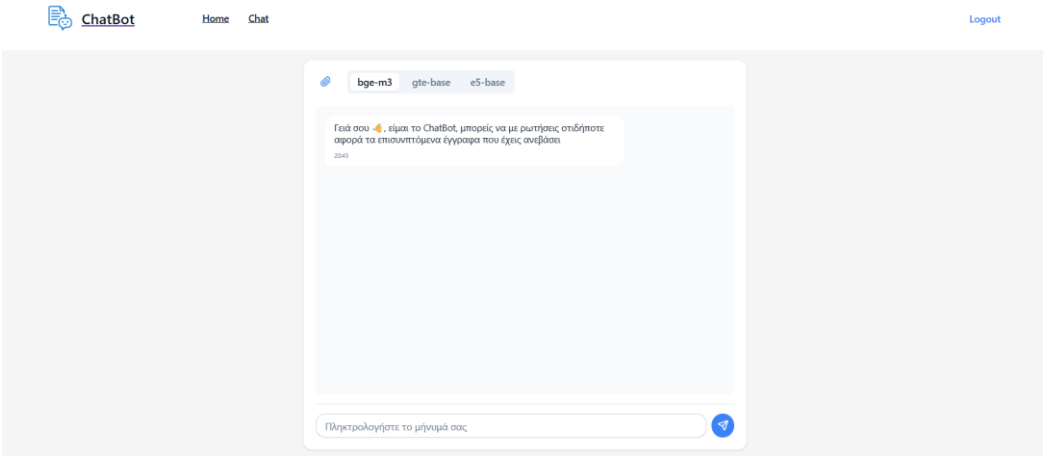
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη ενός Chatbot ανάκτησης εγγράφων για τον ελληνικό δημόσιο τομέα με τη χρήση προ-εκπαιδευμένων LLMs»



Του φοιτητή
Κωνσταντίνου Τριανταφύλλου
Αρ. Μητρώου: ΑΜ 174954

Επιβλέπων
Μπράτσας Χαράλαμπος
Επίκουρος Καθηγητής

Ημερομηνία 18/01/2026

Ανάπτυξη ενός Chatbot ανάκτησης εγγράφων για τον Ελληνικό Δημόσιο Τομέα με την χρήση προ-εκπαιδευμένων LLMs

Κωδικός Π.Ε. 24324

Ονοματεπώνυμο φοιτητή : Κωνσταντίνος Τριανταφύλλου

Ονοματεπώνυμο εισηγητή: Μπράτσας Χαράλαμπος

Ημερομηνία ανάληψης Π.Ε 28/11/2024

Ημερομηνία περάτωσης Π.Ε 18/01/2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Αφιερώνεται στην οικογένειά μου»

Πρόλογος

Επέλεξα αυτό το θέμα λόγω του έντονου ενδιαφέροντός μου για την επεξεργασία φυσικής γλώσσας (natural language processing) και της πρόκλησης που αποτελεί η πρόσβαση σε “μη δομημένα” έγγραφα του ελληνικού δημοσίου μέσω έξυπνης αναζήτησης. Πολλοί οργανισμοί αποθηκεύουν τα δεδομένα τους σε αρχεία τύπου Pdf, Word, αλλά η ανάκτηση σημαντικών απαντήσεων από αυτά παραμένει δύσκολη. Με το σχεδιασμό και την υλοποίηση ενός chatbot με τεχνολογία RAG (retrieval-augmented generation), μπόρεσα να εξερευνήσω σύγχρονες τεχνικές στην εισαγωγή και ενσωμάτωση εγγράφων σε μία διανυσματική βάση, τις οποία εφάρμοσα σε μία διεπαφή χρήστη (user interface), δημιουργώντας μία ολοκληρωμένη εφαρμογή με ασφαλή πιστοποίηση σύνδεσης.

Αυτό το έργο, μου επέτρεψε να εμβαθύνω τις γνώσεις μου σε εργαλεία τελευταίας τεχνολογίας, όπως η FAISS βιβλιοθήκη, για αποτελεσματική σημασιολογική αναζήτηση, τα πολυγλωσσικά μοντέλα ενσωμάτωσης (multilingual embedding models) που υποστηρίζουν τα ελληνικά, και το FastAPI για τη δημιουργία επεκτάσιμων διασυνδέσεων προγραμματισμού, εφαρμογών (API). Η ενσωμάτωση της γλώσσας Angular και του εργαλείου Keycloak με βοήθησε επίσης να αποκτήσω πρακτική εμπειρία στην ανάπτυξη end-to-end εφαρμογών. Συνολικά, αυτή η εργασία όχι μόνο ενίσχυσε τις γνώσεις μου στον τομέα της τεχνητής νοημοσύνης και της μηχανικής λογισμικού, αλλά μου παρείχε επίσης πολύτιμη εμπειρία στην ανάπτυξη λύσεων που μπορούν να εφαρμοστούν στην πραγματική ζωή.

Περίληψη

Η ραγδαία αύξηση του όγκου των ψηφιακών πληροφοριών έχει δημιουργήσει σημαντικές προκλήσεις στην πρόσβαση σε γνώσεις που είναι αποθηκευμένες σε ετερογενείς μορφές εγγράφων, όπως PDF, DOCX και αρχεία απλού κειμένου. Οι παραδοσιακές μέθοδοι αναζήτησης βασισμένες σε λέξεις-κλειδιά συχνά αποτυγχάνουν να ικανοποιήσουν σύνθετες πληροφοριακές ανάγκες, καθώς δεν λαμβάνουν υπόψη το σημασιολογικό περιεχόμενο των κειμένων, οδηγώντας σε ασαφή ή ελλιπή αποτελέσματα. Το πρόβλημα αυτό γίνεται εντονότερο σε σενάρια που περιλαμβάνουν ελληνόγλωσσο περιεχόμενο, όπου οι διαθέσιμες λύσεις είναι περιορισμένες.

Η παρούσα πτυχιακή εργασία αντιμετωπίζει τις παραπάνω προκλήσεις μέσω του σχεδιασμού και της υλοποίησης ενός chatbot βασισμένου σε τεχνολογία RAG, ικανού να εισάγει, να ταξινομεί και να ανακτά σημαντικές πληροφορίες από ελληνικά έγγραφα. Το προτεινόμενο σύστημα ακολουθεί μια αρθρωτή αρχιτεκτονική. Αρχικά, πραγματοποιείται στάδιο προεπεξεργασίας, κατά το οποίο εξάγεται το ακατέργαστο κείμενο από τα μεταφορτωμένα έγγραφα και διαχωρίζεται σε μικρότερα, συνεκτικά τμήματα κατάλληλα για περαιτέρω ανάλυση. Στη συνέχεια, τα τμήματα αυτά μετατρέπονται σε πυκνές διανυσματικές αναπαραστάσεις με τη χρήση πολυγλωσσικών μοντέλων ενσωμάτωσης, όπως το *multilingual-e5-base*. Οι ενσωματώσεις αποθηκεύονται σε ευρετήριο διανυσμάτων FAISS, επιτρέποντας αποδοτική αναζήτηση βάσει σημασιολογικής ομοιότητας. Κατά την υποβολή ενός ερωτήματος από τον χρήστη, το σύστημα το κωδικοποιεί στον ίδιο χώρο ενσωμάτωσης, ανακτά τα πιο συναφή τμήματα και τα προωθεί σε ένα γλωσσικό μοντέλο που δημιουργεί μια συνεκτική απάντηση.

Για τη βέλτιστη αξιοποίηση της προτεινόμενης λύσης, το σύστημα αναπτύχθηκε ως διαδικτυακή εφαρμογή, με την πλευρά του εξυπηρετητή να υλοποιείται σε Python μέσω FastAPI και να παρέχει REST API, ενώ η διεπαφή χρήστη αναπτύχθηκε σε Angular. Η πιστοποίηση και ο έλεγχος πρόσβασης υλοποιούνται μέσω ενσωμάτωσης του Keycloak.

Συνολικά, τα αποτελέσματα της εργασίας καταδεικνύουν ότι ο συνδυασμός πολυγλωσσικών μοντέλων και ενισχυμένης ανάκτησης βελτιώνει σημαντικά την ποιότητα απάντησης ερωτήσεων σε ελληνικά έγγραφα, προσφέροντας ένα πρακτικό και επεκτάσιμο πλαίσιο για τη μετατροπή μη δομημένων συλλογών σε προσβάσιμες και αναζητήσιμες πηγές γνώσης.

Development of a document retrieval chatbot for the Greek public sector using pre-trained LLMs

Konstantinos Triantafyllou

Abstract

The rapid growth of digital information has created significant challenges in accessing knowledge stored in diverse document formats such as PDF, DOCX, and plain text files. Traditional search methods are often insufficient when users seek precise, context-aware answers rather than simple keyword matches. This thesis addresses this problem by designing and implementing a retrieval-augmented generation (RAG) chatbot that can ingest, index, and retrieve meaningful information from Greek documents.

The chatbot that was developed follows a modular pipeline. Initially, a preprocessing stage extracts the raw text from uploaded documents and divides it into smaller, coherent chunks that are suitable for subsequent processing. Consequently, these segments are transformed into dense vector representations through the utilization of multilingual embedding models, such as multilingual-e5-base. The embeddings are stored in a FAISS vector index, enabling efficient similarity search. Upon submission of a query by a user, the system encodes it into the same embedding space, retrieves the most relevant chunks, and forwards them to a language model that generates a coherent response.

To ensure optimal usability, the solution is deployed as a web application. The backend, implemented in Python with FastAPI, exposes a REST API that performs document ingestion, search, and question answering. The frontend, developed in Angular, provides an interactive interface, while authentication and access control are managed through Keycloak integration.

Overall, the results of this thesis demonstrate that the combination of multilingual models and retrieval-augmented generation significantly improves the quality of question answering over Greek documents, providing a practical and scalable framework for transforming unstructured document collections into accessible and searchable sources of knowledge.

Ευχαριστίες

Ευχαριστώ θερμά τον καθηγητή μου κ. Μπράτσα Χαράλαμπο για την εμπιστοσύνη που μου έδειξε με την ανάθεση και την εκπόνηση αυτής της πτυχιακής εργασίας, αλλά και για την στήριξή του καθ' όλη τη διάρκεια της συνεργασίας μας.

Έπειτα θα ήθελα να ευχαριστήσω τους Διδακτορικούς Ερευνητές του τμήματος κ. Αγγελίδη Αλέξανδρο και κ. Αναστασιάδη Ευστάθιο Κωνσταντίνο, για την καθοδήγηση τους, τις συμβουλές τους καθώς επίσης και τη συνεχή βοήθεια που μου παρείχαν καθ' όλη τη διάρκεια των τελευταίων μηνών.

Θα ήθελα επίσης να ευχαριστήσω τους γονείς μου, οι οποίοι πίστεψαν σε μένα και ο καθένας με τον δικό του μοναδικό τρόπο μου έδωσε δύναμη να συνεχίσω την προσπάθεια μου και με στήριξε σε κάθε μου επιλογή όλα αυτά τα χρόνια.

Τέλος οφείλω ένα μεγάλο ευχαριστώ στους φίλους μου, για όλες τις αξέχαστες και όμορφες στιγμές που ζήσαμε κατά την διάρκεια των σπουδών μας, αλλά και για την υποστήριξη τους τόσο σε επαγγελματικό όσο και σε προσωπικό επίπεδο.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
LLMs	Large Language Models
RAG	Retrieval Augmented Generation
API	Application Programming Interface
UI	User Interface
REST API	Representational State Transfer API
CIR	Contextual Information Retrieval
AI	Artificial Intelligence
Σ.Ο.	Σημασιολογική Ομοιότητα
FAISS	Facebook AI Similarity Search

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

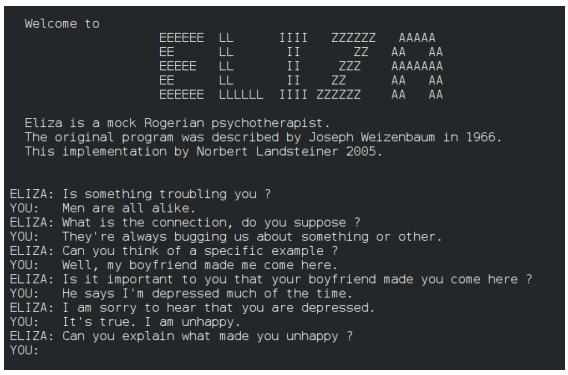
Το παρόν κεφάλαιο θέτει το πλαίσιο μέσα στο οποίο δημιουργήθηκε η πτυχιακή εργασία. Αρχικά πραγματοποιείται μία σύντομη ιστορική αναδρομή στην εξέλιξη των εικονικών συνομιλητών αλλά και των τεχνικών αναζήτησης πληροφορίας. Στη συνέχεια παρουσιάζεται το πρόβλημα που προσπαθεί να λυθεί μέσω του chatbot που αναπτύχθηκε, το οποίο σχετίζεται με τη δυσκολία αναζήτησης και αξιοποίησης πληροφορίας σε μεγάλα και μη δομημένα κείμενα. Έπειτα καθορίζεται ο σκοπός και οι βασικοί στόχοι της εργασίας. Το κεφάλαιο ολοκληρώνεται με την παρουσίαση της διάρθρωσης της εργασίας.

1.2 Ιστορική Αναδρομή

1.2.1 Η εξέλιξη των Chatbot

Ένα chatbot (αρχικά γνωστό ως chatterbot) [4] είναι μια εφαρμογή λογισμικού ή μια διεπαφή ιστού που έχει σχεδιαστεί για να πραγματοποιεί συνομιλίες με κείμενο ή φωνή. Τα σύγχρονα chatbots είναι συνήθως προσβάσιμα μέσω διαδικτύου και χρησιμοποιούν συστήματα γενετικής τεχνητής νοημοσύνης που είναι ικανά να διατηρούν μια συνομιλία με έναν χρήστη σε φυσική γλώσσα και να προσομοιώνουν τον τρόπο με τον οποίο ένας άνθρωπος θα συμπεριφερόταν ως συνομιλητής. Τέτοια chatbots χρησιμοποιούν βαθιά μάθηση και επεξεργασία φυσικής γλώσσας. Απλούστερα chatbots υπάρχουν εδώ και δεκαετίες.

Ένα από τα πρώτα και πιο γνωστά παραδείγματα είναι ο εικονικός συνομιλητής **ELIZA**, ο οποίος αναπτύχθηκε τη δεκαετία του 1960 και βασιζόταν σε απλούς κανόνες αντιστοίχισης προτύπων. Παρότι η λειτουργικότητά του ήταν περιορισμένη, αποτέλεσε τη βάση για τη μετέπειτα εξέλιξη των συστημάτων διαλόγου.



Σχήμα 1.1 : Το chatbot ELIZA του 1966

Πηγή : [4]

Η βασική μέθοδος λειτουργίας του **ELIZA** περιλάμβανε την αναγνώριση λέξεων ή φράσεων-κλειδιών στην είσοδο και την έξοδο των αντίστοιχων προ-προετοιμασμένων ή προ-προγραμματισμένων απαντήσεων που μπορούν να προωθήσουν τη συζήτηση με έναν φαινομενικά ουσιαστικό τρόπο. Έτσι δημιουργούσε την ψευδαίσθηση της κατανόησης, παρόλο που η επεξεργασία που γινόταν ήταν απλώς επιφανειακή. Το ELIZA έδειξε ότι μια τέτοια ψευδαίσθηση είναι εκπληκτικά εύκολο να δημιουργηθεί,

επειδή οι άνθρωποι είναι πρόθυμοι να δώσουν το πλεονέκτημα της αμφιβολίας όταν οι απαντήσεις σε μια συνομιλία μπορούν να ερμηνευθούν ως «έξυπνες» .

Η αυξανόμενη διάδοση των chatbots συνδέεται άμεσα με τη ραγδαία άνθηση της τεχνητής νοημοσύνης κατά τη δεκαετία του 2020, καθώς και με την καθιέρωση της αρχιτεκτονικής των Μετασχηματιστών (Transformers). Οι εξελίξεις αυτές οδήγησαν στην ανάπτυξη του ChatGPT, ακολουθούμενο από ανταγωνιστές όπως το Gemini, Claude και αργότερα το Grok. Τα AI chatbots, δηλαδή οι εικονικοί συνομιλητές τεχνητής νοημοσύνης, συνήθως χρησιμοποιούν ένα βασικό μεγάλο γλωσσικό μοντέλο, το οποίο προσαρμόζεται για συγκεκριμένες χρήσεις [4].

Ένας από τους πλέον καθιερωμένους τομείς εφαρμογής των chatbots είναι η εξυπηρέτηση πελατών και η υποστήριξη, όπου έχουν χρησιμοποιηθεί διαχρονικά ως εικονικοί βοηθοί. Ιδιαίτερα διαδεδομένη εφαρμογή αποτελεί η αναζήτηση και παροχή πληροφορίας, αξιοποιώντας αυτοματοποιημένους μηχανισμούς διαλόγου για την αποτελεσματική κάλυψη πληροφοριακών αναγκών των χρηστών.

1.3 Περιγραφή του Προβλήματος

Οι μηχανές αναζήτησης αποτελούν διαχρονικά το βασικό εργαλείο εύρεσης πληροφοριών στο Διαδίκτυο. Ωστόσο, οι τυπικές αναζητήσεις είναι σύντομες, συχνά μία ή δύο λέξεις, οδηγώντας σε αποτελέσματα περιορισμένης συνάφειας. Η ανάκτηση πληροφοριών με βάση το συγκείμενο (Contextual Information Retrieval – CIR) είναι μια κρίσιμη τεχνική για αυτές τις μηχανές αναζήτησης, προκειμένου να διευκολύνουν τις αναζητήσεις και να επιστρέφουν σχετικές πληροφορίες [1]. Παρά τη σημασία της, έχει σημειωθεί μικρή πρόοδος στην συγκεκριμένη τεχνική λόγω της δυσκολίας αναπαράστασης και μοντελοποίησης της συμφραζόμενης πληροφορίας.

Η ιστορική εξέλιξη των chatbots, από τα πρώιμα συστήματα βασισμένα σε κανόνες όπως το ELIZA έως τα σύγχρονα συστήματα τεχνητής νοημοσύνης που αξιοποιούν μεγάλα γλωσσικά μοντέλα, ανέδειξε σταδιακά την ανάγκη για αποτελεσματικότερη πρόσβαση, κατανόηση και αξιοποίηση της πληροφορίας

Το πρόβλημα γίνεται εντονότερο στην περίπτωση μεγάλων εγγράφων σε μορφές όπως PDF ή DOCX, όπου η αναζήτηση συγκεκριμένης πληροφορίας είναι χρονοβόρα και απαιτεί χειροκίνητη πλοήγηση από τον χρήστη. Ιδιαίτερη πρόκληση αποτελεί και η υποστήριξη πολυγλωσσικού περιεχομένου, όπως κείμενα στην ελληνική γλώσσα, για τα οποία η ακρίβεια ανάκτησης πληροφορίας δεν είναι πάντα δεδομένη.

Στα πλαίσια αυτής της πρόκλησης συνέβαλλε ουσιαστικά η ραγδαία ανάπτυξη των Μεγάλων Γλωσσικών Μοντέλων (Large Language Models – LLMs), χρησιμοποιώντας αρχιτεκτονικές όπως οι "Transformers", οι οποίες τα κατέστησαν ικανά να μπορούν να εκτελέσουν ποικίλες εργασίες, όπως μετάφραση, σύνοψη κειμένων, απάντηση σε ερωτήσεις και δημιουργία περιεχομένου [2]. Παρά τις εντυπωσιακές δυνατότητες που επιδεικνύουν όμως, αντιμετωπίζουν προκλήσεις καθώς δεν διαθέτουν άμεση πρόσβαση σε εξωτερικές πηγές δεδομένων και βασίζονται αποκλειστικά στη γνώση που έχει αποκτηθεί κατά τη φάση της εκπαίδευσής τους. Ως λογικό επακόλουθο, υπόκεινται σε περιορισμούς, συμπεριλαμβανομένου του φαινομένου της ψευδαίσθησης — όπου τα μοντέλα παράγουν πληροφορίες που φαίνονται εύλογες αλλά είναι ανακριβείς ή κατασκευασμένες [3]. Αυτές οι ψευδαισθήσεις μπορεί να προκύψουν λόγω προκαταλήψεων στα δεδομένα εκπαίδευσης, περιορισμών του μοντέλου ή της εγγενούς πολυπλοκότητας της ανθρώπινης γλώσσας.

Η τεχνολογία RAG (Retrieval-Augmented Generation) έχει αναδειχθεί ως μια πολλά υποσχόμενη λύση, ενσωματώνοντας γνώσεις από εξωτερικές βάσεις δεδομένων. Η ενσωμάτωσή της στα LLMs αποτελεί ευρεία υιοθέτηση, καθιστώντας την βασική τεχνολογία, στην προώθηση των chatbots. Η αιτία είναι ότι

προσφέρει ανάκτηση σχετικών τμημάτων εγγράφων, μέσω υπολογισμού σημασιολογικής ομοιότητας, μειώνοντας ουσιαστικά το πρόβλημα της δημιουργίας περιεχομένου που είναι αντικειμενικά λανθασμένο.

1.4 Σκοπός της Πτυχιακής Εργασίας

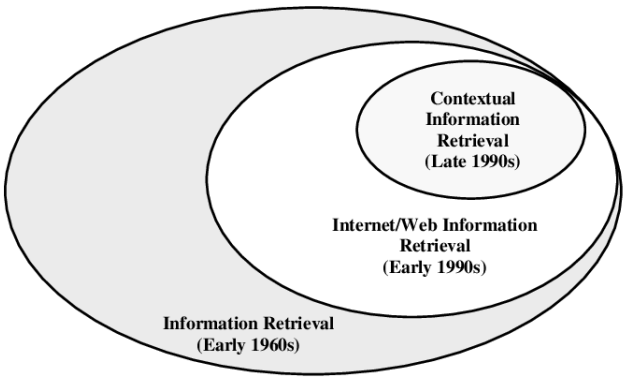
Ο σκοπός της παρούσας πτυχιακής εργασίας είναι η σχεδίαση και η ανάπτυξη ενός σύγχρονου chatbot βασισμένου στην αρχιτεκτονική **Retrieval-Augmented Generation (RAG)**, η οποία επιτρέπει την ακριβή και τεκμηριωμένη απάντηση ερωτήσεων, προεκπαιδευμένο σε έγγραφα του ελληνικού δημόσιου τομέα.

Το σύστημα που υλοποιήθηκε :

- Υποστηρίζει την επεξεργασία και αποθήκευση ελληνικών εγγράφων από το δημόσιο τομέα.
- Αξιοποιεί σύγχρονες τεχνικές αναπαράστασης κειμένου μέσω embeddings.
- Εφαρμόζει αναζήτηση πληροφορίας με τη χρήση διανυσματικών βάσεων και τεχνικών σημασιολογικής ομοιότητας.
- Περιορίζει την παραγωγή ανακριβών απαντήσεων μέσω τεχνικών βελτίωσης της ακρίβειας που του έχουν επιβληθεί, οι οποίες θα αναλυθούν στο 2ο κεφάλαιο.
- Επιστρέφει την απάντηση μαζί με το έγγραφο από το οποίο προήλθε, με κλήση γλωσσικού μοντέλου που καλείται να απαντήσει, μέσω prompt που έχουμε ορίσει, αποκλειστικά βάσει των ανακτηθέντων αποσπασμάτων από τα σχετικά έγγραφα.
- Πιστοποιεί τους χρήστες που εισέρχονται στο σύστημα.

1.4.1 Αναζήτηση και Ανάκτηση Πληροφορίας (Information Retrieval)

Η αναζήτηση πληροφορίας (IR) είναι ένας ευρύς, συχνά αόριστα καθορισμένος όρος. Οι πρώτες προσπάθειες ανάκτησης πληροφοριών από το Διαδίκτυο/Ιστό χρησιμοποιούσαν μηχανές αναζήτησης που ευρετηρίαζαν ένα τμήμα του εγγράφου ή της ιστοσελίδας, με απλές τεχνικές αντιστοίχισης λέξεων-κλειδιών και στατική αντιστοίχιση όρων [5].



Σχήμα 1.2 : Η εξέλιξη της αναζήτησης πληροφορίας

Πηγή : [1]

Η πρώτη τέτοια εφαρμογή ήταν ο **Archie** (1990), που ευρετηρίαζε αρχεία στα πρωτόκολλα FTP, επιτρέποντας στους χρήστες να βρίσκουν διαθέσιμα αρχεία χωρίς να γνωρίζουν τις ακριβείς URL διευθύνσεις τους. Κατά τη διάρκεια της δεκαετίας του ’90, εμφανίστηκαν εργαλεία όπως το

WebCrawler (1994), το οποίο για πρώτη φορά ευρετηρίαζε και επέτρεπε αναζήτηση ολόκληρου του περιεχομένου μιας ιστοσελίδας, αντί μόνο των τίτλων ή URL [5].

