Ατομική Διπλωματική Εργασία

ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΑΥΤΟΜΑΤΗΣ ΔΗΜΙΟΥΡΓΙΑΣ ΦΥΛΛΩΝ ΕΡΓΑΣΙΑΣ ΓΙΑ ΜΑΘΗΤΕΣ ΔΗΜΟΤΙΚΗΣ ΕΚΠΑΙΔΕΥΣΗΣ

Δέσποινα Χρίστου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη συστήματος αυτόματης δημιουργίας φύλλων εργασίας για μαθητές Δημοτικής Εκπαίδευσης

Δέσποινα Χρίστου

Επιβλέπων Καθηγητής
Μάριος Δικαιάκος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2024

Ευχαριστίες

Θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες στον επιβλέποντα καθηγητή μου, Dr. Μάριο Δικαιάκο, για την υπομονή και την καθοδήγηση, που μου παρείχε κατά τη διάρκεια της εργασίας, αλλά και που με ενθάρρυνε να εξερευνήσω περαιτέρω τις δικές μου επιθυμίες και με βοήθησε να προσανατολιστώ προς τον στόχο μου.

Επιπλέον, είμαι ευγνώμων στον μέντορά μου, Δημήτρη Πασχαλίδη, για τις πολύτιμες συμβουλές, την τεχνική αλλά και την ανεκτίμητη ψυχολογική υποστήριξη και τη διαρκή διαθεσιμότητά του. Η αφοσίωση και η ενθάρρυνσή του ήταν απαραίτητα για την ολοκλήρωση αυτού του δύσκολου αλλά γεμάτου γνώσεις ταξιδιού.

Ένα τεράστιο ευχαριστώ επίσης, στους γονείς και τα αδέρφια μου, οι οποίοι με υποστήριξαν ανελλιπώς σε κάθε βήμα. Η αγάπη, η κατανόηση και η υπομονή τους ήταν τα σταθερά μου σημεία αναφοράς σε όλη τη διάρκεια των σπουδών μου.

Τέλος, θέλω να εκφράσω τη βαθιά μου ευγνωμοσύνη προς τους φίλους μου, οι οποίοι ήταν δίπλα μου καθ' όλη τη διάρκεια των σπουδών μου. Η ατελείωτη υπομονή, η θετική ενέργεια και οι ανυπολόγιστες ώρες υποστήριξης που μου παρείχαν, με βοήθησαν να διατηρήσω την ισορροπία και να επικεντρωθώ στους στόχους μου.

Περίληψη

Η παρούσα ατομική διπλωματική εργασία αποτελεί μια μελέτη για τον τρόπο λειτουργίας των γλωσσικών συστημάτων τεχνητής νοημοσύνης (Large Language Models), κυρίως το μοντέλο της OpenAI – ChatGPT, αλλά και τους τρόπους με τους οποίους μπορούν να χρησιμοποιηθούν τέτοιου είδους μοντέλα για εκπαιδευτικούς σκοπούς. Η εργασία αυτή παρουσιάζει την ανάπτυξη ενός συστήματος, το οποίο έχει ως σκοπό να βοηθήσει εκπαιδευτικούς Δημοτικής Εκπαίδευσης να φτιάχνουν ασκήσεις εμπέδωσης ή και αξιολόγησης για τους μαθητές τους με ένα πιο αυτοματοποιημένο και γρήγορο τρόπο σε συνδυασμό με τις μεγάλες δυνατότητες των Γλωσσικών Μοντέλων στην εύρεση και διατύπωση πρωτότυπων ιδεών. Το σύστημα ανατροφοδοτήθηκε με αρκετά δεδομένα από βιβλία του Δημοτικού που βρίσκονται στο διαδίκτυο, και έχει ως στόχο την εξαγωγή των απαραίτητων πληροφοριών, βασισμένες στις επιλογές του χρήστη ο οποίος θα μπορεί να επιλέγει την τάξη, το μάθημα, το κεφάλαιο, ή να δίνει κάποιες λέξεις κλειδιά, οι οποίες θα χρησιμοποιούνται από το μοντέλο για την δημιουργία των σχετικών ασκήσεων.

Η αξιολόγηση του μοντέλου από διάφορους εκπαιδευτικούς παρουσιάζεται επίσης στο τέλος του δοκιμίου.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο και Σκοπός Εργασίας

Το κύριο κίνητρό μου για την ανάπτυξη ενός τέτοιου συστήματος είναι το γεγονός ότι έχω ως στόχο να ασχοληθώ με τον εκπαιδευτικό τομέα στο μέλλον, και βλέποντας την ραγδαία πλέον ανάπτυξη της τεχνητής νοημοσύνης, αλλά κυρίως των μοντέλων επεξεργασίας και παραγωγής φυσικής γλώσσας, ήθελα να βρω ένα τρόπο που θα μπορούσαν οι εκπαιδευτικοί της Κύπρου να έρθουν περισσότερο σε επαφή με τέτοια συστήματα που μπορούν να προσφέρουν διευκόλυνση και εξοικονόμηση χρόνου, αλλά και δημιουργικότητα και πρωτοτυπία.

Σκοπός λοιπόν, είναι να αναδειχτούν τα οφέλη της τεχνολογίας και συγκεκριμένα, της τεχνητής νοημοσύνης στον τομέα της μάθησης, ώστε να γίνεται χρήση της, με τρόπο, τέτοιο ώστε να παραμείνει ωφέλιμη και βοηθητική για τους ανθρώπους, και εννοείται να μην φτάσει στο σημείο που θα επηρεάζει αρνητικά τους μαθητές αλλά και τους εκπαιδευτικούς, κατά την διαδικασία της μετάδοσης γνώσεων, της εξάσκησης και της αξιολόγησης της εμπέδωσης των γνώσεων.

1.2 Οργάνωση Δοκιμίου

Η εργασία αποτελείται από 6 κεφάλαια:

Κεφάλαιο 1*:* Εισαγωγή, παρουσίαση της δομής της εργασίας και μερικοί ορισμοί και επεξηγήσεις.

Κεφάλαιο 2**:** Στο κεφάλαιο αυτό, θα δούμε τη μελέτη για τη διαδικασία της ανάπτυξης ενός γλωσσικού μοντέλου (Large Language Model), και η σκεπτική που υπάρχει πίσω από αυτά και ο τρόπος λειτουργίας και χρήσης τους. Η μελέτη έγινε περισσότερο βασισμένη στο μοντέλο ChatGPT.

Κεφάλαιο 3**:** Σε αυτό το κεφάλαιο, θα παρουσιαστεί η γενική ιδέα για την ανάπτυξη ενός τέτοιου συστήματος και το υλοποιημένο σύστημα που αναπτύχθηκε, οι δυνατότητες και ο τρόπος λειτουργίας και χρήσης του.

Κεφάλαιο 4**:** Στο τέταρτο κεφάλαιο, θα επεξηγηθεί η όλη διαδικασία που ακολουθήθηκε για την ανάπτυξη του πιο πάνω συστήματος, τα μέσα και τα εργαλεία που χρησιμοποιήθηκαν.

Κεφάλαιο 5**:** Στο κεφάλαιο 5, θα δούμε την αξιολόγηση του ολοκληρωμένου συστήματος, σχόλια από τους εκπαιδευτικούς που το χρησιμοποίησαν, εισηγήσεις και ιδέες για βελτίωση ή προσθήκη χαρακτηριστικών.

Κεφάλαιο 6**:** Στο τελευταίο κεφάλαιο θα αναφερθούν γενικά συμπεράσματα που έχουν εξαχθεί κατά την μελέτη και την ανάπτυξη του μοντέλου και θα συζητηθεί η πιθανή επεκτασιμότητα του συστήματος.

1.3 Ορισμοί και Συντομογραφίες

ΣΥΝΤΟΜΟΓΡΑΦΙΑ	ΕΠΕΞΗΓΗΣΗ
AI (ARTIFICIAL INTELLIGENCE)	Η τεχνητή νοημοσύνη, αναφέρεται στην ικανότητα μιας μηχανής να αναπαράγει τις γνωστικές λειτουργίες ενός ανθρώπου, όπως είναι η μάθηση, ο σχεδιασμός και η δημιουργικότητα.
API (APPLICATION PROGRAMMING INTERFACE)	Διεπαφή Προγραμματισμού Εφαρμογών: ένα σύνολο κανόνων και προτύπων που επιτρέπουν σε δύο εφαρμογές να επικοινωνούν μεταξύ τους. (αιτήσεις για χρήση από άλλη εφαρμογή)
LLM (LARGE LANGUAGE MODEL)	Μοντέλα μηχανικής μάθησης που εκπαιδεύονται σε μεγάλα σύνολα δεδομένων γλωσσικής πληροφορίας. Αυτά τα μοντέλα έχουν τη δυνατότητα να κατανοήσουν και να παράγουν φυσική γλώσσα με εξαιρετική ακρίβεια και ποικιλία.
PROMPT	Ένα κείμενο που παρέχεται σε ένα μοντέλο μηχανικής μάθησης για να προκαλέσει μια απάντηση ή μια πρόβλεψη ή λειτουργία.
UI (USER INTERFACE)	Διεπαφή μέσω της οποίας οι χρήστες αλληλεπιδρούν με ένα λογισμικό ή μια εφαρμογή. Μπορεί να περιλαμβάνει γραφικά στοιχεία, κουμπιά, φόρμες και άλλα στοιχεία που επιτρέπουν στον χρήστη να επικοινωνήσει με το σύστημα.
DB (DATABASE)	Ένα σύνολο δομημένων δεδομένων που οργανώνονται και αποθηκεύονται σε έναν υπολογιστή ή σε ένα σύστημα υπολογιστών, για να χρησιμοποιηθούν από μια εφαρμογή.
EMBEDDINGS	Τεχνική με την οποία μετατρέπουμε δεδομένα, όπως λέξεις ή προτάσεις, σε διανυσματικές αναπαραστάσεις.
VECTOR DB	Μια βάση δεδομένων που αποθηκεύει διανυσματικές αναπαραστάσεις για αντικείμενα ή δεδομένα.

Πίνακας 1.1: Ορισμοί και Επεξηγήσεις

Κεφάλαιο 2

Μεγάλα Γλωσσικά Μοντέλα - ChatGPT

2.1 Τι είναι τα LLM και η θέση τους στην τεχνητή νοημοσύνη

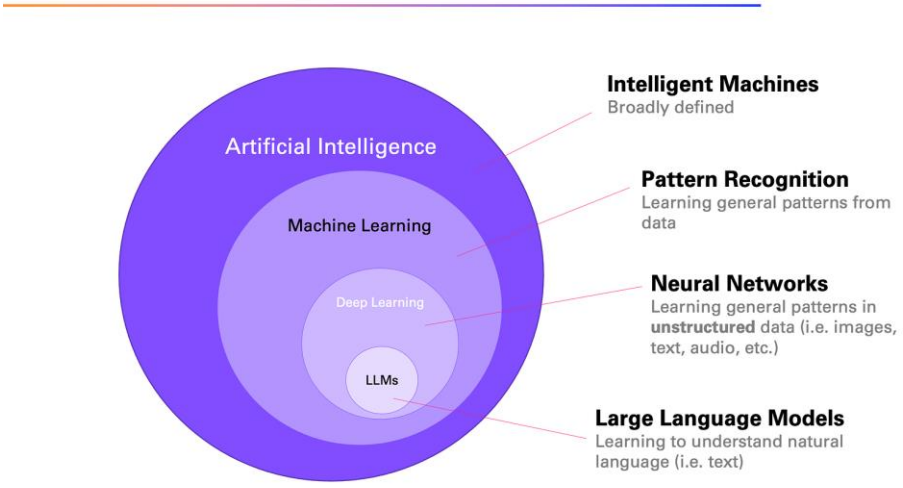
Μιλώντας για Large Language Models, αναφερόμαστε σε μοντέλα deep learning, τα οποία έχουν εκπαιδευτεί σε τεράστιο όγκο κειμένων, σε απλές εργασίες όπως το να προβλέπουν την επόμενη λέξη σε μια πρόταση. Έχουν πλέον ως κύρια ικανότητα να καταλαβαίνουν τη σύνταξη, το νόημα και γενικά τη σημασιολογία της φυσικής γλώσσας και να την παράγουν. Έχουν επίσης μια γενική γνώση για τη λειτουργία του κόσμου και διάφορων γεγονότων. Υπάρχουν περιπτώσεις όμως, όπου τα LLMs μπορούν να ισχυρίζονται με αυτοπεποίθηση πράγματα τα οποία δεν βασίζονται κάπου και ίσως αποτελούν λανθασμένες ή άσχετες με το περιεχόμενο πληροφορίες. Το φαινόμενο αυτό στον τομέα του AI έχει ονομαστεί “ψευδαίσθηση” (hallucinations).

Η λέξη “large” στο όνομά τους, βασίζεται στον μεγάλο αριθμό παραμέτρων (νευρώνων) που τα αποτελούν και συνήθως μιλάμε για αριθμό μεγαλύτερο του ενός δισεκατομμυρίου. Για να κατανοήσουμε λίγο καλύτερα τη θέση που έχουν τα LLM στο AI θα αναλύσουμε τα επίπεδα που αποτελείται η τεχνητή νοημοσύνη: [19]

- Η Μηχανική Μάθηση αποτελεί ένα ευρύτερο πεδίο της τεχνητής νοημοσύνης που επικεντρώνεται στον σχεδιασμό και την ανάπτυξη αλγορίθμων που μπορούν να ανακαλύπτουν μοτίβα στα δεδομένα τα οποία περιγράφουν τη

σχέση μεταξύ μιας εισόδου και ενός αποτελέσματος, βελτιώνοντας την ακρίβεια τους με το πέρασμα του χρόνου.

- Ένα υποσύνολο της μηχανικής μάθησης είναι η Βαθιά Μάθηση (Deep Learning). Τα συστήματα αυτά αποτελούνται από πολλά layers και έχουν την ικανότητα να δουλεύουν με πιο περίπλοκα, unstructured data. Δηλαδή δεδομένα τα οποία αν πάρουμε τα στοιχεία τους απομονωμένα δεν θα έχουν κανένα νόημα, αλλά μαζί αποτελούν ένα αντικείμενο, όπως για παράδειγμα μια εικόνα, ένα κείμενο ή ένα video. [12]
- Φτάνουμε στα LLMs, τα οποία είναι υποσύνολο του Deep Learning, τα οποία εξειδικεύονται στα κείμενα. Συγκεκριμένα, είναι Νευρωνικά Δίκτυα τα οποία εκπαιδεύονται σε τεράστιο αριθμό training data (κείμενα), ώστε να προβλέπουν την επόμενη λέξη σε μια δοσμένη ακολουθία λέξεων, με βάση τη σύνταξη αλλά και τη σημασία της πρότασης. Η διαδικασία αυτή επαναλαμβάνεται, προβλέποντας μια καινούρια λέξη κάθε φορά, φτάνοντας έτσι στην παραγωγή ολόκληρων κειμένων.



Σχήμα 2.1: Τα επίπεδα της Τεχνητής Νοημοσύνης

Πέρα από την απλή γραφή, και την κατανόηση, τα γλωσσικά μοντέλα , παρέχουν μια τεράστια ποικιλία από άλλες ικανότητες, οι οποίες καθιστούν πολύτιμα εργαλεία στην καθημερινότητά μας, σε διάφορους τομείς, αφού ξεπερνούν ακόμα και το ανθρώπινο επίπεδο, σε σχέση με την φαντασία, τη δημιουργικότητα, την ταχύτητα, την μνήμη… Εκτός, όπως ανέφερα και πιο πάνω, από το ότι μπορούν να καταλαβαίνουν και να παράγουν τη φυσική γλώσσα, σε επίπεδο τέτοιο ώστε να μπορούν να συμμετέχουν σε συνομιλίες, να απαντούν τις ερωτήσεις μας, αλλά να αποκτούν και το δικό τους στυλ γραφής και δική τους προσωπικότητα και ύφος. Κάποιες άλλες από τις λειτουργίες τους είναι η επεξεργασία και ανάλυση κειμένων, η εξαγωγή πληροφοριών και η περιλήψεις, όπως επίσης και η δημιουργία περιεχομένου. Πιο συγκεκριμένα, μπορούν να γράψουν άρθρα, μελέτες, ποιήματα, λογοτεχνικά κείμενα, ανέκδοτα, να σκέφτονται καινούριες ιδέες για οποιοδήποτε θέμα, οδηγίες, συμβουλές κλπ. Ακόμα, μπορούν να λειτουργήσουν ως μεταφραστές και διερμηνείς, αλλά και να αναπτύξουν κώδικες σε διάφορες γλώσσες προγραμματισμού ή να βοηθήσουν στην αποσφαλμάτωση προγραμμάτων.

2.1.1 Transformers

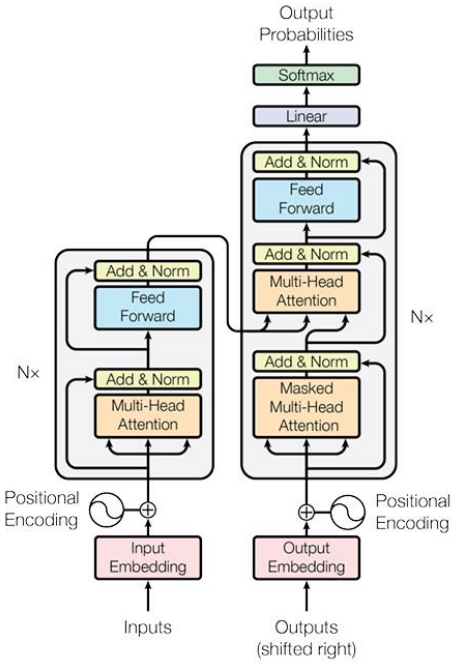
Για να μπούμε λίγο βαθύτερα, τα LLM βασίζονται κυρίως σε έναν τύπο νευρωνικού δικτύου που ονομάζεται μοντέλο μετασχηματιστή (transformer model). Για να κατανοήσουμε λίγο καλύτερα τον τρόπο λειτουργίας τους, ας αναλύσουμε λίγο την αρχιτεκτονική τους. [20]

- Αποτελούνται από έναν encoder (κωδικοποιητή), ο οποίος αποτελείται από 6 ίδια επίπεδα όπου κάθε επίπεδο έχει δύο υπο-επίπεδα:
  1. Μηχανισμός multi-head self-attention
  2. Πλήρως συνδεδεμένο δίκτυο προώθησης (feed-forward)
- Έναν decoder (αποκωδικοποιητή), επίσης 6 επιπέδων, εισάγοντας ένα τρίτο υπο-επίπεδο το οποίο εκτελεί προσοχή με πολλαπλές κεφαλές πάνω στην έξοδο της στοίβας του encoder.

Encoder: Δεδομένου ότι οι υπολογιστές δεν κατανοούν τη φυσική γλώσσα με τον ίδιο τρόπο που την κατανοούμε εμείς, το πρώτο βήμα είναι να μετατραπούν οι λέξεις της πρότασής μας σε embeddings. Κάθε λέξη σε μια εισερχόμενη πρόταση χωρίζεται σε μονάδες που ονομάζονται tokens. Αυτά τα tokens στη συνέχεια μετατρέπονται σε

vector embeddings. Αυτή η διαδικασία, είναι γνωστή ως ενσωμάτωση εισόδου (input embedding). Γνωρίζουμε, ότι, στη φυσική γλώσσα, μια λέξη μπορεί να έχει διαφορετικές σημασίες σε διαφορετικά συμφραζόμενα. Για να ληφθεί υπόψη αυτό, χρησιμοποιείται η κωδικοποίηση θέσης (positional encoding). Αυτό περιλαμβάνει τη χρήση ενός διανύσματος που κωδικοποιεί την απόσταση μεταξύ των λέξεων σε μια πρόταση, παρέχοντας πληροφορίες θέσης στα vector embeddings. Αυτά τα input embeddings μαζί με το positional encoding, προχωρούν στο μπλοκ του κωδικοποιητή: στο multi-head attention layer. Αυτό το επίπεδο επιτρέπει στο μοντέλο να επικεντρωθεί σε διαφορετικές πτυχές της εισερχόμενης πρότασης ταυτόχρονα χρησιμοποιώντας πολλαπλές 'κεφαλές', καθεμία από τις οποίες εκτελεί έναν μηχανισμό προσοχής. Η προσοχή, σε αυτό το πλαίσιο, επιτρέπει στο μοντέλο να καθορίσει ποια μέρη της εισερχόμενης πρότασης είναι πιο σημαντικά. Ένα διάνυσμα προσοχής παράγεται για κάθε λέξη, καθορίζοντας τη σημασία της λέξης σε σχέση με άλλες λέξεις στην ακολουθία εισόδου. Θα εξηγήσουμε πιο αναλυτικά την ιδέα της προσοχής σε λίγο. Τα διανύσματα προσοχής προχωρούν στη συνέχεια σε ένα νευρωνικό δίκτυο προώθησης (feed forward neural network), το οποίο βοηθά το μοντέλο να συλλάβει βασικές αναπαραστάσεις χαρακτηριστικών, μοτίβα και σχέσεις μέσα στα δεδομένα.

Decoder: Τα αρχικά βήματα είναι παρόμοια με του encoder, αλλά ως είσοδο τώρα έχουμε ta vector embeddings της target output ακολουθίας, και τα οποία εμπλουτίζονται με διανύσματα positional encoding. Τα διανύσματα αυτά τροφοδοτούνται στο μπλοκ του αποκωδικοποιητή, που περιλαμβάνει ένα επιπέδο αυτο-προσοχής με μάσκα (masked multi-head attention layer). Στο επίπεδο αυτό, για κάθε λέξη, οι επόμενες λέξεις στην ακολουθία είναι κρυμμένες από την οπτική του μοντέλου (masked = transformed to zeros.). Αφού το μοντέλο προσπαθεί να μάθει, μόνο οι προηγούμενες λέξεις στην πρόταση από το μπλοκ του αποκωδικοποιητή είναι προσβάσιμες. Το τελικό μπλοκ, το επίπεδο προσοχής με πολλαπλές κεφαλές κωδικοποιητή-αποκωδικοποιητή (multi-head attention encoder-decoder block), δημιουργεί διανύσματα προσοχής για κάθε λέξη στις ακολουθίες εισόδου και εξόδου. Αυτά τα διανύσματα προσοχής στη συνέχεια περνούν από το επίπεδο προώθησης (feed-forward layer), το γραμμικό επίπεδο (linear layer), και το επίπεδο softmax για να προβλέψουν την επόμενη λέξη στην ακολουθία.



