Τελικο Project Στην Νευροασαφη Υπολογιστικη

ΒΑΣΙΛΗΣ ΣΤΕΡΓΙΟΥΛΗΣ 03166

ΓΙΩΡΓΟΣ ΚΑΠΑΚΟΣ 03165

Πίνακας περιεχομένων

Αν θελήσετε να δείτε τις υπόλοιπες παραλλαγές υλοποίησης της εργασίας , μπορείτε να μεταβείτε στο github της ομάδας , το οποίο παραθέτουμε στην τελευταία σελίδα της αναφοράς

DATASET

Το Σετ Δεδομένων , πάνω στο οποίο γίνεται η ταξινόμηση των άρθρων , αποτελείται από 17 κατηγορίες 1^{ου} επιπέδου (*Σχήμα 1.1 όπως φαίνεται και από την κατανομή των άρθρων στις κατηγορίες αυτές*) , 109 υποκατηγορίες 2^{ου} επιπέδου ή υποκατηγορίες και συνολικά από 10917 ειδησεογραφικά άρθρα . Στην πραγματικότητα και έπειτα από ενδελεχή έλεγχο του Σετ (*μαζί με αρκετές ώρες αναζήτησης των σφαλμάτων που προέκυπταν*) , καταλήξαμε στο ότι το Σετ αποτελείται εν τέλει από 10915 άρθρα , καθώς το άρθρο υπ' αριθμόν 371 (*αριθμώντας από το 0 , βλέπε Εικόνα 1.1*) που ανήκει στην κατηγορία *Lifestyle and Leisure* εμφανίζεται συνολικά 2 φορές (*και ως εγγραφή 374*) και το άρθρο 6528 της κατηγορίας *Sport* , είναι ένα κενό άρθρο που αποτελείτε από δύο χαρακτήρες που δηλώνουν αλλαγή σειράς (*βλέπε Εικόνα 1.2*) . Ενδεικτικά , μερικές από τις κατηγορίες 1^{ου} είναι οι εξής :

(A) *Καιρός ή Weather* ,

(B) *Έγκλημα , νόμος και Δικαιοσύνη ή Crime , law and Justice*

Και μερικές από τις κατηγορίες 2^{ου} επιπέδου :

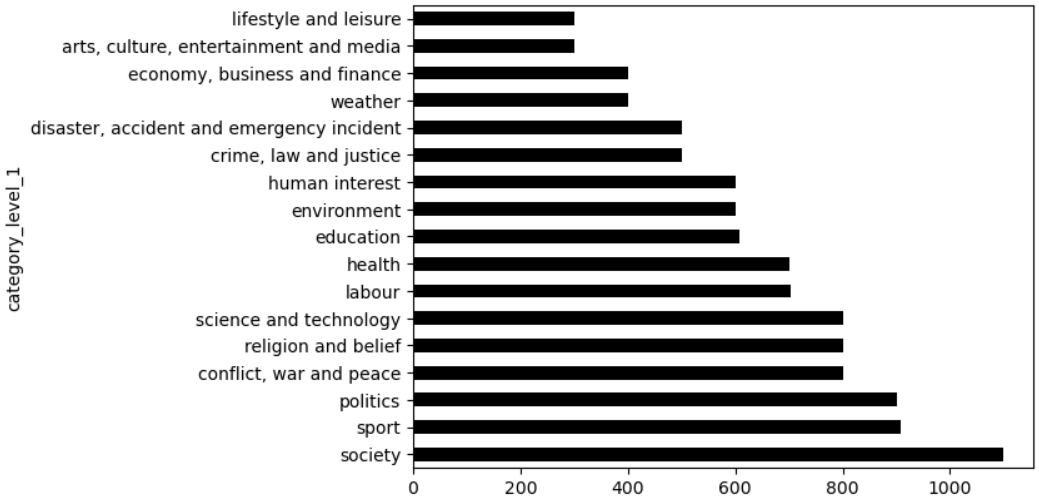
(A) *Τρομοκρατική ενέργεια ή Act of terror και*

(B) *Καιρικά φαινόμενα ή Weather Phenomena*

Τέλος , οφείλουμε να αποφασίσουμε ποια χαρακτηριστικά από το Σετ θα κρατήσουμε , διότι δεν είναι όλα απαραίτητα , για την τελική απόφαση (*παραδείγματος χάριν , δεν χρειάζεται να προβλέψουμε τον τίτλο κάθε άρθρου*) . Με τον τρόπο αυτό , τα χαρακτηριστικά – στήλες που χρειάζονται είναι τα :

(A) *Data ID* , ως δείκτης (*index*)
(B) *Content* , διότι περιέχει το κάθε άρθρο
(C) *Category Level 1* , ως οι κλάσεις πρώτου επιπέδου και
(D) *Category Level 2* , ως οι κλάσεις δεύτερου επιπέδου

Τώρα το *Dataset* μας , έχει την κατάλληλη μορφή για να συνεχίσουμε με το κομμάτι της Προ-επεξεργασίας (*Preprocessing*) .



Σχήμα 1.1

```
In [18]: df.content[371]
```
executed in 6ms, finished 16:24:21 2024-02-15

```
Out[18]: 'A fitness class has become the first in Scotland to practice pilates - with goats.\n\nThe bizarre exercise , which involves tw
o baby goats climbing on top of participants, has taken off at Bellcraig Farm, Glenrothes, Fife.\n\nOrganiser Jo Munro said one
-year-old kids Hazel and Mabel love to interact with humans at the fitness class.\n\nThe loveable animals help release feel-goo
d hormones such as serotonin and make pilate-goers relax more as they stretch on their mats in a barn.\n\nInstructor Jo, 39, wa
s inspired to launch the first classes in Scotland after taking part in goat yoga classes while on holiday in the US.\n\nThe cl
asses of up to six participants start by collecting the kids from the field before standing on mats in a stone built cottage.\n
\nThe group, which is the first in the UK to use African pygmy goats, warms up, while their four-legged friends inquisitively s
niff around.\n\nJo, from Edinburgh, said: "I went to California last year and it\'s massive there so I went and tried it and th
ought I have to bring this back to Scotland.\n\n"It\'s not sore but the goats are reasonably heavy so it does make it more chal
lenging.\n\n"The kids are quite agile and nimble so it can be like a massage with their hooves, but you wouldn\'t want it on yo
ur bare skin.\n\n"When people arrive we go and collect the goats from the field and take them into this little stone cottage wh
ich has a stove so it\'s nice and cosy.\n\n"There is a maximum of six people to two goats.\n\n"Goat pilates is something everyo
ne has seen on YouTube and I think you either love it or hate it.\n\n"It is complete hilarity and it makes everyone really happ
y and they leave on a high feeling really joyful.\n\n"There is an animal therapy side to it. You let out endorphins that make y
ou feel good, like when you pet an animal.\n\n"The goats add that extra element to the pilates. If you are holding a plank with
a 10kg goat on your back it is going to be much harder."\n\nOnce the classes are finished participants will be able to feed the
lambs and play with the farm\'s dogs.\n\nAnd it has proved a hit with Scots as classes book up.\n\nJo said: "We have been doing
it for a couple of months now and it has been really popular with fully booked classes.\n\n"It seems to have really captured pe
ople who were intrigued and finding it really cute."'
```

```
In [23]: df.content[374]
```
executed in 7ms, finished 16:25:11 2024-02-15

```
Out[23]: 'A fitness class has become the first in Scotland to practice pilates - with GOATS.\n\nThe bizarre exercise, which involves two
baby goats climbing on top of participants, has taken off at Bellcraig Farm, Glenrothes, Fife.\n\nOrganiser Jo Munro said one-y
ear-old kids Hazel and Mabel love to interact with humans at the fitness class.\n\nThe loveable animals help release "feelgood"
hormones such as serotonin and make pilate-goers relax more as they stretch on their mats in a barn.\n\nInstructor Jo, 39, was
inspired to launch the first classes in Scotland after taking part in goat yoga classes while on holiday in the US.\n\nThe clas
ses of up to six participants start by collecting the kids from the field before standing on mats in a stone built cottage.\n\n
The group, which is the first in the UK to use African pygmy goats, warms up, while their four-legged friends inquisitively sni
ff around.\n\nJo, from Edinburgh, said: "I went to California last year and it\'s massive there so I went and tried it and thou
ght I have to bring this back to Scotland.\n\n"It\'s not sore but the goats are reasonably heavy so it does make it more challe
nging.\n\n"The kids are quite agile and nimble so it can be like a massage with their hooves, but you wouldn\'t want it on your
bare skin.\n\n"When people arrive we go and collect the goats from the field and take them into this little stone cottage which
has a stove so it\'s nice and cosy.\n\n"There is a maximum of six people to two goats.\n\n"Goat pilates is something everyone h
as seen on YouTube and I think you either love it or hate it.\n\n"It is complete hilarity and it makes everyone really happy an
d they leave on a high feeling really joyful.\n\n"There is an animal therapy side to it. You let out endorphins that make you f
eel good, like when you pet an animal.\n\n"The goats add that extra element to the pilates. If you are holding a plank with a 1
0kg goat on your back it is going to be much harder."\n\nOnce the classes are finished participants will be able to feed the la
mbs and play with the farm\'s dogs.\n\nAnd it has proved a hit with Scots as classes book up.\n\nJo said: "We have been doing i
t for a couple of months now and it has been really popular with fully booked classes.\n\n"It seems to have really captured peo
ple who were intrigued and finding it really cute."'
```

