ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Επεξεργασία Φωνής και Φυσικής Γλώσσας
Χειμερινό Εξάμηνο 2020-21

Προπαρασκευή 3ου Εργαστηρίου

Εισαγωγή

Σκοπός είναι η υλοποίηση μοντέλου για επεξεργασία και κατηγοριοποίηση κειμένων, με την χρήση βαθιών νευρωνικών δικτύων (Deep Neural Networks - DNN). Για την ανάπτυξη των μοντέλων θα πρέπει να χρησιμοποιήσετε την βιβλιοθήκη PyTorch[1]. Αρχικά, χρησιμοποιώντας προ-εκπαιδευμένες διανυσματικές αναπαραστάσεις λέξεων (pretrained word embeddings), καλείστε να δημιουργήσετε αναπαραστάσεις για κάθε κείμενο. Στη συνέχεια, θα χρησιμοποιήσετε τις αναπαραστάσεις των κειμένων, ώστε να κάνετε την κατηγοριοποίηση. Στόχος είναι να εκπαιδεύσετε μοντέλα, τα οποία θα μπορούν να κάνουν ανάλυση συναισθήματος (sentiment analysis) σε προτάσεις. Σας παρέχονται 2 σύνολα δεδομένων (περιέχονται ήδη στο αποθετήριο της άσκησης).

- Sentence Polarity Dataset[2] [Pang and Lee, 2005]. Το dataset αυτό περιέχει 5331 θετικές και 5331 αρνητικές κριτικές ταινιών, από το Rotten Tomatoes και είναι binary-classification πρόβλημα (positive, negative).
- Semeval 2017 Task4-A[3] [Rosenthal et al., 2017]. Το dataset αυτό περιέχει tweets τα οποία είναι κατηγοριοποιημένα σε 3 κλάσεις (positive, negative, neutral) με 49570 παραδείγματα εκπαίδευσης και 12284 παραδείγματα αξιολόγησης.

Περιβάλλον Ανάπτυξης

Σε πρώτη φάση θα πρέπει να στήσετε το περιβάλλον ανάπτυξης στον υπολογιστή σας. Αρχικά κατεβάσετε το αποθετήριο: `https://github.com/slp-ntua/slp-labs/tree/master/lab3`. Στην συνέχεια κατεβάστε προεκπαιδευμένα διανύσματα λέξεων της επιλογής σας στον φάκελο */embeddings* του project. Ενδεικτικά σας προτείνουμε να κατεβάσετε τα GloVe[4] embeddings, καθώς υπάρχουν διαθέσιμα embeddings σε χαμηλές διαστάσεις, το οποίο σημαίνει λιγότερες υπολογιστικές απαιτήσεις. Εναλλακτικά θα μπορούσατε να χρησιμοποιήσετε και τα FastText[5] embeddings, τα οποία όμως είναι διαθέσιμα μόνο σε 300 διαστάσεις. Ανάλογα με το dataset θα παρατηρήσετε ότι διαφορετικά embeddings πετυχαίνουν καλύτερα αποτελέσματα. Για παράδειγμα, στο dataset του Semeval 2017 Task4-A, θα δείτε καλύτερα αποτελέσματα με τα Twitter Glove embeddings, καθώς είναι εκπαιδευμένα πάνω σε tweets. Η επιλογή είναι δική σας και δεν θα αξιολογηθείτε ως προς τις επιδόσεις του μοντέλου σας, αλλά ως προς την ορθότητα των απαντήσεων σας.

Θα πρέπει να εγκαταστήσετε ορισμένες βιβλιοθήκες για την υλοποίηση της άσκησης. Προτείνεται να τις εγκαταστήσετε σε ένα ανεξάρτητο περιβάλλον, ώστε να μην υπάρξει οποιαδήποτε ανεπιθύμητη συνέπεια στις υπόλοιπες βιβλιοθήκες στον υπολογιστή σας. Η πιο εύκολη λύση είναι να χρησιμοποιήσετε το εργαλείο `conda`. Αν δεν το έχετε ήδη εγκατεστημένο στον υπολογιστή σας, κατεβάσετε την σωστή έκδοση του Miniconda για την **έκδοση 3** της Python και εγκαταστήστε το ακολουθώντας τις οδηγίες. Τέλος, δημιουργήστε ένα νέο περιβάλλον[6] και ενεργοποιήστε το ως εξής:

[1] `https://pytorch.org/`
[2] `http://www.cs.cornell.edu/people/pabo/movie-review-data/`
[3] `http://alt.qcri.org/semeval2017/task4/index.php?id=data-and-tools`
[4] `https://nlp.stanford.edu/projects/glove/`
[5] `https://fasttext.cc/docs/en/english-vectors.html`
[6] `https://conda.io/docs/user-guide/tasks/manage-environments.html`

```
conda create -n slp3 python=3
source activate slp3
```

Εγκαταστήστε το PyTorch ακολουθώντας τις οδηγίες και στη συνέχεια όλες τις υπόλοιπες βιβλιοθήκες με την εντολή:

```
pip install -r requirements.txt
```

Εξοικειωθείτε με την δομή του project και μελετήστε προσεκτικά τον κώδικα που έχει ήδη υλοποιηθεί. Το κόκκινο χρώμα υποδηλώνει τα αρχεία τα οποία θα πρέπει να επεξεργαστείτε στα πλαίσια της άσκησης:

`/datasets`	φάκελος με τα δεδομένα εκπαίδευσης
`/embeddings`	φάκελος με τα προεκπαιδευμένα word embeddings
`/utils`	φάκελος με βοηθητικά Python scripts
`config.py`	ρυθμίσεις σχετικά με το project
`dataloading.py`	προεπεξεργασία και προετοιμασία των παραδειγμάτων
`main.py`	κυρίως script - αφετηρία
`models.py`	περιέχει τα μοντέλα με το/τα νευρωνικά δίκτυα
`training.py`	περιέχει τις συναρτήσεις για την εκπαίδευση των μοντέλων

Η συγκεκριμένη εργασία έχει την λογική της συμπλήρωσης κενών στον κώδικα. Πολλά από τα τετριμμένα μέρη της άσκησης, έχουν ήδη υλοποιηθεί και εσείς καλείστε να επέμβετε σε συγκεκριμένα σημεία στον κώδικα, ανάλογα με το ερώτημα. Στην πλειοψηφία των ερωτημάτων, θα πρέπει να συμπληρώσετε ελάχιστες μόνο γραμμές. Ανοίγοντας το αρχείο **main.py**, θα δείτε ότι έχει ήδη υλοποιηθεί για εσάς το βήμα της φόρτωσης των δεδομένων εκπαίδευσης, καθώς και των pretrained word embeddings. Η θέση στην οποία θα πρέπει να υλοποιήσετε τη λύση κάθε ερωτήματος στον πηγαίο κώδικα, θα σημειώνεται με **FILE:EXi**, όπου **FILE** είναι το αρχείο και **EXi** το αναγνωριστικό του ερωτήματος, το οποίο θα υπάρχει σε σχόλια στον κώδικα[7].

1 Προεπεξεργασία Δεδομένων

Σε αυτό το βήμα θα πρέπει να επεξεργαστείτε τα δεδομένα, ώστε να μπορείτε να εκπαιδεύσετε στη συνέχεια το νευρωνικό δίκτυο. Για την καλύτερη διαχείριση των δεδομένων εκπαίδευσης θα χρησιμοποιήσετε τα εργαλεία που παρέχει το PyTorch (κλάσεις Dataset και Dataloader κλπ.[8]), κληρονομώντας τις αντίστοιχες κλάσεις και φτιάχνοντας δικές σας, ανάλογα με τις ανάγκες της άσκησης. Η κλάση `torch.utils.data.Dataset` μετατρέπει κάθε παράδειγμα στην μορφή που απαιτείται για την εκπαίδευση του νευρωνικού δικτύου και η κλάση `torch.utils.data.Dataloader`, χρησιμοποιεί ένα αντικείμενο της κλάσης `Dataset` για να μετατρέψει τα παραδείγματα του σε torch Tensors και να τα οργανώσει σε mini-batces. Η επέκταση της κλάσης `Dataset` που θα φτιάξετε, θα περιέχει τις μεταβλητές με τα παραδείγματα και τις επισημειώσεις (labels) κάθε συνόλου δεδομένων, καθώς και τις μεθόδους για την επεξεργασία και την προετοιμασία τους.

