ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΑΞΙΟΠΟΙΗΣΗ ΜΕΓΑΛΩΝ ΓΛΩΣΣΙΚΩΝ ΜΟΝΤΕΛΩΝ ΓΙΑ ΤΗΝ ΑΝΑΠΤΥΞΗ ΑΥΤΟΜΑΤΩΝ ΕΛΕΓΧΩΝ ΑΠΟΔΟΧΗΣ ΛΟΓΙΣΜΙΚΟΥ»

ΠΛΑΤΙΑΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: P3200157

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

ΔΙΑΜΑΝΤΙΔΗΣ ΝΙΚΟΛΑΟΣ

ΑΘΗΝΑ, ΑΥΓΟΥΣΤΟΣ 2024

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1.ΕΙΣΑΓΩΓΗ 1
2.ΜΕΓΑΛΑ ΓΛΩΣΣΙΚΑ ΜΟΝΤΕΛΑ 2
2.1 Τι είναι ένα νευρωνικό δίκτυο και πώς λειτουργεί 2
2.1.1 Ορισμός Νευρωνικών Δικτύων 2
2.1.2 Ορισμός της μείωσης βαρών με καθοδική κλίση 3
2.1.3 Ορισμός του αλγορίθμου Ανάστροφης Μετάδοσης 5
2.2 Τι είναι ένα μεγάλο γλωσσικό μοντέλο 6
2.2.1 Ορισμός Μεγάλου Γλωσσικού Μοντέλου 6
2.2.2 Τι κάνει ένα Transformer 6
2.2.3 Τι είναι το Attention layer 7
2.2.4 Τι είναι το Feed Forward Step 8
2.3 Ιστορία και εξέλιξη των μεγάλων γλωσσικών μοντέλων 8
2.3.1 Το chatbot ELIZA 8
2.3.2 Άνοδος των Νευρωνικών Δικτύων 9
2.3.3 Δημιουργία των LSTM 10
2.3.4 Δημιουργία Gated Recurrent Network 10
2.3.5 Άνοδος του συστατικού Attention 10
2.3.6 Η εφεύρεση των Transformers 11
2.3.7 Εμφάνιση Μεγάλων Γλωσσικών Μοντέλων 11
2.4 Κύριες εφαρμογές και χρήσεις 12
2.4.1 Ανάλυση ήχου 12
2.4.2 Δημιουργία περιεχομένου 12
2.4.3 Υποστήριξη πελατών 12
2.4.4 Μετάφραση γλωσσών 13
2.4.5 Εκπαίδευση 13
2.4.6 Κυβερνοασφάλεια 13
2.5 Παραδείγματα Μεγάλων Γλωσσικών Μοντέλων 13
2.5.1 BERT 13
2.5.2 GEMINI 14
2.5.3 GPT –3 14
2.5.4 GPT -3.5 και GPT –3.5 Turbo 14
2.5.5 GPT –4 15

2.5.6 GPT –4o 15
3.ΜΕΓΑΛΑ ΓΛΩΣΣΙΚΑ ΜΟΝΤΕΛΑ ΣΤΗΝ ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ 15
3.1 Ρόλος των μεγάλων γλωσσικών μοντέλων στην ανάπτυξη λογισμικού 15
3.1.1 Εισαγωγή στην έννοια των γλωσσικών μοντέλων στην ανάπτυξη λογισμικού 15
3.1.2 Παραδείγματα χρήσης μεγάλων γλωσσικών μοντέλων στην ανάπτυξη λογισμικού 16
3.1.3 Αυτοματοποίηση κώδικα 16
3.1.4 Ανάλυση και διόρθωση λαθών 17
3.1.5 Ανάλυση των απαιτήσεων μετατροπή τους σε τεχνικές προδιαγραφές 17
3.2 Χρήσεις σε διάφορα στάδια της ανάπτυξης λογισμικού 18
3.2.1 Στάδιο Ανάλυσης Απαιτήσεων 18
3.2.2 Στάδιο Σχεδιασμού 18
3.2.3 Στάδιο Κωδικοποίησης 19
3.2.4 Στάδιο ελέγχου και βελτιστοποίησης 19
3.3 Πλεονεκτήματα και περιορισμοί 20
3.3.1 Πλεονεκτήματα 20
3.3.2 Περιορισμοί 21
3.4 Μελλοντικές προοπτικές και ευκαιρίες ανάπτυξης 21
3.4.1 Σύμπραξη μεγάλων γλωσσικών μοντέλων 22
3.4.2 Κατανόηση διαφορετικού τύπου εισόδων 22
3.4.3 Προσαρμογή στις ανάγκες των χρηστών 22
3.4.4 Βελτίωση στις ήδη υπάρχουσες εργασίες 22
4.Behavior Driven Development (BDD) 23
4.1 Εισαγωγή στο Behavior Driven Development (BDD) και Test Driven Development (TDD) 23
4.1.1 Ορισμός και βασικές αρχές του TDD 23
4.1.2 Ορισμός και βασικές αρχές του BDD 24
4.2 Η σημασία της συνεργασίας στο BDD 25
4.2.1 Ομοιογενής Γλώσσα (Ubiquitous Language) 25
4.2.2 Σενάρια και Ιστορίες Χρηστών στο BDD 25
4.3 Οι τρείς πρακτικές του BDD 26
4.4 Τεχνικές Ανάλυσης Συνεργασίας 27
4.4.1 Η συνάντηση των Τριών Φίλων (Three Amigos Meeting) 28
4.5 Αυτοματισμός Δοκιμών με το Cucumber 29
4.5.1 Εισαγωγή στο Cucumber 29
4.5.2 Η Σύνταξη Gherkin 29

4.5.3 Δημιουργία Αρχείων Χαρακτηριστικών (Feature Files) 31
4.5.4 Ανάπτυξη των Step Definitions 32
4.6 Αναφορές του Cucumber 33
4.7 Ζωντανή Τεκμηρίωση 34
5.ΒΙΒΛΙΟΓΡΑΦΙΑ 36

1. ΕΙΣΑΓΩΓΗ

2. ΜΕΓΑΛΑ ΓΛΩΣΣΙΚΑ ΜΟΝΤΕΛΑ

2.1 Τι είναι ένα νευρωνικό δίκτυο και πώς λειτουργεί

2.1.1 Ορισμός Νευρωνικών Δικτύων

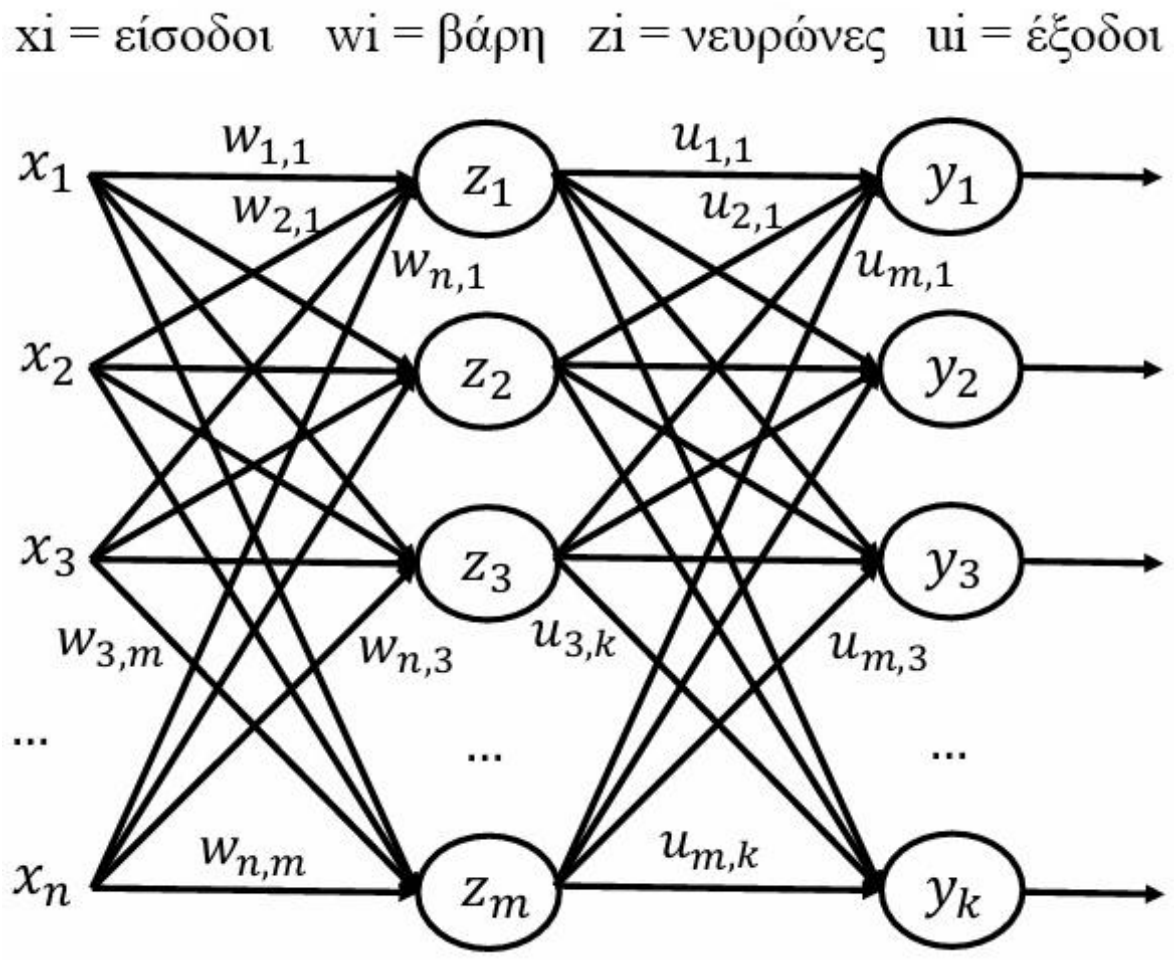
Ξεκινώντας τη συζήτηση για τα μεγάλα γλωσσικά μοντέλα και τη χρήση τους, δεν θα μπορούσε να παραληφθεί η αναφορά στη βασική συνιστώσα που τα αποτελεί, το **«νευρωνικό δίκτυο»** αλλά και στους τρόπους με τους οποίους αυτό λειτουργεί. Ένα νευρωνικό δίκτυο, όπως γίνεται αντιληπτό από το όνομά του, είναι ένα δίκτυο πολλών συνδεδεμένων νευρώνων, χωρισμένων σε διαφορετικά «στρώματα». Κάθε νευρώνας μπορεί να θεωρηθεί ως μία μονάδα που λαμβάνει κάποιες εισόδους, εκτελεί κάποιους πολύ απλούς υπολογισμούς με αυτές και στη συνέχεια παράγει κάποια έξοδο, ένα νούμερο, το οποίο με την σειρά του περνάει ως είσοδος στους επόμενους νευρώνες.

Οι απλοί αυτοί υπολογισμοί είναι το άθροισμα των εισόδων πολλαπλασιασμένοι με έναν αριθμό, το λεγόμενο «βάρος», το οποίο το νευρωνικό δίκτυο μαθαίνει από χιλιάδες δεδομένα κατά την εκπαίδευσή του. Στην συνέχεια, αφού εφαρμοστεί ένας αριθμητικός μετασχηματισμός, το αποτέλεσμα αυτό μεταφέρεται στους νευρώνες του επόμενου στρώματος ως είσοδος και η διαδικασία αυτή συνεχίζεται μέχρι το τελευταίο στρώμα νευρώνων του νευρωνικού.

Για παράδειγμα, ένας νευρώνας να προωθεί στους νευρώνες του επόμενου στρώματος τον αριθμό μηδέν εάν οι είσοδοι που δέχτηκε ήταν αρνητικοί ή κοντά στο μηδέν, και τον αριθμό ένα στην αντίθετη περίπτωση. Αυτό είναι γνωστό ως «σιγμοειδής συνάρτηση ενεργοποίησης».

Συνεχίζοντας, τα μοντέλα βαθιάς μάθησης χρησιμοποιούν εκατομμύρια νευρώνες, χωρισμένους σε πολυάριθμα στρώματα με δισεκατομμύρια βάρη και πολύπλοκες διατάξεις νευρώνων, αλλά η βασική ιδέα παραμένει η ίδια. (Ανδρουτσόπουλος, 2024). Στην **Εικόνα 1**, παρουσιάζεται μία απλοϊκή μορφή ενός νευρωνικού δικτύου, η οποία αποτελείται από ένα στρώμα εισόδων, ένα εσωτερικό στρώμα και ένα στρώμα εξόδων.

Εικόνα 1. Παράδειγμα απλού νευρωνικού δικτύου



Σημείωση: Ανάκτηση από «Τεχνητή Νοημοσύνη και Μεγάλα Γλωσσικά Μοντέλα», Ι.Ανδρουτσόπουλος, ΟΠΑ News Εφημερίδα Οικονομικού Πανεπιστημίου Αθηνών Τεύχος 51, 2024, σελ.8,(σύνδεσμος).

