IMAGE CAPTIONING
using Deep Learning models

Deep Learning Final Assignment

2023

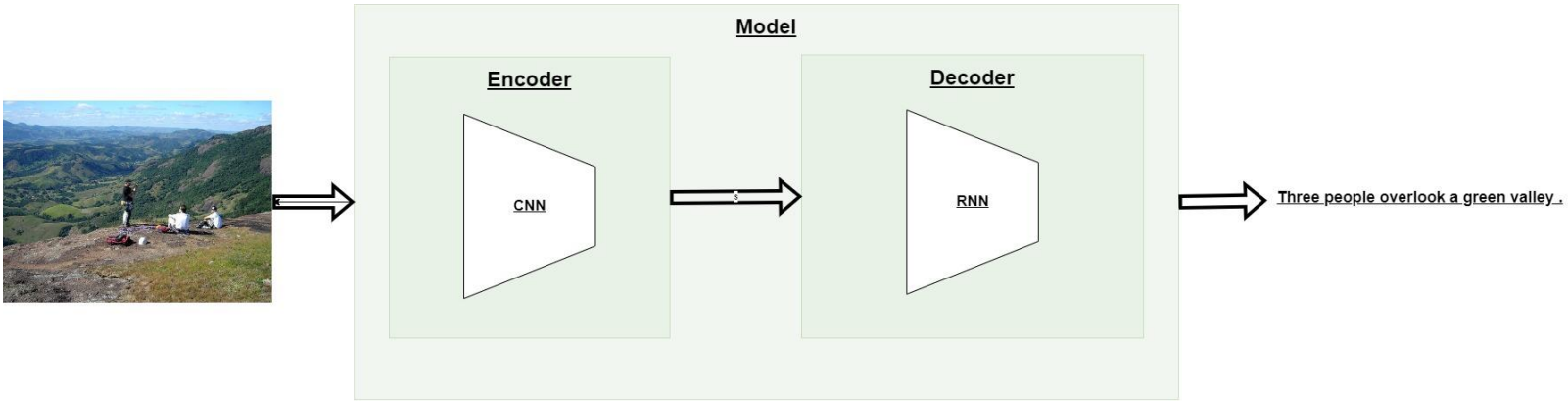
Ονοματεπώνυμο: Βασίλης Σταυριανουδάκης

AM: mtn2215

Περιεχόμενα

Περιγραφή προβλήματος και δεδομένων

Στόχος του συγκεκριμένου προβλήματος είναι η αυτόματη εξαγωγή caption από εικόνες. Για την επίλυση του γίνεται κυρίως χρήση Deep Learning μοντέλων, οπού ο τρόπος που λειτουργούν, σε ένα υψηλό επίπεδο, παρουσιάζεται στην παρακάτω εικόνα:



Μια εικόνα εισάγεται στο μοντέλο, το οποίο αποτελείται από δυο κομμάτια και στο τέλος το μοντέλο εξάγει το caption που έχει παράξει για την συγκεκριμένη εικόνα.

Τα δυο επιμέρους κομμάτια του μοντέλου είναι ο Encoder και ο Decoder. Ο Encoder έχει ως σκοπό να επεξεργαστεί την εικόνα που εισάγεται και να την αναπαραστήσει με ένα feature vector, ή αλλιώς image context. Στην συνέχεια το feature vector εισάγεται στον Decoder, ο οποίος αρχίζει και παράγει κείμενο με βάση το συγκεκριμένο vector που δέχτηκε. Η παραγωγή του κείμενο σταματάει όταν ο Decoder παράξει ένα ειδικό σύμβολο που αντιπροσωπεύει το τέλος της παραγωγής ή αν ένας προκαθορισμένος αριθμός από λέξεις έχει παραχθεί.

Για αυτήν την υλοποίηση το dataset που χρησιμοποιήθηκε είναι το Flickr-8K, το οποίο περιλαμβάνει περίπου 8.000 μοναδικές εικόνες και από 5 εναλλακτικά captions για κάθε εικόνα. Ένα παράδειγμα φαίνεται στην παρακάτω εικόνα:

two dogs on pavement moving toward each other .

two dogs of different breeds looking at each other on the road .

a black dog and a white dog with brown spots are staring at each other in the street .

a black dog and a tri-colored dog playing with each other on the road .

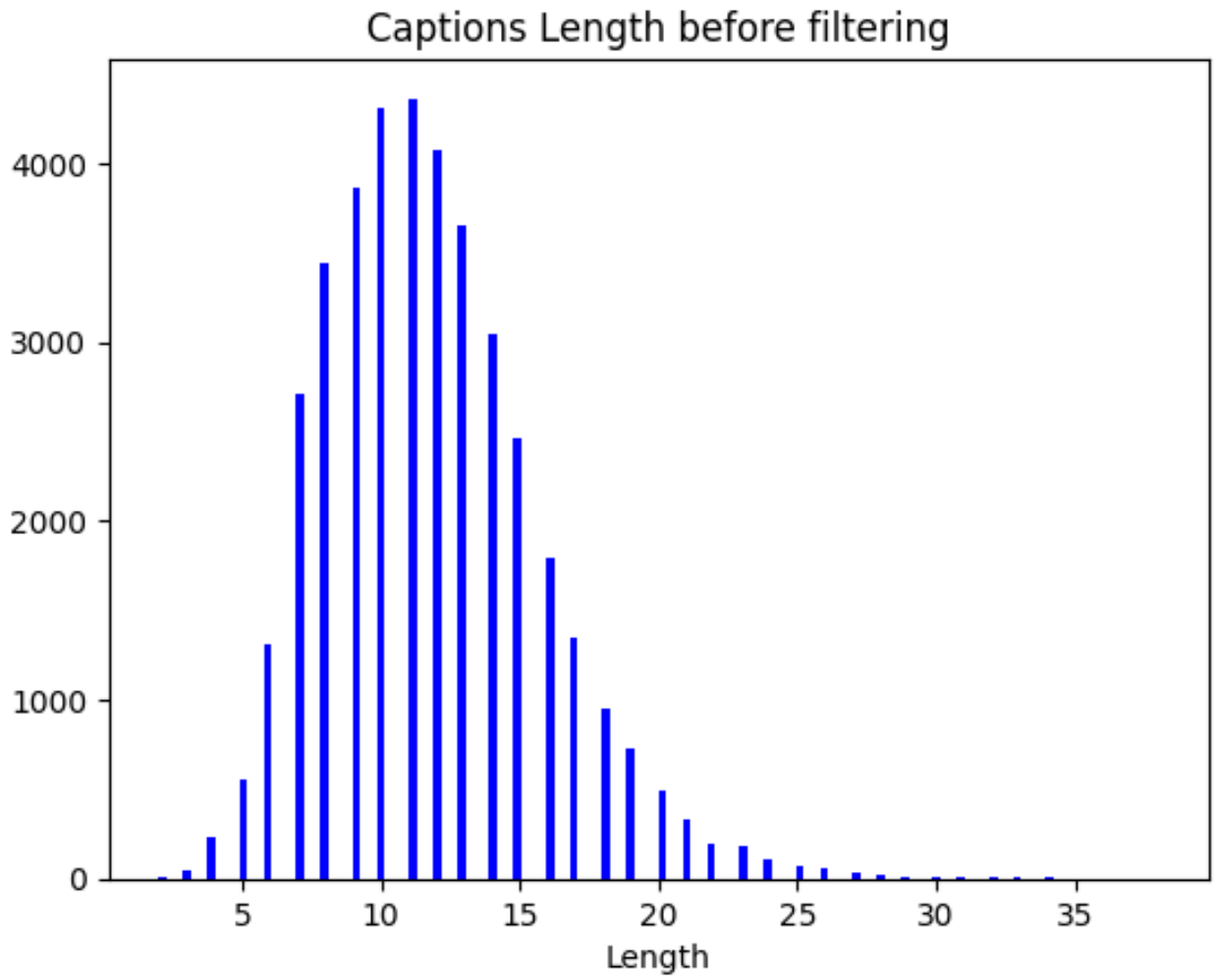
a black dog and a spotted dog are fighting

Προεπεξεργασία Δεδομένων

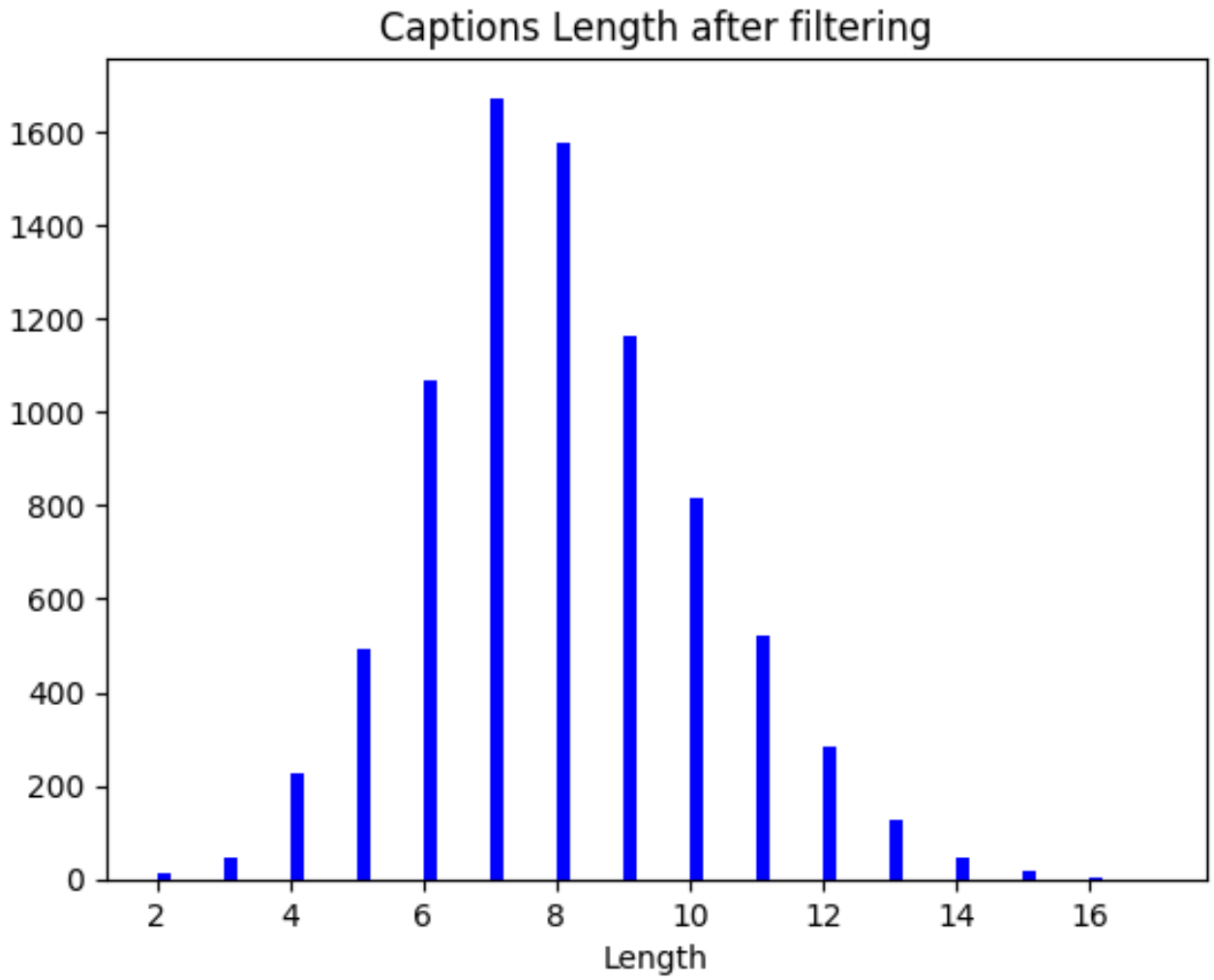
Αρχικά, για να απλοποιηθεί λίγο το πρόβλημα στα πλαίσια του μαθήματος, από κάθε εικόνα χρησιμοποιήθηκε ένα caption από τα πέντε. Συγκεκριμένα χρησιμοποιήθηκε το caption με το μικρότερο πλήθος λέξεων. Επιπλέον, από την στιγμή που έχουμε εξασφαλίσει ότι κάθε εικόνα - caption ζευγάρι είναι μοναδικό, με random τρόπο χωρίστηκαν τα δεδομένα. Συγκεκριμένα περίπου 6.400 εικόνες - captions ζευγάρια χρησιμοποιήθηκαν στο training set και από 810 στο validation και test set.

Στις παρακάτω εικόνες μπορούμε να παρατηρήσουμε την κατανομή των μεγεθών των captions όταν έχουμε και τα πέντε από κάθε εικόνα και μετά, όταν πλέον έχουμε μόλις ένα.

Με τα πέντε captions ανά εικόνα (8.000 x 5 = 40.000 captions):



Με ένα caption ανά εικόνα (8.000 x 1 = 8.000 captions):



‘Οπου στον X άξονα βλέπουμε πόσες λέξεις υπάρχουν σε κάθε caption και στον Y ποσά captions έχουν τέτοιο πλήθος από λέξεις.

Αφού πλέον έχουμε καταλήξει στο dataset που θα χρησιμοποιηθεί, είναι σημαντικό να προεπεξεργαστούμε τόσο τις εικόνες, όσο και τα captions.

