Machine Learning for NLP
Lab Report of Exercise 1

October 15, 2022

1 Data Inspection

The train-dev data consist of 52675 records. Each record consists of a tweet and a class label that indicates the language of the tweet. In total, there are 69 different class labels in the train-dev set.

As for all classification problems, there is a risk that the test data contains class labels that are not contained in the train set and, hence, were never seen by the model. A possible approach to resolve this issue would be to classify a tweet as *Unknown* if the Softmax output (probability) for all labels is below a certain threshold.

Another challenge is illustrated in Figure 1. Most of the class labels appear only very rarely in the train-dev set. For example, the three most frequent labels (English, Japanese and Spanish) account for 66.2% of the train-dev set (see Table 1). Moreover, 59 classes together share just 4.9% of all train-dev records. This class imbalance could potentially lead to challenges for the classifiers as there is only very little data on a lot of the classes.

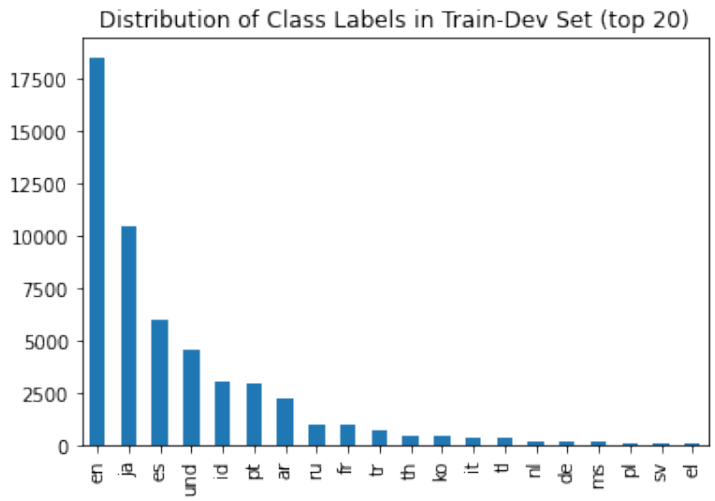

Figure 1: Distribution of top 20 Class Labels in Train-Dev set

Label	Cumulative Sum (in %)
en	35.1
ja	54.9
es	66.2
und	74.8
id	80.5
pt	86.0
ar	90.1
ru	92.0
fr	93.8
tr	95.1

Table 1: Relative Frequencies of top 10 Class Labels in Train-Dev set (Cumulative)

2 Language Identification with Linear Model (Part 1)

Given a tweet, the task at hand is to predict the tweet's language using a linear model (logistic regression (LR)).

2.1 Preprocessing Steps

The following preprocessing steps were applied:

(i) **Clean URLs:** Replace all URLs in a tweet with the empty string. Reasoning: A URL does not contain any information about the language of a tweet.

(ii) **Clean Usernames:** Replace all usernames in a tweet with the empty string. Reasoning: A username does not contain any information about the language of a tweet.

(iii) **Clean Emojis:** Replace all emojis in a tweet with their corresponding description in words. For this, we made use of the package *emoji* and its function *demojize()*. Reasoning: Some cultures (and their languages) might use emojis in tweets more frequently than other cultures (and their languages). Hence, this could be of valuable predictive information.

2.2 Additional Linguistic Features

Besides the TF.IDF-vectorizer we use the following linguistic features in our model:

(i) **Average Word Length:** This feature captures the average length of a word in a given tweet. The reasoning behind this feature is that some languages might contain fewer characters per word than others. This characteristic could then be exploited by our classifier.

(ii) **Tweet Length:** This feature captures the length of a tweet. The reasoning behind this feature is that for some languages, fewer words might be needed to express a given semantic meaning of a sentence.

Since both linguistic features are numerical it is crucial to perform *Standardization* before feeding them into a model. This is especially true for models that use distance-based elements in their solvers such as L1 or L2 regularization of linear models. Consequently, we standardized these two features to have zero mean and unit variance. This step is also integrated in our sklearn pipeline.

2.3 Grid Search to Find Optimal Hyper-Parameters

In order to find the optimal hyper-parameters of the Logistic Regression, the grid search space specified in Table 2 was explored. Please note that due to computational limitations, the maximum number of iterations taken for the solvers to converge was set to an extremely low value of 2.

