UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: *Panagiotis Zazos*
sdi: 7115112300009

Course: *Artificial Intelligence II (M138, M226, M262, M325)*
Semester: *Fall Semester 2023*

Contents

1 Abstract **2**

2 Data processing and analysis **2**
2.1 Pre-processing . . . 2
2.2 Analysis . . . 3
2.3 Data partitioning for train, test and validation . . . 5
2.4 Vectorization . . . 5

3 Algorithms and Experiments **5**
3.1 Experiments/Steps . . . 5
3.2 Hyper-parameter tuning . . . 8
3.3 Evaluation . . . 9
3.3.1 Learning Curve . . . 9
3.3.2 Confusion matrix . . . 10

4 Results and Overall Analysis **13**
4.1 Results Analysis . . . 13
4.1.1 Best trial . . . 13
4.1.2 Validation dataset predictions . . . 13
4.2 Comparison with the first project . . . 13
4.3 Comparison with the second project . . . 14
4.4 Comparison with the third project . . . 14

5 Bibliography **14**

1. Abstract

The task at hand is to create a sentiment analysis model for Twitter data about Greek Elections in the Greek language. Sentiment analysis involves classifying text into one of three categories: Negative, Neutral, Positive. The objective here is to develop a model which accurately predicts the sentiment of given tweets. To achieve this, a combination of ML and NLP techniques were used.

2. Data processing and analysis

2.1. Pre-processing

The data underwent a comprehensive cleaning and pre-processing to ensure the model's training on high-quality and meaningful text. Through the implementation of the following data cleaning techniques, the text has reduced the irrelevant information, noise and inconsistencies. The following data cleaning techniques were applied:

- **Links/Mentions**: URLs, hyperlinks and mentions were eliminated from the text. These might not contribute to the sentiment analysis.
- **Hashtag character**: Hashtags might contain important information about the sentiment of the tweet, therefore the elimination of the character # was necessary.
- **Special characters**: Special characters like left/right pointing double angle quotation mark and other quotation marks are most common in tweets, so their removal is necessary.
- **Lowercasing**: All text was converted to lowercase to guarantee case insensitivity, ensuring that words were recognizable regardless of their capitalization. This step eliminates the duplication of features as well.
- **Lemmatization**: Lemmatization was performed using the spaCy library "el-core-news-sm". This process reduced words to their base form, ensuring that variations of a word were treated as a single entity. **Figure 1** is an example of spaCy's lemmatization along with the findings of more stopwords

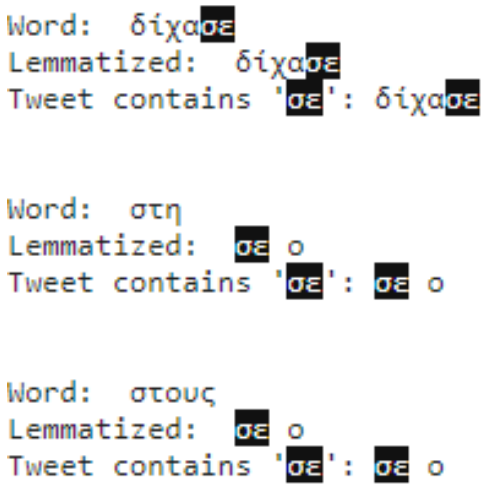

Figure 1: Greek Lemmatization example with stopwords

- **Greek Stopwords**: Since nltk's stopwords didn't match the needs of the data, a custom Greek stopwords list was constructed to eliminate common words that do not provide valuable information for sentiment analysis. The most important inclusion was common words with and without accents, as both variations can appear on the tweets.
  Note: Additionally, given the prevalence of mixed Greek and English text in Greek tweets, it may be beneficial to explore the inclusion of English stopwords in the pre-processing pipeline to handle such linguistic diversity.

2.2. Analysis

Wordcloud: The word cloud provides a visual represantion of the most frequently occuring terms in the training dataset. Larger-sized words represent higher frequency, making it immediately apparent which terms are most prominent. Of course this is heavenly dependent of the pre-process that's been made beforehand. **Figure 2** shows the wordcloud for the training dataset.

