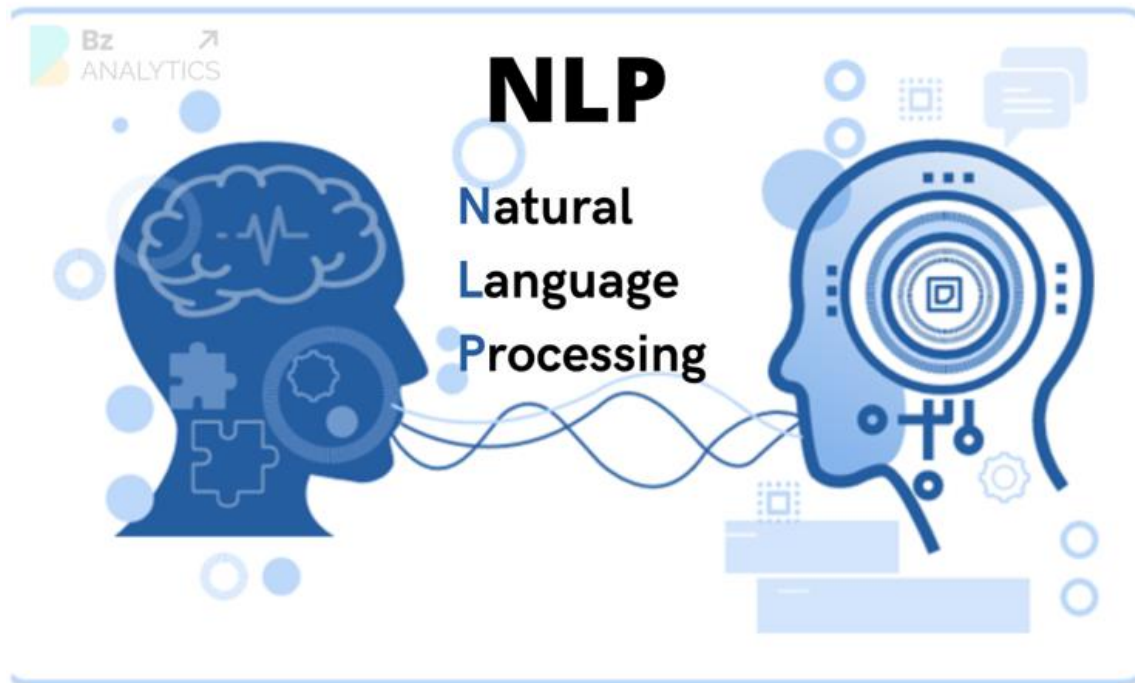


*Εθνικό Μετσόβιο Πολυτεχνείο*  
*Επεξεργασία Φωνής και Φυσικής Γλώσσας*  
*Εργαστήριο 2: Sentiment Analysis*



*Συγγραφείς: Κατσαϊδώνης Νικόλαος 03121868*  
*Τζαμουράνης Γεώργιος 03121141*

# ΠΡΟΠΑΡΑΣΚΕΥΑΣΤΙΚΗ ΕΡΓΑΣΙΑ

! Στα παρακάτω ερωτήματα, σε όσα δεν απαιτούνταν χρήση και των 2 datasets, χρησιμοποιήθηκε το Semeval !

## ΖΗΤΟΥΜΕΝΟ 4

Συμπληρώστε τα κενά στη θέση main.py:EX1 και τυπώστε τα πρώτα 10 labels από τα δεδομένα εκπαίδευσης και τις αντιστοιχίες τους σε αριθμούς.

Τυπώνουμε τα ζητούμενα, καθώς και ένα παράδειγμα από κάθε label.

```
Mapping κατηγορίας → αριθμού:
negative → 0
neutral → 1
positive → 2

Original labels (πριν το encoding):
[1 2 1 2 2 2 1 2 0 1]

So the corresponding list with the actual values is:
['neutral', 'positive', 'neutral', 'positive', 'positive', 'positive', 'neutral', 'positive', 'negative', 'neutral']

Lets print one neutral, one postive and one negative tweet:
Neutral:
05 Beat it - Michael Jackson - Thriller (25th Anniversary Edition) [HD] http://t.co/A4K2B86PBv
Positive:
Jay Z joins Instagram with nostalgic tribute to Michael Jackson: Jay Z apparently joined Instagram on Saturday and.. http://t.co/Qj9I4eCvxy
Negative:
@etbowser do u enjoy his 2nd rate Michael Jackson bit? Honest ques. Like the can't feel face song but god it's so obvious they want MJ 2.0
PS c:\git_repo\slp-labs\lab3>
```

## ZΗΤΟΥΜΕΝΟ 2:

**Συμπληρώστε τα κενά στη θέση dataloading.py:EX2 και τυπώστε τα πρώτα 10 παραδείγματα από τα δεδομένα εκπαίδευσης.**

Παρατηρούμε πώς τα data είναι λίστες από tokens – λέξεις:

```
Lets print some tokenized examples:

Tokenized data 0: ['0s', 'beat', 'it', '-', 'michael', 'jackson', '-', 'thriller', '(', '25th', 'anniversary', 'edition', ')', '[', 'hd', ']', 'http://t.co/a4k2b8spbv']

Tokenized data 1: ['jay', 'z', 'joins', 'instagram', 'with', 'nostalgic', 'tribute', 'to', 'michael', 'jackson', ':', 'jay', 'z', 'apparently', 'joined', 'instagram', 'on', 'saturday', 'and', '...', 'http://t.co/qj9i4ecvay']

Tokenized data 2: ['michael', 'jackson', ':', 'bad', '25th', 'anniversary', 'edition', '(', 'picture', 'vinyl', ')', 'this', 'unique', 'picture', 'disc', 'vinyl', 'includes', 'the', 'original', '1', 'http://t.co/fh0t0aaaw']

Tokenized data 3: ['i', 'liked', 'a', '@youtube', 'video', 'http://t.co/aar3pjp2pi', 'one', 'direction', 'singing', '', 'man', 'in', 'the', 'mirror', '', 'by', 'michael', 'jackson', 'in', 'atlanta', 'a', 'ga', '[', 'june', '26', ',']

Tokenized data 4: ['18th', 'anniv', 'of', 'princess', 'diana's', 'death', '-', 'i', 'still', 'want', 'to', 'believe', 'she', 'is', 'living', 'on', 'a', 'private', 'island', 'away', 'from', 'the', 'public', '-', 'with', 'michael', 'jackson', '.']

Tokenized data 5: ['@bridganzjazz', 'the', '1st', 'time', 'i', 'heard', 'michael', 'jackson', 'sing', 'was', 'in', 'honolulu', ',', 'hawaii', '@', 'a', 'restaurant', 'on', 'radio', '-', 'it', 'was', 'a', '-', 'b', '-', 'c', '-', 'i', 'was', '13', '-', 'i', 'loved', 'it', 'i']

Tokenized data 6: ['', 'michael', 'jackson', '', 'appeared', 'on', 'saturday', '29', 'at', 'the', '9th', 'place', 'in', 'the', 'top', '20', 'of', 'miami's', 'trends', ':', 'http://t.co/danzfagubb', '#trndnl']
```

## ZΗΤΟΥΜΕΝΟ 3

**Υλοποιήστε τη μέθοδο getitem της κλάσης SentenceDataset (θέση dataloading.py:EX3) και τυπώστε 5 παραδείγματα στην αρχική τους μορφή και όπως τα επιστρέφει η κλάση SentenceDataset.**

```
Lets print 5 examples of the encoded sentences:

Encoded example: [ 17261   961    21    12   786   1755    12  8966    24  8962
 2350  2493   25  2824 12315  5281 400001    0    0    0
    0    0    0    0    0    0    0    0    0
    0    0    0    0    0]
Example's label: 1
Example's length: 17

Encoded example: [  4791  9027  7698 109263    18  20557  5079    5   786  1755
  46  4791  9027  1897  1031 109263    14   278    6 400001
400001    0    0    0    0    0    0    0    0    0
    0    0    0    0    0]
Example's label: 2
Example's length: 21

Encoded example: [  786  1755    46   979  8962  2350  2493    24  1836 11193
400001   38  3007  1836  5977 11193  1013    1   930   177
400001    0    0    0    0    0    0    0    0    0
    0    0    0    0    0]
Example's label: 1
Example's length: 21

Encoded example: [    42  5573    8 400001   975 400001    49  2192  4100    9
  301    7    1  6462    9    22   786  1755    7  1098
    2 12132  2824   345  1077    2    0    0    0    0
    0    0    0    0    0]
Example's label: 2
Example's length: 26
```

Πλέον, τα encoded data είναι λίστες από αριθμούς όπου κάθε αριθμός αντιστοιχίζει την κάθε λέξη στο αντίστοιχο embedding διάνυσμα. Παρατηρούμε πως το length είναι μικρότερο πολλές φορές από το μέγεθος της λίστας καθώς αυτό είναι το length πριν το zero padding που κάνουμε ώστε όλα τα embeddings να είναι μήκους max\_length.

## **ZΗΤΟΥΜΕΝΟ 4**

### **1) Γιατί αρχικοποιούμε το embedding layer με τα προ-εκπαιδευμένα word embeddings?**

Γιατί τα προ-εκπαιδευμένα embeddings (όπως GloVe) έχουν **εκπαιδευτεί πάνω σε τεράστια σώματα κειμένου** (π.χ. Wikipedia ) περιέχοντας ήδη **σημασιολογικές και συντακτικές πληροφορίες**. Μας επιτρέπουν να **ξεκινήσουμε με «γλωσσική γνώση» έτοιμη**, χωρίς να τη μάθουμε από το μηδέν. Δηλαδή, από το να φτιάξουμε εμείς από την αρχή τα word vectors που περιέχουν πληροφορία για τη συσχέτιση των λέξεων, παίρνουμε έτοιμες αυτές τις συσχετίσεις. Αυτό μπορεί να αυξήσει ιδιαίτερα την απόδοση ειδικά όταν το dataset μας είναι μικρό ή δε διαθέτουμε τον χρόνο να εκπαιδεύσουμε ένα τέτοιο σύστημα.

