Bidirectional stacked RNNs with LSTM/GRU cells Project 3

Γεώργιος Κωνσταντίνος Ζαχαρόπουλος sdi1900061

Ιανουάριος 27, 2023

0.1 How to test the model

- Αρχικά βάζετε το δικό σας test dataset στο block του notebook Load Data Set στην μεταβλητή test bf.
- Κάνετε upload το αρχείο imdb-reviews.pt
- Κάνετε upload το αρχείο myNN.pt
- Τώρα το notebook είναι έτοιμο να τρέξει πατώντας την επιλογή ¨εκτέλεση όλων¨.

  Άμα το τρέξετε έτσι θα κάνει και train και test. Το οποίο δεν παίρνει πολύ ώρα καθώς το κάνω train για 8 εποχές (περίπου 1,2 λεπτά ανά εποχή) και το pre processing θέλει περίπου 1,5 λεπτά.

  Αν όμως δεν θέλετε να γίνει train και απλά θέλετε να το κάνετε test για το δικό σας αρχείο τότε μπορείτε να πάτε στο block Train and Test που βρίσκεται κάτω απο το Final model block και να κάνετε comment την μοναδική εντολή. Προσοχή άμα το τρέξετε έτσι το notebook θα χαλάσουν οι γραφικές, οπότε θα ήταν καλό να γίνουν comment από πριν.
- Συνοψίζοντας, όποια από τις δύο πρακτικές και να διαλέξετε για να τρέξετε το notebook πρέπει να πατήσετε εκτέλεση όλων των κελιών καθώς έτσι φορτώνονται τα imports και άλλες χρήσιμες συναρτήσεις και μεταβλητές.

0.2 Introduction

Στόχος της εργασίας είναι η υλοποίηση ενός sentiment classifier για ένα data set από imdb movie reviews με την χρήση Bidirectional stacked RNNs με LSTM ή GRU cells.

0.3 Download GloVe Embeddings

Κατεβάζουμε τα GloVe Embeddings τα οποία θα χρησιμοποιηθούν ως είσοδος στο δικό μας μοντέλο.
Μετατρέπουμε το GloVe σε Word2Vec για να έχουμε πιο έυκολη πρόσβαση στα vectors των λέξεων.

0.4 Imports

Κάνουμε import ότι είναι απαραίτητο από τις βιβλιοθήκες: torch, matplotlib, pandas, numpy, re, sklearn, gensim, nltk, torchmetrics

0.5 Set seed

Χρησιμοποιώ την συνάρτηση που δόθηκε από το φροντηστήριο για να έχω σταθερό seed στο notebook ώστε να παίρνω τα ίδια αποτελέσματα στις διάφορες εκτελέσεις.

0.6 Loading Data Set

Διαβάζουμε το data set και δημιουργούμε μια στήλη ¨αποτελέσματος' στην οποία βάζουμε 0 αν η κριτική είναι αρνητική και 1 αν είναι θετική. Αυτές θα είναι και οι κλάσεις μας.

0.7 Data Pre-processing

Χρειάζεται να καθαρίσουμε το data set καθώς δεν εκφράζουν όλες οι λέξεις sentiment και αυτό μπορεί να μπερδέψει στο training του μοντέλου.
Για παράδειγμα στα reviews υπάρχουν html tags, αριθμοί που εκφράζουν σκορ, special characters και stop words που δεν προσφέρουν στην εκμάθηση του μοντέλου. Όλα αυτά θα τα αφαιρέσουμε από το data set. Επίσης εφαρμόζουμε lemmatization στις λέξεις του data set. Ετσι μετατρέπουμε τις λέξεις στο αρχικό τους λήμμα, το οποίο θα αυξήσει τις πιθανότητες να βρούμε τα αντίστοιχα vectors στα embeddings. Τέλος αποφεύγουμε την χρήση stemming καθώς μπορεί να ¨χαλάσουν' οι λέξεις και να μην υπάρχει αντιστοιχία με τα embeddings.

