Imre Gémes 2024.06.16.

Machine Learning Project – Multi-Label Prediction for News Articles

In this short paper, I summarise the final project of the Machine Learning course held at the Rajk College for Advanced Studies in the 2024 Spring semester. The goal of the project is to construct an algorithm to predict news article labels (such as ’Sports’, ’Tech’, etc.). Using Telex articles, I construct different multi-label prediction models that enable articles to get multiple labels at the same time. Creating efficient and uniform labeling can enable more sophisticated recommendations and filtering and provide a better consumer experience. Also, the project provided an excellent learning opportunity for me to experiment with different ML models and to dive deeper into the semantic analysis of texts.

This paper is organized into four chapters. In the first and second chapters, I briefly describe the data collection and the data preparation procedure respectively. In the third chapter, I explain in more detail four different models that I constructed for label prediction. The final chapter concludes.

1. Data Collection

I construct a scraping algorithm to download news articles from telex.hu from 1. January 2023 to 12, May 2024. In this more than one-year interval, I obtain the link of the article, the title, the author(s), the label(s), and the text of the article. Using the label search engine of the website, I iterate through the most frequent labels and download the corresponding articles. These labels are 'Belföld', 'Külföld', 'Gazdaság', 'Életmód', 'Sport', 'Techtud', 'After', and 'English' labels. I download the data in slices: 100 observations for each .csv file. Due to the wide variety of text structures used by Telex, some articles proved to be hard to handle (eg.: bullet point articles and pictures only). I dropped these articles.

2. Data Preparation and EDA

The label prediction consists of two fundamental parts: the labels as outcomes and the features that the predictions are based on.

It is important that a single article can take multiple labels at the same time. To investigate article labels, I first created a binary variable for each label using one-hot encoding. This way, the binary variables indicate if an article can be categorized into a label group or not.

The features are constructed by adding the semantic vectors corresponding to the texts of the articles. I use the „Fasttext" vectorizer which was trained explicitly on Hungarian texts. This model uses n-grams to subtract subword information, which is beneficial for processing morphologically rich languages like Hungarian. With „Fasttext", I create 300 new variables, representing the semantic vectors, based on the texts exclusively.

To see if there is an apparent separation between labels in the semantic vectors, I carried out DBSCAN and HDBSCAN clustering. After clustering, I did a dimension reduction using UMAP, and plotted the semantic content of the articles in a 2-dimensional space, colored by the clustering results.

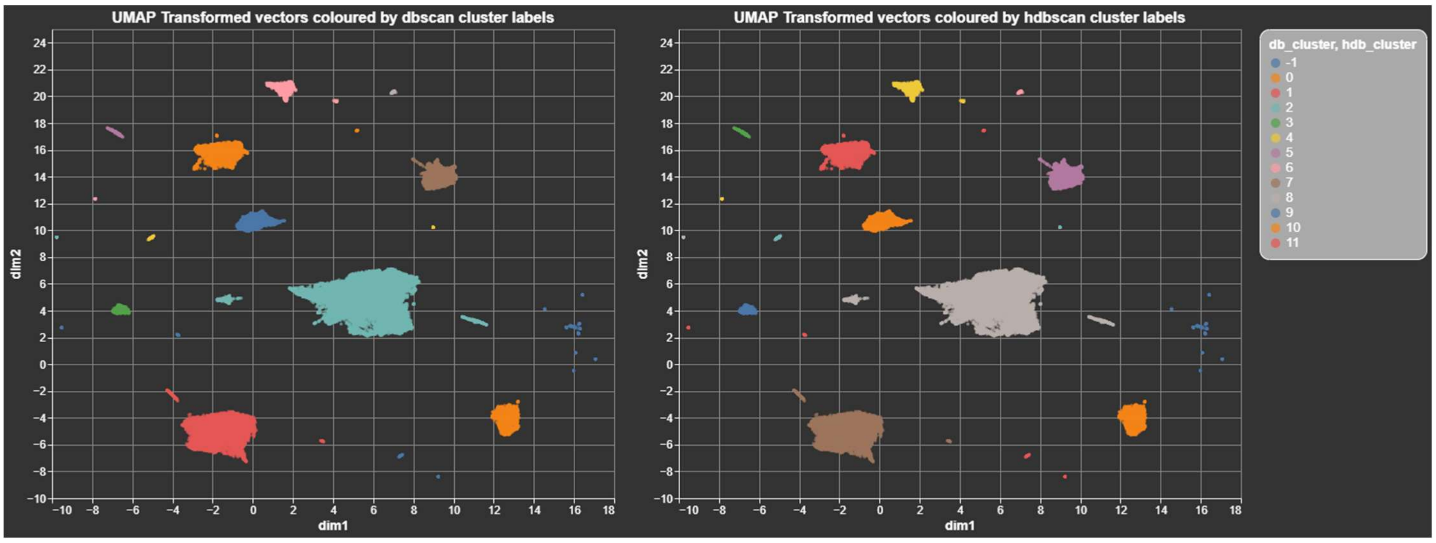

Figure 1: Clustering results based on semantic vectors. Dimension reduction from 300 to 2 variables. Own figure

Figure 1 suggests that based on the semantic vectors, the articles are well separable from each other. Since the semantic vectors are supposed to describe the content of the texts, just like the labels, it is reasonable to assume that article labels are well predictable from these features.