### **2) Γιατί κρατάμε παγωμένα τα βάρη του embedding layer κατά την εκπαίδευση?**

Αν αφήσουμε τα embeddings **να εκπαιδεύονται (trainable)**, μπορεί να αλλοιωθούν και να χάσουν τη σημασιολογική τους πληροφορία. Ιδιαίτερα αν έχουμε **λίγα δεδομένα**, το μοντέλο μπορεί να "ξεμάθει" ή να κάνει overfit. Έτσι, εμπιστευόμαστε όπως είναι τα embeddings που έχουν προκύψει από μελέτη μεγάλων συνόλων δεδομένων ακόμη κι αν αυτά είναι πιθανόν να μην είναι αρκετά καλά για το συγκεκριμένο τύπο δεδομένων που έχουμε (συγκεκριμένα για tweets).

## **ZΗΤΟΥΜΕΝΟ 5**

**Γιατί βάζουμε μία μη γραμμική συνάρτηση ενεργοποίησης στο προτελευταίο layer; Τι διαφορά θα είχε αν είχαμε 2 ή περισσότερους γραμμικούς μετασχηματισμούς στη σειρά;**

Γενικά, γνωρίζουμε πως οι μη γραμμικές συναρτήσεις ενεργοποίησης, όπως η ReLU, η tanh κλπ, είναι κατάλληλες για την δημιουργία πιο πολύπλοκων σχέσεων-μοτίβων πάνω στα δεδομένα. Από την άλλη, οι γραμμικές activation functions είναι της μορφής:

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

Ωστόσο, για δεδομένα με πολύπλοκες συσχετίσεις όπως το text δεν είναι κατάλληλες. Εξετάζοντας τώρα το ενδεχόμενο παράθεσης πολλών γραμμικών συναρτήσεων ενεργοποίησης στη σειρά, συμπεραίνουμε ότι το αποτέλεσμα είναι εξίσου ένας γραμμικός μετασχηματισμός της εισόδου. Για παράδειγμα, για  $n=2$  γραμμικές AFs στη σειρά θα είχαμε:

$$\mathbf{y} = (\mathbf{W}_2 \cdot \mathbf{W}_1)\mathbf{x} + (\mathbf{W}_2 \cdot \mathbf{b}_1 + \mathbf{b}_2) \equiv \mathbf{W}\mathbf{x} + \mathbf{b}$$

Συνεπώς, θα είχαμε πάλι ένα γραμμικό αποτέλεσμα χωρίς την δυνατότητα εκμάθησης πιο πολύπλοκων μοτίβων.

## **ZHTOYΜΕΝΟ 6**

- 1) Αν θεωρήσουμε ότι κάθε διάσταση του embedding χώρου αντιστοιχεί σε μία αφηρημένη έννοια, μπορείτε να δώσετε μία διαισθητική ερμηνεία για το τι περιγράφει η αναπαράσταση που φτιάξατε (κέντρο-βάρους);**

Αυτό που κάναμε είναι κάθε πρόταση να την χωρήσουμε σε tokens και έπειτα κάθε token να αντιστοιχεί σε ένα vector διάστασης πχ 50. Συνεπώς, έχουμε για κάθε πρόταση 32 vectors (ένα) για κάθε λέξη. Παίρνουμε τον μέσω όρο αυτών των 32 vectors και έτσι κάθε στήλη του vector αυτού αθροίζεται και διαιρείται με το πλήθος των non zero τιμών. Στην ουσία, αυτό που κάναμε είναι να δώσουμε μία ισοδύναμη «βαρύτητα» σε κάθε λέξη/token χωρίς να τονίζουμε την σημασία κάποιας λέξης και έτσι βρίσκουμε κάτι σαν το κέντρο βάρους της πρότασης.

- 2) Αναφέρετε πιθανές αδυναμίες της συγκεκριμένης προσέγγισης για να αναπαραστήσουμε κείμενα.**

Η παραπάνω προσέγγιση, μπορεί να είναι απλή και γρήγορη όμως μπορεί να μην οδηγήσει σε επιθυμητά αποτελέσματα. Για παράδειγμα, η απόδοση ίδιας βαρύτητας σε άρθρα, σημεία στίξης, ουσιαστικά, ρήματα κλπ. μπορεί να μην είναι ο πιο έξυπνος τρόπος να χειριστεί κάποιος την κατάσταση. Επιπλέον, λέξεις στην αρχή ή στο τέλος της πρότασης μπορεί να μην έχουν τόσο σημασιολογικό ενδιαφέρον σε σχέση με τις «μεσαίες» λέξεις. Γενικά, το mean pooling υστερεί στον τρόπο με τον οποίο αποδίδει βαρύτητα στις λέξεις των προτάσεων τόσο ως προς την σημασιολογία, όσο και ως προς τη σειρά των λέξεων.

## **ZΗΤΟΥΜΕΝΟ 7**

### **1) Τι συνέπειες έχουν τα μικρά και μεγάλα mini-batches στην εκπαίδευση των μοντέλων;**

Η λογική πίσω από τα mini batches είναι ότι φέρνουμε δεδομένα «κατά ομάδες» αντί για ένα-ένα. Αυτό έχει ως αποτέλεσμα να κερδίζουμε σε απόδοση/χρόνο ιδίως σε περίπτωση GPU που υποστηρίζει παράλληλη επεξεργασία. Επιπλέον, τα βάρη, που ανανεώνονται ανά mini batch, ανανεώνονται πλέον ανά ομάδες δεδομένων αντί να ανανεώνονται σε κάθε νέο δεδομένο που έρχεται ένα-ένα. Με τον τρόπο αυτό, οδηγούμε το μοντέλο στο να μάθει από περισσότερα δεδομένα πριν ανανεώσει επιπόλαια τα βάρη του και υπολογίσει τα gradients για κάθε δεδομένο.

Με βάση τα παραπάνω, όσο πιο μικρά είναι τα mini batches, τόσο πιο θορυβώδεις είναι οι τροποποιήσεις των βαρών αφού γίνονται πάνω σε μικρό μέρος των data. Επιπλέον, μικρά batches ενδέχεται να μας βοηθήσουν να αποφύγουμε τοπικά ελάχιστα. Από την άλλη, για μεγάλα mini batches έχουμε πιο σταθερά gradients ενώ ενδέχεται να «κολλήσουμε» σε τοπικά ελάχιστα πιο εύκολα.

### **2) Συνήθως ανακατεύουμε την σειρά των mini-batches στα δεδομένα εκπαίδευσης σε κάθε εποχή. Μπορείτε να εξηγήσετε γιατί;**

Φέρνοντας τα batches με την ίδια σειρά, σίγουρα δεν βοηθάει το μοντέλο στο να μαθαίνει τα δεδομένα ανεξάρτητα από την σειρά με την οποία έρχονται. Ανακατεύοντας τα δεδομένα, οδηγούμε το μοντέλο στην αγνόηση της αλληλουχίας με την οποία αυτά έρχονται και το βοηθάμε να μάθει αντί να «αποστηθίσει.»

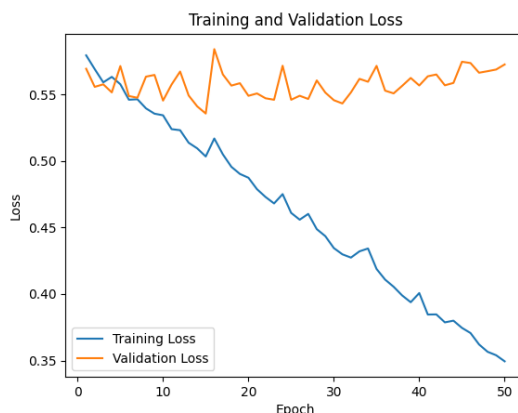
## ΖΗΤΟΥΜΕΝΟ 10

Για κάθε ένα από τα 2 datasets που σας παρέχονται αναφέρετε τις επιδόσεις του μοντέλου στις μετρικές: accuracy, F1 score (macro average), recall (macro average). Επίσης, δημιουργήστε γραφικές παραστάσεις, στις οποίες θα φαίνονται οι καμπύλες εκπαίδευσης του μοντέλου (training και test loss) ανά εποχή.

Για καθένα από τα 2 data sets, εκπαιδεύσαμε το μοντέλο μας.

Παρατηρώντας **overfitting** κυρίως στο dataset Semeval, αποφασίσαμε να προσθέσουμε **dropout**. Έτσι, εκπαιδεύσαμε τα μοντέλα για κάθε data set με και χωρίς dropout. Για τις μετρικές που αναφέρονται, προτιμήσαμε να επιλέξουμε τις **μετρικές της καλύτερης εποχής**, δηλαδή αυτής με το μικρότερο loss και να τις εκτυπώσουμε. Επιπλέον, φτιάξαμε διαγράμματα loss – epochs τα οποία και παραθέτουμε παρακάτω:

- **MR dataset χωρίς dropout:**



Best epoch: 31  
Accuracy: 0.7039274924471299  
F1: 0.7009550851832731  
Recall: 0.712371085024476



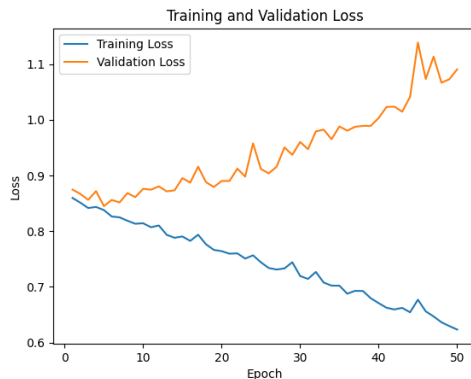
## MR dataset με dropout:



Best epoch: 31  
Accuracy: 0.7084592145015106  
F1: 0.705962630286544  
Recall: 0.7157879818594104

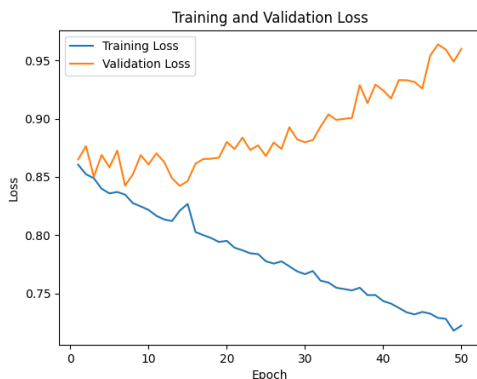
Παρατηρούμε πως το dropout δεν βελτίωσε το μοντέλο μας στην περίπτωση του MR dataset.