2.1.2 Ορισμός της μείωσης βαρών με καθοδική κλίση

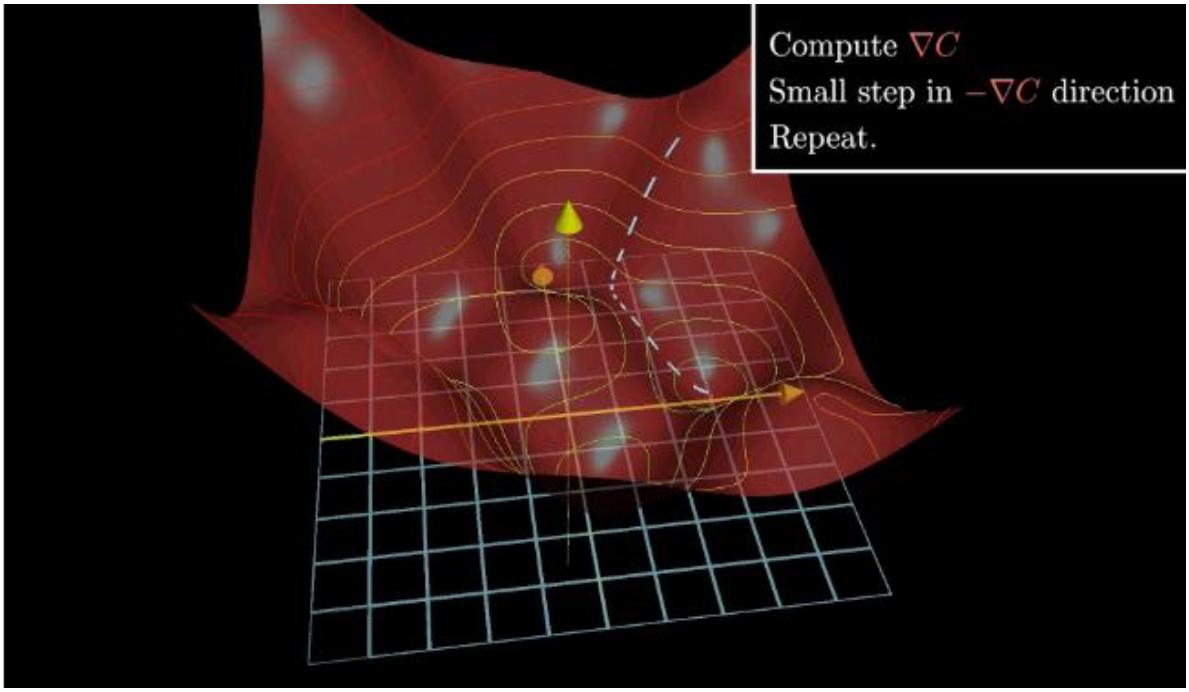
Αυτό που κάνει τα νευρωνικά δίκτυα τόσο ενδιαφέροντα, και κατά συνέπεια τη μηχανική μάθηση, είναι ότι στην ουσία, ποτέ δεν γράφεται κάποιος αλγόριθμος που να ορίζει συγκεκριμένα τι θα πρέπει να κάνει ένα νευρωνικό δίκτυο και πως θα πρέπει να παράγει αποτελέσματα. Αντίθετα, γράφεται ένας αλγόριθμος που, μέσω της εισαγωγής εκατομμυρίων παραδειγμάτων και δεδομένων με τις ορθές «ετικέτες» τους (δηλαδή το επιθυμητό αποτέλεσμα),

μπορεί να μεταβάλλει τα εκατομμύρια βάρη από τα οποία αποτελείται το δίκτυο, ώστε να αποδίδει καλύτερα στα παραδείγματα αυτά.

Τα δεδομένα αυτά ονομάζονται «δεδομένα εκπαίδευσης», και μαζί με τα «δεδομένα δοκιμής», (παραδείγματα που το νευρωνικό δίκτυο δεν έχει «ξαναδεί» και χρησιμοποιούνται για την αξιολόγηση της αποτελεσματικότητάς του), αποτελούν την συνολική αξιολόγηση του συστήματος. Το πρόβλημα αυτό της μεταβολής των βαρών που καλείται να λύσει ο αλγόριθμος καταλήγει να είναι η εύρεση του ελαχίστου μιας «συνάρτησης κόστους».

Η συνάρτηση αυτή υπολογίζεται με βάση τα αποτελέσματα που παράγει το νευρωνικό δίκτυο και τα επιθυμητά αποτελέσματα που έχουμε για κάθε παράδειγμα, και συνεπώς έχει μεγάλη τιμή εάν τα αποτελέσματα διαφέρουν σε μεγάλο βαθμό από τα επιθυμητά, και μικρή τιμή στην αντίθετη περίπτωση. Σκοπός του αλγορίθμου είναι, με βάση τον μέσο όρο όλων αυτών των τιμών κόστους για κάθε παράδειγμα, να προσπαθήσει να μεταβάλλει τις τιμές των βαρών της έτσι ώστε η συνάρτηση κόστους να φτάσει σε ένα τοπικό ελάχιστο.

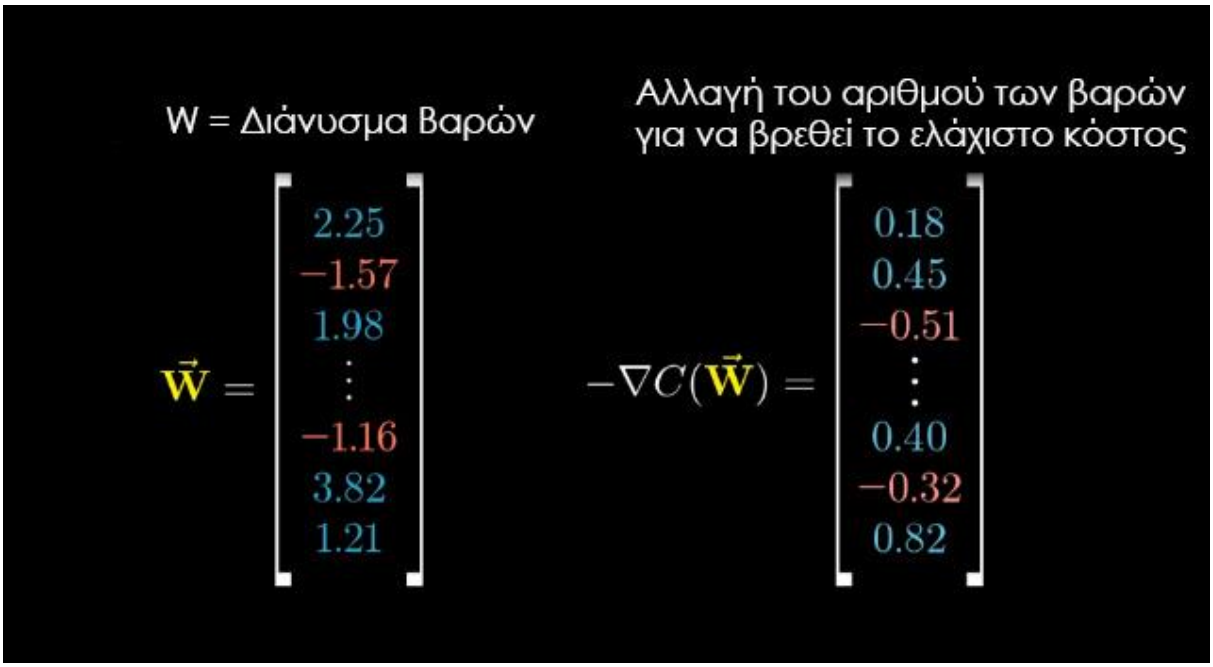
Εικόνα 2. Παράδειγμα αναπαράστασης του gradient descent



Σημείωση: Ανάκτηση από «Gradient descent, how neural networks learn», G. Sanderson-3blue1brown,2017 (σύνδεσμος).

Η τεχνική αυτή ονομάζεται «**καθοδική κλίση**» ή όπως είναι γνωστή, gradient descent, καθώς προσπαθεί να βρει ένα τοπικό ελάχιστο μίας συνάρτησης με πολλές χιλιάδες μεταβλητές (τις εισόδους και τα βάρη). Εάν αναπαρασταθεί σε έναν διανυσματικό χώρο, η συνάρτηση έχει την εικόνα ενός «γεωγραφικού τοπίου» στο οποίο πρέπει να βρεθεί ένα τοπικό «χαμηλότερο σημείο», όπως φαίνεται και στην ***Εικόνα 2***. Χρησιμοποιώντας μαθηματικές έννοιες, όλα τα παραπάνω καταλήγουν τελικά στην εύρεση του αρνητικού του «ανάδελτα» ή ∇ της συνάρτησης κόστους, το οποίο δείχνει προς την κατεύθυνση της πιο απότομης μείωσης της συνάρτησης (Sanderson, 2017).

Εικόνα 3. Παράδειγμα μαθηματικής αναπαράστασης του ανάδελτα



Σημείωση: Ανάκτηση από «Gradient descent, how neural networks learn», G. Sanderson--3blue1brown,2017 (σύνδεσμος).

Κατά συνέπεια, σε αυτό το σημείο θα αναφερθεί περιληπτικά ο ορισμός του αλγορίθμου που προσπαθεί να επιτύχει όλα τα παραπάνω, δηλαδή να υπολογίσει αυτό το τοπικό ελάχιστο της συνάρτησης κόστους, ο οποίος αναφέρεται ως «**αλγόριθμος ανάστροφης μετάδοσης**».

2.1.3 Ορισμός του αλγορίθμου Ανάστροφης Μετάδοσης

Όπως παρουσιάστηκε παραπάνω, ο αλγόριθμος ανάστροφης μετάδοσης είναι ένας αλγόριθμος που βρίσκει το τοπικό ελάχιστο μέσω του υπολογισμού του αρνητικού ανάδελτα μίας

συνάρτησης κόστους. Αρχικά, ο αλγόριθμος αρχικοποιεί όλα τα βάρη του νευρωνικού δικτύου με τυχαίες μικρές τιμές. Για μία δεδομένη είσοδο ή παράδειγμα εκπαίδευσης, υπολογίζει το συνολικό σφάλμα ή τη συνάρτηση κόστους στην τελική έξοδο, συγκρίνοντας την πραγματική έξοδο με την επιθυμητή έξοδο. Στην συνέχεια, το σφάλμα μεταδίδεται από την έξοδο προς την είσοδο, υπολογίζοντας παράλληλα τους παραγώγους ως προς κάθε ξεχωριστό βάρος με τον κανόνα της αλυσίδας. Κάθε βάρος ενημερώνεται ώστε να κατευθύνεται, με την χρήση της καθοδικής κλίσης, προς την μείωση του σφάλματος. Η διαδικασία αυτή επαναλαμβάνεται για κάθε παράδειγμα εκπαίδευσης που δίνεται στο νευρωνικό δίκτυο, οι οποίες ονομάζονται «εποχές». Η εκπαίδευση τελειώνει είτε όταν το σύστημα ξεπεράσει έναν μέγιστο αριθμό εποχών, είτε όταν το συνολικό σφάλμα μειωθεί σε έναν επιθυμητό αριθμό (Ανδρουτσόπουλος, 2023).

2.2 Τι είναι ένα μεγάλο γλωσσικό μοντέλο

2.2.1 Ορισμός Μεγάλου Γλωσσικού Μοντέλου

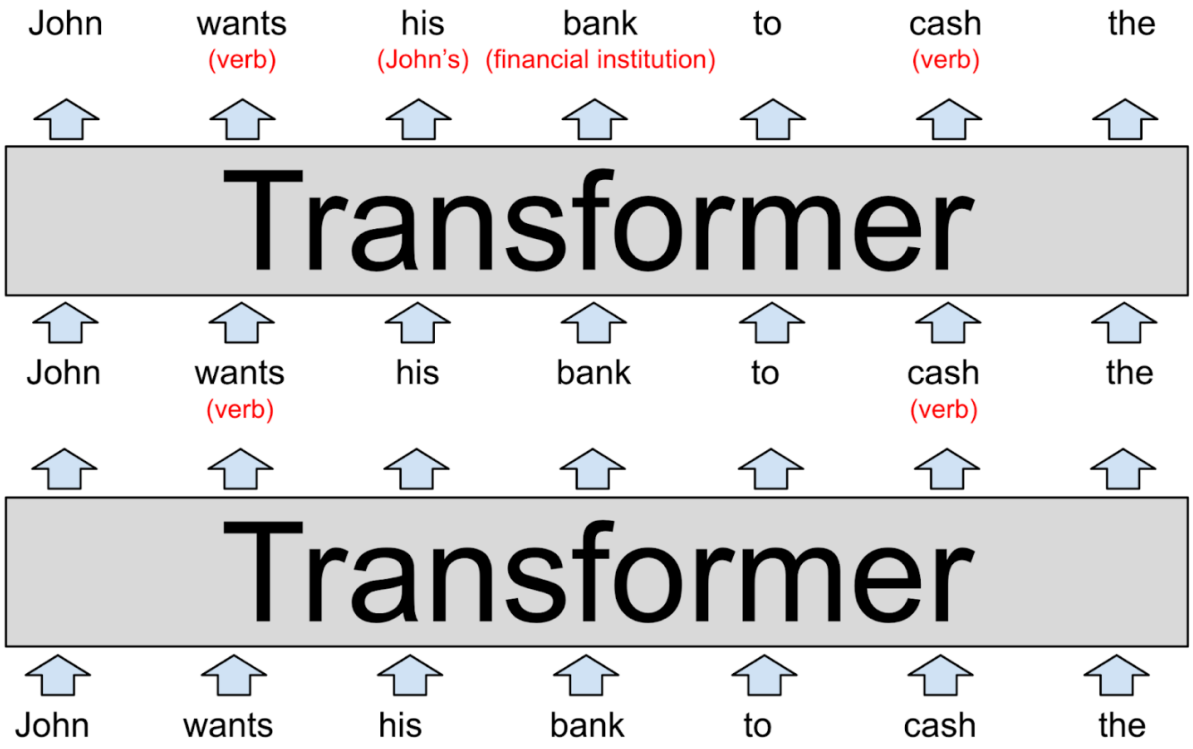
Όπως αναφέρει ο Ανδρουτσόπουλος(2024), ένα μεγάλο γλωσσικό μοντέλο είναι ένα νευρωνικό δίκτυο που δέχεται ως εισόδους λέξεις ή καλύτερα «tokens», σε μορφή αριθμών, τα οποία μπορεί να αποτελούν ένα ημιτελές κείμενο, και παράγει ως εξόδους όλες τις πιθανές λέξεις που θα μπορούσαν να είναι η επόμενη λέξη, επιλέγοντας αυτήν με την μεγαλύτερη πιθανότητα ορθότητας. Έτσι, βασιζόμενοι στο μεγάλο γλωσσικό μοντέλο, μπορούμε να πάρουμε την νέα πρόταση που παρήγαγε, να την ξαναδώσουμε σαν είσοδο και να παράγει ξανά άλλη μία επόμενη πιθανή λέξη. Μετά από έναν αριθμό επαναλήψεων της διαδικασίας αυτής, παράγεται μια ολοκληρωμένη απάντηση που βγάζει νόημα.

