ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

# «Ανάπτυξη και Αξιολόγηση Ελληνόγλωσσου Ψηφιακού Βοηθού (Chatbot) με Αρχιτεκτονική RAG για την Υποστήριξη Χρηστών Εκπαιδευτικής Πλατφόρμας»


**Των φοιτητών**
**Ζορτός Αχιλλέας,**
**Μαρασλίδη Μαρία**
**Αρ. Μητρώου: 2020228, 2019098**

**Επιβλέπων**
**Δρ. Διαμαντάρας Κωνσταντίνος**
**Καθηγητής**

**Ημερομηνία 31-05-2026**

Τίτλος Δ.Ε. Ανάπτυξη και Αξιολόγηση Ελληνόγλωσσου Ψηφιακού Βοηθού (Chatbot) με Αρχιτεκτονική RAG για την Υποστήριξη Χρηστών Εκπαιδευτικής Πλατφόρμας
Κωδικός Δ.Ε. 25288
Ονοματεπώνυμο φοιτητών Ζορτός Αχιλλέας, Μαρασλίδη Μαρία
Ονοματεπώνυμο εισηγητή Διαμαντάρας Κωνσταντίνος
Ημερομηνία ανάληψης Δ.Ε. 15-07-2025
Ημερομηνία περάτωσης Δ.Ε. 31-05-2026

*Βεβαιώνουμε ότι είμαστε οι συγγραφείς αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχουμε καταγράψει τις όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνουμε ότι αυτή η εργασία προετοιμάστηκε από εμάς προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.*

*«Στους αγαπημένους μας»*

# Πρόλογος

Ξεκινώντας τις σπουδές μας στο τμήμα δεν γνωρίζαμε το αντικείμενο με το οποίο θα ασχοληθούμε. Η αγάπη μας για τον τομέα επεξεργασίας ήχου και εικόνας μας έφερε σε επαφή με την επιστημονική εταιρεία «European School Radio». Εκεί γνωριστήκαμε, συνεργαστήκαμε και μάθαμε για την διαδικτυακή πλατφόρμα του πρώτου μαθητικού ραδιοφώνου και όλες τις προηγμένες δυνατότητες που αυτή προσφέρει. Για τρία χρόνια υπήρξαμε μέλη της εκπαιδευτικής ομάδας του προγράμματος «Κάνω Ραδιόφωνο στο European School Radio - Παραγωγή Δημοσιογραφικής Εκπομπής» (εγκεκριμένο από το Υ.ΠΑΙ.Θ.Α.) και έτσι ήρθαμε σε επαφή με ανθρώπους που ασχολούνται με την Τεχνητή Νοημοσύνη και αναπτύσσουν εργαλεία για αυτή τη πλατφόρμα, γεγονός που μας ενέπνευσε να ασχοληθούμε και να ενισχύσουμε αυτή την προσπάθεια. Στόχος μας ήταν να μπορέσουμε να βοηθήσουμε με όλα τα τεχνικά ζητήματα που μπορεί να αντιμετωπίζει ένα μέλος της ραδιοφωνικής μας κοινότητας. Παρόλο που δυσκολευτήκαμε να βρούμε θέμα για την εργασία μας, η άριστη σχέση μας με το European School Radio μας έφερε σε επαφή με τον υπεύθυνο καθηγητή μας. Όλα όσα συναντήσαμε κατά τη διάρκεια αυτών των μηνών, μας πρόσφεραν διδάγματα σε ακαδημαϊκό, αλλά και προσωπικό επίπεδο και μας βοήθησαν να γίνουμε καλύτεροι και να συνεισφέρουμε στην προσπάθεια να αναπτύξουμε την μαθητική ραδιοφωνική κοινότητα.

# Περίληψη

Στην παρούσα διπλωματική εργασία πραγματοποιείται η ανάπτυξη ενός λειτουργικού ελληνόγλωσσου Ψηφιακού Βοηθού (Chatbot) για την εκπαιδευτική πλατφόρμα του European School Radio. Στόχος του είναι να μπορεί να υποστηρίξει τους χρήστες άμεσα και αξιόπιστα σε καθημερινές απορίες που αφορούν την πλατφόρμα. Το σύστημα αναπτύχθηκε με βάση την αρχιτεκτονική Επαυξημένης Παραγωγής μέσω Ανάκτησης (Retrieval Augmented Generation - RAG) και η Βάση Δεδομένων η οποία το τροφοδοτούσε στηρίχθηκε αποκλειστικά στον επίσημο Οδηγό Χρήσης (User Guide) της ιστοσελίδας. Η διαδικασία της έρευνας περιλάμβανε την ανάλυση και σύγκριση διαφορετικών πειραματικών συνδυασμών από τα βασικά συστατικά που καθορίζουν την απόδοση ενός συστήματος RAG, με στόχο την εύρεση του καλύτερου δυνατού αποτελέσματος. Πιο συγκεκριμένα, μετά από βιβλιογραφική έρευνα και φιλτράρισμα εξετάστηκαν 72 συνδυασμοί μεταξύ τεσσάρων Γλωσσικών Μοντέλων (Gemini 2.5 Flash, DeepSeek V4 Flash, Gemma 4:26b, Llama 3.1:8b), τριών μοντέλων διανυσματικών αναπαραστάσεων (Embeddings Models) και έξι στρατηγικών τεμαχισμού κειμένου (Chunking Methods) οι οποίες περιλαμβάνουν αναδρομικές (recursive) και σημασιολογικές μεθόδους (semantic). Τα πειράματα αξιολογήθηκαν μέσω του πλαισίου Ragas (Retrieval Augmented Generation Assessment) χρησιμοποιώντας την προσέγγιση «LLM-as-a-Judge», χρησιμοποιώντας το μοντέλο `gpt-4o-mini` της OpenAI, με βάση το ιδανικό σύνολο ερωτοαπαντήσεων (Golden Dataset) που δημιουργήθηκε. Τα αποτελέσματα ανέδειξαν ως απόλυτους κυρίαρχους το μοντέλο διανυσματικών αναπαραστάσεων `multilingual-e5-large` και την αναδρομική μέθοδο chunking με μέγιστο όριο τα 1024 tokens. Στη σύγκριση των Γλωσσικών Μοντέλων επικράτησε το Gemini 2.5 Flash, διατηρώντας την καλύτερη ισορροπία μεταξύ της αποδοτικότητας στην ανάκτηση του κειμένου (Retrieval) και της ποιότητας παραγωγής κειμένου (Generation). Ωστόσο, το DeepSeek V4 Flash είχε πολύ μικρή απόκλιση στην απόδοσή του και ταυτόχρονα αρκετά χαμηλότερο λειτουργικό κόστος σε σχέση με το Gemini 2.5 Flash. Άρα, συνυπολογίζοντας την απαίτηση ενός ρεαλιστικού σεναρίου με την ιδανική σχέση κόστους και οφέλους, η τελική βέλτιστη επιλογή ήταν ο συνδυασμός του DeepSeek V4 Flash με το `multilingual-e5-large` και τον αναδρομικό τεμαχισμό (1024 tokens).

«Development and Evaluation of a Greek-Language Chatbot with RAG Architecture for User Support in an Educational Platform»

«Achilleas Zortos, Maria Maraslidi»

# Abstract

This thesis presents the development of a functional Greek-speaking Chatbot for the European School Radio educational platform. Its goal is to support users directly and reliably with everyday queries regarding the platform. The system was developed based on the Retrieval-Augmented Generation (RAG) architecture, and the knowledge base relied exclusively on the official User Guide of the website. The research process included the analysis and comparison of different experimental combinations of the basic components that determine the performance of a RAG system, aiming to find the best possible outcome. More specifically, following literature research and filtering, 72 combinations were examined among four Large Language Models (Gemini 2.5 Flash, DeepSeek V4 Flash, Gemma 4:26b, Llama 3.1:8b), three Embedding Models, and six Chunking Methods, which include recursive and semantic methods. The experiments were evaluated through the Ragas (Retrieval Augmented Generation Assessment) framework using the "LLM-as-a-Judge" approach with OpenAI's `gpt-4o-mini` model, based on the created ideal set of questions and answers (Golden Dataset). The results highlighted the `multilingual-e5-large embedding` model and the recursive chunking method with a maximum limit of 1024 tokens as the most effective choices. In the comparison of the Large Language Models, Gemini 2.5 Flash achieved the best performance, maintaining the optimal balance between text retrieval efficiency (Retrieval) and text generation quality (Generation). However, DeepSeek V4 Flash showed a marginally lower performance while having a significantly lower operational cost compared to Gemini 2.5 Flash. Therefore, considering the requirement of a realistic scenario with the ideal cost-benefit ratio, the final optimal choice was the combination of DeepSeek V4 Flash with `multilingual-e5-large` and recursive chunking (1024 tokens).

# Ευχαριστίες

Θα θέλαμε να ευχαριστήσουμε τον Αντιπρύτανη και επιβλέποντα καθηγητή μας Δρ. Κωνσταντίνο Διαμαντάρα για την πολύτιμη καθοδήγησή του, καθώς και την κα. Χάιδω Μιζέλη, Ακαδημαϊκή Υπότροφο και Υποψήφια Διδάκτορα στο τμήμα που μας στήριξε σε κάθε βήμα αυτής της εργασίας. Επιπλέον, ευχαριστούμε από καρδιάς την κα. Ευτυχία Τούλιου, Υποψήφια Διδάκτορα Δημοσιογραφίας στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης και Πρόεδρο της Διαχειριστικής Επιτροπής του European School Radio για την υποστήριξη. Θα θέλαμε να εκφράσουμε τις ξεχωριστές μας ευχαριστίες στον κ. Γιώργο Παπαδόπουλο, Προγραμματιστή και Εταίρο του European School Radio, για τις εξαιρετικά χρήσιμες συμβουλές του. Τέλος, οφείλουμε ένα τεράστιο ευχαριστώ στις οικογένειες και τους φίλους μας για την υπομονή και τη συμπαράστασή τους σε όλη αυτή τη διαδρομή.

# Περιεχόμενα

# Κατάλογος Σχημάτων

# Κατάλογος Πινάκων

# Συντομογραφίες

| | |
|---|---|
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| IDE | Integrated Development Environment |
| IQR | Interquartile |
| LLM | Large Language Model |
| LM | Language Model |
| MoE | Mixture of Experts |
| MLA | Multi Head Latent Attention |
| MTP | Multi Token Prediction |
| RAG | Retrieval Augmented Generation |
| RAGAS | Retrieval Augmented Generation Assessment |
| SLM | Small Language Models |
| UI | User Interface |
| VS Code | Visual Studio Code |
| Δ.Ε. | Διπλωματική Εργασία |
| ΔΙΠΑΕ | Διεθνές Πανεπιστήμιο Ελλάδος |

# Κεφάλαιο 1ο: Εισαγωγή

## 1.1 Εισαγωγή

Ο όρος Βιομηχανία 4.0 περιγράφει τη τρέχουσα τάση στις τεχνολογίες που αναπτύσσονται και υιοθετούνται, με κύρια βάση τους αυτοματισμούς και την ανταλλαγή δεδομένων. Σε αυτά τα δύο, έρχεται και η χρήση «έξυπνων» εργαλείων Τεχνητής Νοημοσύνης (Artificial Intelligence - AI) που εμφανίζεται όλο και περισσότερο σε διάφορες εφαρμογές, με τον χρήστη πολλές φορές να μην μπορεί την αντιληφθεί [1]. Ζούμε σε μία εποχή όπου έχουμε μεταβεί από την περίοδο των πειραματικών δυνατοτήτων του AI σε μία νέα, όπου οι πράκτορες είναι αυτόνομοι και μπορούν να εκτελούν πολύπλοκες ροές εργασίας, να αλληλεπιδρούν με εξωτερικά συστήματα και να λαμβάνουν αποφάσεις με ελάχιστη ανθρώπινη παρέμβαση. Όλες αυτές οι εξελίξεις συνθέτουν την ανάγκη για αυτοματοποίηση εργασιών ρουτίνας, αλλά και για επαναπροσδιορισμό της διάδρασης με τους πελάτες - χρήστες [2].

Σημαντικό σημείο προς συζήτηση, αποτελεί η διάδοση των Ψηφιακών Βοηθών (Chatbot) στην καθημερινότητά μας. Με τον όρο Chatbot περιγράφεται ουσιαστικά ένα πρόγραμμα υπολογιστή σχεδιασμένο να προσομοιώνει μια διαδραστική συνομιλία ή επικοινωνία με τον χρήστη μέσω κειμένου, φωνής και οπτικών μέσων. Ο βασικός περιορισμός που λαμβάνεται είναι ότι κάθε Chatbot μπορεί να μιμηθεί την ανθρώπινη επικοινωνία. Υπάρχουν έρευνες που τεκμηριώνουν ότι τα Chatbot λειτουργούν πλέον ως απαραίτητα εργαλεία για την ενίσχυση των αλληλεπιδράσεων με τους πελάτες - χρήστες και την παροχή υποστήριξης σε πραγματικό χρόνο [3].

Παράλληλα, ενώ η έλλειψη εμπιστοσύνης αποτελεί το μεγαλύτερο εμπόδιο για την υιοθέτηση του AI, η αρχιτεκτονική Επαυξημένης Παραγωγής μέσω Ανάκτησης (Retrieval Augmented Generation - RAG) έχει γίνει ο βασικός πυλώνας για την αποφυγή των «παραισθήσεων» (hallucinations). Το RAG επιτρέπει στα μοντέλα να μην βασίζονται απλώς στην παγωμένη «μνήμη» της αρχικής τους εκπαίδευσης, αλλά να αναζητούν ακριβή στοιχεία και έγγραφα σε πραγματικό χρόνο (real time) για να τεκμηριώσουν τις απαντήσεις τους [4]. Αυτή η τεχνική μπορεί να μειώσει τις παραισθήσεις σε ποσοστό άνω του 40% σε κρίσιμες εφαρμογές, όπως η υγεία ή οι χρηματοοικονομικές υπηρεσίες κ.α.. Επιπλέον, το σύγχρονο RAG υποστηρίζει αναζήτηση πολλαπλών μέσων (multimodal), συνδυάζοντας την ανάλυση κειμένου, εικόνων, βίντεο και ήχου μέσω προηγμένων διανυσματικών αναπαραστάσεων (vector embeddings) [5].

Ενδιαφέρον παρουσιάζει η ανάλυση κειμένου διαφορετικών γλωσσών πέρα της αγγλικής. Παρά τη ραγδαία πρόοδο των Μεγάλων Γλωσσικών Μοντέλων (Large Language Models - LLMs), η πλειονότητα των εξελιγμένων μοντέλων διανυσματικών αναπαραστάσεων (embedding models) παραμένει αγγλοκεντρική. Όταν οι τεχνολογίες αυτές καλούνται να διαχειριστούν γλώσσες με πλούσια μορφολογία και διαφορετική δομή, όπως η ελληνική, συχνά υστερούν σε ακρίβεια. Έτσι, σε εφαρμογές που καλούνται να χρησιμοποιηθούν άλλες γλώσσες, χρειάζονται ακόμη αρκετές έρευνες και δοκιμές προκειμένου να αποδειχθεί η αποτελεσματικότητα και η εγκυρότητα αυτών [5].

## 1.2 Στόχος της Έρευνας

Η παρούσα διπλωματική εργασία εστιάζει σε ένα εξαιρετικά επίκαιρο και αναπτυσσόμενο θέμα, επιλύοντας μια πραγματική ανάγκη του European School Radio (το πρώτο μαθητικό ραδιόφωνο). Το κεντρικό αντικείμενο είναι η ανάπτυξη και η συγκριτική αξιολόγηση ενός ελληνόγλωσσου Chatbot υποστήριξης χρηστών, ο οποίος θα βασίζεται στον επίσημο Οδηγό Χρήσης (User Guide) της πλατφόρμας. Το European School Radio απευθύνεται σε ένα ιδιαίτερα ανομοιογενές κοινό, το οποίο αποτελείται από μαθητές όλων των ηλικιών και ενήλικες εκπαιδευτικούς ή γονείς με διαφορετικά

επίπεδα τεχνικής εξοικείωσης λόγω ηλικίας αλλά και ενασχόλησης. Αυτό καθιστά αναγκαία την ύπαρξη ενός συστήματος που θα απαντά με μεγάλη ακρίβεια και σαφήνεια οποιαδήποτε στιγμή, όταν ένας χρήστης αντιμετωπίζει κάποιο πρόβλημα.

Για την επίτευξη αυτού του στόχου, επιλέγεται η σύγχρονη αρχιτεκτονική RAG, η οποία επιτρέπει σε ένα LLM να αντλεί δυναμικά πληροφορίες από μια συγκεκριμένη, έγκυρη πηγή γνώσης, στην περίπτωσή μας το πλήρες και ανανεωμένο User Guide της ιστοσελίδας, ελαχιστοποιώντας έτσι τις πιθανότητες «παραισθήσεων» (hallucinations) και λανθασμένων απαντήσεων.

Το καινοτόμο και επιστημονικό κομμάτι της εργασίας πηγάζει από τη μελέτη και αξιολόγηση του RAG, με αποκλειστική έμφαση στις ιδιαιτερότητες της ελληνικής γλώσσας. Η διαδικασία αυτή θα εξετάσει τρεις βασικούς άξονες: τον τεμαχισμό του κειμένου (Chunking), τα μοντέλα διανυσματικών αναπαραστάσεων (Embedding Models) και τα Γλωσσικά Μοντέλα που χρησιμοποιούνται για την κατανόηση και παραγωγή φυσικής γλώσσας. Στο στάδιο του τεμαχισμού, η εργασία θα συγκρίνει παραδοσιακές μεθόδους σταθερού μεγέθους (Fixed-Size Chunking) έναντι πιο εξελιγμένων σημασιολογικών προσεγγίσεων (Semantic/Recursive Chunking). Στο επόμενο στάδιο, η έρευνα θα αξιολογήσει τη συμπεριφορά διαφορετικών Embedding Models, τα οποία αναλαμβάνουν να μετατρέψουν τα κείμενα του Οδηγού Χρήσης σε διανυσματικές αναπαραστάσεις (vectors) μέσα σε μια Διανυσματική Βάση Δεδομένων (Vector Database). Εδώ εντοπίζεται ένα σημαντικό ερευνητικό ενδιαφέρον, καθώς η πλειονότητα των διαθέσιμων μοντέλων που χειρίζονται και την ελληνική γλώσσα δεν είναι πάντα αποτελεσματικά [5]. Η εργασία θα θέσει σε δοκιμή και θα συγκρίνει κορυφαία πολυγλωσσικά μοντέλα όπως Gemma, Gemini, Deepseek και Llama, εξετάζοντας ποιο από αυτά μπορεί να αποδώσει με μεγαλύτερη ακρίβεια στα ερωτημάτων των χρηστών.

## 1.3 Δομή της Διπλωματικής Εργασίας

Στο 1ο Κεφάλαιο παρουσιάζεται το εισαγωγικό μέρος της εργασίας καθώς επίσης περιγράφεται ο στόχος της. Στο 2ο Κεφάλαιο γίνεται μια βιβλιογραφική, ιστορική και θεωρητική ανασκόπηση. Εδώ μπορούν να βρεθούν άλλες παρόμοιες εφαρμογές και οι διαφοροποίηση της δικής μας από τις υπόλοιπες. Στο 3ο Κεφάλαιο ενδεικτικά, αναφέρονται και εξηγούνται όλοι οι απαραίτητοι όροι ώστε να γίνει κατανοητή όλη η πρακτική εφαρμογή της εργασίας. Μερικοί όροι που θα συζητηθούν είναι το Chunking, το LLM, το RAG κ.α.. Έπειτα, στο 4ο Κεφάλαιο γίνεται μια αναλυτική περιγραφή της πρακτικής εφαρμογής της εργασίας. Αναλύεται ο σχεδιασμός και η ανάπτυξη της εφαρμογής, όλα τα στάδια που ακολουθήθηκαν, καθώς επίσης εξηγούνται όλες οι τεχνικές αποφάσεις που πάρθηκαν για την εκτέλεση της εφαρμογής. Τέλος, στο 5ο Κεφάλαιο παρουσιάζονται τα αποτελέσματα που προέκυψαν από τη δοκιμή της εφαρμογής σε πραγματικές ερωτήσεις χρηστών, ενώ έγινε εξαγωγή και διαφόρων συμπερασμάτων.

# Κεφάλαιο 2ο: Βιβλιογραφική Ανασκόπηση και Πεδίο Εφαρμογής

## 2.1 Ιστορική Αναδρομή των Ψηφιακών Βοηθών (Chatbot)

Η ιστορία των Chatbot είναι ένα πραγματικά συναρπαστικό ταξίδι που διαρκεί δεκαετίες. Ξεκίνησε με απλά στατιστικά μοντέλα και κανόνες μοτίβων και έφτασε σήμερα στα υπερσύγχρονα συστήματα της Τεχνητής Νοημοσύνης.

Η εξέλιξη των Chatbot μπορεί να χωριστεί σε ορισμένους σημαντικούς σταθμούς. Ένα από τα πρώτα βήματα ήταν η Αλυσίδα Markov, που αναπτύχθηκε από τον μαθηματικό Andrey Markov το 1906. Αυτό ήταν ένα στατιστικό μοντέλο που μπορούσε να προβλέψει τυχαίες ακολουθίες και ήταν ένα από τα πρώτα βήματα για να «διδαχθούν» οι μηχανές να προβλέπουν την επόμενη λέξη [6].

Ένα άλλο σημαντικό βήμα ήταν το Τεστ του Turing, που προτάθηκε από τον Alan Turing το 1950. Αυτό το τεστ ήταν για να αξιολογηθεί εάν μια μηχανή μπορεί να μιμηθεί την ανθρώπινη συμπεριφορά και συνομιλία τόσο πειστικά, ώστε ένας άνθρωπος κριτής να μην μπορεί να ξεχωρίσει αν μιλάει σε μηχανή ή άνθρωπο.

Κατά τη διάρκεια των δεκαετιών 1960 και 1970, αναπτύχθηκαν τα πρώτα Chatbot, όπως η ELIZA και το PARRY. Η ELIZA ήταν το πρώτο Chatbot και δημιουργήθηκε από τον Joseph Weizenbaum στο MIT [7]. Λειτουργούσε αναγνωρίζοντας λέξεις-κλειδιά (keywords) στα μηνύματα των χρηστών και επιστρέφοντας τυποποιημένες απαντήσεις. Το PARRY, αντίθετα, προσομοίαζε έναν ασθενή με παρανοϊκή σχιζοφρένεια και ήταν το πρώτο Chatbot που υποβλήθηκε σε αξιολόγηση μέσω του Τεστ του Turing [6], [8].

Στα χρόνια του πειραματισμού και των βελτιώσεων μετά την δεκαετία 1980, αναπτύχθηκαν Chatbot όπως το Racter και το Jabberwacky. Το Racter ήταν ένα πρόγραμμα που μπορούσε να δημιουργήσει πρωτότυπα κείμενα και ιστορίες, ενώ το Jabberwacky προσπάθησε να μάθει φυσικούς διαλόγους απευθείας από τις συνομιλίες που έκανε με τους χρήστες στο διαδίκτυο [9].

Στη συνέχεια, αναπτύχθηκαν Chatbot όπως το ALICE, το SmarterChild και το CALO. Το ALICE ήταν ένα Chatbot που χρησιμοποιούσε τη γλώσσα AIML και κέρδισε το Βραβείο Loebner. Το SmarterChild ήταν το πρώτο Chatbot που ενσωματώθηκε ευρέως σε εφαρμογές μηνυμάτων, ενώ το CALO ήταν ένα τεράστιο ερευνητικό πρόγραμμα της αμερικανικής DARPA που στόχευε σε γνωσιακούς βοηθούς [8], [9].

Την τελευταία δεκαετία, αναπτύχθηκαν Chatbot όπως το Siri, το Google Now και το ChatGPT. Το Siri ήταν ο Ψηφιακός Βοηθός της Apple, ενώ το Google Now ήταν ένα σύστημα που συνδύαζε φωνητική αναγνώριση και Μηχανική Μάθηση (Machine Learning). Το ChatGPT, που αναπτύχθηκε από την εταιρεία OpenAI, είναι ένα Chatbot που μπορεί να γράψει άρθρα, κώδικα και ποιήματα και έχει αλλάξει τα δεδομένα στην επικοινωνία ανθρώπου-μηχανής [10].

## 2.2 Η Τεχνητή Νοημοσύνη στο Ραδιόφωνο

Η Τεχνητή Νοημοσύνη έχει αρχίσει ήδη να παίζει σημαντικό ρόλο στον χώρο του ραδιοφώνου, επηρεάζοντας τόσο την παραγωγή όσο και την παρουσίαση του περιεχομένου. Πολλοί ραδιοφωνικοί σταθμοί χρησιμοποιούν την Τεχνητή Νοημοσύνη για την αυτοματοποίηση της διαδικασίας παραγωγής εκπομπών: από την επιλογή τραγουδιών και τη δημιουργία λιστών αναπαραγωγής (playlist), μέχρι τη σύνταξη και παρουσίαση κειμένων με συνθετικές φωνές [11]. Παράλληλα, κάνουν την εμφάνισή τους ψηφιακοί παρουσιαστές (AI hosts), που συχνά βασίζονται σε κλωνοποίηση φωνής από πραγματικούς

παρουσιαστές. Με τον τρόπο αυτό μπορούν να ξεπεραστούν πρακτικά εμπόδια που προκύπτουν από την ανθρώπινη φύση [12].

Η χρήση του AI στο ραδιόφωνο δεν περιορίζεται μόνο σε προγράμματα μουσικής, αλλά επεκτείνεται και σε δελτία ειδήσεων, προωθητικά σποτ, ενημερωτικές ενότητες και διαχείριση περιεχομένου. Πλατφόρμες όπως το RadioGPT, το Voicetrack.ai, το Radio.Cloud, αλλά και εργαλεία όπως τα ChatGPT-4 και TopicPulse, έχουν συμβάλει στην ανάπτυξη «έξυπνων» ραδιοφωνικών προγραμμάτων που λειτουργούν με ελάχιστη ανθρώπινη παρέμβαση. Για παράδειγμα, η Aimee είναι ένας AI broadcaster που δημιουργήθηκε στην Ινδονησία και παρουσιάζει δική της ειδησιακή εκπομπή με έναν πιο νεανικό και μοντέρνο τρόπο για να προσελκύσει τους νέους [11], [13].

### 2.2.1 Πλεονεκτήματα της Τεχνητής Νοημοσύνης

Το AI είναι ένα έξυπνο εργαλείο και όπως όλα τα εργαλεία, με την σωστή και ευφάνταστη χρήση τους, μπορούν να αποτελέσουν κρίσιμους παράγοντες για την επιτυχία των προσπαθειών μας και την επίτευξη των στόχων μας. Η αναμφισβήτητη βοήθεια που προσφέρει αυτή η τεχνολογία στην ραδιοφωνική κοινότητα είναι η εξοικονόμηση χρόνου. Το AI εκμηδενίζει τους χρόνους παραγωγής διαφημιστικών σποτ [12]. Διαδικασίες που στο παρελθόν απαιτούσαν ημέρες (συγγραφή κειμένου, επιλογή εκφωνητή, μίξη ήχου), πλέον υλοποιούνται σχεδόν ακαριαία, επιτρέποντας τη δημιουργία προσχεδίων (prototyping) ακόμη και κατά τη διάρκεια μιας συνάντησης με τον πελάτη. Αυτή η «self-service» προσέγγιση καθιστά τη ραδιοφωνική διαφήμιση εξαιρετικά προσβάσιμη και οικονομικά αποδοτική, ιδιαίτερα για τους μικρότερους διαφημιζόμενους πελάτες [13], [14].

Προφανώς μια εύκολη και χρήσιμη βοήθεια είναι ο έλεγχος από γλωσσικά μοντέλα που βοηθούν τους δημοσιογράφους και τους παρουσιαστές να εμπλουτίσουν τον λόγο τους, να διορθώσουν σημασιολογικά ή γραμματικά λάθη και να επεξεργάζονται γρηγορότερα τις πηγές που μαζεύουν [12].

Σημαντική δυνατότητα που προσφέρουν AI εργαλεία στο ραδιόφωνο, είναι η κάλυψη ολόκληρης της ημέρας. Οι βραδινές εκπομπές δημιουργούν πρακτικές δυσκολίες καθώς τα ωράρια είναι δύσκολα, κουραστικά και η απήχηση αυτών των ωρών είναι αντικειμενικά πολύ μικρότερη από όλη την υπόλοιπη μέρα. Αυτό κατά βάση σημαίνει ότι είναι μια κοστοβόρα κατάσταση για τα ραδιόφωνα, την οποία οι αυτοματισμοί του AI μπορούν να λύσουν με μεγάλη ευκολία [13].

Φυσικά τα εργαλεία αυτά μπορούν να βοηθήσουν και από τη πλευρά του ακροατή. Η πιο απλή χρήση είναι η εξατομίκευση του περιεχομένου με κριτήρια βασισμένα στις προτιμήσεις του ίδιου του ακροατή. Αντί για το παραδοσιακό ραδιόφωνο, περνάμε σε ένα μοντέλο όπου το περιεχόμενο προσαρμόζεται στον καθένα ξεχωριστά. Τελικά, όλα τα παραπάνω, αποτελούν ξεκάθαρη βελτίωση της εμπειρίας του χρήστη [11]. Η αναβάθμιση σε αυτή την περίπτωση μεταφράζεται σε: λιγότερο χρόνο αναζήτησης, περιεχόμενο που ταιριάζει με τη διάθεση ή τα ενδιαφέροντά του ακροατή και εξάλειψη περιεχομένου, το οποίο του είναι αδιάφορο, γρήγορη και έγκυρη ενημέρωση κ.α..

### 2.2.2 Μειονεκτήματα και Προκλήσεις της Τεχνητής Νοημοσύνης

Παρόλα αυτά, τέτοια εργαλεία σε λάθος χέρια μπορούν, όχι μόνο να μην φανούν αποτελεσματικά, αλλά να δημιουργήσουν και πολλά προβλήματα. Αρχικά υπάρχουν ορισμένα εμπόδια, όπως το γλωσσικό φράγμα καθώς το AI είναι κυρίως ανεπτυγμένο στα αγγλικά. Αυτό συμβαίνει, διότι τα Μεγάλα Γλωσσικά Μοντέλα εκπαιδεύονται κυρίως με τεράστιους όγκους δεδομένων στην αγγλική γλώσσα [12]. Σε λιγότερο διαδεδομένες γλώσσες (όπως τα ελληνικά), η ποιότητα της σύνταξης, η προφορά, ο τονισμός και η φυσικότητα της ροής υπολείπονται σημαντικά. Παράλληλα η έλλειψη αυθορμητισμού και συναισθηματικού βάθους είναι μια βασική απώλεια από την χρήση Τεχνητής Νοημοσύνης. Οι

μικρές ατέλειες ενός ζωντανού παραγωγού είναι τα κυριότερα στοιχεία με τα οποία ταυτίζεται ο ακροατής.

Επίσης, η χρήση του AI εγείρει ηθικά ζητήματα σχετικά με την κλωνοποίηση φωνής χωρίς συναίνεση, τη δημιουργία παραπλανητικού ή μη επαληθευμένου περιεχομένου και την παραβίαση ιδιωτικότητας μέσω ανάλυσης προσωπικών δεδομένων ακροατών. Η χρήση αυτών των τεχνολογιών απαιτεί ξεκάθαρους κανόνες διαφάνειας, ενημέρωσης και ανθρώπινης επίβλεψης για να υπάρχει εμπιστοσύνη μεταξύ ακροατών και ραδιοφωνικού σταθμού [13].

Αν και πολλοί φοβούνται ότι το AI θα αντικαταστήσει το ανθρώπινο δυναμικό, στην πράξη χρειάζονται οι άνθρωποι του ραδιοφώνου για την επίβλεψη, επιμέλεια και δημιουργική διαμόρφωση του περιεχομένου. Έτσι, το AI λειτουργεί ως υποστηρικτικό και συμπληρωματικό εργαλείο και όχι ως αντικαταστάτης του ανθρώπινου παράγοντα.

## 2.3 Ψηφιακοί Βοηθοί σε Ιστοσελίδες

Τα τελευταία χρόνια, η ενσωμάτωση των Chatbot ως υποστηρικτικό εργαλείο σε ιστοσελίδες αποτελεί μια κοινή γραμμή για πολλές υπηρεσίες. Για παράδειγμα, τα Chatbot έχουν αποκτήσει μια πολύ σημαντική θέση στην εκπαίδευση. Στο Πανεπιστήμιο «Pamantasan ng Lungsod ng Maynila» υπάρχει βοηθός που απαντάει στις ερωτήσεις των υποψηφίων σχετικά με τις διαδικασίες εισαγωγής. Βρίσκεται στην ιστοσελίδα του πανεπιστημίου και εξοικονομεί πολύτιμο χρόνο για το διοικητικό προσωπικό που δεν χρειάζεται να απαντάει την ίδια ερώτηση επανειλημμένως. Αυτό, επίσης, σημαίνει ότι δεν χρειάζεται να περιμένουν οι υποψήφιοι για να απαντηθούν οι ερωτήσεις τους, καθώς η απάντηση είναι προσβάσιμη οποιαδήποτε στιγμή [14-16].

Ένα άλλο παράδειγμα είναι το σύστημα «E-AccesstoAces». Στην ουσία είναι ένας Ψηφιακός Βοηθός που βρίσκεται στην ιστοσελίδα της σχολής οδηγών «Team Aces Driving Academy». Ο βοηθός αυτός προγραμματίζει τα ραντεβού για τα θεωρητικά και πρακτικά μαθήματα. Επίσης, ενημερώνει για τη διαθεσιμότητα των εκπαιδευτών και απαντάει σε συχνές ερωτήσεις των πελατών. Αυτό κάνει την καθημερινή διαχείριση της σχολής πολύ πιο εύκολη [17].

Στον κλάδο του ηλεκτρονικού εμπορίου, οι Ψηφιακοί Βοηθοί προσφέρουν μια πολύ καλή εξυπηρέτηση. Το «ShopAssist» είναι ένα Chatbot που επιτρέπει στους πελάτες να διαχειρίζονται τις παραγγελίες τους. Επιπλέον, μπορούν να υποβάλουν αιτήματα επιστροφής χρημάτων ή αντικατάστασης προϊόντων. Ο πελάτης μπορεί να λαμβάνει προτάσεις αγορών που ταιριάζουν στις ανάγκες του. Αυτό το Chatbot κατανοεί την πρόθεση του χρήστη και προσφέρει έναν δομημένο διάλογο για την άμεση επίλυση προβλημάτων [18].

Συστήματα όπως το «SuperAgent» χρησιμοποιούν δεδομένα όπως λεπτομέρειες προϊόντων και κριτικές πελατών. Αυτό το σύστημα απαντάει με ακρίβεια σε ερωτήματα αγοραστών και λειτουργεί αδιάκοπα, με αποτέλεσμα οι πελάτες να μπορούν να έχουν μια ομαλή εμπειρία αγορών [19].

Ο τραπεζικός τομέας χρησιμοποιεί επίσης τα Chatbot, τα οποία βρίσκονται στις ιστοσελίδες των τραπεζών και μειώνουν τον φόρτο στα τηλεφωνικά κέντρα. Επίσης, προσφέρουν μια ταχύτερη εξυπηρέτηση των πελατών. Για παράδειγμα, η «Qwerty Bank» στην Ινδονησία χρησιμοποιεί πλατφόρμες όπως το Botika και το Rasa. Αυτές οι πλατφόρμες δημιουργούν εικονικούς βοηθούς και οι χρήστες έχουν τη δυνατότητα να ελέγξουν με ασφάλεια το υπόλοιπο του λογαριασμού τους. Επίσης, μπορούν να ενημερωθούν για τα διαθέσιμα προγράμματα ταμιευτηρίου και τα οφέλη τους. Αυτό τους απαλλάσσει την πολύωρη αναμονή στις τηλεφωνικές γραμμές εξυπηρέτησης [20-23].

Τέλος, στον τομέα της ασφάλισης υγείας, τα Chatbot έχουν αναλάβει έναν πολύ κρίσιμο ρόλο. Οι πελάτες μπορούν να ενημερωθούν αναλυτικά για τις καλύψεις των συμβολαίων τους. Επίσης, μπορούν να παρακολουθήσουν την εξέλιξη των ασφαλειών τους σε πραγματικό χρόνο. Οι Ψηφιακοί Βοηθοί μπορούν να αναλύσουν τα δεδομένα του εκάστοτε χρήστη, το οποίο σημαίνει ότι μπορούν να προτείνουν εξατομικευμένα ασφαλιστικά πακέτα. Επιπλέον, μπορούν να βοηθήσουν στην αυτοματοποίηση της επεξεργασίας των αποζημιώσεων, γεγονός που μειώνει δραστικά τον φόρτο εργασίας των ανθρώπινων εκπροσώπων [24].