Εικόνες

Σχετικά με τις εικόνες, ένα αρχικό πρόβλημα είναι ότι η κάθε μια έχει διαφορετικές διαστάσεις. Για να τις φέρουμε όλες στις ιδίες διαστάσεις, ώστε έχουμε ένα ομοιόμορφο dataset αλλά και για να περιορίσουμε λίγο την πληροφορία που υπάρχει σε αυτές, ώστε το training του μοντέλου να γίνεται πιο γρήγορα, κάθε εικόνα έγινε resized σε 256x256. Επιπλέον, επειδή κάθε pixel μιας εικόνας παίρνει τιμές από 0 έως 255, έγινε κανονικοποίηση των τιμών και πλέον κάθε pixel έχει μια τιμή από 0 έως 1. Τέλος, για να μην επαναλαμβάνεται αυτή η διαδικασία κάθε φορά που γίνεται το training ενός μοντέλου και επιπλέον καταναλώνεται χρόνος, οι προεπεξεργασμένες εικόνες αποθηκεύονται. Έτσι κατά το training αρκεί να φορτώσουμε τις εικόνες που θέλουμε και να τις χρησιμοποιήσουμε στο μοντέλο. Αξίζει να αναφερθεί ότι επειδή το σύνολο αυτής της διαδικασίας μπορεί πάλι να πάρει αρκετό χρόνο, ειδικά αν μελλοντικά θέλουμε να προσθέσουμε και άλλες εικόνες ή αλλά βήματα προεπεξεργασίας, ο κώδικας χρησιμοποιεί threads. Κατά αυτόν τον τρόπο μειώνουμε κατά ένα τεράστιο ποσοστό τον συνολικό χρόνο εκτέλεσης και μπορούμε να εξασφαλίσουμε ότι η διαδικασία θα εκτελεστεί με τον αποδοτικότερο δυνατό τρόπο.

Captions

Όπως ειπώθηκε και στην εισαγωγή, η παραγωγή κειμένου από το μοντέλο, σταματάει είτε μόλις παραχθεί ένα ειδικό σύμβολο είτε μόλις ξεπεράσει έναν προκαθορισμένο αριθμό από λέξεις. Αντί λοιπόν να χρησιμοποιηθεί κάποιο αυθαίρετο σύμβολο που να σηματοδοτεί τον τερματισμό της παραγωγής, χρησιμοποιήθηκε το «.». Ένα βήμα της προεπεξεργασίας των κείμενων μας, εξασφαλίζει ότι κάθε caption θα τελειώνει με αυτό το σύμβολο, ώστε ειδικά κατά την εκπαίδευση του, το μοντέλο να είναι σε θέση να μάθει ποτέ και πως τελειώνει μια πρόταση.

Για να καθοριστούν τα υπόλοιπα βήματα, θα πρέπει πρώτα να εξάγουμε κάποιες πληροφορίες από αυτά πριν είμαστε σε θέση να αποφασίσουμε τι προεπεξεργασία απαιτούν. Ένας γρήγορος πρώτος έλεγχος μπορεί να γίνει εξάγοντας τους μοναδικούς χαρακτήρες που υπάρχουν από όλα τα κείμενα. Αυτή η πληροφορία φαίνεται στον παρακάτω πίνακα:

"z",	".",	"O",	"I",	"S",	"'",	"Z",	"c",	"?",
";",	"&",	"w",	"-",	"i",	"d",	"y",	"C",	"j",
"E",	" ",	"T",	"X",	"\"",	"Q",	"p",	"v",
":",	"0",	"B",	"#",	"Y",	"M",	"g",	"P",
"!",	"e",	")",	"b",	"R",	"N",	"D",	"8",
"=",	"u",	"s",	"a",	"4",	"x",	"U",	"3",
"r",	"V",	"W",	"q",	"o",	"7",	"2",	"J",
"m",	"f",	"A",	"9",	"6",	"K",	"t",	"G",
"F",	"(",	"L",	"n",	",",	"I",	"1",	"H",	"k",
"h",	"5"							

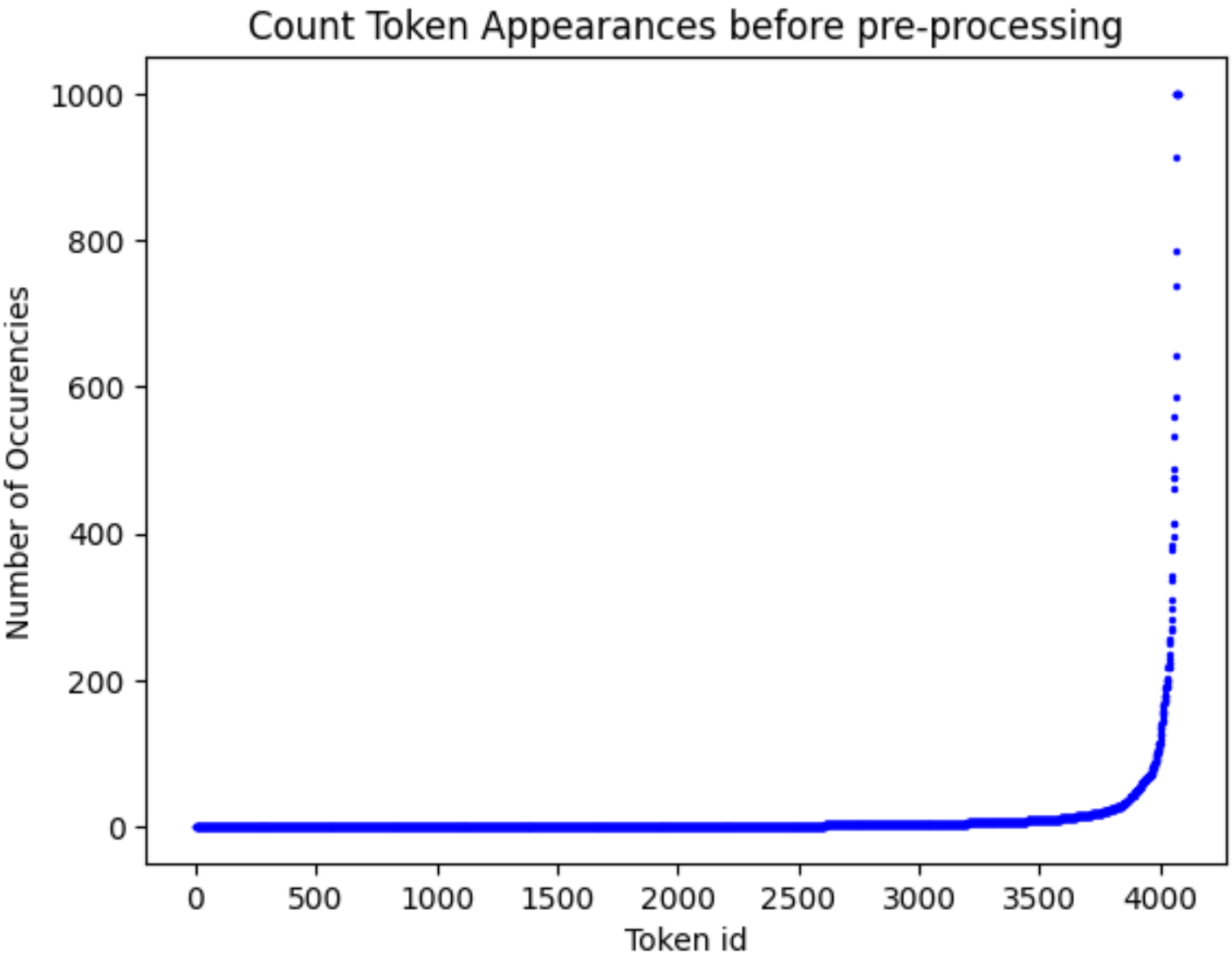
Το πρώτο που μπορεί να παρατηρηθεί είναι ότι υπάρχουν και κεφαλαίοι αλλά και πεζοί χαρακτήρες. Επίσης, υπάρχουν αριθμοί αλλά και κάποια σύμβολα. Εμφανίζοντας, τυχαία, μερικές προτάσεις όπου περιέχονται αριθμοί ή σύμβολα μπορούμε να διαμορφώσουμε την κατάλληλη διαδικασία προεπεξεργασίας των κείμενων μας. Τα βήματα της διαδικασίας, μαζί με κάποια παραδείγματα από captions, συνοψίζονται στην παρακάτω λίστα:

1. Μετατροπή όλων των κείμενων σε πεζούς χαρακτήρες.
    a. «A girl going into a wooden building .» → « a girl going into a wooden building .»
2. Αντικατάσταση του συμβόλου «#» με την λέξη «number» και του «&» με την λέξη «and».
    a. «Closeup of football player # 25 .» → «closeup of football player number 25 .»
    b. «A brown & white greyhound dog sniffs the snow .» → «a brown and white greyhound dog sniffs the snow .»
3. Μετατροπή όλων των αριθμών σε λέξεις.
    a. «Closeup of football player # 25 .» → «closeup of football player number twenty-five .»
4. Αφαίρεση όλων των λέξεων που βρίσκονται μέσα σε παραθέσεις.
    a. «its a distorted lens (almost fish eye) of a teenage boy skateboarding on a concrete block .» → «its a distorted lens of a teenage boy skateboarding on a concrete block .»
5. Αφαίρεση των συμβολών «;!:()».
    a. «A man just caught a fish and it is feisty !» → «a man just caught a fish and it is feisty»
6. Κάθε caption να τελειώνει με το σύμβολο «.».

Υπάρχουν αρκετοί λόγοι που κάνουν την διαδικασία καθαρισμού των κείμενων, εξαιρετικά σημαντική. Αρχικά κατά αυτόν τον τρόπο περιορίζουμε την «ποικιλία» των λέξεων και καταστάσεων που υπάρχουν μέσα στα δεδομένα μας. Αυτό βοηθάει πάντα, αφού, περιορίζοντας το πλήθος των διαφορετικών και μοναδικών καταστάσεων «φέρνουμε» τα δεδομένα μας λίγο πιο κοντά, ελαχιστοποιώντας έτσι, την δυσκολία ενός οποιοδήποτε προβλήματος όπου γίνεται η επίλυση με την χρήση ενός μοντέλου Βαθιάς Μηχανικής μάθησης.

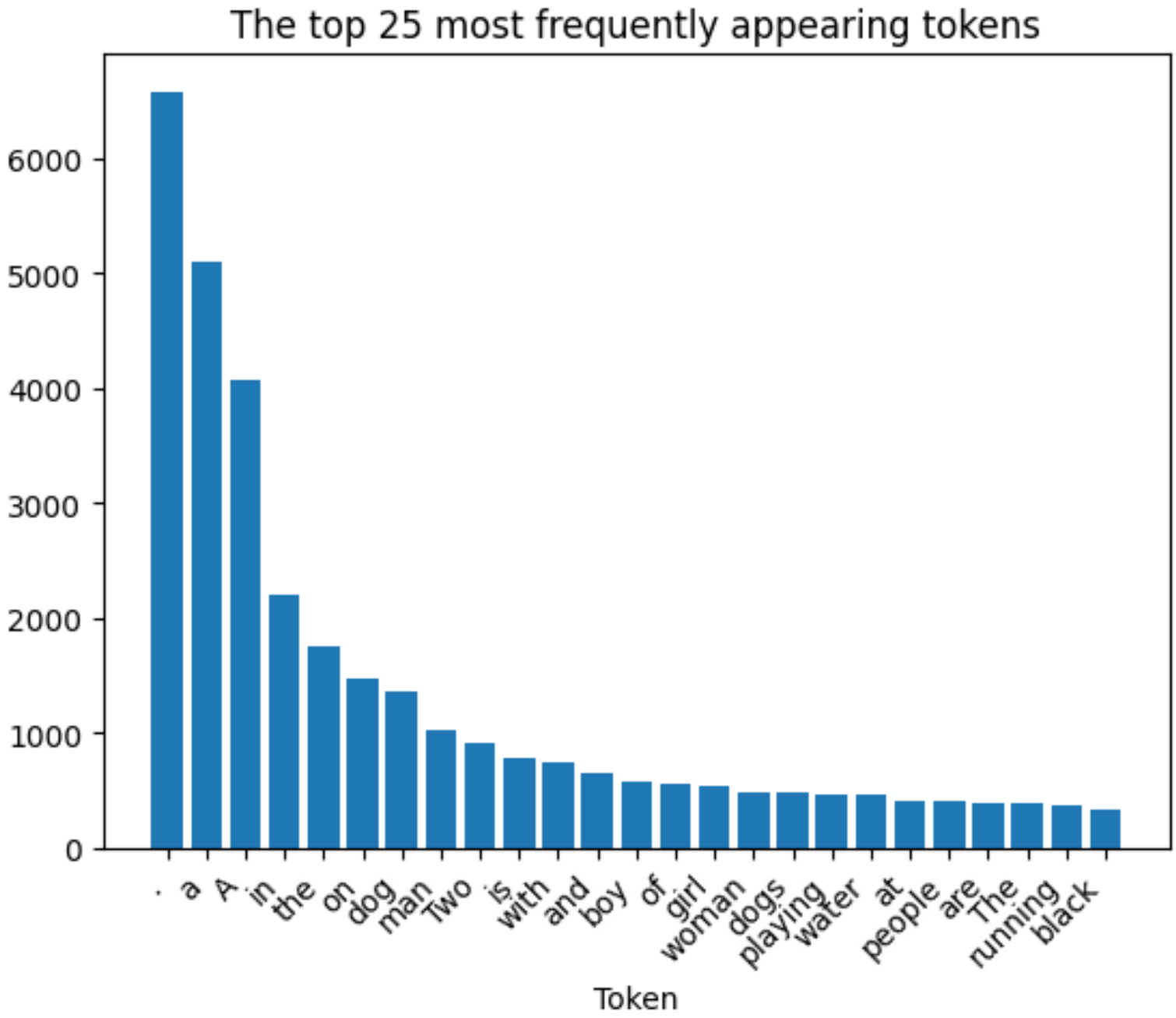
Στο θέμα όμως του Image Captioning όπου περιέχεται και το στάδιο της παραγωγής κειμένου, όπως θα δούμε και στο επόμενο κεφάλαιο, σε κάθε χρονική στιγμή το μοντέλο θα πρέπει να προβλέψει ποια λέξη, από τις διαθέσιμες που έχει στο λεξιλόγιο του, είναι η καταλληλότερη. Κατά συνέπεια, περιορίζοντας το διαθέσιμο λεξιλόγιο του, το μοντέλο θα έχει λιγότερες λέξεις (ή κλάσεις) από τις οποίες πρέπει να διαλέξει.

