

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ

Επεξεργασία Φωνής και Φυσικής Γλώσσας



3ο Εργαστήριο:

Αναφορά

Μάρκος Γιαννόπουλος 03118103

Νικόλαος Καραφύλλης 03119890

Ιούλιος 2023

Στα ερωτήματα 1-5 χρησιμοποιούμε το dataset MR.

Ερώτημα 1

1.1

Τρέχουμε την main.py με είσοδο από το terminal “MaxPoolingDNN”.

Έχουμε υλοποιήσει το MaxPoolingDNN στο models.py.

Υπολογίζουμε την αναπαράσταση κάθε πρότασης ως την συνένωση του μέσου όρου (mean pooling) και του μεγίστου (max pooling) ανά διάσταση των word embedding κάθε πρότασης.

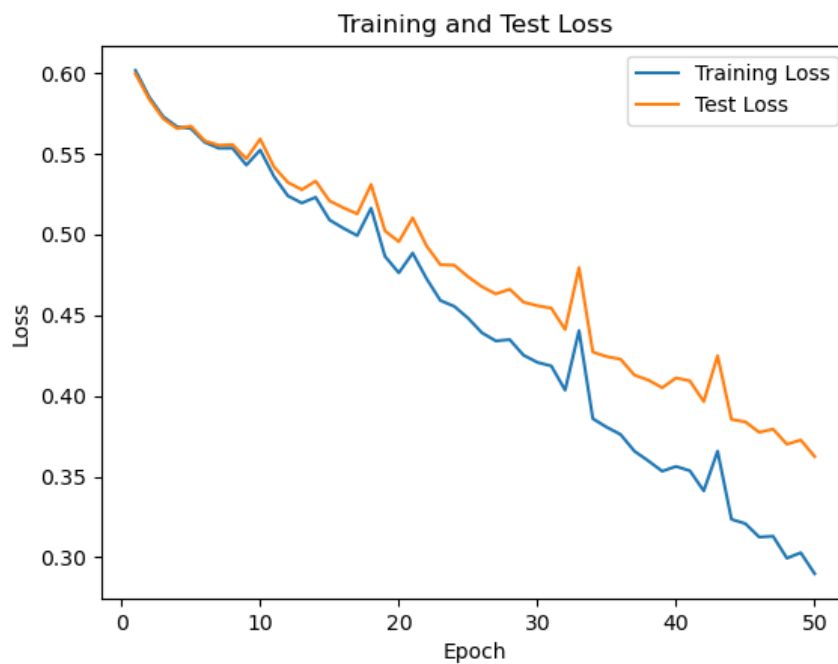
$$u = [mean(E)||max(E)]$$

1.2

Κάθε διάσταση των embeddings των λέξεων αναπαριστά θεωρητικά κάποια έννοια. Επομένως όταν κάνουμε mean pooling, κρατάμε πληροφορία για τον βαθμό που εκφράζεται κάθε έννοια κατά μέσο όρο σε κάθε λέξη της πρότασης.

Όμως κάποιες έννοιες μπορεί να εκφράζονται πολύ έντονα σε κάποιες λέξεις (η αντίστοιχη διάσταση στο embedding της λέξης έχει μεγάλη τιμή), όμως να εκφράζεται λίγο στις υπόλοιπες. Έτσι με το max pooling μπορούμε να κρατήσουμε πληροφορία για κάποιες σημαντικές έννοιες οι οποίες όμως μπορεί να μην εκφράζονται σε πολλές λέξεις της πρότασης.

Ο συνδυασμός mean pooling και max pooling αναμένουμε να έχει καλύτερα αποτελέσματα και πράγματι το test accuracy αυξήθηκε σε 0.8555 μετά από 50 epochs.



Train accuracy: 0.894625

Test accuracy: 0.8555

Train F1 score: 0.8946152370896985

Test F1 score: 0.8554306631003296

Train Recall: 0.8950273324835774

Test Recall: 0.8561833163213636

Ερώτημα 2

2.1

Χρησιμοποιούμε την συνάρτηση `torch_train_val_split` για να δημιουργήσουμε validation set από το training set.

Χρησιμοποιώντας την κλάση `EarlyStopper`, διακόπτουμε την εκπαίδευση όταν το validation loss αυξάνεται συνεχώς για 5 εποχές.