1.1 Κωδικοποίηση Επισημειώσεων (Labels)

Αρχικά οι επισημειώσεις των παραδειγμάτων εκπαίδευσης έχουν την μορφή κειμένου (positive, neutral, negative...). Θα πρέπει να τις κωδικοποιήσετε, ώστε κάθε κλάση να αντιστοιχεί σε έναν συγκεκριμένο αριθμό. Η αντιστοίχηση θα πρέπει να είναι ίδια για training και test sets.

Βοήθεια: Χρησιμοποιήστε τον `LabelEncoder`[9] του scikit-learn.

Ζητούμενο 1: Συμπληρώστε τα κενά στη θέση `main.py:EX1` και τυπώστε τα πρώτα 10 labels από τα δεδομένα εκπαίδευσης και τις αντιστοιχίες τους σε αριθμούς.

[7] απλά κάντε αναζήτηση στον κώδικα του αρχείου βάση του αναγνωριστικού και θα βρείτε όλες τις αναφορές
[8] https://pytorch.org/docs/stable/data.html
[9] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

1.2 Λεκτική Ανάλυση (Tokenization)

Στην διαδικασία αυτή ένα κείμενο, μετατρέπεται από μία ακολουθία χαρακτήρων, σε μία ακολουθία από λεκτικές μονάδες ή όρους (tokens ή terms), όπως λέξεις, σημεία στίξης, αριθμούς κλπ. Στο αρχείο `dataloading.py` έχει δηλωθεί η κλάση `SentenceDataset`, η οποία επεκτείνει την κλάση `torch.utils.data.Dataset`. Εκτελέσετε το tokenization κατά την αρχικοποίηση του Dataset σε όλα τα δεδομένα του και διατηρήσετε τα επεξεργασμένα δεδομένα σε μεταβλητή στην κλάση. Μπορείτε να χωρίσετε κάθε κείμενο βάση του χαρακτήρα του κενού, ή να εκτελέσετε κάποια πιο εξελιγμένη λεκτική ανάλυση ανάλογα τις ιδιαιτερότητες του dataset (όπως NLTK, spaCy, ekphrasis).

Ζητούμενο 2: Συμπληρώστε τα κενά στη θέση `dataloading.py:EX2` και τυπώστε τα πρώτα 10 παραδείγματα από τα δεδομένα εκπαίδευσης.

1.3 Κωδικοποίηση Παραδειγμάτων (Λέξεων)

Αυτό είναι το τελικό βήμα, στο οποίο θα πρέπει να προετοιμάσετε κάθε παράδειγμα στην κατάλληλη μορφή για την εκπαίδευση του νευρωνικού δικτύου. Απαιτείται να υλοποιήσετε τα εξής:

1. κάθε όρος (token - term) πρέπει να χαρτογραφηθεί σε ένα αριθμό, ώστε να μπορεί το embedding layer να τον αντιστοιχίσει στη σωστή διανυσματική αναπαράσταση (word embedding). Η συνάρτηση φόρτωσης των προ-εκπαιδευμένων word embeddings `load_word_vectors` σας επιστρέφει ένα python dictionary το οποίο έχει την μορφή `word:id`. Χρησιμοποιήστε το. Αν ένας όρος δεν υπάρχει στο dictionary, τότε θα πρέπει να τον αντιστοιχίσετε στον όρο `<unk>`.

2. όλα τα παραδείγματα θα πρέπει να έχουν το ίδιο μήκος. Αυτό απαιτείται για να μπορούν να εκτελεστούν πράξεις γραμμικής άλγεβρας, όπως πολλαπλασιασμός πινάκων. Για τον λόγο αυτό θα πρέπει να επιλέξετε ένα μέγιστο μήκος προτάσεων και να συμπληρώνετε με μηδενικά στοιχεία τις προτάσεις μικρότερου μήκους (zero-padding) ή να αφαιρείτε λέξεις που ξεπερνούν το επιλεγμένο μήκος. Μπορείτε να τυπώσετε την κατανομή με τα μήκη των προτάσεων στο training set και να διαλέξετε ένα μήκος το οποίο να καλύπτει την πλειοψηφία, αγνοώντας τους outliers.