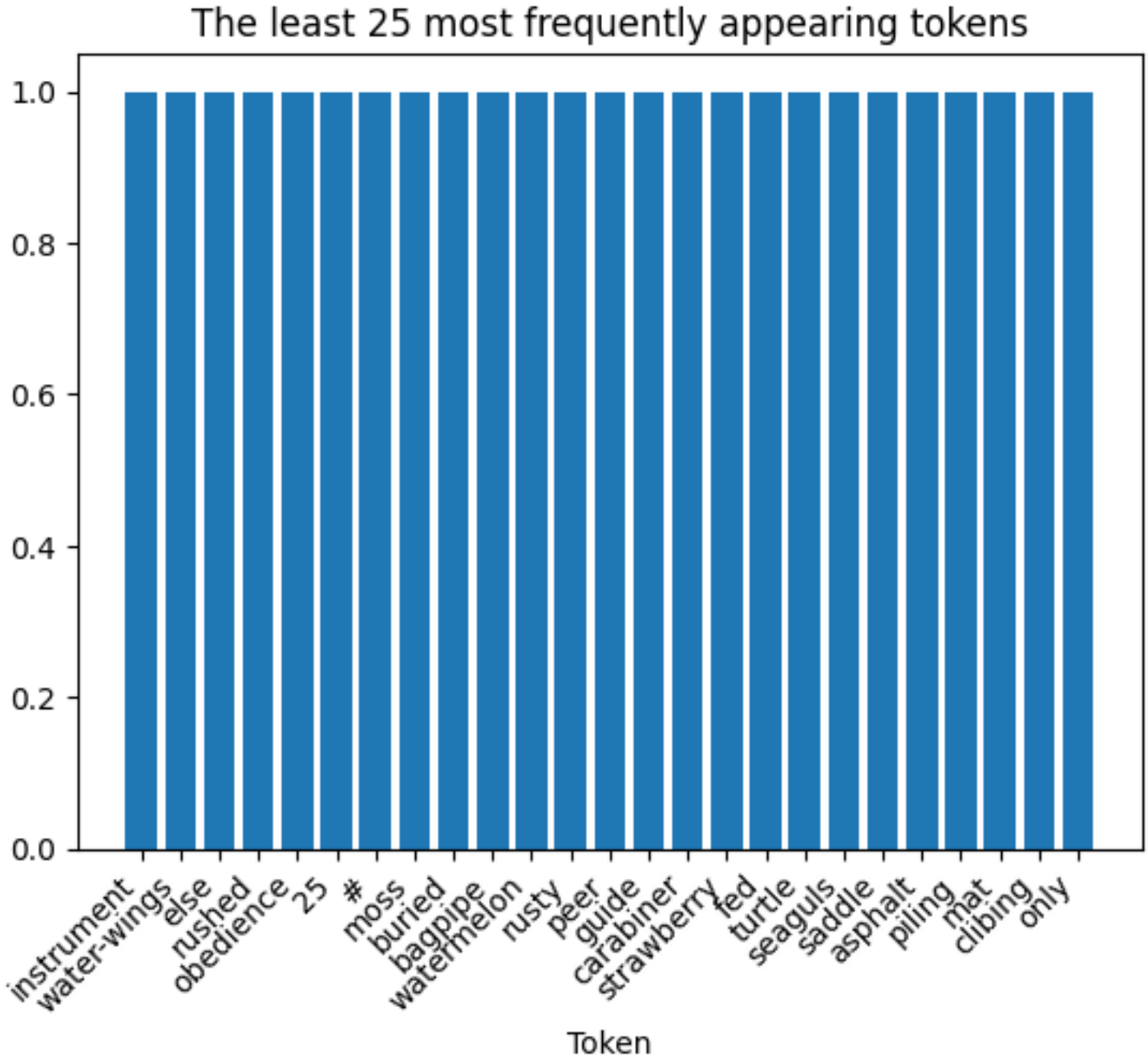
Για να τονίσουμε την δυσκολία του προβλήματος αλλά και την σημασία του σταδίου της προεπεξεργασίας των κείμενων, αρκεί να δούμε το παρακάτω διάγραμμα το οποίο απεικονίζει στον Χ άξονα τις πόσες μοναδικές λέξεις έχουμε και στον Υ άξονα πόσες φορές εμφανιζεται η κάθε λέξη.



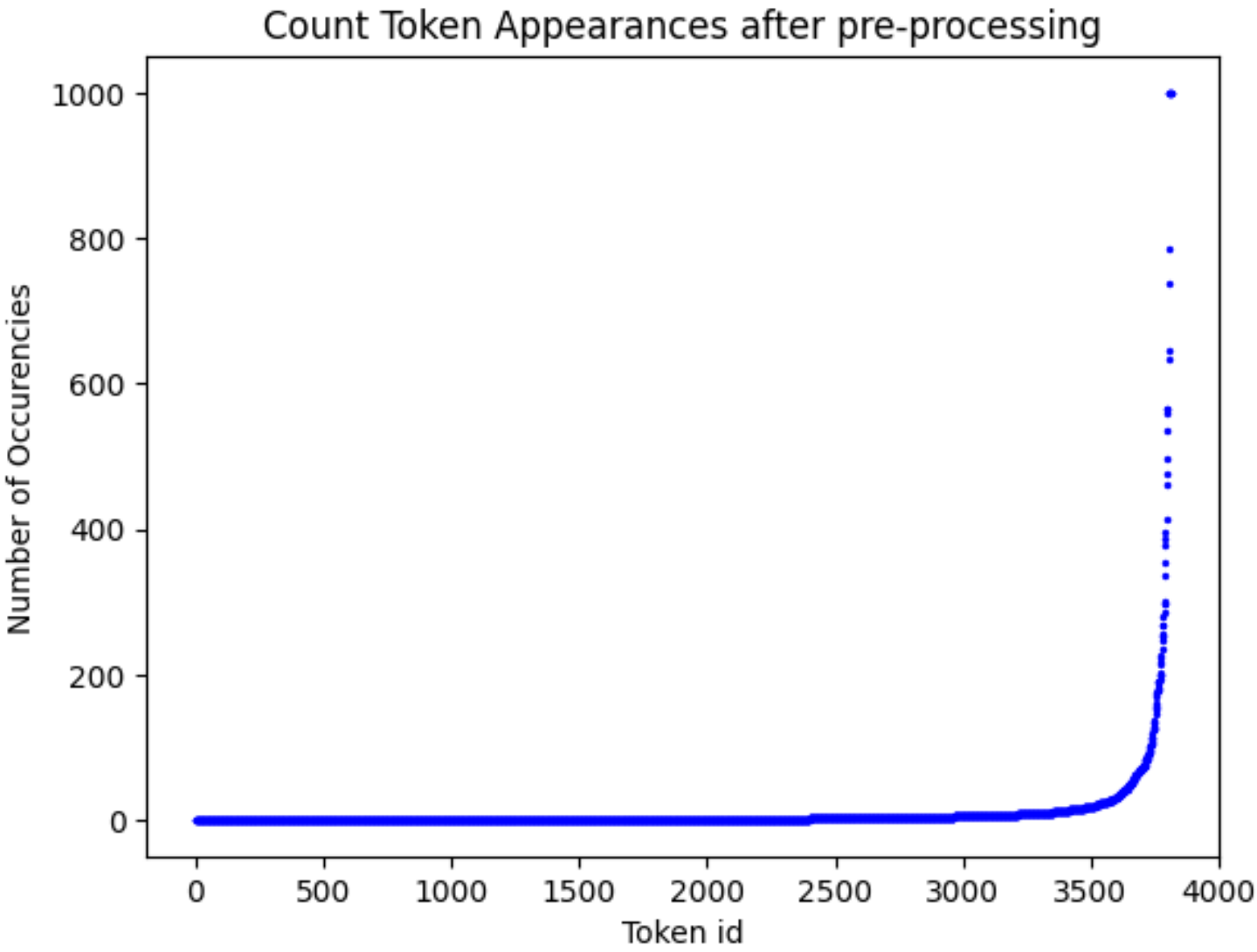
Όπως μπορεί να παρατηρηθεί, έχουμε πάνω από 4.000 μοναδικές λέξεις όπου η πλειοψηφία τους, συγκεκριμένα 2.500 λέξεις εμφανίζονται μόλις 1 φορά σε όλα μας τα κείμενα. Αν χρησιμοποιούσαμε όλες αυτές τις λέξεις θα σήμαινε ότι σε κάθε χρονική στιγμή, το μοντέλο μας θα έπρεπε να διαλέξει ποια από τις 4.000 λέξεις είναι η καταλληλότερη, έχοντας δει τις περισσότερες ελάχιστες φορές.

Επιπλέον, στα παρακάτω διαγράμματα απεικονίζονται οι 25 πιο συχνές και λιγότερο συχνές λέξεις.

