ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών

«Επιστήμη Δεδομένων και Μηχανική Μάθηση»

Predicting Failures in HPC systems with Data Mining and Deep Learning techniques

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΑΡΙΑ-ΙΩΑΝΝΑ Λ. ΤΖΩΡΤΖΗ

Επιβλέπων : Γεώργιος Γκούμας
Επίκουρος Καθηγητής Ε.Μ.Π.

Συνεπιβλέπων : Νικέλα Παπαδοπούλου
Μεταδιδακτορική Ερευνήτρια Ε.Μ.Π.

Αθήνα, Ιούλιος 2020

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Predicting Failures in HPC systems with Data Mining and Deep Learning techniques

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΑΡΙΑ-ΙΩΑΝΝΑ Λ. ΤΖΩΡΤΖΗ

Επιβλέπων : Γεώργιος Γκούμας
Επίκουρος Καθηγητής Ε.Μ.Π.

Συνεπιβλέπων : Νικέλα Παπαδοπούλου
Μεταδιδακτορική Ερευνήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10η Ιουλίου 2020.

.....	.....	.....
Γεώργιος Γκούμας Επίκουρος Καθηγητής Ε.Μ.Π.	Νεκτάριος Κοζύρης Καθηγητής Ε.Μ.Π.	Διονύσιος Πνευματικάτος Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2020

……

ΜΑΡΙΑ-ΙΩΑΝΝΑ Λ. ΤΖΩΡΤΖΗ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τα τελευταία χρόνια, καθώς εξελίσσεται η υπολογιστική επίδοση των συστημάτων HPC (High Performance Computing - Υπολογισμοί υψηλής επίδοσης), η πολυπλοκότητά τους οδηγεί σε περίπλοκη εκδήλωση σφάλματος. Τα σφάλματα είναι πιο συχνά και αναμένεται να αυξηθούν στα συστήματα της τρέχουσας και της επόμενης γενιάς. Προς το παρόν σημαντική υπολογιστική δυνατότητα (computational capability) και ισχύς χάνεται κατά την ανάκτηση αποτυχημένων components. Για την ανθεκτικότητα αυτών των συστημάτων σε σφάλματα, προτείνονται και αξιοποιούνται εφαρμογές που είναι ανεκτικές σε σφάλματα, προσεγγίσεις ανάκτησης (recovery), όπως τα checkpoints, restart, καθώς και καλύτερη κατανόηση των logs (αρχεία καταγραφής) του συστήματος. Μια εναλλακτική και αναγκαία τώρα λύση είναι η πρόβλεψη failure με καθορισμένο lead time καθώς και ο εντοπισμός του κόμβου στον οποίο πρόκειται να εμφανιστεί.

Ο στόχος της διπλωματικής αυτής εργασίας είναι να μελετήσει/οπτικοποιήσει αρχικά τα logs του υπερυπολογιστή MIRA και στη συνέχεια να αναπτύξει μια γενική μεθοδολογία για την πρόβλεψη αποτυχιών. Αυτή η μεθοδολογία συμπεραίνει failure chains και τελικά εντοπίζει τα ids των κόμβων για τον χωροταξικό εντοπισμό της αποτυχίας. Επιπλέον, χρησιμοποιούντας αυτή τη μεθοδολογία μπορούμε να εξάγουμε το lead time.

Για την επίτευξη των παραπάνω αρχικά τα logs αναλύονται και οπτικοποιούνται χρησιμοποιώντας python3 και pandas. Στη συνέχεια χρησιμοποιούνται τα LSTM (Long Short-Term Memory) για την πρόβλεψη failure chains που οδηγούν σε διακοπή της εκτέλεσης των εφαρμογών που εκτελούνται σε συγκεκριμένους κόμβους. Πραγματοποιείται μια ανάλυση μη ετικετοποιημένων logs, τα οποία μπορούν ή όχι να οδηγήσουν σε failure chain. Υπάρχει μια προσέγγιση τριών φάσεων βαθιάς μάθησης, πρώτα γίνεται εκπαίδευση για την πρόβλεψη των επόμενων φράσεων, δεύτερον γίνεται επανεκπαίδευση μόνο για ακολουθίες φράσεων που οδηγούν σε failure chains επαυξημένες με τα αναμενόμενα lead times και τρίτον προβλέπεται το lead time κατά τη διάρκεια της δοκιμής ώστε να προβλέπεται ποιος συγκεκριμένος κόμβος αποτυγχάνει σε πόσα λεπτά.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

Failure Chains, Lead Time, Artificial Intelligence, Deep Learning, Natural Language Processing, Skip-Gram, LSTM

Abstract

While the computation capacity of HPC systems increases, their complexity leads to complex fault manifestation. Faults are frequent and expected to increase in the systems of this and the next generation. Currently, substantial compute capacity and power is wasted in recovering failed components. Towards the resiliency of HPC systems, the community proposes and implements various solutions, such as fault-tolerant application, recovery techniques (e.g. checkpoint-and-restart), as well as a more thorough understanding of system logs. An alternative, yet imperative, solution is the ability to predict failures with a known lead time, and also to pin-point the node of impending failures.

The aim of this diploma thesis is firstly to study/visualize logs of IBM's supercomputer MIRA and also develop a generic methodology for predicting failures. This methodology infers failure chains and ultimately tracks the node ids to pin-point failure location. What's more using this methodology we can extract the lead time.

To achieve all of the above, at first logs are being visualized and analyzed using python3 and pandas. Following, LSTMs (Long Short-Term memory) are used to predict failure chains leading to the cessation of the execution of applications running at specific nodes. A phrase analysis of unlabeled log entries is performed, which may or may not belong to the failure chain. There is a three-phase deep learning approach to first train to predict next phrases, second re-train only sequence of phrases leading to failure chains augmented with expected lead times and third predict lead times during testing deployment to predict which specific node fails in how many minutes.

KEY WORDS

Failure Chains, Lead Time, Artificial Intelligence, Deep Learning, Natural Language Processing, Skip-Gram, LSTM

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο του Μεταπτυχιακού Προγράμματος Σπουδών Επιστήμη Δεδομένων και Μηχανική Μάθηση της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Πριν όμως από οποιαδήποτε αναφορά στη διαδικασία που ακολουθήθηκε και στα αποτελέσματα που προέκυψαν, θα ήθελα να ευχαριστήσω θερμά τους ανθρώπους με τους οποίους συνεργάστηκα και συνέβαλαν στην ολοκλήρωση της εργασίας αυτής.

Κατ'αρχάς απευθύνω τις ευχαριστίες μου στους επιβλέποντες μου κ. Γεώργιο Γκούμα, Επίκουρο Καθηγητή Ε.Μ.Π. και κ.Νεκτάριο Κοζύρη Καθηγητή Ε.Μ.Π., για την δυνατότητα που μου προσέφεραν να εργαστώ σε ένα αντικείμενο ιδιαίτερα ελκυστικό για μένα και να διευρύνω τις γνώσεις μου. Παράλληλα θα ήθελα να ευχαριστήσω τον κ.Διονύσιο Πνευματικάτο καθηγητή Ε.Μ.Π. που με τίμησε με την παρουσία του στην τριμελή επιτροπή εξέτασης.

Επίσης οφείλω ιδιαίτερες ευχαριστίες στη Νικέλα Παπαδοπούλου, μεταδιδακτορική ερευνήτρια στο Ε.Μ.Π., για το χρόνο που αφιέρωσε και τη θεμελιώδη συνεισφορά της στην εκπόνηση της παρούσας εργασίας. Τόσο η επιστημονική όσο και η πνευματική της στήριξη, ιδιαίτερα στα τελευταία στάδια της εργασίας, ήταν ιδιαίτερα σημαντικές για εμένα. Η εμπειρία και οι γνώσεις της στάθηκαν καθοριστικές και η συνεργασία μας θεωρώ πως ήταν άκρως επιτυχημένη και εποικοδομητική.

Θέλω να ευχαριστήσω τους γονείς μου, την αδελφή μου Πένυ, το αγόρι μου Γιάννη και τους φίλους μου οι οποίοι βρίσκονται κοντά μου και με στηρίζουν όλα αυτά τα χρόνια.

Μαρία - Ιωάννα Τζώρτζη

Αθήνα, 10/07/2020

Contents

Κεφάλαιο 1

Εισαγωγή

1.1 Υπερυπολογιστές

Ένας υπερυπολογιστής (supercomputer) είναι ένας υπολογιστής με υψηλό επίπεδο απόδοσης, σε σύγκριση με έναν υπολογιστή γενικής χρήσης. Η απόδοση ενός υπερυπολογιστή μετράται συνήθως σε υπολογισμούς κινητής υποδιαστολής ανά δευτερόλεπτο, floating-point operations per second (FLOPS), αντί για εκατομμύρια οδηγίες ανά δευτερόλεπτο (MIPS). Οι υπερυπολογιστές εισήχθησαν τη δεκαετία του 1960 και για πολλές δεκαετίες οι γρηγορότεροι έγιναν από τον Seymour Cray στη Control Data Corporation (CDC). Από το 2017 υπάρχουν υπερυπολογιστές οι οποίοι μπορούν να εκτελέσουν πάνω από εκατό τετράκις εκατομμύρια FLOPS (petaFLOPS). Από το Νοέμβριο του 2017, όλοι οι ταχύτεροι υπερυπολογιστές τρέχουν λειτουργικά συστήματα που βασίζονται στα Linux. Για την κατασκευή ταχύτερων, ισχυρότερων και τεχνολογικά ανώτερων υπερσύγχρονων exascale υπολογιστών, έρευνα διεξάγεται στην Κίνα, τις Ηνωμένες Πολιτείες, την Ευρωπαική Ένωση, την Ταϊβάν και την Ιαπωνία.

Οι υπερυπολογιστές διαδραματίζουν σημαντικό ρόλο στον τομέα της υπολογιστικής επιστήμης και χρησιμοποιούνται για ένα ευρύ φάσμα υπολογιστικών εντατικών εργασιών, σε διαφορετικούς τομείς, συμπεριλαμβανόμενης της κβαντικής μηχανικής, της πρόβλεψης του καιρού, της κλιματικής έρευνας, της εξερεύνησης πετρελαίου και φυσικού αερίου, της μοριακής μοντελοποίησης και φυσικών προσομοιώσεων (όπως προσομοιώσεις των πρώτων στιγμών του σύμπαντος, αεροδυναμική αεροπλανών και διαστημικών οχημάτων). Επίσης είναι απαραίτητοι στον τομέα της κρυπτοανάλυσης (cryptanalysis) ("Supercomputer").

Σχήμα 1.1: *Summit - Ο πιο γρήγορος υπερυπολογιστής από το Νοέμβριο του 2019.("Summit").*

Ειδικότερα την τελευταία δεκαετία, έχουν γίνει σημαντικές προσπάθειες για την ανθεκτικότητα τέτοιων συστημάτων Υψηλής Απόδοσης (High Perfomance Computing). Παραδείγματα τέτοιων προσπαθειών είναι οι προσεγγίσεις reactive recovery, όπως τα checkpoint και restart οι οποίες κερδίζουν καλύτερη κατανόηση των αρχείων καταγραφής του συστήματος (system logs) και με τις οποίες γίνονται πιο κατανοητές οι απαιτήσεις όσον αφορά την επίδοση, την αντοχή και τις αντισταθμίσεις ισχύος (power trade-offs). Ενώ λοιπόν τα υπολογιστικά συστήματα έχουν ωριμάσει σχετικά με την αποδοτικότητα των υπολογιστών και τις εξελιγμένες αρχιτεκτονικές, η εκδήλωση σφαλμάτων έχει γίνει πιο πολύπλοκη. Οι υπάρχουσες υποδομές υπολογιστών υψηλής απόδοσης με τα ολοένα κι αυξανόμενα στοιχεία (components) που απαιτούνται για exascale-10^6 κόμβους κάνει δύσκολη την ακριβή πρόβλεψη αποτυχίας (fault prediction). Οι εργασίες που αποτυγχάνουν (aborted jobs) λόγω αποτυχιών κόμβου (node failures) επιφέρουν σημαντικό κόστος ενέργειας. Πάνω από το 20% της υπολογιστικής ικανότητας σπαταλάται σε αποτυχίες και ανάκτηση (failures and recovery) (Elnozahy et al., 2008).

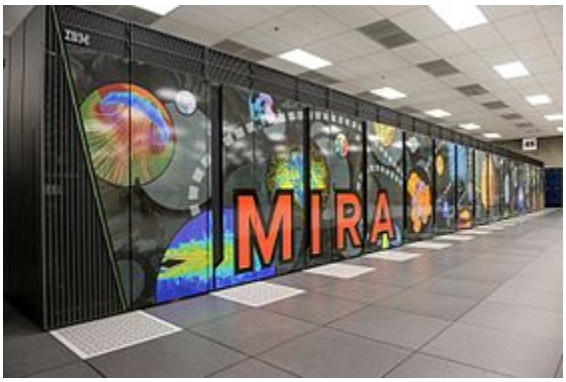
Με τον αυξανόμενο αριθμό κόμβων, ο μέσος χρόνος ανάμεσα στις αποτυχίες (MTBF) μειώνεται για κάθε κόμβο, καθιστώντας ακόμα πιο δύσκολη την αναγνώριση αποτυχιών (fault identification) και την επίλυση τους. Η αυξανόμενη πολυπλοκότητα στα αναδυόμενα συστήματα επόμενης γενιάς παραβλέπει την προσαρμοστική επίλυση αποτυχιών ώστε να αντιμετωπιστεί αυτή η κρίσιμη πρόκληση αξιοπιστίας. Για την καταπολέμηση τέτοιων πρωτοφανών σφαλμάτων και components που "αποτυγχάνουν" είναι αναγκαία η πρόβλεψη failures με ορισμένο lead time.

Πολλές διαφορετικές μελέτες για (failure characterization) [(Gupta et al., 2015), (Gupta et al., 2017)] και επιλύσεις μηχανικής μάθησης [(Lan et al., 2009), (Gainaru et al., 2012)] έχουν γίνει για ανίχνευση ανωμαλιών πάνω σε υπολογιστικά συστήματα μεγάλης κλίμακας. Ωστόσο οι μελέτες που έχουν γίνει και είναι τεχνολογία αιχμής έχει έλλειψη σε δύο βασικές πτυχές. Αρχικά, τα σφάλματα πρέπει να προβλέπονται ακόμη και όταν οι lead χρόνοι είναι μικροί (της τάξης των λεπτών) μαζί με την ακριβή θέση του σφάλματος.

1.2 MIRA

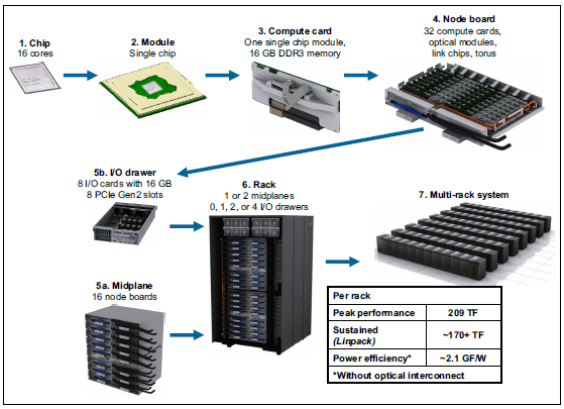
Στη διπλωματική αυτή εργασία έχουν χρησιμοποιηθεί δεδομένα από το Mira. Το μηχάνημα Mira ήταν ένας petascale υπερυπολογιστής Blue Gene / Q. Ο

υπερυπολογιστής αυτός κατασκευάστηκε από την IBM για το Εθνικό Εργαστήριο Argonne με την υποστήριξη του Υπουργείου Ενέργειας των Ηνωμένων Πολιτειών και χρηματοδοτείται μερικώς από το Εθνικό Ίδρυμα Επιστημών. Το Argonne National Laboratory είναι ένα εθνικό εργαστήριο έρευνας επιστήμης και μηχανικής που λειτουργεί από το Πανεπιστήμιο του Σικάγο Argonne LLC για το Υπουργείο Ενέργειας των Ηνωμένων Πολιτειών που βρίσκεται στο Lemont, Illinois, έξω από το Σικάγο. Είναι το μεγαλύτερο εθνικό εργαστήριο ανά μέγεθος και εμβέλεια στη Midwest. Το εργαστήριο αυτό διεξάγει έρευνα σε ένα ευρύ φάσμα κλάδων, λειτουργεί πολλές εγκαταστάσεις και παρέχει πρόσβαση σε πόρους που είναι γενικά υπερβολικά μεγάλοι και ακριβοί ώστε τα πανεπιστημία ή οι εταιρείες να διαθέτουν. Μία από αυτές τις εγκαταστάσεις είναι και η Argonne Leadership Computing Facility (ALCF), στην οποία βρισκόταν και ο υπερυπολογιστής Mira μέχρι και το τέλος του 2019, όπου και αποσύρθηκε για να πάρει τη θέση του ένας πιο ισχυρός υπερυπολογιστής, ο Theta.



Σχήμα 1.2: *Blue Gene/Q supercomputer("Mira").*

Το Mira είναι ένα 10-petaflop σύστημα Blue Gene της IBM, με 48 racks, 49,152 υπολογιστικούς κόμβους, εξοπλισμένο με 786,432 πυρήνες και 768 terabytes μνήμης. Αποτελείται από κόμβους των 16 πυρήνων και κάθε 512 κόμβοι σχηματίζουν ένα 5D τόρο (πλέγμα) σε τοπολογία $4x4x4x4x2$ αποτελώντας ένα midplane. Κάθε rack αποτελείται από δύο midplanes και το σύστημα έχει 48 racks όπως αναφέρθηκε και παραπάνω. Τα 48 αυτά υπολογιστικά racks δηλώνονται από τα R00-R0F, R10-R1F και R20-R2F και τα δύο midplanes του κάθε rack δηλώνονται ως M0 και M1. Από το Νοέμβριο του 2017, είναι στους TOP500, ως ο 9ος ταχύτερος υπερυπολογιστής στον κόσμο, ενώ έκανε το ντεμπούτο του τον Ιούνιο του 2012 στη 3η θέση. (Top500). Διαθέτει διασύνδεση 5Δ-τόρου (5D torus), η οποία μειώνει το μέσο αριθμό των hops και της καθυστέρησης ανάμεσα στους υπολογιστικούς κόμβους και έτσι επιτυγχάνει υψηλή αποδοτικότητα υπολογισμού και εξοικονομεί ενέργεια για τη μεταφορά δεδομένων σε μεγάλες αποστάσεις. Το μικρότερο partition στο Mira είναι 512 κόμβοι. Jobs τα οποία είναι μικρότερα από το ελάχιστο partition τρέχουν αποκλειστικά σε ένα partition. Το μηχάνημα καταγράφει αρχείο από προγραμματισμένα γεγονότα batch jobs συμπεριλαμβανομένων των αιτημάτων και των χρησιμοποιημένων πόρων (π.χ. ο απαιτούμενος-requested χρόνος, ο χρησιμοποιούμενος-used χρόνος, οι χρησιμοποιούμενοι κόμβοι κλπ) καθώς επίσης και timestamps σημαντικών προγραμματιζόμενων γεγονότων όπως ώρα υποβολής, ώρα έναρξης, ώρα λήξης, επίσης δεν λείπει και η καταγραφή πληροφοριών σχετικά με το περιβάλλον της εκτέλεσης ενός job, όπως τα partitions του μηχανήματος.



Σχήμα 1.3: *Blue Gene/Q Hardware Overview(Lakner et al., 2013).*

Μερικές από τις πιο αξιοσημείωτες ερευνητικές εξελίξεις που έγιναν δυνατές από το Mira είναι οι παρακάτω:

1. Μοντελοποίηση υπερκαινοφανών αστέρων (supernovae) με πρωτοφανές επίπεδο λεπτομερειών.

2. Προσομοίωση συγκρούσεων συματιδίων για το Μεγάλο Επιταχυντή Αδρονίων (Large Hadron Collider).

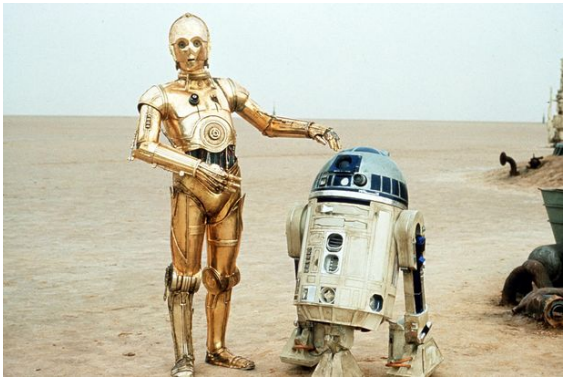
3. Προχωρημένη τεχνική μικροσκοπίας για έγκαιρη ανίχνευση καρκίνου.

4. Επιτάχυνση του σχεδιασμού πιο αποδοτικών κινητήρων.

1.3 Τεχνητή Νοημοσύνη

Από την αρχαιότητα ακόμα είχαν κάνει την εμφάνιση τους μέσα από παραμύθια και ιστορίες τεχνητά όντα με δυνατότητα σκέψης. Ωστόσο η σύγχρονη *Τεχνητή Νοημοσύνη* ("Artificial Intelligence") ξεκινά από την δεκαετία του 1940. Η πρόσβαση των ερευνητών στους ηλεκτρονικούς υπολογιστές καθώς επίσης και η εισαγωγή νέων ιδεών όπως η θεωρία παιγνίων του John von Neumann το 1944 και τα τεχνητά νευρωνικά δίκτυα (McCulloch-Pitts, 1943) οδήγησαν ερευνητές από διάφορους τομείς στο να εξετάσουν την πιθανότητα δημιουργίας ενός τεχνητού εγκεφάλου ή μιας σκεπτόμενης μηχανής. Μετά το Β' Παγκόσμιο Πόλεμο ο Βρετανός μαθηματικός Alan Turing με το άρθρο του Computing Machinery and Intelligence συζήτησε τις συνθήκες για την εξέταση μιας μηχανής ως έξυπνης. Ισχυρίστηκε ότι εάν η μηχανή μπορούσε να προσποιηθεί με επιτυχία ότι είναι άνθρωπος σε έναν έμπειρο παρατηρητή τότε σίγουρα θα πρέπει να θεωρηθεί έξυπνο - το περίφημο Turing test(Harasim, 2012). Το 1956, το πεδίο της έρευνας της Τεχνητής Νοημοσύνης γεννήθηκε σε ένα συνέδριο στο Darthmouth College στις Ηνωμένες Πολιτείες της Αμερικής. Κατά τη διάρκεια του συνεδρίου ο John McCarthy όρισε την Τεχνητή Νοημοσύνη ως η επιστήμη και η μηχανική της δημιουργίας ευφυών μηχανών και ειδικότερα ευφυών προγραμμάτων(McCarthy et al., 2006). Οι περισσότεροι από αυτούς που παρακολούθησαν το συνέδριο έγιναν οι πρώτοι ερευνητές της νέας επιστήμης, και ίδρυσαν εργαστήρια στα μεγαλύτερα πανεπιστήμια των Ηνωμένων Πολιτειών. Από τη στιγμή αυτή και μέχρι τα μέσα της δεκαετίας του 1970, η έρευνα στον τομέα της Τεχνητής Νοημοσύνης άνθησε. Τα αποτελέσματα ήταν εντυπωσιακά: οι ηλεκτρονικοί

υπολογιστές μάθαιναν να λύνουν προβλήματα άλγεβρας, να καταλαβαίνουν και να εκτελούν απλές εντολές σε φυσική γλώσσα. Η χρηματοδότηση ήταν γενναιόδωρη και χωρίς περιορισμούς και οι προσδοκίες των ερευνητών ήταν υψηλές: ο Marvin Minsky το 1967 δήλωσε ότι μέσα σε μια γενιά, το πρόβλημα της δημιουργίας τεχνητής νοημοσύνης θα είχε λυθεί(Minsky, 1967).



Σχήμα 1.4: *C-3PO και R2-D2 από την ταινία ο Πόλεμος των Άστρων (imdb).*

Ωστόσο η έρευνα στην πορεία σκόνταψε σε κάποια προβλήματα τα οποία δεν λύνονταν εύκολα, και τα πρώτα χρόνια της δεκαετίας του 70 έφτασαν χωρίς τα αναμενόμενα αποτελέσματα. Η έλλειψη αποτελεσμάτων οδήγησε σταδιακά στην διακοπή της χρηματοδότησης και άρχισε ο πρώτος "χειμώνας" της Τεχνητής Νοημοσύνης.

Στις αρχές της δεκαετίας του 1980, το ενδιαφέρον για την Τεχνητή Νοημοσύνη άρχισε να ανακάμπτει με την δημιουργία ενός νέου είδους προγραμμάτων τεχνητής νοημοσύνης, των "έμπειρων συστημάτων" (McCorduck, 2004). Τα έμπειρα συστήματα είναι προγράμματα που απαντάνε σε ερωτήσεις ή βγάζουν χρήσιμα συμπεράσματα για έναν πολύ συγκεκριμένο τομέα, χρησιμοποιώντας ένα σύνολο από κανόνες και μια βάση δεδομένων που περιέχει γνώσεις σχετικές με τον τομέα. Οι κανόνες και οι γνώσεις εισάγονται στο σύστημα από ειδικούς με εμπειρία στον συγκεκριμένο τομέα. Τα έμπειρα συστήματα άρχισαν να χρησιμοποιούνται με επιτυχία από επιχειρήσεις παγκοσμίως, με αποτέλεσμα να δημιουργηθεί μια αγορά δισεκατομμυρίων από την πώληση των εξειδικευμένων μηχανημάτων που έτρεχαν τα συστήματα. Οι ερευνητές την δεκαετία αυτή επικεντρώθηκαν σε συστήματα βασισμένα στην γνώση και την αναπαράστασή της.

Το ενδιαφέρον όμως δεν κράτησε για πολύ μιας και οι ηλεκτρονικοί υπολογιστές αποκτούσαν όλο και αυξανόμενη ταχύτητα, ξεπερνώντας τα μηχανήματα έμπειρων συστημάτων σε υπολογιστική ισχύ.

Από τις αρχές της δεκαετίας του 1990 μέχρι και σήμερα, ο τομέας της Τεχνητής Νοημοσύνης έχει σταδιακά ανακάμψει και έχει πετύχει αρκετούς από τους αρχικούς στόχους που είχαν θέσει οι πρώτοι ερευνητές. Τα αποτελέσματα της έρευνας αυτών των χρόνων όλο και περισσότερο χρησιμοποιούνται σε πρακτικές εφαρμογές. Η Τεχνητή Νοημοσύνη χρησιμοποιείται στην εξόρυξη δεδομένων, την ιατρική διάγνωση και άλλους τομείς. Η επιτυχία αυτή οφείλεται στην αύξηση της υπολογιστικής ισχύος, στην επίλυση συγκεκριμένων προβλημάτων, στους νέους δεσμούς μεταξύ της Τεχνητής Νοημοσύνης και άλλων τομέων (όπως στατιστική, οικονομικά και μαθηματικά) και στην αφοσίωση των ερευνητών σε μαθηματικές μεθόδους και επιστημονικά πρότυπα.

Σύμφωνα με τον Jack Clark του Bloomberg, το 2015 ήταν ένα έτος ορόσημο

για την Τεχνητή Νοημοσύνη, με τον αριθμό των προγραμμάτων λογισμικού που χρησιμοποιούν Τεχνητή Νοημοσύνη στη Google να αυξάνεται ραγδαία. Πλέον έχουμε ένα τεράστιο όγκο δεδομένων, διότι η ανθρωπότητα λόγω της δραστηριότητάς της στο Ίντερνετ έχει αφήσει ένα τεράστιο ψηφιακό αποτύπωμα, το οποίο μπορούμε να εκμεταλλευτούμε για να προπονήσουμε τους αλγόριθμους τεχνητής νοημοσύνης. Πολλοί ερευνητές και επιστήμονες εκφράζουν βέβαια μια πιο δυστοπική εκδοχή του μέλλοντος, σύμφωνα με την οποία η Τεχνητή Νοημοσύνη είναι δυνατόν να προκαλέσει σοβαρές κοινωνικές αναταραχές και να αποτελέσει άμεση απειλή για τη δημοκρατία και την παγκόσμια ειρήνη. Ωστόσο όποιος προβληματισμός κι αν υπάρχει δεν μπορεί να σταματήσει το κύμα της προόδου της τεχνολογίας: "Είναι σαν να γυρίζουμε την πλάτη σε ένα κύμα που έρχεται να παρασύρει τα πάντα". Το πιο σημαντικό είναι να σκεφτόμαστε πώς μπορούμε να κατευθύνουμε αυτή την αναμορφωτική δύναμη με τρόπο ώστε να αλλάξει θετικά τη ζωή του ανθρώπου (Δασκαλάκης, 2018).

1.4 Μηχανική Μάθηση

Η *Μηχανική Μάθηση* (Machine Learning) είναι ένα υποπεδίο της επιστήμης των υπολογιστών, το οποίο προέκυψε από την τομή της Επιστήμης των Υπολογιστών με την Στατιστική (Mitchell, 2006). Μία πολύ σημαντική ανακάλυψη από τον Arthur Samuel το 1959, είναι η υλοποίηση ότι αντί να διδάσκεται στους υπολογιστές ό,τι πρέπει να ξέρουν για τον κόσμο και τον τρόπο εκτέλεσης καθηκόντων, ίσως είναι δυνατόν να διδαχθούν πως να μάθουν για τον εαυτό τους. Ο Samuel ορίζει τη μηχανική μάθηση ως "Πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί". Την περασμένη δεκαετία, η μηχανική μάθηση μας έδωσε αυτόματη-οδήγηση αυτοκινήτων, πρακτική αναγνώριση ομιλίας, αποτελεσματική αναζήτηση ιστού καθώς και πολύ βελτιωμένη κατανόηση του ανθρώπινου γονιδιώματος. Η μηχανική μάθηση είναι τόσο διαδεδομένη σήμερα που πιθανώς να χρησιμοποιείται από τον κάθε άνθρωπο δεκάδες φορές την ημέρα χωρίς να γίνεται άμεσα αντιληπτό. Στην ουσία η μηχανική μάθηση είναι μια κατηγορία αλγορίθμων που είναι σε θέση να κάνουν προβλέψεις και να προσαρμοστούν όταν παρουσιάζονται νέα δεδομένα. Ένας αλγόριθμος μηχανικής μάθησης στην αρχή είναι τυπικά ανακριβής, αλλά βελτιώνεται με την πρόβλεψη πάνω σε νέα δεδομένα και προσαρμόζεται βάσει μιας διαδικασίας δοκιμής και σφάλματος. Η μέθοδος αυτή είναι σε θέση να λύσει προβλήματα που ο κλασσικός προγραμματισμός αδυνατεί, με μεγαλύτερο όμως κόστος σε υπολογιστικούς πόρους και χρόνο. Επίσης με την αύξηση των δεδομένων που έχει στην διάθεση του ένας τέτοιος αλγόριθμος, είναι σε θέση να επιλύει δυσκολότερα προβλήματα, όπως αναγνώριση εικόνας, ήχου, φυσικής γλώσσας με μεγαλύτερη ακρίβεια.

Υπάρχουν τρεις μεγάλες κατηγορίες αλγορίθμων μηχανικής μάθησης, ανάλογα με τον τρόπο που επεξεργάζονται τα δεδομένα εισόδου και την ανατροφοδότηση που υπάρχει στο σύστημα εκμάθησης(J Russell and Norvig, 1995):

1. Επιβλεπόμενη μάθηση (Supervised learning): Οι αλγόριθμοι δέχονται ως είσοδο τα δεδομένα μαζί με τις ετικέτες τους και καλούνται να μάθουν ένα γενικό κανόνα προκειμένου να βρίσκουν την αντιστοίχιση (συσχέτιση) μεταξύ δεδομένων και ετικετών.

2. Μη-επιβλεπόμενη μάθηση (Unsupervised learning): Οι αλγόριθμοι δέχονται ως είσοδο δεδομένα χωρίς ετικέτες (χωρίς να παρέχεται δηλαδή κάποια εμπειρία

στον αλγόριθμο μάθησης) και καλούνται να βρουν την δομή τους. Η μη-επιβλεπόμενη μάθηση μπορεί να ανακαλύπτει κρυμμένα μοτίβα σε δεδομένα ή μπορεί να αποτελέσει μέσο για την εύρεση χαρακτηριστικών, που στη συνέχεια χρησιμοποιούνται στη διαδικασία της μάθησης.

3. Ενισχυτική μάθηση (Reinforcement learning): Αλγόριθμοι που αλληλεπιδρούν με ένα δυναμικό περιβάλλον, για την επίτευξη ενός συγκεκριμένου στόχου. Στην περίπτωση αυτή δεν υπάρχει πληροφορία που να υποδεικνύει πόσο κοντά ή μακρυά έχει φτάσει η εκπαίδευση από τον στόχο. Παράδειγμα τέτοιας μάθησης θα μπορούσε να είναι η αυτόνομη οδήγηση ενός οχήματος.

Στη παρούσα εργασία θα εφαρμόσω αρχικά μη-επιβλεπόμενη μάθηση και στη συνέχεια επιβλεπόμενη.

1.5 Βαθιά Μάθηση

Ο όρος *Βαθιά Μάθηση* (Deep Learning) εισήχθη στην κοινότητα της Μηχανικής Μάθησης από τη Rina Dechter το 1986. Η Βαθιά Μάθηση-Deep Learning είναι μια υποπεριοχή της μηχανικής μάθησης. Για τη Βαθιά Μάθηση σε αντίθεση με τα απλά τεχνητά νευρωνικά δίκτυα (Artificial Neural Network - ANN) ισχύει ότι υπάρχουν περισσότερα επίπεδα στοιβαγμένα το ένα μετά το άλλο με αποτέλεσμα να δίνεται η δυνατότητα να εξάγονται περισσότερα υψηλού επιπέδου χαρακτηριστικά.

Αρχιτεκτονικές βαθιάς μάθησης όπως τα δίκτυα βαθιάς πίστης, (deep belief networks), τα CNN, τα RNN έχουν εφαρμοστεί σε πεδία που περιλαμβάνουν την όραση υπολογιστών, (computer vision), την αναγνώριση ομιλίας, (speech recognition), την επεξεργασία φυσικής γλώσσας, (natural language processing), το φιλτράρισμα κοινωνικών δικτύων, τη βιοπληροφορική και τον σχεδιασμό φαρμάκων και έχουν παράγει συγκρίσιμα αποτελέσματα και σε ορισμένες περιπτώσεις καλύτερα από αυτά των ανθρώπων. Οι τεχνικές Βαθιάς Μάθησης λοιπόν έχουν βελτιώσει την ικανότητα ταξινόμησης, αναγνώρισης, ανίχνευσης και περιγραφής - με μια λέξη, την κατανόηση. Παραδείγματα χρήσης της Βαθιάς Μάθησης είναι τα συστήματα Siri και Cortana της Apple και της Microsoft αντίστοιχα.