Εικόνα 1.1

```
df.content[6528]
```
executed in 4ms, finished 16:25:29 2024-02-15

```
'\n\n'
```

Εικόνα 1.2

PREPROCESSING

TEXT CLEANING

Πρώτο μέρος της Προ-επεξεργασίας , που γίνεται στα κείμενα μας , αποτελεί ο "καθαρισμός" κειμένου . Δηλαδή , από το αρχικό μας κείμενο , πρέπει να βγάλουμε όσα κομμάτια δεν αφορούν το περιεχόμενο (*δηλαδή τις λέξεις κάθε άρθρου*) . Συνοπτικά η διαδικασία περιγράφεται από τις παρακάτω ενέργειες :

(Α) Αφαίρεση όσων χαρακτήρων δεν είναι ASCII , όπως μη λατινικών χαρακτήρων (*επειδή τα κείμενα είναι στα αγγλικά και οι άσχετοι χαρακτήρες θα " μπερδέψουν " τον αλγόριθμο Word2Vec , ως αυτός αναλυθεί στην συνέχεια*) ,

(Β) Αφαίρεση χαρακτήρων που δηλώνουν αλλαγή γραμμής ,

(Γ) Αφαίρεση αριθμών και ημερομηνίας και αντικατάσταση τους με κενό χαρακτήρα,

(Δ) Αφαίρεση σημείων που φέρουν HTML Ετικέτα και

(Ε) Αφαίρεση των σημείων στίξης από το κείμενο .

Το αποτέλεσμα των παραπάνω , εμφανίζεται στην *Εικόνα 2.1.1*

TEXT NORMALIZATION

Τώρα που το κείμενο μας έχει προ-επεξεργαστεί κατάλληλα και έχει " καθαρισθεί " , από τα " κακά " σημεία , είμαστε έτοιμοι να ξεκινήσουμε το κομμάτι της Κανονικοποίησης [1] κειμένου , όπως αυτή προβλέπεται σχεδόν σε όλα τα προβλήματα *Επεξεργασίας Φυσικής Γλώσσας* (*Natural Language Processing ή NLP*) . Αυτή αποτελείται από δύο κύρια μέρη : Την *Λημματοποίηση* (*Lemmatization*) των λέξεων και τον " *Διαχωρισμό* " (*Tokenization*) . Προτού όμως προβούμε σε αυτά , οφείλουμε να διαμορφώσουμε (*για άλλη μία φορά*) κατάλληλα το κείμενο μας , δηλαδή να εφαρμόσουμε άλλες δύο ακόμα τεχνικές καθαρισμού κειμένου . Η πρώτη από αυτές , είναι η απεικόνιση κάθε λέξης με μικρά (*lowercase*) γράμματα για να υπάρχει ομοιομορφία στο κείμενο και η δεύτερη είναι η αφαίρεση των λεγόμενων " *ενδιάμεσων λέξεων* " ή απλώς *stopwords* . Τα τελευταία , (*όπως περιγράφονται και από το πακέτο stopwords της βιβλιοθήκης nltk* [2]) , αποτελούνται κατά κόρων από αντωνυμίες και άρθρα άχρηστα ως προς την συνολική εικόνα του κειμένου και κυρίως με την **σημασιολογική έννοια** των λέξεων , αλλά και από ρήματα (*όπως για παράδειγμα το ρήμα είναι και δη στην αγγλική του μορφή και κλίση : am , are , is*) . Με άλλα λόγια (*και όπως θα*

[1] Dan Jurafsky : Speech and Language Processing , Third Edition , pg 24-29 .

[2] https://gist.github.com/sebleier/554280

φανεί στην συνέχεια) στην πρόταση : " *Ο Χ είναι δάσκαλος* " ή " *Αυτή η ομάδα έκανε ...* " , το κύριο περιεχόμενο της πρώτης πρότασης είναι η ιδιότητα του *Χ* και όχι το φύλο του , ενώ στην δεύτερη μας η αντωνυμία χρησιμοποιείται για να αποφευχθεί η τυχόν επανάληψη του ονόματος της ομάδας , ή για να δοθεί έμφαση και τα λοιπά . Το προκύπτον κείμενο , εμφανίζεται στην *Εικόνα 2.2.1* και δίνει μία πρώτη μορφή στην σύνδεση των μεταξύ τους λέξεων .

WORD LEMMATIZATION

Η *Λημματοποίηση* των λέξεων είναι μία από τις κυριότερες και πιο σημαντικές τεχνικές προ-επεξεργασίας κειμένου . Η ίδια περιγράφεται ως μία διαδικασία [3] , κατά την οποία προσδιορίζεται αν δύο λέξεις έχουν την ίδια ρίζα παρά τις " *επιφανειακές διαφορές* " (*surface differences*) τους . Για παράδειγμα οι λέξεις : **τραγούδησε** , **τραγουδούσε** και **τραγουδά** (***sang*** , ***sung*** , ***sings***), είναι διαφορετικές μορφές του ρήματος **τραγουδώ** (***sing***) , το οποίο αποτελεί και το κοινό λήμμα – ρίζα των λέξεων και ένας *λημματοποιητής* (*lemmatizer* , *δηλαδή η συνάρτηση που υλοποιεί την λημματοποίηση*) " *χαρτογραφεί* " (*maps*) τις παραπάνω λέξεις στο **τραγουδώ** . Ένα ακόμα (*λίγο πιο δύσκολο*) παράδειγμα είναι οι λέξεις : **είμαι** , **είσαι** και **είναι** (***am*** , ***are*** , ***is***) , οι οποίες έχουν κοινή ρίζα – λήμμα το ρήμα **είναι** (***be*** , *οι αρχαίοι Έλληνες όμως θα μας διόρθωναν και θα έλεγαν απαρέμφατο αντί για ρήμα*) . Η *λημματοποιημένη* μορφή μίας πρότασης ,όπως της :

Τα παιδιά παίζουν μπάλα
(The kids are playing football)

Είναι η :

Το παιδί παίζω μπάλα
(The child play football)

Μία πιο απλή εκδοχή της αποτελεί το *Stemming* , που απλώς αφαιρεί τα επιθέματα των λέξεων , χαρτογραφώντας τις **τραγούδησε** , **τραγουδούσε** και **τραγουδά** στο *stem* **τραγούδ** . Το κείμενο που προκύπτει μετά την διαδικασία της *λημματοποίησης* , εμφανίζεται στην *Εικόνα 2.2.2* .

[3] Dan Jurafsky : Speech and Language Processing , Third Edition , pg 5 .
Υποσημείωση : Στην παρούσα εργασία εφαρμόζουμε την τεχνική της λημματοποίησης . Αν θελήσετε να δείτε και την τεχνική του Stemming , όπως και τις υπόλοιπες παραλλαγές υλοποίησης της εργασίας , μπορείτε να μεταβείτε στο github της ομάδας , το οποίο παραθέτουμε στην τελευταία σελίδα του Project .

TOKENIZATION

Τέλος σε ό,τι αφορά το *tokenization* , είναι μία τεχνική , η οποία χωρίζει τις λέξεις μεταξύ τους στο κείμενο σε διαφορετικά *νήματα* (*strings*) , όπως φαίνεται και στην *Εικόνα 2.2.3* . Με την βοήθειά της , δημιουργούμε το *σώμα κειμένου* μας (*corpus*) , μέσα στο οποίο υπάρχουν όλες οι λέξεις από το Σετ δεδομένων μας (*από τις οποίες οι 21958 είναι ξεχωριστές*) , μετατρέποντας τες στην ιδανική μορφή για την δημιουργία ενός Word2Vec μοντέλου .

```
'Authorities are trying to determine if anyone helped two inmates who escaped from a California jail, traveled hundreds of mile
s and crossed into Mexico before being captured trying to walk back into the United States.\n\nJonathan Salazar, 20, and Santos
Fonseca, 21, were arrested by U.S. Customs and Border Protection officials at a port of entry in San Ysidro — the nation\'s lar
gest border crossing — early Wednesday, Monterey County Sheriff\'s Office Capt. John Thornburg said.\n\nThornburg said the two
are in the custody of Monterey County officials and have been returned to a jail in Salinas, a farming city of about 160,000 pe
ople roughly 100 miles (160 kilometers) south of San Francisco.\n\nSalazar and Fonseca escaped Sunday from the lockup in Salina
s after climbing through a hole they made in the ceiling of a bathroom, squeezing through a hollow wall and kicking open a hatc
h. Officials found their jail-issued jumpsuits outside the building. Inmates often wear their own clothes under the jumpsuits,
```

```
'Authorities are trying to determine if anyone helped two inmates who escaped from a California jail traveled hundreds of miles
and crossed into Mexico before being captured trying to walk back into the United States Jonathan Salazar  and Santos Fonseca
were arrested by U S Customs and Border Protection officials at a port of entry in San Ysidro the nation s largest border cross
ing early Wednesday Monterey County Sheriff s Office Capt John Thornburg said Thornburg said the two are in the custody of Mont
erey County officials and have been returned to a jail in Salinas a farming city of about  people roughly  miles  kilometers s
outh of San Francisco Salazar and Fonseca escaped Sunday from the lockup in Salinas after climbing through a hole they made in
the ceiling of a bathroom squeezing through a hollow wall and kicking open a hatch Officials found their jail issued jumpsuits
```

