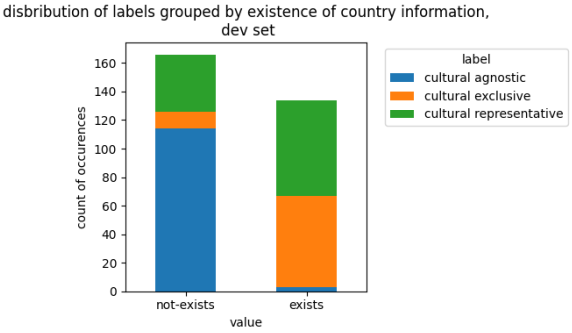

Figure 32: Country label presence in development data.

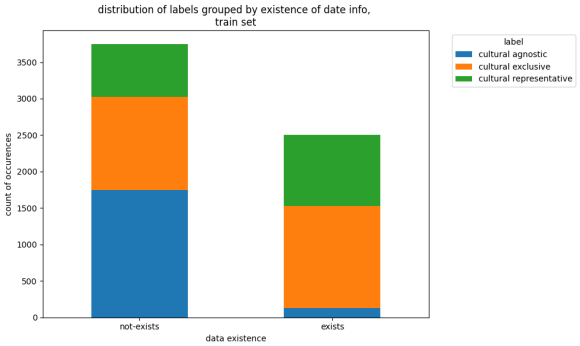

Figure 33: Date attribute presence in training data.

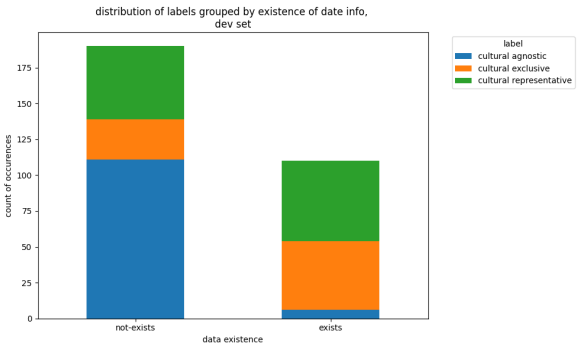

Figure 34: Date attribute presence in development data.

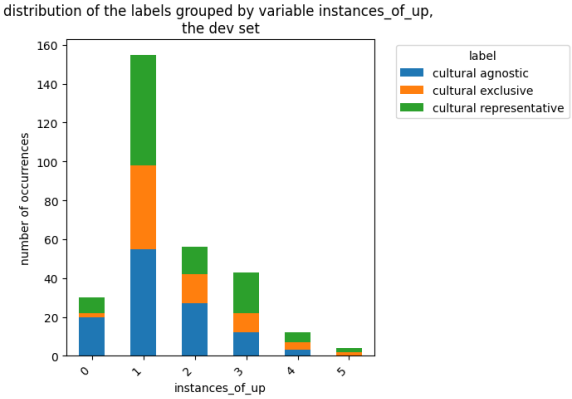

Figure 35: Distribution of upward instance-of labels in dev set.

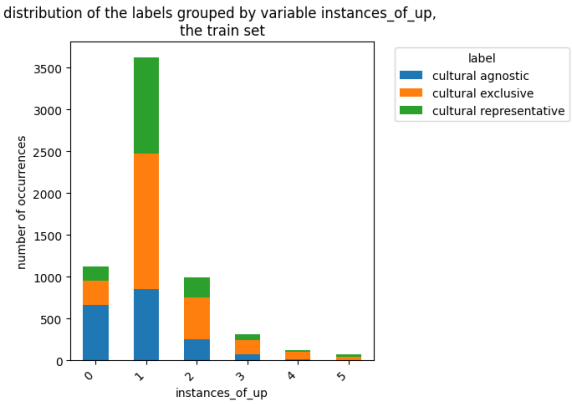

Figure 36: Distribution of upward instance-of labels in training set.

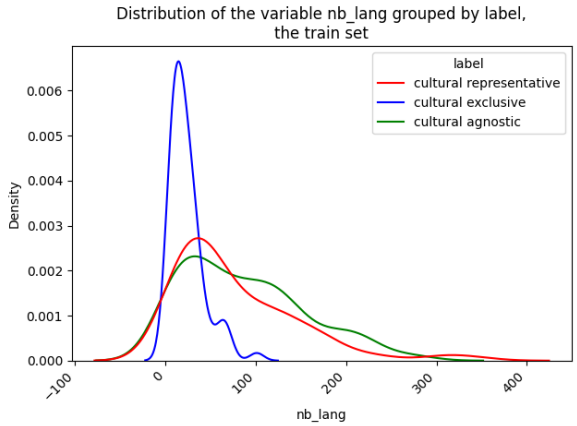

Figure 37: Language grouping distribution in training data.

