News article classification

Karmen Gostiša

October 6, 2020

1 Introduction

This report describes my news classification project.

2 Data

Data consisted of 4036 news article URLs and their category from Zurnal24 web portal in the following structure:

URL	Category
https://zurnal24.si/galleries/gallery-39287	svet
http://zurnal24.si/galleries/gallery-12787	sport
http://zurnal24.si/galleries/gallery-168411	svet
...	...

2.1 Web scraping

The first step of acquiring the dataset was to write a web scraper that parsed HTML files and extracted relevant content (title and text of the article).

After the manual inspection of HTML I found out the relevant content is contained in various HTML tags such as `title`, `meta content` and `article__content`. For pulling the data out of a HTML file I used a library Beautiful Soup. Before saving the text into a file I removed redundant HTML tags, unicode characters and new lines. The final text contained the title and content, all in a human readable form (for now).

I removed one article from the original dataset (line 4035) as it was from different web journal and it needed other parser.

The web scraper is contained in a script named `download.py`. The script downloads the articles as TXT files and outputs them in a folder.

2.2 Exploratory analysis

I was interested to see the list of categories and the number of articles falling under each category. I also calculated the average length of article in each category. Following are the results (obtained by calling the `explore` function in `evaluate.py`):

Category	Number of articles	Average article character length
sport	1717	1562.90
magazin	1307	982.07
slovenija	556	1391.03
svet	355	1138.66
avto	100	2581.25
	4035	

The results tells us the classes are unbalanced and that the "avto" category contains the longest texts which is good as this category has the smallest number of articles in the dataset.

2.3 Cleaning

Before creating the text features I performed the basic text cleaning taking the following steps:

1. **Punctuation and special characters:** Removed those using a function from the Python string module.
2. **Lowercase:** Lowercased all text.
3. **Stop words:** Removed those with the help of a list containing Slovenian stop words I found on Github (link). Later I also added few new stopwords to the list.
4. **Lemmatisation:** Lemmatised words using the LemmaGen module (link) supporting various languages, also Slovenian, among others.

I stored data in pandas DataFrame into following columns: URL, Category, Category_Code (numeric ID for Category) and Text (cleaned).

2.4 Text representation

I chose to represent the articles as TF-IDF vectors with the following parameters:

- N-gram range: 1,
- Maximum Document Frequency: 1,
- Minimum Document Frequency: 10,
- Maximum features: 300.

3 Classification results

I developed two classification models: Multinomial Naive Bayes and Multinomial Logistic Regression. I also implemented the Random Search to find the best parameters for the Multinomial Logistic Regression.

For evaluation I split the dataset into training (85%) and test (15%) set. Then I trained models, made predictions and calculated accuracy values on both training and test set. For the baseline model I took the classifier that always predicts the majority class so its accuracy represents the percentage of samples classified in majority class (in our case the sport category). The accuracy results for training and test dataset are summarised in Table 1. Table 2 depicts the test accuracy values per category in test dataset.

	Training accuracy	Test accuracy
Baseline	0.421	0.449
M. Naive Bayes	0.833	0.830
M. Logistic Regression	0.892	0.876

Table 1: Training and test accuracy values for different models.

	M. Naive Bayes	M. Logistic Regression
avto	8/9 = 0.889	11/12 = 0.917
magazin	168/222 = 0.757	171/212 = 0.806
slovenija	47/70 = 0.671	54/71 = 0.761
sport	263/285 = 0.923	266/275 = 0.967
svet	17/20 = 0.850	29/36 = 0.806
	503/606 = 0.830	531/606 = 0.876

Table 2: Test accuracy values per category and overall for both models.

For better insight into classification results I plotted the confusion matrices depicted in Figure 1 and Figure 2.

4 Conclusion

The results showed both supervised models are good classifiers for our data, with Multinomial Logistic Regression having slightly higher accuracy. Training accuracy values showed the models did not overfit.

The results could be further improved by fine-tuning the parameters, for example in Logistic Regression we could perform a more exhaustive search around the values of hyper parameters we got from the Random Search (currently using the training data, but this should be a validation set, so this is another thing to consider in future work). Different combinations of parameter values for constructing TD-IDF vectors could also be tried.