Figure 2: Wordcloud

Token frequencies: The token frequency list provides precise information on the occurrence of each term, which is essential for understanding the weight of each term. In **Figure 3**, the first 20 most prominent tokens were displayed.

```
Top 20 most frequent words:
νδ: 6564 occurences
τσίπρα: 5144 occurences
μητσοτάκη: 3824 occurences
μητσοτάκης: 3716 occurences
συριζα: 3619 occurences
κκε: 2903 occurences
τσίπρας: 2580 occurences
κάνω: 2242 occurences
τσιπρα: 2060 occurences
νέος: 2044 occurences
κιναλ: 1898 occurences
εκλογή: 1801 occurences
μην: 1750 occurences
εκλογες_2019: 1742 occurences
συριζας: 1732 occurences
λέγω: 1475 occurences
σκαι: 1456 occurences
κυριάκος: 1418 occurences
πασοκς: 1334 occurences
θέλω: 1318 occurences
```

Figure 3: Token frequencies

Note: It is obvious that the special underscore character "_" should have been erased, although that could heavily cripple the semantics of tweets, as these tend to contain very useful information in the form of hashtags, i.e. "κυβέρνηση_πολιτικοσχολιο.

However, one can just replace the underscore character with a white space and split the hashtag.

2.3. Data partitioning for train, test and validation

train_test_split: The choice of an 80-20 has been made for the training dataset, meaning that 80% of the data was used for training the model and the remaining 20% was used for testing and evaluating the model's performance. This split is common practice in machine learning for evaluating model performance. It offers a sizeable amount of data for training while still having a decent separate dataset to assess the model's predictions on unseen data.

2.4. Vectorization

TF-IDF: Term Frequency-Inverse Document Frequency is a technique used to convert text documents into numerical vectors. This helps to understand the importance of words in the tweets. TF-IDF is consisted of:

- **TF:** Term Frequency, which measures how often a word appears in a document, giving insight about the relevance within it.
- **IDF:** Inverse Document Frequency, which indicates how unique or rare a word is across the collection of documents.

Combining TF and IDF we get TF-IDF score for each word in each tweet. Words with higher score are considered more significant. This process allowed the representation of text data as numerical features, suitable for machine learning.

3. Algorithms and Experiments

3.1. Experiments/Steps

The ultimate goal is to enhance the accuracy of the model through a series of experiments. The following are the experiments/steps that were conducted to determine the best way to approach the assignment:

Experiment 1 - Hyperparameter Tuning: This first experiment was focused on optimizing the hyperparameters of Logistic Regression. Instead of manually tweaking each value of **C** (Inverse of regularization strength; smaller values indicate stronger regularization) and **solver** (Algorithm to optimize the problem), a grid was used to test every combination, through exhaustive search using sklearn's GridSearchCV.
Results: With just a regular train/test split, GridSearchCV resulted in the lbfgs to be the better solver and C= 0.1, as shown in **Figure 4**.

```
Best Hyperparameters: {'C': 0.1, 'solver': 'lbfgs'}
Accuracy: 0.3808353808353808
Classification Report:
              precision    recall  f1-score   support

           0       0.37      0.44      0.40      2447
           1       0.38      0.33      0.35      2450
           2       0.39      0.37      0.38      2429

    accuracy                           0.38      7326
   macro avg       0.38      0.38      0.38      7326
weighted avg       0.38      0.38      0.38      7326
```

Figure 4: GridSearchCV results

Experiment 2 - Dimension Reduction: This technique offers a way to reduce the number of features in a dataset while preserving its important information. The idea was to use dimension reduction to reduce overfitting by reducing the number of features, as models can become overly complex and prone to overfitting, where they perform well on the training data but poorly on unseen data. Moreover, dimension reduction may reduce noise and lighten the issue of sparse data. The usage of PCA (Principal Component Analysis) was not possible, as it's designed for dense data. So, TruncatedSVD, which is more suitable for sparse data, such as text data represented as TF-IDF matrices, was used in the training data. To achieve better results, experiments were made by tweaking the components that are created after reducing the dimensionality of the training dataset. The following are the results of 10, 50 and 100 components using TruncatedSVD:

```
Accuracy: 0.3579033579033579
Classification Report:
              precision    recall  f1-score   support

           0       0.36      0.49      0.42      2447
           1       0.34      0.16      0.22      2450
           2       0.36      0.43      0.39      2429

    accuracy                           0.36      7326
   macro avg       0.35      0.36      0.34      7326
weighted avg       0.35      0.36      0.34      7326
```