Εικόνα 2.1.1

```
'authorities trying determine anyone helped two inmates escaped california jail traveled hundreds miles crossed mexico captured
trying walk back united states jonathan salazar santos fonseca arrested u customs border protection officials port entry san ys
idro nation largest border crossing early wednesday monterey county sheriff office capt john thornburg said thornburg said two
custody monterey county officials returned jail salinas farming city people roughly miles kilometers south san francisco salaza
r fonseca escaped sunday lockup salinas climbing hole made ceiling bathroom squeezing hollow wall kicking open hatch officials
found jail issued jumpsuits outside building inmates often wear clothes jumpsuits thornburg said investigators yet determined l
```

Εικόνα 2.2.1

```
'authority trying determine anyone helped two inmate escaped california jail traveled hundred mile crossed mexico captured tryi
ng walk back united state jonathan salazar santos fonseca arrested u custom border protection official port entry san ysidro na
tion largest border crossing early wednesday monterey county sheriff office capt john thornburg said thornburg said two custody
monterey county official returned jail salina farming city people roughly mile kilometer south san francisco salazar fonseca es
caped sunday lockup salina climbing hole made ceiling bathroom squeezing hollow wall kicking open hatch official found jail iss
ued jumpsuit outside building inmate often wear clothes jumpsuit thornburg said investigator yet determined long worked making
```

Εικόνα 2.2.2

```
['authority',
 'trying',
 'determine',
 'anyone',
 'helped',
 'two',
 'inmate',
 'escaped',
 'california',
 'jail',
 'traveled',
 'hundred',
 'mile',
 'crossed',
 'mexico',
 'captured',
 'trying',
 'walk',
 'back',
 'united',
 'state',
 'jonathan',
 'salazar',
```

Εικόνα 2.2.3

WORD2VEC

Το *Word2Vec* , ανήκει στην ευρύτερη " *οικογένεια* " των μεθόδων *ενσωματώσεων* ή *ενσωματωμένων λέξεων* ή αλλιώς *word embeddings* . Οι ενσωματώσεις που προκύπτουν (*embeddings*) είναι στατικές ενσωματώσεις , δηλαδή η μέθοδος μαθαίνει μία σταθερή ενσωμάτωση για κάθε λέξη στο λεξιλόγιο . Η κύρια ιδέα [4] , πίσω από το *Word2Vec* , είναι ότι αντί να μετράμε πόσο συχνά κάθε λέξη εμφανίζεται κοντά σε κάποια άλλη , έστω $\boldsymbol{X}$, θα εκπαιδεύσουμε έναν ταξινομητή στο πρόβλημα δυαδικής πρόβλεψης : " *Είναι η λέξη* $\boldsymbol{Y}$ *δυνατόν να εμφανιστεί κοντά στην λέξη* $\boldsymbol{X}$ *;* " · στην πραγματικότητα όμως , δεν μας ενδιαφέρει η πρόβλεψη ως διαδικασία , αλλά η τελική μορφή των βαρών του ταξινομητή που θα αποτελέσουν και τα *embeddings* μας . Η " επαναστατική " αυτή τεχνική [5] στην οποία βασίζεται το *Word2Vec* ονομάζεται *εαυτό-επιβλεπόμενη* (*self-supervised*) .

Στο παρόν κομμάτι της εργασίας μας , η μέθοδος *Word2Vec* που χρησιμοποιήσαμε για τις ενσωματώσεις λέξεων , είναι αυτή του *skip-gram* (*η άλλη μέθοδος , είναι αυτή του Bag of Words*) που ο αλγόριθμος της συχνά και με " χαλαρά " όρια , σχετίζεται με το ίδιο το *Word2Vec* . Συνοπτικά , αυτή η μέθοδος (*που κυρίως έχει αναλυθεί πιο πάνω*) περιγράφεται ως εξής :

Το μοντέλο *Skip-gram* μαθαίνει 2 δισδιάστατα (*διαστάσεων : [d X 1]*) διανύσματα ενσωματώσεων λέξεων για κάθε λέξη *w* που εμπεριέχεται σε ένα λεξικό - λεξιλόγιο με $|V|$ λέξεις . Τα διανύσματα αυτά είναι τα :

(A) *v* , ως το διάνυσμα κεντρικής λέξης και

(B) *u* , ως το διάνυσμα συμφραζόμενων

Με τον τρόπο αυτό , τελικά μαθαίνει δύο πίνακες: *V* και *U* διαστάσεων [*d X* $|V|$] , με τον πρώτο πίνακα να αποτελεί και τα τελικά *word embeddings* .Ένα παράδειγμα τέτοιου μοντέλου , εμφανίζεται στην *Εικόνα 3.1* .

Τελευταίο κομμάτι του *skip-gram* , είναι η συνάρτηση εξόδου του , εφόσον δεν χρησιμοποιούμε την κλασική μέθοδο του *Negative Sampling* ή *Αρνητικής Δειγματοληψίας* , αλλά αυτής του *softmax* αλγόριθμου . Γενικά , η συνάρτηση *softmax* , είναι μία γενικοποίηση της σιγμοειδούς για τον υπολογισμό της πιθανότητας $p(\boldsymbol{y_k} = 1 \,|\boldsymbol{x})$ και δέχεται ως είσοδο ένα διάνυσμα $\boldsymbol{z} = [\, z_0, z_1, \dots, z_k \,]$, Κ αυθαίρετων τιμών και τα χαρτογραφεί σε μία πιθανοτική

[4] Dan Jurafsky : Speech and Language Processing , Third Edition , pg 120 .

[5] Benzio et al. (2003) & Collobert et al.(2011) .
Υποσημείωση : Περισσότερες πληροφορίες για την σημασιολογικές (*semantic*) ιδιότητες των *embeddings* , μπορείτε να βρείτε επίσης στο github , στο οποίο εμπεριέχεται και ο κώδικας .

κατανομή μεταξύ των τιμών [0,1] έτσι ώστε το συνολικό άθροισμα όλων των τιμών να είναι ίσο με το 0 .

Για ένα διάνυσμα **z** διάστασης K , ο *softmax* :

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_i)} \, , 1 \leq i \leq K$$

Έτσι ,το *softmax* ένός διάνυσμα $\boldsymbol{z} = [\, z_0, z_1, \ldots, z_k \,]$, είναι ένα διάνυσμα και το ίδιο :

$$softmax(z) = [\, \frac{\exp(z_0)}{\sum_{j=0}^{K} \exp(z_i)} \, , \frac{\exp(z_1)}{\sum_{j=0}^{K} \exp(z_i)} \, , \ldots , \frac{\exp(z_K)}{\sum_{j=0}^{K} \exp(z_i)} \,]$$

Ο παρονομαστής , χρησιμοποιείται για να " κανονικοποιήσει " τις τιμές σε πιθανότητες !