Σήμερα πολλές είναι οι εξελίξεις που προωθούν τη βαθιά μάθηση, όπως για παράδειγμα οι βελτιώσεις στις αρχιτεκτονικές και στους αλγορίθμους μάθησης. Επίσης οι νέες προσεγγίσεις μηχανικής μάθησης έχουν βελτιώσει την ακρίβεια των μοντέλων, έχουν αναπτυχθεί νέες τάξεις νευρωνικών δικτύων που ταιριάζουν καλά σε εφαρμογές όπως η μετάφραση κειμένου και η ταξινόμηση των εικόνων. Διαθέτουμε πολύ περισσότερα δεδομένα για την κατασκευή νευρωνικών δικτύων, με πολλά βαθιά επίπεδα. Πολύ βασικό να αναφερθεί είναι ότι τώρα πια υπάρχει στη διάθεση των ανθρώπων απίστευτη υπολογιστική ισχύ, η οποία είναι απαραίτητη για την κατάρτιση βαθιών αλγορίθμων. Με την χρήση GPU επιταχύνονται οι αλγόριθμοι εκπαίδευσης κατά τάξεις μεγέθους, μειώνοντας τους χρόνους λειτουργίας από εβδομάδες σε ημέρες.

Η Βαθιά Μάθηση λοιπόν αλλάζει τον τρόπο σκέψης της αναπαράστασης των προβλημάτων από την υπόδειξη προς τον υπολογιστή στο πως να λύσει το πρόβλημα, στην εκπαίδευση του υπολογιστή για να λύσει το ίδιο το μηχάνημα το πρόβλημα. Η υπόσχεση της Βαθιάς Μάθησης είναι ότι μπορεί να οδηγήσει σε συστήματα πρόγνωσης που γενικεύουν καλά, προσαρμόζονται καλά, βελτιώνεται η απόδοση τους με περισσότερα δεδομένα και είναι πιο δυναμικά από τα συστήματα πρόβλεψης που βασίζονται σε αυστηρούς κανόνες. Δεν χρειάζεται πια να εγκατασταθεί ένα μοντέλο, αντ'αυτού εκπαιδεύεται η εργασία.

1.6 Επεξεργασία Φυσικής Γλώσσας

Η Επεξεργασία Φυσικής Γλώσσας (Natural Language Proccessing - NLP)είναι ένα πεδίο της Τεχνητής Νοημοσύνης το οποίο δίνει τη δυνατότητα στις μηχανές να διαβάζουν, να καταλαβαίνουν και να αντλούν νόημα από τις ανθρώπινες γλώσσες.

Είναι μια αρχή που επικεντρώνεται στην αλληλεπίδραση μεταξύ της επιστήμης δεδομένων και της ανθρώπινης γλώσσας και κλιμακώνεται σε πολλές βιομηχανίες. Σήμερα η επεξεργασία φυσικής γλώσσας εξελίσσεται χάρη στις τεράστιες βελτιώσεις στην πρόσβαση στα δεδομένα και στην αύξηση της υπολογιστικής ισχύος, οι οποίες επιτρέπουν στους ερευνητές να επιτύχουν σημαντικά αποτελέσματα σε τομείς όπως η υγειονομική περίθαλψη, τα μέσα ενημέρωσης, τα οικονομικά και το ανθρώπινο δυναμικό μεταξύ άλλων.

Η Επεξεργασία Φυσικής Γλώσσας θεωρείται ένα από τα πολύ δύσκολα προβλήματα στην επιστήμη υπολογιστών. Είναι η φύση της ανθρώπινης γλώσσας που καθιστά τα NLP δύσκολα. Οι κανόνες που υπαγορεύουν τη μετάδοση πληροφοριών χρησιμοποιώντας φυσικές γλώσσες δεν είναι εύκολο να τις κατανοήσουν οι υπολογιστές. Ορισμένοι από αυτούς τους κανόνες μπορεί να είναι υψηλού επιπέδου (high-leveled) και αφηρημένοι, όπως για παράδειγμα όταν κάποιος χρησιμοποιεί μια σαρκαστική παρατήρηση για να περάσει πληροφορίες. Από την άλλη πλευρά κάποιοι από αυτούς τους κανόνες μπορεί να είναι χαμηλού επιπέδου (low-leveled), όπως για παράδειγμα, χρησιμοποιώντας τον χαρακτήρα "s" για τη δήλωση της πλειονότητας των αντικειμένων. Η πλήρης κατανόηση της ανθρώπινης γλώσσας απαιτεί κατανόηση τόσο των λέξεων όσο και του τρόπου με τον οποίο συνδέονται οι έννοιες για την παράδοση του επιθυμητού μηνύματος. Ενώ λοιπόν οι άνθρωποι μπορούν εύκολα να κυριαρχήσουν μια γλώσσα, η ασάφεια και τα ασαφή χαρακτηριστικά των φυσικών γλωσσών είναι αυτό που καθιστούν τα NLP ένα δύσκολο πρόβλημα ώστε να εφαρμοστεί από τις μηχανές.

Με απλά λόγια λοιπόν η Επεξεργασία Φυσικής Γλώσσας αντιπροσωπεύει τον αυτόματο χειρισμό της φυσικής ανθρώπινης γλώσσας, όπως του λόγου, του κειμένου και παρόλο που η ίδια η έννοια είναι συναρπαστική, η πραγματική αξία πίσω από αυτή την τεχνολογία προέρχεται από τις περιπτώσεις χρήσης της. Η Επεξερασία Φυσικής Γλώσσας μας βοηθάει σε πολλές εργασίες και τα πεδία εφαρμογής της φαίνεται να αυξάνονται καθημερινά. Κάποια από αυτά τα παραδείγματα είναι:

1. Τα NLP επιτρέπουν την αναγνώριση και την πρόβλεψη ασθενειών που βασίζονται σε ηλεκτρονικά αρχεία υγείας και στην ομιλία του ασθενούς. Αυτή η δυνατότητα διερευνάται σε συνθήκες υγείας όπως καρδιαγγειακές παθήσεις μέχρι κατάθλιψη και ακόμη και σχιζοφρένεια. Για παράδειγμα η Amazon Comprehend Medical είναι μια υπηρεσία που χρησιμοποιεί NLP ώστε να εξάγει τις συνθήκες ασθένειας, φαρμακευτική αγωγή και θεραπεία από τις σημειώσεις ασθενών, κλινικών δοκιμών και άλλων ηλεκτρονικών αρχείων υγείας.

2. Οι οργανισμοί μπορούν να καθορίσουν τι λένε οι πελάτες σχετικά με μια υπηρεσία ή ένα προϊόν, προσδιορίζοντας και εξάγοντας πληροφορίες από πηγές όπως τα μέσα κοινωνικής δικτύωσης. Αυτή η ανάλυση συναισθημάτων μπορεί να παρέχει πολλές πληροφορίες σχετικά με τις επιλογές των πελατών και τους οδηγούς αποφάσεων τους.

3. Εταιρείες όπως οι Yahoo, Google, φιλτράρουν και ταξινομούν τα emails με χρήση NLP αναλύοντας το κείμενο στα emails που ρέουν μέσω των servers τους και σταματάνε τα ανεπιθύμητα μηνύματα προτού εισέλθουν στα εισερχόμενα.

4. Για την αναγνώριση ψεύτικων ειδήσεων, η ομάδα NLP στο MIT ανέπτυξε ένα νέο σύστημα για να διαπιστώσει εάν μια πηγή είναι ακριβής ή πολιτικά προκατειλημμένη, εντοπίζοντας αν μια πηγή νέων μπορεί να είναι αξιόπιστη ή όχι.

5. Η Alexa της Amazon και η Siri της Apple είναι παραδείγματα έξυπνων φωνητικών διεπαφών που χρησιμοποιούν NLP για να ανταποκρίνονται στις φωνητικές υποδείξεις και να κάνουν τα πάντα όπως για παράδειγμα να βρίσκουν ένα συγκριμένο κατάστημα, να λένε την πρόγνωση του καιρού, να προτείνουν την καλύτερη διαδρομή για το γραφείο, ή να ενεργοποιούν τα φώτα στο σπίτι.

6. Node Failure Prediction based on unstructured logs.

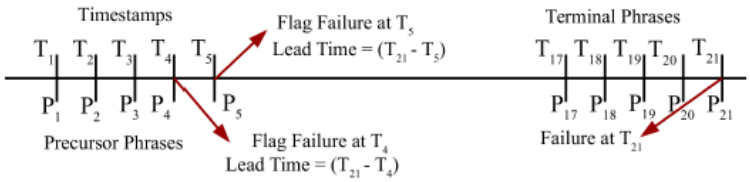
Στην παρούσα εργασία θα επικεντρωθούμε στην πρόβλεψη job failure μέσω της ανάλυσης των logs-αρχείων καταγραφής που παράγονται από τους υπολογιστές μεγάλης κλίμακας. Ωστόσο, η φυσική γλώσσα των αδόμητων logs που παράγονται από τα υπολογιστικά συστήματα αναδεικνύουν δύο προβλήματα. Πρώτον, δεδομένου ότι τα δεδομένα δεν έχουν καμία δομή ούτε ετικέτες, οι συμβατές τεχνικές Μηχανικής Μάθησης υποφέρουν από περιορισμούς όσον αφορά την προεπεξεργασία των δεδομένων αυτών, για παράδειγμα ο σχηματισμός διανυσμάτων χαρακτηριστικών ή ταξινομητών δεν είναι εύκολος. Δεύτερον, είναι ανέφικτο να συμπεράνουμε σύνθετα patterns από τα δεδομένα υψηλής διάστασης γρήγορα, εκτός αν τα δεδομένα αυτά υποβληθούν σε επεξεργασία και τροφοδοτηθούν με κατάλληλη αναπαράσταση εισόδου. Η Βαθιά Μάθηση έχει κάνει τεράστια πρόοδο σε αυτές τις πτυχές πρόσφατα, ειδικά όσον αφορά την κατανόηση της φυσικής γλώσσας. Αυτό ενθαρρύνει την ανάγκη να διερευνυθούν scalable τεχνικές μη επιβλεπόμενης μάθησης στο πλαίσιο της πρόβλεψης αποτυχίας κόμβου. Οι ερευνητές συμφωνούν ότι η πρόβλεψη αποτυχίας (failure prediction) είναι χρήσιμη ακόμα κι αν είναι ατελής και με περιορισμένη ακρίβεια (DOD et al., 2012). Αν υποθέσουμε ότι το 50% των αποτυχιών κόμβου προβλέπονται σωστά και τα υπόλοιπα προβλέπονται εσφαλμένα (ψευδώς θετικά-false positives) τότε μπορούμε να αποτρέψουμε τα μισά από τα ακριβά σε κόστος checkpoints/restarts τα οποία απαιτούν συνολικό συντονισμό με πολύ φθηνότερες παραποιήσεις της διαδικασίας.

1.7 Στόχος Διπλωματικής Εργασίας

Ο σκοπός της διπλωματικής εργασίας εντοπίζεται στη μελέτη των jobs που υποβάλλονται στον υπερυπολογιστή που αναφέραμε στην ενότητα 1.2 και διαχωρίζεται σε δύο βασικά σημεία. Αρχικά γίνεται χρήση ενός εργαλείου που έχει αναπτυχθεί από το Argonne National Laboratory σε συνεργασία με το Πανεπιστήμιο του Illinois για την εύρεση πιθανών συσχετίσεων των logs από HPC. Το εργαλείο αυτό ονομάζεται LogAider. Επιπλέον γίνεται μια **απεικόνιση** και ανάλυση των δεδομένων που έχουμε όσον αφορά τα logs που αναφέρονται στα jobs και στη συνέχεια **προβλέπονται** τα lead times ώστε να παρεμποδιστούν τα failures των jobs. Τα δεδομένα που χρησιμοποιούνται στη διπλωματική αυτή εργασία παράγονται όπως έχει αναφερθεί και προηγουμένως από το MIRA και πρόκειται για ανοιχτά και public δεδομένα (Argonne). Στο επόμενο κεφάλαιο θα αναλυθούν περισσότερο.

Σε τέτοια συστήματα όπως προαναφέρθηκε η δυσλειτουργία του υλικού, του λογισμικού ή των εφαρμογών προκαλεί errors. Τα errors διαδίδονται ως faults οδηγώντας σε failures. Αυτή η εργασία αναπτύσσει μια καλύτερη κατανόηση των

εκδηλώσεων των failures στα συστήματα αυτά και προτείνει μια αποτελεσματική τεχνική για την πρόβλεψη failures στο σύστημα, οπότε κι ένας unhealthy κόμβος σταματήσει να ανταποκρίνεται, άρα και ένα job να σταματήσει. Για τον σκοπό αυτό αναλύονται τα logs που παράγονται από τον υπερυπολογιστή MIRA και ανπτύσσονται τεχνικές μηχανικής μάθησης για να διευκολύνουν την πρόληψη για την ανθεκτικότητα των HPC. Στην εργασία ακολουθώ κάποια ορολογία την οποία κατά κύριο λόγο δε θα μεταφράζω. Αυτή είναι η παρακάτω:



Σχήμα 1.5: *Lead time to a failure (Das et al., 2018).*

- **Phrase-Φράση:** Μια φράση αναφέρεται σε ένα γεγονός ή σε ένα log μήνυμα σε οποιοδήποτε log αρχείο το οποίο λαμβάνεται από ένα υπολογιστικό σύστημα. Συνήθως τα log μηνύματα εμφανίζονται με χρονική σήμανση (timestamped), ωστόσο δεν είναι απαραίτητο να είναι έτσι.

- **Lead Time:** Τα μηνύματα ή τα logs καταγράφονται σε κάποια χρονική στιγμή(timestamp). Το Lead Time είναι η χρονική διαφορά ανάμεσα στον χρόνο κατά τον οποίο ένα component έγινε unresponsive και στον χρόνο κατά τον οποίο μία επικείμενη αποτυχία-impeding failure σημειώνεται προληπτικά σε πρότερη φράση. Αυτή η "πρόδρομη" φράση μπορεί ναι είναι ή μπορεί να μην είναι ενδεικτικό από κανένα error και να προηγείται των τερματικών μηνυμάτων που οδηγούν σε failure. Το παραπάνω σχήμα 1.5 απεικονίζει τον ορισμό του Lead Time. Έστω ότι ένα τερματικό μήνυμα το οποίο υποδεικνύει μια επιβεβαιωμένη failure ενός κόμβου εμφανίζεται στο log message P21 την χρονική στιγμή T21. Αν η failure προβλεπτική δομή μας σημειώνει-flags αυτή τη failure μετά τον έλεγχο της φράσης P5, ο υπολογιζόμενος lead time χρόνος είναι $\Delta T = (T21 - T5)$. Αν σημειώνεται νωρίτερα αφότου ελέγχεται η φράση P4 τότε το lead time αυξάνεται σε $\Delta T = (T21 - T4)$, η αύξηση είναι $(T5 - T4)$. Όσο μεγαλύτερο είναι το lead time, τόσο περισσότερος χρόνος παραμένει για να πάρουμε κατάλληλες ενέργειες ανάκαμψης. Ωστόσο σημειώνοντας ενδεχόμενα failures σε πολύ νωρίτερα "πρόδρομα" γεγονότα μπορεί επίσης να οδηγήσει σε λανθασμένες προβλέψεις (false negatives και false positives), μειώνοντας έτσι την ακρίβεια. Αυτό βέβαια καθιστά απαραίτητη και τη μελέτη του lead time sensitivity.

- **Fault:** Η βλάβη είναι μια κατάσταση η οποία οδηγεί το λογισμικό να αποτύχει να εκτελέσει την απαιτούμενη λειτουργία.

- **Error:** Αναφέρεται στη διαφορά ανάμεσα στην πραγματική έξοδο (actual output) και στην αναμενόμενη (expected output).

- **Failure:** Είναι η αδυναμία ενός συστήματος ή στοιχείου-component να εκτελεί την απαιτούμενη λειτουργία σύμφωνα με τις προδιαγραφές του. Τα failures μπορεί να είναι στο hardware, software, ή στο application. Αυτά τα failures

διαφέρουν από τον σκόπιμο μαζικό τερματισμό των κόμβων ή τη περιοδική επανεκκίνηση της υπηρεσίας που σχετίζονται με τη συντήρηση.

- **Failure Chain:** Είναι μια αλυσίδα-ακολουθία μηνυμάτων η οποία οδηγεί σε κάποιο failure του συστήματος. Μια failure chain καθορίζεται από την ύπαρξη κάποιου μηνύματος τερματισμού (failure indicator).

Στην εργασία αυτή δεν εξετάζεται η ακριβής αιτία της ανωμαλίας που οδηγεί σε failure, αντίθετα επιμένει στην ανάλυση και "εξόρυξη" των φράσεων για την αναγνώριση και πρόβλεψη επικείμενων failures πριν αυτές συμβούν.

Ειδικότερα, ο στόχος της εργασίας επιτυγχάνεται με την πρόβλεψη πιθανών failure chains με χρήση LSTM. Κάνουμε ανάλυση των φράσεων μη ετικετοποιημένων logs, οι οποίες φράσεις μπορεί να ανήκουν ή όχι σε failure chains. Τα failure chains αυτά είναι αναγνωρίσιμα από ένα τερματικό log μήνυμα, το οποίο επαληθεύεται σε συννενόηση με τους διαχειριστές του συστήματος. Για την αποτελεσματική επιτυχία της εργασίας ακολουθείται μια προσέγγιση βαθιάς μηχανικής μάθησης τριών φάσεων (Das et al., 2018).

Αρχικά γίνεται εκπαίδευση ώστε να αναγνωρίζονται αλυσίδες από log events τα οποία οδηγούν σε failures, στη συνέχεια εκπαιδεύουμε εκ νέου τις αλυσίδες αναγνώρισης των γεγονότων αυτή τη φορά όμως έχοντας και την πληροφορία των αναμενόμενων lead times μέχρι το failure. Τέλος προβλέπουμε τα lead times κατά τη διάρκεια της ανάπτυξης του testing/inference όπου προβλέπουμε τον απρόβλεπτο τερματισμό ενός job και σε ποιο συγκεκριμένο κόμβο λαμβάνει χώρα το failure αυτό.

Η μεθοδολογία παρουσιάζεται εκτενώς στην ενότητα 3.2.

Κεφάλαιο 2

Θεωρία

2.1 Δεδομένα

Όπως αναφέρθηκε και στην ενότητα για τον στόχο της διπλωματικής εργασίας, 1.7, χρησιμοποιώ logs από τον υπερυπολογιστή Mira, τα οποία και αναλύω ώστε μετέπειτα να αναπτύξω τη μεθοδολογία για την ανίχνευση failure chains. Τα δεδομένα αυτά είναι ανοιχτά και δημόσια (Argonne) και έτσι δίνεται η δυνατότητα να τα αναλύσουμε για να καταλάβουμε τη συμπεριφορά των χρηστών, την επίδοση των μηχανημάτων, την εμφάνιση σφαλμάτων και επομένως να εφαρμόσουμε διάφορες τεχνικές που βοηθάνε στο να αξιοποιούμε καλύτερα την υπολογιστική ισχύ των υπερυπολογιστών. Συγκεκριμένα για το πρώτο σκέλος του visualization χρησιμοποιούμε τα job logs. Στη συνέχεια για την ανάπτυξη της μεθοδολογίας πρόβλεψης failure συνδυάζουμε δύο είδη logs του συστήματος, τα RAS και τα JOB logs ώστε να εκμεταλλευτούμε τα χαρακτηριστικά των jobs βασισμένα στις καταστάσεις εκτέλεσης τους.

Τα **RAS logs** (reliability, availability and serviceability) είναι από τα πιο σημαντικά logs του συστήματος επειδή υποδηλώνει την αξιοπιστία του. Κάθε ένα από τα στοιχεία στα RAS logs αντιπροσωπεύεται ως συγκεκριμένο γεγονός με ένα από τα τρία επίπεδα σοβαρότητας (INFO, WARN, FATAL). Ωστόσο στην εργασία αυτή δεν λαμβάνονται υπόψη τα επίπεδα αυτά σοβαρότητας. Αυτό συμβαίνει γιατί δεν αποτελούν αποτελεσματική κατηγοριοποίηση των μηνυμάτων σε μακροχρόνια δεδομένα. Κάποια φαινομενικά "ασφαλή" μηνύματα μπορεί να οδηγήσουν σε failures. Από τα 17 πεδία, μερικά μόνο είναι κρίσιμα για τη μελέτη μας, όπως MESSAGE ID, EVENT_TIME, BLOCK, LOCATION, MESSAGE. Τα RAS logs είναι τα πιο σημαντικά logs που συνδέονται με την αξιοπιστία του συστήματος, όπως node failure και διακοπές ρεύματος.

Η υποδομή λοιπόν του reliability, availability και του serviceability (RAS) παρέχει τα μέσα για να αναφερθούν τα συμβάντα που συμβαίνουν στο hardware και στο software για το σύστημα Blue Gene/Q. Υπάρχει ενσωματωμένο ένα προκαθορισμένο σύνολο μηνυμάτων, αλλά ο σχεδιασμός επίσης επιτρέπει στους χρήστες να καθορίζουν τα δικά τους RAS logs που καταγράφονται στη βάση δεδομένων. Αυτά τα logs είναι η κύρια πηγή πληροφορίας που μπορεί να χρησιμοποιήσει ένας διαχειριστής του συστήματος για να κατανοήσει τα failures.

Στο MIRA οι χρήστες που θέλουν να τρέξουν μια εφαρμογή ή προσομοίωση υψηλής υπολογιστικής απόδοσης πρέπει να υποβάλλουν ένα job στον job scheduler του συστήματος. Τα **JOB logs** περιλαμβάνουν πληροφορίες σχετικά με τα jobs που έχουν υποβληθεί. Από τα 57 πεδία που υπάρχουν στα logs που

καταγράφουν την κατάσταση του κάθε job, όπως την κατάσταση της ουράς, του τρεξίματος, τον αριθμό των κόμβων ή πυρήνων που έχουν χρησιμοποιηθεί, τον χρόνο που ολοκληρώνονται τα jobs και το exit status, εμείς χρησιμοποιούμε κάποια από αυτά (START_TIMESTAMP, END_TIMESTAMP, QUEUE_NAME, WALLTIME_SECONDS κλπ).

Γιατί είναι σημαντική όμως η πρόβλεψη πιθανών failures στα υψηλής απόδοσης συστήματα;

Έχοντας πιο αξιόπιστα συστήματα οδηγούμαστε σε πιο παραγωγικούς επιστημονικούς υπολογισμούς και έτσι, γρηγορότερη επιστημονική εξέλιξη. Ωστόσο η βελτίωση και η διατήρηση υψηλού επιπέδου αξιοπιστίας του συστήματος έχει πολλές προκλήσεις για πολλούς λόγους. Κατ'αρχάς ο αριθμός των components αυξάνεται ραγδαία σε μεγάλης κλίμακας HPC συστημάτων, ώστε να μπορούν να ανταποκριθούν στις απαιτήσεις υπολογιστικής δύναμης από τους επιστήμονες και έτσι η πιθανότητα αποτυχίας αυξάνεται. Κατά δεύτερον λόγω της συρρίκνωσης σε μέγεθος των επεξεργαστών γίνονται πιο εύκολα λάθη που σχετίζονται με την αλλαγή της επεξεργασίας και των κατασκευαστικών ελαττωμάτων (Cappello et al., 2014). Τρίτον η πολυπλοκότητα των συστημάτων αυξάνεται με αποτέλεσμα η διαχείριση της αξιοπιστίας του συστήματος να γίνεται πιο δύσκολη (Lucas et al., 2014).

Με τη μελέτη που έγινε στη συγκεκριμένη εργασία ερευνάμε πώς τα fatal γεγονότα του συστήματος επηρεάζουν τις εκτελέσεις των jobs από την πλευρά του job scheduling του συστήματος. Σε αντίθεση με εργασίες όπως (Di et al., 2018), οι οποίες επικεντρώνονται μόνο σε συσχετίσεις μεταξύ των γεγονότων του συστήματος (πχ RAS), στην εργασία αυτή ακολουθούμε μια μεθοδολογία που αρχικά συσχετίζει τα RAS events του συστήματος με τις εκτελέσεις των Jobs. Αυτή η συσχέτιση είναι πολύ σημαντική στους διαχειριστές συστημάτων, στους χρήστες εφαρμογών και στους ερευνητές καθώς υποδεικνύει την αξιοπιστία του συστήματος από την προοπτική των jobs.

Βέβαια η μελέτη των logs δεν είναι εύκολη καθώς είναι πάρα πολλά σε πλήθος και η συσχέτιση μόνο μεταξύ των RAS και των JOB logs είναι χρονοβόρα και απαιτεί μεγάλη κατανάλωση πόρων. Απαιτητικός ήταν και ο σχεδιασμός και η υλοποίηση όλης της μεθοδολογίας που αναπτύχθηκε στα πλαίσια της διπλωματικής και κατά την οποία αρχικά έπρεπε τα μηνύματα των logs να έρθουν σε μορφή κατάλληλη για να μπορούν να εκπαιδευθούν και στη συνέχεια να γίνουν οι δύο φάσεις εκπαίδευσης. Όλη αυτή η διαδικασία ήταν αρκετά απαιτητική σε επίπεδο πόρων.

Τέλος θα ήθελα να προσθέσω ότι η κατανόηση των job failures στον συγκεκριμένο υπερυπολογιστή έχει ευρύτερη σημασία καθώς πολλοί υπερυπολογιστές όπως Sequoia (USA), Vulcan (USA) και Blue Joule (UK), υιοθετούν επίσης την ίδια αρχιτεκτονική με το σύστημα Blue Gene/Q και τα οποία βρίσκονται ακόμα σε λειτουργία.

2.2 Word-Embeddings

Η εργασία με μη δομημένα δεδομένα κειμένου είναι αρκετά δύσκολη, ειδικά όταν στόχος είναι η δημιουργία ενός ευφυούς συστήματος που ερμηνεύει και κατανοεί την ελεύθερη φυσική γλώσσα όπως ακριβώς και οι άνθρωποι. Θα πρέπει να γίνεται επεξεργασία και μετασχηματισμός από τα θυρυβώδη μη δομημένα δεδομένα κειμένου σε δεδομένα δομημένης και διανυσματικής μορφής τα οποία μπορούν να γίνουν κατανοητά από οποιονδήποτε αλγόριθμο μηχανικής μάθησης. Κάθε αλγόριθμος μηχανικής μάθησης βασίζεται σε αρχές στατιστικής, μαθηματικών και

βελτιστοποίησης. Ως εκ τούτου, δεν είναι αρκετά έξυπνοι για να ξεκινήσουν την επεξεργασία κειμένου στην αρχική, φυσική τους μορφή.

Τα Word-Embeddings είναι μία από τις πιο διάσημες αναπαραστάσεις λεξιλογίου κειμένου στην Επεξεργασία Φυσικής Γλώσσας, όπου λέξεις ή φράσεις από το λεξιλόγιο αναπαρίστανται σε διανύσματα πραγματικών αριθμών. Εννοιολογικά περιλαμβάνει μια μαθηματική ενσωμάτωση από ένα χώρο με πολλές διαστάσεις ανά λέξη σε ένα συνεχές χώρο διανύσματος με πολύ χαμηλότερη διάσταση. Είναι ικανά να συλλάβουν το περιεχόμενο μιας λέξης σε ένα έγγραφο, τη σημασιολογική και συντακτική ομοιότητα, την σχέση με άλλες λέξεις κτλ. Τα word και phrase embeddings όταν χρησιμοποιούνται ως η υποκείμενη αναπαράσταση εισόδου, έχουν αποδείξει ότι ενισχύουν την απόδοση σε εργασίες Επεξεργασίας Φυσικής Γλώσσας, όπως η συντακτική ανάλυση και η ανάλυση συναισθημάτων.

Γιατί τα χρειαζόμαστε; Έστω ότι έχουμε τις ακόλουθες παρόμοιες προτάσεις: Have a good day και Have a great day. Δύσκολα έχουν διαφορετικό νόημα. Αν κατασκευάσουμε ένα εξαντλητικό λεξιλόγιο έστω V, τότε θα ήταν V = Have, a, good, great, day. Στη συνέχεια δημιουργούμε ένα one-hot κωδικοποιημένο διάνυσμα για κάθε μία από αυτές τις λέξεις στο V. Το μήκος του one-hot κωδικοποιημένου διανύσματος μας θα είναι ίσο με το μέγεθος του V(=5). Θα έχουμε ένα διάνυσμα μηδενικών, εκτός από το στοιχείο στο ευρετήριο που αντιπροσωπεύει την αντίστοιχη λέξη στο λεξιλόγιο. Αυτό το συγκεκριμένο στοιχείο θα είναι 1. Οι παρακάτω κωδικοποιήσεις το εξηγούν καλύτερα.

Have = [1,0,0,0,0]‘; a=[0,1,0,0,0]‘ ; good=[0,0,1,0,0]‘ ; great=[0,0,0,1,0]‘ ; day=[0,0,0,0,1]‘ (‘ represents transpose).

Αν προσπαθήσουμε να απεικονίσουμε αυτές τις κωδικοποιήσεις, μπορούμε να σκεφτούμε ένα χώρο 5 διαστάσεων, όπου κάθε λέξη καταλαμβάνει μία από τις διαστάσεις και δεν έχει καμία σχέση με τα υπόλοιπα (καμία προβολή στις άλλες διαστάσεις). Αυτό σημαίνει ότι τα good και great είναι τόσο διαφορετικά όσο και τα day και have, κάτι που δεν είναι αλήθεια. Ο στόχος μας είναι να κατέχουν λέξεις με παρόμοιο περιεχόμενο κοντινές χωρικές θέσεις.

2.3 Word2Vec Model

Το Word2Vec μοντέλο δημιουργήθηκε το 2013 από τη Google (Mikolov et al., 2013b) και είναι ένα προγνωστικό μοντέλο βασισμένο στη βαθιά μάθηση για τον υπολογισμό και τη δημιουργία υψηλής ποιότητας, κατανεμημένων και συνεχών πυκνών διανυσματικών αναπαράστεων των λέξεων, οι οποίες αποτυπώνουν τη συμφραζόμενη και σημασιολογική ομοιότητα. Ουσιαστικά αυτά είναι μοντέλα χωρίς επίβλεψη (unsupervised models), τα οποία μπορούν να καταλάβουν τεράστια κείμενα, να δημιουργήσουν ένα λεξιλόγιο πιθανών λέξεων και να δημιουργήσουν πυκνά word embeddings για κάθε λέξη στο διανυσματικό χώρο που αντιπροσωπεύει αυτό το λεξιλόγιο. Συνήθως καθορίζεται το μέγεθος των διανυσμάτων των word embeddings και ο συνολικός αριθμός των διανυσμάτων είναι ουσιαστικά το μέγεθος του λεξιλογίου. Αυτό καθιστά τη διαστατικότητα αυτού του πυκνού διανυσματικού χώρου πολύ χαμηλότερη από τον υψηλής διαστάσεων αραιό διανυσματικό χώρο που κατασκευάζεται χρησιμοποιώντας τα παραδοσιακά μοντέλα Bag of Words. Υπάρχουν δύο διαφορετικές αρχιτεκτονικές μοντέλων που μπορούν να αξιοποιηθούν από το Word2Vec για τη δημιουργία αυτών των αναπαραστάσεων word embeddings. Αυτές είναι (Mikolov et al., 2013a):

1. **Continuous Bag of Words (CBOW) Model**

2. **Skip-gram Model**

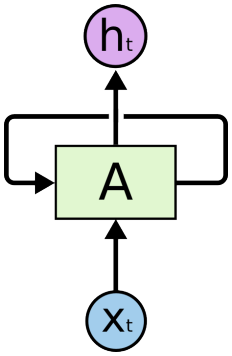
Στην παρούσα διπλωματική εργασία θα χρησιμοποιήσω το Skip-gram μοντέλο το οποίο και παρουσιάζεται στην ενότητα 3.1.

2.4 Recurrent Neural Networks

Οι άνθρωποι δε ξεκινούν τη σκέψη τους από το μηδέν κάθε δευτερόλεπτο. Διαβάζοντας για παράδειγμα ένα βιβλίο κάποιος μπορεί να καταλάβει κάθε λέξη με βάση την κατανόηση των προηγούμενων λέξεων. Δε ξεχνάμε τα πάντα και αρχίζουμε εκ νέου να σκεφτόμαστε. Οι σκέψεις μας έχουν επιμονή.

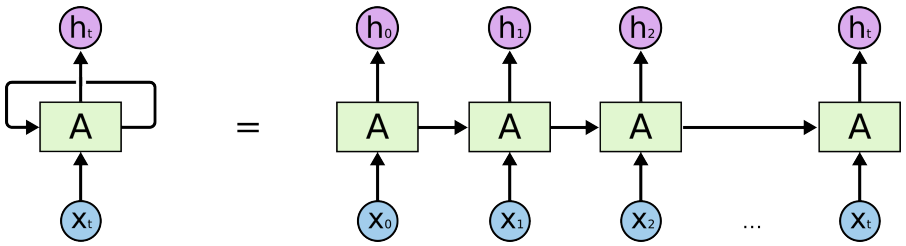
Τα παραδοσιακά νευρωνικά δίκτυα δεν μπορούν να διατηρήσουν όμως την πληροφορία, κάτι που αποτελεί πολύ σημαντικό μειονέκτημά τους. Για παράδειγμα, φανταστείτε ότι θέλετε να ταξινομήσετε τι είδους γεγονός συμβαίνει σε κάθε σημείο μιας ταινίας. Δεν είναι σαφές πώς ένα παραδοσιακό νευρωνικό δίκτυο θα μπορούσε να χρησιμοποιήσει τη συλλογιστική του σχετικά με προηγούμενα γεγονότα της ταινίας για να ενημερώσει τα νεότερα.

Τα Recurrent Neural Networks - RNN αντιμετωπίζουν αυτό το ζήτημα. Είναι δίκτυα με βρόχους μέσα σ'αυτά, επιτρέποντας έτσι την πληροφορία να διατηρείται.