- **Semeval dataset χωρίς dropout:**



Best epoch: 5  
Accuracy: 0.5990719635297949  
F1: 0.5722971038767537  
Recall: 0.5977653503880589

- **Semeval dataset με dropout:**



Best epoch: 14  
Accuracy: 0.6023282318463041  
F1: 0.5798631991790472  
Recall: 0.5970020934442773

Παρατηρήσαμε ότι στο Semeval dataset το validation loss έχει μια μεγάλη άνοδο το οποίο υποδηλώνει overfitting αφού το μοντέλο δεν αποδίδει στο testing τόσο καλά όσο στο train (train loss διαρκώς πέφτει). Με προσθήκη dropout καταφέραμε να πάρουμε κατά έναν μικρό βαθμό καλύτερο μοντέλο τόσο ως προς το loss όσο και ως προς το accuracy.

### **Ζητούμενο 11 – MR Dataset: Αναγνώριση Συναισθήματος με ChatGPT**

Στο πλαίσιο του Ζητήματος 11, πραγματοποιήθηκε πειραματισμός πάνω στην αναγνώριση συναισθήματος με τη χρήση του μοντέλου ChatGPT, χρησιμοποιώντας το MR (Movie Reviews) dataset. Το σύνολο δεδομένων περιλάμβανε κριτικές κινηματογράφου κατηγοριοποιημένες σε δύο κατηγορίες: θετικές (positive) και αρνητικές (negative).

#### **Διαδικασία**

- Επιλέχθηκαν 20 προτάσεις από κάθε κατηγορία (positive/negative), συνολικά 40 προτάσεις.
- Οι προτάσεις τυπώθηκαν με χρήση script και αποθηκεύτηκαν σε JSON αρχείο (samples\_MR.json).
- Οι προτάσεις υποβλήθηκαν σε ChatGPT με τέσσερις διαφορετικές προτροπές (prompts).
- Οι απαντήσεις του μοντέλου αξιολογήθηκαν ως προς την ακρίβειά τους με βάση τα πραγματικά labels.

#### **Prompts που Χρησιμοποιήθηκαν**

**Prompt 1:** Απλή ταξινόμηση χωρίς αιτιολόγηση, με απάντηση 1 λέξης ανά πρόταση.

Παρακάτω σου δίνω κινηματογραφικές κριτικές. Για κάθε μία, πες μου αν είναι "positive" ή "negative". Απάντησε με μία μόνο λέξη για κάθε πρόταση.

1. A visually stunning yet emotionally hollow film.
2. The performances were top-notch and the script witty.
3. ...

**Prompt 2:** Ταξινόμηση με σύντομη αιτιολόγηση για κάθε πρόταση.

Για κάθε μία από τις παρακάτω κινηματογραφικές κριτικές, πες μου:

1. Αν είναι "positive" ή "negative"
2. Μια σύντομη αιτιολόγηση για την απόφασή σου

Κριτικές:

1. ...
2. ...

**Prompt 3:** Ταξινόμηση με παροχή λέξεων-κλειδιών που επηρέασαν την απόφαση.

Θέλω να αναλύσεις τις παρακάτω κριτικές. Για κάθε μία:

1. Πες μου αν είναι "positive" ή "negative"
2. Εντόπισε τις πιο σημαντικές λέξεις ή φράσεις που επηρέασαν την απόφασή σου
3. Εξήγησε σύντομα γιατί κατέληξες σε αυτό το συμπέρασμα

**Prompt 4:** Ταξινόμηση σε μορφή πίνακα με sentiment, keywords, αιτιολόγηση.

Για κάθε μία από τις παρακάτω κριτικές, βάλε τα αποτελέσματα σε πίνακα με τις εξής στήλες:

| Αριθμός | Συναίσθημα (positive/negative) | Λέξεις-κλειδιά | Αιτιολόγηση |

### Αποτελέσματα Ακρίβειας

Prompt	Ακρίβεια (Accuracy)
Prompt 1 – Απλή απάντηση	45.0%
Prompt 2 – Με αιτιολόγηση	92.5%
Prompt 3 – Με keywords & αιτιολόγηση	90.0%
Prompt 4 – Πίνακας με keywords & εξήγηση	90.0%

### Ανάλυση Λαθών

Το Prompt 1 παρουσίασε τη χαμηλότερη ακρίβεια, καθώς δεν παρείχε στο ChatGPT καμία καθοδήγηση ή συμφραζόμενα πέρα από την πρόταση. Το μοντέλο δυσκολεύτηκε σε ειρωνικές, σύνθετες ή αμφίσημες κριτικές. Αντίθετα, τα Prompts 2–4 είχαν πολύ υψηλότερη απόδοση (>90%).

Αιτιολόγηση Απαντήσεων και Λέξεις-Κλειδιά

Τα Prompts 2–4 παρείχαν πολύτιμη πληροφόρηση για τον τρόπο λήψης απόφασης του μοντέλου. Οι λέξεις-κλειδιά που εντοπίστηκαν από το ChatGPT συχνά συσχετίζονταν με ισχυρό θετικό ή αρνητικό συναίσθημα. Παραδείγματα:

**Θετικές λέξεις-κλειδιά:** great, charming, stunning, intelligent, entertaining

**Αρνητικές λέξεις-κλειδιά:** weak, forgettable, pointless, boring, cliché

Επίσης, το μοντέλο παρείχε σχετικές αιτιολογήσεις για τις περισσότερες απαντήσεις, όπως π.χ. "η κριτική αναφέρεται σε αδύναμη πλοκή και μέτρια σκηνοθεσία".

### **Συμπεράσματα**

Η πειραματική διαδικασία έδειξε πως η απόδοση του ChatGPT στη ταξινόμηση συναισθήματος επηρεάζεται σημαντικά από το πόσο σαφής και επεξηγηματική είναι η προτροπή. Η παροχή συμπραζομένων, λέξεων-κλειδιών και αιτιολογήσεων βελτιώνει εμφανώς την ποιότητα της ταξινόμησης.

### **Ζητούμενο 11 – Semeval2017A: Αναγνώριση Συναισθήματος με ChatGPT**

Για το δεύτερο μέρος του Ζητήματος 11, εφαρμόσαμε τεχνικές αναγνώρισης συναισθήματος με το ChatGPT χρησιμοποιώντας το dataset Semeval 2017 Task 4A. Το συγκεκριμένο σύνολο δεδομένων περιλαμβάνει σύντομα tweets κατηγοριοποιημένα σε τρεις κατηγορίες: θετικό (positive), αρνητικό (negative) και ουδέτερο (neutral).

#### **Διαδικασία**

- Επιλέχθηκαν τουλάχιστον 20 tweets από κάθε κατηγορία, συνολικά 60.
- Οι προτάσεις εκτυπώθηκαν και αποθηκεύτηκαν σε JSON για αναφορά (samples\_Semeval2017A.json).
- Τα tweets αναλύθηκαν από το ChatGPT με χρήση τεσσάρων διαφορετικών prompts.
- Καταγράφηκαν οι προβλέψεις και υπολογίστηκε η ακρίβεια για κάθε προτροπή.

## **Prompts που Χρησιμοποιήθηκαν**

**Prompt 1:** Απλή ταξινόμηση, μία λέξη ανά tweet.

Παρακάτω σου δίνω tweets. Για κάθε ένα, πες μου αν είναι "positive", "negative", ή "neutral".

Απάντησε με μία μόνο λέξη για κάθε πρόταση.

1. Just watched the keynote! Impressive work by Apple.
2. This service keeps getting worse. I'm tired of it.
3. Well, the day went by faster than I expected.

...

**Prompt 2:** Ταξινόμηση και αιτιολόγηση για κάθε tweet.

Για κάθε ένα από τα παρακάτω tweets:

1. Πες μου το συναίσθημα ("positive", "negative", "neutral")
2. Πρόσφερε μια σύντομη αιτιολόγηση για την επιλογή σου

Tweets:

1. ...
2. ...

**Prompt 3:** Ταξινόμηση με λέξεις-κλειδιά και σύντομη εξήγηση.

Για κάθε ένα από τα tweets:

1. Πες μου αν είναι "positive", "negative", ή "neutral"
2. Γράψε 2-3 λέξεις-κλειδιά που επηρέασαν την απόφασή σου
3. Μια σύντομη εξήγηση γιατί κατέληξες σε αυτή την κατηγορία

**Prompt 4:** Πλήρης πίνακας με στήλες για συναίσθημα, λέξεις-κλειδιά και αιτιολόγηση.

Για κάθε tweet, συμπλήρωσε τα εξής σε πίνακα:

| Αρ. | Συναίσθημα (positive/negative/neutral) | Λέξεις-κλειδιά | Εξήγηση |

### Αποτελέσματα Ακρίβειας

Prompt	Ακρίβεια (Accuracy)
Prompt 1 – Μία λέξη	50.0%
Prompt 2 – Με αιτιολόγηση	33.3%
Prompt 3 – Με λέξεις-κλειδιά	58.3%
Prompt 4 – Πίνακας	70.0%

### Ανάλυση και Λάθη

Παρατηρήθηκε σημαντική διακύμανση στις επιδόσεις μεταξύ των διαφορετικών prompts. Το Prompt 2 είχε την πιο αδύναμη επίδοση πιθανώς λόγω ασαφών οδηγιών και ασυνεπούς ερμηνείας. Το Prompt 4 είχε την καλύτερη επίδοση, αποδεικνύοντας τη σημασία της δομημένης παρουσίασης πληροφοριών προς το μοντέλο.

