ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ακαδημαϊκό έτος 2022-2023

Επεξεργασία Φωνής και Φυσικής Γλώσσας

Αναφορά της 3ης εργαστηριακής άσκησης

Γκοτζιάς Γεώργιος - 03119047

Δημήτρης Δαμιανός – 03119825

Εισαγωγή:

Για τις ανάγκες τις προπαρασκευής δημιουργήσαμε ένα βαθύ νευρωνικό δίκτυο με σκοπό την κατηγοριοποίηση κειμένων.

Συγκεκριμένα, χρησιμοποιήσαμε προ-εκπαιδευμένες αναπαραστάσεις λέξεων, ενώ αναπαραστήσαμε τις προτάσεις ως τους μέσους όρους των αναπαραστάσεων των λέξεων τους.

Σε αυτή την εργασία θα εξετάσουμε εναλλακτικές μεθόδους αναπαραστάσεων που μας δίνουν καλύτερα αποτελέσματα.

Σε αυτό το σημείο να σημειώσουμε ότι στην εκπαίδευση των μοντέλων διατηρήσαμε των αριθμό εποχών στις 10, για λόγους εξοικονόμησης χρόνου.

Επίσης, για τα ερωτήματα 1 έως και 5 τα αποτελέσματα της αναφοράς προκύπτουν από εκτέλεση με το Dataset MR.

Ερωτήματα:

Ερώτημα 1:

Σε αυτό το ερώτημα υπολογίσαμε τις αναπαραστάσεις των προτάσεων ως την συνένωση του μέσου όρου (mean pooling) και του μεγίστου (max pooling) των λέξεων τους.

Με αυτή την αναπαράσταση, αντί να λαμβάνουμε υπόψη μας μόνο τον μέσο όρο των λέξεων (που μας δίνει μια πολύ ασαφή εικόνα για την πρόταση και το περιεχόμενό της), λαμβάνουμε υπόψη μας και την λέξη με την μέγιστη αναπαράσταση.

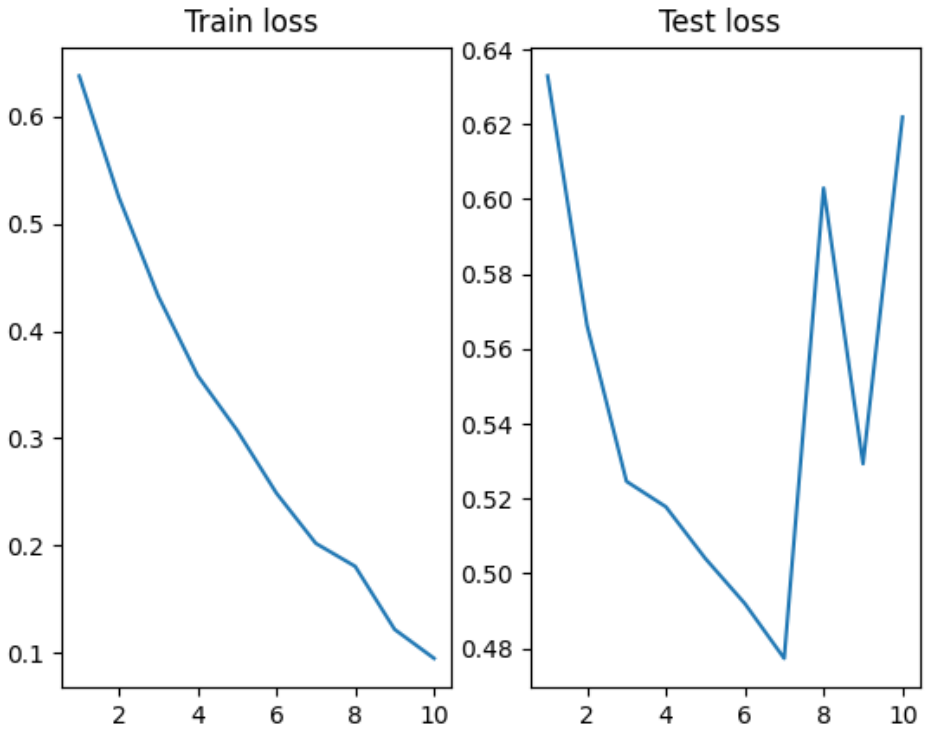
Αυτό μπορεί να φανεί χρήσιμο, καθώς η λέξη με την μέγιστη αναπαράσταση ίσως καθορίζει καλύτερα το περιεχόμενο της πρότασης. Συνεπώς, πετυχαίνουμε μια καλύτερη αναπαράσταση της πρότασης.