2.2.2 Τι κάνει ένα Transformer

Πιο συγκεκριμένα, τα μεγάλα γλωσσικά μοντέλα διασπούν το κείμενο που δέχονται ως είσοδο σε διαφορετικά tokens, τα οποία δεν είναι απαραίτητα λέξεις, αλλά μπορεί να είναι και κομμάτια λέξεων, και στην συνέχεια τα αναπαριστούν σε διανύσματα. Τα διανύσματα αυτά αποτελούνται από χιλιάδες μεταβλητές και αριθμούς και τοποθετούνται σε έναν πολυδιάστατο χώρο, όπου λέξεις με παρόμοια σημασία, όπως «γάτα» και «σκύλος», βρίσκονται πολύ κοντά. Η χρήση διανυσμάτων επιτρέπει στα μεγάλα γλωσσικά μοντέλα να πραγματοποιούν μαθηματικές πράξεις που αποκαλύπτουν σχέσεις μεταξύ λέξεων. Για παράδειγμα έχει διαπιστωθεί ότι αν από το διάνυσμα της λέξης "μεγαλύτερος" αφαιρεθεί το διάνυσμα της λέξης "μεγάλος" και προστεθεί το διάνυσμα της λέξης "μικρός", το αποτέλεσμα θα είναι το διάνυσμα της λέξης "μικρότερος".

Τα μεγάλα γλωσσικά μοντέλα μπορούν να αναπαριστούν τις λέξεις με διαφορετικά διανύσματα ανάλογα με τα συμφραζόμενα. Αυτό επιτυγχάνεται με τη χρήση ενός αρχιτεκτονικού μοντέλου νευρωνικού δικτύου, γνωστού ως «**transformer»,** που ενημερώνει τα διανύσματα των λέξεων μέσω πολλαπλών επιπέδων. Κάθε μεγάλο γλωσσικό μοντέλο αποτελείται από πολλά στρώματα transformers συνδεδεμένα μεταξύ τους, με σκοπό να εμπλουτίζουν κάθε token με πληροφορίες βάσει των συμφραζομένων, τροποποιώντας έτσι το διάνυσμα του ώστε να έχει την ορθή σημασία που αποκαλύπτεται από τα συμφραζόμενα. Κάθε transformer προσθέτει πληροφορία

Εικόνα 4. Παράδειγμα επικοινωνίας των transformers σε ένα μεγάλο γλωσσικό μοντέλο



Σημείωση: Ανάκτηση από «Large Language Models, explained with a minimum of math and jargon», T. Lee and S. Trott, 2023, (σύνδεσμος).

και βελτιώνει τα διανύσματα έως το τελικό στρώμα (Lee, 2023).

2.2.3 Τι είναι το Attention layer

Ένα κύριο συστατικό που επιτρέπει στο transformer να εμπλουτίζει τα διανύσματα των λέξεων είναι το **«attention layer».** Συγκεκριμένα, το στοιχείο αυτό επιτρέπει στα tokens να ανταλλάσσουν πληροφορίες μεταξύ τους και να εμπλουτίζουν το ένα το άλλο, έτσι ώστε να προκύπτει η σωστή σημασία κάθε λέξης. Για παράδειγμα, η λέξη «μοντέλο» έχει διαφορετική

σημασία ανάλογα τα συμφραζόμενα, όπως στις φράσεις «μοντέλο μαθηματικών» και «μοντέλο του Χόλυγουντ». Η λειτουργία του attention layer είναι να πραγματοποιεί τον σωστό αυτό διαχωρισμό για κάθε διαφορετικό token με βάση τις λέξεις που το περιτριγυρίζουν. Η διαδικασία αυτή επιτρέπει στα μεγάλα γλωσσικά μοντέλα να μαντεύουν σωστά την επόμενη λέξη σε κάθε κείμενο (Lee, 2023)

2.2.4 Τι είναι το Feed Forward Step

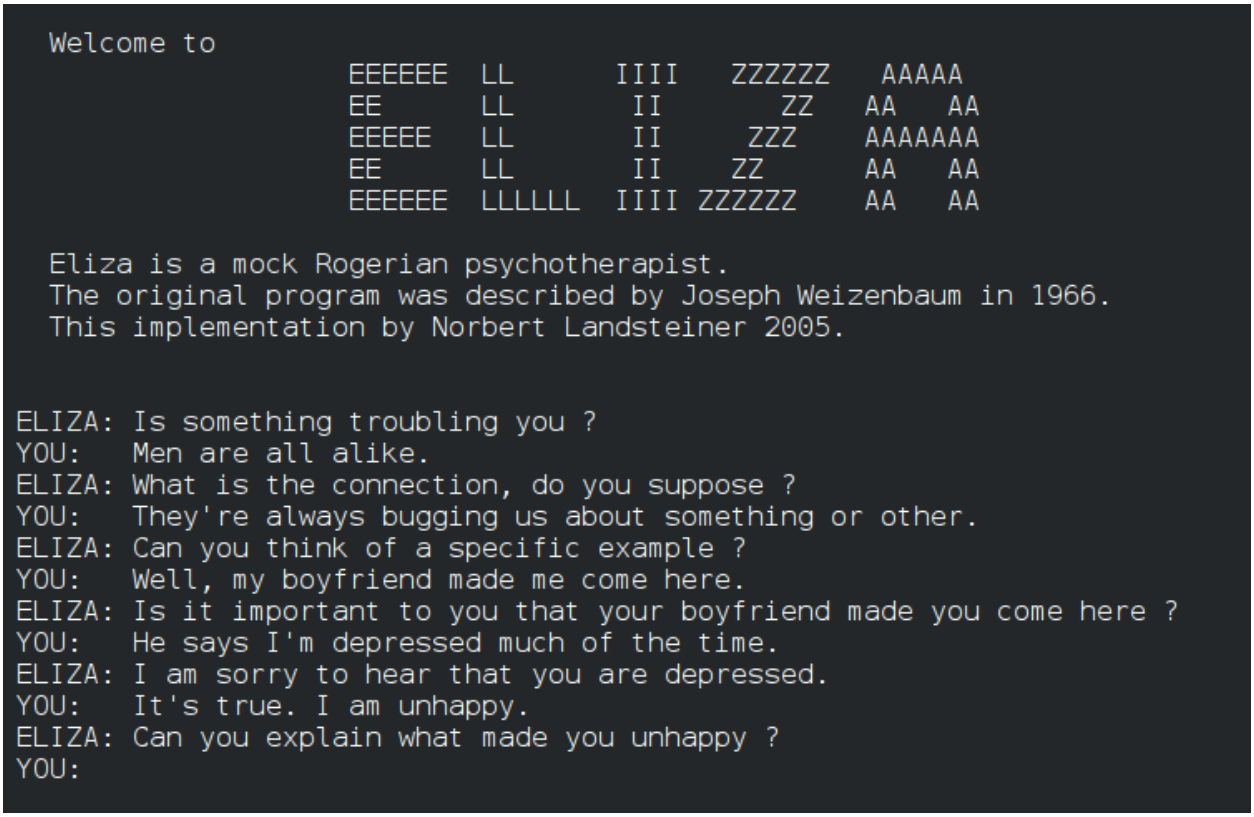
Μετά τη μεταφορά πληροφοριών ανάμεσα στα διανύσματα λέξεων από τα attention heads, το transformer περιλαμβάνει ένα ακόμα συστατικό, το στρώμα **«feed forward»**. Το στρώμα αυτό «εξετάζει» κάθε διάνυσμα λέξης και προσπαθεί να προβλέψει την επόμενη λέξη. Σε αυτό το στάδιο, δεν γίνεται ανταλλαγή πληροφοριών μεταξύ των λέξεων, το στρώμα feed forward αναλύει κάθε λέξη μεμονωμένα. Ωστόσο, το στρώμα feed forward έχει πρόσβαση σε οποιαδήποτε πληροφορία έχει αντιγραφεί από όλα τα προηγούμενα attention heads, ώστε να μπορέσει αποτελεσματικότερα να προβλέψει την επόμενη λέξη (Lee, 2023).

2.3 Ιστορία και εξέλιξη των μεγάλων γλωσσικών μοντέλων

2.3.1 Το chatbot ELIZA

Ξεκινώντας μια ιστορική αναδρομή της εξέλιξης των μεγάλων γλωσσικών μοντέλων, θα αναφερθούμε στο chatbot **«ELIZA»**, το οποίο κατασκευάστηκε το 1996 και θεωρείται το πρώτο chatbot που δημιουργήθηκε από ανθρώπους. Ο δημιουργός του, Joseph Weizenbaum, τοι ανέπτυξε στο πανεπιστήμιο του MIT. Ο τρόπος λειτουργίας του ELIZA, ήταν να δημιουργήσει την ψευδαίσθηση συνομιλίας με την τεχνική της αναδιατύπωσης δηλώσεων των χρηστών ως ερωτήσεις. Εκείνη την εποχή, δημιουργήθηκαν πολλές παραλλαγές του συγκεκριμένου chatbot οι οποίες λειτουργούσαν με παρόμοιο τρόπο και μια από τις πιο γνωστές ονομάζεται «DOCTOR», το οποίο ανταποκρινόταν σαν ψυχοθεραπευτής. Αυτή η αρχή έβαλε τις βάσεις για περεταίρω έρευνα στον τομέα των chatbots και της επεξεργασίας φυσικής γλώσσας. (Pi, 2024)

Εικόνα 5. Ένα παράδειγμα συζήτησης με το chatbot ELIZA



Σημείωση: Ανάκτηση από «ELIZA», In Wikipedia, The Free Encyclopedia (2024, July 15), (σύνδεσμος).

2.3.2 Άνοδος των Νευρωνικών Δικτύων

Όπως αναφέρει ο Pi(2024) στο άρθρο του, προς τα τέλη του 20ου αιώνα εμφανίστηκαν τα νευρωνικά δίκτυα, εμπνευσμένα βαθιά από τον ανθρώπινο εγκέφαλο, όπως γίνεται φανερό από την ονομασία τους αλλά και την αρχιτεκτονική τους με διασυνδεδεμένους νευρώνες. Το 1986, αναφέρεται ως η χρονιά που έκαναν την εμφάνισή τους τα «Επαναλαμβανόμενα Νευρωνικά Δίκτυα (RNN), τα οποία σε αντίθεση με τα παραδοσιακά προωθητικά νευρωνικά δίκτυα, όπου η ροή των πληροφοριών είχε μονάχα μία κατεύθυνση, μπορούσαν να θυμούνται προηγούμενες εισόδους και να απαντούν με βάση το ευρύτερο πλαίσιο. Έτσι, εκπαιδεύονταν να επεξεργάζονται και να μετατρέπουν μια ακολουθία δεδομένων εισόδου σε συγκεκριμένη ακολουθία δεδομένων εξόδου. Ωστόσο, τα RNN είχαν τον πολύ μεγάλο περιορισμό στην «μνήμη», κάτι αντίστοιχο με το σημερινό context size των μεγάλων γλωσσικών μοντέλων όπως του ChatGPT, το οποίο τα κάνει να φαίνονται σαν να «ξεχνούν» πληροφορίες από προηγούμενα μηνύματα.

2.3.3 Δημιουργία των LSTM

Το 1997 εμφανίστηκε η Μνήμη Μακράς Βραχείας Διάρκειας (LSTM), μια εξειδικευμένη μορφή RNN που βελτίωνε το πρόβλημα της διατήρησης πληροφορίας για μεγάλες ακολουθίες. Συγκεκριμένα, το εργαλείο αυτό είχε μια μοναδική αρχιτεκτονική που αποτελούνταν από πύλες εισόδου, λήθης και πύλες εξόδου, επιτρέποντας έτσι τη διατήρηση και τη διαχείριση πληροφοριών για μεγαλύτερα χρονικά διαστήματα (Pi, 2024).

2.3.4 Δημιουργία Gated Recurrent Network

Το 2014, εμφανίστηκαν οι Μονάδες Επαναλαμβανόμενων Δικτύων με Πύλες (GRU), σχεδιασμένες για να επιλύουν τα ίδια προβλήματα με τα LSTM, αλλά με απλούστερη δομή. Οι GRU χρησιμοποιούσαν δύο πύλες: μια πύλη ενημέρωσης και μια πύλη επαναφοράς. (Pi, 2024)

2.3.5 Άνοδος του συστατικού Attention

Τελικά, οι τεχνολογίες RNN, LSTM και GRU αποδείχτηκε ότι δεν είναι τόσο αποτελεσματικές στην διατήρηση των συμφραζόμενων όταν αυτά επεκτείνονται σε μεγάλο βαθμό. Έτσι, δημιουργήθηκε ο μηχανισμός που ονομάστηκε **«Attention»**, ο οποίος προσέφερε μία νέα οπτική στα μεγάλα γλωσσικά μοντέλα. Συγκεκριμένα, το attention επέτρεψε στο μοντέλο να «κοιτάει»

πίσω σε ολόκληρο το διαθέσιμο υλικό δυναμικά, επιλέγοντας τα πιο σημαντικά κομμάτια που προσθέτουν σημασία στις υπόλοιπες προτάσεις. (Pi, 2024)

Εικόνα 6. Σύγκριση της απόδοσης Attention και RNN

Σημείωση: Ανάκτηση από «Brief Introduction to the History of Large Language Models (LLMs)», W. P,2024 (σύνδεσμος)

2.3.6 Η εφεύρεση των Transformers

Το 2017 ήταν το έτος που πρωτοεμφανίστηκε η έννοια των **«Transformers»,** στο paper με τίτλο «Attention is all you need» από τον Vaswani και τους συνεργάτες του στην Google. Η νέα αυτή αρχιτεκτονική, όπως εξηγήθηκε και στην προηγούμενη υποενότητα, χρησιμοποιούσε ως βασικό της εργαλείο τον μηχανισμό attention για να επεξεργαστεί τα δεδομένα εισόδου και ήταν ικανή να επεξεργάζεται ακολουθίες παράλληλα, χωρισμένη σε πολλά στρώματα. Αυτή η προσέγγιση έθεσε τα θεμέλια για μεταγενέστερα μοντέλα όπως το ChatGPT (Pi, 2024).

2.3.7 Εμφάνιση Μεγάλων Γλωσσικών Μοντέλων

Με την μεγάλη επιτυχία των transformers, το επόμενο λογικό βήμα ήταν η αύξηση της κλίμακας. Αυτό ξεκίνησε με το μοντέλο BERT της Google το 2018 και συνεχίστηκε με την

κυκλοφορία των GPT-2 το 2019,του GPT-3 το 2020, καθώς και των νέων εκδόσεων αυτού όπως το GPT-3.5, GPT-4 και GPT-4o (Pi, 2024).