Figure 5: TruncatedSVD components=10

```
Accuracy: 0.3682773682773683
Classification Report:
              precision    recall  f1-score   support

           0       0.36      0.48      0.41      2447
           1       0.37      0.28      0.32      2450
           2       0.37      0.34      0.35      2429

    accuracy                           0.37      7326
   macro avg       0.37      0.37      0.36      7326
weighted avg       0.37      0.37      0.36      7326
```

Figure 6: TruncatedSVD components=50

```
Accuracy: 0.3714168714168714
Classification Report:
              precision    recall  f1-score   support

           0       0.37      0.48      0.41      2447
           1       0.37      0.28      0.32      2450
           2       0.38      0.36      0.37      2429

    accuracy                           0.37      7326
   macro avg       0.37      0.37      0.37      7326
weighted avg       0.37      0.37      0.37      7326
```

Figure 7: TruncatedSVD components=100

Experiment 3 - KFold Cross-Validation: This technique ensures that the model is tested on different subsets of data, making the evaluation more robust and less dependent on the particular random split of data, as it partitions the data into K subsets (folds) and then trains and tests the model K times, each time using a different fold as the test set and the remaining K-1 folds as the training set. Moreover, it helps detecting and mitigating overfitting, as it allows to assess how well the model generalizes by evaluating its performance on multiple tests sets. Overall, the "rotation" of the test sets ensures that every data point is used for both training and testing, which is essential. **Figure 8** shows the accuracy of the model when using a KFold, where k=10

```
Mean Classification Report:
              precision    recall  f1-score   support

           0       0.39      0.44      0.41      1238
           1       0.40      0.35      0.37      1217
           2       0.39      0.38      0.39      1208

    accuracy                           0.39      3663
   macro avg       0.39      0.39      0.39      3663
weighted avg       0.39      0.39      0.39      3663

Cross-Validation Accuracy: 0.39 ± 0.00
```

Figure 8: KFold on training dataset (k=10)

Experiment 4 - Dimension Reduction with KFold Cross-Validation: The model underwent more experiments when using a cross-validation technique, where a dimension reduction was applied at every fold. Same as before, TruncatedSVD was applied with various number of components (10, 50, 100).

```
Mean Classification Report:
              precision    recall  f1-score   support

           0       0.37      0.50      0.43      1238
           1       0.39      0.18      0.25      1217
           2       0.35      0.41      0.37      1208

    accuracy                           0.36      3663
   macro avg       0.37      0.36      0.35      3663
weighted avg       0.37      0.36      0.35      3663

Cross-Validation Accuracy: 0.36 ± 0.00
```

Figure 9: TruncatedSVD KFold components=10

```
Mean Classification Report:
              precision    recall  f1-score   support

           0       0.38      0.51      0.44      1238
           1       0.40      0.28      0.33      1217
           2       0.39      0.37      0.38      1208

    accuracy                           0.39      3663
   macro avg       0.39      0.39      0.38      3663
weighted avg       0.39      0.39      0.38      3663

Cross-Validation Accuracy: 0.37 ± 0.01
```

Figure 10: TruncatedSVD KFold components=50

```
Mean Classification Report:
              precision    recall  f1-score   support

           0       0.38      0.49      0.43      1238
           1       0.40      0.29      0.34      1217
           2       0.38      0.37      0.37      1208

    accuracy                           0.39      3663
   macro avg       0.39      0.38      0.38      3663
weighted avg       0.39      0.39      0.38      3663

Cross-Validation Accuracy: 0.38 ± 0.01
```

Figure 11: TruncatedSVD KFold components=100

3.2. Hyper-parameter tuning

GridSearchCV: As stated in the section above (Experiment 1), the usage of this tool was crucial to identify the best combination of the regularization strength and the algorithm to be used.