0.8 Create the Set of Tensors for the Reviews

Αρχικά μετατρέπουμε το word2vec σε ένα dictionary το οποίο θα μας βοηθήσει να έχουμε γρήγορη αναζήτηση των vectors των λέξων. Επειτα διατρέχουμε τις λέξεις του κάθε review και κρατάμε μόνο τις ξεχωριστές λέξεις σε ένα άλλο dictionary. Ετσι κάθε λέξη θα έχει την ίδια επίδραση στον τελικό tensor του review. Οπότε όταν έχουμε τις ξεχωριστές λέξεις του review βρίσκουμε τους αντίστοιχους tensors και τους προσθέτουμε. Στο τέλος για κάθε review διαιρούμε τον tensor με το πλήθος των ξεχωριστών λέξων του review. Προσθέτουμε τους τελικούς tensors των reviews σε μια λίστα και στο τέλος τους κάνουμε stack.

Σημείωση: Δοκίμασα να χρησιμοποιήσω τον tfidf στα reviews για να βρίσκω τα vectors μόνο των k σημαντικών λέξεων, αλλά δεν είδα βελτίωση στην απόδοση, οπότε δεν το συμπεριέλαβα.

0.9 Split Train and Test Sets

Μετά χωρίζουμε το x και το y σε δύο σύνολα, train και test. Έτσι θα έχουμε ένα άγνωστο για το μοντέλο test set πάνω στο οποίο θα το βαθμολογήσουμε και ένα train set με το οποίο θα εκπαιδεύσουμε το μοντέλο μας. Ετσι ελέγχουμε την απόδοση του και αν κάνει overfit ή undefit.

0.10 Define RNN class

Τα χαρακτηριστικά του αρχικού μοντέλου είναι τα εξής:

- Χρησιμοποιούνται τα Embeddings των 50 διαστάσεων ως (input size), ενώ το output size είνα 1 το οποίο και δεν θα αλλάξει.
- Το hidden size είναι 64.
- Χρησιμοποιούνται 2 recurrent layers, που είναι και το ελάχιστο για να έχουμε stacked RNN.
- Χωρίς dropout μεταξύ των layers.
- Είναι bidirectional όπως ζητάει η εκφώνωση.
- Χρησιμοποιούνται για το training 10 εποχές.
- Και έχει ένα linear layer με sigmoid.
- Το batch value είναι 64
- Το learning rate θα ξεκινήσει από $1e-3$
- Και τέλος δεν υπάρχει clipping.

0.10.1 LSTM - GRU cells and HyperParameters Tuning

Η βασική ιδέα που ακολούθησα για να κάνω tune τις υπερπαραμέτρους είναι να εξετάζω διαφορετικές τιμές για κάθε μια υπερπαράμετρο με την χρήση LSTM και μετά με GRU cells. Δηλαδή θα κάνω tune 2 μοντέλα ταυτόχρονα που το ένα θα έχει γίνει tune με LSTM cells και το άλλο με GRU cells. Οπότε στο τέλος θα τα συγκρίνω και θα επιλέξω το καλύτερο.

0.10.2 Hidden size tuning

LSTM Model

$$HiddenSize = 64$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.47347	0.77510	0.77716	0.77221	0.77112
Test Set	0.48909	0.76664	0.78234	0.74582	0.76175

$$HiddenSize = 128$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.46832	0.77723	0.77869	0.77572	0.77337
Test Set	0.49814	0.76134	0.76222	0.76567	0.76112

$$HiddenSize = 256$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.46539	0.77938	0.78114	0.77828	0.77587
Test Set	0.50108	0.75408	0.74745	0.77486	0.75839

Παρατηρούμε πως με hidden size = 64 έχουμε στο test set το καλύτερο f1 score = 0.76175