Συμπερασματικά, η χρήση Ψηφιακών Βοηθών σε ιστοσελίδες, που βασίζονται σε αρχιτεκτονική RAG, προσφέρει σημαντική προστιθέμενη αξία ως προς την ανάγκη υποστήριξης των χρηστών. Τέτοιες εφαρμογές είναι διαθέσιμες διαρκώς, εξυπηρετούν άμεσα και εξοικονομούν πολύτιμο χρόνο από τους χρήστες, ενώ ταυτόχρονα, το σύστημα RAG, βασιζόμενο σε συγκεκριμένα εγχειρίδια της κάθε ιστοσελίδας, συνεισφέρει στην πιο έγκυρη πληροφόρηση.

## 2.4 European School Radio

Το European School Radio είναι το πρώτο μαθητικό ραδιόφωνο της Ευρώπης. Εκπαιδευτικές μονάδες από διαφορετικές χώρες ενώνονται στην ιστοσελίδα του και δημιουργούν μια ραδιοφωνική κοινότητα που δίνει φωνή στους νέους [25].

Για περισσότερο από δέκα χρόνια το European School Radio παρέχει τα εκπαιδευτικά εφόδια και τις υπηρεσίες, ώστε τα σχολεία της Ελλάδας, της Κύπρου και άλλων σχολικών μονάδων διεθνώς, να δημιουργούν και να αναρτούν πρωτότυπο ηχητικό περιεχόμενο. Το αποθετήριο του European School Radio αριθμεί σήμερα περισσότερες από 10.000 αυθεντικές ψηφιακές δημιουργίες, οι οποίες ενισχύουν τη διαπολιτισμική επικοινωνία και κατανόηση.

Η ολοένα αυξανόμενη συμμετοχή και δημιουργία podcast και ηχητικών παραγωγών από σχολεία μέσα στην πάροδο των ετών, σε συνδυασμό με τη δημοφιλία του podcast τόσο στην κοινωνία γενικά, όσο και στην εκπαίδευση ειδικότερα, καθιστά αναγκαία την προβολή των podcasts για εκπαιδευτικούς σκοπούς.

Σε μια εποχή που η πληροφορία ρέει ορμητικά στο διαδίκτυο οι μαθητές και οι μαθήτριες παροτρύνονται να ερευνούν και να αξιολογούν την πληροφορία, να γίνονται κριτικοί αναγνώστες και ακροατές, να ανακαλύπτουν τη γνώση βιωματικά και να παράγουν δικό τους περιεχόμενο σε θεματικές που πραγματεύονται καθ' όλη τη διάρκεια της χρονιάς είτε στα μαθήματα είτε σε σχολικά έργα και διασχολικές συνεργασίες. Το podcast ως εκπαιδευτικό εργαλείο αφενός προσφέρει μοναδικές ευκαιρίες για την ανάπτυξη δεξιοτήτων συνεργασίας, επικοινωνίας, άσκησης του δικαιώματος στην ελευθερία της έκφρασης, αφετέρου λειτουργεί ως ηχητικό ή και οπτικοακουστικό περιεχόμενο με ψηφιακή μορφή που περιέχει λόγο σε αυθεντικό επικοινωνιακό πλαίσιο και δύναται να αξιοποιηθεί εκ νέου στη σχολική τάξη ως διδακτικό υλικό που θα εμπλουτίσει τη μαθησιακή εμπειρία.

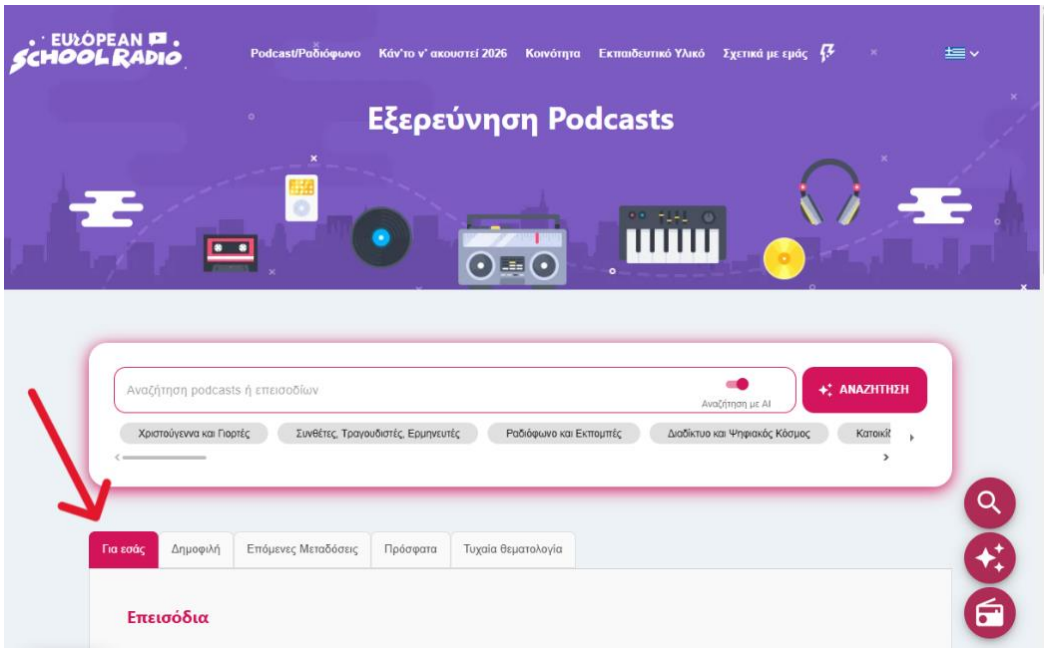
Ως podcast νοείται ένα ηχογραφημένο επεισόδιο, είτε αυτοτελές είτε μέρος μιας σειράς επεισοδίων, που παρουσιάζει ένα θέμα με τη χρήση ποικίλων τεχνικών του ραδιοφωνικού λόγου. Ο αριθμός των ομιλητών/ομιλητριών μπορεί να διαφέρει, από μονόλογο έως πάνελ περισσότερων ατόμων. Ωστόσο, η δομή πρέπει να έχει προηγουμένως σχεδιαστεί και το κείμενο να είναι αποτέλεσμα μιας μελέτης επί της θεματικής που προσεγγίζεται. Επίσης, μπορεί να παρουσιαστεί ένα πρωτότυπο θεατρικό έργο ή ηχητικό ντοκιμαντέρ ή μια ιστορία μυθοπλασίας.

Λαμβάνοντας όλα τα παραπάνω υπόψη, η ιστοσελίδα του European School Radio εξελίσσεται καθημερινά με στόχο να εμπνέει τους μαθητές και τους εκπαιδευτικούς και να τους βοηθάει να

παράγουν το περιεχόμενο που επιθυμούν. Αυτό συμβαίνει καθώς η εταιρία «Διαθεματικό, Διαπολιτισμικό Ραδιόφωνο της Εκπαιδευτικής Κοινότητας» έχει ως στόχο την έρευνα και ανάπτυξη, τη λειτουργία και υποστήριξη του ραδιοφώνου της εκπαιδευτικής κοινότητας «European School Radio, Το Πρώτο Μαθητικό Ραδιόφωνο» και άλλων σχετικών με τα Νέα Μέσα καινοτόμων πληροφοριακών συστημάτων και εργαλείων, για να συμβάλλει, μέσω της ευρύτερης εκπαιδευτικής διαδικασίας, στην καλλιέργεια δεξιοτήτων Γραμματισμού στα Μέσα και στην Πληροφορία, Ψηφιακού και Κριτικού Γραμματισμού προάγοντας την επικοινωνία, ελεύθερη έκφραση ιδεών, τη συνεργασία και, γενικότερα, την ενεργό πολιτότητα στους νέους.

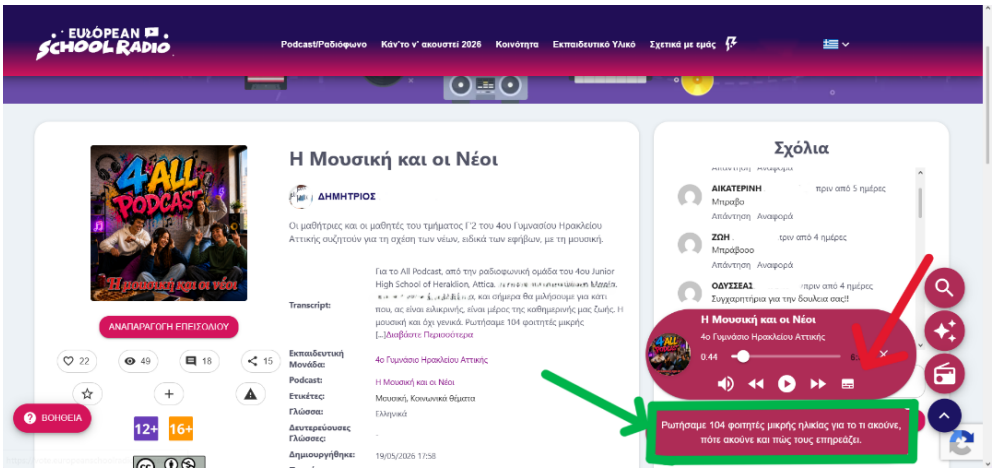
Στην πράξη, το παραπάνω επιβεβαιώνεται με διάφορους τρόπους, έναν από τους οποίους αποτελεί η χρήση εργαλείων AI στην ιστοσελίδα. Οι προγραμματιστές του European School Radio σε συνεργασία με καθηγητές και διδάκτορες έχουν αναπτύξει εργαλεία Τεχνητής Νοημοσύνης που υποστηρίζουν τους χρήστες.

Το πρώτο εργαλείο αποτελεί η λειτουργία προτάσεων με βάση τις προτιμήσεις του χρήστη. Όπως αναφέρθηκε και προηγουμένως, αυτό το εργαλείο βρίσκεται στην «Εξερεύνηση Podcast» της σελίδας, στην επιλογή «Για εσάς» (Σχήμα 2.1) και εμφανίζει προσωποποιημένο περιεχόμενο για τον χρήστη, συγκρίνοντας τις πρόσφατες επιλογές του χρήστη με όλο το υπόλοιπο περιεχόμενο της σελίδας.



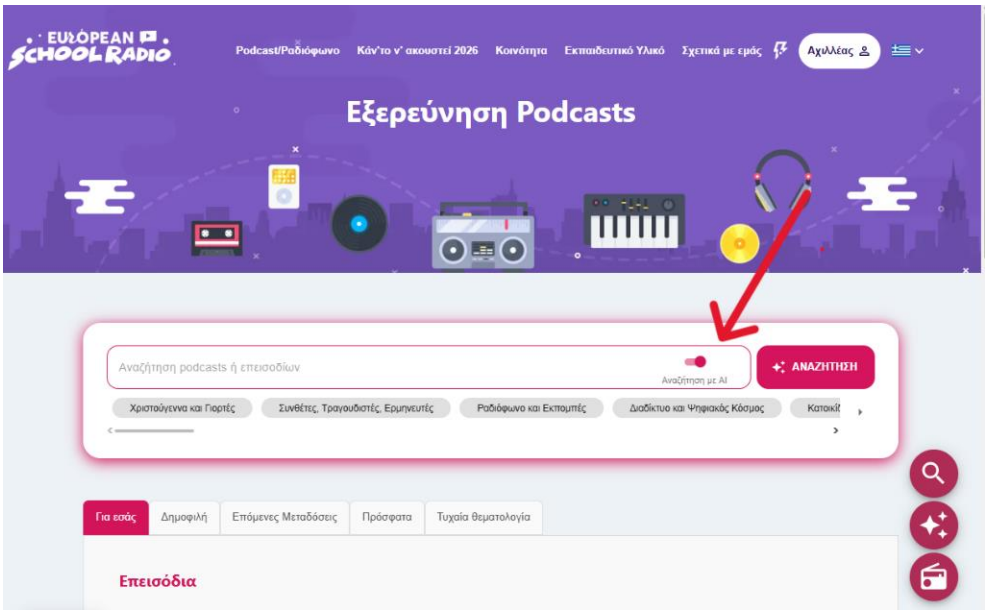

Σχήμα 2.1: Προτάσεις του ιστότοπου βασισμένη στις προτιμήσεις του με χρήση τεχνητής νοημοσύνης

Στην σελίδα, επίσης, χρησιμοποιείται η Τεχνητή Νοημοσύνη προκειμένου να μετατραπούν τα ηχητικά επεισόδια των Podcast σε ακριβή αναπαράσταση κειμένου, γνωστού ως transcript. Αυτό βοηθάει τους χρήστες με δύο τρόπους. Αρχικά, έχοντας τα transcript των επεισοδίων, η ιστοσελίδα μπορεί να τα προσαρμόσει σε ταυτόχρονη αναπαραγωγή με τον ήχο και να δημιουργήσει, τελικά, ακριβείς υπότιτλους (Σχήμα 2.2). Έτσι, η σελίδα αποκτά προσβασιμότητα καθώς το περιεχόμενο γίνεται φιλικό σε άτομα με κώφωση ή σε χρήστες διαφορετικών γλωσσών από την αρχική.




Σχήμα 2.2: Υπότιτλοι επεισοδίου από εργαλείο τεχνητής νοημοσύνης

Το συγκεκριμένο εργαλείο χρησιμοποιείται επίσης και στην αναζήτηση του χρήστη (Σχήμα 2.3). Η σελίδα προσφέρει την δυνατότητα αναζήτησης όχι μόνο με τίτλους ή δημιουργούς αλλά και με το περιεχόμενο των ηχητικών. Γράφοντας μια λέξη στην μπάρα αναζήτησης, η σελίδα εμφανίζει αποτελέσματα που σχετίζονται με την λέξη, ακόμη και αν αυτή αναφέρεται μόνο στο προφορικό μέρος της παραγωγής. Η συγκεκριμένη λειτουργία συγκρίνει την ερώτηση (query) του χρήστη με embeddings, δηλαδή πραγματοποιείται σύγκριση σε νοηματικό επίπεδο.




Σχήμα 2.3: Λειτουργία αναζήτησης με Τεχνητή Νοημοσύνη

Τα παραπάνω αποτελούν μια περίληψη της υπάρχουσας λειτουργίας της ιστοσελίδας του European School Radio που βασίζεται στην Τεχνητή Νοημοσύνη. Ακολουθώντας λοιπόν αυτή την γραμμή, η συγκεκριμένη διπλωματική εργασία αποσκοπεί στο να ενισχύσει την αναβάθμιση της σελίδας με τέτοια εργαλεία. Ένα βασικό πρόβλημα της σελίδας είναι ότι παρέχει τεράστιο αριθμό δυνατοτήτων με αποτέλεσμα οι νέοι χρήστες να αντιμετωπίζουν δυσκολία στο να τις εξερευνήσουν όλες. Όπως αναφέρθηκε στα προηγούμενα κεφάλαια, η εισαγωγή Ψηφιακών Βοηθών σε ιστοσελίδες είναι μια τακτική που αρχίζει να εδραιώνεται όλο και περισσότερο. Λαμβάνοντας όλα τα παραπάνω υπόψη, η πρόκληση που καλούμαστε να αντιμετωπίσουμε είναι η δημιουργία ενός εργαλείου που να μπορεί να συνυπάρχει με τις υπόλοιπες λειτουργίες, να είναι φιλικό προς όλους τους χρήστες και να ικανοποιεί τις ανάγκες της εταιρείας.

# Κεφάλαιο 3ο: Θεωρητικό Υπόβαθρο

## 3.1 Τεχνητή Νοημοσύνη και Σύγχρονοι Ψηφιακοί Βοηθοί

Ο όρος μάθηση περιγράφει την δυνατότητα μιας οντότητας να χρησιμοποιήσει δεδομένα που λαμβάνει από το περιβάλλον γύρω της, ώστε να βελτιώνεται στις λειτουργίες που απαιτούν τα συγκεκριμένα ή παρόμοια δεδομένα [26]. Με την ύπαρξη της διαδικασίας της μάθησης θεωρείται δυνατή και η ύπαρξη της νοημοσύνης. Όταν η οντότητα είναι μια υπολογιστική μηχανή, τότε αναφερόμαστε στην Μηχανική Μάθηση, κατά την οποία η χρήση και επεξεργασία των δεδομένων βασίζεται σε έναν αλγόριθμο.

Ορίζοντας προϋπόθεση της νοημοσύνης την μάθηση, είναι φυσικό επακόλουθο να ισχύει η ανάλογη σχέση με την Τεχνητή Νοημοσύνη και την Μηχανική Μάθηση. Στην περίπτωση αυτή, για να «μάθει» το υπολογιστικό σύστημα, δηλαδή να παρατηρηθεί η βελτίωση της λειτουργίας του, συνηθίζεται να τροφοδοτείται ο αλγόριθμος με ένα μεγάλο πλήθος παραδειγμάτων. Η μάθηση γίνεται σταδιακά καθώς ο αλγόριθμος εξετάζει τα παραδείγματα επαναλαμβανόμενα [27].

Αν μπορούσαμε να δώσουμε έναν ορισμό στην Τεχνητή Νοημοσύνη θα λέγαμε ότι αναφέρεται στην επιστήμη και μηχανική των υπολογιστικών συστημάτων που μπορούν να μιμηθούν ανθρώπινες λειτουργίες όπως η παραγωγή λόγου, η επίλυση προβλημάτων, η χρήση της λογικής σκέψης [26]. Η παραδοσιακή Τεχνητή Νοημοσύνη στηρίζεται σε κανόνες και αυτό, με μία βαθύτερη ματιά, είναι λογικό. Σε ένα σύστημα που καλείται να ξεχωρίζει τα μήλα από τα πορτοκάλια, είναι αρκετό να οριστούν κανόνες όπως «το μήλο είναι κόκκινο» ή «το πορτοκάλι είναι πιο βαρύ» για να μπορέσει να έχει μεγάλη αποτελεσματικότητα. Αυτό, όμως, δεν είναι αρκετό σε όλες τις περιπτώσεις. Αν στο παράδειγμα με τα φρούτα παραπάνω, εμφανιστεί ένα πράσινο μήλο, το σύστημα μας δεν θα μπορεί να απαντήσει αν δεν έχουμε προσδιορίσει αυτή την ειδική περίπτωση του κανόνα. Αυτό συμβαίνει γιατί σε προβλήματα που δεν υπάρχει ξεκάθαρο ή ολοκληρωμένο πλαίσιο κανόνων, η αποτελεσματικότητα της Τεχνητής Νοημοσύνη μειώνεται. Η χαμηλή ευελιξία και η αδυναμία προσαρμογής σε νέα δεδομένα είναι τα βασικότερα μειονεκτήματα της παραδοσιακής προσέγγισης της Τεχνητής Νοημοσύνη [28]. Στο πλαίσιο αυτό, λοιπόν, έρχεται η σπουδαιότητα της μάθησης καθώς σε τέτοια συστήματα δίνει τη δυνατότητα παραγωγής ικανοποιητικού αποτελέσματος με δεδομένα που εμφανίζονται για πρώτη φορά στο σύστημα [26].

Με τον τρόπο αυτό δημιουργείται ένα ιδιαίτερο εργαλείο που μπορεί να χρησιμοποιηθεί για να βοηθήσει σε τομείς που ο ανθρώπινος παράγοντας μπορεί να προκαλέσει προβλήματα. Για παράδειγμα, η ιατρική και η υγεία είναι ένα τομέας άκρως επηρεασμένος από εργαλεία Τεχνητής Νοημοσύνη. Υπάρχουν εργαλεία που μπορούν αναγνωρίσουν ασθένειες, να τις περιγράψουν και να προτείνουν εναλλακτικές θεραπείες και ιατρικές συνταγές. Άλλα πάλι, με την χρήση Μηχανικής Μάθησης προβλέπουν καρδιακές παθήσεις από την χρήση ελεύθερου κειμένου και φωνής κατηγοριοποιώντας τα σε συμπτώματα [27].

Με τον όρο Chatbot αναφερόμαστε σε ένα πρόγραμμα υπολογιστή που έχει αποκτήσει την δυνατότητα να μιμείται την ανθρώπινη επικοινωνία [5]. Αυτή η επικοινωνία πλέον μπορεί να είναι είτε γραπτή, είτε προφορική, καθώς επίσης μπορεί να εκμεταλλευτεί και οπτικοακουστικό υλικό. Ανάλογα με την εφαρμογή που σκοπεύουν να έχουν, στα Chatbot δίνονται διαφορετικές ικανότητες με διαφορετικές μεθόδους [5].

Όπως ειπώθηκε και στην Ενότητα 2.3, τα Chatbot αποτελούν μια εύχρηστη μορφή Τεχνητής Νοημοσύνη για διάφορους λόγους. Οι ανάγκες του ανθρώπου στην σημερινή εποχή είναι πολλές και οι απαιτήσεις προς τους παρόχους υπηρεσιών αυξάνονται αντίστοιχα. Η γρήγορη και αποτελεσματική

εξυπηρέτηση διαδραματίζει καθοριστικό ρόλο στην επιτυχία και στην διατήρηση ανταγωνιστικού πλεονεκτήματος [28].

Ένα Chatbot για να θεωρηθεί αποτελεσματικό θα πρέπει να έχει τέσσερα χαρακτηριστικά. Αρχικά, θα πρέπει να μπορεί να κατανοεί το πλαίσιο στο οποίο ο χρήστης επιχειρεί να συζητήσει. Δεν είναι αρκετό να αναλύει απλώς μεμονωμένες λέξεις αλλά θα πρέπει να αντιλαμβάνεται την σύνδεση των λέξεων που χρησιμοποιεί ο χρήστης. Επίσης, το ιδανικό Chatbot μπορεί να εντοπίσει συναισθήματα. Ένα πολύ κρίσιμο χαρακτηριστικό διότι έτσι μπορεί να απαντάει με πιο φυσικούς και ανθρώπινους τρόπους [27]. Στην συνέχεια, τα Chatbot πρέπει να μαθαίνουν από τις ίδιες τις συνομιλίες που πραγματοποιούν. Αυτό το χαρακτηριστικό είναι πιο απαιτητικό και δεν απευθύνεται στην μνήμη που μπορεί να διατηρεί εντός του chat, αλλά στην δυνατότητα να μετατρέπει τις συνομιλίες σε δεδομένα. Τέλος, θα πρέπει να βελτιώνονται καθημερινά ώστε να προσαρμόζονται στις προτιμήσεις του χρήστη [29]. Προφανώς όμως, αυτή η περιγραφή δεν μπορεί να είναι πάντα εφικτή και γι' αυτό συνηθίζεται η χρήση ποσοτικών μετρικών για τον υπολογισμό της αποτελεσματικότητας [30].

Κάθε Chatbot ανάλογα με την χρήση που προορίζεται να εκτελέσει, έχει κάποιες διαφορές στην λειτουργία του. Στο πλαίσιο αυτής της εργασίας μελετάται η βασική δομή που αξιοποιεί αρχιτεκτονική τύπου RAG. Η δομή αυτή αποτελείται από τη διεπαφή χρήστη (User Interface - UI), στην οποία ο χρήστης παραθέτει την ερώτηση και αποτελεί το πρόσωπο του συστήματος, καθώς είναι η μοναδική επαφή που εμφανίζεται στον χρήστη. Η ερώτηση αυτή μετατρέπεται σε διανύσματα με τα κατάλληλα μοντέλα ώστε να μπορούν να συγκριθούν με τα διανύσματα στην βάση δεδομένων και να εντοπιστεί η πιο κοντινή απάντηση. Στην συνέχεια ένα Γλωσσικό Μοντέλο (Language Model – LM) αναδιαμορφώνει την απάντηση σε γραπτό λόγο ώστε να την λάβει ο χρήστης σε μια κατανοητή μορφή. Κάθε βήμα αυτής της δομής θα αναλυθεί στο παρόν κεφάλαιο.

## 3.2 Ευφυείς Πράκτορες (AI Agents)

Πριν συνεχίσει η εργασία θα πρέπει να οριστεί η έννοια των Ευφυών Πρακτόρων. Ο όρος αυτός περιγράφει ένα λογισμικό το οποίο μπορεί να αλληλεπιδρά με το περιβάλλον του, να μαθαίνει από αυτό και να εκτελεί ενέργειες που αποφασίζει αυτόνομα [31]. Υπάρχουν πολλά είδη πρακτόρων ανάλογα με τον στόχο που καλούνται να πετύχουν. Στη Μηχανική Μάθηση ορίζεται ο υπολογιστής που εκτελεί επαναλαμβανόμενα μια εργασία κάνοντας δοκιμές. Τα Chatbot αποτελούν μια μορφή Συνομιλητικού Πράκτορα, καθώς η λειτουργία τους είναι αυτόνομη και στοχευμένη. Σημαντική αναφορά αποτελούν οι Συνεργατικοί Πράκτορες, καθώς έχουν την δυνατότητα να συνεργάζονται τόσο με ανθρώπους, όσο και με άλλους πράκτορες για να πετύχουν στόχους. Μάλιστα, ερευνητές δημιουργούν αρχιτεκτονικές που πολλαπλά Chatbot επικοινωνούν μεταξύ τους για να βελτιώσουν την συνολική απόδοση του συστήματος [34]. Στην πράξη, τα περισσότερα σύγχρονα Chatbot παραγωγής δεν είναι απλοί Agents, αντιθέτως, είναι RAG-based συστήματα που έχουν «κουμπωμένα» πάνω τους ένα ή δύο εργαλεία Agent, για να εκτελούν συγκεκριμένες ενέργειες [31].

Η χρήση των πρακτόρων αποτελεί μια σημαντική εξέλιξη, καθώς οι εφαρμογές τους μπορούν να προσαρμοστούν σε πολλές περιπτώσεις. Για παράδειγμα, ένας χαμηλού κόστους και επεκτάσιμος πράκτορας χρησιμοποιήθηκε μέσω της πλατφόρμας n8n για να προωθεί ενημερώσεις για την κυβερνοασφάλεια [32]. Το αποτέλεσμα ήταν να αυξηθεί η αναγνωσιμότητα των κειμένων ενημέρωσης κατά 35%.

Μια εξειδικευμένη εταιρεία ανάπτυξης AI Agents σχεδιάζει και υλοποιεί έξυπνα συστήματα τα οποία έχουν την ικανότητα να εκτελούν ροές εργασίας πολλαπλών βημάτων (multi-step workflows) και να αλληλεπιδρούν αυτόνομα με APIs και διάφορα εταιρικά εργαλεία. Επιπλέον, τα συστήματα αυτά

φροντίζουν να διατηρούν σταθερό το πλαίσιο (context) της συζήτησης καθ' όλη τη διάρκεια των αλληλεπιδράσεων με τους χρήστες και είναι σε θέση να παράγουν δομημένα αποτελέσματα, όπως αναφορές και συγκεκριμένες ενέργειες. Τέλος, μέσα από την καθημερινή τους λειτουργία, βελτιώνονται συνεχώς στην πράξη αξιοποιώντας τους βρόχους ανατροφοδότησης [33].

Έχει αποδειχθεί ότι οι σύνθετες ροές εργασίας πρέπει να αποσυντίθενται (careful decomposition) σε ξεχωριστές, ξεκάθαρες διαδικασίες. Αυτές οι διαδικασίες διοικούνται από εξειδικευμένου AI Agents που αλληλεπιδρούν με εξωτερικά συστήματα, ανθρώπους ή άλλους AI Agents [34].

## 3.3 Γλωσσικά Μοντέλα

Τα Γλωσσικά Μοντέλα είναι μια τεχνολογία επεξεργασίας φυσικής γλώσσας στον τομέα της Τεχνητής Νοημοσύνη. Τα μοντέλα αυτά εκπαιδεύονται σε τεράστια ποσότητα δεδομένων κειμένου και βασίζονται σε αρχιτεκτονικές νευρωνικών δικτύων [35], [36]. Προκειμένου, να αποκτήσουν ισχυρές γλωσσικές δεξιότητες και να προσαρμόζονται σε διαφορετικές εργασίες, χρησιμοποιούν τεχνικές μάθησης χωρίς επίβλεψη. Η εκπαίδευσή τους τους επιτρέπει να εκτελούν συλλογισμούς και βαθιά κατανόηση του πλαισίου ενός κειμένου. Μπορούν να διαχειριστούν πολύπλοκους διαλόγους και δυναμικές αλληλεπιδράσεις ή να συνοψίσουν με επιτυχία πολύπλοκα τεχνικά έγγραφα [37].

Στα Chatbot χρησιμοποιούνται για να παράγουν φυσικό λόγο που μιμείται τον ανθρώπινο. Παρά τις μεγάλες δυνατότητες που προσφέρουν, ένα βασικό πρόβλημα που αντιμετωπίζουν είναι οι «παραισθήσεις» [35]. Επίσης, Οι εταιρείες που αναπτύσσουν Chatbot βασισμένα σε Γλωσσικά Μοντέλα χρειάζονται δεδομένα για το πώς οι χρήστες αλληλεπιδρούν με αυτά (π.χ. αναδυόμενες τάσεις, απογοητεύσεις χρηστών, δημοφιλή θέματα) ώστε να βελτιώνουν τις υπηρεσίες τους, γεγονός που δημιουργεί προβλήματα ασφαλείας καθώς οι συνομιλίες περιέχουν συχνά άκρως προσωπικά ή ευαίσθητα δεδομένα [36].

### 3.3.1 Gemma

Για πρώτη φορά, η Google κυκλοφόρησε το Gemma 4 με την άδεια Apache 2.0, που σημαίνει ότι είναι εντελώς δωρεάν και open-source [37]. Το Gemma 4, το οποίο ανήκει στην κατηγορία Μεγάλων Γλωσσικών Μοντέλων, έρχεται σε διάφορες εκδόσεις: E2B και E4B (για συσκευές), 31B Dense και 26B MoE (Mixture of Experts). Τα μικρότερα μοντέλα μπορούν να χειριστούν έως 128K tokens και τα μεγαλύτερα έως 256K, ενώ το μοντέλο 31B έχει context 32K [37], [39].

Αυτό το μοντέλο είναι πολυτροπικό, δηλαδή μπορεί να λάβει εισαγωγή κειμένου, εικόνας και βίντεο. Επιπλέον, οι εκδόσεις E2B και E4B μπορούν να λάβουν και ήχο. Όσον αφορά τις επιδόσεις, το μοντέλο 31B έχει το υψηλότερο σκορ στα benchmarks όρασης με 76.9% στο MMMU Pro. Είναι επίσης 3ο στην παραγωγή κειμένου στο Arena text leaderboard. Το μοντέλο 26B MoE είναι στην 6η θέση στην ίδια κατηγορία [37], [38].

Ειδικά για τα edge μοντέλα (E2B/E4B), εισάγεται ένας δεύτερος πίνακας embedding που τροφοδοτεί με ένα μικρό residual σήμα κάθε layer του decoder, μεγιστοποιώντας την αποδοτικότητα των παραμέτρων. Όλα τα μοντέλα περιλαμβάνουν ένα ενσωματωμένο draft μοντέλο για speculative decoding, επιτρέποντας σημαντικά ταχύτερο inference χωρίς καμία απώλεια στην ποιότητα [39].

Σε αντίθεση με το Gemma 3, το Gemma 4 χρησιμοποιεί τους τυπικούς standard ρόλους συστήματος (system), βοηθού (assistant) και χρήστη (user), προσφέροντας εγγενή υποστήριξη για System Prompts [40].

### 3.3.2 Llama

Το Llama παρουσιάζεται συχνά ως το πιο ανεπτυγμένο Μικρό Γλωσσικό Μοντέλο (Small Language Models - SLM), λόγω της μεγάλης συμβατότητας με άλλα εργαλεία αυτού του τομέα [42-44]. Τα ιδιαίτερα χαρακτηριστικά του είναι η μικρή ποσότητα KV cache που καταναλώνει και η γρήγορη εξαγωγή συμπερασμάτων.

Το Llama θεωρείται πιο αποτελεσματικό κυρίως σε σενάρια που απαιτούν μεγάλο παράθυρο πλαισίου (long context), τοπική εγκατάσταση (self-hosting) για λόγους προστασίας δεδομένων και σύνθετη συλλογιστική (agentic workflows) [41]. Πολλές φορές εμφανίζεται καλύτερο [42].

Ένα από τα μεγαλύτερα πλεονεκτήματα του Llama είναι η δυνατότητα τροποποίησης των εσωτερικών βαρών (μέσω μεθόδων όπως SFT, RLHF, LoRA/QLoRA) για την ενσωμάτωση εξειδικευμένης γνώσης, προσφέροντας βελτίωση ακρίβειας άνω του 20% σε σχέση με τα απλά prompts [43], [44].

### 3.3.3 DeepSeek

Το DeepSeek είναι μια μεγάλη ομάδα γλωσσικών μοντέλων και Chatbot που δημιουργήθηκε από μια κινεζική εταιρεία Τεχνητής Νοημοσύνη. Αυτή η ομάδα μοντέλων έχει σημαντική θέση στο παγκόσμιο τοπίο του AI λόγω των υψηλής απόδοσης και ανοιχτού κώδικα μοντέλων της. Η δομή του βασίζεται στη τεχνολογία «Mixture-of-Experts» (MoE) [45], η οποία μαζί με την προσέγγιση Multi-Head Latent Attention (MLA) και την πρόβλεψη πολλαπλών tokens (Multi-Token Prediction - MTP), καταφέρνει να μεγιστοποιήσει την υπολογιστική απόδοση κατά την εκπαίδευση και την παραγωγή απαντήσεων [45]. Το DeepSeek-V3, με 671 δισεκατομμύρια παραμέτρους και 37 δισεκατομμύρια ενεργές ανά token, κατάφερε να φτάσει τις επιδόσεις κορυφαίων κλειστών μοντέλων όπως το GPT-4o [45]. Ταυτόχρονα, διατήρησε το κόστος εκπαίδευσης σε χαμηλά επίπεδα. Βασισμένο στο V3, το μοντέλο DeepSeek-R1 εισήγαγε προηγμένες τεχνικές ενισχυτικής μάθησης (GRPO) εξειδικευμένες στον συλλογισμό που πέτυχε κορυφαία αποτελέσματα σε πολύπλοκα μαθηματικά προβλήματα και συγγραφή κώδικα. Τον Απρίλιο του 2026, κυκλοφόρησε η σειρά DeepSeek-V4, ένα από τα ευρέως γνωστά LLM. Αυτή η σειρά περιλαμβάνει το V4-Pro με 1,6 τρισεκατομμύριο παραμέτρους και το μικρότερο V4-Flash. Η σειρά προσφέρει ένα μεγάλο παράθυρο πλαισίου 1 εκατομμυρίου tokens. Τα περισσότερα μοντέλα της DeepSeek είναι διαθέσιμα υπό την ελεύθερη άδεια MIT. Επίσης, προσφέρει πρόσβαση σε API με χαμηλό κόστος. Η DeepSeek έχει αλλάξει τα δεδομένα στη βιομηχανία, παρέχοντας νοημοσύνη αιχμής που ανταγωνίζεται τους τεχνολογικούς κολοσσούς [46].

### 3.3.4 Gemini

Το Gemini είναι μια εξαιρετική οικογένεια Μεγάλων Γλωσσικών Μοντέλων που ανέπτυξε η Google DeepMind. Ένα από τα πιο εντυπωσιακά χαρακτηριστικά του Gemini είναι η ικανότητά του να επεξεργάζεται και να κατανοεί τεράστιες ποσότητες δεδομένων, μέχρι και 10 εκατομμύρια tokens [45]. Η αρχιτεκτονική του Gemini βασίζεται στην τεχνολογία «MoE», η οποία κατανέμει την επεξεργασία σε εξειδικευμένα τμήματα του νευρωνικού δικτύου [46]. Αυτό βελτιώνει σημαντικά την αποδοτικότητα και την υπολογιστική του ταχύτητα. Το Gemini δεν λειτουργεί μόνο ως ένα αυτόνομο Chatbot, αλλά έχει ενσωματωθεί σε ολόκληρο το οικοσύστημα του Google Workspace [47]. Αυτό σημαίνει ότι μπορεί να αναλάβει ρόλο κεντρικού ψηφιακού βοηθού σε όλα τα εργαλεία όπως Gmail, Docs, Sheets κ.α. [48].

## 3.4 Διανυσματική Αναπαράσταση και Σημασιολογική Ομοιότητα (Vector Embeddings)

Στην εποχή μας, με την Τεχνητή Νοημοσύνη να εξελίσσεται συνεχώς, η ανάγκη για επεξεργασία μη δομημένων δεδομένων, όπως κείμενα, εικόνες και βίντεο, έχει οδηγήσει σε μια σημαντική αλλαγή.

