Natural Language Processing - INF210
Exercise 17 of part 2

Gketsopoulos Petros | $F3321804$
Gikas Athanasios | $F3321808$
Euagelou Iordanis | $F3321801$
Kougia Vasiliki | $F3321805$

March 25, 2019

1 Dataset

We used the Semeval 2016 dataset created for sentiment analysis on tweets.[1] Tweets are classified into three classes: positive, negative and neutral. The initial dataset consists of a training set (6,000 tweets), a dev set (2,000 tweets) and a devtest set (2,000 tweets), which we retrieved through the Twitter API using the provided code.[2] Some of the tweets could not be retrieved and had a "Not Available" text. These tweets were 2,644 in total and were removed from the dataset.

We additionally observed that some of the tweeted texts existed more than once in the same set, while in some cases the same tweets appeared with different labels. In the case of duplicates consisting of same labels we just removed them and kept only one instance of the tweet at random. To deal with the inconsistency of labels of the same tweets we completely removed these instances from the sets. During this process, a total of 40 tweets were removed from our base. We also observed the same issues occurring across the given sets e.g. the same tweet appeared both in the training and the devtest set. We handled this issue the same as before by keeping only one of the duplicates and discarding duplicates with different labels, removing 17 tweets in total.

The remaining tweets in all three sets were 7,299, from which 3643 were positive, 2516 were negative and 1140 were neutral. We decided to merge the three provided sets into one, which we shuffled and randomly split to train and test. We used 80% (5,839 tweets) of the merged set for train and 20% (1,460 tweets) for test. For the extraction of feature vectors and the classification process, we used algorithms provided by the *scikit-learn* library. Before the classification step, we preprocessed each text individually and also tuned the hyper-parameters using cross-validation on the training set. The algorithms that we decided to apply and the evaluation of our experiments are further described in the following sections.

[1] `http://alt.qcri.org/semeval2016/task4/index.php?id=data-and-tools`
[2] `https://github.com/aritter/twitter_download`

2 Preprocessing and Feature Extraction

Several text filters were applied to the initial dataset either with handcrafted methods or with *nltk* toolset. Our filters consist of lowercase, punctuation and stopwords transformers. Additionally, we subtracted several symbols such as *hashtags*, *@user* and URL links. We further preprocessed basic *emoticon* symbols, as they can be useful for sentiment analysis. In this manner, we replace each one of them with an actual keyword from a list of pairs with the key being an emoticon and value a text keyword depending to the symbols meaning (e.g. ":)" is paired with the value "HAPPY_EMOTICON").

In this work, the transformation between the text domain and the feature space is handled by the TF-IDF vectorizer algorithm provided from the *scikit-learn* toolkit. The input parameters are the following :

ngram range	**max features**	**norm**	**smooth idf**	**sublinear tf**
(1,3)	2281	l2	true	true

We tuned the vectorizer's feature selection prior to the classifier tuning process. This way, we aim to benefit from the reduced search space during tuning, establish a constant parameter for the feature dimension for every classifier but also prevent them to select the highest possible dimensional feature and overfit in the given trainset. In order to select the appropriate dimensionality, which minimizes redundancy and maximizes the information gain of our features, we attempt to approximate the true dimension, by measuring the geometric mean from a series of projected domain dimensions that satisfy the $87 - 93\%$ of the initial domains variance after an *SVD* decomposition. More precisely, we incrementally apply the vectorization process with an initial feature size of 1000 to a maximum of 13000 and in each step we deploy a binary search logic in order to find a projected space which retains the inputs variance in a ratio of $87 - 93\%$. Finally, we compute the tendency of our series and use it as an input parameter for our feature dimension. We briefly demonstrate our experimental results in Fig.1.