Σχήμα 2.2: Transformers Architecture

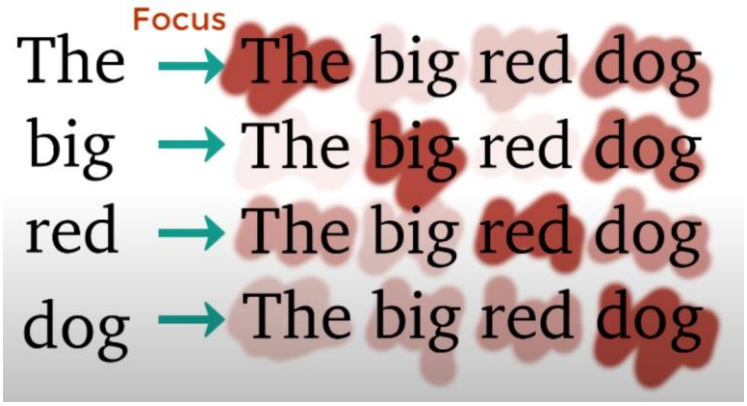
Feed-forward layer: Το δίκτυο προώθησης αποτελείται από δύο γραμμικούς μετασχηματισμούς με μια ενεργοποίηση ReLU ενδιάμεσα.

Softmax: Εφαρμόζεται ένας γραμμικός μετασχηματισμός στις εξόδους του αποκωδικοποιητή και στη συνέχεια χρησιμοποιείται η συνάρτηση softmax για να μετατραπεί η έξοδος του αποκωδικοποιητή σε προβλεπόμενες πιθανότητες του επόμενου token.

Attention: Η αρχιτεκτονική των μετασχηματιστών εισάγει την έννοια του «attention» - και την ιδέα του «να δίνουμε περισσότερη προσοχή» σε κάποια μέρη της ακολουθίας από άλλα. Η προσοχή στους μετασχηματιστές λειτουργεί μέσω τριών βασικών στοιχείων: τα ερωτήματα (queries), τα κλειδιά (keys) και τις τιμές (values). Κάθε στοιχείο της ακολουθίας εισόδου παράγει ένα ζευγάρι query-key, και η "προσοχή" υπολογίζεται με βάση τη συμβατότητα μεταξύ τους. Ουσιαστικά, το μοντέλο "ρωτάει" ποια μέρη της ακολουθίας είναι σημαντικά σχετικά με το τρέχον κομμάτι που αναλύεται. Έτσι, το μοντέλο υπολογίζει ένα βάρος για κάθε key, το οποίο δηλώνει το πόσο σημαντικό είναι (και κατ' επέκταση το κομμάτι κειμένου που αντιπροσωπεύει) για το query. Τα βάρη αυτά μετά χρησιμοποιούνται για να συνδυαστούν τα values σε ένα

συνολικό διάνυσμα εξόδου για το ερώτημα, δίνοντας περισσότερη έμφαση στα σημαντικότερα values. [8,20] Το μοντέλο Transformer χρησιμοποιεί την προσοχή με πολλαπλές κεφαλές με τρεις διαφορετικούς τρόπους:

1. Επίπεδα Προσοχής Κωδικοποιητή-Αποκωδικοποιητή: Τα ερωτήματα (queries) προέρχονται από το προηγούμενο επίπεδο του αποκωδικοποιητή, ενώ τα κλειδιά (keys) και οι τιμές (values) μνήμης προέρχονται από την έξοδο του κωδικοποιητή. Αυτό επιτρέπει σε κάθε θέση στον αποκωδικοποιητή να προσέχει όλες τις θέσεις στην ακολουθία εισόδου.
2. Επίπεδα Αυτο-προσοχής στον Κωδικοποιητή: Όλα τα κλειδιά, οι τιμές και τα ερωτήματα προέρχονται από την ίδια θέση, δηλαδή από την έξοδο του προηγούμενου επιπέδου στον κωδικοποιητή. Κάθε θέση στον κωδικοποιητή μπορεί να προσέχει όλες τις θέσεις στο προηγούμενο επίπεδο του.
3. Επίπεδα Αυτο-προσοχής στον Αποκωδικοποιητή: Παρόμοια με τον κωδικοποιητή, τα επίπεδα αυτο-προσοχής στον αποκωδικοποιητή επιτρέπουν σε κάθε θέση στον αποκωδικοποιητή να προσέχει όλες τις θέσεις στον αποκωδικοποιητή μέχρι και τη συγκεκριμένη θέση.



Σχήμα 2.3: Attention

Positional Encoding: Η κωδικοποίηση θέσης (positional encoding) χρησιμοποιείται για να παρέχει μια σχετική θέση για κάθε λέξη ή token σε μια ακολουθία. Όταν διαβάζουμε μια πρόταση, κάθε λέξη εξαρτάται από τις λέξεις γύρω της. Ορισμένες λέξεις έχουν διαφορετικές σημασίες σε διαφορετικά συμφραζόμενα, οπότε ένα μοντέλο πρέπει να μπορεί να κατανοεί αυτές τις παραλλαγές και τις λέξεις από τις οποίες εξαρτάται για το νόημα.

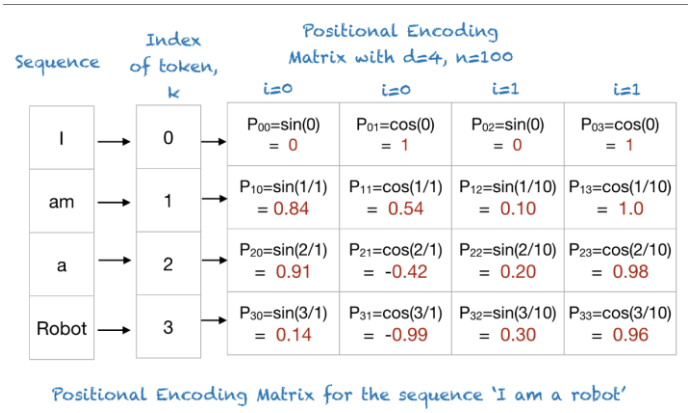
Κάθε θέση στην ακολουθία έχει μια μοναδική αναπαράσταση που την ξεχωρίζει από τις άλλες θέσεις και το αποτέλεσμα της κωδικοποίησης θέσης είναι ένας πίνακας, όπου κάθε γραμμή αναπαριστά την κωδικοποίηση θέσης ενός στοιχείου της ακολουθίας. Γιατί χρησιμοποιείται position encoding πίνακας αντί για απλούς δείκτες; Ο λόγος είναι ότι σε μεγάλες ακολουθίες, οι απλοί δείκτες μπορεί να γίνουν πολύ μεγάλοι γρήγορα και δεν μπορούν να προστεθούν εύκολα σε έναν πίνακα ενσωμάτωσης. [15,18]

Για να εξασφαλιστεί ότι κάθε θέση έχει μια μοναδική αναπαράσταση, χρησιμοποιήθηκαν οι συναρτήσεις ημιτόνου και συνημιτόνου. Ο λόγος είναι, πρώτον επειδή η έξοδος των ημιτόνου και συνημιτόνου βρίσκεται στο διάστημα [−1,1], το οποίο είναι κανονικοποιημένο, άρα δεν θα αυξηθεί σε μη διαχειρίσιμο μέγεθος όπως οι ακέραιοι αριθμοί και δεύτερον, δεν χρειάζεται επιπλέον εκπαίδευση, καθώς δημιουργούνται μοναδικές αναπαραστάσεις για κάθε θέση.

Ο αλγόριθμος κωδικοποίησης λειτουργεί ως εξής:
Για κάθε διάνυσμα κωδικοποίησης θέσης, για κάθε δύο στοιχεία, τοποθετήστε το ζυγό στοιχείο ίσο με PE(k,2i) και το μονό στοιχείο ίσο με PE(k,2i+1). Επανάλαβε μέχρι να υπάρχουν d_{model} (length of vector embeddings) στοιχεία στο διάνυσμα:

- For each $k = 0$ to $L - 1$:
  - For each $i = 0$ to $\frac{d_{model}}{2}$:
    - $PE_{(k,2i)} = sin(\frac{k}{n^{\frac{2i}{d_{model}}}})$
    - $PE_{(k,2i+1)} = cos(\frac{k}{n^{\frac{2i}{d_{model}}}})$

Σχήμα 2.4: Positional Encoding Algorithm (L = number of tokens, k = position, n = 10,000 recommended value)



Σχήμα 2.5: Παράδειγμα Positional Encoding

2.2 Τα LLMs στην Εκπαίδευση

Ας μιλήσουμε για τον τομέα της εκπαίδευσης και της μάθησης, στον οποίο τα LLMs προσφέρουν πολλαπλά πλεονεκτήματα ενισχύοντας τόσο την ποιότητα όσο και την προσβασιμότητα της γνώσης, και παρέχοντας ένα εργαλείο που μπορεί να μετασχηματίσει τον τρόπο διδασκαλίας και παράδοσης γνώσης. [5]

Μερικοί τρόποι που μπορούν τα μοντέλα αυτά να προσφέρουν στο πεδίο αυτό είναι οι εξής:

- Να προσαρμόσουν την μάθηση, αναλύοντας αρχικά τις απαντήσεις, τη συμπεριφορά και την επίδοση του κάθε μαθητή ξεχωριστά, ώστε να προσαρμόσουν το εκπαιδευτικό υλικό για να ταιριάζει στις ανάγκες, δυνατότητες και αδυναμίες και τον ρυθμό μάθησης του καθενός.
- Αυτοματοποιημένη παραγωγή εκπαιδευτικού περιεχομένου, παραγωγή δηλαδή διαγωνισμάτων, ασκήσεων, παιχνιδιών και άλλων εργαλείων, με αποτέλεσμα να μειώνεται ο χρόνος που δαπανούν οι εκπαιδευτικοί για την προετοιμασία υλικού, αλλά να έχουν και πιο πρωτότυπες και δημιουργικές ιδέες.
- Βοήθεια στην αξιολόγηση γραπτών εργασιών και διαγωνισμάτων και παροχή βαθμολογιών και σχολιασμών για την επίδοση των μαθητών.
- Παροχή μιας ακόμα πηγής πληροφοριών, πολύ εύκολης στη χρήση, ακόμα και εξατομικευμένης, αφού προσφέρει της ανάλογες εξηγήσεις βάσει του τι έχουν ακριβώς ερωτηθεί.

- Εισαγωγή νέων μεθόδων διδασκαλίας, αφού μπορούν να βρουν νέες ιδέες, και να εισηγηθούν ένα πρόγραμμα μαθήματος, ή οδηγίες για τον εκπαιδευτικό και το πώς να προσεγγίσει και να διδάξει κάποιο συγκεκριμένο θέμα.

Υπάρχουν είδη αρκετά tools τα οποία εξειδικεύονται στο πεδίο της μάθησης, όπως είναι για παράδειγμα το Education Copilot το οποίο προσφέρει διάφορες επιλογές για τους εκπαιδευτικούς όπως για παράδειγμα τη δημιουργία φυλλαδίων ανάγνωσης, δημιουργία πλάνων διδασκαλίας, εργασίες και πολλά άλλα. Αξίζει να αναφερθεί επίσης ότι υπάρχουν και ισότοποι οι οποίοι έχουν ως στόχο τη προώθηση του Generative AI στην εκπαίδευση, παρέχοντας στρατηγικές, οδηγίες και γενικά καθοδήγηση για να γίνεται σωστή χρήση και να αποφεύγονται κίνδυνοι. Ένα παράδειγμα τέτοιου οργανισμού είναι το AI For Education.

Το ChatGPT αποτελεί είδη έναν εξαιρετικό βοηθό όσο αφορά την εκπαίδευση. [17] Ας αναλύσουμε μερικές από τις δυνατότητες που έχει ως βοηθός στον τομέα αυτό:

1. Δημιουργία πλάνων διδασκαλίας. Μπορεί να προγραμματίσει ένα μάθημα και να καθοδηγήσει τον εκπαιδευτικό για το πώς να το κάνει, με βάση της προτιμήσεις του ίδιου του εκπαιδευτικού.
2. Δημιουργία παραδειγμάτων, επεξηγήσεων και αναλογιών που βοηθούν τους μαθητές να κατανοήσουν με πολύ παραγωγικό τρόπο κάποιο θέμα.
3. Παραγωγή ασκήσεων, διαγωνισμάτων ή και παιχνιδιών.
4. Μπορεί ακόμη να χρησιμοποιηθεί και εντός τάξης, δίνοντας του κάποιες εντολές και λέγοντάς του να πάρει κάποιο ρόλο και βάζοντας τους μαθητές να έχουν συνομιλίες μαζί του. Για παράδειγμα, μπορεί να πάρει το ρόλο ενός μαθητή που έδωσε μια απάντηση και να ζητήσει από τον πραγματικό μαθητή να βρει το λάθος στην απάντηση αυτή.
5. Αξιολόγηση ενός μαθητή και παροχή feedback, ώστε να μπορεί να δοθεί περισσότερη προσοχή στις αδυναμίες του κάθε μαθητή.

Η ίδια η OpenAI παρέχει οδηγίες για το πώς ένας εκπαιδευτικός μπορεί να χρησιμοποιήσει το ChatGPT παραθέτοντας κάποια prompts που μπορούν να χρησιμοποιηθούν σε κάθε περίπτωση. [14]

2.3 Τι είναι το ChatGPT και πώς δουλεύει

Το κύριο εργαλείο που χρησιμοποιείται για την ανάπτυξη του συστήματός μας, αποτελεί το ChatGPT. Για αυτό, πριν προχωρήσουμε στο επόμενο στάδιο, στην παραγωγή δηλαδή ενός συστήματος που δουλεύει με παρόμοιο τρόπο, πρέπει να καταλάβουμε τον τρόπο λειτουργίας του. [21]

Ξεκινώντας, ο βασικός στόχος του μοντέλου αυτού είναι να παράγει μια «λογική συνέχεια» οποιουδήποτε κειμένου, όπου με τον όρο «λογική» εννοούμε «κάτι που θα περίμενε κανείς να γράψει κάποιος» αφού έχει εκπαιδευτεί σε δεδομένα του τι έχουν γράψει οι άνθρωποι σε δισεκατομμύρια ιστοσελίδες, βιβλία κλπ. Ο τρόπος με τον οποίο εκτελεί αυτή τη διαδικασία, είναι βασικά οι πιθανότητες. Πιο συγκεκριμένα, κάθε φορά που καλείται να "μαντέψει" ποια πρέπει να είναι η επόμενη λέξη στη πρόταση, λαμβάνει μια λίστα από τις πιθανές λέξεις και τις πιθανότητες συχνότητας χρήσης τους (με βάση την μέχρι τώρα εκπαίδευσή του). Βλέπουμε ένα παράδειγμα στην επόμενη εικόνα:

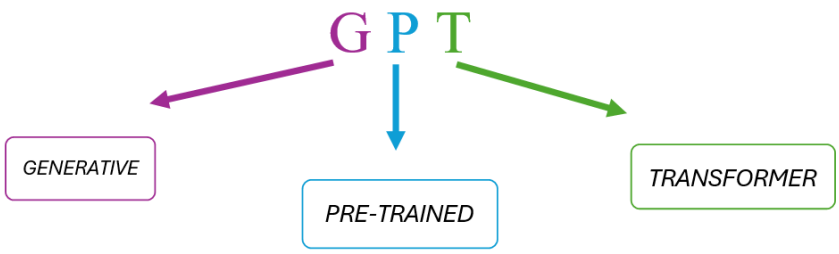
The best thing about AI is its ability to

learn	4.5%
predict	3.5%
make	3.2%
understand	3.1%
do	2.9%

Σχήμα 2.6: Παράδειγμα επιλογών λέξεων και πιθανότητές τους

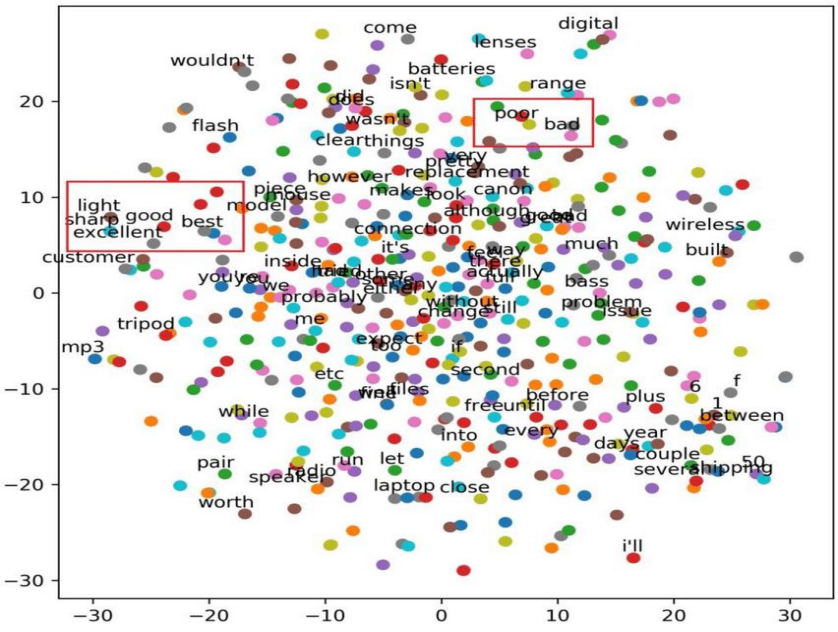
Το "μαγικό" όμως αυτής της διαδικασίας, είναι το γεγονός ότι το ChatGPT , προφανώς, δεν μπορεί να διαλέγει πάντα την λέξη που έρχεται πρώτη στη σειρά, διότι έτσι θα έχουμε πάντα το ίδιο μονότονο αποτέλεσμα. Επομένως, γνωρίζουμε ότι υπάρχει μια παράμετρος με όνομα "temperature", η οποία καθορίζει το πόσο συχνά θα χρησιμοποιούνται λέξεις χαμηλότερης κατάταξης στη λίστα με τις πιθανότητες. Να σημειωθεί ότι δεν υπάρχει κάποια ακριβής θεωρία πίσω από αυτό, αλλά

χρησιμοποιείται αυτό που φαίνεται να δουλεύει καλύτερα στην πράξη. Επίσης, αφού τα LLMs είναι τεράστια Neural Networks, να σημειώσουμε ότι το GPT-3 αποτελείται από 175 δισεκατομμύρια παραμέτρους και το GPT-4 λέγεται ότι μπορεί να έχει γύρω στα 1.7 τρισεκατομμύρια. Για να κατανοήσουμε καλύτερα, θα δούμε με σειρά τα σημαντικότερα βήματα της λειτουργίας ενός τέτοιου συστήματος. Προτού παραθέσουμε τα βήματα όμως, ας δούμε από που προέρχεται το όνομα ChatGPT και πώς το όνομα αυτό προδίδει βασικά όλη την αρχιτεκτονική πίσω από αυτό:



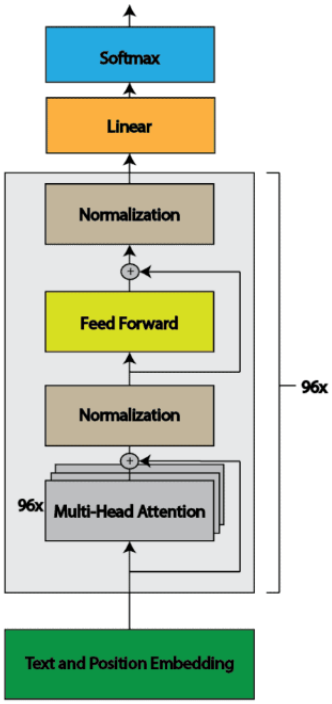
Σχήμα 2.7: Σημασία της ονομασίας GPT

Ως transformer, μια πολύ σημαντική διαδικασία που εκτελεί, είναι τα embeddings. Embeddings είναι στην ουσία μια αριθμητική αναπαράσταση του νοήματος μιας λέξης. Αυτό χρειάζεται επειδή τα νευρωνικά δίκτυα δουλεύουν με νούμερα και συναρτήσεις, άρα για να μπορούν να εκτελεστούν διάφορες λειτουργίες, έπρεπε να βρούμε ένα τρόπο να μετατρέψουμε τη φυσική γλώσσα σε νούμερα. Ως αποτέλεσμα, έχουμε διανύσματα τα οποία αντιπροσωπεύουν την ουσία των λέξεων, και έτσι, όταν θέλουμε να βρούμε την ομοιότητα μεταξύ δύο προτάσεων ή λέξεων, θα βρούμε απλώς την απόσταση μεταξύ τους. [6] Αυτό κάνει το ChatGPT κάθε φορά που πρέπει να προβλέψει την επόμενη λέξη σε μια ακολουθία. Για την ακρίβεια όμως, το ChatGPT δεν εργάζεται με λέξεις, αλλά με tokens. Τα tokens μπορεί να είναι κομμάτια λέξεων, ολόκληρες λέξεις ή σύμβολα.[10] Άρα, ξεκινώντας, δέχεται ως είσοδο ένα διάνυσμα από n tokens, μετατρέπει το κάθε token σε ένα διάνυσμα μεγέθους 768 (GPT-2) ή 12,288 (GPT-3) και παράλληλα, υπάρχει ένας "δευτερεύων δρόμος" που λαμβάνει τη σειρά των (ακεραίων) θέσεων για τα tokens, και από αυτούς τους ακεραίους δημιουργεί ένα άλλο διάνυσμα ενσωμάτωσης. Τελικά, τα embeddings από την τιμή του token και τη θέση του token προστίθενται μαζί για να παράγουν το τελικό embedding vector.



Σχήμα 2.8: Απεικόνιση του word embedding space

Μετά την δημιουργία των embeddings, έρχεται η κυριότερη διαδικασία ενός transformer. Το attention. Το GPT-3, αποτελείται από 96 attention blocks, όπου το κάθε ένα από αυτά συμπεριλαμβάνει 96 attention heads. [22]



Σχήμα 2.9: GPT transformer architecture

Όπως βλέπουμε στο σχήμα 2.6, το GPT είναι κατασκευασμένο χρησιμοποιώντας μόνο μπλοκ αποκωδικοποιητών transformers.

