Τεχνητή Νοημοσύνη 2

Εργασία 2

Δημητρακόπουλος Κωνσταντίνος ΑΜ: 1115201500034

Άσκηση 1

$$
\begin{aligned}
CE(y,\hat{y}) &= \sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log \left(\hat{p_k}^{(i)}\right) \\
&= \sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log \left(\frac{e^{\theta_k^{(i)}}}{\sum_{j=1}^{K} e^{\theta_j^{(i)}}}\right) \\
&= \sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \left[\log \left(e^{\theta_k^{(i)}}\right) - \log \left(\sum_{j=1}^{K} e^{\theta_j^{(i)}}\right)\right] \\
&= \sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \left[\Theta_k^{(i)} - \log \left(\sum_{j=1}^{K} e^{\theta_j^{(i)}}\right)\right] \\
\frac{\mathrm{d}CE(y,\hat{y})}{\mathrm{d}\theta_k} &= \sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \left[\frac{\mathrm{d}\Theta_k^{(i)}}{\mathrm{d}\Theta_k} - \frac{\mathrm{d}\log \left(\sum_{j=1}^{K} e^{\theta_j^{(i)}}\right)}{\mathrm{d}\theta_k}\right] \\
&= \sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \left[diag(1) - \frac{e^{\theta_k^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^{(i)}}}\right] \\
&= \sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \left[diag(1) - \hat{y}^{(i)}\right] \\
&= y(diag(1) - \hat{y}) \\
&= \hat{y} - y
\end{aligned}
$$

Άσκηση 2

$$\frac{\mathrm{d}mull}{\mathrm{d}x} = m$$
$$\frac{\mathrm{d}mull}{\mathrm{d}m} = \mathrm{x}$$
$$\frac{\mathrm{d}plusll}{\mathrm{d}mul} = 1$$
$$\frac{\mathrm{d}plusl}{\mathrm{d}b} = 1$$
$$\frac{\mathrm{d}RELUl}{\mathrm{d}a} = a > 0\,?\,1 : 0$$
$$\frac{\mathrm{d}Lossl}{\mathrm{d}y^*} = \frac{\mathrm{d}(y-y^*)^2}{\mathrm{d}y^*} = -2(y-y^*)$$
$$\frac{\mathrm{d}Lossl}{\mathrm{d}y} = \frac{\mathrm{d}(y-y^*)^2}{\mathrm{d}y^*} = 2(y-y^*)$$
$$\frac{\mathrm{d}Loss}{\mathrm{d}Loss} = 1$$
$$\frac{\mathrm{d}Loss}{\mathrm{d}y} = \frac{\mathrm{d}Loss}{\mathrm{d}Loss}\frac{\mathrm{d}Lossl}{\mathrm{d}y} = 2(y-y^*)$$
$$\frac{\mathrm{d}Loss}{\mathrm{d}y^*} = \frac{\mathrm{d}Loss}{\mathrm{d}Loss}\frac{\mathrm{d}Lossl}{\mathrm{d}y^*} = -2(y-y^*)$$
$$\frac{\mathrm{d}Loss}{\mathrm{d}a} = \frac{\mathrm{d}Loss}{\mathrm{d}y}\frac{\mathrm{d}RELUl}{\mathrm{d}a} = 2(y-y^*)(a>0?1:0)$$
$$\frac{\mathrm{d}Loss}{\mathrm{d}mul} = \frac{\mathrm{d}Loss}{\mathrm{d}a}\frac{\mathrm{d}plusl}{\mathrm{d}mul} = 2(y-y^*)(a>0?1:0)$$
$$\frac{\mathrm{d}Loss}{\mathrm{d}b} = \frac{\mathrm{d}Loss}{\mathrm{d}a}\frac{\mathrm{d}plusl}{\mathrm{d}b} = 2(y-y^*)(a>0?1:0)$$
$$\frac{\mathrm{d}Loss}{\mathrm{d}x} = \frac{\mathrm{d}Loss}{\mathrm{d}mul}\frac{\mathrm{d}mull}{\mathrm{d}x} = 2(y-y^*)(a>0?1:0)m$$
$$\frac{\mathrm{d}Loss}{\mathrm{d}m} = \frac{\mathrm{d}Loss}{\mathrm{d}mul}\frac{\mathrm{d}mul}{\mathrm{d}m} = 2(y-y^*)(a>0?1:0)x$$

Άσκηση 3

Παρακάτω αναλύεται η στρατηγική που ακολουθήθηκε κατά τη σχεδίαση των τελικών μοντέλων.