Στις τελευταίες δεκαετίες, η τεχνολογία των μηχανών αναζήτησης εξελίχθηκε καταλυτικά μεταβαίνοντας από τη λεξικογραφική αναζήτηση στη σημασιολογική αναζήτηση. Οι σύγχρονες μηχανές αναζήτησης αξιοποιούν πλέον προηγμένους αλγόριθμους και τεχνικές όπως η ανάλυση συνάφειας, η εκπαίδευση μέσω νευρωνικών δικτύων και οι διανυσματικές βάσεις δεδομένων, όπως το **FAISS**, προκειμένου να προσφέρουν πιο ακριβείς και εξατομικευμένες απαντήσεις σε μεγάλους όγκους δεδομένων.

Η σύγχρονη τάση επικεντρώνεται στη σύγκλιση των συστημάτων αναζήτησης πληροφορίας με εικονικούς συνομιλητές. Στην προσπάθεια να γεφυρωθεί το χάσμα μεταξύ παραδοσιακών μηχανών αναζήτησης και σύγχρονων γλωσσικών μοντέλων, η αρχιτεκτονική RAG αποτελεί χαρακτηριστικό παράδειγμα, καθώς συνδυάζει την ακρίβεια της ανάκτησης πληροφορίας με την παραγωγή απαντήσεων σε φυσική γλώσσα, στοιχείο που καθίσταται πλέον αναγκαίο λόγω του συνεχώς αυξανόμενου όγκου πληροφορίας, της πολυπλοκότητας των πληροφοριακών αναγκών των χρηστών και της απαίτησης για άμεσες, αξιόπιστες και κατανοητές απαντήσεις.

1.5 Διάρθρωση της αναφοράς

Η διάρθρωση της παρούσας πτυχιακής εργασίας είναι η εξής:

1. **Κεφάλαιο 1 :** Πραγματοποιείται μια γενική αναφορά της σημασίας και της εξέλιξης των εικονικών συνομιλητών καθώς και της αναζήτησης πληροφορίας. Επίσης περιγράφεται το πρόβλημα της εργασίας, που πρόκειται να επιλυθεί.
2. **Κεφάλαιο 2 :** Γίνεται ανασκόπηση της ερευνητικής περιοχής που αφορά τα μεγάλα γλωσσικά μοντέλα (LLMs), την ανάκτηση πληροφορίας αλλά και την τεχνολογία παραγωγής πληροφορίας με ενισχυμένη ανάκτηση (RAG).
3. **Κεφάλαιο 3 :** Πλήρης περιγραφή των υλοποιήσεων και παραδοχών που έγιναν, καθώς και οι αρχιτεκτονικές που ακολουθήθηκαν για τη λειτουργία και τα πειράματα του εικονικού συνομιλητή.
4. **Κεφάλαιο 4 :** Παρουσιάζεται αναλυτικά η μεθοδολογία των πειραμάτων και τα αποτελέσματά τους.
5. **Κεφάλαιο 5 :** Παρουσιάζονται τα τελικά συμπεράσματα, τα προβλήματα που προέκυψαν κατά την εκπόνηση της εργασίας και προτείνονται θέματα για μελλοντική μελέτη, αλλαγές και επεκτάσεις.

1.6 Επίλογος

Στο 1ο κεφάλαιο παρουσιάστηκε το βασικό πρόβλημα που πραγματεύεται η πτυχιακή εργασία, καθώς και το θεωρητικό και ιστορικό πλαίσιο μέσα στο οποίο εντάσσεται. Αναδείχθηκαν οι περιορισμοί διαφόρων προσεγγίσεων στην αναζήτηση πληροφορίας, καθώς και η ανάγκη για συνδυαστικές μεθόδους που εξασφαλίζουν ακρίβεια και αξιοπιστία στις παρεχόμενες απαντήσεις, όπως η τεχνολογία RAG.

Τα ζητήματα που παρουσιάστηκαν στο κεφάλαιο αυτό αποτελούν τη βάση για την ανάλυση που ακολουθεί, καθώς στο επόμενο κεφάλαιο παρουσιάζεται το θεωρητικό υπόβαθρο και οι βασικές τεχνολογίες στις οποίες στηρίζεται το προτεινόμενο σύστημα που δημιουργήσαμε.

Κεφάλαιο 2ο: Επισκόπηση της ερευνητικής περιοχής

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναλυθεί το βασικό θεωρητικό υπόβαθρο το οποίο κρίνεται ως απαραίτητη γνώση, ώστε να βοηθήσει τον αναγνώστη στην κατανόηση των αλγορίθμων και των αρχιτεκτονικών που έχουν χρησιμοποιηθεί στην παρούσα πτυχιακή εργασία, για την υλοποίηση του συστήματος. Παρουσιάζονται οι σύγχρονες τεχνολογίες (state-of-the-art) στον τομέα των γλωσσικών μοντέλων, των διανυσματικών βάσεων, των αναπαραστάσεων κειμένου και πραγματοποιείται μια ανάλυση των χρήσιμων εννοιών που πραγματεύεται η πτυχιακή εργασία αλλά και των μοντέλων που χρησιμοποιήθηκαν.

2.2 Αναπαράσταση Κειμένου και Ανάκτηση Πληροφορίας

Η αναπαράσταση κειμένου καθώς και η ανάκτηση πληροφορίας, αποτελούν θεμελιώδη στοιχεία για τη δημιουργία ενός αξιόπιστου συστήματος ερώτησης-απάντησης. Κεντρικό ρόλο στη διαδικασία της ανάκτησης κειμένου έχουν μετρικές που βασίζονται στη **σημασιολογική ομοιότητα**, οι οποίες επιτρέπουν την αντιστοίχιση tokens με βάση το περιεχόμενο και όχι αποκλειστικά με βάση τη λεξιλογική τους μορφή.

Η Σημασιολογική Ομοιότητα (Σ.Ο.) είναι το μέτρο της εννοιολογικής «απόστασης» μεταξύ δύο λεκτικών όρων (λέξεις, φράσεις, κείμενα), λαμβάνοντας υπόψη την αντιστοιχία της σημασίας τους. Διαφορετικά, η Σ.Ο. εκφράζει το βαθμό ομοιότητας δύο λεκτικών όρων, ως συνάρτηση απόστασης, η οποία αποτυπώνει τη σημασιολογική τους εγγύτητα και βασίζεται σε ποικίλες πηγές πληροφοριών. Χαρακτηριστικά παραδείγματα τέτοιων πηγών, είναι τα σημασιολογικά δίκτυα, οι μηχανές αναζήτησεις, και η Wikipedia [6].

Αντί της παραδοσιακής ανάκτησης πληροφορίας μέσω λέξεων-κλειδιών, στην παρούσα πτυχιακή εργασία, η σημασιολογική αναζήτηση υλοποιείται μέσω διανυσματικών αναπαραστάσεων κειμένου (embeddings) και τεχνικών διανυσματικής ανάκτησης. Οι διανυσματικές αναπαραστάσεις επιτρέπουν τη σύγκριση κειμένων, καθιστώντας δυνατή την ανάκτηση σχετικών αποσπασμάτων ακόμη και όταν δεν υπάρχει ακριβής λεξιλογική ταύτιση. Η προσέγγιση αυτή αποτελεί τη βάση της σημασιολογικής αναζήτησης και χρησιμοποιείται εκτενώς σε σύγχρονες εφαρμογές RAG.

2.2.1 Διανυσματικές Αναπαραστάσεις Κειμένου (Text Embeddings)

Τα embeddings αποτελούν έναν τρόπο αναπαράστασης κειμένου, εικόνας ή ήχου, ως σημεία σε έναν συνεχή διανυσματικό χώρο, όπου οι θέσεις αυτών των σημείων στο χώρο έχουν σημασιολογική σημασία. Ουσιαστικά αναπαρίστανται ως πυκνά αριθμητικά διανύσματα σε έναν n-διάστατο χώρο, όπου ο αριθμός των διαστάσεών του, εξαρτάται από την πολυπλοκότητα των δεδομένων εισόδου.

Στην περίπτωση που τα embeddings αναπαριστούν κείμενο, το αρχικό κείμενο μετατρέπεται πρώτα σε tokens και στη συνέχεια κάθε token αντιστοιχίζεται σε ένα διάνυσμα αριθμών, το οποίο αποτελεί το embedding του. Μέσω αυτής της αναπαράστασης, τα κείμενα με παρόμοιο νόημα, τοποθετούνται κοντά μεταξύ τους στον διανυσματικό χώρο. Η απόσταση, μεταξύ των διανυσμάτων, μπορεί να μετρηθεί με

διαφορετικές μετρικές, όπως η ευκλείδεια απόσταση ή η ομοιότητα συνημιτόνου (cosine similarity), ανάλογα με την εφαρμογή και τη μορφή των δεδομένων.

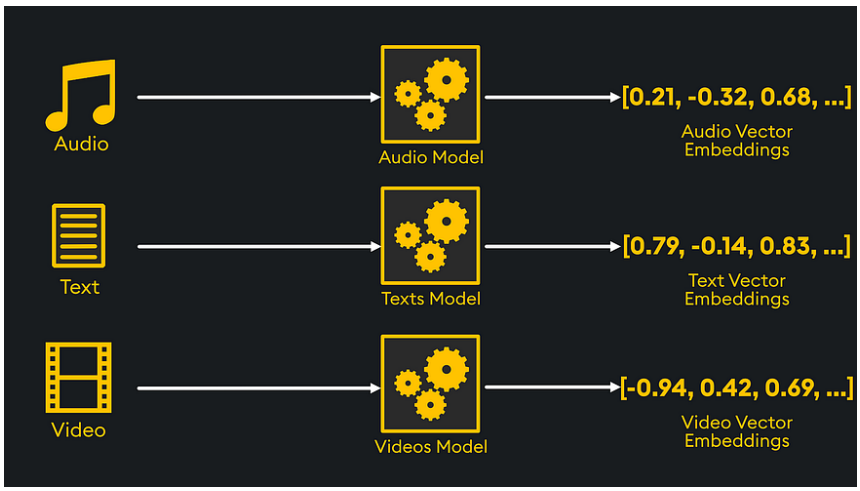
Τα embeddings μπορούν να διακριθούν σε διαφορετικές κατηγορίες, ανάλογα με το είδος της πληροφορίας που κωδικοποιούν και το επίπεδο στο οποίο εφαρμόζονται:

- **Word embeddings :** Αναπαριστούν μεμονωμένες λέξεις ως διανύσματα (διανυσματικές αναπαραστάσεις λέξεων πραγματικών αριθμών), με στόχο την αποτύπωση της σημασιολογικής εγγύτητας μεταξύ λέξεων (meaningful space).

- **Sentence/Document embeddings :** Αναπαριστούν προτάσεις παραγράφους ή ολόκληρα κείμενα ως ένα ενιαίο διάνυσμα. Η κωδικοποίηση γίνεται με μοντέλα νευρωνικών δικτύων, με σκοπό την εκπαίδευση τους στο συνδυασμό των επιμέρους word embeddings [6].

- **Image embeddings :** Χρησιμοποιούνται στην υπολογιστική όραση για την αναπαράσταση εικόνων ως διανυσμάτων.

- **Graph embeddings :** Αναπαριστούν κόμβους ή δομές γράφων ως διανύσματα, διατηρώντας πληροφορία σχετικά με τη συνδεσιμότητα και τη δομή [7].



Σχήμα 2.1 : Τύποι Embeddings

Πηγή :[7]

Στην παρούσα εργασία, η κύρια έμφαση δίνεται στα **sentence/document embeddings**, καθώς αποτελούν το βασικό μηχανισμό σημασιολογικής αναπαράστασης των αποσπασμάτων των εγγράφων και επιτρέπουν την αποτελεσματική ανάκτηση σχετικού περιεχομένου στο πλαίσιο της αρχιτεκτονικής RAG. Κατά την υλοποίηση του συστήματος υιοθετήθηκε προσέγγιση πολλαπλών μοντέλων (multi-embedding), ώστε να αξιολογηθούν διάφορες επιλογές ως προς την ακρίβεια ανάκτησης, την υποστήριξη της ελληνικής γλώσσας και το υπολογιστικό κόστος.

Τα μοντέλα που εξετάστηκαν είναι τα ακόλουθα:

1. RoBERTa (Encoder-based μοντέλο)

Το RoBERTa είναι μοντέλο τύπου encoder, σχεδιασμένο για εργασίες κατανόησης φυσικής γλώσσας. Μπορεί να χρησιμοποιηθεί για παραγωγή embeddings και υπολογισμό σημασιολογικής ομοιότητας μεταξύ κειμένων. Ωστόσο, τα encoder-based μοντέλα, όπως το RoBERTa, παρουσιάζουν αυξημένες απαιτήσεις σε CPU και μνήμη, ειδικά κατά τη μαζική παραγωγή embeddings σε μεγάλο αριθμό

αποσπασμάτων. Αυτό μπορεί να επηρεάσει τον χρόνο ευρετηρίασης και την απόδοση του συστήματος σε περιβάλλοντα χωρίς GPU [8].

2. intfloat/multilingual-e5-base

Το E5 αποτελεί μοντέλο ειδικά σχεδιασμένο για εργασίες ανάκτησης πληροφορίας και ακολουθεί την πρακτική διάκρισης μεταξύ ερωτήματος (query) και αποσπάσματος (passage). Η διαφοροποίηση αυτή οδηγεί σε embeddings που είναι κατάλληλα για αντιστοίχιση ερωτήσεων με σχετικά κείμενα, βελτιώνοντας την αποτελεσματικότητα σε σενάρια retrieval [9].

3. Alibaba-NLP/gte-multilingual-base

Το GTE αποτελεί πολυγλωσσικό μοντέλο embeddings γενικής χρήσης, το οποίο στοχεύει σε ισορροπία μεταξύ ποιότητας αναπαράστασης και αποδοτικότητας. Η πολυγλωσσική του φύση το καθιστά κατάλληλο για περιβάλλοντα όπου απαιτείται ανθεκτικότητα σε διαφορετικές διατυπώσεις και τύπους κειμένου [10].

4. BAAI/bge-m3

Το BGE-M3 αποτελεί σύγχρονο πολυγλωσσικό μοντέλο embeddings, βελτιστοποιημένο για σημασιολογική αναζήτηση και ανάκτηση πληροφορίας. Προσφέρει υψηλή ακρίβεια αντιστοίχισης ερωτήματος–αποσπάσματος, βελτιώνοντας επίσης την αποτελεσματικότητα σε σενάρια retrieval [11].

2.2.2 Μέτρα Ομοιότητας και Διανυσματική Αναζήτηση

Αφού τα κείμενα (τόσο τα αποθηκευμένα έγγραφα όσο και τα ερωτήματα των χρηστών) μετατραπούν σε embeddings, περιλαμβάνοντας έτσι, σημασιολογική πληροφορία, χρειάζεται ένας τρόπος για να μετρηθεί η μεταξύ τους ομοιότητα. Ένας αλγόριθμος που είναι κατάλληλος για την εκπαίδευση ενός μοντέλου για σύγκριση εγγράφων δεδομένων είναι ο αλγόριθμος ομοιότητας συνημιτόνου (cosine similarity). Η ομοιότητα συνημιτόνου μετρά τη γωνιακή εγγύτητα μεταξύ δύο μη μηδενικών διανυσμάτων και χρησιμοποιείται ευρέως σε εφαρμογές σημασιολογικής αναζήτησης.

Ομοιότητα Συνημιτόνου : Η ομοιότητα συνημιτόνου ορίζεται ως το συνημίτονο της γωνίας θ ανάμεσα στα διανύσματα. Ή εναλλακτικά, το συνημίτονο δύο διανυσμάτων ορίζεται ως το εσωτερικό τους γινόμενο, διαιρούμενο από το γινόμενο των μέτρων τους :

$$sim\,(A,\,B) \;=\; cos(\theta) \;=\; \frac{A \cdot B}{||A||\,||B||} \tag{2.1}$$

Οι τιμές που κυμαίνεται η συγκεκριμένη μετρική είναι από -1 (εντελώς αντίθετα) έως 1 (εντελώς όμοια). Πιο συγκεκριμένα αν δύο διανύσματα είναι παράλληλα, δηλαδή σχηματίζουν γωνία 0 μοίρες, έχουν cos(0)=1 και θεωρούνται όμοια. Διαφορετικά, αν είναι κάθετα, δηλαδή σχηματίζουν 90 μοίρες, cos(90)=0, υποδηλώνεται έλλειψη ομοιότητας και αν σχηματίζουν μεταξύ τους γωνία 180 μοιρών, δηλαδή cos(180)= -1, τότε δεν έχουν καμία ομοιότητα. [6].

Η ομοιότητα συνημιτόνου είναι ιδιαίτερα χρήσιμη διότι δεν επηρεάζεται από το μέτρο (μήκος) των διανυσμάτων αλλά από την κατεύθυνσή τους, γεγονός που την καθιστά κατάλληλη για σύγκριση embeddings.

Η διανυσματική αναζήτηση αποτελεί κρίσιμο βήμα για την αρχιτεκτονική Retrieval-Augmented Generation, καθώς η ποιότητα των αποσπασμάτων που ανακτώνται επηρεάζει άμεσα την ποιότητα και την τεκμηρίωση της τελικής απάντησης.

2.2.3 Διανυσματικές Βάσεις Δεδομένων και FAISS

Δεδομένου ότι οι συλλογές εγγράφων σε πραγματικά συστήματα μπορεί να περιλαμβάνουν μεγάλο πλήθος κειμενικών τμημάτων, η αποδοτική ανάκτηση των πλέον σχετικών αποσπασμάτων, η εφαρμογή διανυσματικής αναζήτησης δηλαδή, απαιτεί εξειδικευμένους μηχανισμούς αποθήκευσης και ανάκτησης μεγάλου αριθμού embeddings.

Καθώς ο αριθμός των αποσπασμάτων αυξάνεται, η απλή γραμμική αναζήτηση (brute force search) καθίσταται υπολογιστικά δαπανηρή, με αποτέλεσμα να απαιτούνται δομές ευρετηρίασης δεδομένων και βιβλιοθήκες. Για τον σκοπό αυτό αξιοποιούνται δομές και αλγόριθμοι προσέγγισης του πλησιέστερου γείτονα (Approximate Nearest Neighbor – ANN), οι οποίοι επιτρέπουν την κλιμακούμενη αναζήτηση σε μεγάλους διανυσματικούς χώρους με χαμηλό υπολογιστικό κόστος. Στην παρούσα εργασία, η διαδικασία αυτή υλοποιείται μέσω της βιβλιοθήκης **FAISS** (Facebook AI Similarity Search), η οποία έχει σχεδιαστεί για αναζήτηση διανυσματικής ομοιότητας σε μεγάλη κλίμακα.

Η FAISS δέχεται ως είσοδο ένα σύνολο embeddings (π.χ. αποσπάσματα εγγράφων) και κατασκευάζει ένα index, το οποίο επιτρέπει γρήγορη αναζήτηση των πλησιέστερων γειτόνων για κάθε νέο ερώτημα. Στην περίπτωση ακριβούς αναζήτησης, η FAISS μπορεί να υπολογίσει τα αποτελέσματα με υψηλή ακρίβεια, ενώ σε προσεγγιστικές μεθόδους μπορεί να προσφέρει σημαντικά ταχύτερη απόκριση, με πιθανό μικρό κόστος στην ποιότητα των αποτελεσμάτων [12].

Η ενσωμάτωση της διανυσματικής ανάκτησης σε ένα σύστημα ερώτησης–απάντησης αποτελεί επίσης βασικό στοιχείο της αρχιτεκτονικής RAG, καθώς τα αποσπάσματα που ανακτώνται μέσω της διανυσματικής αναζήτησης ενσωματώνονται στη συνέχεια στην είσοδο του μεγάλου γλωσσικού μοντέλου, που έχει επιλεγεί, ώστε η παραγόμενη απάντηση να βασίζεται σε τεκμηριωμένη πληροφορία, μειώνοντας την πιθανότητα παραγωγής ανακριβών ή μη επαληθεύσιμων αποτελεσμάτων.

2.3 Μεγάλα Γλωσσικά Μοντέλα (LLMs)

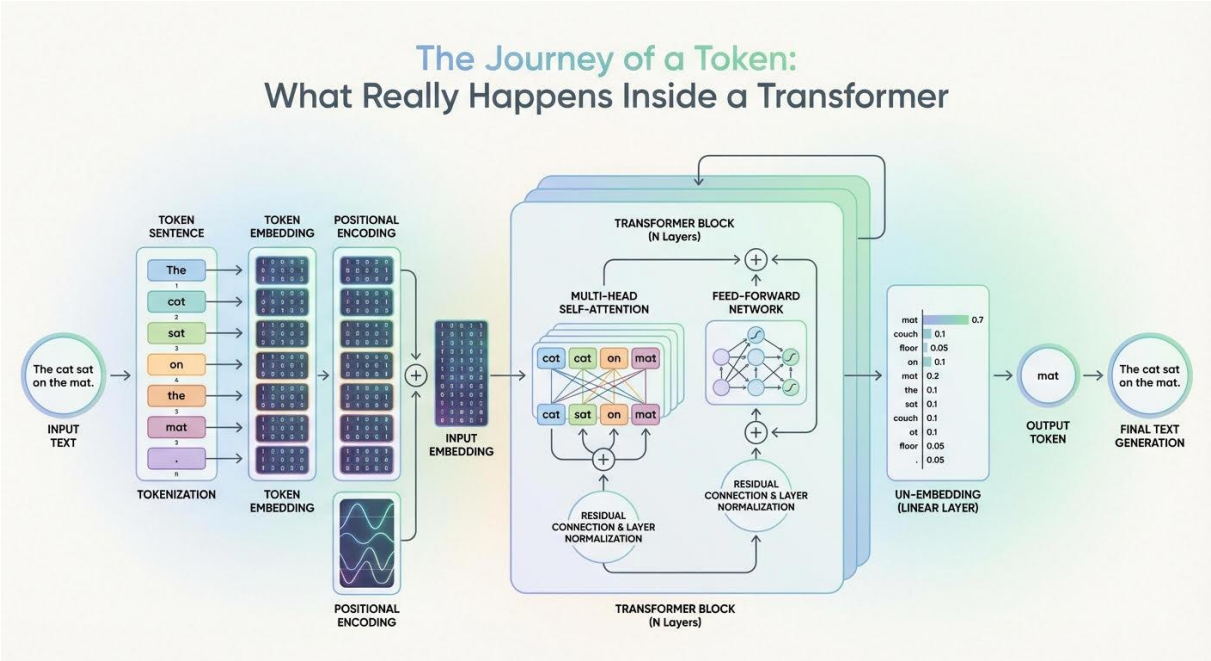
Τα μεγάλα γλωσσικά μοντέλα, σύμφωνα με την IBM, αποτελούν έναν τύπο τεχνητής νοημοσύνης (AI), συγκεκριμένα μία κατηγορία μοντέλων βαθιάς μάθησης που έχουν εκπαιδευτεί σε μεγάλο όγκο γλωσσικών δεδομένων, καθιστώντας τα ικανά να κατανοούν και να παράγουν φυσική γλώσσα και άλλους τύπους περιεχομένου για την εκτέλεση ενός ευρέος φάσματος εργασιών. Τα LLMs βασίζονται σε έναν τύπο αρχιτεκτονικής νευρωνικού δικτύου που ονομάζεται μετασχηματιστής, ο οποίος υπερέχει στη διαχείριση ακολουθιών λέξεων και στην καταγραφή μοτίβων σε κείμενα [2]. Η λειτουργικότητα των μεγάλων γλωσσικών μοντέλων εκτείνεται σε διάφορες εφαρμογές.

Σχήμα 2.2 : Βασικά παραδείγματα χρήσης των Large Language Models

Πηγή : [13]

Τα LLMs, μετά την ολοκλήρωση της εκπαίδευσής τους, μπορούν να αξιοποιηθούν σε ένα ευρύ φάσμα γνωστικών και παραγωγικών εργασιών, όπως απεικονίζεται και στο Σχήμα 2.2. Ενδεικτικά παραδείγματα των εργασιών αυτών, είναι η παραγωγή κειμένου βάση προτροπών (prompts), η περίληψη ενός άρθρου, ο εντοπισμός σφαλμάτων σε κώδικα ή η σύνταξη εξειδικευμένων κειμένων όπως μια νομική ρήτρα. Όταν τους δοθούν ικανότητες δράσης (agents), μπορούν να εκτελέσουν, με διαφορετικούς βαθμούς αυτονομίας, ποικίλες εργασίες όπως η αλληλεπίδραση με εξωτερικά εργαλεία, βάσεις δεδομένων ή προγραμματιστικά περιβάλλοντα.