2.4 Κύριες εφαρμογές και χρήσεις

Με την τεράστια εξέλιξη των μεγάλων γλωσσικών μοντέλων, αυτά έχουν γίνει καθημερινό εργαλείο για πολλούς ανθρώπους, εξοικονομώντας χρόνο, καθώς το μέγεθός τους από δεδομένα και η πολυπλοκότητά τους επιτρέπει σε στιγμιαίες απαντήσεις χωρίς καθυστέρηση. Το κόστος τους είναι ελάχιστο, αφού τα περισσότερα είναι διαθέσιμα δωρεάν σε απλούστερες αλλά πολύ αποτελεσματικές μορφές, ενώ η χρήση τους απαιτεί ελάχιστες γνώσεις. Όλα τα παραπάνω έχουν ωθήσει τα μεγάλα γλωσσικά μοντέλα να βρουν πολυάριθμες χρήσεις σε διάφορους τομείς. Όπως αναφέρει και ο Sumrak(2024), κάποια από τα δημοφιλέστερα παραδείγματα χρήσης των μεγάλων γλωσσικών μοντέλων περιλαμβάνουν την ανάλυση ήχου, τη δημιουργία περιεχομένου, την υποστήριξη πελατών, τη μετάφραση γλωσσών, την εκπαίδευση και την κυβερνοασφάλεια.

2.4.1 Ανάλυση ήχου

Τα μεγάλα γλωσσικά μοντέλα έχουν αναθεωρήσει τον τρόπο με τον οποίο οι άνθρωποι χειρίζονται τα δεδομένα ήχου. Συγκεκριμένα, έχουν την ικανότητα να ακούν πολύωρες συζητήσεις και να παράγουν αποτελεσματικές περιλήψεις καθώς και να απαντούν ερωτήσεις σχετικά με μακροσκελείς συναντήσεις. Ακόμη, μπορούν, με βάση ένα μεγάλο ποσοστό κλήσεων ως δεδομένα, να εξάγουν πολύπλοκα αποτελέσματα και συμβουλές βελτίωσης (Sumrak, 2024).

2.4.2 Δημιουργία περιεχομένου

Τα μεγάλα γλωσσικά μοντέλα χρησιμοποιούνται αποτελεσματικά από συγγραφείς και εμπόρους για την δημιουργία αρχικών σχεδίων(drafts), την πρόταση διάφορων αλλαγών και για την ταχεία εύρεση άρθρων και αναφορών στο διαδίκτυο. Αυτά τα εργαλεία επιταχύνουν σημαντικά την παραγωγικότητα, επιτρέποντάς στους χρήστες να επικεντρωθούν παραπάνω στο πιο απαιτητικό και δημιουργικό κομμάτι της δουλείας τους, αφήνοντας τα μοντέλα να ασχοληθούν με τα μηχανικά στοιχεία της εργασίας τους (Sumrak, 2024).

2.4.3 Υποστήριξη πελατών

Ένας τομέας στον οποίο τα μεγάλα γλωσσικά μοντέλα έχουν κάνει ραγδαία άνοδο και χρησιμοποιούνται κατά κόρον, είναι η εξυπηρέτηση πελατών. Πολλές εταιρείες τηλεφωνίας, σελίδες του δημοσίου τομέα και μεγάλες επιχειρήσεις έχουν υιοθετήσει μοντέλα εξυπηρέτησης

πελατών μέσω μεγάλων γλωσσικών μοντέλων. Αυτά είναι διαθέσιμα 24/7 χωρίς την ανάγκη ανθρώπινης παρέμβασης, προσφέροντας συνεχή βοήθεια σε χρήστες από όλο τον κόσμο με πολύ χαμηλό κόστος (Sumrak, 2024).

2.4.4 Μετάφραση γλωσσών

Τα μεγάλα γλωσσικά μοντέλα βοηθούν στην άρση των γλωσσικών φραγμών, δίνοντας την δυνατότητα στις επιχειρήσεις να προσεγγίζουν πελάτες αλλά και να προσλαμβάνουν άτομα από όλες τις χώρες. Αυτά τα μοντέλα προσφέρουν ακριβείς υπηρεσίες μετάφρασης σε πραγματικό χρόνο, κάνοντας ιστότοπους, εφαρμογές και ψηφιακό περιεχόμενο παγκοσμίως προσβάσιμα (Sumrak, 2024).

2.4.5 Εκπαίδευση

Στον τομέα της εκπαίδευσης, τα μεγάλα γλωσσικά μοντέλα χρησιμοποιούνται για την παροχή εξατομικευμένης μάθησης προσαρμόζοντας το περιεχόμενο στις ατομικές ανάγκες των μαθητών, δημιουργώντας ερωτήσεις κατανόησης και παρέχοντας λεπτομερείς εξηγήσεις προσαρμοσμένες στις ανάγκες των μαθητών (Sumrak, 2024).

2.4.6 Κυβερνοασφάλεια

Τα μεγάλα γλωσσικά μοντέλα μπορούν να χρησιμοποιηθούν στην ανάλυση και την ερμηνεία μεγάλων ποσών δεδομένων κυβερνοασφάλειας, προβλέποντας, αναγνωρίζοντας και ανταποκρινόμενα σε πιθανές απειλές ασφαλείας. Η στοχευμένης εκπαίδευσή τους επιτρέπει ταχύτερη και πιο ακριβή ανίχνευση και ανταπόκριση στις απειλές, ενισχύοντας την ασφάλεια των επιχειρήσεων (Sumrak, 2024).

2.5 Παραδείγματα Μεγάλων Γλωσσικών Μοντέλων

Ολοκληρώνοντας την ανάλυση των μεγάλων γλωσσικών μοντέλων, είναι σημαντικό να αναφερθούν κάποια από τα πιο πολυχρησιμοποιούμενα μοντέλα και οι κυριότερες λειτουργίες και διαφοροποιήσεις τους.

2.5.1 BERT

Το πρώτο μεγάλο γλωσσικό μοντέλο που αναπτύχθηκε ήταν το **BERT**(Bidirectional Encoder Representations from Transformers), δημιουργημένο από την Google το έτος 2018. Το BERT είναι ένα μοντέλο που χρησιμοποιεί την αρχιτεκτονική των transformers με πολυάριθμα

στρώματα συνδεδεμένα μεταξύ τους και διαθέτει 342 εκατομμύρια παραμέτρους για την επεξεργασία εισόδων. Εκπαιδεύτηκε σε τεράστιο όγκο δεδομένων για να παράγει απαντήσεις σε φυσική γλώσσα που να είναι κατανοητές από τους ανθρώπους (Lutkevich, 2024).

2.5.2 GEMINI

Ακολουθώντας την Google, το ανανεωμένο μοντέλο της ονομάστηκε **GEMINI**, το οποίο έχει την ονομασία του από το chatbot της εταιρείας. Το GEMINI είναι ένα πολυτροπικό μοντέλο, ικανό να επεξεργάζεται ήχο, εικόνα, βίντεο και κείμενο, σε αντίθεση με άλλα γλωσσικά μοντέλα που περιορίζονται μόνο σε κείμενο. Το GEMINI διατίθεται σε τρία «μεγέθη», το Ultra, το Pro και το Nano, από το μεγαλύτερο και πιο ικανό έως το μικρότερο και λιγότερο ικανό. Υπάρχουν αναφορές ότι το GEMINI υπερτερεί σε δύναμη από το μοντέλο GPT -4 της OpenAI, στο οποίο θα γίνει αναφορά στην συνέχεια. (Lutkevich, 2024).

2.5.3 GPT –3

Το **GPT -3**(Generative Pre-trained Transformer 3) αποτελεί το πρώτο ισχυρό μοντέλο της OpenAI το οποίο παρουσιάστηκε το 2020 με περισσότερες από 175 δισεκατομμύρια παραμέτρους (το BERT έχει 342 εκατομμύρια). Ενσωματώνει την αρχιτεκτονική των συνδεδεμένων στρωμάτων transformers και είναι δέκα φορές μεγαλύτερο από τον προκάτοχο του, το GPT -2. Είναι εκπαιδευμένο με εκατομμύρια δεδομένα και αποτελεί το τελευταίο μοντέλο από την σειρά μοντέλων που παρήγαγε η OpenAI, για το οποίο ήταν δημόσια γνωστός ο αριθμός των παραμέτρων που χρησιμοποιούσε (Lutkevich, 2024).

2.5.4 GPT -3.5 και GPT –3.5 Turbo

Το **GPT –3.5** αποτελεί την ενημερωμένη έκδοση του GPT –3 με λιγότερες παραμέτρους αλλά ποιοτικότερη εκπαίδευση, και διαδραμάτισε κρίσιμο ρόλο στην πλατφόρμα ChatGPT, η οποία το 2023 άλλαξε ριζικά τη δημοτικότητα και χρήση των μεγάλων γλωσσικών μοντέλων. Το GPT –3.5 εκπαιδεύτηκε με γνώσεις μέχρι και τον Σεπτέμβρη του 2021 και δεν έχει την δυνατότητα πρόσβασης στο διαδίκτυο συγκριτικά με άλλα μεγάλα γλωσσικά μοντέλα Παράλληλα, η πιο ισχυρή έκδοση του GPT-3.5 είναι το **GPT-3.5 Turbo**, που χρησιμοποιείται από το GitHub Copilot της Microsoft για την υποστήριξη των προγραμματιστών με παραγωγή κώδικα και ανίχνευση σφαλμάτων (Lutkevich, 2024).

2.5.5 GPT –4

Το **GPT-4** αποτελεί το μεγαλύτερο μοντέλο της σειράς GPT της OpenAI, κυκλοφορώντας το 2023. Όπως και τα προηγούμενα μοντέλα, βασίζεται στην αρχιτεκτονική των transformers. Ωστόσο, ο αριθμός των παραμέτρων του δεν έχει δημοσιοποιηθεί, με φήμες να αναφέρουν πάνω από 170 **τρισεκατομμύρια** παραμέτρους. Η OpenAI περιγράφει το GPT-4 ως ένα πολυτροπικό μοντέλο(όπως και το GEMINI), πράγμα που σημαίνει ότι είναι ικανό να επεξεργάζεται κείμενο, ήχο και εικόνα, σε αντίθεση με τα προηγούμενα μοντέλα της OpenAI που περιορίζονταν μόνο στο κείμενο. Το GPT-4, υποστηρίζεται ότι πλησίασε την τεχνητή γενική νοημοσύνη (AGI), που σημαίνει ότι είναι εξίσου έξυπνο ή εξυπνότερο από έναν άνθρωπο και σημαντικότερα έχει την δυνατότητα της πρόσβασης στο διαδίκτυο, σε αντίθεση με το GPT –3.5 (Lutkevich, 2024).

2.5.6 GPT –4o

Το **GPT -4o (**GPT -4 Omni)είναι ο διάδοχος του GPT-4 και το πιο πρόσφατο μεγάλο γλωσσικό μοντέλο της OpenAI . Προσφέρει αρκετές βελτιώσεις σε σχέση με τοGPT -4 με μια πιο φυσική ανθρώπινη αλληλεπίδραση για το ChatGPT. Το GPT-4o είναι επίσης πολυτροπικό, αλλά με τη δυνατότητα να βλέπει φωτογραφίες ή οθόνες και να κάνει σχετικές ερωτήσεις κατά τη διάρκεια της αλληλεπίδρασης.

3. ΜΕΓΑΛΑ ΓΛΩΣΣΙΚΑ ΜΟΝΤΕΛΑ ΣΤΗΝ ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ

3.1 Ρόλος των μεγάλων γλωσσικών μοντέλων στην ανάπτυξη λογισμικού

3.1.1 Εισαγωγή στην έννοια των γλωσσικών μοντέλων στην ανάπτυξη λογισμικού

Τα μεγάλα γλωσσικά μοντέλα είναι προηγμένα συστήματα τεχνητής νοημοσύνης που ανταποκρίνονται σε ερωτήσεις και συζητήσεις χρηστών με απαντήσεις σε ανθρώπινη γλώσσα. Έχουν την ικανότητα να κατανοούν βαθιά τα συμφραζόμενα και τις ανάγκες του χρήστη, κάνοντάς τα χρήσιμα σε πολλούς τομείς της καθημερινότητας και ιδιαίτερα στην ανάπτυξη λογισμικού. Τα γλωσσικά μοντέλα μπορούν να βοηθήσουν προγραμματιστές, αυξάνοντας την παραγωγικότητά τους και μειώνοντας τον χρόνο εκσφαλμάτωσης, καθώς έχουν εκπαιδευτεί σε δισεκατομμύρια έργα ανοιχτού κώδικα και ενημερώνονται συνεχώς με νέα δεδομένα.

Ο εσωτερικός τρόπος λειτουργίας των μεγάλων γλωσσικών μοντέλων τα καθιστά εξαιρετικά στην υποβοήθηση πάρα πολλών έργων και στην λύση των δυσκολιών που μπορεί να προκύψουν για έναν προγραμματιστή κατά την διάρκεια της διαδικασίας της ανάπτυξης λογισμικού, αυξάνοντας έτσι σημαντικά την παραγωγικότητα, μειώνοντας τον χρόνο εκσφαλμάτωσης και συνεπώς και τον χρόνο που απαιτείται για την ολοκλήρωση ενός έργου. Όλα αυτά οφείλονται στο γεγονός ότι τα μεγάλα γλωσσικά μοντέλα έχουν την ικανότητα να γράφουν, να διορθώνουν και να βελτιστοποιούν τον κώδικα πολύ γρήγορα και με μεγάλη ακρίβεια, καθώς έχουν εκπαιδευτεί σε δισεκατομμύρια έργα ανοιχτού κώδικα, όπως αυτά στο GitHub, ενώ παράλληλα ενημερώνονται συνεχώς με νέα δεδομένα.

3.1.2 Παραδείγματα χρήσης μεγάλων γλωσσικών μοντέλων στην ανάπτυξη λογισμικού

Ορισμένα από τα σημαντικότερα παραδείγματα χρήσης των μεγάλων γλωσσικών μοντέλων στην ανάπτυξη λογισμικού περιλαμβάνουν την αυτοματοποίηση γραφής κώδικα, την ανάλυση και διόρθωση λαθών καθώς και την ανάλυση των απαιτήσεων και την μετατροπή τους σε τεχνικές προδιαγραφές.

