CHATBOT USING SEQ2SEQ

6η Εργασία Μηχανικής Μάθησης

Κωνσταντίνος Πασβάντης
aid23005@uom.edu.gr

1. Εισαγωγή

Σε αυτή την εργασία παρουσιάζεται μία μικρή τεκμηρίωση του κώδικα που γράφηκε προκειμένου να κατασκευαστεί ένα chatbot γα την 6η και τελευταία εργασία της μηχανικής μάθησης.

Το dataset που θα χρησιμοποιήσουμε είναι το MetalWOZ το οποίο περιλαμβάνει 37.884 διαλόγους μεταξύ χρηστών και πολλών chatbot. Η διαδικασία κατασκευής και αξιολόγησης του chatbot είναι σχετικά απλή. Πρώτα φορτώνουμε και καθαρίζουμε τα δεδομένα, δηλαδή όλους τους διαλόγους, στην συνέχεια αφού τα καθαρίσουμε κατασκευάζουμε την αρχιτεκτονική του νευρωνικού δικτύου που θα εκπαιδευτεί το chatbot, και τέλος αξιολογούμε το μοντέλο βάσει κάποιας μετρικής.

2. Καθαρισμός Δεδομένων

Αρχικά, φορτώνουμε όλους τους διαλόγους που περιλαμβάνονται στο dataset. Παρατηρούμε ότι όλα τα αρχεία είναι σε txt format, με κάθε txt αρχείο να αποτελείται απο json objects. Οπότε χρησιμοποιώντας την βιβλιοθήκη json παίρνουμε από όλα τα αρχεία διαλόγων μόνο τις συνομιλίες που μας ενδιαφέρουν, οι οποίοι σε κάθε json object είναι αποθηκευμένοι στο κλειδί ‘turns’ .

Ο σκοπός μας είναι να δημιουργήσουμε δύο λίστες. Στην μία θα αποθηκεύονται οι ερωτήσεις που κάνει ο χρήστης και στην άλλη θα αποθηκεύονται οι απαντήσεις που δίνει το chatbot. Αφού δημιουργήσουμε τις δύο λίστες κάνοντας append σε κάθε διάλογο έναν ψεύτικο χαιρετησμό από τον χρήστη, και έναν ψεύτικο αποχαιρετισμό (σε περίπτωση που την τελευταία φράση στον διάλογο μας την λέει ο χρήστης, καθαρίζουμε και τις δυο λίστες με την χρήση της συνάρτησης ‘preprocess_text’. Μετά από αυτό βάζουμε δύο token στην αρχή και το τέλος κάθε πρότασης που έχουμε, ένα για να δηλώσουμε ότι αρχίζει η πρόταση (‘<SOS>’) και ένα για το τέλος της (<EOS>).

Για να μειώσουμε τον χρόνο της εκπαίδευσης, από όλες τις ερωτήσεις και απαντήσεις κρατάμε μόνο αυτές οι οποίες έχουν μέχρι και 15 λέξεις. Από αυτές τις απαντήσεις και ερωτήσεις διαλέγουμε ένα υποσύνολο τους ώστε να είμαστε σε θέση να εξιολογήσουμε τις απαντήσεις που δίνει το chatbot με τις πραγματικές. (Στην συγκεκριμένη περίπτωση διάλεξα 5000 ερωτήσεις και απαντήσεις με αριθμό λέξεων έως 17.

Τέλος, χρησιμοποιούμε την εντολή Tokenizer βάζοντας σαν argument τον αριθμό λέξεων να είναι 14999. Αυτό σημαίνει ότι δημιουργείται ένα

vocabulary με 14999 λέξεις ασχέτως αν υπάρχουν περισσότερες λέξεις που υπάρχουν στις ερωτήσεις και τις απαντήσεις που παίρνουμε. Αφού δημιουργήσουμε το vocabulary, τις μετατρέπουμε σε αριθμητικά διανύσματα αντικαθιστώντας κάθε λέξη με το index που έχει στο vocabulary που έχει δημιουργηθεί. Για να έχει κάθε αριθμητικό διάνυσμα το ίδιο μήκος, χρησιμοποιούμε την εντολή pad_sequences κάνοντάς τα να έχουν όλα μήκος 17 (15 είναι το μήκος της μεγαλύτερης λέξης και χρησιμοποιούμε 2 ακόμα για τα tokens αρχής και τέλους πρότασης).