### Σημαντικές Λέξεις και Ερμηνείες

Από τα αποτελέσματα του Prompt 3 και 4 προκύπτει ότι το ChatGPT βασίστηκε κυρίως σε λέξεις όπως: "love", "awesome", "great", "boring", "worst", "alone", "yay", "#fail", "unbelievable", κ.ά. Αρκετά ουδέτερα tweets ταξινομήθηκαν λανθασμένα λόγω διαφορετικών εκφράσεων ή ειρωνείας.

### Συμπεράσματα

Η ανάλυση του Semeval dataset ανέδειξε τη δυσκολία ταξινόμησης ουδέτερων προτάσεων από το μοντέλο, λόγω έλλειψης σαφούς συναισθηματικού περιεχομένου. Ωστόσο, διαπιστώθηκε ότι η δομή και πληρότητα της προτροπής βελτιώνει σημαντικά την ακρίβεια. Το ChatGPT παρείχε επαρκείς αιτιολογήσεις και λέξεις-κλειδιά όταν του ζητήθηκαν, γεγονός που ενισχύει την ερμηνευσιμότητα των αποτελεσμάτων.

# ΕΡΩΤΗΜΑΤΑ

## Ερώτημα 1

- 1.1) Υπολογίστε την αναπαράσταση κάθε πρότασης  $u$  (Εξίσωση 2) ως την συνένωση (concatenation) του μέσου όρου (mean pooling) και του μεγίστου ανά διάσταση (max pooling) των wordembeddings κάθε πρότασης,  $E = (e_1, e_2, \dots, e_N)$ .

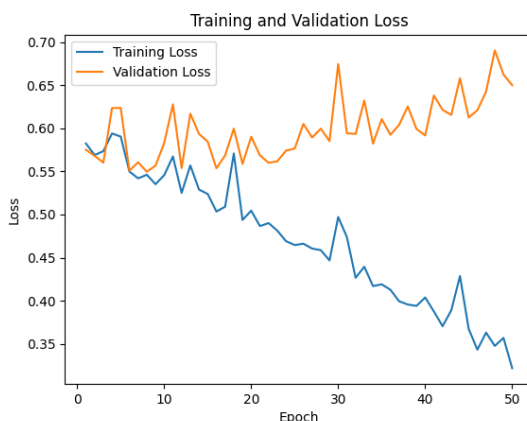
$$u = [\text{mean}(E) \parallel \text{max}(E)]$$

- 1.2) Τι διαφορά(ές) έχει αυτή η αναπαράσταση με την αρχική; Τι παραπάνω πληροφορία θα μπορούσε να εξάγει; Απαντήστε συνοπτικά.

Η mean pooling αναπαράσταση που είχαμε έως τώρα, έδινε μία γενική ιδέα για το νόημα της πρότασης η οποία στηριζόταν στην ισοβαρή σημασία κάθε λέξης εφόσον πέραναμε τον μέσο όρο. Πλέον, με την συνένωση mean-max pooling παίρνουμε για κάθε πρόταση ένα διάνυσμα διπλάσιων διαστάσεων το οποίο περιέχει πληροφορία και για τις πιο «σημαντικές» λέξεις της πρότασης. Με το max pooling, κρατάμε ανά διάσταση την μέγιστη τιμή από όλες τις λέξεις και έτσι προσθέτουμε στο μοντέλο την ικανότητα να δίνει βάρος στις λέξεις με μεγάλο συναισθηματικό – σημασιολογικό ενδιαφέρον.

Παρακάτω, παραθέτουμε τα αντίστοιχα διαγράμματα και τις μετρικές όπως σε προηγούμενα ερωτήματα, με χρήση **mean-max pooling**:

- **MR dataset χωρίς dropout και mean-max pooling:**



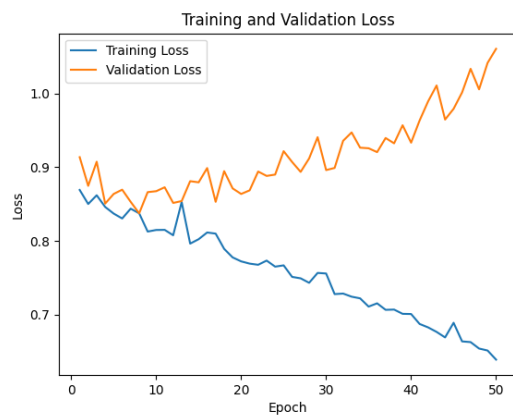
Best epoch: 10  
Accuracy: 0.6888217522658611  
F1: 0.6871771344680461  
Recall: 0.6928778541260711

- **MR dataset με dropout και mean-max pooling:**



Best epoch: 22  
 Accuracy: 0.6782477341389728  
 F1: 0.6725930225808324  
 Recall: 0.6914758020236882

- **Semeval dataset χωρίς dropout και mean-max pooling:**



Best epoch: 8  
 Accuracy: 0.6011071312276132  
 F1: 0.5807617160491193  
 Recall: 0.5952204843941504

- **Semeval dataset με dropout και mean-max pooling:**



Best epoch: 11  
 Accuracy: 0.6055845001628134  
 F1: 0.5749072818015479  
 Recall: 0.6079614969092957



Με βάση τα αποτελέσματα μας, το accuracy με χρήση mean-max pooling βελτιώθηκε (κατά μικρό ποσοστό) για το μοντέλο του dataset Semeval όμως όχι του MR dataset. Αξίζει να σημειωθεί επίσης, πως σε γενικές γραμμές, μοντέλα με dropout είναι κατά έναν μικρό βαθμό καλύτερα τόσο ως προς το loss όσο και ως προς το accuracy. Σε περιπτώσεις που το μοντέλο με dropout είναι ελαφρώς χειρότερο, μάλλον θα έπρεπε να χρησιμοποιηθούν περισσότερες εποχές ώστε να μπορέσει το μοντέλο να μάθει καλύτερα.

Για καλύτερο visualization των αποτελεσμάτων παραθέτουμε τα παραπάνω συμπεράσματα ως προς τις μετρικές σε πίνακες:

- **MR dataset:**

-	-	Accuracy	F1 score	Recall
mean pooling	not dropout	0.7039	0.7009	0.7124
	dropout	0.7085	0.7059	0.7158
mean-max pooling	not dropout	0.6888	0.6872	0.6929
	dropout	0.6782	0.6726	0.6915

- **Semeval dataset:**

-	-	Accuracy	F1 score	Recall
mean pooling	not dropout	0.5991	0.5723	0.5978
	dropout	0.6023	0.5799	0.5970
mean-max pooling	not dropout	0.6011	0.5808	0.5952
	dropout	0.6056	0.5749	0.6080

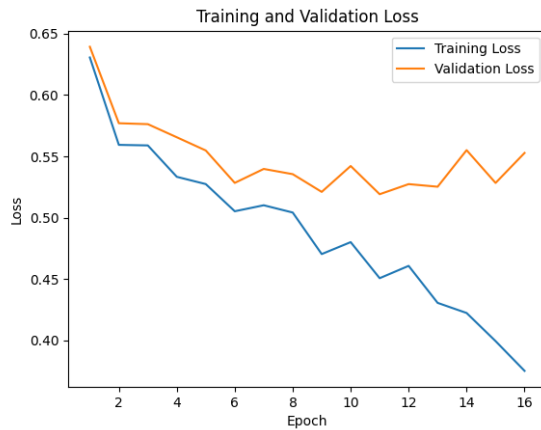
## Ερώτημα 2

Στην συνέχεια, θα πειραματιστούμε την χρήση αναδρομικών νευρωνικών δικτύων και συγκεκριμένα **LSTM**. Όπως μας ζητείται, κατά την εκπαίδευση, θα χρησιμοποιήσουμε **early stopping** μετά από 5 συνεχόμενες αυξήσεις του validation loss.

Στην συνέχεια θα δοκιμάσουμε **το αμφίδρομο LSTM** και θα συγκρίνουμε τις μετρικές μας για τα δύο αυτά πειράματα. Το Bidirectional LSTM αποτελεί μια επέκταση του κλασικού LSTM, η οποία επιτρέπει στο δίκτυο να επεξεργάζεται την είσοδο και προς τις δύο κατευθύνσεις, τόσο από αριστερά προς τα δεξιά όσο και από δεξιά προς τα αριστερά. Με αυτόν τον τρόπο, κάθε λέξη αποκτά μια αναπαράσταση που ενσωματώνει πληροφορία από ολόκληρο το συμφραζόμενο της πρότασης. Αυτή η προσέγγιση είναι ιδιαίτερα χρήσιμη σε γλωσσικά tasks όπως η ανάλυση συναισθήματος, όπου η σημασία μιας λέξης μπορεί να επηρεάζεται τόσο από προηγούμενες όσο και από επόμενες λέξεις.

Παρακάτω, παραθέτουμε τα διαγράμματα του validation/test loss για το LSTM με χρήση early stopping καθώς και τις μετρικές που εξάγαμε από το test set.