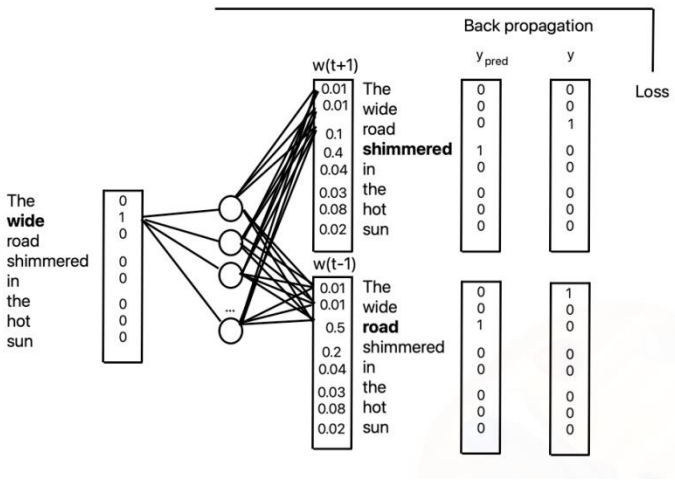
Στο μοντέλο μας , ο αλγόριθμος χρησιμοποιείται στο τελευταίο στάδιο ως εξής :

$$\underline{y_{out}} = softmax(W\underline{x} + b)$$

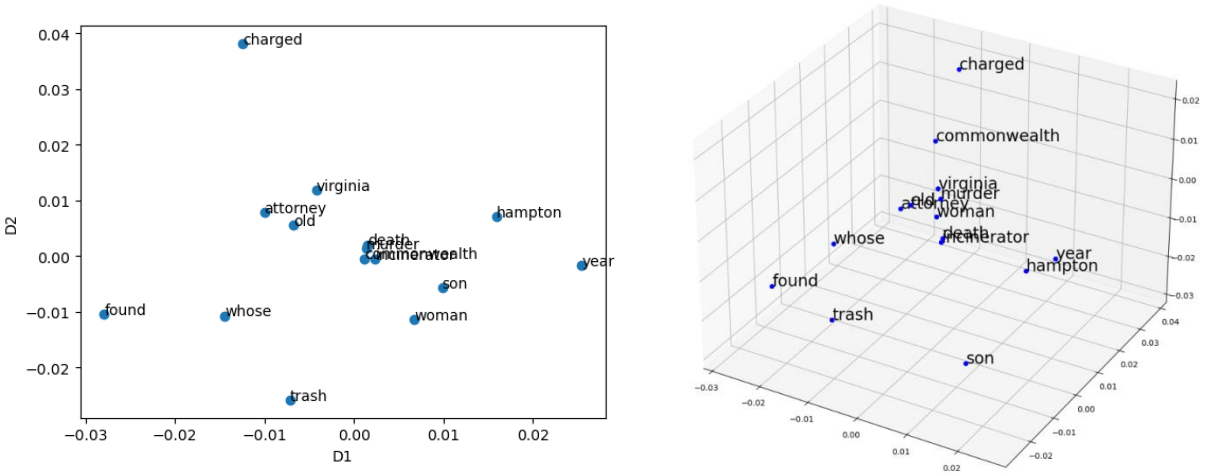
Με τα παραπάνω , έχουμε καταφέρει να χαρτογραφήσουμε - μεταφράσουμε τις λέξεις σε αριθμούς (*όπως φαίνεται και από τις Εικόνες 3.2 και 3.3 , που καταγράφουν την σημασιολογική σύνδεση των λέξεων στο δισδιάστατο και τρισδιάστατο επίπεδο*) και είμαστε έτοιμοι για την ταξινόμηση των κειμένων μας . Προσοχή όμως , διότι χρησιμοποιούμε μία παραλλαγή [6] για την ταξινόμηση των κειμένων μας (με βάση το Νευρωνικό δίκτυο)· αντί να δημιουργήσουμε στο δίκτυο μας ένα *στρώμα ενσωμάτωσης* (*embedding layer*) , κρατάμε τον μέσο όρο των *embeddings* , ο οποίος αποτελεί και την είσοδο του δικτύου μας ! Τα κρυφά επίπεδα από τα οποία αποτελείται το μοντέλο μας και τα χρησιμοποιούμε ως χαρακτηριστικά είναι ίσα με 600 , άρα και οι ενσωματώσεις μας , διαθέτουν 600 χαρακτηριστικά .

Τέλος και για την ταξινόμηση ενός αγνώστου προς το σετ δεδομένων μας κείμενο με βάση το Νευρωνικό μας δίκτυο , οφείλουμε να εκπαιδεύσουμε το *Word2Vec* μοντέλο μας στο σύνολο των κειμένων εξ αρχής . Αυτό συμβαίνει , λόγω της στατικής φύσης των *embeddings* , διότι αν επιλέγαμε παραδείγματος χάριν να δημιουργήσουμε δύο μοντέλα *Word2Vec* , ένα για το σετ δεδομένων και ένα για το άγνωστο , τότε τα μεταξύ τους *embeddings* , θα ήταν άσχετα ! .

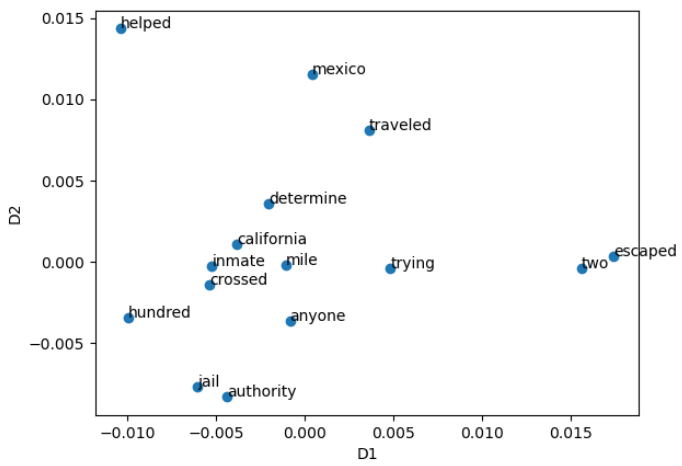
[6] Dan Jurafsky : Speech and Language Processing , Third Edition , pg 148-152 .

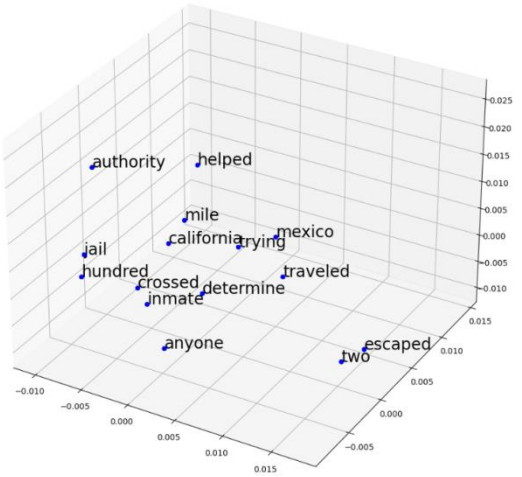


Εικόνα 3.1 ,
το μοντέλο Skip-gram



Εικόνα 3.2 ,
τα embeddings του
1ου κειμένου και η
αναπαράστασή τους
(μετά από PCA) .





Εικόνα 3.3 , τα embeddings του 2ου κειμένου και η αναπαράστασή τους (μετά από PCA) .

NEURAL NETWORK

ARCHITECTURE

Το Νευρωνικό δίκτυο το οποίο δημιουργήσαμε αποτελεί ένα είδος ***Fully-Connected*** νευρωνικού δικτύου. Αποτελείται από ένα στρώμα (*layer*) εισόδου , δύο κρυφά στρώματα – επίπεδα (*hidden layers , ίσως και 3 αν θεωρήσουμε το Dropout Layer ως ένα εξτρά κρυφό στρώμα*) και ένα επίπεδο (*layer*) εξόδου όπως φαίνεται και από την *Εικόνα 4.1.1* [7] . Ο κάθε νευρώνας εφαρμόζει έναν γραμμικό μετασχηματισμό στο εκάστοτε διάνυσμα εισόδου μέσω ενός πίνακα βαρών . Ακόμα , ο κάθε νευρώνας συνδέεται με όλους τους υπόλοιπους νευρώνες του επόμενου στρώματος (*προφανώς εξαιρουμένων αυτών του επιπέδου εξόδου*) , πράγμα το οποίο σημαίνει ότι η παραμικρή αλλαγή σε έναν νευρώνα επηρεάζει όλο το δίκτυο μας . Προσοχή , διότι τα νευρωνικά δίκτυα που χρησιμοποιούμε για την ταξινόμηση των κατηγοριών 1^{ου} και 2^{ου} επιπέδου , ακολουθούν την ίδια δομή (*η οποία θα περιγραφεί παρακάτω*) , αλλάζοντας μόνο το πλήθος των νευρώνων στο τελευταίο επίπεδο .

Το στρώμα εισόδου (*input layer*) του δικτύου μας , δηλώνει τη διάσταση των διανυσμάτων – χαρακτηριστικών (*όπως αυτά προκύπτουν από το Word2Vec μοντέλο*), που είναι 600 . Το πρώτο *Hidden Layer* είναι επί της ουσίας ένα *Dense Layer* , με 200 νευρώνες και ως *activation function* ή αλλιώς *συνάρτηση ενεργοποίησης* , την *Leaky ReLU* . Αυτό συνδέεται με ένα δεύτερο *Dense Layer* , που αποτελείται από 360 νευρώνες, με συνάρτηση ενεργοποίησης επίσης την *leaky ReLU* . Οι νευρώνες του δικτύου μας , ακολουθούν την αναλογία-λόγο ίση με 1.8 (*νευρώνες δευτέρου / νευρώνες πρώτου*) , ακολουθώντας μία τεχνική παρόμοια με αυτή των ασύγχρονων νευρωνικών δικτύων που χρησιμοποιούνται στην αναγνώριση συναισθήματος μέσω φωνής [8] . Στην συνέχεια , προσθέτουμε ένα *Dropout Layer* με πιθανότητα " αφαίρεσης " νευρώνων ίση με 20%, για να βελτιώσουμε την ακρίβεια του νευρωνικού μας , λόγω του μεγάλου αριθμού των μονάδων-νευρώνων που χρησιμοποιούμε . Τέλος, το *output layer* (*Dense layer*) , για τις κατηγορίες 1^{ου} επιπέδου αποτελείται από 17 νευρώνες εξόδου , εφόσον έχουμε 17 κλάσεις-κατηγορίες άρθρων και για του 2^{ου} από 109 νευρώνες εξόδου , αφού έχουμε 109 κλάσεις-υποκατηγορίες άρθρων .