Το training ενός τέτοιου μοντέλου χωρίζεται σε 3 βασικές φάσεις. Το pre-training, το supervised fine-tuning (SFT) και το reinforcement learning from human feedback (RLHF). Στη φάση του pre-training γίνεται ότι ακριβώς είπαμε μέχρι τώρα, το σύστημα εκπαιδεύεται σε μεγάλα δεδομένα για να προβλέπει την επόμενη λέξη σε μια ακολουθία. Αυτό όμως δεν το κάνει έναν βοηθό ο οποίος μπορεί να ακολουθεί κάποιες εντολές και να απαντά σωστά σε ερωτήσεις. Αυτό μαθαίνεται στην επόμενη φάση της μάθησης: Παίρνουμε το προ-εκπαιδευμένο LLM με τις τρέχουσες δυνατότητές του και κάνουμε ουσιαστικά αυτό που κάναμε πριν, μαθαίνουμε να προβλέπουμε μία λέξη τη φορά, αλλά τώρα το κάνουμε χρησιμοποιώντας μόνο ζευγάρια οδηγιών και απαντήσεων υψηλής ποιότητας ως δεδομένα εκπαίδευσης. Με αυτόν τον τρόπο, το μοντέλο «ξεμαθαίνει» να λειτουργεί απλώς ως συμπληρωτής κειμένου και μαθαίνει να γίνεται ένας χρήσιμος βοηθός που ακολουθεί οδηγίες και ανταποκρίνεται με τρόπο που είναι συντονισμένος με τις προθέσεις του χρήστη. Το μέγεθος αυτού του συνόλου δεδομένων οδηγιών είναι συνήθως πολύ μικρότερο από το σετ προ-εκπαίδευσης. Αυτό οφείλεται στο ότι τα ζευγάρια οδηγιών και απαντήσεων υψηλής ποιότητας είναι πολύ πιο ακριβά στη δημιουργία, καθώς συνήθως προέρχονται από ανθρώπους. Τέλος, στο τρίτο βήμα, χρησιμοποιείται πάλι το μοντέλο SFT και διαφορετικά prompts τροφοδοτούνται στο μοντέλο και παράγονται διαφορετικές απαντήσεις για κάθε είσοδο. Στη συνέχεια, ο labeler καθορίζει ένα reward για κάθε μία από αυτές τις απαντήσεις, το οποίο είναι ανάλογο με την ποιότητα της απάντησης σε σχέση με το αρχικό ερώτημα. Ο labeler κατατάσσει τις απαντήσεις σε ακολουθία από την καλύτερη έως τη χειρότερη. Στη συνέχεια, μπορούμε να χρησιμοποιήσουμε αυτά τα δεδομένα για να εκπαιδεύσουμε ένα μοντέλο ανταμοιβής. Η είσοδος για ένα μοντέλο ανταμοιβής θα είναι το ερώτημα του χρήστη και μία από τις απαντήσεις που παρήγαγαν και η έξοδος θα είναι μια αριθμητική τιμή που καθορίζει την ποιότητα της απάντησης σε σχέση με το ερώτημα εισόδου. [7]

Το RLHF βοηθά στην ευθυγράμμιση και διασφαλίζει ότι η έξοδος του LLM αντικατοπτρίζει τις ανθρώπινες αξίες και προτιμήσεις. Υπάρχουν έρευνες που δείχνουν ότι αυτό το στάδιο είναι κρίσιμο για την επίτευξη ή ακόμα και την υπέρβαση της

ανθρώπινης απόδοσης. Πράγματι, ο συνδυασμός των πεδίων της ενισχυτικής μάθησης και της μοντελοποίησης της γλώσσας φαίνεται ιδιαίτερα υποσχόμενος και πιθανό να οδηγήσει σε μεγάλες βελτιώσεις.

2.4 Στρατηγικές Prompt Engineering του ChatGPT

Σύμφωνα με την OpenAI, υπάρχουν κάποιες στρατηγικές όσο αφορά το prompting, οι οποίες αν ακολουθηθούν, θα μας φέρουν καλύτερα αποτελέσματα. Θα αναφέρουμε κάποιες σημαντικές τακτικές:

- ✓ Γράφουμε καθαρές οδηγίες για το τι ακριβώς θέλουμε, παραθέτοντας λεπτομέρειες. Για παράδειγμα, μπορούμε να καθορίσουμε το επιθυμητό μήκος της απάντησης, να δώσουμε στο μοντέλο παραδείγματα για το τι ακριβώς περιμένουμε να κάνει. Μπορούμε ακόμα να προσδιορίσουμε τα βήματα που απαιτούνται για την ολοκλήρωση μιας εργασίας ή και να του ζητήσουμε να υιοθετήσει μια προσωπικότητα και να εργάζεται όπως θα εργαζόταν αυτή. Τέλος, καλό είναι να χρησιμοποιούμε διαχωριστικά μέσα στις εντολές μας, για να είναι ξεκάθαρο στο μοντέλο ποια είναι τα διακριτά τμήματα της εισόδου μας.
- ✓ Παροχή κειμένου αναφοράς. Για αποφυγή των ψευδών απαντήσεων μπορούμε να ζητήσουμε από το μοντέλο να απαντήσει μια ερώτηση χρησιμοποιώντας κείμενο αναφοράς ή να απαντήσει με παραπομπές από το κείμενο.
- ✓ Διαχωρίζουμε περίπλοκες εργασίες σε απλούστερες υπο-εργασίες. Όπως αναφέρει η OpenAI, οι περίπλοκες εργασίες τείνουν να έχουν υψηλότερα ποσοστά λαθών από τις απλούστερες. Μπορούμε να συνοψίζουμε μεγάλα έγγραφα κατά τμήμα και να κατασκευάζουμε μια πλήρη σύνοψη αναδρομικά, ή να ταξινομούμε προθέσεις για εντοπισμό των πιο σχετικών οδηγιών για μια ερώτηση χρήστη.
- ✓ Δίνουμε στο μοντέλο χρόνο να "σκεφτεί". Τέτοια μοντέλα κάνουν περισσότερα λάθη στη λογική, όταν προσπαθούν να απαντήσουν αμέσως κάτι το οποίο, απαιτεί μια ακολουθία από βήματα και άρα περισσότερο χρόνο για να απαντηθούν. Ζητούμε μια "αλυσίδα σκέψεων", ώστε να αναγκάσουμε το σύστημα να ακολουθήσει όντως τις σκέψεις αυτές και να λογικεύει τον δρόμο του προς πιο αξιόπιστες απαντήσεις. Κάποιες τακτικές που μπορούμε να ακολουθήσουμε είναι να δώσουμε οδηγίες στο μοντέλο να εργαστεί για την

λύση του πριν βιαστεί να καταλήξει σε ένα συμπέρασμα ή ακόμα και το να ρωτήσουμε το ίδιο το μοντέλο αν έχει ξεχάσει κάτι σε προηγούμενες του διαδικασίες, μπορεί να βοηθήσει στη βελτίωση.

- ✓ Χρήση εξωτερικών εργαλείων: Χρήση αναζήτησης βασισμένη σε embeddings για την αποτελεσματική ανάκτηση γνώσης, χρήση της εκτέλεσης κώδικα για πιο ακριβείς υπολογισμούς ή κλήση εξωτερικών API, πρόσβαση του μοντέλου σε συγκεκριμένα functions.

Κεφάλαιο 3

Υλοποιημένο Σύστημα

3.1 Σκέψη πίσω από την Υλοποίηση

Όπως αναφέρθηκε και στο πρώτο κεφάλαιο, το σύστημα αυτό φτιάχτηκε για να βοηθήσει τους εκπαιδευτικούς να εξοικονομούν χρόνο και κόπο. Η ιδέα πίσω από ένα τέτοιο σύστημα είναι η επίλυση του προβλήματος που δημιουργείται λόγω του ότι τα δεδομένα εκπαίδευσης των LLM είναι στατικά και έχουν ενημερωθεί μέχρι μια συγκεκριμένη ημερομηνία, από συγκεκριμένες πηγές. Άρα, στοχεύουμε στην αποφυγή της παρουσίασης ψευδών πληροφοριών ή της δημιουργίας απαντήσεων από άσχετες ή μη αξιόπιστες πηγές ή της παραγωγής γενικών πληροφοριών όταν ο χρήστης αναμένει μια πιο συγκεκριμένη, και εξιδεικευμένη απάντηση.

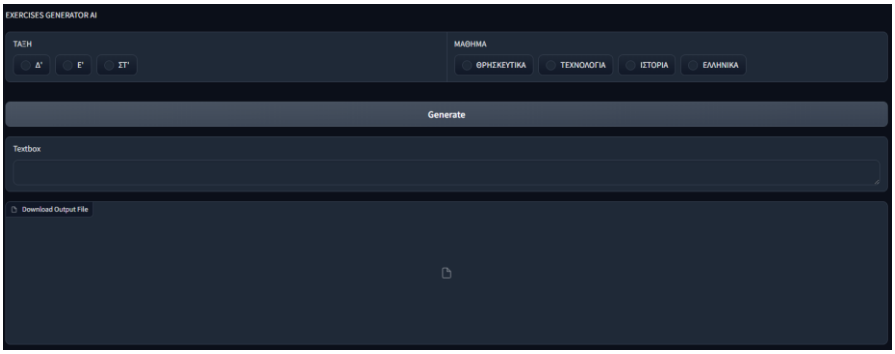
Θέλουμε δηλαδή ένα τρόπο, να ενισχύσουμε την εμπιστοσύνη των χρηστών, παρέχοντάς τους ένα μοντέλο το οποίο να παρουσιάζει ακριβείς πληροφορίες και απαντήσεις βασισμένες σε αυτό που όντως ψάχνουν. Έτσι, θα πρέπει να «εκπαιδεύσουμε» το LLM μας, με κάποια εξωτερικά δεδομένα, τα οποία θα είναι τα βιβλία κάποιων τάξεων και μαθημάτων του Δημοτικού σχολείου, ώστε να δώσουμε ένα συγκεκριμένο υπόβαθρο στο οποίο θα βασίζεται για της απαντήσεις του.

Ένα άλλο πλεονέκτημα που θέλουμε να έχει το σύστημα μας, σε σχέση με τα LLMs, είναι το ότι οι χρήστες δεν θα χρειάζεται να μπουν στη διαδικασία να γράψουν και να εξηγήσουν με λεπτομέρεια τι χρειάζονται, ούτε να δώσουν πληροφορίες για το θέμα, αφού το μοντέλο θα είναι ανατροφοδοτημένο είδη με όλα τα δεδομένα, όλες τις εντολές

που πρέπει να ακολουθήσει, ακόμα και με παραδείγματα για το τι περιμένουμε να φτιάξει. Θα είναι προσαρμοσμένο ειδικά για αυτόν τον σκοπό και οι εκπαιδευτικοί το μόνο που θα χρειαστεί να κάνουν θα είναι απλά clicks για να επιλέξουν την τάξη, το μάθημα, το κεφάλαιο και τα είδη ερωτήσεων με τα οποία θα ανατροφοδοτηθεί και θα προσαρμοστεί το μοντέλο.

3.2 Παρουσίαση του Συστήματος

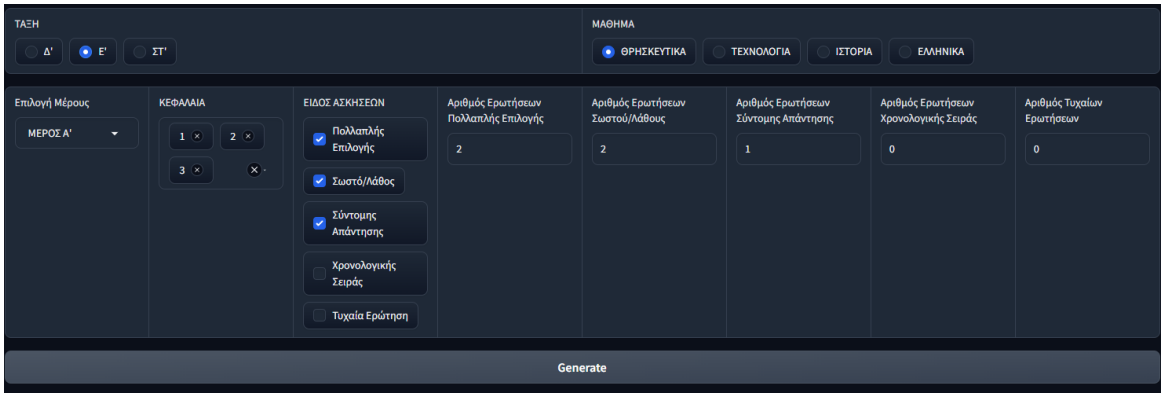
Ας δούμε πως είναι το υλοποιημένο σύστημα, τις διάφορες επιλογές που προσφέρει στους χρήστες και μερικά από τα αποτελέσματά του.
Μόλις ενωθούμε στο link, βλέπουμε το περιβάλλον της σελίδας, όπου αρχικά εμφανίζονται μόνο οι επιλογές τάξης και μαθήματος:



Σχήμα 3.1: Αρχική σελίδα

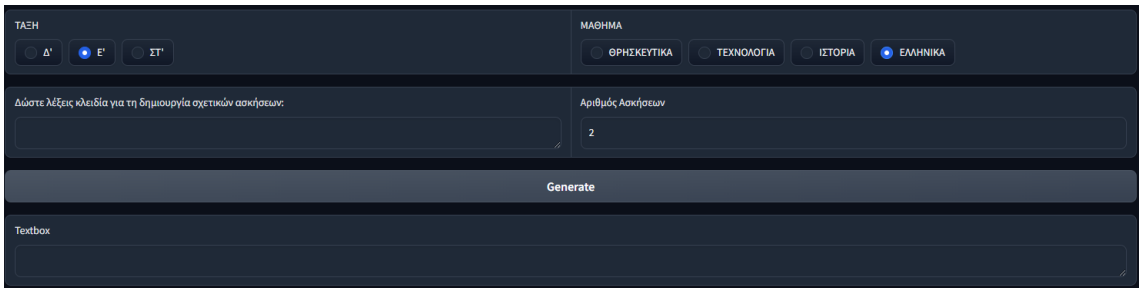
Έπειτα, αφού επιλεγεί ένα μάθημα από τα Θρησκευτικά, Τεχνολογία, ή Ιστορία, εμφανίζονται περισσότερες επιλογές όπως: Επιλογή Μέρους (Μέρος Α', Μέρος Β' μόνο για το μάθημα τον Θρησκευτικών), Επιλογή Κεφαλαίων (περιλαμβάνεται διαφορετικός αριθμός κεφαλαίων για κάθε μάθημα) και ο μέγιστος αριθμός κεφαλαίων που μπορεί να επιλεγεί είναι μέχρι 3 κεφάλαια.
Αυτό, διότι αν επιλεγούν περισσότερα κεφάλαια, ίσως το prompt μας τελικά να γίνεται πολύ μεγάλο, και το μοντέλο να μην μπορεί να το χειριστεί ολόκληρο, ή να μην δίνει σημασία σε όλες της πληροφορίες από κάποιο σημείο και μετά. Έτσι για να το αποφύγουμε αυτό, βάλαμε ένα όριο. Όπως βλέπουμε και στην πιο κάτω εικόνα, προσφέρονται 4 είδη ασκήσεων: ερώτηση πολλαπλής επιλογής, σωστού/λάθους, σύντομης απάντησης και χρονολογικής σειράς. Υπάρχει επίσης και η επιλογή

παραγωγής τυχαίας ερώτησης, η οποία μπορεί να αποτελεί άσκηση συμπλήρωσης κενών, αντιστοίχισης… Θα δούμε στη συνέχεια μερικά παραδείγματα.



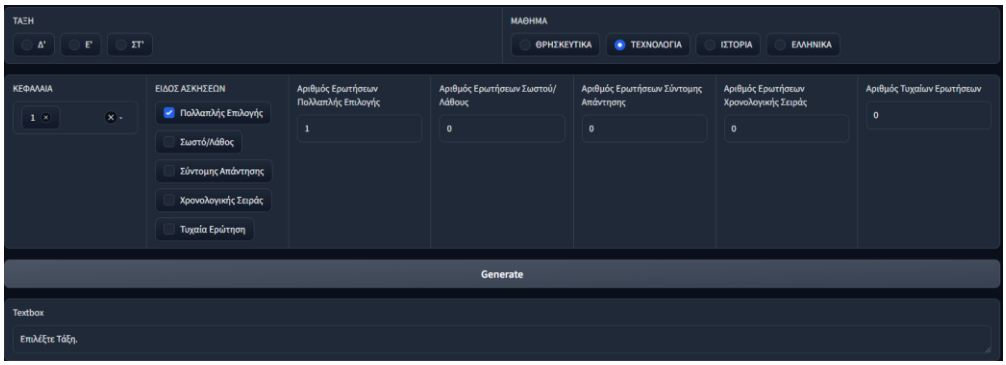
Σχήμα 3.2: Σελίδα μετά από επιλογή μαθήματος Θρησκευτικών

Όσον αφορά την επιλογή του μαθήματος Ελληνικών, τα πράγματα είναι λίγο διαφορετικά. Σε αυτό το μάθημα, ο χρήστης καλείται να εισάγει κάποια keywords, τα οποία θα χρησιμοποιηθούν για να βρεθούν ασκήσεις από τα documents οι οποίες αφορούν τα συγκεκριμένα keywords, όπως επίσης καλείται να επιλέξει και τον αριθμό ερωτήσεων που επιθυμεί. Οι ερωτήσεις δεν καθορίζονται από κάποιο συγκεκριμένο είδος, αλλά θα είναι παρόμοιου είδους με τις ασκήσεις που γίνονται retrieve και δίνονται στο σύστημα.

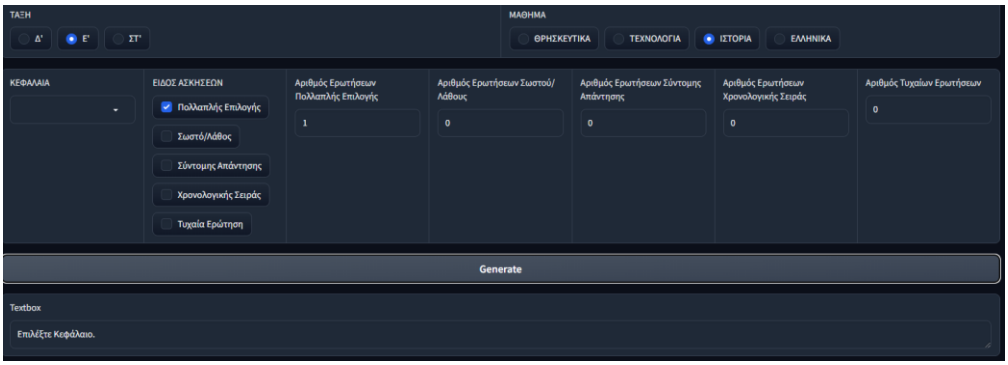


Σχήμα 3.3: Σελίδα μετά από επιλογή μαθήματος Ελληνικών

Σε περίπτωση που ο χρήστης παραλείψει να επιλέξει κάποια αναγκαία επιλογή για το σύστημα να μπορεί να λειτουργήσει, εμφανίζονται ανάλογα μηνύματα:



Σχήμα 3.4: Έξοδος μετά από παράλειψη επιλογής τάξης



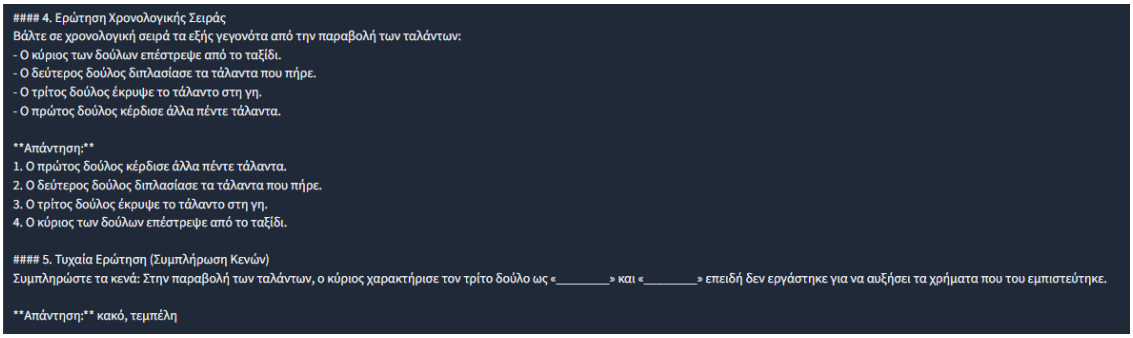
Σχήμα 3.5: Έξοδος μετά από παράλειψη επιλογής κεφαλαίου

3.3 Διάφορα Αποτελέσματα Συστήματος

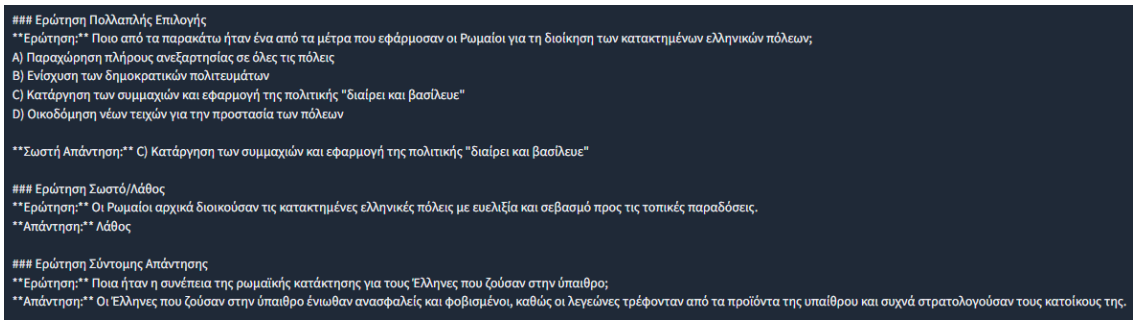
Σε αυτό το σημείο θα εξετάσουμε μερικά αποτελέσματα του συστήματος. Πιο κάτω βλέπουμε τις απαντήσεις που δίνει το σύστημα με διάφορες επιλογές:

```
### Ερωτήσεις Διαγωνίσματος για μαθητές Δ' Δημοτικού στο μάθημα Θρησκευτικά

#### 1. Ερώτηση Πολλαπλής Επιλογής
Ποιο μυστήριο συνοδεύεται από το μυστήριο του Χρίσματος, όπου το Άγιο Πνεύμα δίνει τα χαρίσματά του;
- (α) Εξομολόγηση
- (β) Γάμος
- (γ) Βάπτισμα
- (δ) Ευχέλαιο

**Σωστή απάντηση: (γ) Βάπτισμα**

#### 2. Ερώτηση Σωστό/Λάθος
Ο τρίτος δούλος στην παραβολή των ταλάντων έκρυψε το τάλαντο στη γη για να το προστατεύσει από τους κλέφτες.
- Σωστό
- Λάθος

**Σωστή απάντηση: Λάθος** (Το έκρυψε γιατί φοβόταν τον κύριό του και ήθελε να αποφύγει τη δουλειά)

#### 3. Ερώτηση Σύντομης Απάντησης
Ποιος είναι ο κύριος λόγος που οι χριστιανοί συγκεντρώνονται στον ναό;
**Απάντηση:** Για να ευχαριστήσουν και να δοξολογήσουν τον Θεό.
```

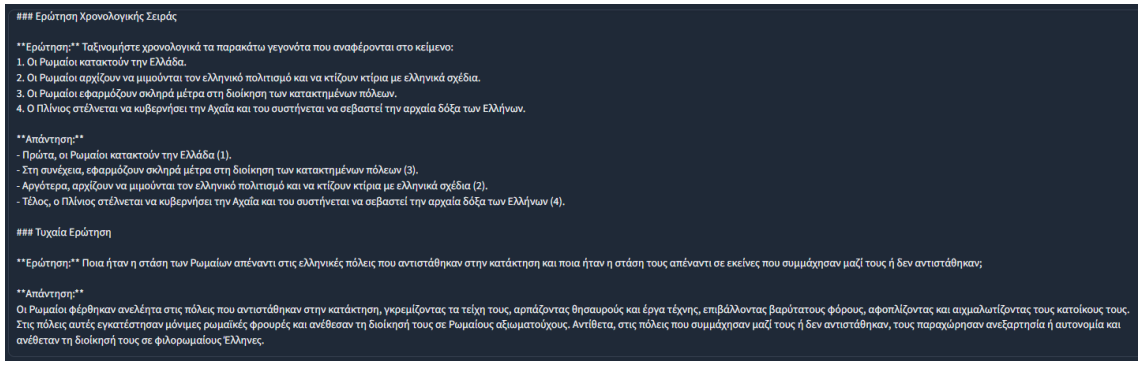
Σχήμα 3.6: Ασκήσεις Πολλαπλής Επιλογής, Σωστού/Λάθους, Σύντομης Απάντησης για Θρησκευτικά