Σχήμα 2.1: *Recurrent Neural Networks have loops. (Olah, 2015).*

Στο παραπάνω διάγραμμα, ένα κομμάτι νευρωνικού δικτύου, A, κοιτάει μια είσοδο x_t και εξάγει μια τιμή h_t. Ένας βρόχος επιτρέπει τη μετάδοση της πληροφορίας από ένα βήμα του δικτύου στο επόμενο. Αυτοί οι βρόχοι κάνουν τα Recurrent Neural δίκτυα να φαίνονται αρκετά μυστηριώδη. Ωστόσο αν κάποιος σκεφτεί για λίγο θα διαπιστώσει ότι δε διαφέρουν πολύ από ενά κανονικό νευρωνικό δίκτυο. Ένα Recurrent Neural δίκτυο μπορεί να θεωρηθεί ως πολλαπλά αντίγραφα του ίδιου του δικτύου, το καθένα από τα οποία διαβιβάζει ένα μήνυμα σε ένα διάδοχο. Η παρακάτω εικόνα είναι όταν ξετυλίγεται ένας βρόχος:



Σχήμα 2.2: *An unrolled recurrent neural network. (Olah, 2015).*

Αυτή η μορφή που μοιάζει με αλυσίδα αποκαλύπτει ότι τα Recurrent Neural δίκτα σχετίζονται στενά με αλληλουχίες και λίστες. Είναι η φυσική αρχιτεκτονική του νευρωνικού δικτύου που θα χρησιμοποιήσει τέτοια δεδομένα.

Τα τελευταία χρόνια υπήρξε απίστευτη επιτυχία στην εφαρμογή των Recurrent Neural δικτύων σε διάφορα προβλήματα: αναγνώριση φωνής, μοντελοποίηση γλώσσας, μετάφραση, υπότιτλος εικόνας...η λίστα συνεχίζεται.

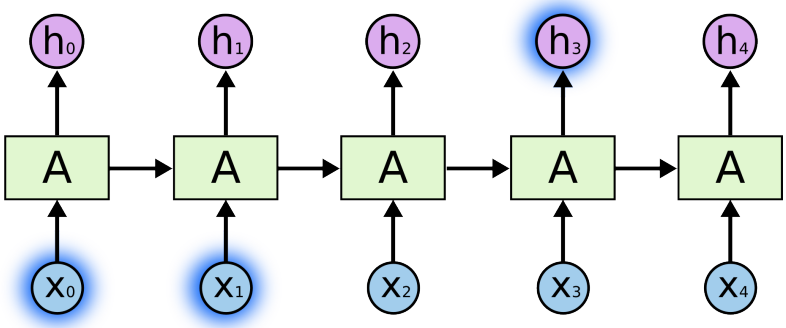
Απαραίτητη για αυτές τις επιτυχίες είναι η χρήση των LSTMs, ένα πολύ ιδιαίτερο είδος Recurrent Neural δικτύου το οποίο λειτουργεί, για πολλές εργασίες, πολύ καλύτερα από την κανονική έκδοση. Σχεδόν όλα τα συναρπαστικά αποτελέσματα που βασίζονται σε Recurrent Neural δίκτυα επιτυγχάνονται με τα LSTMs.

2.4.1 The Problem of Long-Term Dependencies

Μία από τις προσεγγίσεις των RNNs είναι η ιδέα ότι μπορούν να συνδέσουν προηγούμενη πληροφορία με την παρούσα εργασία, όπως η χρήση προηγούμενων video frames θα μπορούσε να βοηθήσει στην κατανόηση των παρόντων video frames. Αν λοιπόν τα RNNs θα μπορούσαν να το κάνουν αυτό θα ήταν εξαιρετικά χρήσιμο. Αλλά μπορούν; Εξαρτάται.

Μερικές φορές χρειάζεται απλά να εξετάσουμε μόνο τις πρόσφατες πληροφορίες για την εκτέλεση της παρούσας εργασίας. Για παράδειγμα, έστω ότι ένα μοντέλο γλώσσας προσπαθεί να προβλέψει την επόμενη λέξη με βάση τις προηγούμενες. Αν προσπαθούμε να προβλέψουμε την τελευταία λέξη στο "the clouds are in the sky," δεν χρειαζόμαστε κανένα άλλο περιεχόμενο - είναι αρκετά προφανές ότι η επόμενη λέξη θα είναι "sky".

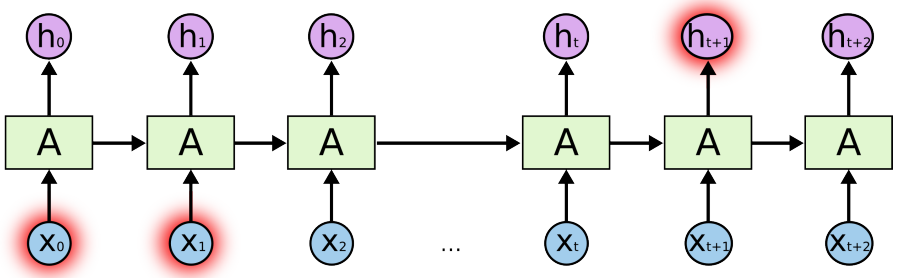
Σε τέτοιες περιπτώσεις, όπου το χάσμα μεταξύ των σχετικών πληροφοριών και του χώρου που χρειάζεται είναι μικρός, τα RNNs μπορούν να μάθουν να χρησιμοποιούν προηγούμενες πληροφορίες.



Σχήμα 2.3: *(Olah, 2015).*

Υπάρχουν όμως και περιπτώσεις όπου χρειαζόμαστε περισσότερο περιεχόμενο (context). Έστω ότι θέλουμε να προβλέψουμε την τελευταία λέξη στο κείμενο "I grew up in France... I speak fluent *French*". Οι πρόσφατες πληροφορίες δείχνουν ότι η επόμενη λέξη είναι πιθανώς το όνομα μιας γλώσσας, αλλά αν θέλουμε να περιορίσουμε ποια γλώσσα, χρειαζόμαστε το περιεχόμενο της Γαλλίας, που βρίσκεται πολύ πίσω. Είναι πολύ πιθανό λοιπόν το χάσμα ανάμεσα στην σχετική πληροφορία και το σημείο όπου αυτή χρειάζεται να είναι πολύ μεγάλο.

Δυστυχώς όσο το χάσμα αυτό μεγαλώνει, γίνεται όλο και πιο δύσκολο για τα RNNs να μάθουν να συνδέουν την πληροφορία.



Σχήμα 2.4: *(Olah, 2015).*

Θεωρητικά τα RNNs είναι απολύτως σε θέση να χειριστούν τέτοιες "μακροχρόνιες εξαρτήσεις". Ένας άνθρωπος θα μπορούσε να επιλέξει προσεχτικά παραμέτρους γι'αυτά ώστε να λυθούν προβλήματα αυτής της φύσης. Δυστυχώς όμως στην πράξη τα RNNs δε φαίνεται να είναι σε θέση να τα μάθουν. Το πρόβλημα διερευνήθηκε σε βάθος από τους Hochreiter (Hochreiter, 1991) και Bengio, et al. (Bengio et al., 1994), ο οποίος βρήκε κάποιους θεμελειώδης λόγους για τους οποίους μπορεί να είναι δύσκολο.

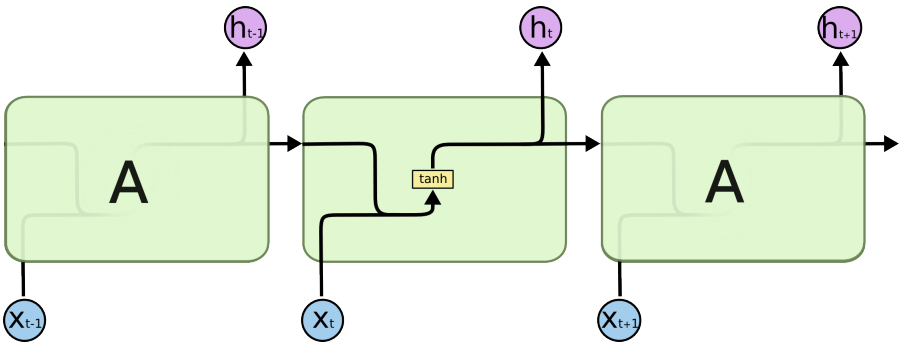
Ευτυχώς τα LSTM δεν έχουν αυτό το πρόβλημα!

2.5 LSTM Networks

Τα δίκτυα Long Short Term Memory - συνήθως LSTM - είναι ένα ειδικό είδος RNN, ικανό να μαθαίνει μακροπρόθεσμες εξαρτήσεις. Εισήχθησαν από τους Hochreiter & Schmidhuber (1997) (Hochreiter and Schmidhuber, 1997) και βελτιώθηκαν και χρησιμοποιήθηκαν από πολλούς ανθρώπους σε εργασίες που ακολούθησαν. Δουλεύουν αρκετά καλά σε μια μεγάλη ποικιλία προβλημάτων και χρησιμοποιούνται ευρέως.

Τα LSTM έχουν σχεδιαστεί ειδικά για να αποφευχθεί το πρόβλημα της μακροχρόνιας εξάρτησης. Το να θυμούνται πληροφορίες για μεγάλες χρονικές περιόδους είναι πρακτικά η προεπιλεγμένη συμπεριφορά τους, όχι κάτι που αγωνίζονται να μάθουν!

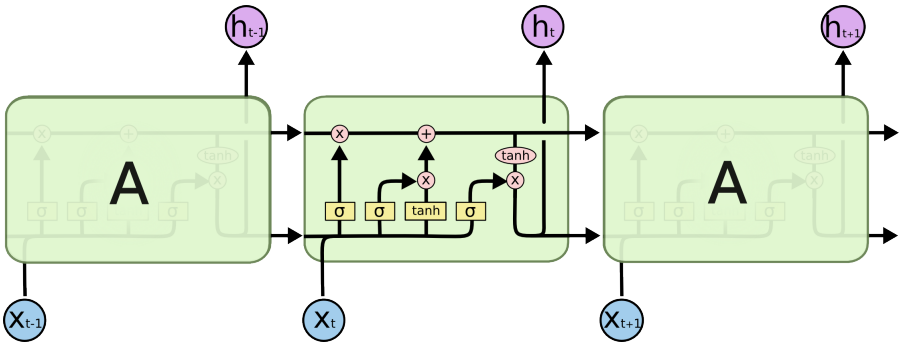
Όλα τα recurrent neural networks έχουν τη μορφή μιας αλυσίδας επαναλαμβανόμενων ενοτήτων του νευρωνικού δικτύου. Στα πρότυπα RNNs αυτή η επαναλαμβανόμενη ενότητα θα έχει μια πολύ απλή δομή, όπως ένα tanh layer μόνο.



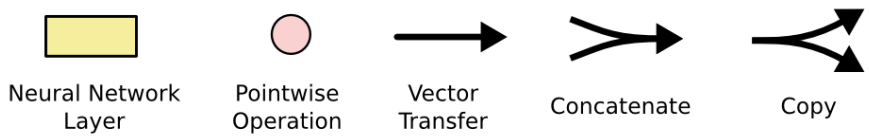
Σχήμα 2.5: *The repeating module in a standard RNN contains a single layer.(Olah, 2015).*

Τα LSTM έχουν επίσης αυτή τη δομή σαν αλυσίδα, αλλά η επαναλαμβανόμενη

ενότητα έχει διαφορετική δομή. Αντί να έχουμε ένα μόνο neural network layer, υπάρχουν τέσσερα, που αλληλεπιδρούν με ένα πολύ ιδιαίτερο τρόπο.



Σχήμα 2.6: *The repeating module in a LSTM contains four interacting layers.(Olah, 2015).*

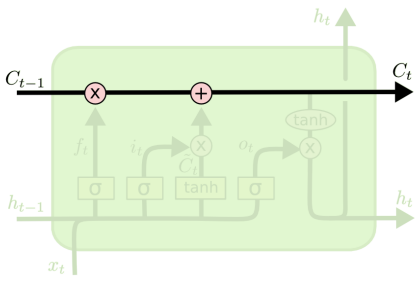


Σχήμα 2.7: *(Olah, 2015).*

Στο παραπάνω διάγραμμα, κάθε γραμμή μεταφέρει ένα ολόκληρο διάνυσμα, από την έξοδο ενός κόμβου στις εισόδους των άλλων κόμβων. Οι ροζ κύκλοι αντιπροσωπεύουν σημειακές λειτουργίες, όπως την προσθήκη διανυσμάτων, ενώ τα κίτρινα κουτιά είναι εκπαιδευμένα neural network layers. Οι γραμμές που συγχωνεύονται υποδηλώνουν τη συνένωση, ενώ μια γραμμή που διακλαδώνεται υποδηλώνει ότι το περιεχόμενο του αντιγράφεται και τα αντίγραφά του μεταφέρονται σε διαφορετικές τοποθεσίες.

2.5.1 Η βασική Ιδέα πίσω από τα LSTM

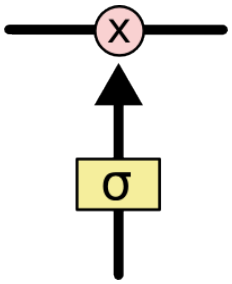
Το κλειδί για τα LSTMs είναι το cell state, η οριζόντια γραμμή που διατρέχει το πάνω μέρος του διαγράμματος. Το cell state είναι κάπως σα μεταφορικός ιμάντας. Διατρέχει κατευθείαν ολόκληρη την αλυσίδα, με μερικές μόνο μικρές γραμμικές αλληλεπιδράσεις. Είναι πολύ εύκολο να μεταφέρεται η πληροφορία κατά μήκος αμετάβλητη.



Σχήμα 2.8: *(Olah, 2015).*

Το LSTM έχει την ικανότητα να αφαιρεί ή να προσθέτει πληροφορία στο cell state, προσεκτικά ρυθμισμένο από δομές που ονομάζονται πύλες-gates.

Τα gates είναι ένας τρόπος να αφαιθεί πληροφορία μέσω αυτών. Αποτελούνται από ένα sigmoid neural net layer και μια σημειακή λειτουργία πολλαπλασιασμού.



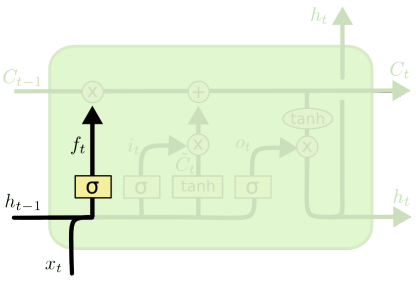
Σχήμα 2.9: *(Olah, 2015).*

Οι έξοδοι από το sigmoid layer εξάγει αριθμούς μεταξύ του 0 και του 1, περιγράφοντας το πόσο από το κάθε component πρέπει να περάσει. Μια τιμή 0 σημαίνει "να μην αφήσετε τίποτα να περάσει", ενώ η τιμή 1 σημαίνει "αφήστε τα πάντα να περάσουν". Ένα LSTM έχει τρεις από αυτές τις πύλες-gates, για να προστατεύει και να ελέγχει τα cell states.

2.5.2 Βήμα-Βήμα τα LSTM

Το πρώτο βήμα στα LSTM είναι να αποφασίσουμε ποια πληροφορία θα πετάξουμε από τα cell states. Αυτή η απόφαση θα παρθεί από ένα sigmoid layer το οποίο ονομάζεται "forget gate layer". Εξετάζει τα h_{t-1} και x_t και έχει σαν έξοδο έναν αριθμό μεταξύ 0 και 1 για κάθε αριθμό στο cell state C_{t-1}. Το 1 αντιπροσωπεύει "κρατήστε αυτή την πληροφορία", ενώ το 0 αντιπροσωπεύει "πετάξτε αυτή την πληροφορία".

Έστω λοιπόν το παράδειγμα του γλωσσικού μοντέλου το οποίο προσπαθεί να προβλέψει την επόμενη λέξη με βάση όλες τις προηγούμενες. Σε ένα τέτοιο πρόβλημα, cell state μπορεί να περιλαμβάνει το φύλο του παρόντος θέματος, έτσι ώστε να μπορούν να χρησιμοποιηθούν οι σωστές αντωνυμίες. Όταν βλέπουμε ένα καινούριο θέμα, θέλουμε να ξεχάσουμε το φύλο του προηγούμενου θέματος.

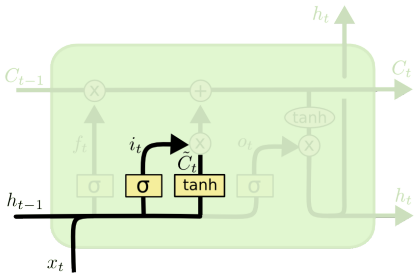


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Σχήμα 2.10: *(Olah, 2015).*

Το επόμενο βήμα είναι να αποφασίσουμε ποιες νέες πληροφορίες θα πρέπει να αποθηκεύσουμε στο cell state. Αυτό έχει δύο μέρη. Κατ'αρχάς ένα sigmoid layer το οποίο ονομάζεται "input gate layer" αποφασίζει ποιες τιμές θα ενημερώσουμε. Στη συνέχεια ένα tanh layer δημιουργεί ένα διάνυσμα πιθανών τιμών, $\tilde{C}_t$, το οποίο θα μπορούσε να προστεθεί στο state. Στο επόμενο βήμα θα συνδυάσουμε αυτά τα δύο, για να δημιουργήσουμε μια ενημέρωση για το state.

Έτσι λοιπόν για το παράδειγμα του γλωσσικού μοντέλου, θα θέλαμε να προσθέσουμε το φύλο του νέου θέματος στο cell state, για να αντικαταστήσουμε το παλιό που ξεχνάμε.

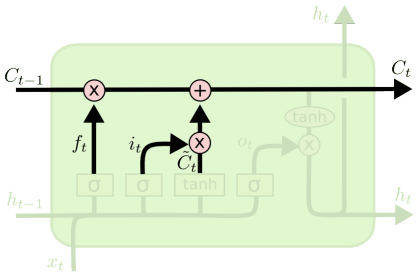


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Σχήμα 2.11: *(Olah, 2015).*

Είναι πλέον καιρός να ενημερώσουμε το παλιό cell state, C_{t-1} στο νέο cell state C_t. Τα προηγούμενα βήματα έχουν ήδη αποφασίσει τι πρέπει να κάνουμε, το μόνο που μας μένει είναι να το κάνουμε. Πολλαπλασιάζουμε την παλιά κατάσταση επί f_t, αφήνοντας αυτά τα πράγματα που αποφασίσαμε να αφήσουμε νωρίτερα. Μετά προσθέτουμε $i_t * \tilde{C}_t$. Αυτές είναι οι νέες υποψήφιες τιμές, κλιμακούμενες από το πόσο αποφασίσαμε να ενημερώσουμε την κάθε κατάσταση.

Στο παράδειγμα του γλωσσικού μοντέλου, αυτό το σημείο είναι που έχουμε αφήσει τις πληροφορίες σχετικά με το φύλο του παλιού υποκειμένου και προσθέτουμε νέες πληροφορίες, όπως αποφασίσαμε στα προηγούμενα βήματα.

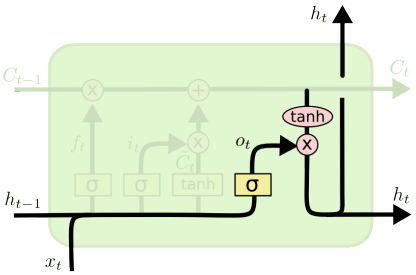


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Σχήμα 2.12: *(Olah, 2015).*

Τελικά, πρέπει να αποφασίσουμε τι θα έχουμε για έξοδο. Αυτή η έξοδος θα βασίζεται στο cell state, αλλά θα είναι μια φιλτραρισμένη έκδοση. Αρχικά, έχουμε sigmoid layer το οποίο αποφασίζει ποια μέρη του cell state θα βγάλουμε στην έξοδο. Στη συνέχεια, περνάμε το cell state μέσα από tanh (για να ωθήσουμε τις τιμές ανάμεσα σε -1 και 1) και το πολλαπλασιάζουμε με την έξοδο από τη sigmoid gate, έτσι ώστε να εξάγουμε τα μέρη που αποφασίσαμε.

Άρα για το παράδειγμα του γλωσσικού μοντέλου, δεδομένου ότι είδε ένα θέμα, μπορεί να θέλει να παράγει πληροφορίες σχετικά με ένα ρήμα, σε περίπτωση που αυτό έρχεται μετά. Για παράδειγμα μπορεί να θέλει να εξάγει αν ένα υποκείμενο είναι στον ενικό ή στον πληθυντικό, έτσι ώστε να γνωρίζουμε σε ποια μορφή θα πρέπει το ρήμα να κλιθεί αν αυτό ακολουθεί.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Σχήμα 2.13: *(Olah, 2015).*

Τα LSTMs είναι ένα μεγάλο βήμα σε αυτά που μπορούμε να επιτύχουμε με τα RNNs.

Στην παρούσα διπλωματική εργασία θα γίνει χρήση των LSTMs.

2.6 Επιπρόσθετα layers και Activation Συναρτήσεις

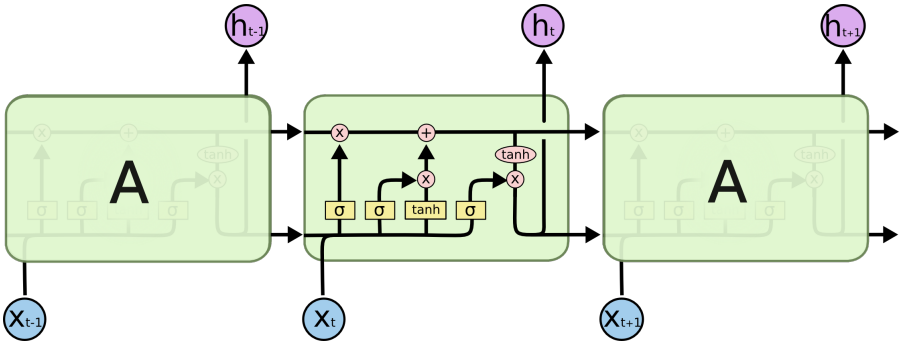
Στην ενότητα αυτή αρχικά θα αναφέρω layers που χρησιμοποιήθηκαν για την εκπαίδευση με LSTM. Επιπλέον θα αναφέρω και τις activation συναρτήσεις που έχω χρησιμοποιήσει συνολικά στη συγκεκριμένη εργασία.

Όπως αναφέρθηκε στην προηγούμενη ενότητα, τα μοντέλα LSTM είναι ένας τύπος από recurrent neural networks τα οποία έχουν τη δυνατότητα να μαθαίνουν ακολουθίες παρατηρήσεων. Αυτό τα κάνει ένα δίκτυο κατάλληλο για προβλέψεις χρονοσειρών (time series). Ένα θέμα που προκύπτει με τα LSTM είναι ότι μπορούν εύκολα να κάνουν overfit τα δεδομένα εκπαίδευσης, μειώνοντας έτσι την προγνωστική τους ικανότητα.

Το **Dropout** είναι μια μέθοδος regularization όπου βοηθά στο να μειώνει σημαντικά την *υπερπροσαρμογή-(overfitting)*[1] αγνοώντας τυχαία επιλεγμένους νευρώνες κατά τη διάρκεια της εκπαίδευσης και συνεπώς μειώνει την "ευαισθησία" προς τα συγκεκριμένα βάρη των μεμονωμένων νευρώνων.

Στην προκειμένη περίπτωση, αντί να χρησιμοποιήσουμε μεμονώμενα dropout layers, χρησιμοποιούμε το dropout μέσα στα LSTM cells. Άμα είχαν χρησιμοποιηθεί ξεχωριστά dropout layers τότε θα επηρεάζονταν μόνο η τελική έξοδος του LSTM, ενώ στη δική μας περίπτωση εφαρμόζεται το dropout σε 4 διαφορετικές sub neural network λειτουργίες μέσα στο LSTM.

[1] Η *υπερπροσαρμογή-(overfitting)* αναφέρεται σε ένα μοντέλο που αποδίδει πολύ καλά τα δεδομένα εκπαίδευσης αλλά δε γενικεύει καλά. Η υπερπροσαρμογή συμβαίνει όταν ένα μοντέλο μαθαίνει να μοντελοποιεί την λεπτομέρεια και το θόρυβο στα δεδομένα εκπαίδευσης, σε βαθμό που επηρεάζει αρνητικά την απόδοση του μοντέλου σε νέα δεδομένα. Αυτό σημαίνει ότι ο θόρυβος ή οι τυχαίες διακυμάνσεις στα δεδομένα εκπαίδευσης (training data) συλλέγονται και εκπαιδεύονται ως έννοιες από το μοντέλο. Το πρόβλημα είναι ότι αυτές οι έννοιες δεν ισχύουν για νέα δεδομένα και επηρεάζουν αρνητικά την ικανότητα των μοντέλων να γενικεύουν. Η υπερπροσαρμογή είναι πιθανότερη με μη-παραμετρικά και μη-γραμμικά μοντέλα που έχουν μεγαλύτερη ευελιξία και ελευθερία στην οικοδόμηση του μοντέλου με βάση το σύνολο δεδομένων και συνεπώς μπορούν πραγματικά να χτίσουν μη ρεαλιστικά μοντέλα. Ως εκ τούτου, πολλοί μη-παραμετρικοί αλγόριθμοι μηχανικής μάθησης περιλαμβάνουν επίσης παραμέτρους ή τεχνικές για τον περιορισμό του πόση λεπτομέρεια μαθαίνει το μοντέλο.



Στην παραπάνω εικόνα φαίνονται στα κίτρινα κουτιά οι 4 fully connected λειτουργίες του δικτύου (η καθεμία με το δικό της βάρος), που συμβαίνουν κάτω από το LSTM.

Συγκεκριμένα στα LSTM έχουμε δύο είδη dropout παραμέτρων, το dropout, που εφαρμόζεται στα ninputs (τα οποία μετατρέπονται σε συμβατή διάσταση) και το recurrent_dropout, το οποίο εφαρμόζεται στα recurrent inputs (προηγούμενα outputs και/ή states). Άρα, οι είσοδοι και τα recurrent connections στα LSTM units αποκλείονται πιθανοτικά από τα activations και τις ενημερώσεις των βαρών κατά την εκπαίδευση του δικτύου. Αυτό όπως είπαμε έχει ως αποτέλεσμα τη βελτίωση της εκπαίδευσης.

Αφότου τα LSTM layers έκαναν όλη τη δουλειά για να μετατρέψουν την είσοδο ώστε να είναι δυνατές οι προβλέψεις για την επιθυμητή έξοδο πρέπει να μειώσουμε (ή σε σπάνιες περιπτώσεις να επεκτείνουμε) το σχήμα ώστε να ταιριάζει με την επιθυμητή έξοδο. Στην περίπτωση μας, καθώς η ανάλυση των logs είναι multi-class [2] πρόβλημα, θα έχουμε τόσες εξόδους όσες και το συνολικό πλήθος του λεξιλογίου από το οποίο αποτελούνται τα logs. Αυτή η διαδικασία γίνεται στο **Dense Layer**. Σύμφωνα και με την περιγραφή του Keras, το layer αυτό είναι η εφαρμογή της εξίσωσης *output = activation(dot(input,kernel)+bias)*. Αυτό σημαίνει ότι λαμβάνουμε το γινόμενο ανάμεσα στον τένσορα εισόδου και σε οτιδήποτε εμφανίζει ο πυρήνας του πίνακα των βαρών (weight kernel matrix) στο dense layer. Μετά μπορούμε να προσθέσουμε το διάνυσμα πόλωσης (bias vector) (αν θέλουμε) και λαμβάνουμε μία element-wise activation των τιμών εξόδου.

Η πιο σημαντική παράμετρος που λαμβάνει αυτό το layer είναι η παράμετρος *units*, ένας θετικός αριθμός που υποδηλώνει το μέγεθος της εξόδου του. Η παράμετρος αυτή στην ουσία υπαγορεύει το μέγεθος του πίνακα βαρών (weight matrix) και του διανύσματος πόλωσης (το διάνυσμα πόλωσης θα έχει το ίδιο μέγεθος, αλλά ο πίνακας βαρών θα υπολογιστεί με βάση το μέγεθος των δεδομένων εισόδου, έτσι ώστε το γινόμενο να παράγει δεδομένα που έχουν μέγεθος εξόδου τα units.

Η επόμενη παράμετρος που θα αναφέρω είναι τα **Activations**. Τα Activations μπορούν να χρησιμοποιηθούν είτε μέσω ενός activation layer, είτε μέσω μιας activation παραμέτρου. Στην περίπτωση μας θα περιλαμβάνεται μέσα στο dense layer.

Αν δεν εφαρμοστεί μια συνάρτηση Ενεργοποίησης, τότε το σήμα εξόδου θα ήταν απλά μια απλή γραμμική συνάρτηση, η οποία είναι απλώς ένα πολυώνυμο ενός βαθμού. Μια γραμμική εξίσωση είναι εύκολο να λυθεί, αλλά είναι περιορισμένη στην πολυπλοκότητά της και έχει λιγότερη ισχύ για να μάθει σύνθετες λειτουργικές αντιστοιχίσεις από τα δεδομένα. Ένα επίπεδο νευρωνικού δικτύου χωρίς συνάρτηση ενεργοποίησης θα ήταν απλώς ένα μοντέλο γραμμικής παλινδρόμησης (Linear

[2] Κάθε δείγμα μπορεί να ανήκει σε μία από τις C κλάσεις. Το LSTM θα έχει c νευρώνες στην έξοδο του. Η (ground-truth) θα είναι ένα διάνυσμα one-hot με μία θετική κλάση και $c-1$ αρνητικές κλάσεις.

regression Model), το οποίο έχει περιορισμένη ισχύ και δεν αποδίδει καλά τις περισσότερες φορές. Επιπλέον αυτό που χρειάζεται είναι το Νευρωνικό Δίκτυο όχι μόνο να μαθαίνει και να υπολογίζει μια γραμμική λειτουργία, αλλά κάτι πιο περίπλοκο από αυτό.

Επίσης, χωρίς την activation συνάρτηση, το νευρωνικό μας δίκτυο δε θα είναι σε θέση να μάθει και να μοντελοποιήσει άλλα περίπλοκα είδη δεδομένων, όπως εικόνες, βίντεο, ήχο, ομιλία κλπ. Αυτός είναι και ο λόγος για τον οποίο χρησιμοποιούνται τεχνικές Τεχνητού Νευρωνικού Δικτύου, όπως η Βαθιά Μάθηση, για να κατανοηθούν πιο πολύπλοκα, μεγάλης διαστάσεως, μη γραμμικά μεγάλα σύνολα δεδομένων, όπου το μοντέλο έχει πολλά κρυφά επίπεδα μεταξύ τους και έχει μια πολύ περίπλοκη αρχιτεκτονική που βοηθά στο να κατανοηθούν και να εξαχθούν γνώσεις από τέτοια πολύπλοκα μεγάλα σύνολα δεδομένων.

Οπότε γιατί είναι αναγκαίες οι μη-γραμμικότητες; Οι μη γραμμικές λειτουργίες είναι αυτές που έχουν βαθμό περισσότερο από ένα και έχουν καμπυλότητα όταν σχεδιάζεται μια μη γραμμική συνάρτηση. Τα νευρωνικά δίκτυα θεωρούνται **Προσεγγιστές Καθολικής Συνάρτησης (Universal Function Approximators)**. Αυτό σημαίνει, ότι μπορούν να υπολογίσουν και να μάθουν οποιαδήποτε λειτουργία. Σχεδόν κάθε διαδικασία που μπορεί κάποιος να σκεφτεί μπορεί να εκπροσωπείται ως λειτουργικός υπολογισμός στα Νευρωνικά Δίκτυα. Για το λόγο αυτό, πρέπει να εφαρμοστεί μια συνάρτηση ενεργοποίησης f(x) ώστε να καταστεί το δίκτυο πιο ισχυρό και να προστεθεί η ικανότητα να μάθει κάτι περίπλοκο από τα δεδομένα και να αντιπροσωπεύει μη-γραμμικές σύνθετες αυθαίρετες λειτουργικές αντιστοιχίσεις μεταξύ εισόδων και εξόδων. Ως εκ τούτου, χρησιμοποιώντας μη-γραμμική ενεργοποίηση, είναι δυνατό να παράγονται μη-γραμμικές απεικονίσεις από τις εισόδους στις εξόδους.

Το ποια συνάρτηση ενεργοποίησης θα χρησιμοποιήσουμε εξαρτάται από το πρόβλημα που έχουμε κάθε φορά. Για την εκπαίδευση του μοντέλου Skipgram χρησιμοποιείται η συνάρτηση ενεργοποίησης **sigmoid**, ενώ για την ανάλυση των logs στην πρώτη φάση, που όπως αναφέρθηκε και παραπάνω είναι ένα multi-class πρόβλημα, χρησιμοποιείται η συνάρτηση ενεργοποίησης **softmax**. Ακόμη για τη δεύτερη φάση η οποία είναι εκπαίδευση σε multivariate time series χρησιμοποιείται η συνάρτηση ενεργοποίησης **elu**. Οι συναρτήσεις αυτές θα περιγραφούν στην επόμενη υποενότητα 2.6.1.

2.6.1 Softmax-Sigmoid-ELU

- **Softmax**

  Το τελευταίο layer του νευρωνικού δικτύου, χωρίς την activation συνάρτηση, είναι αυτό που ονομάζουμε "logits layer". Απλώς παρέχει τις τελικές εξόδους για το νευρωνικό δίκτυο. Στην περίπτωση μιας multi-class ταξινόμησης, με n κλάσεις, θα είναι n νευρώνες - και ως εκ τούτου n έξοδοι. Βέβαια αυτοί οι αριθμοί "logits" δε μας παρέχουν κάποια χρήσιμη πληροφορία.

  Κατά κάποιον τρόπο, ωστόσο, η πρόβλεψη σε ποια κατηγορία στόχου (target class) ανήκει κάποια είσοδος σχετίζεται με μια κατανομή πιθανότητας. Για το παραπάνω, εαν γνωρίζουμε τις πιθανότητες μιας τιμής να ανήκει σε ένα από τα n πιθανά αποτελέσματα, θα μπορούσαμε απλά να πάρουμε το argmax αυτών των διακριτών πιθανοτήτων και να βρούμε το αποτέλεσμα της κλάσης. Επομένως, αρκεί να μετατρέψουμε τα παραπάνω logits σε μια κατανομή πιθανότητας. Η κατάλληλη κατανομή πιθανότητας είναι η *διακριτή κατανομή πιθανότητας*

(discrete probability distribution), η οποία μπορεί να πάρει έναν μετρήσιμο αριθμό τιμών, δηλαδή μια πιθανότητα για κάθε τιμή.