	Hyper-Parameter	**Explored Values**
Classifier	Penalty	*L2, None*
	C	*0.1, 0.5, 1.0*
	Solver	*newton-cg, lbfgs*
	Max iteration	*2*
Vectorizer	Ngram range	*(1,1), (1,2)*
	Max features	*500, 1000*

Table 2: Grid Search Space of Linear Model

As specified in the exercise statement, the different models were trained on 90% of the train-dev set (i.e., on the train set) and validated on the remaining 10% (i.e., the development set). The hyper-parameter combination of the best-performing model on the validation set is reported in Table 3.

	Hyper-Parameter	**Best Value**
Classifier	Penalty	*None*
	C	*0.1*
	Solver	*newton-cg*
	Max iteration	*2*
Vectorizer	Ngram range	*(1,1)*
	Max features	*1000*

Table 3: Hyper-Parameters of Best-Performing Linear Model

Grid search cross-validation combines two ideas of training a machine-learning model:

- **Grid Search:** This is the idea of trying out different combinations of a model's parameters. The reasoning being that depending on the specific data at hand, different parameter combinations might perform better than others.
- **Cross-Validation:** This is the idea of further splitting up the data used during the training process into a training and development (or, validation) set. A model is then trained on the training set and validated on the held-out validation set. This gives a more realistic picture on a model's performance as the validation data was not seen during the training/model-fitting process.

Combining these two ideas yields grid-search cross-validation where different parameter combinations are systematically explored (using the predefined grid search space) and realistically compared using cross-validation. Once specified, this entire process happens automated and without any user interaction. Consequently, cross-validated grid search is an effective way to find the best parameters of a machine-learning model for a given task and data set.

2.4 Confusion Matrix and Error Analysis

The overall accuracy of the best-performing model (see Table 3) on the unseen test set is **59.9%**.

The Confusion Matrix of the top 10 classes with values normalized to true labels is displayed in Figure 2. This is, each row sums to 1.0 and the cell value c(i, j) indicates the percentage of class i being predicted as class j. For example, given a Japanese tweet, the model (falsely) predicts the tweet to be English in 63 out of 100 cases.

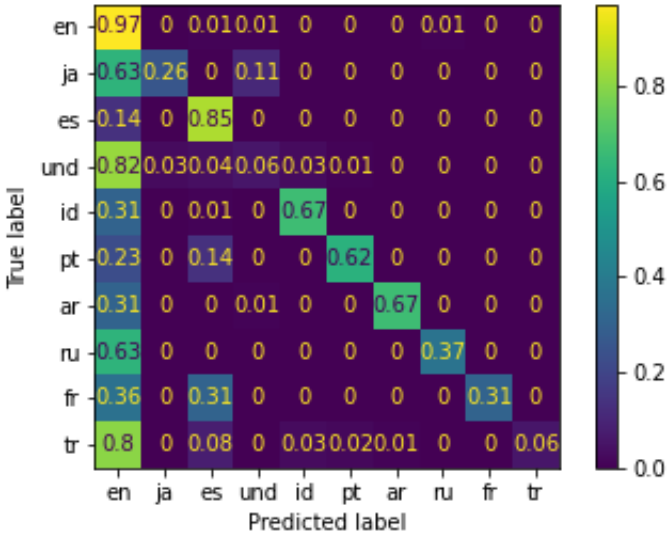

Figure 2: Confusion Matrix for Predictions on Test Set (Normalized to True Labels) of LR

Even though the percentage of 97% for English tweets being correctly classified as English might seem high at first glance, this performance is set into perspective when looking at the entire first column. It becomes apparent that the model simply predicts the English label very often. Combined with the class imbalance observed in section 1, the high percentage of 97% has an obvious explanation.

Overall, the model seems to perform reasonably well for English (en) and Spanish (es) tweets and mediocre for Indonesian (id), Portuguese (pt) and Arabic (ar) ones. For the remaining languages in the top 10, the model performs rather poorly.

2.5 Feature Importance Table

The top 10 feature importance table of the best-performing linear model for the languages English, Spanish and Japanese is shown in Table 4.

It can be observed that for both English and Spanish, the outputs of the vectorizer seem to be more important than the extra linguistic features. For Japanese, however, the two additional linguistic features (average word length and tweet length) rank among the most important features. Even more interesting is the fact that two Arabic words (الله, which translates to "allah" in English, and من) seem to be highly important in classifying a Japanese tweet.