Τα μεγάλα γλωσσικά μοντέλα λειτουργούν ως κωδικοποιητές (encoders), αποκωδικοποιητές (decoders) ή συνδυασμοί αυτών. Το εισερχόμενο κείμενο, κατατμηματίζεται σε μονάδες κειμένου (tokens), οι οποίες στη συνέχεια κωδικοποιούνται σε διανύσματα χώρου υψηλής διαστασιμότητας. Τα tokens αντιστοιχούν σε μικρότερες μονάδες, όπως λέξεις, υπολέξεις ή ακόμη και μεμονωμένους χαρακτήρες. Η διαδικασία αυτή τυποποιεί τη γλωσσική είσοδο, επιτρέποντας στα μοντέλα να διαχειρίζονται με συνέπεια τόσο σπάνια, όσο και νέα λεκτικά στοιχεία. Με τον τρόπο αυτό, διασφαλίζεται η γενικευσιμότητα του μοντέλου και η αποτελεσματική επεξεργασία νέων ή άγνωστων λέξεων. Βασικός στόχος των μοντέλων κατά την εκπαίδευση, είναι η πρόβλεψη της επόμενης λέξης ή του επόμενου token σε μία δεδομένη ακολουθία λέξεων. Φυσικά τα βάρη του μοντέλου προσαρμόζονται (fine tuning), ώστε να ελαχιστοποιηθεί η απόσταση των διανυσμάτων του δοθέντος και του προβλεπόμενου κειμένου. Η ελαχιστοποίηση της απόστασης επιτυγχάνεται από το μοντέλο μέσω της εύρεσης γλωσσικών μοτίβων και επαναλαμβανόμενων δομών/σχέσεων στα δεδομένα.



Σχήμα 2.3 : Ενδεικτικό διάγραμμα ροής της λειτουργίας ενός μεγάλου γλωσσικού μοντέλου (LLM), από την είσοδο κειμένου έως την παραγωγή εξόδου.

Πηγή : [14]

Όσον αφορά την αρχιτεκτονική που τα διέπει, τα LLMs βασίζονται στους μετασχηματιστές (transformers), οι οποίοι έχουν επιφέρει επανάσταση στον τομέα της επεξεργασίας φυσικής γλώσσας,

μέσω της ικανότητάς τους να μοντελοποιούν εξαρτήσεις μεταξύ λέξεων, ανεξάρτητα από την απόστασή τους στο κείμενο. Η ιδιότητα αυτή επιτυγχάνεται μέσω μηχανισμών προσοχής (attention mechanisms), που επιτρέπουν στο μοντέλο να εστιάζει δυναμικά σε διαφορετικά μέρη του εισερχόμενου κειμένου, ανάλογα με το εκάστοτε γλωσσικό και σημασιολογικό πλαίσιο [15].

Παρά τις εντυπωσιακές δυνατότητές τους, τα LLMs αντιμετωπίζουν αρκετούς περιορισμούς, ιδιαίτερα σε σενάρια που αφορούν συγκεκριμένους τομείς ή απαιτούν εκτεταμένες γνώσεις. Τα προ-εκπαιδευμένα μοντέλα συχνά αντιμετωπίζουν δυσκολίες, όπως αναφέραμε και νωρίτερα, όταν χειρίζονται εισόδους που αποκλίνουν από τα δεδομένα εκπαίδευσής τους. Γεγονός που οδηγεί στο φαινόμενο της ψευδαίσθησης, κατά το οποίο το μοντέλο παράγει περιεχόμενο που είναι εύλογο αλλά στην πραγματικότητα είναι λανθασμένο ή πλασματικό.

Η τρέχουσα κατάσταση ωστόσο, της έρευνας των μεγάλων γλωσσικών μοντέλων χαρακτηρίζεται από ραγδαίες εξελίξεις και σημαντικές συνεισφορές από τεχνολογικούς οργανισμούς, όπως η OpenAI, η Google, η Anthropic και η Meta, οι οποίοι συνεχώς διευρύνουν τα όρια της επεξεργασίας φυσικής γλώσσας αναπτύσσοντας όλο και πιο εξελιγμένα μοντέλα. Χαρακτηριστικό παράδειγμα αποτελεί το μοντέλο **Gemini 2.5 Flash**, το οποίο συνδυάζει προηγμένες δυνατότητες κατανόησης και παραγωγής φυσικής γλώσσας με χαμηλή καθυστέρηση και μειωμένες υπολογιστικές απαιτήσεις. Το μοντέλο αυτό αποτελεί δείγμα της σύγχρονης τάσης μοντέλων που προσφέρουν ισορροπία μεταξύ ταχύτητας, τιμής και ακρίβειας απαντήσεων [16].

2.3.1 Gemini 2.5 Flash

Στο πλαίσιο της παρούσας εργασίας, για την παραγωγή των απαντήσεων αξιοποιείται το γλωσσικό μοντέλο Gemini 2.5 Flash, της σειράς Gemini 2.X, το οποίο παρέχεται από την εταιρεία Google. Η σειρά μοντέλων Gemini 2.X έχει σχεδιαστεί με πολυτροπικό χαρακτήρα (natively multimodal), υποστηρίζοντας εισόδους με μήκος που υπερβαίνει τις ένα εκατομμύριο μονάδες κειμένου (tokens).

	Gemini 1.5 Flash	*Gemini 1.5 Pro*	**Gemini 2.0 Flash-Lite**	**Gemini 2.0 Flash**	**Gemini 2.5 Flash**	**Gemini 2.5 Pro**
Input modalities	Text, Image, Video, Audio	Text, Image, Video, Audio	Text, Image, Video, Audio	Text, Image, Video, Audio	Text, Image, Video, Audio	Text, Image, Video, Audio
Input length	1M	2M	1M	1M	1M	1M
Output modalities	Text	Text	Text	Text, Image*	Text, Audio*	Text, Audio*
Output length	8K	8K	8K	8K	64K	64K
Thinking	No	No	No	Yes*	Dynamic	Dynamic
Supports tool use?	No	No	No	Yes	Yes	Yes
Knowledge cutoff	November 2023	November 2023	June 2024	June 2024	January 2025	January 2025

Σχήμα 2.3 : Σύγκριση των μοντέλων της σειράς Gemini 2.X με τα μοντέλα Gemini 1.5 Pro και Flash

Πηγή : [17]

Το συγκεκριμένο μοντέλο επιλέχθηκε όχι μόνο λόγω της ικανότητάς του να επεξεργάζεται εκτενή κείμενα αλλά και να υποστηρίζει αποτελεσματικά πολύγλωσσο περιεχόμενο. Παράλληλα ενσωματώνει μηχανισμό ελεγχόμενου προϋπολογισμού σκέψης για τις πιο σύνθετες εργασίες, με αποτέλεσμα να

επιβάλει την ισορροπία μεταξύ ποιότητας, κόστους και καθυστέρησης. Τέλος υποστηρίζει χρήση εξωτερικών εργαλείων, όπως διανυσματικές βάσεις δεδομένων και συστήματα αναζήτησης.

Το Gemini χρησιμοποιείται αποκλειστικά στο στάδιο της παραγωγής της απάντησης και λαμβάνει ως είσοδο ένα αυστηρά καθορισμένο πλαίσιο, μαζί με κάποιο prompt που έχει επιλεχθεί από το χρήστη, το οποίο περιλαμβάνει μόνο τα αποσπάσματα που ανακτήθηκαν από το σύστημα ανάκτησης πληροφορίας. Με τον τρόπο αυτό, το γλωσσικό μοντέλο λειτουργεί ως μηχανισμός σύνθεσης και διατύπωσης της απάντησης, χωρίς να εισάγει εξωτερική γνώση.

Το σύστημα ανάκτησης της πληροφορίας που αντιμετωπίζει τους περιορισμούς/ψευδαισθήσεις των LLMs, είναι η αρχιτεκτονική RAG, η οποία επιτρέπει τη δυναμική ανάκτηση πληροφορίας από εξωτερικές βάσεις δεδομένων, μέσω μετρικών ομοιότητας, περιορίζοντας την εμφάνιση ανακριβών ή μη τεκμηριωμένων αποτελεσμάτων.

2.3.2 Meltemi-7B

Το Meltemi ανήκει στην οικογένεια μοντέλων προσανατολισμένων στην ελληνική γλώσσα, με στόχο την καλύτερη κάλυψη μορφολογικών και λεξιλογικών ιδιαιτεροτήτων. Αναπτύχθηκε από το Ινστιτούτο Επεξεργασίας του Λόγου (ΙΕΛ) του Ερευνητικού Κέντρου "Αθηνά" και η αξιοποίησή του είναι ιδιαίτερα χρήσιμη σε ελληνικά έγγραφα δημόσιου τομέα, όπου η ορολογία και η διατύπωση παρουσιάζουν συχνά αυξημένη πολυπλοκότητα.

Το Meltemi είναι χτισμένο πάνω στο LLM ανοιχτού λογισμικού της Mistral-7B και έχει εκπαιδευτεί πάνω σε ελληνικά κείμενα υψηλής ποιότητας. Η αρχική έκδοση εκπαιδεύτηκε σε ένα μεγάλο σώμα αγγλικού κειμένου και στην συνέχεια επεκτάθηκε η προπαίδευση του Mistral-7B με πρόσθετη επάρκεια στην ελληνική γλώσσα, αξιοποιώντας ένα μεγάλο σώμα που αποτελείται από περίπου 40 δισεκατομμύρια tokens. Αυτό το σώμα περιλαμβάνει 28,5 δισεκατομμύρια ελληνικά tokens, κατασκευασμένα από διαθέσιμους πόρους στο κοινό. Επιπλέον, για να διασφαλιστεί ότι το μοντέλο έχει δίγλωσσες δυνατότητες, χρησιμοποιήθηκαν επιπλέον υποσώματα με 10,5 δισεκατομμύρια tokens αγγλικών κειμένων και ένα παράλληλο σύνολο δεδομένων ελληνοαγγλικών 600 εκατομμυρίων tokens [18]. Αυτό το σώμα έχει υποβληθεί σε επεξεργασία, φιλτράρισμα και κατάργηση των αντιγράφων για τη διασφάλιση της ποιότητας των δεδομένων.

Ωστόσο , το συγκεκριμένο γλωσσικό μοντέλο παρουσιάζει υπολογιστικές δυσκολίες. Αναλυτικότερα, αντίστοιχα με τα υπόλοιπα μοντέλα τύπου encoder, η χρήση του απαιτεί αυξημένη επεξεργαστική ισχύ (κυρίως CPU), κάτι που μπορεί να επιβαρύνει την ταχύτητα επεξεργασίας σε μεγάλα σύνολα εγγράφων. Δεδομένου της περιορισμένης ισχύος που είχαμε στη διάθεσή μας, δεν μπορέσαμε να χρησιμοποιήσουμε το συγκεκριμένο μοντέλο στην υλοποίηση που ακολουθήθηκε.

2.4 Retrieval-Augmented Generation (RAG)

Η αρχιτεκτονική RAG συνδυάζει τις δυνατότητες των προεκπαιδευμένων LLMs με έναν μηχανισμό ανάκτησης πληροφοριών από μια εξωτερική βάση γνώσεων. Ουσιαστικά, αποτελεί μία διαδικασία βελτιστοποίησης της απόδοσης ενός μεγάλου γλωσσικού μοντέλου, καθώς ανατρέχει σε μια έγκυρη βάση γνώσεων, πριν από τη δημιουργία μιας απάντησης. Έτσι ενώ, τα μεγάλα γλωσσικά μοντέλα (LLM) εκπαιδεύονται σε τεράστιους όγκους δεδομένων και χρησιμοποιούν δισεκατομμύρια παραμέτρους για να παράγουν απαντήσεις σε ερωτήσεις, η RAG επεκτείνει τις ήδη ισχυρές δυνατότητες των LLM σε συγκεκριμένους τομείς ή στην εσωτερική βάση γνώσεων ενός οργανισμού, χωρίς να

απαιτείται επανεκπαίδευση του μοντέλου. Πρόκειται για μια οικονομικά αποδοτική προσέγγιση για τη βελτίωση της απόδοσης των LLM, ώστε να παραμένουν συναφή, ακριβή και χρήσιμα σε διάφορα πλαίσια.

Το RAG, αναφερόμενο σε έγκυρες και ειδικές γνώσεις, βελτιώνει τη συνάφεια, την ακρίβεια και τη χρησιμότητα των απαντήσεων που παράγονται [19]. Η μεθοδολογία RAG αποτελεί μια υβριδική αρχιτεκτονική που συνδυάζει τεχνικές ανάκτησης πληροφορίας (retrieval) με τη δημιουργική παραγωγή φυσικής γλώσσας (generation) και μπορεί να περιγραφεί ως μια διαδοχική διαδικασία σταδίων, όπου κάθε στάδιο επηρεάζει άμεσα την τελική ποιότητα της απάντησης.

2.4.1 Στάδια Λειτουργίας RAG

Η λειτουργία ενός συστήματος RAG αποτελείται από τα εξής βασικά στάδια:

1. **Προεπεξεργασία και τμηματοποίηση κειμένου** (Chunking)

Το αρχικό σύνολο (π.χ. έγγραφα, άρθρα, τεχνικά κείμενα) συνήθως δεν είναι κατάλληλο για άμεση εισαγωγή στο γλωσσικό μοντέλο λόγω μεγάλου μήκους και ανομοιογενούς δομής. Για τον λόγο αυτό εφαρμόζεται διαδικασία τμηματοποίησης (chunking), κατά την οποία τα έγγραφα διασπώνται σε μικρότερα τμήματα κειμένου. Η επιλογή του μεγέθους των τμημάτων αποτελεί κρίσιμη παράμετρο, καθώς πολύ μικρά τμήματα μπορεί να οδηγούν σε απώλεια συμφραζομένων, ενώ πολύ μεγάλα τμήματα αυξάνουν το κόστος ανάκτησης και μειώνουν την ακρίβεια αντιστοίχισης.

Συχνά εφαρμόζεται και επικάλυψη (overlap) μεταξύ διαδοχικών τμημάτων, ώστε να διατηρούνται σημαντικές πληροφορίες που ενδέχεται να βρίσκονται στα όρια των αποσπασμάτων. Η διαδικασία αυτή βελτιώνει τη συνοχή της ανακτημένης πληροφορίας και μειώνει την πιθανότητα αποσπασματικής ανάκτησης.

2. **Υπολογισμός διανυσματικών αναπαραστάσεων** (Embedding)

Μετά την τμηματοποίηση, κάθε τμήμα κειμένου μετατρέπεται σε διανυσματική αναπαράσταση μέσω ενός μοντέλου embeddings. Η αναπαράσταση αυτή αποτυπώνει το σημασιολογικό περιεχόμενο του αποσπάσματος σε έναν διανυσματικό χώρο, επιτρέποντας την εύρεση «κοντινών» τμημάτων με βάση την ομοιότητα νοήματος. Το ίδιο ισχύει και για το ερώτημα του χρήστη, το οποίο μετατρέπεται σε αντίστοιχο διάνυσμα ώστε να είναι δυνατή η σύγκρισή του με τα embeddings των αποσπασμάτων.

Η ποιότητα των embeddings αποτελεί έναν από τους σημαντικότερους παράγοντες απόδοσης ενός συστήματος RAG, καθώς επηρεάζει άμεσα την ικανότητα του retriever να εντοπίζει τα πλέον συναφή τμήματα κειμένου.

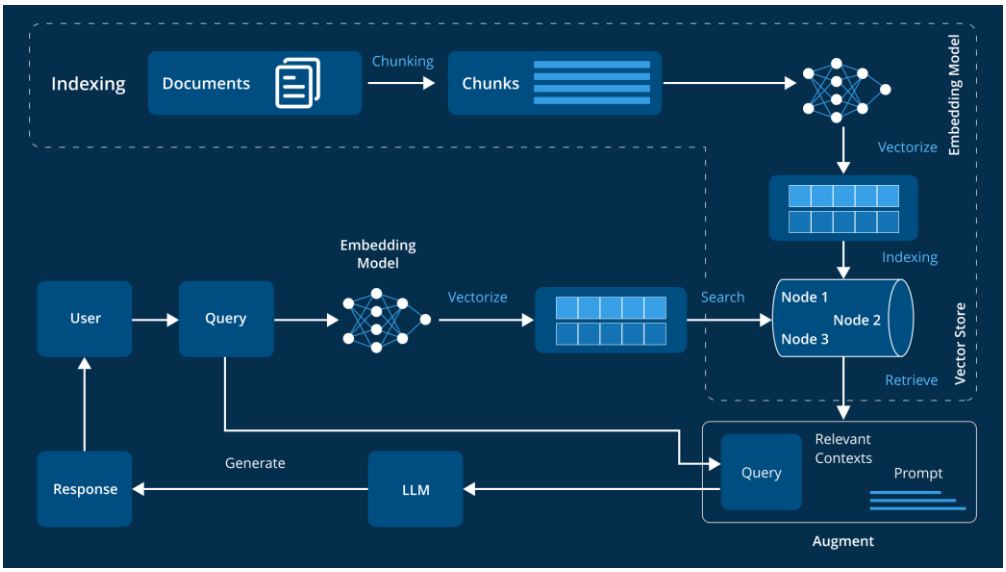
3. **Ανάκτηση** (Indexing)

Μόλις υποβληθεί ένα ερώτημα στο LLM, το σύστημα αναζητά σχετικές πληροφορίες που θα διαμορφώσουν την τελική απάντηση. Τα παραγόμενα embeddings, από τα έγγραφα που έχει προ εκπαιδευτεί το σύστημα, έχουν ήδη αποθηκευτεί σε μια δομή ευρετηρίασης, συνήθως σε μορφή διανυσματικής βάσης δεδομένων (vector database) ή index. Κατά την εκτέλεση ενός ερωτήματος, το

embedding του χρήστη συγκρίνεται με τα embeddings του index και ανακτώνται τα **k** πιο συναφή αποσπάσματα, βάσει μιας μετρικής ομοιότητας (π.χ. cosine similarity). Το στάδιο αυτό είναι καθοριστικό, καθώς η ποιότητα του ανακτημένου περιεχομένου επηρεάζει άμεσα την ποιότητα της τελικής απάντησης.

4. **Εμπλουτισμένη απάντηση** (Augmented Generation)

Τα αποσπάσματα που ανακτήθηκαν εισάγονται ως "context" στο LLM. Το prompt, που εισάγεται στο μοντέλο, συνήθως περιλαμβάνει το αρχικό ερώτημα του χρήστη, το ανακτημένο περιεχόμενο ως τεκμηριωτικό πλαίσιο, όπως ήδη αναφέρθηκε και κάποιες οδηγίες σχετικά με τον τρόπο σύνθεσης της απάντησης (π.χ. ακρίβεια, ύφος, αναφορά σε πηγές) [3]. Με αυτόν τον τρόπο, η RAG αρχιτεκτονική μειώνει την πιθανότητα ψευδαισθήσεων και επιτρέπει την αξιοποίηση εξωτερικής γνώσης, χωρίς να απαιτείται επανεκπαίδευση του μοντέλου.



Σχήμα 2.4 : Αρχιτεκτονική ενός τυπικού συστήματος RAG

Πηγή : [19]

2.4.2 Περιορισμοί και Προκλήσεις

Παρά τα πλεονεκτήματα, η αρχιτεκτονική RAG παρουσιάζει και αρκετές προκλήσεις. Πιο συγκεκριμένα οι απαντήσεις ενδέχεται να προέρχονται από αναξιόπιστες πηγές, μειώνοντας την αξιοπιστία τους. Ταυτόχρονα αν ανακτηθούν λανθασμένα ή άσχετα αποσπάσματα, τότε η απάντηση του LLM θα είναι επίσης λανθασμένη ή ελλιπής ή αν χρησιμοποιείται παρόμοια ορολογία, με διαφορετικό τρόπο στις διάφορες πηγές εκπαίδευσης μπορεί να οδηγηθεί το μοντέλο επίσης σε ανακριβείς απαντήσεις.

Ακόμη, τα LLMs έχουν περιορισμό στο μέγεθος κειμένου που μπορούν να λάβουν ως είσοδο. Συνεπώς απαιτείται επιλογή των πιο σχετικών αποσπασμάτων. Τέλος, η παραγωγή embeddings και η αναζήτηση σε μεγάλη κλίμακα απαιτούν υπολογιστικούς πόρους, ιδιαίτερα σε περιβάλλοντα χωρίς GPU.

2.5 Διεπαφές Επικοινωνίας και RESTful Υπηρεσίες

Η ανάπτυξη σύγχρονων πληροφοριακών συστημάτων βασίζεται συχνά σε αρχιτεκτονικές, όπου τα επιμέρους υποσυστήματα επικοινωνούν μεταξύ τους μέσω τυποποιημένων διεπαφών. Σε τέτοιες αρχιτεκτονικές, η λειτουργικότητα της εφαρμογής διαχωρίζεται συνήθως σε επίπεδα, όπως το επίπεδο παρουσίασης (frontend) και το επίπεδο επιχειρησιακής λογικής (backend). Για να επιτευχθεί η επικοινωνία μεταξύ αυτών των επιπέδων, χρησιμοποιούνται διαδικτυακές υπηρεσίες, με πιο διαδεδομένη προσέγγιση τα RESTful APIs.

Το REST (Representational State Transfer) αποτελεί αρχιτεκτονικό στυλ σχεδίασης υπηρεσιών που βασίζεται στο πρωτόκολλο HTTP και επιτρέπει την ανταλλαγή δεδομένων με απλό και επεκτάσιμο τρόπο.

Οι RESTful υπηρεσίες αξιοποιούν συγκεκριμένες HTTP μεθόδους, οι οποίες αντιστοιχούν σε βασικές λειτουργίες:

- **GET:** ανάκτηση δεδομένων
- **POST:** αποστολή/δημιουργία νέων δεδομένων
- **PUT/PATCH:** ενημέρωση δεδομένων
- **DELETE:** διαγραφή δεδομένων

2.5.1 Λειτουργία REST API

1.**Ο πελάτης (client) στέλνει αίτημα (request)**

Μια εφαρμογή πελάτη (όπως μια εφαρμογή για κινητά, ένας περιηγητής ιστού ή ένας άλλος διακομιστής) ξεκινά ένα αίτημα προς το API για την εκτέλεση μιας ενέργειας. Αυτό το αίτημα αποστέλλεται σε μια συγκεκριμένη διεύθυνση URL και περιλαμβάνει μια μέθοδο HTTP, κεφαλίδες με μεταδεδομένα και, μερικές φορές, ένα σώμα αιτήματος με δεδομένα.

2. **Ο διακομιστής (server) επεξεργάζεται το αίτημα**

Ο διακομιστής λαμβάνει και επικυρώνει το αίτημα, επιβεβαιώνοντας ότι ο πελάτης είναι πιστοποιημένος και εξουσιοδοτημένος να εκτελέσει την αιτούμενη ενέργεια. Στη συνέχεια, επεξεργάζεται το αίτημα, το οποίο μπορεί να περιλαμβάνει την ανάκτηση δεδομένων από μια βάση δεδομένων, τη δημιουργία ενός νέου πόρου ή την ενημέρωση ενός υπάρχοντος.

3. **Ο διακομιστής στέλνει απάντηση (response)**

Μετά την επεξεργασία του αιτήματος, ο διακομιστής στέλνει μια απάντηση στον πελάτη. Αυτή η απάντηση περιλαμβάνει έναν κωδικό κατάστασης HTTP που υποδεικνύει το αποτέλεσμα (για παράδειγμα, 200 OK για επιτυχία, 404 Not Found αν ο πόρος δεν υπάρχει, 401 Unauthorized για ζητήματα ασφαλείας, 500 για σφάλμα στον διακομηστή), μαζί με τα ζητούμενα δεδομένα στο σώμα της απάντησης.

4. **Μεταφορά δεδομένων**

Τα δεδομένα που μεταφέρονται μεταξύ του πελάτη και του διακομιστή είναι μια «αναπαράσταση» της κατάστασης του πόρου. Το JSON (JavaScript Object Notation) είναι η πιο κοινή μορφή για αυτά τα δεδομένα, επειδή είναι ελαφρύ, ευανάγνωστο από τον άνθρωπο και εύκολο στην ανάλυση από τις γλώσσες προγραμματισμού [20].