GRU Model

$$HiddenSize = 64$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.47987	0.76937	0.77115	0.76758	0.76614
Test Set	0.50136	0.75920	0.74754	0.78125	0.76141

$$HiddenSize = 128$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.47951	0.77142	0.77399	0.76976	0.76837
Test Set	0.50562	0.75942	0.73877	0.80195	0.76625

$$HiddenSize = 256$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.46700	0.77893	0.78069	0.77880	0.77591
Test Set	0.51766	0.74644	0.76187	0.72015	0.73659

Παρατηρούμε πως με hidden size = 128 έχουμε στο test set το καλύτερο f1 score = 0.7663

0.10.3 Number of Layers tuning

LSTM Model

$$NumberofLayers = 2$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.49703	0.76224	0.76574	0.75476	0.75683
Test Set	0.50492	0.75365	0.78639	0.69976	0.73834

$$NumberofLayers = 3$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.49634	0.76125	0.76554	0.75268	0.75549
Test Set	0.50461	0.75233	0.78173	0.70450	0.73890

$$NumberofLayers = 4$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.49566	0.76150	0.76679	0.75172	0.75545
Test Set	0.50431	0.75343	0.78714	0.69869	0.73794

$$NumberofLayers = 5$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.49644	0.76058	0.76600	0.75087	0.75464
Test Set	0.50542	0.74991	0.77985	0.70052	0.73578

Παρατηρούμε πως με number of layers = 3 έχουμε στο test set το καλύτερο f1 score = 0.73890

GRU Model

$$Number of Layers = 2$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.47919	0.77034	0.77248	0.76997	0.76754
Test Set	0.50547	0.75502	0.73916	0.78880	0.76006

$$Number of Layers = 3$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.47746	0.77227	0.77334	0.77290	0.76957
Test Set	0.50206	0.76008	0.75864	0.76486	0.75865

$$Number of Layers = 4$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.48258	0.76922	0.77312	0.76748	0.76597
Test Set	0.50533	0.75854	0.74754	0.78312	0.76182

$$Number of Layers = 5$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.48659	0.76518	0.76972	0.76198	0.76140
Test Set	0.50478	0.75656	0.77086	0.73261	0.74782

Παρατηρούμε πως με number of layers = 4 έχουμε στο test set το καλύτερο f1 score = 0.7618

0.10.4 Dropout tuning

LSTM Model

$$Dropout = 0.1$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.50003	0.75995	0.76314	0.75320	0.75474
Test Set	0.50331	0.75189	0.77827	0.70814	0.73925

$$Dropout = 0.25$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.50326	0.75861	0.76234	0.75170	0.75345
Test Set	0.50186	0.75233	0.76716	0.72866	0.74527

$$Dropout = 0.4$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.50866	0.75567	0.75912	0.74896	0.75059
Test Set	0.50406	0.75563	0.76642	0.73852	0.75006

$$Dropout = 0.5$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.51428	0.75224	0.75548	0.74678	0.74745
Test Set	0.50710	0.75167	0.76599	0.72741	0.74410

Παρατηρούμε πως με dropout = 0.4 έχουμε στο test set το καλύτερο f1 score = 0.75006

GRU Model

$$Dropout = 0.1$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.47574	0.77441	0.77704	0.77503	0.77174
Test Set	0.50169	0.75986	0.76133	0.75761	0.75594

$$Dropout = 0.25$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.48324	0.76991	0.77416	0.76664	0.76642
Test Set	0.50319	0.75744	0.75046	0.77153	0.75774

$$Dropout = 0.4$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.49327	0.76461	0.76722	0.76537	0.76165
Test Set	0.50743	0.75722	0.74075	0.79166	0.76279

$$Dropout = 0.5$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.50062	0.76085	0.76566	0.75749	0.75695
Test Set	0.50634	0.75524	0.75449	0.75853	0.75346

Παρατηρούμε πως με dropout $= 0.4$ έχουμε στο test set το καλύτερο f1 score $= 0.7628$

Σημείωση: Παρακάτω θα παραθέτω τις μετρήσεις μόνο για τις επιλεγμένες τιμές των υπερπαραμέτρων.