3.3. Evaluation

While studying the classification reports one can notice something interesting, which is also reasonable: the predictions for the Neutral(1) tweets are never good enough comparatively to Positive(2) or Negative(0). In every technique that was used, the **f1-score** of the Neutral tweets is the lowest to the others. This obviously states that the imbalance between **precision** and **recall** is noticeable. Moreover, the model's ability to identify all positive instances of Negative(0) tweets is reasonably good, close to 50% (recall).
While it would be later discussed, the need fore more powerful and unique tools to better classify the tweets is mandatory, like adding weights to words and most importantly, word embeddings.

3.3.1. Learning Curve.

The Learning Curve at **Figure 12** shows the performance of the model over time as it learns from a training dataset, using a train_test_split technique of 0.2 and without cross-validation and/or dimension reduction.
Red Line: The training starts high and decreases as the number of training examples increases. Due to overfitting, the model performed well on a smaller scale. As more data were introduced, the model must generalize from a larger set of examples, which can reduce the training score.
Green Line: The validation score begins low and increases slightly as more data are introduced. Any increase of this line mean that the model is learning and generalizing better as it is exposed to more tweets.
The gap between the lines indicates the overfitting, as the model performs better on the training set than on the unseen validation data.
Moreover, the shaded area around the red line suggests the variability in the training score across different training sets (epochs). This larger area around the beginning of the line indicates more variability in performance with fewer training examples, which is expected as the model might not be stable. As the number of examples increases, the shaded area becomes narrower, which means that the model's becoming more consistent.
In conclusion, since the validation score is increasing, more data could help improve the model's performance, as well as more work on feature engineering, improving the input features for better predictions.

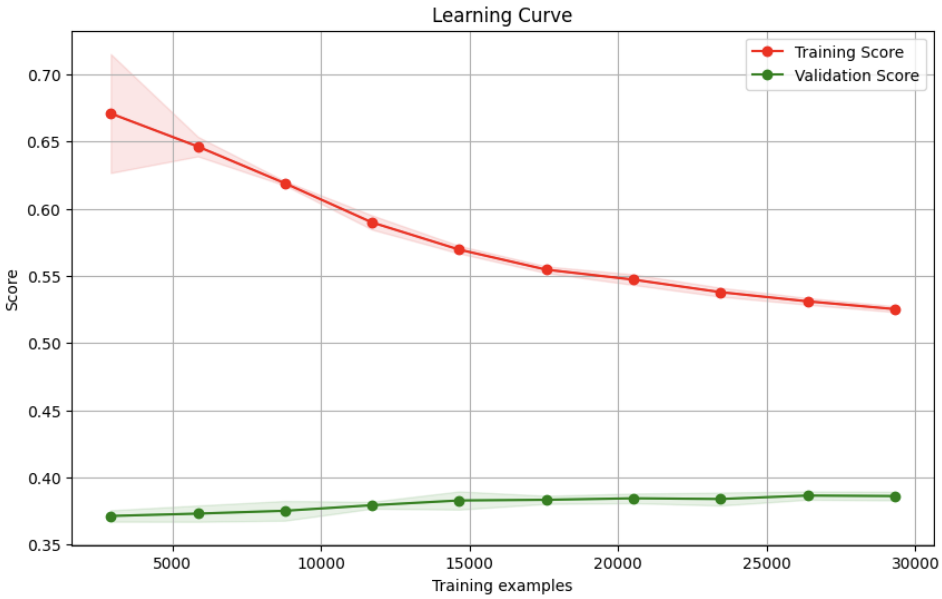

Figure 12: Learning Curve

3.3.2. Confusion matrix. A confusion matrix is a table used to describe the performance of a classification model on a set of data for which the true values are known. The matrix for the training dataset was implemented using a train_test_split of 0.2, without using cross-validation and/or dimension reduction.
Let's preface with the encoding of the categories:

- **0: Negative**
- **1: Neutral**
- **2: Positive**

Describing the confusion matrix of the training dataset, Figure 13

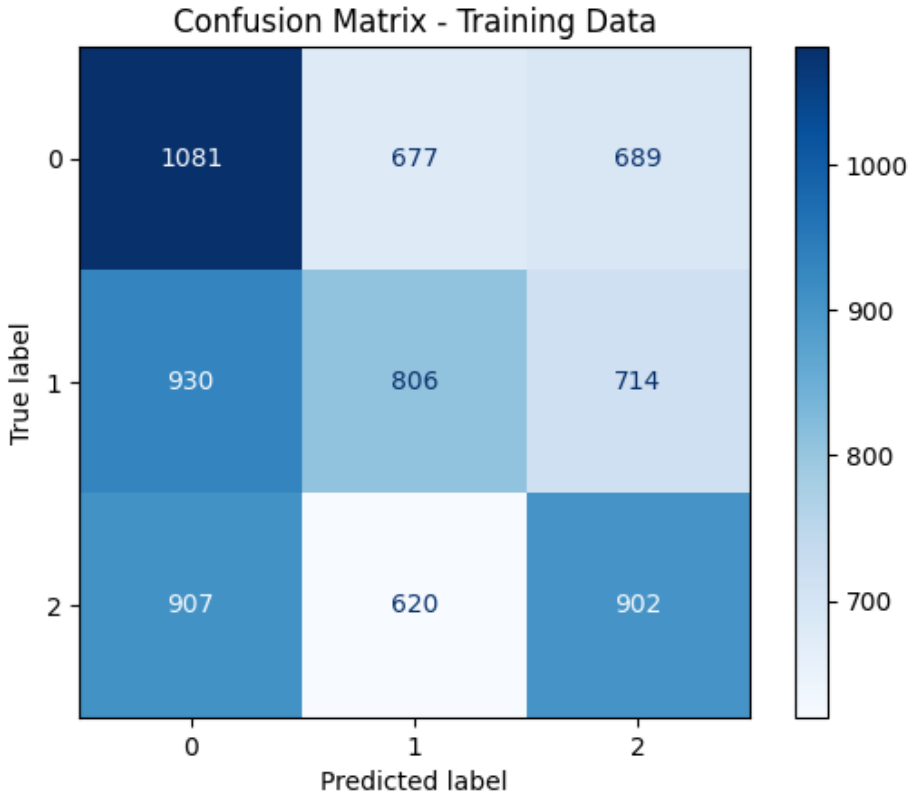

Figure 13: Confusion Matrix training

Diagonal Values (True Positives):

- For Negatives tweets, the model correctly predicted 1081 instances.
- For Neutral tweets, the model correctly predicted 806 instances.
- For Positive tweets, the model correctly predicted 902 instances.

Off-diagonal Values (Misclassifications):

- Negatives were incorrectly predicted as Neutrals 677 times, and as Positives 689 times.
- Neutrals were incorrectly predicted as Negatives 930 times, and as Positives 714 times.
- Positives were incorrectly predicted as Negatives 907 times, and as Neutrals 620 times.

Insight: The model, as stated before when analyzing the mean classification reports, struggles the most to distinguish Neutral tweets from Negative and Positive tweets. The misclassification of Positives and Negatives is also significant, not to be overlooked.