Η αυτοματοποίηση γραφής κώδικα μπορεί να εξοικονομήσει πολύ χρόνο και να μειώσει τα λάθη, καθώς τα γλωσσικά μοντέλα μπορούν να δημιουργούν κώδικα βάσει των περιγραφών των προγραμματιστών και να προτείνουν βελτιώσεις. Η ανάλυση και η διόρθωση λαθών επιτρέπει στους προγραμματιστές να εντοπίζουν και να διορθώνουν σφάλματα γρήγορα και αποτελεσματικά και η μετατροπή των απαιτήσεων σε τεχνικές προδιαγραφές διευκολύνει την κατανόηση των αναγκών του έργου και την ακριβή μετάφρασή τους σε τεχνικές λεπτομέρειες.

3.1.3 Αυτοματοποίηση κώδικα

Στον τομέα του αυτοματοποιημένου κώδικα, τα μεγάλα γλωσσικά μοντέλα μπορούν να προτείνουν ή να γράφουν αποσπάσματα κώδικα βάσει των περιγραφών των προγραμματιστών. Παρέχουν επίσης προτάσεις κώδικα και σχόλια σε πραγματικό χρόνο, όπως το GitHub Copilot της Microsoft, το οποίο αποτελεί ένα από τα μοντέλα που εξετάστηκαν στην εργασία.

Τα μεγάλα γλωσσικά μοντέλα μπορούν να κατανοούν περιγραφές σε φυσική γλώσσα και να μετατρέπουν αυτές τις περιγραφές σε λειτουργικό κώδικα, με αποτέλεσμα να μειώνεται η ανάγκη για λεπτομερή καθοδήγηση και να επιτρέπει στους προγραμματιστές να εστιάσουν σε πιο δημιουργικές και πολύπλοκες πτυχές της ανάπτυξης λογισμικού. Η ικανότητα να παρέχουν

συνεχείς ενημερώσεις και βελτιώσεις στον κώδικα καθιστά τα μεγάλα γλωσσικά μοντέλα έναν πολύτιμο συνεργάτη στην ανάπτυξη λογισμικού.

3.1.4 Ανάλυση και διόρθωση λαθών

Τα μεγάλα γλωσσικά μοντέλα είναι ικανά να αναγνωρίζουν με μεγάλη ευκολία και ταχύτητα σφάλματα που έχουν συμβεί κατά την διάρκεια της συγγραφής κώδικα, να τα αναλύουν και να προσφέρουν λύσεις με βάση πολύπλοκες μεθόδους εκσφαλμάτωσης, ενώ μπορούν να δίνουν και τις ίδιες τις διορθώσεις σε όλα τα πιθανά λάθη που έχουν εντοπιστεί. Ακόμη, βελτιώνουν την ποιότητα του κώδικα εντοπίζοντας σημεία που επιδέχονται βελτίωση και παρέχοντας προτάσεις για πιο ποιοτικό αποτέλεσμα.

Η ικανότητα των μεγάλων γλωσσικών μοντέλων να κατανοούν σε τόσο μεγάλο βάθος τον κώδικα, τους επιτρέπει να εντοπίζουν λάθη που μπορεί να μην είναι προφανή άμεσα σε έναν προγραμματιστή. Αυτό μπορεί να περιλαμβάνει σφάλματα σύνταξης, λογικά σφάλματα ή ακόμη και προβλήματα βελτιστοποίησης. Η παροχή συγκεκριμένων προτάσεων για τη διόρθωση αυτών των λαθών βοηθά στην εξασφάλιση της ποιότητας και της αξιοπιστίας του κώδικα.

3.1.5 Ανάλυση των απαιτήσεων μετατροπή τους σε τεχνικές προδιαγραφές

Τα μεγάλα γλωσσικά μοντέλα μπορούν να διαβάζουν και να κατανοούν τα διάφορα έγγραφα απαιτήσεων, να τα διορθώνουν και να χρησιμοποιούν φυσική γλώσσα για να εξάγουν τις βασικές ανάγκες και προδιαγραφές που πρέπει να περιγραφούν στους εμπλεκόμενους ενός μεγάλου έργου.

Η κατανόηση και η μετάφραση των απαιτήσεων σε τεχνικές προδιαγραφές είναι ένα κρίσιμο βήμα στην διαδικασία της ανάπτυξης λογισμικού. Συγκεκριμένα, τα μεγάλα γλωσσικά μοντέλα μπορούν να αναλύουν πολύπλοκες περιγραφές και να τις μετατρέπουν σε κατανοητές προδιαγραφές που μπορούν να χρησιμοποιηθούν από όλα τα εμπλεκόμενα μέρη. Συνεπώς αυτό μειώνει την πιθανότητα ασαφειών(έννοια που περιγράφεται σε επόμενη υποενότητα) και εξασφαλίζει ότι όλες οι απαιτήσεις πληρούνται με ακρίβεια.

3.2 Χρήσεις σε διάφορα στάδια της ανάπτυξης λογισμικού

3.2.1 Στάδιο Ανάλυσης Απαιτήσεων

Η ανάλυση των απαιτήσεων είναι μία κρίσιμη διαδικασία για την επιτυχή ανάπτυξη ενός συστήματος λογισμικού. Ένα σύνηθες πρόβλημα κατά το στάδιο της ανάλυσης των απαιτήσεων, είναι η ύπαρξη ασαφειών, δηλαδή η διαφορετική ερμηνεία της ίδιας απαίτησης από πολλά άτομα, το οποίο μπορεί να οδηγήσει σε σοβαρά προβλήματα στα υπόλοιπα στάδια της ανάπτυξης λογισμικού. Σύμφωνα με τον Hou και συν. (2023), το ChatGPT, υπερτερεί σε μεγάλο βαθμό σε σχέση με τα άλλα γλωσσικά μοντέλα. Παράλληλα, όπως ανέδειξε και μια έρευνα που αναφέρει ο Hou και οι συν. στην οποία δόθηκαν κάποιες απαιτήσεις με μεγάλες προκλήσεις στην ασάφεια, το γλωσσικό μοντέλο κατάφερε να εντοπίσει με ακρίβεια τα λάθη και να τα διορθώσει σε κάθε περίπτωση. Επομένως, τα μεγάλα γλωσσικά αποτελούν ένα πολύ σημαντικό εργαλείο στην εξάλειψη των ασαφειών στο στάδιο της ανάλυσης απαιτήσεων, συμβάλλοντας σε ένα ποιοτικότερο αποτέλεσμα.

3.2.2 Στάδιο Σχεδιασμού

Στο στάδιο του σχεδιασμού είναι επίσης κρίσιμο σημείο για την επιτυχία ενός έργου ανάπτυξης λογισμικού. Όπως αναφέρει ο Hou και συν.(2023), ενώ η έρευνα γύρω από τα μεγάλα γλωσσικά μοντέλα στον τομέα του σχεδιασμού δεν έχει προχωρήσει σε βάθος σε σύγκριση με άλλα στάδια της ανάπτυξης, όπως η κωδικοποίηση και η βελτιστοποίηση, τα μεγάλα γλωσσικά μοντέλα προσφέρουν σημαντική υποστήριξη με ποικίλους άλλους τρόπους. Ειδικότερα, τα γλωσσικά μοντέλα μπορούν να βελτιώσουν την υπάρχουσα αρχιτεκτονική και να προτείνουν διάφορες εναλλακτικές λύσεις, εάν αυτό κριθεί απαραίτητο. Για παράδειγμα, μπορούν να προτείνουν συγκεκριμένα πρότυπα σχεδίασης που να ταιριάζουν περισσότερο με τις ανάγκες του έργου.

Ακόμη, τα μεγάλα γλωσσικά μοντέλα μπορούν να συνεισφέρουν στην δημιουργία διαγραμμάτων, όχι μέσω της παραγωγής τους απευθείας αλλά μέσω της περιγραφής μέσω μηνυμάτων, στις περιγραφές συστημάτων και στις τεχνικές προδιαγραφές.

Επιπρόσθετα, τα μεγάλα γλωσσικά μοντέλα μπορούν να εντοπίζουν σημεία όπου μπορεί να υπάρξουν πιθανά σφάλματα απόδοσης ή ασφάλειας και να προτείνουν βελτιώσεις, συμβάλλοντας σε μία αποδοτικότερη και ασφαλέστερη αρχιτεκτονική.

3.2.3 Στάδιο Κωδικοποίησης

Το στάδιο της κωδικοποίησης είναι το πιο γνωστό και συχνά το πιο χρονοβόρο στάδιο στην ανάπτυξη λογισμικού. Σε αυτό το στάδιο, οι προγραμματιστές γράφουν τον κώδικα που θα υλοποιήσει τις απαιτήσεις και τις προδιαγραφές του έργου τους. Τα μεγάλα γλωσσικά μοντέλα παρουσιάζουν συνεχώς βελτιωμένες επιδόσεις με κάθε νέο μοντέλο που κυκλοφορεί. Όπως αναφέρει ο Hou και συν. (2023), μοντέλα όπως τα GPT-4, GPT-3, GPT-3.5, BERT series, Codex, CodeGen, InCoder, Copilot και CodeGeeX έχουν πολύ σημαντικό ρόλο στην κωδικοποίηση. Η εκπαίδευσή τους σε μεγάλους όγκους δεδομένων και συγκεκριμένα κειμένων, τους επιτρέπει να κατανοούν βαθιά τη φυσική γλώσσα και να τη μετατρέπουν σε κώδικα, ενώ παράλληλα τα μοντέλα αποδίδουν εξαιρετικά στην παραγωγή κώδικα με περιγραφές σε φυσική γλώσσα, προτείνοντας διορθώσεις και συμπληρώσεις σε πραγματικό χρόνο.

Ακόμη, τα μοντέλα αυτά συνεισφέρουν και στην δημιουργία τεκμηρίωσης, ένα σημαντικό αλλά συχνά παραμελημένο κομμάτι της ανάπτυξης λογισμικού. Τα γλωσσικά μοντέλα μπορούν να παράγουν αυτόματα σχόλια και τεκμηρίωση για τον κώδικα, διευκολύνοντας την κατανόηση και τη συντήρηση του κώδικα από άλλους προγραμματιστές.

Επιπλέον, τα μεγάλα γλωσσικά μοντέλα υποστηρίζουν πολλές γλώσσες προγραμματισμού, καθιστώντας τα χρήσιμα για πολυγλωσσικά έργα που απαιτούν χρήση πολλών γλωσσών, εφόσον μπορούν να αναγνωρίζουν τα συμφραζόμενα και να προσαρμόζουν τις προτάσεις τους ανάλογα με τη γλώσσα προγραμματισμού που χρησιμοποιείται κάθε φορά.

3.2.4 Στάδιο ελέγχου και βελτιστοποίησης

Το στάδιο του ελέγχου και της βελτιστοποίησης είναι ζωτικής σημασίας για τη διασφάλιση της ποιότητας και της απόδοσης του λογισμικού. Κατά τη διάρκεια αυτού του σταδίου, οι προγραμματιστές και οι δοκιμαστές εντοπίζουν και διορθώνουν σφάλματα, βελτιστοποιούν την απόδοση και διασφαλίζουν ότι το λογισμικό πληροί τις απαιτήσεις των χρηστών. Ο έλεγχος και η βελτιστοποίηση διαρκούν συνήθως αρκετό καιρό, καθώς απαιτείται λεπτομερής έλεγχος όλων των κομματιών κώδικα που έχουν δημιουργηθεί. Επιπλέον, η διαδικασία αυτή συνεχίζεται και μετά την παράδοση του τελικού προϊόντος, εφόσον προκύψουν νέες απαιτήσεις από τους χρήστες και τον εμπλεκόμενους του έργου.

Τα μεγάλα γλωσσικά μοντέλα προσφέρουν πολύτιμη υποστήριξη σε αυτό το στάδιο, μέσω της δημιουργίας αυτοματοποιημένων τεστ, τον εντοπισμό σφαλμάτων και την βελτιστοποίηση του κώδικα.

Συγκεκριμένα, τα μεγάλα γλωσσικά μοντέλα μπορούν να δημιουργούν αυτόματα μονάδες ελέγχου (unit tests) και ενσωματωμένα τεστ (integration tests) βάση του κώδικα που έχει γραφτεί. Με αυτόν τον τρόπο εξασφαλίζεται ότι κάθε κομμάτι κώδικα λειτουργεί σωστά και αποδοτικά.

Ακόμη, τα μεγάλα γλωσσικά μοντέλα μπορούν να αναλύουν μεγάλα κομμάτια κώδικα και να εντοπίζουν πιθανά συντακτικά και λογικά σφάλματα που έχουν συμβεί. Παρέχουν επίσης προτάσεις για τη διόρθωση αυτών των σφαλμάτων, διευκολύνοντας τους προγραμματιστές στη διαδικασία της εκσφαλμάτωσης.

Ακόμη, τα μοντέλα αυτά έχουν την δυνατότητα να εντοπίζουν αναποτελεσματικά κομμάτια κώδικα που μπορεί να δίνουν το επιθυμητό αποτέλεσμα, αλλά όχι με τον βέλτιστο τρόπο, και να προτείνουν βέλτιστες πρακτικές και βελτιώσεις, οδηγώντας σε πιο ποιοτικό κώδικα.

3.3 Πλεονεκτήματα και περιορισμοί

Η χρήση των μεγάλων γλωσσικών μοντέλων στην ανάπτυξη λογισμικού προσφέρει πληθώρα πλεονεκτημάτων και νέες δυνατότητες στους προγραμματιστές, με ορισμένους όμως περιορισμούς στην ορθή χρήση και τον έλεγχο των αποτελεσμάτων.

3.3.1 Πλεονεκτήματα

Καταρχάς, η ικανότητα των μεγάλων γλωσσικών μοντέλων να δημιουργούν με τόσο μεγάλη ταχύτητα κώδικα βελτιώνει την αποδοτικότητα και μειώνει τα σφάλματα κατά τη διαδικασία της κωδικοποίησης, ενώ παράλληλα λειτουργεί και ως ένα εκπαιδευτικό εργαλείο για τον προγραμματιστή, αφού προσφέρει άμεσα εξηγήσεις και απαντήσεις σχετικά με τις διορθώσεις του κώδικα.

Ακόμη, μέσω των διορθώσεων και των συμπληρώσεων σε πραγματικό χρόνο που προσφέρουν αυτά τα μοντέλα, επιταχύνουν την ανάπτυξη λογισμικού και διασφαλίζουν υψηλή ποιότητα στον παραγόμενο κώδικα.

