ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΗΜΑΤΩΝ ΕΛΕΓΧΟΥ ΚΑΙ ΡΟΜΠΟΤΙΚΗΣ

Επεξεργασία Φωνής & Φυσικής Γλώσσας

Ροή Σ: Σήματα, Έλεγχος, Ρομποτική

7ο Εξάμηνο

3η Εργαστηριακή Άσκηση

ΘΕΜΑ:

RNNs/LSTMs for Sentiment Analysis

Χρήστος Δημόπουλος

03117037

ABSTRACT

Σκοπός της άσκησης είναι η υλοποιήση μοντέλων για την επεξεργασία και κατηγοριοποίηση κειμένων, με τη χρήση νευρωνικών δικτύων. Έχοντας σχεδιάσει στο προπαρασκευαστικό μέρος του εργαστηρίου απλές αρχιτεκτονικές νευρωνικών δικτύων (DNN) με τη χρήση προ-εκπαιδευμένων word embeddings, εμβαθύνουμε περαιτέρω στην αναγνώριση συναισθήματος μέσω κειμένου, χρησιμοποιώντας **Ανατροφοδοτούμενα Νευρωνικά Δίκτυα** (Recurrent Neural Networks), τεχνικές **Μεταφοράς Γνώσης** (Transfer Learning) και μηχανισμούς **Προσοχής** (Attention mechanisms).

Για την ανάπτυξη των παραπάνω μηχανισμών Μηχανικής Μάθησης, χρησιμοποιείται το dataset ‘’***Sentence Polarity Dataset’’*** [Pang and Lee, 2005] [1], το οποίο περιέχει 5331 θετικές και 5331 αρνητικές κριτικές ταινιών, από το Rotten Tomatoes και είναι binary-classification πρόβλημα (positive, negative). Επιπλέον, για την εκπαίδευση των μοντέλων επιλέγονται τα προεκπαιδευμένα word embeddings "***glove.6B.100d.txt***", ενώ ως τιμές των υπερπαραμέτρων ορίζονται **batch_size = 64**, ***BCEWithLogitsLoss*** ως συνάρτηση κόστους, ***ADAM optimizer*** με **learning rate = 0.001** και **15 εποχές** – εκτός αν αναφέρεται ρητά κάποια άλλη παραμετροποίηση.

Contents

1. Mean & Max Pooling

Έχοντας προσδιορίσει τις διανυσματικές αναπαραστάσεις κάθε λέξης μέσω του embedding layer του έκαστου μοντέλου, επιλέγεται ως αναπαράσταση κάθε πρότασης u η συνένωση του μέσου όρου (mean pooling) και του μέγιστου ανά διάσταση (max pooling) των word embeddings κάθε πρότασης $E = (e_1, e_2, \dots, e_N)$.

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

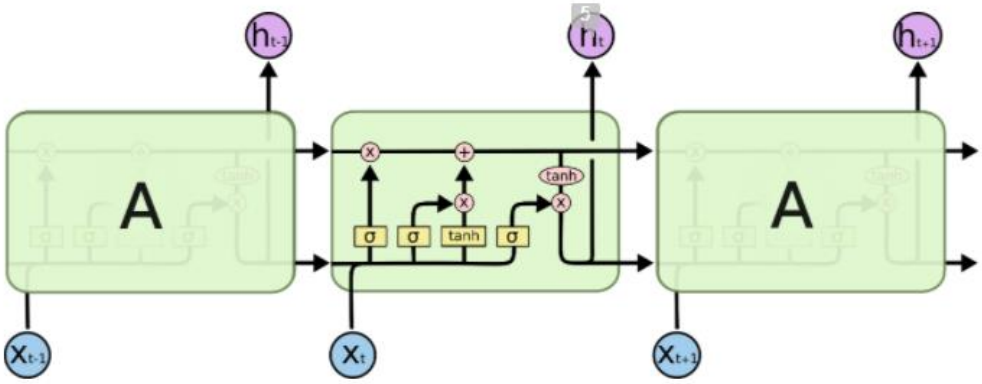
Η αναπαράσταση των προτάσεων με συνδυασμό mean και max pooling μπορεί να υλοποιηθεί ως εξής σε κώδικα python (εφαρμόζεται σε hidden states όπως ζητείται παρακάτω, αλλά η ίδια πορεία μπορεί να ακολουθηθεί και για word embeddings):

```
# Sentence representation as the final hidden state of the model
representations = torch.zeros(batch_size, self.hidden_dim).float()
mean_pooling = torch.sum(ht, dim=1)
for i in range(lengths.shape[ 0]):
     # Discard padded zeros at the end
     last = lengths[i] - 1 if lengths[i] <= max_length
     else  max_length - 1
     representations[i] = ht[i, last, :]

     # Mean pooling of outputs
     mean_pooling[i] = mean_pooling[i] / lengths[i]



# Max pooling of outputs
h = torch.transpose(ht, 1, 2)  # [N,L,C] -> [N,C,L]
m = nn.MaxPool1d(max_length)
max_pooling = m(h)
max_pooling=max_pooling.squeeze() # discard dimension that is 1

# Concatenate all three of them
representations = torch.cat((representations, mean_pooling, max_pooling),1)
```

Τα πλεονεκτήματα της εν λόγω αναπαράστασης προτάσεων μπορούν να αναδειχθούν, αν αναλογιστεί κανείς ότι μολονότι μια πρόταση μπορεί να αποτελείται από αρκετές λέξεις, λίγες εξ αυτών συμβάλλουν ουσιαστικά στην κατηγοριοποίηση της σε θετική ή αρνητική ως προς το συναίσθημα που εκφέρει, συμβάλλοντας εν τέλει καταλυτικά στην τελική πρόβλεψη του μοντέλου. Έτσι, λαμβάνοντας το μέγιστο ανα διάσταση των embeddings που απαρτίζουν μια πρόταση – κατ'αντιστοιχία με την τεχνική max-over-time pooling στα συνελικτικά Νευρωνικά Δίκτυα - ωθούμε το μοντέλο μάθησης να αγνοήσει τις λέξεις εκείνες που είναι ασήμαντες ή ασυσχέτιστες με την τελική κατηγοριοποίηση και να εστιάσει στα λεγόμενα **key words**. Συνδυάζοντας την τεχνική με αυτήν του mean pooling, στην οποία όλες οι λέξεις μιας πρότασης συνεισφέρουν ισότιμα στην τελική απόφαση, δημιουργείται μια πιο εύρωστη και αποτελεσματική αναπαράσταση προτασιακών μονάδων.