Η μέθοδος `__getitem__` της κλάσης `SentenceDataset` πρέπει να επιστρέφει τα εξής: 1) την κωδικοποιημένη μορφή μίας πρότασης, 2) το id της επισημείωσης 3) το **πραγματικό** μήκος της πρότασης (δηλαδή εξαιρουμένων των μηδενικών στοιχείων).

Παράδειγμα κωδικοποίησης μίας πρότασης, για μέγιστο_μήκος = 8:

```
dataitem = "this is a simple example", label = "neutral"

Return values:
example = [  533  3908  1387   649   88     0     0     0]
label = 1
length = 5
```

Ζητούμενο 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`.

2 Μοντέλο

Σε αυτό το βήμα θα πρέπει να σχεδιάσετε ένα νευρωνικό δίκτυο, το οποίο 1) θα δημιουργεί μία συνεχή διανυσματική αναπαράσταση για κάθε όρο σε μία πρόταση με την χρήση ενός Embedding layer, 2) θα δημιουργεί μία διανυσματική αναπαράσταση για όλο το κείμενο ενός παραδείγματος, 3)

θα κατηγοριοποιεί το κείμενο βάση της αναπαράστασης του στην σωστή κλάση. Ο πηγαίος κώδικας του μοντέλου βρίσκεται στο αρχείο `models.py` και θα πρέπει να υλοποιήσετε τις μεθόδους `__init__` και `__forward__` της κλάσης `BaselineDNN`. Στη μέθοδο `__init__` θα δηλώσετε τα layers του δικτύου και θα αρχικοποιήσετε τα βάρη τους. Στη μέθοδο `__forward__` θα ορίσετε τους μετασχηματισμούς των δεδομένων εισόδου για την παραγωγή της τελικής εξόδου του μοντέλου (forward pass).

2.1 Embedding Layer

Αρχικά, θα χρησιμοποιήσετε ένα embedding layer, για να προβάλετε κάθε όρο/λέξη ενός κειμένου σε ένα συνεχή χώρο (embedding space). Το embedding layer τοποθετεί τις λέξεις στον χώρο ανάλογα με τις σχέσεις που έχουν μεταξύ τους. Λέξεις οι οποίες συνεμφανίζονται συχνά, θα βρίσκονται κοντά μεταξύ τους. Το διάνυσμα που αντιστοιχεί στην θέση κάθε λέξης ονομάζεται αναπαράσταση της λέξης ή χαρακτηριστικά της λέξης (features) ή word embedding. Τα βάρη του embedding layer μπορείτε να τα αρχικοποιήσετε με τυχαίες τιμές και να τα ενημερώσετε κατά την εκπαίδευση του μοντέλου ή να τα αρχικοποιήσετε από προ-εκπαιδευμένα word embeddings.
Υλοποιήστε τα εξής:

- Δημιουργήστε ένα embedding layer[10].
- Αρχικοποιήσετε τα βάρη του δικτύου από τα προεκπαιδευμένα word embeddings. Η συνάρτηση φόρτωσης των προ-εκπαιδευμένων word embeddings `load_word_vectors` σας επιστρέφει την διδιάστατη μήτρα με τα βάρη (`shape: num_embeddings, emb_dim`). Χρησιμοποιήστε την.
- Παγώστε το embedding layer, δηλαδή δηλώστε ότι τα βάρη του δικτύου δεν θα ενημερωθούν περαιτέρω κατά την εκπαίδευση του μοντέλου.

Απαντήστε συνοπτικά στα εξής:

- Γιατί αρχικοποιούμε το embedding layer με τα προ-εκπαιδευμένα word embeddings?
- Γιατί κρατάμε παγωμένα τα βάρη του embedding layer κατά την εκπαίδευση?

Ζητούμενο 4: Συμπληρώστε τα κενά στις θέσεις `models.py:EX4` και απαντήστε στα παραπάνω ερωτήματα.