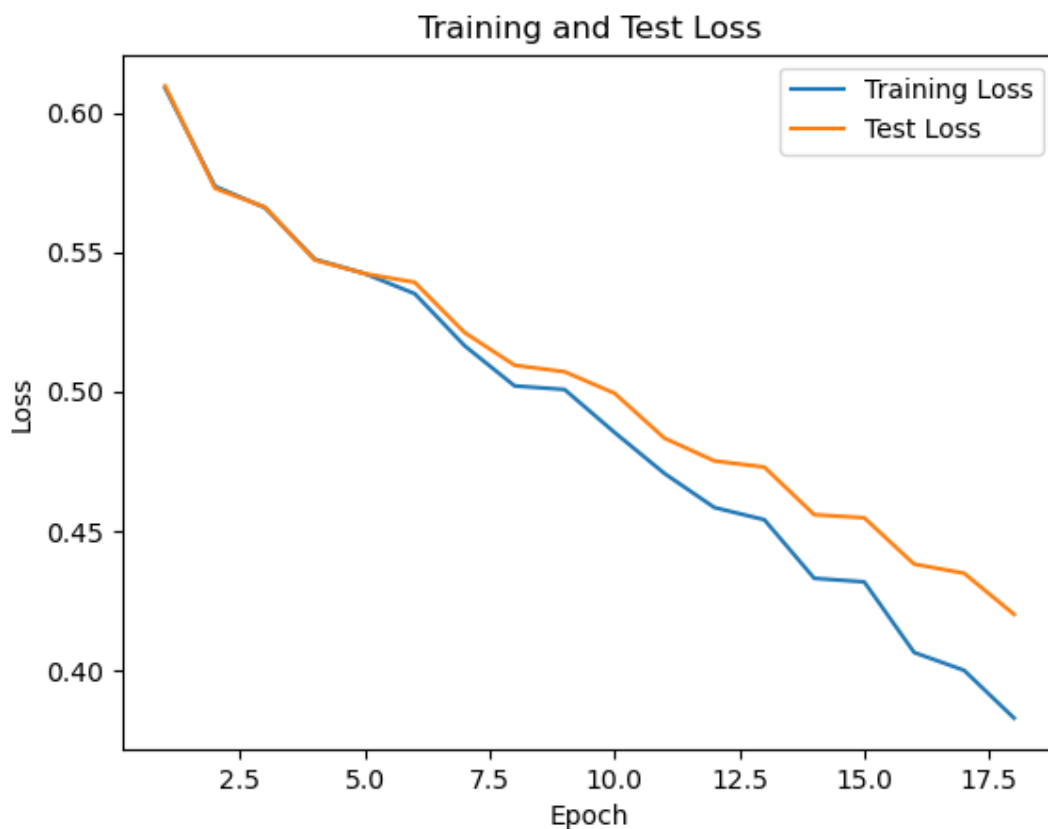
2.2

Υλοποιούμε το LSTM στο `models.py`.

Χρησιμοποιούμε την τελευταία έξοδο του LSTM ως την αναπαράσταση του κειμένου. Η τελευταία έξοδος μπορεί να βρεθεί από τα lengths αφού έχουμε αγνοήσει τα μηδενικά.

Τρέχουμε την main.py με είσοδο από το terminal “LSTM”.

Είχαμε early stopping μετά από 18 εποχές και το μοντέλο είχε test accuracy 0.8107.