Describing the confusion matrix of the validation dataset, Figure 14

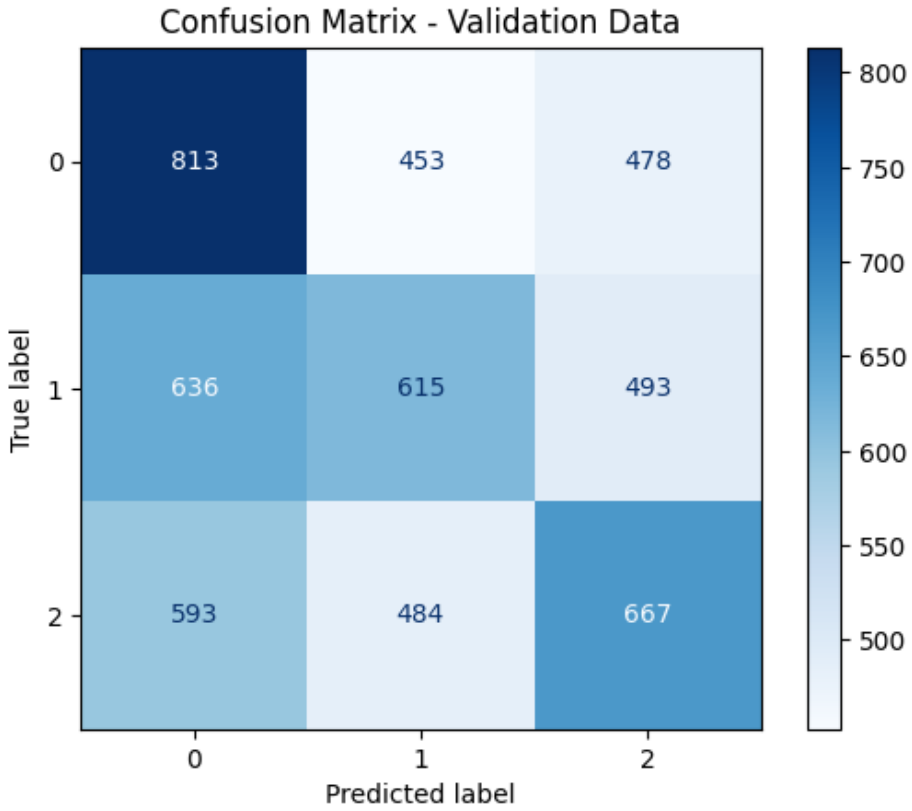

Figure 14: Confusion Matrix validation

Diagonal Values (True Positives):

- For Negatives tweets, the model correctly predicted 813 instances.
- For Neutral tweets, the model correctly predicted 615 instances.
- For Positive tweets, the model correctly predicted 667 instances.

Off-diagonal Values (Misclassifications):

- Negatives were incorrectly predicted as Neutrals 453 times, and as Positives 478 times.
- Neutrals were incorrectly predicted as Negatives 636 times, and as Positives 493 times.
- Positives were incorrectly predicted as Negatives 593 times, and as Neutrals 484 times.

Insight: The model's performance drop from training to validation indicates that it may be overfitting the training data. As on the training, so does in the validation data, the model's performance is not consistent across the classes. Further evaluation to determine why it is struggling with certain classes, especially on the Neutral sentiment, is mandatory.

4. Results and Overall Analysis

4.1. Results Analysis

The accuracy achieved is around 0.39 and can be considered mediocre. However, branding the accuracy good or bad depends on the specific problem, dataset and techniques used to approach them. A necessary tool that must be manipulated is the use of the Word Embeddings. These include techniques such Word2Vec, GloVe, fastText, which capture semantic relationships between words, thus enhancing the model's understanding of the tweet.
Pre-processing aside, different machine learning models beyond logistic regression can be tested, such as Random Forest, SVM (Support Vector Machine) or neural networks.

4.1.1. Best trial.

```
Mean Classification Report:
              precision    recall  f1-score   support

           0       0.39      0.44      0.41      1238
           1       0.40      0.35      0.37      1217
           2       0.39      0.38      0.39      1208

    accuracy                           0.39      3663
   macro avg       0.39      0.39      0.39      3663
weighted avg       0.39      0.39      0.39      3663

Cross-Validation Accuracy: 0.39 ± 0.00
```

Figure 15: KFold training dataset k10

4.1.2. Validation dataset predictions.

```
Validation Accuracy: 0.4004204892966361
Classification Report:
              precision    recall  f1-score   support

           0       0.40      0.47      0.43      1744
           1       0.40      0.35      0.37      1744
           2       0.41      0.38      0.39      1744

    accuracy                           0.40      5232
   macro avg       0.40      0.40      0.40      5232
weighted avg       0.40      0.40      0.40      5232
```

Figure 16: Validation dataset predictions

4.2. Comparison with the first project

<Use only for projects 2,3,4>
<Comment the results. Why the results are better/worse/the same?>

4.3. Comparison with the second project

<Use only for projects 3,4>
<Comment the results. Why the results are better/worse/the same?>

4.4. Comparison with the third project

<Use only for project 4>
<Comment the results. Why the results are better/worse/the same?>

5. Bibliography

References

[1] Donald E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.

<You should link and cite whatever you used or "got inspired" from the web. Use the / cite command and add the paper/website/tutorial in refs.bib>
<Example of citing a source is like this:> [1] <More about bibtex>