- **MR dataset LSTM:**



Early stopping triggered at epoch 16

✅ Final Evaluation (Test Set):

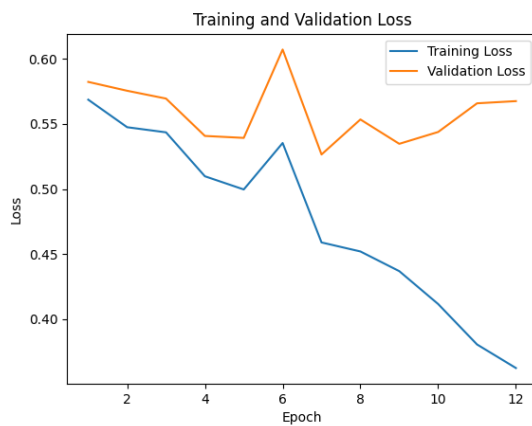
Test Loss: 0.5021

Accuracy: 0.7432

F1 Score (macro): 0.7432

Recall (macro): 0.7433

- **MR dataset LSTM µε bidirectional:**



Early stopping triggered at epoch 12

✅ Final Evaluation (Test Set):

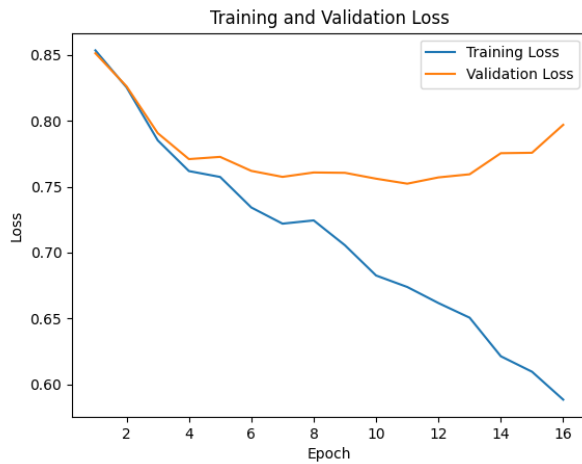
Test Loss: 0.5010

Accuracy: 0.7447

F1 Score (macro): 0.7438

Recall (macro): 0.7482

- **Semeval dataset LSTM:**



Early stopping triggered at epoch 16

✓ Final Evaluation (Test Set):

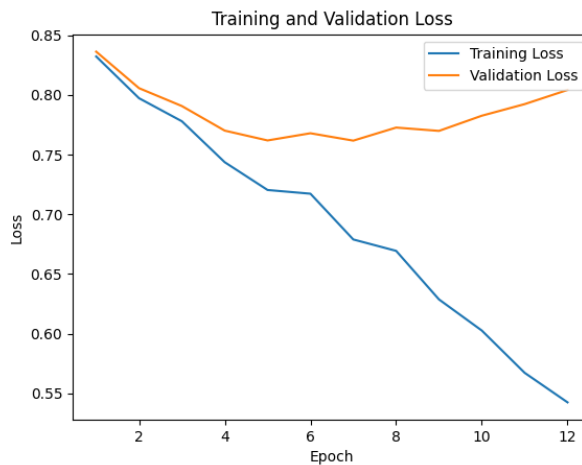
Test Loss: 0.8467

Accuracy: 0.6030

F1 Score (macro): 0.5907

Recall (macro): 0.6035

- **Semeval dataset LSTM  $\mu\epsilon$  bidirectional:**



Early stopping triggered at epoch 12

✓ Final Evaluation (Test Set):

Test Loss: 0.8108

Accuracy: 0.6217

F1 Score (macro): 0.6095

Recall (macro): 0.6139

- **MR dataset LSTM:**

	<b>Accuracy</b>	<b>F1 score</b>	<b>Recall</b>
<b>not bidirectional</b>	0.7432	0.7432	0.7433
<b>bidirectional</b>	0.7447	0.7438	0.7482

- **Semeval dataset LSTM:**

	<b>Accuracy</b>	<b>F1 score</b>	<b>Recall</b>
<b>not bidirectional</b>	0.6030	0.5907	0.6035
<b>bidirectional</b>	0.6217	0.6095	0.6139

Παρατηρούμε πως το bidirectional LSTM αυξάνει την ποιότητα του μοντέλου κατά ελάχιστο και στα δύο datasets σε σχέση με το απλό LSTM, το οποίο ήταν αναμενόμενο καθώς λέξεις επηρεάζονται και από επόμενες και από προηγούμενες λέξεις.

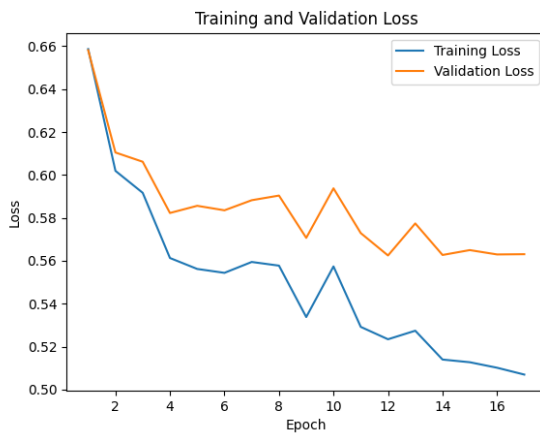
Επιπλέον, από τις καμπύλες φαίνεται ότι ο μηχανισμός attention μείωσε αισθητά το overfitting σε σχέση με προηγούμενα μοντέλα. (φθίνουσα καμπύλη)

Αξίζει να σημειωθεί, πως στην περίπτωση του MR dataset σημειώθηκε αύξηση στο accuracy σε σχέση και με το απλό DNN μοντέλο. Ωστόσο, στο Semeval πήραμε σχεδόν ίδιες (κατά μερικά εκατοστά χειρότερες) μετρικές σε σχέση με το DNN. Υπενθυμίζουμε όμως πως οι μετρικές που τυπώσαμε στο DNN αφορούν το validation ανά εποχή κρατώντας τις καλύτερες από αυτές (minimum loss) και συνεπώς η σύγκριση παραμένει εν μέρη άδικη για τις μετρικές του LSTM.

### Ερώτημα 3

Στο ερώτημα αυτό χρησιμοποιούμε attention μηχανισμό. Εφαρμόσαμε average pooling στις τελικές αναπαραστάσεις. Σε σχέση με το LSTM παρατηρήσαμε μεγάλη διαφορά στον χρόνο εκπαίδευσης ο οποίος μειώθηκε αισθητά.

- **MR dataset Attention:**



Early stopping triggered at epoch 17

Final Evaluation (Test Set):

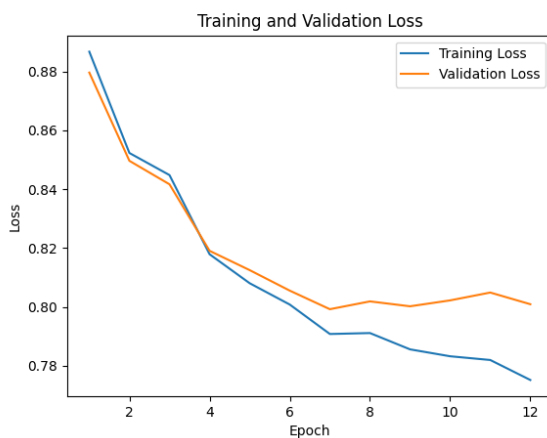
Test Loss: 0.5324

Accuracy: 0.7145

F1 Score (macro): 0.7145

Recall (macro): 0.7146

- **Semival dataset Attention:**



Early stopping triggered at epoch 12

Final Evaluation (Test Set):

Test Loss: 0.8467

Accuracy: 0.6055

F1 Score (macro): 0.5807

Recall (macro): 0.6082

**3.2) Τι είναι τα queries, keys και values που υπάρχουν στη κλάση `attention.Head` και τα `position_embeddings` που ορίζονται στην `attention.SimpleSelfAttentionModel`; Μπορείτε να συμβουλευτείτε το tutorial “Let’s build GPT: from scratch, in code, spelled out” για την απάντησή σας.**

Στο πλαίσιο του μηχανισμού self-attention, τα queries (Q), keys (K) και values (V) είναι διαφορετικές προβολές (γραμμικοί μετασχηματισμοί) των word embeddings μιας πρότασης. Πιο συγκεκριμένα:

- Keys: Κωδικοποιούν τα περιεχόμενα κάθε λέξης — "τι σημαίνει" κάθε λέξη.
- Queries: Εκφράζουν την πληροφορία που “αναζητά” κάθε λέξη — "σε τι δίνω προσοχή".
- Values: Είναι η πραγματική πληροφορία που θα συγκεντρωθεί, ζυγισμένη βάσει των attention scores.

Ουσιαστικά, κάθε λέξη συγκρίνει το query της με όλα τα keys (μέσω εσωτερικού γινομένου), ώστε να αποφασίσει σε ποιες λέξεις να δώσει σημασία. Στη συνέχεια, υπολογίζεται ένας σταθμισμένος μέσος όρος των αντίστοιχων values, δίνοντας έτσι το attention output.

Επιπλέον, τα position embeddings που υπάρχουν στην κλάση `SimpleSelfAttentionModel` προστίθενται στα word embeddings για να ενσωματωθεί πληροφορία θέσης (δηλαδή σειράς λέξεων) στο input, καθώς το self-attention από μόνο του είναι ανεξάρτητο από τη σειρά των tokens.