Επιπλέον, η δυνατότητά τους να υποστηρίζουν πολλές γλώσσες προγραμματισμού τα καθιστά ιδιαίτερα χρήσιμα για πολυγλωσσικά έργα, όπου η κατανόηση μεγάλου όγκου κώδικα σε διαφορετικές γλώσσες προγραμματισμού μπορεί να είναι μία πρόκληση για τους εμπλεκόμενους προγραμματιστές.

Τέλος, η εκπαίδευσή των μεγάλων γλωσσικών μοντέλων σε μεγάλα σύνολα δεδομένων και projects, επιτρέπει βαθιά κατανόηση και εφαρμογή των βέλτιστων τεχνικών της ανάπτυξης λογισμικού, κάνοντάς τα πολύτιμα εργαλεία για προγραμματιστές και μεγάλες ομάδες ανάπτυξης.

3.3.2 Περιορισμοί

Σε συνδυασμό με τα πλεονεκτήματα, η εκτεταμένη χρήση των μεγάλων γλωσσικών μοντέλων κατά την διαδικασία της ανάπτυξης λογισμικού συνεπάγεται και ορισμένους περιορισμούς.

Συγκεκριμένα, ενώ τα μεγάλα γλωσσικά μοντέλα διαπρέπουν σε απλά σενάρια και έργα, σε πολλές περιπτώσεις παρουσιάζουν περιορισμένη ακρίβεια σε πιο εξειδικευμένα ή σύνθετα σενάρια, λόγω της πιθανής ελλιπής εκπαίδευσής τους σε παρόμοια δεδομένα, με αποτέλεσμα την παραγωγή ανεπαρκών αποτελεσμάτων.

Ακόμη, η χρήση κώδικα η οποία έχει παραχθεί από μεγάλα γλωσσικά μοντέλα, αυξάνει την ανάγκη για επαλήθευση και διορθώσεις από τους προγραμματιστές, καθώς το μοντέλο ενδέχεται να μην κατανοεί πλήρως τις απαιτήσεις ή να παράγει κάτι παραπλήσιο από αυτό που ζητήθηκε. Αυτό μπορεί να αυξήσει σημαντικά τον χρόνο διόρθωσης σφαλμάτων, σε σύγκριση με τη συγγραφή κώδικα από τους ίδιους τους προγραμματιστές.

Τέλος, από τον τρόπο κατασκευής τους, τα μεγάλα γλωσσικά μοντέλα εξαρτώνται άμεσα από το σύνολο των δεδομένων όπου έχουν εκπαιδευτεί, με αποτέλεσμα η ποιότητα και το περιεχόμενο των δεδομένων να επηρεάζουν σε μεγάλο βαθμό τα παραγόμενα αποτελέσματα.

3.4 Μελλοντικές προοπτικές και ευκαιρίες ανάπτυξης

Τα μεγάλα γλωσσικά μοντέλα, ήδη ισχυρά εργαλεία στην ανάπτυξη λογισμικού, συνεχώς εξελίσσονται, με νέες και υποσχόμενες εφαρμογές και βελτιώσεις να μπορούν να επεκτείνουν περαιτέρω τις δυνατότητές τους. Με τις βελτιώσεις αυτές, στις οποίες θα γίνει αναφορά στην

συνέχεια, αναμένεται ότι τα μοντέλα αυτά θα προσφέρουν ακόμη πιο προηγμένα εργαλεία και λειτουργικότητες για να κάνουν χρήση οι προγραμματιστές.

3.4.1 Σύμπραξη μεγάλων γλωσσικών μοντέλων

Μία υποσχόμενη προοπτική η οποία θα άξιζε να ερευνηθεί σε μεγαλύτερο βαθμό, όπως αναφέρει και ο Hou και συν.(2023), είναι η ενσωμάτωση πολλαπλών διαφορετικών μεγάλων γλωσσικών μοντέλων σε ένα ενιαίο σύστημα, όπου τα μοντέλα θα συνεργάζονται για την επίλυση σύνθετων εργασιών στην ανάπτυξη λογισμικού. Αυτή η σύμπραξη θα αξιοποιούσε τα διαφορετικά πλεονεκτήματα κάθε μοντέλου, δημιουργώντας ένα εξαιρετικά αποτελεσματικό σύστημα.

3.4.2 Κατανόηση διαφορετικού τύπου εισόδων

Επιπρόσθετα, μια σημαντική ευκαιρία για την ανάπτυξη των μεγάλων γλωσσικών μοντέλων στον τομέα της ανάπτυξης λογισμικού, είναι η βελτίωση της κατανόησης διαφορετικών τύπων εισόδου, όπως εικόνες, σχήματα και φωνή, καθώς και η αυτόματη παραγωγή πολύπλοκων διαγραμμάτων όπως UML. Παρόλο που πολυτροπικά μοντέλα, όπως το GPT-4, έχουν σημειώσει πρόοδο σε αυτούς τους τομείς, δεν έχουν ακόμα φτάσει σε ένα πλήρως υποσχόμενο επίπεδο ώστε να αποτελούν ένα έμπιστο εργαλείο για την υλοποίηση των παραπάνω(Hou και συν., 2023).

3.4.3 Προσαρμογή στις ανάγκες των χρηστών

Επιπλέον, όπως αναφέρει και η OpenAI, μία πολύ σημαντική εξέλιξη αναμένεται να ενσωματωθεί στα ήδη μεγάλα γλωσσικά μοντέλα της, τα οποία θα διαθέτουν πιο προηγμένες ικανότητες μάθησης και προσαρμογής, επιτρέποντάς τους να μαθαίνουν από τις αλληλεπιδράσεις με τους χρήστες και να προσαρμόζονται στις συγκεκριμένες ανάγκες και προτιμήσεις τους. Αυτές οι δυνατότητες θα μπορούσαν να βελτιώσουν την ακρίβεια των προτάσεων κώδικα και να προσφέρουν εξατομικευμένες λύσεις που ανταποκρίνονται καλύτερα στις απαιτήσεις των έργων, αφού το μεγάλο γλωσσικό μοντέλο θα επικεντρώνεται σε συγκεκριμένους τομείς.

3.4.4 Βελτίωση στις ήδη υπάρχουσες εργασίες

Ωστόσο, είναι εξίσου σημαντικό τα μεγάλα γλωσσικά μοντέλα να συνεχίσουν να βελτιώνονται και στις ήδη υπάρχουσες εργασίες που εκτελούν καλά, μέσω της εκπαίδευσής τους σε

περισσότερα και ποιοτικότερα δεδομένα. Αυτό περιλαμβάνει την περαιτέρω βελτίωση της ικανότητάς τους να παράγουν κώδικα με βάση τις απαιτήσεις σε φυσική γλώσσα, την ακριβή πρόταση διορθώσεων και συμπληρώσεων και τη μείωση των σφαλμάτων κατά τη διάρκεια της ανάπτυξης. Η ενίσχυση σε αυτούς τους τομείς θα βελτιώσει τη συνολική αποδοτικότητα των εργαλείων αυτών και θα συμβάλει στην ακόμα καλύτερη ενσωμάτωσή τους και αποδοχή στη διαδικασία ανάπτυξης λογισμικού.

4. Behavior Driven Development (BDD)

Το κεφάλαιο αυτό εστιάζει στη μεθοδολογία του **Behavior Driven Development** (BDD), μια εξέλιξη του Test Driven Development (TDD) που στοχεύει στην προώθηση της συνεργασίας μεταξύ προγραμματιστών, ειδικών ελέγχου ποιότητας και πελατών. Η μεθοδολογία αυτή ενισχύει τη συνεργασία και την επικοινωνία μεταξύ των μελών μιας ομάδας ανάπτυξης λογισμικού, βελτιώνοντας έτσι την ποιότητα και τη σαφήνεια των απαιτήσεων και της λειτουργικότητας του λογισμικού.

4.1 Εισαγωγή στο Behavior Driven Development (BDD) και Test Driven Development (TDD)

Ξεκινώντας το κεφάλαιο, θα γίνει μία συνοπτική αναφορά στην μεθοδολογία TDD και τις αρχές τις, έτσι ώστε στην συνέχεια να γίνει κατανοητή η έννοια του BDD που αποτελεί μία εξέλιξη αυτής.

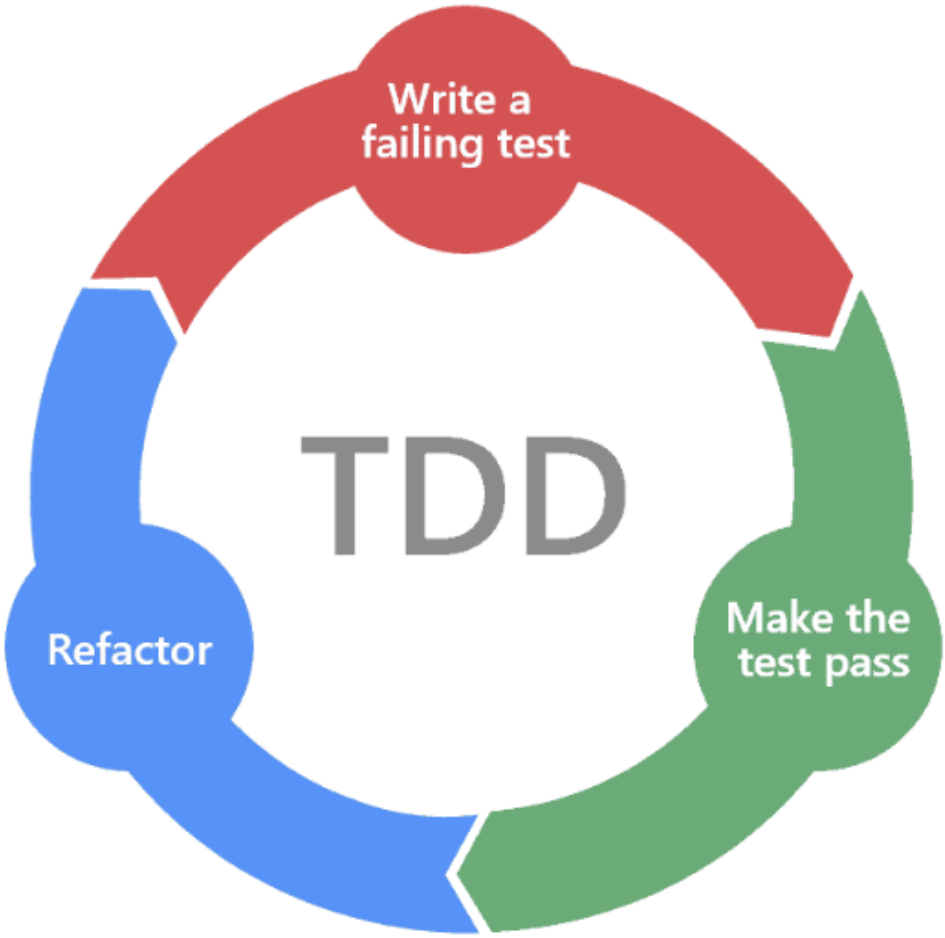
4.1.1 Ορισμός και βασικές αρχές του TDD

Το **Test Driven Development (TDD)** είναι μία μεθοδολογία ανάπτυξης λογισμικού που δίνει έμφαση στην δημιουργία δοκιμών (tests) πριν αρχίσει η διαδικασία της κωδικοποίησης, κάτι το οποίο φαίνεται αντίθετο με τις παραδοσιακές πρακτικές, που τοποθετούν τις δοκιμές σαν το τελευταίο στάδιο στην διαδικασία ανάπτυξης λογισμικού.

Το TDD υποστηρίζει ότι οι δοκιμές αρχικά σχεδιάζονται με την δεδομένη συνθήκη ότι θα αποτύχουν, αφού δεν υπάρχει ακόμη ο απαραίτητος κώδικας για να περάσουν οι έλεγχοι. Στη συνέχεια, αναπτύσσεται ο κώδικας της νέας μονάδας ή λειτουργικότητας με την πιο απλή λύση ώστε απλά να περάσει το τεστ. Μετά την υλοποίηση, οι δοκιμές εκτελούνται ξανά για να επαληθευτεί ότι η υλοποίηση καλύπτει τις απαιτήσεις των δοκιμών και τέλος θα πρέπει να

γίνουν οι απαραίτητες τροποποιήσεις για να βελτιωθεί η ποιότητα του υλοποιημένου ελέγχου της μονάδας (Test-driven development, 2024).

Εικόνα 7. Ο κύκλος ανάπτυξης του TDD



Σημείωση: Ανάκτηση από «Why Test-Driven Development (TDD)», MARNSER,(σύνδεσμος)

Το κλειδί στο TDD είναι ότι η κάλυψη του κώδικα δεν είναι ο τελικός στόχος, αλλά το αποτέλεσμα της σωστής εφαρμογής της μεθοδολογίας TDD. Το TDD προάγει την ποιότητα του κώδικα, καθώς και τη συνεχή βελτίωση μέσα από επαναλαμβανόμενους κύκλους δοκιμών και υλοποίησης.

4.1.2 Ορισμός και βασικές αρχές του BDD

Η μεθοδολογία **Behavior Driven Development** (BDD) βασίζεται κυρίως στην ιδέα του TDD και αποτελεί μία εξέλιξη της, επηρεασμένη παράλληλα και από διάφορες άλλες agile

μεθοδολογίες, η οποία έχει ως βασικό της στόχο την ενθάρρυνση της συνεργασίας μεταξύ όλων των ενδιαφερόμενων μερών στην διαδικασία της ανάπτυξης του λογισμικού ενός έργου. Συγκεκριμένα, όπως αναφέρει και η Fitzgibbons(2021), το BDD προωθεί την τεκμηρίωση και τον σχεδιασμό μίας εφαρμογής με βάσει της συμπεριφοράς που αναμένεται να βιώσει ο χρήστης, ενθαρρύνοντας τους προγραμματιστές να επικεντρώνονται στις σημαντικές συμπεριφορές που πρέπει να έχει το σύστημα. Έτσι, αποφεύγεται η δημιουργία περιττού κώδικα και υπερβολικών χαρακτηριστικών.