0.10.5 Batch Value tuning

Δομιμάστηκαν τα εξής batch sizes : $64, 128, 256, 512, 1024$. Παρατηρήθηκε ότι και για τα δύο μοντέλα το καλύτερο batch size είναι το 256.

LSTM Model

$$Batchsize = 256$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.52836	0.74239	0.74444	0.73802	0.73999
Test Set	0.51354	0.75084	0.75243	0.75791	0.75468

GRU Model

$$Batchsize = 256$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.49635	0.76141	0.76547	0.75615	0.75907
Test Set	0.50851	0.75030	0.73608	0.77942	0.75628

0.10.6 Learning rate tuning

Δομιμάστηκαν τα εξής learning rates : $1e-3, 1e-4, 1e-5$. Παρατηρήθηκε ότι και για τα δύο μοντέλα το καλύτερο learning rate είναι το $1e-3$.

LSTM Model

$$Learningrate = 1e-3$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.49536	0.76319	0.76788	0.75654	0.76033
Test Set	0.49052	0.76163	0.77863	0.74093	0.75872

GRU Model

$$Learningrate = 1e-3$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.49553	0.76136	0.76468	0.75775	0.75953
Test Set	0.51041	0.75068	0.74098	0.77042	0.75446

0.10.7 Clipping tuning

Δομιμάστηκαν τα εξής clipping values : $1, 2, 3, 4, 5$. Παρατηρήθηκε ότι και για τα LSTM cells το καλύτερο lipping value είναι το 5 ενώ για GRU cells το καλύτερο clipping value είναι το 4.

LSTM Model

$$Clippingvalue = 5$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.49659	0.76259	0.76806	0.75514	0.759465
Test Set	0.49195	0.76147	0.76512	0.76424	0.76410

GRU Model

$$Clippingvalue = 4$$

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.49591	0.76152	0.76681	0.75488	0.75888
Test Set	0.51144	0.75207	0.73554	0.78753	0.75962

0.10.8 LSTM vs GRU

Σε αυτό το σημείο έχουμε βρει τις καλύτερες υπερπαραμέτρους για τα δύο μοντέλα μας, οπότε θα τα συγκρίνουμε και θα επιλέξουμε το καλύτερο. Πριν τα συγκρίνουμε θα κάνουμε μερικές αλλαγές

- Θα χρησιμοποιήσουμε τα Embeddings των 300 διαστάσεων.
- Και θα προσθέσουμε 3 ακόμα linear layers με ενδιάμεσα activation function relu όπως και στην προηγούμενη εργασία.

Final Model with LSMT cells

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.35619	0.84463	0.84346	0.84667	0.84412
Test Set	0.37243	0.83780	0.81629	0.87681	0.84529

Final Model with GRU cells

Epoch 10	Loss	Accuracy	Precision	Recall	f1
Train Set	0.35920	0.84309	0.84608	0.83946	0.84172
Test Set	0.38259	0.84400	0.85123	0.83848	0.84456

Παρατηρούμε πως με LSTM cell έχουμε στο test set το καλύτερο f1 score = 0.84529 οπότε θα επιλέξουμε αυτό ως τελικό μοντέλο.