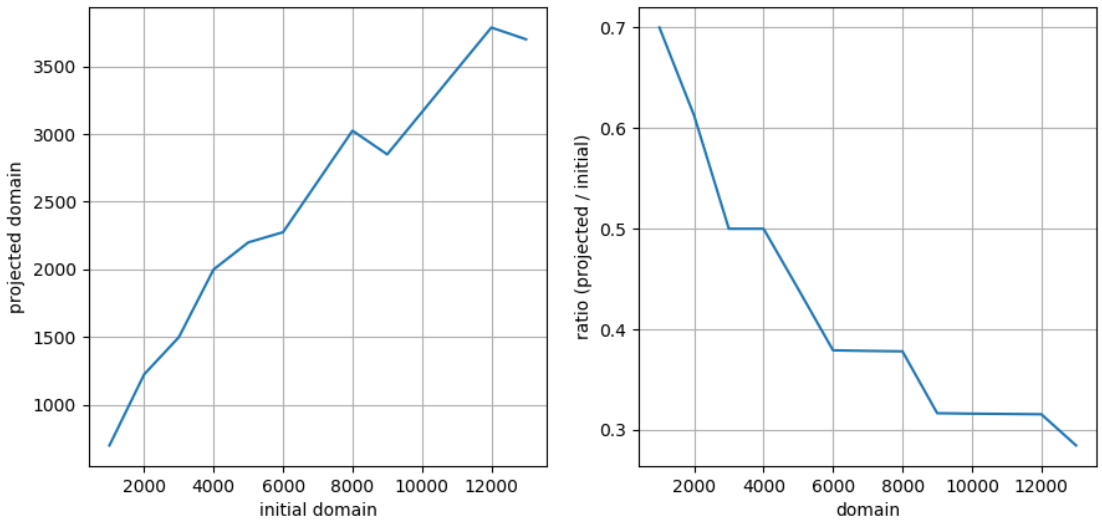


Figure 1: Left, the dimension of projected space per input. Right, the ratio between the projected domain and the initial one.