An important aspect that could also improve classifiers is to balance the dataset, possibly by over or undersampling. For example, since the "avto" category contains the longest articles but is undersampled in our original dataset, we could split those articles in two and that way double the number of "avto" samples. Since the accuracy values for this category is pretty high (as

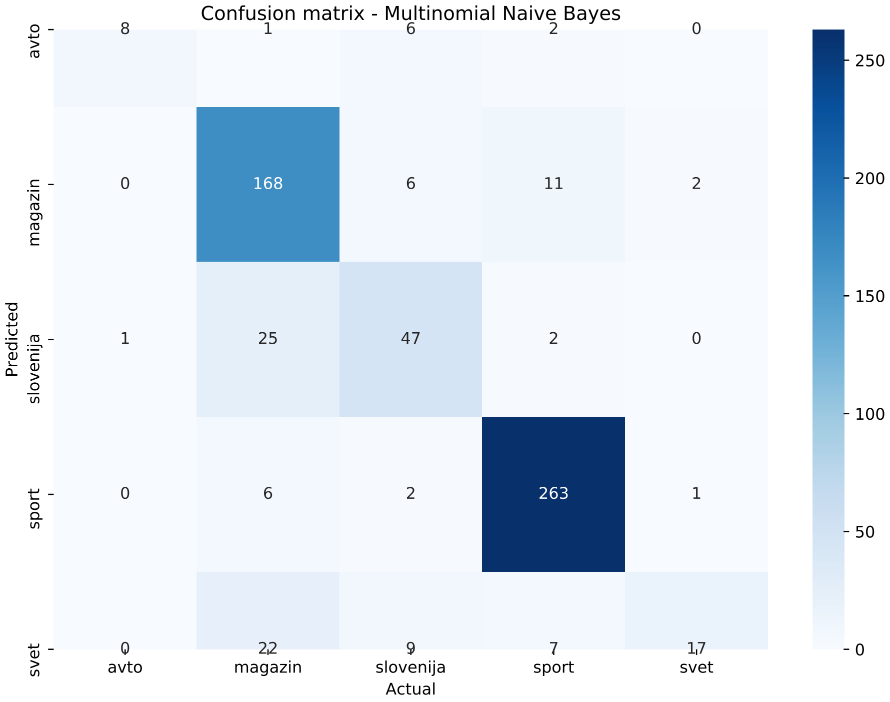

Figure 1: Confusion matrix for the Multinomial Naive Bayes model. Y-axis shows the predicted category and x-axis the actual. Values in the matrix are number of articles.

opposed to “slovenija” or “magazin”) I left samples as they were. The over or undersampling technique could improve the classification of samples into “slovenija” or “magazin”.

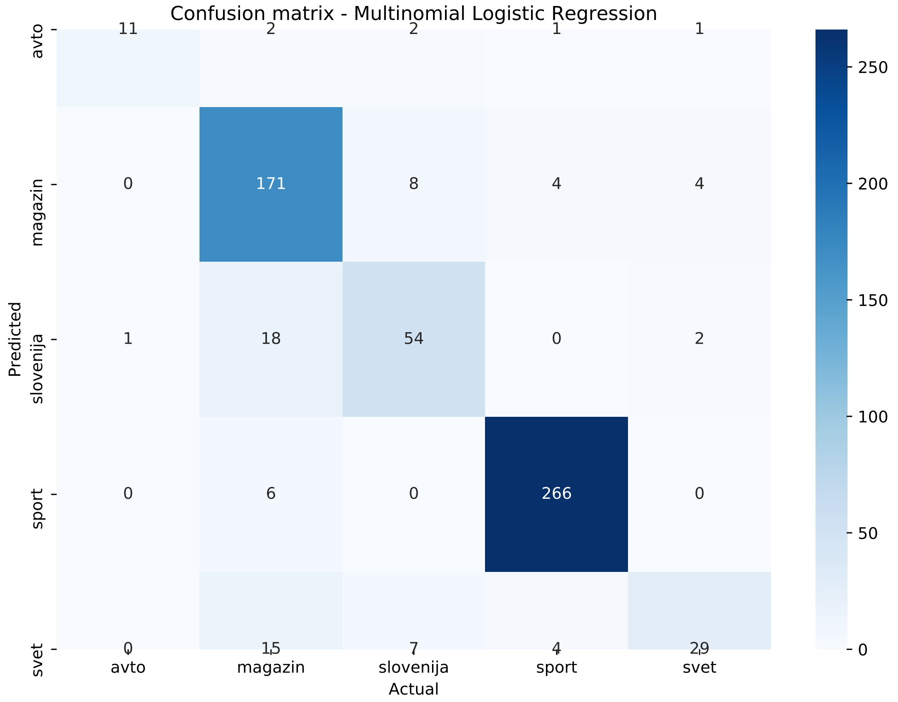

Figure 2: Confusion matrix for the Multinomial Logistic Regression model. Y-axis shows the predicted category and x-axis the actual. Values in the matrix are number of articles.