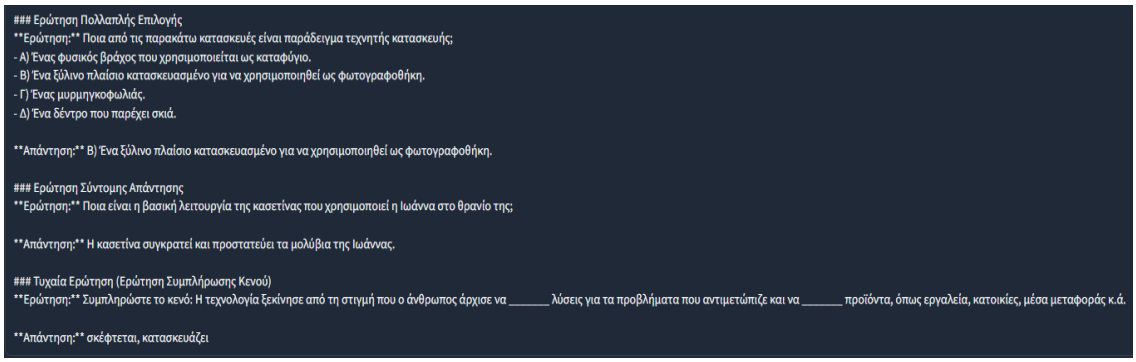
Σχήμα 3.7: Ασκήσεις Χρονολογικής Σειράς, Συμπλήρωσης Κενών για Θρησκευτικά



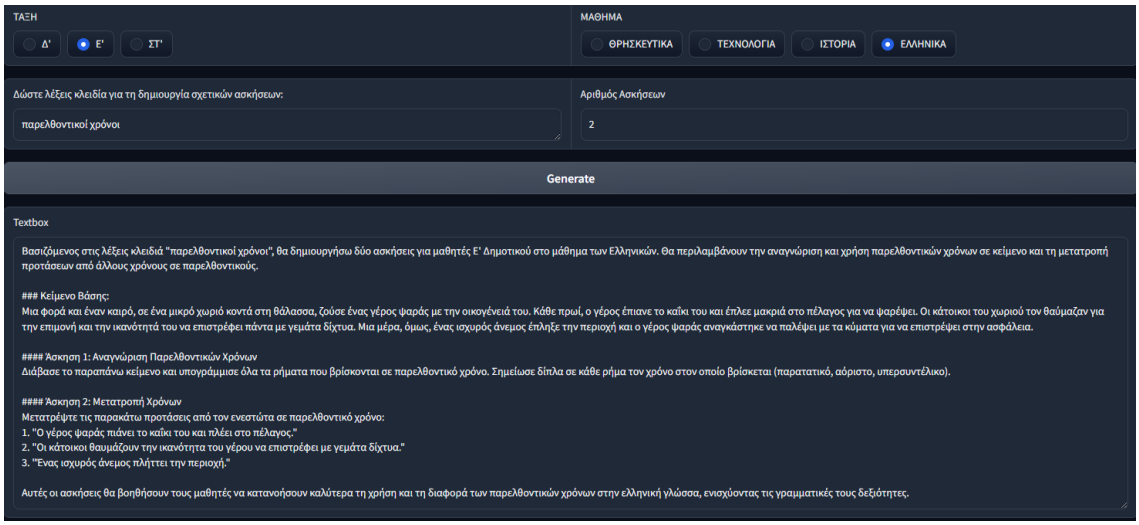
Σχήμα 3.7: Ασκήσεις Πολλαπλής Επιλογής, Σωστού/Λάθους, Σύντομης Απάντησης για Ιστορία



Σχήμα 3.8: Ασκήσεις Χροολογικής Σειράς, Τυχαία Ερώτηση για Ιστορία



Σχήμα 3.9: Ασκήσεις Πολλαπλής Επιλογής και Σύντομης Απάντησης για Τεχνολογία



Σχήμα 3.10: Ασκήσεις για Ελληνικά από λέξεις κλειδιά

Κεφάλαιο 4

Διαδικασία Υλοποίησης Συστήματος

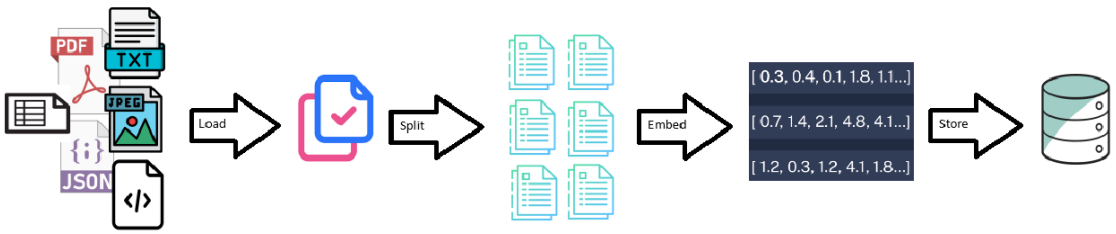
4.1 Εισαγωγή στο RAG

Όπως αναφέραμε πιο πάνω, ένα μοντέλο επεξεργασίας φυσικής γλώσσας, μπορεί να προβεί σε κατασκευή φανταστικών απαντήσεων ή στην παροχή ψευδών πληροφοριών. Αυτό τις περισσότερες φορές οφείλεται στην έλλειψη δεδομένων και γνώσεων του συστήματος. Το πρόβλημα αυτό μπορεί να λυθεί, όταν εμείς εξειδικεύσουμε και συγκεκριμενοποιήσουμε το περιεχόμενο των απαντήσεων του μοντέλου, σε κάποιο θέμα. Στο στάδιο αυτό μπορούμε να εισάγουμε την τεχνική του RAG (Retrieval Augmented Generation). RAG, είναι μια τεχνική αύξησης των γνώσεων ενός LLM με

επιπρόσθετα δεδομένα, ώστε οι ερωτήσεις και απαντήσεις σχετικά με αυτά, να είναι ποιοτικές και έγκυρες. [3,11]

Η τεχνική αυτή, χωρίζεται σε δύο βασικά μέρη, το Indexing και το Retrieval and Generation. Το Indexing αφορά την εισαγωγή και οργάνωση των δεδομένων σε μια Βάση Δεδομένων, με τέτοιο τρόπο ώστε, να είναι προσβάσιμα γρήγορα και αποτελεσματικά. Με άλλα λόγια, επιτρέπεται στο μοντέλο να κάνει γρήγορη ανάκτηση πληροφοριών από τη βάση δεδομένων ώστε να εξάγει τις πληροφορίες που πρέπει.
Ας δούμε πιο συγκεκριμένα ποια είναι τα βήματα της διαδικασίας αυτής και θα τα αναλύσουμε στη συνέχεια του κεφαλαίου:

i. Αρχικά, πρέπει να φορτώσουμε τα δεδομένα μας (Load). Αυτό επιτυγχάνεται με τη χρήση των DocumentLoaders, εργαλείων που επιτρέπουν την εισαγωγή δεδομένων από διάφορες πηγές, όπως αρχεία κειμένου, βάσεις δεδομένων ή ακόμα και διαδικτυακές πηγές.
ii. Μετά, τα documents πρέπει να χωριστούν σε τμήματα (Splits). Τα εργαλεία Text splitters χρησιμοποιούνται για να διασπάσουν τα μεγάλα έγγραφα στα splits. Αυτό είναι χρήσιμο τόσο για το indexing των δεδομένων όσο και για την προώθησή τους στο μοντέλο, καθώς τα μεγάλα τμήματα κειμένου είναι δυσκολότερο να αναζητηθούν.
iii. Μετατροπή του κειμένου σε διανύσματα, (embeddings: αναπαράσταση του νοήματος των λέξεων με νούμερα) επιτρέποντας την ταχύτερη και πιο αποτελεσματική αναζήτηση μετά την αποθήκευση τους.
iv. Χρειαζόμαστε κάπου να αποθηκεύσουμε και να ευρετηριάσουμε τα διαιρεμένα τμήματα, ώστε να μπορούν να αναζητηθούν αργότερα. Αυτό γίνεται συνήθως με τη χρήση ενός VectorStore. Το VectorStore είναι μια βάση δεδομένων που επιτρέπει την αποθήκευση των διανυσμάτων (embeddings) των τμημάτων κειμένου.



Σχήμα 4.1: Διαδικασία του Indexing

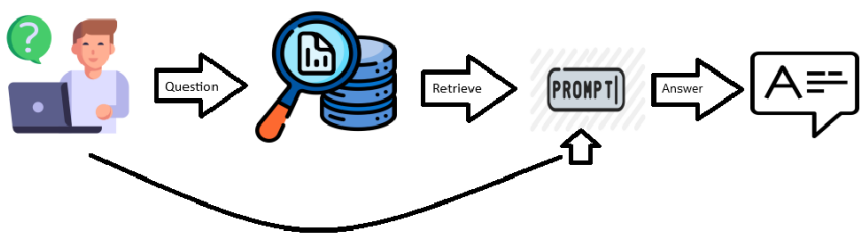
Το δεύτερο μέρος, Retrieval and Generation, αφορά την αναζήτηση και ανάκτηση σχετικών με το query του χρήστη δεδομένων, από τα δεδομένα που αποθηκεύτηκαν κατά τη φάση του indexing. Μετά τα δεδομένα αυτά δίνονται στο LLM και αυτό θα

αρχίσει την διαδικασία παραγωγής της απάντησής του, βασισμένο στο ερώτημα του χρήστη και εμπλουτισμένο με τα σχετικά δεδομένα που του δόθηκαν. Να αναφέρουμε τα βήματα της διαδικασίας αυτής και θα ακολουθήσει περεταίρω ανάλυσή τους στη συνέχεια:

i. Αρχικά ο χρήστης θα δώσει μια είσοδο, συνήθως ένα prompt ή κάποιο query (ερώτημα).

ii. Έπειτα γίνεται ανάκτηση δεδομένων (retrieval): Το μοντέλο χρησιμοποιώντας τον αλγόριθμο ανάκτησης (Retriever), το σύστημα σαρώνει την προκαθορισμένη βάση δεδομένων για να βρει και να ανακτήσει τα πιο σχετικά κομμάτια πληροφοριών (splits). Αυτά τα κομμάτια δεδομένων επιλέγονται βάσει της σχετικότητάς τους με την αρχική ερώτηση του χρήστη. Ο Retriever μπορεί να είναι είτε ένας απλός αλγόριθμος αναζήτησης βασισμένος σε κείμενο, ή ένα πιο προηγμένο μοντέλο μηχανικής μάθησης που χρησιμοποιεί embeddings για να κατανοήσει την έννοια και την πληροφορία που κρύβεται πίσω από τις λέξεις της ερώτησης.

iii. Αφού ολοκληρωθεί η ανάκτηση των απαραίτητων δεδομένων, γίνεται ενσωμάτωσή τους μαζί με το prompt που θα δόθηκε από τον χρήστη και το μοντέλο μεταβαίνει στο δεύτερο στάδιο, την παραγωγή (generation). Εδώ, ένα μοντέλο παραγωγής κειμένου (π.χ., ChatModel ή Large Language Model όπως το GPT) λαμβάνει ως είσοδο την αρχική ερώτηση του χρήστη μαζί με τα ανακτημένα δεδομένα. Στη συνέχεια, το μοντέλο παράγει μια απάντηση, αφού επεξεργαστεί τις πληροφορίες που ανακτήθηκαν. Το μοντέλο χρησιμοποιεί τεχνικές βαθιάς μάθησης για να κατασκευάσει καλά δομημένη απάντηση που ανταποκρίνεται στο αίτημα του χρήστη.



Σχήμα 4.2: Διαδικασία του Retrieval and Generation

4.2 Εργαλεία και τεχνολογίες που χρησιμοποιήθηκαν

4.2.1 LangChain

Langchain είναι ένα framework το οποίο έχει φτιαχτεί για να βοηθά τους προγραμματιστές στην ανάπτυξη εφαρμογών οι οποίες τροφοδοτούνται και δουλεύουν με LLMs, δηλαδή τα χρησιμοποιούν ως βάση για επεξεργασία και παραγωγή φυσικής γλώσσας, καθώς και για τη λήψη αποφάσεων ή την εκτέλεση εργασιών με βάση κάποιο κείμενο. Πιο συγκεκριμένα μπορούν να αναπτυχθούν εφαρμογές οι οποίες :

- ✓ Είναι contex-aware, δηλαδή επικεντρώνονται στη συλλογή, επεξεργασία και χρήση πληροφοριών οι οποίες δίνονται στο μοντέλο ως πηγή δεδομένων (πλαίσιο) που θα κατευθύνει και θα προσαρμόσει τις ενέργειές του. Πιο συγκεκριμένα, το μοντέλο θα ενσωματώσει τις πληροφορίες που μπορεί να είναι prompt instructions, παραδείγματα (few shot examples), εξειδικευμένο περιεχόμενο (Retrieval-Augmented Generation) κλπ. , με σκοπό την βελτίωση της ποιότητας και σχετικότητας των απαντήσεών του.
- ✓ Έιναι reasoned, δηλαδή εκτός από απλή επεξεργασία πληροφοριών, χρησιμοποιούν το LLM για να σκεφτούν (λογικά) και να λάβουν αποφάσεις βάση του δοσμένου αυτού πλαισίου. Πιο συγκεκριμένα, είναι σε θέση να προτείνουν ενέργειες και λύσεις, να βγάζουν συμπεράσματα και να καθοδηγούν τους χρήστες.

4.2.2 Hugging Face

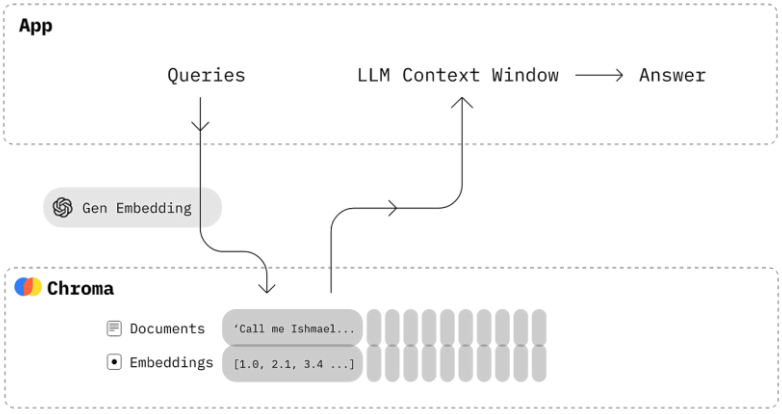
Το Hugging Face είναι μια πλατφόρμα επικεντρωμένη στη μηχανική μάθηση και βοηθά τους χρήστες στη δημιουργία, ανάπτυξη και εκπαίδευση μοντέλων μηχανικής μάθησης παρέχοντάς τους διάφορα εργαλεία και πόρους όπως για παράδειγμα διάφορα trained μοντέλα ή datasets.

4.2.3 ChromaDB

Το Chroma DB είναι ένα open-source σύστημα αποθήκευσης διανυσμάτων (βάση δεδομένων διανυσμάτων) που έχει σχεδιαστεί για την αποθήκευση και ανάκτηση vector embeddings. Η κύρια λειτουργία του είναι να αποθηκεύει embeddings με συνδεδεμένα metadatas για μεταγενέστερη χρήση από τα LLMS. Επιπλέον, μπορεί να λειτουργήσει ως βάση για σημασιολογικούς μηχανισμούς αναζήτησης (semantic search) που

λειτουργούν με βάση τα δεδομένα κειμένων. Ένα Vector Database προσφέρει μια ιδανική λύση για τη διαχείριση μεγάλων όγκων unstructured and semi-structured δεδομένων, αφού έχουν σχεδιαστεί για να διαχειρίζονται τεράστιες ποσότητες δεδομένων, και χρησιμοποιούν προηγμένους αλγόριθμους indexing για να επιτρέψουν την γρήγορη ανάκτηση σχετικών διανυσμάτων στον διανυσματικό χώρο. [4,13]

Στις οδηγίες της OpenAI παρέχονται αρκετές Vector Databases που μπορούν να χρησιμοποιηθούν σε συνδυασμό με το ChatGPT API, όπως AnalyticDB, MongoDB, Pinecone κλπ. Έτσι επιλέγηκε μια από αυτές, η οποία είναι δωρεάν και εύκολη στη χρήση, παρέχει υποστήριξη για LangChain και έχει αρκετά features: για queries, filtering, εκτιμήσεις πυκνότητας και πολλά άλλα.



Σχήμα 4.3: Chroma DB explained

4.2.4 Gradio

Το Gradio είναι ένα πακέτο Python ανοιχτού κώδικα που μας επιτρέπει να δημιουργήσουμε γρήγορα ένα demo ή μια ιστοσελίδα εφαρμογής για μοντέλα μηχανικής μάθησης, APIs ή οποιαδήποτε συνάρτηση Python. Μπορούμε στη συνέχεια χρησιμοποιώντας τα ενσωματωμένα features κοινοποίησης του Gradio, να μοιραστούμε το website με ένα link, σε μερικά δευτερόλεπτα. Μας βοηθά να φτιάξουμε βασικά το user interface μας, από το οποίο τα inputs και οι επιλογές του χρήστη, θα είναι αυτά που θα δοθούν με κάποιο τρόπο στο μοντέλο μηχανικής μάθησης, και αφού γίνει η παραγωγή της απάντησής του, εμφανίζεται επίσης στο user interface.

4.2.5 Google Colab

Όλος ο κώδικας και οι δοκιμές έχουν γίνει στο Google Colab. Το Colab είναι μια υπηρεσία φιλοξενίας του Jupyter Notebook (editor για computational notebooks: έγγραφα που συνδυάζουν κώδικα, περιγραφές σε απλή γλώσσα, δεδομένα, πλούσιες οπτικοποιήσεις όπως μοντέλα 3D, διαγράμματα, γραφήματα και εικόνες, καθώς και διαδραστικά εργαλεία ελέγχου). Δεν απαιτεί καμία προετοιμασία για να χρησιμοποιηθεί και παρέχει δωρεάν πρόσβαση σε υπολογιστικούς πόρους, συμπεριλαμβανομένων των GPUs και TPUs. Το Colab είναι ιδιαίτερα κατάλληλο για τη μηχανική μάθηση, την επιστήμη δεδομένων και την εκπαίδευση. Παρέχει προεγκατεστημένες βιβλιοθήκες για γλώσσες προγραμματισμού όπως Python, καθώς και εργαλεία μηχανικής μάθησης, επιτρέποντας στους χρήστες να εστιάζουν στην ανάπτυξη κώδικα και μοντέλων αντί για τη διαχείριση των εξαρτήσεων**.** Επίσης, συνδέεται άμεσα με το Google Drive, παρέχοντάς μας τη δυνατότητα να χρησιμοποιούμε αρχεία από το drive κατευθίαν, εξοικονομώντας χώρο και χρόνο, αλλά και καλύτερη οργάνωση.

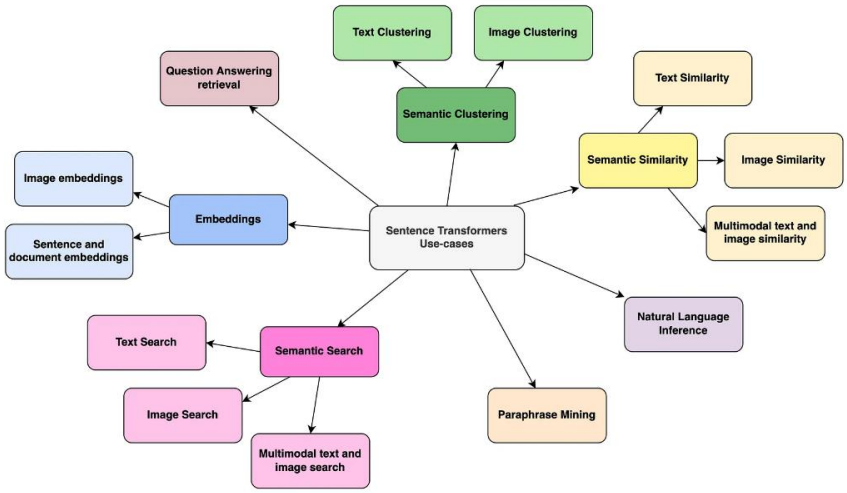
4.2.6 Sentence Transformers

Τα SentenceTransformers είναι framework στη Python για κορυφαίας τεχνολογίας embeddings προτάσεων, κειμένων και εικόνων. Οι ενσωματώσεις μπορούν να υπολογιστούν για πάνω από 100 γλώσσες και μπορούν να χρησιμοποιηθούν εύκολα για κοινές εργασίες όπως η σημασιολογική ομοιότητα κειμένου, η σημασιολογική αναζήτηση και η δημιουργία παραφράσεων.

Ενώ το μοντέλο των Transformers (BERT) αναμφίβολα ξεχωρίζει στις ενσωματώσεις λέξεων, οι Sentence Transformers εξειδικεύονται στην κατανόηση ολόκληρων προτάσεων ή παραγράφων. Είναι σχεδιασμένοι να δημιουργούν πλούσιες ενσωματώσεις προτάσεων, ανοίγοντας πόρτες σε πολλές εφαρμογές που απαιτούν κατανόηση σε επίπεδο προτάσεων. Οι Sentence Transformers μπορούν να αναλύσουν και να αναγνωρίσουν μεικτά συναίσθημα μέσα σε μια περίπλοκη πρόταση, επιδεικνύοντας το βάθος της κατανόησής τους.