2. LSTM Recurrent Neural Network

Στο συγκεκριμένο ερώτημα, χρησιμοποιείται ένα LSTM για την κωδικοποίηση της πρότασης. Το LSTM διαβάζει τα word embeddings e_i και παράγει μια νέα αναπαράσταση για κάθε λέξη h_i, η οποία λαμβάνει υπόψη και τα συμφραζόμενα (context-sensitive). Ο μη γραμμικός μετασχηματισμός στο τελευταίο layer παραλείπεται.



Εικόνα 1: Εσωτερική Δομή LSTM ξετυλιγμένη στον χρόνο

2.1 Αναπαράσταση με τελευταία έξοδο h_N

Αρχικά, ως αναπαράσταση του κειμένου u χρησιμοποιείται η τελευταία έξοδος του LSTM h_N. Εκπαιδεύοντας το μοντέλο **LSTMTagger**, πάνω σε 15 εποχές λαμβάνουμε τα εξής αποτελέσματα ως προς την επίδοση του:

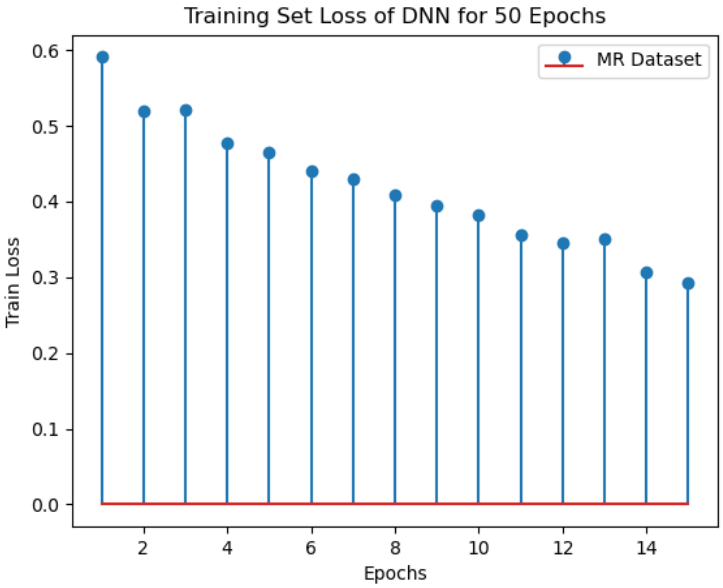
TRAIN DATASET EVALUATION:

Epoch Loss = 0.524524162339

Accuracy = 0.7377587579617835

Recall = 0.7372630930649166

F1 Score = 0.7342734281125456



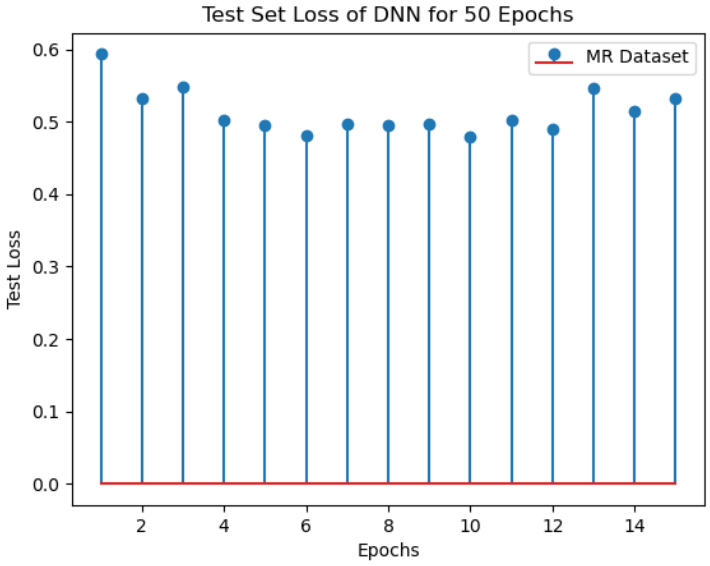
TEST DATASET EVALUATION:

Epoch Loss = 0.5367207770997827

Accuracy = 0.7395402892561983

Recall = 0.7381667410437377

F1 Score = 0.7368574113573735



2.2 Αναπαράσταση με concatenation h_N, mean(E), max(E)

Στη συνέχεια, χρησιμοποιείται ως αναπαράσταση του κειμένου u η συνένωση της τελευταίας εξόδου του LSTM h_N , του μέσου όρου των εξόδων του LSTM, και του μεγίστου ανά διάσταση των εξόδων του LSTM.

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

Σταματάμε τη διαδικασία εκπαίδευσης του μοντέλου **Concat_LSTM** στην 8η εποχή, ώστε να προλάβουμε το overfitting του μοντέλου. Λαμβάνουμε τα εξής αποτελέσματα:

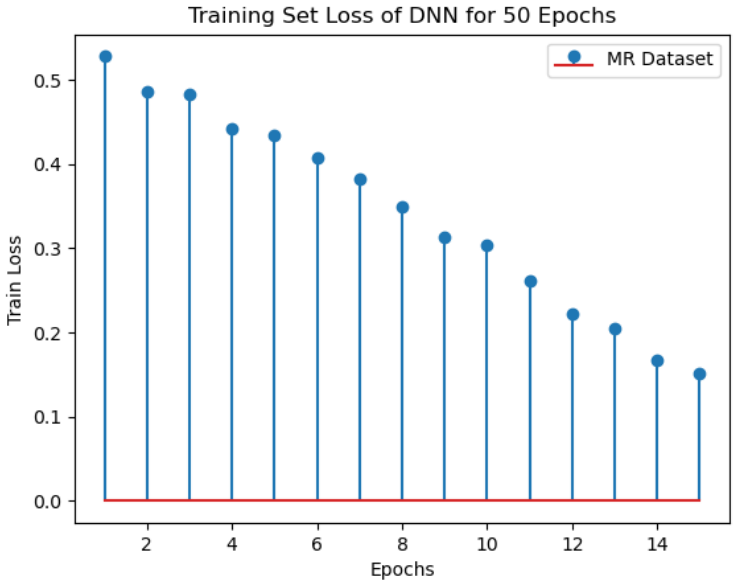
TRAIN DATASET EVALUATION:

Epoch Loss = 0.34952246127234904

Accuracy = 0.8458399681528662

Recall = 0.8452118623010606

F1 Score = 0.8429928449207502



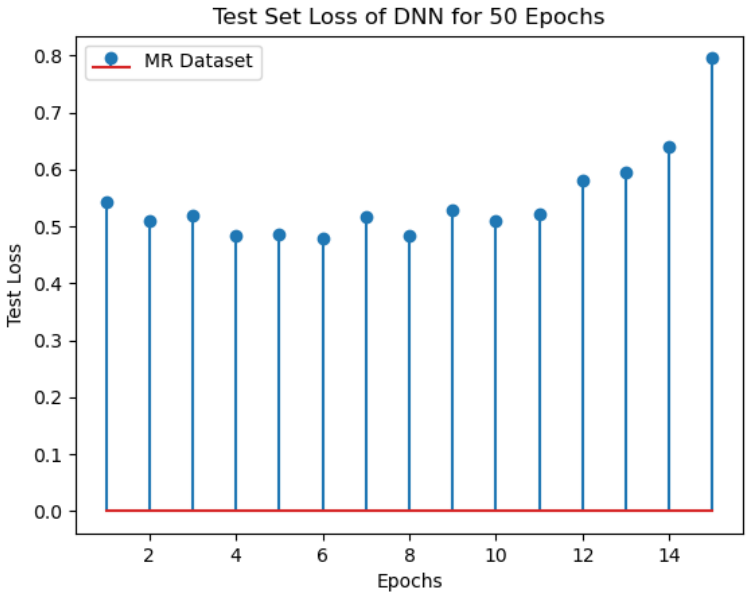
TEST DATASET EVALUATION:

Epoch Loss = 0.4840939776463942

Accuracy = 0.7731146694214875

Recall = 0.7761228692802646

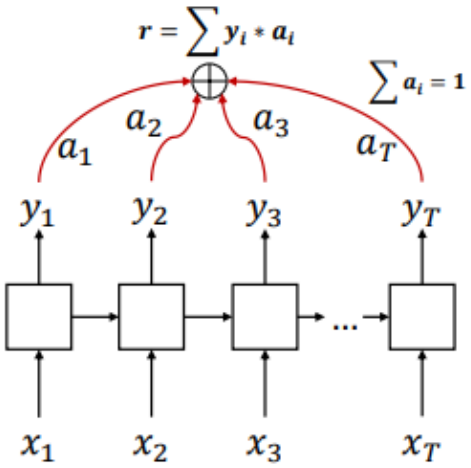
F1 Score = 0.7684382124125254



Παρατηρούμε ότι ο δεύτερος τρόπος αναπαράστασης φέρνει λίγο καλύτερα αποτελέσματα από τον πρώτο, ωστόσο απαιτεί τον τερματισμό της διαδικασίας εκπαίδευσης νωρίτερα, ειδάλλως ελλοχεύει ο κίνδυνος να επέλθει overfitting.

3. Attention Mechanism

Στο επόμενο ερώτημα, χρησιμοποιείται ένας μηχανισμός Προσοχής, με βάση την έτοιμη υλοποίηση [2], ο οποίος επιχειρεί να ενισχύσει την συνεισφορά των πιο σημαντικών στοιχείων στην τελική αναπαράσταση.



Εικόνα 2: RNN με μηχανισμό attention. Η τελική αναπαράσταση της ακολουθίας είναι το σταθμισμένο άθροισμα των ενδιάμεσων εξόδων yi.

3.1 Weighted Sum of Word Embeddings

Ως πρώτη υλοποίηση, ο μηχανισμός attention χρησιμοποιείται για την αναπαράσταση ενός κειμένου ως το σταθμισμένο άθροισμα των word embeddings:

$$v_i = \tanh(We_i + b)$$

$$\alpha_i = \frac{\exp(v_i)}{\sum_{t=1}^{N} \exp(v_t)}$$

$$u = \sum_{i=1}^{N} \alpha_i e_i$$

Εκπαιδεύοντας, λοιπόν, το μοντέλο ***LSTM_Attention*** με παράμετρο *focus* = ***'embeddings'***, λαμβάνουμε τα ακόλουθα αποτελέσματα:

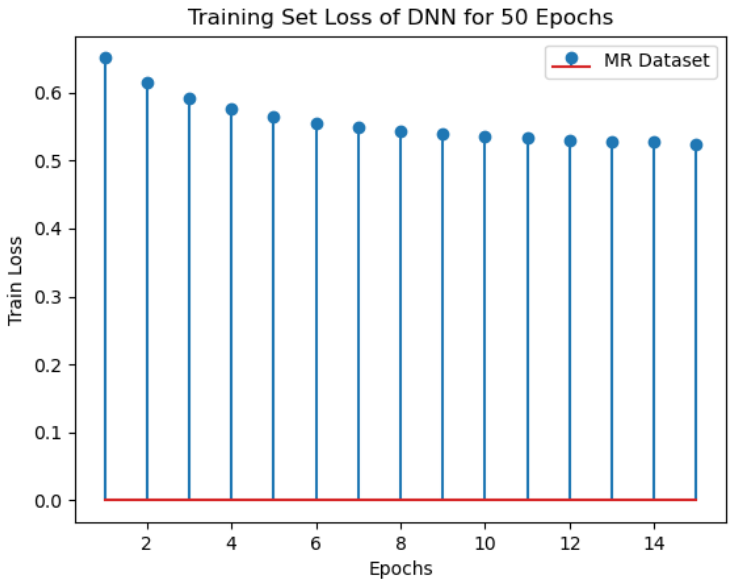
TRAIN DATASET EVALUATION:

Epoch Loss = 0.5245241623395568

Accuracy = 0.7377587579617835

Recall = 0.7372630930649166

F1 Score = 0.7342734281125456



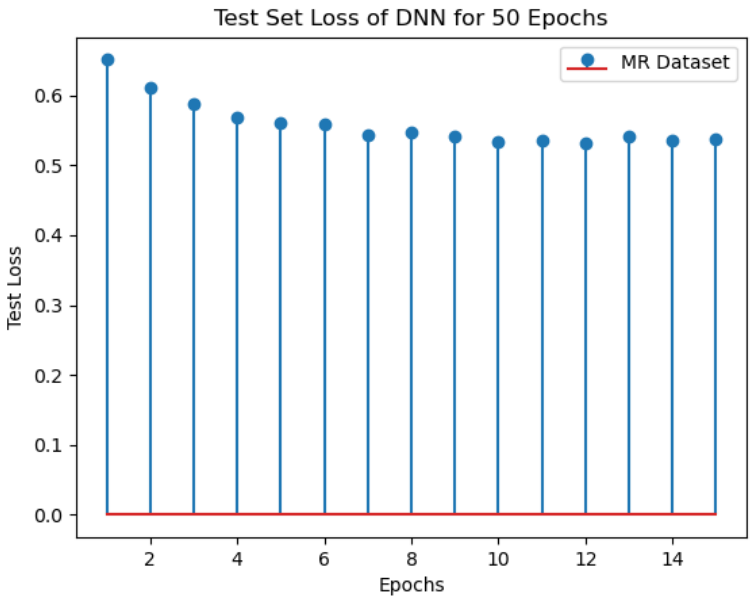
TEST DATASET EVALUATION:

Epoch Loss = 0.536720777099

Accuracy = 0.7395402892561983

Recall = 0.7381667410437377

F1 Score = 0.7368574113573735



3.2 Weighted Sum of Hidden States

Ως δεύτερη υλοποίηση, ο μηχανισμός attention χρησιμοποιείται για την αναπαράσταση ενός κειμένου ως το σταθμισμένο άθροισμα των εξόδων του LSTM:

$$v_i = \tanh(Wh_i + b)$$

$$\alpha_i = \frac{\exp(v_i)}{\sum_{t=1}^{N} \exp(v_t)}$$

$$u = \sum_{i=1}^{N} \alpha_i h_i$$

Εκπαιδεύοντας, λοιπόν, το μοντέλο ***LSTM_Attention*** με παράμετρο *focus* = ***' hidden_state'***, και τερματίζοντας τη διαδικασία εκπαίδευσης στη **10η εποχή**, ώστε να αποφευχθεί το overfitting, λαμβάνουμε τα ακόλουθα αποτελέσματα:

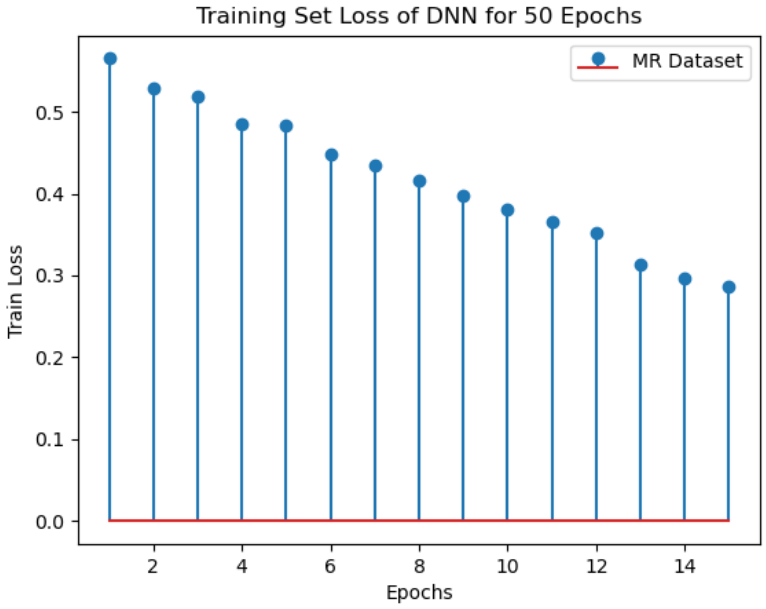
TRAIN DATASET EVALUATION:

Epoch Loss = 0.38142702932

Accuracy = 0.8285230891719745

Recall = 0.8279831472377152

F1 Score = 0.8259610197958868



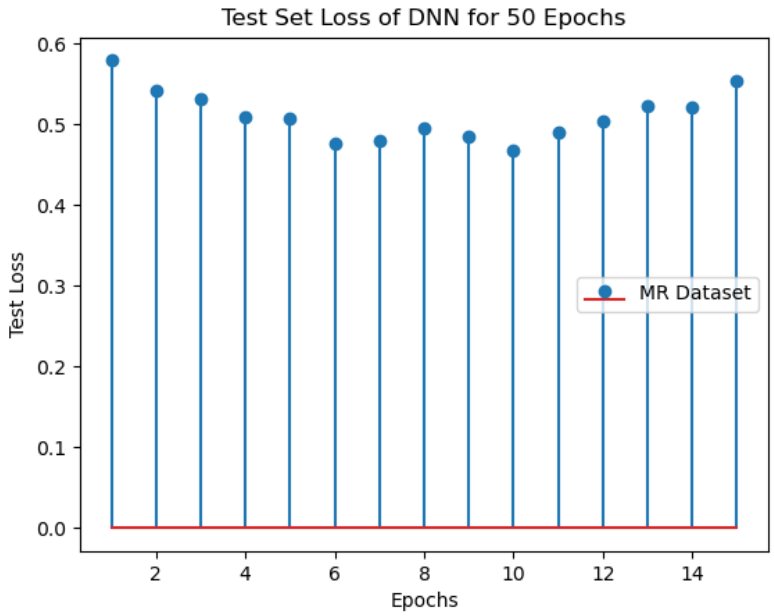
TEST DATASET EVALUATION:

Epoch Loss = 0.46816087581894

Accuracy = 0.7815082644628099

Recall = 0.7796683993171897

F1 Score = 0.7780865700668481



4. Bidirectional LSTM

Στο συγκεκριμένο μέρος της άσκησης υλοποιείται ένα αμφίδρομο RNN (bidirectional RNN), το οποίο αποτελείται από τον συνδυασμό δύο διαφορετικών RNN, όπου το κάθε ένα επεξεργάζεται την ακολουθία με διαφορετική φορά. Το κίνητρο αυτής της τεχνικής, είναι η δημιουργία μίας σύνοψης του εγγράφου και από τις δύο κατευθύνσεις, ώστε να σχηματιστεί μία καλύτερη αναπαράσταση. Έτσι έχουμε ένα δεξιόστροφο RNN → f , το οποίο διαβάζει μία πρόταση από το x_1 προς x_T και ένα αριστερόστροφο RNN ← f , το οποίο διαβάζει μία πρόταση από το x_T προς x_1.