Η συνάρτηση **softmax** μας επιτρέπει να εκφράσουμε τις εισόδους μας ως διακριτή κατανομή πιθανότητας. Μαθηματικά αυτό ορίζεται ως:

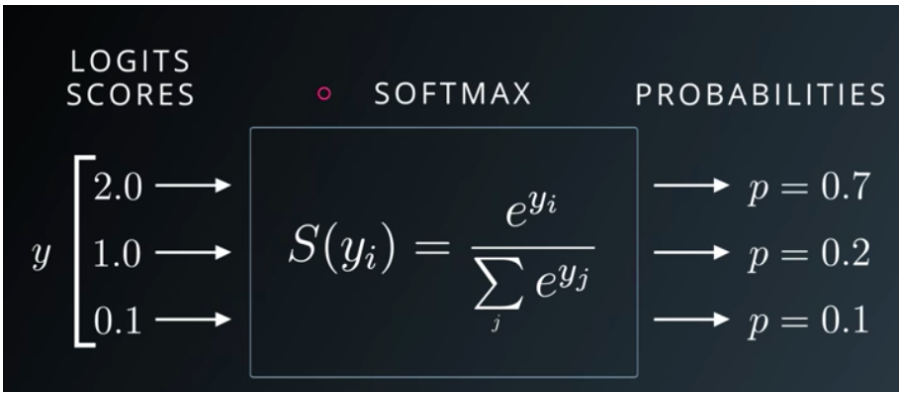
$$Softmax(x_i) = \frac{\exp(x_i)}{\sum_{j=1} exp(x_j)} \tag{2.1}$$

Διαισθητικά αυτό μπορεί να μεταφραστεί ως εξής: Για κάθε τιμή (πχ είσοδος) στο διάνυσμα εισόδου μας, η τιμή Softmax είναι ο εκθέτης της μεμονομένης εισόδου διαιρούμενης από ένα άθροισμα των εκθετών όλων των εισόδων.

Αυτό εξασφαλίζει ότι συμβαίνουν πολλά πράγματα:

- Οι αρνητικές είσοδοι θα μετατραπούν σε μη αρνητικές τιμές, χάρη στην εκθετική συνάρτηση.
- Κάθε είσοδος θα βρίσκεται στο διάστημα (0,1).
- Καθώς ο παρανομαστής σε κάθε υπολογισμό Softmax είναι ο ίδιος, οι τιμές γίνονται ανάλογες μεταξύ τους, πράγμα που εξασφαλίζει ότι αθροίζονται μαζί στο 1.

Άρα συνοψίζοντας, η συνάρτηση Softmax μετατρέπει τα logits σε πιθανότητες οι οποίες αθροίζονται στο 1. Οι συναρτήσεις Softmax έχουν ως έξοδο ένα διάνυσμα το οποίο αντιπροσωπεύει κατανομές πιθανοτήτων μιας λίστας πιθανών αποτελεσμάτων.



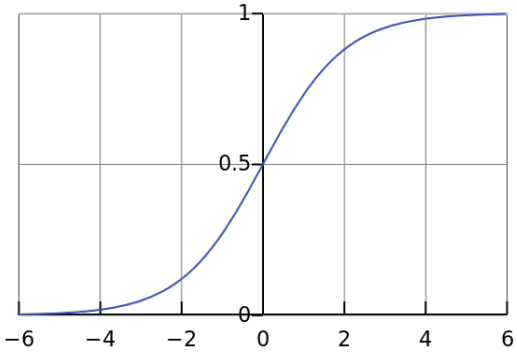
Σχήμα 2.14: *Udacity Deep Learning Slide on Softmax.*

Η παραπάνω εικόνα μας δείχνει ότι η συνάρτηση Softmax μετατρέπει τα logits [2.0, 1.0, 0.1] σε πιθανότητες [0.7, 0.2, 0.1] και οι πιθανότητες αθροίζονται στο 1.

- **Sigmoid**

  Η Σιγμοειδής συνάρτηση ενεργοποίησης έχει την ακόλουθη μαθηματική εξίσωση και φαίνεται παρακάτω.

$$A = \frac{1}{1 + e^{-x}}$$



Σχήμα 2.15: *Σιγμωειδής Συνάρτηση (Sharma, 2017).*

Ο κύριος λόγος που χρησιμοποιούμε τη sigmoid συνάρτηση είναι γιατί υπάρχει ανάμεσα στο (0 ως το 1). Επομένως, χρησιμοποιείται ειδικά για μοντέλα όπου θα πρέπει να προβλέψουμε την πιθανότητα ως έξοδο. Δεδομένου ότι η πιθανότητα από οτιδήποτε υπάρχει μόνο μεταξύ του εύρους 0 και 1, η sigmoid είναι η σωστή επιλογή.

Έχει ομαλή κλίση (smooth gradient) αποτρέποντας τα "jumps" στις τιμές εξόδου. Οι τιμές εξόδου δεσμεύονται όπως είπαμε ανάμεσα στο 0 και το 1, κανονικοποιώντας την έξοδο κάθε νευρώνα. Οι προβλέψεις λοιπόν με τη συνάρτηση αυτή είναι σαφείς - για X πάνω από το 2 πχ ή κάτω από το -2, τείνει να φέρει τη Y τιμή (την πρόβλεψη) στην άκρη της καμπύλης, πολύ κοντά στο 1 ή το 0.

Βέβαια ένα βασικό της μειονέντημα είναι το φαίνομενο του *Vanishing gradient* - για πολύ υψηλές ή χαμηλές τιμές του X, δεν υπάρχει σχεδόν καμία αλλαγή στην πρόβλεψη προκαλώντας ένα πρόβλημα vanishing gradient [3]. Αυτό μπορεί να οδηγήσει στο δίκτυο να αρνηθεί να μάθει παραπάνω ή να είναι πολύ αργό για να επιτύχει μια ακριβή πρόβλεψη. Επίσης είναι υπολογιστικά δαπανηρή επιλογή.

- **Exponential Linear Unit - ELU**

  Αυτή η activation function επιδιορθώνει ορισμένα από τα προβλήματα με τη ReLU και διατηρεί μερικά από τα θετικά πράγματα. Για αυτή τη συνάρτηση ενεργοποίησης μία alpha α τιμή λαμβάνεται, μια συνηθισμένη τιμή είναι ανάμεσα στο 0.1 και 0.3.

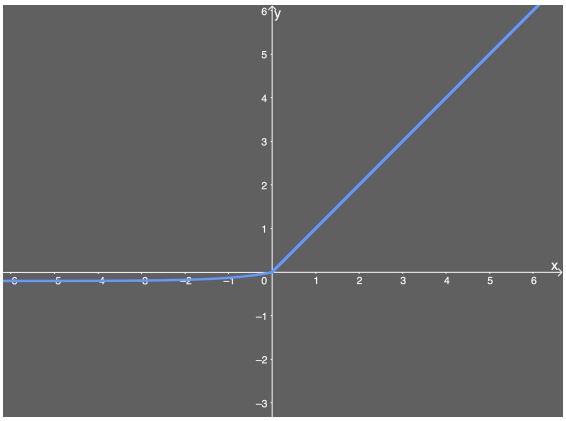
$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases}$$

  Αν εισάγουμε μια τιμή x μεγαλύτερη από το 0, τότε είναι η ίδια με τη ReLU - το αποτέλεσμα θα είναι μια τιμή y ίση με τη τιμή x. Αλλά με την ELU, αν η τιμή εισόδου x είναι μικρότερη από το 0, έχουμε μια τιμή ελαφρώς πιο κάτω από το μηδέν.

[3] Ο αριθμός των weights ή biases λαμβάνουν ουσιαστικά πολύ μικρές ενημερώσεις. Αυτό αναγκάζει τους κόμβους του δικτύου να απέχουν πολύ από τη βέλτιστη τιμή τους, εμποδίζοντας εν τέλει το νευρωνικό δίκτυο από τη μάθηση. Έχει παρατηρηθεί ότι γίνεται ακόμα μεγαλύτερο το πρόβλημα, αν υπάρχουν διαφορετικά layers που να μαθαίνουν σε διαφορετικές ταχύτητες. Τα layers θα μαθαίνουν με διαφορετικές ταχύτητες και το πρώτο θα είναι πάντα χειρότερο όσον αφορά το ρυθμό μάθησής του.

Η τιμή-y που λαμβάνεται εξαρτάται τόσο από την είσοδο της τιμής x, όσο και από την alpha α παράμετρο, η οποία μπορεί να προσαρμοστεί ανάλογα με τις ανάγκες μας κάθε φορά. Επιπλέον, εισάγουμε μια εκθετική λειτουργία e^x, πράγμα που σημαίνει ότι η ELU είναι υπολογιστικά πιο ακριβή απ'ότι η ReLU.(HANSEN)

Η ELU απεικονίζεται παρακάτω με a=0.2.



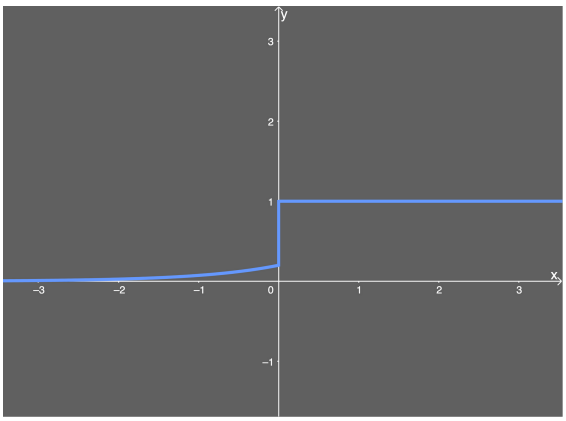
Σχήμα 2.16: *Το διάγραμμα της ELU συνάρτησης ενεργοποίησης(HANSEN).*

Αλλά τι γίνεται με την παράγωγο της ELU?

Πριν αναφερθώ για την παράγωγο, να σημειώσω ότι είναι πολύ σημαντική, γιατί καθώς ενημερώνεται η καμπύλη, για να ξέρουμε σε ποια κατεύθυνση και πόσο πρέπει να αλλάξουμε ή να ενημερώσουμε την καμπύλη εξαρτάται από την κλίση. Γι'αυτό και είναι τόσο σημαντική. Η εξίσωση της είναι η παρακάτω:

$$\mathrm{ELU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ \mathrm{ELU}(x) + \alpha & \text{if } x \leq 0 \end{cases}$$

Η y - τιμή είναι 1 αν το x είναι μεγαλύτερο από το 0. Αν πάλι η τιμή x είναι μικρότερη από το μηδέν, τότε η έξοδος είναι η ELU συνάρτηση, συν την τιμή α. Η γραφική παράσταση γι'αυτό είναι η παρακάτω:



Σχήμα 2.17: *Το διάγραμμα της παραγώγου της ELU συνάρτησης ενεργοποίησης(HANSEN).*

Όπως ίσως έχετε παρατηρήσει, αποφεύγουμε το "dead relu" πρόβλημα, διατηρώντας παράλληλα κάποια από την υπολογιστική ταχύτητα που αποκτάται από τη ReLU συνάρτηση ενεργοποίησης.

Πλεονεκτήματα:

- Αποφεύγει το πρόβλημα του dead relu.
- Παράγει αρνητικές εξόδους, οι οποίες βοηθούν το δίκτυο να ωθεί τα βάρη (weights) και τις μεροληψίες (biases) προς τις σωστές κατευθύνσεις.
- Κατά τον υπολογισμό της gradient, δημιουργεί ενεργοποιήσεις αντί να τα αφήνει στο μηδέν.

Μειονεκτήματα:

- Εισάγει μεγαλύτερο χρόνο υπολογισμού, εξαιτίας της εκθετικής λειτουργίας που συμπεριλαμβάνεται.
- Δεν αποφεύγει το πρόβλημα του exploding gradient descent[4].
- Το νευρωνικό δίκτυο δε μαθαίνει την τιμή α.

2.7 Ανισορροπία Κλάσεων - Imbalance classes

Οι περισσότερες ταξινομήσεις στον πραγματικό κόσμο εμφανίζουν κάποιο επίπεδο **ανισορροπίας κλάσης (class imbalance)**, το οποίο σημαίνει ότι δεν αντιπροσωπεύεται κάθε κλάση με μια ισότιμη μερίδα του συνόλου δεδομένων. Είναι σημαντικό όμως να προσαρμοστούν σωστά οι μετρήσεις και οι μέθοδοι στους στόχους της κάθε έρευνας. Για παράδειγμα, έστω ότι υπάρχουν 2 κλάσεις - A και B. Η κλάση A είναι το 90% του συνόλου δεδομένων και η κλάση B είναι το άλλο 10%, αλλά το μεγαλύτερο ενδιαφέρον είναι για τον εντοπισμό περιπτώσεων κλάσης B. Είναι εύκολο να επιτευχθεί ακρίβεια 90%, απλά προβλέποντας την κλάση A κάθε φορά, αλλά αυτό δεν προσφέρει κάτι στον αρχικό στόχο της έρευνας. Αντ'αυτού, μια κατάλληλα βαθμονομημένη μέθοδος μπορεί να επιτύχει μια χαμηλότερη ακρίβεια, αλλά θα έχει πολύ υψηλότερο πραγματικό θετικό ρυθμό (true positive rate) (ή ανάκληση-recall).

Στην συγκεκριμένη εργασία αντιμετώπιζεται και εδώ το πρόβλημα της ανισορροπίας κλάσης. Συγκεκριμένα ο μεγαλύτερος αριθμός των φράσεων που συναντώνται στα logs είναι φράσεις safe ή unknown. Για την αντιμετώπιση αυτού του προβλήματος έγινε χρήση class weights - βαρών κλάσεων. Με τη μέθοδο αυτή εισάγονται διαφορετικά βάρη σε διαφορετικές κλάσεις. Συγκεκριμένα έγινε με την εντολή του sklearn.utils.class_weight.compute_class_weight('balanced', unique_classes2, y_ints2) (Pedregosa et al., 2011).

Στην ενότητα 2.9.2 γίνεται αναφορά σε μετρικές που είναι πιο αντικειμενικές όταν υπάρχει ανισορροπία κλάσεων.

[4]Τα βάρη εδώ "εκρήγνυνται", δηλαδή οι τιμές τους αυξάνονται γρήγορα. Ουσιαστικά για να έχουμε vanishing gradient θα έχουμε $0 < w < 1$ και exploding gradient όταν $w > 1$.

2.8 Transfer Learning

Η Επεξεργασία Φυσικής Γλώσσας είναι ένα ισχυρό εργαλείο, αλλά στον πραγματικό κόσμο συναντάμε συχνά tasks τα οποία έχουν έλλειψη δεδομένων και φτωχή γενίκευση των μοντέλων. Ειδικά το κείμενο είναι τόσο διαφορετικό, θορυβώδη και αδόμητο. Το **Transfer Learning** λύνει αυτό το πρόβλημα επιτρέποντας μας να πάρουμε ένα προ-εκπαιδευμένο (pre-trained) μοντέλο μιας εργασίας και να το χρησιμοποιήσουμε σε άλλες. Πρόκειται για μια πολύ δημοφιλή μέθοδο, ιδιαίτερα στο τομέα του Computer vision η οποία μας επιτρέπει να χτίζουμε ακριβή μοντέλα με "timesaving way" (Rawat and Wang, 2017). Ειδικότερα για τα NLP, αυτή η έννοια είναι μια στροφή από την χρήση των word embeddings, τα οποία έχουν χρησιμοποιηθεί σε πολλές εργασίες των NLP τα τελευταία χρόνια, στην χρήση πιο εξελιγμένων και αφηρημένων αναπαραστάσεων.

Με τη μέθοδο αυτή, αντί να αρχίσει η διαδικασία εκπαίδευσης από την αρχή, αρχίζει από μια προηγούμενη γνώση η οποία εκπαιδεύθηκε κατά τη λύση ενός διαφορετικού προβλήματος, έτσι αξιοποιούνται προηγούμενες γνώσεις και δε μαθαίνουμε κάτι καινούριο από την αρχή. Ένας πιο επίσημος ορισμός του Transfer Learning είναι ο παρακάτω:

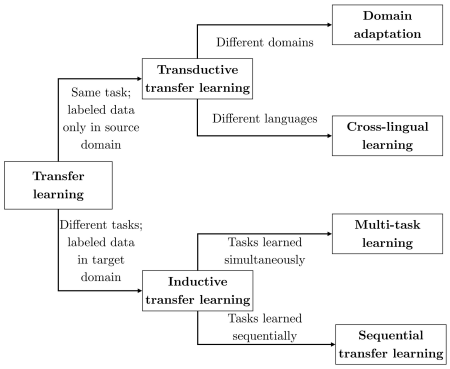
Έστω ότι έχουμε το Ds ένα source domain με το αντίστοιχο source task του, Ts, καθώς επίσης και ένα target domain Dt με το δικό του target task του, Tt. Ο στόχος του **Transfer Learning** είναι να μας επιτρέψει να μάθουμε την επιδιωκώμενη δεσμευμένη κατανομή πιθανότητας (target conditional probability distribution) $P(Yt|Xt)$ στο Dt με την πληροφορία που αποκτήθηκε από το Ds όπου $Ds \neq Dt$ ή $Ts \neq Tt$. Η μέθοδος αυτή θα μπορούσε να χωριστεί στους παρακάτω τύπους:

- Domain Adaptation
- Cross-lingual learning
- Multi-task learning
- Sequential transfer learning

Στην παρούσα διπλωματική ακολουθείται ο τελευταίος τύπος transfer learning, ο διαδοχικός-sequential.

2.8.1 Sequential Transfer Learning

Όπως υποδηλώνει και το όνομά του, το διαδοχικό transfer learning (STL) περιλαμβάνει τη μεταφορά γνώσης με μια ακολουθία βημάτων, όπου το source και το target task δεν είναι αναγκαστικά παρόμοια. Το STL αποτελείται από δύο στάδια. Στην πρώτη φάση της προεκπαίδευσης (pretraining), το μοντέλο εκπαιδεύεται στα δεδομένα του source και στη δεύτερη φάση της προσαρμογής (adaptation) το μοντέλο source εκπαιδεύεται για το target task.



Σχήμα 2.18: *Types of transfer learning.(Ruder, 2019).*

Το task της προεκπαίδευσης είναι συνήθως δαπανηρό, αλλά εκτελείται μόνο μια φορά. Το task της προσαρμογής είναι συνήθως πιο γρήγορο καθώς λειτουργεί σαν ένα βήμα fine-tuning. Το STL είναι συνήθως χρήσιμο σ'αυτά τα τρία σενάρια:

- Τα δεδομένα source και target task δεν είναι διαθέσιμα την ίδια στιγμή.
- Το source task έχει περισσότερα δεδομένα από το target task.
- Απαιτείται προσαρμογή σε πολλά target tasks.

Όσον αφορά την προεκπαίδευση, γενικότερα για να πάρουμε το μέγιστο όφελος θέλουμε ένα source training το οποίο θα ωφελήσει πολλά target tasks. Είναι δύσκολο να βρούμε τέτοιο task στην πράξη, αλλά είναι πάντα καλύτερα από το ξεκινάς από την αρχή. Σχετικά με την εκπαίδευση του source μπορούμε να την επιτύχουμε με τρεις τρόπους:

- Distant supervision
- Traditional supervision
- No supervision

Εμείς ακολουθούμε τον τρίτο τρόπο, της χωρίς επίβλεψης. Με τον τρόπο αυτό, που είναι γνωστός και ως language modelling, απαιτείται η πρόσβαση σε πολλά μη ετικετοποιημένα δεδομένα, για μας αυτά τα δεδομένα είναι τα logs. Σε σύγκριση με την εποπτευόμενη μάθηση, είναι πολύ πιο επεκτάσιμη προσέγγιση καθώς το κείμενο για οποιοδήποτε τομέα είναι εύκολα διαθέσιμο. Αυτή η προσέγγιση καλύπτει πολύ περισσότερες γενικές γνώσεις σχετικά με τη γλώσσα, σε σύγκριση με την εποπτευόμενη μάθηση που συλλαμβάνει μόνο εκείνα τα χαρακτηριστικά που απαιτούνται για το task.

Όσον αφορά για το δεύτερο βήμα του STL, την προσαρμογή, έχουμε δύο προσεγγίσεις για την χρήση προ-εκπαιδευμένου μοντέλου για το target task - feature extraction και fine-tuning. Η πρώτη χρησιμοποιεί αναπαραστάσεις ενός προ-εκπαιδευμένου μοντέλου και το τροφοδοτεί σε ένα άλλο μοντέλο, ενώ το fine-tuning περιλαμβάνει την εκπαίδευση του προ-εκπαιδευμένου μοντέλου πάνω στο target task.

- **Feature extraction(EX):** τα βάρη του μοντέλου παγώνουν και η έξοδος από αυτό αποστέλλεται απευθείας σε ένα άλλο μοντέλο. Τα χαρακτηριστικά

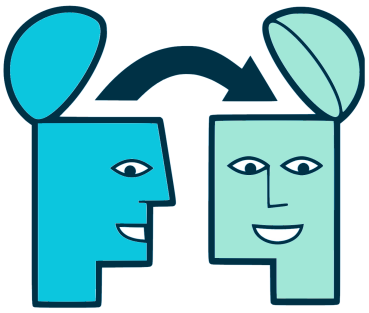
μπορούν να σταλούν είτε σε ένα fully connected model είτε μπορούμε απλά να εκπαιδεύσουμε ένα κλασσικό μοντέλο όπως Support Vector Machine ή Random Forest σε αυτά. Το όφελος χρήσης του είναι ότι το task-specific μοντέλο μπορεί να χρησιμοποιηθεί για παρόμοια δεδομένα. Επίσης, αν τα ίδια δεδομένα χρησιμοποιούνται επανειλημμένα, η εξαγωγή χαρακτηριστικών μια φορά μπορεί να εξοικονομήσει πολλούς υπολογιστικούς πόρους.

- **Fine-tuning(FT):** τα βάρη διατηρούνται trainable και fine-tuned για το target task. Έτσι το προεκπαιδευμένο μοντέλο λειτουργεί ως σημείο εκκίνησης για το μοντέλο, οδηγώντας σε ταχύτερη σύγκλιση συγκριτικά με την τυχαία αρχικοποίηση.

Συνοψίζοντας από τα παραπάνω, στο EX έχουμε το πλεονέκτημα να δημιουργούμε χαρακτηριστικά μια φορά και να δοκιμάζουμε διαφορετικά μοντέλα μ'αυτά, εξοικονομώντας έτσι πολύτιμους υπολογιστικούς πόρους για επανεκπαίδευση και πειραματικούς σκοπούς. Εναλλακτικά, το FT είναι ιδανικό και για τη βελτίωση του μοντέλου ώστε να χρησιμοποιηθεί για πολλά διαφορετικά tasks, και για τη διευκόλυνση της εργασίας μας, καθώς δε χρειάζεται να πειραματιστούμε για καμία παραλλαγή του μοντέλου μας.

Οι Peters et al.2019 (Peters et al., 2019), προέβησαν σε ανάλυση για την επίδραση των EX και FT πάνω σε επτά διαφορετικά tasks. Διαπίστωσαν ότι και οι δύο προσεγγίσεις επιτυγχάνουν τις ίδιες επιδόσεις τις περισσότερες φορές, αλλά το fine tuning αποδίδει καλύτερα όταν τα source και target tasks είναι παρόμοια, με μειονέκτημα βέβαια ότι μόνο οι λέξεις που εμφανίζονται στην εκπαίδευση θα έχουν ανανεωμένα embeddings, ενώ τα embeddings των μη ορατών λέξεων θα ξεχαστούν [5]. Ενώ το feature extraction αποδίδει καλύτερα όταν source και target tasks είναι διαφορετικά.

Σε γενικές γραμμές λοιπόν, εάν το σύνολο δεδομένων μας δεν είναι δραστικά διαφορετικό ως προς το περιεχόμενο σε σχέση με το σύνολο δεδομένων στο οποίο έχει εκπαιδευθεί το προ-εκπαιδευμένο μοντέλο, θα πρέπει να κάνουμε fine tuning.



Για το δικό μας σύνολο δεδομένων, που είναι τα logs του HPC θα ακολουθήσουμε την προσέγγιση του fine-tuning.

Συγκεκριμένα για τη μέθοδο του **fine-tuning** υπάρχουν ορισμένες **τεχνικές** που μπορούμε να χρησιμοποιήσουμε, όπως:

1. Η συνήθης πρακτική που χρησιμοποιείται είναι να περικόψουμε το τελευταίο layer (**truncate the last layer**) (softmax layer) του προ-εκπαιδευμένου

[5] Αυτό μπορεί να επηρεάσει την απόδοση όταν το σύνολο για εκπαίδευση είναι πολύ μικρό ή το σύνολο test περιέχει πολλά out-of-vocabulary (OOV).

μοντέλου και να το αντικαταστήσουμε με το καινούριο softmax layer που σχετίζεται με το δικό μας πρόβλημα. Για παράδειγμα, το προ-εκπαιδευμένο μοντέλο στο ImageNet έρχεται με ένα softmax layer με 1000 κατηγορίες. Εάν η εργασία μας ήταν μια ταξινόμηση σε 10 κατηγορίες, το νέο softmax layer του δικτύου μας θα είναι 10 κατηγοριών αντί για 1000. Στη συνέχεια, κάνουμε back propagation στο δίκτυο, για να κάνουμε fine-tune τα προ-εκπαιδευμένα βάρη. Εδώ είναι αναγκαίο το cross validation ώστε το δίκτυο να μπορεί να γενικεύσει καλά.

2. Να χρησιμοποιήσουμε ένα μικρότερο ρυθμό μάθησης (**smaller learning rate**) για την εκπαίδευση του δικτύου μας. Εφόσον περιμένουμε ότι τα προ-εκπαιδευμένα βάρη να είναι ήδη αρκετά καλά σε σύγκριση με τα τυχαία αρχικά βάρη, δε θέλουμε να τα παραμορφώσουμε πολύ γρήγορα και πάρα πολύ.

3. Είναι πολύ κοινή πρακτική να παγώνουμε τα βάρη των πρώτων layers του προ-εκπαιδευμένου μοντέλου (**freeze the weights of the first few layers**). Αυτό συμβαίνει γιατί τα πρώτα layers καταγράφουν καθολικά χαρακτηριστικά, όπως για παράδειγμα σε ένα πρόβλημα με εικόνες, καταγράφουν καμπύλες, και ακμές, που σχετίζονται με το καινούριο πρόβλημα των εικόνων. Θέλουμε να κρατήσουμε αυτά τα βάρη άθικτα. Αντ'αυτού, θα κάνουμε το δίκτυο μας να επικεντρωθεί στην εκμάθηση συγκεκριμένων χαρακτηριστικών του συνόλου δεδομένων μας στα επόμενα layers.

Στην εργασία αυτή χρησιμοποίησα τη δεύτερη και την τρίτη από τις τεχνικές για fine-tuning που παρουσίασα παραπάνω.

2.9 Απόδοση Νευρωνικών Δικτύων

2.9.1 Σφάλματα

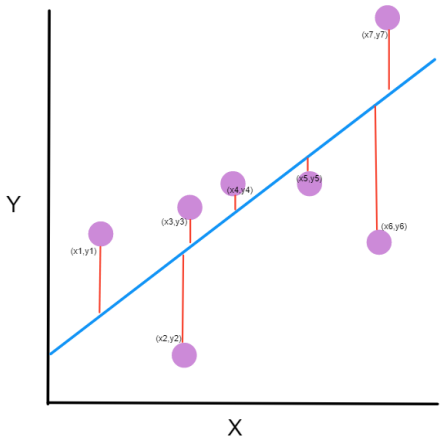
Στη Μηχανική Μάθηση ο κύριος στόχος μας είναι να ελαχιστοποιήσουμε το error το οποίο ορίζεται από τη **Loss Function**. Στα περισσότερα neural networks, το σφάλμα υπολογίζεται ως η διαφορά μεταξύ της πραγματικής τιμής και της προβλεπόμενης τιμής.

$$J(w) = p - \overline{p} \tag{2.2}$$

Η συνάρτηση που χρησιμοποιείται για τον υπολογισμό αυτού του σφάλματος είναι γνωστή ως Συνάρτηση Σφάλματος (Loss Function), J. Οι διαφορετικές συναρτήσεις σφάλματος θα δώσουν διαφορετικά σφάλματα για την ίδια πρόβλεψη και έτσι θα έχουν σημαντική επίδραση στην απόδοση του μοντέλου. Η συνάρτηση σφάλματος είναι μία από τις δύο παραμέτρους που απαιτούνται για την κατάρτιση ενός μοντέλου (KERAS). Έτσι αν βελτιστοποιούμε τη λάθος συνάρτηση loss, θα φτάσουμε στη λάθος απάντηση, γι'αυτό και πρέπει να διαλέξουμε την κατάλληλη. Στην παρούσα διπλωματική εργασία γίνεται χρήση της **Categorical Crossetnropy** κατά τη φάση 1 και της **Mean Squared Error** κατά την εκπαίδευση του skipgram και τη φάση 2.

- **Mean Squared Error-MSE**

  Υπολογίζεται ως ο μέσος όρος των τετραγωνικών διαφορών ανάμεσα στις προβλεπόμενες και τις πραγματικές τιμές. Το MSE είναι μια risk συνάρτηση που αντιστοιχεί στην αναμενόμενη τιμή του squared error loss.



Σχήμα 2.19: *Points on a simple graph.*

- Οι μωβ τελείες είναι τα σημεία στο γράφημα. Κάθε σημείο έχει x και y συντεταγμένες.
- Η μπλε γραμμή είναι η γραμμή πρόβλεψή μας. Αυτή η γραμμή περνάει από όλα τα σημεία και τα ταιριάζει με τον καλύτερο τρόπο. Αυτή η γραμμή περιλαμβάνει τα προβλεπόμενα σημεία.
- Η κόκκινη γραμμή μεταξύ κάθε μωβ σημείου και της γραμμής πρόβλεψης είναι τα errors. Κάθε error είναι η απόσταση από το σημείο στο προβλεπόμενο σημείο.

Το mean squared error ή αλλιώς το μέσο τετραγωνικό σφάλμα λέει πόσο κοντά είναι μια γραμμή παλινδρόμησης σε ένα σύνολο σημείων. Αυτό το επιτυγχάνει λαμβάνοντας τις αποστάσεις των σημείων από την γραμμή παλινδρόμησης και τετραγωνίζοντας αυτές. Ο τετραγωνισμός αυτός είναι απαραίτητος για την εξάλειψη πιθανών αρνητικών τιμών - το αποτέλεσμα πρέπει να είναι πάντα θετικό ανεξάρτητα από το πρόσημο των προβλεπόμενων και πραγματικών τιμών. Επίσης δίνει μεγαλύτερο βάρος στις μεγαλύτερες διαφορές. Ονομάζεται έτσι καθώς εντοπίζεται ο μέσος όρος ενός συνόλου σφαλμάτων. Όσο μικρότερο είναι το MSE, τόσο πιο κοντά βρισκόμαστε στο να βρούμε την γραμμή με το καλύτερο fit.

- **Categorical Crossentropy**

  Τι είναι η Cross-Entropy?

  Η **Cross-Entropy** είναι ένα μέτρο της διαφοράς ανάμεσα σε δύο κατανομές πιθανοτήτων για μια δεδομένη τυχαία μεταβλητή ή σύνολο γεγονότων. Η πληροφορία, όπως είναι γνωστό, ποσοτικοποιεί τον αριθμό των bits που απαιτείται για την κωδικοποίηση και τη μετάδοση ενός γεγονότος. Όπως είναι γνωστό τα μικρότερης πιθανότητας γεγονότα-μηνύματα έχουν περισσότερη πληροφορία, ενώ τα μεγαλύτερης πιθανότητας γεγονότα-μηνύματα έχουν λιγότερη πληροφορία.

  Η **Entropy-Εντροπία** είναι συνάρτηση της κατάστασης του συστήματος, οπότε η αλλαγή της εντροπίας ενός συστήματος καθορίζεται από τις αρχικές και τελικές καταστάσεις. Στην ιδανική περίπτωση ότι μια διαδικασία είναι αναστρέψιμη, η εντροπία δεν αλλάζει, ενώ οι μη αναστρέψιμες διαδικασίες πάντα αυξάνουν τη συνολική εντροπία. Επειδή καθορίζεται από τον αριθμό

των τυχαίων microstates, η εντροπία σχετίζεται με την ποσότητα της επιπρόσθετης πληροφορίας που απαιτείται για τον καθορισμό της ακριβούς φυσικής κατάστασης ενός συστήματος δεδομένων των μακροσκοπικών του προδιαγραφών. Γι'αυτό το λόγο, συχνά λέγεται ότι η εντροπία είναι μια έκφραση της διαταραχής, ή της τυχαιότητας ενός συστήματος, ή της έλλειψης πληροφορίας σχετικά μ'αυτό. *Η έννοια της εντροπίας παίζει κεντρικό ρόλο στη θεωρία της πληροφορίας*. Ειδικότερα, η εντροπία είναι ένα λογαριθμικό μέτρο του αριθμού των καταστάσεων με σημαντική πιθανότητα αυτών να είναι "occupied".

Μια skewed[6] κατανομή έχει μικρότερη entropy, ενώ μια κατανομή όπου τα γεγονότα-μηνύματα έχουν την ίδια πιθανότητα έχει μεγαλύτερη entropy. Στη θεωρία της πληροφορίας, θέλουμε να περιγράφουμε την "έκπληξη" ενός γεγονότος.

- **Probability Distribution** (unsurprising): Χαμηλή εντροπία.
- **Balanced Probability Distribution**(surprising): Υψηλή εντροπία.

Η Cross-Entropy (CE) λοιπόν βασίζεται στην ιδέα της εντροπίας από τη θεωρία της πληροφορίας και υπολογίζει των αριθμό των bits που απαιτούνται για την αντιπροσώπευση ή τη μετάδοση ένα γεγονότος από μία κατανομή συγκριτικά με μια άλλη. Η CE loss μπορεί να ορισθεί ως:

$$CE = -\sum_{i}^{C} t_i log(s_i)$$

όπου t_i και s_i είναι η ground truth και το score για κάθε κλάση i στο C.

Η **Categorical Cross-Entropy** χρησιμοποιείται για multi-class προβλήματα. Οι κλάσεις-κατηγορίες έχουν κωδικοποιηθεί one-hot, αυτό σημαίνει ότι υπάρχει μια δυαδική αναπαράσταση για κάθε κλάση, μόνο η θετική κλάση C_p κρατάει τον όρο της στο loss. Υπάρχει μόνο ένα στοιχείο του Target vector t το οποίο δεν είναι μηδέν $t_i = t_p$. Επίσης οι προβλέψεις πρέπει να είναι προβλεπόμενες πιθανότητες για κάθε μία από τις κατηγορίες. Η CE μπορεί να γραφεί ως:

$$CE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)$$

όπου S_p είναι το σκορ για τη θετική κλάση.