## Ερώτημα 4

Στο ερώτημα αυτό, εκπαιδεύουμε ένα multihead attention μηχανισμό και τυπώνουμε την επίδοση του μοντέλου. Συγκεκριμένα, επιλέγουμε 3 κεφαλές με ίσο μέγεθος η κάθε μία ( $\text{head\_size} = \text{dim} // \text{n\_head}$ ). Κάθε head, ουσιαστικά επεξεργάζεται διαφορετικό τμήμα των vectors (εφόσον έχουμε διαφορετικά keys, values, queries και άρα διαφορετικούς πίνακες βαρών για κάθε head) και άρα εξάγει πληροφορία διαφορετικής σημασιολογίας. (πχ άλλο head για ρήματα, άλλο για επιθετικούς προσδιορισμούς κλπ. )

- **MR dataset Multihead Attention:**



Early stopping triggered at epoch 17

Final Evaluation (Test Set):

Test Loss: 0.5243

Accuracy: 0.7221

F1 Score (macro): 0.7218

Recall (macro): 0.7227

- **Semival dataset Multihead Attention:**



Early stopping triggered at epoch 23

Final Evaluation (Test Set):

Test Loss: 0.8145

Accuracy: 0.6228

F1 Score (macro): 0.6038

Recall (macro): 0.6242



- **MR dataset Attention μηχανισμοί:**

	<b>Accuracy</b>	<b>F1 score</b>	<b>Recall</b>
<b>One head</b>	0.7145	0.7145	0.7146
<b>Multihead</b>	0.7221	0.7218	0.7227

- **Semeval dataset Attention μηχανισμοί:**

	<b>Accuracy</b>	<b>F1 score</b>	<b>Recall</b>
<b>One head</b>	0.6055	0.5807	0.6082
<b>Multihead</b>	0.6228	0.6038	0.6232

Όπως φαίνεται και από τους παραπάνω πίνακες, η εφαρμογή πολλαπλών κεφαλών βελτίωσε την επίδοση του μοντέλου μας. Αυτό είναι αναμενόμενο, καθώς ο μηχανισμός multi-head attention επιτρέπει στο μοντέλο να εστιάζει ταυτόχρονα σε διαφορετικές πτυχές της πρότασης μέσω πολλών "κεφαλών" προσοχής. Κάθε κεφαλή μπορεί να μαθαίνει διαφορετικά μοτίβα ή σχέσεις μεταξύ των λέξεων, ενισχύοντας έτσι την εκφραστικότητα της τελικής αναπαράστασης της πρότασης.

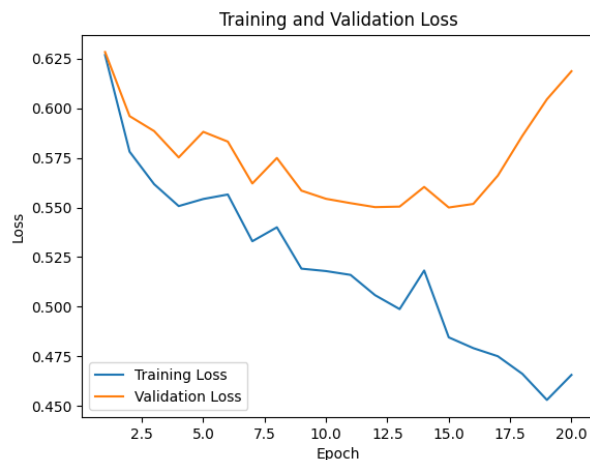
## Ερώτημα 5

### 5.1) Ποια είναι η διαφορά του Transformer σε σχέση με το MultiHead Attention Model;

Στο ερώτημα αυτό, θα πειραματιστούμε με τη χρήση του μοντέλου Transformer. Το μοντέλο αυτό βασίζεται αποκλειστικά στον μηχανισμό attention ωστόσο διαφοροποιείται ως προς την πολυπλοκότητα του. Ενώ το μοντέλο Multi Head Attention εφαρμόζει στην είσοδο μία φορά **attention** και έπειτα **feedforward** ( Linear -> ReLU -> Linear -> Dropout ) ενώ στον Transformer έχουμε **πολλαπλά επαναλαμβανόμενα blocks** (layers) από τέτοια επίπεδα. Έτσι, έχει τη δυνατότητα να κατανοεί πιο σύνθετες σχέσεις αφού το attention εφαρμόζεται αρκετές παραπάνω φορές.

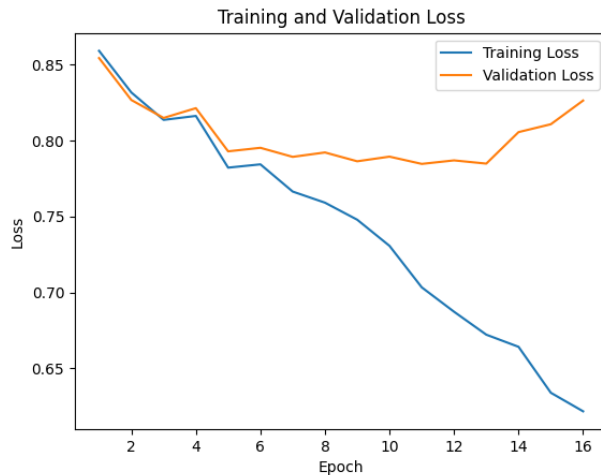
Αρχικά, φτιάξαμε έναν Transformer με **n\_heads = 3** και **n\_layer=3**. Παρακάτω παραθέτουμε τα αποτελέσματά μας:

- **MR dataset Transformer:**



Final Evaluation (Test Set):  
Test Loss: 0.5377  
Accuracy: 0.7372  
F1 Score (macro): 0.7372  
Recall (macro): 0.7372

- **Semeval dataset Transformer:**



Early stopping triggered at epoch 16

Final Evaluation (Test Set):

Test Loss: 0.8159

Accuracy: 0.6232

F1 Score (macro): 0.6050

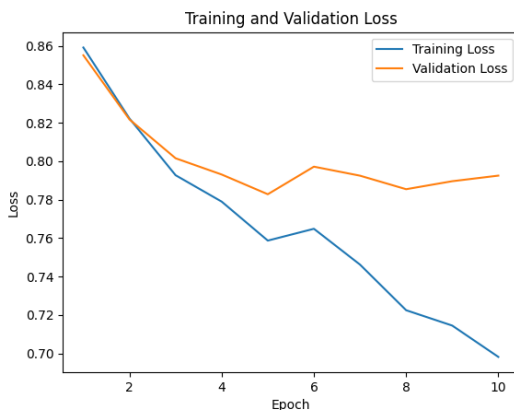
Recall (macro): 0.6204

Συγκριτικά με το MultiHead μοντέλο, το Transformer model αύξησε κατά μικρό ποσοστό την επίδοση και στα δύο data sets. **Συγκεντρωτικές συγκρίσεις θα παραθέσουμε σε πίνακες παρακάτω.**

## 5.2) Πειραματιστείτε με διαφορετικές τιμές των παραμέτρων;

Διαλέξαμε το **Semeval** data set για πειραματισμό με τις παραμέτρους:

- **n\_heads=6, n\_layer = 3**



Early stopping triggered at epoch 10

Final Evaluation (Test Set):

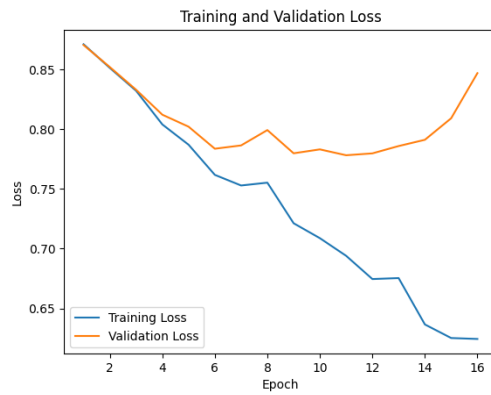
Test Loss: 0.8302

Accuracy: 0.6221

F1 Score (macro): 0.5980

Recall (macro): 0.6278

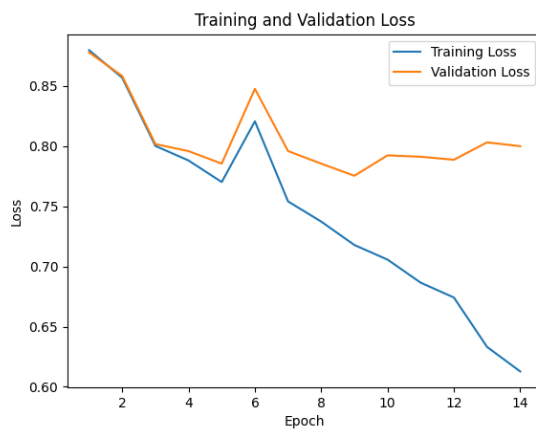
- **n\_heads=8, n\_layer = 4**



Early stopping triggered at epoch 16

Final Evaluation (Test Set):  
 Test Loss: 0.8234  
 Accuracy: 0.6233  
 F1 Score (macro): 0.6144  
 Recall (macro): 0.6112

- **n\_heads=8, n\_layer = 6 (default)**



Early stopping triggered at epoch 14

Final Evaluation (Test Set):  
 Test Loss: 0.8142  
 Accuracy: 0.6279  
 F1 Score (macro): 0.5973  
 Recall (macro): 0.6439

### 5.3) Ποιες είναι οι default τιμές τους στην κλασική αρχιτεκτονική του Transformer;

Με βάση το paper “Attention is ALL You Need” οι default παράμετροι είναι:

**n\_head = 8**

**n\_layer = 6**

**embedding\_dim = 512**

**feedforward\_dim = 2048 (4 x embedding\_dim)**

**dropout = 0.1**

Όπως θα δείτε παραπάνω, δοκιμάσαμε κι εμείς n\_head=8, n\_layer=6 αλλά τα αποτελέσματα προφανώς δεν είναι συγκρίσιμα με το paper εφόσον έχουμε διαφορετικές διαστάσεις embeddings.