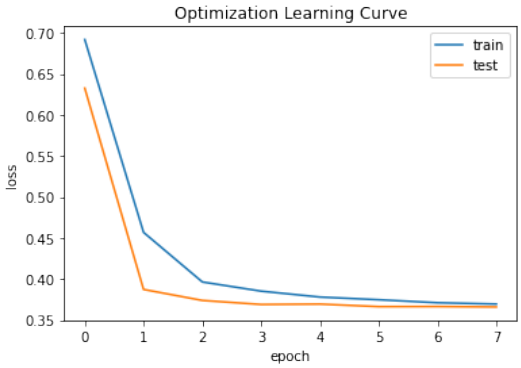
Σημείωση: Βλέπουμε ότι δεν χρειάζονται πάνω απο 10 εποχές καθώς το loss φαίνεται να μην μειώνεται και το accuracy να μην αυξάνεται σημαντικά στις 2 τελευταίες εποχές. Οπότε μπορούμε να κρατήσουμε ως $nepoch = 8$

0.11 Plots

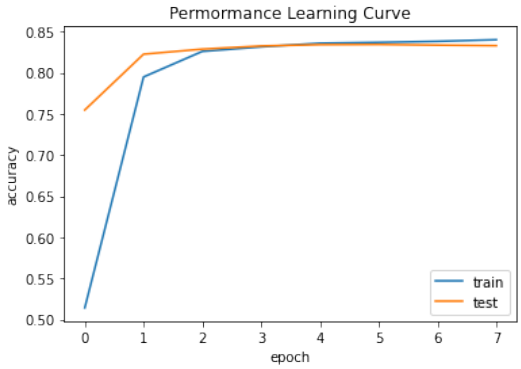
0.11.1 Comments

Παρατηρούμε ότι δεν παρουσιάζεται overfitting στις γραφικές καθώς οι καμπύλες όλο και συγκλίνουν μεταξύ τους και βρίσκονται πολύ κοντά η μία στην άλλη. Επίσης φαίνεται ότι το μοντέλο μας δεν κάνει ούτε underfit καθώς έχει αρκετά καλές επιδόσεις στο training data, τάξης του 0.84. Τέλος το roc curve δείχνει ότι το μοντέλο μας είναι αποδοτικό στο classification ενώ το loss graph δείχνει ότι μετά από ένα σημείο το μοντέλο δεν μπορεί να μάθει παραπάνω όποτε και πρέπει να το σταματήσουμε καθώς θα είχαμε θέμα με το overfit.