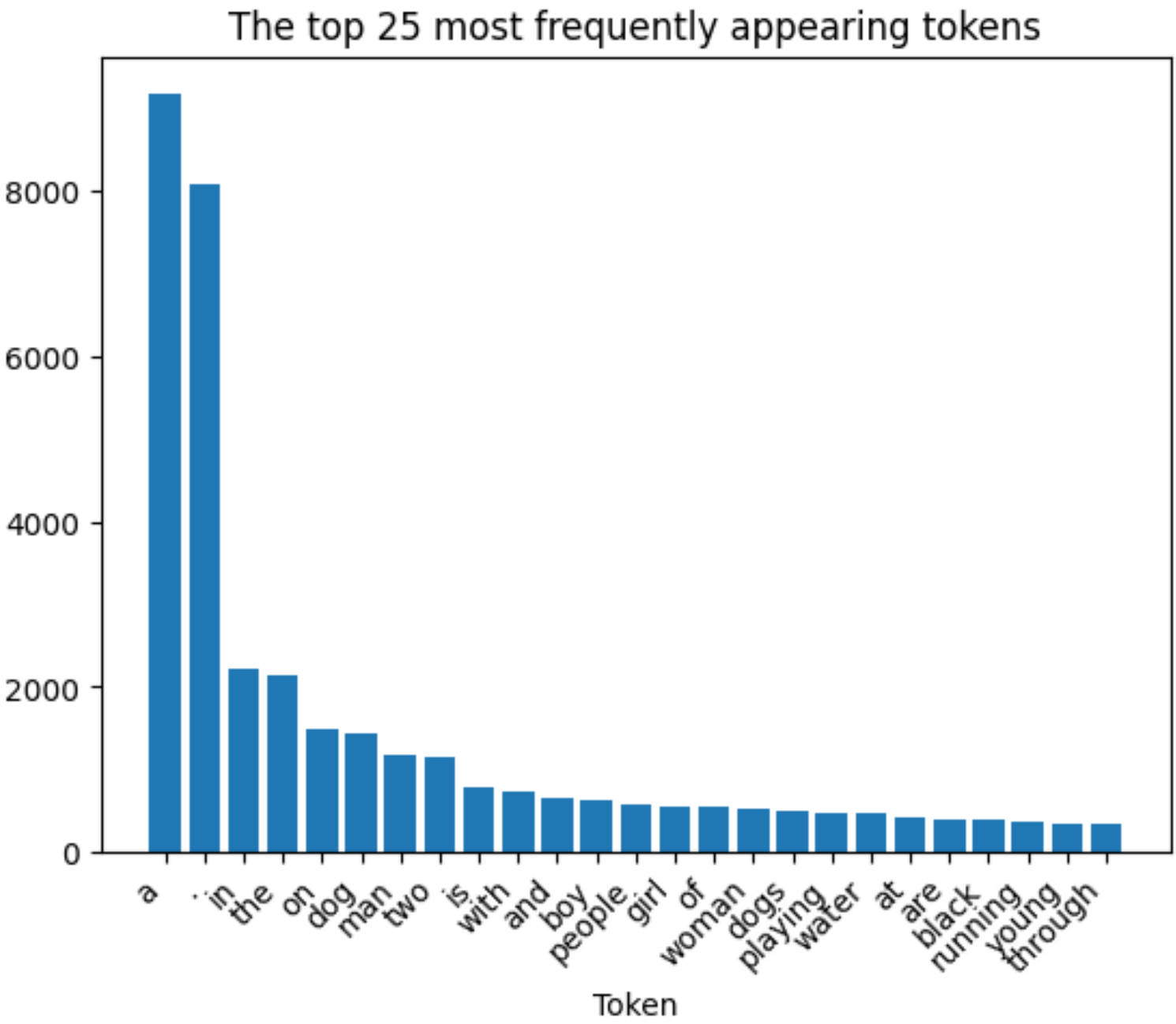


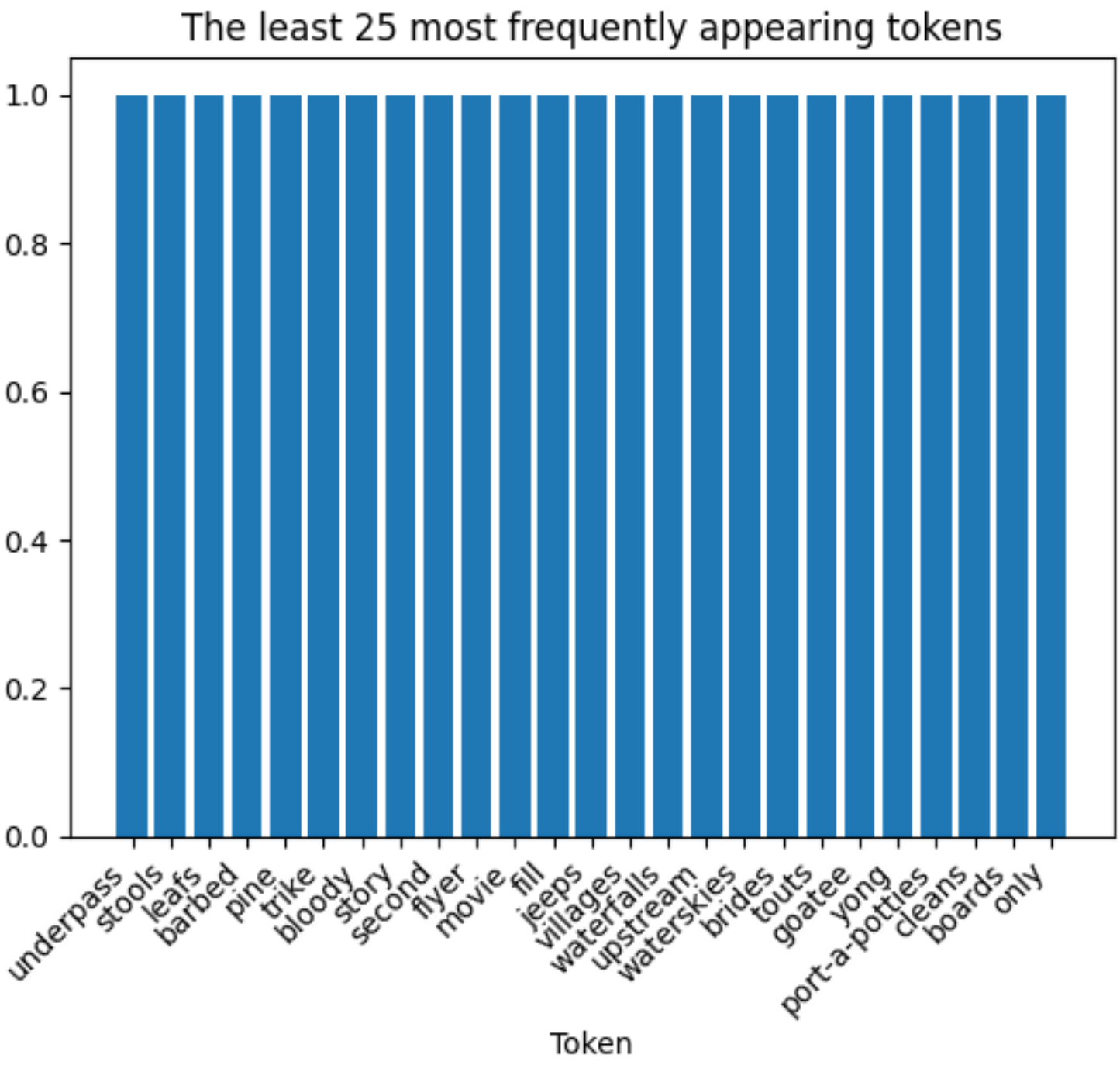

The least 25 most frequently appearing tokens
1.0
0.8
0.6
0.4
0.2
0.0
instrument
water-wings
else
rushed
obedience
25
#
moss
buried
bagpipe
watermelon
rusty
peer
guide
carabiner
strawberry
fed
turtle
seaguls
saddle
asphalt
piling
mat
clibing
only
Token

Σε αντίθεση, στα παρακάτω διαγράμματα μπορούμε να παρατηρήσουμε, μετά την προεπεξεργασία, πως διαμορφώνονται τα κείμενα μας.



Παρόλο που οι μοναδικές λέξεις είναι πλέον κάτω από 4.000 και συγκεκριμένα είναι περίπου 3.700, το πρόβλημα ότι οι περισσότερες λέξεις εμφανίζονται ελάχιστες φορές παραμένει. Αυτό το πρόβλημα όμως θα αντιμετωπιστεί στην συνέχεια.


The top 25 most frequently appearing tokens
8000
6000
4000
2000
0
a
.
in
the
on
dog
man
two
is
with
and
boy
people
girl
of
woman
dogs
playing
water
at
are
black
running
young
through
Token


The least 25 most frequently appearing tokens
1.0
0.8
0.6
0.4
0.2
0.0
underpass
stools
leafs
barbed
pine
trike
bloody
story
second
flyer
movie
fill
jeeps
villages
waterfalls
upstream
waterskies
brides
touts
goatee
yong
port-a-potties
cleans
boards
only
Token

Μετά τον καθαρισμό των κείμενων μας, παρατηρείται και ένα άλλο πρόβλημα. Αν εξαιρέσουμε το σύμβολο «.» που όπως είπαμε τοποθετείται στο τέλος κάθε πρότασης (και γι’ αυτό τον λόγο εμφανιζεται 8.000 φορές, όσες δηλαδή και τα captions μας), βλέπουμε ότι το άρθρο «a» είναι η κυρίαρχη λέξη μέσα στα κείμενα μας. Συγκεκριμένα εμφανιζεται πάνω από 4 φορές περισσότερο σε σχέση με την επόμενη πιο συχνή λέξη το «in».

Επειδή το λεξιλόγιο είναι και οι διαθέσιμες κλάσεις του μοντέλου, αυτό σημαίνει ότι η συγκεκριμένη λέξη δημιουργεί ένα τεράστιο imbalance στα δεδομένα μας. Επιπλέον, γνωρίζοντας ότι αν αφαιρεθεί από τα κείμενα, τότε το νόημα τους επηρεάζεται ελάχιστα, πάρθηκε η απόφαση να αφαιρεθεί αυτή η λέξη από τα συνολικά μας κείμενα.

Το τελικό στάδιο της διαδικασίας είναι η αντιμετώπιση των σπάνιων λέξεων. Για να την επίλυση του, αφού πρώτα έχουν διαμοιραστεί τα δεδομένα μας σε training, validation και test set, δημιουργούμε το vocabulary μας από το training set. Κατά την δημιουργία του, όσες λέξεις εμφανίζονται κάτω από 10 φορές αφαιρούνται, ενώ οι υπόλοιπες γίνονται map σε έναν αύξοντα αριθμό. Τόσο στο training όσο και στο validation και test set, όσες λέξεις δεν υπάρχουν μέσα στο vocabulary μας αντικαθίστανται από το ειδικό token «<UNK>».

Κατά το πέρας αυτής της διαδικασίας, μένουμε με ένα πλήθος από μοναδικές λέξεις, το οποίο προσεγγίζει τις 500, όπου και θα είναι και ο αριθμός από κλάσεις του μοντέλου.

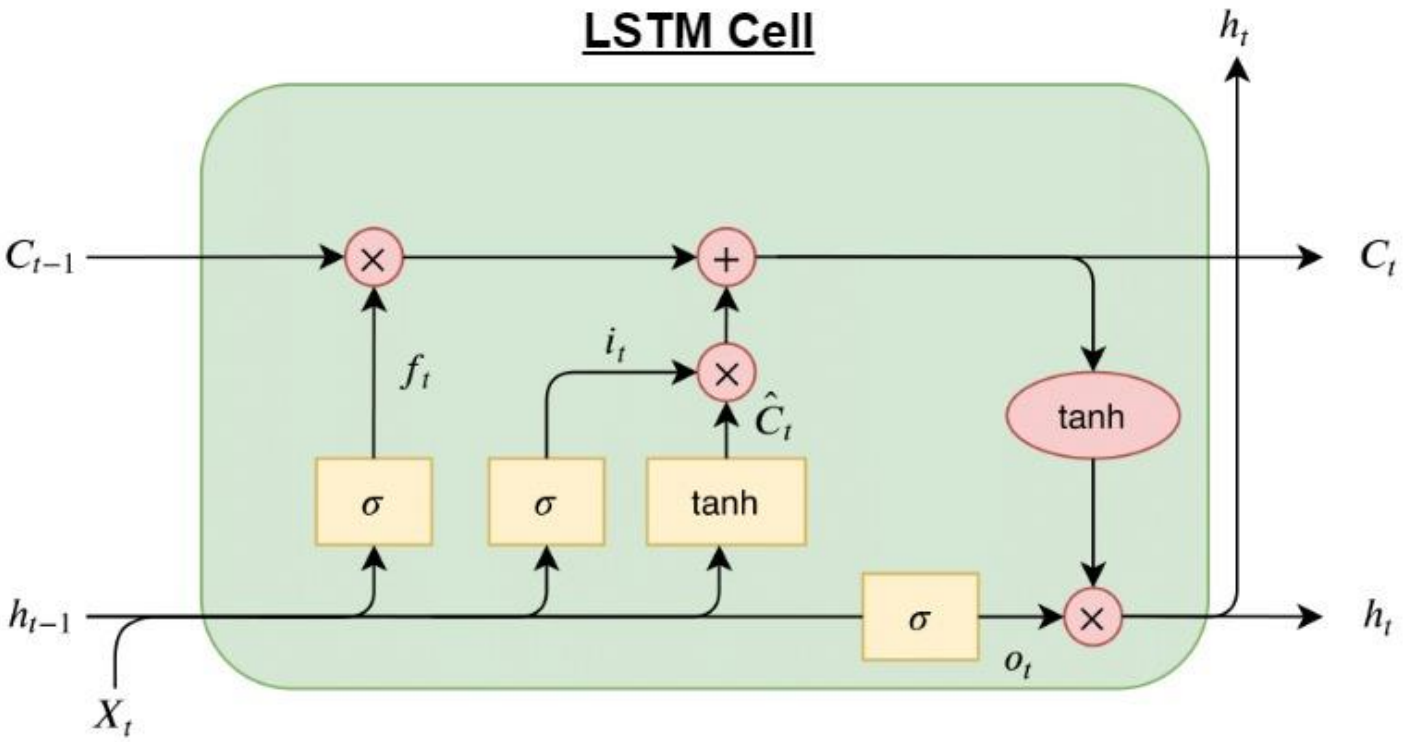
Υλοποιήσεις Μοντέλων

Έχοντας τα δεδομένα μας προεξεργασμένα και έτοιμα, παίρναμε στην φάση της εκπαίδευσης ενός μοντέλου πάνω στο συγκεκριμένο πρόβλημα. Το μοντέλο, αποτελείται από δυο υπο-μοντέλα, ένα CNN Encoder και έναν RNN Decoder, όπου και τα δυο εκπαιδεύονται end-to-end. Για τον CNN Encoder γίνεται χρήση Convolution και Linear Layers, ενώ ο Decoder αποτελείται από ένα Embedding, LSTM και μερικά Linear Layers.

Η δομή που θα ακολουθηθεί είναι ότι πρώτα θα περιγράφουν οι δυο κυρίες αρχιτεκτονικές με βάση τις οποίες μπορεί να εκπαιδευτεί ένα μοντέλο. Στην συνέχεια θα γίνει η σύγκριση τους, θα παρουσιαστεί ένας τρόπος που θα μπορούσε να βελτιώσει τα αποτελέσματα τους και τέλος θα δούμε τα αποτελέσματα από το καλύτερο μοντέλο και άλλη μια μέθοδο που θα μπορούσε να βελτιώσει επιπλέον τα αποτελέσματα του.