Στο επίπεδο εξόδου χρησιμοποιούμε ως συνάρτηση ενεργοποίησης , την συνάρτηση *softmax* , επειδή θέλουμε να κανονικοποιήσουμε την έξοδο του νευρωνικού σε μία κατανομή πιθανοτήτων με βάση τις προβλεπόμενες κλάσεις εξόδου . Μετά την χρήση του *softmax* ο κάθε νευρώνας που αντιστοιχεί σε κάθε μία κλάση , ανατίθεται μία πιθανότητα και χρησιμοποιώντας ένα σχήμα : " *ο νικητής – μεγαλύτερη τιμή τα παίρνει όλα* " παίρνουμε και το αποτέλεσμα της κατηγοριοποίησης . Δηλαδή , αν ο όγδοος νευρώνας έχει την μεγαλύτερη πιθανότητα , τότε το δείγμα ταξινομείται στην όγδοη κλάση – ομάδα .

Η συνάρτηση ενεργοποίησης ***Leaky ReLU*** χρησιμοποιείται , επειδή είναι πιο αποτελεσματική από την *ReLU* , καθώς λύνει το *dying ReLU* πρόβλημα και βελτιώνει την

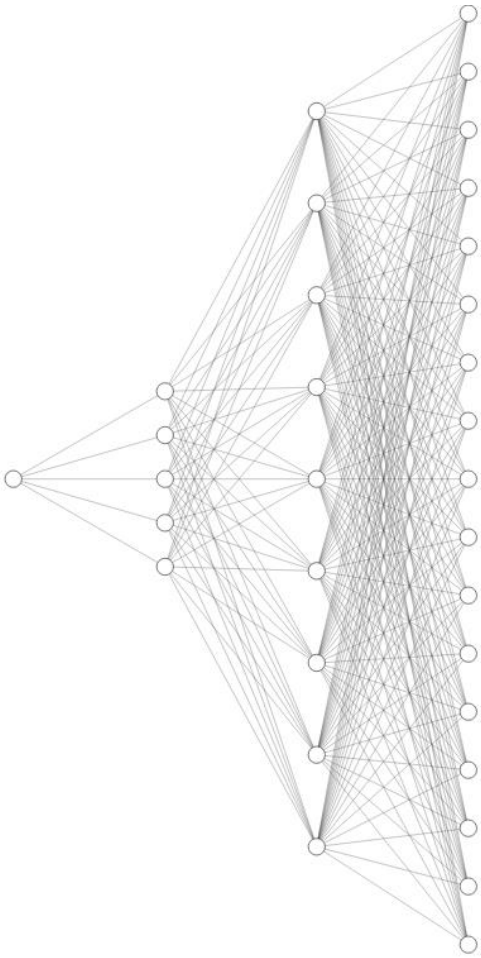
[7] Η εικόνα δημιουργήθηκε με την βοήθεια του εργαλείου *NN SVG* .

[8] Για τον αριθμό των νευρώνων χρησιμοποιήθηκαν αρκετές διαφορετικές τιμές , όμως τα καλύτερα αποτελέματα ήρθαν όταν ο αριθμός των νευρώνων του 1^{ου} κρυφού επιπέδου ήταν μεταξύ του 180 και 220 , ενώ του 2^{ου} μεταξύ του 340 και του 400 , κρατώντας πάντα τον μεταξύ τους λόγο – αναλογία ίση με 1.8

απόδοση του νευρωνικού μας δικτύου (*η επιλογή της , όπως και των περισσότερων παραμέτρων έγινε μετά από αρκετή δοκιμή και αποτυχία , αναζητώντας τις καλύτερες παραμέτρους , όπως φαίνεται και από το github repository της ομάδας*) .

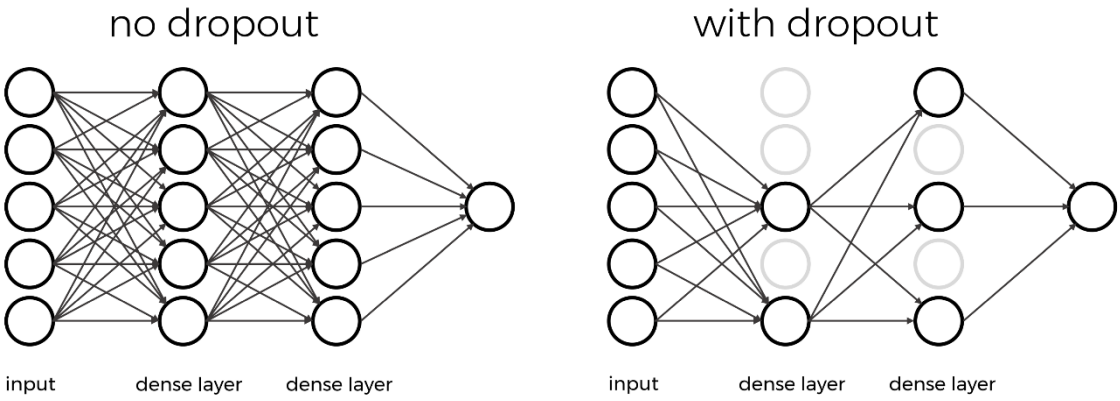
Το ***Dropout Layer*** , χρησιμοποιείται, για την αποφυγή του *overfitting* . Με τη διαδικασία αυτή, ανεξάρτητοι κόμβοι αποκλείονται σε πολλαπλά *training runs* , με την πιθανότητα 20%, σαν να μην ήταν μέρος της αρχιτεκτονικής του δικτύου . Ένα παράδειγμα νευρωνικού δικτύου με την χρήση αυτού του *layer* , εμφανίζεται στην *Εικόνα 4.1.2* [9] .

Τέλος , ο διαμοιρασμός των δεδομένων του σετ Δεδομένων για την εκπαίδευση και αποτίμηση του νευρωνικού μας δικτύου είναι ο εξής : 60% των δεδομένων για εκπαίδευση , 20 % για *επαλήθευση* (*validation*) και 20% για *δοκιμή* (*testing*)



Εικόνα 4.1.1
, μία προσεγγιστική
απεικόνιση του νευρωνικού μας δικτύου
με βάση το πρόβλημα των κατηγοριών
1ου επιπέδου

[9] Πηγή : https://carpentries-incubator.github.io/deep-learning-intro/instructor/bonus-material.html


no dropout
input
dense layer
dense layer
with dropout
input
dense layer
dense layer

Εικόνα 4.1.2

TRAINING ALGORITHM - OPTIMIZER

Ο αλγόριθμος εκπαίδευσης , που επιλέξαμε είναι ο *AdamW* . Είναι μία *stochastic gradient* μέθοδος , που βασίζεται στην προσαρμοστική εκτίμηση ορμών (*momentum*) $1^{ης}$ και $2^{ης}$ τάξης με μια πρόσθετη μέθοδο για την *αποσύνθεση των βαρών* [10](*weight decay*) . Η μέθοδος *AdamW , είναι υπολογιστικά αποτελεσματική , απαιτεί ελάχιστη μνήμη ,είναι αμετάβλητη στην διαγώνια αύξηση ή μείωση υπό κλίμακα (rescaling) των κλίσεων και ταιριάζει καλά για προβλήματα , τα οποία έχουν αρκετές παράμετρούς και δεδομένα* [11] , όπως και στα περισσότερα *NLP* προβλήματα . Ο *AdamW* , είναι μία βελτιωμένη εκδοχή του *Adam* , διότι η μείωση βάρους εκτελείται μόνο μετά τον έλεγχο του μεγέθους του βήματος ως προς τις παραμέτρους .