2.9.2 Μετρικές

Στη διπλωματική αυτή εργασία χρησιμοποιούνται διάφορες Μετρικές (Metrics). Μια μετρική είναι μια συνάρτηση που χρησιμοποιείται για να κρίνεται η απόδοση του

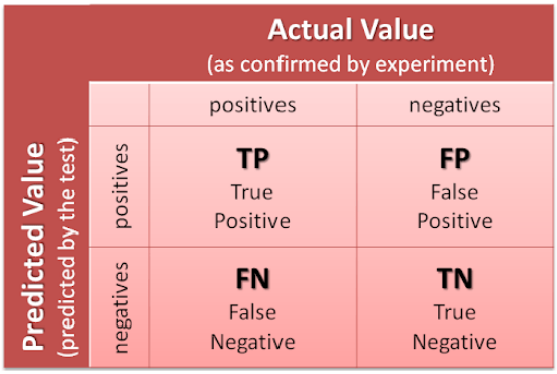
[6]Μια κατανομή λέγεται skewed όταν τα σημεία των δεδομένων συγκεντρώνονται περισσότερο προς μία πλευρά της κλίμακας από την άλλη, δημιουργώντας μια καμπύλη που δεν είναι συμμετρική. Με άλλα λόγια, η δεξιά και η αριστερή πλευρά της κατανομής διαμορφώνονται διαφορετικά μεταξύ τους.

μοντέλου. Με δεδομένες τις προβλεπόμενες τιμές και τις πραγματικές (ground truth)[7] τιμές η σύναρτηση αυτή δίνει ένα βαθμωτό μέτρο της καταλληλότητας του μοντέλου, με τα υπάρχοντα δεδομένα. Μια μετρική συνάρτηση είναι παρόμοια με τη συνάρτηση σφάλματος, εκτός από το ότι τα αποτελέσματα από την αξιολόγηση μιας μέτρησης δεν χρησιμοποιούνται κατά την εκπαίδευση του μοντέλου.

- **Πίνακας Σύγχυσης - Confusion Matrix**

Στο πεδίο της μηχανικής μάθησης, ένας πίνακας σύγχυσης-Confusion Matrix, επίσης γνωστός ως πίνακας σφάλματος, (Stehman, 1997) είναι μια συγκεκριμένη διάταξη πίνακα που επιτρέπει την απεικόνιση της απόδοσης ενός supervised learning algorithm (σε μη-επιβλεπόμενη εκπαίδευση, συνήθως ονομάζεται πίνακας αντιστοιχίας (Matching Matrix). Κάθε γραμμή του πίνακα αντιπροσωπεύει τα στιγμιότυπα σε μια προβλεπόμενη κλάση ενώ κάθε στήλη αντιπροσωπεύει τα στιγμιότυπα σε μια πραγματική κλάση (ή αντίστροφα) (Powers, 2011). Το όνομα πηγάζει από το γεγονός ότι καθιστά εύκολο το να διαπιστωθεί εάν το σύστημα προκαλεί σύγχυση σε δύο κλάσεις (π.χ. συχνά δίνοντας λάθος ετικέτες η μία με την άλλη).

Πρόκειται για ένα ειδικό είδος πίνακα ενδεχομένων, με δύο διαστάσεις ("πραγματικό", και "προβλεπόμενο") και πανομοιότυπα σύνολα "κλάσεων" και στις δύο διαστάσεις (κάθε συνδυασμός διαστάσεων και κλάσης είναι μια μεταβλητή στον πίνακα ενδεχομένων).

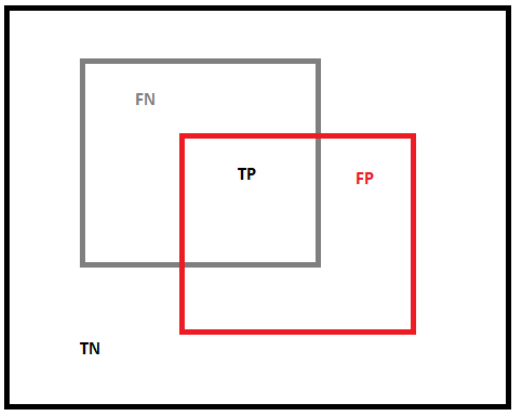


Σχήμα 2.20: *Πίνακας Σύγχυσης .*

Όλες οι σωστές προβλέψεις εντοπίζονται στη διαγώνιο του πίνακα, επομένως είναι εύκολο να ελεχθεί οπτικά ο πίνακας για σφάλματα πρόβλεψης, καθώς θα αντιπροσωπεύονται από τιμές εκτός της διαγωνίου.

True Positive-TP: Αληθώς θετική ονομάζεται μια πρόβλεψη που εκτιμήθηκε ότι ανήκει σε μια συγκεκριμένη κλάση και αυτό ισχύει.
False Positive-FP: Ψευδώς θετική ονομάζεται μια πρόβλεψη που εκτιμήθηκε ότι ανήκει σε μια συγκεκριμένη κλάση και αυτό δεν ισχύει.
False Negative-FN: Ψευδώς αρνητική ονομάζεται μια πρόβλεψη που εκτιμήθηκε ότι δεν ανήκει σε μια συγκεκριμένη κλάση και αυτό δεν ισχύει.
True Negative-TN: Αληθώς αρνητική ονομάζεται μια πρόβλεψη που εκτιμήθηκε ότι δεν ανήκει σε μια κλάση και αυτό ισχύει.

[7]Η ground truth είναι αυτό που μετριέται για τη μεταβλητή-στόχο για τα παραδείγματα εκπαίδευσης και δοκιμών. Συνήθως οι τιμές ground truth αντιμετωπίζονται όπως και οι ετικέτες. Υπάρχουν βέβαια και περιπτώσεις που δεν είναι ακριβώς η ίδια με την ετικέτα.



Σχήμα 2.21: *FP-FN-TP-TN, Η κλάση είναι το γκρι κουτάκι και το TP είναι στην τομή από το γκρι και το κόκκινο κουτάκι, οι προβλέψεις που εκτιμήθηκαν ότι ανήκουν στην κλάση και ισχύει. Στο γκρι κουτάκι το FN είναι οι προβλέψεις που εκτιμήθηκαν ότι δεν ανήκουν στην κλαση και αυτό δεν ισχύει. Στο κόκκινο κουτάκι οι προβλέψεις που εκτιμήθηκαν ότι ανήκουν στην κλάση αλλά δεν ισχύει. Τέλος έξω από όλα τα εσωτερικά κουτάκια και μέσα στο μαύρο κουτάκι είναι το TN που είναι οι προβλέψεις που εκτιμήθηκαν ότι δεν ανήκουν στην κλάση και ισχύει..*

Στη διπλωματική χρησιμοποιούνται οι εξής μετρικές: categorical accuracy και Jaccard coefficient για τη φάση 1, accuracy για τη φάση 2 και το recall, το precision και το f1 score για τη φάση 3.

- **accuracy(ακρίβεια)**

Η ακρίβεια είναι μια μετρική η οποία αξιολογεί όπως δηλώνει και το όνομα της, πόσο ακριβής είναι η πρόβλεψη του μοντέλου σε σχέση με τα πραγματικά δεδομένα. Δηλαδή, η ακρίβεια είναι ο λόγος των πραγματικών αποτελεσμάτων (τόσο αληθινών όσο και αρνητικών) μεταξύ του συνολικού αριθμού των περιπτώσεων που εξετάστηκαν.

$$\frac{TP + TN}{TP + FP + FN + TN} \tag{2.3}$$

- **recall**

Το recall (ή sensitivity) είναι μια παραδοσιακή μέτρηση αξιολόγησης, η οποία όπως όλα τα μέτρα, υποθέτουν μια έννοια βασικής αλήθειας (ground truth). Είναι το κλάσμα των επιτυχών προβλέψεων μιας κλάσης προς τα συνολικά παραδείγματα αυτής. Η μετρική αυτή μπορεί να ερμηνευθεί ως το ποσό των θετικών δειγμάτων του test που ταξινομήθηκαν πράγματι ως θετικά. Όσα λιγότερα False Negatives - FN ένας ταξινομητής δίνει τόσο μεγαλύτερο είναι το recall του.
Το recall λοιπόν, μπορεί να θεωρηθεί ως μέτρο πληρότητας ή ποσότητας. Χρησιμοποίειται και αυτή στο κομμάτι της αξιολόγησης.

$$\frac{TP}{TP + FN} \tag{2.4}$$

- **precision**

Δείχνει το ποσοστό των επιτυχών προς των συνολικών προβλέψεων μιας κλάσης από ένα μοντέλο. Διαισθητικά το precision μπορεί να θεωρηθεί ως η ικανότητα του ταξινομητή (classifier) να προβλέπει μόνο πραγματικά θετικά δείγματα ως θετικά. Άρα όταν δεν υπάρχουν False Positives - FP, δηλαδή ταξινομεί μόνο τα True Positive ως θετικά, θα έχει precision 1. Έτσι, στην ουσία όσο λιγότερα FP δίνει ένας ταξινομητής, τόσο υψηλότερο είναι το precision του.
Η μετρική αυτή μπορεί να θεωρηθεί ως μέτρο ποιότητας.

$$\frac{TP}{TP + FP} \tag{2.5}$$

Έτσι, όσο υψηλότερο είναι το precision και το recall, τόσο καλύτερη είναι η απόδοση του ταξινομητή επειδή ανιχνεύει τα περισσότερα από τα positive δείγματα (υψηλό recall) και δεν ανιχνεύει πολλά δείγματα που δεν πρέπει να ανιχνευθούν (υψηλό precision).

Για να το ποσοτικοποιήσουμε, χρησιμοποιούμε μια άλλη μετρική που ονομάζεται F1 score.

- **F1 score**

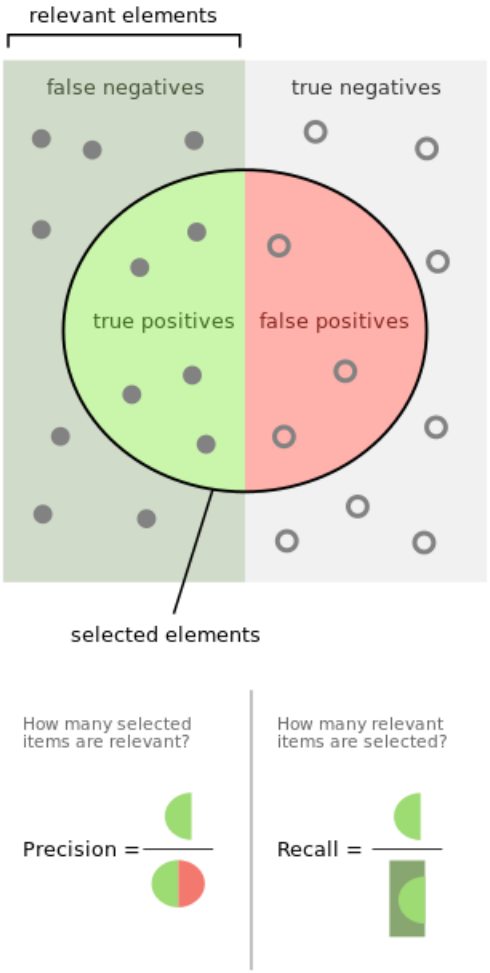
Η μετρική αυτή χρησιμοποιείται στο κομμάτι της αξιολόγησης και συνδυάζει το precision και το recall. Συγκεκριμένα είναι ο σταθμισμένος (weighted) μέσος όρος του precision και του recall. Το F1 score φτάνει στην καλύτερη τιμή του στο 1 (τέλειο precision και recall) και χειρότερη στο 0.

$$\frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.6}$$

Οι παραπάνω μετρικές χρησιμοποιούνται πέρα από την αξιολόγηση των μοντέλων που εκπαιδεύονται και για την αξιολόγηση της συνολικής μεθοδολογίας που ακολουθείται για την πρόβλεψη εν τέλει της αποτυχίας των κόμβων. Στον παρακάτω πίνακα φαίνεται και ο τρόπος και η λογική με τα οποία υπολογίζονται οι μετρικές αυτές στο πλαίσιο της πρόβλεψης αποτυχίας των κόμβων.

Metric	**Formula & Implication**
Recall	TP/(TP+FN) # Nodes failures correctly predicted
Precision	TP/(TP+FP) # Total node failures predicted
FP Rate	FP/(FP+TN) # False Positive Rate
True Positive (TP)	# Actual node failures, successfully predicted
True Negative (TN)	# Nodes actually didn't fail, are not be predicted as failures
False Positive (FP)	# Nodes actually didn't fail, but are predicted as failure
False Negative (FN)	# Actual node failures, but failed to be predicted

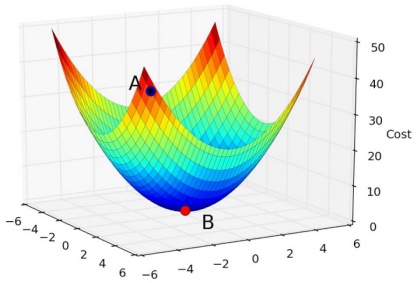
Σχήμα 2.22: *Μετρικές Αξιολόγησης.*



Σχήμα 2.23: *Precision και Recall .*

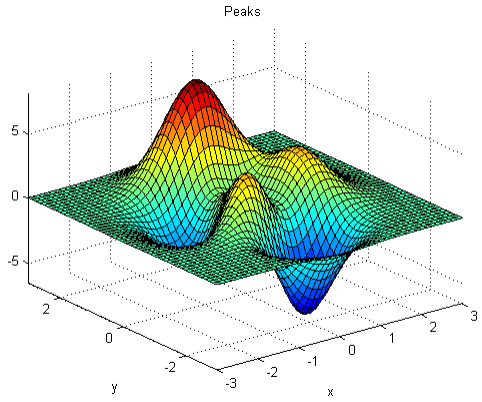
2.10 Βελτιστοποίηση

Οι αλγόριθμοι βελτιστοποίησης ελαχιστοποιούν μια συνάρτηση σφάλματος J(x), η οποία είναι απλώς μια μαθηματική συνάρτηση που εξαρτάται από τις εσωτερικές παραμέτρους του μοντέλου, που χρησιμοποιούνται για τον υπολογισμό των τιμών στόχων (Υ) από το σύνολο των προγνωστικών τιμών (X) που χρησιμοποιούνται στο μοντέλο. Οι εσωτερικές παράμετροι ενός μοντέλου διαδραματίζουν πολύ σημαντικό ρόλο στην αποτελεσματική και αποδοτική εκπαίδευση του μοντέλου και παράγουν ακριβή αποτελέσματα. Αυτός είναι ο λόγος που χρησιμοποιούμε διάφορες στρατηγικές και αλγορίθμους βελτιστοποίησης για να ενημερώσουμε και να υπολογίσουμε τις κατάλληλες και βέλτιστες τιμές των παραμέτρων αυτού του μοντέλου που επηρεάζουν τη διαδικασία μάθησης του μοντέλου μας καθώς και την έξοδο του. Με απλά λόγια, υπολογίζει πώς να αλλάξουμε τα βάρη του νευρωνικού μας δικτύου, ώστε το error να γίνει χαμηλότερο σε κάθε επανάληψη. Ο στόχος όλων των βελτιστοποιητών λοιπόν είναι να φθάσουν το global minima όπου η συνάρτηση κόστους επιτυγχάνει την ελάχιστη δυνατή τιμή. Αν προσπαθήσουμε να απεικονίσουμε τη λειτουργία κόστους σε 3 διαστάσεις θα ήταν κάτι σαν το σχήμα που φαίνεται παρακάτω:



Σχήμα 2.24: *convex cost function.*

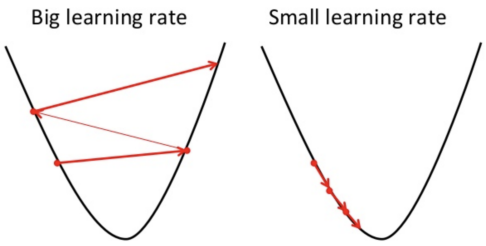
Βέβαια δεν είναι πάντα η συνάρτηση κόστους τόσο ομαλή όσο στην εικόνα 2.24. Τις περισσότερες φορές οι συναρτήσεις κόστους θα είναι μη-κυρτές (non-convex). Το πρόβλημα με τη non-convex συνάρτηση είναι ότι υπάρχει η πιθανότητα να κολλήσουμε σε ένα τοπικό ελάχιστο και η loss να μη μπορεί ποτέ να συγκλίνει σε ένα global maxima. Όπως στην παρακάτω εικόνα:



Σχήμα 2.25: *Non-Convex cost function.*

- **Learning Rate**

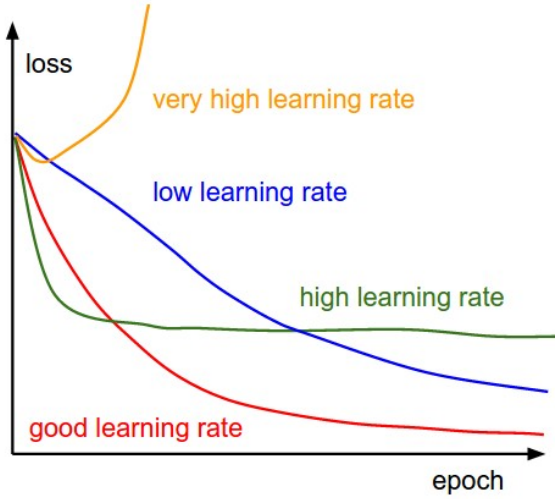
Ο **ρυθμός εκμάθησης - learning rate** είναι ίσως η πιο σημαντική πτυχή της gradient descent και πολλών άλλων βελτιστοποιητών. Ο learning rate - η είναι μια **υπερ-παράμετρος** που ελέγχει πόσο προσαρμόζονται τα βάρη του δικτύου σε σχέση με την κλίση της απώλειας loss gradient. Καθορίζει το μέγεθος των βημάτων που παίρνονται για να προσεγγιστεί ένα (τοπικό) ελάχιστο. Με άλλα λόγια, ακολουθείται η κατεύθυνση της κλίσης της επιφάνειας που δημιουργείται από την αντικειμενική συνάρτηση προς τα κάτω μέχρι να προσεγγιστεί η κοιλάδα. Έστω δηλαδή ότι έχουμε τη συνάρτηση cost ως ένα λάκκο, θα ξεκινήσουμε από την κορυφή του λάκκου με στόχο να φτάσουμε στο κάτω μέρος του-κοιλάδα. Άρα όπως είπαμε και παραπάνω μπορούμε να θεωρήσουμε το learning rate ως το βήμα που θα πάρουμε για να φτάσουμε στο κάτω μέρος του λάκου (global minima).



Σχήμα 2.26: *Διαφορετικά learning rates.(Gandhi).*

Η επιλογή κατάλληλου ρυθμού μάθησης μπορεί να είναι δύσκολη. Αν επιλέξουμε μεγάλο learning rate θα κάνουμε μεγάλες αλλαγές στα βάρη, τιμές μεροληψίας κλπ.-άρα τεράστια άλματα για να φτάσουμε στο κάτω μέρος. Αυτό όμως κρύβει τον κίνδυνο να υπερβούμε το global minima (κοιλάδα) και να καταλήξουμε στην άλλη πλευρά του λάκκου. Με ένα μεγάλο λοιπόν learning rate δε θα είμαστε ποτέ σε θέση να συγκλίνουμε με τα global minima και πάντα θα περιπλανιώμαστε γύρω από αυτά. Αν από την άλλη επιλέξουμε ένα μικρό learning rate, δεν έχουμε το ρίσκο να υπερβούμε το παγκόσμιο ελάχιστο, αλλά ο αλγόριθμος μας θα χρειάζεται περισσότερο χρόνο για να συγκλίνει-παίρνουμε μικρότερα βήματα, αλλά θα πρέπει να πάρουμε και περισσότερα. Ως εκ τούτου, θα πρέπει να εκπαιδεύσετε για μεγαλύτερο χρονικό διάστημα. (Gandhi) Επίσης, αν η cost function είναι non-convex ο αλγόριθμός μπορεί εύκολα να παγιδευτεί σε κάποιο τοπικό ελάχιστο και δε θα μπορέσει να συγκλίνει στο παγκόμιο-ολικό ελάχιστο.

Δεν υπάρχει μια γενική σωστή τιμή για το learning rate. Προγράμματα ρυθμού εκμάθησης (Robbins and Monro, 1985) προσπαθούν να προσαρμόσουν τον ρυθμό εκμάθησης κατά τη διάρκεια της εκπαίδευσης, π.χ. με ανόπτηση, δηλαδή μείωση του ρυθμού εκμάθησης σύμφωνα με ένα προκαθορισμένο χρονοδιάγραμμα ή όταν η αλλαγή στόχου μεταξύ των εποχών πέσει κάτω από ένα κατώφλι. Ωστόσο τα προγράμματα και τα κατώφλια πρέπει να καθοριστούν εκ των προτέρων και συνεπώς δεν είναι σε θέση να προσαρμοστούν στα χαρακτηριστικά ενός συνόλου δεδομένων (Darken et al., 1992). Επιπλέον, ο ίδιος ρυθμός εκμάθησης ισχύει για όλες τις ενημερώσεις παραμέτρων. Εάν τα δεδομένα είναι αραιά και τα χαρακτηριστικά έχουν πολύ διαφορετικές συχνότητες, ίσως να μην είναι επιθυμητό να ενημερώνονται όλα τους στον ίδιο βαθμό, αλλά να εκτελείται μια μεγαλύτερη ενημέρωση για σπάνια χαρακτηριστικά.



Σχήμα 2.27: *Η επίδραση διαφόρων learning rate στη σύγκλιση.*

Στην διπλωματική εργασία οι αλγόριθμοι βελτιστοποίησης που χρησιμοποιούνται είναι η SGD στη φάση 1 και ο RMSprop στη φάση 2 και στο skipgram.

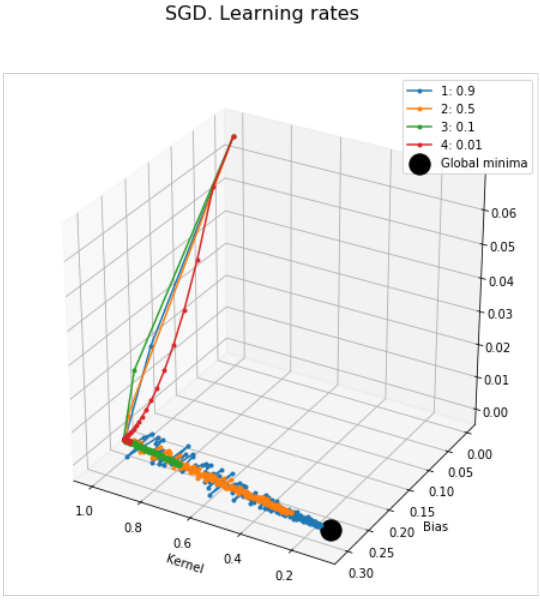
2.10.1 Gradient Descent

Η Gradient Descent είναι η πιο σημαντική τεχνική και το θεμέλιο του τρόπου με τον οποίο εκπαιδεύουμε και βελτιστοποιούμε τα Ευφυή Συστήματα. Είναι ο πιο δημοφιλής αλγόριθμος βελτιστοποίησης ενός νευρωνικού δικτύου. Χρησιμοποιείται σε μεγάλο βαθμό για την πραγματοποίηση ενημερώσεων των βαρών σε ένα μοντέλο νευρωνικών δικτύων, δηλαδή την ενημέρωση και τον συντονισμό των παραμέτρων του μοντέλου σε μια κατεύθυνση ώστε να ελαχιστοποιήσουμε τη συνάρτηση σφάλματος.

Ταυτόχρονα, κάθε σύγχρονη βιβλιοθήκη Deep Learning περιλαμβάνει υλοποιήσεις διαφόρων αλγορίθμων για τη βελτιστοποίηση της Gradient Descent (π.χ. caffe και keras). Αυτοί οι αλγόριθμοι, ωστόστο, χρησιμοποιούνται συχνά ως βελτιστοποιητές μαύρου κουτιού, καθώς οι πρακτικές εξηγήσεις σχετικά με τις δυνατότητες και αδυναμίες είναι δύσκολο να γίνουν.

Η Gradient Descent είναι ένας επαναληπτικός αλγόριθμος βελτιστοποίησης λοιπόν, που χρησιμοποιείται στη Μηχανική Μάθηση για να βρεθεί το καλύτερο αποτέλεσμα (το ελάχιστο μιας καμπύλης δηλαδή).

Gradient νοείται ο ρυθμός αύξησης ή μείωσης μιας κλίσης. Descent είναι η περίπτωση της φθίνουσας. Ο αλγόριθμος είναι επαναληπτικός γιατί πρέπει να πάρουμε τα αποτελέσματα πολλές φορές για να έχουμε το βέλτιστο αποτέλεσμα. Η επαναληπτική κατάσταση της gradient descent βοηθά ένα μη καλά fitted γράφο για παράδειγμα, να κάνει βέλτιστο fit στα δεδομένα.



Σχήμα 2.28: *Visualization of the training process for SGD optimizer with various learning rates.*

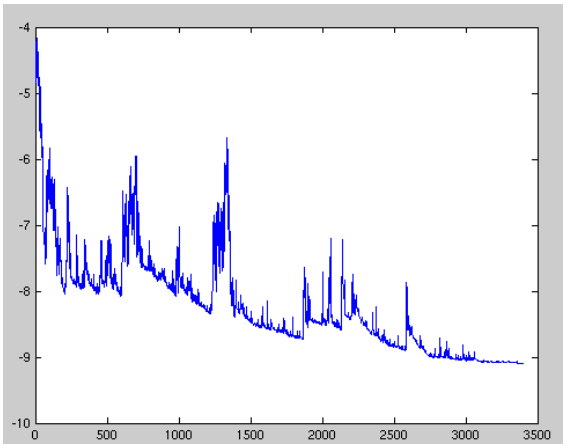
Στην παραπάνω εικόνα βλέπουμε ότι στις περισσότερες περιπτώσεις οι βελτιστοποιητές δεν ήταν σε θέση να πλησιάσουν τα ελάχιστα στο δεδομένο χρόνο. Εδώ μεταξύ πολλών άλλων λόγων, τα step sizes ήταν πολύ μικρά για να μπορούν να κινηθούν αρκετά γρήγορα, έτσι σταματήσαν αρκετά πιο μακριά από το ελάχιστο σημείο. Ο βελτιστοποιητής με το σχετικά πιο υψηλό learning rate από την άλλη πλευρά ήταν σε θέση να φτάσει το ελάχιστο και με αρκετό χρόνο εκπαίδευσης, τελικά θα φτάσει στο επιθυμητό σημείο.

2.10.2 Stochastic Gradient Descent-SGD

Η Stochastic Gradient Descent - SGD εκτελεί μια ενημέρωση παραμέτρων για κάθε παράδειγμα εκπαίδευσης. Είναι συνήθως πολύ πιο γρήγορη τεχνική. Εκτελεί μία ενημέρωση κάθε φορά για κάθε παράδειγμα εκπαίδευσης $x^{(i)}$ και ετικέτα $y^{(i)}$:

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.7}$$

Η batch gradient descent εκτελεί περιττούς υπολογισμούς για μεγάλα σύνολα δεδομένων, διότι επαναπροσδιορίζει τις κλίσεις για παρόμοια παραδείγματα πριν από κάθε ενημέρωση παραμέτρων. Η SGD απομακρύνει αυτό τον πλεονασμό εκτελώντας μία ενημέρωση τη φορά. Είναι επομένως συνήθως πολύ ταχύτερη και μπορεί επίσης να χρησιμοποιηθεί για να μάθει online. Η SGD εκτελεί συχνές ενημερώσεις με μεγάλη διακύμανση με συνέπεια οι ενημερώσεις παραμέτρων να έχουν μεγάλη διακύμανση η οποία προκαλεί μεγάλη αντικειμενική μεταβολή της αντικειμενικής λειτουργίας όπως στην εικόνα 2.29:



Σχήμα 2.29: *Διακυμάνσεις SGD.(Ruder, 2016).*

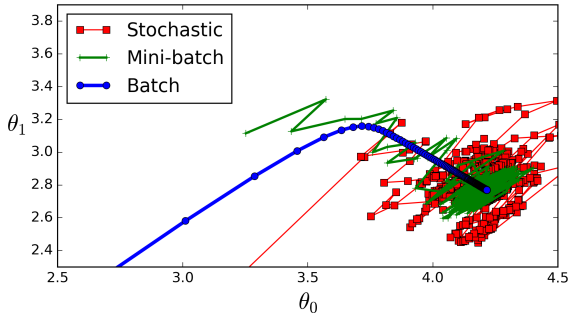
Οι διακυμάνσεις της SGD προκαλούν τη διακύμανση της συνάρτησης σφάλματος σε διαφορετικές εντάσεις. Αυτό είναι πράγματι καλό γιατί βοηθά να ανακαλυφθούν νέα και ίσως καλύτερα τοπικά ελάχιστα (Walia, 2017). Από την άλλη πλευρά, αυτό τελικά περιπλέκει τη σύγκλιση στο ακριβές ελάχιστο καθώς η SGD θα συνεχίσει να υπερβαίνει. Ωστόσο, έχει αποδειχθεί ότι όταν μειώνεται αργά ο ρυθμός εκμάθησης, το SGD σχεδόν σίγουρα συγκλίνει σε ένα τοπικό ή παγκόσμιο ελάχιστο για μη κυρτή και κυρτή βελτιστοποίηση αντίστοιχα (Ruder, 2016).

Μέγεθος της Παρτίδας (Batch Size)

Ορίζει τον αριθμό των δειγμάτων που βλέπει το μοντέλο πριν ενημερώσει τα βάρη. Για παράδειγμα, έστω ότι έχουμε 1050 δείγματα εκπαίδευσης και μέγεθος παρτίδας ίσο με 100. Ο αλγόριθμος παίρνει τα πρώτα 100 δείγματα (από το 1ο έως το 100ο) από το σύνολο δεδομένων εκπαίδευσης και εκπαιδεύει το δίκτυο. Στη συνέχεια παίρνει τα δεύτερα 100 δείγματα (από το 101ο έως το 200ο) και πάλι εκπαιδεύει το δίκτυο. Αυτή η διαδικασία μπορεί να συνεχιστεί μέχρι να διαδωθεί μέσω των δικτύων σε όλα τα δείγματα. Το πρόβλημα συνήθως συμβαίνει με το τελευταίο σετ δειγμάτων. Στο συγκεκριμένο παράδειγμα που αναφέρθηκε υπάρχουν 1050 δείγματα, που όμως το 1050 δεν διαρείται με το 100 χωρίς υπόλοιπο. Η απλούστερη λύση είναι απλά να περαστούν τα τελικά 50 δείγματα στον αλγόριθμο και να εκπαιδευτεί το δίκτυο. Σε αυτό το σημείο ο αλγόριθμος έχει εκτελέσει μια εποχή.

Με αυτό τον τρόπο απαιτείται λιγότερη μνήμη, δεδομένου ότι εκπαιδεύεται δίκτυο χρησιμοποιώντας μικρότερο αριθμό δειγμάτων. Είναι ιδιαίτερα σημαντικό σε περίπτωση που δεν υπάρχει η δυνατότητα να προσαρμοστεί το σύνολο δεδομένων στη μνήμη. Συνήθως τα δίκτυα εκπαιδεύονται ταχύτερα με μίνι-παρτίδες (mini-batches). Αυτό επειδή ενημερώνονται τα βάρη μετά από κάθε διάδοση. Στο παράδειγμα αυτό έχουν διαδοθεί 11 παρτίδες (10 από αυτά είχαν 100 δείγματα και 1 είχε 50 δείγματα) και μετά από κάθε μία από αυτές ενημερώθηκαν οι παράμετροι του δικτύου. Εάν χρησιμοποιούνταν όλα τα δείγματα κατά τη διάδοση, θα γινόταν μόνο 1 ενημέρωση για την παράμετρο του δικτύου.

Βέβαια όσο μικρότερη είναι η παρτίδα τόσο λιγότερο ακριβής είναι η εκτίμηση της κλίσης.



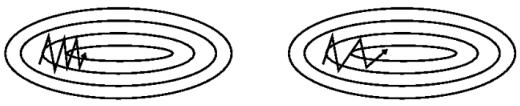
Σχήμα 2.30: *Στο σχήμα είναι εύκολο να παρατηρήσει κάποιος ότι η κατεύθυνση κλίσης της μίνι-παρτίδας (πράσινο χρώμα) κυμαίνεται σε σύγκριση με την πλήρη παρτίδα (μπλε χρώμα).*

Η **Stochastic Gradient Descent - SGD** είναι μόνο μία μίνι-παρτίδα (mini-batch) με μέγεθος (batch_size) ίσο με 1. Η κλίση αλλάζει την κατεύθυνση της ακόμη πιο συχνά από μια μίνι-παρτίδα.

Όταν γίνεται η εκπαίδευση (training) του νευρωνικού δικτύου, ή η αξιολόγηση (evaluation) η διάδοση των δεδομένων γίνεται με τις παρτίδες (batches) όπως αναφέρθηκε και παραπάνω. Επίσης από τα παραπάνω είναι προφανές ότι το νευρωνικό δίκτυο θα τροφοδοτηθεί με όλες τις παρτίδες. Η χρήση όλων των παρτίδων μία φορά είναι 1 εποχή. Εάν λοιπόν υπάρχουν 10 εποχές σημαίνει ότι θα χρησιμοποιηθούν όλα τα δεδομένα (χωριστά σε παρτίδες) 10 φορές.

Momentum-Ορμή

Η sgd με momentum σχεδόν πάντοτε συγκλίνει ταχύτερα από την τυπική sgd. Στην τυπική sgd, θα παίρναμε μεγαλύτερα βήματα προς τη μία κατεύθυνση και μικρότερα βήματα προς την άλλη κάτι το οποίο επιβραδύνει τον αλγόριθμο. Στην παρακάτω εικόνα, αριστερά βλέπουμε ότι η τυπική sgd παίρνει μεγαλύτερα βήματα κατά την κατεύθυνση-y και μικρότερα βήματα κατά την κατεύθυνση-x. Αν ο αλγόριθμός μας είναι σε θέση να μειώσει τα βήματα που λαμβάνονται κατά τον άξονα y και να συγκεντρώσει την κατεύθυνση του βήματος στην κατεύθυνση x τότε ο αλγόριθμος θα συγκλίνει ταχύτερα. Αυτό κάνει το momentum, περιορίζει την ταλάντωση ως προς μία κατεύθυνση έτσι ώστε ο αλγόριθμος να μπορεί να συγκλίνει ταχύτερα. Επίσης, δεδομένου ότι ο αριθμός των βημάτων που λαμβάνονται κατά την κατεύθυνση y περιορίζεται, μπορούμε να ορίσουμε ένα υψηλότερο learning rate (Gandhi).