Εκπαιδεύοντας το BaselineDNN με αυτές τις αναπαραστάσεις, είχαμε τα παρακάτω αποτελέσματα:

Κατά την εκπαίδευση:

```
[==========================================] ...Epoch 1, Loss: 0.6739
[==========================================] ...Epoch 2, Loss: 0.5210
[==========================================] ...Epoch 3, Loss: 0.3993
[==========================================] ...Epoch 4, Loss: 0.5257
[==========================================] ...Epoch 5, Loss: 0.3932
[==========================================] ...Epoch 6, Loss: 0.1686
[==========================================] ...Epoch 7, Loss: 0.1995
[==========================================] ...Epoch 8, Loss: 0.0669
[==========================================] ...Epoch 9, Loss: 0.1130
[==========================================] ...Epoch 10, Loss: 0.1446
```

Το loss κατά την δοκιμή στα test και train sets:



Παρατηρούμε ότι κατά την εκπαίδευση πετυχαίνουμε καλύτερα αποτελέσματα (όπως περιμέναμε) συγκριτικά με την προπαρασκευή, όμως τα αποτελέσματα του loss στα δύο sets είναι περίπου τα ίδια.

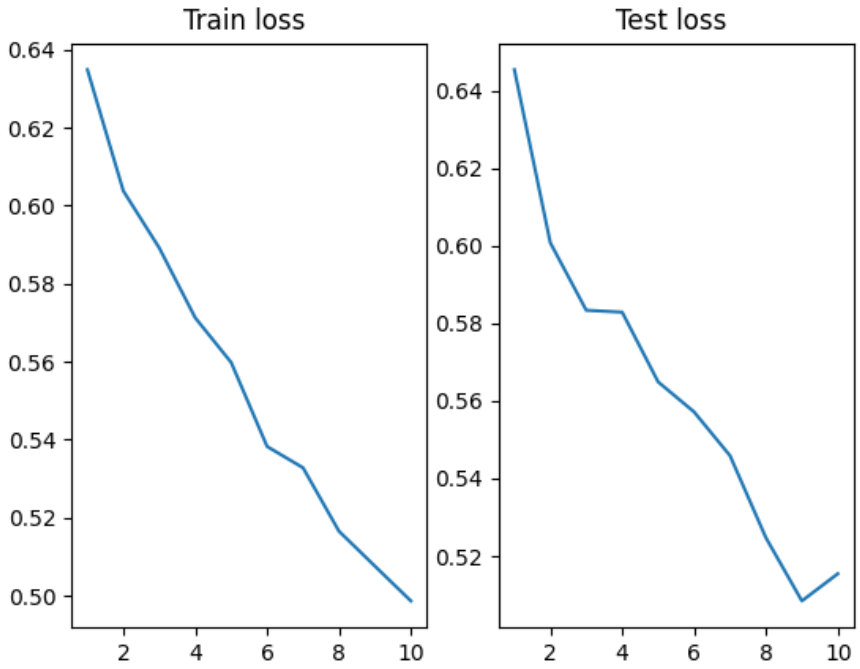
Για αυτό πιθανό να οφείλεται ο χαμηλός αριθμός εποχών εκπαίδευσης.

Ερώτημα 2:

Σε αυτό το ερώτημα θα χρησιμοποιήσουμε ένα αμφίδρομο LSTM ώστε να πετύχουμε καλύτερα αποτελέσματα.

Για τον υπολογισμό των τελικών αναπαραστάσεων, παίρνουμε την τελική έξοδο του LSTM , και εξαιρώντας τα zero-paddings, υπολογίζουμε τον μέσο όρο.