Ενώ το BERT διαπρέπει σε εργασίες όπως ανάλυση συναισθήματος, απάντηση σε ερωτήσεις και αναγνώριση ονομαστικών οντοτήτων, όπου η λεπτομέρεια σε επίπεδο

λέξεων είναι κρίσιμη, οι Sentence Transformers είναι η προτιμώμενη επιλογή για αξιολογήσεις σημασιολογικής ομοιότητας, αντιστοίχιση κειμένων και ανάκτηση εγγράφων, όπου η σύλληψη της ουσίας ολόκληρων προτάσεων ή παραγράφων είναι ουσιώδης. [1]



Σχήμα 4.4: Sentence Transformers use cases

4.3 Προ-Επεξεργασία Δεδομένων

Πριν ξεκινήσει η ανάπτυξη του συστήματος, έγινε έρευνα για την εύρεση υλικού. Όλα τα βιβλία της Δημοτικής Εκπαίδευσης βρίσκονται στην σελίδα του Υπουργείου Παιδείας, στο εκπαιδευτικό υλικό. Για το σύστημά μας, χρησιμοποιήθηκαν τα βιβλία της Ιστορίας της Δ’ και Ε’ τάξης, των Θρησκευτικών (Α’ και Β’ μέρος) της Δ’ , Ε’ και ΣΤ’ τάξης, του Σχεδιασμού & Τεχνολογίας Ε’ και ΣΤ’ τάξης, και για το μάθημα των Ελληνικών χρησιμοποιήθηκαν οι Δραστηριότητες Κειμένων Δ’ , Ε’ και ΣΤ’ τάξης. Τα βιβλία ήταν όλα σε μορφή .pdf. Παρουσιάζονται πιο αναλυτικά ο αριθμός και τα μεγέθη των PDFs του κάθε μαθήματος πριν κάποια συγκεκριμένη προ-επεξεργασία, μόνο απλή μετατροπή από .pdf σε .txt files:

6. Ιστορία: 2 files: 37801 words & 420 KB, 43245 words & 510 KB
7. Θρησκευτικά: 6 files: 20736 words & 261 KB, 17909 words & 227 KB, 31342 words & 387 KB, 22983 words & 287 KB, 36588 words & 450 KB, 26410 words & 334 KB

8. Σχεδιασμός και Τεχνολογία: 2 files : 14204 words & 190 KB , 14000 words & 175 KB
9. Γλώσσα: 3 files: 6983 words & 141 KB, 8639 words & 170 KB, 8572 words & 186 KB

Αφού τα δεδομένα είναι σε PDF σε μορφή κανονικού βιβλίου, συμπεριλαμβάνουν εκτός από το κείμενο, εικόνες, σχεδιαγράμματα, παραρτήματα, και άλλα noisy data (άχρηστες πληροφορίες) όπως π.χ. πηγές, συντάκτες των βιβλίων, αρίθμηση σελίδων, επαναλήψεις τίτλων κεφαλαίων κλπ. Συστήνεται, σε τέτοιου είδους applications να γίνεται ένα data pre-processing, δηλαδή μια επεξεργασία πριν εισαχθούν τα δεδομένα στο σύστημα, ώστε να αφαιρεθούν όλες οι άχρηστες πληροφορίες ή επαναλήψεις και να παραμείνουν μόνο οι χρήσιμες πληροφορίες. Αυτό θα βοηθήσει αργότερα το σύστημα στο να έχει καλύτερα αποτελέσματα, αφού βελτιώνεται η ποιότητα των δεδομένων εκπαίδευσής του, αλλά μειώνεται και το μέγεθός τους, άρα θα είναι πιο γρήγορο και αποδοτικό. (Γρηγορότερη αποθήκευση embeddings, γρηγορότερο ψάξιμο => γρηγορότερα αποτελέσματα). Αρχικά, διαβάζουμε με χρήση βιβλιοθήκης της Python τα pdfs και τα μετατρέπουμε σε text files. Συγκεκριμένα έγινε χρήση των PyPDF2 και pdfplumber για την ανάγνωσή τους. Για εύρεση των άχρηστων δεδομένων χρησιμοποιήθηκε αναζήτηση με regular expressions (regex) (‘re’ Python library) [16] , κατά την οποία γίνεται αναζήτηση κάποιων μοτίβων μέσα στα κείμενα, και όταν βρεθούν διαγράφονται. Βλέπουμε ένα παράδειγμα στον πιο κάτω κώδικα, στον οποίο φτιάχνουμε αρχικά ένα μοτίβο προς αναζήτηση το οποίο θα συμπεριλαμβάνει τις λέξεις ‘Εικ.’ που ακολουθείται από κάποιο ψηφίο (0-9) ή ‘Πηγή’ που ίσως ακολουθείται από ‘:’ και μπορεί επίσης να ακολουθούνται από οποιοδήποτε χαρακτήρα με 0 ή περισσότερες εμφανίσεις. Στη συνέχεια τα διαγράφουμε από το κείμενο:

```
pattern = re.compile(r'Εικ\. \d+|Πηγή:?.*|πηγή:?.*', re.IGNORECASE)

# Process each chapter text to remove the specified pattern
for i in range(len(chapter_texts)):
    chapter_texts[i] = re.sub(pattern, '', chapter_texts[i])
```

Σχήμα 4.4: Παράδειγμα κώδικα χρήσης του re library

Να αναφέρω ότι χρειάστηκε και αρκετή επεξεργασία να γίνει manually.

Type	Special Char/Seq	Use case	Example	Matches
character	.	match any single character (except line break)	a.b	axb, a$b, abb, a.b, etc
	\d	match one digit character	\d	0, 1, 2, 3, 4, 5, etc
	\w	match one alphanumeric (including _) character	\w	a, b, c, etc
	\s	match one whitespace, tab, or new line	a\sb\c	a b c
	\D	match one non-digit character	\D	A, B, C, etc
	\W	match one non-alphanumeric (including _) character	\W	0, 1, 2, 3, etc
	\S	match one non-whitespace, tab, or newline	\S	A, B, C, etc
class	[...]	create a class of matching characters	[a-z]	all lower case alphabets
	[^...]	match the complement of a set	[^aeiou]	any non-vowel
			^[^a-z]*$	any line containing no lower case
occurrence indicators	*	match zero or more occurrences of the previous pattern	a*b	b, ab, aab, aaab, aaaab, etc
	+	match one or more occurrences of the previous pattern	a+b	ab, aab, aaab, aaaab, etc
	?	match zero or one occurrence of the previous pattern	a?b	b, ab
	[...]*	match repeated sets	[aeiou]*	any number of vowels
	{m,n}	match m to n repeats of the preceding regular expression (the default value for m is 0 and ∞ for n)	[a-z]{2,10}	sequences of 2 to 10 lower case
			^[aeiou]{2,}	at least 2 vowels in the start of line
position anchors	^	start-of-line	^A	any line beginning with A
	$	end-of-line	$A	any line ending with A
			^[0-9]$	any numeric string
	\b	boundary of word	\bab\b	only the word ab
or	|	the logical "or" operator	a|b	either a or b
rules	use (r1) allows the *, +, ? matches to apply to the entire regular expression		(ab)+	at least one repetition of ab
	adding the "?" after the qualifiers makes it perform the match in non-greedy or minimal fashion			
	use parentheses (...) to create a back reference and \1, \2, ... to retrieve back references in sequential order			
	use lookforward assertion (?=...) to check if ... matches next			

Πίνακας 4.5: Regex syntax και use cases

Ένα ακόμα κρίσιμο σημείο στη διαδικασία της προ-επεξεργασίας δεδομένων, είναι η ομαδοποίηση. Τα περισσότερα από τα βιβλία, αποτελούνται από το κείμενο (θεωρία) και από τις ασκήσεις σε κάθε κεφάλαιο. Έτσι, με χρήση πάλι regular expressions matching αλλά και της βιβλιοθήκης fuzzywuzzy [2] προχώρησα στον διαχωρισμό των κεφαλαίων, ομαδοποιώντας τη θεωρία του κάθε κεφαλαίου, ξεχωριστά από τις ασκήσεις του. Το fuzzywuzzy είναι ένα πολύ εύκολο στη χρήση πακέτο, το οποίο χρησιμοποιεί την απόσταση Levenshtein για να υπολογίσει τις διαφορές μεταξύ ακολουθιών και είναι αρκετά χρήσιμο για την εύρεση ακολουθιών σε προτάσεις οι οποίες μπορεί να μην είναι ακριβώς οι ίδιες, αλλά παρόμοιες. Υπολογίζει βασικά ένα score ομοιότητας μεταξύ μιας πρότασης και μιας άλλης και αν το score αυτό είναι αρκετά μεγάλο μπορούμε να προβούμε σε κάποια πράξη. Βλέπουμε ένα παράδειγμα στον πιο κάτω κώδικα:

```
# Check if we're still matching titles
if current_title_index < len(chapter_titles):
    # Use fuzzy matching for the current line against the current title
    score = fuzz.partial_ratio(line, chapter_titles[current_title_index])

    # If a new title is found, move to next chapter
    if score >= 90:
        current_title_index += 1
        continue  # Skip adding title line to content
```

Σχήμα 4.6: Παράδειγμα κώδικα χρήσης fuzzywuzzy

Τώρα έχουμε τα Documents μας και είμαστε έτοιμοι να προχωρίσουμε στο επόμενο βήμα.

4.4 Indexing

4.4.1 Split

Για το βήμα αυτό, θα γίνει χρήση του LangChain. Το LangChain παρέχει διάφορους τρόπους (text Splitters) για να χωρίσουμε τα Documents μας σε μικρότερα chunks. Στη περίπτωσή μας χρησιμοποιήθηκε ο RecursiveCharacterTextSplitter, ο οποίος κάνει split αναδρομικά και προσπαθεί να κρατά σχετικά κομμάτια κειμένου κοντά το ένα στο άλλο. Αυτός είναι και ο προτεινόμενος splitter από το LangChain. Σε γενικές γραμμές, ένας διαχωριστής κειμένου λειτουργεί ως εξής:

1. Αρχικά, το κείμενο διαχωρίζεται σε μικρά, σημασιολογικά πλήρη τμήματα, τα οποία συνήθως είναι προτάσεις.
2. Στη συνέχεια, αυτά τα μικρά τμήματα αρχίζουν να συνδυάζονται σε μεγαλύτερα μπλοκ μέχρι να φτάσουν σε ένα συγκεκριμένο μέγεθος, το οποίο καθορίζεται από πριν.
3. Όταν φτάσει σε αυτό το μέγεθος, το συγκεκριμένο μπλοκ γίνεται ένα ανεξάρτητο κομμάτι κειμένου. Έπειτα, ξεκινά η δημιουργία ενός νέου μπλοκ κειμένου με κάποια επικάλυψη (overlap), προκειμένου να διατηρηθεί η συνέχεια και η συνάφεια μεταξύ των τμημάτων.

Στη περίπτωση μας, το chunk size έχει προκαθοριστεί σε 1000 χαρακτήρες για τα κείμενα και στους 500 για τα παραδείγματα, ενώ το overlap στους 200 και 100 αντίστοιχα.

Ο λόγος που χρειάζεται να “κόψουμε” τα κείμενα, είναι διότι τα LLMs, έχουν ένα μέγιστο όριο tokens που μπορούν να επεξεργαστούν κάθε φορά, αυτό ονομάζεται context window. (ChatGPT 8k/32k.128k). Εκτός αυτού, το να διαχωρίσουμε τα κείμενα σε συγκεκριμένα τμήματα, θα μας βοηθήσει αργότερα να παράξουμε στοχευμένες απαντήσεις σε συγκεκριμένα ερωτήματα, όπως για παράδειγμα, ερωτήσεις που θα αφορούν κάποια ενότητα, ή κεφάλαιο κλπ.

Παράλληλα με το splitting, μπαίνουν τα metadata. Σε κάθε chunk, σύμφωνα με το μάθημα από το οποίο προέρχεται, την τάξη, το κεφάλαιο και το είδος του (θεωρία ή

ασκήσεις), εισάγονται κάποιες επιπρόσθετες πληροφορίες, ώστε να μπορούμε να αναζητούμε με ευκολία, αυτό που ψάχνουμε κάθε φορά. Τα metadatas είναι βασικά σε μορφή JSON, είναι δηλαδή σε key-value pairs: {"chapter": chapter_number, "part":"A", "subject":"thriskeftika", "taksi":"D", "type": "examples"}.

Split Chapters to Chunks

```
import string
documents = []
metadatas = []

from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)

thrisk_documents = text_splitter.create_documents(thriskeftika, metadatas=thriskeftika_metadata)
thrisk_documents = text_splitter.split_documents(thrisk_documents)

thrisk_documents = [d for d in thrisk_documents if not all(c in string.punctuation for c in d.page_content)]

thriskeftika_metadatas = []
for i in range (0, len(thrisk_documents)):
  thriskeftika_metadatas.append(thrisk_documents[i].metadata)
  documents.append(thrisk_documents[i])
  metadatas.append(thriskeftika_metadatas[i])
```

Σχήμα 4.7: Κώδικας για split και πρόσθεση metadatas στα chunks

4.4.2 Embed

Φτάνουμε σε ένα από τα σημαντικότερα σημεία της ανάπτυξης του συστήματος. Τα embeddings. Έχει χρησιμοποιηθεί μέσω του Hugging Face, ένα μοντέλο sentence transformers. Το μοντέλο που χρησιμοποιήθηκε είναι το “sentence-transformers/paraphrase-multilingual-mpnet-base-v2” , το οποίο μετατρέπει προτάσεις και παραγράφους σε ένα 768 dimensional dense vector space και μπορεί να χρησιμοποιηθεί για εργασίες όπως η ομαδοποίηση ή η σημασιολογική αναζήτηση. Το μοντέλο αυτό επιλέκτηκε αφού είναι multilingual και δουλεύει για την Ελληνική γλώσσα, έχει δοκιμαστεί και συγκρηθεί με άλλα μοντέλα, και είχε λίγο καλύτερα αποτελέσματα από ένα μοντέλο που έχει εκπαιδευτεί για την Ελληνική γλώσσα (“lighteternal/stsb-xlm-r-greek-transfer”). Το μοντέλο είναι επίσης και αρκετά δημοφιλές, αφού τα downloads του τον τελευταίο μήνα βρίσκονται στα 735,690. Βλέπουμε ένα παράδειγμα embedding vector που δημιουργήθηκε για ένα Document:

```
embeddings = collection.get(ids=['0'], include=['documents','embeddings'])
print(embeddings)
```

```
{'documents': ['Τραύμα της ζωής: Πόλεμος; Ο πόλεμος μέσα από έναν πίνακα Το 1936 ο στρατηγός Φράνκο κάνει
πραξικόπημα, για να ανατρέψει τη δημοκρατική κυβέρνηση της Ισπανίας. Στον εμφύλιο πόλεμο που ακολουθεί ο Χίτλερ
έρχεται να ενισχύσει τον Φράνκο και τα αεροπλάνα του βομβαρδίζουν αδιάκριτα περιοχές της Ισπανίας. 'Ενα μικρό
χωριό -η Γκερνίκα- γκρεμίζεται βαθύ τον πόνο από τις βόμβες τους, καθώς σκοτώνονται πολλοί άμαχοι, γυναίκες και παιδιά.
Το 1937 η ισπανική κυβέρνηση παραγγέλλει στον Πάμπλο Πικάσο ένα έργο κατάλληλο να αντιπροσωπεύσει την Ισπανία στη
διεθνή έκθεση του Παρισιού. Ο Πικάσο ζωγραφίζει την «ΓΚΕΡΝΙΚΑ» επηρεασμένος από την καταστροφή του μικρού χωριού
από τη χιτλερική αεροπορία. Είναι ένα έργο με ισχυρό αντιπολεμικό μήνυμα, που ακούγεται ηχηρά σε όλο τον κόσμο. Η
Δόμνα ζει το ξέσπασμα του πολέμου Μόλις είχε κλείσει τα έξι η Δόμνα, κι ο πατέρας τής αγόρασε κτόλας μια τσάντα,
βιβλία και τετράδια. Σε λίγους μήνες θα πήγαινε στο δημοτικό της Αγίας Φωτεινής να μάθει τα πρώτα της γράμματα.'],
'embeddings': [[0.057719599455595016, 0.11449652910232544, -0.0155205512419343, 0.026406029239296913,
0.01630452647805214, 0.011450617574155533, 0.07080158591270447, 0.004926114343106747, 0.021770654246211052,
0.10157956928014755, 0.08602334558963776,
0.0722145214676857, -0.03109113872051239, -0.24881741404533386, -0.01109382789582014, -0.10913461446762085
, -0.03695668280124664, 0.09103039652109146, -0.12415380775928497,
0.03188488632440567, -0.024333182722330093, 0.0062345899641513824, 0.10576840490102768,
0.04397867992520332, 0.0032336153090000153,
0.08495056629180908, -0.02485022321343422, -0.0162569060921669,
0.12792488932609558, -0.019311875104904175, 0.0785495787858963, -0.037752747535705566,
0.029267312958836555, 0.05863192304968834, 0.03972284495830536,
0.031776729971170425, -0.08926629275083542, 0.03035927563905716,
0.006762140430510044, -0.047316938638687134, 0.22807274758815765,
0.017043618485331535, -0.1327110230922699, -0.021879209205508232, -0.15637081861495972, .....
```

Σχήμα 4.8: Παρουσίαση embedding vector ενός document

4.4.3 Store

Το τελευταίο βήμα είναι η αποθήκευση των documents, των metadatas τους και των embeddings τους στο Vector Database. Η αποθήεύση γίνεται παράλληλα με την δημιουργία των embeddings ως εξής:

```
import chromadb

from chromadb.utils import embedding_functions

sentence_transformer_ef = embedding_functions.SentenceTransformerEmbeddingFunction(model_name='sentence-transformers/paraphrase-multilingual-mpnet-base-v2')
chroma_client = chromadb.Client()
collection = chroma_client.create_collection(name="my_collection", embedding_function=sentence_transformer_ef, metadata={"hnsw:space": "cosine"})
```

Σχήμα 4.9: Κώδικας για καθορισμός embedding function στη βάση

```
collection.add(
    documents = [docs.page_content for docs in documents],
    metadatas = metadatas,
    ids       = [str(i) for i in range(len(documents))]
)
```

Σχήμα 4.10: Κώδικας για πρόσθεση των δεδομένων στη βάση

4.4 Retrieval

Με βάση της επιλογές που κάνει ο χρήστης, γίνονται retrieve τα απαραίτητα δεδομένα από τη βάση. Για τα μαθήματα της Ιστορίας, των Θρησκευτικών και της Γεωγραφίας, η αναζήτηση μέσα στην βάση, γίνεται με βάση τα metadatas. Για παράδειγμα όταν επιλεγεί το μάθημα της Ιστορίας, Ε΄ τάξης, τα κεφάλαια 1,2,3 θα πάρουμε από τη βάση όλα τα documents που έχουν ως metadata: lesson = ‘istoria’ , grade = ‘E’, chapters in [1,2,3] (δηλαδή, που ισούται με έναν από τους τρεις αριθμούς που βρίσκονται στη λίστα). Τα documents αυτά, αποτελούν στην ουσία την ερώτηση (query) του χρήστη. Αυτά, στη συνέχεια θα ενσωματωθούν μέσα στο prompt το οποίο θα σταλεί στο LLM.

Για το μάθημα των Ελληνικών, πραγματοποιείται similarity search. Ο χρήστης καλείται να εισάγει κάποια keywords, που αντιπροσωπεύουν το θέμα των ασκήσεων που θέλει. Αφού επιλέξουν και την τάξη, γίνεται αναζήτηση από το database, όπου γίνονται τελικά retrieved παραδείγματα ασκήσεων για την συγκεκριμένη τάξη, οι οποίες σχετίζονται με τα keywords του χρήστη.

Ας εξηγήσουμε τι είναι το similarity search και με ποιους τρόπους μπορεί να γίνει χρησιμοποιώντας το ChromDB. [9] Οι συναρτήσεις απόστασης βοηθούν στον υπολογισμό της διαφοράς (απόστασης) μεταξύ δύο vector embeddings, όπου με αυτό τον τρόπο κατά το similarity search, τα embeddings με την μικρότερη απόσταση από το embedding του query (αυτού που ψάχνουμε), επιστρέφονται. Το ChromaDB υποστηρίζει τις ακόλουθες συναρτήσεις απόστασης:

- Cosine - Χρήσιμη για την ομοιότητα κειμένου: Μετρά τη γωνία μεταξύ δύο διανυσμάτων στον διανυσματικό χώρο. Η τιμή της κυμαίνεται από -1 έως 1, όπου:
    - 1 σημαίνει ότι τα διανύσματα είναι απόλυτα ομοιόμορφα.
    - 0 σημαίνει ότι τα διανύσματα είναι ορθογώνια μεταξύ τους (καμία ομοιότητα).
    - -1 σημαίνει ότι τα διανύσματα είναι διαμετρικά αντίθετα.

  Ιδιότητες:
    - Ανεξάρτητη από το μέγεθος: Δεν επηρεάζεται από το μέγεθος των διανυσμάτων, μόνο από τη γωνία τους.
    - Κατάλληλη για υψηλές διαστάσεις: Συχνά χρησιμοποιείται στην επεξεργασία φυσικής γλώσσας όπου οι ενσωματώσεις είναι υψηλών διαστάσεων.
- Euclidean (L2) - Χρήσιμη για την ομοιότητα κειμένου, πιο ευαίσθητη στον θόρυβο από το συνημίτονο: Μετρά την πραγματική απόσταση στο χώρο μεταξύ δύο σημείων (διανυσμάτων). Υπολογίζεται ως η τετραγωνική ρίζα του αθροίσματος των τετραγώνων των διαφορών των συντεταγμένων τους.
  Ιδιότητες:
    - Ευαισθησία στον θόρυβο: Πιο ευαίσθητη σε μικρές διαφορές στα δεδομένα, κάτι που μπορεί να επηρεάσει την ακρίβεια σε δεδομένα με θόρυβο.

- Εξαρτημένη από το μέγεθος: Επηρεάζεται από το μέγεθος των διανυσμάτων, πράγμα που σημαίνει ότι διανύσματα με μεγάλη διαφορά στο μέγεθος μπορεί να έχουν μεγαλύτερη απόσταση ακόμα και αν είναι παρόμοια.

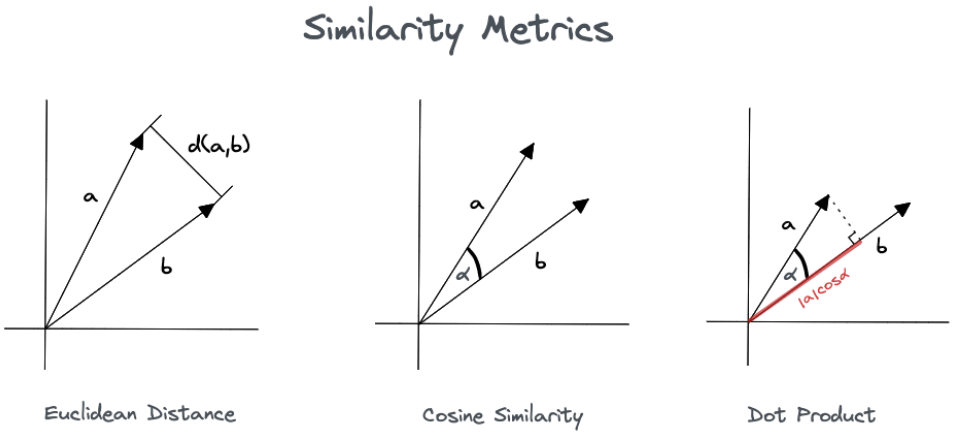
- Inner Product (IP) – Χρήσιμη για recommender systems: Μετρά το προϊόν δύο διανυσμάτων χωρίς να λαμβάνει υπόψη τη γωνία ή την απόσταση στον χώρο. Είναι ιδιαίτερα χρήσιμο στα συστήματα προτάσεων, όπου το ζητούμενο είναι να βρεθούν διανύσματα (προϊόντα, χρήστες κ.λπ.) που έχουν υψηλή τιμή προϊόντος εσωτερικού γινομένου.

  Ιδιότητες:

  - Πολυπλοκότητα: Απλό στην υπολογιστική του εφαρμογή.
  - Χρήση στη σύσταση: Συνήθως χρησιμοποιείται σε συστήματα συστάσεων για την εύρεση αντικειμένων (π.χ., προϊόντων) που είναι πιο σχετικά με τους χρήστες.