Προεπεξεργασία των δεδομένων

Δοκιμάστηκαν οι παρακάτω μέθοδοι:

1. Αφαίρεση συνδέσμων.
2. Αφαίρεση hashtags.
3. Αφαίρεση mentions.
4. Αντικατάσταση συνδέσμων σε <link>.
5. Αντικατάσταση hashtags σε <hashtag>.
6. Αντικατάσταση mentions σε <mention>.
7. Αντικατάσταση αριθμών σε <number>.
8. Αντικατάσταση κεφαλαίων χαρακτήρων σε <upper>.
9. Διατήρηση μόνο των αλφαριθμητικών χαρακτήρων.
10. Μετατροπή σε πεζά γράμματα.
11. Lemmatizing.
12. Αφαίρεση stop words.

Οι περιπτώσεις 4 έως 8 συνεισφέρουν στη διανυσματοποίηση των οντοτήτων που περιγράφουν από τα glove twitter pretrained embeddings που χρησιμοποιήθηκαν για ελέγχους κατά τη σχεδίαση του μοντέλου 2.

Μετά από έλεγχο των μεθόδων στα τελικά μοντέλα (κατά την κατασκευή τους και μετά την ολοκλήρωση τους), παρατηρήθηκε πως οι μέθοδοι που βελτιστοποιούν τα αποτελέσματα είναι οι 10-12.

- Οι μέθοδοι 4-8 δεν αξιοποιήθηκαν αφού τελικά δεν χρησιμοποιήθηκαν τα **twitter** pretrained embeddings.
- Τα σημεία στίξης δεν συνεισφέρουν στη καλή ταξινόμηση για τα συγκεκριμένα μοντέλα - σετ δεδομένων, αφού η σωστή αντιστοίχιση τους στα pre trained embeddings έφερε μείωση στα αποτελέσματα των μετρικών (f1=0.61 στα περισσότερα test).

Αριθμός των Hidden Layers και Units

Ο αρχικός έλεγχος πραγματοποιήθηκε με μηδενικό αριθμό από hidden layers με τα αποτελέσματα της f1 μετρικής να είναι 0.45.

Κατά την εισαγωγή του πρώτου hidden layer και με σταδιακή αύξηση των unit που το αποτελούν τα αποτελέσματα εμφανίζονταν βελτιωμένα.

Όταν το πρώτο hidden layer αποτελούταν από πάρα πολλά units υπήρξε μείωση στην ταχύτητα training και σταθεροποίηση των αποτελεσμάτων των μετρικών.

Τότε η εισαγωγή ενός δεύτερου hidden layer και ο διαμοιρασμός του συνολικού αριθμού από units έκαναν το μοντέλο περισσότερο πολύπλοκο, γρηγορότερο και ικανότερο να αντιμετωπίσει το σύνθετο πρόβλημα.

Η διαδικασία αυτή συνεχίστηκε με πειράματα μέχρις ότου το μοντέλο να ήταν αρκετά πολύπλοκο για να περιγράψει το πρόβλημα και την εμφάνιση του φαινομένου overfitting.

Τελικά διατηρήθηκαν δύο layers [1024,512] και [512,3], πέρα του input, που διατηρούν μια καλή ισορροπία στην χρονική απόδοση και στα αποτελέσματα των μετρικών για τα δύο μοντέλα.

Επιλογή Loss Function

Τα δύο μοντέλα ελέγθηκαν με τις multi-class classification Loss functions **Cross Entropy** και **Negative Log Likelihood**.

Υπόλοιπες loss functions όπως L1, MSE, Quadratic Loss, Mean Absolute Error, Mean Bias Error κ.α. χρησιμοποιούνται σε προβλήματα **regression** που είναι διαφορετικού σκοπού.

Scheduled Learning Rate

Στη πλειοψηφεία των δοκιμών που αναφέρονται χρησιμοποιήθηκαν πολλαπλές τιμές ως learning rates. Στις περιπτώσεις όπου το learning rate είναι μεγάλο η σύγκλιση προς το επιθυμητό αποτέλεσμα επιταχύνεται αλλά δεν σταθεροποιείται σε ένα καλό score. Αντίθετα, στις περιπτώσεις όπου το learning rate είναι πολύ μικρό η σύγκλιση είναι αργή και πολλές φορές το τελικό αποτέλεσμα δεν είναι το αναμενόμενο.