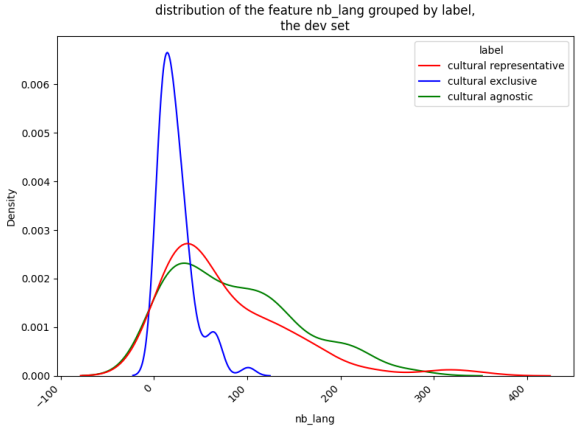

Figure 38: Language grouping distribution in dev data.

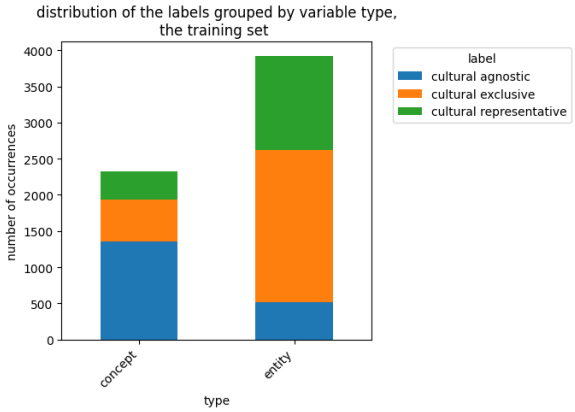

Figure 39: Entity type label distribution in training set.

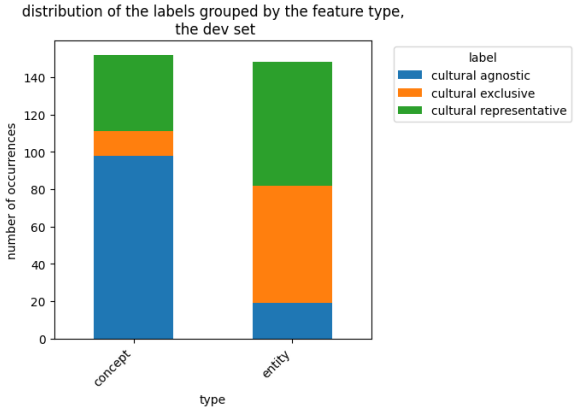

Figure 40: Entity type label distribution in development set.