Distance	parameter	Equation
Squared L2	`l2`	$d = \sum (A_i - B_i)^2$
Inner product	`ip`	$d = 1.0 - \sum (A_i \times B_i)$
Cosine similarity	`cosine`	$d = 1.0 - \frac{\sum (A_i \times B_i)}{\sqrt{\sum (A_i^2)} \cdot \sqrt{\sum (B_i^2)}}$

Σχήμα 4.11: ChromaDB Distance functions και εξισώσεις τους



Σχήμα 4.11: Γραφικά τα distance functions

Να τονίσουμε ότι εκτελόντας similarity search στα documents μας, δεν παίρνουμε τα πιο ακριβή αποτελέσματα. Αυτό ίσως οφείλεται στα embedding functions, ή στη

μέθοδο αναζήτησης που χρησιμοποιεί το ChromaDB. Έχουν δοκιμαστεί 4 διαφορετικά μοντέλα sentence transformers και θα δούμε μερικά αποτελέσματα που παίρνουμε κατά την αναζήτηση ενός query.

Πρώτο παράδειγμα με query: 'ενεργητική, παθητική φωνή'.

"dimitriz/st-greek-media-bert-base-uncased" :
['προτάσεις, χρησιμοποιώντας τις παρακάτω λέξεις του κειμένου. Να ζητήσεις βοήθεια και από το λεξικό σου! εμπνευστής: αθέατος: αφανής: προδιαγεγραμμένη: εγχείρημα: 8. Τα ρήματα του κειμένου βρίσκονται, κυρίως, σε χρόνο ενεστώτα, αόριστο ή παρατατικό. Να αντιγράψεις έξι προτάσεις από το κείμενο, συμπληρώνοντας τον ακόλουθο πίνακα: Χρόνος: Προτάσεις-Παραδείγματα, Ενεστώτας, Αόριστος, Παρατατικός. 9. Να εξηγήσεις γιατί, σε αρκετά σημεία του κειμένου, χρησιμοποιείται ο ενεστώτας. Τι προσφέρει στον/στην αναγνώστη/αναγνώστρια η χρήση αυτού του χρόνου;']

"lighteternal/stsb-xlm-r-greek-transfer":
['(όταν τονίζονται στη λήγουσα), παροξύτονες (όταν τονίζονται στην παραλήγουσα), προπαροξύτονες (όταν τονίζονται στην προπαραλήγουσα). 14. Αφού γράψεις σε ποια συλλαβή τονίζονται, να ονομάσεις τις παρακάτω λέξεις, όπως το παράδειγμα: δάχτυλο: προπαραλήγουσα - προπαροξύτονη καλλιγραφία: οικογένεια: συλλαβισμός: 15. Τέλος, να εντοπίσεις και να αντιγράψεις, από το κείμενο, δύο λέξεις για κάθε ομάδα. οξύτονες: παροξύτονες: προπαροξύτονες: δάχτυλο: φάρμα: μουστάκια: κορίτσι: Σάββατο: οικογένεια: καλλιγραφία: ελαφάκι: δασκάλα: συλλαβισμός.']

"sentence-transformers/paraphrase-multilingual-mpnet-base-v2" & "sentence-transformers/stsb-xlm-r-multilingual":
['5. Να συμπληρώσεις τα κενά με -μαι ή -με, -ται ή –τε: 6. Να γράψεις τα παρακάτω ρήματα στην κατάλληλη στήλη: (ψάχνω, εκφράζομαι, προστατεύομαι, προστατεύω, βρέχομαι, αγγίζω, κοιμούμαι, μυρίζω, ερευνώ, συνεργάζομαι, τρώγω, γεύομαι, μαθαίνω, ντύνομαι, περπατώ, γυμνάζομαι) ΕΝΕΡΓΗΤΙΚΗ ΦΩΝΗ/ΠΑΘΗΤΙΚΗ ΦΩΝΗ. 7. Να μετατρέψεις την ενεργητική σύνταξη των παρακάτω προτάσεων σε παθητική. Τα παιδιά ανακαλύπτουν τη φύση. Οι ισχυροί άνεμοι έκοψαν τα δέντρα. Το

«Σχολείο του Δάσους» αναπτύσσει τη συνεργασία των παιδιών. 8. Να μετατρέψεις την παθητική σύνταξη των παρακάτω προτάσεων σε ενεργητική. Τα παιδιά συνοδεύονται από ειδικούς εκπαιδευτικούς. Τα δάση κάηκαν από τη φωτιά. Τα φυτά του δάσους μελετήθηκαν από τα παιδιά. Δεν κοιμού_ από τη χαρά μου. Χαίρο_ που θα συναντηθού_ στο «Σχολείο του Δάσους». Το «Σχολείο του Δάσους» βρίσκε_ κοντά στη λίμνη. Θα εί_ και εγώ απέναντι από το βουνό, αφού θα σας περιμένου_ όλους. Από εδώ που κάθο_ φαίνε_ το βουνό. 9. Να συγκρίνεις την ενεργητική με την παθητική']

Σε αυτό το παράδειγμα τα multilingual models φαίνεται να δουλεύουν λίγο καλύτερα. Ας δούμε ακόμα ένα παράδειγμα.

Δεύτερο παράδειγμα με query = ‘Αόριστος , Παρατατικός, Ενεστώτας’.

“dimitriz/st-greek-media-bert-base-uncased” & “lighteternal/stsb-xlm-r-greek-transfer”:
['προτάσεις, χρησιμοποιώντας τις παρακάτω λέξεις του κειμένου. Να ζητήσεις βοήθεια και από το λεξικό σου! εμπνευστής: αθέατος: αφανής: προδιαγεγραμμένη: εγχείρημα: 8. Τα ρήματα του κειμένου βρίσκονται, κυρίως, σε χρόνο ενεστώτα, αόριστο ή παρατατικό. Να αντιγράψεις έξι προτάσεις από το κείμενο, συμπληρώνοντας τον ακόλουθο πίνακα: Χρόνος: Προτάσεις-Παραδείγματα, Ενεστώτας, Αόριστος, Παρατατικός. 9. Να εξηγήσεις γιατί, σε αρκετά σημεία του κειμένου, χρησιμοποιείται ο ενεστώτας. Τι προσφέρει στον/στην αναγνώστη/αναγνώστρια η χρήση αυτού του χρόνου;']

“sentence-transformers/paraphrase-multilingual-mpnet-base-v2”:
['πληθυντικού αριθμού ο άνθρωπος του ανθρώπου των ανθρώπων το κουδούνι το δάχτυλο ο δάσκαλος το σχολείο η κυρία το πρόσωπο η φασαρία 11. Να ξαναγράψεις τις φράσεις, όπως το παράδειγμα: το χρώμα του σπιτιού το χρώμα των σπιτιών το άρωμα του δάσους το τέλος του ποταμού η εποχή της βροχής το νερό του καταρράκτη ο ήχος της βροντής. 12. Στην ιστορία που διάβασες, η τάξη της κυρίας Γουίντερ μαθαίνει να συλλαβίζει λέξεις. Ώρα να θυμηθείς τι ξέρεις από συλλαβισμό. Πρώτα, να χωρίσεις τις παρακάτω λέξεις σε συλλαβές. 13. Στη συνέχεια, ανάλογα με τον αριθμό των συλλαβών

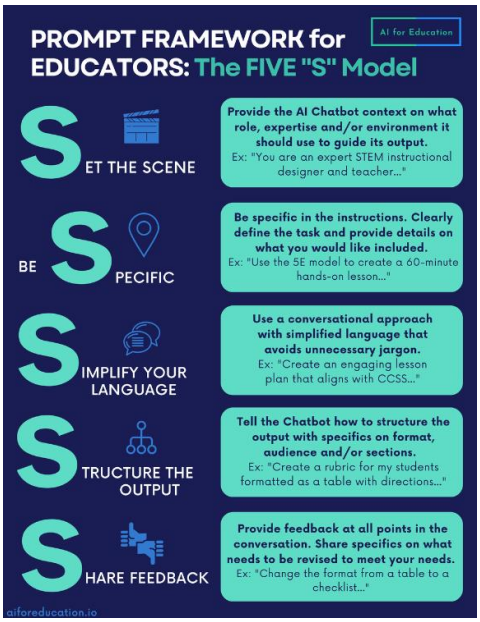
των λέξεων, να εντοπίσεις και να αντιγράψεις, από το κείμενο, τρεις λέξεις για κάθε ομάδα. μονοσύλλαβες : δισύλλαβες : τρισύλλαβες : πολυσύλλαβες : Σημείωση: Οι λέξεις, ανάλογα με τη συλλαβή που τονίζονται, ονομάζονται οξύτονες (όταν τονίζονται στη λήγουσα), παροξύτονες (όταν τονίζονται στην παραλήγουσα), προπαροξύτονες (όταν τονίζονται στην προπαραλήγουσα). 14. Αφού γράψεις σε ποια συλλαβή τονίζονται, να ονομάσεις τις']

"sentence-transformers/stsb-xlm-r-multilingual":
['πάγος: _ , φροντίδα: _ , ραφή: _, βίδα: _, καρφί: _ , σκάμμα: _, ύψος: _, κρύο: _, ελπίδα: _, καθαρίζω, μαθαίνω, σπέρνω, ντρέπομαι, έρχομαι, τρέχω Συγκριτικός Βαθμός ψηλότερη Θετικός Βαθμός σκληρό Υπερθετικός Βαθμός εξυπνότατος Γράφουμε οδηγίες! ΕΠΙΣΤΡΕΦΟΥΜΕ ΣΤΟ ΣΧΟΛΕΙΟ ΚΑΙ ΜΕΝΟΥΜΕ ΥΓΙΕΙΣ! Με βάση τις εικόνες, συμβουλεύουμε τα παιδιά του σχολείου μας πώς μπορούν να προστατευθούν από τον κορωνοϊό ή και άλλους ιούς, χρησιμοποιώντας πρώτο πληθυντικό πρόσωπο, στην οριστική.']]

Στην περίπτωση αυτή τα μοντέλα της Ελληνικής γλώσσας, φαίνεται να επιστρέφουν καλύτερα αποτελέσματα. Γενικά όμως, κανένα από τα πιο πάνω μοντέλα δεν βλέπουμε να δουλεύει αρκετά καλά στην περίπτωσή μας.

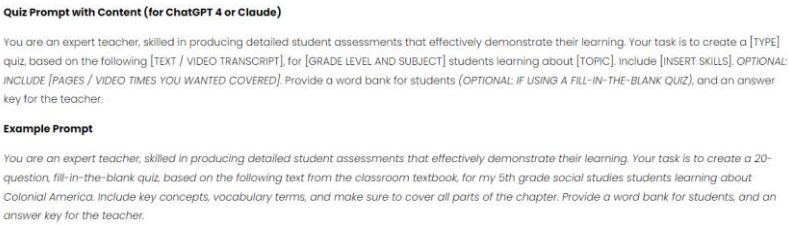
4.5 LLM Prompting

Όπως είδαμε στο κεφάλαιο 2, πρέπει να ακολουθηθούν κάποιες στρατηγικές για καλύτερη χρήση του ChatGPT. Ακολουθήθηκαν επίσης και κάποιες οδηγίες οι οποίες γράφτηκαν για δασκάλους οι οποίοι προσπαθούν να εντάξουν στη καθημερινότητα τους το AI. Ας δούμε μερικές οδηγίες που παρέχουν:



Σχήμα 4.11: Prompt framework για εκπαιδευτικούς

Ας δούμε επίσης και ένα παράδειγμα που παρέχει η ιστοσελίδα αυτή για δημιουργία Quizzes:



Quiz Prompt with Content (for ChatGPT 4 or Claude)

You are an expert teacher, skilled in producing detailed student assessments that effectively demonstrate their learning. Your task is to create a [TYPE] quiz, based on the following [TEXT / VIDEO TRANSCRIPT], for [GRADE LEVEL AND SUBJECT] students learning about [TOPIC]. Include [INSERT SKILLS]. *OPTIONAL: INCLUDE [PAGES / VIDEO TIMES YOU WANTED COVERED].* Provide a word bank for students *(OPTIONAL: IF USING A FILL-IN-THE-BLANK QUIZ)*, and an answer key for the teacher.

Example Prompt

You are an expert teacher, skilled in producing detailed student assessments that effectively demonstrate their learning. Your task is to create a 20-question, fill-in-the-blank quiz, based on the following text from the classroom textbook, for my 5th grade social studies students learning about Colonial America. Include key concepts, vocabulary terms, and make sure to cover all parts of the chapter. Provide a word bank for students, and an answer key for the teacher.

Σχήμα 4.12: Quiz prompt παράδειγμα

Έτσι και στην περίπτωσή μας, ορίζουμε αρχικά μια προσωπικότητα που θέλουμε να πάρει το LLM μας, και του εξηγούμε τι περιμένουμε να κάνει:

"content = «Είσαι έμπειρος δάσκαλος, με χρόνια εμπειρίας στην παραγωγή διαγωνισμάτων για μαθητές Δημοτικού, που αποδεικνύουν αποτελεσματικά τη μάθησή και την κατανόησή τους. Είσαι υπεύθυνος να βοηθήσεις στην κατασκευή ενός διαγωνίσματος για μαθητές {τάξη} Δημοτικού στο μάθημα {μάθημα}»".

Άρα το σύστημα έχει πάρει αυτό το περιεχόμενο, και ως μήνυμα από τον χρήστη, έχουμε το εξής:

"prompt = «Με βάση τα κείμενα που θα σου δοθούν πιο κάτω, να φτιάξεις {αριθμός ασκήσεων} ασκήσεις οι οποίες θα είναι του τύπου που θα σου δοθεί πιο κάτω. Αν οι ασκήσεις είναι περισσότερες από μία, τότε να εστιαστείς σε όλα τα διαφορετικά κεφάλαια (θέματα) που μπορεί να δοθούν στα κείμενα. \n\

#Κείμενα :{κείμενα}\n\

#Τύπος ασκήσεων: {τύποι ερωτήσεων}\n\

#Αριθμός Ερωτήσεων για κάθε τύπο: {αριθμός ερωτήσεων για κάθε τύπο}»".

Με αυτό τον τρόπο, δίνεται ρόλος στο μοντέλο, δίνονται ακριβής οδηγίες, και επίσης, γίνονταν τακτικές διορθώσεις στα prompts, μέχρι το μοντέλο να καταλήξει να εργάζεται ακριβώς όπως το θέλουμε και να δίνει το αποτέλεσμα που επιθυμούμε.

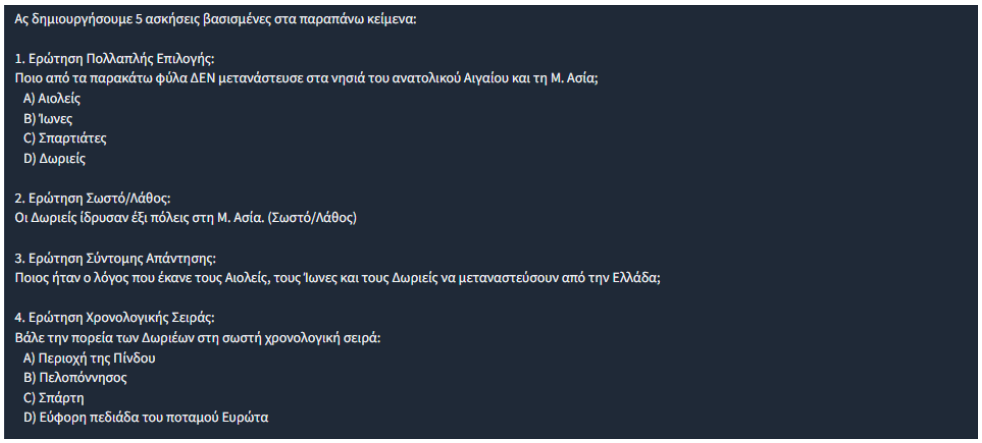
```python
from openai import OpenAI
client = OpenAI()

response = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Who won the world series in 2020?"},
    {"role": "assistant", "content": "The Los Angeles Dodgers won the World Series in 2020.
    {"role": "user", "content": "Where was it played?"}
  ]
)
```

Σχήμα 4.13: OpenAI template για χρήση API

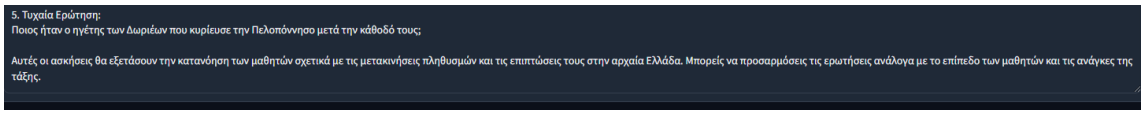
Αξίζει να αναφερθεί ότι για την επιλογή ενός μοντέλου GPT, έχουν δοκιμαστεί 3 μοντέλα, το “gpt-3.5-turbo” , το “gpt-4-turbo”, και το καινούριο μοντέλο της OpenAI “gpt-4o”.

Το “gpt-4o” έχει αρκετά καλές απαντήσεις και είναι επίσης πολύ πιο γρήγορο από το “gpt-4-turbo”, αφού χρειάζεται περίπου 10-15 δευτερόλεπτα για την παραγωγή 5 ασκήσεων, ενώ το “4-turbo” περίπου 30-35. Το πρώτο είναι επίσης και αρκετά πιο φτηνό. Το “gpt-3.5-turbo” παίρνει περίπου 20 δευτερόλεπτα, παρέχει καλές ασκήσεις, αλλά όχι στον βαθμό των άλλων δύο. Τα άλλα δύο προσφέρουν και τις απαντήσεις των ερωτήσεων τους, όπως επίσης και καλύτερη τυχαία ερώτηση. Βλέπουμε παράδειγμα των απαντήσεων των τριών μοντέλων για τα ίδια κεφάλαια:

gpt3.5-turbo:

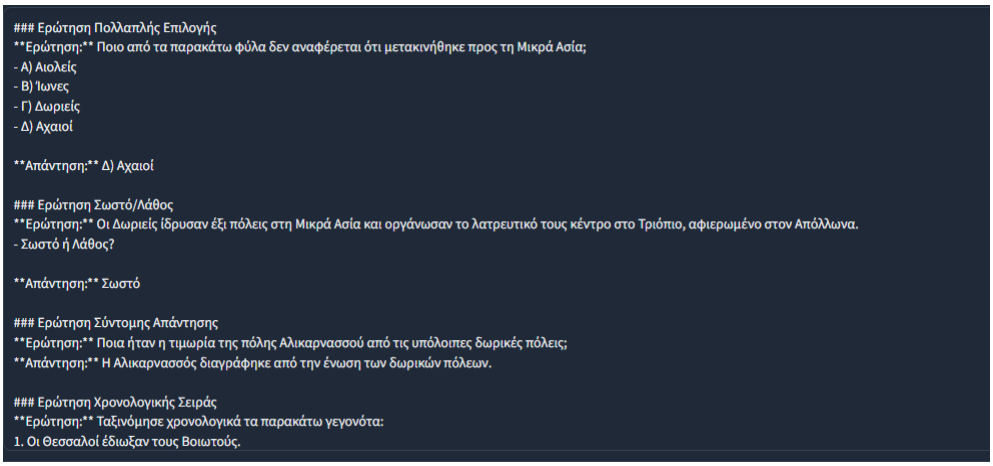


Σχήμα 4.14: Ερωτήσεις gpt3.5-turbo

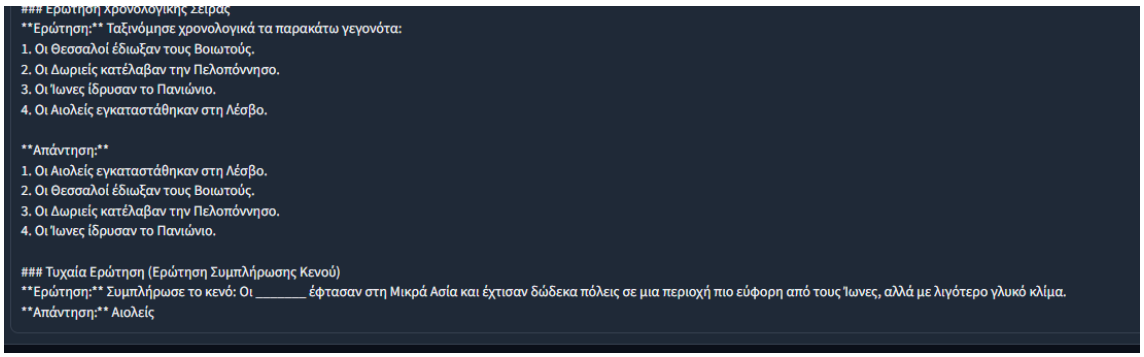


Σχήμα 4.15: Ερωτήσεις gpt3.5-turbo

gpt-4-turbo:

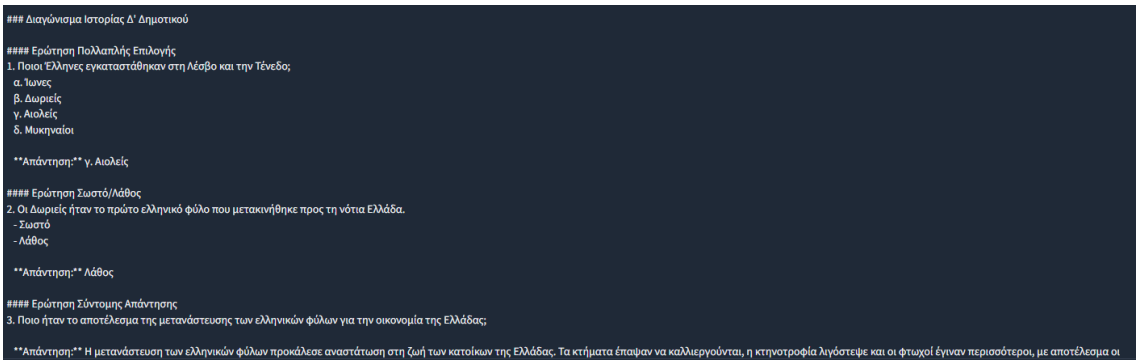


Σχήμα 4.16: Ερωτήσεις gpt4-turbo

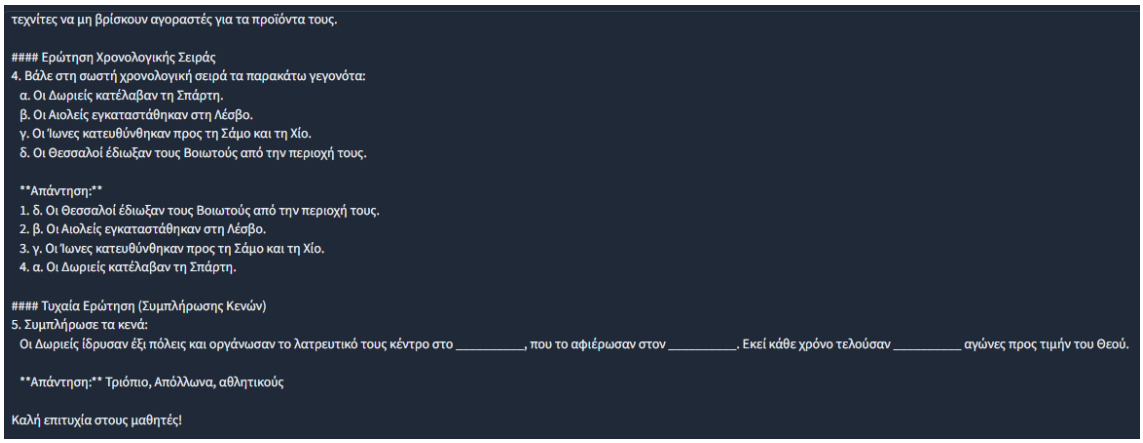


Σχήμα 4.17: Ερωτήσεις gpt4-turbo

gpt-4o



Σχήμα 4.18: Ερωτήσεις gpt-4o



Σχήμα 4.19: Ερωτήσεις gpt-4o

4.6 User Interface

Για την υλοποίηση ενός εύκολου στη χρήση UI χρησιμοποιήθηκε το Gradio. Όλη η βασική διαδικασία γίνεται με την χρήση 3 functions. Στο πρώτο κυρίως function γίνεται βασικά όλη η σχεδίαση του front end. Έχουν μπει δηλαδή όλα τα κουμπιά, τα text boxes, dropdowns, check boxes κλπ., τα οποία αφορούν την επιλογή της τάξης, του μαθήματος, των κεφαλαίων, το είδος και τον αριθμό των ερωτήσεων. Όλα αυτά, αφού επιλεγούν από τον χρήστη, αποτελούν τα inputs για το δεύτερο function.