Train accuracy: 0.829875

Test accuracy: 0.8107

Train F1 score: 0.8293180193597698

Test F1 score: 0.8102499717555129

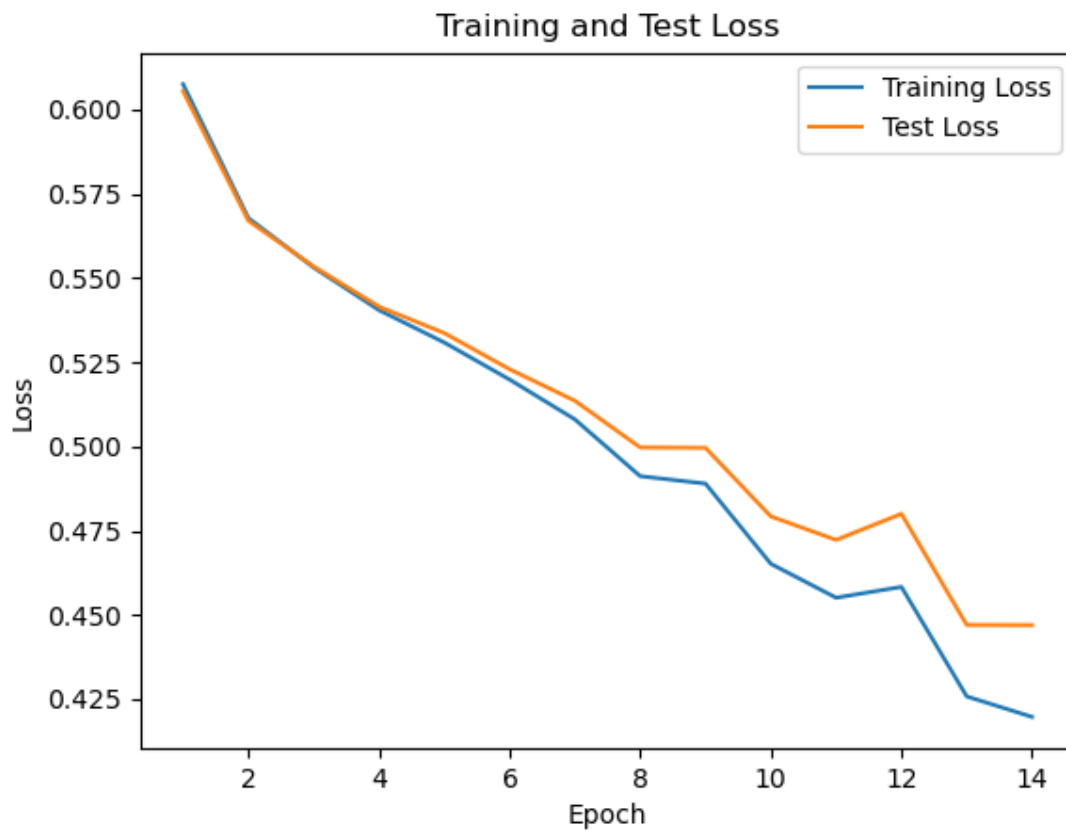
Train Recall: 0.8332116471099522

Test Recall: 0.813675766716657

2.3

Τρέχουμε ξανά την `main.py` με είσοδο “LSTMbi”, θέτοντας την παράμετρο `bidirectional = True`.

Τα αποτελέσματα είναι παρόμοια με το απλό LSTM, ίσως λίγο καλύτερα γιατί σταμάτησε μετά από 14 epochs.