Σχήμα 2.31: *Η sgd χωρίς momentum αριστερά και η sgd με momentum δεξιά .*

Η ορμή-momentum (Qian, 1999) είναι μια μέθοδος που βοηθά στην επιτάχυνση της sgd στη σχετική κατεύθυνση και μειώνει τις ταλαντώσεις όπως φαίνεται και στη δεξιά εικόνα 2.31. Αυτό επιτυγχάνεται προσθετόντας ένα κλάσμα γ του διανύσματος ενημέρωσης του προηγούμενου βήματος χρόνου στο τρέχον διάνυσμα ενημέρωσης:

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \\ \theta &= \theta - v_t \end{aligned} \tag{2.8}$$

Σημείωση: Ορισμένες εφαρμογές ανταλλάσουν τα σημεία στις εξισώσεις. Ο όρος ορμής γ ορίζεται συνήθως σε 0.9 ή σε κάποια παρόμοια τιμή.

Ουσιαστικά όταν χρησιμοποιείται η ορμή, είναι σα να πιέζεται μια σφαίρα κάτω σε ένα λόφο. Η σφαίρα συσσωρεύει την ορμή, καθώς κυλάει προς τα κάτω, γρηγορότερα και ταχύτερα στο δρόμο (μέχρι να φτάσει στην τελική της ταχύτητας αν υπάρχει αντίσταση αέρα, δηλαδή $\gamma < 1$). Το ίδιο συμβαίνει με τις ενημερώσεις των παραμέτρων μας: Ο όρος ορμή αυξάνεται για τις διαστάσεις των οποίων οι κλίσεις δείχνουν προς τις ίδιες κατευθύνσεις και μειώνουν τις ενημερώσεις για τις διαστάσεις των οποίων οι κλίσεις αλλάζουν κατευθύνσεις. Ως αποτέλεσμα, επιτυγχάνεται ταχύτερη σύγκλιση και μειωμένη ταλάντωση.

2.10.3 RMSprop (Root Mean Square Propagation)

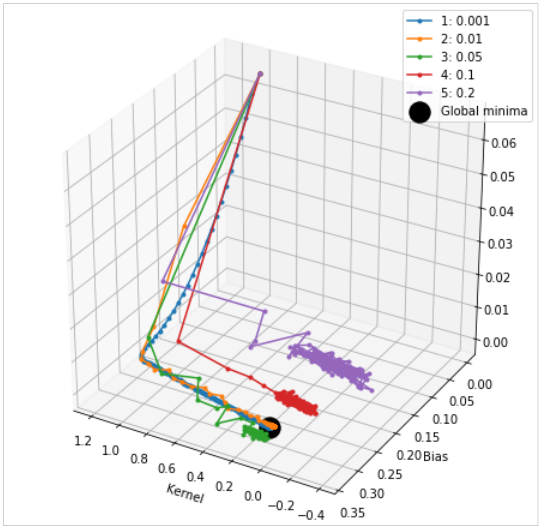
Ο RMSprop βελτιστοποιητής εμφανίστηκε πρώτη φορά στις διαλέξεις από μια σειρά online μαθημάτων στο Coursera πάνω στα νευρωνικά δίκτυα από τον καθηγητή Geoffrey Hinton του Πανεπιστημίου Toronto. Ο Hinton δε δημοσίευσε το RMSprop σε επίσημο ακαδημαϊκό έγγραφο, αλλά χρησιμοποιείται ευρέως στο deep learning. Το ανέπτυξε ώστε να αντιμετωπίσει το πρόβλημα που ήταν σύνηθες όταν χρησιμοποιούνταν το rprop με mini-batches, που είναι ότι τα βάρη θα προσαρμόζονταν ανάλογα με το μέγεθος της gradient για κάθε mini-batch, με αποτέλεσμα να προκύπτουν πολύ μεγάλες αυξήσεις βάρους ή μειώσεις αν τα διαδοχικά mini-batches δεν έχουν παρόμοια gradients. Αυτό βέβαια έρχεται αντιμέτωπο με τα επιθυμητά αποτελέσματα της sgd, που είναι να κάνεις μικρές προσαρμογές στα βάρη και στα biases έτσι ώστε να βαθμονομηθεί ένα νευρωνικό δίκτυο για να λειτουργεί όλο και καλύτερα σε ένα συκγεκριμένο task με κάθε επανάληψη του αλγορίθμου βελτιστοποίησης.

Στο RMSprop, το πρόβλημα που μπορεί να δημιουργηθεί στο rprop αν τα διαδοχικά mini-batches ποικίλλουν κατά πολύ, μετριάζεται χρησιμοποιώντας ένα moving average of the squared gradient για κάθε βάρος. Αυτό σημαίνει ότι η gradient από κάθε mini-batch διαιρείται με τη τετραγωνική ρίζα του MeanSquare, όπου το MeanSquare υπολογίζεται ως:

$$MeanSquare(w, t) = 0.9\ MeanSquare(w,\ t-1) + 0.1\left(\partial E \big/ \partial w(t)\right)^2$$

Αυτή η διαδικασία υπολογίζει κατά μέσο όρο τις gradients σε διαδοχικά mini-batches έτσι ώστε τα βάρη να μπορούν να βαθμονομηθούν με ακρίβεια.

Πιο πρακτικά, ο RMSprop βελτιστοποιητής είναι παρόμοιος με τον αλγόριθμο sgd με momentum. Ο RMSprop στην ουσία περιορίζει τις ταλαντώσεις στην κάθετη κατεύθυνση.



Σχήμα 2.32: *Visualization of the training process for RMSprop optimizer with various learning rates.*

Εδώ όλοι οι βελτιστοποιητές ήταν σε θέση να πλησιάσουν την περιοχή που περιβάλλει το ελάχιστο, αλλά αυτά με το υψηλότερο learning rate (0.2, 0.1, και 0.05) έχουν "κολλήσει" λίγο πιο μακριά από αυτό. Τα αρχικά τους βήματα ήταν πιο μακριά, επομένως χάσανε ελάχιστα το μονοπάτι προς τη σωστή κατεύθυνση και δεν μπορούσαν να το ξαναβρούν.

Κεφάλαιο 3

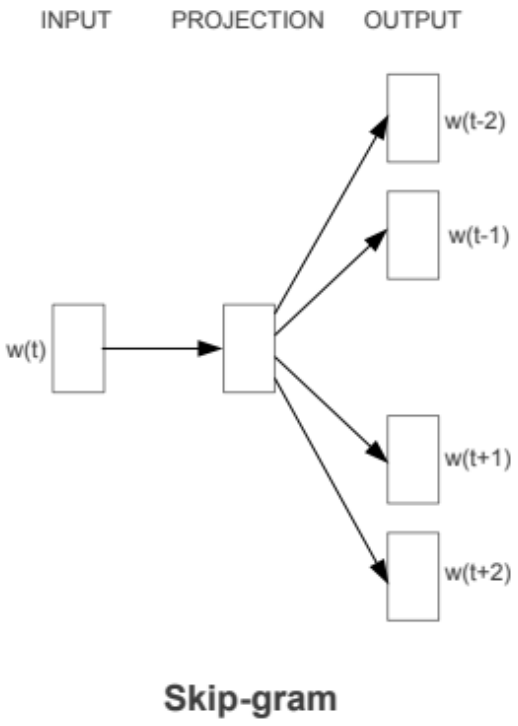
Μεθοδολογία και Ειδικά Μοντέλα

3.1 Skip-Gram

Στην εργασία αυτή μέρος της μεθοδολογίας για την πρόβλεψη failure chains είναι και η εκπαίδευση του μοντέλου Skip-gram για την απόκτηση των word embeddings. Το μοντέλο Skip-gram συνήθως προσπαθεί να πετύχει το ακριβώς αντίθετο από αυτό που προσπαθεί να επιτύχει το μοντέλο CBOW. Προσπαθεί να προβλέψει τις λέξεις περιεχομένου της πηγής (λέξεις γύρω από τη λέξη στόχο) δοθέντος μια λέξη στόχο (κεντρική λέξη).

Έστω ότι έχουμε την πρόταση: **the quick brown fox jumps over the lazy dog**. Αν χρησιμοποιήσουμε το μοντέλο CBOW, παίρνουμε ζευγάρια από **(context window, target word)**, όπου αν εξετάσουμε ένα μέγεθος παραθύρου 2, θα έχουμε παραδείγματα όπως **([quick, fox], brown), ([the, brown], quick), ([the, dog], lazy)** και ούτω καθεξής. Θεωρώντας τώρα ότι ο στόχος του μοντέλου Skip-gram είναι να προβλέψει το περιεχόμενο (context) από τη λέξη στόχο, το μοντέλο συνήθως αναστρέφει τα περιεχόμενα και τους στόχους και προσπαθεί να προβλέψει κάθε λέξη-πλαίσιο(context word) από τη λέξη-στόχο του (target word).

Ως εκ τούτου η εργασία μετατρέπεται στο να προβλέπουμε το πλαίσιο-context, ([quick, fox]) δοσμένης της λέξης στόχου-target brown, ή ([the brown]) δοσμένης της λέξης quick και ούτω καθεξής. Έτσι το μοντέλο προσπαθεί να προβλέψει το (context word) με βάση το (target word).



Σχήμα 3.1: *Η αρχιτεκτονική Skip-gram model (Mikolov et al., 2013a).*

Μοντελοποιούμε την αρχιτεκτονική του Skip-gram ως ένα μοντέλο ταξινόμησης βαθιάς μάθησης, τέτοιο ώστε να παίρνουμε την (target word) ως είσοδο και να προσπαθούμε να προβλέψουμε τις (context words). Αυτό γίνεται πιο πολύπλοκο εφόσον υπάρχουν πολλές λέξεις στο περιεχόμενο μας (context). Απλοποιούμε περαιτέρω τη διαδικασία διασπώντας **κάθε ζεύγος (target, context words)** σε **ζεύγη (target, context)** έτσι ώστε κάθε context να αποτελείται από μία μόνο λέξη. Συνεπώς το αρχικό σύνολο δεδομένων μας μετασχηματίζεται σε ζεύγη όπως (brown, quick), (brown, fox), (quick, the), (quick, brown) και ούτω καθεξής. Αλλά πώς να εκπαιδευτεί ή να επιβλέπει το μοντέλο ώστε να γνωρίζει τι είναι συμφραζόμενο και τι δεν είναι;

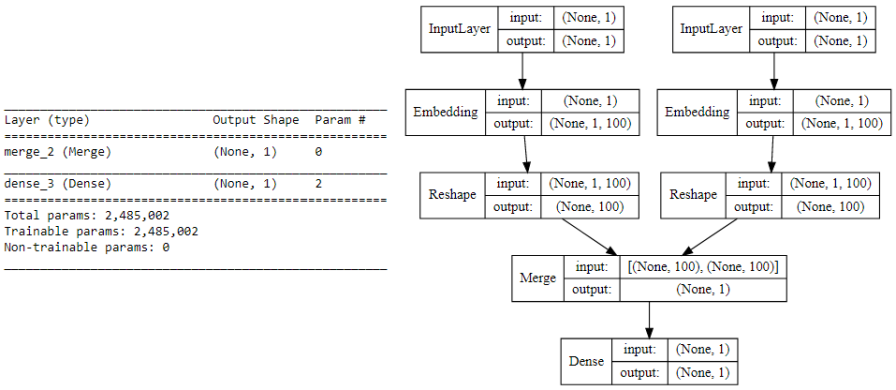
Για αυτό τον λόγο, τροφοδοτούμε το μοντέλο Skip-gram με ζευγάρια **(X,Υ)** όπου X είναι η είσοδος μας και Υ είναι η ετικέτα μας. Το κάνουμε αυτό χρησιμοποιώντας ζευγάρια **[(target, context), 1]** ως θετικά δείγματα εισόδου, όπου (target είναι η λέξη ενδιαφέροντος μας και context είναι η λέξη περιεχομένου που συμβαίνει κοντά στη (target λέξη και η θετική ετικέτα 1 υποδεικνύει ότι αυτό είναι ένα συναφές ζεύγος συμφραζομένων. Επίσης το τροφοδοτούμε με ζευγάρια **[(target, random), 0]** ως αρνητικά δείγματα εισόδου, (target είναι ξανά η λέξη ενδιαφέροντος μας, αλλά το random είναι μια τυχαία επιλεγμένη λέξη από το λεξιλόγιο μας η οποία δεν έχει κανένα περιεχόμενο ή σχέση με τη target λέξη μας). Οπότε η αρνητική ετικέτα 0 υποδεικνύει ότι αυτό είναι άσχετο ζευγάρι όσον αφορά τα συμφραζόμενα. Αυτό το κάνουμε έτσι ώστε το μοντέλο μας να μπορεί να μάθει ποια ζεύγη λέξεων είναι σχετικά όσον αφορά το περιεχόμενο και ποια δεν είναι με συνέπεια να δημιουργηθούν παρόμοια embeddings για σημασιολογικά παρόμοιες λέξεις.

Για την απόκτηση του ζεύγους των λέξεων με την συνάφεια τους χρησιμοποιούμε μια χρήσιμη βοηθητική συνάρτηση από το Keras (KERAS), το skipgrams.

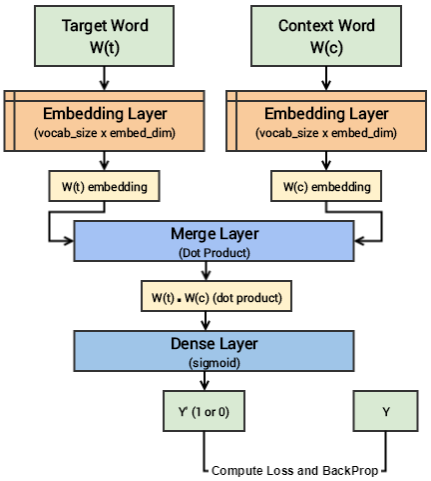
Αυτή η συνάρτηση μετατρέπει μια ακολουθία από δείκτες λέξεων (λίστα από ακεραίους) σε tuples λέξεων της μορφής:

- (word,word in the same window) με ετικέτα 1 (θετικά δείγματα).
- (word,random word from the vocabulary) με ετικέτα 0 (αρνητικά δείγματα).

Για την αρχιτεκτονική βαθιάς μάθησης του μοντέλου skip-gram θα έχουμε σαν εισόδους τα target word και τα context ή random word ζευγάρια. Καθέ ένα από αυτά μεταφέρεται σε ένα δικό του embedding layer (αρχικοποιημένο με τυχαία βάρη) με μέγεθος ($vocab\_size x embed\_size$) τα οποία θα μας δώσουν dense word embeddings για κάθε μία από αυτές τις δύο λέξεις ($1 x embed\_size for each word$). Αφότου αποκτήσουμε τα word embeddings για το target και το context word αντίστοιχα, τα περνάμε σε ένα merge layer στο οποίο υπολογίζουμε το γινόμενο αυτών των δύο διανυσμάτων. Στη συνέχεια, μεταφέρουμε το γινόμενο αυτό σε ένα dense layer με sigmoid activation function το οποίο προβλέπει είτε 1 είτε 0 ανάλογα με το αν το ζεύγος λέξεων σχετίζεται όσον αφορά το περιεχόμενο τους, ή απλά είναι τυχαίες λέξεις (Y'). Το συγκρίνουμε αυτό με την πραγματική ετικέτα relevance (Y), υπολογίζουμε την απώλεια αξιοποιώντας την απώλεια mean_squared_error και εκτελούμε backpropagation για τα σφάλματα για να προσαρμόσουμε τα βάρη στο embedding layer και επαναλαμβάνουμε τη διαδικασία για όλα τα ζεύγη (target, context) για πολλαπλά epochs.



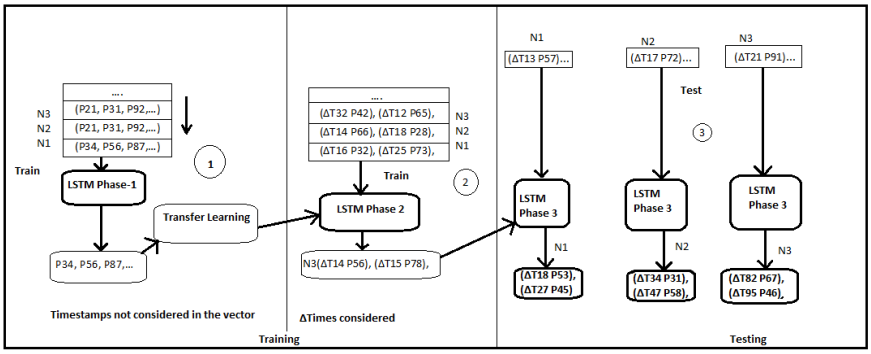
Σχήμα 3.2: *skip_gram model summary.(Sarkar).*



Σχήμα 3.3: *visual depiction of the skip-gram deep learning model(Sarkar).*

3.2 Μεθοδολογία

Σκοπός της διπλωματικής αυτής εργασίας είναι να προβλέψει τα failures του συστήματος ώστε να παρεμποδιστούν τα failures των jobs καθώς και να υπολογίσει τα lead times. Αυτό το πετυχαίνουμε με την πρόβλεψη πιθανών failure chains.



Σχήμα 3.4: *LSTM φάσεις.*

- **Φάση 1**

Η πρώτη φάση περιλαμβάνει την εκπαίδευση για failure chains από ακατέργαστα δεδομένα. Τα ακατέργαστα logs από το μηχάνημα BlueGene της IBM περιέχουν φράσεις με ανωμαλίες διάσπαρτες μαζί με σημαντικό αριθμό θορύβου και ασφαλών γεγονότων. Ένα log μήνυμα έχει ένα timestamp T1, μια φράση συμβάντος P1 και έναν κόμβο N1. Ομοίως, πολλές αντίστοιχες timestamped φράσεις, δηλαδή φράσεις με χρονική σήμανση, θα αντιστοιχούν σε συγκεκριμένους κόμβους.

Η παρακολούθηση των ids των κόμβων συμβάλλει στη διατήρηση της συγκεκριμένης θέσης όπου λαμβάνει χώρα η βλάβη-σφάλμα.

Εκπαιδεύω το σύνολο δεδομένων με βάση τους κόμβους στη φάση 1. Με άλλα λόγια, τα logs από κάθε κόμβο "συγκολλούνται" και τροφοδοτούνται στο ίδιο LSTM. Αυτό έχει δύο πλεονεκτήματα. Καταρχάς, δεν υπάρχει καμία επιβάρυνση για την αποθήκευση του node id και την επεξεργασία του στο διάνυσμα, κάτι το οποίο εξοικονομεί μνήμη και κόστος υπολογισμού. Δεύτερον, για να μπορέσει το μοντέλο μας να μάθει τα patterns από τα failure chains που παρατηρούνται, η ταυτότητα του κόμβου δεν έχει καμία συνέπεια, αντίθετα, ποιες φράσεις εμφανίζονται σε μια ακολουθία που οδηγεί σε διάφορα node failures είναι αυτά που απαιτούνται. Σ'αυτή τη φάση δεν προβλέπεται χρόνος, αλλά ταξινομούμε τα log μηνύματα με βάση το timestamp τους γι'αυτό το λόγο. Η σειρά των φράσεων έχει σημασία εδώ, όχι η χρονική διαφορά τους.

Στη φάση 1 χρησιμοποιούνται τα phrase ids ως διανύσματα. Κάθε φράση-event phrase στη συνέχεια διαχωρίζεται σε στατικό και δυναμικό περιεχόμενο, έτσι ώστε να προσδιορισθεί το σταθερό μέρος της φράσης από το μεταβλητό μέρος αυτής. Έτσι λοιπόν η P1 φράση διασπάται στο στατικό μέρος και στο μεταβλητό μέρος. Το δυναμικό μέρος απορρίπτεται. Παραδείγματα του διαχωρισμού των φράσεων σε στατικό και δυναμικό μέρος αποτελούν τα παρακάτω:

static	dynamic
DDR Correctable Error Summary	count=8713 MCFIR error status: [POWERBUS_WRITE_BUFFER_CE] This bit is set when a PBUS ECC CE is detected on a PBus write buffer read op;
L1P Correctable Error Summary	count=27355 cores=0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 L1P_ESR : [ERR_RELOAD_ECC_X2] correctable reload data ECC error;
DDR Correctable Error Summary	count=236 MCFIR error status: [MEMORY_CE] This bit is set when a memory CE is detected on a non-maintenance memory read op;
L1P Correctable Error Summary	count=2 cores=3,13 L1P_ESR : [ERR_RELOAD_ECC_X2] correctable reload data ECC error;

Σχήμα 3.5: *Παραδείγματα στατικού και δυναμικού μέρους φράσεων.*

Όταν όλα τα σταθερά μέρη εξαχθούν, κωδικοποιούνται σε ένα μοναδικά αναγνωρίσιμο αριθμό. Αυτές οι φράσεις είναι συζευγμένες οντότητες πολλών λέξεων (για παράδειγμα ddrcorrectableerrorsummary). Οπότε θα έχουμε μια ακολουθία κωδικοποιημένων φράσεων, δηλαδή για παράδειγμα {45, 67, 89, 40}, για τον κόμβο N1. Όμως τα LSTM δεν μπορούν να κατανοήσουν τη σημασιολογική ή συντακτική σχέση τους σε αυτή τη διακριτή μορφή.

Σ'αυτό το σημείο έρχονται τα vector space models με κατανεμημένη αναπαράσταση, τα οποία συμβάλλουν στη δημιουργία σημασιολογικού συσχετισμού. Σ'αυτές τις κωδικοποιημένες φράσεις λοιπόν, τους δίνεται στη συνέχεια διανυσματική μορφή χρησιμοποιώντας τα word embeddings. Τα embeddings όπως αναφέρω και στην ενότητα 2.2 είναι καθορισμένα πλαίσια που ελέγχουν τι εμφανίζεται πριν και μετά από ένα target event phrase σε μια σειρά από γεγονότα. Στη μελέτη αυτή χρησιμοποιώ το skip-gram για να εξάγω τα word embeddings, το οποίο και περιγράφω στην ενότητα 3.1. Άρα για κάθε κωδικοποιημένη φράση στα δεδομένα έχω ένα σύνολο phrase embeddings που αναφέρονται στο περιεχόμενο τους. Επομένως χρησιμοποιώντας τα embeddings και τι εμφανίζεται πριν και μετά από ένα στόχο, το vector space δημιουργείται.

Άρα στην ουσία για μια ακολουθία {45, 67, 89, 40, 89, 102}, έχουμε τη πιθανότητα ότι το 40 παρατηρείται αφότου εμφανιστεί η ακολουθία {45, 67, 89} και πριν εμφανιστεί η ακολουθία {89, 102}. Αυτό συμφωνεί με τη διανυσματική αναπαράσταση που ανέφερα παραπάνω. Αυτά τα διανύσματα στη συνέχεια τροφοδοτούνται σε stacked lstm χρησιμοποιώντας δύο κρυφά layers για να πραγματοποιήσουν 3-step πρόβλεψη (να προβλέψουν τις 3 επόμενες φράσεις). Τα μεγέθη παραθύρου που χρησιμοποιούνται είναι 15 και 3, για να ληφθεί, αντίστοιχα, υπόψη ο αριθμός των φράσεων αριστερά και δεξιά μιας συγκεκριμένης φράσης στόχου. Το μοντέλο μας στη φάση 1 χρησιμοποιεί τον βελτιστοποιητή sgd με categorical cross-entropy loss όπως αναφέρθηκε και στις προηγούμενες ενότητες. Η φάση 1 είναι semi-supervised και παράγει εκπαιδευμένες ακολουθίες των διανυσμάτων των φράσεων.

Phrase Labeling. Στη συνέχεια φιλτράρω τα logs αφότου έχω κρατήσει το σταθερό τους μέρος ακολουθώντας την παραπάνω διαδικασία και τα κατηγοριοποιώ σε τρεις κατηγορίες: **safe, error** και **unknown**. Οι Safe αντιπροσωπεύουν τις "ασφαλείς" φράσεις, οι οποίες σε καμία περίπτωση δε συσχετίζονται με καμία ανωμαλία του συστήματος. Οι Error αναφέρονται σ'αυτές τις φράσεις, οι οποίες είναι σίγουρα ένδειξη κάποιας ανωμαλίας. Η Unknown ετικέτα δίνεται σε αυτές τις φράσεις που μπορεί να είναι ή να μην είναι ένδειξη κάποιας ανωμαλίας.

Σημαντικό είναι να σημειωθεί ότι η επισήμανση μιας φράσης ως Error δε σημαίνει ότι θα είναι πάντοτε μέρος των node failures, αλλά μπορεί να είναι ή μπορεί να μην είναι μέρος της αποτυχίας του κόμβου. Ωστόσο, αυτές οι φράσεις θα είναι είτε τερματικά μηνύματα, είτε θα είναι μηνύματα σημαντικής δυσλειτουργίας υλικού ή λογισμικού

που εντοπίζεται στα Linux logs. Επίσης όπως ανέφερα και στην ενότητα 2.1 δεν παίρνουμε υπόψη τα severity levels από τα logs. Αυτό διότι η άμεση ταξινόμηση των logs μηνυμάτων που βασίζεται σε περιστασιακά εμφανιζόμενα tags δεν είναι αποτελεσματική για long-term sensitive δεδομένα, καθώς πολλά μη-κρίσιμα μηνύματα θα μπορούσαν να είναι ένας καλύτερος δείκτης για failures στον χρόνο. Εξάλλου, όπως ήδη αναφέρθηκε, κάποια φαινομενικά "ασφαλή" γεγονότα σε ένα context μπορεί να οδηγήσουν σε fatal events σε ένα άλλο context. Ως εκ τούτου, θεωρούμε φράσεις ανεξάρτητα από flags/severity levels.

Παραδείγματα φράσεων που έχουν κατηγοριοποιηθεί σε Safe, Error, Unknown αποτελούν τα παρακάτω:

Safe	Unknown	Error
ddrcorrectableerrorsummary	memorycontrollerinitializationwarning	badphywasdetected
l1pcorrectableerrorsummary	warningmc1rank1	unabletoupdatethelcddisplay
serviceaction6585restartednodedcar13	encounteredanexception	Kernelunexpectedoperation (Indicator – Panic)
serviceaction6585startedtoservicer13	warningmc0rank1	cfamlivelockbusterfailure (Indicator – Hardware(H/W))

Σχήμα 3.6: *Παραδείγματα φράσεων με βάση την ετικετοποίησή τους. Οι φράσεις παρουσιάζονται ως συζευγμένες οντότητες πολλών λέξεων. Στην τρίτη στήλη παρουσιάζονται κάποιες Error φράσεις, δύο από τις οποίες αποτελούν και τερματικά μηνύματα που σηματοδοτούν failures. Η μία ανήκει στην κλάση Hardware, καθώς αναφέρεται σε τέτοια errors, ενώ η άλλη σε Panic, καθώς αναφέρεται σε kernel panic μηνύματα.*

Μετά την κατηγοριοποίηση έχουμε φράσεις με Safe(S), Error(E) ή Unknown(U) ετικέτες. Οι safe φράσεις εξαλείφονται καθώς το πρωταρχικό μας ενδιαφέρον είναι οι error και unknown φράσεις.

Η ετικετοποίηση των φράσεων σκόπιμα δε γίνεται πριν από τη διανυσματική αναπαράσταση, δεδομένου ότι η εκπαίδευση γίνεται πιο ισχυρή με το θόρυβο. Επιπλέον, οι δείκτες για βλαβερά ή ασφαλή γεγονότα δεν είναι a priori για τη μη επιβλεπόμενη μάθηση. Η ετικετοποίηση γίνεται για τη βελτιστοποίηση του κόστους υπολογισμού στη φάση 2. Συνεπώς, μια ακολουθία γεγονότων που οδηγεί σε node failure σχηματίζεται χρησιμοποιώντας ετικετοποιημένες φράσεις Unknown (U) και Error (E), εφόσον και τα τερματικά μηνύματα, που υποδεικνύουν ότι ένας κόμβος έχει πέσει, είναι γνωστά.

- **Φάση 2**

Στη φάση 1 η παρουσία του θορύβου έκανε τον υπολογισμό του ΔT ανέφικτο, δεδομένου ότι θα πρέπει να θεωρούμε φράσεις με ανωμαλίες που οδηγούν σε node failures χωρίς να υπάρχουν διασκορπισμένες safe φράσεις. Σε αυτή τη φάση λοιπόν, φάση 2, διαχωρίζουμε τις φράσεις που σχηματίζουν failure chains από τις υπόλοιπες (που δεν αποτελούν μέρος μιας failure chain) και υπολογίζουμε τις χρονικές διαφορές ανάμεσα σ'αυτές τις φράσεις της αλυσίδας για την πρόβλεψη του lead time.

Πώς γίνεται ο υπολογισμός **ΔT**;

Ο στόχος μας είναι να προβλέψουμε το lead time που οδηγεί σε ένα node failure. Σε μια ακολουθία μηνυμάτων που οδηγεί σε failure, υπάρχουν "ανώμαλα" μηνύματα, πριν από τα τερματικά μηνύματα. Η "αποτυχία"-failure που εκδηλώνεται, υποδεικνύεται από την ανώτερη σε σειρά διάταξης time-series. Επομένως, ταξινομούμε τα δεδομένα σε φθίνουσα σειρά των timestamps και υπολογίζουμε τα ΔT, τα οποία είναι η αθροιστικά χρονική διαφορά ανάμεσα στη τρέχουσα φράση και τη τελευταία

φράση χρονικά (υψηλότερη σειρά) στην ακολουθία. Στην υψηλότερη timestamped φράση της ακολουθία έχει εκχωρηθεί το ΔT=0, εφόσον δεν έχει μείνει καμία άλλη φράση στην ακολουθία για να υπολογισθεί η χρονική διαφορά.

#	Timestamp	Label	Phrase Vector
P1	03:59:58.466 (T1)	U	ΔT1=07.822, P1
P2	03:59:59.543 (T2)	U	ΔT2=06.745, P2
P3	04:00:00.477 (T3)	U	ΔT3=05.811, P3
P4	04:00:01.706 (T4)	E	ΔT4=04.582, P4
P5	04:00:01.731 (T5)	E	ΔT5=04.557, P5
P6	04:00:06.288 (T6)	E	ΔT6=00:000, P6

Σχήμα 3.7: *Παράδειγμα Failure Chain.(Das et al., 2018).*

Στο σχήμα 3.7, για παράδειγμα, στο ΔT6 έχει εκχωρηθεί η τιμή 0. Στη συνέχεια, υπολογίζουμε τα ΔT αφαιρώντας τα timestamps της κάθε φράσης από το T6 timestamp στην failure chain αντίστοιχα. Άρα έχουμε T6-T5, T6-T4, T6-T3, T6-T2, T6-T1 και μ'αυτόν τον τρόπο λαμβάνουμε τις χρονικές διαφορές σε δευτερόλεπτα (7.822, 6.745, 5.811, 4.582, 4.557, 0).

Στη συνέχεια εκπαίδευουμε ένα καινούριο μοντέλο αποτελούμενο πάλι από stacked lstm δύο κρυφών layers αυτή τη φορά με failure chains ώστε να μάθουν τα διάφορα ΔT με τα ids των φράσεων και τελικά να προβλέψουμε τι χρόνους θα περιμένουμε στο μέλλον. Η είσοδος στο LSTM στη φάση 2 θα είναι ένα διάνυσμα 2 καταστάσεων, που θα περιέχει το ΔT και το id της αντίστοιχης φράσης, όπως φαίνεται και στη τελευταία στήλη του σχήματος 3.7. Αυτή η εκπαίδευση αποτελεί ένα multivariate time-series πρόβλημα, το οποίο περιμένει οι προβλεπόμενες τιμές να είναι αρκετά κοντά με τις πραγματικές τιμές που υπάρχουν στο σύνολο εκπαίδευσης. Στη φάση 2 το LSTM τροφοδοτείται με διανύσματα με history size 10 και εκτελούν 1-step πρόβλεψη για κάθε σειρά με 2 κρυφά layers. Επιπλέον χρησιμοποιώ transfer learning από τη φάση 1 στη φάση 2, ώστε το εκπαιδευμένο μοντέλο της φάσης 1 να αποτελέσει σημείο εκκίνησης για την εκπαίδευση του μοντέλου της φάσης 2, με αποτέλεσμα και τη ταχύτερη σύγκλιση συγκριτικά με τη τυχαία αρχικοποίηση.

Η μεθοδολογία που ακολουθείται για τις φάσεις 1 και 2 είναι εκπαίδευση των διανυσμάτων συνενωμένων, δηλαδή τα διανύσματα αντιστοιχούν στον έναν κόμβο μετά τον άλλον. Άρα τα δύο μοντέλα μαθαίνουν failure sequences από διαφορετικούς κόμβους διαδοχικά. Η διαφορά είναι στο περιεχόμενο του διανύσματος εισόδου για κάθε log message που σχετίζεται με έναν κόμβο. Ενώ η φάση 1 περιέχει μόνο τα ids των φράσεων, η φάση 2 περιέχει τις χρονικές διαφορές ανάμεσα στις φράσεις, μαζί με τα ids αυτών. Κατά τη φάση 2 δηλαδή με δεδομένο τα ids των φράσεων και τις χρονικές διαφορές ανάμεσα τους για προηγούμενο χρόνο, προβλέπουμε τα ids των φράσεων τις επόμενες ώρες.

Στη φάση 3 από την άλλη, εντοπίζονται failure που βασίζονται σε προηγούμενη εκπαίδευση και εφαρμόζονται στα καινούρια test δεδομένα (διαφορετικά από αυτά της εκπαίδευσης). Ωστόσο, τα διανύσματα που τροφοδοτούνται στο LSTM στη φάση 3 προέρχονται από συγκεκριμένο κόμβο (μη συνενωμένο). Έτσι μαθαίνουμε πρώτα από όλους τους κόμβους και χρησιμοποιούμε αυτή τη συνολική μάθηση για την ανίχνευση failures ανά μεμονωμένο κόμβο κατά τη διάρκεια του testing και του inference στη φάση 3.