Το BDD επιτυγχάνει τον στόχο της ενθάρρυνσης της συνεργασίας μεταξύ όλων των ενδιαφερόμενων μερών της ανάπτυξης ενός έργου, μέσω της χρήσης μίας κοινής γλώσσας, κατανοητής από όλους, η οποία αναφέρεται ως «**Ομοιογενής Γλώσσα**» (Ubiquitous Language). Χρησιμοποιώντας μία κοινή γλώσσα, βελτιώνεται η κατανόηση των απαιτήσεων του έργου και η ποιότητα του παραγόμενου λογισμικού, αφού όλοι οι εμπλεκόμενοι, ανεξαρτήτως της θέσης τους, μπορούν να συμμετέχουν και να έχουν λόγο στο τελικό αποτέλεσμα.

4.2 Η σημασία της συνεργασίας στο BDD

4.2.1 Ομοιογενής Γλώσσα (Ubiquitous Language)

Το κεντρικό σημείο της συνεργασίας μεταξύ όλων των ενδιαφερόμενων μερών είναι η χρήση μίας γλώσσας κατανοητής από όλα τα εμπλεκόμενα δια τμηματικά μέλη των ομάδων. Αυτή η γλώσσα ονομάζεται **Ομοιογενής Γλώσσα** και χρησιμοποιείται σε κάθε επικοινωνία σχετικά με το λογισμικό. Ένα παράδειγμα μίας τέτοιας γλώσσας, που χρησιμοποιείται στην πράξη στο εργαλείο Cucumber και θα γίνει αναλυτική αναφορά παρακάτω, είναι η **Gherkin**. Η Gherkin είναι μία δομημένη γλώσσα με την οποία δημιουργούνται οι έλεγχοι στο BDD και χρησιμοποιεί μία συγκεκριμένη σύνταξη, εύκολη στην ανάγνωση από όλους και γραμμένη σε φυσική γλώσσα (Cucumber, n.d.).

4.2.2 Σενάρια και Ιστορίες Χρηστών στο BDD

Η πρώτη διαδικασία με την οποία αρχίζει κάθε έργο που αναπτύσσεται με την μεθοδολογία BDD, είναι ο ορισμός της συμπεριφοράς του συστήματος από την οπτική των τελικών χρηστών. Αυτό περιλαμβάνει την δημιουργία **ιστοριών χρηστών** και **σεναρίων** που περιγράφουν πώς θα πρέπει να συμπεριφέρεται το σύστημα σε διάφορες καταστάσεις που μπορεί να έρθει αντιμέτωπο κατά την λειτουργία του.

Οι **ιστορίες χρηστών** γράφονται σε φυσική γλώσσα και είναι της μορφής «Ως [ρόλος του χρήστη], Θέλω να [κάποια ενέργεια του χρήστη με το σύστημα], Ώστε να [κάποιο τελικό αποτέλεσμα, στόχος ή πλεονέκτημα]». Ένα παράδειγμα είναι αυτό της **Εικόνας 8**, όπου παρουσιάζεται μία υποθετική ιστορία χρήστη σε ένα σύστημα βιβλιοθήκης, όπου ο χρήστης ή βιβλιοθηκάριος στην συγκεκριμένη περίπτωση, θέλει μέσω του συστήματος να μπορεί να χειρίζεται αποδοτικά την διαδικασία δανεισμού ώστε οι δανειζόμενοι να μπορούν να δανείζονται εύκολα βιβλία από την βιβλιοθήκη. (Cucumber, n.d.)

Εικόνα 8. Παράδειγμα ιστορίας χρήστη

```
User Story: As a librarian,
I want to efficiently manage the loaning process of books to registered borrowers
So that borrowers can easily borrow books from the library.
```

Πολλά **σενάρια** μπορούν να ανήκουν σε ένα κοινό χαρακτηριστικό του συστήματος, το ονομαζόμενο «**feature**». Για παράδειγμα, για το χαρακτηριστικό/feature "Εγγραφή χρήστη", μπορούν να υπάρξουν σενάρια που εξετάζουν τι γίνεται στην περίπτωση της επιτυχούς εγγραφής, στην περίπτωση που ο χρήστης έχει ήδη λογαριασμό και πολλά άλλα.

Τα σενάρια γράφονται με την ομοιογενή γλώσσα **Gherkin**, όπως αναφέρθηκε παραπάνω, η οποία έχει την μορφή «**Given-When-Then**»(συνήθως ακολουθείται αυτή η σειρά) και στην οποία θα γίνει αναφορά παρακάτω.

Τα σενάρια που γράφονται με αυτή τη δομή μπορούν εύκολα να είναι κατανοητά από όλους τους ενδιαφερόμενους και βοηθούν στην εξασφάλιση ότι οι απαιτήσεις των χρηστών ικανοποιούνται πλήρως από το σύστημα (Cucumber, n.d.).

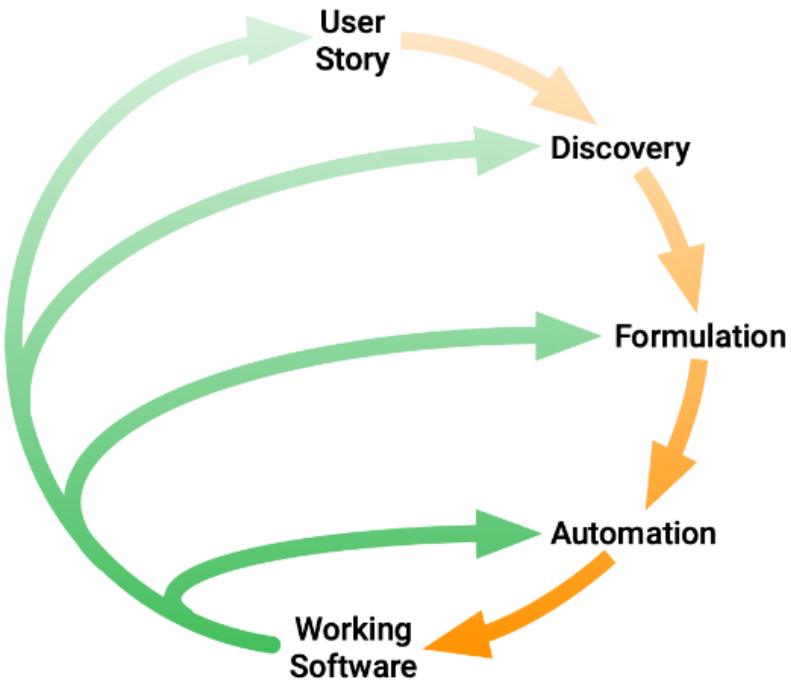
4.3 Οι τρείς πρακτικές του BDD

Όπως αναφέρεται και στο documentation της επίσημης σελίδας του Cucumber, το BDD, όπως και το TDD, ενθαρρύνει την εργασία σε γρήγορες επαναλήψεις(rapid iterations), διασπώντας τα προβλήματα των χρηστών σε μικρά κομμάτια ώστε να μπορούν να επιλυθούν γρήγορα και ορθά.

Ουσιαστικά, η καθημερινότητα ενός έργου που αναπτύσσεται με BDD αποτελείται από τρείς διαδικασίες:

- **Ανάλυση παραδειγμάτων** : Ξεκινά με μια μικρή επερχόμενη αλλαγή, μια ιστορία χρήστη, όπως αναλύθηκε και πριν, και την αναλύει σε πραγματικά παραδείγματα (σενάρια).
- **Αυτοματοποίηση παραδειγμάτων** : Τροποποιεί τα παραδείγματα ώστε να μπορούν να αυτοματοποιηθούν ως σενάρια Gherkin.
- **Υλοποίηση κώδικα** : Υλοποιεί τον κώδικα του συστήματος που περιγράφεται από αυτά τα παραδείγματα.

Εικόνα 9. Οι τρείς πρακτικές του BDD



Σημείωση: Ανάκτηση από «Behavior-Driven Development», Cucumber (σύνδεσμος)

4.4 Τεχνικές Ανάλυσης Συνεργασίας

Ένα κομβικό σημείο στην εύρυθμη λειτουργία της μεθοδολογίας BDD, είναι μία συνάντηση η οποία ονομάζεται «Three Amigos Meeting», όπου όπως θα αναλυθεί και παρακάτω, αποτελεί μία «συζήτηση» μεταξύ των διαφορετικών ενδιαφερομένων για την κατανόηση και δημιουργία των απαιτήσεων ή σεναρίων.

4.4.1 Η συνάντηση των Τριών Φίλων (Three Amigos Meeting)

Η συνάντηση των τριών φίλων, ή όπως είναι γνωστή «**Three Amigos Meeting**», έχει ως βασικό σκοπό της την μετατροπή των ιστοριών χρηστών σε κατανοητά και ολοκληρωμένα από όλους σενάρια Gherkin. Περιέχει τουλάχιστον τρία διαφορετικά μέλη :

- **Ιδιοκτήτης του προϊόντος (Product Owner)** - Εστιάζει στο εύρος της εφαρμογής και μετατρέπει τις ιστορίες χρηστών σε features , αποφασίζοντας τι είναι εντός και εκτός του εύρους.
- **Δοκιμαστής (Tester)** - Δημιουργεί σενάρια και ακραίες περιπτώσεις, εξετάζοντας ποιες ιστορίες χρηστών δεν έχουν καλυφθεί.
- **Προγραμματιστής (Developer)** - Προσθέτει βήματα στα σενάρια και εξετάζει τις λεπτομέρειες της εφαρμογής, όπως την εκτέλεση και τα πιθανά εμπόδια κατά την δημιουργία του κώδικα.

Οι παραπάνω συζητήσεις δημιουργούν πολύ καλούς ελέγχους, διότι ο το κάθε άτομο βλέπει το σύστημα από διαφορετική οπτική γωνία. Για τον λόγο αυτό, κρίνεται αναγκαίο όλα τα παραπάνω άτομα να παίρνουν μέρος στην συνάντηση, με σκοπό την ανακάλυψη νέων παραδειγμάτων χρήσης. (Cucumber, n.d.)

Εικόνα 10. The Three Amigos Meeting

Σημείωση: Ανακτήθηκε από «The 3 Amigos Meeting», AWH, 2020, Medium. (σύνδεσμος)

4.5 Αυτοματισμός Δοκιμών με το Cucumber

4.5.1 Εισαγωγή στο Cucumber

Το **Cucumber** είναι ένα εργαλείο λογισμικού το οποίο υποστηρίζει την μεθοδολογία του Behavior Driven Development βασιζόμενο στην χρήση της γλώσσας **Gherkin**, για την οποία έγινε μία μικρή εισαγωγή παραπάνω. Η Gherkin επιτρέπει την περιγραφή των απαιτήσεων του συστήματος και την αυτοματοποίηση των ελέγχων. Συγκεκριμένα, το Cucumber έχει έναν βοηθητικό ρόλο ως προς τον αυτοματισμό των ελέγχων ενός συστήματος, προσφέροντας μία κοινή γλώσσα που είναι κατανοητή από όλους τους ενδιαφερόμενους.

4.5.2 Η Σύνταξη Gherkin

Η Gherkin είναι μία δομημένη γλώσσα που χρησιμοποιείται από το εργαλείο Cucumber για τη δημιουργία ελέγχων στο BDD. Η σύνταξή της είναι απλή και κατανοητή, επιτρέποντας σε τεχνικούς και μη τεχνικούς χρήστες να μπορούν να γράφουν σενάρια.

Η γλώσσα Gherkin χρησιμοποιεί την μορφή «Given-Then-When» και με αυτό τον τρόπο επιτρέπει την εύκολη γραφή σεναρίων κατανοητών από όλα τα μέλη.

Οι λέξεις αυτές κλειδιά αποτελούν τον βασικό τρόπο γραφής των σεναρίων ενός feature και κάθε μία από αυτές έχει μία διαφορετική σημασία, όπως αναφέρεται και από το documentation της επίσημης σελίδας του Cucumber :

- Η λέξη κλειδί **Given**, ορίζει το γενικό πλαίσιο του σεναρίου και βάζει το σύστημα σε μία συγκεκριμένη κατάσταση ώστε να ξεδιπλωθεί το σενάριο στα επόμενα βήματα. Αυτό μπορεί να περιλαμβάνει την προετοιμασία του συστήματος, τη δημιουργία δεδομένων ή την εξασφάλιση ότι το σύστημα βρίσκεται σε μια συγκεκριμένη αρχική κατάσταση.
- Η λέξη κλειδί **When**, είναι μια ενέργεια, η οποία μεταβάλλει το σύστημα από την κατάσταση «ηρεμίας» του και το αναγκάζει να προβεί σε κάποιο αποτέλεσμα. Αυτή η ενέργεια είναι συνήθως η κεντρική πράξη του σεναρίου και αποτελεί το γεγονός που θα προκαλέσει την αντίδραση του συστήματος.
- Η λέξη κλειδί **Then**, είναι το αποτέλεσμα, δηλαδή η ενέργεια που αναμένουμε να κάνει το σύστημα όταν γίνεται η ενέργεια στο βήμα When. Αυτό το τμήμα του σεναρίου περιγράφει την αναμενόμενη συμπεριφορά του συστήματος και επιτρέπει την επαλήθευση ότι το σύστημα λειτουργεί όπως πρέπει.

Στην **Εικόνα 11** παρακάτω, παρουσιάζεται ένα παράδειγμα ενός σεναρίου γραμμένου σε Gherkin στο Cucumber, το οποίο αφορά την ενημέρωση των στοιχείων ενός δανειζόμενου σε ένα υποθετικό σύστημα βιβλιοθήκης και το οποίο ανήκει σε ένα feature το οποίο αφορά την «Διαχείριση Δανειζόμενων». Όπως φαίνεται, κάθε σενάριο περιέχει επίσης έναν σύντομο τίτλο ο οποίος περιγράφει την περίπτωση χρήσης που πραγματεύεται το σενάριο που ακολουθεί.

Εικόνα 11. Παράδειγμα σεναρίου με την μορφή Given-When-Then

```
Scenario: Updating the borrower's details when he is registered
This scenario describes the process of updating the details of a borrower who has already registered before
  Given George Red is registered as a borrower
  When George Red updates his borrowing details
  Then the system saves the changes
```

Στην **Εικόνα 12** παρουσιάζεται ένα ακόμη παράδειγμα ενός υποθετικού σεναρίου στο ίδιο σύστημα με την Εικόνα 11, το οποίο αφορά την περίπτωση της επιτυχούς εγγραφής ενός δανειζόμενου σε ένα σύστημα βιβλιοθήκης.