4.1 BiLSTM with Concatenated Representation

Αρχικά, υλοποιείται ένα μοντέλο LSTM ανάλογο με αυτό του ερωτήματος 2.2, με τη διαφορά ότι πρόκειται για αμφίδρομης κατεύθυνσης. Εκπαιδεύοντας το εν λόγω μοντέλο, **Bidirectional_Concat_LSTM,** πάνω σε 10 εποχές και σταματώντας στην **6η** λαμβάνουμε τα εξής αποτελέσματα:

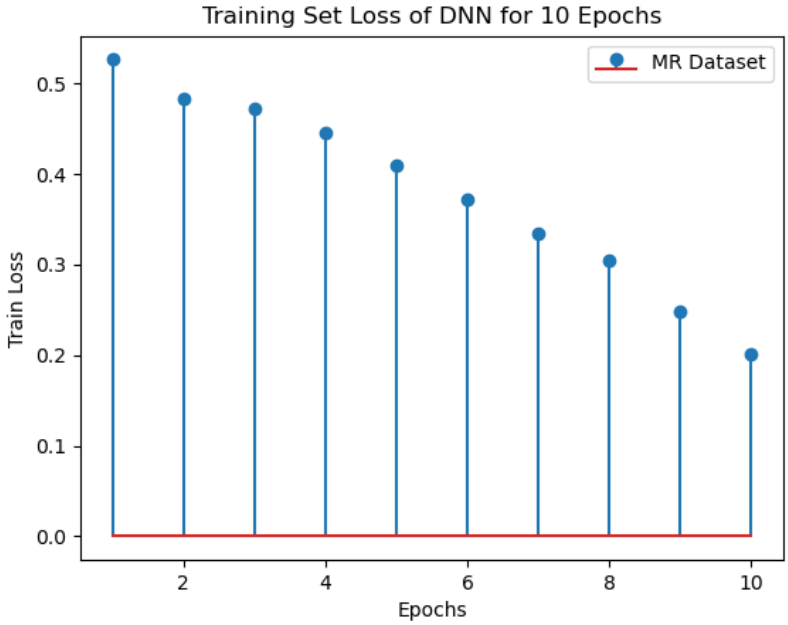
TRAIN DATASET EVALUATION:

Epoch Loss = 0.3715295002908

Accuracy = 0.8370820063694

Recall = 0.8379941913918744

F1 Score = 0.83521662943207



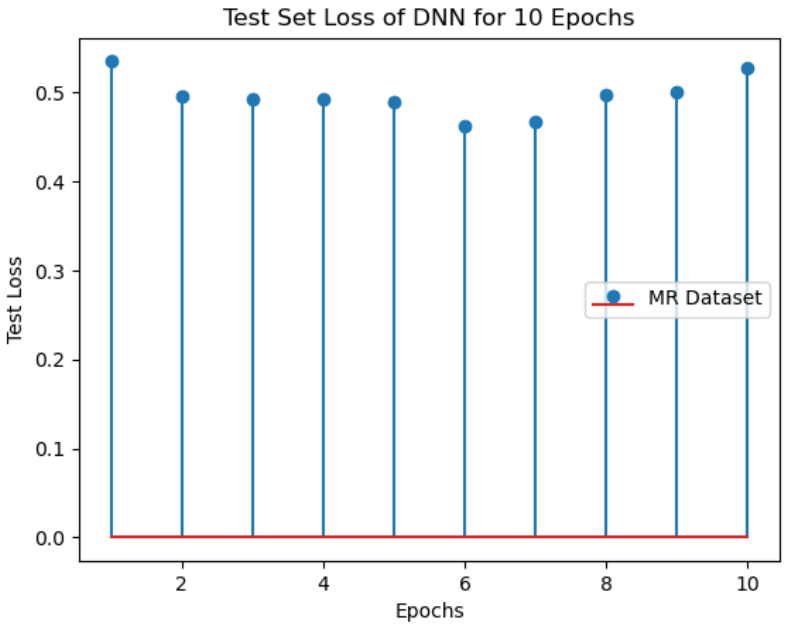
TEST DATASET EVALUATION:

Epoch Loss = 0.4628233056176

Accuracy = 0.78434917355371

Recall = 0.7840926103983912

F1 Score = 0.7815412638827



4.2 BiLSTM with Attention Mechanism

Ακολούθως, υλοποιείται ένα μοντέλο LSTM ανάλογο με αυτό του ερωτήματος 3.2, με τη διαφορά ότι πρόκειται για αμφίδρομης κατεύθυνσης. Εκπαιδεύοντας το εν λόγω μοντέλο, **Bidirectional_LSTM_Attention,** πάνω σε 11 εποχές λαμβάνουμε τα εξής αποτελέσματα:

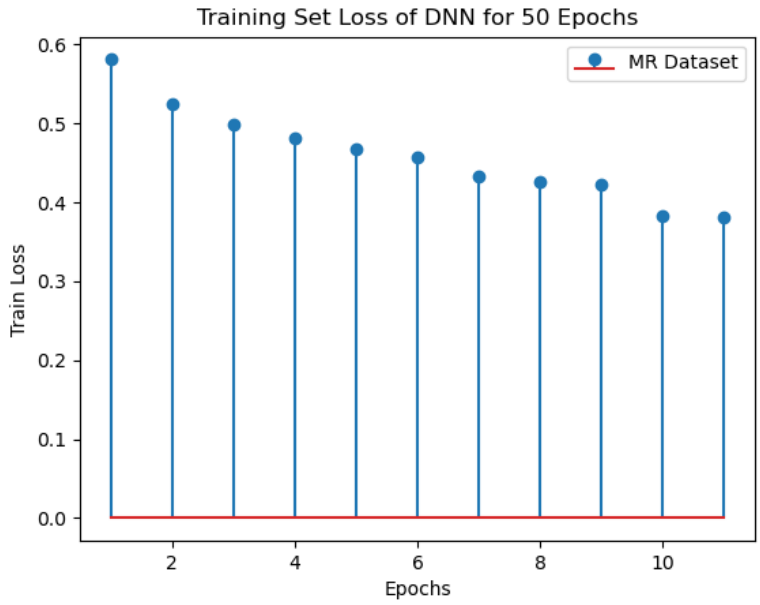
TRAIN DATASET EVALUATION:

Epoch Loss = 0.38161652601217

Accuracy = 0.8255537974683544

Recall = 0.8266886695715353

F1 Score = 0.8243247015444781



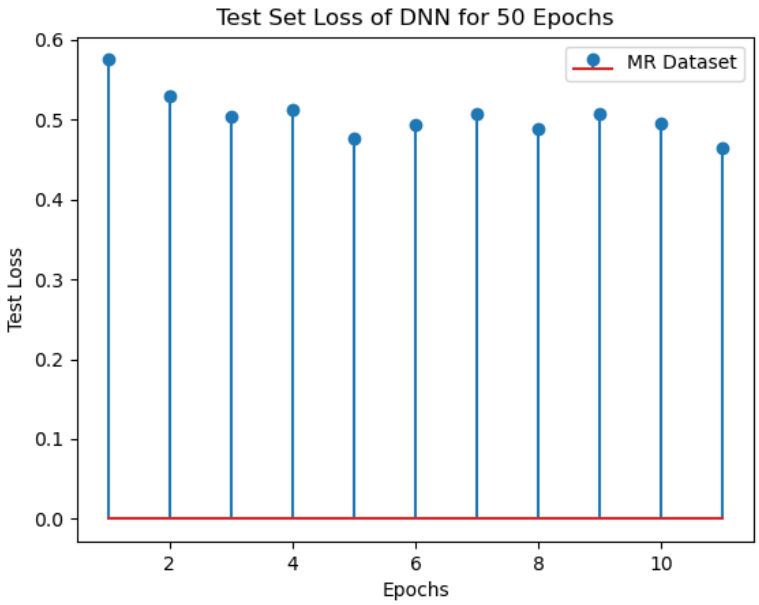
TEST DATASET EVALUATION:

Epoch Loss = 0.465206176042556

Accuracy = 0.7869318181818182

Recall = 0.7871450052809604

F1 Score = 0.7855360630975826



5. Comparison & Visualisation

5.1 Επίδοση Καλύτερου Μοντέλου

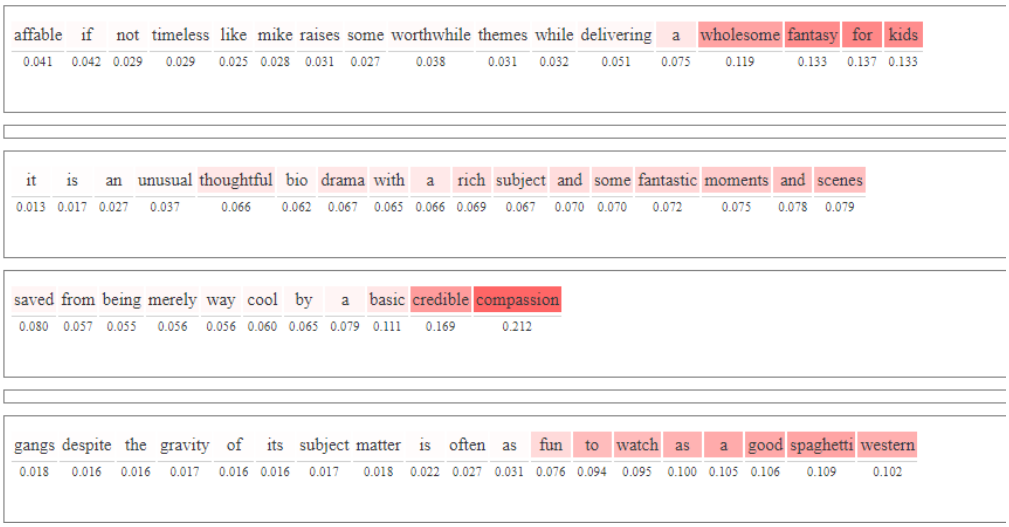
Παρατηρώντας τις επιδόσεις των παραπάνω μοντέλων στην κατηγοριοποίηση συναισθήματος από κειμενική πληροφορία, συμπεραίνει κανείς ότι το χαμηλότερο σφάλμα στο validation set επιτυγχάνεται **στις δύο τελευταίες περιπτώστεις**, δηλαδή στο LSTM Bi-RNN που χρησιμοποιεί την concatenated αναπαράσταση των embeddings (***Test Loss = 0.4628***), καθώς και στο Bi-RNN εδοφιασμένο με μηχανισμό προσοχής πάνω στις κρυφές καταστάσεις h του LSTM (***Test Loss = 0.4652***). Η ευστοχία των παραπάνω μοντέλων αγγίζει το **78%** πάνω στο validation set.

Εστιάζοντας στο ***Attention Bi-LSTM*** του ερωτήματος (4.2), αποθηκεύουμε ένα **checkpoint** του μοντέλου που εκπαιδεύσαμε πάνω σε 11 εποχές ως **''best_model''**. Στη συνέχεια, με τη χρήση του script *'visualise.py'* αποθηκεύουμε τις προβλέψεις του εν λόγω μοντέλου πάνω στο validation set (δίχως να εφαρμόσουμε shuffle) σε ένα αρχείο με το όνομα ***'best_predictions.txt'***. Το αρχείο αυτό περιέχει μια ακολουθία από labels (0 and 1) που αντιστοιχούν στα predicted classifications του μοντέλου πάνω στις προτάσεις του validation set (0: negative, 1: positive).

5.2 Οπτικοποίηση Καλύτερου Μοντέλου

Ακολούθως, τροποποιώντας τη συνάρτηση forward του καλύτερου μοντέλου ώστε να επιστρέφει και τα attention scores του τελευταίου layer, καθώς και το script 'training.py' ώστε να συλλέξουμε τα εν λόγω scores, επιδιώκουμε να οπτικοποιήσουμε τα βάρη των κατανομών του Μηχανισμού Προσοχής, με τη χρήση του online εργαλείου **NeAt-vision** [3].

Πιο συγκεκριμένα, με τη χρήση του python script 'visualise.py' δημιουργούμε 2 .json αρχεία, data.json και label.json, τα οποία αποθηκεύουμε στο directory ./json. Παρακάτω φαίνονται οι κατανομές των βαρών του μηχανισμού προσοχής σε μερικά παραδείγματα προτάσεων του validation set, όπως αυτές απεικονίστηκαν με χρήση του NeAt-vision:



Εικόνα 3: Κατανομές Μηχανισμού Προσοχής best model

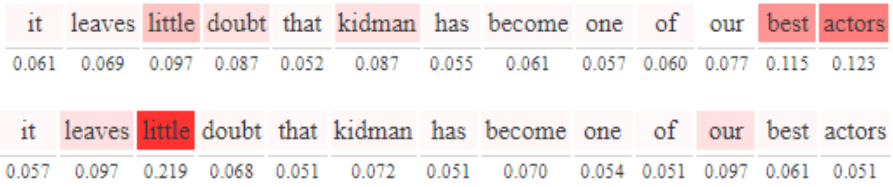
5.3 Οπτικοποίηση Μηχανισμών Προσοχής

Χρησιμοποιώντας περαιτέρω το εργαλείο οπτικοποίησης, μπορούμε να κατανοήσουμε καλύτερα γιατί το μοντέλο με μηχανισμό προσοχής πάνω στις κρυφές καταστάσεις h_i του LSTM, φαίνεται να αποδίδει καλύτερα από το μοντέλο με Μηχανισμό Προσοχής πάνω στα word embeddings e_i.