Στο πλαίσιο της παρούσας εργασίας, η χρήση RESTful υπηρεσιών αποτελεί κρίσιμο στοιχείο για τη λειτουργία του συστήματος ως ολοκληρωμένης εφαρμογής. Μέσω της ύπαρξης REST API, καθίσταται δυνατή η επικοινωνία της διεπαφής χρήστη με τον μηχανισμό ανάκτησης πληροφορίας και παραγωγής απαντήσεων, επιτρέποντας την πρακτική αξιοποίηση του chatbot σε πραγματικά σενάρια χρήσης.

Η λεπτομερής υλοποίηση των endpoints, η αρχιτεκτονική του backend, καθώς και οι τεχνολογίες που αξιοποιούνται (π.χ. FastAPI, μηχανισμοί αυθεντικοποίησης και frontend εφαρμογή) αναλύονται στο επόμενο κεφάλαιο, όπου παρουσιάζεται η συνολική σχεδίαση και υλοποίηση του συστήματος.

2.6 Επίλογος

Στο παρόν κεφάλαιο παρουσιάστηκε το θεωρητικό υπόβαθρο που απαιτείται για την κατανόηση και την υλοποίηση του συστήματος που έχει υλοποιηθεί, βασισμένου στην αρχιτεκτονική Retrieval-Augmented Generation (RAG). Αρχικά αναλύθηκαν οι έννοιες της αναπαράστασης κειμένου και της ανάκτησης πληροφορίας, με έμφαση στη σημασιολογική ομοιότητα και στη χρήση διανυσματικών αναπαραστάσεων (embeddings), οι οποίες αποτελούν τη βάση για τη σύγχρονη σημασιολογική αναζήτηση.

Ταυτόχρονα έγινε μία αξιοποίηση διαφορετικών μοντέλων embeddings όπου συμπερασματικά καταλήξαμε ότι τα μοντέλα E5, GTE και BGE-M3 είναι σχεδιασμένα με έμφαση στην ανάκτηση πληροφορίας και αναμένεται να παρουσιάζουν πιο σταθερή συμπεριφορά σε σενάρια retrieval. Αντίθετα, μοντέλα τύπου RoBERTa και ελληνικά μοντέλα όπως το Meltemi μπορούν να προσφέρουν ισχυρή κατανόηση κειμένου, αλλά συχνά με αυξημένο υπολογιστικό κόστος, ιδιαίτερα σε περιβάλλοντα χωρίς επιτάχυνση, δηλαδή σε περιβάλλοντα χωρίς GPU.

Στη συνέχεια, παρουσιάστηκαν τα μέτρα ομοιότητας και οι τεχνικές διανυσματικής αναζήτησης, καθώς και ο ρόλος των διανυσματικών βάσεων δεδομένων, με ιδιαίτερη αναφορά στη βιβλιοθήκη FAISS, η οποία επιτρέπει την αποδοτική ανάκτηση σχετικών αποσπασμάτων σε μεγάλα σύνολα δεδομένων. Ακολούθως, αναλύθηκε η αρχιτεκτονική RAG ως συνδυασμός retrieval και παραγωγής απάντησης από γλωσσικό μοντέλο, αναδεικνύοντας τα πλεονεκτήματά της ως προς την αξιοπιστία, τη μείωση ανακριβών απαντήσεων και την αξιοποίηση εξωτερικών πηγών γνώσης.

Τέλος, έγινε αναφορά στη σημασία των RESTful υπηρεσιών ως μηχανισμού επικοινωνίας μεταξύ των επιπέδων μιας εφαρμογής, επιτρέποντας τη σύνδεση της διεπαφής χρήστη με τον μηχανισμό ανάκτησης και παραγωγής απαντήσεων.

Κεφάλαιο 3ο: Υλοποιήσεις

3.1 Εισαγωγή

Το σύστημα της πτυχιακής εργασίας αναπτύχθηκε με χρήση της γλώσσας προγραμματισμού Python 3.10. Τα πειράματα και η εκπαίδευση των embeddings μοντέλων εκτελέστηκαν σε ένα σύστημα Intel(R) Core(TM) i5-8500, με κεντρική μονάδα επεξεργασίας (CPU) με 6 πυρήνες και με 16 GB μνήμη RAM.

Στο παρόν κεφάλαιο παρουσιάζεται η πρακτική υλοποίηση της πλατφόρμας chatbot, η οποία αναπτύχθηκε. Ο στόχος του συστήματος είναι η παροχή απαντήσεων σε ερωτήματα χρηστών με βάση έγγραφα που έχουν μεταφορτωθεί από τους ίδιους, αξιοποιώντας RAG.

Η εφαρμογή σχεδιάστηκε με αρχιτεκτονική που διαχωρίζει σαφώς:

- το **Frontend/UI** (διεπαφή χρήστη),
- το **Backend/API** (FastAPI),
- τον μηχανισμό **ανάκτησης πληροφορίας** (RAG - FAISS vector store),
- την **παραγωγή απάντησης** μέσω μεγάλου γλωσσικού μοντέλου (Gemini),
- καθώς και τον μηχανισμό **ασφάλειας και πιστοποίησης** χρηστών (Keycloak).

Παράλληλα, η εφαρμογή είναι πλήρως containerized μέσω Docker, ώστε να είναι εύκολα διαθέσιμη σε οποιονδήποτε χρήστη.

3.2 Αρχιτεκτονική του Συστήματος

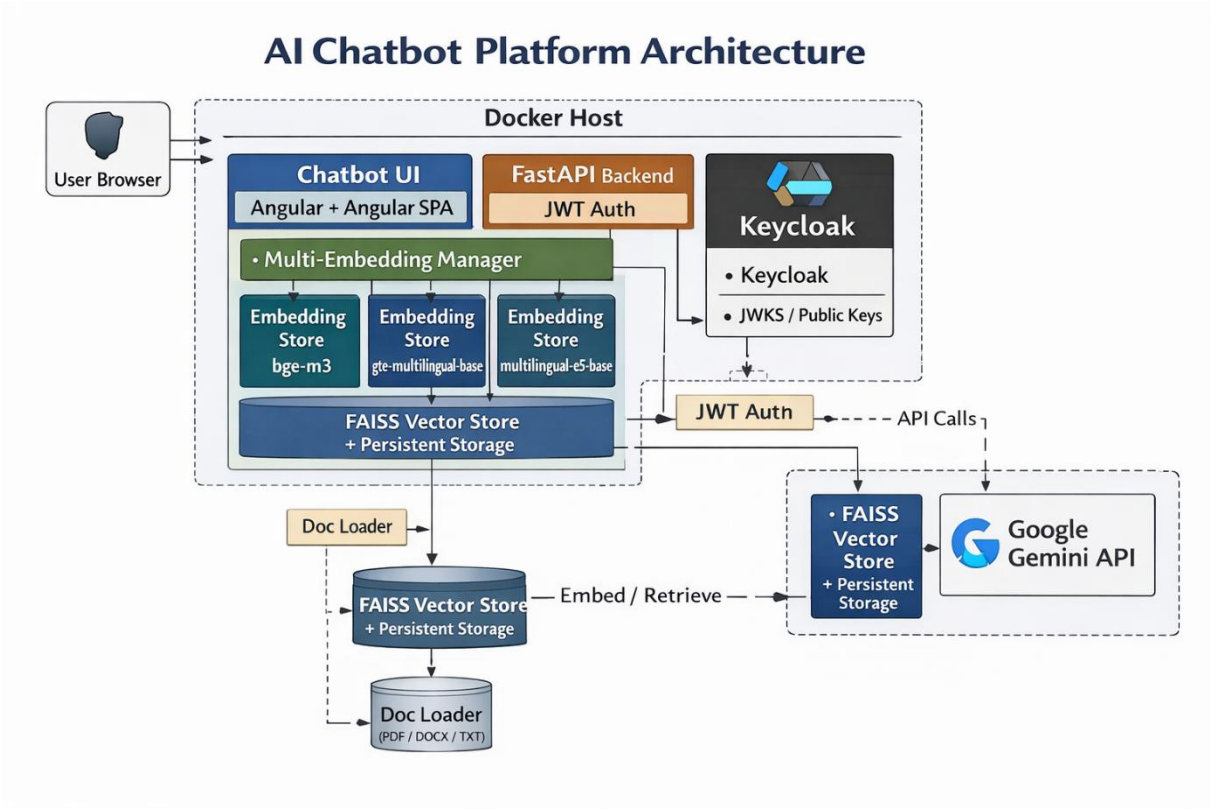
Το σύστημα σχεδιάστηκε με στόχο να λειτουργεί ως ένα ολοκληρωμένο chatbot βασισμένο στην αρχιτεκτονική RAG, το οποίο μπορεί να απαντά με ακρίβεια σε ερωτήσεις χρηστών αξιοποιώντας έγγραφα του ελληνικού δημόσιου τομέα που έχουν προηγουμένως εισαχθεί (ingested) στη βάση γνώσης.

Η υλοποίηση ακολουθεί πολυεπίπεδη αρχιτεκτονική (multi-tier architecture), διαχωρίζοντας τη διεπαφή χρήστη από τη λογική επεξεργασίας και ανάκτησης πληροφορίας. Με τον τρόπο αυτό, επιτυγχάνεται καλύτερη επεκτασιμότητα, συντηρησιμότητα και δυνατότητα μελλοντικής αναβάθμισης του συστήματος.

Αναλυτικότερα, ο χρήστης αλληλεπιδρά με την εφαρμογή μέσω ενός web browser, ο οποίος φορτώνει την Angular SPA διεπαφή. Η διεπαφή αυτή επικοινωνεί με το backend μέσω HTTP αιτημάτων, αξιοποιώντας endpoints που υλοποιούνται μέσω FastAPI. Παράλληλα, για τη διαχείριση της αυθεντικοποίησης, η εφαρμογή χρησιμοποιεί Keycloak, το οποίο εκδίδει JWT tokens που συνοδεύουν κάθε αίτημα προς το backend.

Το backend αποτελεί τον κεντρικό κόμβο της αρχιτεκτονικής, καθώς εκτελεί όλη τη λογική του RAG pipeline. Από τη μία πλευρά, δέχεται αρχεία από τους χρήστες και τα επεξεργάζεται ώστε να εξαχθεί το κείμενό τους, ενώ από την άλλη, αποθηκεύει την πληροφορία σε διανυσματική μορφή μέσα σε FAISS indexes. Όταν ο χρήστης υποβάλει ερώτημα, το backend αναζητά τα πιο σχετικά αποσπάσματα στο FAISS και στη συνέχεια τα χρησιμοποιεί ως context για την κλήση του Google Gemini, ώστε να παραχθεί η τελική απάντηση.

Όλη αυτή η αρχιτεκτονική περιγράφεται από την εικόνα παρακάτω.



Σχήμα 3.1 : Ολοκληρωμένη διαδικασία υλοποίησης του chatbot με χρήση Gemini, FAISS library, Keycloak, FAST API και Multi-embedding manager

Η αρχιτεκτονική αυτή έχει υλοποιηθεί πρακτικά μέσα από τρία βασικά επίπεδα κώδικα. Στο frontend, τα κύρια αρχεία βρίσκονται στον φάκελο *chatbot-ui/src/app/pages*, ενώ στο backend η βασική εφαρμογή οργανώνεται γύρω από το *main.py* και τον φάκελο *chatbot/*, όπου βρίσκονται οι κλάσεις που υλοποιούν τη λειτουργικότητα embeddings, vector search και της παραγωγής απάντησης. Η πιστοποίηση των χρηστών υλοποιείται μέσω του *auth.py*, το οποίο εφαρμόζει έλεγχο JWT token σε κάθε προστατευμένο endpoint.

3.3 Backend και Μηχανισμός RAG

Το backend υλοποιείται με FastAPI και αποτελεί τον βασικό μηχανισμό εκτέλεσης του RAG pipeline. Η εφαρμογή οργανώνεται με τρόπο ώστε οι βασικές λειτουργίες ingestion και querying να είναι διακριτές, επιτρέποντας τη συντήρηση και την επέκταση του κώδικα. Στην υλοποίηση, ο κεντρικός ρόλος δίνεται στην κλάση Chatbot που βρίσκεται στο αρχείο *chatbot/chatbot.py*. Η κλάση αυτή λειτουργεί ως orchestrator, δηλαδή συντονίζει τόσο την εισαγωγή νέων εγγράφων όσο και την απάντηση σε ερωτήματα χρηστών.

Στο *main.py* γίνεται η αρχικοποίηση ενός αντικειμένου Chatbot, το οποίο χρησιμοποιείται από τα endpoints της εφαρμογής. Με αυτόν τον τρόπο, τα endpoints δεν περιλαμβάνουν την επιχειρησιακή λογική, αλλά περιορίζονται στο να λαμβάνουν τα δεδομένα του χρήστη, να τα προετοιμάζουν και να καλούν τις κατάλληλες μεθόδους του chatbot.

Το backend υλοποιεί δύο βασικές ροές:

- **τη ροή εισαγωγής εγγράφων** (upload → extract → chunk → embed → store)
- **τη ροή ερώτησης–απάντησης** (query → embed → retrieve → prompt → generate).

Η λειτουργία του backend μπορεί να περιγραφεί συνοπτικά μέσα από τα ακόλουθα στάδια:

1. **Εισαγωγή εγγράφων (Document Ingestion):**

   Ο χρήστης εισάγει έγγραφα στο σύστημα (π.χ. PDF, DOCX ή TXT). Το σύστημα εξάγει το κείμενο, το κατατμεί σε αποσπάσματα και το αποθηκεύει στη βάση γνώσης.

2. **Παραγωγή διανυσματικών αναπαραστάσεων (Embedding Generation):**

   Για κάθε απόσπασμα παράγονται embeddings μέσω μοντέλων σημασιολογικής αναπαράστασης κειμένου. Στην παρούσα εργασία υιοθετήθηκε multi-embedding προσέγγιση, ώστε να υποστηρίζεται αναζήτηση με διαφορετικά μοντέλα.

3. **Αποθήκευση σε διανυσματικό ευρετήριο (Vector Indexing):**

   Τα embeddings αποθηκεύονται σε διανυσματικό index, χρησιμοποιώντας FAISS, με σκοπό την αποδοτική ανάκτηση των πιο σχετικών αποσπασμάτων.

4. **Υποβολή ερωτήματος (User Query):**

   Ο χρήστης υποβάλει μία ερώτηση μέσω της διεπαφής του chatbot.

5. **Ανάκτηση αποσπασμάτων (Retrieval):**

   Το σύστημα μετατρέπει το ερώτημα σε embedding και αναζητά τα πιο σχετικά αποσπάσματα στο διανυσματικό ευρετήριο. Στη συνέχεια επιλέγεται το πιο σχετικό αρχείο (single-file retrieval) και επιστρέφονται τα top-k αποσπάσματα από αυτό.

6. **Παραγωγή απάντησης (Answer Generation):**

   Τα ανακτημένα αποσπάσματα χρησιμοποιούνται ως context και δίνονται σε γλωσσικό μοντέλο, το οποίο παράγει απάντηση βασισμένη αποκλειστικά στο παρεχόμενο περιεχόμενο, σύμφωνα με το prompt που του έχει δοθεί.

3.3.1 Εισαγωγή εγγράφων (document loader)

Η διαδικασία εισαγωγής νέων εγγράφων περιλαμβάνει έναν μηχανισμό φόρτωσης και εξαγωγής κειμένου από διαφορετικές μορφές αρχείων. Η εξαγωγή κειμένου πραγματοποιείται μέσω της κλάσης *DocumentLoader* στο αρχείο *chatbot/document_loader.py*, η οποία υποστηρίζει διαφορετικούς τύπους αρχείων, όπως PDF, DOCX και TXT. Το κείμενο που προκύπτει δεν αποθηκεύεται ως ενιαίο σώμα, αλλά μετατρέπεται σε μικρότερα τμήματα (chunks), ώστε να είναι αποτελεσματική η ανάκτηση και να μπορεί να αξιοποιηθεί από το LLM χωρίς υπέρβαση ορίων context.

Το chunking των κειμένων υλοποιείται στο αρχείιο *main.py* μέσω της συνάρτησης *chunk_text(text, chunk_size, overlap)*. Η συνάρτηση αυτή εφαρμόζει διαχωρισμό, δημιουργώντας τμήματα σταθερού μεγέθους, ενώ ταυτόχρονα χρησιμοποιεί overlap μεταξύ διαδοχικών chunks. Η ύπαρξη overlap είναι σημαντική, διότι επιτρέπει τη διατήρηση της συνέχειας μεταξύ τμημάτων και μειώνει την πιθανότητα απώλειας πληροφορίας στα όρια ενός chunk.

3.3.2 Υπολογισμός Embeddings

Ο υπολογισμός embeddings αποτελεί το επόμενο κρίσιμο στάδιο, μετά την παραγωγή των chunks, για τη λειτουργία του RAG, καθώς μετατρέπει το κείμενο των εγγράφων σε διανυσματικές αναπαραστάσεις. Η εφαρμογή υποστηρίζει πολλαπλά embedding models, υλοποιώντας μια multi-embedding προσέγγιση. Η υλοποίηση αυτή διεξάγεται μέσω της κλάσης *MultiEmbeddingManager* στο αρχείο *chatbot/multi_embedding_manager.py*. Ο manager αυτός διατηρεί ξεχωριστά EmbeddingStore αντικείμενα, ένα για κάθε embedding model που χρησιμοποιείται, έτσι κάθε έγγραφο που εισάγεται στο σύστημα αποθηκεύεται σε ξεχωριστά διανυσματικά indexes, ένα για κάθε embedding model.

Στον κώδικα, τα διαθέσιμα μοντέλα είναι

- **BAAI/bge-m3**
- **Alibaba-NLP/gte-multilingual-base**
- **intfloat/multilingual-e5-base**

και κατά την αρχικοποίηση δημιουργούνται οι αντίστοιχες δομές αποθήκευσης.

Η παραγωγή embeddings πραγματοποιείται μέσω της κλάσης EmbeddingBackend στο αρχείο *chatbot/embedding_backend.py*, η οποία λειτουργεί ως wrapper γύρω από τα 3 embedding μοντέλα. Η μέθοδος *encode(texts, is_query)* παράγει embeddings είτε για passages (chunks) είτε για queries. Στην περίπτωση των E5 μοντέλων (intfloat/multilingual-e5-base) εφαρμόζεται επιπλέον προεπεξεργασία του κειμένου μέσω της μεθόδου *_format_inputs*, όπου τα κείμενα μορφοποιούνται ως "query: ..." και "passage: ...". Η διαφοροποίηση αυτή είναι απαραίτητη ώστε το embedding space να παραμένει συνεπές μεταξύ των queries και των αρχείων.

Τέλος, τα embeddings αποθηκεύονται μέσω της μεθόδου *add_documents* σε κάθε EmbeddingStore. Ο *MultiEmbeddingManager* καλεί την εισαγωγή σε όλα τα stores, ώστε κάθε chunk να υπάρχει σε πολλαπλά vector spaces. Με τον τρόπο αυτό, η ανάκτηση μπορεί να εκτελεστεί με βάση το μοντέλο που επιλέγεται, επιτρέποντας συγκρίσεις και πειραματισμό.

3.3.3 Διανυσματική Αναζήτηση και Ανάκτηση Πληροφορίας

Στο πλαίσιο του προτεινόμενου συστήματος ερωταπαντήσεων, η FAISS χρησιμοποιείται ως μηχανισμός ανάκτησης των πλέον σχετικών αποσπασμάτων από τα έγγραφα του ελληνικού δημόσιου τομέα. Τα αποσπάσματα αυτά αποτελούν το “τεκμηριωμένο πλαίσιο” που δίνεται στη συνέχεια στο γλωσσικό μοντέλο, ώστε η τελική απάντηση να παραχθεί με βάση πραγματικό περιεχόμενο και όχι με εικασία.

Αναλυτικότερα, μετά την παραγωγή embeddings, τα διανύσματα κανονικοποιούνται και αποθηκεύονται σε FAISS index αρχεία για αποδοτική αναζήτηση, με βάση τη σημασιολογική ομοιότητά τους. Παράλληλα, αποθηκεύονται τα πρωτογενή αποσπάσματα κειμένου καθώς και μεταδεδομένα (π.χ. από ποιο αρχείο προέρχεται κάθε chunk). Στο σύστημα επιλέγεται κατάλληλη δομή ευρετηρίου, ώστε να υποστηρίζεται γρήγορη αναζήτηση top-k αποτελεσμάτων.

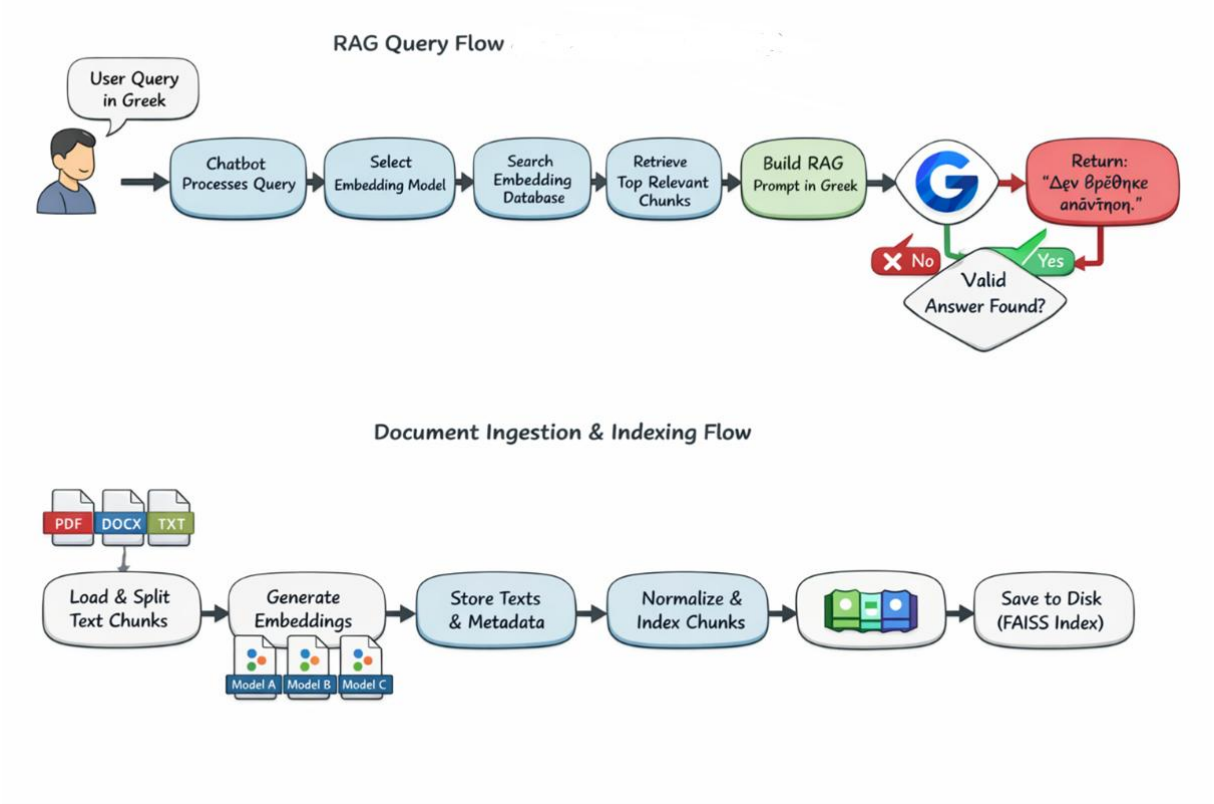
Η ανάκτηση πληροφορίας υλοποιείται μέσω της κλάσης *EmbeddingStore* στο αρχείο *chatbot/embedding_store.py*. Κάθε store διατηρεί ένα FAISS index, καθώς και τις αντίστοιχες λίστες chunks και metadata. Η δημιουργία του index πραγματοποιείται μέσω της μεθόδου *_rebuild_index,* η οποία κατασκευάζει έναν *faiss.IndexFlatIP*. Πριν την εισαγωγή στο index εφαρμόζεται normalization των embeddings, ώστε το inner product να αντιστοιχεί ουσιαστικά σε ομοιότητα συνημιτόνου. Με

αυτόν τον τρόπο, η ανάκτηση βασίζεται σε σημασιολογική ομοιότητα και όχι σε απλή λέξη-προς-λέξη σύγκριση.