2.2 Output Layer(s)

Για την κατηγοριοποίηση, θα πρέπει να προβάλετε τις αναπαραστάσεις των κειμένων στον χώρο των κλάσεων. Δηλαδή, αν οι τελικές αναπαραστάσεις είναι 500 διαστάσεων και το πρόβλημα κατηγοριοποίησης έχει 3 κλάσεις, τότε θα πρέπει το τελευταίο layer να κάνει προβολή $R^{500} \rightarrow R^3$.
Υλοποιήστε τα εξής:

- Δημιουργήστε ένα layer με μία μη γραμμική συνάρτηση ενεργοποίησης (π.χ. ReLU, tanh...), το οποίο θα μαθαίνει ένα μετασχηματισμό της αναπαράστασης κάθε παραδείγματος.
  *Υποθέστε ότι οι διαστάσεις των αναπαραστάσεων των παραδειγμάτων, είναι όσες οι διαστάσεις των word embeddings. Για την έξοδο του layer διαλέξτε εσείς ότι διαστάσεις επιθυμείτε.
- Δημιουργήστε το τελευταίο layer το οποίο προβάλει τις τελικές αναπαραστάσεις των κειμένων στις κλάσεις.

Απαντήστε συνοπτικά στα εξής:

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

Ζητούμενο 5: Συμπληρώστε τα κενά στις θέσεις `models.py:EX5` και απαντήστε στα παραπάνω ερωτήματα.

[10] https://pytorch.org/docs/stable/generated/torch.nn.Embedding.html#torch.nn.Embedding

2.3 Forward pass

Αφού έχετε σχεδιάσει τα layers τα οποία θα χρησιμοποιεί το μοντέλο σας, το τελευταίο βήμα είναι να σχεδιάσετε τον τρόπο με τον οποίο θα μετασχηματίσει το δίκτυο τα δεδομένα εισόδου στις αντίστοιχες εξόδους (προβλέψεις). Η είσοδος στο μοντέλο θα είναι ένα mini-batch, με διαστάσεις (`batch_size, max_length`). Σε αυτό το ερώτημα σας ζητείται να δημιουργήσετε ένα πολύ απλοϊκό μοντέλο το οποίο θα εκτελεί τους ακόλουθους μετασχηματισμούς:

1. Προβολή των λέξεων κάθε πρότασης με το embedding layer που δημιουργήσατε, όπου κάθε όρος (λέξη) θα αντιστοιχεί σε ένα διάνυσμα. Η είσοδος θα έχει διαστάσεις (`batch_size, max_length`) και η έξοδος (`batch_size, max_length, emb_dim`).

2. Δημιουργήστε μία αναπαράσταση για κάθε πρόταση. Οι προτάσεις έχουν μεταβλητό μήκος, όμως θα πρέπει όλες οι αναπαραστάσεις των κειμένων να βρίσκονται στον ίδιο χώρο (ίδιες διαστάσεις). Μπορείτε να δημιουργήσετε μία απλή αναπαράσταση, υπολογίζοντας τον μέσο όρο (κέντρο βάρους) των word embeddings σε μία πρόταση. Έτσι όλες οι προτάσεις θα έχουν μία διανυσματική αναπαράσταση με διαστάσεις, όσες οι διαστάσεις των word embeddings.

3. Εφαρμόστε τον μη-γραμμικό μετασχηματισμό που σχεδιάσατε στο προηγούμενο ερώτημα σε κάθε αναπαράσταση και δημιουργήστε καινούριες αναπαραστάσεις (deep-latent representations).

4. Προβάλετε τις τελικές αναπαραστάσεις στον χώρο των κλάσεων.

Απαντήστε συνοπτικά στα εξής:

- 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).

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

Βοήθεια: Για τον υπολογισμό του μέσου όρου των word embeddings μίας πρότασης, πρέπει να λάβετε υπόψη σας και να **εξαιρέσετε** τα zero-padded timesteps. Για αυτό τον λόγω θα πρέπει να χρησιμοποιήσετε τα πραγματικά μήκη κάθε πρότασης, τα οποία έχετε υπολογίσει στην μέθοδο `__getitem__` της κλάσης `SentenceDataset` και τα οποία έχετε περάσει ως είσοδο στη μέθοδο `__forward__` του μοντέλου. Για παράδειγμα, στην ακολουθία $[3, 5, 2, 8, 0, 0, 0, 0]$ ο μέσος όρος θα πρέπει να είναι 4.5 και όχι 2.25, καθώς το πραγματικό μήκος της ακολουθίας (όπως εννοείται στα πλαίσια του προβλήματος) είναι 4 και όχι 8.