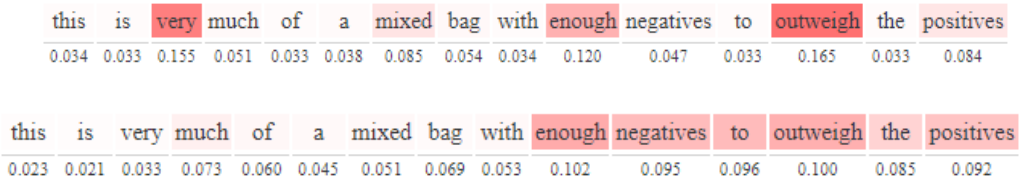
Πιο συγκεκριμένα, εφαρμόζοντας τον Μηχανισμό Προσοχής πάνω στα word embeddings e_i, το μοντέλο φαίνεται να εστιάζει τα βάρη των κατανομών προσοχής κυρίως σε **λέξεις κλειδιά**, οι οποίες έχουν εμφανώς αρνητική ή θετική χροιά. Κάτι τέτοιο, όμως, συχνά **αποτυγχάνει να κωδικοποιήση την ειρωνική ή μεταφορική χρήση των λέξεων** από τους χρήστες, ωθώντας συχνά το μοντέλο σε misclassifications. Αντιθέτως, εφαρμόζοντας τον Μηχανισμό Προσοχής στις κρυφές καταστάσεις h_i, το μοντέλο εξακολουθεί να εστιάζει σε λέξεις – κλειδιά των προτάσεων, ωστόσο λαμβάνει περισσότερο υπόψη το **context** μέσα στο οποίο αυτές

χρησιμοποιούνται (**content-based addressing**), προσδίδοντας κατάλληλη σημασία και σε γειτονικές τους λέξεις.

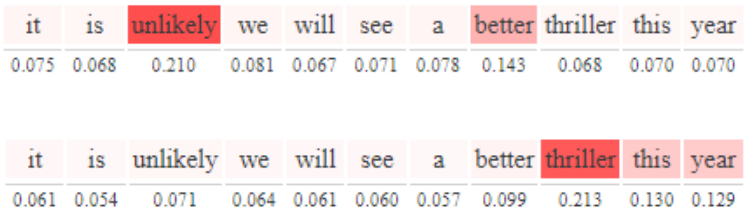
Παρατίθενται τα εξής ενδεικτικά παραδείγματα οπτικοποίησης για το 1ο και το 2ο μοντέλο αντίστοιχα, πάνω στις ίδιες προτάσεις προς classification, προκειμένου να γίνει περισσότερο αντιληπτή η παραπάνω διαφορά:



Στο εν λόγω παράδειγμα, ο Embedding-based Attention Mechanism εστιάζει την προσοχή του πάνω στην λέξη '**little'**, η οποία έχει αρνητική χροιά και κατηγοριοποιεί εσφαλμένα την παραπάνω πρόταση ως negative. Αντιθέτως, ο HiddenState-based Attention Mechanism λαμβάνει υπόψη και το υπόλοιπο context της πρότασης ('**little DOUBT'** και '**best actors**') αποκρυσταλλώνοντας τον θετικό χαρακτήρα της πρότασης.



Στο εν λόγω παράδειγμα, ο Embedding-based Attention Mechanism εστιάζει την προσοχή του πάνω στις λέξεις '**very'**, '**enough'**, '**outweigh'** και '**positive'**, οι οποίες έχουν θετική χροιά και κατηγοριοποιεί εσφαλμένα την παραπάνω πρόταση ως positive. Αντιθέτως, ο HiddenState-based Attention Mechanism εστιάζει στο context ολόκληρης της φράσης '**enough negatives to outweigh the positives**' και ορθώς κατατάσσει την πρόταση ως negative.



Ομοίως, ο Embedding-based Attention Mechanism εστιάζει σχεδόν αποκλειστικά την προσοχή του πάνω στη λέξη '**unlikely'**, η οποία έχει αρνητική χροιά και κατηγοριοποιεί εσφαλμένα την παραπάνω πρόταση ως negative. Αντιθέτως, ο HiddenState-based Attention

Mechanism εστιάζει στο **context** ολόκληρης της φράσης και ορθώς κατατάσσει την πρόταση ως positive.

6. Bag-of-Words Characteristics

Προκειμένου να ρυθμίσουμε τον τρόπο με τον οποίο ένα μοντέλο Μηχανικής Μάθησης εστιάζει σε λέξεις των προτάσεων που κατηγοριοποιεί με βάση τη συχνότητα εμφάνισης αυτών των λέξεων, θα μπορούσε κανείς να συνδυάσει τις αναπαραστάσεις που εξάγει το Νευρωνικό Δίκτυο με **Bag-of-Words (BoW) ή tf-idf αναπαραστάσεις**. Στο συγκεκριμένο μέρος της αναφοράς, εξετάζεται η επίδραση των tf-idf χαρακτηριστικών στο μοντέλο με τις καλύτερες επιδόσεις (Bidirectional RNN with Attention Mechanism).

Σύμφωνα με την [4], τα tf-idf χαρακτηριστικά χρησιμοποιούνται προκειμένου να προσδίδεται περισσότερο έμφαση σε λέξεις που εμφανίζονται συχνά, αλλά όχι υπερβολικά συχνά (όπως πχ ‘the’, ‘and’ κλπ). Για τον σκοπό αυτό, ορίζουμε τα εξής μεγέθη:

$$tf_{i,d} = \log_{10}(count(t,d) + 1)$$

$$idf_t = \log_{10}(\frac{N}{df_t})$$

$$w_{t,d} = tf_{i,d} \times idf_t$$

Όπου:

$\boldsymbol{tf_{i,d}}$: *η συχνότητα εμφάνισης της λέξης t στο document d (term frequency)*

$\boldsymbol{N}$: *πλήθος των documents*

$\boldsymbol{df_t}$: *πλήθος των documents στα οποία εμφανίζεται η λέξη t*

$\boldsymbol{idf_t}$: *Inverse Document frequency*

$\boldsymbol{w_{t,d}}$: *το $tf - idf$ βάρος της λέξης t στο document d*

6.1 Υλοποίηση tf-idf Vectorizer

Χρησιμοποιώντας την βιβλιοθήκη scikit-learn της python, υλοποιούμε έναν **tf-idf Vectorizer** στην κλάση **SentenceDataset** του script ‘*dataloading.py*’ ως εξής:

```
# 6.1 Implement tf-idf vectorizer of dataset
self.vectorizer = TfidfVectorizer()
self.vectorizer.fit_transform(X)
```