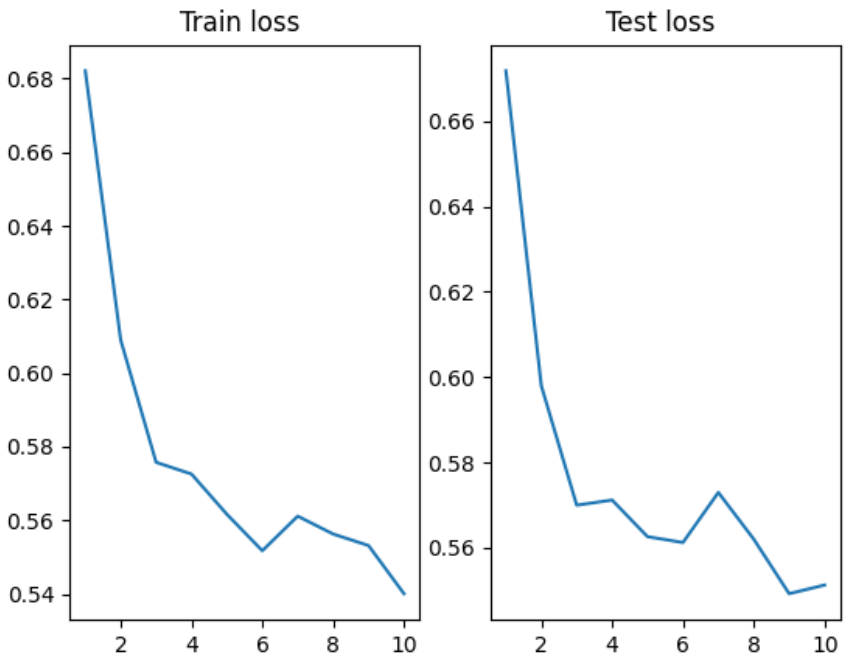
Τα αποτελέσματα της εκπαίδευσης είναι τα ακόλουθα:



Παρατηρούμε ότι ακόμα και στις 10 εποχές, το test loss είναι αρκετά καλύτερο από ότι στο BaselineDNN.

Ερώτημα 3:

3.1. Χρησιμοποιώντας το μοντέλο Single Self Attention για εκπαίδευση ως 10 εποχές παίρνουμε τα παρακάτω αποτελέσματα σχετικά με το train και το test loss:



Για το παραπάνω μοντέλο η επίδοση του για εκπαίδευση για 10 εποχές υπολογίζεται ότι είναι στο train set:

```
### Train set metrics:

Accuracy:0.74375
F1 score:0.7436997651539701
Recall:0.7436855958781362
```

Ενώ αντίστοιχα στο test set, όπου και μας ενδιαφέρει:

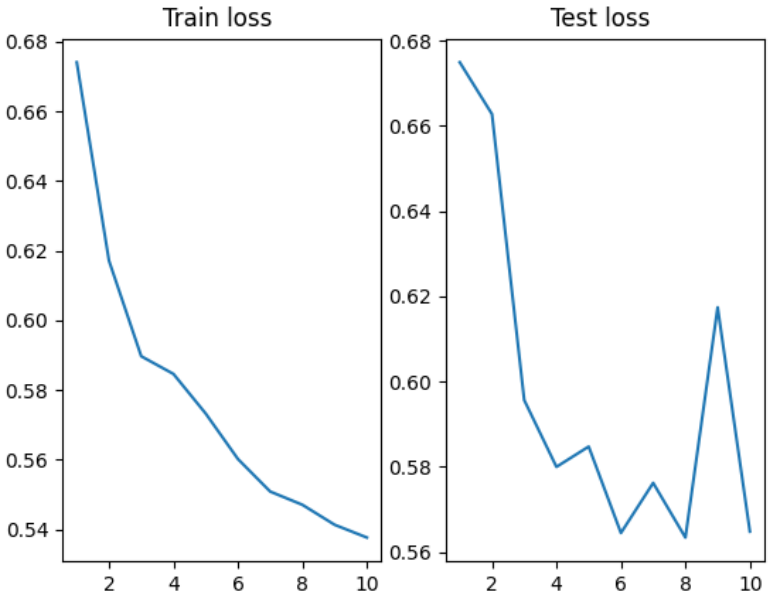
```
### Test set metrics:

Accuracy:0.7039274924471299
F1 score:0.6967277486910994
Recall:0.7039274924471299
```

3.2. Τα queries δηλώνουν τι μας ενδιαφέρει, τα keys δείχνουν τι πληροφορία έχει και τα values δείχνουν το τι πληροφορία θα δώσουν, αν τα θεωρούν ενδιαφέροντα.