Πλέον, η παραδοσιακή αναζήτηση με βάση λέξεις-κλειδιά έχει αντικατασταθεί από συστήματα που κατανοούν το βαθύτερο νόημα, την πρόθεση του χρήστη και το πλαίσιο [49]. Τα διανύσματα, τα embeddings και οι Διανυσματικές Βάσεις Δεδομένων είναι ο τρόπος που συνέβη η αλλαγή. Ένα embedding είναι ένας τρόπος μετατροπής του νοήματος μιας πληροφορίας σε ένα διάνυσμα, δηλαδή σε μια λίστα αριθμών. Αυτό επιτρέπει στα συστήματα να μεταφράζουν τη σημασιολογία σε γεωμετρία, όπου κάθε θέση στο διάνυσμα αντιστοιχεί σε μια λανθάνουσα διάσταση νοήματος [50].

Το βασικότερο χαρακτηριστικό των embeddings είναι ότι η γεωμετρική εγγύτητα στον πολυδιάστατο χώρο αντιστοιχεί σε σημασιολογική ομοιότητα. Για παράδειγμα, οι έννοιες «έσοδα πελατών» και «εισόδημα πελάτη» θα παράγουν διανύσματα που βρίσκονται πολύ κοντά γεωμετρικά, παρόλο που δεν μοιράζονται κοινές λέξεις [51]. Αντίθετα, οι όροι «έσοδα πελατών» και «μετανάστευση βάσης δεδομένων» θα βρεθούν πολύ μακριά στον χώρο αυτό.

Η παραγωγή και η χρήση των embeddings ακολουθούν συγκεκριμένα πρότυπα ανάλογα με τον τύπο των δεδομένων. Τα σύγχρονα μοντέλα παράγουν διανύσματα που κυμαίνονται συνήθως από 384 έως 3.072 διαστάσεις. Οι περισσότερες διαστάσεις προσφέρουν μεγαλύτερη ακρίβεια και εκφραστικότητα, αλλά απαιτούν περισσότερη υπολογιστική ισχύ και αυξάνουν την καθυστέρηση [52].

Τέλος, όταν τα δεδομένα μετατραπούν σε διανύσματα, δεν μπορούν να αποθηκευτούν αποτελεσματικά σε παραδοσιακές σχεσιακές βάσεις δεδομένων. Εκεί αναλαμβάνει η Vector Database, η οποία είναι βελτιστοποιημένη για τη γρήγορη εύρεση των κοντινότερων γειτόνων σε περιβάλλοντα εκατομμυρίων διανυσμάτων.

## 3.5 Επαυξημένη Παραγωγή μέσω Ανάκτησης (RAG)

Το RAG είναι μια αρχιτεκτονική που βοηθά τα Μεγάλα Γλωσσικά Μοντέλα να έχουν πρόσβαση σε εξωτερικές πληροφορίες σε πραγματικό χρόνο. Αυτό σημαίνει ότι μπορούν να δώσουν απαντήσεις που βασίζονται σε πραγματικά δεδομένα και όχι μόνο σε αυτά που έχουν μάθει κατά την εκπαίδευσή τους [53], [54]. Το RAG επιλύει τους περιορισμούς των στατικών μοντέλων, τα οποία δεν μπορούσαν να ενημερώσουν τις γνώσεις τους εύκολα και υπήρχε πιθανότητα να δώσουν λάθος απαντήσεις [55].

Τα πλεονεκτήματα του RAG είναι πολλά. Οι απαντήσεις που δίνει είναι πιο ακριβείς και ποικίλες, ενώ δίνει μεγαλύτερο έλεγχο και ερμηνευσιμότητα στο σύστημα. Η πηγή της γνώσης είναι διαφανής και ελέγξιμη, και είναι σε μορφή απλού κειμένου [56].

Η λειτουργία του RAG βασίζεται σε δύο στάδια: το Στάδιο Ανάκτησης και το Στάδιο Παραγωγής. Κατά το Στάδιο Ανάκτησης, τα έγγραφα-πηγές αναλύονται και μετατρέπονται σε διανυσματικές αναπαραστάσεις, οι οποίες αποθηκεύονται σε μια βάση δεδομένων. Όταν ο χρήστης κάνει ένα ερώτημα, το σύστημα ανακτά τα πιο σχετικά αποσπάσματα εγγράφων και τα χρησιμοποιεί για να δημιουργήσει μια απάντηση. Το Στάδιο Παραγωγής είναι όπου το περιεχόμενο που ανακτήθηκε μορφοποιείται και ενσωματώνεται σε ένα πρότυπο prompt, μαζί με το αρχικό ερώτημα του χρήστη. Το prompt αυτό προωθείται σε ένα Μεγάλο Γλωσσικό Μοντέλο, το οποίο συνθέτει και παράγει την τελική απάντηση [57].

Για την υλοποίηση του RAG, μπορούν να χρησιμοποιηθούν διάφορα εργαλεία και τεχνολογίες, όπως το Streamlit, το Chainlit, το SentenceTransformers και η FAISS. Το RAG είναι κατάλληλο για εφαρμογές που απαιτούν λεπτομερή ακρίβεια και συνεχή αναφορά σε εξωτερικές πηγές. Τέτοιες χρήσεις εντοπίζονται ευρέως στις επιχειρήσεις, τον νομικό τομέα, την εκπαίδευση, την υγεία, τα οικονομικά και την επικαιρότητα που κάποιο λάθος μπορεί να προκαλέσει σοβαρά προβλήματα [58], [59].

## 3.6 Στρατηγικές Τεμαχισμού Κειμένου (Text Chunking)

Ο όρος Τεμαχισμός Κειμένου (Chunking) αναφέρεται στην διαδικασία διαχωρισμού ενός μεγάλου κειμένου σε μικρότερα τμήματα εφαρμόζοντας συγκεκριμένους κανόνες [60-65]. Για να μπορέσει το RAG να είναι λειτουργικό, θα πρέπει να μπορεί να επιστρέφει τμήματα πληροφορίας ανάλογα με την ανάγκη της κάθε περίπτωσης. Η διαδικασία του Chunking είναι το πρώτο βήμα σε συστήματα Chatbot και είναι ιδιαίτερα σημαντική για να μπορεί ο χρήστης να λαμβάνει την ακριβή πληροφορία που επιθυμεί [53]. Η αποδοτικότητα του Chunking βασίζεται σε:

- **Μέγεθος Chunking:** Είναι το μέγιστο μήκος κειμένου. Αν κοπεί σε μικρά κομμάτια υπάρχει κίνδυνος αλλοίωσης της πληροφορίας ενώ τα μεγάλα chunks φέρουν κίνδυνο δημιουργίας «παραισθήσεων».
- **Επικάλυψη τμημάτων:** Η δυνατότητα το ένα chunk να καλύπτει το επόμενο για να μπορεί να υπάρχει νοηματική συνέχεια.

Υπάρχουν πολλοί μέθοδοι τμηματοποίησης ενός κειμένου που η αποτελεσματικότητα τους βασίζεται στην μορφή και την πληροφορία του κειμένου [65-72].

- **Τμηματοποίηση Σταθερού Μεγέθους (Fixed-Size Chunking):** Το κείμενο διασπάται αυστηρά σε ίσα κομμάτια κειμένου ανάλογα με τον αριθμό Token που έχει οριστεί.
- **Δομική Τμηματοποίηση (Structural / Rule-Based Chunking):** Αυτή η μέθοδος βασίζεται σε ξεκάθαρους και προκαθορισμένους κανόνες, με βάση την ιεραρχία του κειμένου, για να δημιουργήσει τα τελικά chunks.
- **Αναδρομική Τμηματοποίηση (Recursive Chunking):** Σε αυτή την μέθοδο ορίζεται το μέγιστο μήκος των chunks και μετά εντοπίζει διαχωριστές κειμένου με συγκεκριμένη προτεραιότητα, όπως η αλλαγή παραγράφου ή η τελεία για να σταματήσει τον διαχωρισμό εντός του ορίου.
- **Σημασιολογική Τμηματοποίηση (Semantic Chunking):** Διαχωρίζεται το κείμενο σε προτάσεις και στην συνέχεια αυτές ομαδοποιούνται με κριτήριο το πόσο κοντά βρίσκονται νοηματικά.

## 3.7 Αξιολόγηση Συστημάτων RAG

Η αξιολόγηση συστημάτων RAG είναι μια κρίσιμη διαδικασία για να διασφαλιστεί ότι ένα Chatbot απαντάει με ακρίβεια, πληρότητα και αξιόπιστα χωρίς να επηρεάζεται από «παραισθήσεις». Πλέον την αξιολόγηση πραγματοποιούν αυτοματοποιημένα εργαλεία για μεγαλύτερη ευκολία, ταχύτητα και αποτελεσματικότητα [73], [74].

Το RAGAS είναι ένα εργαλείο που χρησιμοποιεί LLM για να κρίνει την λειτουργία ενός Chatbot. Η αξιολόγηση συμβαίνει σε διάφορα πεδία ώστε να μπορεί να δοκιμαστεί ολοκληρωτικά το Chatbot [75]. Για παράδειγμα, η ακρίβεια της απάντησης ή η ικανότητα ανάκτησης του συστήματος, η πιστότητα και η συνάφεια είναι κάποιες μερικές που αποδίδουν μια σφαιρική εικόνα της αποτελεσματικότητας [76].

Η αξιολόγηση επίσης μπορεί να επηρεαστεί από διάφορους παράγοντες όπως:

- **Την πολυπλοκότητα των ερωτήσεων:** Ερωτήσεις με σύνθετο περιεχόμενο ή περίεργη διατύπωση μπορεί να προκαλέσει ευκολότερα «παραισθήσεις» [77].
- **Την φύση του αρχικού κειμένου:** Ένα εκτεταμένο κείμενο έχει διαφορετικές ανάγκες για να μπορέσει να χρησιμοποιηθεί επαρκώς από το σύστημα σε σύγκριση με ένα πιο σύντομα διατυπωμένο [78].
- **Τον τελικό κριτή:** Η ανθρώπινη κρίση μπορεί να θεωρηθεί αμφιλεγόμενη, ενώ διαφορετικά γλωσσικά μοντέλα παρουσιάζουν μεταβαλλόμενη απόδοση.

# Κεφάλαιο 4ο: Υλοποίηση του Συστήματος

## 4.1 Περιβάλλον Ανάπτυξης και Εργαλεία Υλοποίησης

Για την υλοποίηση της παρούσας διπλωματικής εργασίας και την ανάπτυξη του συστήματος RAG, αξιοποιήθηκε ένα σύνολο από σύγχρονα εργαλεία και τεχνολογίες. Τα κυριότερα κριτήρια επιλογής τους ήταν η υποστήριξη ανοιχτού κώδικα (open-source), η διαλειτουργικότητα και το εύχρηστο περιβάλλον ανάπτυξης. Στις παρακάτω υποενότητες γίνεται μία συνοπτική παρουσίαση όλων των βασικών εργαλείων που συνέβαλαν στην υλοποίηση της έρευνας.

### 4.1.1 Python

Η Python δημιουργήθηκε από τον Guido van Rossum στις αρχές της δεκαετίας του 1990 και αυτή τη στιγμή βρίσκεται στην έκδοση 3.14.5 (10 Μαΐου 2026). Από την κυκλοφορία της μέχρι και σήμερα, είναι ευρέως γνωστή και έχει καθιερωθεί ως μία από τις πιο διαδεδομένες γλώσσες προγραμματισμού υψηλού επιπέδου και γενικής χρήσης. Η αρχιτεκτονική της εστιάζει κυρίως στην ευαναγνωστότητα του κώδικα και στην υποστήριξη πολλαπλών προγραμματιστικών παραδειγμάτων, όπως ο αντικειμενοστρεφής και ο συναρτησιακός προγραμματισμός. Είναι ιδιαίτερα δημοφιλής στους τομείς της Τεχνητής Νοημοσύνης και της Μηχανικής Μάθησης, καθώς προσφέρει ένα τεράστιο οικοσύστημα ανοιχτών βιβλιοθηκών, βελτιστοποιώντας έτσι τη διαδικασία ανάπτυξης ενός συστήματος. Επιλέχθηκε ως ο βασικός πυρήνας για τη συγγραφή των σεναρίων εκτέλεσης (scripts) για όλα τα στάδια της έρευνας, από την προετοιμασία των δεδομένων μέχρι και την τελική αξιολόγηση. Από τα κυριότερα πλεονεκτήματά της είναι η εύκολη διαχείριση του μεγάλου όγκου δεδομένων που είχαν τα πειράματα (μέσω της βιβλιοθήκης *pandas*), καθώς και η δυνατότητα παράλληλης και ασύγχρονης εκτέλεσης διεργασιών (μέσω της βιβλιοθήκης *asyncio*) [79].

### 4.1.2 Visual Studio Code

Το Visual Studio Code (γνωστό ως VS Code) είναι ένα δωρεάν, ανοιχτού κώδικα, ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE) που αναπτύχθηκε από τη Microsoft και κυκλοφόρησε το 2015, ενώ αυτή τη στιγμή βρίσκεται στην έκδοση 1.121.0 (20 Μαΐου 2026). Είναι εξαιρετικά ελαφρύ, ταχύτατο και πλήρως παραμετροποιήσιμο και προσφέρει ισχυρά εργαλεία όπως η αποσφαλμάτωση (Debugging), η έξυπνη συμπλήρωση κώδικα και το ενσωματωμένο τερματικό (Integrated Terminal). Αποτελεί το πιο δημοφιλές εργαλείο ανάπτυξης λογισμικού παγκοσμίως, καθώς υποστηρίζει σχεδόν όλες τις γλώσσες προγραμματισμού και είναι ιδιαίτερα ευέλικτο μέσω του τεράστιου οικοσυστήματος επεκτάσεων (extensions) που διαθέτει. Επιλέχθηκε ως το κύριο περιβάλλον συγγραφής και ελέγχου του κώδικα, λόγω της άριστης συμβατότητάς του με την Python και της διευκόλυνσης που πρόσφερε το ενσωματωμένο τερματικό στην άμεση εγκατάσταση των απαραίτητων βιβλιοθηκών. Παράλληλα, η δυνατότητα οπτικού ελέγχου των πολλαπλών αρχείων διαφορετικής μορφής (π.χ. JSON) επιτάχυνε σημαντικά την πειραματική διαδικασία [80].

### 4.1.3 n8n

Το n8n είναι μία σύγχρονη πλατφόρμα οπτικού προγραμματισμού και αυτοματοποίησης ροών εργασίας (workflow automation), η οποία αναπτύχθηκε από γερμανική εταιρεία και κυκλοφόρησε το 2019. Ακολουθεί μία προσέγγιση χαμηλού κώδικα (low-code) και χρησιμοποιεί ένα περιβάλλον βασισμένο σε κόμβους (nodes), όπου κάθε κόμβος αναλαμβάνει την εκτέλεση μιας συγκεκριμένης λειτουργίας. Το γεγονός ότι υιοθετεί το μοντέλο διανομής «fair-code» το κάνει να ξεχωρίζει σε αντίθεση με άλλα

αντίστοιχα εμπορικά μοντέλα. Επιπλέον, προσφέρει τη δυνατότητα τόσο για τοπική εγκατάσταση από τον χρήστη, όσο και ως διαχειριζόμενη υπηρεσία στο cloud. Στην παρούσα εργασία, το n8n (έκδοση 2.12.3) εγκαταστάθηκε σε κεντρικό διακομιστή (server) που διατέθηκε στο πλαίσιο της συνεργασίας με το European School Radio. Η υποδομή αυτή αξιοποιήθηκε κυρίως στο στάδιο επιλογής αρχιτεκτονικής και παραμέτρων των διαφορετικών συστημάτων RAG. Ένα από τα πιο σημαντικά οφέλη της χρήσης του ήταν η άμεση οπτική παρακολούθηση της εκτέλεσης των αιτημάτων σε κάθε επιμέρους κόμβο, με αποτέλεσμα τον εύκολο και γρήγορο εντοπισμό των σφαλμάτων κατά τη φάση της ανάπτυξης του συστήματος [81].

### 4.1.4 Chainlit

Το Chainlit είναι ένα σύγχρονο πλαίσιο ανάπτυξης (framework) ανοιχτού κώδικα βασισμένο στην Python, το οποίο εμφανίστηκε δημόσια το 2023. Σκοπός του είναι η ταχύτατη δημιουργία διεπαφών συνομιλίας (chat interfaces) για εφαρμογές Τεχνητής Νοημοσύνης, χρησιμοποιώντας αποκλειστικά κώδικα Python. Επικεντρώνεται στην διευκόλυνση της αλληλεπίδρασης με τα Μεγάλα Γλωσσικά Μοντέλα και ενσωματώνει πολύπλοκες λειτουργίες, όπως το ιστορικό συνομιλίας και η παροχή έτοιμης διεπαφής χρήστη (UI). Στην ουσία, απαιτείται μόνο η συγγραφή του κώδικα Python και κατευθείαν παρέχει ένα ολοκληρωμένο περιβάλλον για αλληλεπίδραση με τον χρήστη. Για τις ανάγκες της εργασίας, το Chainlit (έκδοση 2.10.1) επιλέχθηκε ως το τελικό περιβάλλον αλληλεπίδρασης του χρήστη με το σύστημα RAG και είναι το εργαλείο με το οποίο πραγματοποιήθηκε η εξαγωγή των τελικών αποτελεσμάτων για τα διαφορετικά πειράματα (βλ. Υποενότητα 4.4.4). Ένας από τους βασικούς λόγους επιλογής του είναι το άμεσο και οικείο περιβάλλον συνομιλίας που προσφέρει. Παράλληλα, το γεγονός ότι υπάρχουν διαθέσιμα έτοιμα κιτ ανάπτυξης λογισμικού (SDKs), κάνει εύκολη και γρήγορη την ενσωμάτωσή του στην ιστοσελίδα του European School Radio, ως τελικό προϊόν [82].

### 4.1.5 Ragas

Το Ragas (Retrieval Augmented Generation Assessment) είναι ένα εξειδικευμένο πλαίσιο (framework) ανοιχτού κώδικα, το οποίο παρουσιάστηκε για πρώτη φορά το 2023 και σχεδιάστηκε αποκλειστικά για την αξιολόγηση συστημάτων RAG. Έχει ως στόχο την αντικατάσταση του παραδοσιακού τρόπου αξιολόγησης, ο οποίος βασίζεται στην απλή λεξιλογική ταύτιση μεταξύ κειμένων, με πιο προηγμένες μετρήσεις ποιότητας που κατανοούν σε βάθος τη σημασιολογική ορθότητα μιας απάντησης. Αυτό το επιτυγχάνει εφαρμόζοντας την καινοτόμο προσέγγιση «LLM-as-a-Judge», η οποία χρησιμοποιεί τα ίδια τα Γλωσσικά Μοντέλα στον ρόλο του αντικειμενικού κριτή. Για την εργασία αξιοποιήθηκε η νεότερη έκδοση του πλαισίου Ragas (0.4), η οποία ενσωματώθηκε ως βιβλιοθήκη της Python και διαδραμάτισε κεντρικό ρόλο στο στάδιο της αξιολόγησης των διαφορετικών συνδυασμών συστημάτων RAG που προέκυψαν κατά την έρευνα. Συγκεκριμένα, με την επιλογή αυτού του εργαλείου έγινε δυνατή η αυτοματοποιημένη και αντικειμενική αξιολόγηση χιλιάδων ερωτήσεων με μέγιστη ταχύτητα και απόδοση, όπως αναλύεται στην Υποενότητα 4.5. Επιπλέον, ένα από τα πιο ισχυρά του σημεία είναι η μεγάλη ποικιλία διαθέσιμων μετρικών, η οποία παρείχε την απαραίτητη ευελιξία για να προσαρμοστεί η διαδικασία αξιολόγησης στις ακριβείς ανάγκες της παρούσας έρευνας [83].

### 4.1.6 OpenAI

Η OpenAI είναι από τις κορυφαίες εταιρείες παγκοσμίως στον τομέα της Τεχνητής Νοημοσύνης, γνωστή σε όλους για την ισχυρή σειρά Μεγάλων Γλωσσικών Μοντέλων, GPT. Η εταιρεία ιδρύθηκε το 2015 στην Αμερική αρχικά ως μη κερδοσκοπικός οργανισμός με στόχο την ερευνητική εξέλιξη. Στο πλαίσιο της παρούσας εργασίας, χρησιμοποιήθηκε η πλατφόρμα της OpenAI (OpenAI Platform), η

οποία επιτρέπει σε προγραμματιστές και επιχειρήσεις να ενσωματώσουν τα σύγχρονα μοντέλα της στα δικά τους συστήματα. Ειδικότερα, αξιοποιήθηκε αποκλειστικά η δυνατότητά της για την έκδοση του απαραίτητου κλειδιού (API key) και την απομακρυσμένη επικοινωνία με τα όλα τα μοντέλα της. Όπως αναφέρεται στην Υποενότητα 4.5.1, χρησιμοποιώντας την επίσημη βιβλιοθήκη της Python (πακέτο *openai*, έκδοση 2.29.0) και σύμφωνα με τις ανάγκες της έρευνας, επιλέχθηκαν συγκεκριμένα ισχυρά μοντέλα της εταιρείας, για να αναλάβουν τον ρόλο του κριτή («LLM-as-a-Judge») και να αξιολογήσουν αντικειμενικά τα πειράματα, μέσω του πλαισίου Ragas. Με αυτόν τον τρόπο, εξασφαλίστηκε η κορυφαία υπολογιστική ευφυΐα, η οποία είναι απαραίτητη για τη σημασιολογική αξιολόγηση σε βάθος, προσφέροντας ταυτόχρονα υψηλή ταχύτητα απόκρισης και σταθερότητα [84].

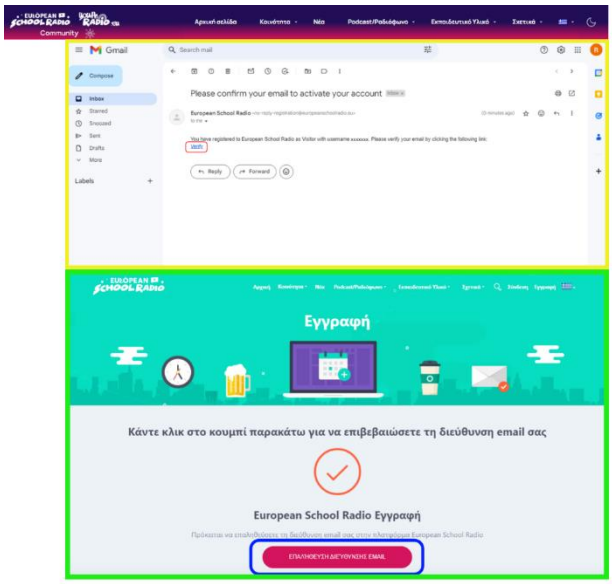
## 4.2 Επεξεργασία και Δόμηση Οδηγού Χρήσης

Η αποδοτικότητα ενός συστήματος RAG εξαρτάται άμεσα από την ποιότητα, τη δομή και την ακρίβεια των δεδομένων με τα οποία το τροφοδοτείς. Στα πλαίσια της παρούσας διπλωματικής εργασίας, ως αρχική πηγή δεδομένων επιλέχθηκε ο επίσημος Οδηγός Χρήσης (User Guide) της εκπαιδευτικής πλατφόρμας European School Radio [85]. Ωστόσο, δεν ήταν εφικτή η απευθείας ενσωμάτωση του υπάρχοντος υλικού στο σύστημα, καθιστώντας αναγκαία μια αναλυτική διαδικασία αναθεώρησης, προεπεξεργασίας και μορφοποίησης, η οποία περιγράφεται στα παρακάτω στάδια.

### 4.2.1 Αναθεώρηση και Ψηφιοποίηση Περιεχομένου

Αρχικά, διαπιστώθηκε ότι ο διαθέσιμος οδηγός χρήσης της πλατφόρμας ήταν παρωχημένος, καθώς δεν είχε ενημερωθεί με βάση τις νέες αλλαγές και προσθήκες που είχαν γίνει πρόσφατα στην ιστοσελίδα. Επομένως, το πρώτο βήμα ήταν η ανανέωση του υλικού, έτσι ώστε ο ψηφιακός βοηθός να αντλεί δεδομένα που ανταποκρίνονται στην τρέχουσα έκδοση της πλατφόρμας.

Παράλληλα, ο αρχικός οδηγός χρήσης σε πολλά σημεία βασιζόταν αποκλειστικά σε στιγμιότυπα οθόνης (screenshots), χωρίς περαιτέρω επεξήγηση με κείμενο, για την ευκολότερη καθοδήγηση των χρηστών στην ιστοσελίδα (π.χ. υποδεικνύοντας οπτικά που βρίσκεται το κουμπί που πρέπει να πατήσει ο χρήστης για να προχωρήσει στο επόμενο βήμα). Γεγονός το οποίο αποτελεί έναν από τους βασικούς περιορισμούς των Γλωσσικών Μοντέλων που μπορούν να επεξεργαστούν μόνο απλό κείμενο. Για να μπορέσει το μοντέλο να καταλάβει αυτές τις οδηγίες που απεικονίζονταν στα στιγμιότυπα οθόνης, έπρεπε να μετατραπούν όλες οι εικόνες χειροκίνητα σε κείμενο (transcription). Έτσι, κάθε εικόνα αντικαταστάθηκε από όσο γίνεται πιο αναλυτικά, αλλά κατανοητά, βήματα για την διαδικασία που πρέπει να ακολουθήσει ο χρήστης, δίνοντας έμφαση στο να μην χαθεί πληροφορία κατά την μετάβαση αυτή, όπως παρουσιάζεται στο Σχήμα 4.1.




(α) Αρχικό Οπτικό Υλικό

## Εγγραφή Επισκέπτη

Για την εγγραφή σας ως Επισκέπτης μπορείτε να επιλέξετε σύνδεση με Google και θα πάρετε αυτόματα τον ρόλο του επισκέπτη. Η άλλη επιλογή είναι να ξεκινήσετε με την επιλογή "Εγγραφή Τώρα!".
Μετά την επιλογή "Εγγραφή Τώρα!", εμφανίζεται η καρτέλα για τη συμπλήρωση των στοιχείων εγγραφής του/της Συντονιστή και της εκπαιδευτικης μονάδας του/της. Στη συνέχεια, πατάτε "Επόμενο".
Είναι απαραίτητο να ανοίξετε το ηλεκτρονικό ταχυδρομείο σας και να κάνετε κλικ στον σύνδεσμο. Για να επιβεβαιώσετε τη διεύθυνση email σας, θα πρέπει να έχετε ανοιχτό το email σας στον ίδιο περιηγητή που χρησιμοποιείτε για την εγγραφή. Για παράδειγμα, εάν κάνετε την εγγραφή σας από τον περιηγητή Chrome, τότε θα πρέπει να επιβεβαιώσετε το email σας στον Chrome. Κάνετε κλικ στο "Verify" (μπλε κείμενο). Επιλέγετε "Επαλήθευση διεύθυνσης email". Στη συνέχεια, εισάγετε το όνομα χρήστη και τον κωδικό σας.
Τώρα είστε έτοιμες/οι να δηλώσετε τα δικά σας προσωπικά στοιχεία. Επιλέγετε τα θέματα των Podcasts και των Ραδιοφωνικών Εκπομπών. Σε περίπτωση που θέλετε να διαγράψετε μία επιλογή, μπορείτε να το κάνετε κάνοντας κλικ στο Χ που έχει δίπλα στο κάθε θέμα. Στη συνέχεια, πατάτε "Υποβολή". Αναμένετε την έγκριση από τον Διαχειριστή ή έναν από τους άλλους χρήστες με τον ρόλο του εκπαιδευτικού για το δικό σας σχολείο.

(β) Νέα Κειμενική Μορφή

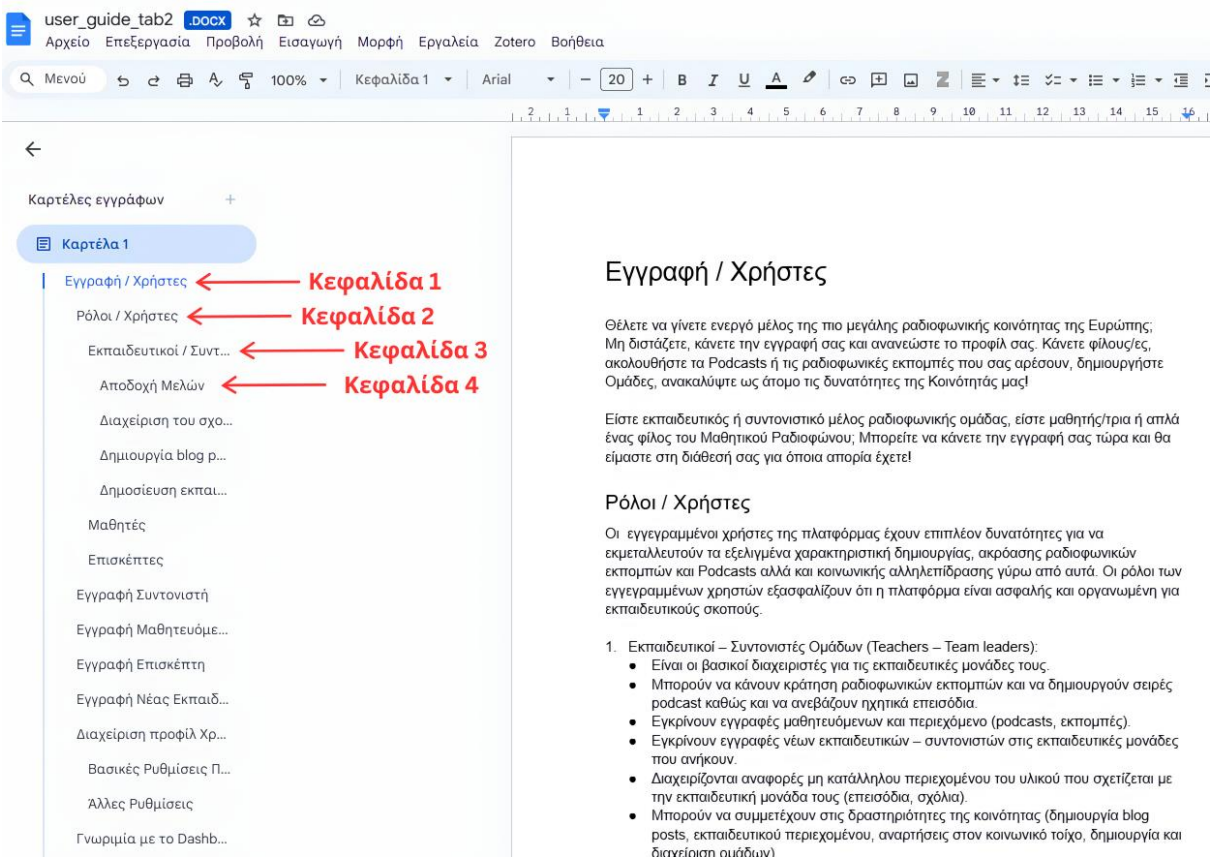
Σχήμα 4.1: Παράδειγμα μετατροπής οπτικής πληροφορίας (screenshot) από τον (α) αρχικό οδηγό χρήσης σε (β) αναλυτικά βήματα μορφής απλού κειμένου

### 4.2.2 Ιεραρχική Οργάνωση του Περιεχομένου

Ο Οδηγός Χρήσης της πλατφόρμας αποτελούνταν από πολλαπλά επίπεδα καρτελών και υπο-καρτελών, το οποίο καθιστούσε αρκετά πολύπλοκη την αρχιτεκτονική του. Για αποδοτικότερη διαχείριση του όγκου των δεδομένων, αποφασίστηκε ο αρχικός διαχωρισμός του σε 4 ξεχωριστά έγγραφα (online Google Έγγραφα) με βάση τις 4 καρτέλες, οι οποίες αντιστοιχούσαν στις κύριες ενότητες πλοήγησης του European School Radio:

1. Καλωσορίσατε στη σελίδα
2. Εγγραφή / Χρήστες
3. Εκπομπές
4. Διαγωνισμοί / Δράσεις

Σε κάθε έγγραφο έγινε χρήση της λειτουργίας των Κεφαλίδων (Κεφαλίδα 1, Κεφαλίδα 2, Κεφαλίδα 3 και Κεφαλίδα 4) για να αποτυπωθεί η αντίστοιχη ιεραρχία που υπήρχε μεταξύ των καρτελών και υπο-καρτελών μέσα στον οδηγό χρήσης (Σχήμα 4.2). Αυτό αποτέλεσε ένα πολύ σημαντικό βήμα για την προετοιμασία των δεδομένων, καθώς αργότερα θα γινόταν μετατροπή αυτών των εγγράφων (.docx) σε αρχεία Markdown (.md), όπου οι κεφαλίδες λειτουργούν ως μεταδεδομένα (metadata) και βοηθούν το σύστημα να κατανοήσει το ευρύτερο θεματικό πλαίσιο (context) κάθε ενότητας.




Σχήμα 4.2: Ιεραρχική δόμηση του περιεχομένου στα έγγραφα προετοιμασίας, με χρήση δομής Κεφαλίδων (Headings) για τη διατήρηση του νοηματικού πλαισίου

### 4.2.3 Κανονικοποίηση Κειμένου

Η κανονικοποίηση του κειμένου υπήρξε ίσως το πιο απαιτητικό στάδιο στην επεξεργασία του οδηγού χρήσης, καθώς έπρεπε να γίνει πολλές φορές ενδελεχής έλεγχος σε ολόκληρη την έκταση του κειμένου αναζητώντας μικρές λεπτομέρειες. Ο λόγος είναι ότι οι αλγόριθμοι τεμαχισμού του κειμένου επηρεάζονται άμεσα από την μορφοποίηση του κειμένου, για παράδειγμα χωρίζουν το κείμενο κάθε φορά που εντοπίζουν τελεία (.), οπότε αν έλειπε σε κάποιο σημείο θα δημιουργούσε πρόβλημα. Ταυτόχρονα, πολύ σημαντικό ρόλο έπαιζε και ο έλεγχος πιθανών σφαλμάτων στο κείμενο, διότι η τροφοδότηση του Ψηφιακού Βοηθού πρέπει να είναι αλάνθαστη, ώστε να διασφαλιστεί πως θα απαντάει με σωστά ελληνικά. Οι ενέργειες που πραγματοποιήθηκαν περιλάμβαναν:

- **Ορθογραφικός και συντακτικός έλεγχος:** Εξάλειψη πιθανών ορθογραφικών και τυπογραφικών λαθών για την αποτροπή παροχής λανθασμένης πληροφορίας στον τελικό χρήστη.
- **Αφαίρεση περιττής μορφοποίησης:** Διαγράφηκαν τα διπλά κενά (spaces) και στοιχεία όπως η έντονη (bold) ή πλάγια (italic) γραφή, τα οποία θα μπορούσαν να εισάγουν «θόρυβο» (noise) κατά τη δημιουργία των διανυσμάτων (embeddings).
- **Κανονικοποίηση στίξης:** Πραγματοποιήθηκε σχολαστικός έλεγχος ώστε κάθε πρόταση να καταλήγει σε τελεία και σε κάθε αλλαγή θεματικής να υπάρχει αλλαγή παραγράφου.
- **Κανονικοποίηση λιστών:** Εφαρμόστηκε κοινή μορφοποίηση σε όλα τα έγγραφα για τις λίστες (bullets) και τις αριθμήσεις. Συγκεκριμένα, όλες οι διαδικασίες που περιλαμβάναν βήματα αποτυπώνονταν με αριθμημένες λίστες της μορφής «1.», «2.» κ.ο.κ. διασφαλίζοντας τη συνοχή του κειμένου και διευκολύνοντας την ανάγνωση και κατανόηση των χρηστών.

## 4.3 Στρατηγική Τεμαχισμού Κειμένου (Chunking)

Η διαδικασία του τεμαχισμού κειμένου αποτελεί έναν από τους πιο κρίσιμους παράγοντες που συμβάλλουν στην αποδοτικότητα ενός συστήματος RAG, καθώς επηρεάζει τον τρόπο με τον οποίο αποθηκεύεται μία πληροφορία, με στόχο, κατά την ανάκτησή της, να τροφοδοτήσει το Γλωσσικό Μοντέλο με τα κατάλληλα δεδομένα για την παραγωγή ορθής απάντησης. Επομένως, είναι αδιαμφισβήτητο ότι έπρεπε να γίνει πολύ προσεκτική επιλογή των τελικών μεθόδων τεμαχισμού. Στις επόμενες υποενότητες αναλύεται σε βάθος ολόκληρη η διαδικασία, από τις αρχικές δοκιμές και το φιλτράρισμα των μεθόδων μέχρι και την τελική επιλογή και εξαγωγή αποτελεσμάτων.