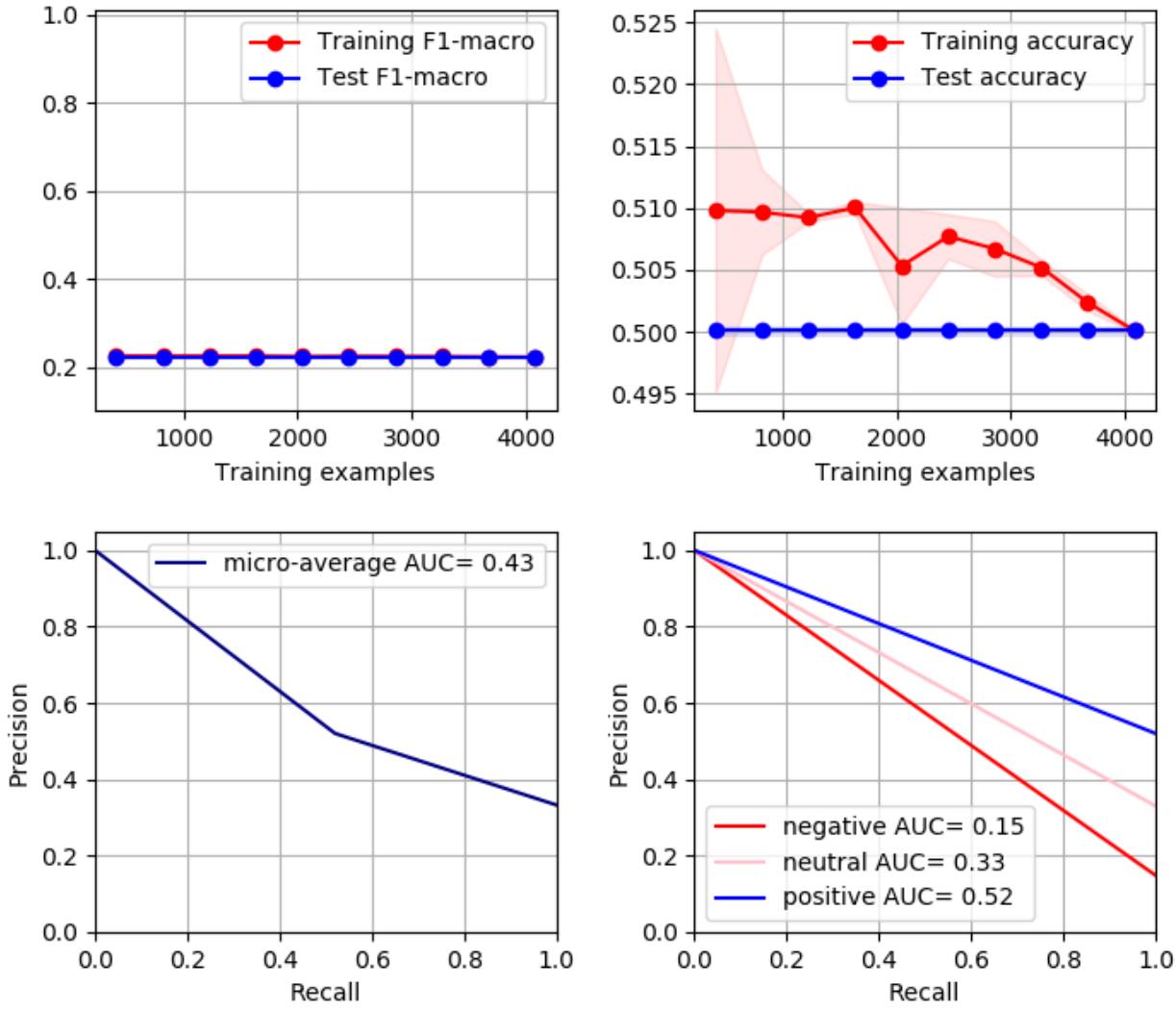
3 Evaluation

Prior to the evaluation of each classifier, we employ a grid search approach to tune their hyper-parameters on the training subset of our dataset and then train them in order to infer on the unseen test subset. In the following subsections, we briefly describe the hyper-parameters chosen for each classifier (marked as underlined) and visualize our results regarding the F1-score, accuracy and precision recall curves.

3.1 Dummy classifier

Estimators parameters :

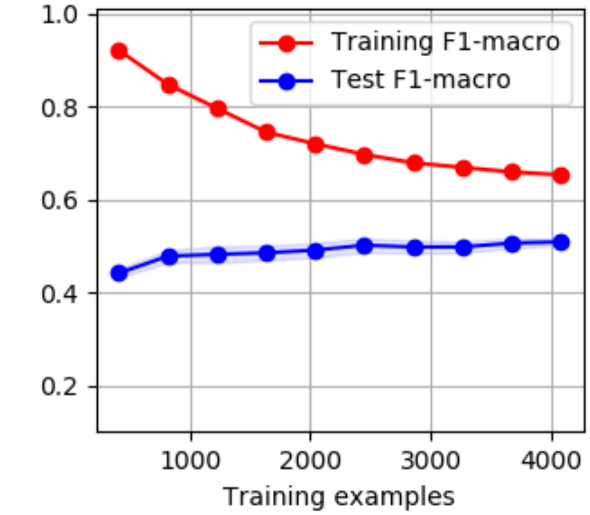
- **Stategy** : most frequent, stratified