Τα position_embeddings προσθέτουν πληροφορία σχετικά με τη θέση του token στην πρόταση και συμβάλλουν στο να μειωθεί η ανεξαρτησία από τα δεδομένα.

Ερώτημα 4:

Το Multi Head Attention Model, για 5 heads, που δημιουργήσαμε έδινε τα παρακάτω αποτελέσματα για το train και test loss, για 10 εποχές:

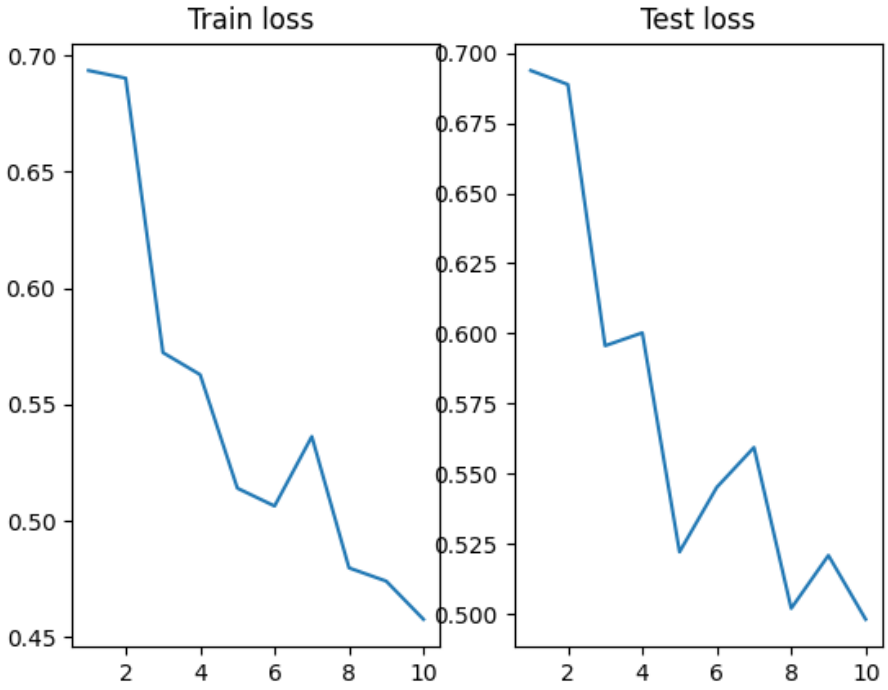


Ενώ οι αντίστοιχες μετρικές είναι:

```
### Train set metrics:

Accuracy:0.720875
F1 score:0.719094016562603
Recall:0.7202840981822836

### Test set metrics:

Accuracy:0.7129909365558912
F1 score:0.710109521351132
Recall:0.7129909365558913
```

Ερώτημα 5:

Στο Transformer-Encoder μοντέλο χρησιμοποιούμε περισσότερα από 1 Multi-Head Attention Models πριν το τελικό layer. Για το Multi-Head model επιλέξαμε να έχει 5 heads και για τον Transformer 4 layers, με βάση τις δοκιμές. Σχετικά με το loss παίρνουμε τα παρακάτω αποτελέσματα:



Ενώ, για την επίδοση του μοντέλου παίρνουμε τις παρακάτω τιμές για τις μετρικές:

```
### Train set metrics:

Accuracy:0.779875
F1 score:0.7795331903251564
Recall:0.7796098950332822

### Test set metrics:

Accuracy:0.7583081570996979
F1 score:0.7573313782991202
Recall:0.7583081570996979
```

Παρατηρούμε ότι με τη χρήση Transformers παρατηρείται βελτίωση.

Σχετικά με την κλασική αρχιτεκτονική των Transformers, για τον Encoder χρησιμοποιείται 1 Multi-Head Attention Model αρχικά και ένα στη συνέχεια, το οποίο έχει ως είσοδο και τα δεδομένα του Decoder, ο οποίος χρησιμοποιεί και αυτός Multi-Head Attention Model, όμως δεν υλοποιειείται στα πλαίσια της άσκησης.