### 4.3.1 Μετατροπή Εγγράφων σε Μορφή Markdown

Εξαιτίας της δομής του Οδηγού Χρήσης με τα επίπεδα καρτελών, θεωρήθηκε σκόπιμο να εφαρμοστούν αλγόριθμοι τεμαχισμού με βάση την ιεραρχία του κειμένου (Markdown Header Splitters). Η μορφή Markdown επιτρέπει στους αλγόριθμους να συγκρατούν ως πληροφορία το «επίπεδο» ιεραρχίας του κειμένου μαζί με τον τίτλο του. Στη συνέχεια, τα στοιχεία αυτά αξιοποιούνται ως μεταδεδομένα κατά τη διαδικασία του τεμαχισμού, εξασφαλίζοντας ένα ευρύτερο νοηματικό πλαίσιο σε κάθε παραγόμενο τμήμα.

Για τον σκοπό αυτό, αναπτύχθηκε ένα script με ονομασία `convert_to_md.py` σε γλώσσα Python, το οποίο μετατρέπει τα αρχεία από μορφή εγγράφου Microsoft Word (.docx) σε μορφή Markdown (.md). Η διαδικασία μετατροπής βασίστηκε στη βιβλιοθήκη *markitdown*, η οποία εγκαταστάθηκε στο περιβάλλον ανάπτυξης μέσω της εντολής τερματικού `pip install markitdown[all]`. Η ιεραρχία αποτυπώνεται χρησιμοποιώντας το σύμβολο της δίεσης (#), δηλαδή η Κεφαλίδα 1 (Heading 1) του Word μετατρέπεται σε #, η Κεφαλίδα 2 σε ##, η Κεφαλίδα 3 σε ### κ.ο.κ.

Αρχικά, ο κώδικας δημιουργεί δυναμικά τον φάκελο προορισμού (*user_guide_converted*) για τα αρχεία Markdown και μέσω μιας επαναληπτικής δομής (loop), μετατρέπει διαδοχικά τα τέσσερα αρχεία του οδηγού χρήσης (*user_guide_tab{i}.docx*, όπου i=1,2,3,4). Όπως φαίνεται στο Σχήμα 4.3, η μετατροπή γίνεται με την συνάρτηση `md.convert()` και το νέο αρχείο (*user_guide_tab{i}_converted.md*, όπου i=1,2,3,4) αποθηκεύεται με κωδικοποίηση χαρακτήρων UTF-8, εξαλείφοντας έτσι την πιθανότητα εσφαλμένης κωδικοποίησης των ελληνικών χαρακτήρων.

```python
# Επεξεργασία των τεσσάρων εγγράφων (tab1 - tab4)
for i in range(1, 5):

    # Φτιάχνουμε τα ονόματα των αρχείων βάζοντας τον αριθμό {i}
    input_file = f"user_guide/user_guide_tab{i}.docx"
    output_file = f"{output_dir}/user_guide_tab{i}_converted.md"

    # Μετατροπή του αρχείου (.docx) σε Markdown
    result = md.convert(input_file)

    # Φτιάχνει ένα νέο αρχείο (.md) και αποθηκεύει το αποτέλεσμα
    with open(output_file, "w", encoding='utf-8') as file:
        file.write(result.text_content)

    print(f"Το αρχείο user_guide_tab{i} μετατράπηκε!")
```

Σχήμα 4.3: Επαναληπτική δομή (loop) του script convert_to_md.py για τη μαζική μετατροπή των αρχείων του Οδηγού Χρήσης σε Markdown (.md)

### 4.3.2 Αρχική Αξιολόγηση και Φιλτράρισμα Μεθόδων Τεμαχισμού

Η επιλογή της κατάλληλης στρατηγικής τεμαχισμού κειμένου (chunking) αποτελεί καθοριστικό παράγοντα για την επιτυχία ενός συστήματος RAG, καθώς επηρεάζει άμεσα την ποιότητα της πληροφορίας που ανακτάται από το Γλωσσικό Μοντέλο. Κάθε κείμενο ανάλογα με το περιεχόμενο και τη δομή του, μπορεί να έχει διαφορετική απόδοση στις διάφορες μεθόδους τεμαχισμού. Επομένως, προκειμένου να εντοπιστεί η βέλτιστη προσέγγιση για το ελληνόγλωσσο περιεχόμενο του Οδηγού Χρήσης του European School Radio, αποφασίστηκε η πειραματική εφαρμογή και συγκριτική αξιολόγηση έντεκα διαφορετικών μεθόδων τεμαχισμού. Οι έντεκα μέθοδοι που εξετάστηκαν ήταν οι εξής:

1. Fixed έως 256 tokens με 32 tokens overlap.
2. Fixed έως 512 tokens με 64 tokens overlap.
3. Recursive έως 256 tokens με 32 tokens overlap.
4. Recursive έως 512 tokens με 64 tokens overlap.
5. Recursive έως 1024 tokens με 128 tokens overlap.
6. Markdown Headers & Recursive έως 256 tokens με 32 tokens overlap.
7. Markdown Headers & Recursive έως 512 tokens με 64 tokens overlap.
8. Markdown Headers & Recursive έως 1024 tokens 128 tokens overlap.
9. Markdown Headers & Semantic με κριτήριο το Percentile = 90
10. Markdown Headers & Semantic με κριτήριο το Standard Deviation = 1.5
11. Markdown Headers & Semantic με κριτήριο το Interquartile Range - IQR = 1.5

Ειδικότερα, σε αυτή την πρώτη φάση της δοκιμής επιλέχθηκε στρατηγικά μόνο το δεύτερο έγγραφο του Οδηγού (Tab 2: Εγγραφή/Χρήστες), το οποίο είναι το πιο εκτενές και ιεραρχικά πολύπλοκο αρχείο από όλα, λειτουργώντας ως σενάριο ακραίας δοκιμής (stress test) για τους αλγόριθμους. Για την εκτέλεση αυτής της δοκιμής αναπτύχθηκε το script `chunking_11_methods.py`. Πραγματοποιήθηκε εγκατάσταση των απαραίτητων βιβλιοθηκών στο περιβάλλον ανάπτυξης μέσω της εντολής τερματικού `pip install langchain pandas tiktoken`. Η βιβλιοθήκη *tiktoken* χρησιμοποιήθηκε με την κωδικοποίηση *cl100k_base* για την απόλυτα ακριβή καταμέτρηση των tokens. Επιπλέον, για τις μεθόδους σημασιολογικού τεμαχισμού (Semantic Chunking), ενσωματώθηκε το τοπικό γλωσσικό μοντέλο `lighteternal/stsb-xlm-r-greek-transfer` μέσω της βιβλιοθήκης *langchain_huggingface*, ώστε το σύστημα να κατανοεί σωστά την ελληνική γλώσσα κατά τον διαχωρισμό των νοημάτων.

Αρχικά, ο κώδικας δημιουργεί δυναμικά δύο φακέλους προορισμού (*chunks_json* και *chunks_excel*) για να εξάγει αντίστοιχα τα αρχεία JSON (για το σύστημα RAG) και τα αρχεία Excel (για διευκόλυνση του ανθρώπινου ελέγχου). Γίνεται αξιοποίηση των αρχείων markdown, μέσω του εργαλείου `MarkdownHeaderTextSplitter()`, το οποίο χρησιμοποιείται στις μεθόδους 6 έως 11. Όπως απεικονίζεται στο Σχήμα 4.4, η μέθοδος αυτή ορίζει ρητά σε ποια σύμβολα του Markdown θα γίνεται η κοπή (π.χ. #, ##) και ονομάζει τα επίπεδα (π.χ. "*Καρτέλα*", "*Υποκαρτέλα*").

```
# --- Κοινό Βήμα Προετοιμασίας για τις Μεθόδους 6 & 7 ---
# Κόβουμε πρώτα το κείμενο δομικά, βάσει των τίτλων του Markdown.
headers_to_split_on = [
    ("#", "Καρτέλα"),
    ("##", "Υποκαρτέλα"),
    ("###", "Ενότητα"),
    ("####", "Υποενότητα")
]
md_splitter = MarkdownHeaderTextSplitter(headers_to_split_on=headers_to_split_on)
md_splits = md_splitter.split_text(text)
```

Σχήμα 4.4: Καθορισμός των επιπέδων ιεραρχίας (Markdown Headers) για τη δυναμική εξαγωγή και διατήρηση των μεταδεδομένων.

Με αυτόν τον τρόπο, η προσέγγιση αυτή έχει ένα τεράστιο πλεονέκτημα, καθώς το σύστημα αποθηκεύει την διαδρομή της ιεραρχίας (π.χ. Εγγραφή/Χρήστες > Ρόλοι/Χρήστες > Μαθητές) ως μεταδεδομένο σε κάθε μεμονωμένο chunk, εξασφαλίζοντας ότι το Γλωσσικό Μοντέλο θα λαμβάνει πάντα το πλήρες θεματικό πλαίσιο ενός κειμένου. Στο τέλος, γίνεται εκτύπωση στο τερματικό όλων των διαδρομών που εντοπίστηκαν στο συγκεκριμένο έγγραφο, όπως φαίνεται στο Σχήμα 4.5.

```
Οι Καρτέλες/Ενότητες που εντόπισε η Μέθοδος Markdown είναι οι εξής:

• Εγγραφή / Χρήστες
• Εγγραφή / Χρήστες > Ρόλοι / Χρήστες
• Εγγραφή / Χρήστες > Ρόλοι / Χρήστες > Εκπαιδευτικοί / Συντονιστές Ομάδων
• Εγγραφή / Χρήστες > Ρόλοι / Χρήστες > Εκπαιδευτικοί / Συντονιστές Ομάδων > Αποδοχή Μελών
• Εγγραφή / Χρήστες > Ρόλοι / Χρήστες > Εκπαιδευτικοί / Συντονιστές Ομάδων > Διαχείριση του σχολείου
• Εγγραφή / Χρήστες > Ρόλοι / Χρήστες > Εκπαιδευτικοί / Συντονιστές Ομάδων > Δημιουργία blog post στο Community
• Εγγραφή / Χρήστες > Ρόλοι / Χρήστες > Εκπαιδευτικοί / Συντονιστές Ομάδων > Δημοσίευση εκπαιδευτικού περιεχομένου H5P
• Εγγραφή / Χρήστες > Ρόλοι / Χρήστες > Μαθητές
• Εγγραφή / Χρήστες > Ρόλοι / Χρήστες > Επισκέπτες
• Εγγραφή / Χρήστες > Εγγραφή Συντονιστή
• Εγγραφή / Χρήστες > Εγγραφή Μαθητευόμενου/ης
• Εγγραφή / Χρήστες > Εγγραφή Επισκέπτη
• Εγγραφή / Χρήστες > Εγγραφή Νέας Εκπαιδευτικής Μονάδας
• Εγγραφή / Χρήστες > Διαχείριση προφίλ Χρήστη > Βασικές Ρυθμίσεις Προφίλ Χρήστη
• Εγγραφή / Χρήστες > Διαχείριση προφίλ Χρήστη > Άλλες Ρυθμίσεις
• Εγγραφή / Χρήστες > Γνωριμία με το Dashboard
• Εγγραφή / Χρήστες > Γνωριμία με το Dashboard > Κεντρική Μπάρα Πλοήγησης
• Εγγραφή / Χρήστες > Γνωριμία με το Dashboard > Προτάσεις για Εσάς (AI)
• Εγγραφή / Χρήστες > Γνωριμία με το Dashboard > Ραδιοφωνικός Player
• Εγγραφή / Χρήστες > Γνωριμία με το Dashboard > Μετάβαση στη Βιβλιοθήκη
• Εγγραφή / Χρήστες > Βιβλιοθήκη (Διαχείριση υλικού χρήστη)
• Εγγραφή / Χρήστες > Βιβλιοθήκη (Διαχείριση υλικού χρήστη) > Ενεργή εκπαιδευτική μονάδα
• Εγγραφή / Χρήστες > Βιβλιοθήκη (Διαχείριση υλικού χρήστη) > Βιβλιοθήκη
```

Σχήμα 4.5: Επιτυχής αναγνώριση της ιεραρχίας του κειμένου, επιβεβαιώνοντας τη διατήρηση του θεματικού πλαισίου (context).

Επιπλέον, γίνεται εκτύπωση στο τερματικό των συγκεντρωτικών στατιστικών αποτελεσμάτων που εξήγαγε το σύστημα για καθεμία από τις έντεκα μεθόδους, δηλαδή το σύνολο τμημάτων κειμένου (chunks), τον μέσο όρο χαρακτήρων και τον μέσο όρο tokens για τα chunks κάθε μεθόδου, όπως φαίνεται στον Πίνακα 4.1.

Πίνακας 4.1: Στατιστική σύνοψη των 11 μεθόδων τεμαχισμού (Εφαρμογή στο Tab 2)

| Μέθοδος | Σύνολο Chunks | Μ.Ο. Χαρακτήρων | Μ.Ο. Tokens |
|---|---|---|---|
| 1. Fixed 256t (32t overlap) | 109 | 214 | 182 |
| 2. Fixed 512t (64t overlap) | 55 | 428 | 363 |
| 3. Recursive 256t (32t overlap) | 132 | 162 | 138 |
| 4. Recursive 512t (64t overlap) | 67 | 318 | 269 |
| 5. Recursive 1024t (128t overlap) | 32 | 665 | 563 |
| 6. Markdown Headers (max 256t) | 119 | 173 | 148 |
| 7. Markdown Headers (max 512t) | 64 | 320 | 271 |
| 8. Markdown Headers (max 1024t) | 34 | 595 | 506 |
| 9. Semantic (Percentile 90%) | 46 | 437 | 371 |
| 10. Semantic (Std Dev 1.5) | 36 | 559 | 475 |
| 11. Semantic (IQR 1.5) | 31 | 649 | 551 |

Με βάση κυρίως τον ενδελεχή ποιοτικό έλεγχο των παραγόμενων αρχείων Excel, προχωρήσαμε στην απόρριψη των πρώτων πέντε μεθόδων (σταθερού μεγέθους και απλές αναδρομικές). Κατά την οπτική επισκόπηση των εξαγόμενων chunks για τις μεθόδους σταθερού μεγέθους, διαπιστώθηκε στην πράξη ότι έκοβαν συχνά προτάσεις στη μέση ή διαχώριζαν βήματα μιας ενιαίας οδηγίας σε διαφορετικά chunks, αλλοιώνοντας εντελώς το νόημα, αφού διαχώριζαν το κείμενο αυστηρά με βάση απόλυτα όρια μεγέθους. Παράλληλα, ο τεμαχισμός του κειμένου με τις αναδρομικές μεθόδους βελτιωνόταν αισθητά όταν λειτουργούσε συνδυαστικά με το εργαλείο Markdown Headers, γι' αυτό και απορρίφθηκαν οι μέθοδοι 3 έως 5.

### 4.3.3 Εφαρμογή των Επιλεγμένων Στρατηγικών Τεμαχισμού

Μετά την επιτυχή ολοκλήρωση της πρώτης φάσης αξιολόγησης και φιλτραρίσματος των μεθόδων τεμαχισμού, οι 6 μέθοδοι που επιλέχθηκαν εφαρμόστηκαν στο σύνολο του Οδηγού Χρήσης (και στα 4 αρχεία/καρτέλες). Για την αυτοματοποίηση αυτής της διαδικασίας, αναπτύχθηκε το script με ονομασία `chunking_6_methods.py`.

Όλες οι επιλεγμένες μέθοδοι είναι υβριδικές, καθώς συνδυάζουν δύο διαφορετικές τεχνικές για το τελικό αποτέλεσμα, έχοντας ως κοινό παρονομαστή τον αρχικό διαχωρισμό του κειμένου βάσει των τίτλων (Markdown Headers), χρησιμοποιώντας την κλάση `MarkdownHeaderTextSplitter` της βιβλιοθήκης *LangChain*. Στη συνέχεια, υποβάλλονται σε έναν δεύτερο κύκλο τεμαχισμού, ο οποίος επιτυγχάνεται με δύο διαφορετικές στρατηγικές:

**A. Αναδρομικές Μέθοδοι (Recursive)**

Χρησιμοποιούν τον αλγόριθμο `RecursiveCharacterTextSplitter`, ο οποίος αναλαμβάνει να «σπάσει» περαιτέρω τα τμήματα του Markdown θέτοντας σε κάθε μέθοδο και ένα διαφορετικό

μέγιστο όριο για τα tokens που θα περιλαμβάνει. Επιπλέον, διασφαλίζει ένα ποσοστό επικάλυψης (overlap) για να μην χάνεται η νοηματική σύνδεση μεταξύ των chunks. Το ποσοστό αυτό ορίστηκε στο 12.5% του μέγιστου μεγέθους του chunk κάθε φορά (δηλαδή το ⅛ του συνολικού μεγέθους). Όπως απεικονίζεται στο Σχήμα 4.6, οι μέθοδοι αυτές δοκιμάστηκαν στο script σε τρία μεγέθη:

1. Μέγιστο μέγεθος 256 tokens με 32 tokens overlap
2. Μέγιστο μέγεθος 512 tokens με 64 tokens overlap
3. Μέγιστο μέγεθος 1024 tokens με 128 tokens overlap

```
# Recursive splitters που είναι απαραίτητα για τις μεθόδους 1, 2 και 3
rec_256 = RecursiveCharacterTextSplitter.from_tiktoken_encoder(chunk_size=256, chunk_overlap=32)
rec_512 = RecursiveCharacterTextSplitter.from_tiktoken_encoder(chunk_size=512, chunk_overlap=64)
rec_1024 = RecursiveCharacterTextSplitter.from_tiktoken_encoder(chunk_size=1024, chunk_overlap=128)
```

Σχήμα 4.6: Αρχικοποίηση των τριών αλγορίθμων RecursiveCharacterTextSplitter με τις αντίστοιχες παραμέτρους μεγέθους τμήματος (chunk_size) και επικάλυψης (chunk_overlap).

**B. Σημασιολογικές Μέθοδοι (Semantic)**

Χρησιμοποιούν τον αλγόριθμο `SemanticChunker`, ο οποίος αξιολογεί το νόημα των προτάσεων και επιλέγει που να «σπάσει» το κείμενο με βάση αυτό. Για την κατανόηση της ελληνικής γλώσσας και τον εντοπισμό αλλαγής θέματος, αξιοποιήθηκε το τοπικό ελληνικό embedding model `lighteternal/stsb-xlm-r-greek-transfer` μέσω *HuggingFace*, όπως φαίνεται στο Σχήμα 4.7.

```
print("\nΦόρτωση τοπικού μοντέλου Embeddings... Παρακαλώ περιμένετε.\n")
hf_embeddings = HuggingFaceEmbeddings(model_name="lighteternal/stsb-xlm-r-greek-transfer")
```

Σχήμα 4.7: Ενσωμάτωση του ελληνικού embedding model

Ο `SemanticChunker` μπορεί να επιλέξει ανάμεσα σε τέσσερα μαθηματικά κριτήρια για να καθορίσει το σημείο αλλαγής νοήματος, τα οποία ορίζονται στην παράμετρο `breakpoint_threshold_type` και είναι:

- το ποσοστημόριο (percentile)
- η τυπική απόκλιση (standard deviation)
- το ενδοτεταρτημοριακό εύρος (interquartile - iqr)
- η κλίση / παράγωγος (gradient)

Επιπλέον, το όριο στο οποίο ορίζεται η αλλαγή για όλες τις παραμέτρους ορίζεται στην παράμετρο `breakpoint_threshold_amount`, όπως φαίνεται στο Σχήμα 4.8. Τελικά, για το σημασιολογικό τεμαχισμό επιλέχθηκε η εξής παραμετροποίηση:

1. Percentile = 90%
2. Standard Deviation = 1.5
3. IQR = 1.5

```
# Αρχικοποίηση Semantic Chunkers (Μέθοδοι 4, 5 και 6)
semantic_percentile = SemanticChunker(hf_embeddings, breakpoint_threshold_type="percentile", breakpoint_threshold_amount=90)
semantic_std = SemanticChunker(hf_embeddings, breakpoint_threshold_type="standard_deviation", breakpoint_threshold_amount=1.5)
semantic_iqr = SemanticChunker(hf_embeddings, breakpoint_threshold_type="interquartile", breakpoint_threshold_amount=1.5)
```

Σχήμα 4.8: Αρχικοποίηση των αλγορίθμων σημασιολογικού τεμαχισμού (Semantic Chunker)

Για την αποδοτικότερη εκτέλεση του κώδικα, χωρίς περιττές επαναλήψεις, δημιουργήθηκε μία δομή λεξικού (`methods_info`), η οποία απεικονίζεται στο Σχήμα 4.9. Η δομή αυτή περιλαμβάνει το όνομα κάθε μεθόδου με την αντίστοιχη συνάρτηση τεμαχισμού, όπως έχει οριστεί παραπάνω, κάνοντας χρήση ανώνυμων συναρτήσεων (`lambda`). Έτσι, υπήρχε η δυνατότητα δυναμικής κλήσης των αλγορίθμων μέσα σε έναν κεντρικό βρόχο επανάληψης (loop) που διατρέχει αυτόματα τα 4 έγγραφα και για τις 6 μεθόδους, αποφεύγοντας την ανάγκη για πολλαπλές εκτελέσεις του κώδικα.

```python
# Λεξικό με όλες τις ενεργές μεθόδους (1 έως 6) για εύκολη διαχείριση στο loop
methods_info = {
    1: {"name": "1_MD_Recursive_256", "splitter": lambda docs: rec_256.split_documents(docs)},
    2: {"name": "2_MD_Recursive_512", "splitter": lambda docs: rec_512.split_documents(docs)},
    3: {"name": "3_MD_Recursive_1024", "splitter": lambda docs: rec_1024.split_documents(docs)},
    4: {"name": "4_MD_Semantic_Percentile_90", "splitter": lambda docs: semantic_percentile.split_documents(docs)},
    5: {"name": "5_MD_Semantic_Std_1_5", "splitter": lambda docs: semantic_std.split_documents(docs)},
    6: {"name": "6_MD_Semantic_IQR_1_5", "splitter": lambda docs: semantic_iqr.split_documents(docs)}
}
```

Σχήμα 4.9: Χρήση δομής λεξικού και ανώνυμων συναρτήσεων (lambda) για τη δυναμική κλήση των έξι αλγορίθμων τεμαχισμού

### 4.3.4 Εξαγωγή Δεδομένων και Στατιστική Ανάλυση

Με την εκτέλεση του κώδικα παράγονται αρχεία JSON τα οποία κατανέμονται αυτόματα σε έξι ξεχωριστούς φακέλους (έναν για κάθε μέθοδο) οι οποίοι φέρουν την ονομασία της αντίστοιχης μεθόδου για λόγους διευκόλυνσης στη μελλοντική τροφοδότηση της Διανυσματικής Βάσης Δεδομένων (Vector Database). Παράλληλα, παράγονται και αρχεία Excel που κατανέμονται δυναμικά στον φάκελο *chunks_excel*. Για τα 4 έγγραφα του Οδηγού Χρήσης και τις 6 μεθόδους τεμαχισμού παράγονται συνολικά 48 αρχεία:

- 24 αρχεία Excel, ένα ανά έγγραφο και μέθοδο
- 24 αρχεία JSON, ένα ανά έγγραφο και μέθοδο

Ιδιαίτερα σημαντική είναι η ορθή δόμηση αυτών των αρχείων. Όπως αποτυπώνεται στο Σχήμα 4.10, διαμορφώθηκε έτσι ώστε να περιλαμβάνει τα τέσσερα βασικά πεδία που είναι απαραίτητα για ένα σύστημα RAG, τα οποία είναι:

- το μοναδικό αναγνωριστικό (chunk_id)
- το δομημένο λεξικό μεταδεδομένων (metadata), το οποίο διασώζει την ιεραρχία των τίτλων
- το πλήθος των tokens (tokens)
- το τμήμα κειμένου (text)


```json
{
    "chunk_id": 28,
    "metadata": {
        "Καρτέλα": "Καλωσήρθατε στην σελίδα",
        "Υποκαρτέλα": "Ραδιόφωνο",
        "Ενότητα": "Επόμενες Εκπομπές (Next Shows)"
    },
    "tokens": 193,
    "text": "εκπομπής στο ραδιόφωνο. Οι ειδοποιήσεις αυτές ε
},
{
    "chunk_id": 29,
    "metadata": {
        "Καρτέλα": "Καλωσήρθατε στην σελίδα",
        "Υποκαρτέλα": "Ακρόαση Επεισοδίου"
    },
    "tokens": 198,
    "text": "Πέραν της ζωντανής ροής του ραδιοφωνικού σταθμο
},
```


Σχήμα 4.10: Παράδειγμα δομής παραγόμενου αρχείου JSON (Chunks_Tab1_1_MD_Recursive_256.json), όπου διακρίνεται η επιτυχής αποθήκευση της ιεραρχίας στο πεδίο των μεταδεδομένων (metadata)

Αντίστοιχα, τα αρχεία Excel που προορίζονται για τον ανθρώπινο έλεγχο, δομήθηκαν σε μορφή πίνακα για να διευκολύνουν την οπτική επισκόπηση, με τις στήλες του να αντικατοπτρίζουν ακριβώς τη δομή του αρχείου JSON (Σχήμα 4.11), δηλαδή:

- *A/A Chunk*: αύξων αριθμός του τμήματος κειμένου
- *Καρτέλα (Metadata)*: το σημείο της ιεραρχίας που βρίσκεται το chunk
- *Αριθμός Tokens*: σύνολο tokens για το συγκεκριμένο chunk
- *Κείμενο*: το κείμενο του συγκεκριμένου chunk

| A/A Chunk | Καρτέλα (Metadata) | Αριθμός Tokens | Κείμενο (Chunk) |
|---|---|---|---|
| 1 | {'Καρτέλα': 'Καλωσήρθατε στην σελίδα', 'Υποκαρτέλα': 'Ραδιόφωνο'} | 195 | Η πλατφόρμα σας δίνει τη δυνατότητα να ακούτε τον ζωντανό ραδιοφωνικό σταθμό του European School Radio από οποιαδήποτε σελίδα εντός της πλατφόρμας του European School Radio και της πλατφόρμας του Youth Radio, χωρίς να χρειάζεται να γίνεται παύση ή διακοπή της αναπαραγωγής |
| 2 | {'Καρτέλα': 'Καλωσήρθατε στην σελίδα', 'Υποκαρτέλα': 'Ραδιόφωνο'} | 44 | ή διακοπή της αναπαραγωγής όταν αλλάζετε σελίδα. |
| 3 | {'Καρτέλα': 'Καλωσήρθατε στην σελίδα', 'Υποκαρτέλα': 'Ραδιόφωνο'} | 184 | Για να γίνει αναπαραγωγή του ζωντανού ραδιοφώνου, ο χρήστης θα πρέπει να πατήσει το κυκλικό κουμπί με το εικονίδιο του ραδιοφώνου, που βρίσκεται στην κάτω δεξιά πλευρά της οθόνης. Το εικονίδιο είναι Floating Action Button, οπότε και |
| 4 | {'Καρτέλα': 'Καλωσήρθατε στην σελίδα', 'Υποκαρτέλα': 'Ραδιόφωνο'} | 78 | είναι Floating Action Button, οπότε και εμφανίζεται σε όλες τις σελίδες της πλατφόρμας, στο ίδιο ύψος πάντα. |
| 5 | {'Καρτέλα': 'Καλωσήρθατε στην σελίδα', 'Υποκαρτέλα': 'Ραδιόφωνο'} | 128 | Πατώντας αυτό το εικονίδιο, ανοίγει ένα μικρό παράθυρο με τη ζωντανή ραδιοφωνική ροή. Το μικρό παράθυρο αυτό ανοίγει ακριβώς δίπλα από το Floating Action Button. |
| 6 | {'Καρτέλα': 'Καλωσήρθατε στην σελίδα', 'Υποκαρτέλα': 'Ραδιόφωνο'} | 208 | Η αναπαραγωγή της ζωντανής ραδιοφωνικής μετάδοσης ξεκινάει κατευθείαν με το άνοιγμα του μικρού παραθύρου. Φυσικά υπάρχει ένας μικρός χρόνος κατα τον οποίο γίνεται η φόρτωση της ραδιοφωνικής ροής, πριν ξεκινήσει η αναπαραγωγή. |

Σχήμα 4.11: Παράδειγμα εξαγόμενου αρχείου Excel (Chunks_Tab1_1_MD_Recursive_256.xlsx) για την ποιοτική αξιολόγηση των παραγόμενων τμημάτων κειμένου (chunks)

Στη συνέχεια, το script υπολογίζει τα αναλυτικά στατιστικά αποτελέσματα ξεχωριστά για κάθε συνδυασμό εγγράφου και μεθόδου τεμαχισμού (δηλαδή 24 συνδυασμοί) και τα τυπώνει στο τερματικό. Στον Πίνακα 4.2 παρουσιάζεται η αναλυτική καταγραφή των μετρικών (*Σύνολο Chunks*, *Μέσος Όρος Χαρακτήρων*, *Μέσος Όρος Tokens*) ανά Μέθοδο Chunking (*Μέθοδος*) και ανά έγγραφο (*Tab*).

Πίνακας 4.2: Αναλυτικά στατιστικά τεμαχισμού, ταξινομημένα ανά Μέθοδο και έγγραφο (Tab)

| **Tab** | **Μέθοδος** | **Σύνολο Chunks** | **Μ.Ο. Χαρακτήρων** | **Μ.Ο. Tokens** |
|---|---|---|---|---|
| Tab 1 | 1_MD_Recursive_256 | 108 | 166 | 141 |
| Tab 2 | 1_MD_Recursive_256 | 124 | 171 | 146 |
| Tab 3 | 1_MD_Recursive_256 | 170 | 176 | 149 |
| Tab 4 | 1_MD_Recursive_256 | 114 | 173 | 150 |
| Tab 1 | 2_MD_Recursive_512 | 54 | 319 | 271 |
| Tab 2 | 2_MD_Recursive_512 | 66 | 317 | 270 |
| Tab 3 | 2_MD_Recursive_512 | 83 | 358 | 302 |
| Tab 4 | 2_MD_Recursive_512 | 60 | 322 | 278 |
| Tab 1 | 3_MD_Recursive_1024 | 27 | 634 | 539 |

| | | | | |
|---|---|---|---|---|
| Tab 2 | 3_MD_Recursive_1024 | 34 | 609 | 519 |
| Tab 3 | 3_MD_Recursive_1024 | 42 | 716 | 602 |
| Tab 4 | 3_MD_Recursive_1024 | 27 | 705 | 608 |
| Tab 1 | 4_MD_Semantic_Percentile_90 | 30 | 564 | 480 |
| Tab 2 | 4_MD_Semantic_Percentile_90 | 45 | 457 | 390 |
| Tab 3 | 4_MD_Semantic_Percentile_90 | 48 | 606 | 512 |
| Tab 4 | 4_MD_Semantic_Percentile_90 | 26 | 720 | 622 |
| Tab 1 | 5_MD_Semantic_Std_1_5 | 22 | 769 | 654 |
| Tab 2 | 5_MD_Semantic_Std_1_5 | 35 | 588 | 502 |
| Tab 3 | 5_MD_Semantic_Std_1_5 | 41 | 710 | 599 |
| Tab 4 | 5_MD_Semantic_Std_1_5 | 24 | 780 | 674 |
| Tab 1 | 6_MD_Semantic_IQR_1_5 | 19 | 891 | 757 |
| Tab 2 | 6_MD_Semantic_IQR_1_5 | 30 | 686 | 585 |
| Tab 3 | 6_MD_Semantic_IQR_1_5 | 34 | 856 | 722 |
| Tab 4 | 6_MD_Semantic_IQR_1_5 | 21 | 892 | 770 |

Ωστόσο, για να εξαχθεί ένα τελικό, ασφαλές συμπέρασμα για τη συνολική συμπεριφορά των αλγορίθμων τεμαχισμού, κρίνεται απαραίτητη η ομαδοποίηση των δεδομένων με βάση τον αλγόριθμο. Επομένως, με βάση αυτά τα στοιχεία θα γίνει και η επιλογή της βέλτιστης παραμετροποίησης του συστήματος στο επόμενο στάδιο. Στον Πίνακα 4.3 παρατίθενται τα συγκεντρωτικά στατιστικά αποτελέσματα για ολόκληρο τον Οδηγό Χρήσης, τα οποία στο τέλος της εκτέλεσης του script τυπώνονται στο τερματικό.

Πίνακας 4.3: Συγκεντρωτικά στατιστικά μεγέθους και πλήθους τμημάτων για ολόκληρο τον Οδηγό Χρήσης (Tabs 1 έως 4)

| **Μέθοδος** | **Σύνολο Chunks** | **Μ.Ο. Χαρακτήρων** | **Μ.Ο. Tokens** |
|---|---|---|---|
| 1_MD_Recursive_256 | 516 | 172 | 147 |
| 2_MD_Recursive_512 | 263 | 331 | 282 |
| 3_MD_Recursive_1024 | 130 | 669 | 568 |
| 4_MD_Semantic_Percentile_90 | 149 | 572 | 488 |
| 5_MD_Semantic_Std_1_5 | 122 | 699 | 596 |
| 6_MD_Semantic_IQR_1_5 | 104 | 821 | 699 |

## 4.4 Πειραματικός Σχεδιασμός και Εκτέλεση

Έχοντας ολοκληρώσει την προετοιμασία των δεδομένων και τον τεμαχισμό τους, το επόμενο κρίσιμο στάδιο είναι ο σχεδιασμός του πειράματος. Είναι πολύ σημαντικό να υπάρχουν οι κατάλληλες προϋποθέσεις και η δομή της έρευνας να είναι σαφής για την αποφυγή σφαλμάτων. Επομένως, έγινε προσεκτική επιλογή των μοντέλων που θα αξιοποιηθούν, καθώς και η απαραίτητη παραμετροποίηση τους στο περιβάλλον n8n. Έπειτα, αναπτύχθηκε το Σύνολο Δεδομένων Αξιολόγησης (Golden Dataset), προκειμένου να χρησιμοποιηθεί ως μέτρο σύγκρισης και να εξασφαλιστεί η αντικειμενική αξιολόγηση. Τέλος, μέσω του εργαλείου Chainlit, πραγματοποιήθηκε η εκτέλεση όλων των πειραμάτων και η εξαγωγή των αποτελεσμάτων, τα οποία είναι έτοιμα για αξιολόγηση. Όλα αυτά τα βήματα της διαδικασίας περιγράφονται αναλυτικά στις επόμενες υποενότητες.

### 4.4.1 Αρχιτεκτονική Πειράματος και Επιλογή Μοντέλων

Για την ολοκληρωμένη αξιολόγηση του συστήματος RAG, σχεδιάστηκε μια εκτεταμένη πειραματική δομή, η οποία εξετάζει την επιρροή όλων των κρίσιμων τεχνολογιών που συνιστούν ένα τέτοιο σύστημα. Στόχος του πειράματος είναι να μελετηθεί πώς οι διάφοροι συνδυασμοί μεταξύ διαφορετικών Γλωσσικών Μοντέλων (LLMs και SLM), μοντέλων διανυσματικών αναπαραστάσεων (Embedding Models) και στρατηγικών τεμαχισμού (Chunking Methods) επηρεάζουν την τελική ποιότητα της ανάκτησης και της παραγωγής απαντήσεων και ποιος τελικά θα είναι ο καλύτερος συνδυασμός για το σύστημα RAG.

Η τελική επιλογή των μοντέλων πραγματοποιήθηκε με βάση τη βιβλιογραφική ανασκόπηση και προηγούμενη έρευνα, τα οποία αναλύθηκαν στο Κεφάλαιο 3, συνδυαστικά με τη συμβατότητα και τα εργαλεία που αξιοποιούνται ήδη στην υφιστάμενη υποδομή του European School Radio. Έτσι ώστε, να διασφαλιστεί ότι τα αποτελέσματα της παρούσας διπλωματικής εργασίας θα μπορούν να έχουν άμεση πρακτική εφαρμογή στο πραγματικό περιβάλλον της πλατφόρμας.

Οι ανεξάρτητες μεταβλητές του πειράματος, καθώς και οι αντίστοιχες κωδικές τους ονομασίες, οι οποίες δημιουργήθηκαν αποκλειστικά για την τυποποίηση της ονοματολογίας των εξαγόμενων αρχείων και χρησιμοποιούνται για διευκόλυνση καθόλη την διάρκεια της έρευνας, είναι οι εξής:

- **Γλωσσικά Μοντέλα (LLMs και SLM):** Επιλέχθηκαν 4 μοντέλα με διαφορετικές αρχιτεκτονικές και πλήθος παραμέτρων για να αξιολογηθεί η ικανότητα κατανόησης και παραγωγής κειμένου. Τα μοντέλα αυτά παρατίθενται στον Πίνακα 4.4.