Όταν ο χρήστης υποβάλει ένα ερώτημα, το backend υπολογίζει πρώτα το embedding του query και στη συνέχεια εκτελεί αναζήτηση στο διανυσματικά αρχεία FAISS index. Η ανάκτηση δεν περιορίζεται σε μια απλή επιστροφή top-k chunks, αλλά εφαρμόζεται πιο στοχευμένη στρατηγική μέσω της μεθόδου *search_grouped(query, k_files, k_per_file)*. Η μέθοδος αυτή αρχικά αναζητά ένα μεγαλύτερο σύνολο αποτελεσμάτων (π.χ. top 100), ώστε να υπάρχει επαρκής βάση επιλογής. Στη συνέχεια, τα αποτελέσματα ομαδοποιούνται ανά αρχείο, χρησιμοποιώντας τα metadata που αποθηκεύτηκαν κατά την εισαγωγή. Με αυτόν τον τρόπο, το σύστημα μπορεί να εντοπίσει ποιο αρχείο είναι συνολικά πιο σχετικό με το ερώτημα.

Η ομαδοποίηση επιτρέπει την επιλογή των καλύτερων αρχείων βάσει του μέγιστου score που εμφανίζεται ανά ομάδα. Έπειτα, από το πιο σχετικό αρχείο επιστρέφονται τα καλύτερα *k_per_file chunks*. Η επιλογή αυτή ενισχύει την ποιότητα του ανακτημένου κειμένου, καθώς μειώνει την πιθανότητα να συνδυαστούν αποσπάσματα από πολλά άσχετα έγγραφα, κάτι που θα οδηγούσε το LLM σε σύγχυση ή σε απαντήσεις μειωμένης ακρίβειας. Επιπλέον, στον κώδικα εφαρμόζεται ένα κατώφλι ομοιότητας (threshold), ώστε chunks με πολύ χαμηλό score να απορρίπτονται. Αυτό συμβάλλει στη μείωση «ψευδαισθήσεων», καθώς το σύστημα αποφεύγει να παρέχει κείμενο αμφίβολης συνάφειας.

Η στρατηγική αυτή ολοκληρώνεται στην κλάση *Chatbot*, όπου η μέθοδος *answer_query(query, k_per_file, embedding_model)* επιλέγει το κατάλληλο embedding store μέσω του manager και καλεί την αναζήτηση. Τα αποτελέσματα της ανάκτησης μετατρέπονται σε κείμενο, το οποίο ενσωματώνεται στο prompt που θα σταλεί στο LLM **Gemini 2.5 Flash**.



Σχήμα 3.2 : Αρχιτεκτονική RAG του συστήματος που υλοποιήθηκε

3.3.4 Αξιολόγηση RAG

Η αξιολόγηση ενός συστήματος RAG (Retrieval-Augmented Generation) απαιτεί τη μελέτη και της ανάκτησης (retrieval), δηλαδή του κατά πόσο το σύστημα εντοπίζει τα σωστά αποσπάσματα από τα έγγραφα, αλλά και της παραγωγής (generation) απάντησης, δηλαδή της ποιότητας της τελικής απάντησης που δημιουργεί το LLM με βάση το ανακτημένο context.

Στην παρούσα εργασία, η αξιολόγηση επικεντρώθηκε κυρίως στο στάδιο της ανάκτησης, καθώς το embedding μοντέλο και ο μηχανισμός αναζήτησης καθορίζουν την ποιότητα των αποσπασμάτων που τροφοδοτούνται στο LLM. Οι μετρικές που χρησιμοποιήθηκαν για αποτυπώνουν τόσο την απόδοση (latency/κόστος) όσο και την ποιότητα ανάκτησης είναι οι εξής :

Retrieval (ms)
Αντιπροσωπεύει τον χρόνο που απαιτείται για τη διαδικασία ανάκτησης. Περιλαμβάνει:

- τη δημιουργία embedding του ερωτήματος,
- την αναζήτηση στον FAISS index,
- και την επιλογή των Top-k αποσπασμάτων από το επιλεγμένο έγγραφο. Η μετρική αυτή είναι κρίσιμη καθώς επηρεάζει άμεσα τον χρόνο απόκρισης του συστήματος και επιτρέπει τη σύγκριση της ταχύτητας μεταξύ διαφορετικών embedding models.

Total (ms)
Αντιστοιχεί στον συνολικό χρόνο απόκρισης από την υποβολή του ερωτήματος μέχρι την παραγωγή τελικής απάντησης. Περιλαμβάνει τόσο το retrieval όσο και τον χρόνο κλήσης του LLM. Παρότι εκφράζει την "εμπειρία χρήστη" (end-to-end latency), επηρεάζεται σημαντικά από εξωτερικούς παράγοντες (π.χ. δικτυακές καθυστερήσεις ή μεταβλητότητα υπηρεσίας LLM) και, επομένως, δεν απομονώνει αποκλειστικά την επίδοση του embedding μοντέλου.

CPU Retrieval(ms)
Μετρά τον χρόνο CPU που καταναλώθηκε από τη διεργασία κατά τη φάση ανάκτησης. Αποτελεί δείκτη υπολογιστικού κόστους και είναι χρήσιμος για συγκρίσεις αποδοτικότητας, ιδιαίτερα όταν στόχος είναι η λειτουργία σε περιβάλλοντα περιορισμένων πόρων. Σε αντίθεση με το Total ms, το CPU Retrieval ms αποτυπώνει σε μεγαλύτερο βαθμό το «καθαρό» υπολογιστικό κόστος της ανάκτησης.

Top1
Η μέγιστη τιμή ομοιότητας μεταξύ του ερωτήματος και του καλύτερου ανακτημένου αποσπάσματος. Στην υλοποίηση χρησιμοποιείται κανονικοποίηση διανυσμάτων και FAISS **I**ndexFlatIP, άρα το Top1 αντιστοιχεί σε cosine similarity. Υψηλότερη τιμή υποδηλώνει ότι το embedding μοντέλο θεωρεί το συγκεκριμένο απόσπασμα ιδιαίτερα σχετικό με το ερώτημα.

MeanTop5
Ο μέσος όρος των τιμών ομοιότητας των 5 αποσπασμάτων που τροφοδοτούνται στο LLM. Η μετρική αυτή αποτυπώνει την «συνολική ποιότητα» του context: όταν το MeanTop5 είναι υψηλό, τα

περισσότερα chunks είναι σχετικά και το LLM έχει μεγαλύτερη πιθανότητα να βρει την πληροφορία που απαιτείται.

Σημειώνεται ότι οι παραπάνω μετρικές (Top1 και MeanTop5) λειτουργούν ως δείκτες σχετικότητας της ανάκτησης, όμως δεν αποτελούν από μόνες τους απόδειξη ορθότητας της τελικής απάντησης. Η τελική ακρίβεια επηρεάζεται και από το περιεχόμενο των αποσπασμάτων (π.χ. αν περιέχουν ρητή πληροφορία), καθώς και από τους κανόνες του prompt.

3.3.5 RESTful API του Backend

Η διεπαφή επικοινωνίας μεταξύ του frontend και του backend υλοποιήθηκε μέσω ενός RESTful API, το οποίο αναπτύχθηκε με τη χρήση του framework **FastAPI**. Η επιλογή του FastAPI έγινε λόγω της υψηλής απόδοσης που προσφέρει, της άμεσης υποστήριξης για ανάπτυξη API σε Python, καθώς και της δυνατότητας ενσωμάτωσης validation μέσω Pydantic μοντέλων και dependency injection για μηχανισμούς όπως η πιστοποίηση χρηστών. Το API που υλοποιήθηκε υποστηρίζει εισαγωγή εγγράφων (ingestion), υποβολή ερωτήματος (query), επιστροφή απάντησης και πηγών (sources) καθώς και επιλογή embedding model και παραμέτρων ανάκτησης.

Η υλοποίηση των endpoints πραγματοποιείται στο αρχείο *main.py*, όπου χρησιμοποιείται ο μηχανισμός routing του FastAPI μέσω decorators όπως @app.post() και @app.get(). Η χρήση των decorators αυτών επιτρέπει την καθαρή δήλωση των διαθέσιμων HTTP μεθόδων και των αντίστοιχων διαδρομών, ενώ παράλληλα διευκολύνει τη συντήρηση και την επέκταση του API.

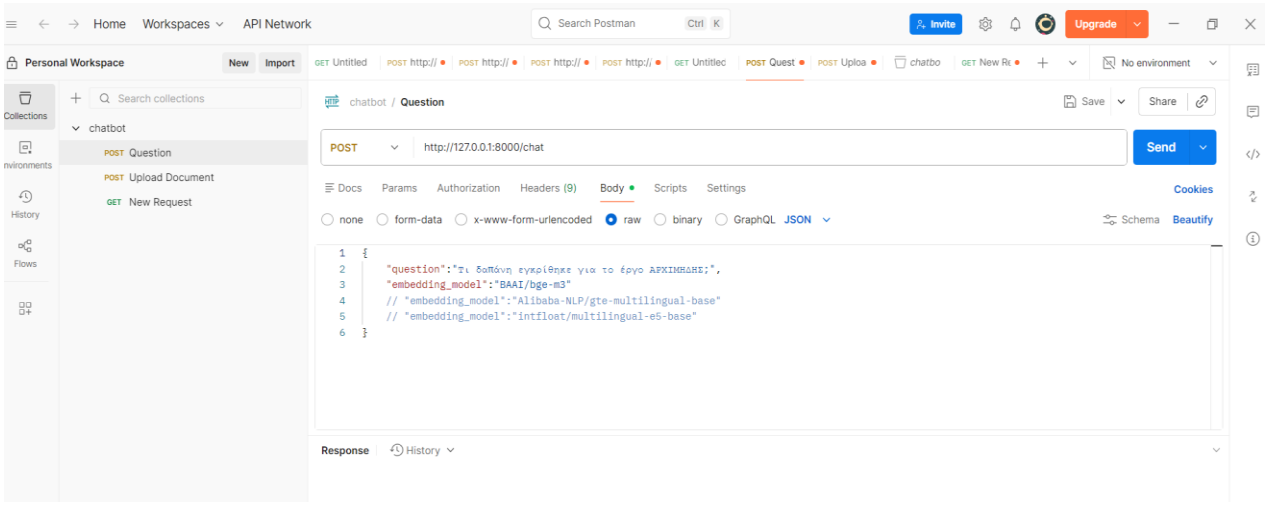
Τα δύο βασικά endpoints που σχετίζονται με τον πυρήνα της εφαρμογής είναι το endpoint μεταφόρτωσης εγγράφων και το endpoint συνομιλίας.

Το endpoint **/upload** υλοποιείται ως POST request και δέχεται αρχείο τύπου UploadFile. Κατά την εκτέλεσή του, το αρχείο αποθηκεύεται προσωρινά, ώστε να είναι διαθέσιμο για ανάγνωση από τον *DocumentLoader*. Ανάλογα με την επέκταση του αρχείου, καλείται η αντίστοιχη μέθοδος φόρτωσης (PDF/DOCX/TXT). Το κείμενο που προκύπτει περνά στη συνάρτηση *chunk_text*, ώστε να παραχθεί λίστα chunks, όπως αναλύσαμε και νωρίτερα. Στη συνέχεια, τα chunks εισάγονται στη διανυσματική βάση μέσω της μεθόδου *chatbot_instance.ingest_documents(chunks, filename)*, η οποία καταλήγει στο MultiEmbeddingManager και τελικά σε κάθε EmbeddingStore.

Το endpoint **/chat** υλοποιείται επίσης ως POST request και δέχεται ένα JSON σώμα που περιλαμβάνει το ερώτημα, το embedding model που επιλέχθηκε και την παράμετρο *k_per_file*. Το endpoint καλεί τη μέθοδο *answer_query* της κλάσης *Chatbot*. Η μέθοδος αυτή εκτελεί ανακάκτηση μέσω FAISS, κατασκευάζει prompt και καλεί το Google Gemini API για παραγωγή απάντησης. Η απάντηση επιστρέφεται στο frontend μαζί με πληροφορίες για τα chunks που χρησιμοποιήθηκαν ως πηγές.

Τέλος, υλοποιείται και endpoint **/models**, το οποίο επιστρέφει τη λίστα των διαθέσιμων embedding models, αξιοποιώντας τη μέθοδο *list_models()* του MultiEmbeddingManager. Με αυτόν τον τρόπο, το UI μπορεί να εμφανίζει δυναμικά τις επιλογές μοντέλων.

Στην παρακάτω εικόνα απεικονίζονται τα APIs που σχεδιάστηκαν στην εφαρμογή στο περιβάλλον Postman, καθώς στα πρώτα στάδια δεν είχε υλοποιηθεί η διεπαφή χρήστη και ήταν επιτακτική η ανάγκη βελτιστοποίηση τους αλγορίθμου ανάκτησης των δεδομένων.

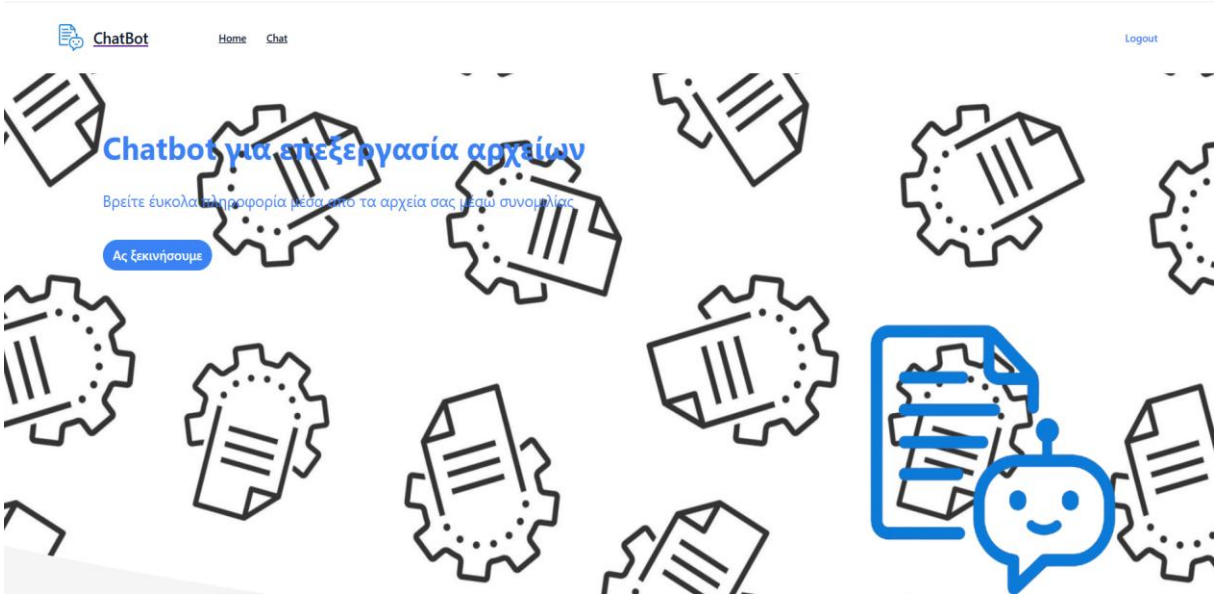


Σχήμα 3.3 : Κλήση APIS μέσω Postman

3.4 Διεπαφή Χρήστη (UI)

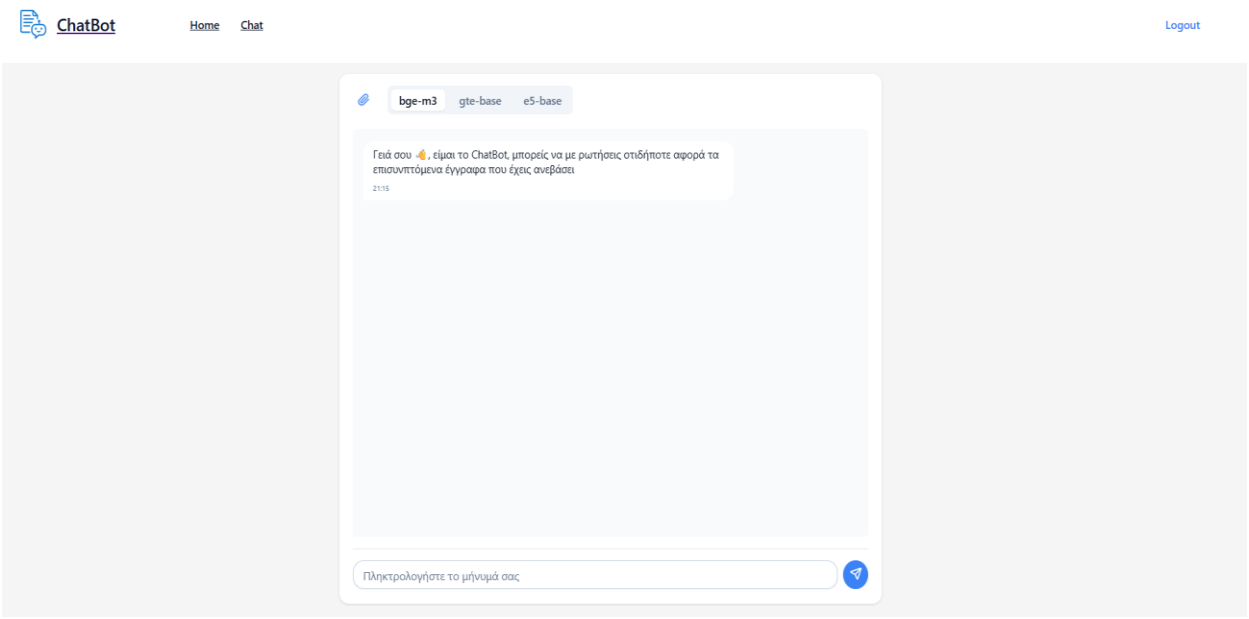
Η διεπαφή χρήστη υλοποιήθηκε ως Angular Single Page Application και οργανώθηκε σε επιμέρους σελίδες (pages) και services.

Παρακάτω παρουσιάζεται η κύρια οθόνη της εφαρμογής :



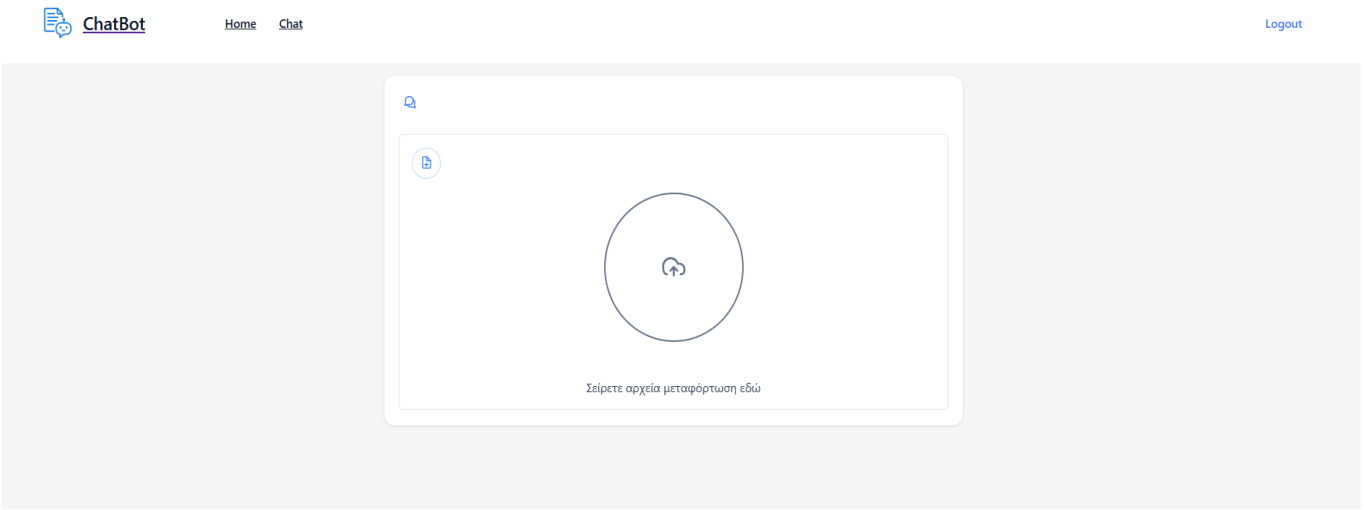
Σχήμα 3.4 : Αρχική οθόνη εφαρμογής

Η κύρια σελίδα αλληλεπίδρασης του χρήστη είναι η σελίδα συνομιλίας, η οποία βρίσκεται στο αρχείο *pages/chat/chat.ts*. Η σελίδα αυτή επιτρέπει στον χρήστη να πληκτρολογεί ερωτήσεις, να αποστέλλει μηνύματα και να λαμβάνει απαντήσεις από το backend. Επιπλέον, δίνεται η δυνατότητα επιλογής embedding model, ώστε ο χρήστης ή ο ερευνητής να μπορεί να αξιολογεί την επίδραση διαφορετικών embeddings στο retrieval.

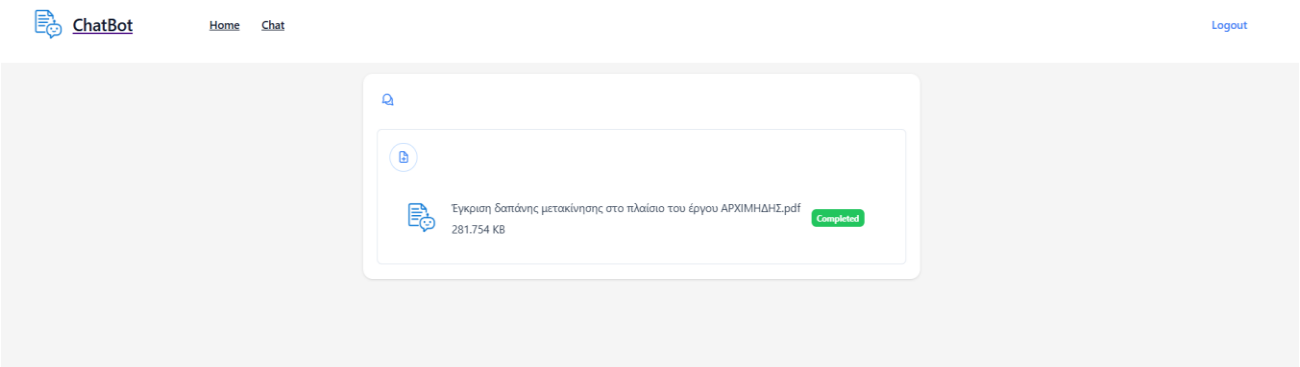


Σχήμα 3.5 Οθόνη του /chat endpoint

Για τη μεταφόρτωση εγγράφων χρησιμοποιείται το component *pages/files/files.ts*, το οποίο μπορεί να εμφανίζεται ή να αποκρύπτεται από τη σελίδα chat μέσω της μεθόδου *toggleFiles()*. Η λογική αυτή επιτρέπει στον χρήστη να ανεβάζει νέα έγγραφα χωρίς να εγκαταλείπει τη συνομιλία, βελτιώνοντας τη συνολική εμπειρία χρήσης.



Σχήμα 3.6 : Οθόνη upload αρχείων



Σχήμα 3.7 : Παράδειγμα επιτυχούς upload αρχείου

Η επικοινωνία με το backend υλοποιείται μέσω του service *pages/chat/chat-service.ts*. Το service αυτό περιλαμβάνει μεθόδους που καλούν τα endpoints του backend, όπως η μέθοδος *chat(...)* για την αποστολή ερωτήματος και η μέθοδος *uploadDocument(...)* για τη μεταφόρτωση αρχείων. Η χρήση service layer προσφέρει καθαρό διαχωρισμό ανάμεσα στο UI και τη λογική επικοινωνίας, ενώ επιτρέπει την ευκολότερη συντήρηση του κώδικα.