Εικόνα 12. Παράδειγμα σεναρίου με την μορφή Given-When-Then

```
Scenario: Registering a new borrower
This scenario describes the process of registering a new borrower in the library system
  Given George Red is not registered as a borrower
  When George Red gets registered in the system with a unique borrower number and his details
  Then the system successfully stores the borrower's details
```

4.5.3 Δημιουργία Αρχείων Χαρακτηριστικών (Feature Files)

Τα **αρχεία χαρακτηριστικών** (feature files), όπως αναφέρθηκε και παραπάνω, είναι αρχεία που περιέχουν τα σενάρια Gherkin που περιγράφουν τις διάφορες περιπτώσεις χρήσης ενός συγκεκριμένου feature του συστήματος. Η δομή των feature files φαίνεται στην **Εικόνα 13** και αποτελείται από την λέξη κλειδί «feature» που αρχικοποιεί το αρχείο. Στην συνέχεια, ακολουθεί το όνομα του feature, το οποίο θα είναι και το όνομα του αρχείου και μία μικρή περιγραφή του feature και των σεναρίων που θα ακολουθήσουν. Ακόμη, μία καλή πρακτική αποτελεί η προσθήκη μίας ιστορία χρήστη, για να γίνει καλύτερα κατανοητό το σύνολο των σεναρίων που θα ακολουθήσουν.

Εικόνα 13. Παράδειγμα δομής ενός feature file

```
Feature: Borrower handling by the system
  The system can register a new person, modify their credentials or delete their account

  User Story: As a librarian,
  I want to efficiently manage the loaning process of books to registered borrowers
  So that borrowers can easily borrow books from the library.

  Scenario: Registering a new borrower
  This scenario describes the process of registering a new borrower in the library system
    Given George Red is not registered as a borrower
    When George Red gets registered in the system with a unique borrower number and his details
    Then the system successfully stores the borrower's details

  Scenario: Borrower trying to register has registered before
  This scenario describes what happens when the system tries to register a new borrower who has already registered before
    Given George Red is registered as a borrower
    When the system attempts to register George Red with a unique borrower number and his details
    Then the system informs that the user already exists
```

Μια καλή πρακτική είναι τα feature files να περιέχουν το πολύ δέκα σενάρια και να εστιάζουν σε ένα συγκεκριμένο χαρακτηριστικό.

4.5.4 Ανάπτυξη των Step Definitions

Τα **Step Definitions** είναι τμήματα κώδικα που συνδέουν τα σενάρια γραμμένα σε Gherkin με την πραγματική υλοποίηση κώδικα ώστε να περάσουν οι έλεγχοι και να δοκιμαστεί το σύστημα. Τα Step Definitions μπορούν να γραφτούν σε διάφορες γλώσσες προγραμματισμού όπως Java, Ruby κ.ά., και προσδιορίζουν την συμπεριφορά κάθε βήματος ενός σεναρίου Gherkin.

Το Cucumber εκτελεί τα σενάρια που βρίσκονται στα feature files τρέχοντας τον αντίστοιχο κώδικα των Step Definitions, διασφαλίζοντας ότι τα βήματα εκτελούνται με σωστή σειρά, προσομοιάζοντας την συμπεριφορά που περιγράφεται στα σενάρια Gherkin.

Η δομή των Step Definitions, όπως φαίνεται και στην **Εικόνα 14**, περιλαμβάνει βοηθητικές δομές δεδομένων και αντικείμενα που χρησιμοποιούνται στον κώδικα των Step Definitions. Για να συνδέσουμε τα βήματα ενός σεναρίου με ένα Step Definition, χρησιμοποιούμε τη μορφή @Given, @When, ή @Then ακολουθούμενη από το κείμενο του βήματος στο σενάριο Gherkin.

Εικόνα 14. Παράδειγμα δομής ενός αρχείου Step Definitions

```
public class StepDefinitions {
    2 usages
    private Person sean;
    3 usages
    private Person lucy;
    2 usages
    private String messageFromSean;

    @Given("Lucy is {int} metres from Sean")
    public void lucy_is_located_metres_from_Sean(Integer distance){
        lucy = new Person();
        sean = new Person();
        lucy.moveTo(distance);
    }
    @When("Sean shouts {string}")
    public void sean_shouts(String message) {
        sean.shout(message);
        messageFromSean = message;
    }
    @Then("Lucy should hear Sean's message")
    public void lucy_should_hear_sean_s_message() {
        assertEquals(asList(messageFromSean), lucy.getMessagesHeard());
    }
}
```

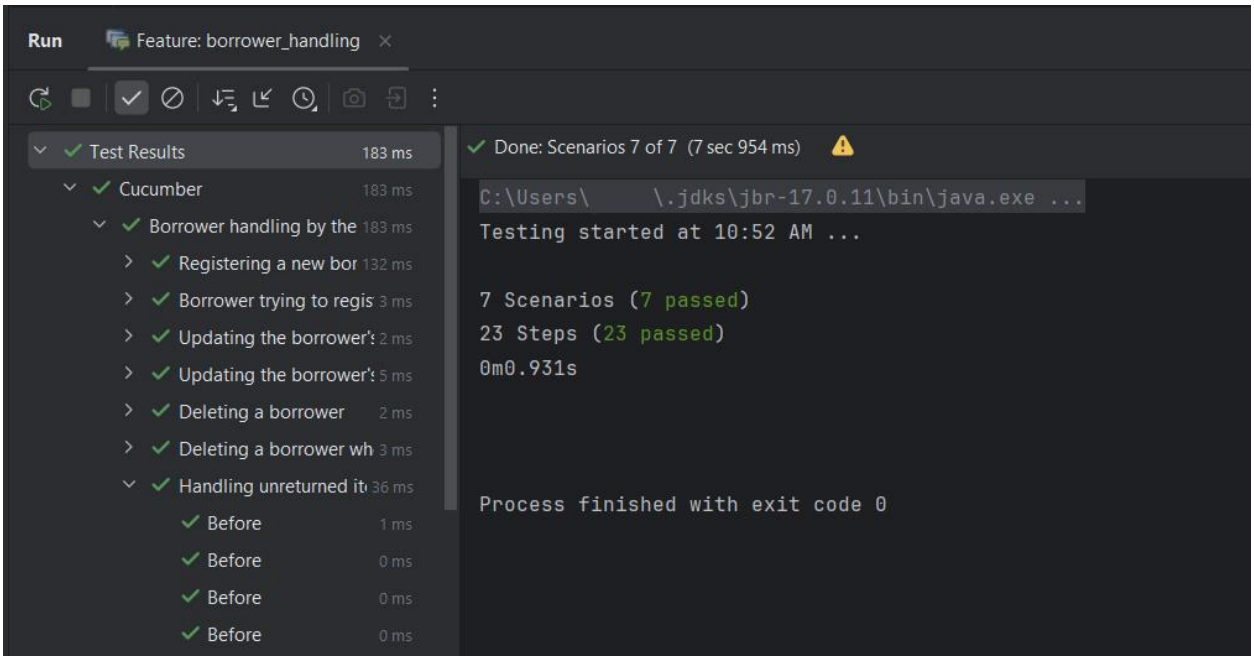
Στην περίπτωση της **Εικόνας 14,** το πρώτο βήμα του Step Definition αρχικοποιεί το περιβάλλον και τα αντικείμενα που θα χρησιμοποιηθούν στα παρακάτω βήματα του σεναρίου, σε ένα υποθετικό σύστημα μίας εφαρμογής που παραδίδει μηνύματα σε άτομα μίας συγκεκριμένης απόστασης.

Τα Step Definitions πρέπει να είναι μοναδικά ανεξαρτήτως feature file ώστε κάθε βήμα ενός σεναρίου να μπορεί να αντιστοιχιστεί με ένα μόνο Step Definition, αποφεύγοντας την σύγχυση και διασφαλίζοντας την σωστή εκτέλεση των ελέγχων.

4.6 Αναφορές του Cucumber

Μετά την εκτέλεση των βημάτων των σεναρίων των Step Definitions, το Cucumber παράγει **λεπτομερείς αναφορές** και σχόλια για την εκτέλεση που ακολούθησε. Αυτές οι αναφορές περιλαμβάνουν πληροφορίες για τα βήματα που πέρασαν επιτυχώς, τα βήματα που απέτυχαν και εκείνα που παραλήφθηκαν λόγω κάποιου σφάλματος. Οι αναφορές αυτές αποτελούν σημαντικό εργαλείο για προγραμματιστές και τους ειδικούς ελέγχου, καθώς τους βοηθούν να κατανοήσουν την κατάσταση του συστήματος και να εντοπίσουν τυχόν προβλήματα. (Cucumber, n.d.)

Εικόνα 15. Παράδειγμα αναφοράς με το Cucumber



4.7 Ζωντανή Τεκμηρίωση

Όταν χρησιμοποιείται το BDD ως μεθοδολογία ανάπτυξης ενός συστήματος, γράφονται παραδείγματα ή σενάρια τα οποία καθοδηγούν την ανάπτυξη. Αυτά τα παραδείγματα χρησιμεύουν επίσης ως κριτήρια αποδοχής (acceptance criteria) και είναι γραμμένα σε μια κοινή γλώσσα, ώστε να είναι κατανοητά από όλους τους εμπλεκόμενους. Επομένως, τα παραδείγματα ή σενάρια αυτά λειτουργούν και ως μία «**ζωντανή» τεκμηρίωση** του συστήματος. Σε αντίθεση με τα παραδοσιακά έγγραφα κριτηρίων αποδοχής, αυτή η τεκμηρίωση ενημερώνεται συνεχώς με κάθε νέα αλλαγή ή προσθήκη λειτουργικότητας στο σύστημα.

Η σημασία της ζωντανής τεκμηρίωσης είναι ιδιαίτερα σημαντική σε ένα περιβάλλον ανάπτυξης λογισμικού, καθώς επιτρέπει σε όλα τα μέλη της ομάδας να κατανοούν τις απαιτήσεις του συστήματος. Αυτό είναι ιδιαίτερα χρήσιμο για νέα μέλη της ομάδας, που δεν γνωρίζουν από πριν το σύστημα που αναπτύσσεται. Η ζωντανή τεκμηρίωση διασφαλίζει ότι η γνώση του συστήματος διατηρείται και διαμοιράζεται αποτελεσματικά σε όλη την ομάδα (Cucumber, n.d.).

4.8 Καλές πρακτικές για γραφή με Cucumber

Η χρήση του Cucumber για τον αυτοματισμό των ελέγχων και την υιοθέτηση της μεθοδολογίας BDD μπορεί να συμβάλλει σημαντικά στην βελτίωση της ποιότητας του παραγόμενου λογισμικού. Ωστόσο, η επίτευξη αυτού του στόχου εξαρτάται σε μεγάλο βαθμό από τον τρόπο που θα γίνει η γραφή των σεναρίων και η οργάνωσή τους σε feature files. Μερικές καλές πρακτικές για την συγγραφή σεναρίων με Cucumber, όπως αναφέρει και ο Pathak (2022), είναι οι εξής:

- **Μικρά σενάρια** : Τα σενάρια που γράφονται για ένα feature και τα οποία περιγράφουν μία περίπτωση χρήσης του συστήματος, θα πρέπει να είναι μικρά και να εστιάζουν σε μία μόνο λειτουργία του συστήματος.
- **Χρήση παραδειγμάτων από τον πραγματικό κόσμο** : Τα σενάρια θα πρέπει να βασίζονται σε πραγματικά παραδείγματα ώστε να διασφαλίζουν ότι είναι ρεαλιστικά και ότι το σύστημα θα λειτουργήσει σωστά σε πραγματικές συνθήκες
- **Επαναχρησιμοποίηση βημάτων** : Όταν είναι δυνατόν, θα πρέπει να γίνεται επαναχρησιμοποίηση των βημάτων των σεναρίων ώστε να μειωθεί η επανάληψη και να διευκολυνθεί η συντήρηση των σεναρίων.

- **Οργάνωση των feature files** : Τα feature files θα πρέπει να οργανώνονται με τρόπο που να διευκολύνει την εύρεση και κατανόηση των σεναρίων. Κάθε αρχείο χαρακτηριστικών θα πρέπει να περιέχει σενάρια που σχετίζονται με ένα συγκεκριμένο χαρακτηριστικό ή λειτουργία του συστήματος, χρησιμοποιώντας κατανοητούς τίτλους και περιγραφές.

5. ΒΙΒΛΙΟΓΡΑΦΙΑ

Cucumber. (χ.χ.). *Gherkin Reference*. Ανάκτηση από Cucumber documentation: https://cucumber.io/docs/gherkin/reference/

Fitzgibbons, L. (2021, Οκτώβριος). *behavior-driven development (BDD).* Ανάκτηση από TechTarget: https://www.techtarget.com/searchsoftwarequality/definition/Behavior-driven-development-BDD

Lee, T. B. (2023, Ιούλιος 31). Ανάκτηση από arstechnica.com: https://arstechnica.com/science/2023/07/a-jargon-free-explanation-of-how-ai-large-language-models-work/6/

Lutkevich, B. (2024, Ιούνιος 21). *TechTarget.* Ανάκτηση από https://www.techtarget.com/whatis/feature/12-of-the-best-large-language-models

Pathak, K. (2022, Σεπτέμβριος 12). *Cucumber Best Practices to follow for efficient BDD Testing*. Ανάκτηση από Medium: https://kailash-pathak.medium.com/cucumber-best-practices-to-follow-for-efficient-bdd-testing-b3eb1c7e9757

Pi, W. (2024, Μάϊος 7). *Research Graph.* Ανάκτηση από Medium: https://medium.com/@researchgraph/brief-introduction-to-the-history-of-large-language-models-llms-3c2efa517112

Sanderson, G. (2017, Οκτώβριος 16). *3blue1brown.* Ανάκτηση από 3blue1brown: https://www.3blue1brown.com/lessons/gradient-descent#another-way-to-think-about-the-gradient

Sumrak, J. (2024, Μάρτιος 11). *7 LLM use cases and applications in 2024.* Ανάκτηση από AssemblyAI: https://www.assemblyai.com/blog/llm-use-cases/

Ανδρουτσόπουλος, Ί. (2024, Φεβρουάριος). Τεχνητή Νοημοσύνη και Μεγάλα Γλωσσικά Μοντέλα. *ΟΠΑ News Εφημερίδα Οικονομικού Πανεπιστημίου Αθηνών Τεύχος 51*, σσ. 8-9.