Ζητούμενο 6: Συμπληρώστε τα κενά στις θέσεις `models.py:EX6` στη μέθοδο `__forward__` και απαντήστε στα παραπάνω ερωτήματα.

3 Διαδικασία Εκπαίδευσης

Σε αυτό το βήμα θα πρέπει να υλοποιήσετε την διαδικασία εκπαίδευσης του δικτύου, δηλαδή θα οργανώσετε τα παραδείγματα σε mini-batches και να εκτελέσετε stochastic gradient descent για να ενημερώσετε τα βάρη του δικτύου.

3.1 Φόρτωση Παραδειγμάτων (DataLoaders)

Αφού υλοποιήσατε την κλάση `SentenceDataset` για την μετατροπή των παραδειγμάτων στην κατάλληλη μορφή, θα πρέπει στη συνέχεια να σχεδιάσετε τον τρόπο με τον οποίο το νευρωνικό δίκτυο θα εκπαιδεύεται από τα αντίστοιχα παραδείγματα. Χρησιμοποιήστε την κλάση DataLoader και φτιάξτε ένα στιγμιότυπο για κάθε dataset. Δώστε μία σύντομη απάντηση στα εξής:

- Τι συνέπειες έχουν τα μικρά και μεγάλα mini-batches στην εκπαίδευση των μοντέλων?
- Συνήθως ανακατεύουμε την σειρά των mini-batches στα δεδομένα εκπαίδευσης σε κάθε εποχή. Μπορείτε να εξηγήσετε γιατί?

Ζητούμενο 7: Συμπληρώστε τα κενά στις θέσεις `main.py:EX7` και απαντήστε στα παραπάνω ερωτήματα.

3.2 Βελτιστοποίηση

Για την βελτιστοποίηση του μοντέλου πρέπει να ορίσετε τα εξής:

1. Κριτήριο. Αν το πρόβλημα κατηγοριοποίησης έχει 2 κλάσεις τότε χρησιμοποιήστε το `BCEWithLogitsLoss`, ενώ αν έχει περισσότερες τότε χρησιμοποιήστε το `CrossEntropyLoss`. Προσοχή στις διαστάσεις του output layer του νευρωνικού ανάλογα με το κριτήριο που θα επιλέξετε.
2. Παράμετροι. Επιλέξτε τις παραμέτρους οι οποίες θα βελτιστοποιηθούν. Προσοχή, καθώς δεν θέλουμε να εκπαιδεύσουμε όλες τις παραμέτρους του δικτύου.
3. Optimizer. Επιλέξτε έναν αλγόριθμο βελτιστοποίησης. Προτείνεται να χρησιμοποιήσετε έναν αλγόριθμο ο οποίος προσαρμόζει αυτόματα την ταχύτητα ενημέρωσης των βαρών (Adam, Adagrad, RMSProp...).

Ζητούμενο 8: Συμπληρώστε τα κενά στις θέσεις `main.py:EX8`.

3.3 Εκπαίδευση

Το τελευταίο βήμα για την εκπαίδευση του μοντέλου, είναι να υλοποιήσετε τις μεθόδους για την εκπαίδευση και αξιολόγηση κάθε mini-batch. Η συνάρτηση `train_dataset` καλείται για κάθε batch σε μία εποχή, δίνει τα δεδομένα εκπαίδευσης στο μοντέλο, υπολογίζει το σφάλμα και ενημερώνει τα βάρη του δικτύου με τον αλγόριθμο backpropagation. Ομοίως, η συνάρτηση `eval_dataset` καλείται στο τέλος κάθε εποχής, για να αξιολογήσει το μοντέλο.

Βοήθεια: Για τον υπολογισμό των προβλέψεων του δικτύου (πιθανότερη κλάση) κατά την αξιολόγηση, όπου πρέπει να υπολογίσετε το argmax των posteriors, μπορείτε να χρησιμοποιήσετε την συνάρτηση max[11]. Αν το πρόβλημα είναι binary-classification τότε σας ενδιαφέρει αν το logit είναι μεγαλύτερο του μηδέν (πιθανότητα πάνω από 0.5).