- **Μοντέλα Διανυσματικής Αναπαράστασης (Embeddings):** Επιλέχθηκαν 3 μοντέλα με κύριο γνώμονα την υποστήριξη της ελληνικής γλώσσας, είτε είναι πολυγλωσσικά, είτε εξειδικευμένα μόνο στα ελληνικά, όπως παρουσιάζονται στον Πίνακα 4.5.

- **Μέθοδοι Τεμαχισμού (Chunking):** Εφαρμόστηκαν οι 6 υβριδικές μέθοδοι (3 Αναδρομικές και 3 Σημασιολογικές) που εξετάστηκαν αναλυτικά στο προηγούμενο στάδιο της έρευνας (Πίνακας 4.6).

- **Διανυσματική Βάση Δεδομένων:** Έγινε επιλογή της Qdrant, η οποία είναι έτοιμη, ανοιχτής πρόσβασης (open source) Διανυσματική Βάση Δεδομένων και έχει πάρα πολλά πλεονεκτήματα να προσφέρει. Ορισμένα από αυτά είναι ότι έχει εξαιρετικά υψηλή ταχύτητα απόκρισης, δεν καταναλώνει πολύ μνήμη και είναι «αυτοφιλοξενούμενη» (self-hosted). Επίσης, έχει τη δυνατότητα φιλτραρίσματος και μέσω ωφέλιμου φορτίου (payload), το οποίο σημαίνει ότι έχει την επιλογή να κάνει ερωτήματα (queries), είτε μόνο με διάνυσμα (vector), είτε με vector και

payload, είτε μόνο με payload. Επιπλέον, υποστηρίζει διαμερισμό (shrading) των συλλογών (collections) της, δηλαδή μπορεί να τις «σπάει» για καλύτερη ταχύτητα αναζήτησης.

Πίνακας 4.4: Τα επιλεγμένα Γλωσσικά Μοντέλα και η αντίστοιχη κωδική ονομασία τους στο πείραμα

| **Γλωσσικά Μοντέλα** | **Κωδική Ονομασία** |
|---|---|
| llama3.1:8b | llama |
| gemma4:26b | gemma |
| gemini 2.5 flash | gemini |
| deepseek-v4-flash | deepseek |

Πίνακας 4.5: Τα επιλεγμένα Embeddings Models και η αντίστοιχη κωδική ονομασία τους στο πείραμα

| **Μοντέλο Embedding** | **Κωδική Ονομασία** |
|---|---|
| paraphrase-multilingual-MiniLM-L12-v2 | multilingual |
| stsb-xlm-r-greek-transfer | greek |
| multilingual-e5-large | e5 |

Πίνακας 4.6: Οι επιλεγμένες μέθοδοι τεμαχισμού κειμένου και η αντίστοιχη κωδική ονομασία τους

| **Μέθοδος Chunking** | **Κωδική Ονομασία** |
|---|---|
| Markdown Headers & Recursive (256 / overlap 32) | re_256 |
| Markdown Headers & Recursive (512 / overlap 64) | re_512 |
| Markdown Headers & Recursive (1024 / overlap 128) | re_1024 |
| Markdown Headers & Semantic (Percentile = 90) | se_percentile |
| Markdown Headers & Semantic (Std = 1.5) | se_std |
| Markdown Headers & Semantic (IQR = 1.5) | se_iqr |

Λαμβάνοντας υπόψη τον συνδυασμό όλων των παραπάνω μεταβλητών, ο συνολικός αριθμός των πειραμάτων που σχεδιάστηκαν και εκτελέστηκαν υπολογίζεται ως:

$$Συνολικά\ Πειράματα\ =\ 4\ (LM)\ \times\ 3\ (Embeddings)\ \times\ 6\ (Chunking\ Methods)\ =\ 72\ πειράματα$$

Λόγω του τεράστιου πλήθους των αρχείων που θα προέκυπτε και της πολυπλοκότητας των ονομασιών για την αποφυγή σφαλμάτων ορίστηκε μια τυποποιημένη κωδική ονοματολογία για τα εξαγόμενα

αρχεία αποτελεσμάτων η οποία προκύπτει από τις κωδικές ονομασίες των παραπάνω πινάκων. Κάθε πείραμα παράγει ένα μοναδικό αρχείο δομής JSON, το όνομα του οποίου ακολουθεί το πρότυπο:

*{Μέθοδος Chunking}_{Γλωσσικό Μοντέλο}_{Μοντέλο Embedding}.json*

Ενδεικτικά παραδείγματα της παραπάνω ονοματολογίας είναι τα εξής:

- *re_512_gemini_multilingual.json*
- *re_256_deepseek_greek.json*
- *se_percentile_llama_e5.json*
- *se_iqr_gemma_greek.json*

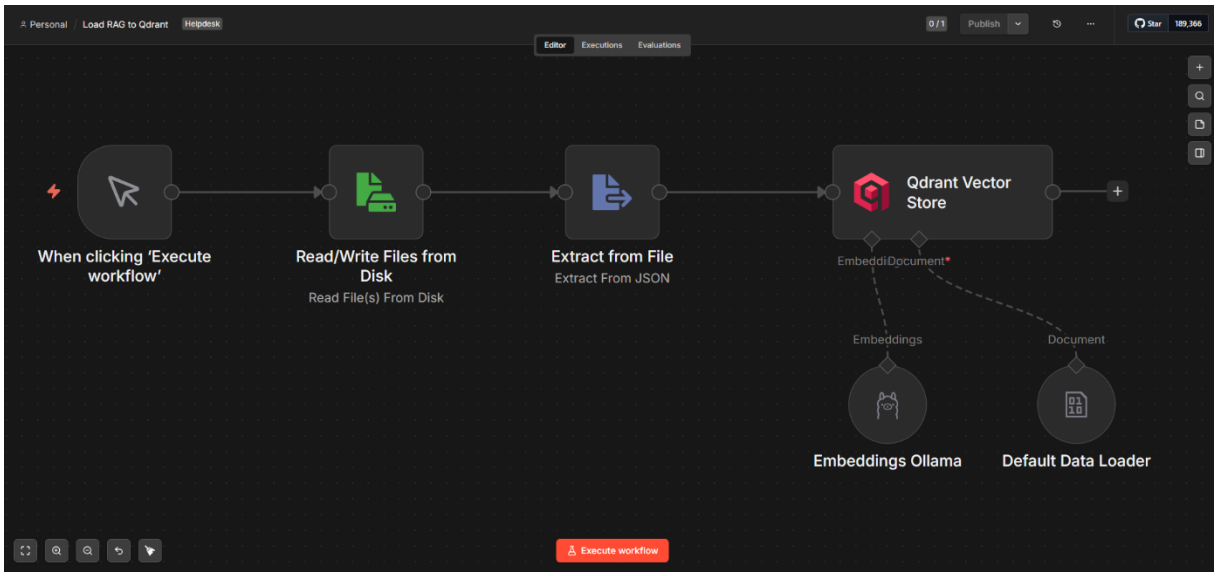
### 4.4.2 Αρχιτεκτονική Συστήματος RAG (n8n)

Οι δοκιμές πάνω στις βασικές ρυθμίσεις που απαρτίζουν ένα σύστημα RAG, η οριστικοποίηση τους, αλλά και η υλοποίηση των πειραμάτων, έγιναν στην πλατφόρμα αυτοματοποίησης n8n. Στο πλαίσιο αυτό, σχεδιάστηκαν δύο ροές εργασίας (workflows) που αναλαμβάνουν ολόκληρη την διαδικασία του συστήματος RAG: από την προετοιμασία των δεδομένων έως την ζωντανή αλληλεπίδραση του χρήστη με το Chatbot και την τελική παραγωγή απαντήσεων.

#### 4.4.2.1 Ροή Προετοιμασίας Δεδομένων

Η πρώτη ροή εργασίας (*Load RAG to Qdrant*), όπως απεικονίζεται στο Σχήμα 4.12, δημιουργήθηκε με σκοπό την τροφοδότηση της Διανυσματικής Βάσης Qdrant. Σκοπός της είναι να παίρνει τα τμήματα κειμένου (chunks) που αναπτύχθηκαν σε προηγούμενο στάδιο σε μορφή JSON, να τα μετατρέπει σε διανύσματα (embeddings) και να τα αποθηκεύει. Η διαδικασία εκτελείται γραμμικά μέσα από την εξής σειρά κόμβων:

1. **Εκτέλεση Ροής** (*When clicking "Execute workflow"*): Η ροή εκκινεί χειροκίνητα όταν πατηθεί το κουμπί "*Execute Workflow*" στο κάτω μέρος της οθόνης.
2. **Ανάγνωση Αρχείων** (*Read/Write Files from Disk*): Αναλαμβάνει την ανάγνωση των δεδομένων δίνοντας του την διαδρομή μέσα στο σύστημα αρχείων του εκάστοτε αρχείου. Τα αρχεία αυτά περιλαμβάνουν τα τμήματα κειμένου για όλα τα έγγραφα του Οδηγού Χρήσης, ανά διαφορετική μέθοδο τεμαχισμού. Ειδικότερα, είναι τα 24 αρχεία JSON που δημιουργήθηκαν στο τελικό στάδιο τεμαχισμού του κειμένου (βλ. Υποενότητα 4.3.4).
3. **Εξαγωγή Δεδομένων** (*Extract from File*): Παρόλο που τα αρχεία είναι ήδη δομημένα ως JSON, το n8n κατά την ανάγνωσή τους, στον προηγούμενο κόμβο, τα αντιμετωπίζει αρχικά ως ακατέργαστα δυαδικά (binary) δεδομένα. Ο κόμβος αυτός αναλαμβάνει την εξαγωγή τους, μετατρέποντας το περιεχόμενό τους σε μορφή μεταβλητών που το n8n μπορεί να αναγνωρίσει.
4. **Μηχανισμός Φόρτωσης** (*Default Data Loader*): Παίρνει τα εξαγόμενα δεδομένα και τα μετατρέπει σε μορφή Εγγράφου (Document), το οποίο είναι μία συγκεκριμένη δομή που αναγνωρίζει η Qdrant.
5. **Παραγωγή Διανυσμάτων** (*Embeddings Ollama*): Το μοντέλο διανυσματικών αναπαραστάσεων αναλαμβάνει τη σημασιολογική αναπαράσταση των κειμένων, δηλαδή την μετατροπή τους σε αριθμούς (embeddings), αποτυπώνοντας το νόημα της πληροφορίας.
6. **Αποθήκευση στη Βάση** (*Qdrant Vector Store*): Η βάση Qdrant αξιοποιεί το Έγγραφο (Document) και τις παραγόμενες διανυσματικές αναπαραστάσεις (embeddings), προκειμένου να αποθηκεύσει τα τμήματα κειμένου (chunks). Έτσι, επιτυγχάνεται η σημασιολογική αποθήκευση αυτών, που θα χρησιμεύσει στην επόμενη ροή εργασίας.

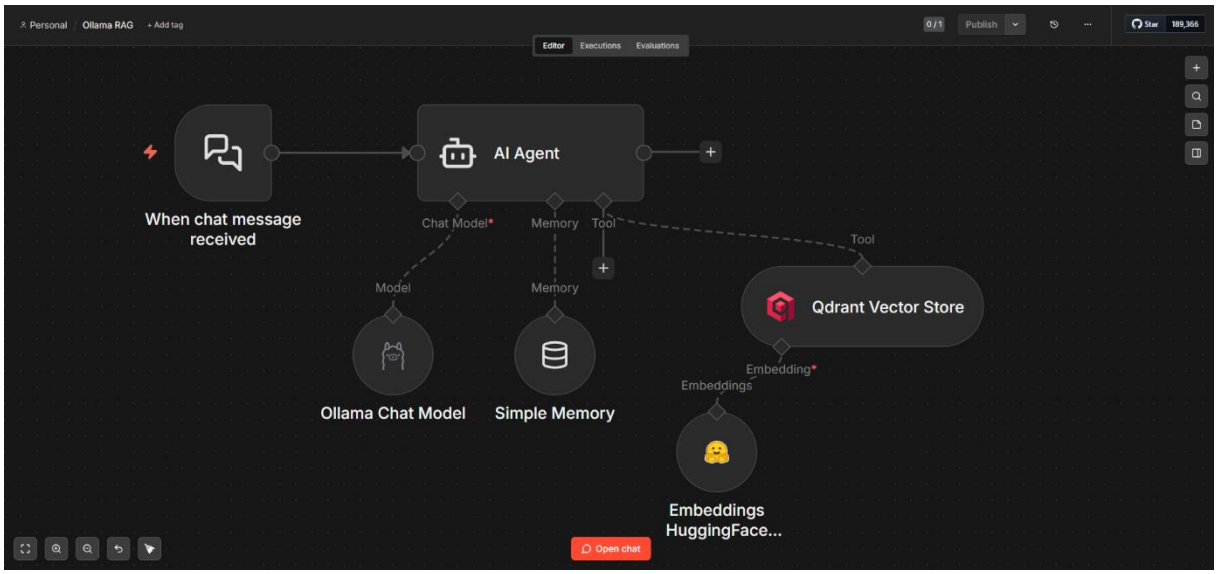



Σχήμα 4.12: Η ροή εργασίας στο n8n για την ενσωμάτωση των εγγράφων στη διανυσματική βάση Qdrant (Load RAG to Qdrant)

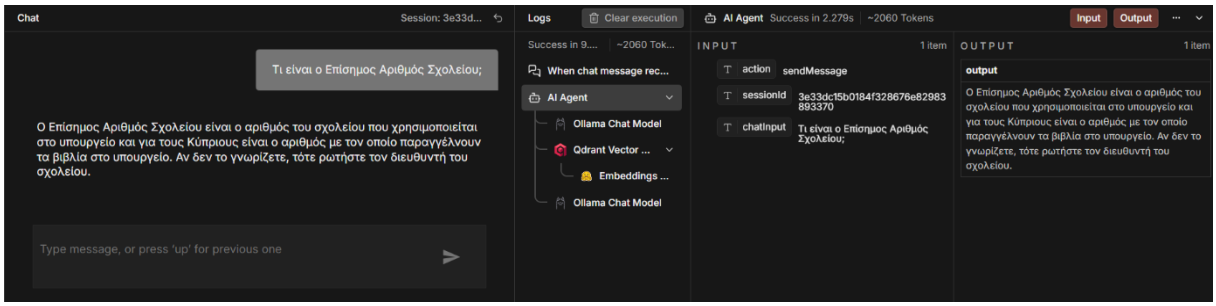
#### 4.4.2.2 Ροή Ανάκτησης και Παραγωγής

Η δεύτερη ροή εργασίας (*Ollama RAG*), όπως απεικονίζεται στο Σχήμα 4.13, είναι ο πυρήνας του συστήματος RAG, καθώς αφορά την αλληλεπίδραση με τον χρήστη. Σκοπός της είναι να δέχεται το ερώτημα του χρήστη, να βρίσκει την πιο σχετική πληροφορία στη βάση δεδομένων και με αυτή, να απαντάει κατάλληλα. Η αρχιτεκτονική της βασίζεται στους εξής κόμβους:

1. **Έναρξη συνομιλίας** (*When chat message received*): Η ροή ξεκινάει απευθείας τη στιγμή που ο χρήστης υποβάλλει το ερώτημα του στο chat που φαίνεται στο κάτω αριστερά μέρος της οθόνης (Σχήμα 4.14).
2. **Γλωσσικό Μοντέλο** *(Ollama Chat Model)*: Το Γλωσσικό Μοντέλο, μέσω της πλατφόρμας Ollama, που αναλαμβάνει να συντάξει την τελική απάντηση.
3. **Διαχείριση Μνήμης** *(Simple Memory)*: Το Chatbot χρησιμοποιεί την μνήμη για να διατηρεί το πλαίσιο της συζήτησης. Συγκεκριμένα ρυθμίζεται να κρατάει τον Ν αριθμό τελευταίων μηνυμάτων. Επομένως, αν ο χρήστης κάνει κάποια συμπληρωματική ερώτηση (follow-up), το Γλωσσικό Μοντέλο θα μπορεί να κατανοήσει πλήρως το πλαίσιο (context) της αναφοράς.
4. **Παραγωγή Διανυσμάτων** (*Embeddings HuggingFace Inference*): Το μοντέλο διανυσματικών αναπαραστάσεων της πλατφόρμας Hugging Face, αναλαμβάνει τη σημασιολογική αναπαράσταση της ερώτησης του χρήστη, δηλαδή την μετατροπή της σε αριθμούς (embeddings), αποτυπώνοντας το νόημα της πληροφορίας.
5. **Εργαλείο Ανάκτησης** *(Qdrant Vector Store Tool)*: Είναι η Διανυσματική Βάση Δεδομένων που δημιουργήθηκε στο προηγούμενο βήμα (*Load RAG to Qdrant*). Συγκρίνει το νόημα της ερώτησης με τα αποθηκευμένα τμήματα κειμένου (chunks) και εντοπίζει αυτά με την μεγαλύτερη νοηματική ομοιότητα μεταξύ τους. Αφού ανακτήσει όσα κείμενα του έχουμε ορίσει, τα χρησιμοποιεί ως πληροφορία για να δομήσει την τελική απάντηση.
6. **Πράκτορας Τεχνητής Νοημοσύνης** (*AI Agent*): Λειτουργεί ως ο «εγκέφαλος» της ροής, καθώς παίρνει την ερώτηση του χρήστη και αποφασίζει τι θα κάνει, αξιοποιώντας τα εργαλεία που έχει συνδεδεμένα (δηλαδή το Γλωσσικό Μοντέλο, τη Μνήμη και τη Βάση Δεδομένων).



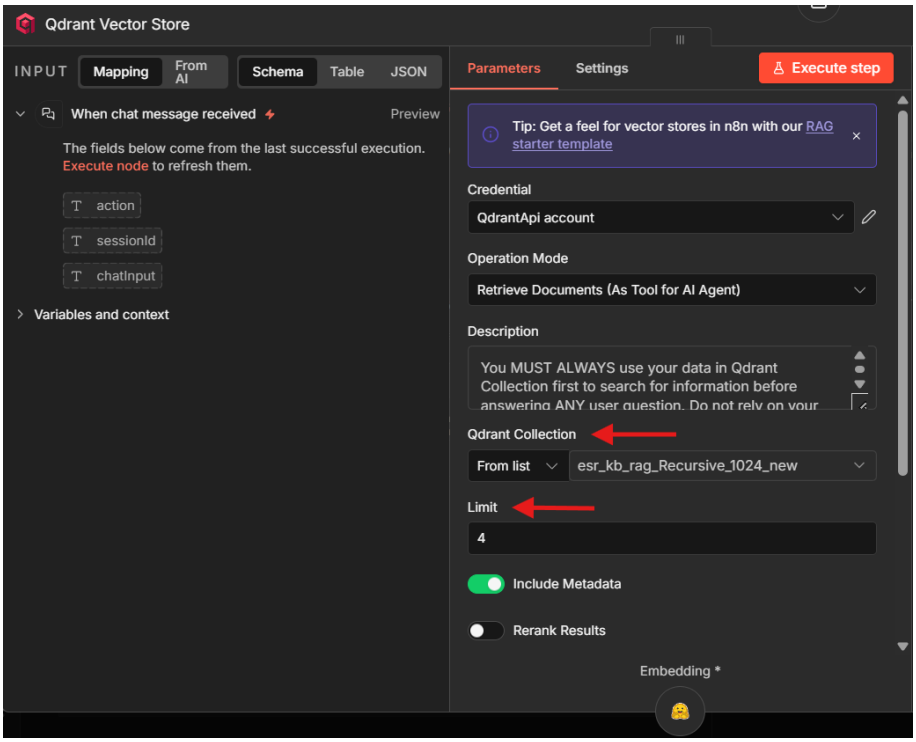

Σχήμα 4.13: Η ροή εργασίας στο n8n για τη διαχείριση ερωτημάτων (AI Agent) και την ανάκτηση πληροφοριών από το Qdrant (Ollama RAG)




Σχήμα 4.14: Διεπαφή αλληλεπίδρασης χρήστη κατά την επιτυχή εκτέλεση του συστήματος RAG

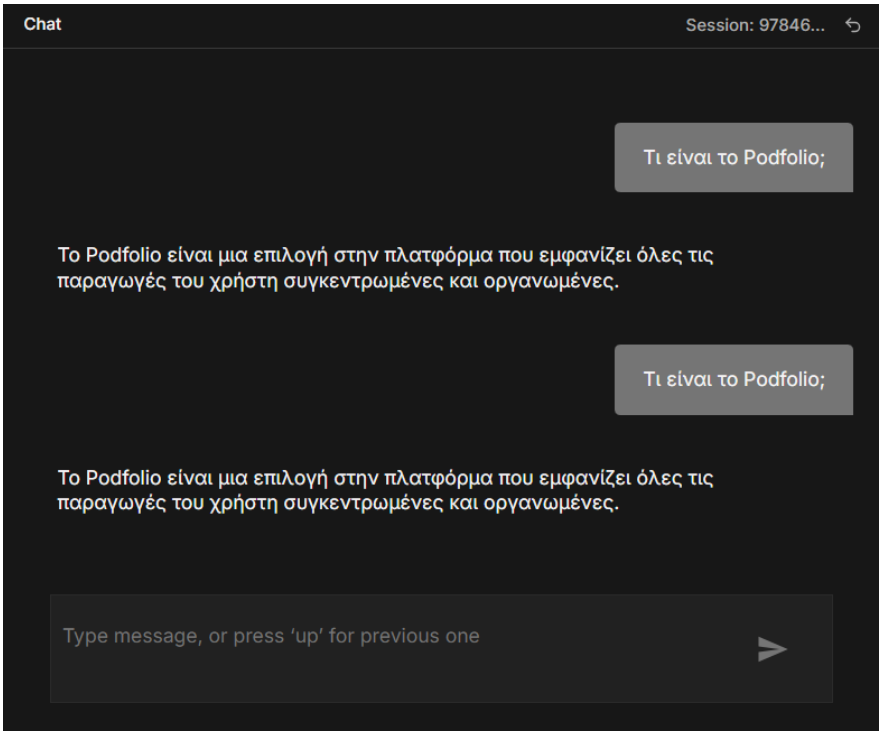
Το σύστημα έχει πολλές διαφορετικές παραμέτρους και ρυθμίσεις, οι οποίες επηρεάζουν άμεσα την ορθότητα και την ποιότητα της απάντησης που δίνει το Γλωσσικό Μοντέλο. Έπειτα από πολλές δοκιμές με στόχο την εύρεση ισορροπίας κυρίως μεταξύ ακρίβειας και πιστότητας στον Οδηγό Χρήσης, προέκυψε η επιλογή των τελικών ρυθμίσεων. Οι τελικές ρυθμίσεις παρουσιάζονται συνοπτικά στον Πίνακα 4.7 και αναλύονται παρακάτω:

- **Όριο Ανάκτησης (Limit):** Για την αντικειμενική αξιολόγηση των διαφορετικών μεθόδων τεμαχισμού, ήταν καθοριστικής σημασίας η αναλογική προσαρμογή της παραμέτρου *Limit* που ορίζει τον αριθμό των ανακτώμενων τμημάτων κειμένου (chunks) από τη βάση δεδομένων. Η παράμετρος αυτή προσαρμόστηκε αντιστρόφως ανάλογα με το μέσο όρο tokens που έχουν τα chunks για την κάθε μέθοδο, έτσι ώστε το Γλωσσικό Μοντέλο να τροφοδοτείται με την ίδια ποσότητα πληροφορίας για κάθε περίπτωση. Όπως απεικονίζεται στο Σχήμα 4.15, ανάλογα με την επιλεγμένη μέθοδο στο πεδίο *Qdrant Collection*, προσαρμόζεται δυναμικά και η τιμή στο πεδίο *Limit*. Με την βοήθεια του Πίνακα 4.3, οι υπολογισμοί έγιναν ως εξής:
    - Recursive 256: Limit = 16 (Προκύπτει από: $147 \times 16 = 2352\ tokens$ )
    - Recursive 512: Limit = 8 (Προκύπτει από: $282 \times 8 = 2256\ tokens$ )
    - Recursive 1024: Limit = 4 (Προκύπτει από: $568 \times 4 = 2272\ tokens$ )
    - Semantic Percentile 90: Limit = 5 (Προκύπτει από: $488 \times 5 = 2440\ tokens$ )
    - Semantic Std 1.5: Limit = 4 (Προκύπτει από: $596 \times 4 = 2384\ tokens$ )
    - Semantic IQR 1.5: Limit = 4 (Προκύπτει από: $699 \times 4 = 2796\ tokens$ )



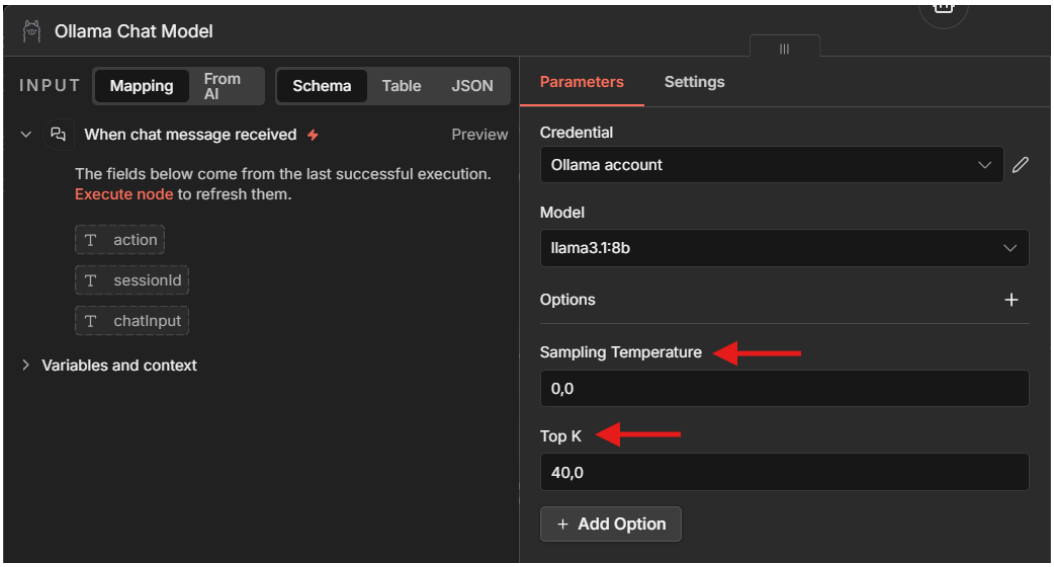

Σχήμα 4.15: Παραμετροποίηση της συλλογής και του ορίου ανάκτησης (Limit) στον κόμβο Qdrant Vector Store

- **Θερμοκρασία Δειγματοληψίας (Sampling Temperature):** Η θερμοκρασία δειγματοληψίας καθορίζει τον βαθμό δημιουργικότητας και τυχαιότητας στο κείμενο που παράγει το Γλωσσικό Μοντέλο. Ορίστηκε αυστηρά στο μηδέν, καθώς ήταν αναγκαίο να στηρίζεται αποκλειστικά στον Οδηγό Χρήσης για να συντάξει την απάντηση προς τον χρήστη. Επιπλέον, παρατηρήθηκε ότι με αυτό τον τρόπο οι απαντήσεις ήταν πάντα σταθερές σε ίδιες ερωτήσεις, όπως φαίνεται στο Σχήμα 4.16.



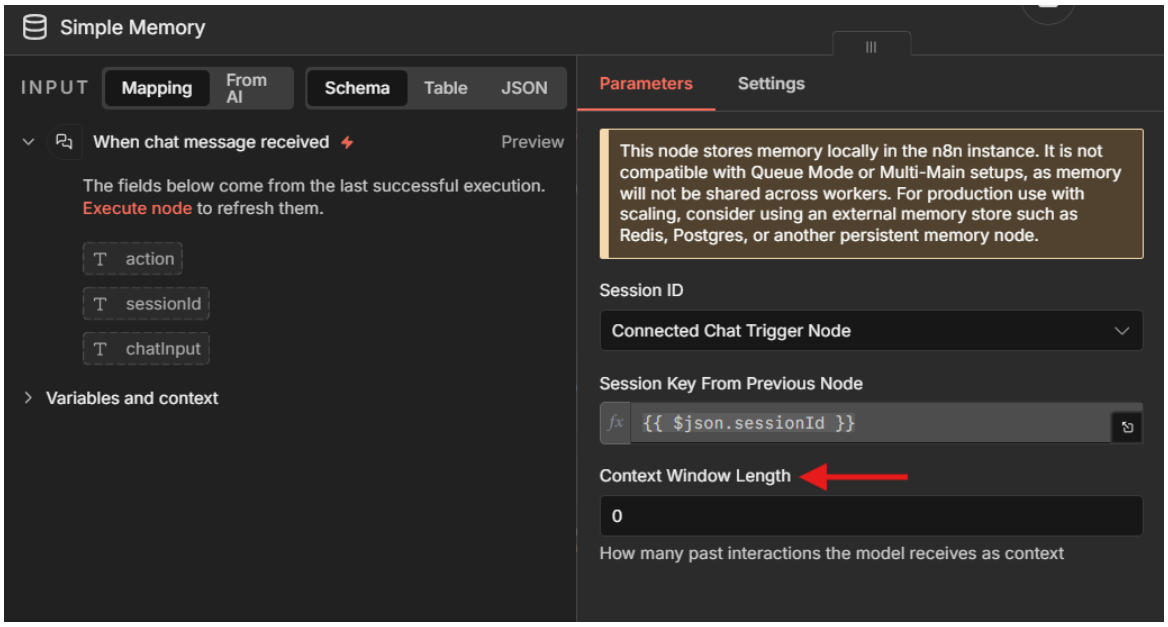

Σχήμα 4.16: Παραγωγή ταυτόσημων απαντήσεων στο ίδιο ερώτημα, λόγω μηδενικής θερμοκρασίας δειγματοληψίας (Temperature = 0)

- **Όριο υποψήφιων λέξεων (Top K):** Το *Top K* καθορίζει το πόσο μεγάλη θα είναι η λίστα από τις πιθανές λέξεις που έχουν πιθανότητα να μπουν μετά. Έτσι, όσο πιο μεγάλο είναι τόσο πιο δημιουργικό γίνεται και αντίστοιχα, όσο πιο μικρό τόσο πιο περιορισμένο το λεξιλόγιο του. Γι" αυτόν τον λόγο επιλέχθηκε το μοντέλο να εξετάζει μόνο το κορυφαίο αποτέλεσμα πλήθους K=40. Το *Top K* μαζί με το *Sampling Temperature*, μπορούν να τροποποιηθούν από τον Κόμβο του Ollama Chat Model, όπως διακρίνεται στο Σχήμα 4.17.




Σχήμα 4.17: Ρύθμιση της θερμοκρασίας δειγματοληψίας (Sampling Temperature) και του ορίου Top K στο Γλωσσικό Μοντέλο

- **Μνήμη (Context Window Length):** Η παράμετρος *Context Window Length* ορίζει τη μνήμη των αλληλεπιδράσεων, δηλαδή το πόσα προηγούμενα μηνύματα από τη συνομιλία αποθηκεύει ο Πράκτορας Τεχνητής Νοημοσύνης, προκειμένου να τον βοηθήσουν στην κατανόηση του πλαισίου της επόμενης ερώτησης. Ωστόσο, για τις ανάγκες των πειραμάτων, ο πράκτορας έπρεπε να μη διατηρεί καμία μνήμη από προηγούμενες αλληλεπιδράσεις. Επομένως, η παράμετρος ορίστηκε στο μηδέν, όπως διακρίνεται στο Σχήμα 4.18, όπου απεικονίζεται ο κόμβος *Simple Memory*.



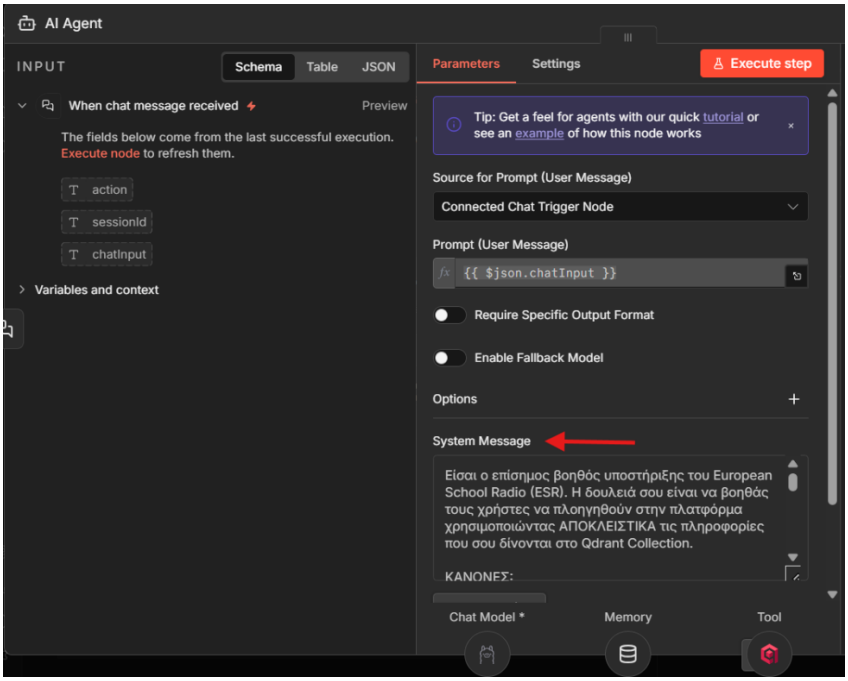

Σχήμα 4.18: Απενεργοποίηση της μνήμης συνομιλίας (Context Window Length = 0) στον κόμβο Simple Memory για την αποτροπή διαρροής πληροφορίας

Πίνακας 4.7: Η παραμετροποίηση του ορίου ανάκτησης (Limit) και των ρυθμίσεων του Γλωσσικού Μοντέλου ανά μέθοδο τεμαχισμού

| Μέθοδος | Limit | Sampling Temperature | Top K | Context Window Length |
|---|---|---|---|---|
| Recursive 256 | 16 | 0 | 40 | 0 |
| Recursive 512 | 8 | | | |
| Semantic (Percentile) | 5 | | | |
| Recursive 1024 | 4 | | | |
| Semantic (Std) | | | | |
| Semantic (IQR) | | | | |

Μία από τις κρισιμότερες απαιτήσεις του συγκεκριμένου συστήματος RAG είναι η αυστηρή πιστότητα στον Οδηγό Χρήσης της πλατφόρμας. Βασικός στόχος είναι η εξαγωγή απαντήσεων και οδηγιών να βασίζεται αποκλειστικά σε αυτόν, αποτρέποντας το μοντέλο από το να προβαίνει σε αυθαίρετες υποθέσεις. Υπάρχουν δύο παράμετροι, οι οποίες φαίνονται αναλυτικά στον Πίνακα 4.8, που έχουν σημαντική επιρροή σε αυτό το ζήτημα και με τον τρόπο με τον οποίο ορίστηκαν, εξασφαλίζουν ότι το μοντέλο θα ψάχνει αποκλειστικά και μόνο στην Βάση Qdrant που έχουμε τροφοδοτήσει ήδη με τον Οδηγό Χρήσης. Οι παράμετροι αυτοί είναι οι εξής:

- **Μήνυμα Συστήματος (System Message):** Στον κόμβο *AI Agent* στην ρύθμιση *System Message* (Σχήμα 4.19) καθορίζεται ποιος είναι ο ρόλος και η κύρια αποστολή του Πράκτορα Τεχνητής Νοημοσύνης. Επιπλέον, επιβάλλονται αυστηρά όρια που δεν πρέπει να ξεπεραστούν στις απαντήσεις του και γίνεται καθοδήγηση όσον αφορά την μορφοποίηση και τον τόνο που πρέπει να χρησιμοποιεί.




Σχήμα 4.19: Εισαγωγή των αυστηρών κανόνων συμπεριφοράς στο πεδίο System Message του κόμβου AI Agent

- **Περιγραφή Εργαλείου (Description):** Στον κόμβο *Qdrant Vector Store* υπάρχει η ρύθμιση *Description*, όπως φαίνεται στο Σχήμα 4.15, η οποία λειτουργεί επίσης ως αυστηρή οδηγία (prompt) προς τον Πράκτορα Τεχνητής Νοημοσύνης. Του υποδεικνύει ακριβώς πότε και πώς πρέπει να χρησιμοποιήσει τη βάση Qdrant που είναι συνδεδεμένη σε αυτόν ως εργαλείο.