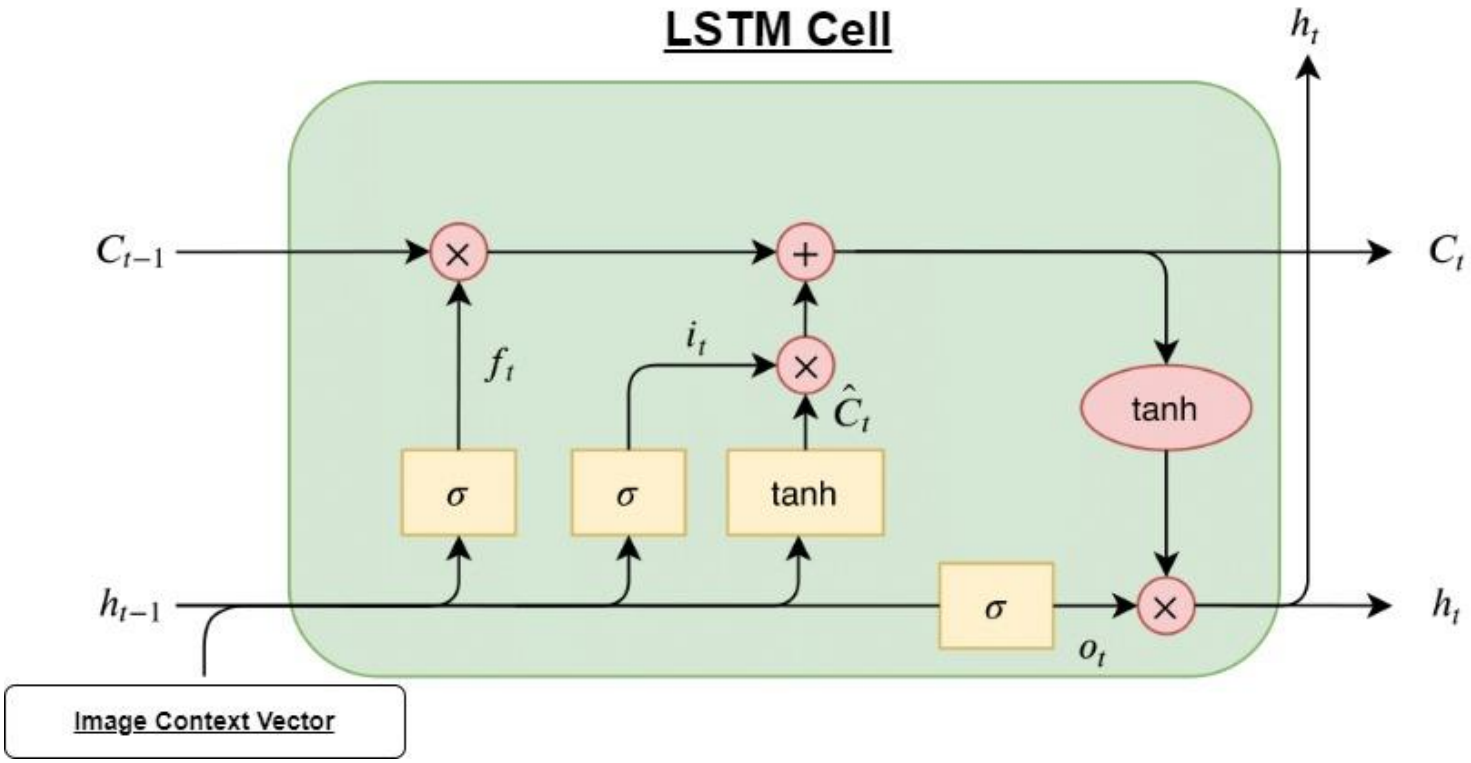
Τις δυο κύριες αρχιτεκτονικές θα τις ονομάσουμε Image context as embedding input και Image context as gate input.

Image Context as Embedding Input

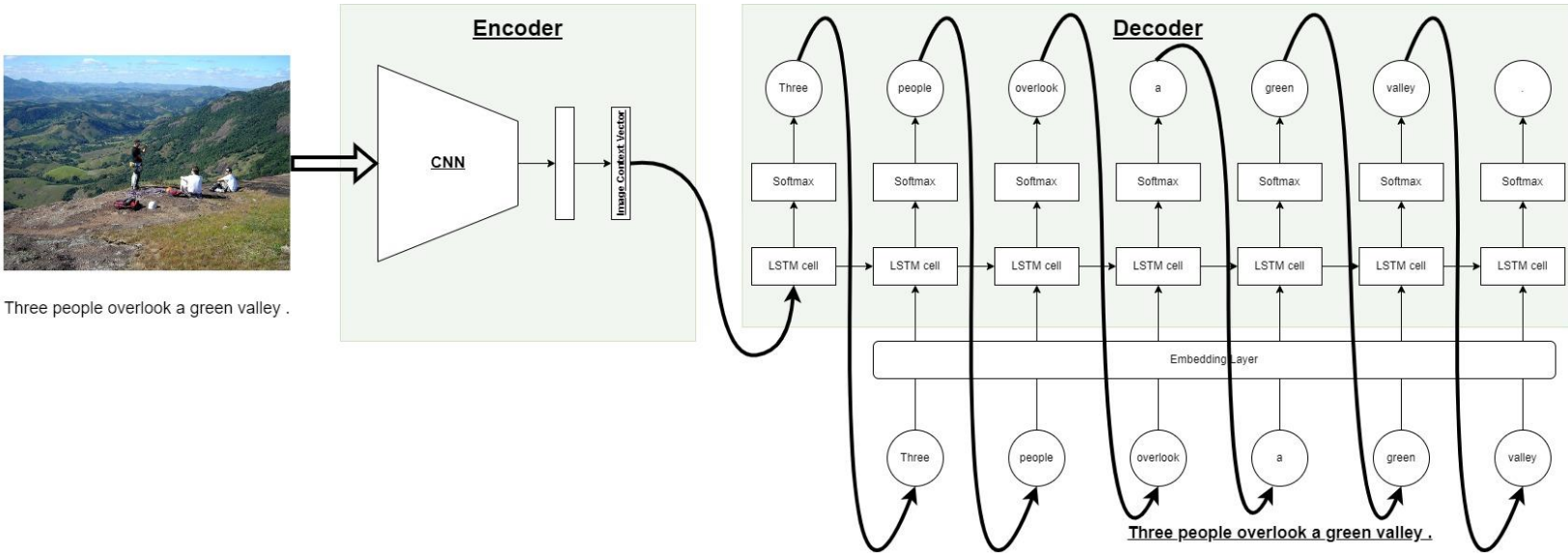


Όπως φαίνεται στην παραπάνω εικόνα, ένας τυπικός νευρώνας ενός LSTM, δέχεται τρεις εισόδους. Το X που περιέχει τα features της χρονικής στιγμής T, το H που είναι το hidden state και εξάγεται από την χρονική στιγμή T-1 και το C που είναι το cell state και ομοίως εξάγεται από την T-1.

Η συγκεκριμένη υλοποίηση κάνει χρήση της εισόδου X του νευρώνα κατά αυτόν τον τρόπο:

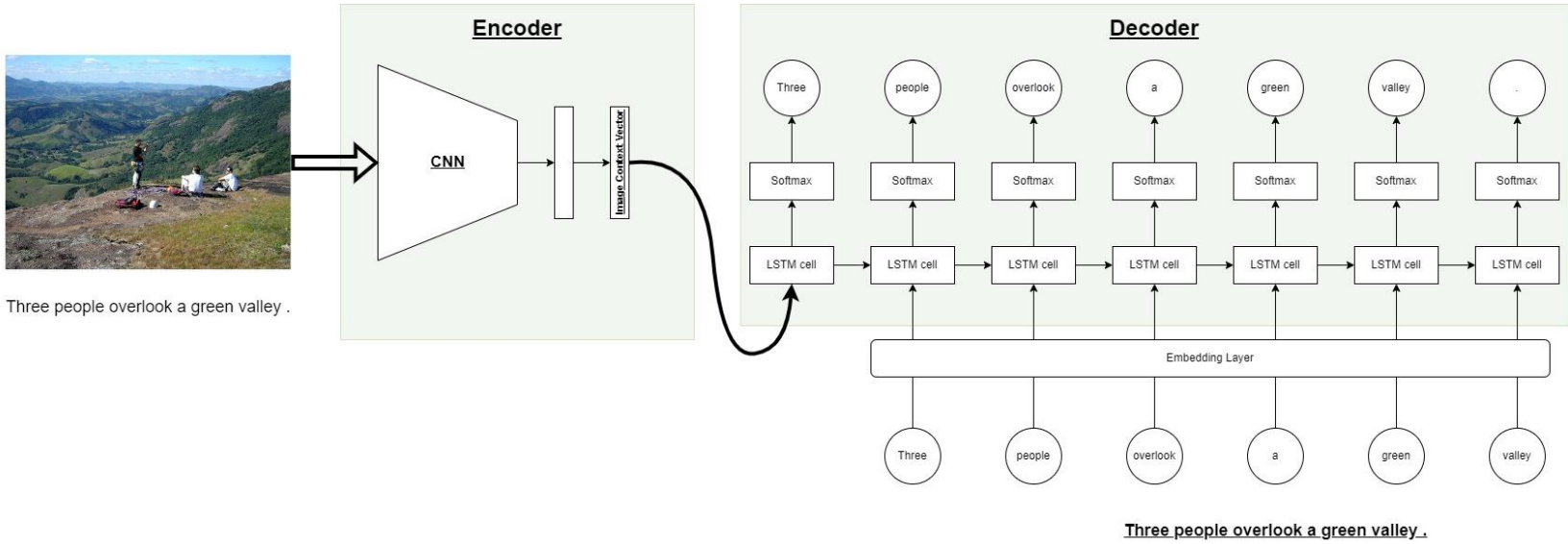


Με περισσότερες λεπτομέρειες, κατά το inference του μοντέλου, ο τρόπος που λειτουργεί είναι:



Το παραπάνω διάγραμμα περιγράφει σε ένα υψηλό επίπεδο, τα κύρια μέρη του μοντέλου. Όπως μπορούμε να παρατηρήσουμε, μια εικόνα εισάγεται στο μοντέλο μας και συγκεκριμένα στον Encoder του, όπου και εξάγεται από αυτήν ένα feature vector ή αλλιώς image context. Το context αυτό στην συνέχεια εισάγεται στον Decoder σαν το πρώτο στοιχείο της ακολουθίας μας και ο Decoder παράγει την πρώτη λέξη με βάση αυτό το context. Μετά, η λέξη αυτή μαζί με τα H και C που εξάγει το LSTM ξανά τροφοδοτούνται σε αυτό, ώστε να παραχθεί η επόμενη λέξη. Η διαδικασία αυτή συνεχίζεται μέχρι να παραχθεί το σύμβολο «.» ή αν ένας προκαθορισμένος αριθμός από λέξεις έχει παραχθεί.

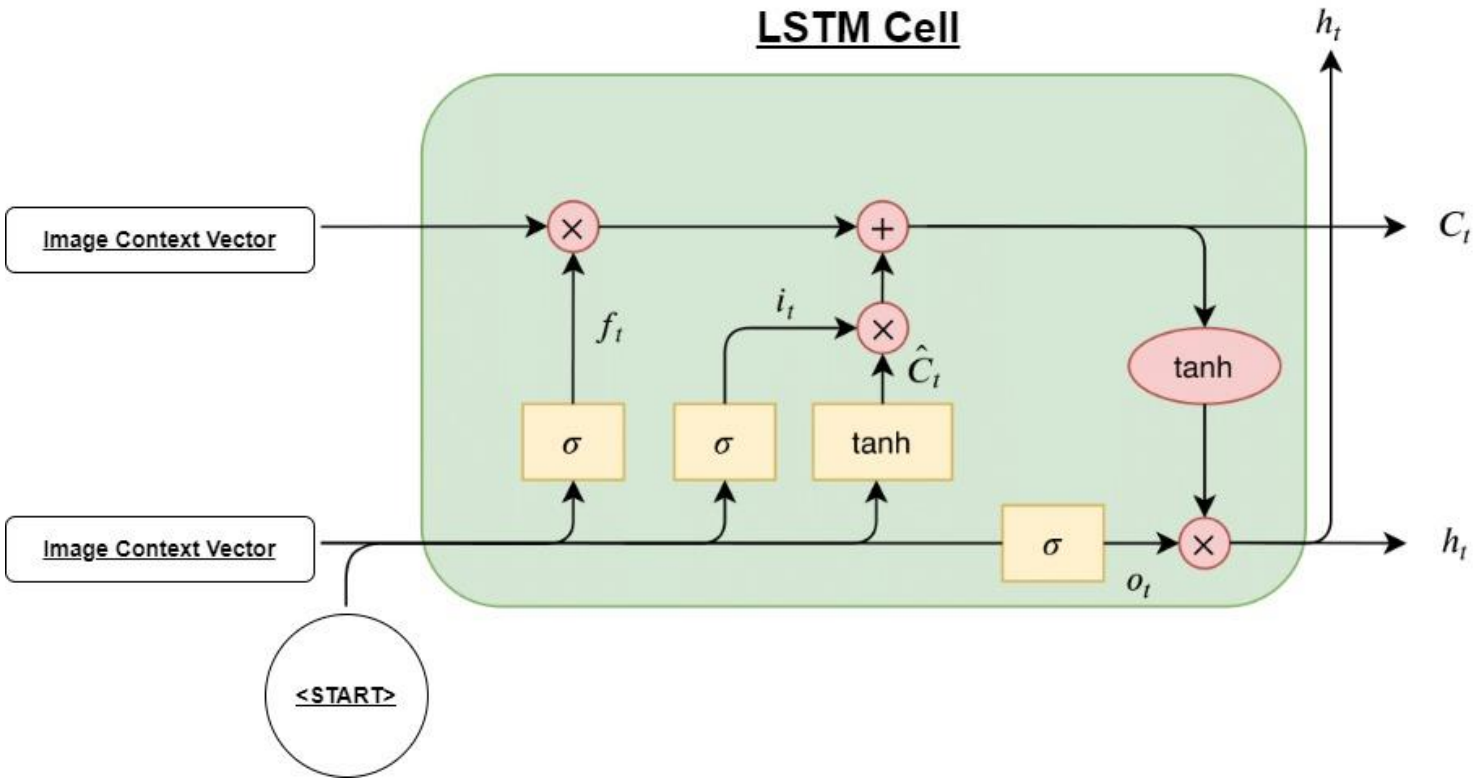
Κατά την εκπαίδευση όμως του μοντέλου, η παραπάνω διαδικασία εισάγει ένα πιθανό πρόβλημα. Κυρίως στα πρώτα στάδια της εκπαίδευσης όπου το μοντέλο παράγει λέξεις με λίγο τυχαίο τρόπο, ένα λάθος σε ένα σημείο της ακολουθίας θα επηρεάσει και όλες τις υπόλοιπες λέξεις που παράγονται. Για να αντιμετωπιστεί αυτό, γίνεται η χρήση της παρακάτω λογικής, όπου δηλαδή, όπως και πριν, χρησιμοποιούμε το μοντέλο για να μας παράξει μια λέξη σε κάθε χρονική στιγμή και στην συνέχεια με αυτές τις προβλέψεις κάνουμε ενημέρωση των βαρών του μοντέλου. Για να παραχθεί όμως κάθε επόμενη λέξη, δίνουμε σαν είσοδο την προηγουμένη σωστή λέξη που έχουμε ήδη από το caption που υπάρχει στα training δεδομένα.