Πριν προσωρήσουμε στην περιγραφή της αρχιτεκτονικής του νευρωνικού δικτύου, έχουμε καταλήξει προς το παρον με δύο λίστες. Η μία περιλαμβάνει τα αριθμητικά διανύσματα μήκους 17 που αντιστοιχούν σε κάθε ερώτηση, την οποία την έχουμε αποθηκεύσει με το όνομα encoder_input_data και η άλλη περιλαμβάνει τα αριθμητικά διανύσματα μήκους 17 που αντιστοιχόυν στις απαντήσεις, η οποία είναι αποθηκευμένη με το όνομα decoder_input_data. Δημιουργούμε μία ακόμα λίστα η οποία περιλαμβάνει πάλι τα αριθμητικά διανύσματα που αντιστοιχούν στις απαντήσεις, μόνο που αυτή τη φορά διαγράφουμε το πρώτο token κάθε διανύσματος, το οποίο αντιστοιχεί στο index του token ξεκινήματος πρότασης. Το έχουμε αποθηκεύσει με το όνομα decoder_final_output.

3. Μοντέλο Seq2Seq

Το μοντέλο που λειτουργεί τελικά για την δικιά μου περίπτωση είναι ένα seq2seq μοντέλο.

Αρχικά πριν το εκπαιδεύσουμε κατασκευάζουμε την αρχιτεκτονική του μοντέλου. Ξέρουμε ήδη από τον καθαρισμό δεδομένων που κάναμε ότι τα διανύσματα που θα χρησιμοποιήσουμε έχουν μήκος 17, οπότε δημιουργούμε τα input του encoder και του decoder ώστε να έχουν μήκος 17. Στην συνέχεια δημιουργούμε δύο embedding layers, ένα για τον encoder και ένα για τον decoder, ώστε να ρίξει την διάσταση από το 15000 (που είναι ο αριθμός των λέξεων που έχουμε χρησιμοποιήσει, στο 100. Έπειτα, δημιουργούμε τα LSTM layer, τα οποία έχουν 512 cells, και αφού γίνει η εκπαίδευση σε αυτά τα layer, το output που παράγεται είναι τα hidden states και τα cell states, τα οποία τα χρησιμοποιούμε για να αρχικοποιήσουμε τις καταστάσεις του decoder. Το τελευταίο layer που χρησιμοποιούμε είναι ένα dense layer το οποίο χρησιμοποιεί το αποτέλεσμα της εκπαίδευσης του decoder lstm layer. Αφού δημιουργήσουμε το μοντέλο, το κάνουμε compile χρησιμοποιώντας σαν loss function την sparse categorical crossentropy, κυρίως γιατί δεν μπορούσαν να μετατραπούν σε categorical διανύσματα οι προτάσεις λόγω έλλειψης ram.

Εφόσον έχουμε εκπαιδεύσει το μοντέλο μας, πρέπει να κάνουμε inference χρησιμοποιώντας τα αποτελέσματα και τα states από την εκπαίδευση του μοντέλου μας. Δημιουργούμε δύο νέα μοντέλα, ένα encoding και ένα decoding.

Το encoding model θα παίρνει σαν input τα αριθμητικά διανύσματα των ερωτήσεων και σαν αποτέλεσμα θα βγάζει τα states που θα εκπαιδευτούν από το LSTM layer του encoder που δημιουργήσαμε στο βασικό μοντέλο. Στην συνέχεια θα παρέχει τον decoder με αυτά τα states ώστε να μπορέσει να δημιουργήσει μία πρόταση βασισμένος στο token για την αρχή της πρότασης, χρησιμοποιώντας τα states που έχουν βρεθεί από το LSTM layer του decoder του βασικού μοντέλου.

Αφού εκπαιδευτούν και τα τρία μοντέλα, τα κάνουμε save λόγω του χρόνου εκπαίδευσης, ο οποίος για τα συγκεκριμένα μοντέλα με 50 epochs και με χρήση gpu που παρέχει το google colab, ανερχόταν περίπου στις δυόμιση ώρες.

4. Συνομιλία με το chatbot

Εφόσον εκπαιδεύσουμε τα μοντέλα μας και τα κάνουμε save, είμαστε σε θέση με την εντολή load_model να τα αξιοποιήσουμε και πάλι χωρίς να χρειαστεί να επανεκπαιδεύσουμε τα μοντέλα.