Τελικά χρησιμοποιήθηκαν lr schedulers με σκοπό η τιμή του learning rate να μειώνεται σταδιακά για την καλύτερη σύγκλιση. Συγκεκριμένα έγιναν δοκιμές με τους **exponential learning rate** (με παράγοντα 0.9) και **step learning rate** (ανά 5 epochs πολλαπλασιασμός με το παράγοντα 0.1).

Μοντέλο 1

Δοκιμές κατά τη δημιουργία

Αρχικά χρησιμοποιήθηκε ο TF-IDF Vectorizer για τη διανυσματοποίηση των προεπεξεργασμένων δεδομένων, με παραμέτρους min_df=0.01, max_df=0.4, ngrams=(1,5), οι οποίοι συνεισφέρουν στη μείωση του θορύβου στα δεδομένα και βελτιστοποιούν τα αποτελέσματα των μετρικών (όπως στην εργασία 1).

Παρακάτω φαίνονται δοκιμές με διάφορες υπερπαραμέτρους και optimizers, activation και loss functions για το μοντέλο 1, οι οποίες εφαρμόστηκαν στο νευρωνικό δίκτυο της παραπάνω ενότητας.

SGD με χρήση Momentum=0.5

→ Loss function: Cross entropy

→ Activation Function: ReLU μετά από κάθε layer

Παρουσιάζεται έντονο φαινόμενο overfit.

→ Γίνεται χρήση dropout όπου μειώνει σημαντικά αλλά όχι ολοκληρωτικά το overfit που παρουσιάστηκε.

→ Δοκιμάζεται l2 regularization για τις τιμές [1e-4,1e-3,1e-2], μέσω της παραμέτρου weight_decay όπου συνεισφέρει στην αντιμετώπιση του overfitting.

→ Δοκιμάζονται οι activation funtions

- SELU (επιθυμητή σύγκλιση με score f1=0.67),
- ELU (score=0.67),
- LeakyRelu (score=0.67),
- RRELU (score=0.67).

Χρήση της loss function NLLLoss

→ Εισαγωγή της activation function softMax()

→ Εμφάνιση φαινομένου vanishing gradients και προσπάθεια απόριψης του με χρήση batch normalization στα output των layers.

→ Χρήση sgd loss function χωρίς momentum (score=0.67)

→ Πειραματισμός με διάφορες τιμές για την παράμετρο momentum.

- momentum = 0.9 -> f1 score = 0.67
- momentum = 0.7 -> f1 score = 0.67
- momentum = 0.3 -> f1 score = 0.66

Υπόλοιποι optimizers

→ Χρήση του optimizer **nesterov momentum**
→ Εμφάνιση ελάχιστου overfitting με f1 score = 0.67
→ Χρήση του optimizer **adagrad** με f1 score 0.67
→ Χρήση του optimizer **rmsprop** με αποτέλεσμα εμφάνιση overfitting και μεταβολή της παραμέτρου alpha χωρίς βελτίωση των αποτελεσμάτων.
→ Χρήση **dropout**, για αποφυγή του overfitting, χωρίς τη βελτίωση του μοντέλου.
→ Χρήση του optimizer **adam** με f1 score = 0.67.
→ Μεταβολή των παραμέτρων betas του optimizer με αποτέλεσμα τη διατήρηση και σταδιακή μείωση των αποτελεσμάτων των μετρικών.
→ Χρήση της παραμέτρου amsgrad=true του optimizer **adam** με f1 score = 0.67
→ Χρήση του optimizer **adamax** με f1 score = 0.67
→ Μεταβολή των betas του optimizer με αποτέλεσμα τη διατήρηση και σταδιακή μείωση των αποτελεσμάτων των μετρικών.
→ Χρήση του optimizer **nadam** με f1 score = 0.64
→ Μεταβολή της παραμέτρου momentum_decay=4e-1 με f1 score = 0.67

Το τελικό μοντέλο

Το τελικό μοντέλο απαρτίζεται από:

- **Layers**: [input,1024] , [1024,512] , [512,3]
- **Activation function**: ELU και Dropout στην έξοδο κάθε layer
- **Loss function**: Cross Entropy
- **Optimizer**: Stochastic Gradient Descent με χρήση Momentum
- **LR scheduler**: Exponential Learning Rate
- **Vectorizer**: TF-IDF
- **Batch size**: 32
- **Epochs**: 12

Pre Trained Embeddings

Πραγματοποιήθηκαν δοκιμές με τα twitter glove pre trained embeddings (διαστάσεις 25, 50, 100, 200) και με τα glove pre trained embeddings (διαστάσεις 50, 100, 200, 300). Εμφανώς καλύτερα αποτελέσματα παρουσίασαν τα **glove pre trained embeddings διάστασης 300**. Για την καλύτερη εξέταση των νέων διανυσμάτων του input δημιουργήθηκαν οι βοηθητικές συναρτήσεις check_model, check_model2 όπου τυπώνουν το πλήθος και τα tokens του σετ δεδομένων που βρέθηκαν στα pre trained embeddings και όχι.

Μοντέλο 2

Δοκιμές κατά τη δημιουργία

Εξετάζοντας το σετ pre trained embeddings με διάσταση 50:

- Χρήση optimizer sgd με activation function:
  - → sigmoid εμφανίζει vanishing grandients
  - → Tanh με χαμηλό f1 score = 0.59
  - → LeakyRelu με f1 score = 0.61 overfitting

Τελικά δημιουργήθηκε ο πίνακας διανυσμάτων του σετ δεδομένων από τα glove pre trained embeddings διάστασης 300 όπου εμφανίζουν το καλύτερο score. Συγκεκριμένα κάθε tweet αποτελείται από το μέσο διάνυσμα των διανυσμάτων των tokens του.

- Με μία εκτέλεση (sgd,relu,ce) είναι εμφανές το έντονο φαινόμενο overfitting.
  - → Για την αποφυγή του χρησιμοποιείται l2 regularization μέσω weight_decay, το οποίο εμφανίζει νέο φαινόμενο vanishing gradients.
  - → Δοκιμάστηκαν οι τιμές (0.6 , 0.5 , 0.7) στο activation function dropout για την αποφυγή του φαινομένου overfitting. Εμφανίζονται υψηλά f1 score (0.67 , 0.68 , 0.71) με το φαινόμενο του overfitting να παραμένει ενεργό. Κατά την τιμή 0.75 φαίνεται τελικό f1 score = 0.67 με αρκετά μειωμένο overfitting.
  - → Χρησιμοποιώντας κατάλληλα τις τιμές 1e-4, 1e-3, 1e-2 στην παράμετρο weight_decay παρατηρούμε τη διατήρηση του φαινομένου.
  - → Ο συνδυασμός dropout με πιθανότητα 0.75 και l2 regularization 1e-4 ή 1e-3 εμφανίζει f1 score 0.68 πριν την αρχή του φαινομένου overfitting.
  - → Χρήση relu -> f1 score = 0.69

    - → Χρήση selu -> Παρατηρούμε vanishing gradients, μειώνουμε το dropout ενώ εισάγεται batch normalization με αποτέλεσμα έντονο το φαινόμενο overfit.
    - → Χρήση elu -> f1 score = 0.67 με καλή σύγκλιση ενώ δεν υπάρχει αύξηση/μείωση του score/loss ανα epoch.
- Με χρήση του optimizer nesterov momentum (momentum = 0.7) παράγεται f1 score = 0.68
- Με χρήση του optimizer adagrad
    - → παράγεται f1 score = 0.65 με καλή σύγκλιση
    - → αφαιρώντας το dropout παρουσιάζεται overfit.
    - → Με χρήση l2 regularization για την αντιμετώπιση του overfit παράγεται f1 score = 0.68 με αργό training
- Με χρήση του optimizer rmsprop
    - → Παράγεται f1 score = 0.67
    - → Αφαιρώντας το dropout εμφανίζεται έντονο overfitting
    - → Μεταβάλοντας τη παράμετρο alpha σε 0.6 παράγεται overfit παρά το οποιοδήποτε regularization
- Με χρήση του optimizer adam
    - → Παράγεται f1 score = 0.68
    - → Μεταβάλοντας τι τιμές betas στο σετ {(0.8,0.999), (0.7,0.999), (0.7,0.8), (0.7,0.7), (0.5,0.5)} παρατηρούμε τη σταδιακή μείωση των αποτελεσμάτων των μετρικών και overfitting.
    - → Χρησιμοποιώντας l2 regularization τα αποτελέσματα είναι άρτια (f1 score = 0.68 χωρίς overfit)
- Με χρήση του optimizer adamax
    - → Παρατηρούμε f1 score = 0.69 με έντονο overfitting
    - → Με l2 regularization αντιμετωπίζεται το φαινόμενο overfitting μεταβάλλοντας το score σε 0.68
    - → Πραγματοποιήθηκε παρόμοια μεταβολή των betas χωρίς βελτίωση.
- Με χρήση του optimizer nadam
    - → Παρατηρουμε overfitting με f1 score = 0.65 στην αρχή του φαινομένου.
    - → Μετά από regularization αποφεύγεται το overfit με μείωση του f1 score σε 0.63