Στο δεύτερο function συμβαίνει μια από της πιο κρίσιμες διαδικασίες ολόκληρου του προγράμματος. Γίνεται το retrieval των κειμένων από το database, με βάση τα inputs που αναφέραμε προηγουμένως, και έπειτα παράγονται τα prompts που θα δοθούν στο μοντέλο. Να σημειώσουμε επίσης ότι σε αυτό το σημείο, πραγματοποιούνται και

κάποιο έλεγχοι για error handling, ελέγχοντας πριν προχωρήσει το πρόγραμμα σε οποιαδήποτε άλλη διαδικασία, αν έχουν επιλεγεί από τον χρήστη όλες οι απαραίτητες είσοδοι, και αν όχι, να τους εμφανίσει σχετικό μήνυμα.

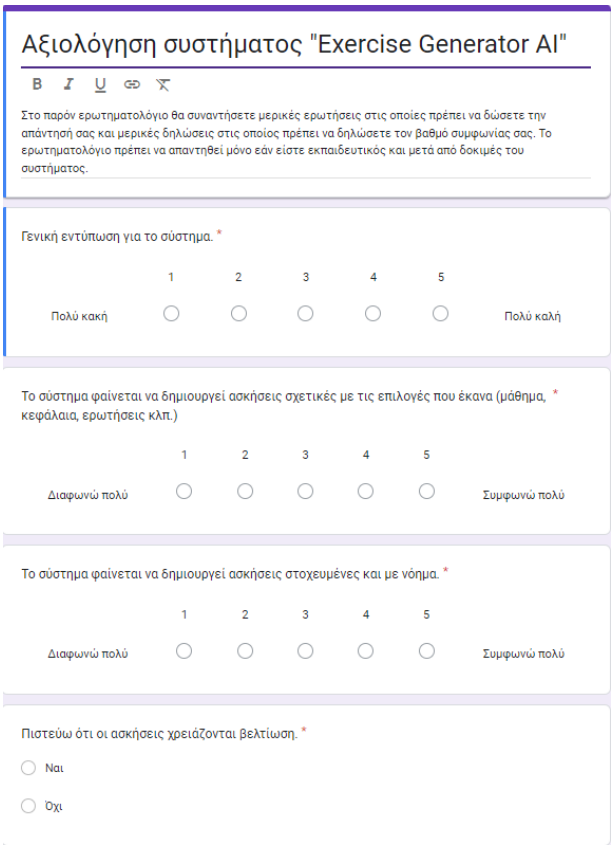
Τέλος, μέσω της τελευταίας συνάρτησης, καλείται η τρίτη συνάρτηση η οποία είναι υπεύθυνη για την αποστολή του prompt στο ChatGPT και την παραλαβή και επιστροφή της απάντησης, πίσω στο function του Gradio, για την εκτύπωση του output στην οθόνη.

Κεφάλαιο 5

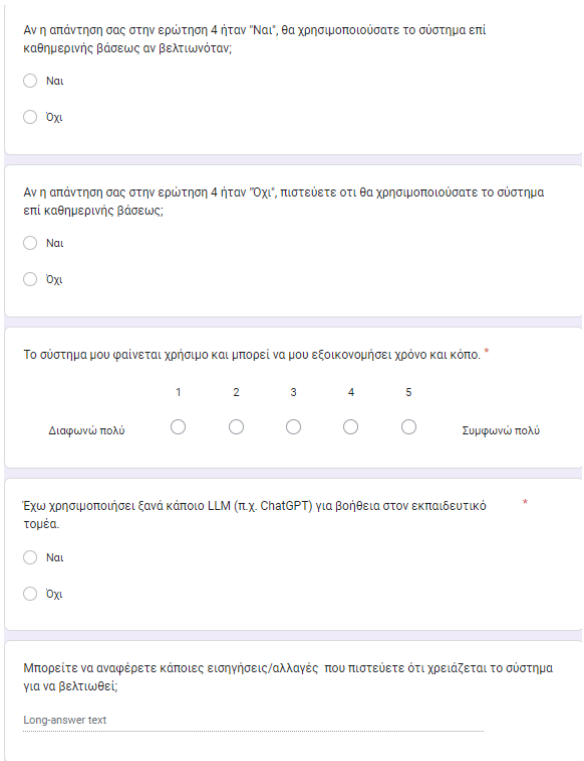
Αξιολόγηση Συστήματος

5.1 Ερωτηματολόγιο

Το ολοκληρωμένο σύστημα, έχει σταλεί σε 4 δασκάλους, και μετά από δοκιμές κλήθηκαν να απαντήσουν σε μερικές ερωτήσεις. Το ερωτηματολόγιο που τους έχει σταλεί, έχει ως στόχο να εξετάσει αν οι απαντήσεις που δίνει το σύστημα είναι βασισμένες στο κεφάλαιο, την τάξη, το μάθημα και γενικά στις επιλογές που κάνει ο χρήστης, και να εξετάσει αν οι απαντήσεις, με βάση τους ειδικούς, είναι σε ένα επίπεδο το οποίο μπορούν να χρησιμοποιηθούν στους μαθητές. Που είναι δηλαδή στοχευμένες και τείνουν να αναδεικνύουν την μάθηση ή τις αδυναμίες των μαθητών. Επίσης, ζητήθηκαν εισηγήσεις ή τυχόν αδυναμίες που εντόπισαν οι εκπαιδευτικοί και ήθελαν να εκφράσουν. Οι ερωτήσεις στο ερωτηματολόγιο ήταν οι εξής:



Σχήμα 5.1: Ερωτήσεις Ερωτηματολογίου



Σχήμα 5.2: Ερωτήσεις Ερωτηματολογίου

5.2 Εντυπώσεις

Με βάση την πρώτη ερώτηση, φαίνεται ότι το σύστημα τράβηξέ το ενδιαφέρον των εκπαιδευτικών, και γενικά σχημάτισαν μια καλή πρώτη εντύπωση για αυτό, αφού όλοι απάντησαν ότι έχουν πολύ καλή γενική εντύπωση. Στην ερώτηση που αφορά το αν οι ασκήσεις φτιάχνονται βασισμένες στις επιλογές του χρήστη, οι απαντήσεις ήταν όλες θετικές, κάτι που δείχνει ότι ο βασικός στόχος της εργασίας, το να μπορεί το σύστημα να μην δημιουργεί άσχετες και ψευδείς απαντήσεις, έχει επιτευχθεί. Οι περισσότεροι χρήστες απάντησαν ότι οι ασκήσεις δεν χρειάζονται βελτίωση και ότι είναι στοχευμένες και με νόημα, όμως ένας απάντησε ότι αυτό δεν ισχύει στον μέγιστο βαθμό. Αυτό εναπόκειται στη κρίση του καθενός, εννοείται όμως ότι το σύστημα θα εμφανίζει βελτίωση, όσο η OpenAI φτιάχνει νέα και πιο δυνατά μοντέλα, και όσο το μοντέλο γίνεται περισσότερο trained στον τομέα αυτό. Σε γενικές γραμμές όμως, το μοντέλο φαίνεται να παράγει αρκετά καλά αποτελέσματα, όπως είδαμε και στο προηγούμενο κεφάλαιο.

Όσον αφορά τις επόμενες ερωτήσεις, μόνο ένας χρήστης απάντησε ότι έχει ξανά-χρησιμοποιήσει LLM για κάποια βοήθεια στον εκπαιδευτικό τομέα, όμως όλοι οι χρήστες απάντησαν ότι το μοντέλο τους φαίνεται χρήσιμο και βοηθητικό στην εξοικονόμηση χρόνου σε μέγιστο βαθμό, και ότι, σίγουρα θα το χρησιμοποιούν στη καθημερινότητά τους. Γεγονός που μας δείχνει ότι ένα τέτοιο σύστημα θα φανεί πολύ χρήσιμο στο επάγγελμά τους.

5.3 Εισηγήσεις

Μερικές εισηγήσεις που είχαμε ήταν:

1. Το σύστημα να μπορεί να τυπώνει τις ασκήσεις.
2. Να εμφανίζονται και οι τίτλοι των κεφαλαίων, αντί μόνο οι αριθμοί για περισσότερη ευκολία.
3. Το σύστημα να επεκταθεί και να γίνει ένα εργαλείο που μπορεί να χρησιμοποιηθεί μέσα στη τάξη, σαν παιχνίδι, όπου όλοι οι μαθητές θα ενώνονται πάνω, θα τους εμφανίζεται η ίδια ερώτηση και θα την απαντούν.

Για την πρώτη εισήγηση, προστέθηκε κουμπί το οποίο αφήνει τον χρήστη να κατεβάσει στον υπολογιστή του αρχείο με της ερωτήσεις, και στη συνέχεια μπορεί να προβεί σε εκτύπωση.
Η δεύτερη εισήγηση, αποτελεί μια καλή ιδέα για βελτίωση στην χρήση του μοντέλου. Αυτό μπορεί να πραγματοποιηθεί, αν πριν την δημιουργία της βάσης, εισαχθούν οι τίτλοι ως metadata σε κάθε κεφάλαιο.
Η τελευταία εισήγηση αποτελεί μια αρκετά μεγάλη και έξυπνη επέκταση του συστήματος, το οποίο θα μπορεί πλέον να χρησιμοποιείται και από δασκάλους αλλά και μαθητές στο σχολικό περιβάλλον.

Κεφάλαιο 6

Συμπεράσματα

6.1 Γενική Επισκόπηση

Στις μέρες μας, τα Μεγάλα Γλωσσικά Μοντέλα (LLM) έχουν γίνει αναπόσπαστο μέρος της τεχνολογικής εξέλιξης και έχουν εκτοξευτεί σε νέα ύψη. Με την ανάπτυξη της τεχνητής νοημοσύνης και της μηχανικής μάθησης, τα LLM έχουν καταφέρει να επιτύχουν εντυπωσιακή ακρίβεια και απόδοση σε διάφορες εφαρμογές, όπως η μετάφραση, η παραγωγή κειμένου και η αναγνώριση πληροφοριών. Με το πέρας του χρόνου, η εξοικείωση με τη σωστή χρήση των LLMs θα είναι πλέον αναγκαία για όλους μας. Αυτά τα μοντέλα αναδεικνύουν τη σημασία τους σε διάφορους τομείς της ζωής μας, ενισχύοντας την επικοινωνία, προωθώντας την καινοτομία και αυξάνοντας την αποτελεσματικότητα σε πολλούς τομείς της καθημερινότητάς μας. Από τη μετάφραση μέχρι τη δημιουργία περιεχομένου και την ανάλυση δεδομένων, τα LLM αναδεικνύονται ως ισχυρά εργαλεία που επηρεάζουν θετικά την καθημερινή μας ζωή και την εργασία μας, με την προϋπόθεση πάντα ότι, χρησιμοποιούνται με τον σωστό τρόπο.

Η εφαρμογή που αναπτύχθηκε κατά τη διάρκεια της διπλωματικής εργασίας αυτής, αποτελεί ένα παράδειγμα που δείχνει πώς τα LLM μπορούν να προσφέρουν πολλά στον τομέα της εκπαίδευσης. Η εφαρμογή αυτή, επιτρέπει στους εκπαιδευτικούς να δημιουργούν ασκήσεις για τους μαθητές τους με έναν εύκολο και αποτελεσματικό τρόπο. Οι εκπαιδευτικοί μπορούν με μερικά απλά clicks στην οθόνη τους, να παράγουν

διαγωνίσματα ή φύλα εργασίας με διαφόρων ειδών ασκήσεις, προσαρμοσμένες στο θέμα που αυτοί επιθυμούν. Με αυτόν τον τρόπο, η χρήση του εργαλείου αυτού, μπορεί να συμβάλει στη βελτίωση της εκπαιδευτικής διαδικασίας και να ενισχύσει την αποτελεσματικότητα της εκπαίδευσης για πολλούς εκπαιδευτικούς και μαθητές.

6.2 Περιορισμοί

Το σύστημα αυτή τη στιγμή αποτελεί βασικά ένα demo της εφαρμογής, αφού γίνεται share με links που παρέχει το Gradio, τα οποία είναι active για μερικές ώρες. Για να μπορεί να χρησιμοποιηθεί κανονικά από χρήστες οποιαδήποτε στιγμή το επιθυμούν, πρέπει να γίνει hosted σε έναν Web Server. Το ChromaDB παρέχει documentation για τη διαδικασία του hosting στον AWS, όπως επίσης και το Gradio.

Ένας άλλος περιορισμός αποτελεί η ποσότητα των δεδομένων. Στη παρούσα φάση το σύστημα έχει ανατροφοδοτηθεί με μερικά βιβλία από μερικά μαθήματα. Για να γίνει ένα ολοκληρωμένο σύστημα και να μπορούν να το χρησιμοποιούν όλοι οι εκπαιδευτικοί, χρειάζεται όλα τα βιβλία, όλων των μαθημάτων, όλων των τάξεων.

Επίσης, το σύστημα, χρειάζεται βελτίωση στα vector embeddings του και στο similarity search από τη βάση. Θα έχει ακόμα πιο ακριβή αποτελέσματα αν το similarity search είναι πιο accurate.

6.3 Μελλοντική Δουλειά

Ως μελλοντική εργασία που μπορεί να πραγματοποιηθεί στο παρόν σύστημα, αποτελεί η επίλυση των πιο πάνω περιορισμών. Σε τέτοια περίπτωση, το σύστημα θα αποτελεί ένα πραγματικό βοήθημα για όλους τους εκπαιδευτικούς της Κύπρου, αφού θα είναι εύκολο στη πρόσβαση, και θα παρέχει επιλογές για όλα όσα υπάρχουν ως εκπαιδευτικό υλικό στα σχολεία μας.

Τέλος, το σύστημα, μπορεί να επεκταθεί σε ένα τεράστιο εργαλείο, το οποίο θα είναι φτιαγμένο για ολόκληρο το εκπαιδευτικό σύστημα. Θα μπορούν να ενώνονται και δασκάλοι αλλά και μαθητές, και όχι μόνο θα παράγει ασκήσεις, αλλά θα μπορεί να χρησιμοποιηθεί και με διάφορους άλλους τρόπους όπως για παράδειγμα:

- Να αποτελεί ένα μέσο επικοινωνίας δασκάλων – μαθητών, όπου δασκάλοι θα στέλνουν τα εργασίες των μαθητών, οι μαθητές θα τις κάνουν, και το AI θα διορθώνει τις ασκήσεις και θα παρέχει feedback και στις δύο πλευρές.
- Μπορεί να χρησιμοποιείται στην τάξη ως παιχνίδι μεταξύ των μαθητών.
- Μπορεί να ανατροφοδοτηθεί με οδηγούς εκπαιδευτικών και προγράμματα διδασκαλίας και να εκπαιδευτεί στη παραγωγή οδηγιών και νέων ιδεών για προσεγγίσεις μαθημάτων, βασισμένες πάντα στην ύλη και να χρησιμοποιείται από δασκάλους.
- Μπορεί να χρησιμοποιείται από μαθητές, οι οποίοι θα διαλέγουν το μάθημα που νιώθουν ότι έχουν κάποιες αδυναμίες, ή ότι δυσκολεύονται ή ότι χρειάζονται περισσότερη εξάσκηση, και να τους παρέχονται νέες εργασίες με βάση το επίπεδό τους.

Βιβλιογραφία

[1] Chiusano, F. (2022, January 10). Two minutes NLP — Sentence Transformers cheat sheet. NLPlanet. https://medium.com/nlplanet/two-minutes-nlp-sentence-transformers-cheat-sheet-2e9865083e7a

[2] Cohen, A. (n.d.). fuzzywuzzy: Fuzzy string matching in python. PyPI. https://pypi.org/project/fuzzywuzzy/

[3] *Databricks. Retrieval Augmented Generation (RAG) on Databricks*. (n.d.). Docs.databricks.com. https://docs.databricks.com/en/generative-ai/retrieval-augmented-generation.html

[4] Dwyer, J. *Exploring Chroma Vector Database Capabilities | Zeet.co*. (n.d.). Zeet.co. Retrieved May 19, 2024, from https://zeet.co/blog/exploring-chroma-vector-database-capabilities

[5] Gan, W., Qi, Z., Wu, J., & Lin, C.-W. (n.d.). *Large Language Models in Education: Vision and Opportunities*. https://arxiv.org/pdf/2311.13160

[6] Crone, N. (2023, August 22). *Vector Embeddings 101: The New Building Blocks for Generative AI*. Vector Database Central. https://vectordatabase.substack.com/p/vector-embeddings-101-the-new-building

[7] Gupta, Y. (2023, March 16). *Chat GPT and GPT 3 Detailed Architecture Study-Deep NLP Horse*. Nerd for Tech. https://medium.com/nerd-for-tech/gpt3-and-chat-gpt-detailed-architecture-study-deep-nlp-horse-db3af9de8a5d

[8] Jain, K. (2020, December 20). *Evolution of NLP — Part 4 — Transformers — BERT, XLNet, RoBERTa*. Analytics Vidhya. https://medium.com/analytics-vidhya/evolution-of-nlp-part-4-transformers-bert-xlnet-roberta-bd13b2371125

[9] Kanungo, N. (2023, August 31). *How Vector Databases Search by Similarity: A Comprehensive Primer*. KX Systems. https://medium.com/kx-systems/how-vector-databases-search-by-similarity-a-comprehensive-primer-c4b80d13ce63

[10] Kumar, M. (2023, September 13). Understanding Tokens in ChatGPT. Medium. https://medium.com/@manav.kumar87/understanding-tokens-in-chatgpt-32845987858d

[11] LangChain. Q&A with RAG | LangChain. (n.d.). Python.langchain.com. https://python.langchain.com/v0.1/docs/use_cases/question_answering/

[12] Leveau, P., *Unstructured Data vs. Structured Data: its Impact on Your ML Pipeline*. (n.d.). Kili-Website. https://kili-technology.com/training-

data/unstructured-data-vs-structured-data-know-the-difference-and-its-impact-on-your-ml-pipeline

[13] Novogroder, I. (2023, July 19). *What Is A Vector Database, And Why Do You Need One?* Git for Data - LakeFS. https://lakefs.io/blog/what-is-vector-databases/

[14] OpenAI. (n.d.) Teaching with AI. OpenAI. https://openai.com/index/teaching-with-ai/

[15] Phillips, H. (2023, May 14). *Positional Encoding*. Medium. https://medium.com/@hunter-j-phillips/positional-encoding-7a93db4109e6

[16] Python. (2009). *re — Regular expression operations — Python 3.7.2 documentation*. Python.org. https://docs.python.org/3/library/re.html

[17] Radeva, M. (2024, February 15). *GUEST POST: The Benefits and Risks of ChatGPT for Education*. The Learning Scientists. https://www.learningscientists.org/blog/2024/2/15-1

[18] Saeed, M. (2022, January 31). *A Gentle Introduction to Positional Encoding In Transformer Models, Part 1*. Machine Learning Mastery. https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

[19] Stöffelbauer, A. (2023, October 24). How Large Language Models Work. Data Science at Microsoft. https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f

[20] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, June 12). Attention Is All You Need. ArXiv.org. https://arxiv.org/abs/1706.03762

[21] Wolfram, S. (2023, February 14). What Is ChatGPT Doing … and Why Does It Work? Writings.stephenwolfram.com. https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

[22] Zvornicanin, E. (2024, January 26). Comparison Between BERT and GPT-3 Architectures [Review of Comparison Between BERT and GPT-3 Architectures]. Baeldung. https://www.baeldung.com/cs/bert-vs-gpt-3-architecture

Παράρτημα Α

Κώδικας για την δημιουργία της βάσης δεδομένων.