Η κανονικοποίηση στον *Adam* συνήθως υλοποιείται με την παρακάτω μετατροπή , όπου w_t ο ρυθμός μείωσης των βαρών σε χρόνο t :

$$g_t = \nabla f(\theta_t) + w_t \theta_t$$

Ενώ ο *AdamW* , προσαρμόζει τον όρο της μείωσης των βαρών , να εμφανίζεται στην ενημέρωση της κλίσης :

$$\theta_{t+1,i} = \theta_{t,i} - \eta \left(\frac{1}{\sqrt{\widehat{u_t}} + \varepsilon} \widehat{m_t} + w_{t,i} \theta_{t,i} \right) \forall t$$

, όπου :

θ , *είναι η παράμετρος προς αλλαγή*
η , *είναι το αρχικό LR*
ε , *είναι ένας μικρός αριθμός για την αποφυγή της διαίρεσης με το* 0
$\nabla f(\theta)$, *είναι η παράγωγος της συνάρτησης κόστους* (*loss function*) ,

$\widehat{m_t} = \frac{m_t}{1 - \beta_{1,t}}$, *όπου :*
m_t , *το μέσο ή η πρώτη ορμή*
β_1 , *ο αρχικός ρυθμός αποσύνθεσης* (*decay*) *για την πρώτη ορμή και*
$\widehat{u_t} = \frac{u_t}{1 - \beta_{2,t}}$, *όπου :*
u_t , *είναι μη κεντρισμένη διασπορά ή δεύτερη ορμή και*
β_2, , *ο αρχικός ρυθμός αποσύνθεσης* (*decay*) *για την δεύτερη ορμή*

[10] Decoupled Way Decay Regularization , Loshchilov , Hutter et al. 2019 .
[11] Kingma et al. 2014 .

Τέλος , στον *optimizer* μας , θα χρησιμοποιήσουμε την *EMA ορμή* (*exponential moving average*) , η οποία αποτελείται από τον υπολογισμό ενός εκθετικού κινητού μέσου όρου των βαρών του μοντέλου και την περιοδική αντικατάστασή τους με αυτόν τον κινητό μέσο όρο , με βάση τον τύπο :

*new_average = ema_momentum * old_average + (1 – ema_momentum) *current+variable_value*

Οι τιμές των παραμέτρων που επιλέγουμε για τον *AdamW* , είναι οι εξής :

(I) η = 0.007.

(II) β_1 = 0.9

(III) β_2 = 0.999

(IV) *ema_momentum* = 0.9999

Neural Network Evaluation

Αφού εκπαιδεύσουμε το νευρωνικό μας δίκτυο , για δέκα επαναλήψεις , με το *news-classification dataset* , το νευρωνικό αποφέρει με τα μετρικά της βιβλιοθήκης *sklearn* ακρίβεια από **76%** έως **81%**, όσον αφορά την ταξινόμηση, για τα δεδομένα των κατηγοριών $1^{ου}$ επιπέδου και για $2^{ου}$ επιπέδου αποφέρει ακρίβεια από **57%** έως **60%** . Για την αξιολόγησή τους μέσω των μετρικών του *keras* , αποφέρει ακρίβεια γύρω στο 94%, για δεδομένα του $1^{ου}$ επιπέδου , ενώ για τα $2^{ου}$ ακρίβεια κοντά στο 99% . Προσοχή , όμως , διότι η ακρίβεια που υπολογίζεται μέσω της *sklearn* , είναι η πραγματική ακρίβεια των δεδομένων καθώς το keras (, δηλαδή το νευρωνικό μας δίκτυο) , αποτιμάει τις *ετικέτες* (*labels*) των κλάσεων – κατηγοριών σε *One-Hot* κωδικοποίηση , η οποία εμφωλεύει αρκετούς κινδύνους .

Παραδείγματος χάριν , έστω ότι έχουμε τρεις κλάσεις με *labels* τα 1 , 2 , 3 , με τις *One-Hot* τιμές τους να είναι 001 , 010 , 100 και τρία σημεία – πρότυπα , με labels 2 , 1 ,3 ή αντίστοιχα 010 , 001 , 100 . Επίσης έστω ότι το νευρωνικό μας δίκτυο αποτιμά τα σημεία ως εξής : 001 , 001 , 101 , λέγοντας ότι για το πρώτο σημείο έχει επιτύχει ακρίβεια ίση με 33% , διότι έχει πετύχει σωστά την θέση του ενός μηδενικού , για το δεύτερο 100 % και για το τρίτο 67% , διότι μόνο το τελευταίο ψηφίο είναι διαφορετικό σε σχέση με όλα τα υπόλοιπα . Με τον τρόπο αυτό η συνολική ακρίβεια κατά τα μετρικά *keras* είναι ίση με 66 % . Στην πραγματικότητα όμως και λόγω του *One-Hot Encoding* τα labels μετά την *αποκωδικοποίηση* (*decoding*) είναι 1 , 3 και NaN , καθώς η τιμή 101 δεν υπάρχει στα αρχικά *labels* . Έτσι η πραγματική ακρίβεια είναι ίση με 33 % (αφού έχει προβλέψει μόνο ένα στα 3 σημεία σωστά).

Αναλυτικά και για τις κατηγορίες $1^{ου}$ επιπέδου , (*μία*) η εκπαίδευση του νευρωνικού μας δικτύου , εμφανίζεται στην *Εικόνα 4.2.1* , στην οποία , η ακρίβεια του νευρωνικού μας δικτύου είναι ίση με 80.55 % , ενώ στο *Σχήμα 4.2.1* , παρατηρούμε τις αυξομειώσεις τις

ακρίβειας και της συνάρτησης κόστους (*cost function*) . Για ότι αφορά τις κατηγορίες $2^{ου}$ επιπέδου , μία εκπαίδευση φαίνεται στην *Εικόνα 4.2.2* , στην οποία η ακρίβεια είναι ίση 59.79 % και στο *Σχήμα 4.2.2* , παρατηρούμε τις αυξομειώσεις τις ακρίβειας και της συνάρτησης κόστους . Τέλος , η πραγματική δομή του δικτύου μας , όπως και το πλήθος των παραμέτρων του , εμφανίζεται στην *Εικόνα 4.2.3* και τα βάρη και τα *bias* του νευρωνικού μας δικτύου , για κάθε επίπεδο (*ειδικά για τις κατηγορίες* $1^{ου}$ *επιπέδου*) , εμφανίζονται στην *Εικόνα 4.2.4* · οι αρχικές τιμές τους , είναι 0 για τα *bias* και ένας μικρός αριθμός κοντά στο 0 για τα βάρη , ενώ οι τελικές τους τιμές είναι αριθμοί , πολύ κοντά στο 0 !

```
loss: 2.414

accuracy: 0.941
```

	precision	recall	f1-score	support
0	0.92	0.16	0.28	68
1	0.94	0.35	0.51	176
2	0.80	0.41	0.54	97
3	0.90	0.50	0.64	90
4	0.95	0.38	0.54	93
5	0.87	0.43	0.57	108
6	0.86	0.71	0.78	126
7	0.81	0.54	0.65	136
8	0.81	0.55	0.65	122
9	0.84	0.49	0.62	155
10	0.96	0.39	0.56	61
11	0.50	0.50	0.50	172
12	0.85	0.54	0.66	182
13	0.78	0.50	0.61	151
14	0.21	0.93	0.35	200
15	0.81	0.93	0.87	169
16	0.89	0.94	0.91	77
accuracy			0.57	2183
macro avg	0.81	0.54	0.60	2183
weighted avg	0.77	0.57	0.60	2183

```
Precision is 0.8055214855305051
```

Εικόνα 4.2.1

Σχήμα 4.2.1

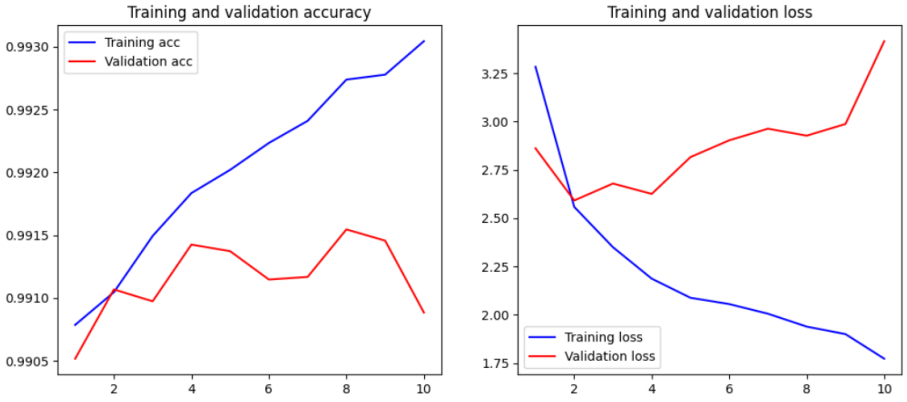
```
loss: 4.329

accuracy: 0.991
```

82	0.83	0.34	0.49	29
83	0.28	0.21	0.24	24
84	1.00	0.05	0.09	22
85	0.36	0.25	0.30	16
86	0.14	0.37	0.20	19
87	0.00	0.00	0.00	16
88	0.15	0.27	0.19	11
89	0.02	0.92	0.04	12
90	0.07	0.62	0.13	13
91	0.67	0.32	0.43	19
92	0.00	0.00	0.00	20
93	0.50	0.06	0.11	16
94	0.60	0.17	0.26	18
95	0.71	0.75	0.73	16
96	0.83	0.42	0.56	12
97	0.25	0.06	0.10	17
98	0.17	0.12	0.14	25
99	1.00	0.76	0.86	25
100	0.88	0.54	0.67	26
101	0.90	0.41	0.56	22
102	1.00	0.05	0.09	21
103	0.20	0.12	0.15	17
104	0.37	0.67	0.48	15
105	0.56	0.94	0.70	16
106	1.00	0.20	0.33	25
107	0.63	0.81	0.71	21
108	1.00	0.44	0.62	18
accuracy			0.31	2183
macro avg	0.60	0.32	0.34	2183
weighted avg	0.62	0.31	0.35	2183

```
Precision is 0.5979821876300838
```