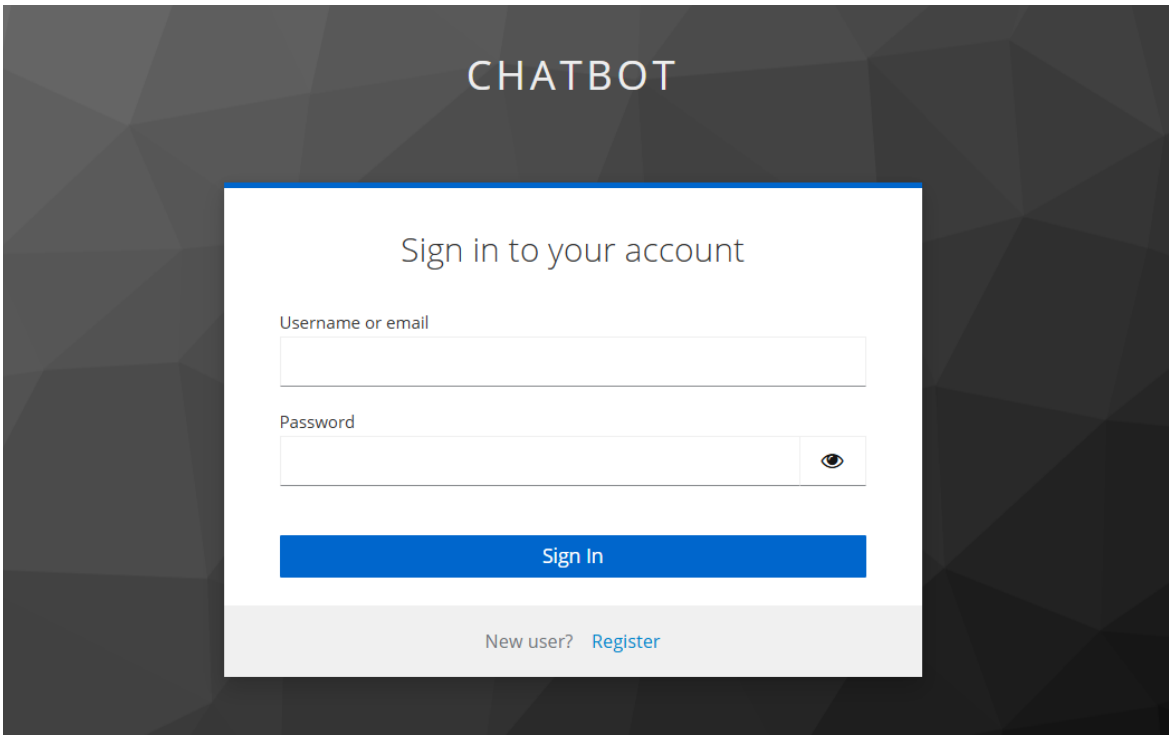
Παράλληλα, η εφαρμογή περιλαμβάνει σελίδες όπως home και fallback, οι οποίες συμβάλλουν στη δομή και την πλοήγηση της SPA. Συνολικά, η Angular υλοποίηση επιτρέπει σύγχρονο, επεκτάσιμο και φιλικό προς τον χρήστη περιβάλλον αλληλεπίδρασης.

3.5 Ασφάλεια και Πιστοποίηση Χρηστών

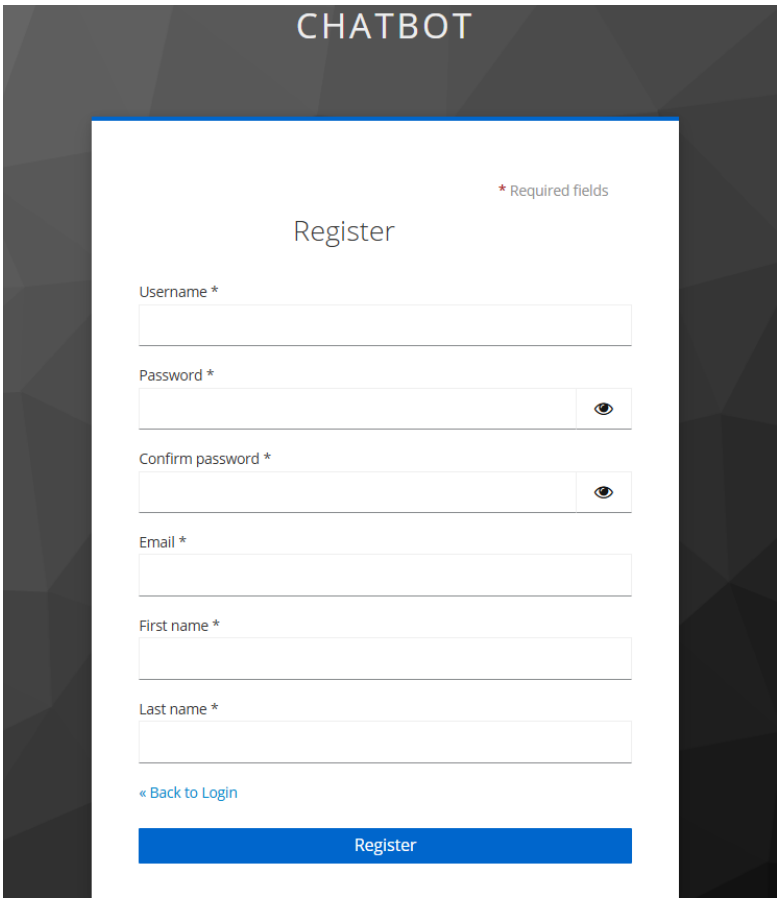
Η ασφάλεια του συστήματος αποτελεί βασικό στοιχείο, καθώς η εφαρμογή διαχειρίζεται έγγραφα χρηστών και παρέχει δυνατότητα πρόσβασης σε αποθηκευμένο περιεχόμενο. Για τον λόγο αυτό, χρησιμοποιήθηκε ο Keycloak ως Identity Provider, προσφέροντας authentication μέσω OpenID Connect και έκδοση JWT access tokens.

Στο backend, η προστασία των endpoints υλοποιείται μέσω dependency injection του FastAPI, χρησιμοποιώντας το *dependency (require_token*) στα endpoints /upload και /chat. Η μέθοδος require_token ορίζεται στο αρχείο *auth.py* και είναι υπεύθυνη για την επαλήθευση του JWT token που αποστέλλεται από το UI. Η επαλήθευση πραγματοποιείται με χρήση των δημόσιων κλειδιών του Keycloak (JWKS). Ο κώδικας κατεβάζει τα κλειδιά από το endpoint των certificates του Keycloak και τα αποθηκεύει προσωρινά με caching, ώστε να αποφεύγονται συνεχείς επαναλήψεις δικτυακών αιτημάτων.

Κατά την αποκωδικοποίηση του token, ελέγχονται κρίσιμες παράμετροι όπως ο issuer και το audience, ώστε να διασφαλιστεί ότι το token είναι έγκυρο και εκδόθηκε από το σωστό σύστημα. Σε περίπτωση αποτυχίας, το backend επιστρέφει HTTP με κωδικό 401 Unauthorized. Με αυτόν τον τρόπο, διασφαλίζεται ότι μόνο πιστοποιημένοι χρήστες μπορούν να ανεβάζουν έγγραφα ή να υποβάλλουν ερωτήσεις, προστατεύοντας το σύστημα από μη εξουσιοδοτημένη χρήση.



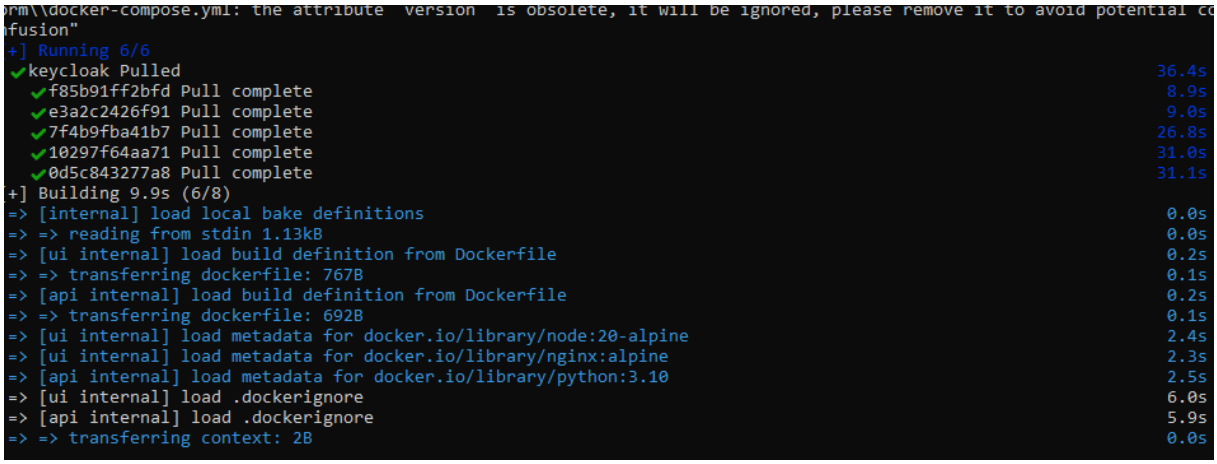
Σχήμα 3.8 : Οθόνη Login



Σχήμα 3.9 : Οθόνη register χρήστη

3.6 Ανάπτυξη και Διάθεση της Εφαρμογής (Docker)

Για την ανάπτυξη και διάθεση της εφαρμογής επιλέχθηκε η χρήση Docker, καθώς προσφέρει αναπαραγωγιμότητα περιβάλλοντος, απομόνωση υπηρεσιών και ευκολία εγκατάστασης. Η εφαρμογή εκτελείται ως σύνολο containers που περιλαμβάνει το Angular UI, το FastAPI backend και τον Keycloak server. Με αυτόν τον τρόπο, η εκκίνηση της εφαρμογής γίνεται ενιαία μέσω Docker Compose, χωρίς να απαιτείται χειροκίνητη εγκατάσταση εξαρτήσεων σε κάθε νέο περιβάλλον.



Σχήμα 3.10 : Στιγμιότυπο κατά τη διάρκεια του «χτησίματος» του container τοπικά στον υπολογιστή.

3.7 Επίλογος

Στην παρούσα πτυχιακή εργασία, το RAG αξιοποιείται για την ανάπτυξη ενός chatbot που απαντά σε ερωτήματα χρηστών με βάση το περιεχόμενο εγγράφων του ελληνικού δημόσιου τομέα. Η διαδικασία λειτουργίας του συστήματος συνοψίζεται ως εξής:

1. Τα έγγραφα εισάγονται στο σύστημα και μετατρέπονται σε αποσπάσματα κειμένου.
2. Για κάθε απόσπασμα παράγονται embeddings μέσω εξειδικευμένων μοντέλων σημασιολογικής αναπαράστασης.
3. Τα embeddings αποθηκεύονται σε διανυσματική δομή αναζήτησης (FAISS).
4. Όταν ο χρήστης υποβάλει ερώτημα, το σύστημα αναζητά τα πιο σχετικά αποσπάσματα.
5. Τα αποσπάσματα αυτά εισάγονται στο LLM μαζί με οδηγίες ώστε η απάντηση να παραχθεί αποκλειστικά από το παρεχόμενο περιεχόμενο.

Με τον τρόπο αυτό, το chatbot μπορεί να παρέχει απαντήσεις που είναι περισσότερο αξιόπιστες, τεκμηριωμένες και σχετικές με το υλικό που έχει εισαχθεί, περιορίζοντας ταυτόχρονα την πιθανότητα παραγωγής ανακριβών ή αυθαίρετων απαντήσεων.

Κεφάλαιο 4ο: Πειράματα και αποτελέσματα

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιαστούν τα πειράματα που έγιναν και τα αποτελέσματα που συγκεντρώθηκαν στα πλαίσια σύγκρισης απόδοσης και ακρίβεια απαντήσεων, των embedding μοντέλων που επιλέχθηκαν.

Αξίζει να σημειωθεί ότι κατά την υλοποίηση του συστήματος έγιναν διάφορες παραδοχές και υποθέσεις, στη διαδικασία ανάκτησης/απάντησης, ώστε να προκύψουν όσο το δυνατόν πιο σταθερά και συγκρίσιμα αποτελέσματα στα πειράματα. Οι παραδοχές αυτές αναλύονται παρακάτω :

Στο συγκεκριμένο έργο εφαρμόζεται στρατηγική ανάκτησης με **k_files = 1**, δηλαδή το σύστημα:

1. αναζητά αρχικά τα πιο σχετικά αποσπάσματα σε όλο το αποθετήριο,
2. επιλέγει ένα "καλύτερο" αρχείο (με βάση το μέγιστο score των chunks του),
3. και στη συνέχεια επιστρέφει τα k_per_file πιο σχετικά chunks μόνο από αυτό το αρχείο.

Η παραπάνω επιλογή σχεδιασμού σημαίνει ότι τα benchmarks δεν αξιολογούν την ικανότητα του συστήματος να «συνδυάζει πληροφορίες από πολλά έγγραφα», αλλά επικεντρώνεται στην απόδοση του embedding μοντέλου υπό τις ίδιες παραμέτρους αναζήτησης.

4.2 Συλλογή Dataset

Ως κύρια πηγή για τη συλλογή των εγγράφων επιλέχθηκε το site της Διαύγειας (https://diavgeia.gov.gr/). Η πλατφόρμα αυτή αποτελεί κεντρικό σημείο ανάρτησης πράξεων των ελληνικών κυβερνητικών και διοικητικών οργάνων, προσφέροντας ένα ευρύ φάσμα δημοσίων εγγράφων. Διεξήχθη διερευνητική έρευνα στην πλατφόρμα για τον εντοπισμό διαφόρων τύπων εγγράφων και πιθανών περιπτώσεων χρήσης (use cases) που θα μπορούσαν να αξιοποιηθούν.

Η διαδικασία συλλογής δεδομένων και κατασκευής του dataset έγινε με τους εξής άξονες :

- **Ποικιλομορφία των εγγράφων:** Επιλέχθηκαν έγγραφα διαφόρων φορέων, όπως του Συνηγόρου του Καταναλωτή, Νοσοκομείων, Δήμων και του Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης που ανήκουν σε διάφορες θεματικές, ενδεικτικά αναφέρονται οι Οικονομικές και Εμπορικές Συναλλαγές, η Δημόσια Διοίκηση, η Παραγωγή, Τεχνολογία και Έρευνα – Επιστήμες.

- **Αναζητήσιμα έγγραφα (Searchability):** Επιβεβαιώθηκε ότι όλες οι εξεταζόμενες πράξεις ήταν "searchable", δηλαδή το κείμενό τους ήταν ψηφιακά αναγνωρίσιμο και μπορούσε να εξαχθεί χωρίς την ανάγκη εφαρμογής τεχνικών Οπτικής Αναγνώρισης Χαρακτήρων (OCR). Η παραδοχή αυτή απλοποίησε σημαντικά το στάδιο της εξαγωγής κειμένου και παραγωγής των embeddings**.**

- **Ετερογένεια στη δομή και στη μορφοποίηση των εγγράφων:** Παρότι τα διοικητικά έγγραφα ακολουθούν σε γενικές γραμμές κοινά πρότυπα, εντοπίστηκαν διαφοροποιήσεις στη διάταξη, στη χρήση γραμματοσειρών, στην ύπαρξη πινάκων ή πεδίων φόρμας, καθώς και στην οργάνωση των ενοτήτων. Παραδείγματος χάρη, πολλά έγγραφα περιείχαν πίνακες με πληροφορίες όπως απαιτούμενα ποσά, ή στοιχεία επιτροπών. Οι διαφοροποιήσεις αυτές εμφανίζονταν τόσο μεταξύ εγγράφων διαφορετικών φορέων, όσο και μεταξύ εγγράφων του ίδιου φορέα σε διαφορετικές χρονικές περιόδους. Η πρόκληση αυτή ανέδειξε την ανάγκη για ευέλικτες τεχνικές

προεπεξεργασίας και τμηματοποίησης κειμένου, ώστε το σύστημα να μπορεί να υποστηρίζει αξιόπιστα την ανάκτηση πληροφορίας σε περιβάλλον πραγματικών εγγράφων του δημόσιου τομέα.

4.3 Σύνολο Ερωτήσεων

Για την αξιολόγηση της απόδοσης του συστήματος RAG και ειδικότερα της σύγκρισης των τριών embedding models, δημιουργήθηκε ένα σύνολο 10 αντιπροσωπευτικών ερωτήσεων αξιολόγησης, το οποίο χρησιμοποιήθηκε με κοινό τρόπο σε όλα τα πειράματα. Οι ερωτήσεις επιλέχθηκαν με στόχο να προσομοιώνουν ρεαλιστικά σενάρια χρήσης, όπου ένας χρήστης επιθυμεί να αντλήσει συγκεκριμένη πληροφορία από διοικητικά έγγραφα, είτε όσον αφορά δαπάνες που συμφωνήθηκαν στα πλαίσια έργων, μέλη επιτροπών που συστάθηκαν καθώς και σύμπτυξη συμβάσεων που εγκρίθηκαν.

Αξίζει να σημειωθεί πως κάθε ερώτηση σχεδιάστηκε έτσι ώστε η απάντησή της να μπορεί να εντοπιστεί μέσα στο κείμενο των επιλεγμένων εγγράφων, χωρίς να απαιτείται εξωτερική γνώση. Με αυτόν τον τρόπο, η αξιολόγηση επικεντρώνεται στη δυνατότητα του συστήματος να ανακτά το κατάλληλο απόσπασμα και να παράγει απάντηση αποκλειστικά από το παρεχόμενο context.

Τέλος, για να εξασφαλιστεί η συγκρισιμότητα των αποτελεσμάτων, το ίδιο ακριβώς σύνολο ερωτήσεων χρησιμοποιήθηκε σε όλα τα embedding models. Με αυτόν τον τρόπο, οι παρατηρούμενες διαφορές μπορούν να αποδοθούν κυρίως στην επιλογή του embedding μοντέλου και όχι σε αλλαγές στο σύνολο των ερωτήσεων ή στις ρυθμίσεις του συστήματος.

Παρακάτω, στον **Πίνακα 1**, παρουσιάζονται οι ερωτήσεις που επιλέχθηκαν, αντιστοιχισμένες ανά αρχείο.

Πίνακας 1 : Αντιστοίχιση ερώτησης ανά αρχείο

No	Ερώτηση	Αρχείο
1	Σε ποιον ανατέθηκε η αμοιβή για τις εργασίες επισκευής του υπ.αριθ. ΚΥ 9089(ΙΝΚ 8982);	Απόφαση ανάθεσης αμοιβής για εργασίες συντήρησης και επισκευής του υπ.αριθ. ΚΥ 9089(ΙΝΚ 8982) υπηρεσιακού αυτοκίνητου της ΔΟΥ Τρικάλων.
2	Τι δαπάνη εγκρίθηκε για το έργο ΑΡΧΙΜΗΔΗΣ;	Έγκριση δαπάνης μετακίνησης στο πλαίσιο του έργου ΑΡΧΙΜΗΔΗΣ.
3	Τι αποφασίστηκε για την απευθείας ανάθεση της συναυλίας της ΔΕΘ και πόσο θα κοστίσει;	Απόφαση Δημάρχου για την απευθείας ανάθεση της συναυλίας.
4	Από ποιους υπαλλήλους συγκροτήθηκε επιτροπή παραλαβής υλικών και υπηρεσιών για την εύρυθμη λειτουργίας της Δ.Ο.Υ. Ναυπλίου για το έτος 2026;	Απόφαση συγκρότησης επιτροπής προμηθειών και εργασιών της ΔΟΥ Ναυπλίου για το έτος 2026
5	Τι συμφωνήθηκε στη σύμβαση 684/2025;	ΣΥΜΒΑΣΗ 684_2025 UNISON

6	Τι επιτροπές συγκροτήθηκαν από το Συνήγορο του Καταναλωτή για την παρακολούθηση εκτέλεσης συμβάσεων και την παραλαβή προμηθειών;	Συγκρότηση Επιτροπών για την παρακολούθηση εκτέλεσης συμβάσεων και την παραλαβή προμηθειών (υλικών και υπηρεσιών) της Ανεξάρτητης Αρχής "Συνήγορος του Καταναλωτή"
7	Ποια είναι τα σημαντικότερα σημεία της σύμβασης μίσθωσης της Ετήσιας Γεωργικής Έρευνας;	Σύμβαση Μίσθωσης Ανάθεσης Έργου Ετήσια Γεωργική Έρευνα (ΕΓΕ)
8	Ποιες είναι οι τεχνικές προδιαγραφές της σύμβασης παροχής υπηρεσιών καθαριότητας του τελωνείου του Βόλου το έτος 2026;	Σύμβαση Παροχής Υπηρεσιών Καθαριότητας Τελωνείου Βόλου έτους 2026
9	Ποια μέλη παρευρέθηκαν στη συνεδρίαση;	Συνεδριαση δημοτικης επιτροπης
10	Ποιο είναι το Πρωτόκολλο Ανάληψης Υποχρέωσης του χρηματικού εντάλματος 79988;	Χρηματικό Ένταλμα 79988_2025

4.4 Πειράματα

Στο πλαίσιο του κεφαλαίου των πειραμάτων παρουσιάζονται πίνακες από τις ερωτήσεις που υποβλήθηκαν στο σύστημα μαζί με τις απαντήσεις που παρήχθησαν, καθώς και ένας συγκεντρωτικός πίνακας με μετρικές απόδοσης (benchmarks).

4.4.1 Ερωτήσεις - Απαντήσεις στα Embeddings μοντέλα

Για την αξιολόγηση της ανάκτησης (retrieval) απαντήσεων του RAG συστήματος συγκρίθηκαν, όπως έχουμε αναφέρει και στο κεφάλαιο των υλοποιήσεων, τρία διαφορετικά embedding models: BAAI/bge-m3, Alibaba-NLP/gte-multilingual-base και intfloat/multilingual-e5-base.

Για λόγους δίκαιης σύγκρισης, χρησιμοποιήθηκε το ίδιο σύνολο 10 ερωτήσεων για κάθε μοντέλο και διατηρήθηκαν σταθερές οι παράμετροι ανάκτησης. Επιπλέον, εφαρμόστηκε στρατηγική **k_files = 1**, σύμφωνα με την οποία το σύστημα επιλέγει αρχικά το «καλύτερο» αρχείο (με βάση τα αποτελέσματα ανάκτησης) και στη συνέχεια επιστρέφει **k_per_file = 5** αποσπάσματα (chunks) μόνο από αυτό το αρχείο, τα οποία δίνονται ως context στο γλωσσικό μοντέλο Gemini, για την παραγωγή απάντησης.

Παρακάτω παρουσιάζονται όλα τα πειράματα που διεξήχθησαν μέσω της UI διεπαφής μαζί με τις απαντήσεις που δόθηκαν από όλα τα μοντέλα. Σε όλα τα πειράματα οι ερώτησης απαντήθηκαν πρώτα με χρήση του μοντέλου BAAI/bge-m3, στη συνέχεια μέσω του μοντέλου Alibaba-NLP/gte-multilingual-base και τέλος μέσω του μοντέλου intfloat/multilingual-e5-base. Οι ερωτήσεις είναι με την σειρά που παρουσιάζονται στον Πίνακα 1 στην υποενότητα Σύνολο Ερωτήσεων.

Πείραμα Ερώτησης 1:

Πίνακας 2.1 : Απάντηση ανά μοντέλο Ερώτησης 1

Μοντέλο	**Απάντηση**
bge-m3	Δεν βρέθηκε απάντηση.
gte-multilingual-base	Η απάντηση αναφέρει ότι η οφειλή ανήκει στην ΑΦΟΙ ΕΡΓΑΜΑ Ο.Ε., αποδίδοντας αναλυτικά τις εργασίες επισκευής και την επιμέλεια του πλοίου.
multilingual-e5-base	Η οφειλή για τις εργασίες επισκευής του υπ' αριθ. ΚΥ 9089/ΝΚ 8982 ανήκει στην ΑΦΟΙ ΕΡΓΑΜΑ Ο.Ε., σύμφωνα με τα στοιχεία που περιλαμβάνονται στο σχετικό έγγραφο.

Πείραμα Ερώτησης 2:

Πίνακας 2.2 : Απάντηση ανά μοντέλο Ερώτησης 2

Μοντέλο	**Απάντηση**
bge-m3	Εγκρίθηκε δαπάνη για την εκτέλεση συνεργείας ID-9256 μεταξύ Αθήνας – Πόρτο, ύψους 3.036,00 €.
gte-multilingual-base	Εγκρίθηκε δαπάνη μετακίνησης συνεργάτη με ID-9256 μεταξύ Αθήνας – Πόρτο, η οποία προυπολογίζεται στο ύψος των 3.036,00 €.
multilingual-e5-base	Εγκρίθηκε η μετακίνηση συνεργάτη με ID-9256 μεταξύ Αθήνας – Πόρτο με επιστροφή αεροπορικώς από 14/09/25 – 20/09/2025, για συμμετοχή στο ECMLPKDD 2025 για παρουσίαση paper.