Furthermore, the bias term ranks as the most important feature for both English and Japanese tweets. An explanation for this could be the fact that these two classes are the most frequent ones among all

y=en		**y=es**		**y=ja**	
Weight	*Feature*	*Weight*	*Feature*	*Weight*	*Feature*
5.147	<BIAS>	15.299	que	3.398	<BIAS>
1.406	from	14.163	de	2.159	avg_word_length
1.268	new	14.033	el	1.319	youtube
1.265	please	13.06	la	... 93 more positive ...	
1.225	by	11.66	en	... 900 more negative ...	
0.992	has	10.222	no	-1.227	tweet_length
0.987	gold	8.376	es	-1.253	la
0.945	when	7.557	te	-1.267	que
0.942	who	7.39	me	-1.297	ya
0.938	have	7.371	los	-1.333	di
... 613 more positive ...		... 191 more positive ...		-1.389	الله
... 380 more negative ...		... 802 more negative ...		-1.414	من

Table 4: Feature Importance Table for Linear Model

classes. Note that the bias of English is higher than the one of Japanese. Hence, if we would have an empty string (i.e., all features equal to 0), then the model would classify the tweet as English. Intuitively, this makes sense.

3 Language Identification with MLP (Part 2)

Given a tweet, the task at hand is to predict the tweet's language using a non-linear model (multi-layer perceptron (MLP)).

3.1 Preprocessing Steps

The same preprocessing steps as in section 2.1 were applied.

3.2 Additional Linguistic Features

The same additional linguistic features as in section 2.2 were used.

3.3 Grid Search to Find Optimal Hyper-Parameters

In order to find the optimal hyper-parameters of the MLP, the grid search space specified in Table 5 was explored. Please note that due to computational limitations, the maximum number of iterations taken for the solvers to converge was set to an extremely low value of 20.

	Hyper-Parameter	**Explored Values**
Classifier	Hidden layer sizes	*(100,), (10,10,10)*
	Activation function	*tanh, relu*
	Solver	*adam, lbfgs*
	Early stopping	*True, False*
	Max_iter	*20*
Vectorizer	Ngram range	*(1,1), (1,2)*
	Max features	*500, 1000*

Table 5: Grid Search Space of MLP

In order to minimize the risk of a model being superior due to different data splits, the MLP was trained and validated with the same training and development split as the linear classifier. The hyper-parameter combination of the best-performing MLP on the validation set is reported in Table 6.

	Hyper-Parameter	**Best Value**
Classifier	Hidden layer sizes	*(100,)*
	Activation function	*relu*
	Solver	*adam*
	Early stopping	*False*
	Max_iter	*20*
Vectorizer	Ngram range	*(1,1)*
	Max features	*1000*

Table 6: Hyper-Parameters of Best-Performing MLP

3.4 Confusion Matrix and Comparison with Linear Model

The overall accuracy of the best-performing model (see Table 6) on the unseen test set is **80.7%**. This is more than 20 percentage points higher than the linear model.

The Confusion Matrix of the top 10 classes with values normalized to true labels is displayed in Figure 3. Again, each row sums to 1.0 and the cell value c(i, j) indicates the percentage of class i being predicted as class j.

Compared to the confusion matrix of the linear model (see Figure 2), the confusion matrix of the MLP exhibits much higher values on the diagonal. Also, in contrast to what we saw for the linear model, the MLP does not just predict the English label for every class (values in the first column are much lower compared to the ones for the linear model).

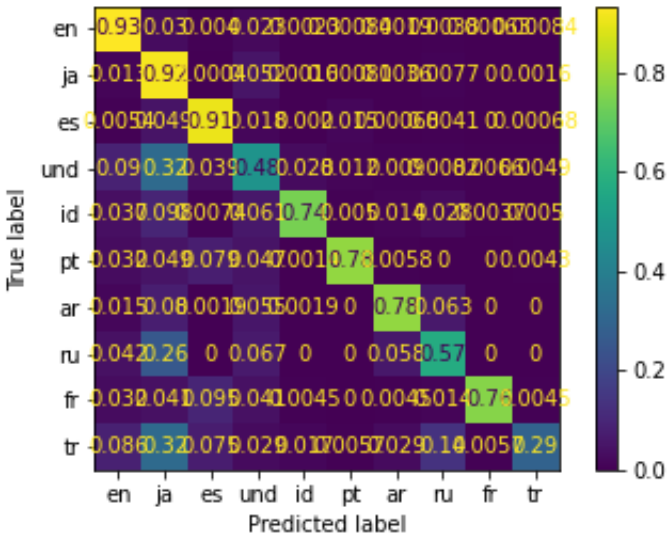

Figure 3: Confusion Matrix for Predictions on Test Set (Normalized to True Labels) of MLP

To summarize, the added model complexity of the MLP pays off as the test accuracy is considerably higher than in the linear case. Since language is highly complex, it makes much sense that the more complex model can capture hidden information better than the linear model.