Έχοντας τα tf-idf χαρακτηριστικά κάθε λέξης του corpus, τα ενσωματώνουμε στις αναπαραστάσεις του Νευρωνικού Δικτύου. Για κάθε πρόταση $(t_1 t_2 \dots t_N)$, στην οποία η λέξη t_i έχει διανυσματική αναπαράσταση e_i και tf-idf βάρος w_i, επιλέγουμε ως νέα διανυσματική αναπαράσταση τον εξής κανονικοποιημένο διανυσματικό όρο:

$$e_i' = \frac{w_i}{w_1 + \cdots + w_n} e_i$$

Η αναπαράσταση αυτή υλοποιείται σε γλώσσα python και ενσωματώνεται στη μέθοδο forward του Νευρωνικού Δικτύου:

```
# Tf-Idf weights on the word embeddings
        shape = embeddings.shape
        for i in range(shape[0]):
            denominator = 0
            for j in range(shape[1]):
                embeddings[i][j] = tfidf[i][j]*embeddings[i][j]
                denominator += tfidf[i][j]
            embeddings[i][:] = embeddings[i][:]/denominator
```

Εκπαιδεύοντας εκ νέου πάνω σε 10 εποχές το μοντέλο με τις καλύτερες επιδόσεις, εμπλουτισμένο με tf-idf χαρακτηριστικά, προκύπτουν τα εξής αποτελέσματα:

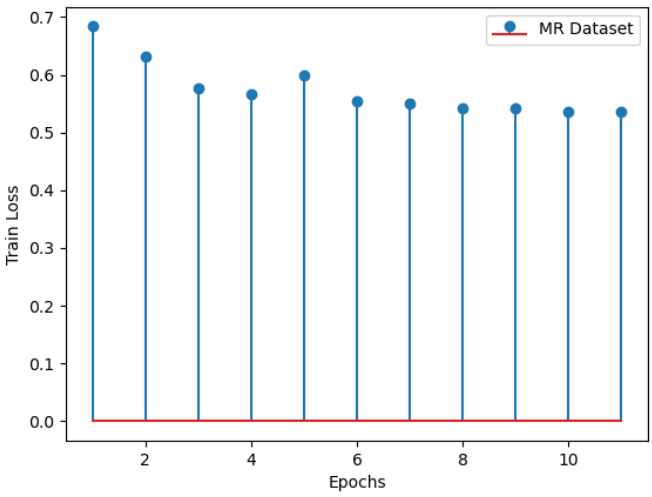
TRAIN DATASET EVALUATION:

Epoch Loss = 0.5350972497010533

Accuracy = 0.7217167721518988

Recall = 0.7216115479622398

F1 Score = 0.7177162996354635



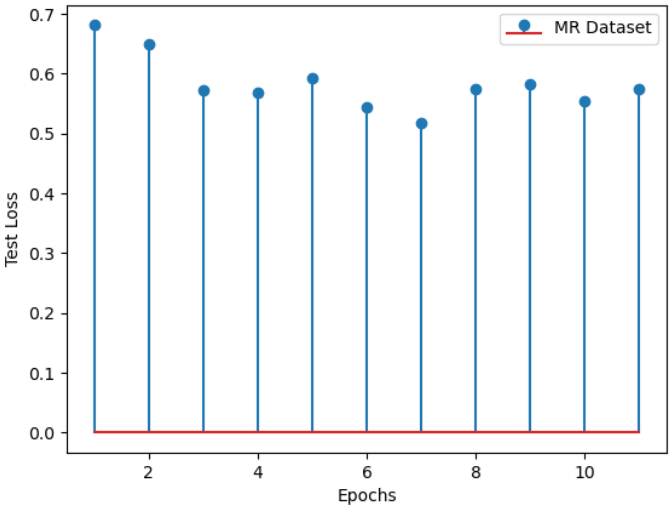
TEST DATASET EVALUATION:

Epoch Loss = 0.5734111269315084

Accuracy = 0.7013494318181818

Recall = 0.7036320791872052

F1 Score = 0.69703064241169



6.2 Συμπεράσματα – Επιδόσεις tf-idf Χαρακτηριστικών

Συνεπώς, παρατηρεί κανείς ότι εμπλουτίζοντας το Νευρωνικό Δίκτυο με tf-idf χαρακτηριστικά, όχι μόνο δεν βελτιώνονται οι επιδόσεις του, αλλά **χειροτερεύουν** καθώς το σφάλμα κατηγοριοποίησης στο validation set **αυξάνεται** (από 0.46 σε 0.57). Αυτό συμβαίνει διότι τα BoW/tf-idf χαρκατηριστικά αγνοούν σημαντική πληροφορία των κειμένων προς κατάταξη, όπως είναι η σειρά των λέξεων και οι αρνήσεις. Επιπροσθέτως, τα χαρακτηριστικά αυτά αδιαφορούν για τη σημασία και την ομοιότητα των λέξεων, ενώ εστιάζουν αποκλειστικά στη συχνότητα εμφάνισης τους. Ως αποτέλεσμα, χρησιμοποιώντας χαρακτηριστικά BoW σε κείμενα με πολύ μεγάλο λεξιλόγιο, η επίδοση των μοντέλων ταξινόμησης επιδεινώνεται.

Παρόλα αυτά, υπάρχουν ορισμένες περιπτώσεις στις οποίες τα BoW χαρακτηριστικά θα μπορούσαν να οδηγήσουν σε καλύτερα αποτελέσματα σε σχέση με αναπαραστάσεις όπως ο μέσος όρος των word embeddings. Πιο συγκεκριμένα:

- Στη δημιουργία **baseline models**. Χρησιμοποιώντας έτοιμες συναρτήσεις της scikit-learn, μπορεί κανείς να δημιουργήσει πρώιμα μοντέλα κατηγοριοποίησης κειμένων σε λίγες γραμμές κώδικα.
- Στη περίπτωση που **το dataset είναι μικρού μεγέθους** και το **context** των κειμένων προς κατηγοριοποίηση είναι **domain specific**. Για την κατηγοριοποίηση κειμένων με εξειδικευμένη ορολογία και domain specific context, δύσκολα μπορεί κανείς να βρει προεκπαιδευμένα word-embeddings (GloVe, fastText κλπ), γάυτό και ενδείκνυται η αναπαράσταση με χαρακτηριστικά tf-idf.

References

[1] [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.

[2] [Graves, 2013] Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. arXiv:1308.0850 [cs]. 00570.

[3] https://github.com/cbaziotis/neat-vision

[4] Speech and Language Processing (3rd ed. draft) - Dan Jurafsky and James H. Martin