Train accuracy: 0.82175

Test accuracy: 0.8037

Train F1 score: 0.8217303457706213

Test F1 score: 0.8036999509249877

Train Recall: 0.8217523761515696

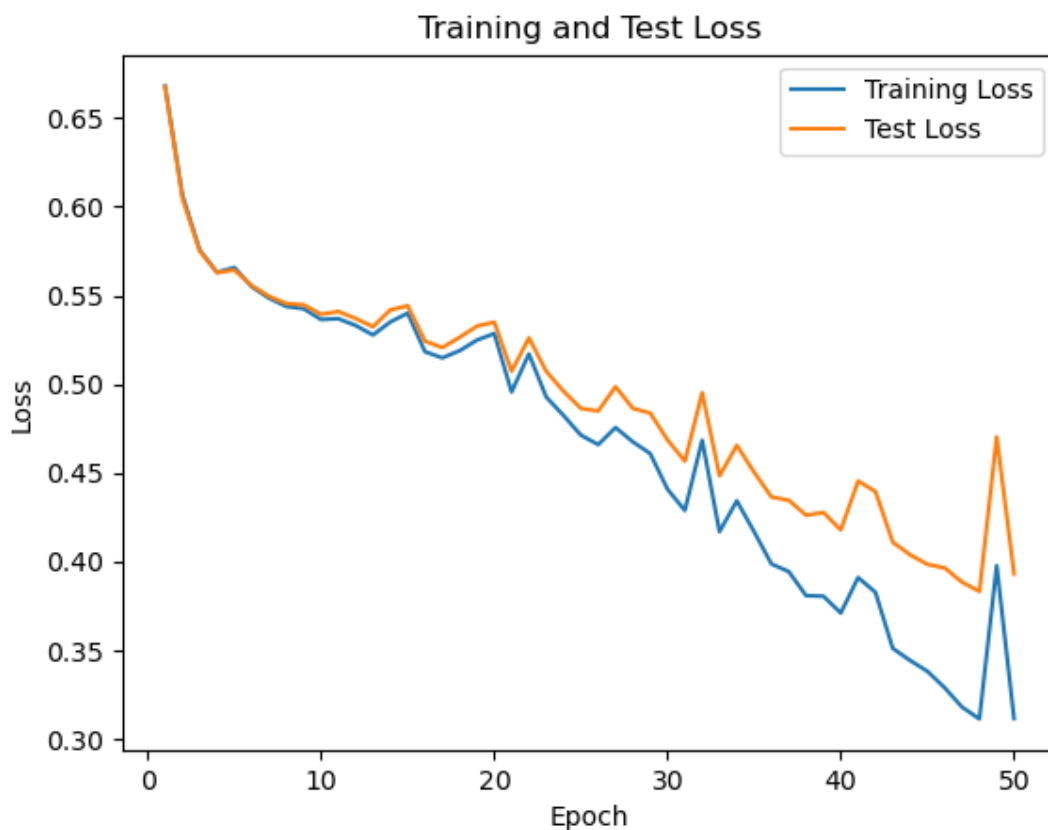
Test Recall: 0.8037003037003037

Ερώτημα 3

3.1

Συμπληρώνουμε τα κενά στο SimpleSelfAttentionModel στο attention.py.

Τρέχουμε την main.py με είσοδο “SimpleSelfAttentionModel”.



Train accuracy: 0.869375

Test accuracy: 0.8342

Train F1 score: 0.8690965007599998

Test F1 score: 0.8336633580723474

Train Recall: 0.8735299998440578

Test Recall: 0.8385692222706336

3.2

Τα queries αντιπροσωπεύουν τι πληροφορίες ψάχνει το κάθε token.

Τα keys αντιπροσωπεύουν τι πληροφορίες έχει το κάθε token.

Με το εσωτερικό γινόμενο του query ενός token A με το key ενός token B παίρνουμε ένα μέτρο της συσχέτισης της πληροφορίας που ψάχνει ο A σε σχέση με αυτή που έχει ο B. Μετά από την κανονικοποίηση και το softmax παίρνουμε για κάθε token τα βάρη προσοχής (attention) για όλα τα υπόλοιπα.

Τα values αντιπροσωπεύουν τις πληροφορίες που θα επικοινωνήσει το κάθε token στα υπόλοιπα και πολλαπλασιάζονται με τα αντίστοιχα βάρη.

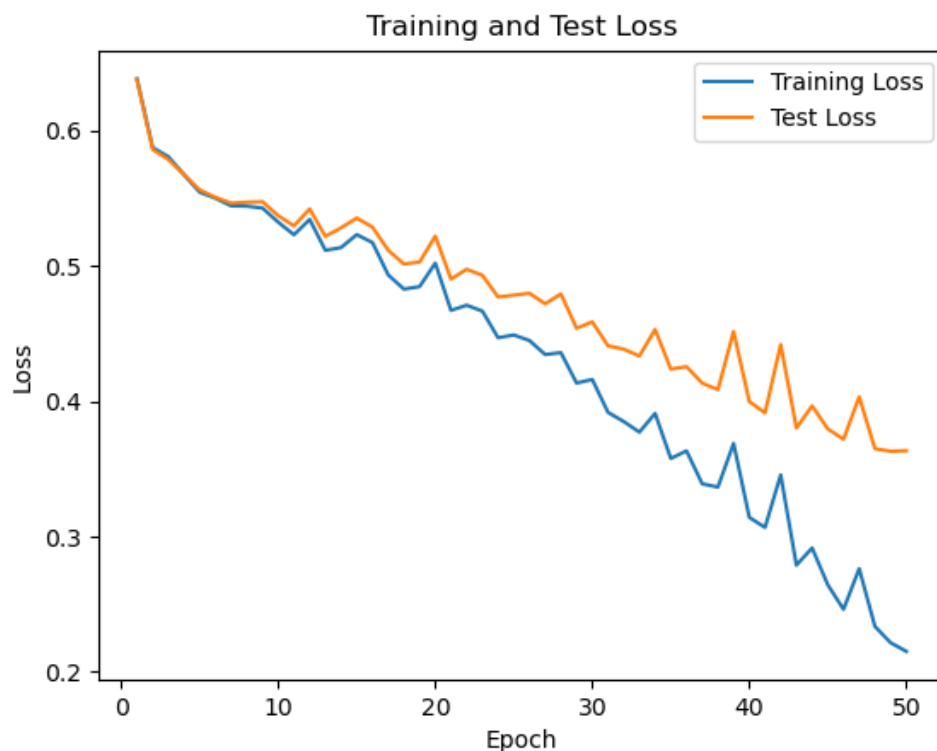
Τα position embeddings κωδικοποιούν την θέση του κάθε token μέσα στην πρόταση και αθροίζονται με τα input embeddings. Προσφέρουν πληροφορία για το πόσο κοντά είναι δύο tokens μεταξύ τους, η οποία είναι χρήσιμη για να κατανοήσει το μοντέλο το context.