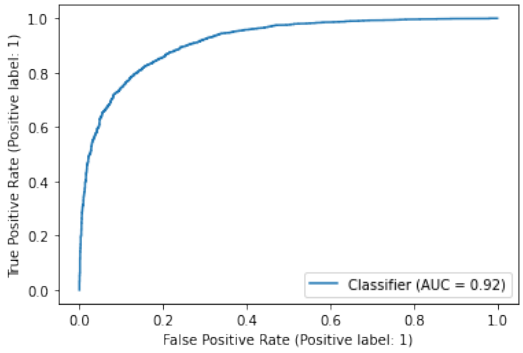
0.11.2 Optimization Learning Curve



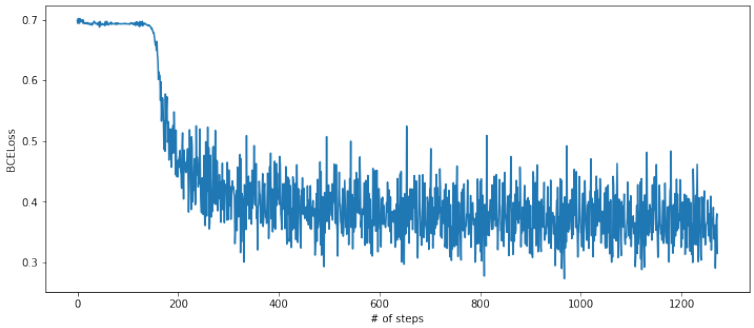
0.11.3 Performance Learning Curve



0.11.4 ROC curve



0.11.5 Loss graph

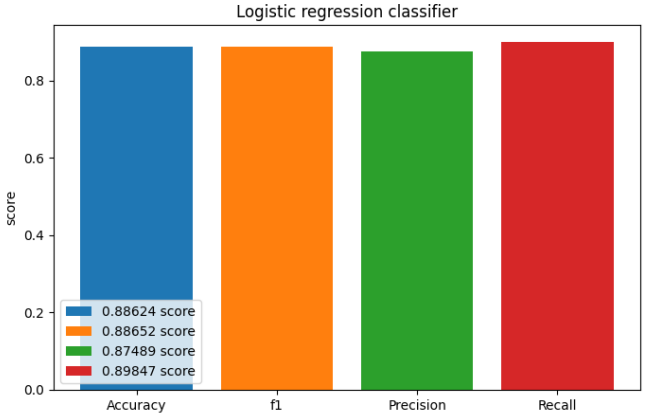


0.12 Compare different models

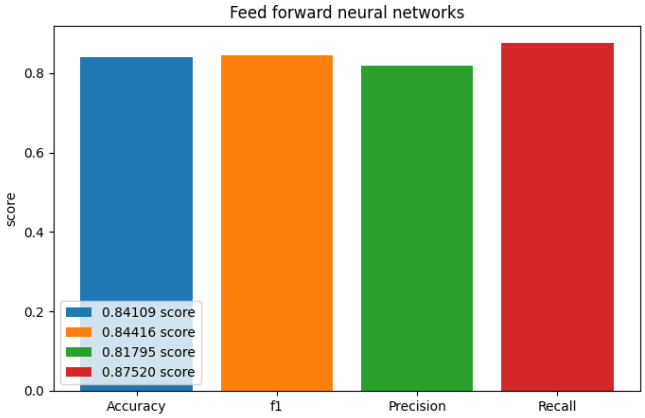
0.13 Comments

Παρατηρούμε πως το υψηλότερο accuracy το έχει πετύχει το μοντέλο της πρώτης εργασίας ενώ οι επιδόσεις των μοντέλων της δέυτερης και της τρίτης εργασίας σχεδόν ταυτίζονται με αυτές της τρίτης να είναι λίγο καλύτερες. Θα περιμέναμε τα πιο σύνθετα μοντέλα να ξεπερνούν την αποδοτικότητα του πρώτου μοντέλου. Αυτό μπορεί να οφείλεται είτε σε κακή μοντελοποίηση είτε λόγο περιορισμένων δεδομένων. Επίσης κάτι που δεν φαίνεται στα διαγράμματα είναι ότι τα losses του τρίτου μοντέλου είναι λιγότερα από αυτά του δευτέρου.

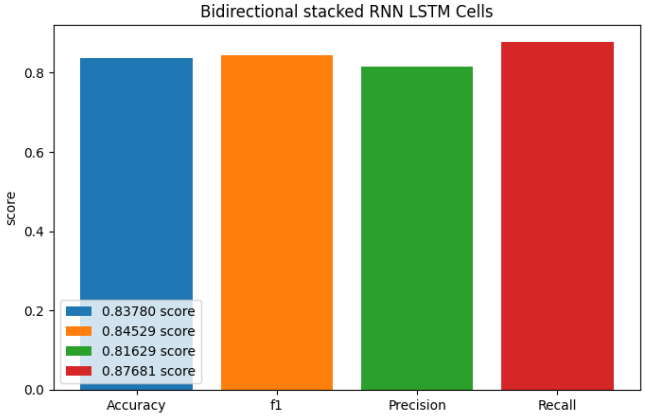
0.13.1 Project 1



0.13.2 Project 2



0.13.3 Project 3



0.14 Comments

Στην αρχή της εργασίας προσπάθησα να κάνω tune τις υπερπαραμέτρους με την χρήση του optuna όμως όταν το έτρεχα έκανε time-out. Οπότε ακολούθησα την ίδια τακτική με τις προηγούμενες εργασίες για την εύρεση των κατάλληλων υπερπαραμέτρων.
Μέσα στο φάκελο με το παραδοτέο έχω και όλες τις μετρήσεις που έκανα για την εργασία - folder (metrics).