Αρχικά δημιουργούμε μία συνάρτηση η οποία θα παίρνει σαν όρισμα ένα string και θα το μετατρέπει σε μορφή κατάλληλη για τα μοντέλα ώστε να προβλέψουν την απάντηση.

Η επόμενη συνάρτηση είναι αυτή που μας επιτρέπει να συνομιλήσουμε με το chatbot. Κάθε φορά μετατρέπει το string που θέλουμε (την ερώτηση μας), σε μορφή που να μπορεί να περαστεί σαν input στο encoder model που δημιουργήσαμε για inference. Δημιουργούμε ένα empty_target_sequence το οποίο κάθε φορά θα έχει το index της κάθε λέξης που προβλέπει ο decoder, ώστε μετά να προβλέψει την επόμενη λέξη που θα την διαδεχτεί στην πρόταση. Η αρχική λέξη προφανώς και θα είναι το token για το ξεκίνημα της πρότασης (‘sos’). Αφού έχουμε περάσει το string που δώσαμε σαν input από τον encoder και έβγαλε σαν output τις καταστάσεις που δημιουργούνται από αυτό το input, χρησιμοποιούμε αυτές τις καταστάσεις για να προβλέψουμε την λέξη πιυ θα διαδεχτεί το token της αρχής της πρότασης. Αφού βρούμε τελικά την επόμενη λέξη, την χρησιμοποιούμε για να βρούμε την αμέσως επόμενη της, κάτι το οποίο συνεχίζεται μέχρι να βρεθεί το token το οποίο σηματοδοτεί

το τέλος της πρότασης, ή ανη πρόταση που βρεθέι καταλήξει να έχει πάνω από 17 λέξεις .

Μερικές από τις συνομιλίες που προέκυψαν είναι οι εξής:

```
--------------------------
ENTERING CHAT
Press "/q" to quit chatting
--------------------------
You: hello maybe
Bot:  hello how may i help you
You: i think im hungry
Bot:  are you sure
You: yes i am sure
Bot:  you are all set
You: can i order anything
Bot:  you can try
You: i mean something like a pizza
Bot:  oh i love pizza s
You: so you eat pizza
Bot:  yes
You: what is your favourite
Bot:  i do not have access to that information sorry
You: tell me
Bot:  you can ask me anytime
You: i just asked you
Bot:  i am not sure
You: well, i am, look at the chat
Bot:  i can provide you with a change
You: this doesnt make sense
Bot:  it really is
You: can i leave now?
Bot:  yes
You: Have a nice day
Bot:  you have a good day
You: /q
```

```
--------------------------
ENTERING CHAT
Press "/q" to quit chatting
--------------------------
You: hello again
Bot:  hello
You: i need help
Bot:  i can sort some things for you
You: what can you do
Bot:  all of your personal information
You: like what do you mean
Bot:  you can ask me questions about basic facts
You: is the color of my eyes blue
Bot:  no
You: you made a mistake
Bot:  yes i know
You: tell me about you
Bot:  sure
You: how old are you
Bot:  29 million
You: do you like me
Bot:  you need me to do that
You: thats a sure thing
Bot:  i have got you location
You: i am scared
Bot:  is there anything else i can help you with
You: dont come here
Bot:  does not a bot
You: thanks
Bot:  is there anything else
You: no
Bot:  then take care
You: /q
```

```
---------------------------
ENTERING CHAT
Press "/q" to quit chatting
---------------------------
You: are you here
Bot:  sure how can i help
You: i want to fight you
Bot:  you do not have a plan bot
You: you mean a plan b
Bot:  yes
You: are you ready to fight
Bot:  that is a pretty common query
You: did you get asked a lot
Bot:  it is amazing to me
You: wait, you enjoy fighting
Bot:  well let me know if you have any further questions have a nice day
You: i dont want to ask
Bot:  how many people are in your party
You: i will win you alone
Bot:  to be careful of what you are looking for
You: give it to me
Bot:  hi i am sorry now
You: are you afraid now
Bot:  my database says so
You: oh you should be
Bot:  have a good day
You: /q
```