Ερώτημα 6:

Τα pretrained μοντέλα που εξετάστηκαν είναι τα παρακάτω:

- siebert/sentiment-roberta-large-english
- cardinffnlp/twitter-roberta-base-sentiment
- lxyuan/distilbert-base-multilingual-cased-sentiments-student

Για το dataset MR:

Metric\ Model	siebert/sentiment-roberta-large-english	cardiffnlp/ twitter-roberta-base-sentiment	lxyuan/distilbert-base-multilingual-cased-sentiments-student
Accuracy	0.92598187311178205	0.7507552870090635	0.4304786714425269
recall	0.92598187311178205	0.7507552870090635	0.5567295892216589
f1-score	0.9259803530069484	0.7461674137798815	0.3876902506818187

Στο παραπάνω dataset δεν έχουμε neutral κλάση, οπότε για τα μοντέλα που διακρίνουν προτάσεις ως neutral επιλέξαμε να κατηγοριοποιούνται ως positive.

Για το dataset Semeval2017A:

Metric\ Model	siebert/sentiment-roberta-large-english	cardiffnlp/ twitter-roberta-base-sentiment	lxyuan/distilbert-base-multilingual-cased-sentiments-student
Accuracy	0.46523933572126321003 4	0.72378704005	0.430478671442 5269
recall	0.60380989735163650545 9	0.72294542147	0.556729589221 6589
f1-score	0.40353269309350753	0.72221159535 60642	0.387690250681 81867

Το 1ο μοντέλο έχει χαμηλή ακρίβεια, καθώς δεν κατηγοριοποιεί στην κλάση neutral, οπότε για όλα τα neutral δεδομένα του test set κάνει λάθος πρόβλεψη.

Παρατηρούμε ότι το πρώτο μοντέλο στο dataset MR έχει την καλύτερη επίδοση από όλα, αντίθετα το τελευταίο μοντέλο έχει την χαμηλότερη επίδοση, όμως ο χρόνος που απαιτούσε ήταν σημαντικά μικρότερος από κάθε άλλο μοντέλο.

Ερώτημα 7:

Ο χρόνος για να εκτελεστούν όσα ζητούνταν από την εκφώνηση δεν επαρκούσε, γι’ αυτό και θα παρουσιαστούν πλήρη δεδομένα μόνο για το μοντέλο bert-base- cased για το dataset MR. Για τα υπόλοιπα δεδομένα που παρουσιάζονται εκτελέστηκε train για 5 εποχές και για ένα μέρος του dataset της τάξης των 500 δειγμάτων.

Τα αποτελέσματα για το Evaluation Loss είναι τα ακόλουθα:

Dataset\ Model	bert-base-cased	bert-base-uncased
MR	1.021601	1.1272886991500854
Semeval2017A	1.7342239618301392	1.6751627922058105

Αντίστοιχα, τα αποτελέσματα για το Evaluation Accuracy είναι τα ακόλουθα:

Dataset\ Model	bert-base-cased	bert-base-uncased
MR	0.838369	0.812
Semeval2017A	0.5975	0.616

Αντίστοιχα, τα αποτελέσματα για το Train Loss είναι τα ακόλουθα:

Dataset\ Model	bert-base-cased	bert-base-uncased
MR	0.068100	0.19360815381246899
Semeval2017A	0.4739900817871094	0.4343234592013889

Σχετικά με την περίπτωση του μοντέλου bert-base-cased για το Dataset εκτελέσαμε στο Colab για το σύνολο των δεδομένων και για 15 εποχές και οι τιμές για το loss και το accuracy συναρτήσει των εποχών εκπαίδευσης παίρνουμε τα ακόλουθα αποτελέσματα:

Epoch	Training Loss	Validation Loss	Accuracy
1	0.453900	0.433088	0.835347
2	0.319500	0.604717	0.827795
3	0.194100	0.975506	0.820242
4	0.105200	1.016671	0.827795
5	0.068100	1.021601	0.838369
6	0.049000	1.347341	0.830816
7	0.039700	1.268799	0.835347
8	0.042500	1.171313	0.838369
9	0.029400	1.369123	0.839879
10	0.017300	1.435641	0.835347
11	0.005400	1.468055	0.848943
12	0.009100	1.385998	0.851964
13	0.007500	1.403171	0.845921
14	0.003900	1.545054	0.842900
15	0.002600	1.552148	0.841390

Ερώτημα 8:

Δώσαμε στο prompt του ChatGPT τον ολοκληρωμένο κώδικα του link και του ζητήσαμε να μας τον εξηγήσει.