#### *Πίνακες Σύγκρισης Υπερπαραμέτρων (Semeval dataset):*

	Accuracy	F1 score	Recall
n_head=3 n_layer=3	0.6232	0.6050	0.6204
n_head=6 n_layer=3	0.6221	0.5980	0.6278
n_head=8 n_layer=4	0.6233	0.6144	0.6112
n_head=8 n_layer=6	0.6279	0.5973	0.6439

*Σημειώνουμε πως όσο αυξάναμε τις υπερπαραμέτρους, τόσο αυξήθηκε και ο χρόνος εκπαίδευσης.*

Στην περίπτωση όπου αυξήσαμε το πλήθος των κεφαλών από 3 σε 6 πήραμε ελαφρώς χειρότερα αποτελέσματα. Ενδεχομένως αυτό να σχετίζεται με το overfitting που προκλήθηκε. Το τρίτο πείραμα είναι ελάχιστα καλύτερο από το πρώτο ενώ οι default τιμές

**Πίνακες ολικής σύγκρισης Attention – MultiHead Attention - Transformer για κάθε dataset:**

- **MR dataset:**

	Accuracy	F1 score	Recall
One head	0.7145	0.7145	0.7146
Multihead	0.7221	0.7218	0.7227
Transformer	0.7372	0.7372	0.7372

- **Semeval dataset (keeping best parameters performance):**

	Accuracy	F1 score	Recall
One head	0.6055	0.5807	0.6082
Multihead	0.6228	0.6038	0.6232
Transformer	0.6279	0.5973	0.6439

Και στα δύο dataset, το **Transformer model** δείχνει να είναι ελαφρώς **καλύτερο** συγκριτικά με τους απλούς μηχανισμούς attention (κρίνοντας βάση accuracy).

## Ερώτημα 6

Σε αυτό το ερώτημα θα χρησιμοποιήσετε Pre-Trained Transformer μοντέλα για την κατηγοριοποίηση συναισθήματος. Επιλέξτε τουλάχιστον 3 μοντέλα για κάθε dataset και επεκτείνετε κατάλληλα τον κώδικα. Συγκρίνετε τις επιδόσεις τους και αναφέρετε σε πίνακες τα αποτελέσματα.

Όπως ζητήθηκε, διαλέξαμε 3 μοντέλα από το **Hugging Face Model Hub** (<https://huggingface.co/models>) για κάθε dataset. Προσέξαμε να είναι εκπαιδευμένα για sentiment analysis καθώς και το πλήθος των ετικετών τους να ταιριάζει σε κάθε data set.

- Για το **MR** dataset διαλέξαμε τα παρακάτω, τα οποία παραθέτουμε μαζί με τις μετρικές που πήραμε κατά το testing:

### Μοντέλο 1: [siebert/sentiment-roberta-large-english](#)

```
Dataset: MR
Pre-Trained model: siebert/sentiment-roberta-large-english
Test set evaluation
  accuracy: 0.9259818731117825
  recall: 0.9259818731117825
  f1-score: 0.9259803530069484
```

### Μοντέλο 2: [distilbert-base-uncased-finetuned-sst-2-english](#)

```
Dataset: MR
Pre-Trained model: distilbert-base-uncased-finetuned-sst-2-english
Test set evaluation
  accuracy: 0.8912386706948641
  recall: 0.8912386706948641
  f1-score: 0.891213847502191
```

### Μοντέλο 3: [textattack/bert-base-uncased-SST-2](#)

```
Dataset: MR
Pre-Trained model: textattack/bert-base-uncased-SST-2
Test set evaluation
  accuracy: 0.8987915407854985
  recall: 0.8987915407854985
  f1-score: 0.898739552393846
```

	Accuracy	F1 score	Recall
<b>siebert</b>	0.926	0.926	0.926
<b>distilbert</b>	0.891	0.891	0.891
<b>textattack</b>	0.899	0.899	0.899

Βάσει accuracy, το μοντέλο **siebert** κρίνεται το καλύτερο για το MR dataset με πολύ μεγάλο ποσοστό accuracy!

Τα μοντέλα **siebert**, **distilbert** και **textattack** είναι προεκπαιδευμένα για ανάλυση συναισθήματος πάνω σε διαφορετικά σύνολα δεδομένων. Το πρώτο έχει εκπαιδευτεί σε ποικιλία πηγών (Amazon, Yelp, **IMDB**, κ.ά.) για γενική χρήση, ενώ τα άλλα δύο είναι εκπαιδευμένα ειδικά στο SST-2 (Stanford Sentiment Treebank), το οποίο **περιλαμβάνει κριτικές ταινιών** με ετικέτες θετικό/αρνητικό. Όλα είναι σχεδιασμένα για **ταξινόμηση 2 κατηγοριών**.

Το γεγονός ότι είναι εκπαιδευμένα ειδικά για ταινίες δικαιολογεί την υψηλή επίδοση και των τριών.



- Για το **Semeval** dataset διαλέξαμε τα παρακάτω, τα οποία παραθέτουμε μαζί με τις μετρικές που πήραμε κατά το testing:

### Μοντέλο 1: [cardiffnlp/twitter-roberta-base-sentiment](#)

```
Dataset: Semeval2017A
Pre-Trained model: cardiffnlp/twitter-roberta-base-sentiment
Test set evaluation
accuracy: 0.7237870400521003
recall: 0.7229454214750545
f1-score: 0.7222115953560642
PS C:\git_repo_nlp\slp-labs\lab3> █
```

### Μοντέλο 2: [finiteautomata/bertweet-base-sentiment-analysis](#)

```
Dataset: Semeval2017A
Pre-Trained model: finiteautomata/bertweet-base-sentiment-analysis
Test set evaluation
accuracy: 0.7177629436665581
recall: 0.7301871228078923
f1-score: 0.718050644575488 █
```

### Μοντέλο 3: [yiyanghkust/finbert-tone](#)

```
Dataset: Semeval2017A
Pre-Trained model: yiyanghkust/finbert-tone
Test set evaluation
accuracy: 0.5097688049495278
recall: 0.3866509029782952
f1-score: 0.3329611976374523 █
```

	Accuracy	F1 score	Recall
<b>cardiffnlp</b>	0.724	0.723	0.722
<b>finiteautomata</b>	0.718	0.730	0.718
<b>yiyanghkust</b>	0.510	0.387	0.333

Βάσει accuracy, το μοντέλο **cardiffnlp** κρίνεται το καλύτερο για το Semeval dataset. Από την άλλη, το yiyannghkust έχει πολύ μικρό accuracy κάτι που περιμέναμε εφόσον είναι εκπαιδευμένο σε οικονομικά κείμενα («financial communication text») και η αντίθεση με τα tweets είναι προφανής.

## Ερώτημα 7

**Σε αυτό το ερώτημα θα εκπαιδεύσετε/κουρδίσετε (fine-tune) Pre-Trained Transformer μοντέλα για την κατηγοριοποίηση συναισθήματος. Αξιοποιήστε τον κώδικα που θα βρείτε στο αρχείο finetune\_pretrained.py και εκτελέστε τοπικά με περιορισμένο dataset για λίγες εποχές (όπως δίνεται). Μεταφέρετε τον κώδικα σε notebook στο Google Colab8, μετατρέψτε τον κατάλληλα και προχωρήστε σε fine-tuning με σκοπό τη βέλτιστη επίδοση. Δοκιμάστε τουλάχιστον 3 μοντέλα για κάθε dataset και αναφέρετε τα αποτελέσματα σε πίνακες.**

Στο ερώτημα αυτό, θα πάρουμε τα μοντέλα που ήδη χρησιμοποιήσαμε (για συγκρίσιμα αποτελέσματα) και θα κάνουμε πάνω τους fine tuning. Το fine tuning είναι η διαδικασία όπου παίρνουμε μοντέλα ήδη εκπαιδευμένα σε τεράστια σύνολα δεδομένων και τα εκπαιδεύουμε επιπλέον πάνω στο δικό μας data set. Έτσι, μοντέλα που έχουν μάθει αρκετά καλά να γενικεύουν και «γνωρίζουν» πολλές λέξεις, αποκτούν μία εξειδίκευση στο δικό μας data set ώστε να κάνουν την δική μας εργασία.

Αρχικά στο αρχείο finetune\_pretrained θα συμπληρώσουμε τον κώδικα που πρέπει εκπαιδεύοντας για λίγες εποχές και μετά θα περάσουμε σε Collab για χρήση GPU με περισσότερες εποχές.

Τρέξαμε τοπικά για το **MR dataset** 5 εποχές και καναμε fine tune το μοντέλο **textattack** . Έπειτα κάναμε και τελικό testing στα test δεδομένα για να έχουμε καλύτερη σύγκριση. Πήραμε accuracy 87.61%.

```
Final test accuracy: 0.8761  
PS C:\git_repo_nlp\slp-labs\lab3> █
```

**Ο χρόνος εκπαίδευσης ήταν αρκετά μεγάλος. Προσπαθήσαμε να τρέξουμε σε GPU τοπικά αλλά πήραμε error για χώρο στη VRAM. Συνεχίζουμε με τον κώδικα στο Collab, ο οποίος θα δίνεται και στα αρχεία ως “fine\_tuning.ipynb” εκτελεσμένο ώστε να μπορείτε να δείτε τα αποτελέσματα. Στο collab, αλλάξαμε το n\_samples=40 σε 100 και κάνουμε train για 5 εποχές αφού με ολόκληρο το dataset παίρναμε run time error.**

Παραθέτουμε την επίδοση κάθε μοντέλου στο testing (μετά την εκπαίδευση) σε πίνακες για τα 2 datasets:

- **MR dataset:**

<b>Model</b>	<b>Accuracy</b>
<b>siebert</b>	0.926
<b>Fine tuned siebert</b>	0.863
<b>distilbert</b>	0.891
<b>Fine tuned distilbert</b>	0.896
<b>textattack</b>	0.899
<b>Fine tuned textattack</b>	0.884

- Semeval dataset:

Model	Accuracy
cardiffnlp	0.724
<b>Fine tuned cardiffnlp</b>	0.690
finiteautomata	0.718
<b>Fine tuned finiteautomata</b>	0.708
yiyanghkust	0.510
<b>Fine tuned yiyanghkust</b>	0.414

Είναι προφανές ότι η μείωση της απόδοσης κάποιων μοντέλων οφείλεται στο μικρό μέρος των data sets που αναγκαστήκαμε να χρησιμοποιήσουμε λόγω έλλειψης πόρων. Το μοντέλο σε αυτήν την περίπτωση κάνει overfit στα λίγα δείγματα που του παρείχαμε.