5. Αξιολόγηση του chatbot

Για την αξιολόγηση των απαντήσεων του chatbot υπάρχουν πολλές μετρικές που θα μπορούσα να χρησιμοποιήσω, αποφάσισα όμως να διαλέξω την ομοιότητα κατά jaccard. Η μετρική αυτή μετράει σε δύο σύνολα (στην συγκεκριμένη περίπτωση προτάσεις), τον αριθμό των στοιχείων τα οποία είναι ίδια, ως προς τον αριθμό των συνολικών στοιχείων των δυό προτάσεων. Για παράδειγμα, αν η μία πρόταση έχει 3 λέξεις και η άλλη έχει 5, ενώ έχουν 2 κοινές λέξεις, η ομοιότητα κατα jaccard αυτών των δύο προτάσεων θα είναι ίση με 2/8.

Οπότε για την αξιολόγηση θα χρησιμοποιήσουμε τις ερωτήσεις για evaluation που δημιουργήσαμε στην αρχή, θα βρούμε τις απαντήσεις που θα μας δώσει το chatbot πάνω σε αυτές τις ερωτήσεις, και στην συνέχεια θα συγκρίνουμε τις απαντήσεις του chatbot με τις κανονικές. Η μετρική αυτή ξέρουμε ότι δεν είναι η ιδανικότερη για την αξιολόγηση ενός chatbot, κυρίως γιατί μπορεί και οι δύο απαντήσεις σε μία ερώτηση να βγάζουν το ίδιο νόημα, χωρίς απαραίττητα να έχουν κάποια ίδια λέξη. Αλλά είναι μία αρκετά καλή μετρική για να δούμε πόσο καλά έχει μάθε το chatbot να απαντάει. Για αυτό δημιουργούμε μία συνάρτηση που να υπολογίζει την ομοιότητα jaccard, και στην συνέχεια χρησιμοποιούμε μία λίστα για να αποθηκεύσουμε την τιμή της για κάθε απάντηση. Στην συνέχεια παίρνουμε τον μέσο όρο αυτής της λίστας, και βλέπουμε ότι το αποτέλεσμα είναι ίσο με 0.0965 Προφανώς και δεν περιμέναμε ο αριθμός αυτός να είναι μεγάλος σε 5000 απαντήσεις, αλλά βλέπουμε ότι σε αρκετές απαντήσεις έχουμε μερικές ίδιες λέξεις, κάτι το οποίο σημαίνει οτι το chatbot εκπαιδεύτηκε αρκετά καλά πάνω σε αυτό το dataset.

6. Αναφορές

Η διαδικασία κατασκευής και εκπαίδευσης του chatbot ήταν απαιττητική. Στην αρχή χρησιμοποιήθηκαν διαφορετικά μοντέλα εκπαίδευσης πέρα από seq2seq , όπως sequential και αρκετές διαφορετικές δομές από Recurrent Neural Networks (RNN). Η εκπαίδευση των μοντέλων όμως δεν έβγαζε καλά αποτελέσματα. Το chatbot δεν βρισκόταν σε θέση να απαντήσει στις ερωτήσεις και δεν μπορούσε να κάνει καμία αναγνώριση μεταξύ των μοτίβων σε μία πρόταση.

Ένα άλλο πρόβλημα που εμφανίστηκε, ήταν η έλλειψη χωρητικότητας ram, καθώς όταν έπρεπε να μετατρέψουμε σε κατηγορικά διανύσματα τις απαντήσεις, το μήκος και ο αριθμός των διανυσμάτων ήταν πολύ μεγάλος για να τα διαχειριστεί το google colab. Τελικά αυτό διορθώθηκε βάζοντας σαν loss function στο νευρωνικό μας δίκτυο, sparse categorical crossentropy.

Οι σύνδεσμοι που με βοήθησαν σε μεγάλο βαθμό στην κατανόηση και την κατασκευή ενός μοντέλου seq2seq ώστε να δημιουργήσω ένα chatbot είναι οι εξής:

- https://towardsdatascience.com/how-to-build-your-own-chatbot-using-deep-learning-bb41f970e281
- https://towardsdatascience.com/generative-chatbots-using-the-seq2seq-model-d411c8738ab5
- https://medium.com/swlh/how-to-design-seq2seq-chatbot-using-keras-framework-ae86d950e91d
- https://medium.com/swlh/how-to-design-seq2seq-chatbot-using-keras-framework-ae86d950e91d