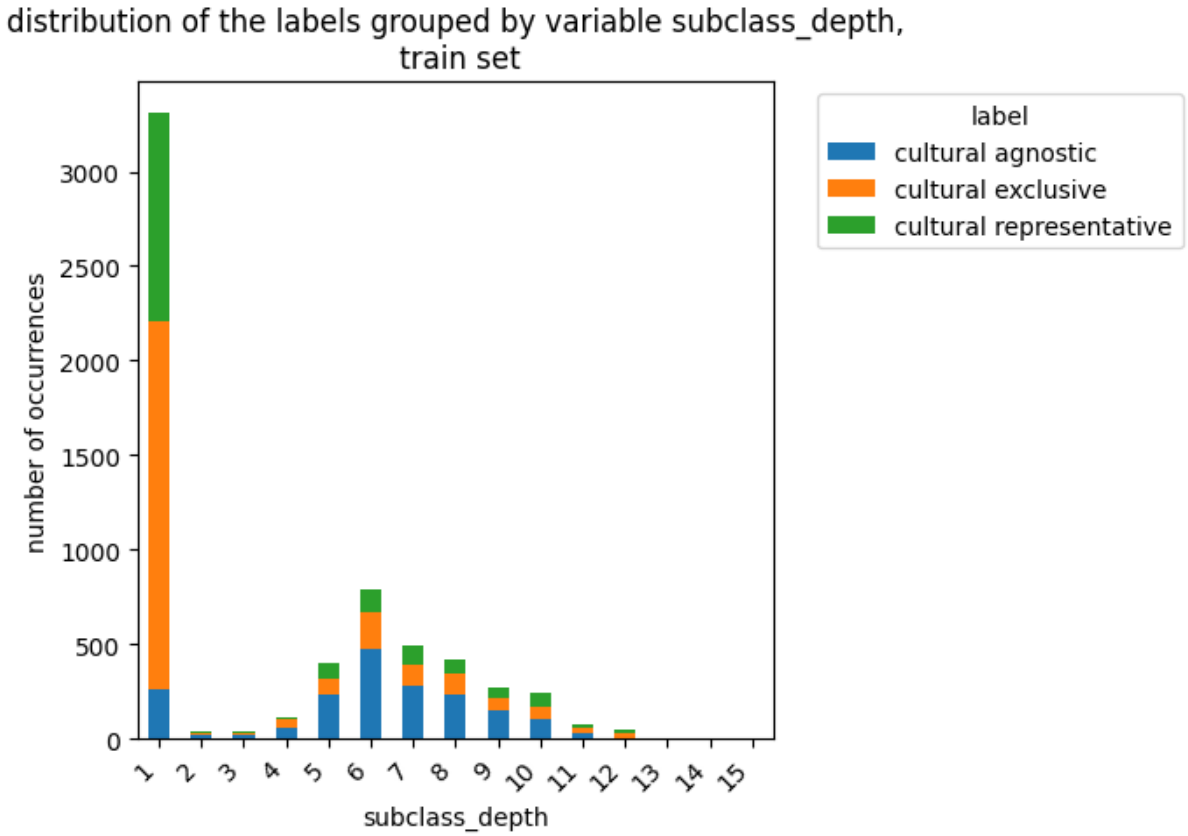

Figure 41: Subclass depth distribution in the training set.

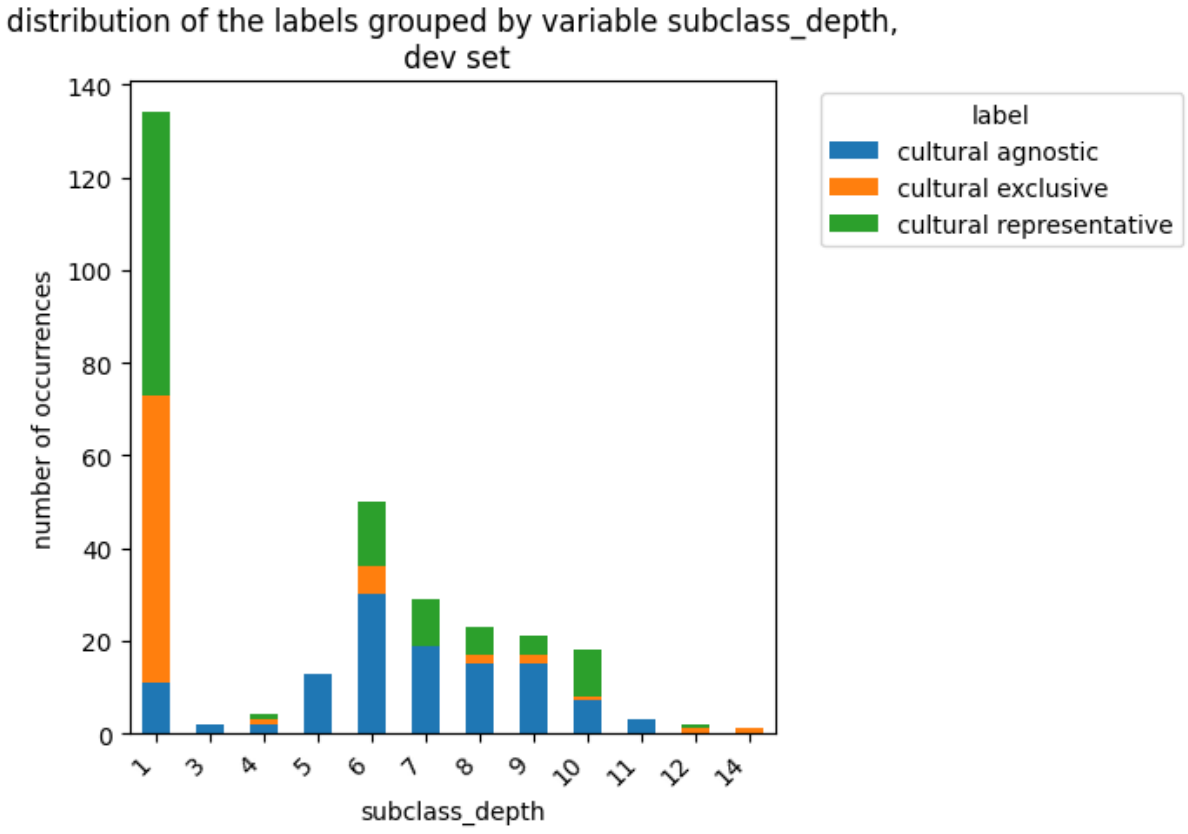

Figure 42: Subclass depth distribution in the development set.