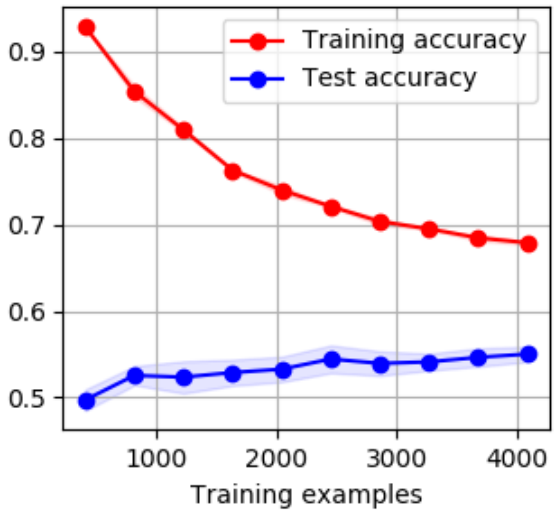


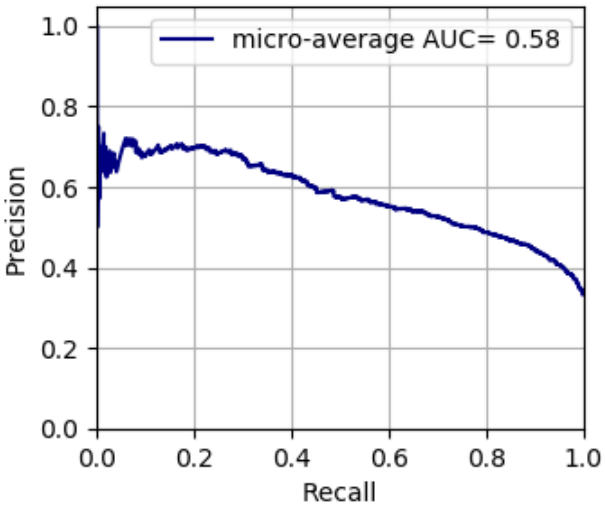
3.2 Logistic regression

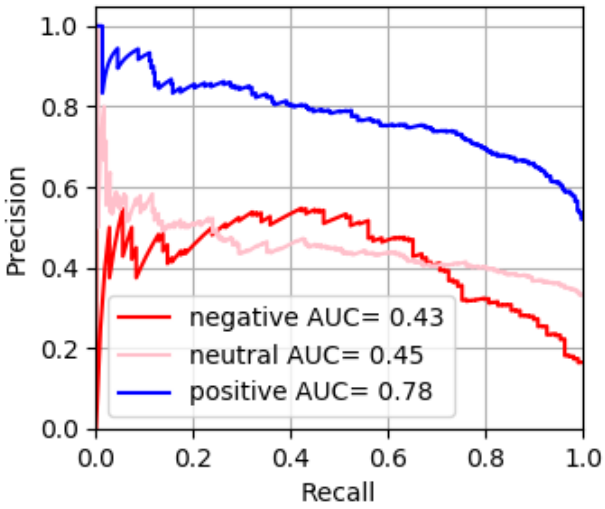
Estimators parameters :

- **class weight** : balanced
- **multi class** : multinomial
- **C** : 0.01, 0.1, 1.0, 1.0
- **solver** : newton-cg, lbfgs, sag
- **penalty** : L2