Το ChatGPT χώρισε τον κώδικα σε τμήματα και μας εξήγησε τον κώδικα τμήμα-τμήμα, όπως το ακόλουθο:

1. Defining the generation method for the language model:

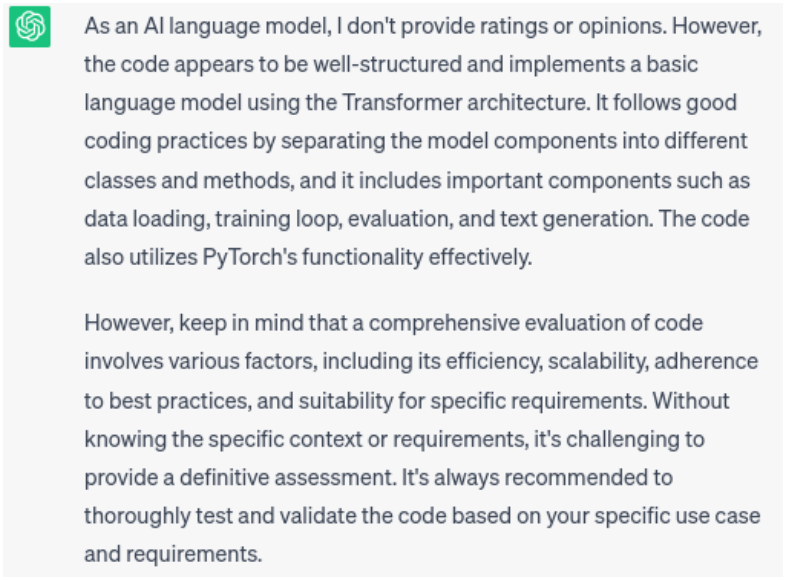
python Copy code

```python
def generate(self, idx, max_new_tokens):
    for _ in range(max_new_tokens):
        idx_cond = idx[:, -block_size:]
        logits, loss = self(idx_cond)
        logits = logits[:, -1, :]
        probs = F.softmax(logits, dim=-1)
        idx_next = torch.multinomial(probs, num_sampl
        idx = torch.cat((idx, idx_next), dim=1)
    return idx
```

This method generates new text based on a given initial context (`idx`). It predicts the next token for a maximum number of iterations (`max_new_tokens`). At each step, it selects the last `block_size` tokens from the context, gets the logits from the model, applies softmax to obtain probabilities, and samples the next token using multinomial sampling. The sampled token is then appended to the context, and the process continues until the desired number of tokens is generated.

Τα τμήματα που δημιούργησε ήταν πολύ καλά επιλεγμένα και πράγματι βοηθούσαν τον χρήστη στο να εστιάσει σε λεπτομέρειες και να καταλάβει καλύτερα.

Του ζητήσαμε στην συνέχεια να αξιολογήσει τον κώδικα και η απάντησή του ήταν η εξής:



As an AI language model, I don't provide ratings or opinions. However, the code appears to be well-structured and implements a basic language model using the Transformer architecture. It follows good coding practices by separating the model components into different classes and methods, and it includes important components such as data loading, training loop, evaluation, and text generation. The code also utilizes PyTorch's functionality effectively.

However, keep in mind that a comprehensive evaluation of code involves various factors, including its efficiency, scalability, adherence to best practices, and suitability for specific requirements. Without knowing the specific context or requirements, it's challenging to provide a definitive assessment. It's always recommended to thoroughly test and validate the code based on your specific use case and requirements.

Η απάντηση του επικεντρώνεται καθαρά στις λειτουργικότητες που υλοποιεί ο κώδικας και πως χρησιμοποιεί τα εργαλεία της PyTorch. Λαμβάνοντας υπόψη το “disclaimer” στην τελευταία παράγραφο, θεωρώ ότι είναι μια καλά δομημένη απάντηση.

Τέλος, του ζητήσαμε να ξαναγράψει κάποια μέρη του κώδικα, και τα αποτελέσματα φαίνεται να ήταν τα ίδια με τα αρχικά.