Στην συνέχεια, προσπαθήσαμε να αυξήσουμε επιπλέον τον αριθμό **n\_samples** και τον αριθμό των **εποχών από 5 σε 15** και να εκπαιδεύσουμε τα μοντέλα εκ νέου με σκοπό να πάρουμε καλύτερη επίδοση. Ωστόσο λόγω έλλειψης πόρων, δεν καταφέραμε να τρέξουμε το training στο Collab. Ωστόσο, ύστερα από αρκετές προσπάθειες, εκπαιδεύσαμε το μικρό μοντέλο yiyanghkust σε **περισσότερα samples με 10 εποχές** και πήραμε accuracy 0.492 (8% πάνω από το προηγούμενο fine tuned μοντέλο)

Μοιάζει λογικό, πως όσο αυξάνουμε τον αριθμό εποχών και των δειγμάτων, γενικά το fine tuning εκπαιδεύεται ολοένα και καλύτερα στα δικά μας datasets και παίρνουμε καλύτερο αποτέλεσμα.

## Ερώτημα 8

Ο κώδικας για το tutorial “Let’s build GPT: from scratch, in code, spelled out”, που προαναφέρθηκε, είναι διαθέσιμος εδώ [https://colab.research.google.com/drive/1JMLa53HDuA-i7ZBmqV7ZnA3c\\_fvtXnx-?usp=sharing](https://colab.research.google.com/drive/1JMLa53HDuA-i7ZBmqV7ZnA3c_fvtXnx-?usp=sharing).

Συνδεθείτε στο ChatGPT10 και ζητήστε:

- να σας εξηγήσει τον κώδικα σε λεπτομέρεια
- να αξιολογήσει τον κώδικα
- να ξαναγράψει κάποια τμήματά του (refactoring)

Αναφέρετε παραδείγματα από το διάλογο που κάνατε και σχολιάστε την απόδοση του στα παραπάνω.

Δώσαμε τον κώδικα στο chatgpt, κυρίως τμήματα που χρειάζοταν περαιτέρω εξήγηση. Παρακάτω παραθέτουμε μερικά μέρη του διαλόγου μας:

### Παράδειγμα 1: Επεξήγηση του Bigram Language Model

**Χρήστης:**

Μπορείς να μου εξηγήσεις τι κάνει η BigramLanguageModel;

**ChatGPT:**

Ναι! Η BigramLanguageModel δημιουργεί προβλέψεις για τον επόμενο χαρακτήρα βασισμένη μόνο στον τρέχοντα χαρακτήρα. Έχει ένα embedding table διαστάσεων (vocab\_size, vocab\_size) και εκπαιδεύεται ώστε να μάθει πιθανότητες μετάβασης από έναν χαρακτήρα στον επόμενο.

## Παράδειγμα 2: Refactoring

### Χρήστης:

Το `generate()` φαίνεται πολύ επαναλαμβανόμενο. Μπορείς να το κάνεις πιο καθαρό;

### ChatGPT:

Σίγουρα! Μπορούμε να εξάγουμε το κομμάτι:

```
python Αντιγραφή

logits = self(idx_cond)
logits = logits[:, -1, :]
probs = F.softmax(logits, dim=-1)
```

## Παράδειγμα 3: Αξιολόγηση του Κώδικα

### Χρήστης:

Πώς σου φαίνεται αυτός ο κώδικας; Θα τον θεωρούσες καθαρό;

### ChatGPT:

Ο κώδικας είναι εξαιρετικά καθαρός και διδακτικός. Χρησιμοποιεί καλές πρακτικές όπως modular κλάσεις, dropout, layer norm, και είναι ιδανικός για εκπαιδευτικούς σκοπούς. Ωστόσο, για παραγωγή θα μπορούσε να γίνει refactor σε ξεχωριστά αρχεία/modules.

## Παράδειγμα 4: Εξήγηση του Μέρους για την Εκπαίδευση του Μοντέλου

### Χρήστης:

Μπορείς να μου εξηγήσεις πώς λειτουργεί η συνάρτηση `generate()` που δημιουργεί νέο κείμενο; Ειδικά αυτό το τμήμα: (δόθηκε κώδικας)

### ChatGPT:

```
def generate(self, idx, max_new_tokens):
    for _ in range(max_new_tokens):
        logits = self(idx)
        logits = logits[:, -1, :] # get the logits for the last token
        probs = F.softmax(logits, dim=-1)
        idx_next = torch.multinomial(probs, 1) # sample from the probabilities
        idx = torch.cat((idx, idx_next), dim=1) # append the predicted token
    return idx
```

#### 1. Επανάληψη για Κάθε Νέο Token:

`for _ in range(max_new_tokens):` – Η συνάρτηση θα δημιουργήσει μέχρι `max_new_tokens` νέα tokens, όπου κάθε νέο token εξαρτάται από τα προηγούμενα.

#### 2. Πρόβλεψη του Επόμενου Token:

`logits = self(idx)` – Χρησιμοποιώντας το τρέχον `idx` (εισαγωγή που περιλαμβάνει ήδη τα προηγούμενα tokens), το μοντέλο προβλέπει τα logits για το επόμενο token.

#### 3. Επιλογή του Τελευταίου Token:

`logits = logits[:, -1, :]` – Εδώ επιλέγουμε μόνο το τελευταίο token από την έξοδο, γιατί το μοντέλο κάνει προβλέψεις για κάθε token ξεχωριστά, και εμείς θέλουμε μόνο την πρόβλεψη για το επόμενο.

#### 4. Υπολογισμός Πιθανοτήτων:

`probs = F.softmax(logits, dim=-1)` – Με τη συνάρτηση `softmax`, τα logits μετατρέπονται σε πιθανότητες για το επόμενο token. Κάθε πιθανότητα αντιπροσωπεύει την πιθανότητα του κάθε χαρακτήρα να είναι το επόμενο στο κείμενο.

#### 5. Δειγματοληψία από τις Πιθανότητες:

`idx_next = torch.multinomial(probs, 1)` – Αντί να επιλέγουμε πάντα την πιο πιθανή πρόβλεψη, κάνουμε δειγματοληψία (sampling) από την κατανομή πιθανοτήτων για να προσδώσουμε τυχαία ποικιλία στο παραγόμενο κείμενο.

#### 6. Ενημέρωση Εισόδου:

`idx = torch.cat((idx, idx_next), dim=1)` – Ενημερώνουμε το `idx` με το νέο προβλεπόμενο token για να το χρησιμοποιήσουμε στην επόμενη επανάληψη και να παράγουμε το επόμενο token.

## Παράδειγμα 5: Εξήγηση του Μέρους για την Εκπαίδευση του Μοντέλου

### Χρήστης:

Μπορείς να μου εξηγήσεις τι ακριβώς κάνει αυτό το τμήμα του κώδικα, το οποίο αφορά την εκπαίδευση του μοντέλου;

### ChatGPT:

```
python Αντιγραφή  
  
for iter in range(max_iters):  
    # forward pass  
    logits, loss = model(x_batch, y_batch)  
  
    # backward pass  
    loss.backward()  
  
    # update weights  
    optimizer.step()  
    optimizer.zero_grad()  
  
    # log progress  
    if iter % log_interval == 0:  
        print(f"iter {iter}/{max_iters}, loss: {loss.item()}")
```

Αυτό το κομμάτι είναι υπεύθυνο για την **εκπαίδευση** του μοντέλου σε κάθε επανάληψη. Αναλύοντας βήμα-βήμα:

#### 1. Προβλέψεις (Forward Pass):

`logits, loss = model(x_batch, y_batch)` – Η είσοδος είναι το `x_batch` (τα δεδομένα εισόδου) και το `y_batch` (οι ετικέτες). Το μοντέλο κάνει προβλέψεις για την επόμενη τιμή και υπολογίζει το `loss` βάσει της διαφοράς με το πραγματικό αποτέλεσμα.

#### 2. Οπισθοδιάδοση (Backward Pass):

`loss.backward()` – Υπολογίζεται το **gradient** του `loss` σε σχέση με τα βάρη του μοντέλου χρησιμοποιώντας την αλγορίθμο οπισθοδιάδοσης (backpropagation).

#### 3. Ενημέρωση Βαρών:

`optimizer.step()` – Ο `optimizer` ενημερώνει τα βάρη του μοντέλου χρησιμοποιώντας τα `gradients` που υπολογίστηκαν στην προηγούμενη φάση. Αυτός ο βήμας προσπαθεί να ελαχιστοποιήσει το `loss`.

#### 4. Μηδενισμός των Gradients:

`optimizer.zero_grad()` – Επαναφέρει τα `gradients` σε 0 για να μην επηρεάσουν τις επόμενες επαναλήψεις.

#### 5. Καταγραφή Προόδου:

`print(f"iter {iter}/{max_iters}, loss: {loss.item()}")` – Κάθε `log_interval` εμφανίζεται η πρόοδος της εκπαίδευσης και το τρέχον `loss` για να παρακολουθούμε την απόδοση του μοντέλου.



Η χρήση του ChatGPT για **ανάλυση, αξιολόγηση και refactoring** του tutorial ήταν ιδιαίτερα αποδοτική. Παρείχε:

- Κατανοητές εξηγήσεις για κάθε στάδιο.
- Υποδείξεις για καθαρότερο κώδικα.
- Ανάλυση της απόδοσης του transformer και του training loop.

## **ΠΑΡΑΔΟΤΕΟ:**

Στον φάκελο του παραδοτέου περιέχονται τα κύρια **αρχεία τα οποία τροποποιήθηκαν**, η **αναφορά**, ό,τι **αρχεία χρησιμοποιήθηκαν στο ζητούμενο 11 προπαρασκευής** καθώς και το **notebook** του ερωτήματος 7. Επιπλέον, τα μοντέλα μας στα περισσότερα ερωτήματα τα αποθηκεύαμε οπότε παραθέτουμε και τον φάκελο με τα **“saved\_models”**.