Τα θετικά αυτής της αρχιτεκτονικής είναι η σχετικά εύκολη υλοποίηση της από πλευράς κώδικα, καθώς και η ελευθέρια αλλαγής και πειραματισμού με τις παραμέτρους του μοντέλου, όπως τα Layers, πλήθος νευρώνων κλπ. Το τελευταίο βασίζεται στο γεγονός ότι το context της εικόνας αποτελεί απλά το πρώτο στοιχείο που εισάγεται στον Decoder και έτσι δεν επηρεάζεται από τυχόν αλλαγές που γίνονται στα δυο υπο-μοντέλα.

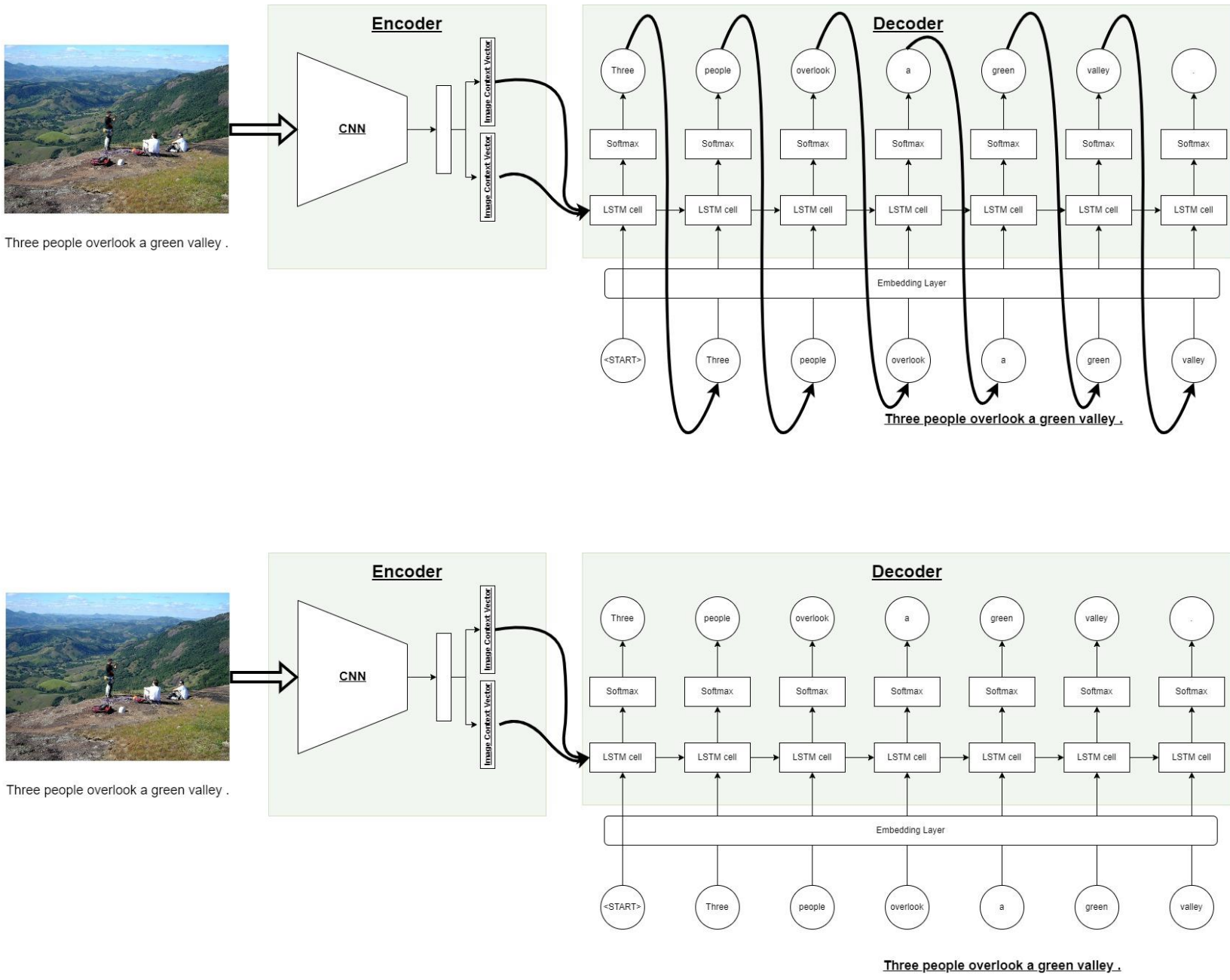
Το κύριο όμως αρνητικό της είναι ότι πρέπει το feature vector που εξάγεται από την εικόνα να έχει τις ίδιες διστάσεις με τα word embeddings που βγαίνουν από το Embedding Layer. Αν σκεφτούμε ότι τυπικά οι διαστάσεις των word embeddings είναι μικρές, της τάξης των 100, 200 ή 300 τότε καταλαβαίνουμε ότι όλο το context μια εικόνας θα πρέπει να αναπαρασταθεί από ένα αρκετά μικρό πλήθος τιμών.

Image Context as Gate Input



Η συγκεκριμένη αρχιτεκτονική, στοχεύει στο να αντιμετωπίσει το κύριο αρνητικό της προηγουμένης, κάνοντας χρήση των δυο gates ή states του νευρωνικού ενός LSTM. Για να πραγματοποιηθεί όμως αυτό, όπως μπορούμε να παρατηρήσουμε και στην παραπάνω εικόνα, θα πρέπει να εξάγουμε δυο Image context vectors, καθώς και να περάσουμε κάτι από την X είσοδο του νευρώνα. Ως πρώτο στοιχείο της ακολουθίας και ως πρώτη X είσοδος του LSTM χρησιμοποιείται ένα προκαθορισμένο token το «<START>».

Τα παρακάτω διαγράμματα περιγράφουν την λογική με την οποία γίνεται το inference και το training με την χρήση αυτής της αρχιτεκτονικής.



Πλέον το Image context vector δεν περιορίζεται από το μέγεθος των word embeddings και μπορεί να έχει οποιαδήποτε διάσταση θέλουμε, αρκεί να ισούται με τις διαστάσεις του output του LSTM.

Σύγκριση αποτελεσμάτων

Για την σύγκριση των αποτελεσμάτων μεταξύ των δυο υλοποιήσεων, οι παράμετροι των μοντέλων παρέμειναν ίδιοι με σκοπό η μοναδική διαφορά να αποτελεί τον τρόπο με τον οποίο εισάγεται το image context vector προς τον Decoder.

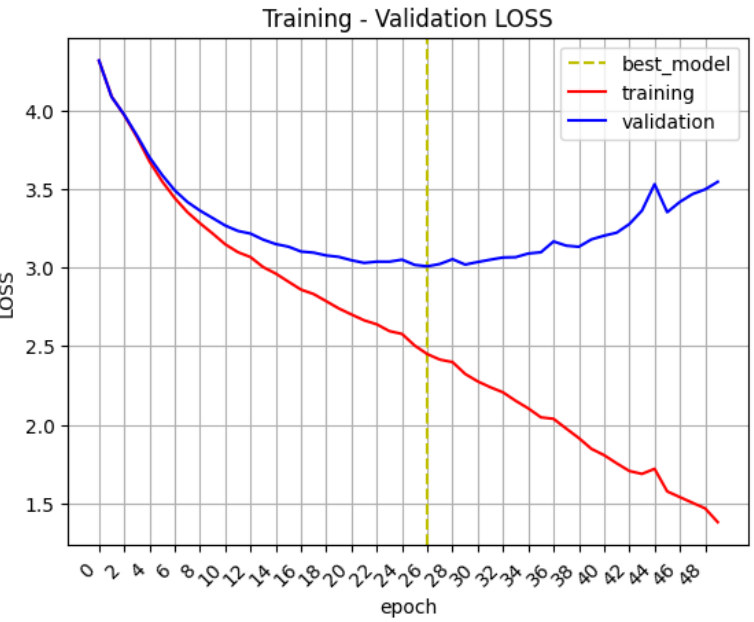
Αριστερά θα απεικονίζονται τα αποτελέσματα από την πρώτη υλοποίηση (Image context as embedding input), ενώ στα δεξιά από την δεύτερη (Image context as gate input). Να σημειωθεί ότι τόσο ο αριθμός των εποχών, όσο και το Learning Rate, ο optimizer, το batch size και το Loss Function ήταν ίδια. Συγκεκριμένα:

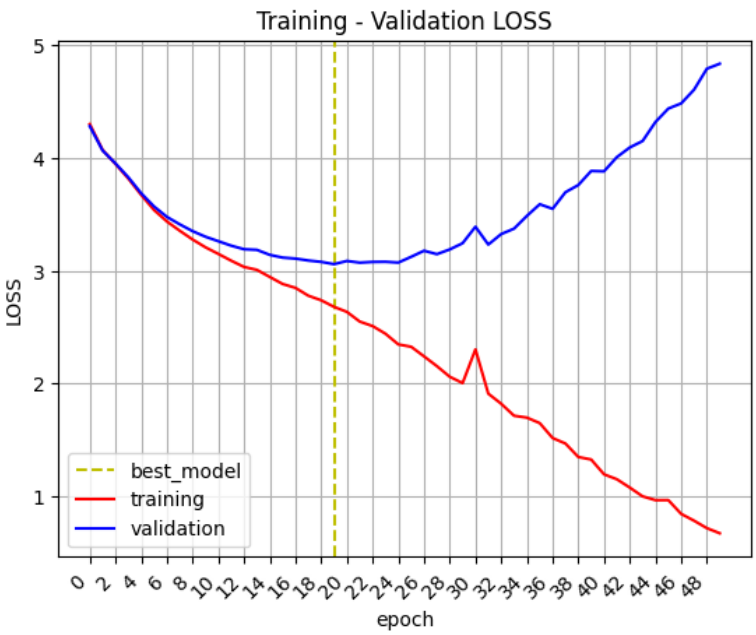
```
"epochs": 50,
"batch_size": 256,
"lr": 0.0001,
"weight_decay": 0,
"decoder_params": {
        "embed_size": 300,
        "lstm_hidden_size": 2048,
        "num_layers": 1,
        "dropout_prob": 0.1,
        "vocab_size": 479
},
"encoder_params": {
        "dropout_prob": 0.0
},
```

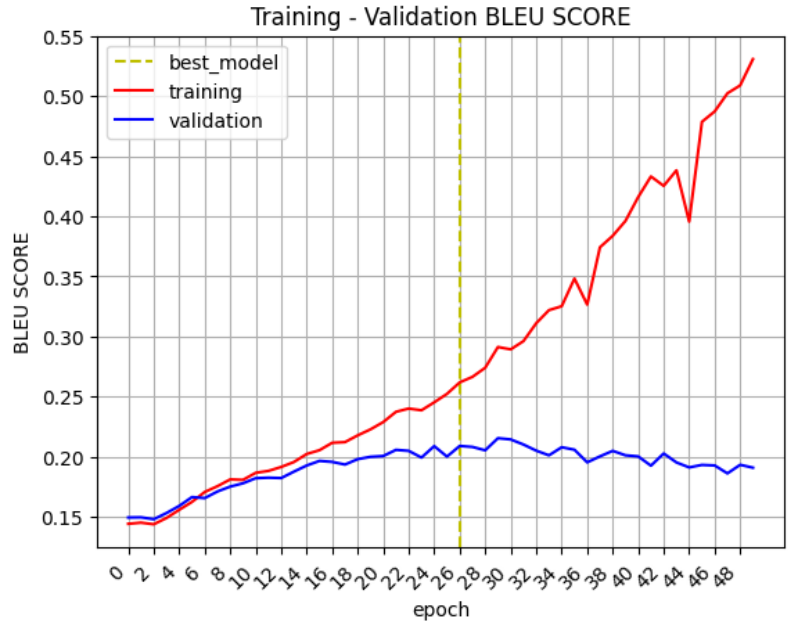
Ως Loss function χρησιμοποιήθηκε το Cross Entropy Loss και ως evaluation metric της απόδοσης των μοντέλων το BLEU score με αριθμό n grams να ισούται με 2.