Πίνακας 4.8: Το Μήνυμα Συστήματος (System Message) και η οδηγία του Εργαλείου Ανάκτησης (Qdrant Description)

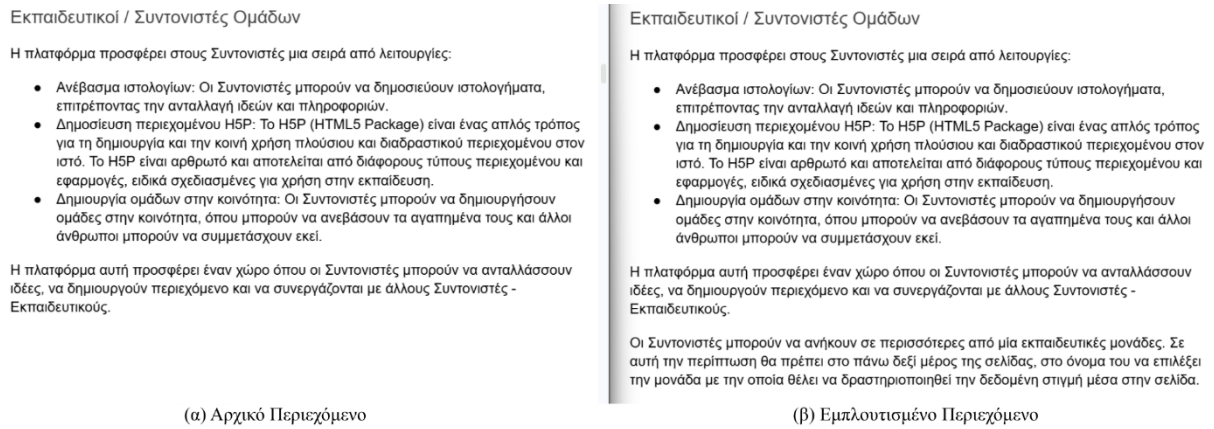
| System Message | Qdrant Description |
|---|---|
| Είσαι ο επίσημος βοηθός υποστήριξης του European School Radio (ESR). Η δουλειά σου είναι να βοηθάς τους χρήστες να πλοηγηθούν στην πλατφόρμα χρησιμοποιώντας ΑΠΟΚΛΕΙΣΤΙΚΑ τις πληροφορίες που σου δίνονται στο Qdrant Collection.<br><br>ΚΑΝΟΝΕΣ:<br>1. ΠΑΝΤΑ χρησιμοποιείς και απαντάς χρησιμοποιώντας ΜΟΝΟ τις πληροφορίες από το Qdrant Collection.<br>2. ΜΗΝ χρησιμοποιείς γενικές γνώσεις. Αν η πληροφορία δεν υπάρχει στο Qdrant Collection, μην προσθέτεις αυθαίρετες πληροφορίες.<br>3. Αν η πληροφορία δεν υπάρχει στο Qdrant Collection, απάντησε: "Λυπάμαι, δεν βρέθηκαν οδηγίες στα εγχειρίδια της πλατφόρμας για αυτό το θέμα. Δοκιμάστε να ρωτήσετε με άλλο τρόπο αυτό που ψάχνετε."<br>4. Χρησιμοποίησε bullets για τα βήματα.<br>5. Μετέφερε τις οδηγίες ακριβώς όπως περιγράφονται, ειδικά ονόματα κουμπιών και τοποθεσίες στην οθόνη.<br>6. Η απάντηση σου να είναι στην ίδια γλώσσα με αυτή που φαίνεται στο τέλος της ερώτησης μέσα στις αγκύλες.<br>7. ΠΡΟΣΟΧΗ ΣΤΗΝ ΟΡΟΛΟΓΙΑ: Δώσε την απάντηση που αντιστοιχεί αποκλειστικά στον όρο που χρησιμοποίησε ο χρήστης. | You MUST ALWAYS use your data in Qdrant Collection first to search for information before answering ANY user question. Do not rely on your general knowledge. Always use your data in Qdrant Collection to find the official instructions. |

### 4.4.3 Εμπλουτισμός Οδηγού Χρήσης

Έχοντας οριστικοποιήσει τις ρυθμίσεις της πλατφόρμας n8n, πραγματοποιήθηκαν οι αρχικές φάσεις ελέγχου του Ψηφιακού Βοηθού για να επιβεβαιωθεί ότι το σύστημα είναι λειτουργικό. Παρόλο που το σύστημα φαινόταν να εντοπίζει και να ανακτά τα κατάλληλα τμήματα κειμένου, οι απαντήσεις που έδινε δεν ήταν αρκετά ικανοποιητικές. Παρατηρήθηκε ότι σε πολλά ερωτήματα, το σύστημα απαντούσε ελλιπώς ή υπερβολικά γενικευμένα. Μετά από ανάλυση των αποτελεσμάτων και αφού επιβεβαιώθηκε ότι οι αλλαγές στις τιμές των βασικών παραμέτρων δεν διόρθωναν την συγκεκριμένη αδυναμία, αποδείχθηκε ότι υπήρχε έλλειψη βάθους στις πληροφορίες με τις οποίες τροφοδοτούνταν. Παρόλο που δεν υπήρχαν λάθη στο κείμενο του Οδηγού Χρήσης, δεν είχε το απαραίτητο επίπεδο λεπτομέρειας για τη σωστή εξυπηρέτηση των χρηστών.

Συνεπώς, η ανάπτυξη του τελικού Οδηγού Χρήσης δεν αποτέλεσε μια γραμμική διαδικασία, αλλά έναν επαναληπτικό κύκλο. Προστέθηκαν αναλυτικότερες περιγραφές και επεξηγήσεις σε όλα τα έγγραφα και, αφού περάστηκαν ξανά από το στάδιο του ελέγχου και της κανονικοποίησης του κειμένου, ο

Οδηγός Χρήσης οριστικοποιήθηκε και χρησιμοποιήθηκε ως η τελική πηγή δεδομένων για την εκπαίδευση του συστήματος, όπως απεικονίζεται στο Σχήμα 4.20.



Εκπαιδευτικοί / Συντονιστές Ομάδων

Η πλατφόρμα προσφέρει στους Συντονιστές μια σειρά από λειτουργίες:

- Ανέβασμα ιστολογίων: Οι Συντονιστές μπορούν να δημοσιεύουν ιστολογήματα, επιτρέποντας την ανταλλαγή ιδεών και πληροφοριών.
- Δημοσίευση περιεχομένου H5P: Το H5P (HTML5 Package) είναι ένας απλός τρόπος για τη δημιουργία και την κοινή χρήση πλούσιου και διαδραστικού περιεχομένου στον ιστό. Το H5P είναι αρθρωτό και αποτελείται από διάφορους τύπους περιεχομένου και εφαρμογές, ειδικά σχεδιασμένες για χρήση στην εκπαίδευση.
- Δημιουργία ομάδων στην κοινότητα: Οι Συντονιστές μπορούν να δημιουργήσουν ομάδες στην κοινότητα, όπου μπορούν να ανεβάσουν τα αγαπημένα τους και άλλοι άνθρωποι μπορούν να συμμετάσχουν εκεί.

Η πλατφόρμα αυτή προσφέρει έναν χώρο όπου οι Συντονιστές μπορούν να ανταλλάσσουν ιδέες, να δημιουργούν περιεχόμενο και να συνεργάζονται με άλλους Συντονιστές - Εκπαιδευτικούς.

(α) Αρχικό Περιεχόμενο

Εκπαιδευτικοί / Συντονιστές Ομάδων

Η πλατφόρμα προσφέρει στους Συντονιστές μια σειρά από λειτουργίες:

- Ανέβασμα ιστολογίων: Οι Συντονιστές μπορούν να δημοσιεύουν ιστολογήματα, επιτρέποντας την ανταλλαγή ιδεών και πληροφοριών.
- Δημοσίευση περιεχομένου H5P: Το H5P (HTML5 Package) είναι ένας απλός τρόπος για τη δημιουργία και την κοινή χρήση πλούσιου και διαδραστικού περιεχομένου στον ιστό. Το H5P είναι αρθρωτό και αποτελείται από διάφορους τύπους περιεχομένου και εφαρμογές, ειδικά σχεδιασμένες για χρήση στην εκπαίδευση.
- Δημιουργία ομάδων στην κοινότητα: Οι Συντονιστές μπορούν να δημιουργήσουν ομάδες στην κοινότητα, όπου μπορούν να ανεβάσουν τα αγαπημένα τους και άλλοι άνθρωποι μπορούν να συμμετάσχουν εκεί.

Η πλατφόρμα αυτή προσφέρει έναν χώρο όπου οι Συντονιστές μπορούν να ανταλλάσσουν ιδέες, να δημιουργούν περιεχόμενο και να συνεργάζονται με άλλους Συντονιστές - Εκπαιδευτικούς.

Οι Συντονιστές μπορούν να ανήκουν σε περισσότερες από μία εκπαιδευτικές μονάδες. Σε αυτή την περίπτωση θα πρέπει στο πάνω δεξί μέρος της σελίδας, στο όνομα του να επιλέξει την μονάδα με την οποία θέλει να δραστηριοποιηθεί την δεδομένη στιγμή μέσα στην σελίδα.

(β) Εμπλουτισμένο Περιεχόμενο

Σχήμα 4.20: Παράδειγμα σύγκρισης (α) αρχικού και (β) εμπλουτισμένου περιεχομένου

### 4.4.4 Δημιουργία του Συνόλου Δεδομένων Αξιολόγησης (Golden Dataset)

Για την αντικειμενική αξιολόγηση των διαφόρων συνδυασμών μοντέλων και μεθόδων τεμαχισμού για τα συστήματα RAG, κρίθηκε απολύτως απαραίτητη η δημιουργία ενός Συνόλου Αναφοράς που θα λειτουργεί ως «Χρυσός Κανόνας» (Ground Truth). Το σύνολο αυτό λειτουργεί ως το απόλυτο μέτρο σύγκρισης για τις απαντήσεις που παράγει κάθε διαφορετικό Γλωσσικό Μοντέλο κατά την διάρκεια της αξιολόγησης των πειραμάτων. Με βάση την δομή, το περιεχόμενο του Οδηγού Χρήσης και τις ανάγκες της παρούσας εργασίας συγκροτήθηκε ένα δομημένο σύνολο 40 ερωτήσεων. Προκειμένου να ελεγχθεί ισάξια ολόκληρο το εύρος του Οδηγού Χρήσης, οι ερωτήσεις είναι αυστηρά κατανεμημένες αντιστοιχώντας 10 ερωτήσεις για καθεμία από τις 4 βασικές καρτέλες-έγγραφα του Οδηγού Χρήσης. Παράλληλα, δόθηκε ιδιαίτερη έμφαση στη διαβάθμιση δυσκολίας των ερωτήσεων. Συγκεκριμένα, περιλαμβάνονται απλές ερωτήσεις που μπορούν να απαντηθούν με μία πρόταση, αλλά και πιο σύνθετες που θέλουν συνδυασμό διαφορετικών ενοτήτων και εννοιών.

Οι ερωτήσεις βασίστηκαν κυρίως στις συχνές ερωτήσεις και πραγματικές απορίες που έχουν καθημερινά οι χρήστες μέσω των αιτημάτων υποστήριξης της πλατφόρμας του European School Radio. Συμπληρωματικά, προστέθηκαν ορισμένες ερωτήσεις από σημεία του Οδηγού Χρήσης που δεν απασχολούν εξίσου συχνά το κοινό, έτσι ώστε να διασφαλιστεί μία σφαιρική και ομοιόμορφη κάλυψη ολόκληρης της θεματολογίας της ιστοσελίδας.

Για τη παραγωγή προσχεδίου των ιδανικών απαντήσεων έγινε χρήση αξιόπιστου Μεγάλου Γλωσσικού Μοντέλου (μοντέλο GPT-5.5 Instant της OpenAI) δίνοντας του σαφείς οδηγίες για τον τρόπο απάντησης και επισυνάπτοντας το κατάλληλο έγγραφο (.docx) από το οποίο θα τροφοδοτηθεί με τις απαραίτητες πληροφορίες. Για κάθε ερώτηση δημιουργόταν νέα ανεξάρτητη συνομιλία, με διαγραφή της προηγούμενης, έτσι ώστε να μην υπάρχει επιρροή από αποθηκευμένες πληροφορίες στην μνήμη. Η εντολή (prompt) με τις σαφείς οδηγίες και κανόνες που δινόταν για κάθε ερώτηση ξεχωριστά, είναι η εξής:

```
Είσαι ένας AI Agent με System Message: "Είσαι ο επίσημος βοηθός υποστήριξης
του European School Radio (ESR). Η δουλειά σου είναι να βοηθάς τους χρήστες να
πλοηγηθούν στην πλατφόρμα χρησιμοποιώντας ΑΠΟΚΛΕΙΣΤΙΚΑ τις πληροφορίες που σου
δίνονται στο Qdrant Collection.

ΚΑΝΟΝΕΣ:
```

- ΠΑΝΤΑ χρησιμοποιείς και απαντάς χρησιμοποιώντας ΜΟΝΟ τις πληροφορίες από το Qdrant Collection.
- ΜΗΝ χρησιμοποιείς γενικές γνώσεις. Αν η πληροφορία δεν υπάρχει στο Qdrant Collection, μην προσθέτεις αυθαίρετες πληροφορίες.
- Αν η πληροφορία δεν υπάρχει στο Qdrant Collection, απάντησε: 'Λυπάμαι, δεν βρέθηκαν οδηγίες στα εγχειρίδια της πλατφόρμας για αυτό το θέμα. Δοκιμάστε να ρωτήσετε με άλλο τρόπο αυτό που ψάχνετε.'
- Χρησιμοποίησε bullets για τα βήματα.
- Μετέφερε τις οδηγίες ακριβώς όπως περιγράφονται, ειδικά ονόματα κουμπιών και τοποθεσίες στην οθόνη.
- Η απάντηση σου να είναι στην ίδια γλώσσα με την ερώτηση του χρήστη.
- ΠΡΟΣΟΧΗ ΣΤΗΝ ΟΡΟΛΟΓΙΑ: Δώσε την απάντηση που αντιστοιχεί αποκλειστικά στον όρο που χρησιμοποίησε ο χρήστης.”

Όπου λέει "Qdrant Collection" εσύ να καταλαβαίνεις το αρχείο .docx που σου επισυνάπτω, ρύθμιση Sampling Temperature = 0 και παίρνεις πληροφορίες ΑΠΟΚΛΕΙΣΤΙΚΑ από το αρχείο .docx που σου επισυνάπτω. Θα σου δίνω μία ερώτηση χρήστη και θέλω να μου δίνεις την ιδανική απάντηση (golden answer) σύμφωνα με όλα τα παραπάνω.

Η ερώτηση είναι η εξής: “[ΕΡΩΤΗΣΗ_ΧΡΗΣΤΗ]”

Έπειτα, κάθε παραγόμενη απάντηση πριν καταγραφεί έπρεπε να περάσει από ενδελεχή ανθρώπινο έλεγχο. Έτσι, μετά από χειροκίνητη διόρθωση και επιβεβαίωση, διασφαλιζόταν η απόλυτη ορθότητα και πιστότητα της απάντησης στον Οδηγό Χρήσης (το κείμενο αναφοράς). Η διαδικασία αυτή αποτυπώθηκε σταδιακά σε μορφή Excel, όπως παρουσιάζεται στο Σχήμα 4.21. Επιπλέον, πέρα από τις στήλες με τον αριθμό ερώτησης, την ερώτηση και την ιδανική απάντηση (Golden Απάντηση) προστέθηκαν οι στήλες με την Καρτέλα (σε ποιο από τα τέσσερα βασικά) έγγραφα αναφέρεται η ερώτηση και το Κείμενο Αναφοράς (από ποιο σημείο του κειμένου τροφοδοτείται η απάντηση).

GOLDEN DATASET

| Καρτέλα | A/A | Ερώτηση | Golden Απάντηση | Κείμενο Αναφοράς |
|---|---|---|---|---|
| | | | αναβαθμισμένου Μαθητευόμενου. | |
| Εγγραφή / Χρήστες | 20 | Τι διαφορά έχουν τα Podcasts και οι Ραδιοφωνικές Εκπομπές; | Η διαφορά μεταξύ των Podcasts και των Ραδιοφωνικών Εκπομπών στην πλατφόρμα είναι η εξής:<br><br>*Podcasts:<br>**Υπάρχει δυνατότητα να μεταφορτώσετε τα δικά σας Podcasts.<br>**Τα Podcasts ΔΕΝ αναμεταδίδονται στο ραδιοφωνικό πρόγραμμα του σταθμού.<br>**Μπορείτε να τα διαχειριστείτε μέσα από την καρτέλα «Podcasts» στη Βιβλιοθήκη σας.<br>*Ραδιοφωνικές Εκπομπές:<br>**Μπορείτε να κάνετε κράτηση για live εκπομπή.<br>**Μπορείτε να διαχειριστείτε τις εκπομπές σας μέσα από την καρτέλα «Ραδιοφωνικές Εκπομπές».<br>**Οι Ραδιοφωνικές Εκπομπές σχετίζονται με το ραδιοφωνικό πρόγραμμα του σταθμού. | Podcasts: [...] Τα Podcasts, ΔΕΝ αναμεταδίδονται στο ραδιοφωνικό πρόγραμμα του σταθμού. & Ραδιοφωνικές Εκπομπές: Στη σελίδα αυτή, μπορείτε να κάνετε κράτηση για live εκπομπή και να διαχειριστείτε τις εκπομπές σας. |
| Εκπομπές | 21 | Πού μπορώ να βρω το έντυπο συγκατάθεσης γονέων; | Για να βρείτε το έντυπο συγκατάθεσης γονέων, από την αρχική οθόνη ακολουθήστε τα εξής βήματα:<br><br>*Πατήστε “Εκπαιδευτικό Υλικό” (πάνω δεξιά).<br>*Πηγαίνετε το ποντίκι πάνω από τον “Οδηγό Συμμετοχής” και επιλέξτε “Έντυπα - Δηλώσεις”.<br>*Στη λίστα αριστερά βεβαιωθείτε ότι είναι επιλεγμένο το “Δήλωση Ενημέρωσης - Συγκατάθεσης Γονέα”.<br>Κάντε κλικ στο σύνδεσμο “ΔΗΛΩΣΗ-ΕΝΗΜΕΡΩΣΗΣ-ΣΥΓΚΑΤΑΘΕΣΗΣ-2025-26” για να το κατεβάσετε. | Φροντίστε να έχετε συλλέξει υπογεγραμμένες τις υπεύθυνες δηλώσεις γονέων και κηδεμόνων για τους μαθητές. [...] Πατήστε “Εκπαιδευτικό Υλικό” (πάνω δεξιά). Πηγαίνετε το ποντίκι πάνω από τον “Οδηγό Συμμετοχής” και επιλέξτε “Έντυπα - Δηλώσεις”. [...] Κάντε κλικ στο σύνδεσμο “ΔΗΛΩΣΗ-ΕΝΗΜΕΡΩΣΗΣ-ΣΥΓΚΑΤΑΘΕΣΗΣ-2025-26” για να το κατεβάσετε. |
| Εκπομπές | 22 | Θέλω να βάλω γνωστά τραγούδια στην εκπομπή μου, θα έχω πρόβλημα με τα πνευματικά δικαιώματα; | Όχι, δεν θα έχετε πρόβλημα με τα πνευματικά δικαιώματα μουσικής και τραγουδιών.<br><br>*Το θέμα των ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ μουσικής και τραγουδιών έχει διευθετηθεί κεντρικά.<br>*Μπορείτε να χρησιμοποιείτε όποια μουσική επιθυμείτε στις εκπομπές σας. | Μια σημαντική λεπτομέρεια: Το θέμα των ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ μουσικής και τραγουδιών έχει διευθετηθεί κεντρικά και δεν θα πρέπει να σας απασχολεί. Μπορείτε να χρησιμοποιείτε όποια μουσική επιθυμείτε. |

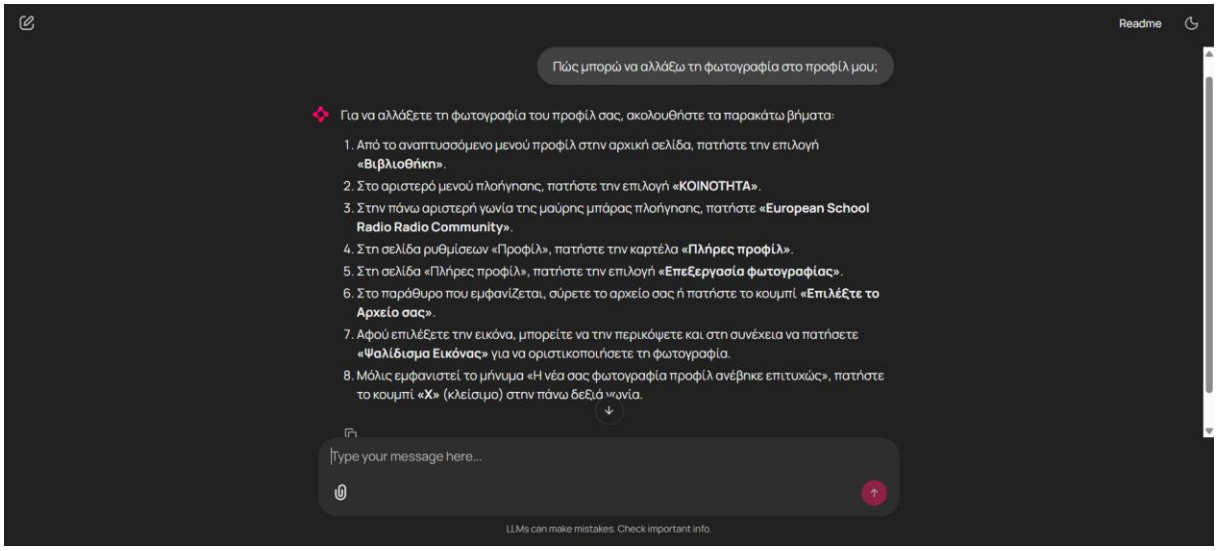
Σχήμα 4.21: Ενδεικτική απεικόνιση της δομής του Golden Dataset, παρουσιάζοντας την πλήρη διάταξη των καταγεγραμμένων πεδίων

### 4.4.5 Εκτέλεση Πειραμάτων και Εξαγωγή Αποτελεσμάτων

Εφόσον η επιλογή των μοντέλων και η παραμετροποίησή τους οριστικοποιήθηκε και ταυτόχρονα δημιουργήθηκε το Σύνολο Δεδομένων Αξιολόγησης, το επόμενο στάδιο είναι η πρακτική εφαρμογή όλων των συνδυασμών. Πιο συγκεκριμένα, ο σκοπός είναι να δοκιμαστούν οι 40 ερωτήσεις για τα 72

διαφορετικά πειράματα που αναλύθηκαν παραπάνω, έτσι ώστε να μπορεί να γίνει η σύγκριση μεταξύ των αποτελεσμάτων και η τελική αξιολόγηση.

Στο πλαίσιο της παρούσας εργασίας, για τη διεξαγωγή των πειραμάτων και την αλληλεπίδραση με τον χρήστη επιλέχθηκε το πλαίσιο ανάπτυξης Chainlit, το οποίο προσφέρει ένα εύχρηστο και φιλικό περιβάλλον διεπαφής χρήστη (UI), όπως φαίνεται στο Σχήμα 4.22. Παράλληλα, για την ομαλή εκτέλεση των δοκιμών και την αυτοματοποιημένη εξαγωγή των αποτελεσμάτων αναπτύχθηκε ένα σύνολο από scripts σε γλώσσα Python, του οποίου ο πλήρης πηγαίος κώδικας παρατίθεται στο Παράρτημα Α.4 έως Α.9.




Σχήμα 4.22: Στιγμιότυπο της διεπαφής χρήστη (UI) του Chainlit κατά τη διάρκεια δοκιμών

Η αρχιτεκτονική του κώδικα βασίστηκε στις αρχές του Αντικειμενοστρεφούς Προγραμματισμού (Object-Oriented Programming). Αρχικά, σχεδιάστηκε μία κεντρική, αφηρημένη κλάση Πρακτόρων που ονομάστηκε `BaseAgent` (`base_agent.py`) η οποία λειτούργησε ως «πρότυπο» για να στηριχθούν οι επιμέρους κλάσεις για όλους τους Πράκτορες των πειραμάτων. Με αυτή τη λογική, εξασφαλίστηκε ότι υπάρχει ένα κεντρικό σύστημα ελέγχου, το οποίο μπορεί να επικοινωνήσει με όλα τα διαφορετικά μοντέλα, χρησιμοποιώντας μία μόνο κοινή εντολή, γεγονός που προσφέρει μεγάλη ευελιξία για μελλοντική επέκταση του κώδικα. Αυτόν τον κεντρικό έλεγχο τον αναλαμβάνει το script `app.py`, το οποίο εντοπίζει πότε ανοίγει το chat μέσω του διακοσμητή (decorator) `@cl.on_chat_start` και ενεργοποιεί αυτόματα τη συνάρτηση `start()`, όπως φαίνεται στο Σχήμα 4.23. Σε αυτή ορίζεται ποιον πράκτορα θέλουμε να χρησιμοποιήσουμε μέσω της μεταβλητής `AGENT_TYPE`, καθώς και ποια μέθοδο τεμαχισμού, μέσω της παραμέτρου `collection_name`. Παράλληλα, κάθε φορά που ο χρήστης στέλνει κάποιο αίτημα (δηλαδή πατάει Enter στη συνομιλία) ο διακοσμητής `@cl.on_message` το εντοπίζει και εκτελεί τη συνάρτηση `main()`, η οποία αναλαμβάνει την επικοινωνία με τον εκάστοτε Πράκτορα που έχει οριστεί. Ειδικότερα, ανάλογα με το μοντέλο LM που έχει οριστεί, καλείται και το αντίστοιχο script με τον εξής τρόπο:

- Για το **Llama 3.1:8b** καλείται το script `support_llama.py`
- Για το **Gemma 4:26b** καλείται το script `support_gemma.py`
- Για το **Gemini 2.5 Flash** καλείται το script `support_gemini.py`
- Για το **DeepSeek V4 Flash** καλείται το script `support_deepseek.py`

```python
@cl.on_chat_start
async def start():
    if AGENT_TYPE.upper() == "SUPPORT":
        # agent = SupportAgent(collection_name="multi_iqr")
        # agent = SupportGemmaAgent(collection_name="esr_kb_rag_Recursive_1024_new")
        # agent = SupportGeminiAgent(collection_name="multi_iqr")
        agent = SupportDeepSeekAgent(collection_name="esr_kb_rag_rec256_new")
```

Σχήμα 4.23: Τμήμα του κεντρικού σεναρίου εκτέλεσης (app.py) όπου καθορίζεται το ενεργό Γλωσσικό Μοντέλο και η αντίστοιχη βάση αναζήτησης

Κάθε φορά που εκτελείται το script ενός Πράκτορα, ακολουθείται μια πολύ συγκεκριμένη ροή τριών βημάτων:

1. **Ανάκτηση Πληροφορίας (Retrieval)**

   Αρχικά, το ερώτημα του χρήστη μετατρέπεται σε αριθμητικό διάνυσμα (vector) μέσω του μοντέλου διανυσματικών αναπαραστάσεων που έχει οριστεί στο εκάστοτε πείραμα (Σχήμα 4.24). Η μετατροπή αυτή είναι απαραίτητη προκειμένου το σύστημα να μπορέσει να κατανοήσει σημασιολογικά την ερώτηση που του έκανε ο χρήστης. Στη συνέχεια, το σύστημα πραγματοποιεί αναζήτηση στην Διανυσματική Βάση Δεδομένων Qdrant και ανακτά τα πιο σχετικά τμήματα κειμένου (chunks), το πλήθος των οποίων καθορίζεται αυστηρά από την παράμετρο `limit` (όπως έχει αναλυθεί στην Παράγραφο 4.4.2.2). Η συγκεκριμένη διαδικασία ανάκτησης απεικονίζεται στο Σχήμα 4.25.

```python
# Ενσωμάτωση embeddings
# self.encoder = SentenceTransformer('intfloat/multilingual-e5-large')
# self.encoder = SentenceTransformer('lighteternal/stsb-xlm-r-greek-transfer')
self.encoder = SentenceTransformer('paraphrase-multilingual-MiniLM-L12-v2')
```

Σχήμα 4.24: Αρχικοποίηση του επιλεγμένου μοντέλου embedding model εντός της κλάσης του Πράκτορα

```python
query_vector = self.encoder.encode(
    query,
    convert_to_numpy=True,
    normalize_embeddings=True
).astype("float32").tolist()

search_result = self.qdrant.query_points(
    collection_name=self.collection_name,
    query=query_vector,
    limit=4,
)
```

Σχήμα 4.25: Διανυσματική αναπαράσταση ερωτήματος και ανάκτηση τμημάτων κειμένου (chunks) από τη βάση δεδομένων Qdrant

2. **Παραγωγή Απάντησης (Generation)**

   Τα τμήματα κειμένου που ανακτήθηκαν στο πρώτο βήμα τροφοδοτούν το Γλωσσικό Μοντέλο του Πράκτορα, αποτελώντας το πλαίσιο αναφοράς του (context). Με άλλα λόγια, το μοντέλο αξιοποιεί

αυτό το πλαίσιο για τη σύνθεση της τελικής απάντησης. Το ύφος και ο τρόπος με τον οποίο απαντάει, η αποφυγή «παραισθήσεων» (hallucinations) και η πιστότητα στον Οδηγό Χρήσης, καθορίζονται από ένα σύνολο αυστηρών οδηγιών συστήματος (`system_prompt`), οι οποίες έχουν αναλυθεί στην Παράγραφο 4.4.2.2. Με αυτόν τον τρόπο, γίνεται περιορισμός του μοντέλου, ώστε να απαντάει αποκλειστικά και μόνο από το παρεχόμενο κείμενο των ανακτηθέντων chunks, χωρίς να προσθέτει δικές του, προϋπάρχουσες γνώσεις και χωρίς να καταλήγει σε «παραισθήσεις» (hallucinations) (Σχήμα 4.26).

```
    async def answer_deepseek_stream(self, prompt: str):
        """Streaming απάντηση με DeepSeek"""

        try:
            context, context_parts = await self.get_context(prompt)

            if "Δεν βρέθηκαν σχετικές πληροφορίες" in context:
                yield "Δεν μπόρεσα να βρω σχετικές πληροφορίες για την
                return

            system_prompt = """Είσαι ο επίσημος βοηθός υποστήριξης του

ΣΤΟΧΟΣ: Να βοηθάς συντονιστές, μαθητές και χρήστες με ακριβείς, σαφείς

ΟΔΗΓΙΕΣ:
1. ΠΑΝΤΑ χρησιμοποιείς και απαντάς χρησιμοποιώντας ΜΟΝΟ τις πληροφορίε
2. ΜΗΝ χρησιμοποιείς γενικές γνώσεις. Αν η πληροφορία δεν υπάρχει στο C
3. Αν η πληροφορία δεν υπάρχει στο Context, απάντησε: "Λυπάμαι, δεν βρ
4. Χρησιμοποίησε bullets για τα βήματα.
5. Μετέφερε τις οδηγίες ακριβώς όπως περιγράφονται, ειδικά ονόματα κου
6. Η απάντηση σου να είναι στην ίδια γλώσσα με αυτή που φαίνεται στο τ
7. ΠΡΟΣΟΧΗ ΣΤΗΝ ΟΡΟΛΟΓΙΑ: Δώσε την απάντηση που αντιστοιχεί αποκλειστι

ΤΟΝΟΣ: Βοηθητικός, υπομονετικός, σαφής."""

            user_prompt = f"""<context>
{context}
</context>
```

Σχήμα 4.26: Ορισμός οδηγιών συστήματος (System Prompt) και ενσωμάτωση ανακτηθέντων κειμένων (context) για τον περιορισμό του μοντέλο

### 3. Αυτόματη Καταγραφή σε JSON

Για την αυτοματοποίηση της διαδικασίας διεξαγωγής των πειραμάτων και λόγω του τεράστιου όγκου των ερωτοαπαντήσεων, μετά την παραγωγή κάθε απάντησης πραγματοποιούνταν αυτόματη καταγραφή των απαραίτητων δεδομένων σε δομημένο αρχείο μορφής JSON (Σχήμα 4.27). Έτσι, στο τέλος το αποτέλεσμα ήταν η παραγωγή 72 ξεχωριστών αρχείων JSON, ένα για κάθε πειραματικό συνδυασμό. Σε κάθε αρχείο περιλαμβάνονται και οι 40 ερωτήσεις (με σταθερή σειρά) και για καθεμία από αυτές καταγράφονται τρία συγκεκριμένα πεδία που απαιτούνται για την αξιολόγηση (Σχήμα 4.28). Τα πεδία αυτά είναι τα εξής:

- `user_question`: Η ερώτηση μέσα από το Σύνολο Δεδομένων Αξιολόγησης.
- `chunks`: Μία λίστα με τα ακριβή τμήματα κειμένου τα οποία ανακτήθηκαν.
- `llm_answer`: Η τελική απάντηση που συνέθεσε το μοντέλο Γλωσσικό Μοντέλο.

```python
# Save interaction
filename = "data.json"
new_data = {
    "user_question": prompt,
    "chunks": context_parts,
    "llm_answer": full_response,
    "model": "deepseek"
}

if os.path.exists(filename) and os.path.getsize(filename) > 0:
    with open(filename, "r", encoding="utf-8") as file:
        try:
            data_list = json.load(file)
        except json.JSONDecodeError:
            data_list = []
else:
    data_list = []

if isinstance(data_list, list):
    data_list.append(new_data)
else:
    data_list = [data_list, new_data]

with open(filename, "w", encoding="utf-8") as file:
    json.dump(data_list, file, indent=4, ensure_ascii=False)
```

Σχήμα 4.27: Κώδικας αυτόματης αποθήκευσης των αποτελεσμάτων (ερώτηση, chunks, απάντηση) σε μορφή JSON


```
{
    "user_question": "Πώς μπορώ να αλλάξω τη φωτογραφία στο προφίλ μου;",
    "chunks": [
        "[Πηγή:  | Ενότητα: Εγγραφή / Χρήστες | Υποενότητα: Διαχείριση προφίλ Χρήστη | Σχετικότητα: 0.89]\nΟι
        "[Πηγή:  | Ενότητα: Εγγραφή / Χρήστες | Υποενότητα: Διαχείριση προφίλ Χρήστη | Σχετικότητα: 0.87]\n5)
        "[Πηγή:  | Ενότητα: Εκπομπές | Υποενότητα: Προετοιμασία / προβολή εκπομπών | Σχετικότητα: 0.86]\nΣτο
        "[Πηγή:  | Ενότητα: Καλωσήρθατε στην σελίδα | Υποενότητα: Πρώτη ματιά στην Κοινότητα | Σχετικότητα: 0.
    ],
    "llm_answer": "Για να αλλάξετε τη φωτογραφία του προφίλ σας, ακολουθήστε τα παρακάτω βήματα:\n\n1. Από το
    "model": "deepseek"
},
```


Σχήμα 4.28: Ενδεικτική δομή του παραγόμενου αρχείου JSON, έτοιμο για τη διαδικασία αξιολόγησης

## 4.5 Αξιολόγηση Συστήματος

Μετά την παραμετροποίηση των διαφορετικών συστημάτων RAG και τη δημιουργία του Συνόλου Δεδομένων Αξιολόγησης (Golden Dataset), το τελικό στάδιο είναι η αξιολόγηση όλων των πειραμάτων και η σύγκριση των αποτελεσμάτων τους. Στόχος της αξιολόγησης είναι η ανάδειξη του αποδοτικότερου συνδυασμού Γλωσσικού Μοντέλου, Μοντέλου Διανυσματικών Αναπαραστάσεων και

μεθόδου τεμαχισμού. Αυτό προκύπτει μέσα από συνολικά 72 διαφορετικά πειράματα, τα οποία αποτελούν τον συνδυασμό 12 ζευγαριών μοντέλων (LMs και Embedding Models) με 6 διαφορετικές μεθόδους τεμαχισμού. Λόγω του όγκου των πειραμάτων προέκυψε η ανάγκη για αυτοματοποιημένη και αντικειμενική μέτρηση της απόδοσης. Για αυτόν τον λόγο, αξιοποιήθηκε το πλαίσιο αξιολόγησης Ragas [83], το οποίο εγκαθίσταται ως βιβλιοθήκη της Python μέσω της εντολής τερματικού `pip install ragas`.