Το τελικό μοντέλο

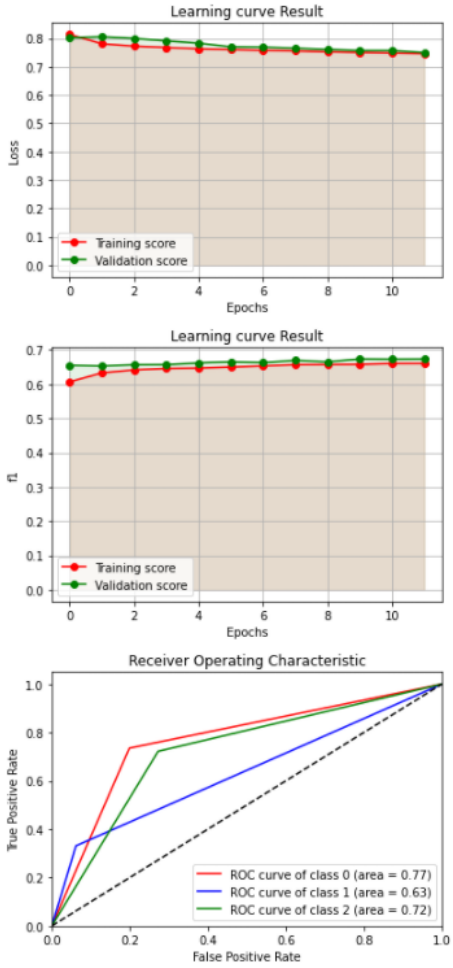
Το τελικό μοντέλο απαρτίζεται από:

- **Layers**: [input,1024] , [1024,512] , [512,3]
- **Activation function**: LeakyRelu και Dropout στην έξοδο κάθε layer
- **Loss function**: Cross Entropy
- **Optimizer**: Stochastic Gradient Descent με χρήση Momentum
- **Regularizer**: L2 regularizer (weight_decay=1e-3)
- **LR scheduler**: Exponential Learning Rate
- **Vectorizer**: glove pre-trained embeddings (size=300 dimensions)
- **Batch size**: 32
- **Epochs**: 14

Σύγκριση των Μοντέλων

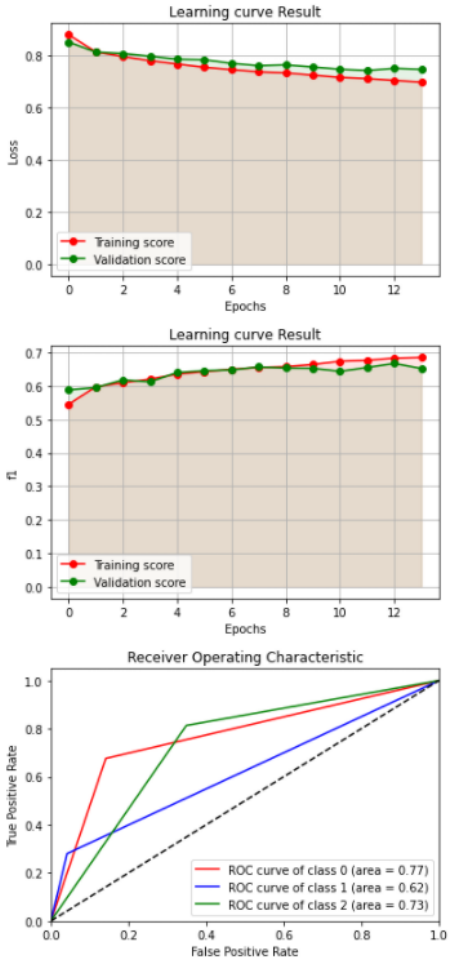
Στις παρακάτω εικόνες φαίνονται τα αποτελέσματα όπως τα παράγουν τα δύο μοντέλα.