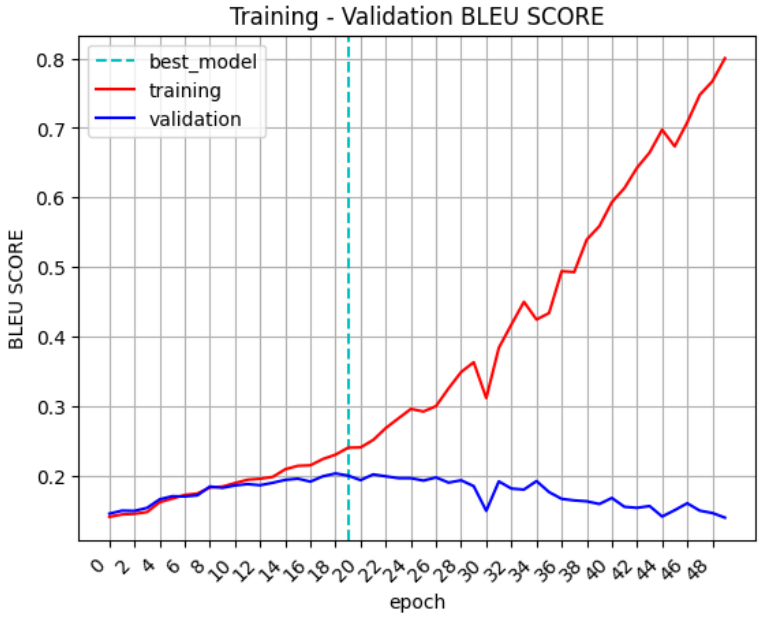
```
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
ImageCaptioningModel                     [256, 14, 478]            --
├─Encoder: 1-1                           [256, 300]                --
│    └─Conv2d: 2-1                       [256, 16, 252, 252]       2,368
│    └─BatchNorm2d: 2-2                  [256, 16, 252, 252]       32
│    └─ReLU: 2-3                         [256, 16, 252, 252]       --
│    └─MaxPool2d: 2-4                    [256, 16, 125, 125]       --
│    └─Conv2d: 2-5                       [256, 32, 123, 123]       12,832
│    └─BatchNorm2d: 2-6                  [256, 32, 123, 123]       64
│    └─ReLU: 2-7                         [256, 32, 123, 123]       --
│    └─MaxPool2d: 2-8                    [256, 32, 41, 41]         --
│    └─Conv2d: 2-9                       [256, 64, 41, 41]         18,496
│    └─BatchNorm2d: 2-10                 [256, 64, 41, 41]         128
│    └─ReLU: 2-11                        [256, 64, 41, 41]         --
│    └─MaxPool2d: 2-12                   [256, 64, 13, 13]         --
│    └─Linear: 2-13                      [256, 300]                3,245,100
│    └─ReLU: 2-14                        [256, 300]                --
├─Decoder: 1-2                           [256, 14, 478]            --
│    └─Embedding: 2-15                   [256, 13, 300]            143,400
│    └─LSTM: 2-16                        [1782, 2048, 1]           19,251,200
│    └─Sequential: 2-17                  [256, 14, 478]            --
│    │    └─Linear: 3-1                  [256, 14, 1024]           2,098,176
│    │    └─ReLU: 3-2                    [256, 14, 1024]           --
│    │    └─Dropout: 3-3                 [256, 14, 1024]           --
│    │    └─Linear: 3-4                  [256, 14, 478]            489,950
==========================================================================
Total params: 25,261,746
Trainable params: 25,261,746
```

```
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
ImageCaptioningModel                     [72, 11, 479]             --
├─Encoder: 1-1                           [72, 2048, 1]             --
│    └─Conv2d: 2-1                       [72, 16, 252, 252]        2,368
│    └─BatchNorm2d: 2-2                  [72, 16, 252, 252]        32
│    └─ReLU: 2-3                         [72, 16, 252, 252]        --
│    └─MaxPool2d: 2-4                    [72, 16, 125, 125]        --
│    └─Conv2d: 2-5                       [72, 32, 123, 123]        12,832
│    └─BatchNorm2d: 2-6                  [72, 32, 123, 123]        64
│    └─ReLU: 2-7                         [72, 32, 123, 123]        --
│    └─MaxPool2d: 2-8                    [72, 32, 41, 41]          --
│    └─Conv2d: 2-9                       [72, 64, 41, 41]          18,496
│    └─BatchNorm2d: 2-10                 [72, 64, 41, 41]          128
│    └─ReLU: 2-11                        [72, 64, 41, 41]          --
│    └─MaxPool2d: 2-12                   [72, 64, 13, 13]          --
│    └─Linear: 2-13                      [72, 2048]                22,153,216
│    └─ReLU: 2-14                        [72, 2048]                --
│    └─Dropout: 2-15                     [72, 2048]                --
│    └─Linear: 2-16                      [72, 2048]                22,153,216
│    └─ReLU: 2-17                        [72, 2048]                --
│    └─Dropout: 2-18                     [72, 2048]                --
├─Decoder: 1-2                           [72, 11, 479]             --
│    └─Embedding: 2-19                   [72, 11, 300]             143,700
│    └─LSTM: 2-20                        [499, 2048, 1]            19,251,200
│    └─Sequential: 2-21                  [72, 11, 479]             --
│    │    └─Linear: 3-1                  [72, 11, 1024]            2,098,176
│    │    └─ReLU: 3-2                    [72, 11, 1024]            --
│    │    └─Dropout: 3-3                 [72, 11, 1024]            --
│    │    └─Linear: 3-4                  [72, 11, 479]             490,975
==========================================================================
Total params: 66,324,403
Trainable params: 66,324,403
```









Test loss: 2.907, Test bleu: 0.21

Test loss: 2.962, Test bleu: 0.2

Αρχικά μπορούμε να παρατηρήσουμε ότι και τα δυο μοντέλα έχουν κάνει overfit πολύ πιο πριν από το τέλος του training. Παρόλα αυτά 50 εποχές επιλέχθηκαν, ώστε να μπορούμε να δούμε την συνολική εικόνα των μοντέλων. Τα μοντέλα είναι λογικό να κάνουν overfit γιατί το dataset μας είναι σχετικά μικρό για το συγκεκριμένο πρόβλημα, ενώ τα χαμηλά BLEU scores στο validation είναι εξίσου λογικά, λόγω του μεγέθους του dataset. Με μόλις 6.400 εικόνες – captions είναι αρκετά δύσκολο να παραχθεί ένα γενικό μοντέλο, που να παράγει υψηλής ποιότητας captions με από εικόνες που δεν έχει ξανά δει.

Επίσης, δεν φαίνεται κάποια αισθητή διαφορά μεταξύ των δυο υλοποιήσεων όσον αφορά το Loss και το BLEU score τόσο του validation set όσο και του test set. Το χαμηλότερο Loss φαίνεται να είναι κοντά στο 3.0, ενώ το καλύτερο BLEU score, λίγο πιο πάνω από το 20%. Η μοναδική διαφορά φαίνεται να είναι, το ποσό γρηγορά το μοντέλο κάνει overfit στα training δεδομένα. Στην δεύτερη υλοποίηση, το μοντέλο κάνει γρηγορότερα overfit, το οποίο είναι λογικό, αφού το image context που εισάγεται στον Decoder είναι μεγαλύτερο και άρα μπορεί να μάθει γρηγορότερα τα σωστά captions που πρέπει να παράξει.

Ως ένα επιπλέον ποιοτικό μέτρο σύγκρισης των αποτελεσμάτων, επιλέχθηκε το πρώτο δεδομένο από το validation set και εξάχθηκε το caption που παράγει το κάθε μοντέλο για αυτό, ανά εποχή. Με αυτόν τον τρόπο μπορούμε να δούμε την συμπεριφορά τους ανά εποχή. Για παράδειγμα, πόσο γρήγορα μαθαίνουν την σωστή σύνταξη μιας πρότασης (και όχι απλά να τοποθετούν τυχαίες λέξεις), πόσο γρήγορα φαίνεται να «καταλαβαίνουν» στοιχεία που υπάρχουν μέσα στην εικόνα και πόσο γρήγορα επηρεάζονται από το overfitting. Η εικόνα είναι η παρακάτω, με το σωστό caption να αναφέρει: «three dogs on grassy hill .»

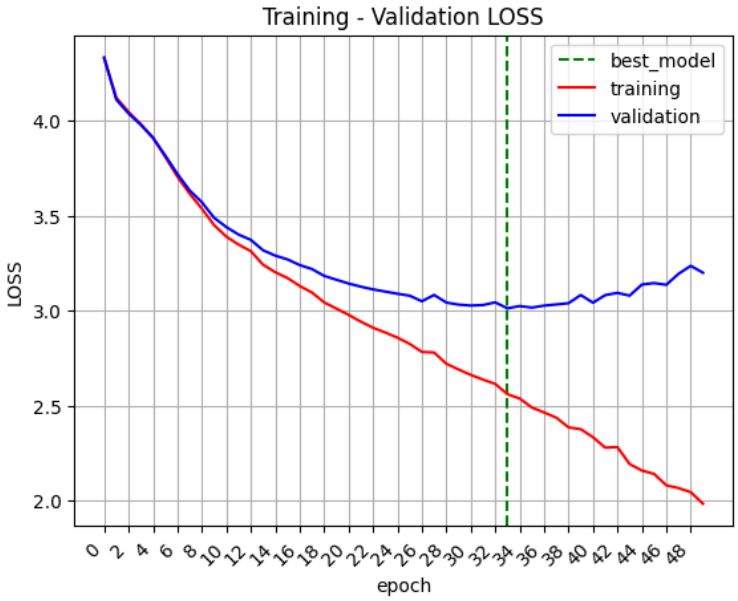
Epoch/Implementation	**Image Context as Embedding Input**	**Image Context as Gate Input**
1	Man <UNK> <UNK> .	Two <UNK> <UNK> .
2	Man <UNK> <UNK> <UNK> .	Two <UNK> <UNK> <UNK> .
3	Man <UNK> <UNK> <UNK> .	Man <UNK> <UNK> <UNK> .
4	Man <UNK> <UNK> <UNK> .	Man <UNK> <UNK> .
5	Two dogs <UNK>	Man <UNK> <UNK> .
6	Man <UNK> <UNK> <UNK> .	Dog is <UNK> in the snow .
7	Dog is <UNK> <UNK> .	Dog is running in the snow .
8	Dog is running in the grass .	Dog is running in the grass .
9	Two dogs playing in the snow .	Dog runs through the grass .
10	Dog runs through the grass .	Dog <UNK> in the snow .
20	Two dogs play in the water .	Man <UNK> on the beach .
30	Dog runs through the grass .	Dog jumps over the water .
40	Dog runs through the sand .	Two dogs play in the grass .
50	Dog runs through the grass .	Dog runs through the grass .

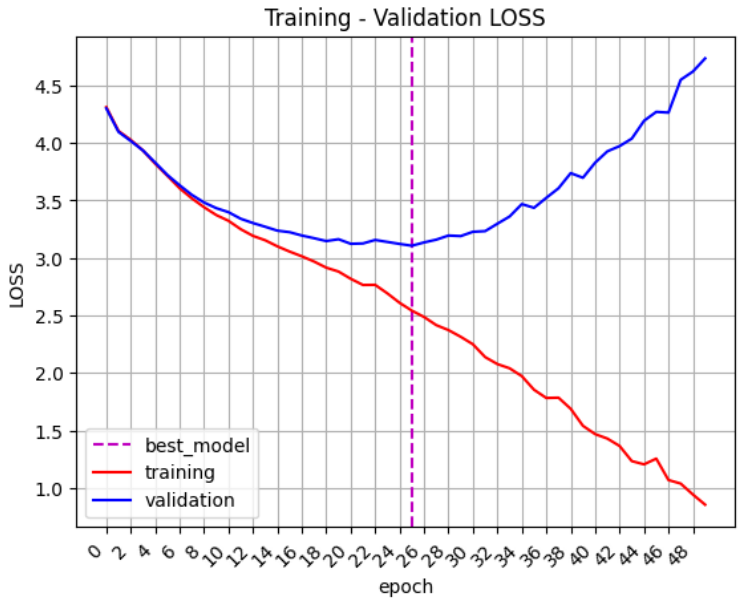
Αυτό επιβεβαιώνει τα προηγούμενα αποτελέσματα, αφού φαίνεται ότι με την δεύτερη υλοποίηση, το μοντέλο είναι σε θέση να καταλάβει νωρίτερα κάποια στοιχεία που υπάρχουν μέσα στην εικόνα, αλλά ταυτόχρονα βλέπουμε ότι λόγω του γρηγορότερου overfitting, όσο προχωράνε οι εποχές, τα αποτελέσματα γίνονται κάπως τυχαία.

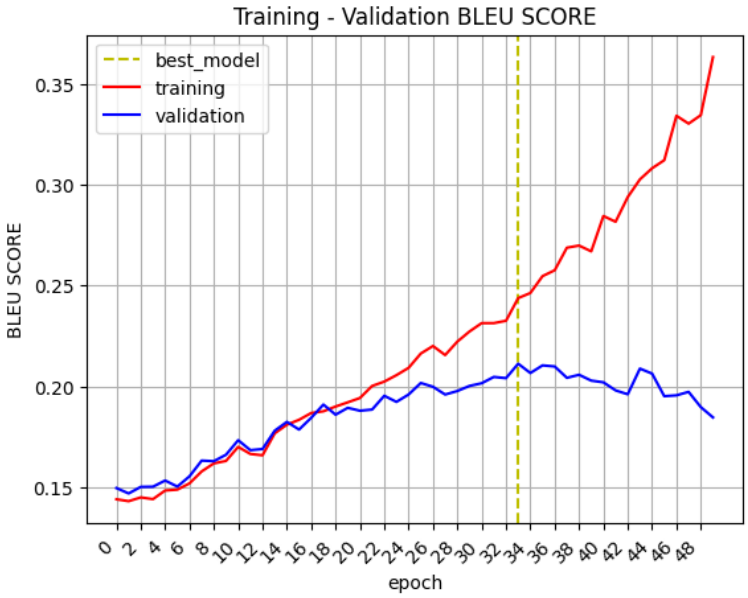
Χρήση pre-trained Embeddings

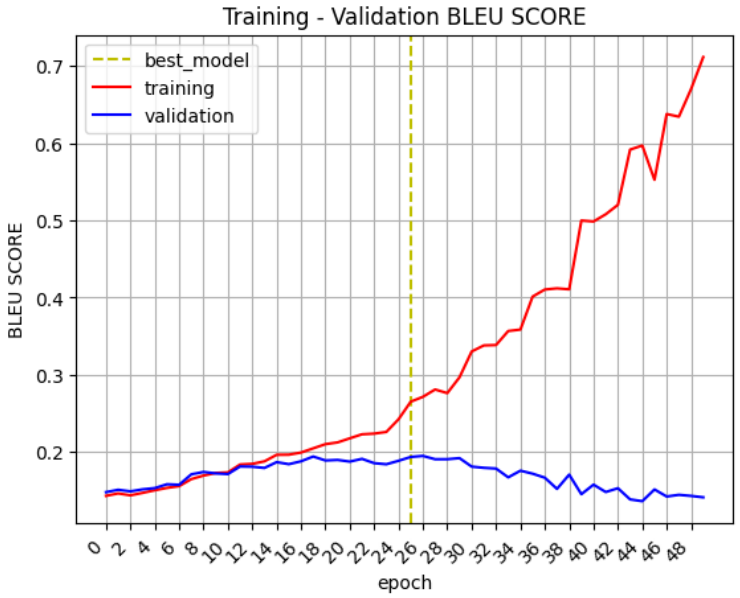
Ως μια πιθανή βελτίωση των αποτελεσμάτων, δοκιμάστηκε η χρήση Pre-trained word embeddings. Συγκεκριμένα δοκιμάστηκαν τα GloVe 6B και 42B με διάσταση 300. Τα παρακάτω αποτελέσματα έχουν παραχθεί με την χρήση των GloVe 42B, αλλά η κατάσταση ήταν ίδια και με τα 6B.