Εικόνα 4.2.2 ,
εμφανίζεται ένα μέρος
των κατηγοριών 2ου επιπέδου



Σχήμα 4.2.2

```
Model: "News_Classifier"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embs (InputLayer)           [(None, 600)]             0

 dense (Dense)               (None, 200)               120200

 dense_1 (Dense)             (None, 360)               72360

 dropout (Dropout)           (None, 360)               0

 dense_2 (Dense)             (None, 17)                6137

=================================================================
Total params: 198697 (776.16 KB)
Trainable params: 198697 (776.16 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Εικόνα 4.2.3 ,
το δίκτυο μας , έχει συνολικά
198697 παραμέτρους
(κατηγορίες 1^{ου} επιπέδου)

```
For Our Dense Layer:
Weights
[<tf.Variable 'dense_18/kernel:0' shape=(600, 200) dtype=float32, numpy=
array([[ 0.02095382,  0.06529135, -0.06944855, ..., -0.11531513,
        -0.0432654 ,  0.04585142],
       [ 0.00288117,  0.11004753, -0.02837353, ...,  0.08460536,
         0.03656631,  0.08883729],
       [-0.1296058 , -0.13677879, -0.06991735, ..., -0.06799284,
        -0.08383609, -0.06283099],
       ...,
       [ 0.07793278,  0.08517528, -0.03700601, ..., -0.04134277,
        -0.04423823, -0.01249834],
       [ 0.02619734, -0.00548243, -0.01220047, ..., -0.02785957,
        -0.02657631,  0.04709461],
       [ 0.0641965 , -0.05564709,  0.06540658, ...,  0.06400902,
         0.06183492,  0.10047821]], dtype=float32)>, <tf.Variable 'dense_18/bias:0' shape=(200,) dtype=float32, numpy=
array([-1.41873565e-02, -6.82184193e-03, -1.04635060e-02, -1.12770544e-02,
       -1.11765210e-02, -1.19109787e-02, -1.10705020e-02, -7.49255065e-03,
       -1.29594747e-02,  5.79170231e-03, -1.18641434e-02, -9.31074563e-03,
       -9.86391678e-03, -7.29316100e-03, -7.91338086e-03, -6.90116687e-03,
       -1.23797879e-02, -1.03994245e-02, -6.21181773e-03, -1.34799797e-02,
       -1.09594017e-02, -1.39021715e-02, -5.66838915e-03, -6.82238070e-03,
       -9.61646624e-03, -1.09056532e-02, -1.00562861e-02, -1.31774526e-02,
       -7.35802343e-03, -1.17802098e-02, -9.86909308e-03, -1.39591061e-02,
       -9.25209001e-03,  1.44033809e-03, -1.13505200e-02, -7.70265656e-03,
       -1.27527621e-02, -9.78736207e-03, -5.60018187e-03, -9.51260980e-03,
       -4.64360556e-03, -9.53522511e-03, -1.64742935e-02, -1.32946037e-02,
       -6.73112366e-03, -1.14291301e-02, -9.14378185e-03, -9.01647564e-03,
       -1.34914173e-02, -1.14788078e-02, -1.16888871e-02, -1.07051786e-02,
       -8.08060076e-03, -1.01749972e-02,  2.25843629e-03, -7.03014480e-03,
       -5.41269407e-03, -9.36887693e-03,  1.41989568e-03, -1.06544243e-02,
       -7.70467008e-03, -3.53418523e-03, -5.78022376e-03, -9.60935839e-03,
       -6.82039186e-03, -1.40709979e-02, -5.59200719e-03, -1.40614565e-02,
       -6.76652696e-03, -1.40940640e-02,  6.29998802e-04, -6.11353852e-03,
       -7.21222954e-03, -1.06841000e-02, -1.25552639e-02, -1.10019930e-02,
       -8.31936672e-03, -1.37927039e-02, -4.84592048e-03, -1.19306156e-02,
       -3.74046038e-03, -7.80789042e-03, -1.16452463e-02, -1.09873014e-02,
       -8.63986742e-03, -9.26875696e-03, -7.81876687e-03, -1.04202833e-02,
       -8.71593785e-03, -1.36002228e-02, -4.92196158e-03, -1.35596683e-02,
       -1.38188675e-02, -1.26347095e-02, -1.49594489e-02, -1.30612766e-02,
       -9.45592579e-03, -1.15871895e-02, -1.45756267e-02, -1.08847613e-04,
       -6.69087330e-03, -8.78108013e-03, -9.56179015e-03, -1.11385575e-02,
       -8.12615547e-03, -9.94833838e-03, -1.53043605e-02, -1.11391619e-02,
       -9.18575190e-03, -1.37364790e-02, -1.15886042e-02, -9.88688879e-03,
       -8.44804291e-03, -8.98966007e-03,  2.03152624e-04, -1.85635919e-03,
       -1.14755975e-02, -6.76113274e-03, -1.06798997e-02, -7.16911303e-03,
       -9.70942993e-03, -1.03948452e-02, -1.38740474e-02, -7.98425078e-03,
       -4.13685944e-03, -1.13450950e-02, -9.02618002e-03, -9.17008892e-03,
       -9.65414383e-03, -1.15652764e-02, -9.09916766e-04, -6.37305155e-03,
       -1.12285167e-02, -1.31830424e-02, -1.40101882e-02, -9.29026958e-03,
       -1.05362982e-02, -1.06790168e-02, -7.49358209e-03, -1.17789125e-02,
       -1.00169405e-02, -6.10035378e-03, -8.69716797e-03, -1.10540604e-02,
       -1.03141107e-02, -1.11001488e-02, -1.12476256e-02, -1.20587144e-02,
       -3.87693360e-03, -8.33660644e-03, -9.14004724e-03, -9.82643943e-03,
       -6.72628265e-03, -9.42522846e-03, -5.14224498e-03, -7.01006595e-03,
       -1.03050740e-02, -1.25302542e-02, -9.81941074e-03, -7.57029140e-03,
       -1.06841978e-02, -9.11874138e-03,  4.01998928e-04, -1.92758464e-03,
       -6.10477570e-03, -1.07379956e-02, -7.43104098e-03, -1.16078984e-02,
       -7.30025908e-03, -8.27895384e-03, -7.56528601e-03, -7.99034629e-03,
       -1.19347293e-02, -1.08713079e-02, -3.39886406e-03, -9.90398414e-03,
       -1.32006723e-02, -1.19574415e-02, -1.09841686e-03, -1.53587237e-02],
```

Ένα απόκομμα από τα βάρη και τα
biases του πρώτου layer
για τις κατηγορίες 1^{ου} επιπέδου

```
For Our Dense Layer:
Weights
[<tf.Variable 'dense_19/kernel:0' shape=(200, 360) dtype=float32, numpy=
array([[ 0.16083297, -0.03679974,  0.05701152, ..., -0.00777473,
        -0.07403651, -0.13363732],
       [ 0.06117297,  0.07225601, -0.09685433, ...,  0.02799341,
         0.03613174, -0.01861896],
       [-0.01065143, -0.05725872,  0.03353362, ..., -0.09649805,
        -0.1338036 , -0.01854796],
       ...,
       [ 0.04743296,  0.06738761,  0.08971276, ...,  0.0265082 ,
         0.05808685,  0.05776981],
       [-0.05420671, -0.02729923,  0.07588771, ...,  0.03473961,
        -0.04263511, -0.011885  ],
       [ 0.13137959,  0.00202987,  0.10173202, ...,  0.01932035,
        -0.01672031, -0.07306501]], dtype=float32)>, <tf.Variable 'dense_19/bias:0' shape=(360,) dtype=float32, numpy=
array([-4.51321155e-02, -5.47489747e-02, -4.68952917e-02, -8.67969822e-03,
       -2.00329963e-02, -4.43031639e-02, -7.46211559e-02, -7.15229511e-02,
       -8.88460875e-02, -4.35733125e-02, -2.58802660e-02, -6.15196750e-02,
       -7.10522458e-02, -7.44278580e-02, -6.89489543e-02, -1.21119404e-02,
       -9.22878757e-02, -5.85444346e-02, -5.93433045e-02, -4.46713306e-02,
       -7.70798177e-02, -3.41474749e-02, -4.78016958e-02, -4.89741489e-02,
       -4.87829670e-02, -5.23409620e-02, -5.86605370e-02, -6.34031519e-02,
       -2.53669042e-02, -4.17570621e-02, -7.19582736e-02, -8.34836066e-02,
       -7.58676901e-02, -6.51195645e-02, -8.00011754e-02, -7.24392831e-02,
        1.19250271e-05, -4.55454960e-02, -4.27782536e-02, -6.47367835e-02,
       -6.61729202e-02, -7.49316961e-02, -6.29157871e-02, -4.46629971e-02,
       -6.34918362e-02, -4.48422432e-02, -7.56853893e-02, -4.55105342e-02,
       -9.18577313e-02, -7.07843006e-02, -4.14030477e-02, -7.56503344e-02,
       -5.84711842e-02, -9.19819921e-02, -4.84936908e-02,  3.67901614e-03,
```

Ένα απόκομμα από τα βάρη και τα
biases του δεύτερου layer
για τις κατηγορίες 1^{ου} επιπέδου