set	precision	f1	recall	accuracy
Training	0.673069	0.674257	0.680583	0.680583
Validation	0.673092	0.673148	0.677476	0.677476



Μοντέλο 1

set	precision	f1	recall	accuracy
Training	0.707495	0.689743	0.698360	0.698360
Validation	0.688531	0.671867	0.680105	0.680105



Μοντέλο 2

Σημείωση: Παρά τον καθορισμό του χειροκίνητου seed σε 42, ενδέχεται τα αποτελέσματα ανά εκτέλεση να είναι διαφορετικά. Πιθανή αιτία είναι η λειτουργία του dropout, όπου με κάποια πιθανότητα αποκλείει κάποια units από τη διαδικασία του forward.

Παρατηρείται πολύ καλή σύγκλιση στα δύο μοντέλα, αποφυγή του φαινομένου overfitting και μια καλύτερη περιγραφή του μοντέλου από της ROC καμπύλες.

Συγκριμένα, τα δύο μοντέλα σταματούν την διαδικασία του training την κατάλληλη στιγμή, όπου αποφεύγεται είτε η διαφορά στο κόστος μεταξύ training και validation, είτε η στασιμότητα των αποτελεσμάτων.

Παρατηρούμε πως η ποσότητα των true positive προβλέψεων είναι διαρκώς αρκετά μεγαλύτερη των false positive προβλέψεων. Οι καμπύλες των αποτελεσμάτων τείνουν προς την πάνω αριστερά κορυφή όπου περιγράφει το ιδανικό μοντέλο. Το εμβαδό των καμπυλών με τη διαγώνιο μας δίνει ικανοποιητικά αποτελέσματα, όπως αναμενόταν. Για την κλάση 1, τα αποτελέσματα είναι κατώτερα των υπόλοιπων κλάσεων, το οποίο είναι αναμενόμενο, αφού οι εγγραφές που κατατάσσονται στην κλάση 1 αποτελούν μόλις το 15% του σετ δεδομένων.

Σύγκριση με το μοντέλο Multinomial Logistic Regression

Τα δύο μοντέλα παράγουν παρόμοιο αποτέλεσμα με το μοντέλο που αναπτύχθηκε κατά την εργασία 1.

set	precision	f1	recall	accuracy
Training	0.680184	0.678775	0.692288	0.692288
Validation	0.673559	0.672845	0.686678	0.686678

Multinomial Logistic Regression

set	precision	f1	recall	accuracy
Training	0.672221	0.668021	0.684089	0.684089
Validation	0.668443	0.663511	0.678791	0.678791

Μοντέλο 1

set	precision	f1	recall	accuracy
Training	0.707495	0.689743	0.698360	0.698360
Validation	0.688531	0.671867	0.680105	0.680105

Μοντέλο 2

Με μια πρώτη σκέψη, ένα πιο πολύπλοκο σύστημα όπως το νευρωνικό δίκτυο που χρησιμοποιήθηκε, θα έπρεπε να προσαρμόζεται καλύτερα στα δεδομένα του προβλήματος και να παράγει καλύτερα αποτελέσματα. Παρά τις πολλαπλές δοκιμές αυτό δεν επιτυγχάνεται, ενώ πολλοί παράγοντες ενδέχεται να επηρεάζουν το αποτέλεσμα αυτό, όπως:

- Ο μέσος όρος των word embeddings που υπολογίζεται και αντιπροσωπεύει το tweet μπορεί να μην χαρακτηρίζει το tweet το ίδιο καλά, όπως ένα σύστημα που θα δεχόταν πολλαπλά embeddings για ένα tweet.
- Μπορεί το πρόβλημα να προσαρμόζεται καλύτερα με ένα πιο απλό (multinomial logistic regression) ή πιο σύνθετο (RNNs) μοντέλο, από τα δύο μοντέλα της εργασίας αυτής.