IART - Assignment 2 - Emotions

Final Delivery June 6th

Group 76_3A
Afonso Monteiro - up201907284@up.pt
Miguel Lopes - up201704590@up.pt
Vasco Alves - up201808031@up.pt

Work Specification

The topic of our assignment is Natural Language Processing. The aim is to process textual data, employing diverse techniques to transform them into appropriate datasets that can then be addressed using supervised learning algorithms. In our particular case the objective is to classify statements according to the emotion present.

Statement	Emotion
How the fu*k! Who the heck! moved my fridge!... should I knock the landlord door. #angry #mad ##	anger
So my Indian Uber driver just called someone the N word. If I wasn't in a moving vehicle I'd have jumped out #disgusted	anger
I asked for my parcel to be delivered to a pick up store not my address #fuming #poorcustomerservice	anger
#ObamaLegacy - weekly #riots and #terror attacks, >400k dead #Syrians, #Jews fleeing #persecution in Europe, #Christian #genocide in ME.....	fear
bad news fam, life is still hard and awful #depression #anxiety #atleastIhaveBuffy	fear
fucking hell mate absolute nightmare ??	fear
What a #lively #lovely #shower ?? ..	joy
What a great training course, lots of photos, fun and laughter. Photo's will be up soon #Boostercourse #fun #laughter	joy
hahahaha you're ridiculous!!! But thank you a joyous evening xx	joy

Tools and algorithms

We will be using python for the programming language and will develop the assignment in a Jupyter Notebook and on source code files.

For the development of the work we will use NLTK, Stanza and scikit-learn. We are also planning on customizing the stopwords set to better serve our purposes.

We will be using the algorithms Naive Bayes, Decision Trees, SVM and XGB, among others.

Data Pre-Processing

On the source code files we removed the asterisk and then joined both parts, we removed non-alpha chars, converted every word to lowercase, applied stemming and removed stopwords. We customized the stopwords set for it to not include negations and added the word ‘amp’. We always used bag of words as the form of representation.

On the notebook, we performed the same data pre-processing. We can choose different forms of representation, such as bag of words, tf-idf, and also each one of these with ngram range. When using the ngram range, it is useful to have the negation words, hence why we do not remove them.

Implementation

On the source code files, we implemented the following algorithms:

- Multinomial Naïve Bayes
- Decision Trees
- Random Forest
- SVC
- Perceptron
- Logistic Regression
- XGBoost

It was explored with grid search whether it was worth tuning some of the hyper-parameters of these algorithms to obtain the best performance possible. The best accuracy, in percentage, was for each: 85.3%, 95.4%, 96.9%, 95.7%, 96.6%, 96.5% and 95.1%, respectively.

Implementation

On the notebook, we tried to not overload it on algorithms, so we implemented, with default parameters:

- Multinomial Naïve Bayes
- Decision Trees
- SVC
- Random Forest

We obtained the following results:

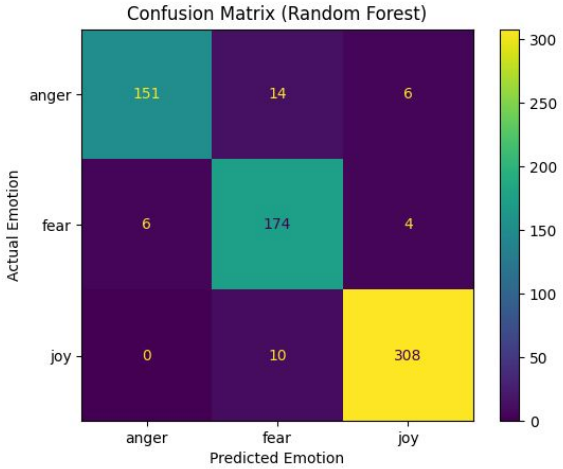
		Algorithm							
		Naive Bayes		Decision Trees		SVC		Random Forest	
		Time*	Accuracy**	Time	Accuracy	Time	Accuracy	Time	Accuracy
Representation	Bag of Words	1s	93.08%	17s	95.39%	47s	95.82%	10s	96.69%
	TF-IDF	1s	94.38%	1s	88.76%	1s	96.69%	1s	96.97%
	Bag of Words with ngram_range=(1,2)	3s	95.24%	1m30s	95.39%	3m	96.11%	30s	97.41%
	TF-IDF with ngram_range=(1,2)	1s	95.39%	1s	93.80%	2s	96.54%	3s	97.12%

*Time was measured on training

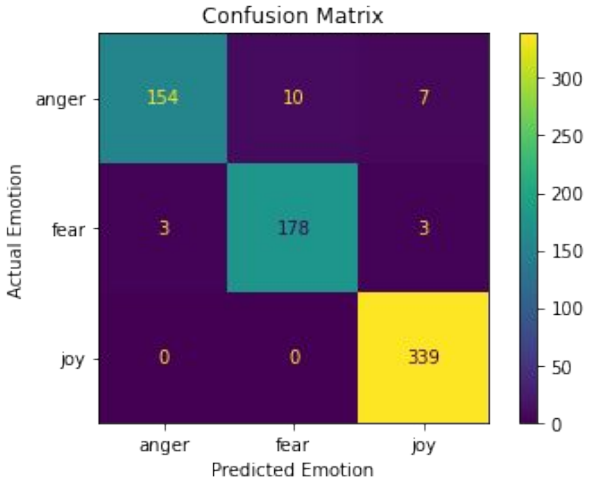
**Accuracy was measured on the test dataset

Evaluation and Comparison

As we can see by the results, squeezing the best parameters usually gives similar results as when the pre-process is done differently and without the tuning. In the case of Random Forest, we can say that using grid search to explore the best parameters might be a viable option, even tho we only analyzed its results for the bag of words form of representation.
In the matrices below, the dataset was pre-processed in the same way, as described in the previous sections:



Random Forest Confusion Matrix with Grid Search Exploring



Random Forest Confusion Matrix with Default Parameters

Evaluation and Comparison

The best result we obtained in the notebook was an accuracy of 96.97%, using Random Forest with TF-IDF representation.
For the algorithms in the source code files, the best one was also Random Forest with an accuracy of 96.9%, but this time using Bag of Words and some parameter tuning with grid search.

The worst result we got was an accuracy of 85.3% with Naive Bayes, using Bag of Words Representation and grid search. However, we believe this result could have been better if we improved the grid search exploration.

Conclusions

In conclusion, assuming that the poor classification of the dataset on the source website is to blame, we can see that the Random Forest classifier gives results similar or better than those by the X-grading Boost one, even though that the latter was the most promising one.

We also discovered that different forms of representation and different parameters can affect the performance of the classifier, which is a thing to take into account when using these types of algorithms.

Even so, we were able to study the steps needed to deal with problems of natural language processing, even coming up with a way to obtain the best parameters possible for each classifier.

This work puts us one step in the world of machine learning, and it is another on the way of becoming full-fledged informatics engineers.

Bibliography

https://www.kaggle.com/datasets/anonymous1972/emotions?resource=download

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=randomforest#sklearn.ensemble.RandomForestClassifier

https://www.datacamp.com/tutorial/xgboost-in-python

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html?highlight=perceptron#sklearn.linear_model.Perceptron

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html?highlight=svc#sklearn.svm.SVC

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html?highlight=nb#sklearn.naive_bayes.MultinomialNB

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?highlight=decision%20tree#sklearn.tree.DecisionTreeClassifier

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highlight=logistic%20regression#sklearn.linear_model.LogisticRegression

https://blog.dataiku.com/narrowing-the-search-which-hyperparameters-really-matter