```
Sorry but our Dropout Layer doesnt have any weights or biases
For Our Dense Layer:
Weights
[<tf.Variable 'dense_20/kernel:0' shape=(360, 17) dtype=float32, numpy=
array([[-0.02348287,  0.06934157, -0.12795876, ..., -0.1149531 ,
         0.08355093, -0.06026338],
       [-0.15985797,  0.13627583,  0.10141062, ..., -0.03007158,
         0.08291596, -0.03432193],
       [ 0.03148609, -0.10998986, -0.00219149, ...,  0.06921351,
        -0.01304887, -0.07222434],
       ...,
       [-0.06934445, -0.04132098,  0.05525555, ..., -0.06054215,
         0.04442093, -0.05421781],
       [ 0.06037645, -0.02180223,  0.1464304 , ..., -0.06833963,
         0.01455124, -0.07211284],
       [-0.0370078 , -0.06326355,  0.00420887, ..., -0.08605266,
        -0.10008698, -0.00372868]], dtype=float32)>, <tf.Variable 'dense_20/bias:0' shape=(17,) dtype=float32, numpy=
array([-0.06659854, -0.08551707,  0.01112162,  0.03434744, -0.00924592,
       -0.03117106, -0.03028885,  0.01037667,  0.04772525, -0.04176733,
       -0.08652899,  0.07190516, -0.01341844,  0.01587007,  0.12661181,
        0.00423385, -0.15229106], dtype=float32)>]
Bias
[-0.06659854 -0.08551707  0.01112162  0.03434744 -0.00924592 -0.03117106
 -0.03028885  0.01037667  0.04772525 -0.04176733 -0.08652899  0.07190516
 -0.01341844  0.01587007  0.12661181  0.00423385 -0.15229106]
```

Ένα απόκομμα από τα βάρη και τα
biases του layer εξόδου για τις κατηγορίες
1^{ου} επιπέδου

Εικόνα 4.2.4

TIME RECORDING

Το μηχάνημα πάνω στο οποίο πραγματοποιήσαμε τα πειράματα έχει τα εξής χαρακτηριστικά :

(I) 11th Gen Intel(R) Core(TM) i5 – 1135G7 @ 2.40 GHz

(II) 16 GB RAM

Ο χρόνος εκπαίδευσης του νευρωνικού δικτύου για την ταξινόμηση των στοιχείων - κατηγοριών του 1^{ου} επιπέδου , κυμαίνεται από 47 έως 52 seconds συνολικά , με την κάθε επανάληψη να χρειάζεται περίπου 4-5s και 10ms/step , για 437 δεδομένα εισόδου τη φορά (*χρησιμοποιώντας batch size = 16*) . Το ίδιο ισχύει και για τον χρόνο εκπαίδευσης του νευρωνικού δικτύου για την ταξινόμηση των στοιχείων του 2^{ου} επιπέδου . Για τις δύο αυτές κατηγορίες προπονούμε τα νευρωνικά μας δίκτυα για συνολικά 10 επαναλήψεις , όπως φαίνεται και από την *Εικόνα 4.3.1* . Τέλος , ο χρόνος για την πρόβλεψη των κλάσεων (*ανεξαρτήτου επιπέδου κατηγοριών*) ισούται με 1s με 5ms/step .

Τέλος , ο (*συνολικός*) χρόνος απόκρισης του δικτύου μας , προκύπτει αν προσθέσουμε τους χρόνους εκπαίδευσης και πρόβλεψης των νευρωνικών μας δικτύων για τις κατηγορίες 1^{ου} και 2^{ου} επιπέδου . Κυμαίνεται μεταξύ των 95 και 104 δευτερολέπτων και αν συνυπολογίσουμε σε αυτόν και τον χρόνο που χρειάζεται για την δημιουργία των *word embeddings* (*= 4min και 6sec*) τότε , ο συνολικός χρόνος είναι ίσος με περίπου 5min και 41 – 52 δευτερόλεπτα .

Στο δεύτερο μηχάνημα πάνω στο οποίο πραγματοποιήσαμε τα πειράματα έχει τα εξής χαρακτηριστικά :

(I) AMD Ryzen 5 3400G with Radeon Vega Graphics

(II) 16 GB RAM

Σε αυτό το μηχάνημα , οι χρόνοι εκπαίδευσης , κατεγράφησαν ίσοι με 17.98 δευτερόλεπτα τόσο για τις κατηγορίες πρώτου και δευτέρου επιπέδου , οι χρόνοι πρόβλεψης ήταν επίσης ίσοι με 1 δευτερόλεπτο και ο συνολικός χρόνος απόκρισης του δικτύου ήταν 5min και 10-20 δευτερόλεπτα , άρα λίγο πιο μικρός .

```
Epoch 3/10
437/437 [==============================] - 4s 10ms/step - loss: 2.3921 - accuracy: 0.9914 - precision: 0.5634 - recall: 0.2671 - val_loss: 2.6728 - va
l_accuracy: 0.9909 - val_precision: 0.5050 - val_recall: 0.2587
Epoch 4/10
437/437 [==============================] - 5s 11ms/step - loss: 2.2158 - accuracy: 0.9918 - precision: 0.6015 - recall: 0.3191 - val_loss: 2.7411 - va
l_accuracy: 0.9914 - val_precision: 0.5577 - val_recall: 0.2959
Epoch 5/10
437/437 [==============================] - 5s 10ms/step - loss: 2.0880 - accuracy: 0.9921 - precision: 0.6220 - recall: 0.3588 - val_loss: 2.7854 - va
l_accuracy: 0.9913 - val_precision: 0.5427 - val_recall: 0.3022
Epoch 6/10
437/437 [==============================] - 5s 10ms/step - loss: 2.0404 - accuracy: 0.9923 - precision: 0.6291 - recall: 0.3864 - val_loss: 2.7467 - va
l_accuracy: 0.9913 - val_precision: 0.5421 - val_recall: 0.3205
Epoch 7/10
437/437 [==============================] - 5s 11ms/step - loss: 1.9404 - accuracy: 0.9925 - precision: 0.6409 - recall: 0.4162 - val_loss: 2.7288 - va
l_accuracy: 0.9914 - val_precision: 0.5609 - val_recall: 0.2977
Epoch 8/10
437/437 [==============================] - 5s 10ms/step - loss: 1.8918 - accuracy: 0.9927 - precision: 0.6526 - recall: 0.4322 - val_loss: 3.1619 - va
l_accuracy: 0.9911 - val_precision: 0.5204 - val_recall: 0.3572
Epoch 9/10
437/437 [==============================] - 5s 10ms/step - loss: 1.9117 - accuracy: 0.9926 - precision: 0.6416 - recall: 0.4434 - val_loss: 3.1054 - va
l_accuracy: 0.9913 - val_precision: 0.5361 - val_recall: 0.3440
Epoch 10/10
437/437 [==============================] - 5s 11ms/step - loss: 1.8299 - accuracy: 0.9930 - precision: 0.6673 - recall: 0.4782 - val_loss: 3.1855 - va
l_accuracy: 0.9913 - val_precision: 0.5370 - val_recall: 0.3486
69/69 [==============================] - 1s 5ms/step
Our Predictions
[ 89  75 105 ...  40  51  77]
Our Predictions based on Keras :
69/69 [==============================] - 1s 6ms/step - loss: 4.3004 - accuracy: 0.9908 - precision: 0.0000e+00 - recall: 0.0000e+00


Total time is:  50.91237664222717
```

Εικόνα 4.3.1 ,
ο χρόνος εκπαίδευσης για
τις εκπαιδεύσεις των παραδειγμάτων
είναι ίσος με 50.91 δευτερόλεπτα

GitHub Repository : https://github.com/vstergioulis/NeuroFuzzy_Computing_Final_Project