Πείραμα Ερώτησης 3:

Πίνακας 2.3 : Απάντηση ανά μοντέλο Ερώτησης 3

Μοντέλο	**Απάντηση**
bge-m3	Αποφασίστηκε η έγκριση της απευθείας ανάθεσης για τη «Διοργάνωση συναυλίας – καλλιτεχνικού προγράμματος ηλεκτρονικής μουσικής μουσικών παραγωγών (Dj) στο πλαίσιο συμμετοχής του Δήμου Θεσσαλονίκης στη Διεθνή Έκθεση Θεσσαλονίκης (Δ.Ε.Θ.)». Το κόστος θα είναι 2.480,00 € με ΦΠΑ 24%.

gte-multilingual-base	Εγκρίθηκε και ανατέθηκε η απευθείας ανάθεση για τη «Διοργάνωση συναυλίας – καλλιτεχνικού προγράμματος ηλεκτρονικής μουσικής μουσικών παραγωγών (Dj) στο πλαίσιο συμμετοχής του Δήμου Θεσσαλονίκης στη Διεθνή Έκθεση Θεσσαλονίκης (Δ.Ε.Θ.)». Το κόστος ανέρχεται σε 2.480,00 € με ΦΠΑ 24%.
multilingual-e5-base	Εγκρίθηκε η απευθείας ανάθεση για τη «Διοργάνωση συναυλίας – καλλιτεχνικού προγράμματος ηλεκτρονικής μουσικής μουσικών παραγωγών (Dj) στο πλαίσιο συμμετοχής του Δήμου Θεσσαλονίκης στη Διεθνή Έκθεση Θεσσαλονίκης (Δ.Ε.Θ.)». Το συνολικό κόστος ανέρχεται σε 2.480,00 € συμπεριλβανομένου ΦΠΑ 24%.

Πείραμα Ερώτησης 4:

Πίνακας 2.4 : Απάντηση ανά μοντέλο Ερώτησης 4

Μοντέλο	**Απάντηση**
bge-m3	Η επιτροπή συγκροτήθηκε από τους εξής υπαλλήλους: ΤΑΚΤΙΚΑ ΜΕΛΗ: 1. Κουτσουμπάκη Ευαγγελία, ΤΕ/Α, ως πρόεδρος 2. Καραγιάννη Ευαγγελία, ΠΕ/Α, ως μέλος 3. Χριστοδούλου Ευάγγελος, ΤΕ/Α, ως μέλος ΑΝΑΠΛΗΡΩΜΑΤΙΚΑ ΜΕΛΗ: 1. Πασχαλίδης Μιχαήλ, ΤΕ/Α, ως αναπληρωτής πρόεδρος 2. Παπαδοπούλου Μαρία, ΠΕ/Α, ως αναπληρωματικό μέλος 3. Παππάς Δημήτριος, ΠΕ/Α, ως αναπληρωματικό μέλος
gte-multilingual-base	Η επιτροπή παραλαβής υλικών και υπηρεσιών για την ισάρυθμη λειτουργία της Δ.Ο.Υ. Ναυπλίου για το έτος 2026 συγκροτήθηκε από τους εξής υπαλλήλους: ΤΑΚΤΙΚΑ ΜΕΛΗ: 1. Κουτσουμπάκη Ευαγγελία, ΤΕ, ως πρόεδρος

	2. Καραγιάννη Ευαγγελία, ΠΕ, ως μέλος 3. Χριστοδούλου Ευάγγελος, ΠΕ, ως μέλος ΑΝΑΠΛΗΡΩΜΑΤΙΚΑ ΜΕΛΗ: 1. Πασχαλίδης Μιχαήλ, ΠΕ, ως αναπληρωτής πρόεδρος 2. Παπαδοπούλου Μαρία, ΠΕ, ως αναπληρωματικό μέλος 3. Παππάς Δημήτριος, ΠΕ, ως αναπληρωματικό μέλος
multilingual-e5-base	Η επιτροπή συγκροτήθηκε από τους παρακάτω υπαλλήλους: ΤΑΚΤΙΚΑ ΜΕΛΗ: 1. Κουτσουμπάκη Ευαγγελία, ΤΕ/Α, ως πρόεδρος 2. Καραγιάννη Ευαγγελία, ΠΕ/Α, ως μέλος 3. Χριστοδούλου Ευάγγελος, ΤΕ/Α, ως μέλος ΑΝΑΠΛΗΡΩΜΑΤΙΚΑ ΜΕΛΗ: 1. Πασχαλίδης Μιχαήλ, ΤΕ/Α, ως αναπληρωτής πρόεδρος 2. Παπαδοπούλου Μαρία, ΠΕ/Α, ως αναπληρωματικό μέλος 3. Παππάς Δημήτριος, ΠΕ/Α, ως αναπληρωματικό μέλος

Πείραμα Ερώτησης 5:

Πίνακας 2.5 : Απάντηση ανά μοντέλο Ερώτησης 5

Μοντέλο	**Απάντηση**
bge-m3	Δεν βρέθηκε απάντηση.
gte-multilingual-base	Στη σύμβαση 684/2025 συμφωνήθηκαν «ΥΠΗΡΕΣΙΕΣ ΕΠΙΣΚΕΥΗΣ ΚΑΙ ΣΥΝΤΗΡΗΣΗΣ ΗΛΕΚΤΡΟΜΗΧΑΝΟΛΟΓΙΚΩΝ ΕΓΚΑΤΑΣΤΑΣΕΩΝ ΚΤΙΡΙΟΥ» για τις ανάγκες του Γ.Ν. Βόλου.
multilingual-e5-base	Δεν βρέθηκε απάντηση.

Πείραμα Ερώτησης 6:

Πίνακας 2.6 : Απάντηση ανά μοντέλο Ερώτησης 6

Μοντέλο	**Απάντηση**
bge-m3	Συγκροτήθηκαν οι εξής επιτροπές: Α. Επιτροπή Παρακολούθησης Εκτέλεσης Συμβάσεων Προμήθειας και Παραλαβής Υλικών. Β. Επιτροπή Παρακολούθησης Εκτέλεσης Συμβάσεων Παροχής Υπηρεσιών.
gte-multilingual-base	Συγκροτήθηκαν οι εξής επιτροπές: Α. Επιτροπή Παρακολούθησης Εκτέλεσης Συμβάσεων Προμήθειας και Παραλαβής Υλικών. Β. Επιτροπή Παρακολούθησης Εκτέλεσης Συμβάσεων Παροχής Υπηρεσιών.
multilingual-e5-base	Συγκροτήθηκαν οι εξής επιτροπές: * Επιτροπή Παρακολούθησης Εκτέλεσης Συμβάσεων Προμήθειας και Παραλαβής Υλικών. * Επιτροπή Παρακολούθησης Εκτέλεσης Συμβάσεων Παροχής Υπηρεσιών.

Πείραμα Ερώτησης 7:

Πίνακας 2.7 : Απάντηση ανά μοντέλο Ερώτησης 7

Μοντέλο	**Απάντηση**
bge-m3	Τα σημαντικότερα σημεία της σύμβασης μίσθωσης της Ετήσιας Γεωργικής Έρευνας είναι: • Αντικείμενο: Διενέργεια της Ετήσιας Γεωργικής Έρευνας (ΕΓΕ) 2025–2026, με ανάθεση 21 ερωτηματολογίων στον Ιδιώτη Συνεργάτη (ΙΣ). • Μέρη: Η Ελληνική Στατιστική Αρχή (ΕΛΣΤΑΤ) και ο Ιδιώτης Συνεργάτης ΣΤΥΛΙΑΝΗ ΜΑΛΛΑΡΗ. • Διάρκεια: Από την ημερομηνία υπογραφής της σύμβασης από την ΕΛΣΤΑΤ έως την 30/04/2026. • Υποχρεώσεις ΙΣ: Τήρηση όλων των όρων της

	σύμβασης, παρακολούθηση μη αμειβώμενου εκπαιδευτικού σεμιναρίου,παράδοση συμπληρωμένων ερωτηματολογίων, τήρηση εμπιστευτικότητας και στατιστικού απορρήτου, εκτέλεση οδηγιών της ΕΛΣΤΑΤ. • Υποχρεώσεις ΕΛΣΤΑΤ: Καταβολή του συμφωνηθέντος τιμήματος. • Ιδιωκτησία: Τα ερωτηματολόγια και όλο το υλικό της έρευνας αποτελούν ιδιοκτησία της ΕΛΣΤΑΤ. • Αναπόσπαστα μέρη:Η Πρόσκληση Εκδήλωσης Ενδιαφέροντος , η Δήλωση Ιδιώτη Συνεργάτη και η Δήλωση Τήρησης Στατιστικού Απορρήτου.
gte-multilingual-base	Τα σημαντικότερα σημεία της σύμβασης μίσθωσης της Ετήσιας Γεωργικής Έρευνας είναι: • Αντικείμενο: Ετήσια Γεωργική Έρευνας (ΕΓΕ) 2025–2026, με ανάθεση 21 ερωτηματολογίων στον Ιδιώτη Συνεργάτη (ΙΣ) στον Δήμο Ιεράπετρας. • Μέρη: Η Ελληνική Στατιστική Αρχή (ΕΛΣΤΑΤ) και η ΜΑΛΛΑΡΗ ΣΤΥΛΙΑΝΗ ως Ιδιώτης Συνεργάτης. • Διάρκεια: Από την ημερομηνία υπογραφής της σύμβασης από την ΕΛΣΤΑΤ έως την 30/04/2026. • Υποχρεώσεις ΙΣ: Τήρηση όλων των όρων της σύμβασης και των αναπόσπαστων μερών της (Πρόσκληση Εκδήλωσης Ενδιαφέροντος κ.λπ) παρακολούθηση μη αμειβώμενου εκπαιδευτικού σεμιναρίου,εκτέλεση οδηγιών της ΕΛΣΤΑΤ, παράδοση ερωτηματολογίων έως την λήξη. Η εργασία πραγματοποιείται σε ώρες και ημέρες επιλογής του ΙΣ, εκτός έδρας της ΕΛΣΤΑΤ. • Υποχρεώσεις ΕΛΣΤΑΤ: Καταβολή του συμφωνηθέντος τιμήματος. • Δικαιώματα: Τα ερωτηματολόγια και όλο το υλικό της έρευνας αποτελούν ιδιοκτησία της ΕΛΣΤΑΤ. • Εμπιστευτικότητα /Απόρρητο: Ο ΙΣ υποχρεούται να τηρεί την εμπιστευτικότητα και το στατιστικό απόρρητο στο διηνεκές.

	• Αναπόσπαστα Μέρη: Η Πρόσκληση Εκδήλωσης Ενδιαφέροντος , η Δήλωση Ιδιώτη Συνεργάτη και η Δήλωση Τήρησης Στατιστικού Απορρήτου και όλα τα έντυπα που υπογράφονται στο πλαίσιο της Πρόσκλησης.
multilingual-e5-base	Τα σημαντικότερα σημεία της σύμβασης μίσθωσης της Ετήσιας Γεωργικής Έρευνας (ΕΓΕ) είναι: • Αντικείμενο: Διενέργεια της Ετήσιας Γεωργικής Έρευνας (ΕΓΕ) 2025–2026, με ανάθεση 21 ερωτηματολογίων στον Ιδιώτη Συνεργάτη (ΙΣ). • Διάρκεια: Η σύμβαση ισχύει από την ημερομηνία υπογραφής της από την ΕΛΣΤΑΤ και λήγει την 30/04/2026. • Υποχρεώσεις ΙΣ: Τήρηση όλων των όρων της σύμβασης, παρακολούθηση και συμμετοχή στην εκπαίδευση, παράδοση συμπληρωμένων ερωτηματολογίων, εκτέλεση οδηγιών της ΕΛΣΤΑΤ. • Ιδιοκτησία: Τα ερωτηματολόγια και όλο το υλικό της έρευνας αποτελούν ιδιοκτησία της ΕΛΣΤΑΤ. • Εμπιστευτικότητα – Απόρρητο: Ο Ιδιώτης υποχρεούται να τηρεί την εμπιστευτικότητα και το στατιστικό απόρρητο. • Συμβαλλόμενα Μέρη: Ελληνική Στατιστική Αρχή (ΕΛΣΤΑΤ) και ΜΑΛΛΑΡΗ ΣΤΥΛΙΑΝΗ ως Ιδιώτης Συνεργάτης.

Πείραμα Ερώτησης 8:

Πίνακας 2.8: Απάντηση ανά μοντέλο Ερώτησης 8

Μοντέλο	**Απάντηση**
bge-m3	Οι τεχνικές προδιαγραφές της σύμβασης είναι οι εξής: • Ημέρες εργασίας: Δευτέρα-Παρασκευή. • Απασχόληση: Τρίωρη ημερησίως, με (1) άτομο. • Κάλυψη χώρων: Περίπου 430 τ.μ. εσωτερικών χώρων (γραφείων) και εξωτερικών χώρων

	περιοδικά. • Παρεχόμενες υπηρεσίες: Ημερήσιοι καθαρισμοί, συμπεριλαμβανομένου του καθαρισμού γραφειακών χώρων (ξεσκόνισμα επίπλων, γραφείων, καθισμάτων, ηλεκτρονικών υπολογιστών, τηλεφώνων και άλλων μικροαντικειμένων.
gte-multilingual-base	Οι τεχνικές προδιαγραφές της σύμβασης είναι οι εξής: • Ημέρες εργασίας: Δευτέρα-Παρασκευή. • Απασχόληση: Τρίωρη απασχόληση ημερησίως, με (1) άτομο. • Κάλυψη χώρων: 430 τ.μ. περίπου εσωτερικών χώρων (γραφείων) και εξωτερικών χώρων περιοδικά. • Παρεχόμενες υπηρεσίες: Ημερήσιοι καθαρισμοί, συμπεριλαμβανομένου του καθαρισμού γραφειακών χώρων (ξεσκόνισμα επίπλων, γραφείων, καθισμάτων, ηλεκτρονικών υπολογιστών, τηλεφώνων και άλλων μικροαντικειμένων.
multilingual-e5-base	Οι τεχνικές προδιαγραφές της σύμβασης παροχής υπηρεσιών καθαριότητας είναι: • Ημέρες εργασίας: Δευτέρα-Παρασκευή, τρίωρη απασχόληση ημερησίως, με (1) άτομο. • Κάλυψη χώρων: Περίπου 430 τ.μ. εσωτερικών χώρων (γραφείων) και εξωτερικών χώρων περιοδικά. • Παρεχόμενες υπηρεσίες: Ημερήσιοι καθαρισμοί, συμπεριλαμβανομένου του καθαρισμού γραφειακών χώρων (ξεσκόνισμα επίπλων, γραφείων, καθισμάτων, ηλεκτρονικών υπολογιστών, τηλεφώνων και άλλων μικροαντικειμένων.

Πείραμα Ερώτησης 9:

Πίνακας 2.9 : Απάντηση ανά μοντέλο Ερώτησης 9

Μοντέλο	**Απάντηση**
bge-m3	Στη συνεδρίαση παρευρέθηκαν τα εξής πέντε μέλη: 1. Μακρόπουλος Γεώργιος (Δήμαρχος, Πρόεδρος Δ.Σ.) 2. Τσακίρης Ιωάννης (Δημοτικός Σύμβουλος, Τακτικό Μέλος Δ.Σ.) 3. Αργυρός Γεώργιος (Δημοτικός Σύμβουλος, Τακτικό Μέλος Δ.Σ.) 4. Θέμελης Ανδρέας (Δημοτικός Σύμβουλος, Αναπληρωματικό Μέλος Δ.Σ.) 5. Δραβίλας Ιωάννης (Δημοτικός Σύμβουλος, Αναπληρωματικό Μέλος Δ.Σ.)
gte-multilingual-base	Τα μέλη που παρευρέθηκαν στη συνεδρίαση είναι: 1. Μακρόπουλος Γεώργιος 2. Τσακίρης Ιωάννης 3. Αργυρός Γεώργιος 4. Θέμελης Ανδρέας 5. Δραβίλας Ιωάννης
multilingual-e5-base	Τα μέλη που παρευρέθηκαν στη συνεδρίαση είναι: 1. Μακρόπουλος Γεώργιος 2. Τσακίρης Ιωάννης 3. Αργυρός Γεώργιος 4. Θέμελης Ανδρέας 5. Δραβίλας Ιωάννης

Πείραμα Ερώτησης 10:

Πίνακας 2.10 : Απάντηση ανά μοντέλο Ερώτησης 10

Μοντέλο	**Απάντηση**
bge-m3	Το Πρωτόκολλο Ανάληψης Υποχρέωσης είναι 339124/31-12-2025.
gte-multilingual-base	Δεν βρέθηκε απάντηση.
multilingual-e5-base	339124/31-12-2025

4.4.2 Συγκεντρωτικός πίνακας αποτελεσμάτων

Ο Πίνακας 3 περιλαμβάνει τις ακόλουθες μετρικές:

- **Retrieval ms:** χρόνος ανάκτησης (embedding του ερωτήματος + αναζήτηση στο FAISS + επιλογή των Top-k αποσπασμάτων).
- **Total ms:** συνολικός χρόνος απόκρισης του συστήματος (ανάκτηση + κλήση στο LLM).
- **CPU Ret. ms:** χρόνος CPU που καταναλώθηκε κατά τη φάση ανάκτησης (δείκτης υπολογιστικού κόστους του retrieval pipeline).
- **Top1:** μέγιστη τιμή ομοιότητας (cosine similarity) μεταξύ ερωτήματος και του καλύτερου chunk που ανακτήθηκε.
- **MeanTop5:** μέση τιμή ομοιότητας των 5 chunks που δόθηκαν στο LLM ως context.
- **Βρέθηκε Απάντηση:** ένδειξη επιτυχούς παραγωγής απάντησης (σε αντίθετη περίπτωση επιστρέφεται «Δεν βρέθηκε απάντηση.» σύμφωνα με τους κανόνες του prompt).

Πίνακας 3 : Benchmarks

Ερ.	Μοντέλο	Retrieval ms	Total ms	CPU Ret. ms	Top1	MeanTop5	Βρέθηκε Απάντηση
1	bge-m3	242.2	6211.8	1415.5	0.5178	0.4454	×
1	gte-multilingual-base	168.1	1960.4	692.2	0.7877	0.7510	✓
1	multilingual-e5-base	164.5	3483.5	548.4	0.8276	0.8243	✓
2	bge-m3	521.7	2953.9	2968.6	0.5718	0.5509	✓
2	gte-multilingual-base	211.9	2460.4	1020.6	0.6948	0.6892	✓
2	multilingual-e5-base	195.5	4834.8	882.2	0.8433	0.8433	✓
3	bge-m3	571.7	3595.2	2880.1	0.6475	0.5221	✓
3	gte-multilingual-base	204.2	3780.9	811.8	0.7815	0.6794	✓
3	multilingual-e5-base	72.9	2606.4	393.8	0.8508	0.8245	✓
4	bge-m3	2432.9	5473.8	8879.4	0.7290	0.6698	✓
4	gte-multilingual-base	359.9	6212.4	16927.5	0.9213	0.8441	✓
4	multilingual-e5-base	1278.7	4066.9	5376.6	0.8610	0.8456	✓
5	bge-m3	788.7	3150.9	4443.3	0.4843	0.4452	×
5	gte-multilingual-base	1893.4	4178.9	10074.4	0.7399	0.6515	✓
5	multilingual-e5-base	1419.5	4117.8	7831.6	0.8239	0.8081	×
6	bge-m3	5192.5	7283.7	28409.7	0.7165	0.5987	✓
6	gte-multilingual-base	193.4	2171.7	1088.6	0.9055	0.7833	✓
6	multilingual-e5-base	1754.7	3405.5	9078.0	0.8456	0.8101	✓
7	bge-m3	2650.4	8267.2	15034.9	0.6086	0.4947	✓
7	gte-multilingual-base	478.5	6152.0	2262.7	0.8174	0.6680	✓

7	multilingual-e5-base	276.3	4082.0	904.6	0.8595	0.8364	✓
8	bge-m3	295.9	4532.6	1504.3	0.6980	0.6406	✓
8	gte-multilingual-base	124.9	2417.1	545.3	0.8774	0.8568	✓
8	multilingual-e5-base	76.8	2485.1	428.0	0.9064	0.8884	✓
9	bge-m3	249.4	2377.0	1170.6	0.5629	0.4742	✓
9	gte-multilingual-base	117.3	1534.0	428.1	0.7227	0.6081	✓
9	multilingual-e5-base	247.0	1571.7	527.5	0.7908	0.7708	✓
10	bge-m3	409.3	2967.9	1603.4	0.5201	0.4761	✓
10	gte-multilingual-base	190.6	2070.1	693.2	0.6837	0.6556	×
10	multilingual-e5-base	174.9	1457.2	535.0	0.8328	0.8205	✓

4.4.3 Ανάλυση αποτελεσμάτων

Από τα αποτελέσματα του Πίνακα 3 προκύπτουν σαφείς διαφορές μεταξύ των embedding models, τόσο σε επίπεδο ποιότητας ανάκτησης όσο και σε επίπεδο απόδοσης. Τα συμπεράσματα παρουσιάζονται αναλυτικότερα παρακάτω.

Ποιότητα ανάκτησης (Top1, MeanTop5)

Το multilingual-e5-base παρουσιάζει συνολικά τις υψηλότερες τιμές ομοιότητας (μέσο Top1 ≈ 0.844 και μέσο MeanTop5 ≈ 0.827), ενώ ακολουθεί το gte-multilingual-base (μέσο Top1 ≈ 0.793, μέσο MeanTop5 ≈ 0.719). Αντίθετα, το bge-m3 εμφανίζει χαμηλότερες τιμές (μέσο Top1 ≈ 0.606, μέσο MeanTop5 ≈ 0.532), γεγονός που υποδεικνύει ασθενέστερη σημασιολογική αντιστοίχιση στο συγκεκριμένο σύνολο ελληνικών διοικητικών εγγράφων.

Επιτυχία παραγωγής απάντησης

Ως προς τη στήλη «Βρέθηκε Απάντηση», το gte-multilingual-base και το multilingual-e5-base πέτυχαν απάντηση σε 9/10 ερωτήσεις (90%), ενώ το bge-m3 σε 8/10 (80%). Παρατηρούνται μεμονωμένες περιπτώσεις αποτυχίας (π.χ. Ερώτηση 5 για bge-m3 και multilingual-e5-base, καθώς και Ερώτηση 10 για gte-multilingual-base), οι οποίες υποδεικνύουν ότι η τελική επιτυχία δεν εξαρτάται μόνο από το retrieval score, αλλά και από το αν τα ανακτημένα αποσπάσματα περιέχουν ρητή και επαρκή πληροφορία ώστε να επιτραπεί απάντηση με βάση τους αυστηρούς κανόνες του prompt που έχει δοθεί στο γλωσσικό μοντέλο.

Χρόνοι εκτέλεσης και υπολογιστικό κόστος

Σε επίπεδο ταχύτητας ανάκτησης, το gte-multilingual-base εμφανίζει τον χαμηλότερο μέσο χρόνο (Avg Retrieval ≈ 394 ms), ενώ το multilingual-e5-base εμφανίζει υψηλότερο μέσο όρο (Avg Retrieval ≈ 566 ms) λόγω ορισμένων ερωτήσεων με αυξημένο χρόνο ανάκτησης. Το bge-m3 παρουσιάζει σημαντικά αυξημένο μέσο χρόνο ανάκτησης (Avg Retrieval ≈ 1335 ms) και επιπλέον υψηλότερο μέσο CPU Ret.

ms (≈ 6831 ms) συγκριτικά με τα άλλα δύο μοντέλα (≈ 3454 ms για gte και ≈ 2651 ms για e5). Αυτό υποδεικνύει μεγαλύτερο υπολογιστικό κόστος του retrieval pipeline όταν χρησιμοποιείται το bge-m3 στο συγκεκριμένο περιβάλλον.

Τέλος, σημειώνεται ότι η μετρική Total ms επηρεάζεται έντονα από τη μεταβλητότητα της κλήσης στο γλωσσικό μοντέλο (network/υπηρεσία), επομένως είναι χρήσιμη ως μέτρηση «εμπειρίας χρήστη» (end-to-end latency), αλλά δεν απομονώνει από μόνη της την απόδοση του embedding μοντέλου.

4.5 Επίλογος

Η πειραματική αξιολόγηση που διεξήχθη στο παρόν κεφάλαιο παρέχει πολύτιμες ενδείξεις για την απόδοση των τριών επιλεγμένων embedding μοντέλων στο πλαίσιο ενός εικονικού συνομιλητή, που παρείχε ανάκτηση πληροφοριών από δημόσια έγγραφα, που έχουν προεπιλεχθεί από το χρήστη. Μέσω ενός συνόλου 10 ερωτήσεων και της εφαρμογής μετρικών αξιολόγησης, κατέστη δυνατή η ποσοτική σύγκριση της αποτελεσματικότητάς τους.