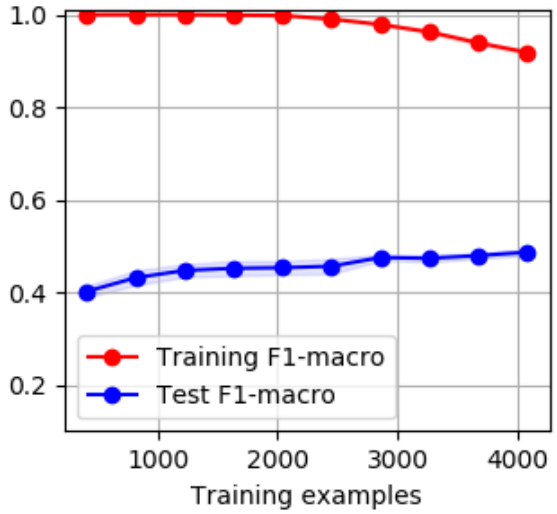


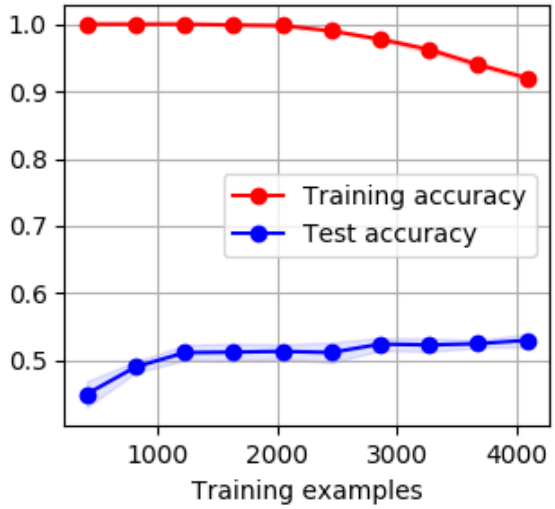


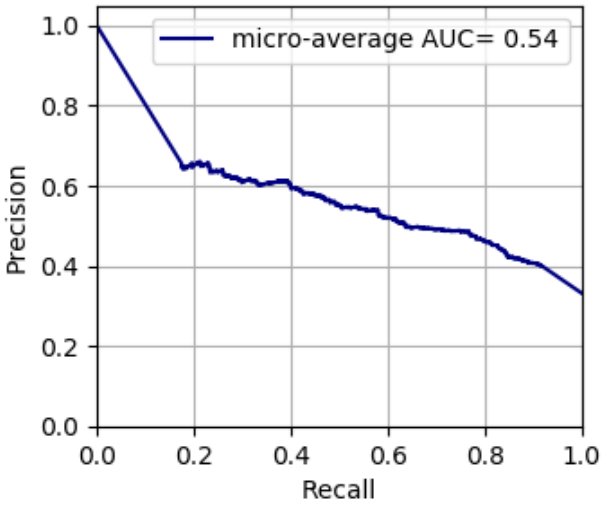
3.3 Stochastic gradient descent

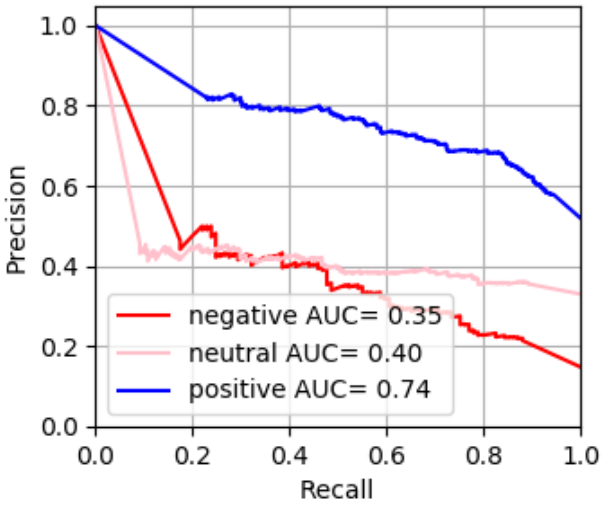
Estimators parameters :

- **class weight** : balanced
- **C** : $0.01, \underline{0.1}, 1.0, 1.0$
- **loss** : <u>modified huber</u>, perceptron
- **penalty** : <u>L2</u>, elasticnet
- **L1 ratio** : $0.0, 0.2, 0.4, 0.6, 0.8, 1.0$