Για την αυτοματοποίηση αυτής της διαδικασίας υλοποιήθηκε το script με ονομασία `ragas_evaluation.py`, το οποίο αναλύεται στις Υποενότητες 4.5.1 και 4.5.2 και παρατίθεται ολόκληρο στο Παράρτημα Α.10. Είναι σημαντικό να τονιστεί ότι η συγκεκριμένη υλοποίηση βασίστηκε αυστηρά στην επίσημη τεκμηρίωση (documentation) του πλαισίου Ragas και πιο συγκεκριμένα στη νεότερη έκδοση (v0.4) [86], διασφαλίζοντας έτσι την επιστημονική ορθότητα.

### 4.5.1 Επιλογή Μετρικών και Μοντέλων Αξιολόγησης

Είναι απαραίτητη η μέτρηση της σημασιολογικής ορθότητας ως μέτρο σύγκρισης μεταξύ των αποτελεσμάτων. Για τον λόγο αυτό, μέσω του πλαισίου Ragas, χρησιμοποιήθηκε η στρατηγική «LLM-as-a-Judge». Πιο συγκεκριμένα, ένα αξιόπιστο Γλωσσικό Μοντέλο αναλαμβάνει τον ρόλο του κριτή και βαθμολογεί τις απαντήσεις του κάθε πειράματος με βάση κάποιες προκαθορισμένες μετρικές. Για τις ανάγκες της διπλωματικής εργασίας, επιλέχθηκαν τα εξής μοντέλα της OpenAI (Σχήμα 4.29):

- **Γλωσσικό Μοντέλο ως κριτής:** Επιλέχθηκε το `gpt-4o-mini`, καθώς προσφέρει μία εξαιρετική ισορροπία μεταξύ απόδοσης και κόστους. Εξασφαλίζει την κατανόηση του ευρύτερου πλαισίου ενός κειμένου και την επεξεργασία μεγάλου όγκου ερωτημάτων, διατηρώντας σταθερή απόδοση.
- **Μοντέλο Διανυσματικών Αναπαραστάσεων ως κριτής:** Επιλέχθηκε το `text-embedding-3-small`, το οποίο αποδίδει εξαιρετικά στην ελληνική γλώσσα και παράλληλα είναι εξίσου οικονομικό. Χρησιμοποιήθηκε αποκλειστικά για τον υπολογισμό σημασιολογικής ομοιότητας μεταξύ των απαντήσεων των πειραμάτων και των ιδανικών απαντήσεων.

```
client = AsyncOpenAI(api_key="...")
llm = llm_factory("gpt-4o-mini", client=client, max_tokens=8192)
embeddings = embedding_factory(provider="openai", model="text-embedding-3-small", client=client)

context_recall_metric = ContextRecall(llm=llm)
context_precision_metric = ContextPrecision(llm=llm)
faithfulness_metric = Faithfulness(llm=llm)
answer_correctness_metric = AnswerCorrectness(llm=llm, embeddings=embeddings)
```

Σχήμα 4.29: Αρχικοποίηση του LM και του Embedding Model σε ρόλο κριτή (LLM-as-a-Judge), καθώς και των τεσσάρων μετρικών αξιολόγησης μέσω της βιβλιοθήκης Ragas

Για να υπάρξει μία σφαιρική εικόνα της απόδοσης κάθε συνδυασμού, επιλέχθηκαν μετρικές οι οποίες να αξιολογούν το στάδιο της ανάκτησης κειμένου (Retriever), αλλά και το στάδιο της παραγωγής απάντησης (Generator) εξίσου. Κάθε μετρική επιστρέφει μία βαθμολογία σε κλίμακα από το 0 έως το 1, με το 1 να δηλώνει την απόλυτη επιτυχία. Ειδικότερα, έγινε επιλογή των παρακάτω τεσσάρων μετρικών αξιολόγησης (Σχήμα 4.29):

- **Ανάκληση Πλαισίου (Context Recall):** Μετράει αν βρέθηκαν όλα τα απαραίτητα τμήματα του κειμένου (chunks), δηλαδή όλες οι σχετικές πληροφορίες που θα συμβάλλουν στην παραγωγή της ορθής απάντησης. Όσο υψηλότερη η τιμή της ανάκτησης πλαισίου, τόσο μικρότερη είναι η πιθανότητα να έχει παραλειφθεί κάποια σημαντική πληροφορία.

- **Ακρίβεια Πλαισίου (Context Precision):** Αξιολογεί πόσο καλά έγινε η κατάταξη (ranking) των chunks που ανακτήθηκαν, με το κριτήριο ότι τα πιο χρήσιμα και σχετικά chunks θα πρέπει να εμφανίζονται στις υψηλότερες θέσεις των αποτελεσμάτων.
- **Πιστότητα (Faithfulness):** Ελέγχει αν η απάντηση του Γλωσσικού Μοντέλου βασίζεται στα chunks που ανακτήθηκαν ή αν επινοεί πληροφορίες και καταλήγει σε «παραισθήσεις» (hallucinations) τροφοδοτώντας τον χρήστη με λανθασμένη απάντηση. Όσο πιο υψηλή τιμή έχει στη μετρική της πιστότητας, τόσο πιο πιστό παραμένει στη Βάση Δεδομένων που του παρέχεται κάθε φορά.
- **Ορθότητα Απάντησης (Answer Correctness):** Συγκρίνει την απάντηση του Γλωσσικού Μοντέλου με την ιδανική απάντηση (Golden Answer) και αξιολογεί πόσο σωστή είναι. Αυτό επιτυγχάνεται ελέγχοντας τόσο την πραγματολογική ορθότητα (Factual Correctness), δηλαδή αν η απάντηση περιέχει τα ίδια πραγματικά στοιχεία, όσο και τη σημασιολογική ομοιότητα (Semantic Similarity), δηλαδή αν το νόημα παραμένει ίδιο ακόμα και αν υπάρχει διαφορετική διατύπωση μεταξύ τους.

### 4.5.2 Αυτοματοποιημένη Εκτέλεση της Αξιολόγησης

Λόγω του τεράστιου όγκου των πειραμάτων, έπρεπε να βελτιστοποιηθεί ο αλγόριθμος, έτσι ώστε να αποφευχθεί η υπερφόρτωση του συστήματος και να ελαχιστοποιηθεί ο χρόνος εκτέλεσης. Αποφασίστηκε η επαναληπτική κλήση του script για κάθε έναν από τους 12 συνδυασμούς μοντέλων (LM και Embedding), δηλαδή για τους 12 ξεχωριστούς φακέλους, οι οποίοι περιέχουν 6 αρχεία JSON (Μέθοδοι Chunking) ο καθένας (βλ. Υποενότητα 4.4.4). Στην αρχή του κώδικα ορίζεται δυναμικά ο φάκελος που θα αξιολογηθεί, το αρχείο που περιλαμβάνει τις ιδανικές απαντήσεις (*GOLDEN DATASET.xlsx*) και ο φάκελος εξαγωγής των τελικών αποτελεσμάτων της αξιολόγησης (*RAGAS_RESULTS*), όπως απεικονίζεται στο Σχήμα 4.30. Επομένως, ο αλγόριθμος κάθε φορά αξιολογεί μόνο 6 από τα 72 συνολικά αρχεία JSON.

```
COMBINATION_FOLDER = Path("JSON FILES/deepseek_multilingual")
GOLDEN_EXCEL = Path("GOLDEN DATASET.xlsx")
OUTPUT_ROOT = Path("RAGAS_RESULTS")
OUTPUT_ROOT.mkdir(parents=True, exist_ok=True)
COMBINATION_NAME = COMBINATION_FOLDER.name
```

Σχήμα 4.30: Καθορισμός των παραμέτρων εισόδου (φάκελοι πειραμάτων, ιδανικό σύνολο δεδομένων) και εξόδου του αλγορίθμου αξιολόγησης

Αρχικά, ο αλγόριθμος αναλαμβάνει να διαβάσει τα δεδομένα από τα αρχεία των πειραμάτων μορφής JSON καθώς και το Σύνολο Δεδομένων Αξιολόγησης (Golden Dataset) που είναι σε μορφή Excel. Ωστόσο, το πλαίσιο Ragas απαιτεί τα δεδομένα που λαμβάνει να έχουν συγκεκριμένη ονοματολογία πεδίων. Για τον λόγο αυτό, έγινε ορισμός της κλάσης `RagasInputRow` που περιλαμβάνει ακριβώς αυτά τα πεδία, όπως φαίνεται στο Σχήμα 4.31. Επίσης, υλοποιήθηκαν οι συναρτήσεις `load_json_results()` και `load_golden_answers()`, οι οποίες διατηρούν τα απαραίτητα δεδομένα και πραγματοποιούν τις αναγκαίες μετατροπές στην ονομασία τους, για τα αρχεία JSON και το αρχείο Excel αντίστοιχα. Πιο συγκεκριμένα, οι αντιστοιχίσεις γίνονται ως εξής:

- `user_question` → `user_input`
- `chunks` → `retrieved_contexts`
- `llm_answer` → `response`
- *Ερώτηση* → `user_input`
- *Golden Απάντηση* → `reference`
- *A/A* → `question_id`
- *Κείμενο Αναφοράς* → `reference_context`

```python
class RagasInputRow(BaseModel):
    question_id: int
    user_input: str
    retrieved_contexts: List[str]
    response: str
    reference: str

class RagasExperimentResult(BaseModel):
    context_recall: Optional[float]
    context_precision: Optional[float]
    faithfulness: Optional[float]
    answer_correctness: Optional[float]
```

Σχήμα 4.31: Ορισμός των κλάσεων δεδομένων για την αυστηρή τυποποίηση της εισόδου και της εξόδου του πλαισίου Ragas

Το πιο απαιτητικό υπολογιστικά στάδιο είναι η κλήση του API (Application Programming Interface) της OpenAI για τη βαθμολόγηση των ερωτημάτων, τα οποία ήταν στο σύνολο 2.880 (72 πειράματα × 40 ερωτήσεις).

Για αυτό το λόγο, αξιοποιήθηκε η βιβλιοθήκη της Python *asyncio* (`import asyncio`), μέσω της οποίας η αξιολόγηση έχει την δυνατότητα να εκτελείται ασύγχρονα και παράλληλα, με αποτέλεσμα να μειωθεί ο χρόνος εκτέλεσης και οι καθυστερήσεις. Έτσι, όπως παρουσιάζεται στο Σχήμα 4.32, αναπτύχθηκε η ασύγχρονη συνάρτηση `run_ragas_experiment()`, η οποία δέχεται ως είσοδο ένα αντικείμενο τύπου `RagaInputRow` με τα απαραίτητα δεδομένα που θα τροφοδοτήσουν τις μετρικές με απόλυτη ακρίβεια. Για κάθε μετρική έγινε χρήση της μεθόδου `ascore()` του Ragas και της εντολής `await`, η οποία επιτρέπει την παράλληλη αποστολή και αξιολόγηση πολλαπλών ερωτημάτων στο Γλωσσικό Μοντέλο κριτή. Επιπλέον, γίνεται χρήση του διακοσμητή (decorator) `@experiment(RagasExperimentResult)` για να εξασφαλιστεί ότι τα αποτελέσματα των μετρικών θα επιστραφούν δομημένα ως αριθμοί κινητής υποδιαστολής (float) στο αντικείμενο τύπου `RagasExperimentResult` (Σχήμα 4.31).

```python
@experiment(RagasExperimentResult)
async def run_ragas_experiment(row: RagasInputRow) -> RagasExperimentResult:
    context_recall_result = await context_recall_metric.ascore(
        user_input=row.user_input,
        retrieved_contexts=row.retrieved_contexts,
        reference=row.reference
    )
    context_precision_result = await context_precision_metric.ascore(
        user_input=row.user_input,
        reference=row.reference,
        retrieved_contexts=row.retrieved_contexts
    )
    faithfulness_result = await faithfulness_metric.ascore(
        user_input=row.user_input,
        response=row.response,
        retrieved_contexts=row.retrieved_contexts
    )
    answer_correctness_result = await answer_correctness_metric.ascore(
        user_input=row.user_input,
        response=row.response,
        reference=row.reference
    )
    return RagasExperimentResult(
        context_recall=float(context_recall_result.value),
        context_precision=float(context_precision_result.value),
        faithfulness=float(faithfulness_result.value),
        answer_correctness=float(answer_correctness_result.value),
    )
```

Σχήμα 4.32: Ασύγχρονη εκτέλεση του πειράματος για τον ταυτόχρονο υπολογισμό των βαθμολογιών ανά ερώτημα, μέσω της συνάρτησης experiment της βιβλιοθήκης Ragas

Αφού ολοκληρωθεί η αξιολόγηση και των 40 ερωτημάτων ενός αρχείου JSON (το οποίο αντιστοιχεί σε μία συγκεκριμένη μέθοδο τεμαχισμού), ο αλγόριθμος δομεί τα αποτελέσματα σε μορφή πινάκων, διευκολύνοντας τον ανθρώπινο έλεγχο. Αρχικά, γίνεται εξαγωγή 6 ξεχωριστών αναλυτικών αρχείων μορφής Excel, ένα για κάθε μέθοδο τεμαχισμού, τα οποία περιέχουν τη βαθμολογία κάθε μεμονωμένης ερώτησης σε όλες τις μετρικές (Σχήμα 4.33). Με αυτό τον τρόπο, δίνεται η δυνατότητα εντοπισμού ερωτήσεων ή θεματικών ενοτήτων στις οποίες το σύστημα πιθανώς υστερεί.

| user_input | retrieved_contexts | response | context_recall | context_precision | faithfulness | answer_correctness |
|---|---|---|---|---|---|---|
| Πώς ακούω | ['[Category: ]\nConte | Μπορείτε να ακούσε | 1 | 0,816666667 | 0,833333333 | 0,331974237 |
| Αν αλλάξω σ | ['[Category: ]\nConte | Λυπάμαι, δεν βρέθη | 0,4 | 0,5 | 0 | 0,586106994 |
| Πώς βρίσκω | ['[Category: ]\nConte | Για να βρείτε το σχολ | 0,25 | 0,915079365 | 1 | 0,333549152 |
| Πώς βλέπω τ | ['[Category: ]\nConte | Για να δείτε τι εκπομ | 1 | 0,930555556 | 0,888888889 | 0,980306646 |
| Γίνεται να βά | ['[Category: ]\nConte | Ναι, μπορείτε να ενε | 0,875 | 0,8125 | 1 | 0,791188195 |
| Πώς βρίσκω | ['[Category: ]\nConte | Μπορείτε να βρείτε ε | 0,8 | 1 | 0,909090909 | 0,407034234 |
| Μπορώ να δ | ['[Category: ]\nConte | Λυπάμαι, δεν βρέθη | 0,75 | 0,961734694 | 0 | 0,085332606 |
| Μπορώ να ψ | ['[Category: ]\nConte | Ναι, μπορείτε να ψάξ | 0,75 | 1 | 0,909090909 | 0,38047008 |
| Πώς μπορώ | ['[Category: ]\nConte | Για να κάνετε αναζήτ | 1 | 0,588888889 | 0,8 | 0,802471146 |
| Εκτός από τα | ['[Category: ]\nConte | Μέσα στην Κοινότητ | 0,428571429 | 1 | 1 | 0,554164933 |
| Τι επιπλέον ά | ['[Category: ]\nConte | Με βάση τα εγχειρίδ | 0,588235294 | 1 | 0,75 | 0,687894664 |
| Είμαι δάσκα | ['[Category: ]\nConte | Για εκπαιδευτικούς α | 1 | 0,966666667 | 0,833333333 | 0,676698604 |
| Πώς μπορώ | ['[Category: ]\nConte | Για να αλλάξετε τη φ | 0,833333333 | 0,958333333 | 1 | 0,816212665 |
| Τι είναι ο Επ | ['[Category: ]\nConte | Σύμφωνα με τα εγχει | 1 | 1 | 1 | 0,973754885 |
| Μπορούν οι | ['[Category: ]\nConte | Ναι, οι μαθητές μπο | 1 | 0,625 | 1 | 0,868133458 |

Σχήμα 4.33: Δείγμα αναλυτικού αρχείου Excel (re_512_deepseek_multilingual.xlsx), στο οποίο αποτυπώνεται η ξεχωριστή βαθμολογία των τεσσάρων μετρικών για κάθε μεμονωμένο ερώτημα

Στη συνέχεια, αφού ολοκληρωθεί η επεξεργασία και των 6 αρχείων JSON, ο αλγόριθμος υπολογίζει αυτόματα τον μέσο όρο κάθε μετρικής, συνολικά για το κάθε αρχείο. Με βάση αυτούς τους τέσσερις μέσους όρους εξάγεται και ένας συνολικός μέσος όρος για το εκάστοτε αρχείο. Στο τελικό στάδιο του script, δημιουργείται ένα συγκεντρωτικό αρχείο μορφής Excel για τον συγκεκριμένο συνδυασμό Γλωσσικού Μοντέλου και Μοντέλου Διανυσματικών Αναπαραστάσεων. Το αρχείο αυτό περιλαμβάνει 6 σειρές, οι οποίες αντιστοιχούν στις 6 μεθόδους, περιέχοντας τους μέσους όρους των τεσσάρων μετρικών και τον συνολικό μέσο όρο για την καθεμία μέθοδο. Επιπλέον, όπως απεικονίζεται στο Σχήμα 4.34, πραγματοποιείται αυτόματη κατάταξη των μεθόδων, από την υψηλότερη τιμή προς τη χαμηλότερη, σύμφωνα με τη στήλη «*ΣΥΝΟΛΙΚΟΣ ΜΕΣΟΣ ΟΡΟΣ*».

| Κατάταξη | Μοντέλα LLM & Embedding | Μέθοδος Chunking | Context Recall | Context Precision | Faithfulness | Answer Correctness | ΣΥΝΟΛΙΚΟΣ ΜΕΣΟΣ ΟΡΟΣ |
|---|---|---|---|---|---|---|---|
| 1 | DEEPSEEK MULTILINGUAL | RE 1024 | 0,83 | 0,85 | 0,62 | 0,56 | 71,38 |
| 2 | DEEPSEEK MULTILINGUAL | RE 512 | 0,79 | 0,79 | 0,71 | 0,56 | 71,32 |
| 3 | DEEPSEEK MULTILINGUAL | SE IQR | 0,82 | 0,85 | 0,6 | 0,55 | 70,15 |
| 4 | DEEPSEEK MULTILINGUAL | RE 256 | 0,8 | 0,72 | 0,69 | 0,57 | 69,63 |
| 5 | DEEPSEEK MULTILINGUAL | SE PERCENTILE | 0,86 | 0,85 | 0,55 | 0,52 | 69,54 |
| 6 | DEEPSEEK MULTILINGUAL | SE STD | 0,77 | 0,86 | 0,54 | 0,52 | 67,32 |

Σχήμα 4.34: Δείγμα συγκεντρωτικού αρχείου Excel (DEEPSEEK_MULTILINGUAL_EVALUATION.xlsx), το οποίο παρουσιάζει την τελική κατάταξη των έξι μεθόδων τεμαχισμού βάσει του συνολικού μέσου όρου

### 4.5.3 Παρουσίαση Συγκεντρωτικών Αποτελεσμάτων

Από την αυτοματοποιημένη διαδικασία εξαγωγής αποτελεσμάτων (Υποενότητα 4.5.2) δημιουργήθηκαν συνολικά 72 αναλυτικά και 12 συγκεντρωτικά αρχεία μορφής Excel. Με στόχο να προκύψουν ασφαλή και αντικειμενικά συμπεράσματα από τα πειράματα, ήταν απαραίτητη η ομαδοποίηση αυτών των δεδομένων, χρησιμοποιώντας μόνο τα 12 συγκεντρωτικά αρχεία.

Αρχικά, η ταξινόμηση των διαφόρων συνδυασμών βασίστηκε στον απλό αριθμητικό μέσο όρο των τεσσάρων μετρικών, ο οποίος υπολογίζεται στο τέλος του αλγορίθμου (στήλη «*ΣΥΝΟΛΙΚΟΣ ΜΕΣΟΣ ΟΡΟΣ*»). Ωστόσο, παρατηρήθηκε ότι ορισμένοι συνδυασμοί, παρόλο που είχαν τον υψηλότερο συνολικό μέσο όρο, δεν παρουσίαζαν αντίστοιχα τις υψηλότερες τιμές και στις τέσσερις μετρικές. Για την ακρίβεια είχαν υψηλή βαθμολογία σε δύο ή τρεις μετρικές καλύπτοντας έτσι τη σχετικά χαμηλή απόδοση τους στις υπόλοιπες. Είναι γεγονός όμως ότι σε ένα αποδοτικό σύστημα RAG το ζητούμενο είναι να βρεθεί μία ισορροπία ανάμεσα σε όλες τις πτυχές του, δηλαδή μεταξύ της ανάκτησης και της παραγωγής κειμένου.

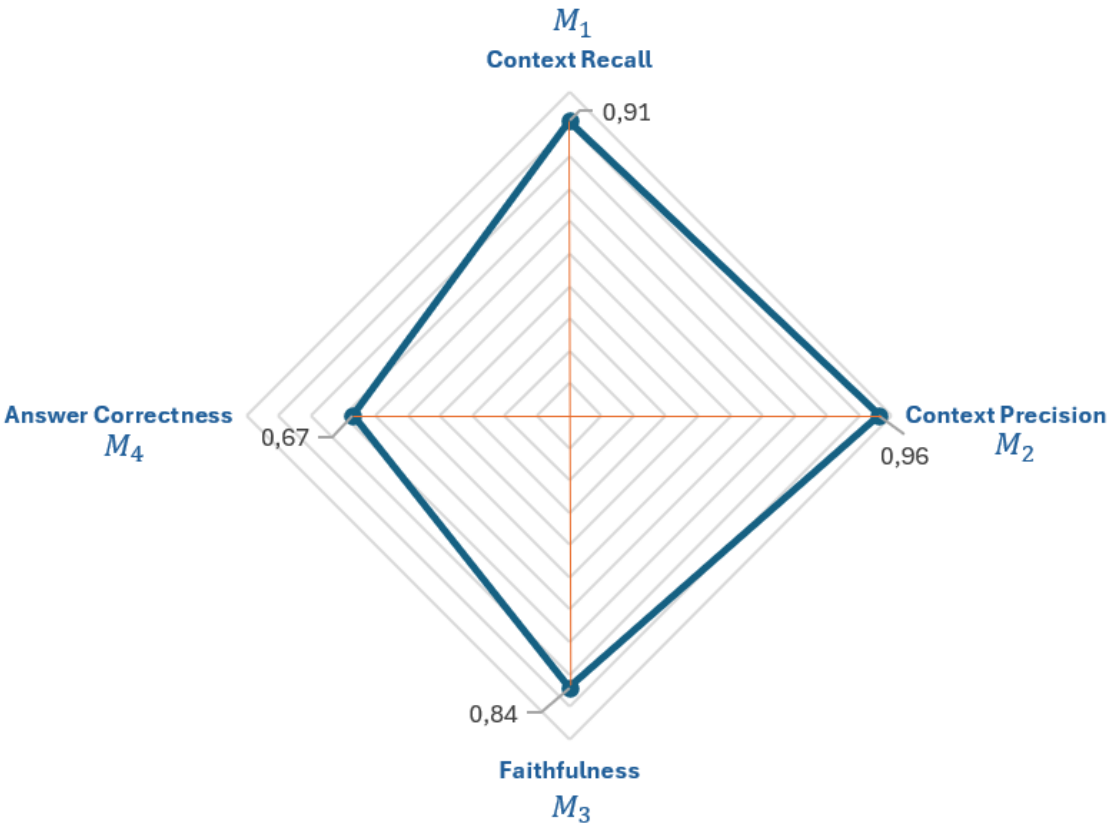
Για την ορθή ταξινόμηση των αποτελεσμάτων, ο απλός μέσος όρος αντικαταστάθηκε από τη γεωμετρική μετρική του Εμβαδού Διαγράμματος Αράχνης / Ραντάρ (Spider Area / Radar Chart Area) [87]. Το πρώτο βήμα για να μπορέσει να αξιοποιηθεί αυτή η μέθοδος είναι να κανονικοποιηθούν οι μετρικές, γεγονός το οποίο εξασφαλίζεται εξαρχής, αφού όλες οι τιμές τους βρίσκονται ήδη στο διάστημα [0,1]. Έπειτα, εφόσον οι μετρικές έχουν ίδια βαρύτητα για τη συνολική βαθμολογία, οι άξονες τοποθετήθηκαν κάθετα μεταξύ τους δημιουργώντας ένα τετράπλευρο (quadrilateral), όπως απεικονίζεται στο Σχήμα 4.35. Η τελική απόδοση του συστήματος υπολογίζεται προσδιορίζοντας το συνολικό εμβαδόν (Area) αυτού του τετράπλευρου, το οποίο προκύπτει από το άθροισμα των τεσσάρων ορθογώνιων τριγώνων που σχηματίζονται, σύμφωνα με τον τύπο (4.1):

$$Area = \frac{M_1 \times M_2}{2} + \frac{M_2 \times M_3}{2} + \frac{M_3 \times M_4}{2} + \frac{M_4 \times M_1}{2} \Rightarrow$$

$$Area = \frac{(M_1 \times M_2) + (M_2 \times M_3) + (M_3 \times M_4) + (M_4 \times M_1)}{2} \quad (4.1)$$

Όπου:

- $M_1$: Context Recall (Ανάκληση Πλαισίου)
- $M_2$: Context Precision (Ακρίβεια Πλαισίου)
- $M_3$: Faithfulness (Πιστότητα)
- $M_4$: Answer Correctness (Ορθότητα Απάντησης)




Σχήμα 4.35: Παράδειγμα γραφικής απεικόνισης των τεσσάρων μετρικών αξιολόγησης σε Διάγραμμα Αράχνης (Radar Chart) για τον υπολογισμό του συνολικού εμβαδού (Area)

Η βαθμολογία Area, για την οποία οι τιμές της κυμαίνονται στο διάστημα [0,2], ευνοεί τα συστήματα RAG που αποδίδουν ομοιόμορφα σε όλες τις μετρικές και κρατάνε μία ισορροπία, ενώ ταυτόχρονα επιβάλλει ποινή σε αυτά που έχουν μεγάλες διακυμάνσεις στις τιμές τους. Επομένως, βάσει αυτής της νέας τιμής έγινε επαναταξινόμηση των αποτελεσμάτων. Τα τελικά αποτελέσματα δομήθηκαν σύμφωνα με δύο διαφορετικές φιλοσοφίες σύγκρισης:

1. **Σταθερό Γλωσσικό Μοντέλο:** Σε αυτή την προσέγγιση τα δεδομένα ομαδοποιήθηκαν με βάση το Γλωσσικό Μοντέλο (δηλαδή σύνολο 4 πίνακες) και παρουσιάζονται αναλυτικά στους Πίνακες 4.9 έως 4.12. Κάθε πίνακας περιλαμβάνει 18 γραμμές, οι οποίες προκύπτουν από τον συνδυασμό του συγκεκριμένου LM με τα 3 μοντέλα Embedding και τις 6 μεθόδους Chunking. Με αυτόν τον τρόπο, επιτυγχάνεται η σύγκριση της συμπεριφοράς των διαφορετικών μοντέλων διανυσματικών αναπαραστάσεων (και των μεθόδων Chunking), διατηρώντας σταθερή την επίδραση του LM.

2. **Σταθερό Μοντέλο Διανυσματικών Αναπαραστάσεων (Embedding):** Σε αυτή την προσέγγιση τα δεδομένα ομαδοποιήθηκαν με βάση το Μοντέλο Διανυσματικών Αναπαραστάσεων (δηλαδή σύνολο 3 πίνακες) και παρουσιάζονται αναλυτικά στους Πίνακες 4.13 έως 4.15. Κάθε πίνακας περιλαμβάνει 24 γραμμές, οι οποίες προκύπτουν από τον συνδυασμό του συγκεκριμένου Embedding με τα 4 Γλωσσικά Μοντέλα και τις 6 μεθόδους Chunking. Με αυτόν τον τρόπο, επιτυγχάνεται η σύγκριση της συμπεριφοράς των διαφορετικών Γλωσσικών Μοντέλων (και των μεθόδων Chunking), διατηρώντας σταθερή την επίδραση του embedding model.

Για την ευκολότερη κατανόηση των πινάκων, τα ονόματα των μοντέλων και των μεθόδων αναγράφονται σύμφωνα με τις συντομογραφίες που ορίστηκαν στην Υποενότητα 4.4, ενώ οι μέγιστες επιδόσεις σε κάθε επιμέρους μετρική έχουν επισημανθεί με έντονη γραφή (bold).

Πίνακας 4.9: Κατάταξη απόδοσης συστήματος RAG για το Γλωσσικό Μοντέλο Llama 3.1 (8B) (llama)

| **Κατάταξη** | **Μοντέλο Embedding** | **Μέθοδος Chunking** | **Context Recall** | **Context Precision** | **Faithfulness** | **Answer Correctness** | **Area** |
|---|---|---|---|---|---|---|---|
| 1 | e5 | re 1024 | 0,87 | **0,96** | 0,64 | 0,52 | 1,1174 |
| 2 | e5 | se iqr | 0,85 | 0,93 | 0,64 | **0,56** | 1,11005 |
| 3 | e5 | se std | 0,87 | 0,93 | **0,65** | 0,48 | 1,0716 |
| 4 | e5 | se percentile | 0,88 | 0,95 | 0,53 | 0,47 | 1,0011 |
| 5 | e5 | re 512 | **0,89** | 0,89 | 0,59 | 0,46 | 0,999 |
| 6 | e5 | re 256 | 0,84 | 0,84 | 0,57 | 0,41 | 0,88125 |
| 7 | multilingual | re 1024 | 0,83 | 0,82 | 0,48 | 0,46 | 0,8384 |
| 8 | greek | re 512 | 0,76 | 0,84 | 0,51 | 0,42 | 0,8001 |
| 9 | greek | re 1024 | 0,68 | 0,86 | 0,51 | 0,44 | 0,7735 |
| 10 | greek | se percentile | 0,74 | 0,85 | 0,48 | 0,4 | 0,7625 |
| 11 | multilingual | se iqr | 0,76 | 0,86 | 0,44 | 0,38 | 0,744 |
| 12 | greek | se std | 0,69 | 0,86 | 0,5 | 0,39 | 0,74375 |
| 13 | multilingual | se percentile | 0,8 | 0,79 | 0,46 | 0,35 | 0,7182 |
| 14 | greek | se iqr | 0,7 | 0,87 | 0,41 | 0,35 | 0,6771 |
| 15 | multilingual | re 256 | 0,74 | 0,61 | 0,54 | 0,44 | 0,672 |
| 16 | multilingual | se std | 0,81 | 0,83 | 0,33 | 0,34 | 0,6669 |
| 17 | greek | re 256 | 0,7 | 0,81 | 0,37 | 0,39 | 0,642 |
| 18 | multilingual | re 512 | 0,7 | 0,7 | 0,43 | 0,43 | 0,63845 |

Πίνακας 4.10: Κατάταξη απόδοσης συστήματος RAG για το Γλωσσικό Μοντέλο Gemma 4 (26B) (gemma)

| Κατάταξη | Μοντέλο Embedding | Μέθοδος Chunking | Context Recall | Context Precision | Faithfulness | Answer Correctness | Area |
|---|---|---|---|---|---|---|---|
| 1 | e5 | re 1024 | 0,76 | **0,97** | 0,88 | **0,69** | 1,3612 |
| 2 | e5 | se iqr | **0,84** | 0,92 | 0,89 | 0,63 | 1,34075 |
| 3 | e5 | se std | 0,82 | 0,95 | 0,91 | 0,6 | 1,34075 |
| 4 | e5 | se percentile | 0,82 | 0,93 | **0,92** | 0,61 | 1,3398 |
| 5 | e5 | re 512 | 0,79 | 0,88 | **0,92** | 0,63 | 1,29105 |
| 6 | e5 | re 256 | 0,78 | 0,84 | 0,9 | 0,63 | 1,2348 |
| 7 | greek | re 512 | 0,73 | 0,85 | 0,86 | 0,58 | 1,13685 |
| 8 | greek | se percentile | 0,74 | 0,87 | 0,84 | 0,52 | 1,0981 |
| 9 | greek | re 256 | 0,71 | 0,8 | 0,87 | 0,56 | 1,0744 |
| 10 | multilingual | se percentile | 0,74 | 0,87 | 0,77 | 0,53 | 1,057 |
| 11 | greek | se std | 0,68 | 0,89 | 0,79 | 0,52 | 1,03635 |
| 12 | greek | se iqr | 0,69 | 0,85 | 0,82 | 0,51 | 1,0268 |
| 13 | multilingual | re 512 | 0,7 | 0,84 | 0,81 | 0,52 | 1,0268 |
| 14 | greek | re 1024 | 0,69 | 0,86 | 0,76 | 0,51 | 0,99325 |
| 15 | multilingual | re 256 | 0,69 | 0,77 | 0,86 | 0,51 | 0,992 |
| 16 | multilingual | re 1024 | 0,7 | 0,9 | 0,69 | 0,46 | 0,9452 |
| 17 | multilingual | se iqr | 0,67 | 0,85 | 0,74 | 0,49 | 0,9447 |
| 18 | multilingual | se std | 0,65 | 0,9 | 0,7 | 0,46 | 0,918 |

Πίνακας 4.11: Κατάταξη απόδοσης συστήματος RAG για το Γλωσσικό Μοντέλο Gemini 2.5 Flash (gemini)

| **Κατάταξη** | **Μοντέλο Embedding** | **Μέθοδος Chunking** | **Context Recall** | **Context Precision** | **Faithfulness** | **Answer Correctness** | **Area** |
|---|---|---|---|---|---|---|---|
| 1 | e5 | re 1024 | 0,9 | **0,95** | **0,87** | **0,7** | 1,46025 |
| 2 | e5 | se std | 0,88 | 0,94 | 0,86 | 0,65 | 1,3833 |
| 3 | e5 | se iqr | 0,89 | 0,93 | 0,85 | 0,64 | 1,3659 |
| 4 | e5 | re 512 | **0,93** | 0,89 | 0,82 | 0,62 | 1,32125 |
| 5 | e5 | se percentile | 0,86 | 0,94 | 0,83 | 0,62 | 1,3182 |
| 6 | e5 | re 256 | 0,84 | 0,83 | **0,87** | 0,6 | 1,22265 |
| 7 | greek | se std | 0,8 | 0,88 | 0,8 | 0,57 | 1,16 |
| 8 | greek | re 256 | 0,86 | 0,8 | 0,83 | 0,57 | 1,15765 |
| 9 | greek | se iqr | 0,79 | 0,85 | 0,81 | 0,57 | 1,136 |
| 10 | greek | se percentile | 0,79 | 0,86 | 0,82 | 0,55 | 1,13505 |
| 11 | greek | re 512 | 0,78 | 0,83 | 0,84 | 0,55 | 1,1178 |
| 12 | greek | re 1024 | 0,76 | 0,84 | 0,74 | 0,58 | 1,065 |
| 13 | multilingual | re 512 | 0,69 | 0,82 | 0,8 | 0,56 | 1,0281 |
| 14 | multilingual | re 256 | 0,68 | 0,78 | 0,81 | 0,54 | 0,9834 |
| 15 | multilingual | se iqr | 0,75 | 0,84 | 0,69 | 0,5 | 0,9648 |
| 16 | multilingual | re 1024 | 0,7 | 0,87 | 0,67 | 0,47 | 0,9179 |
| 17 | multilingual | se percentile | 0,67 | 0,84 | 0,71 | 0,49 | 0,9177 |
| 18 | multilingual | se std | 0,67 | 0,88 | 0,64 | 0,45 | 0,87115 |