- **Φάση 3**

Σ'αυτή τη φάση, το LSTM χρησιμοποιείται στα test δεδομένα για την επικύρωση (validate) των εκπαιδευμένων failure chains από τη φάση 2. Τα test δεδομένα επεξεργάζονται για να σχηματίσουν κωδικοποιημένα διανύσματα με χρονικές διαφορές και φράσεις αντίστοιχα με το πώς είπαμε στη φάση 2. Βέβαια εδώ έχουμε όλων των ειδών φράσεις, safe, error, unknown. Στη φάση 3 θα πρέπει να παρακολουθούμε το id του κόμβου, έτσι ώστε να ξέρουμε ποιος κόμβος αναμένεται να πέσει, με πόσο χρόνο έχει μείνει πριν από οποιαδήποτε failure. Εδώ πέρα λοιπόν τα διανύσματα δε συνδέονται μεταξύ των κόμβων όπως συμβαίνει στις φάσεις 1 και 2. Αντίθετα, τα logs από κάθε κόμβο διαβιβάζονται σε ένα ίδιο εκπαιδευμένο LSTM. Άρα στην ουσία περνάνε σε κάθε εκπαιδευμένο LSTM ακολουθίες από διανύσματα που περιέχουν ΔT και id φράσεων που αφορούν τους κόμβους - αυτά είναι τα test δεδομένα.

Έστω λοιπόν ότι έχουμε 100 διαφορετικούς κόμβους στο σύνολο δοκιμής (test), τότε θα έχουμε 100 "πακέτα" εισόδου για το κάθε LSTM (αν πχ είναι μεγέθους M, τότε θα έχουμε M 2-state διανύσματα στο κάθε "πακέτο" εισόδου). Αυτό αντιπροσωπεύει τα node ids με έναν τρόπο που εξοικονομεί το κόστος και συμμορφώνεται και στη μορφή του διανύσματος εισόδου. Στην ουσία γίνεται evaluation της φάσης 2 στα καινούρια test δεδομένα. Το LSTM προβλέπει το επόμενο δείγμα για αυτά τα test δεδομένα και υπολογίζει το MSE. Καθώς γίνεται η επικύρωση των φράσεων σε μια ακολουθία, η πρόβλεψη εάν θα συμβεί node failure καθορίζεται από την προσπάθεια να βρεθεί ένα close match στην πραγματική target failure chain.

Χρησιμοποιούμε ένα κατώφλι threshold για να συμπεραίνουμε τα node failures. Με άλλα λόγια, όταν το LSTM αποκτά MSE<= "κατώφλι", μπορούμε να θεωρήσουμε αυτά τα αποτελέσματα ύποπτα για να ελέγξουμε για failure. Το κατώφλι που χρησιμοποιούμε είναι το 0.0145 και το βρίσκουμε μετά από πειραματική διαδικασία. Συγκεκριμένα, πάνω από 0.0145 MSE στα test δεδομένα εξάγει αλυσίδες που είναι πολύ διαφορετικές από αυτές που βρίσκονται στις εκπαιδευμένες failure chains. Έστω δηλαδή ότι έχουμε την πρόβλεψη [P1, P5, P6] και η failure chain είναι [P1, P5, P7], θεωρείται ως failure και την επαληθεύουμε με τα πραγματικά δεδομένα. Οπότε μετά μπορούμε να εξάγουμε τα ΔT από το διάνυσμα και να υπολογίσουμε το lead time.

Κεφάλαιο 4

Πειραματική Διαδικασία και Αποτελέσματα

4.1 Εργαλεία

Για την υλοποίηση της παρούσας διπλωματικής εργασίας έγινε χρήση του Keras το οποίο τρέχει πάνω από το TensorFlow και άλλες βιβλιοθήκες. Το Keras περιέχει πολλές υλοποιήσεις των δομικών στοιχείων που χρησιμοποιούνται συνήθως σε νευρωνικά δίκτυα, όπως layers, losses, activation functions, optimizers και metrics. Οι γραφικές παραστάσεις έγιναν με το matplotlib. Όλος ο κώδικας της διπλωματικής έγινε με python3.

4.2 LogAider

Αρχικά γίνεται χρήση ενός εργαλείου που έχει αναπτυχθεί από το Argonne National Laboratory σε συνεργασία με το Πανεπιστήμιο του Illinois για την εύρεση πιθανών συσχετίσεων των logs από HPC (Di et al., 2017). Βασίζεται σε logs όπως τα RAS και τα JOB που αναφέρθηκαν στην ενότητα 2.1. Το εργαλείο αυτό, το οποίο δημιουργήθηκε και χρησιμοποιήθηκε για τα logs του Mira, μπορεί να αποκαλύψει τρεις τύπους πιθανής συσχέτισης: across-field, spatial και temporal.

Η **across-field** συσχέτιση στοχεύει στην κατανόηση της σημαντικότητας των πεδίων και των συνδυασμών τους κατά τον προσδιορισμό των χαρακτηριστικών τους μηνυμάτων, όπως το severity του μηνύματος και η κατηγορία του. Η συσχέτιση αυτή έχει να κάνει με τον υπολογισμό της κατανομής της prior probability κάποιου συγκεκριμένου γεγονότος, όπως για παράδειγμα κάποιας αποτυχίας του συστήματος. Το LogAider επίσης μπορεί να αποκαλύψει στατιστικές συσχετίσεις (όπως posterior probability distribution-δεσμευμένη πιθανότητα), μεταξύ διαφορετικών ομάδων από συνδυασμένα πεδία ή μετρικές. Έτσι για παράδειγμα το LogAider μπορεί να απαντήσει στο ερώτημα ποια είναι η πιθανότητα το σύστημα να εμπίπτει στον τομέα του σφάλματος λογισμικού, για ένα συγκεκριμένο component (όπως για παράδειγμα control node kernel) και για συγκεκριμένο severity level (όπως Fatal). Οπότε για αυτό το είδος συσχέτισης χρησιμοποιείται ένα Bayesian μοντέλο πιθανότητας.

Στην **spatial correlation** το LogAider μπορεί να εξορύξει χωρικές συσχετίσεις με έναν βελτιστοποιημένο αλγόριθμο clustering, πάνω σε μια τοπολογία δικτύου Torus. Αυτό το επιτυγχάνει με την χρήση ενός K-means clustering αλγορίθμου με ένα βελτιωμένο αριθμό από clustering σύνολα. Συγκεκριμένα, βελτιστοποιείται

ο αριθμός των συστάδων, συγχωνεύοντας τα κοντινά τους κέντρα επαναληπτικά προκειμένου να επιτευχθεί το βέλτιστο αποτέλεσμα ομαδοποίησης.

Όσον αφορά την **temporal** συσχέτιση έχει σχεδιαστεί για να εξάγει πιθανή συσχέτιση γεγονότων στη διάσταση του χρόνου. Ειδικότερα, ο στόχος είναι η αναζήτηση ενός συνόλου γεγονότων που έχουν πολύ σημαντική ομοιότητα και επίσης εμφανίζονται πολύ κοντά στον χρόνο (ή εντός μιας ανεκτής περιόδου που καθορίζεται από τον χρήστη). Αυτή η συσχέτιση είναι αρκετά κοντά στην ανθρώπινη συμπεριφορά, καθώς εξερευνά χρονικές συσχετίσεις όχι μόνο βασιζόμενες στα timestamps, αλλά και στην ομοιότητα δύο γεγονότων σε πολλαπλά επίπεδα, όπως messageID, location, category. Επίσης η συνάρτηση ομοιότητας για το κάθε πεδίο είναι προσαρμόσιμη με βάση την ανάγκη. Έτσι για παράδειγμα η ομοιότητα για το πεδίο του messageID, καθώς αυτό στο μηχάνημα Mira είναι 8 ψηφία (πχ 00080012) θα προκύπτει ως εξής:

- 0, αν τα πρώτα 4 ψηφία είναι διαφορετικά.
- 0.5, αν τα πρώτα 4 ψηφία είναι ίδια και τα 4 τελευταία διαφορετικά.
- 1, αν όλα τα 8 ψηφία είναι ίδια.

Το εργαλείο αυτό είναι γραμμένο σε java και όπως αναφέρεται και παραπάνω είναι εργαλείο που εφαρμόστηκε πάνω στο Mira. Το LogAider παρόλο που προσφέρει και αρχεία, τα οποία προορίζονται για να τα αλλάζει ο χρήστης όσον αφορά το "schema" του κάθε συστήματος, παραμένει ένα εργαλείο αρκετά συνδεδεμένο με την τοπολογία του Mira (5D torus topology) κάτι που το κάνει πιο δύσχρηστο για άλλα μηχανήματα με διαφορετική τοπολογία, όπως για παράδειγμα το Shasta Cray System, το οποίο λειτουργεί με το τελευταίας γενιάς interconnect, με το κωδικό-όνομα "Slingshot" (cray).

Επίσης εκτέλεσή του ακόμα και σε διαφορετικές χρονιές από τα logs του ίδιου του μηχανήματος, απαιτεί πολλές αλλαγές μέσα στον κώδικα. Αυτό συμβαίνει γιατί η κάθε χρονιά φαίνεται να έχει διαφορετικά πεδία.

Όλα αυτά το καθιστούν ενά καλό εργαλείο από τη μία, αλλά με αρκετούς περιορισμούς ως προς την εύκολη χρήση του ειδικότερα σε πολύ διαφορετικές τοπολογίες από αυτές των Blue Gene συστημάτων.

4.3 Ανάλυση - Απεικόνιση

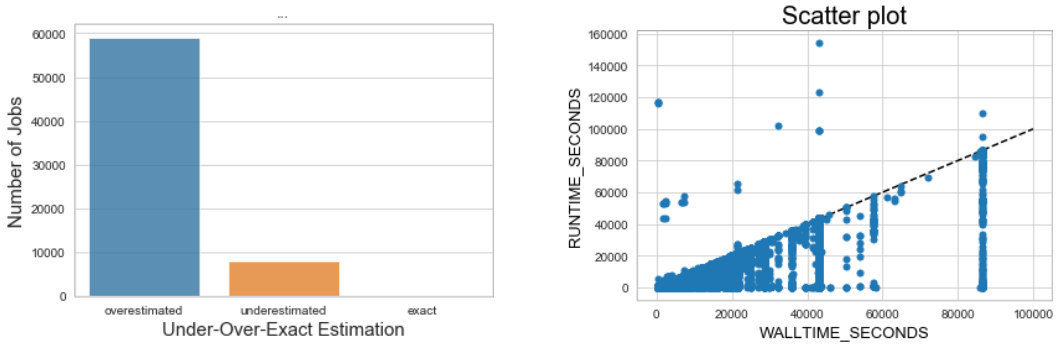
Όπως αναφέρθηκε στην ενότητα 1.7 αρχικά πραγματοποιήθηκε μια απεικόνιση και ανάλυση των δεδομένων των logs που αναφέρονται στα jobs του Mira. Υπάρχουν 125 projects, 298 χρήστες που υπέβαλλαν jobs, 67000 jobs έχουν υποβληθεί και περίπου το 71% των jobs έχουν εξέλθει κανονικά από το σύστημα κατά την περίοδο την οποία έχω αναλύσει σ'αυτή την εργασία.

Basic Information of the job log
Period April 9, 2013 – December 31, 2013
Total number of jobs 67000
Total number of projects 125
Total number of users 298
Percentage of jobs exit system normally 71.004477611940%

Σχήμα 4.1: *Βασική πληροφορία των job logs*

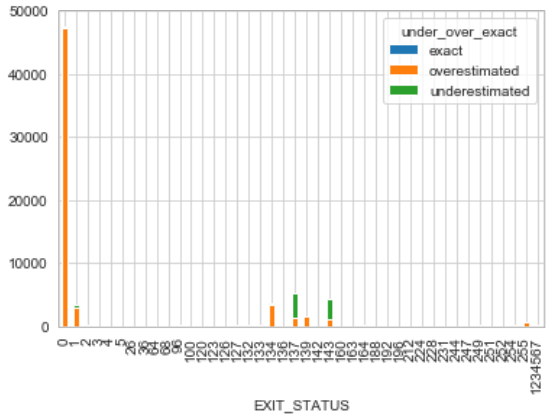
Η πληροφορία από τα logs είναι χρήσιμη για την ανάλυση δεδομένων και για μια νέα αξιολόγηση του χρονοπρογράμματος. Η ανάλυση αυτή βοηθάει τους διαχειριστές να μαθαίνουν την κατάσταση των μηχανών και τα προβλήματα σε ένα σύστημα.

Χαρακτηριστικά διάγραμματα αυτής της ανάλυσης είναι τα παρακάτω:



Σχήμα 4.2: *Αριστερά βλέπουμε ότι τα περισσότερα jobs έχουν υπερεκτιμηθεί. Αντίστοιχη πληροφορία προκύπτει και από το διάγραμμα στα δεξιά.*

Συμπεραίνουμε από τα παραπάνω διαγράμματα ότι τα περισσότερα jobs έχουν υπερεκτιμηθεί. Στο scatter plot έχω στον οριζόντιο άξονα το walltime seconds [1] και στον κάθετο άξονα το runtime seconds [2]. Παρατηρούμε ότι τα περισσότερα jobs είναι κάτω από την γραμμή $y = x$, άρα κάνουν λιγότερο χρόνο από αυτόν που θεωρήθηκε ότι θα χρειαστούν, ενώ τα jobs που είναι πάνω από την γραμμή χρειάζονται περισσότερο χρόνο από αυτόν που θεωρήθηκε ότι θα χρειαστεί. Επιπλέον πληροφορία μας δίνεται και από το παρακάτω διάγραμμα, του αριθμού των jobs συναρτήσει του exit status ομαδοποιημένα με βάση το άμα αυτά έχουν υπερεκτιμηθεί ή όχι.



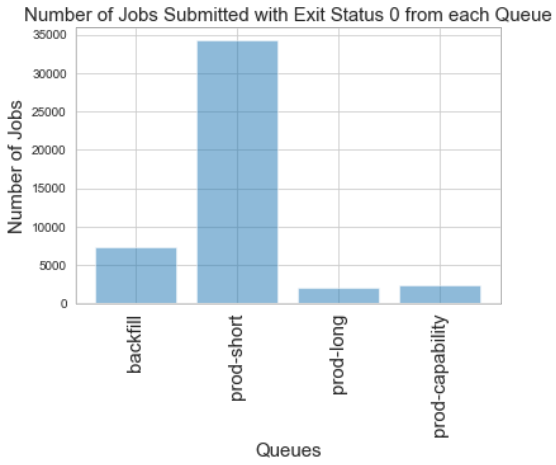
Σχήμα 4.3: *Τα περισσότερα jobs που έχουν exit status 0-δηλαδή αυτά που έχουν τερματίσει με επιτυχία-έχουν υπερεκτιμηθεί.*

Ακόμα, ενδιαφέρον παρουσιάζει η πληροφορία που λαμβάνω από τις πιο συνηθισμένες ουρές του μηχανήματος [3] (backfill, prod-short, prod-long, prod-capability) σχετικά με το πόσα jobs εκτελούνται με επιτυχία (exit code 0).

[1] Ο χρόνος που έχει ζητηθεί για το job, μπορεί να είναι διαφορετικός από τον χρόνο εκτέλεσης.

[2] Ο χρόνος που το job έχει εκτελεστεί.

[3] Γενικά όλα τα jobs που δεν αποτελούν μέρος από κάποια κράτηση που έκανε κάποιος χρήστης υποβάλλονται στις ουρές. Αυτά τα jobs εκτελούνται στη συνέχεια από έναν δρομολογητή ο οποίος εφαρμόζει πολιτική προγραμματισμού για να τα τοποθετήσει σε ουρές ανάλογα με το μέγεθος και το ζητούμενο χρόνο.



Σχήμα 4.4: *Στο διάγραμμα αυτό φαίνεται ο αριθμός των jobs που έχουν γίνει submit και έχουν exit status 0, συναρτήσει των πιο συνηθισμένων ουρών. Όπως μπορούμε να δούμε τα περισσότερα jobs με επιτυχία βρίσκονται στην ουρά prod-short και μετά στη backfill.*

Οι ουρές που αναφέρω στο παραπάνω διάγραμμα έχουν την παρακάτω ερμηνεία:

- Prod-capability: Κάθε job που ζητά (request) ίσους ή περισσότερους από το 20% των κόμβων, δρομολογείται σ'αυτή την ουρά, μέχρι το μέγιστο χρόνο εκτέλεσης των 24 ωρών. Αυτή η ουρά έχει πρόσβαση σε όλους τους κόμβους του μηχανήματος. Ιδανικά, κάθε job θα πληρούσε αυτά τα κριτήρια, δεδομένου ότι αυτά τα "capability" jobs είναι η κύρια αποστολή του ALCF.

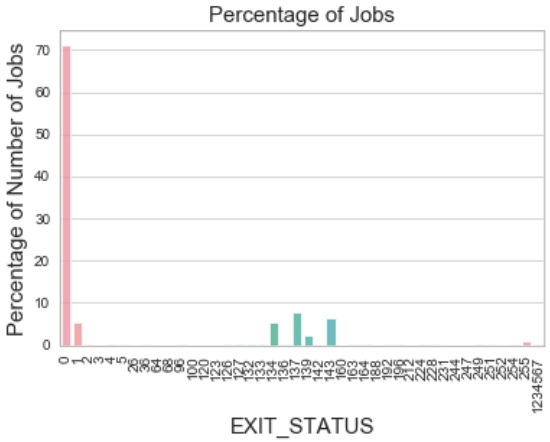
- Prod-short: Κάθε job που απαιτεί λιγότερους από το 20% των κόμβων και έχει ζητήσει walltime μικρότερο ή ίσο από 6 ώρες, δρομολογείται σ'αυτή την ουρά. Αυτή η ουρά έχει πρόσβαση επίσης σε όλους τους κόμβους του μηχανήματος. Αυτά δεν είναι capability jobs, αλλά είναι σχετικά βραχυπρόθεσμα - short, έτσι θα μπλοκάρουν τα μεγαλύτερα jobs για μικρότερο χρονικό διάστημα.

- Prod-long: Κάθε job που απαιτεί λιγότερους από το 20% των κόμβων και έχει ζητήσει (requested) wall-time μεγαλύτερο από 6 ώρες, δρομολογείται στη συγκεκριμένη ουρά. Αυτή η ουρά έχει πρόσβαση μόνο στο 1/3 των κόμβων του μηχανήματος. Αυτά τα jobs είναι μικρά σε μέγεθος και μεγάλα σε διάρκεια, έτσι μπορούν να μπλοκάρουν μεγαλύτερα jobs για ένα σημαντικό χρονικό διάστημα και επομένως είναι πολύ ενοχλητικά στον επιδιωκόμενο στόχο μας να τρέχουμε μεγαλύτερα capability jobs. Περιορίζοντάς τα στο 1/3 του μηχανήματος, εγγυόμαστε ότι έχουμε τουλάχιστον ένα 32K partition και όλα τα sub-partitions διαθέσιμα στην ουρά prod-capability (Allcock et al., 2017).

- Backfill: Επιτρέπεται σε άλλα jobs να χρησιμοποιούν δεσμευμένα job slots, αρκεί τα άλλα αυτά jobs να μην καθυστερούν την έναρξη κάποιου άλλου job. Στην ουσία, ο backfill προγραμματισμός επιτρέπει την χρήση των δεσμευμένων θέσεων job (reserved job slot) από μικρά jobs τα οποία μπορούν να εκτελεστούν και να ολοκληρωθούν πριν ξεκινήσει το μεγάλο job. Έτσι αυξάνεται και η αξιοποίηση των πόρων. Το backfill λειτουργεί πιο αποτελεσματικά όταν όλα τα jobs σε ένα cluster έχουν όριο εκτέλεσης. Τα jobs με μικρότερο όριο εκτέλεσης έχουν περισσότερες πιθανότητες να προγραμματιστούν ως backfill jobs, άρα

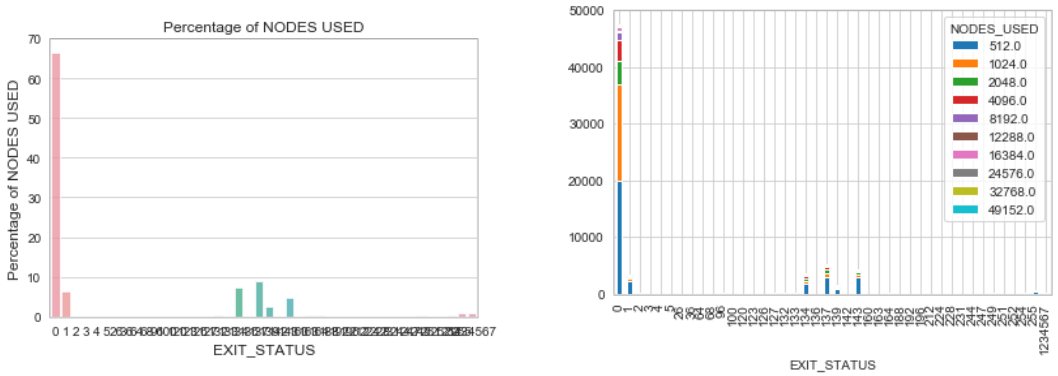
όπως είπαμε μόνο τα jobs με όριο εκτέλεσης που λήγει πριν από την ώρα έναρξης του μεγάλου job επιτρέπεται να χρησιμοποιήσει τη δεσμευμένη θέση (IBM).

Ακόμα ένα σημαντικό διάγραμμα είναι το παρακάτω, το οποίο απεικονίζει το ποσοστό των jobs με τα αντίστοιχα error status.



Σχήμα 4.5: *Από το διάγραμμα αυτό φαίνεται ότι το 70% των jobs που έχουν υποβληθεί έχει exit code 0, αυτό σημαίνει ότι έχει τρέξει με επιτυχία. Το υπόλοιπο 30% είναι ανεπιτυχές, ποσοστό το οποίο όμως είναι πολύ σημαντικό αν σκεφτεί κανείς ότι ένα πιθανό για παράδειγμα restart είναι επώδυνο για τις απαιτήσεις των εργασιών και τη λειτουργία συνολικά του μηχανήματος.*

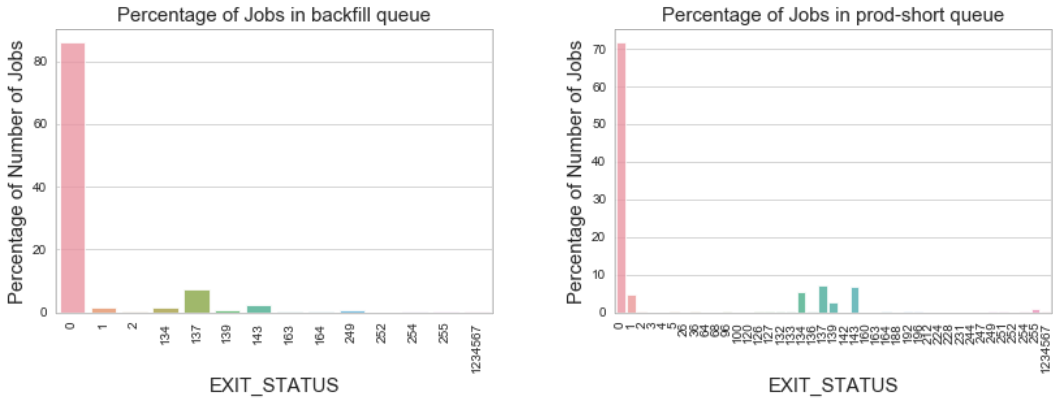
Σε ενδιαφέροντα συμπεράσματα καταλήγουμε και από τα παρακάτω δύο διάγραμμα.



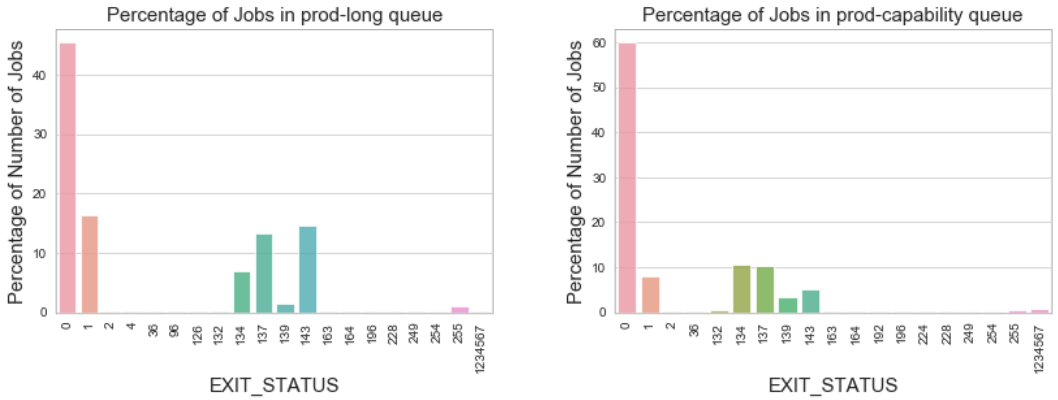
Σχήμα 4.6: *Αριστερά το διάγραμμα απεικονίζει το ποσοστό των συνολικών κόμβων που χρησιμοποιήθηκαν ως προς το exit status. Δεξιά απεικονίζεται η αναλογία του πόσοι κόμβοι χρησιμοποιήθηκαν στα jobs με τα αντίστοιχα exit status.*

Όπως φαίνεται λιγότερο από το 70% των κόμβων που χρησιμοποιήθηκαν συνολικά, αντιστοιχεί σε jobs με exit code 0, ενώ πάνω από το 30% των κόμβων που χρησιμοποιήθηκαν αντιστοιχεί σε jobs που τερματίστηκαν ανεπιτυχώς. Επιπλέον φαίνεται και η αναλογία των πακέτων κόμβων που χρησιμοποιούνται με τα αντίστοιχα exit status. Σε όλα φαίνεται ότι χρησιμοποιείται περισσότερο το "πακέτο" των 512 κόμβων.

Επιπλέον, σημαντική είναι και η πληροφορία που λαμβάνουμε για τα exit codes όσον αφορά τα jobs που βρίσκονται στις ουρές που ανέφερα και προηγουμένως, όπως αυτές εμφανίζονται και στο σχήμα 4.4.



Σχήμα 4.7: *Αριστερά το ποσοστό των jobs με τα αντίστοιχα exit status στην ουρά backfill, ενώ δεξιά για την ουρά prod-short.*



Σχήμα 4.8: *Αριστερά το ποσοστό των jobs με τα αντίστοιχα exit status στην ουρά prod-long, ενώ δεξιά για την ουρά prod-capability.*

Όπως είναι φανερό από τα σχήματα 4.7 και 4.8, βλέπουμε ότι ενώ το μεγαλύτερο ποσοστό των jobs εκτελείται επιτυχώς, με exit status 0, υπάρχει σημαντικό ποσοστό το οποίο δεν εκτελείται επιτυχώς. Διαφορά παρατηρείται και στα ποσοστά ανάμεσα στα είδη ουρών, για παράδειγμα στην ουρά prod-long πάνω από το 10% των jobs έχει exit status 143. [4] Επίσης, πάνω από το 10% των jobs στην ίδια ουρά έχει exit status 137. [5] Στην ίδια ουρά σημαντικό ποσοστό καταλαμβάνει και το exit status

[4]Με αυτό το exit status αποστέλλεται το σήμα Sigterm το οποίο γίνεται για να ζητήσει από μια διαδικασία τον τερματισμό της. Με το σήμα αυτό ο τερματισμός της λειτουργίας γίνεται πιο ομαλά. Όταν η διαδικασία λάβει το σήμα, μπορεί είτε να σταματήσει αμέσως, είτε να σταματήσει μετά από μια μικρή καθυστέρηση αφότου καθαριστούν πόροι, είτε μπορεί να τρέχει επ'αόριστον.

[5]Με τον κωδικό εξόδου 137, αποστέλλεται το σήμα SIGKILL. Το σήμα SIGKILL αποστέλλεται σε μια διαδικασία που την προκαλεί να τερματιστεί αμέσως. Αυτό το σήμα δεν μπορεί να αγνοηθεί και η διαδικασία λήψης δεν μπορεί να εκτελέσει κανένα καθαρισμό κατά τη λήψη αυτού του σήματος, για την ακρίβεια η διαδικασία δεν γνωρίζει καν το σήμα SIGKILL, αφού πηγαίνει κατευθείαν στον kernel. Ωστόσο μπορεί ο kernel να μην είναι σε θέση να καταστρέψει επιτυχώς τη διαδικασία σε μερικές περιπτώσεις. Για παράδειγμα, αν η διαδικασία περιμένει για δίκτυο ή δίσκους I/O, ο kernel δεν μπορεί να την σταματήσει. Επίσης οι διαδικασίες zombie και αυτές που βρίσκονται σε έναν

134. [6] Τέλος στην ουρά prod-long βλέπουμε σημαντικό ποσοστό να κατέχει και το exit status 1, το οποίο σημαίνει λάθη γενικού τύπου, τα οποία προέρχονται από τους χρήστες. Συμπεραίνεται λοιπόν ότι στην συγκεκριμένη ουρά πάνω από το 50% των jobs που έχουν υποβληθεί δεν τερματίζονται με επιτυχία.

Σε αντίστοιχα συμπεράσματα καταφεύγουμε και από τις άλλες τρεις ουρές, με μεγαλύτερα ποσοστά αποτυχίας να έχουμε στην ουρά prod-capability (περίπου 40%) και λιγότερα στις ουρές prod-short (λιγότερο από 30%) και backfill (λιγότερο από 20%).

Από τα παραπάνω διαγράμματα λοιπόν φαίνεται ότι τα jobs στην prod-short και backfill ουρά έχουν μεγαλύτερες πιθανότητες να τερματίσουν με ομαλό τρόπο. Μόνο το 70% των jobs που έχουν υποβληθεί στο μηχάνημα Mira την περίοδο που μελετάμε (9 Απριλίου-31 Δεκεμβρίου 2013) τερματίζει με επιτυχία, το υπόλοιπο 30% δεν τερματίζει. Αυτό δημιουργεί αρκετά προβλήματα τόσο στη διαχείριση των πόρων που παρέχεται προς τους χρήστες, όσο και στον εντοπισμό και στη γρήγορη επίλυση των προβλημάτων που δημιουργούν αυτά τα failures. Αν και μερικά failures είναι προφανή, τα περισσότερα είναι δύσκολο να εντοπιστούν.

Με την αύξηση της πολυπλοκότητας των συστημάτων υψηλής απόδοσης, αυξάνονται ακόμα περισσότερα τα failures στα μεγάλης κλίμακας HPC συστήματα, με αποτέλεσμα να μειώνεται και ο MTBF - Mean Time Between Failures και άρα να σπαταλάται σημαντική υπολογιστική δυνατότητα.

Όλα αυτά μας οδηγούν να ερευνήσουμε τρόπους με τους οποίους θα προβλέπεται κάποια ανωμαλία του συστήματος, σε ποιο σημείο του συστήματος θα γίνει αυτή (σε ποιον κόμβο), καθώς και σε πόση ώρα από τώρα (lead time). Για την επίτευξη αυτών των στόχων οι ως τώρα συνηθισμένες τεχνικές εξόρυξης δεδομένων, όπως support vector machines (SVM) (Fulp et al., 2008), ή principal component analysis (PCA) (Lan et al., 2009) δεν είναι αρκετά επεκτάσιμες για τόσο μεγάλης κλίμακας δεδομένα και σε real-time. Ως εκ τούτου καινοτόμα, επεκτάσιμα και βελτιωμένα εργαλεία απαιτούνται.

Η επεξεργασία γλώσσας από μη δομημένα logs που παράγονται από υπολογιστικά συστήματα αναδεικνύουν δύο προβλήματα όπως αναφέρθηκε και στην ενότητα 1.6. Κατ'αρχάς ότι τα logs δεν είναι ετικετοποιημένα και κατά δεύτερον ότι για τον συμπερασμό σύνθετων patterns πρέπει τα δεδομένα αυτά να υποβληθούν σε επεξεργασία και να τροφοδοτηθούν με κατάλληλη αναπαράσταση εισόδου. Σ'αυτό το σημείο μας βοηθά η ραγδαία ανάπτυξη της βαθιάς μάθησης στην κατανόηση της φυσικής γλώσσας.

Με βάση τα παραπάνω οδηγούμαστε στην ανάπτυξη μιας μεθοδολογίας με την οποία προβλέπουμε αλυσίδες μηνυμάτων που οδηγούν σε failure και παράλληλα εξάγουμε το lead time έχοντας τη δυνατότητα να εντοπίσουμε σε ποιον κόμβο έγινε το failure του job. Μάλιστα τα δεδομένα που χρησιμοποιούμε είναι πραγματικά (Argonne) και δημόσια, χωρίς καμία επιπλέον "πλαστή" προσθήκη failure για ανωμαλία του συστήματος.

αδιάλειπτο ύπνο δεν μπορούν να σταματήσουν. Απαιτείται επανεκκίνηση για την εκκάθαριση αυτών των διαδικασιών από το σύστημα.

[6] Με το 134 λαμβάνουμε το σήμα SIGABRT, το οποίο δηλώνει τον τερματισμό (core dump) - σήμα διακοπής διεργασίας. Αυτό το μήνυμα αποτελεί σοβαρό σήμα τερματισμού (λόγω abort assertion ή double-free of memory). Αυτά τα σφάλματα ανήκουν στην κατηγορία των Bug.

4.4 Failure Prediction

Για την υλοποίηση της πρόβλεψης αποτυχιών των κόμβων χρησιμοποιήθηκαν πραγματικά logs συστήματος από τον υπερυπολογιστή Mira της IBM για την περίοδο από τον Απρίλιο του 2013 μέχρι και το Δεκέμβριο του 2013. Συγκεκριμένα χρησιμοποιήσαμε τα RAS και τα JOB logs, όπως αναφέρθηκε και στην ενότητα 2.1. Συνενώσαμε τα logs αυτά με κοινό κλειδί την κοινή πληροφορία από τα Block (που είναι σε μορφή πχ MIR-40000-737F1-4096) και κρατήσαμε την πληροφορία από γεγονότα χρονικά κοντά. Επιπλέον, κρατήσαμε τα γεγονότα που λαμβάνουν χώρα στο ίδιο Block και των οποίων το Start Timestamp [7] είναι μικρότερο από το Event Time και με τη σειρά του το Event Time είναι μικρότερο από το End Timestamp.

$$StartTimestamp < EventTimestamp < EndTimestamp \tag{4.1}$$

Η συνένωση αυτή ήταν αρκετά χρονοβόρα καθώς τα logs ήταν πάρα πολλά ειδικά τα RAS log ήταν της τάξης των 5 G αλλά και τα JOB log 33 M.