Ζητούμενο 9: Συμπληρώστε τα κενά στις θέσεις `training.py:EX9`.

3.4 Αξιολόγηση

Αφού έχετε ολοκληρώσει την ανάπτυξη του μοντέλου, το τελικό ζητούμενο είναι να αξιολογήσετε τις επιδόσεις του. Το πλήθος των εποχών που θα εκπαιδεύσετε τα μοντέλο μπορείτε να το αποφασίσετε εσείς. Μπορεί το πλήθος να είναι μία προεπιλεγμένη τιμή, ή να σταματάτε την εκπαίδευση όταν το σφάλμα (loss) στο test set σταματάει να μειώνεται.

Ζητούμενο 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.

[11] https://pytorch.org/docs/stable/generated/torch.max.html#torch.max

4 Παραδοτέα

Σε ότι αφορά την βαθμολόγηση της άσκησης, δεν θα αξιολογηθείτε ως προς τις επιδόσεις του μοντέλου σας, αλλά ως προς την ορθότητα των απαντήσεων σας. Θα πρέπει να παραδώσετε τα εξής:

1. Σύντομη αναφορά (σε pdf ή jupyter notebook) που θα περιέχει τις απαντήσεις σε κάθε ερώτημα και τα αντίστοιχα αποτελέσματα.
2. Κώδικας Python, συνοδευόμενος από σύντομα σχόλια.

Συγκεντρώστε τα (1) και (2) σε ένα .zip αρχείο το οποίο πρέπει να υποβληθεί **πριν** από τη διεξαγωγή του 3ου εργαστηρίου (18/1/2021).

References

[Pang and Lee, 2005] Pang, B. and Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

[Rosenthal et al., 2017] Rosenthal, S., Farra, N., and Nakov, P. (2017). SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, Vancouver, Canada. Association for Computational Linguistics.

Παράρτημα - Βοηθητικό Υλικό

Επίσημοι Οδηγοί

Μπορείτε να ξεκινήσετε τη μελέτη σας από τους ακόλουθους γενικούς οδηγούς:

- Deep Learning with PyTorch: A 60 Minute Blitz
- Learning PyTorch with Examples

Και να συνεχίσετε με αυτούς που εστιάζουν σε NLP προβλήματα:

- Introduction to PyTorch
- Deep Learning with PyTorch
- Word Embeddings: Encoding Lexical Semantics

Οδηγοί από Τρίτους

Πολύ αξιόλογοι είναι οι οδηγοί από το μάθημα "CS230 Deep Learning" του Stanford:

- Introduction to Pytorch Code Examples - An overview of training, models, loss functions and optimizers
- Named Entity Recognition Tagging - Defining a Recurrent Network and Loading Text Data

Επιπλέον:

- Το άρθρο "PyTorch - Basic operations" είναι ένα πολύ καλό σημείο αναφοράς με τις βασικές πράξεις που μπορείτε να κάνετε με τανυστές στο PyTorch.

- Η λίστα με υλοποιημένα μοντέλα "Pytorch: Lists of tutorials examples" μπορεί να σας φανεί χρήσιμη.
- Όπως και ο οδηγός "Writing Custom Datasets, DataLoaders and Transforms" για απορίες σχετικά με την προετοιμασία των δεδομένων.

Τέλος, προτείνουμε να δοκιμάζετε επιμέρους λειτουργίες χρησιμοποιώντας κάποιο ξεχωριστό script, ώστε να ελέγχετε εύκολα και γρήγορα την συμπεριφορά τους. Όπως στο ακόλουθο παράδειγμα που αφορά ένα embedding layer:

```
import torch
import torch.nn as nn

batch_size = 16
max_length = 8
n_embeddings = 1000
embedding_size = 50

# create a batch of random sequences of token ids
x = (torch.rand(batch_size, max_length) * n_embeddings).long()
print(x.shape)

embed_layer = nn.Embedding(num_embeddings=n_embeddings, embedding_dim=embedding_size)

embeddings = embed_layer(x)
print(embeddings.shape)
```