Ερώτημα 4

Συμπληρώνουμε τα κενά στο MultiHeadAttentionModel στο attention.py, χρησιμοποιώντας το MultiHeadAttention. Χρησιμοποιούμε 5 heads.

Τρέχουμε την main.py με είσοδο “MultiHeadAttentionModel”.

Η επίδοση ήταν καλύτερη από το SimpleSelfAttention.



Train accuracy: 0.912875

Test accuracy: 0.866

Train F1 score: 0.9128523402467819

Test F1 score: 0.865918859456989

Train Recall: 0.913706229949157

Test Recall: 0.8668881040201153

Ερώτημα 5

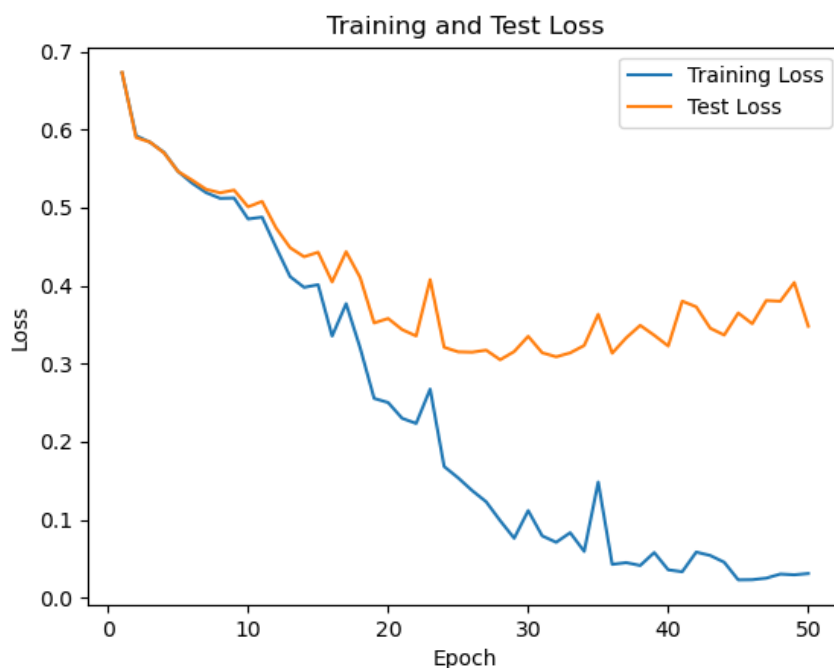
Συμπληρώνουμε τα κενά στο TransformerEncoderModel στο attention.py.

Η διαφορά του σε σχέση με το MultiheadAttentionModel είναι ότι έχει πολλά blocks, που αποτελούνται από multihead attentions ακολουθούμενα από feedforward επίπεδα με LayerNorm, το ένα μετά το άλλο. Στο τέλος υπάρχει ένα τελευταίο επίπεδο LayerNorm και στη συνέχεια γίνεται το average pooling πριν περάσει στο επίπεδο της εξόδου.

Για τις παραμέτρους επιλέγουμε τις τιμές $n_head=5$ και $n_layer=3$.

Στην κλασική αρχιτεκτονική του Transformer δηλαδή στο paper “Attention is all you need”, οι τιμές τους είναι $n_head=8$ και $n_layer=6$.

Τρέχουμε την main.py με είσοδο “TransformerEncoderModel”.