Ομοίως με πριν, αριστερά θα απεικονίζονται τα αποτελέσματα από την πρώτη υλοποίηση (Image context as embedding input), ενώ στα δεξιά από την δεύτερη (Image context as gate input).









Test loss: 2.913, Test bleu: 0.209

Test loss: 3.016, Test bleu: 0.199

Φαίνεται ότι η χρήση των pre-trained embeddings δεν απέδωσε κάποια βελτίωση στα αποτελέσματα των μοντέλων. Να σημειωθεί ότι δεν ήταν δυνατό να γίνουν non-trainable τα embeddings καθώς αυτό θα σήμαινε ότι το backpropagation θα σταματούσε στον Decoder και έτσι ο Encoder δεν θα έκανε update τα βάρη του.

Τα αποτελέσματα είναι άμεσα συγκρίσιμα με τα προηγούμενα, αφού η αρχιτεκτονική και οι παράμετροι των μοντέλων παρέμειναν ακριβώς τα ίδια.

Αποτελέσματα καλυτέρου μοντέλου

Παρόλο που τα αποτελέσματα μεταξύ των δυο αρχιτεκτονικών φαίνεται να είναι παρόμοια, από αυτό το σημείο και έπειτα, ως καλύτερη προτάθηκε η δεύτερη. Η δεύτερη δίνει την μεγαλύτερη ευελιξία, αφού επιτρέπει το image context vector να πάρει πιο εύκολα μεγαλύτερες τιμές.

Σε αυτή την ενότητα παρουσιάζονται οι top 5 καλύτερες και χειρότερες, από άποψη BLEU score, εικόνες από το test set, με σκοπό να έχουμε μια γενική αίσθηση των αποτελεσμάτων του μοντέλου.

Οι 5 με το υψηλότερο BLEU score:

Image name: 1189977786_4f5aaed773.jpg

True Caption: dog jumping into swimming pool .

Model Caption: dog jumping into pool .

BLEU SCORE: 0.709

Image name: 874665322_9ad05c4065.jpg

True Caption: dog runs through the grass .

Model Caption: dog runs in the grass .

BLEU SCORE: 0.707

Image name: 3265162450_5b4e3c5f1b.jpg

True Caption: three dogs running through the snow .

Model Caption: dog jumps through the snow .

BLEU SCORE: 0.535

Image name: 944374205_fd3e69bfca.jpg

True Caption: children play soccer in field .

Model Caption: three dogs play in field .

BLEU SCORE: 0.516

Image name: 3602838407_bf13e49243.jpg

True Caption: black dog in water .

Model Caption: brown dog playing in water .

BLEU SCORE: 0.516

Για να εντοπιστούν οι 5 με το χαμηλότερο BLEU score, από το test set, αφαιρέθηκαν πρώτα οι εικόνες όπου το μοντέλο είχε BLEU score ίσο με 0. Αυτό συνέβη καθώς δεν θα υπήρχε κάτι άξιο σχολιασμού, αφού τα παραγόμενα captions θα ήταν τυχαία.

Image name: 1362128028_8422d53dc4.jpg

True Caption: kids play in blue tub full of water outside .

Model Caption: dog <unk> through of water .

BLEU SCORE: 0.162

Image name: 2584957647_4f9235c150.jpg

True Caption: the white dog brings stick from the water .

Model Caption: white dog swims .

BLEU SCORE: 0.143

Image name: 2066048248_f53f5ef5e2.jpg

True Caption: two men and some horses on snowy mountain .

Model Caption: man <unk> up mountain .

BLEU SCORE: 0.142

Image name: 3621741935_54d243f25f.jpg

True Caption: little girl eats ice cream near her bike .

Model Caption: person on bmx bike .

BLEU SCORE: 0.142

Image name: 3327487011_1372c425fb.jpg

True Caption: group of children mostly wearing white uniform shirts sit and wait .

Model Caption: group of people are <unk> up <unk> .

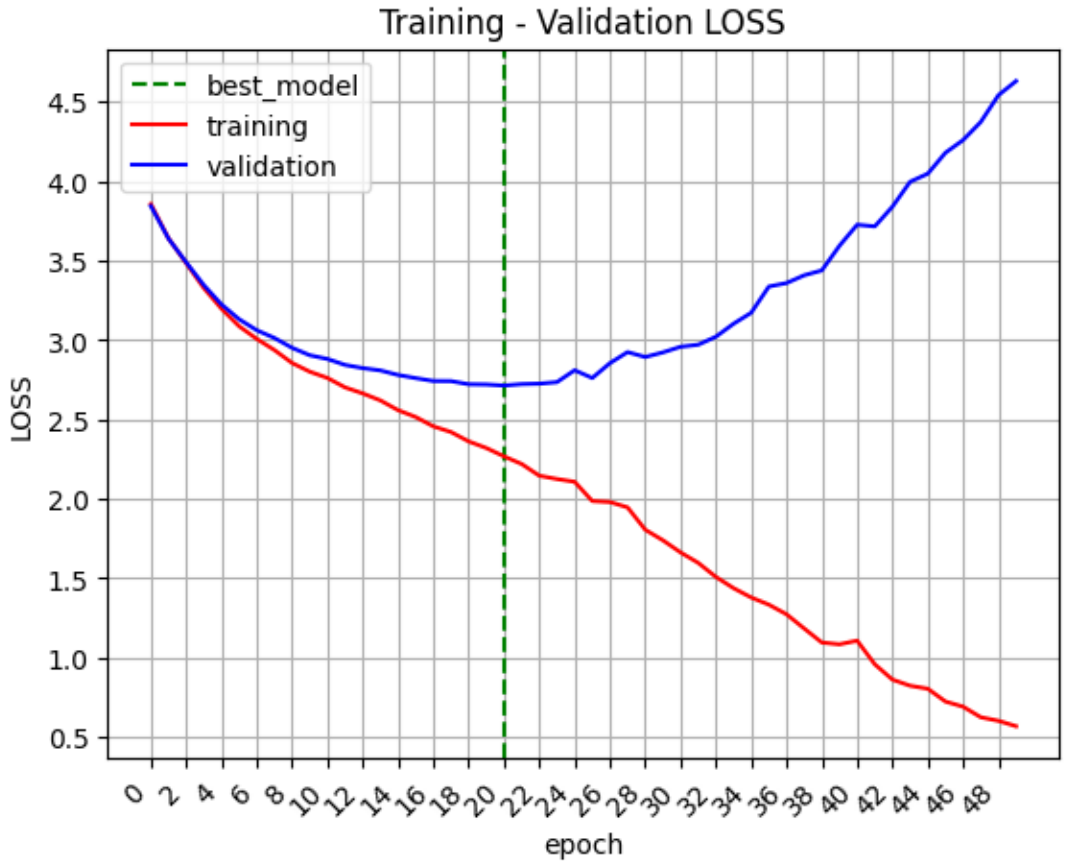
BLEU SCORE: 0.14

Ιδιαίτερο ενδιαφέρον υπάρχει, καθώς παρόλο που τα παραγόμενα captions έχουν διαφορά από τα πραγματικά, το μοντέλο φαίνεται να έχει «καταλάβει» το νόημα που βρίσκεται μέσα σε αυτές. Στην δεύτερη κατάλαβε ότι βρίσκεται ένα άσπρο σκυλί το οποίο είναι μέσα στο νερό. Στην τρίτη ότι υπάρχει βουνό μέσα στην εικόνα. Στην τέταρτη ότι υπάρχει ένας άνθρωπος που κάθεται πάνω σε ένα ποδήλατο. Ενώ στην τελευταία, ότι πρόκειται για ένα group ανθρώπων.

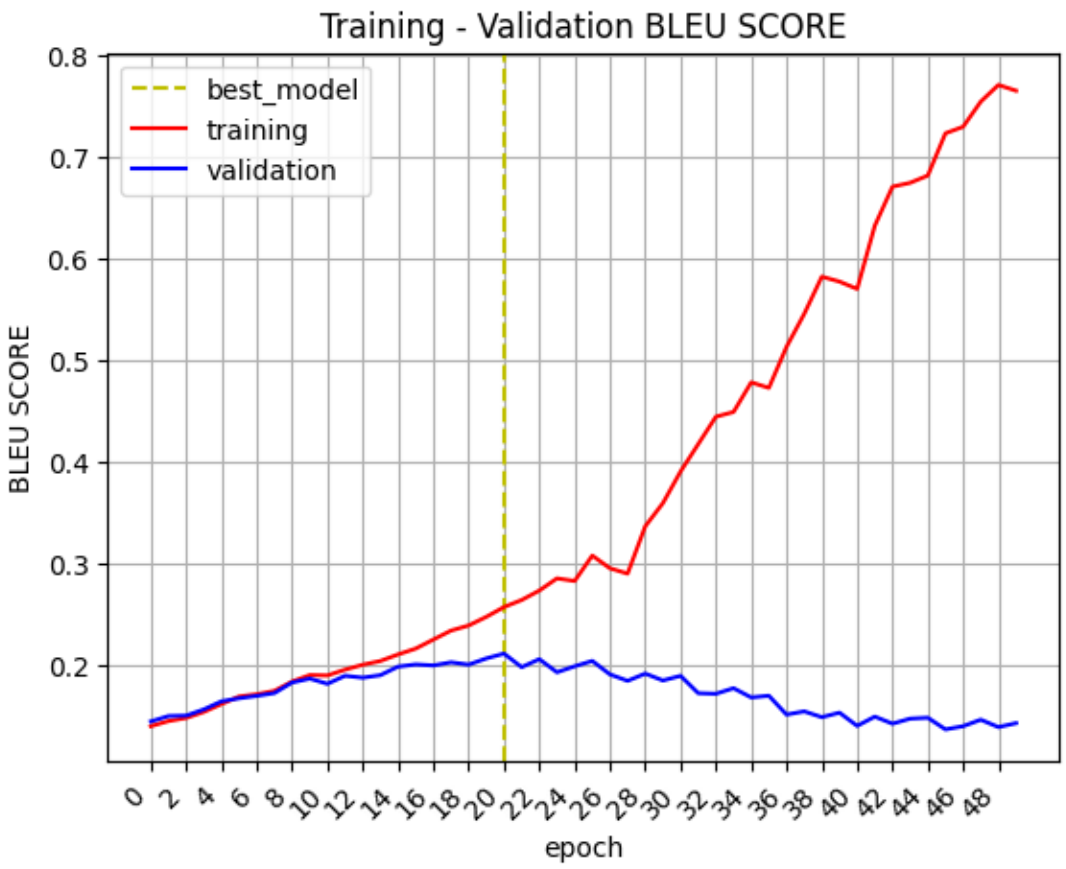
Asymmetric Loss

Όπως παρατηρήσαμε και στην προεξεργασία των κειμένων, υπάρχει ένα μεγάλο Imbalance στην εμφάνιση των λέξεων. Οι περισσότερες λέξεις εμφανίζονται ελάχιστες φορές, ενώ μόνο κάποιες συγκεκριμένες εμφανίζονται αρκετές φορές.

Η τελευταία δοκιμή για την βελτίωση των αποτελεσμάτων, στοχεύει στην αντιμετώπιση αυτού του προβλήματος, αντικαθιστώντας το Cross entropy Loss από το Asymmetric Loss[1][2]. Το συγκεκριμένο Loss επιτρέπει τη δυναμική μείωση της σημασίας και την τοποθέτηση ενός σκληρού threshold στα εύκολα δειγμάτων, ενώ απορρίπτει επίσης πιθανώς λανθασμένα επισημασμένα δείγματα. Η δοκιμή έγινε πάλι με την χρήση της δεύτερης υλοποίησης.



[1] Paper: https://arxiv.org/pdf/2009.14119.pdf
[2] GitHub: https://github.com/Alibaba-MIIL/ASL



Test loss: 2.631, Test bleu: 0.213

Σε σχέση με τα προηγούμενα αποτελέσματα, παρατηρείται μια αισθητή βελτίωση στο Loss, που πλέον αγγίζει το 2.6, ενώ επιπλέον παρατηρείται μια βελτίωση (αν και σχετικά ελάχιστη) και στο BLEU score, τόσο στο validation όσο και στο test set.

Φαίνεται λοιπόν ότι το συγκεκριμένο Loss αποδίδει καλυτέρα σε σχέση με το Cross Entropy. Ο λόγος της βελτίωσης, πιθανόν μπορεί να αποδοθεί στο γεγονός ότι μπορεί να αντιμετωπίσει καλυτέρα το πρόβλημα του class imbalance που υπάρχει στα δεδομένα.