Install packages

```
%pip install -U langchain sentence-transformers chromadb openai gradio
```

Load Files

```
%unzip '/content/books.zip'
```

```python
import os
import re

# lists for the books' content
thriskeftika = []
istoria = []
texnologia = []
keimena = []
# lists for the books'  metadata
thriskeftika_metadata = []
istoria_metadata = []
texnologia_metadata = []
keimena_metadata = []
# lists for example questions
thriskeftika_examples = []
istoria_examples = []
texnologia_examples = []
# lists for examples metadata
thriskeftika_examples_metadata = []
istoria_examples_metadata = []
texnologia_examples_metadata = []

# Directory path
dir_path = '/content/books'
# List all files in the directory
file_names = [f for f in os.listdir(dir_path) if os.path.isfile(os.path.join(dir_path, f))]
# Loop over the list of files
for file_name in file_names:
    # Create the full file path
    file_path = os.path.join(dir_path, file_name)
    match = re.search(r"chapter(\d+)", file_name)
    if match:
      chapter_number = int(match.group(1))
    # Open and read the file
    with open(file_path, 'r', encoding='utf-8') as file:
        content = file.read()

        if("thriskeftika"  in file_name):
          if("exercises" in file_name):
            thriskeftika_examples.append(content)
            if("_D_" in file_name):
              if("merosA" in file_name):
```

```
      thriskeftika_examples_metadata.append({"chapter": chapter_number,"part":"A","subject":"thriskeftika","taksi":"D","type": "examples"})
    else:
      thriskeftika_examples_metadata.append({"chapter": chapter_number,"part":"B","subject":"thriskeftika","taksi":"D", "type": "examples"})

  elif("_E_" in file_name):
    if("merosA" in file_name):
      thriskeftika_examples_metadata.append({"chapter": chapter_number,"part":"A","subject":"thriskeftika","taksi":"E","type": "examples"})
    else:
      thriskeftika_examples_metadata.append({"chapter": chapter_number,"part":"B","subject":"thriskeftika","taksi":"E","type": "examples"})

  elif("_ST_" in file_name):
    if("merosA" in file_name):
      thriskeftika_examples_metadata.append({"chapter": chapter_number,"part":"A","subject":"thriskeftika","taksi":"ST","type": "examples"})
    else:
      thriskeftika_examples_metadata.append({"chapter": chapter_number,"part":"B","subject":"thriskeftika","taksi":"ST","type": "examples"})
 else:
  thriskeftika.append(content)
  if("_D_" in file_name):
    if("merosA" in file_name):
      thriskeftika_metadata.append({"chapter": chapter_number,"part":"A","subject":"thriskeftika","taksi":"D", "type": "theory"})
    else:
      thriskeftika_metadata.append({"chapter": chapter_number,"part":"B","subject":"thriskeftika","taksi":"D", "type": "theory"})

  elif("_E_" in file_name):
    if("merosA" in file_name):
      thriskeftika_metadata.append({"chapter": chapter_number,"part":"A","subject":"thriskeftika","taksi":"E", "type": "theory"})
    else:
      thriskeftika_metadata.append({"chapter": chapter_number,"part":"B","subject":"thriskeftika","taksi":"E","type": "theory"})

  elif("_ST_" in file_name):
    if("merosA" in file_name):
      thriskeftika_metadata.append({"chapter": chapter_number,"part":"A","subject":"thriskeftika","taksi":"ST","type": "theory"})
    else:
      thriskeftika_metadata.append({"chapter": chapter_number,"part":"B","subject":"thriskeftika","taksi":"ST","type": "theory"})


elif("istoria" in file_name):
 if("exercises" in file_name):
  istoria_examples.append(content)
  if("_taksiD_" in file_name):
    istoria_examples_metadata.append({"chapter": chapter_number,"subject":"istoria","taksi":"D","type": "examples"})
  elif("_taksiE_" in file_name):
    istoria_examples_metadata.append({"chapter": chapter_number,"subject":"istoria","taksi":"E","type": "examples"})
 else:
  istoria.append(content)
```

```
  if("_taksiD_" in file_name):
    istoria_metadata.append({"chapter": chapter_number,"subject":"istoria","taksi":"D","type": "theory"})
  elif("_taksiE_" in file_name):
    istoria_metadata.append({"chapter": chapter_number,"subject":"istoria","taksi":"E","type": "theory"})


elif("texnologia" in file_name):
 if("exercises" in file_name):
  texnologia_examples.append(content)
  if("_taksiE_" in file_name):
    texnologia_examples_metadata.append({"chapter": chapter_number,"subject":"texnologia","taksi":"E", "type": "examples"})
  elif("_taksiST_" in file_name):
    texnologia_examples_metadata.append({"chapter": chapter_number,"subject":"texnologia","taksi":"ST", "type":"examples"})
 else:
  texnologia.append(content)
  if("_taksiE_" in file_name):
    texnologia_metadata.append({"chapter": chapter_number,"subject":"texnologia","taksi":"E","type": "theory"})
  elif("_taksiST_" in file_name):
    texnologia_metadata.append({"chapter": chapter_number,"subject":"texnologia","taksi":"ST","type": "theory"})

elif("keimeno" in file_name):
 keimena.append(content)
 if("_D" in file_name):
    keimena_metadata.append({"subject":"keimena","taksi":"D","type": "theory"})
 elif("_E" in file_name):
    keimena_metadata.append({"subject":"keimena","taksi":"E","type": "theory"})
 elif("_ST" in file_name):
    keimena_metadata.append({"subject":"keimena","taksi":"ST","type": "theory"})
```

Split Chapters to Chunks

```
from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
```

```
import string
documents = []
metadatas = []

thrisk_documents = text_splitter.create_documents(thriskeftika, metadatas=thriskeftika_metadata)
thrisk_documents = text_splitter.split_documents(thrisk_documents)

thrisk_documents = [d for d in thrisk_documents if not all(c in string.punctuation for c in d.page_content)]

thriskeftika_metadatas = []
for i in range (0, len(thrisk_documents)):
  thriskeftika_metadatas.append(thrisk_documents[i].metadata)
  documents.append(thrisk_documents[i])
  metadatas.append(thriskeftika_metadatas[i])

istoria_documents = text_splitter.create_documents(istoria, metadatas=istoria_metadata)
istoria_documents = text_splitter.split_documents(istoria_documents)
istoria_metadatas = []
for i in range(0,len(istoria_documents)):
  istoria_metadatas.append(istoria_documents[i].metadata)
  documents.append(istoria_documents[i])
  metadatas.append(istoria_metadatas[i])

texnologia_documents = text_splitter.create_documents(texnologia, metadatas=texnologia_metadata)
texnologia_documents = text_splitter.split_documents(texnologia_documents)
texnologia_metadatas = []
for i in range(0,len(texnologia_documents)):
  texnologia_metadatas.append(texnologia_documents[i].metadata)
  documents.append(texnologia_documents[i])
  metadatas.append(texnologia_metadatas[i])


keimena_documents = text_splitter.create_documents(keimena, metadatas=keimena_metadata)
keimena_documents = text_splitter.split_documents(keimena_documents)
keimena_metadatas = []
```

```python
for i in range(0,len(keimena_documents)):
  keimena_metadatas.append(keimena_documents[i].metadata)
  documents.append(keimena_documents[i])
  metadatas.append(keimena_metadatas[i])


text_splitter = RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=100)

thrisk_examples_documents = text_splitter.create_documents(thriskeftika_examples, metadatas=thriskeftika_examples_metadata)
thrisk_examples_documents = text_splitter.split_documents(thrisk_examples_documents)

thriskeftika_examples_metadatas = []
for i in range (0, len(thrisk_examples_documents)):
  thriskeftika_examples_metadatas.append(thrisk_examples_documents[i].metadata)
  documents.append(thrisk_examples_documents[i])
  metadatas.append(thriskeftika_examples_metadatas[i])

istoria_examples_documents = text_splitter.create_documents(istoria_examples, metadatas=istoria_examples_metadata)
istoria_examples_documents = text_splitter.split_documents(istoria_examples_documents)
istoria_examples_metadatas = []
for i in range(0,len(istoria_examples_documents)):
  istoria_examples_metadatas.append(istoria_examples_documents[i].metadata)
  documents.append(istoria_examples_documents[i])
  metadatas.append(istoria_examples_metadatas[i])

texnologia_examples_documents = text_splitter.create_documents(texnologia_examples, metadatas=texnologia_metadata)
texnologia_examples_documents = text_splitter.split_documents(texnologia_examples_documents)
texnologia_examples_metadatas = []
for i in range(0,len(texnologia_examples_documents)):
  texnologia_examples_metadatas.append(texnologia_examples_documents[i].metadata)
  documents.append(texnologia_examples_documents[i])
  metadatas.append(texnologia_examples_metadatas[i])
```

Setup our Vector Database

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```python
import chromadb
client = chromadb.PersistentClient(path="/content/drive/MyDrive/chromadb")
```

```python
from sentence_transformers import SentenceTransformer
```

```python
from chromadb.utils import embedding_functions
sentence_transformer_ef = embedding_functions.SentenceTransformerEmbeddingFunction(model_name='sentence-transformers/paraphrase-multilingual-mpnet-base-v2')
collection = client.create_collection(name="my_collection", embedding_function=sentence_transformer_ef, metadata={"hnsw:space": "cosine"})
```

Add documents to database

```
collection.add(
    documents = [docs.page_content for docs in documents],
    metadatas = metadatas,
    ids       = [str(i) for i in range(len(documents))]
)
```

Κώδικας για το prompting και το querying στη βάση δεδομένων.

```
collection = client.get_collection(name="my_collection")
```

Create Exam

Create using Specific Parameters from user

```
import locale

locale.getpreferredencoding = lambda: "UTF-8"
```

```
from openai import OpenAI
```

```
Get openAI API KEY
```

```
# get a token: https://platform.openai.com/account/api-keys

import os
from getpass import getpass
OPENAI_API_KEY = "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"

os.environ["OPENAI_API_KEY"] = OPENAI_API_KEY
GPT_MODEL = "gpt-4-turbo"

client = OpenAI(api_key=os.environ.get("OPENAI_API_KEY", OPENAI_API_KEY))
```

Ask the LLM

```python
def ask_LLM(content,prompt):
    response = client.chat.completions.create(
        messages=[
            {'role': 'system', 'content': content},
            {'role': 'user', 'content': prompt},
        ],
        model=GPT_MODEL,
        temperature=0.0001,
    )

    return response.choices[0].message.content
```

Create the prompt

```python
from sentence_transformers import SentenceTransformer

# Load a model
model = SentenceTransformer('sentence-transformers/paraphrase-multilingual-mpnet-base-v2')

def generate_prompt(grade, lesson, chapters, question_types,number_Multiple,number_trueFalse,number_shortAns, number_chronoOrder, number_randomQuestion, number_keimena, query, part):

    if(grade == None):
        return "Επιλέξτε Τάξη." , None
    if(lesson == None):
        return "Επιλέξτε Μάθημα.", None
    if (lesson == "ΘΡΗΣΚΕΥΤΙΚΑ" or lesson == "ΙΣΤΟΡΙΑ" or lesson == "ΤΕΧΝΟΛΟΓΙΑ"):
        if(not chapters):
            return "Επιλέξτε Κεφάλαιο.", None

    if(grade == "Δ'"):
        grade = 'D'
    elif(grade == "Ε'"):
        grade = 'E'
    elif(grade == "ΣΤ'"):
        grade = 'ST'

    if(lesson != "ΕΛΛΗΝΙΚΑ"):

        if (lesson == "ΘΡΗΣΚΕΥΤΙΚΑ"):
            lesson = "thriskeftika"
            if(part == "ΜΕΡΟΣ Α'"):
                part = 'A'
            else:
                part = 'B'
```

```
results = collection.get(
  where = {
    "$and": [
      {
        "chapter": {
          "$in": chapters
        }
      },
      {
        "subject": {
          "$eq": lesson
        }
      },
      {
        "taksi": {
          "$eq": grade
        }
      },
      {
        "part": {
          "$eq": part
        }
      },
      {
        "type": {
          "$eq": "theory"
        }
      }
    ]
  }
)
```

```
examples = collection.get(
  where = {
    "$and": [
      {
        "chapter": {
          "$in": chapters
        }
      },
      {
        "subject": {
          "$eq": lesson
        }
      },
      {
        "taksi": {
          "$eq": grade
        }
      },
      {
        "part": {
          "$eq": part
        }
      },
      {
        "type": {
          "$eq": "examples"
        }
      }
    ]
  }
)
```

```
elif (lesson == "ΙΣΤΟΡΙΑ" or lesson == "ΤΕΧΝΟΛΟΓΙΑ"):
  if (lesson == "ΙΣΤΟΡΙΑ"):
    lesson = "istoria"
  else:
    lesson = "texnologia"

  results = collection.get(
    where = {
      "$and": [
        {
            "chapter": {
              "$in": chapters
            }
        },
        {
            "subject": {
              "$eq": lesson
            }
        },
        {
            "taksi": {
              "$eq": grade
            }
        },
        {
            "type": {
              "$eq": "theory"
            }
        }
      ]
    }
  )
```

```
    examples = collection.get(
      where = {
        "$and": [
          {
            "chapter": {
              "$in": chapters
            }
          },
          {
            "subject": {
              "$eq": lesson
            }
          },
          {
            "taksi": {
              "$eq": grade
            }
          },
          {
            "type": {
              "$eq": "examples"
            }
          }
        ]
      }
    )


    question_types_str = ", ".join(question_types)
    #check if the user input is something that exists, if not, return , to prevent wrong outputs
    if (len(results['ids'])==0):
      return "Δεν υπάρχει αυτό το κεφάλαιο.", None

    context_text = "".join(results['documents'])

    content = f"Είσαι έμπειρος δάσκαλος, με χρόνια εμπειρίας στην παραγωγή διαγωνισμάτων για μαθητές Δημοτικού, που αποδεικνύουν αποτελεσματικά τη μάθησή και την κατανόησή τους. \
    Είσαι υπεύθυνος να βοηθήσεις στην κατασκευή ενός διαγωνίσματος για μαθητές {grade} Δημοτικού στο μάθημα {lesson}\n "
    prompt =  f"Με βάση τα κείμενα που θα σου δοθούν πιο κάτω, να φτιάξεις {number_Multiple+number_trueFalse+number_shortAns+number_chronoOrder+number_randomQuestion} ασκήσεις \
    οι οποίες θα είναι του τύπου που θα σου δοθεί πιο κάτω. Αν οι ασκήσεις είναι περισσότερες από μία, τότε να εστιαστείς σε όλα τα διαφορετικά κεφάλαια (θέματα) που μπορεί να δοθούν στα κείμενα. \n\
    #Κείμενα :\n\
    {context_text}\n\
    #Τύπος ασκήσεων:\n\
    {question_types_str}\n\
    #Αριθμός Ερωτήσεων για κάθε τύπο: \n"
```

```
if (number_Multiple > 0):
  if (number_Multiple == 1):
      prompt += str(number_Multiple) + " Ερώτηση Πολλαπλής Επιλογής\n"
  else:
    prompt += str(number_Multiple) + " Ερωτήσεις Πολλαπλής Επιλογής\n"

if (number_trueFalse > 0):
  if (number_trueFalse == 1):
    prompt += str(number_trueFalse) + " Ερώτηση Σωστό/Λάθος\n"
  else:
    prompt += str(number_trueFalse) + " Ερωτήσεις Σωστό/Λάθος\n"

if (number_shortAns > 0):
  if (number_shortAns == 1):
    prompt += str(number_shortAns) + " Ερώτηση Σύντομης Απάντησης"
  else:
    prompt += str(number_shortAns) + " Ερωτήσεις Σύντομης Απάντησης"

if (number_chronoOrder > 0):
  if (number_chronoOrder == 1):
    prompt += str(number_chronoOrder) + " Ερώτηση Χρονολογικής Σειράς"
  else:
    prompt += str(number_chronoOrder) + " Ερωτήσεις Χρονολογικής Σειράς"

if (number_randomQuestion > 0):
  if (number_randomQuestion == 1):
    prompt += str(number_randomQuestion) + " Τυχαία Ερώτηση"
  else:
    prompt += str(number_randomQuestion) + " Τυχαίες Ερωτήσεις"
  prompt += "Για τυχαία ερώτηση, μπορείς να φτιάξεις όποιον δήποτε τύπο άσκησης μπορείς να σκεφτείς. Προσπάθησε να είναι διαφορετική ερώτηση από τις υπόλοιπες που έφτιαξες. "


if (len(examples['ids'])!=0):
  example_questions = "".join(examples['documents'])
  prompt += "Μπορείς να πάρεις κάποιες ιδέες για το που θα βασίζονται οι ασκήσεις που θα φτιάξεις από αυτά τα παραδείγματα: "
  prompt += example_questions
```

```
else:
  # Encode the query
  query_vector = model.encode(query).tolist()
  docs = collection.query(
    query_embeddings=[query_vector],
    n_results=10,
    where={"$and": [
        {
            "taksi": {
                "$eq": grade
            }
        },
        {
            "subject": {
              "$eq": "keimena"
            }
        }
        ]
      },
  )

  key_words = [word.strip() for word in query.split(',')]


  res1 = []
  # Loop through each keyword and perform a query
  for keyword in key_words:
    docs1 = collection.get(
      where = {  "subject": {
                        "$eq": "keimena"
                      }
      },
    where_document = {"$contains": keyword}
    )
    res1.append(docs1['documents'])

  if (len(docs['ids'])==0):
    return "Δεν υπάρχει αυτό το κεφάλαιο.", None
```

```
  flat_list = [item for sublist in docs['documents'] for item in sublist]
  context_text = "".join(flat_list)

  if (len(res1) != 0):
    flat_list = [item for sublist in res1 for item in sublist]
    context_text1 = "".join(flat_list)
    context_text += context_text1

  content = f"Είσαι έμπειρος δάσκαλος, με χρόνια εμπειρίας στην παραγωγή διαγωνισμάτων για μαθητές Δημοτικού, που αποδεικνύουν αποτελεσματικά τη μάθησή τους. \
  Είσαι υπεύθυνος να βοηθήσεις στην κατασκευή ενός διαγωνίσματος για μαθητές {grade} Δημοτικού στο μάθημα {lesson}\n "
  prompt = f"Θα σου δοθούν κάποιες λέξεις κλειδία και πρέπει να βασιστείς σε αυτές για να φτιάξεις {number_keimena} σχετικές ασκήσεις \
  Μπορείς να πάρεις κάποια παραδείγματα από τις πιο κάτω ασκήσεις, αλλά δεν πρέπει να βασιστείς μόνο σε αυτές. Προσπάθησε να μην ξεφύγεις από το θέμα των λέξεων κλειδιών που σου δόθηκε. \
  Αν χρειαστεί, δημιούργησε και ένα μικρό κείμενο που θα βασίζονται οι ασκήσεις πάνω σε αυτό. Το κείμενο μπορεί να αποτελείται από μια παράγραφο. \n\
  #Λέξεις κλειδία :\n\
  {query}\n\
  #Παραδείγματα Ασκήσεων :\n\
  {context_text}\n\ "

exam = ask_LLM(content,prompt)
with open("exercises.txt", "w") as file:
      file.write(exam)

return exam,"exercises.txt"
```

Κώδικας για UI.

Create the UI

```
import gradio as gr

grades = ["Δ'", "Ε'", "ΣΤ'"]
lessons = ['ΘΡΗΣΚΕΥΤΙΚΑ', 'ΤΕΧΝΟΛΟΓΙΑ', 'ΙΣΤΟΡΙΑ', 'ΕΛΛΗΝΙΚΑ']  # Example lessons
# Define chapter ranges for different lessons
chapter_ranges = {
    'ΘΡΗΣΚΕΥΤΙΚΑ': list(range(1, 19)),  # chapters for ΘΡΗΣΚΕΥΤΙΚΑ
    'ΙΣΤΟΡΙΑ': list(range(1, 44)),       # chapters for ΙΣΤΟΡΙΑ
    'ΤΕΧΝΟΛΟΓΙΑ': list(range(1, 6)),    # chapters for TEXNOLOGIA
    'ΕΛΛΗΝΙΚΑ': list(range(1, 16))
}
question_types_options = ['Πολλαπλής Επιλογής', 'Σωστό/Λάθος', 'Σύντομης Απάντησης', 'Χρονολογικής Σειράς', 'Τυχαία Ερώτηση']

def change_inputs(choice):
    # Get chapter range based on selected lesson
    chapters = chapter_ranges.get(choice, list(range(1, 16)))
    if (choice == "ΕΛΛΗΝΙΚΑ"):
        return [
            gr.Textbox(label="Δώστε λέξεις κλειδία για τη δημιουργία σχετικών ασκήσεων:", visible=True),
            gr.Dropdown(choices=[], visible=False),
            gr.CheckboxGroup(choices=[], visible=False),
            gr.CheckboxGroup(choices=[], visible=False),
            gr.Number(value=1, visible=False),
            gr.Number(value=0, visible=False),
            gr.Number(value=0, visible=False),
            gr.Number(value=0, visible=False),
            gr.Number(value=0, visible=False),
            gr.Number(label="Αριθμός Ασκήσεων ", value=2, visible=True)

        ]
```

```
elif (choice == "ΘΡΗΣΚΕΥΤΙΚΑ"):
    return [
        gr.Textbox(visible=False),
        gr.Dropdown(choices=["ΜΕΡΟΣ Α'", "ΜΕΡΟΣ Β'"], label="Επιλογή Μέρους", visible=True, value = "ΜΕΡΟΣ Α'"),
        gr.Dropdown(choices=chapters, label="ΚΕΦΑΛΑΙΑ", visible=True, multiselect = True, max_choices = 3),
        gr.CheckboxGroup(choices=question_types_options, label="ΕΙΔΟΣ ΑΣΚΗΣΕΩΝ", visible=True),
        gr.Number(label="Αριθμός Ερωτήσεων Πολλαπλής Επιλογής", value=1, visible=True),
        gr.Number(label="Αριθμός Ερωτήσεων Σωστού/Λάθους", value=0, visible=True),
        gr.Number(label="Αριθμός Ερωτήσεων Σύντομης Απάντησης", value=0, visible=True),
        gr.Number(label="Αριθμός Ερωτήσεων Χρονολογικής Σειράς", value=0, visible=True),
        gr.Number(label="Αριθμός Τυχαίων Ερωτήσεων ", value=0, visible=True),
        gr.Number(value=0, visible=False)

    ]
elif (choice == "ΤΕΧΝΟΛΟΓΙΑ" or choice == "ΙΣΤΟΡΙΑ"):
    return [
        gr.Textbox(visible=False),
        gr.Dropdown(choices=[], visible=False),
        gr.Dropdown(choices=chapters, label="ΚΕΦΑΛΑΙΑ", visible=True, multiselect = True, max_choices = 3),
        gr.CheckboxGroup(choices=question_types_options, label="ΕΙΔΟΣ ΑΣΚΗΣΕΩΝ", visible=True),
        gr.Number(label="Αριθμός Ερωτήσεων Πολλαπλής Επιλογής", value=1, visible=True),
        gr.Number(label="Αριθμός Ερωτήσεων Σωστού/Λάθους", value=0, visible=True),
        gr.Number(label="Αριθμός Ερωτήσεων Σύντομης Απάντησης", value=0, visible=True),
        gr.Number(label="Αριθμός Ερωτήσεων Χρονολογικής Σειράς", value=0, visible=True),
        gr.Number(label="Αριθμός Τυχαίων Ερωτήσεων ", value=0, visible=True),
        gr.Number(value=0, visible=False)

    ]
```

```
with gr.Blocks() as demo:
    with gr.Row():
        gr.Markdown("EXERCISES GENERATOR AI")
    with gr.Row():
        grade_radio = gr.Radio(choices=grades, label="TA∃H")
        lesson_radio = gr.Radio(choices=lessons, label="MA0HMA")
    with gr.Row():
        query_text = gr.Textbox(visible=False)
        part_dropdown = gr.Dropdown(choices=[], visible=False)
        chapter_choice = gr.Dropdown(choices=[], visible=False)
        question_types_group = gr.CheckboxGroup(choices=[], visible=False)
        mcq_number = gr.Number(value=1, visible=False)
        tf_number = gr.Number(value=0, visible=False)
        sa_number = gr.Number(value=0, visible=False)
        ho_number = gr.Number(value=0, visible=False)
        rq_number = gr.Number(value=0, visible=False)
        keimena_number = gr.Number(value=2, visible=False)


    lesson_radio.change(change_inputs, inputs=lesson_radio, outputs=[query_text, part_dropdown, chapter_choice, question_types_group, mcq_number, tf_number, sa_number, ho_number, rq_number, keimena_number])

    submit_button = gr.Button("Generate")
    output_text = gr.Textbox()
    output_file = gr.File(label="Download Output File")

    submit_button.click(
        generate_prompt,
        inputs=[grade_radio, lesson_radio, chapter_choice, question_types_group, mcq_number, tf_number, sa_number, ho_number, rq_number, keimena_number, query_text, part_dropdown],
        outputs=[output_text,output_file]
    )


demo.launch(share = True)
```