Train accuracy: 0.990375

Test accuracy: 0.9254

Train F1 score: 0.9903727242022571

Test F1 score: 0.9253752813081805

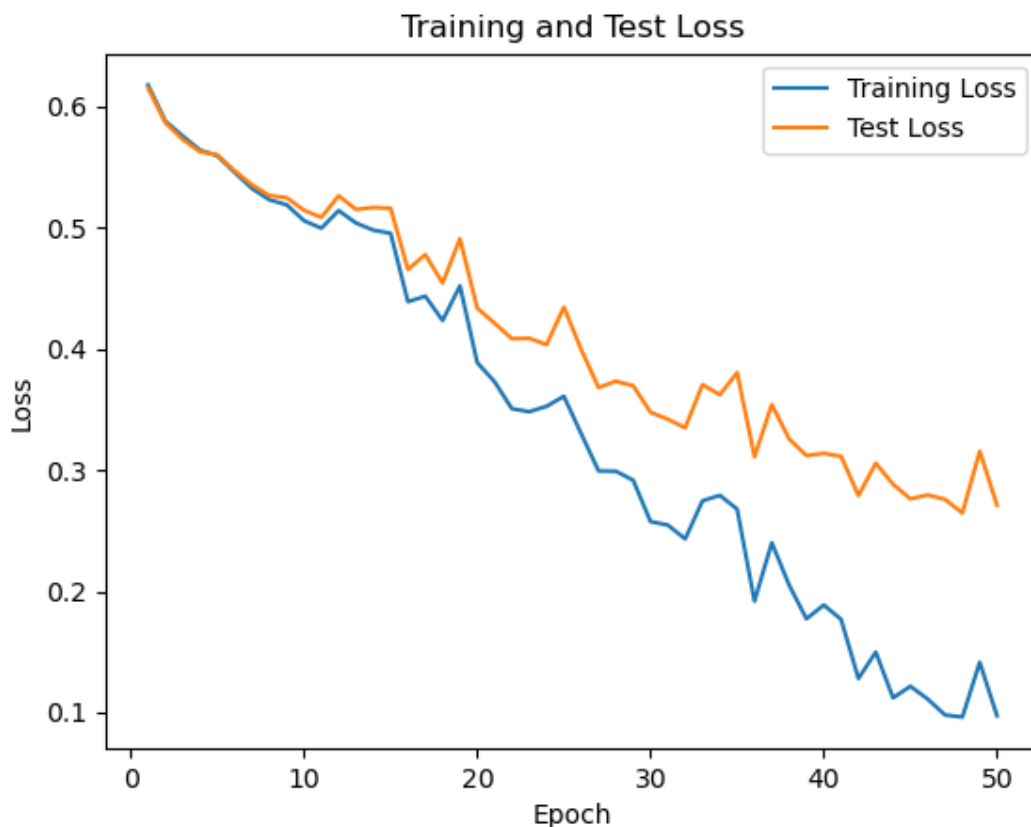
Train Recall: 0.9905378840135948

Test Recall: 0.9259643857725732

Τα αποτελέσματα μετά από 50 εποχές είναι πάρα πολύ καλά σε σημείο που έχουμε overfitting.

Μπορεί να αντιμετωπιστεί αν προσθέσουμε dropout.

Με dropout=0.2 παντού αντιμετωπίζουμε όντως το overfitting.



Train accuracy: 0.96975

Test accuracy: 0.9143

Train F1 score: 0.9697458230326924

Test F1 score: 0.9142994643716522

Train Recall: 0.9698008696589355

Test Recall: 0.914310357758944

Ερώτημα 6

Στην πρώτη γραμμή του πίνακα έχουμε το Pre-Trained model, και στις τρεις επόμενες το Test set evaluation.

Για το “MR” dataset:

Model	siebert/sentiment-roberta-large-english	textattack/bert-base-uncased-imdb	textattack/roberta-base-imdb
Accuracy	0.9259818731117825	0.7960725075528701	0.8625377643504532
Recall	0.9259818731117825	0.7960725075528701	0.8625377643504532
F1-score	0.9259803530069484	0.795500882112677	0.86227346405946

Την καλύτερη επίδοση την πετυχαίνει το siebert/sentiment-roberta-large-english και είναι παρόμοια με την επίδοση του δικού μας transformer ο οποίος όμως είχε εκπαιδευτεί πάνω στο συγκεκριμένο dataset.

Για το “Semeval2017A” dataset:

Model	cardiffnlp/twitter-roberta-base-sentiment	lxyuan/distilbert-base-multilingual-cased-sentiments-student	finiteautomata/bertweet-base-sentiment-analysis
Accuracy	0.7237870400521003	0.4304786714425269	0.7177629436665581
Recall	0.7229454214750545	0.5567295892216589	0.7301871228078923
F1-score	0.7222115953560642	0.38769025068181867	0.718050644575488

Την καλύτερη επίδοση την πετυχαίνει το
cardiffnlp/twitter-roberta-base-sentiment.