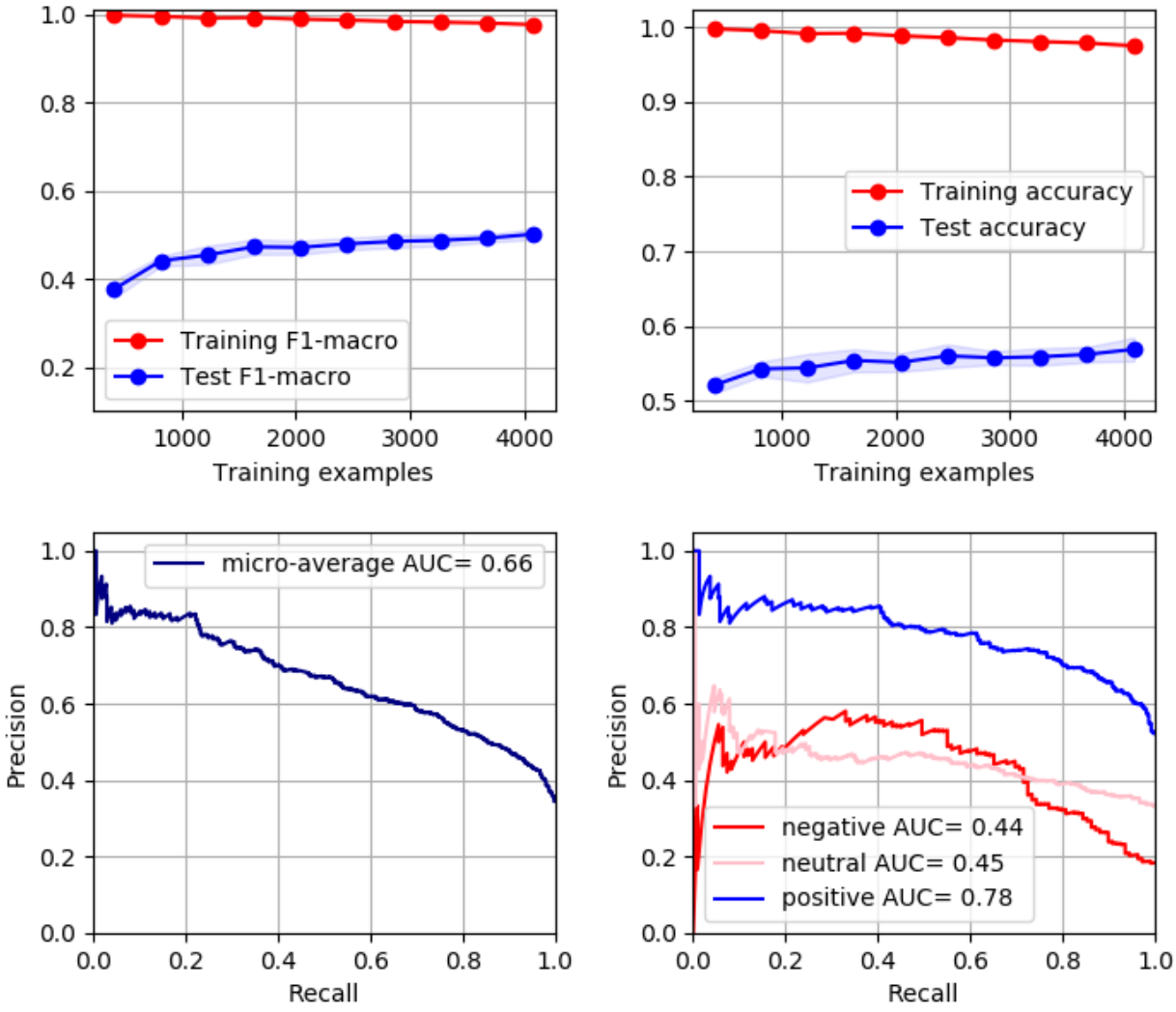




3.4 Support vector machine

Estimators parameters :

- **class weight** : balanced
- **gamma** : scale
- **kernel** : linear, rbf, poly, sigmoid
- **C** : $0.01, 0.1, \underline{1.0}, 10$
- **coef0** : $\underline{3}, 4$



3.5 Conclusions

The *Dummy* classifier had the worst performance as expected. Among the other classifiers we experimented with, we observed that they had similar performance, achieving an accuracy between 50%-60%, an F1-score between 0.4-0.6, as well as having same AUCs, on the

test subset. We believe that our classifiers perform poorly due to the nature of our feature selection algorithm. The lack of feature descriptiveness combined with the available corpus size, constraints the available representation space and the exploration process of our algorithms does not generalize to our task. The latter behavior can be observed when a gradient descent heuristic is applied in order to fit the search space (e.g. *SVM* and *SGD*). The latter, results in a high performance on the training subject but still performs poorly in the unseen subset.