3. Label Prediction Models

In this chapter, I go into more detail about how I predict the labels of the articles. I construct four different models to predict labels. These models are namely a Logit model, a Bagging model, a Random Forest model, and a Gradient Boosting model. All these models are used to predict binary labels one by one, separately from each other. Consequently, I construct four separate models for each label.

Before constructing the models, it is important to understand the implications that an article can have multiple labels. In the notebook, I explain that if there is a huge label overlap, label prediction becomes more challenging. Fortunately, this is unlikely to be the case as labels tend to have a relatively small overlap.

3.1 Logit Model

First, I construct the Logit model. I split the data into three equal sizes: training, validation, and test set. Using all features, I fit the Logit models on the training set. Since label distribution varies, I use balanced class weights to correct for this imbalance.

In the case of a Logit model, the precision and the recall can be improved at the expense of each other. To find a balanced prediction, I experiment with various threshold levels for each label and plot the corresponding accuracy, precision, recall, and F1 scores, obtained on the validation set. I choose the F1 score to be the main performance indicator metric of my models as it provides a single, balanced metric of precision and recall.

It can be seen that different labels take on maximum F1 scores at different thresholds. I use these thresholds in the final model to maximize model efficiency. Using the trained models and the obtained thresholds, I predict labels on the test set. To evaluate the model, I print the performance by label.

Predicting 'English' articles massively overperform every other lables. The reason behind that is that vectorizers use the semantic content of a text to produce a vector. These vectorizers are trained for particular languages, therefore applying the same vectorizer on different language texts produces massively different vectors. As a result, the created vectors are much less meaningful for 'English' texts. But at the same time, the difference between the languages is obvious. This way, the semantic vector of 'English' texts is not meaningful, but they are efficient in predicting the label.

For those labels that appear frequently in the sample, the model performs relatively well but not for those labels that has much fewer articles. To see what kind of mistakes the model makes, I construct the confusion matrix.

Finally, to measure the model performance with a single number, I compute the weighted average of the F1 score, where the weights are the relative frequency of the label in the sample.

I will use this score to compare all the models. I was considering excluding 'English' articles from the evaluation as they form a completely different category. Finally, I included them too because they also needed to be categorized even if the corresponding semantic vectors were not meaningful. Including ’English’ labels did not influence model performance compared to each other because every model efficiently predicts 'English' labels.

3.2 Bagging Model

In Bagging, I split the data into a training and a test set. In this model, I create various samples, using bootstrapping. Then, I fit the same decision tree model on these samples. Since the bootstrap samples are somewhat different, the models built upon them might predict labels differently. These rather simple models capture a different aspect of the classification problem due to the bootstrapped sampling. Condorcet's Jury Theorem suggests that in this case, the majority vote should be better than any single model. Based on this fundamental idea, I predict labels using Bagging.

Again, similarly to the Logit model, I construct the evaluation table, the confusion matrix, and the single performance measure weighted F1 score.

3.3 Random Forest Model

The third model type is the Random Forest model. In this model, I split the data into training and test sets. Similarly to Bagging, Random Forest models also make use of bootstrap sampling. The main advantage of them is that Random Forests build models based on only a limited number of randomly selected features. For the selected sample, the model also selects some variables that the model is built upon. Since this introduces more randomness, the resulting models are less correlated, making them efficient tools in classification problems.

The procedure is similar to the previous ones: I build separate models for each binary variable. These models predict labels considering exclusively the semantic vectors of the articles. Then, I predict the labels in the test set and construct the evaluation metrics, the confusion matrix, and the single performance measure.

3.4 Gradient Boosting Tree Model

Finally, a fourth model is constructed to predict article labels. I use XGBoost from the Gradient Boosting framework. This model uses decision trees as weak learners. This is a sequentially constructed model, which means that after constructing and evaluating a model, it practically learns from its previous mistakes by assigning extra weights to the misclassified observations. As a result, the model gradually gets better after each iteration, until reaching its maximum effectiveness.

Results and Concluding Remarks

The final, weighted average F1 Scores of the models can be seen in Table 1. The Gradient Boosting Tree Model outperformed the others, followed by the Logit model, the Random Forest Model, and the Bagging Model.

Model	Weighted Average F1 Score
Logit Model	0.7759
Bagging Model	0.7578
Random Forest Model	0.7634
Gradient Boosting Tree Model	0.8460

Table 1: Weighted average F1 score by models. Own table.

It might be surprising how efficiently the Logit model performed compared to Bagging and Random Forest. The reason behind that is that the Logit model was optimized for maximum F1 score, while this was not possible directly for the other models. It could be possible to construct the other models with various hyperparameters and carry out a grid search to find the maximum F1 score for those as well. However, due to computation capacity, I did not follow this idea.

It is also noteworthy that feature importance distribution tends to be more uniform across different labels, except for the ’English’ labeled articles, where feature importance distribution is much more skewed.

Further development possibilities for the project could be extending models with a grid search part, optimizing for F1 score. Also, these models predict labels one by one, disregarding other labels. There are more complex, sophisticated methods to predict multiple labels at the same time.