Ερώτημα 7

```
DATASET = 'MR' # 'MR' or 'Semeval2017A'
```

```
PRETRAINED_MODEL = 'bert-base-cased'
```

```
{'eval_loss': 0.693094789981842, 'eval_accuracy': 0.5, 'eval_runtime': 27.9744,  
'eval_samples_per_second': 1.43, 'eval_steps_per_second': 0.179, 'epoch': 1.0}
```

```
{'eval_loss': 0.660019040107727, 'eval_accuracy': 0.675, 'eval_runtime': 46.4,  
'eval_samples_per_second': 0.862, 'eval_steps_per_second': 0.108, 'epoch': 2.0}
```

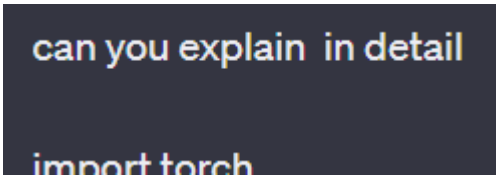
```
{'eval_loss': 0.6627224683761597, 'eval_accuracy': 0.6, 'eval_runtime':  
40.5754, 'eval_samples_per_second': 0.986, 'eval_steps_per_second': 0.123,  
'epoch': 3.0}
```

```
{'eval_loss': 0.6475683450698853, 'eval_accuracy': 0.6, 'eval_runtime': 51.229,  
'eval_samples_per_second': 0.781, 'eval_steps_per_second': 0.098, 'epoch': 4.0}
```

```
{'eval_loss': 0.6323204040527344, 'eval_accuracy': 0.625, 'eval_runtime':  
42.9483, 'eval_samples_per_second': 0.931, 'eval_steps_per_second': 0.116,  
'epoch': 5.0}
```

```
{'train_runtime': 1053.3639, 'train_samples_per_second': 0.19,  
'train_steps_per_second': 0.024, 'train_loss': 0.5124170303344726, 'epoch': 5.0}
```

Ερώτημα 8 (bonus)



can you explain in detail

import torch

The provided code implements a language model based on the Transformer architecture. Let's go through the code step by step to understand its functionality.

1. The code imports the required libraries, including `torch`, `torch.nn`, and `torch.nn.functional` (renamed as `F` for convenience).
2. Hyperparameters are defined, such as `batch_size`, `block_size`, `max_iters`, `eval_interval`, `learning_rate`, `device`, `eval_iters`, `n_embd`, `n_head`, `n_layer`, and `dropout`.
3. The random seed is set to ensure reproducibility of the results.

Φαίνεται ότι μπορεί να εξηγήσει πολύ καλά τον κώδικα.

can you evaluate the code

Overall, the code seems to be well-implemented, following best practices for building and training Transformer-based language models. However, to provide a comprehensive evaluation, it would be necessary to assess the model's performance in terms of loss convergence, text generation quality, and potential overfitting. Additional evaluation metrics and techniques, such as perplexity and test set evaluation, could be considered to further evaluate the model's performance.

Δεν μπορεί να αξιολογήσει τον κώδικα ως προς την απόδοσή του, μόνο ως προς την λειτουργικότητα. Δεν έχει κάποια βαθύτερη γνώση (η μάλλον αυτογνωσία) για το πως επηρεάζει εσωτερικά ο κώδικας την απόδοση.

can you refactor some parts of it

Certainly! Here's a refactored version of some parts of the code that can improve readability and maintainability:

These are just a few suggestions for refactoring the code to improve readability and maintainability. Remember to adapt the changes to the relevant parts of the code and ensure they do not introduce any unintended effects on the overall functionality.

Δεν εμφανίζεται καθόλου σίγουρο για την λειτουργικότητα του refactored κώδικα που έδωσε. Εξ εμπειρίας, αρκετές φορές, ο κώδικας που δίνει έχει bugs τα οποία κάποιες φορές τα λύνει μόνο του ενώ άλλες φορές χρειάζεται αρκετή ώρα και βαθιά κατανόηση από τον προγραμματιστή για διορθώσει τον (φαινομενικά ορθό) hallucinated κώδικα του ChatGPT.