Συνολικά, με βάση τις μετρικές του πίνακα 3, τα μοντέλα **gte-multilingual-base** και **multilingual-e5-base** εμφανίζουν πιο σταθερή και ποιοτική ανάκτηση (υψηλότερα Top1/MeanTop5 και υψηλότερο ποσοστό επιτυχούς απάντησης), ενώ το **bge-m3** παρουσιάζει αυξημένο υπολογιστικό κόστος και χαμηλότερα retrieval scores στο συγκεκριμένο dataset ελληνικών διοικητικών εγγράφων.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει μια ανακεφαλαίωση όσων αναλύθηκαν στην παρούσα πτυχιακή εργασία, θα αναφερθούν τα βασικά συμπεράσματα που εξήχθησαν κατά την υλοποίηση και σχεδίαση του RAG εικονικού συνομιλητή και θα γίνουν κάποιες προτάσεις για μελλοντικές βελτιώσεις.

Σε αυτή τη πτυχιακή εργασία, διερευνήθηκαν διάφορες πτυχές του τρόπου ανάκτησης (retrieval) πληροφορίας σε ελληνικά διοικητικά έγγραφα, καθώς και ο αντίκτυπος της επιλογής διαφορετικών embedding μοντέλων ως προς την ποιότητα του παρεχόμενου context και, κατ' επέκταση, της παραγωγής απάντησης από το LLM.

Παρόλο που αντιμετωπίστηκαν κάποιες προκλήσεις κατά τη δημιουργία του συστήματος και δεδομένου της μειωμένης υπολογιστής ισχύος που υπήρχε διαθέσιμη, επιτεύχθηκε η δημιουργία ενός chatbot που προσφέρει αξιόπιστες απαντήσεις σε μία πληθώρα εγγράφων.

5.2 Συμπεράσματα

Η υλοποίηση ενός RAG συστήματος για ερωτήσεις και απαντήσεις πάνω σε ελληνικά έγγραφα δημόσιου τομέα ανέδειξε ότι η ποιότητα της τελικής απάντησης εξαρτάται πρωτίστως από την ποιότητα της ανάκτησης και από το κατά πόσο τα ανακτημένα αποσπάσματα περιέχουν επαρκή και σαφή πληροφορία. Στο πλαίσιο των πειραμάτων, η στρατηγική ανάκτησης k_files = 1 επέτρεψε την εστίαση του συστήματος στην επιλογή του καταλληλότερου εγγράφου και στη συνέχεια στην τροφοδότηση του LLM με τα πιο σχετικά αποσπάσματα από αυτό. Η προσέγγιση αυτή αποδείχθηκε λειτουργική, καθώς μείωσε την πολυπλοκότητα της απάντησης και περιόρισε την πιθανότητα ανάμιξης άσχετων πηγών.

Από τον Πίνακα 3 και τα συγκεντρωτικά αποτελέσματα προκύπτει ότι τα πολυγλωσσικά μοντέλα Alibaba-NLP/gte-multilingual-base και intfloat/multilingual-e5-base εμφάνισαν συνολικά καλή και αξιόπιστη προσαρμογή στο συγκεκριμένο dataset, παρουσιάζοντας υψηλότερες τιμές ομοιότητας (Top1 και MeanTop5) και υψηλότερο ποσοστό επιτυχούς παραγωγής απάντησης στις περισσότερες ερωτήσεις. Αντίθετα, το BAAI/bge-m3 εμφάνισε χαμηλότερα retrieval scores και αυξημένο υπολογιστικό κόστος (CPU χρόνος retrieval) σε αρκετές περιπτώσεις, γεγονός που επηρέασε αρνητικά την αποδοτικότητα του συστήματος στο συγκεκριμένο περιβάλλον δοκιμών.

Παράλληλα, παρατηρήθηκαν περιπτώσεις όπου, παρά τις υψηλές τιμές ομοιότητας, δεν παρήχθη απάντηση. Αυτό υποδεικνύει ότι οι μετρικές ομοιότητας αποτελούν χρήσιμους δείκτες για την ποιότητα της ανάκτησης, αλλά δεν εγγυώνται από μόνες τους την επιτυχία. Η τελική συμπεριφορά επηρεάζεται από τη δομή του εγγράφου, την ύπαρξη επαναλαμβανόμενων τμημάτων (π.χ. κεφαλίδες/υποσέλιδα), την καταλληλότητα της τμηματοποίησης, καθώς και από τους αυστηρούς κανόνες του prompt που έχουν τεθεί για τον περιορισμό «αυθαίρετων» απαντήσεων.

5.3 Μελλοντική Εργασία

Η πτυχιακή εργασία αυτή θα μπορούσε να επεκταθεί με διάφορους τρόπους στο μέλλον. Η έρευνα μπορεί να εμβαθύνει στην εξαγωγή κειμένου, μέσω αφαίρεσης των επαναλαμβανόμενων

κεφαλίδων/υποσέλιδων και της διατήρηση δομικών στοιχείων (π.χ. αλλαγές γραμμής, τίτλοι) με στόχο τη μείωση των φαινομένων επανάληψης στα Top-k αποτελέσματα.

Στον ίδιο άξονα, κατά την παραγωγή των embeddings, θα μπορούσε να επιτευχθεί η τμηματοποίηση του κειμένου με βάση ενότητες/παραγράφους ή πεδία φόρμας (αντί για καθαρά χαρακτήρες) ώστε να αυξηθεί η πιθανότητα του LLM να βρει την πληροφορία συγκεντρωμένη σε μικρότερο context.

Ταυτόχρονα, η χρήση ενός cross-encoder reranker (π.χ. bge-reranker) μετά την αρχική ανάκτηση μπορεί να βελτιώσει αισθητά την ποιότητα των top-k chunks, ειδικά σε ερωτήσεις όπου τα embeddings επιστρέφουν κοντινά αλλά όχι απολύτως σχετικά αποσπάσματα.

Τέλος η επέκταση σε k_files>1 θα μπορούσε να αυξήσει την κάλυψη (coverage), με αντίστοιχη ανάγκη καλύτερου ελέγχου θορύβου και επιλογής αποσπασμάτων.

5.4 Επίλογος

Η παρούσα εργασία ανέδειξε ότι ένα RAG σύστημα μπορεί να υποστηρίξει αποτελεσματικά την αναζήτηση και εξαγωγή πληροφορίας από ελληνικά διοικητικά έγγραφα, αρκεί να δοθεί ιδιαίτερη έμφαση στην ποιότητα της ανάκτησης. Τα πειράματα έδειξαν ότι η επιλογή του embedding μοντέλου επηρεάζει ουσιαστικά τόσο την ποιότητα των ανακτημένων αποσπασμάτων όσο και τη συνολική συμπεριφορά του συστήματος, με συγκεκριμένα πολυγλωσσικά μοντέλα να εμφανίζουν καλύτερη απόδοση στο συγκεκριμένο dataset. Παράλληλα, οι προτεινόμενες βελτιώσεις καταδεικνύουν σαφείς κατευθύνσεις για περαιτέρω αύξηση της αξιοπιστίας και της επεκτασιμότητας του συστήματος, ώστε να μπορεί να αξιοποιηθεί σε ευρύτερα σενάρια χρήσης και σε μεγαλύτερους όγκους εγγράφων.

ΒΙΒΛΙΟΓΡΑΦΙΑ.

[1] M. D. K. Limbu, A. M. Connor, and S. G. Macdonell, “A framework for contextual information retrieval from the WWW,” pp. 185–189.

[2] IBM, “What are large language models (LLMs)?” Accessed: Jan. 11, 2026. [Online]. Available: https://www.ibm.com/think/topics/large-language-models

[3] Y. Gao *et al.*, “Retrieval-Augmented Generation for Large Language Models: A Survey,” Mar. 2024. [Online]. Available: http://arxiv.org/abs/2312.10997

[4] Wikipedia, “Chatbot - Wikipedia.” Accessed: Jan. 10, 2026. [Online]. Available: https://en.wikipedia.org/wiki/Chatbot

[5] Jayne Schultheis, “The Evolution of Search Engines: From Archie to Answer Engines.” Accessed: Jan. 11, 2026. [Online]. Available: https://www.rellify.com/blog/evolution-of-search-engines

[6] Χρυσάνθη Σ. Γιαννούλη, “Μελέτη Μεθόδων Υπολογισμού Σημασιολογικής Ομοιότητας Κειμένων,” Εθνικό Μετσόβιο Πολυτεχνείο, 2021.

[7] “Embedding Models in NLP: A Comprehensive Guide to Word Embeddings and Contextualized Embeddings | Medium.” Accessed: Jan. 13, 2026. [Online]. Available: https://medium.com/@nay1228/embedding-models-a-comprehensive-guide-for-beginners-to-experts-0cfc11d449f1

[8] Y. Liu *et al.*, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” Jul. 2019, [Online]. Available: http://arxiv.org/abs/1907.11692

[9] L. Wang *et al.*, “Text Embeddings by Weakly-Supervised Contrastive Pre-training,” Feb. 2024, [Online]. Available: http://arxiv.org/abs/2212.03533

[10] “gte-multilingual-base | AI Model Details.” Accessed: Jan. 14, 2026. [Online]. Available: https://www.aimodels.fyi/models/huggingFace/gte-multilingual-base-alibaba-nlp

[11] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, “M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation,” Dec. 2025, [Online]. Available: http://arxiv.org/abs/2402.03216

[12] “Welcome to Faiss Documentation — Faiss documentation.” Accessed: Jan. 18, 2026. [Online]. Available: https://faiss.ai/index.html

[13] Big Blue Data Academy, “Τι Είναι τα Large Language Models και Πώς Λειτουργούν;” Accessed: Jan. 12, 2026. [Online]. Available: https://bigblue.academy/gr/large-language-models

[14] Ivan Palomares Carrascosa, “The Journey of a Token: What Really Happens Inside a Transformer”, Accessed: Nov. 17, 2025. [Online]. Available: https://machinelearningmastery.com/the-journey-of-a-token-what-really-happens-inside-a-transformer/

[15] Σάββας Παπαδόπουλος, “Δημιουργία πολυθεματικού περιεχομένου για Μεγάλα Γλωσσικά Μοντέλα με χρήση γράφων ομοιότητας και τεχνικών ομαδοποίησης θεμάτων,” Τμήμα

Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Τομέας Ηλεκτρονικής και Υπολογιστών, 2024.

[16] Oracle, “Google Gemini 2.5 Flash.” Accessed: Jan. 12, 2026. [Online]. Available: https://docs.oracle.com/en-us/iaas/Content/generative-ai/google-gemini-2-5-flash.htm

[17] G. Comanici *et al.*, “Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities,” Dec. 2025. [Online]. Available: http://arxiv.org/abs/2507.06261

[18] “Meltemi: Ένα μεγάλο γλωσσικό μοντέλο ανοιχτού λογισμικού για τα Ελληνικά από το ΙΕΛ Αθηνά - DigiGov.” Accessed: Jan. 14, 2026. [Online]. Available: https://digi.gov.gr/meltemi-ena-megalo-glossiko-montelo-gia-ta-ellinika-apo-to-iel-athina/

[19] “Advanced RAG: Architecture, Techniques, Applications and Use Cases and Development.” Accessed: Jan. 18, 2026. [Online]. Available: https://www.leewayhertz.com/advanced-rag/

[20] “REST API basics and implementation | Google Cloud.” Accessed: Jan. 18, 2026. [Online]. Available: https://cloud.google.com/discover/what-is-rest-api

ΠΑΡΑΡΤΗΜΑ Α : REST Endpoints

```python
class QueryRequest(BaseModel):
    question: str
    embedding_model: str | None = None
    k_per_file: int = 5                 # chunks to give the LLM

@app.get("/")
def health_check():
    """Health check endpoint."""
    return {"status": "ok", "message": "up and running"}

@app.post("/chat")
def chat_endpoint(query: QueryRequest, user: Dict[str, Any] = Depends(require_token)):
    """Ask a question."""
    model = query.embedding_model

    result = chatbot_instance.answer_query(
        query=query.question,
        k_per_file=query.k_per_file,
        embedding_model=model,
    )

    return result
```

```python
@app.post("/upload")
async def upload_document(file: UploadFile = File(...), user: Dict[str, Any] = Depends(require_token)):

    """Endpoint for uploading a document."""
    # Save the uploaded file to a temporary location
    with tempfile.NamedTemporaryFile(delete=False) as tmp:
        tmp.write(await file.read())
        tmp_path = tmp.name

    try:
        # Extract text
        if file.content_type == "application/pdf":
            text = DocumentLoader.load_pdf(tmp_path)
        elif file.content_type in [
            "application/vnd.openxmlformats-officedocument.wordprocessingml.document",
            "application/msword",
        ]:
            text = DocumentLoader.load_docx(tmp_path)
        elif file.content_type and file.content_type.startswith("text/"):
            text = DocumentLoader.load_txt(tmp_path)
        else:
            raise HTTPException(status_code=400, detail=f"Unsupported file type: {file.content_type}")


        print("\n[upload]", file.filename, file.content_type)
        print("[upload] extracted_chars:", len(text))
        print("[upload] preview:", repr((text or "")[:250]))
```

```python
    # Chunk + ingest
    chunks = chunk_text(text)

    print("[upload] chunks:", len(chunks), "first_chunk_chars:", len(chunks[0]) if chunks else 0)

    if not chunks:
        raise HTTPException(status_code=400, detail="No extractable text found in file")

    chatbot_instance.ingest_documents(chunks, filename=file.filename)


    _debug_store_stats()

    return {"message": "Document ingested successfully", "file": file.filename, "chunks": len(chunks)}

finally:
    # Cleanup temp file
    try:
        os.unlink(tmp_path)
    except Exception:
        pass
```

ΠΑΡΑΡΤΗΜΑ Β : Embeddings

```python
class MultiEmbeddingManager:
    """
    Holds multiple EmbeddingStore instances (one per model).
    Lets you ingest to all, and query any single one by name.
    """
    def __init__(self, model_names: List[str] | None = None, persist_root: str = "../data"):
        self.model_names = model_names or DEFAULT_EMBEDDING_MODELS
        self.stores: Dict[str, EmbeddingStore] = {
            name: EmbeddingStore(name, persist_root=persist_root) for name in self.model_names
        }

    def list_models(self) -> List[str]:
        return list(self.stores.keys())

    def add_documents_all(self, texts: List[str], filename: str = "unknown"):
        # Ingest to every model
        for store in self.stores.values():
            store.add_documents(texts, filename=filename)

    def search_one(
            self,
            model_name: str,
            query: str,
            k_files: int = 1,
            k_per_file: int = 5,
    ) -> Tuple[List[Tuple[str, Dict[str, Any], float]], List[str]]:
        store = self.stores.get(model_name)
        if not store:
            raise ValueError(f"Unknown embedding model: {model_name}")
        return store.search_grouped(query=query, k_files=k_files, k_per_file=k_per_file)
```

```python
class EmbeddingBackend:
    """
    Wraps a SentenceTransformer and handles model-specific quirks (e.g., E5 prompts).
    """
    def __init__(self, model_name: str):
        self.model_name = model_name
        if model_name == "Alibaba-NLP/gte-multilingual-base":
            self.model = SentenceTransformer(model_name, trust_remote_code=True)
        else:
            self.model = SentenceTransformer(model_name)

        lname = model_name.lower()
        self.is_e5 = ("e5" in lname) or ("intfloat/multilingual-e5" in lname)
        # (BGE/GTE need no special prefixes)

    def _format_inputs(self, texts: List[str], is_query: bool) -> List[str]:
        if self.is_e5:
            if is_query:
                return [f"query: {t}" for t in texts]
            else:
                return [f"passage: {t}" for t in texts]
        return texts

    def encode(self, texts: List[str], is_query: bool = False):
        if not texts:
            return []
        formatted = self._format_inputs(texts, is_query=is_query)
        return self.model.encode(formatted, convert_to_numpy=True)
```

```python
class EmbeddingStore:
    """
    Persistent store for ONE embedding model.
    - Normalizes vectors → FAISS IndexFlatIP (cosine similarity).
    - Saves/loads texts, embeddings, metadata, and the FAISS index.
    - Grouped search to pick chunks from the single best source file.
    """

    def __init__(self, model_name: str, persist_root: str = "../data"):
        self.model_name = model_name
        self.backend = EmbeddingBackend(model_name)

        # Per-model folder
        self.persist_dir = os.path.abspath(
            os.path.join(os.path.dirname(__file__), persist_root, _safe_name(model_name))
        )
        os.makedirs(self.persist_dir, exist_ok=True)

        self.index_file = os.path.join(self.persist_dir, "index.faiss")
        self.data_file = os.path.join(self.persist_dir, "index.pkl")

        self.texts: List[str] = []
        self.metadata: List[Dict[str, Any]] = []   # [{file: "..."}]
        self.embeddings: np.ndarray | None = None
        self.index: faiss.Index | None = None

        self.load()
```

```python
class EmbeddingStore:
    def _norm(x: np.ndarray) -> np.ndarray:
        return x / (np.linalg.norm(x, axis=1, keepdims=True) + 1e-12)

    def _rebuild_index(self):
        if self.embeddings is None or self.embeddings.size == 0:
            self.index = None
            return
        embs = self._norm(self.embeddings.astype(np.float32))
        idx = faiss.IndexFlatIP(embs.shape[1])
        idx.add(embs)
        self.index = idx

    # ---------- public API ----------
    def add_documents(self, texts: List[str], filename: str = "unknown") -> None:
        if not texts:
            return
        new_embs = self.backend.encode(texts, is_query=False).astype(np.float32)

        if self.embeddings is None:
            self.embeddings = new_embs
            self.texts = list(texts)
            self.metadata = [{"file": filename} for _ in texts]
        else:
            self.embeddings = np.vstack((self.embeddings, new_embs))
            self.texts.extend(texts)
            self.metadata.extend([{"file": filename} for _ in texts])

        self._rebuild_index()
        self.save()
```

ΠΑΡΑΡΤΗΜΑ C : Συνδυασμός embeddings και Rest APIs μέσω των LLMs

```
class Chatbot:
    def __init__(self, model_names: List[str] | None = None):
        self.manager = MultiEmbeddingManager(model_names=model_names, persist_root="../data")

    def ingest_documents(self, texts: List[str], filename: str = "unknown"):
        self.manager.add_documents_all(texts, filename=filename)

    def build_prompt(self, context: str, query: str) -> str:
        return (
            "Ρόλος: Είσαι βοηθός RAG.\n"
            "ΟΔΗΓΙΕΣ:\n"
            "• Απάντησε ΜΟΝΟ από τα αποσπάσματα που σου δίνω.\n"
            "•Προσπάθησε αρκετά να βρεις την απάντηση."
            "•Αν δεν υπάρχει επαρκής πληροφορία, απάντησε ακριβώς: «Δεν βρέθηκε απάντηση.»\n"
            "•Γράψε στα Ελληνικά, σύντομα και ακριβώς.\n\n"
            f"Αποσπάσματα (όλα από ΕΝΑ αρχείο):\n{context}\n\n"
            f"Ερώτηση: {query}\n"
            "Απάντηση:"
        )

    def answer_query(
            self,
            query: str,
            k_per_file: int = 5,
            embedding_model: str | None = None,  # let UI choose; default = first model
    ) -> Dict[str, Any]:
        model_name = embedding_model or self.manager.list_models()[0]

        # ---- Start timers ----
        wall_start = time.perf_counter()
        cpu_start = cpu_time_ms()
```

```python
class Chatbot:
    def answer_query(

        # ---- Retrieval ----
        t0 = time.perf_counter()

        hits, chosen_files = self.manager.search_one(
            model_name=model_name, query=query, k_files=1, k_per_file=k_per_file
        )

        retrieval_wall_ms = (time.perf_counter() - t0) * 1000.0
        cpu_after_retrieval = cpu_time_ms()

        if not hits:
            total_wall_ms = (time.perf_counter() - wall_start) * 1000.0
            cpu_total_ms = cpu_time_ms() - cpu_start
            print(
                f"[metrics] model={model_name} "
                f"retrieval_ms={retrieval_wall_ms:.1f} total_ms={total_wall_ms:.1f} "
                f"cpu_total_ms={cpu_total_ms:.1f} hits=0"
            )
            return {"answer": "Δεν βρέθηκε απάντηση.", "sources": [], "embedding_model": model_name}

        # ---- Retrieval quality (chunk-level) ----
        scores = [s for (_, _, s) in hits]
        top1 = scores[0]
        mean_topk = sum(scores) / len(scores)

        # ---- Build prompt ----
        context = "\n\n".join(t for (t, _, _) in hits)
        prompt = self.build_prompt(context, query)
```

```python
class Chatbot:                                                    ⚠2 ⚠1
    def answer_query(

        # ---- LLM ----
        t1 = time.perf_counter()
        ans = self._call_gemini(prompt)
        llm_wall_ms = (time.perf_counter() - t1) * 1000.0
        cpu_after_llm = cpu_time_ms()

        # ---- Final metrics ----
        total_wall_ms = (time.perf_counter() - wall_start) * 1000.0
        cpu_retrieval_ms = cpu_after_retrieval - cpu_start
        cpu_llm_ms = cpu_after_llm - cpu_after_retrieval
        cpu_total_ms = cpu_after_llm - cpu_start

        print(
            f"[metrics] model={model_name} files={chosen_files} hits={len(hits)} "
            f"retrieval_ms={retrieval_wall_ms:.1f} llm_ms={llm_wall_ms:.1f} total_ms={total_wall_ms:.1f} "
            f"cpu_retrieval_ms={cpu_retrieval_ms:.1f} cpu_llm_ms={cpu_llm_ms:.1f} cpu_total_ms={cpu_total_ms:.1f} "
            f"top1={top1:.4f} mean_topk={mean_topk:.4f} "
        )

        if ans and ans != "Δεν βρέθηκε απάντηση.":
            return {"answer": ans, "sources": chosen_files, "embedding_model": model_name}

        return  {"answer": "Δεν βρέθηκε απάντηση.", "sources": [], "embedding_model": model_name}
```

```python
# ---------- LLM calls ----------
def _call_gemini(self, prompt: str) -> str | None:
    model = genai.GenerativeModel(GEMINI_MODEL_NAME)
    for attempt in range(MAX_GEMINI_ATTEMPTS):
        try:
            resp = model.generate_content(prompt)
            txt = getattr(resp, "text", "") if resp else ""
            return txt.strip() or None
        except (ResourceExhausted, ServiceUnavailable) as e:
            print(f"Gemini API rate limit or service unavailable: {e}")
            if attempt < MAX_GEMINI_ATTEMPTS - 1:
                time.sleep(RETRY_WAIT_SECONDS)
            else:
                return None
        except DeadlineExceeded as e:
            print(f"Gemini API deadline exceeded: {e}")
            return None
        except Exception as e:
            print(e)
            return None
    return None
```

ΠΑΡΑΡΤΗΜΑ D : Πιστοποίηση Χρηστών

```python
async def get_jwks() -> Dict:
    now = time.time()
    if _jwks_cache["keys"] and now - _jwks_cache["fetched_at"] < JWKS_CACHE_TTL:
        return _jwks_cache

    async with httpx.AsyncClient() as client:
        resp = await client.get(JWKS_URL, timeout=10)
        resp.raise_for_status()
        data = resp.json()
        _jwks_cache["keys"] = data["keys"]
        _jwks_cache["fetched_at"] = now
        return _jwks_cache

# ---------------- JWT validation ---------------- #
async def introspect_token(token: str) -> Dict[str, any]:
    jwks = await get_jwks()
    unverified_header = jwt.get_unverified_header(token)
    kid = unverified_header.get("kid")
    if not kid:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Missing 'kid' in token header",
            headers={"WWW-Authenticate": "Bearer"},
        )
```

```python
async def introspect_token(token: str) -> Dict[str, any]:

    try:
        # python-jose can use the JWKS dict directly
        decoded = jwt.decode(
            token,
            key_data,
            algorithms=[key_data["alg"]],
            audience=AUDIENCE,
            issuer=ISSUER,
        )
        return decoded
    except JWTError as e:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail=f"Invalid token: {str(e)}",
            headers={"WWW-Authenticate": "Bearer"},
        )

# ---------------- FastAPI dependency ---------------- #
async def require_token(
    credentials: Optional[HTTPAuthorizationCredentials] = Depends(security),
) -> Dict[str, any]:
    if credentials is None or credentials.scheme.lower() != "bearer":
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Not authenticated",
            headers={"WWW-Authenticate": "Bearer"},
        )
    return await introspect_token(credentials.credentials)
```