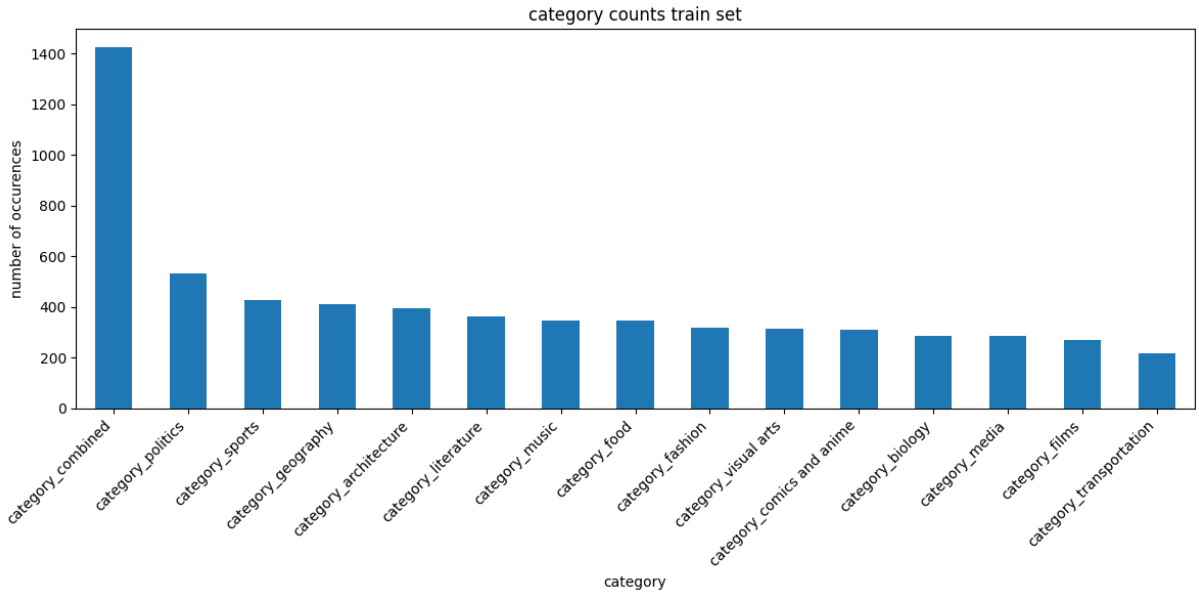

Figure 43: Distribution of categories in the training set.

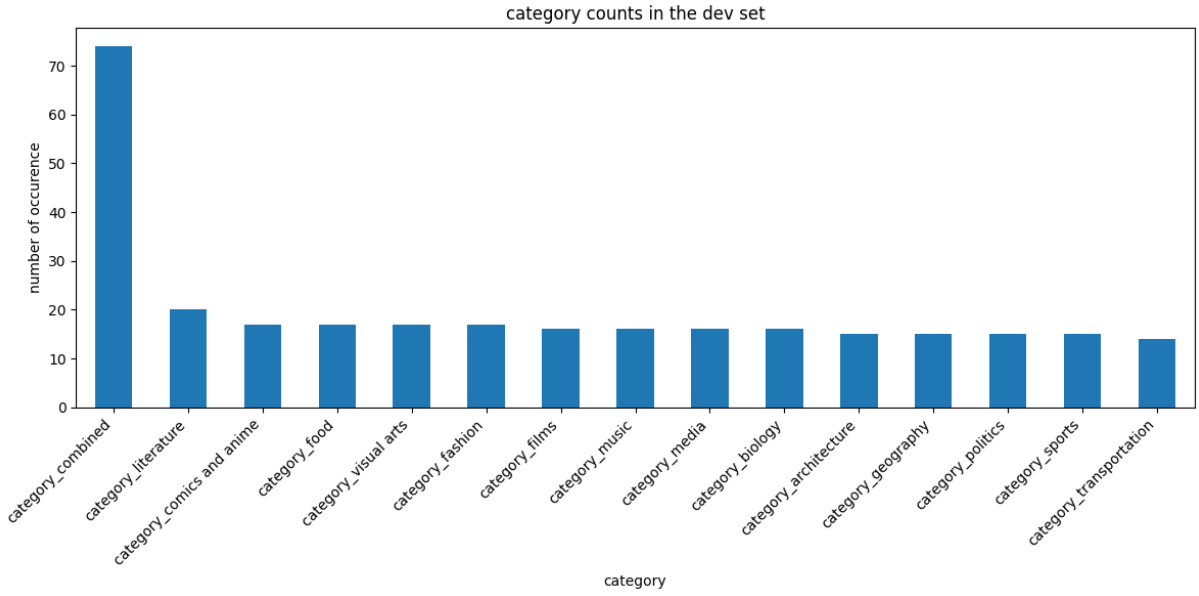

Figure 44: Distribution of categories in the development set.