Με τη μέθοδο που ακολουθήσαμε έχουμε lead time περίπου 10 secs με 1 hour, με recall όχι λιγότερο από 81%. Χωρίζουμε το σύνολο δεδομένων μας σε training και testing, όπου το 30% των δεδομένων χρησιμοποιείται για training και το υπόλοιπο χρησιμοποιείται για testing. Η πειραματική μας αξιολόγηση ποσοτικοποιεί την αποτελεσματικότητα της μεθοδολογίας μας, όσον αφορά τις τυπικές μετρικές απόδοσης.

Αρχικά για την εκπαίδευση του στη φάση 1 έγινε εξαγωγή των word embeddings με τη βοήθεια του μοντέλου skip-gram.

	0	1	2	3	4	...	95	96	97	98	99
ndcorrectableerror	-0.005538	2.680929	24.234669	-8.306913	-24.042770	...	31.209141	-36.268063	-1.395723	1.290433	-7.899679
warningmc0rank1	-0.179043	0.370571	-7.521682	-7.401166	-11.498628	...	11.767866	-7.519443	1.518415	-0.117205	1.809196
serviceaction6585restartednodedcar13	0.016986	0.009768	-16.279041	-9.973520	-11.286119	...	11.680753	-5.424914	2.927208	-0.769646	4.300936
kernelinternalassertionfailure	-6.977578	-0.200877	17.313423	20.057041	-18.064708	...	7.534953	-6.133167	-2.827714	3.030792	3.524452
abqlbiterrorthresholdwasexceeded	0.219328	-3.853128	-11.354437	-1.463878	5.039874	...	-6.411012	8.927105	3.251094	-1.470589	3.760504
alinkchipdidnotalignalongtheaportonswitch2	-0.045018	-0.053850	-4.676926	-2.973093	-3.529989	...	3.492841	-1.598395	0.854124	-0.205243	1.386150
cfamrecoverableerror	0.125706	-1.365464	-3.360045	0.067303	2.761106	...	-3.305687	4.010753	0.965712	-0.484375	1.109738
ndsenderretransmissioncorrectableerror	-0.109295	-0.202403	-15.544802	-7.710242	-7.962912	...	7.159219	-1.393445	2.459007	-0.632582	4.615002
checkforsuccessofporsequencefailed	-0.046419	0.726082	9.227051	-0.840424	-5.163361	...	7.074319	-9.179214	-0.826632	0.494978	-2.958648
alinkchipdidnotbitalignalongthereceivercport	-0.079056	-0.415448	-10.627165	-4.518475	-3.853862	...	3.220978	0.370925	1.666339	-0.529350	3.263497

[10 rows x 100 columns]

Σχήμα 4.9: *Στην παραπάνω εικόνα φαίνονται κάποια από τα word embeddings 10 φράσεων του συνόλου δεδομένων μας.*

Στη συνέχεια για την εκπαίδευση με την χρήση LSTM, μετατρέψαμε το κωδικοποιημένο σύνολο δεδομένων μας σε μορφή κατάλληλη ώστε να μπορούν να εκπαιδευθούν οι χρονοσειρές μας (time-series).

Τεχνικά στην ορολογία πρόβλεψης χρονοσειρών, ο τρέχων χρόνος - current time (t) και οι μελλοντικοί χρόνοι (t+1, t+n) είναι χρόνοι πρόβλεψης και οι προηγούμενες παρατηρήσεις (t-1, t-n) χρησιμοποιούνται για την πραγματοποίηση των προβλέψεων. Αυτό επιτρέπει όχι μόνο την κλασσική πρόβλεψη X− >y, αλλά και την X− >Y, όπου και η είσοδος και η έξοδος μπορούν να είναι ακολουθίες. Επιπλέον, αυτό λειτουργεί και στα προβλήματα των multivariate time series (φάση 2), όπου αντί να έχουμε ένα μόνο σύνολο παρατηρήσεων για μια χρονοσειρά, έχουμε περισσότερα (πχ ids φράσεων και ΔT).

Για να μπορέσουμε λοιπόν να μετατρέψουμε δεδομένα χρονοσειρών σε ένα πρόβλημα επιβλεπόμενης μάθησης, χρησιμοποιήσαμε τη βιβλιοθήκη Pandas και συγκεκριμένα τη συνάρτηση shift().

[7] Το Start Timestamp αναφέρεται στην χρονική στιγμή που ξεκινά ένα job, ενώ το End Timestamp στην χρονική στιγμή που τελειώνει το job. Το Event Timestamp είναι η χρονική στιγμή που καταγράφεται κάποιο event στα RAS logs για το συγκεκριμένο job.

var1(t-11)	var1(t-10)	var1(t-9)	var1(t-8)	var1(t-7)	var1(t-6)	var1(t-5)	var1(t-4)	var1(t-3)	var1(t-2)	var1(t-1)	var1(t)	var1(t+1)	var1(t+2)
10.0	14.0	5.0	5.0	5.0	5.0	5.0	14.0	5.0	5.0	5.0	10	29.0	14.0
14.0	5.0	5.0	5.0	5.0	5.0	14.0	5.0	5.0	5.0	10.0	29	14.0	5.0
5.0	5.0	5.0	5.0	5.0	14.0	5.0	5.0	5.0	10.0	29.0	14	5.0	5.0
5.0	5.0	5.0	5.0	14.0	5.0	5.0	5.0	10.0	29.0	14.0	5	5.0	5.0
5.0	5.0	5.0	14.0	5.0	5.0	5.0	10.0	29.0	14.0	5.0	5	5.0	5.0
5.0	5.0	14.0	5.0	5.0	5.0	10.0	29.0	14.0	5.0	5.0	5	5.0	5.0
5.0	14.0	5.0	5.0	5.0	10.0	29.0	14.0	5.0	5.0	5.0	5	5.0	10.0
14.0	5.0	5.0	5.0	10.0	29.0	14.0	5.0	5.0	5.0	5.0	5	10.0	10.0
5.0	5.0	5.0	10.0	29.0	14.0	5.0	5.0	5.0	5.0	5.0	10	10.0	10.0
5.0	5.0	10.0	29.0	14.0	5.0	5.0	5.0	5.0	5.0	10.0	10	10.0	10.0

Σχήμα 4.10: *Φαίνονται οι στήλες των προηγούμενων παρατηρήσεων (t-1,...,t-n) και οι στήλες των παρατηρήσεων που προβλέπονται (t,...,t+n) για ένα σύνολο δεδομένων αποτελούμενο από χρονοσειρές, σε μια μορφή επιβλεπόμενης μάθησης.*

Αυτή η λειτουργία μας επιτρέπει να δημιουργήσουμε νέα πλαίσια προβλημάτων χρονοσειρών δεδομένου του επιθυμητού μήκους των ακολουθιών εισόδου και εξόδου. Αποτελεί σημαντικό εργαλείο καθώς μας επιτρέπει να διερευνήσουμε διαφορετικά πλαίσια ενός προβλήματος χρονοσειρών με αλγορίθμους μηχανικής μάθησης για να δούμε ποια θα μπορούσαν να οδηγήσουν σε μοντέλα με καλύτερη απόδοση.

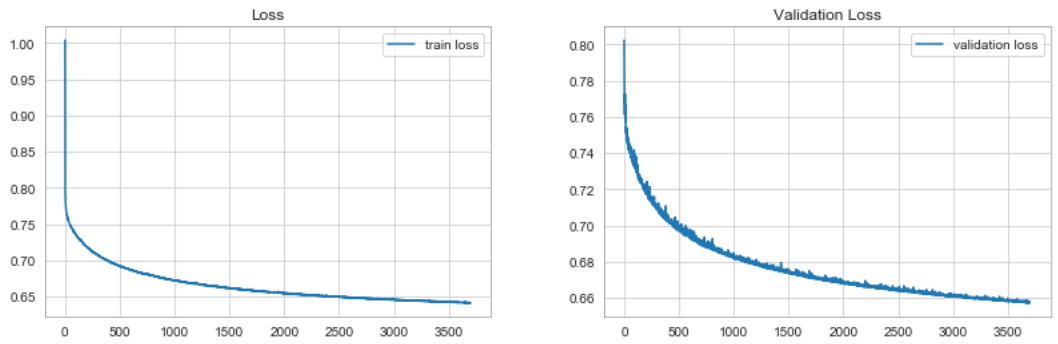
Ακολουθώντας τα παραπάνω, στη συνέχεια κάνουμε την εκπαίδευση των μοντέλων με τα lstm. Αρχικά στη **φάση 1** προεκπαιδεύονται όπως ανέφερα και στις προηγούμενες ενότητες τα δεδομένα του source, το οποίο περιέχει όλο το vocabulary του συνόλου εκπαίδευσης, πριν το phrase labeling. Η εκπαίδευση κατά τη φάση αυτή διαρκεί περίπου 110 ώρες και πετυχαίνουμε accuracy περίπου 76%.

Στη συνέχεια ακολουθεί η **φάση 2** που στην ουσία αποτελεί το δεύτερο μέρος του Sequential Transfer Learning, όπως αυτό περιγράφηκε στην ενότητα 2.8.1. Εδώ πέρα το προεκπαιδευμένο μοντέλο της φάσης 1 χρησιμοποιείται ως σημείο εκκίνησης του μοντέλου-στόχου μας, με σκοπό την ταχύτερη και βέλτιστη σύγκλιση. Η εκπαίδευση στη φάση αυτή διαρκεί περίπου 23.72 λεπτά και πετυχαίνουμε accuracy περίπου 83%.

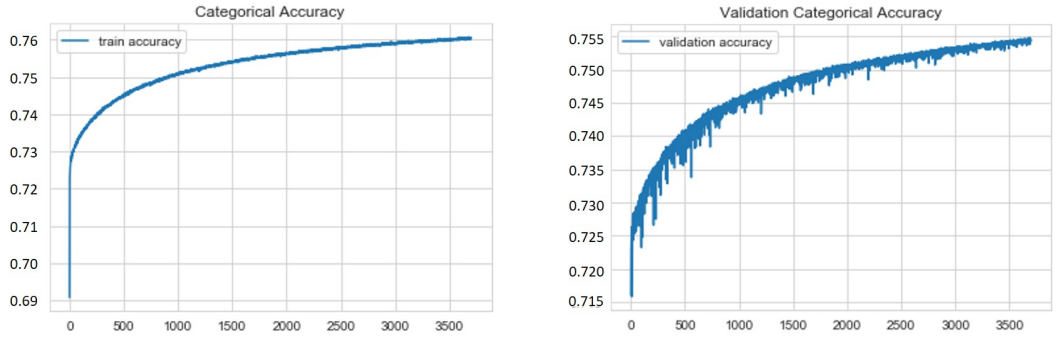
Από τον χρόνο που διαρκεί η κάθε εκπαίδευση φαίνεται και πόσο ωφέλιμο είναι το Transfer learning στο αποτέλεσμά μας. Συγκεκριμένα βλέπουμε τεράστια διαφορά ανάμεσα στη φάση 1 που διαρκεί περίπου 4,5 μέρες και στη φάση 2 που διαρκεί ελάχιστα λεπτά.

Παρακάτω φαίνονται τα διαγράμματα του loss και του accuracy στη φάση 1 και 2 της μεθοδολογίας που ακολούθήσαμε.

- Φάση 1

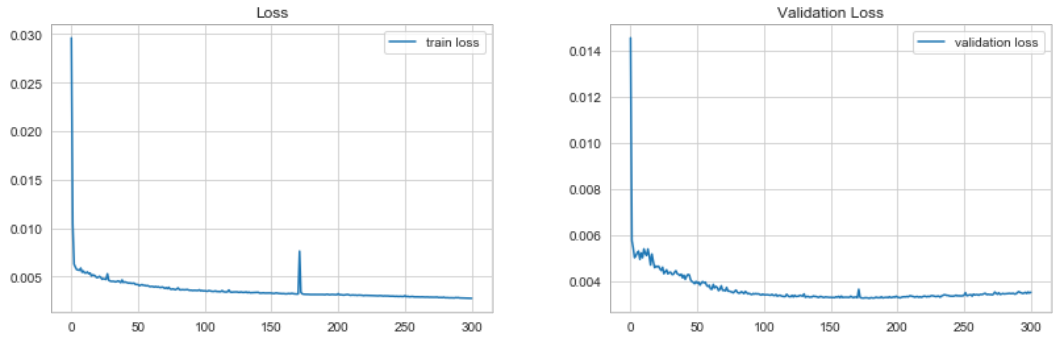


Σχήμα 4.11: *Αριστερά βλέπουμε το διάγραμμα του loss, ενώ δεξιά βλέπουμε το διάγραμμα του validation loss.*

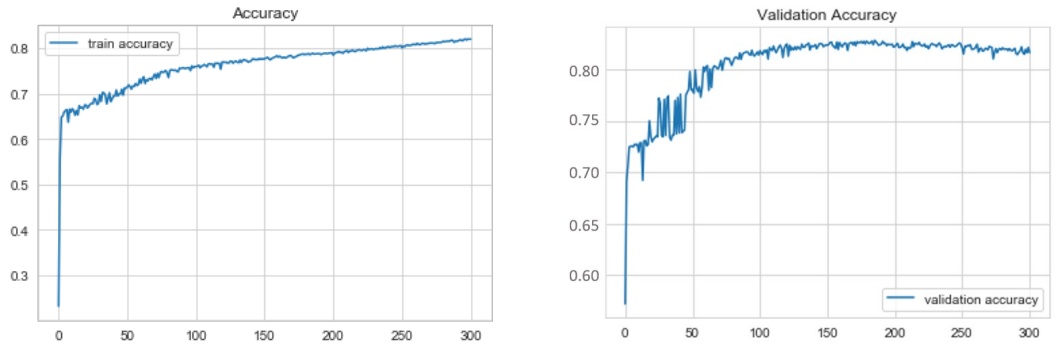


Σχήμα 4.12: *Αριστερά βλέπουμε το διάγραμμα του accuracy, ενώ δεξιά βλέπουμε το διάγραμμα του validation accuracy.*

- Φάση 2



Σχήμα 4.13: *Αριστερά βλέπουμε το διάγραμμα του loss, ενώ δεξιά βλέπουμε το διάγραμμα του validation loss.*

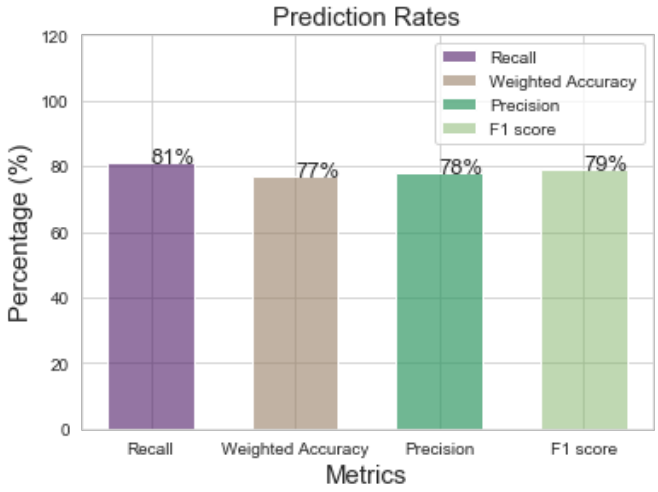


Σχήμα 4.14: *Αριστερά βλέπουμε το διάγραμμα του accuracy, ενώ δεξιά βλέπουμε το διάγραμμα του validation accuracy.*

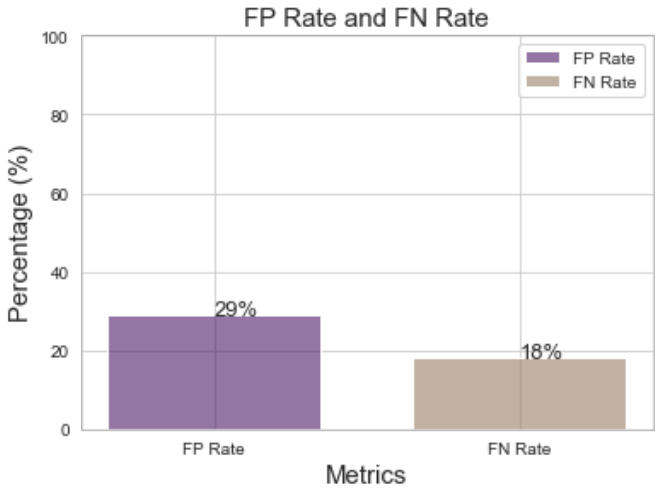
Η απόδοση της μεθοδολογίας μας οφείλεται σε πολύ μεγάλο βαθμό στο **history window size** [8] που έχουμε πάρει. Συγκεκριμένα τα παραπάνω αποτελέσματα προκύπτουν με history size 15 και 10 για τη φάση 1 και 2 αντίστοιχα. Περισσότερο history size βελτιώνει το accuracy, όμως καταναλώνει πολύ περισσότερο χρόνο. Μειώνοντας το history size σε 8 για τη φάση 1 και 5 για τη φάση 2 τα αποτελέσματα

[8]Το history size είναι το window size των δειγμάτων που παρέχονται κατά τη διάρκεια της εκπαίδευσης βάσει των οποίων γίνονται οι προβλέψεις.

δεν είναι τόσο καλά όσο με τα προηγούμενα history sizes, μάλιστα κατεβαίνει το accuracy κατά 10%.
Επιπλέον με το accuracy για την αποτελεσματικότητα της μεθοδολογίας μας υπολογίζουμε κατάλληλες μετρικές οι οποίες όπως αναφέρθηκαν και στην ενότητα 2.9.2, είναι πολύ σημαντικές για την αξιολόγηση της συνολικής μεθοδολογίας μας. Οι προβλεπτικές μετρικές είναι: **recalls, weighted accuracy, precisions, F1 score** καθώς και τα ποσοστά **FP rate, FN rate**. Αυτές οι μετρικές υπολογίζονται στη φάση 3 και εκφράζουν την προβλεπτική δύναμη της μεθόδου για πιθανές failure chains.
Για history size 15 και 10 στις φάσεις 1 και 2 έχουμε συνολικά για τη μεθοδολογία μας 81% recall, 77% weighted accuracy, 78% precision, 79% F1 score. Οπότε βλέπουμε ότι προβλέπεται το μεγαλύτερο ποσοστό των failure chains που υπάρχουν στα test data με πολύ σημαντικό recall ενώ έχουμε ένα αρκετά μεγάλο ποσοστό precision.



Σχήμα 4.15: *Στην παραπάνω εικόνα φαίνονται τα Prediction Rates που πετυχαίνει η μεθοδολογίας μας.*



Σχήμα 4.16: *Στη συγκεκριμένη εικόνα φαίνονται τα FP και FN Rates.*

Ανάλυση των δεδομένων test. Το 71.43% των φράσεων στο σύνολο όλων των test δεδομένων αποτελεί FC-related φράσεις. Το ποσοστό αυτό των

φράσεων είναι μέρος των failure chains, το υπόλοιπο από αυτό το ποσοστό ανήκει στις φράσεις που δεν ανήκουν στις failure chains. Σημαντικό να σημειωθεί ότι το συγκεκριμένο ποσοστό μόνο του δεν υποδεικνύει το αναμενόμενο recall, ή precision, αφού FC-related φράσεις που είναι μέρος και σε ακολουθίες πολύ διαφορετικές από τις failure chains (πχ. δεν τερματίζουν με κάποιο τερματικό μήνυμα που υποδεικνύει κάποιο failure), δεν είναι πραγματικές failures, έχουν μερικές ομοιότητες. Πολλά nodes αντιμετωπίζουν παρόμοια μηνύματα μέσα στη μέρα, ωστόσο δεν αποτυγχάνουν όλα. Αυτή η εκτίμηση αποτελεί απλά ένα δείκτη επικάλυψης κοινών φράσεων που οδηγούν σε failure εν αντιθέση με αυτές που δεν οδηγούν σε failure.

Επιπλέον όπως αναφέρθηκε και στην ενότητα 3.2 στη φάση 3 χρησιμοποιείται ένα κατώφλι του MSE, κάτω από το οποίο μπορούμε να ερευνάμε για πιθανά failure chains (MSE<=0.0145), ενώ πάνω από αυτό είναι πιο απίθανο να βρούμε κάποιο failure chain. Το κατώφλι υπολογίζεται πειραματικά.

Κεφάλαιο 5

Συμπεράσματα

Αρχικά όσον αφορά το **visualization** των job logs που κάναμε παρατηρούμε ότι:

- Τα περισσότερα jobs που έχουν υποβληθεί την περίοδο που μελετάμε τα logs του μηχανήματος, έχουν υπερεκτιμηθεί.
- Τα περισσότερα jobs που έχουν exit status 0, δηλαδή αυτά που έχουν τερματίσει με επιτυχία, έχουν υπερεκτιμηθεί.
- Τα περισσότερα jobs που έχουν τερματίσει με επιτυχία (exit code 0) βρίσκονται στην ουρά prod-short και μετά στη backfill.
- Στις ουρές prod-long και prod-capability το μεγαλύτερο ποσοστό των jobs δεν τερματίζεται με επιτυχία (50% και 40% μην επιτυχημένων jobs αντίστοιχα).
- Πάνω από το 30% των jobs που έχουν υποβληθεί είναι ανεπιτυχή. Είναι αρκετά μεγάλο ποσοστό ωστόσο για τη δυσκολία της ομαλής λειτουργίας συνολικά του μηχανήματος, καθώς και του χρόνου εκτέλεσης των εργασιών. Άρα η ανάγκη για πρόβλεψη των αποτυχιών του συστήματος που οδηγεί εν τέλει στην πρόβλεψη των job failures είναι πολύ σημαντική.

Όσον αφορά το **failure prediction**, στη συγκεκριμένη διπλωματική εργασία παρουσιάστηκε μια ισχυρή μεθοδολογιά για την επεξεργασία HPC logs με την χρήση LSTM για αποτελεσματική πρόβλεψη failure. Χρησιμοποιείται μια προσέγγιση βαθιάς μάθησης τριών φάσεων ώστε αρχικά να εκπαιδεύεται να προβλέπει την επόμενη φράση, στη συνέχεια να ξαναεκπαιδεύεται για αναγνώριση αλυσίδων από logs που οδηγούν σε failure επαυξημένες με τα προβλέπομενα lead times και τρίτον να προβλέπει failure chains και κατ'επέκταση lead time κατά τη διάρκεια του testing. Έτσι προβλέπεται που θα υπάρχει failure και σε πόσα λεπτά/δευτερόλεπτα.

Από τα αποτελέσματα της προηγούμενης ενότητας 4 βλέπουμε ότι:

- Έχουμε αρκετά υψηλό recall - 81%, αλλά και precision - 78%.
  Άρα από τις συνολικές failure chains που υπάρχουν πράγματι στα test δεδομένα, εντοπίζω αρκετά μεγαλό ποσοστό αυτών (recall). Καθώς επίσης ότι από τις συνολικές failure chains που προέβλεψα, το μεγαλύτερο ποσοστό τους είναι όντως failure chains (precision).
- Το υψηλό F1 score - 79%, το οποίο αξιολογεί την ακρίβεια της μεθοδολογίας μας ως προς την πρόβλεψη failure, θεωρώντας ότι είναι ο σταθμισμένος μέσος

όρος του recall και του precision, είναι ακόμα ένας δείκτης του πόσο καλή είναι η μεθοδολογία αυτή.

- Τα παραπάνω συνεπάγονται ότι οι πληροφορίες που αποκτήθηκαν από τις εκπαιδευμένες failure chains (φάση 2) βοήθησαν στην πραγματοποίηση ακριβών προβλέψεων για τα συνολικά test δεδομένα.

- Φαίνεται ότι το Transfer learning που χρησιμοποιείται στη φάση 2 είναι αρκετά αποτελεσματικό και σε απόδοση της μεθόδου και σε χρόνο, άρα και όσον αφορά την χρήση των πόρων που διαθέτουμε.

- Χρησιμοποιώντας μικρότερο history size μειώνεται το accuracy, ενώ μεγαλύτερο αυξάνεται, με κόστος όμως να απαιτείται περισσότερος χρόνος για την εκπαίδευση. Στη συγκεκριμένη εργασία βρέθηκε με πειράματα ότι τα κατάλληλα history sizes είναι 15 για τη φάση 1 και 10 για τη φάση 2.

- Με τη μέθοδο που ακολουθήσαμε έχουμε lead time περίπου 10 secs με 1 hour.

Άρα τα failure chains είναι αρκετά αξιόπιστα ώστε να διατηρήσουν την προβλεπτική δύναμη του μοντέλου μας για τα συμβάντα που συναντώνται κατά τη διάρκεια του testing.

Επιπλέον έχοντας ερευνήσει πολλές διαφορετικές αλυσίδες που οδηγούν σε failed κόμβους, καθορίζονται οι κυρίαρχες φράσεις που οδηγούν σε μη ομαλό node shutdown. Αυτές οι φράσεις ανήκουν σε ορισμένες κλάσεις οι οποίες καταγράφονται παρακάτω:

- MCEs αναφέρονται στα hardware machine check exceptions τα οποία συναντώνται πολύ συχνά.

- FileSystem(FS)(access alert).

- Hardware (H/W) που αναφέρεται σε hardware errors.

- Panic αναφορικά με kernel panic μηνύματα.

Με βάση την κλάση του failure τα lead times διαφέρουν. Τα MCEs και τα Panic failures έχουν συγκριτικά πολύ υψηλότερο lead time από τα άλλα, περίπου 1 ώρα. Σημαντικό επίσης τέλος είναι να αναφέρω ότι το μεγαλύτερο ποσοστό των φράσεων που οδηγεί σε failure ανήκουν στην κλάση των MCE και του Hardware.

Κεφάλαιο 6

Μελλοντική Εργασία

Στο πλαίσιο του failure prediction ένα επόμενο βήμα που θα θέλαμε να γίνει είναι να εφαρμοστεί η μεθοδολογία και σε άλλες χρονιές λειτουργίας του μηχανήματος Mira. Στη συγκεκριμένη εργασία όπως αναφέρθηκε και παραπάνω χρησιμοποιήθηκαν τα logs κατά την περίοδο 9 Απριλίου 2013 μέχρι 31 Δεκεμβρίου 2013. Οπότε θα ήταν σημαντικά και τα αποτελέσματα που θα πάρουμε και από τις άλλες χρονιές λειτουργίας του.

Ακόμα θα ήταν πολύ ενδιαφέρον να εφαρμοστεί και σε logs άλλων HPC καθώς σα μέθοδος είναι generic. Αυτό σημαίνει ότι μπορεί σίγουρα να προσαρμοστεί σε άλλα συστήματα παραγωγής μεγάλης κλίμακας (large-scale) με διαφορετικό logging, κάνοντας κάποιο customization. Η μέθοδος αυτή κατά κύριο λόγο είναι unsupervised και παραμένει ανενόχλητη από τα χάσματα των διαφορετικών υπολογιστικών υποδομών.

Στην εργασία αυτή χρησιμοποιούνται τεχνικές deep learning ώστε να μπορεί να γίνει anomaly detection και failure prediction στο Mira. Αντίστοιχες προσπάθειες έχουν γίνει και από άλλους όπως οι (Fu and Xu, 2007), (Berrocal et al., 2014) και (Du et al., 2017) οι οποίοι ακολουθούν τεχνικές όπως clustering, SVM, PCA, neural networks.

Ωστόσο αυτές οι προσπάθειες που γίνονται και αποκαλύπτουν τον "πλούτο" των πληροφοριών από εργασίες που βασίζονται στο Machine Learning θα αποδόσουν πραγματικά μόνο όταν θα μπορέσουμε να χτίσουμε πραγματικά frameworks για online failure prediction. Η μέθοδος που ακολουθούμε θα μπορούσε να έχει εφαρμογή σε real-time προσεγγίσεις που απαιτούν γρήγορο anomaly detection online.

Κλείνοντας, είναι φανερό ότι αυτή η εργασία έχει αρκετές πιθανές μελλοντικές εξελίξεις. Τα αποτελέσματα καθώς και τα βήματα της μεθόδου παρακινούν να επεκταθεί και για την ανίχνευση και τον εντοπισμό failure σε διαφορετικά υπερυπολογιστικά συστήματα, καθώς δίνει και το κίνητρο για ανάπτυξη ενός online εργαλείου για anomaly detection.

Βιβλιογραφία

W. Allcock, P. Rich, Y. Fan, and Z. Lan. Experience and practice of batch scheduling on leadership supercomputers at argonne. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 1–24. Springer, 2017.

Argonne. This data was generated from resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. https://reports.alcf.anl.gov/data/ANL-ALCF-AUTOPERF-MIRA_20180101_20181231.html.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

E. Berrocal, L. Yu, S. Wallace, M. E. Papka, and Z. Lan. Exploring void search for fault detection on extreme scale systems. In *2014 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–9. IEEE, 2014.

F. Cappello, G. Al, W. Gropp, S. Kale, B. Kramer, and M. Snir. Toward exascale resilience: 2014 update. *Supercomputing Frontiers and Innovations: an International Journal*, 1(1):5–28, 2014.

cray. ARCHITECTURAL INNOVATION FOR THE EXASCALE ERA. https://www.cray.com/resources/Architectural-innovation-for-exascale-era.

C. Darken, J. Chang, and J. Moody. Learning rate schedules for faster stochastic gradient search. In *Neural Networks for Signal Processing [1992] II., Proceedings of the 1992 IEEE-SP Workshop*, pages 3–12. IEEE, 1992.

A. Das, F. Mueller, C. Siegel, and A. Vishnu. Desh: deep learning for system health prediction of lead times to failure in hpc. In *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, pages 40–51, 2018.

S. Di, R. Gupta, M. Snir, E. Pershey, and F. Cappello. Logaider: A tool for mining potential correlations of hpc log events. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 442–451. IEEE, 2017.

S. Di, H. Guo, R. Gupta, E. R. Pershey, M. Snir, and F. Cappello. Exploring properties and correlations of fatal events in a large-scale hpc system. *IEEE Transactions on Parallel and Distributed Systems*, 30(2):361–374, 2018.

J. D. DOD, B. Harrod, T. Hoang, L. Nowell, B. Adolf, S. Borkar, N. DeBardeleben, M. Heroux, D. Rogers, V. Sarkar, et al. Inter-agency workshop on hpc resilience at extreme scale. 2012.

M. Du, F. Li, G. Zheng, and V. Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1285–1298, 2017.

E. Elnozahy, R. Bianchini, T. El-Ghazawi, A. Fox, F. Godfrey, A. Hoisie, K. McKinley, R. Melhem, J. Plank, P. Ranganathan, et al. System resilience at extreme scale. *Defense Advanced Research Project Agency (DARPA), Tech. Rep*, 2008.

S. Fu and C.-Z. Xu. Exploring event correlation for failure prediction in coalitions of clusters. In *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–12, 2007.

E. W. Fulp, G. A. Fink, and J. N. Haack. Predicting computer system failures using support vector machines. *WASL*, 8:5–5, 2008.

A. Gainaru, F. Cappello, M. Snir, and W. Kramer. Fault prediction under the microscope: A closer look into hpc systems. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE, 2012.

R. Gandhi. A Look at Gradient Descent and RMSprop Optimizers. https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b.

S. Gupta, D. Tiwari, C. Jantzi, J. Rogers, and D. Maxwell. Understanding and exploiting spatial properties of system failures on extreme-scale hpc systems. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 37–44. IEEE, 2015.

S. Gupta, T. Patel, C. Engelmann, and D. Tiwari. Failures in large scale systems: long-term measurement, analysis, and implications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2017.

C. HANSEN. Activation Functions Explained - GELU, SELU, ELU, ReLU and more. https://mlfromscratch.com/activation-functions-explained/#elu.

L. Harasim. *Learning theory and online technologies*. Routledge, 2012.

S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

IBM. Backfill scheduling. https://www.ibm.com/support/knowledgecenter/en/SSWRJV_10.1.0/lsf_admin/backfill.html.

imdb. Star Wars. https://www.imdb.com/title/tt0076759/?ref_=ttmi_tt.

S. J Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, chapter Intelligent agents, pages 31–52. Prentice Hall, 01 1995. ISBN 0137903952.

KERAS. Keras Documention. https://keras.io/.

G. Lakner, B. Knudson, et al. *IBM system Blue Gene solution: Blue Gene/Q system administration*. IBM Redbooks, 2013.

Z. Lan, Z. Zheng, and Y. Li. Toward automated anomaly identification in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 21(2):174–187, 2009.

R. Lucas, J. Ang, K. Bergman, S. Borkar, W. Carlson, L. Carrington, G. Chiu, R. Colwell, W. Dally, J. Dongarra, et al. Doe advanced scientific computing advisory subcommittee (ascac) report: top ten exascale research challenges. Technical report, USDOE Office of Science (SC)(United States), 2014.

J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12, 2006.

P. McCorduck. *Machines Who Think (2nd Ed.)*. A. K. Peters, 2004.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

M. L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., 1967.

"Mira". Mira (supercomputer). https://en.wikipedia.org/wiki/Mira_(supercomputer).

T. M. Mitchell. *The discipline of machine learning*, volume 9. Carnegie Mellon University, School of Computer Science, Machine Learning Department, 2006.

C. Olah. Understanding lstm networks. 2015.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

M. Peters, S. Ruder, and N. A. Smith. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*, 2019.

D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

W. Rawat and Z. Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

H. Robbins and S. Monro. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pages 102–109. Springer, 1985.

S. Ruder. An overview of gradient descent optimization algorithms. http://ruder.io/optimizing-gradient-descent/index.html#stochasticgradientdescent, 2016.

S. Ruder. *Neural transfer learning for natural language processing*. PhD thesis, NUI Galway, 2019.

D. D. Sarkar. A hands-on intuitive approach to Deep Learning Methods for Text Data — Word2Vec, GloVe and FastText. https://towardsdatascience.com/understanding-feature-engineering-part-4-deep-learning-methods-for-text-data-96c

A. Sharma. Understanding Activation Functions in Neural Networks. https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0, 2017.

S. V. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89, 1997.

"Summit". Summit (supercomputer). https://en.wikipedia.org/wiki/Summit_(supercomputer).

"Supercomputer". Supercomputer. https://en.wikipedia.org/wiki/Supercomputer.

Top500. Top500. https://www.top500.org/.

A. S. Walia. Types of Optimization Algorithms used in Neural Networks and Ways to Optimize Gradient Descent. https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gr 2017.

Δασκαλάκης. Video: Διάλεξη του Δρ. Κωνσταντίνου Δασκαλάκη — Τεχνητή Νοημοσύνη: Το μέλλον τώρα;. https://www.auth.gr/video/24941, 2018.