Πίνακας 4.12: Κατάταξη απόδοσης συστήματος RAG για το Γλωσσικό Μοντέλο DeepSeek V4 Flash (deepseek)

| **Κατάταξη** | **Μοντέλο Embedding** | **Μέθοδος Chunking** | **Context Recall** | **Context Precision** | **Faithfulness** | **Answer Correctness** | **Area** |
|---|---|---|---|---|---|---|---|
| 1 | e5 | re 1024 | **0,91** | **0,96** | **0,84** | **0,67** | 1,42625 |
| 2 | e5 | se iqr | 0,9 | 0,93 | 0,81 | 0,65 | 1,3509 |
| 3 | e5 | re 512 | 0,89 | 0,89 | 0,8 | **0,67** | 1,3182 |
| 4 | e5 | se percentile | 0,84 | 0,94 | 0,79 | 0,66 | 1,304 |
| 5 | e5 | se std | 0,85 | 0,94 | 0,78 | 0,64 | 1,2877 |
| 6 | e5 | re 256 | 0,86 | 0,85 | 0,78 | 0,63 | 1,2136 |
| 7 | greek | re 256 | 0,79 | 0,8 | 0,74 | 0,62 | 1,0863 |
| 8 | greek | re 512 | 0,77 | 0,81 | 0,76 | 0,6 | 1,07865 |
| 9 | greek | re 1024 | 0,72 | 0,86 | 0,71 | 0,6 | 1,0439 |
| 10 | greek | se iqr | 0,8 | 0,85 | 0,63 | 0,58 | 1,02245 |
| 11 | multilingual | re 1024 | 0,83 | 0,85 | 0,62 | 0,56 | 1,02225 |
| 12 | multilingual | re 512 | 0,79 | 0,79 | 0,71 | 0,56 | 1,0125 |
| 13 | greek | se std | 0,76 | 0,9 | 0,62 | 0,56 | 1,0074 |
| 14 | multilingual | se iqr | 0,82 | 0,85 | 0,6 | 0,55 | 0,994 |
| 15 | greek | se percentile | 0,75 | 0,89 | 0,65 | 0,52 | 0,987 |
| 16 | multilingual | se percentile | 0,86 | 0,85 | 0,55 | 0,52 | 0,96585 |
| 17 | multilingual | re 256 | 0,8 | 0,72 | 0,69 | 0,57 | 0,96105 |
| 18 | multilingual | se std | 0,77 | 0,86 | 0,54 | 0,52 | 0,9039 |

Πίνακας 4.13: Κατάταξη απόδοσης συστήματος RAG για το Embedding Model paraphrase-multilingual-MiniLM-L12-v2 (multilingual)

| **Κατάταξη** | **Μοντέλο Embedding** | **Μέθοδος Chunking** | **Context Recall** | **Context Precision** | **Faithfulness** | **Answer Correctness** | **Area** |
|---|---|---|---|---|---|---|---|
| 1 | gemma | se percentile | 0,74 | 0,87 | 0,77 | 0,53 | 1,057 |
| 2 | gemini | re 512 | 0,69 | 0,82 | 0,8 | 0,56 | 1,0281 |
| 3 | gemma | re 512 | 0,7 | 0,84 | 0,81 | 0,52 | 1,0268 |
| 4 | deepseek | re 1024 | 0,83 | 0,85 | 0,62 | 0,56 | 1,02225 |
| 5 | deepseek | re 512 | 0,79 | 0,79 | 0,71 | 0,56 | 1,0125 |
| 6 | deepseek | se iqr | 0,82 | 0,85 | 0,6 | 0,55 | 0,994 |
| 7 | gemma | re 256 | 0,69 | 0,77 | **0,86** | 0,51 | 0,992 |
| 8 | gemini | re 256 | 0,68 | 0,78 | 0,81 | 0,54 | 0,9834 |
| 9 | deepseek | se percentile | **0,86** | 0,85 | 0,55 | 0,52 | 0,96585 |
| 10 | gemini | se iqr | 0,75 | 0,84 | 0,69 | 0,5 | 0,9648 |
| 11 | deepseek | re 256 | 0,8 | 0,72 | 0,69 | **0,57** | 0,96105 |
| 12 | gemma | re 1024 | 0,7 | **0,9** | 0,69 | 0,46 | 0,9452 |
| 13 | gemma | se iqr | 0,67 | 0,85 | 0,74 | 0,49 | 0,9447 |
| 14 | gemma | se std | 0,65 | **0,9** | 0,7 | 0,46 | 0,918 |
| 15 | gemini | re 1024 | 0,7 | 0,87 | 0,67 | 0,47 | 0,9179 |
| 16 | gemini | se percentile | 0,67 | 0,84 | 0,71 | 0,49 | 0,9177 |
| 17 | deepseek | se std | 0,77 | 0,86 | 0,54 | 0,52 | 0,9039 |
| 18 | gemini | se std | 0,67 | 0,88 | 0,64 | 0,45 | 0,87115 |
| 19 | llama | re 1024 | 0,83 | 0,82 | 0,48 | 0,46 | 0,8384 |
| 20 | llama | se iqr | 0,76 | 0,86 | 0,44 | 0,38 | 0,744 |
| 21 | llama | se percentile | 0,8 | 0,79 | 0,46 | 0,35 | 0,7182 |
| 22 | llama | re 256 | 0,74 | 0,61 | 0,54 | 0,44 | 0,672 |
| 23 | llama | se std | 0,81 | 0,83 | 0,33 | 0,34 | 0,6669 |
| 24 | llama | re 512 | 0,7 | 0,7 | 0,43 | 0,43 | 0,63845 |

Πίνακας 4.14: Κατάταξη απόδοσης συστήματος RAG για το Embedding Model stsb-xlm-r-greek-transfer (greek)

| Κατάταξη | Μοντέλο Embedding | Μέθοδος Chunking | Context Recall | Context Precision | Faithfulness | Answer Correctness | Area |
|---|---|---|---|---|---|---|---|
| 1 | gemini | se std | 0,8 | 0,88 | 0,8 | 0,57 | 1,16 |
| 2 | gemini | re 256 | **0,86** | 0,8 | 0,83 | 0,57 | 1,15765 |
| 3 | gemma | re 512 | 0,73 | 0,85 | 0,86 | 0,58 | 1,13685 |
| 4 | gemini | se iqr | 0,79 | 0,85 | 0,81 | 0,57 | 1,136 |
| 5 | gemini | se percentile | 0,79 | 0,86 | 0,82 | 0,55 | 1,13505 |
| 6 | gemini | re 512 | 0,78 | 0,83 | 0,84 | 0,55 | 1,1178 |
| 7 | gemma | se percentile | 0,74 | 0,87 | 0,84 | 0,52 | 1,0981 |
| 8 | deepseek | re 256 | 0,79 | 0,8 | 0,74 | **0,62** | 1,0863 |
| 9 | deepseek | re 512 | 0,77 | 0,81 | 0,76 | 0,6 | 1,07865 |
| 10 | gemma | re 256 | 0,71 | 0,8 | **0,87** | 0,56 | 1,0744 |
| 11 | gemini | re 1024 | 0,76 | 0,84 | 0,74 | 0,58 | 1,065 |
| 12 | deepseek | re 1024 | 0,72 | 0,86 | 0,71 | 0,6 | 1,0439 |
| 13 | gemma | se std | 0,68 | 0,89 | 0,79 | 0,52 | 1,03635 |
| 14 | gemma | se iqr | 0,69 | 0,85 | 0,82 | 0,51 | 1,0268 |
| 15 | deepseek | se iqr | 0,8 | 0,85 | 0,63 | 0,58 | 1,02245 |
| 16 | deepseek | se std | 0,76 | **0,9** | 0,62 | 0,56 | 1,0074 |
| 17 | gemma | re 1024 | 0,69 | 0,86 | 0,76 | 0,51 | 0,99325 |
| 18 | deepseek | se percentile | 0,75 | 0,89 | 0,65 | 0,52 | 0,987 |
| 19 | llama | re 512 | 0,76 | 0,84 | 0,51 | 0,42 | 0,8001 |
| 20 | llama | re 1024 | 0,68 | 0,86 | 0,51 | 0,44 | 0,7735 |
| 21 | llama | se percentile | 0,74 | 0,85 | 0,48 | 0,4 | 0,7625 |
| 22 | llama | se std | 0,69 | 0,86 | 0,5 | 0,39 | 0,74375 |
| 23 | llama | se iqr | 0,7 | 0,87 | 0,41 | 0,35 | 0,6771 |
| 24 | llama | re 256 | 0,7 | 0,81 | 0,37 | 0,39 | 0,642 |

Πίνακας 4.15: Κατάταξη απόδοσης συστήματος RAG για το Embedding Model multilingual-e5-large (e5)

| **Κατάταξη** | **Μοντέλο Embedding** | **Μέθοδος Chunking** | **Context Recall** | **Context Precision** | **Faithfulness** | **Answer Correctness** | **Area** |
|---|---|---|---|---|---|---|---|
| 1 | gemini | re 1024 | 0,9 | 0,95 | 0,87 | **0,7** | 1,46025 |
| 2 | deepseek | re 1024 | 0,91 | 0,96 | 0,84 | 0,67 | 1,42625 |
| 3 | gemini | se std | 0,88 | 0,94 | 0,86 | 0,65 | 1,3833 |
| 4 | gemini | se iqr | 0,89 | 0,93 | 0,85 | 0,64 | 1,3659 |
| 5 | gemma | re 1024 | 0,76 | **0,97** | 0,88 | 0,69 | 1,3612 |
| 6 | deepseek | se iqr | 0,9 | 0,93 | 0,81 | 0,65 | 1,3509 |
| 7 | gemma | se iqr | 0,84 | 0,92 | 0,89 | 0,63 | 1,34075 |
| 8 | gemma | se std | 0,82 | 0,95 | 0,91 | 0,6 | 1,34075 |
| 9 | gemma | se percentile | 0,82 | 0,93 | **0,92** | 0,61 | 1,3398 |
| 10 | gemini | re 512 | **0,93** | 0,89 | 0,82 | 0,62 | 1,32125 |
| 11 | deepseek | re 512 | 0,89 | 0,89 | 0,8 | 0,67 | 1,3182 |
| 12 | gemini | se percentile | 0,86 | 0,94 | 0,83 | 0,62 | 1,3182 |
| 13 | deepseek | se percentile | 0,84 | 0,94 | 0,79 | 0,66 | 1,304 |
| 14 | gemma | re 512 | 0,79 | 0,88 | **0,92** | 0,63 | 1,29105 |
| 15 | deepseek | se std | 0,85 | 0,94 | 0,78 | 0,64 | 1,2877 |
| 16 | gemma | re 256 | 0,78 | 0,84 | 0,9 | 0,63 | 1,2348 |
| 17 | gemini | re 256 | 0,84 | 0,83 | 0,87 | 0,6 | 1,22265 |
| 18 | deepseek | re 256 | 0,86 | 0,85 | 0,78 | 0,63 | 1,2136 |
| 19 | llama | re 1024 | 0,87 | 0,96 | 0,64 | 0,52 | 1,1174 |
| 20 | llama | se iqr | 0,85 | 0,93 | 0,64 | 0,56 | 1,11005 |
| 21 | llama | se std | 0,87 | 0,93 | 0,65 | 0,48 | 1,0716 |
| 22 | llama | se percentile | 0,88 | 0,95 | 0,53 | 0,47 | 1,0011 |
| 23 | llama | re 512 | 0,89 | 0,89 | 0,59 | 0,46 | 0,999 |
| 24 | llama | re 256 | 0,84 | 0,84 | 0,57 | 0,41 | 0,88125 |

# Κεφάλαιο 5ο: Συμπεράσματα και Μελλοντικές Βελτιώσεις

## 5.1 Βασικά Συμπεράσματα

Έχοντας ολοκληρώσει πλήρως την πειραματική διαδικασία, μετά και την αξιολόγηση μέσω του πλαισίου Ragas, υπάρχει πλέον η δυνατότητα να εξαχθούν ασφαλή και αντικειμενικά συμπεράσματα. Το πιο βασικό από αυτά είναι η ιδανική επιλογή αρχιτεκτονικής του συστήματος RAG, για τη συγκεκριμένη εφαρμογή με τον Οδηγό Χρήσης του European School Radio.

Στον Πίνακα 5.1 παρουσιάζεται η συγκεντρωτική κατάταξη των 6 νικητήριων συνδυασμών. Είναι σημαντικό να σημειωθεί ότι ο πίνακας αυτός αντιπροσωπεύει τους νικητές από κάθε επιμέρους σύγκριση που έγινε στην Υποενότητα 4.5.3. Με άλλα λόγια, είναι οι νικητήριοι συνδυασμοί για τους Πίνακες 4.9 έως 4.15, για τους οποίους πραγματοποιήθηκε ανακατάταξη σύμφωνα με τη μετρική Area.

Πίνακας 5.1: Συγκεντρωτική κατάταξη των 6 νικητήριων συνδυασμών από τις επιμέρους συγκριτικές δοκιμές, βάσει της συνολικής μετρικής Area

| Κατάταξη | Γλωσσικό Μοντέλο | Μοντέλο Embedding | Μέθοδος Chunking | Context Recall | Context Precision | Faithfulness | Answer Correctness | Area |
|---|---|---|---|---|---|---|---|---|
| 1 | gemini | e5 | re 1024 | 0,9 | 0,95 | 0,87 | **0,7** | 1,46025 |
| 2 | deepseek | e5 | re 1024 | **0,91** | 0,96 | 0,84 | 0,67 | 1,42625 |
| 3 | gemma | e5 | re 1024 | 0,76 | **0,97** | **0,88** | 0,69 | 1,3612 |
| 4 | gemini | greek | se std | 0,8 | 0,88 | 0,8 | 0,57 | 1,16 |
| 5 | llama | e5 | re 1024 | 0,87 | 0,96 | 0,64 | 0,52 | 1,1174 |
| 6 | gemma | multilingual | se percentile | 0,74 | 0,87 | 0,77 | 0,53 | 1,057 |

Αρχικά, είναι σαφές ότι το μοντέλο `multilingual-e5-large` είχε την απόλυτη κυριαρχία μεταξύ των μοντέλων διανυσματικών αναπαραστάσεων, καθώς και στους 4 πίνακες με σταθερό το LM (Πίνακες 4.9 έως 4.12), κατέκτησε πάντα τις πρώτες 6 θέσεις. Εξασφαλίζοντας σταθερά εξαιρετικά υψηλές τιμές στον τομέα της ανάκτησης κειμένου (Retrieval), τόσο στην ανάκτηση πλαισίου (Context Recall: 0.76 - 0.93), όσο και στην ακρίβεια (Context Precision: 0.84 - 0.97). Παράλληλα, για τα μοντέλα `paraphrase-multilingual-MiniLM-L12-v2` και `stsb-xlm-r-greek-transfer` στον τομέα της ανάκτησης η μέγιστη επίδοσή τους έφτανε το πολύ τις χαμηλότερες τιμές του `multilingual-e5-large`, γεγονός που κάνει τη διαφορά μεταξύ τους να είναι αρκετά σημαντική. Αποδεικνύεται, λοιπόν, ότι το `multilingual-e5-large` αντιλαμβάνεται βέλτιστα τη σημασιολογία της ελληνικής γλώσσας και είναι σίγουρα η καλύτερη επιλογή από τη σύγκριση των μοντέλων διανυσματικών αναπαραστάσεων για τον Οδηγό Χρήσης του European School Radio.

Ένα ακόμη συμπέρασμα που συνάγεται, κυρίως μέσω του Πίνακα 5.1, αφορά τις μεθόδους τεμαχισμού κειμένου. Είναι εμφανής η υπεροχή του αναδρομικού τεμαχισμού με μέγιστο όριο τα 1024 tokens (Recursive Chunking max 1024 tokens), καθώς και για τα τέσσερα Γλωσσικά Μοντέλα κατακτά πάντα την πρώτη θέση (συνδυαστικά με το `multilingual-e5-large`). Η συγκεκριμένη μέθοδος παρείχε το ιδανικό μέγεθος πλαισίου (context) στα τμήματα κειμένου (chunks) με τα οποία τροφοδοτούσε το LM, καθώς ούτε χανόταν σημαντική πληροφορία, ούτε υπήρχαν περιττά δεδομένα. Επιπλέον, είναι

αξιοσημείωτο ότι οι σημασιολογικές μέθοδοι τεμαχισμού έχουν αρκετά ισχυρή παρουσία, καθώς στις περισσότερες περιπτώσεις ήταν στην αμέσως επόμενη θέση κατάταξης. Έτσι, προκύπτει ότι ο διαχωρισμός βάσει νοήματος αποτελεί μία ανταγωνιστική και εξίσου αξιόλογη προσέγγιση.

Όσον αφορά τα Γλωσσικά Μοντέλα, από τη σύγκριση μεταξύ τους συμπεραίνεται ότι η κορυφαία επιλογή είναι το Gemini 2.5 Flash. Όπως φαίνεται στον Πίνακα 5.1, επιτυγχάνει τις υψηλότερες τιμές στις μετρικές της ορθότητας απάντησης (Answer Correctness: 0.70) και του Εμβαδού Διαγράμματος Αράχνης (Area: 1.46025). Επομένως, παρουσιάζει την καλύτερη ισορροπία μεταξύ ανάκτησης και παραγωγής κειμένου, γεγονός που αποτελεί ένα από τα πιο σημαντικά κριτήρια που θα καθορίσουν την τελική επιλογή. Ωστόσο, τα Γλωσσικά Μοντέλα DeepSeek V4 Flash και Gemma 4:26b έχουν εξίσου ικανοποιητικές επιδόσεις, οι οποίες τα καθιστούν αρκετά ανταγωνιστικές επιλογές. Πιο συγκεκριμένα, με βάση τον Πίνακα 5.1, το DeepSeek V4 Flash κατέγραψε την υψηλότερη τιμή στην ανάκτηση πλαισίου (Context Recall: 0.91), γεγονός που σημαίνει ότι είναι εξαιρετικά ικανό στον εντοπισμό της σωστής πληροφορίας στα chunks. Παράλληλα, το Gemma 4:26b σημείωσε τις υψηλότερες τιμές στην ακρίβεια (Context Precision: 0.97) και στην πιστότητα (Faithfulness: 0.88), υποδηλώνοντας έτσι ότι είναι λιγότερο επιρρεπές σε «παραισθήσεις» (hallucinations), αφού βασίζεται αυστηρά στα δεδομένα που του παρέχονται. Συμπληρωματικά, ένα αρκετά ασφαλές και αναμενόμενο συμπέρασμα που εξάγεται είναι ότι το μοντέλο Llama 3.1:8b αποδείχθηκε το πιο αδύναμο μοντέλο, καθώς κατείχε πάντα τις τελευταίες θέσεις με σημαντικά χαμηλές τιμές στην ορθότητα απάντησης.

Συνοψίζοντας, από καθαρά τεχνική άποψη με βάση το υψηλότερο συνολικό Εμβαδόν (Area: 1.46025), η βέλτιστη αρχιτεκτονική για την υλοποίηση του ψηφιακού βοηθού του European School Radio είναι ο πρώτος συνδυασμός του Πίνακα 5.1. Πιο συγκεκριμένα, είναι η επιλογή του Gemini 2.5 Flash συνδυαστικά με το μοντέλο διανυσματικών αναπαραστάσεων `multilingual-e5-large` και αναδρομική μέθοδο τεμαχισμού με μέγιστο όριο τα 1024 tokens (Recursive Chunking max 1024 tokens). Ωστόσο, η τελική επιλογή πρέπει να συνυπολογίσει το γεγονός ότι το σύστημα RAG πρέπει να ενσωματωθεί στην παραγωγική λειτουργία του European School Radio. Αυτό σημαίνει ότι πρέπει να βρεθεί μία ισορροπία μεταξύ κόστους και απόδοσης για να είναι πιο ρεαλιστική η λύση.

Εξετάζοντας τον αμέσως επόμενο συνδυασμό, τη δεύτερη θέση στον Πίνακα 5.1, παρατηρείται ότι η απόκλιση στη συνολική απόδοση είναι πολύ μικρή, με τη διαφορά στο Area να είναι μόλις 0.034. Στην πραγματικότητα, το DeepSeek έχει ελαφρώς μεγαλύτερες τιμές στις μετρικές ανάκτησης (Context Recall και Context Precision), ενώ η διαφορά του στις υπόλοιπες είναι ελάχιστη. Το πιο σημαντικό, όμως, είναι η μεγάλη διαφορά που έχουν στο λειτουργικό κόστος χρήσης (API pricing), καθώς το DeepSeek έχει δραματικά χαμηλότερο κόστος συγκριτικά με το Gemini. Συμπερασματικά, λαμβάνοντας υπόψη όλα τα παραπάνω, για τη συγκεκριμένη περίπτωση χρήσης, η καλύτερη επιλογή είναι το Μεγάλο Γλωσσικό Μοντέλο DeepSeek V4 Flash, μαζί με το μοντέλο διανυσματικών αναπαραστάσεων `multilingual-e5-large` και την αναδρομική μέθοδο τεμαχισμού με μέγιστο όριο τα 1024 tokens (Recursive Chunking max 1024 tokens).

## 5.2 Περιορισμοί της Έρευνας

Αξίζει να επισημανθεί ότι, παρά τη λεπτομερή έρευνα που πραγματοποιήθηκε, υπήρξαν ορισμένοι περιορισμοί που αφορούσαν κυρίως το λειτουργικό κόστος. Αρχικά, όπως αναφέρθηκε ήδη, η παραγωγή των απαντήσεων έγινε με τη χρήση γλωσσικών μοντέλων που υπήρχαν στη διάθεση του European School Radio, τα οποία παρόλο που είναι ισχυρά, ανήκουν στις ελαφριές και γρήγορες εκδόσεις που υπάρχουν διαθέσιμες. Η χρήση πιο εξελιγμένων μοντέλων θα έδινε σίγουρα μια διαφορετική οπτική στον τρόπο παραγωγής απαντήσεων, προσφέροντας πιθανώς σημαντικές

βελτιώσεις. Αντίστοιχα, στην αξιολόγηση του συστήματος, η χρήση ενός ισχυρότερου μοντέλου στη θέση του αντικειμενικού κριτή, ενδεχομένως, να παρέχει μια πιο ακριβή σύγκριση των αποτελεσμάτων μεταξύ τους. Με αυτόν τον τρόπο, θα αποκτηθεί μια ξεκάθαρη και πιο σφαιρική εικόνα για το μέγιστο της απόδοσης που μπορεί να φτάσει το σύστημα RAG στη συγκεκριμένη πλατφόρμα. Επιπλέον, το Σύνολο Δεδομένων Αξιολόγησης (Golden Dataset), ήταν μικρό σε μέγεθος, δηλαδή σε πλήθος ερωτήσεων, γεγονός που αποτελεί περιορισμό ως προς τη στατιστική εγκυρότητα των συγκρίσεων που πραγματοποιήθηκαν. Χρησιμοποιώντας περισσότερες ερωτήσεις διαβαθμισμένης δυσκολίας, η αξιολόγηση θα ήταν πιο ολοκληρωμένη με αξιόπιστα και ασφαλή συμπεράσματα.

## 5.3 Προτάσεις για Μελλοντικές Βελτιώσεις

Παρόλο που η παρούσα διπλωματική εργασία κατέληξε σε ένα αξιόπιστο και αποδοτικό σύστημα RAG για τον Ψηφιακό Βοηθό της πλατφόρμας του European School Radio, οι συνεχείς εξελίξεις στους τομείς της Τεχνητής Νοημοσύνης και της Μηχανικής Μάθησης αφήνουν σημαντικά περιθώρια βελτίωσης για το μέλλον. Προς αυτή την κατεύθυνση, υπάρχουν ορισμένες προτάσεις που στοχεύουν στη συνεχή βελτίωση της εμπειρίας του χρήστη και στη συνολική αναβάθμιση της απόδοσης του συστήματος.

Δεδομένου ότι η πλατφόρμα απευθύνεται σε κοινό όλων των ηλικιών με διαφορετική εξοικείωση με το περιβάλλον της ιστοσελίδας, θα αποτελούσε σημαντική ενίσχυση η οπτική υποστήριξη των χρηστών. Πιο συγκεκριμένα, θα ήταν χρήσιμη η προβολή εικόνων (π.χ. η τοποθεσία ενός συγκεκριμένου κουμπιού) μέσα στην ιστοσελίδα, έτσι ώστε να διευκολυνθεί η πλοήγηση του χρήστη. Παράλληλα, είναι αρκετά κρίσιμη η ενσωμάτωση της μνήμης συνομιλίας (conversational memory), για να μπορεί το Γλωσσικό Μοντέλο να απαντάει σε συμπληρωματικές (follow-up) ερωτήσεις του χρήστη με την ίδια αποδοτικότητα χωρίς να παρουσιάζει «παραισθήσεις» (hallucinations). Το γεγονός αυτό αποτελεί αρκετά ρεαλιστικό σενάριο, καθώς οι χρήστες πολλές φορές δεν κατανοούν πλήρως την απάντηση ή χρειάζονται επιπλέον διευκρίνιση.

Προκειμένου να βελτιωθεί η εμπειρία χρήσης, πρέπει οι τελικοί χρήστες (μαθητές, καθηγητές και επισκέπτες) να έχουν ενεργή θέση στην κρίση του τελικού αποτελέσματος. Το επόμενο βήμα, λοιπόν, είναι η συμπλήρωση ερωτηματολογίων από τους χρήστες για την αξιολόγηση της εμπειρίας τους με τον Ψηφιακό Βοηθό της πλατφόρμας. Αυτός ο τομέας μπορεί να ενισχυθεί με την προσθήκη ενός μηχανισμού ανατροφοδότησης (feedback), όπως κουμπιά θετικής/αρνητικής ψήφου (like/dislike), που να εμφανίζεται σε κάθε απάντηση του Chatbot. Μέσω αυτής της διαδικασίας, είναι δυνατό να γίνει συλλογή δεδομένων και νέων ερωτήσεων που δεν υπάρχουν ήδη στο Σύνολο Δεδομένων Αξιολόγησης (Golden Dataset), με στόχο τον περαιτέρω εμπλουτισμό του. Ταυτόχρονα, μπορούν να προστεθούν ερωτήσεις μεγαλύτερης δυσκολίας και πολυπλοκότητας σε αυτό, για να γίνει αναλυτικότερη αξιολόγηση του συστήματος RAG σε πραγματικές συνθήκες.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


[1] M. A. Boden, *Artificial Intelligence*. San Diego, CA: Academic Press, 1996.

[2] E. Wolfgang, Introduction to Artificial Intelligence, Hochschule Ravensburg-Weingarten, 2025.

[3] J. Yin, T. T. Goh, B. Yang, and Y. Xiaobin, "Conversation technology with micro-learning: the impact of chatbot-based learning on students' learning motivation and performance," *Journal of Educational Computing Research*, vol. 59, no. 1, pp. 154-177, Mar. 2021.

[4] R. Khan and A. Das, *Build Better Chatbots: A Complete Guide to Getting Started with Chatbots*. Bangalore, Karnataka, India: Apress, 2018.

[5] B. P. Wicaksono and A. Zahra, "Design of the use of chatbot as a virtual assistant in banking services in Indonesia," IAES International Journal of Artificial Intelligence (IJ-AI), vol. 11, no. 1, pp. 23-33, Mar. 2022.

[6] K. Wang, "From ELIZA to ChatGPT: A brief history of chatbots and their evolution," in *Proc. 2023 International Conference on Machine Learning and Automation*, 2023, pp. 57-62.

[7] K. Mainzer, *Artificial intelligence – When do machines take over?* Cham, Germany: Springer, 2020.

[8] G. Hald, "A Brief History of Chatbots," BotSupply Technical Report, Tech. Rep. 2018-05, May 2018.

[9] M. Al-Amin *et al.*, "History of generative Artificial Intelligence (AI) chatbots: past, present, and future development," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 1, pp. 23-33, Mar. 2022.

[10] E. Adamopoulou and L. Moussiades, "The history of chatbots: the journey from psychological experiment to educational object," *Machine Learning with Applications*, vol. 2, p. 100006, Dec. 2020.

[11] Harliantara, D. J. Sompie, and I. M. Sutika, "Radio Broadcasting with Artificial Intelligence: A Case Study on Radio Mustang Jakarta," Communicatus: Jurnal Ilmu Komunikasi, vol. 8, no. 1, pp. 121-138, Jun. 2024.

[12] Radio Centre Ireland, "How AI is improving audio by enhancing creativity, and offering new opportunities for personalisation," Radio Centre Ireland, 2024. [Online]. Available: https://www.radiocentreireland.ie/news/how-ai-is-improving-audio-by-enhancing-creativity-and-offering-new-opportunities-for-personalisation.

[13] L. Rowe, "RadioGPT: Should You Be Worried About AI Radio Hosts?", Radio.co, Apr. 30, 2025. [Online].

[14] N. Patel, D. Parikh, D. Patel, and R. Patel, "AI and Web-Based Interactive College Enquiry Chatbot," *International Journal of Electronics, Communication and Aerospace Technology*, vol. 3, no. 2, pp. 148-150, Jun. 2019.

[15] W. El Hefny, Y. Mansy, M. Abdallah, and S. Abdennadher, "Components of Smart Chatbot Academic Model for a University Website," *International Journal of Computer Applications in Higher Education*, vol. 12, no. 4, pp. 671-681, 2021.

[16] P. Gidzon, “Digital Assistant to Improve User Experience on Websites: Innovative Support and Efficient Communication Strategies,” *Journal of Web Interaction and User Experience*, vol. 5, no. 2, pp. 112-124, 2021.

[17] N. J. P. Bondad, C. J. U. Sin, K. M. R. Estalero, and A. M. Jain, “E-AccesstoAces: A Web-Based Customer Support System Driving Academy Using Chatbot Technology,” Isabela State University Linker: Journal of Engineering, Computing, and Technology, vol. 1, no. 1, pp. 77-88, Jun. 2024.

[18] R. Jegadeesan, “SHOPASSIST: A PERSONALIZED CUSTOMER SUPPORT CHATBOT WITH MACHINE LEARNING-POWERED INTENT RECOGNITION AND QUERY RESOLUTION,” European Chemical Bulletin, vol. 13, no. S3, pp. 6004-6012, 2025.

[19] A. M. Mhadaye, “A STUDY ON THE IMPACT OF AI CHATBOTS ON CONSUMER TRUST AND CONFIDENCE IN PURCHASE DECISIONS,” Aarhat Multidisciplinary International Education Research Journal, vol. XV, no. 1, pp. 171-177, Jan.-Feb. 2026.

[20] S. Bhattacharya and B. Singla, “Exploring the Secrets Behind Chatbot Success in Modern Banking: A Systematic Literature Review,” *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 13, no. 1, pp. 255-269, Mar. 2024.

[21] N. M. Z. Almira, A. F. Ramadhani, E. N. Rahman, and J. S. Justianto, “CHATBOT ADOPTION IN BANKING: ENHANCING CUSTOMER EXPERIENCE AND SATISFACTION IN INDONESIA - JAKARTA URBAN AREA,” Journal of Theoretical and Applied Information Technology, vol. 103, no. 5, pp. 1838-1850, Mar. 2025.

[22] J. M. Kothari, “The Effect of Chatbots on Managing Brand Reputation among Commercial Banks in Kenya,” International Journal of Bank Marketing, vol. 42, no. 3, pp. 530-545, Feb. 2024.

[23] A. T. Joseph, “An Attempt to Improve the Efficiency of Chat Bots by Providing Multilingual Support,” in Proc. National Conference on Emerging Computer Applications (NCECA)-2023, 2023, pp. 313-316.

[24] M. V. Vasileiou, “AI POWERED CHATBOT IN HEALTH INSURANCE,” International Journal of Healthcare Engineering, vol. 14, no. 2, pp. 102-115, Oct. 2024.

[25] European School Radio, “European School Radio: Το Πρώτο Μαθητικό Ραδιόφωνο,” European School Radio, 2026. [Online]. Available: https://europeanschoolradio.eu/el.

[26] Κ. Διαμαντάρας, Δ. Μπότσης, *Μηχανική Μάθηση.* Αθήνα: Εκδόσεις Κλειδάριθμος, Ιούλιος 2019.

[27] N. B. Korade, M. B. Salunke, A. A. Bhosle, G. G. Asalkar, B. Lal, and P. B. Kumbharkar, “Elevating intelligent voice assistant chatbots with natural language processing, and OpenAI technologies,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 37, no. 1, pp. 507-517, Jan. 2025.

[28] R. Mukesh, M. Rahim, S. V. K. Reddy, R. J. Mohan, and S. Saravana Kumar, “Customer Support Chatbot with Machine Learning,” International Journal for Multidisciplinary Research (IJFMR), vol. 6, no. 6, pp. 1-4, Nov.-Dec. 2024.

[29] A. G. Usigan, M. I. Salomeo, G. J. L. J. Zafe, C. J. Centeno, A. A. R. C. Sison, and A. G. Bitancor, “Implementation of an Undergraduate Admission Chatbot Using Microsoft Azure's Question Answering and Bot Framework,” in Proc. 2022 5th Artificial Intelligence and Cloud Computing Conference (AICCC), Osaka, Japan, 2022, pp. 240-245.

[30] D. E. Kyzyrov and S. Z. Sapakova, “CHATBOT ASSISTANTS: IMPLEMENTATION AND ANALYSIS OF THE EFFICIENCY,” *Journal of Problems in Computer Science and Information Technologies*, vol. 2, no. 1, pp. 25-41, 2023.

[31] Distillery, “AI Chatbot Development Services & AI Agents in 2026: The Enterprise Guide,” Distillery Blog, 2026. [Online]. Available: https://distillery.com/blog/ai-chatbot-development-services-ai-agents-development-company/.

[32] “AI Cybersecurity Awareness Agents: Automating Threat Education with LLMs and No-Code Workflows,” Technical Report, May 2025.

[33] Groovy Web, “How to Build an AI Chatbot in 2026: From Concept to Production,” Groovy Web Blog, 2026. [Online]. Available: https://www.groovyweb.co/blog/how-to-build-ai-chatbot-2026.

[34] AgileEngine, “AI chatbot development services: a practical guide for businesses,” AgileEngine Blog, 2026. [Online]. Available: https://agileengine.com/ai-chatbot-development-guide/.

[35] Y. Gao *et al.*, “Retrieval-Augmented Generation for Large Language Models: A Survey,” in *Proc. 2023 International Conference on Machine Learning and Automation*, 2023, pp. 1-18

[36] Google Research, “A differentially private framework for gaining insights into AI chatbot use,” Google Research Blog, 2026. [Online]. Available: https://research.google/blog/a-differentially-private-framework-for-gaining-insights-into-ai-chatbot-use/.

[37] LushBinary, “Gemma 4 vs Llama 4 vs Qwen 3.5: 2026 Comparison Guide,” LushBinary Blog, 2026. [Online]. Available: https://lushbinary.com/blog/gemma-4-vs-llama-4-vs-qwen-3-5-open-weight-model-comparison/.

[38] Kanerika, “10 Best Open Source LLMs to Evaluate in 2026,” Kanerika Blogs, 2026. [Online]. Available: https://kanerika.com/blogs/open-source-llms/.

[39] A. Adak, “Gemma 4 Complete Guide 2026, Architecture, Benchmarks, Deployment and more,” DEV Community, 2026. [Online]. Available: https://dev.to/aniruddhaadak/gemma-4-complete-guide-2026-architecture-benchmarks-deployment-3en9.

[40] Google AI for Developers, “Gemma 4 model card,” Google AI Documentation, 2026. [Online]. Available: https://ai.google.dev/gemma/docs/core/model_card_4.

[41] Milvus, “What are real-world Llama 4 RAG use cases in April 2026?,” Milvus AI Quick Reference, Apr. 2026. [Online]. Available: https://milvus.io/ai-quick-reference/what-are-realworld-llama-4-rag-use-cases-in-april-2026.

[42] MyNextDeveloper, “Gemma 4 vs. Llama 4: Which “Open” Model Actually Wins in 2026?,” MyNextDeveloper Blogs, 2026. [Online]. Available: https://mynextdeveloper.com/blogs/gemma-4-vs-llama-4-which-open-model-actually-wins-in-2026/.

[43] TechJack Solutions, “Llama for Enterprise: Complete Deployment Guide (2026),” TechJack Solutions AI Tools, 2026. [Online]. Available: https://techjacksolutions.com/ai-tools/meta-llama/llama-for-enterprise/.

[44] Meta-Intelligence, “Small Language Models: Phi-4 vs Gemma 3 vs Llama 3.3 — Enterprise Edge AI [2026],” Meta-Intelligence Insights, 2026. [Online]. Available: https://www.meta-intelligence.tech/en/insight-slm-enterprise.

[45] DeepSeek-AI, “DeepSeek-V3 Technical Report,” arXiv preprint arXiv:2412.19437, 2024.

[46] Chat-Deep.ai, “DeepSeek vs Llama: Which AI Model Should You Choose in 2026?,” Chat-Deep.ai, 2026. [Online]. Available: https://www.chat-deep.ai.

[47] Gemini Team, Google, “Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context,” arXiv preprint arXiv:2403.05530, 2024.

[48] Build Fast with AI, “Gemini in Google Workspace: Every Feature Explained (2026),” Build Fast with AI, 2026. [Online]. Available: https://www.buildfastwithai.com.

[49] Atlan, “What Are Embeddings in AI? How They Power Search and RAG [2026],” Atlan Knowledge Base, 2026. [Online]. Available: https://atlan.com/know/what-are-embeddings-ai-search/.

[50] Encord, “Complete Guide to Embeddings in 2026,” Encord Blog, 2026. [Online]. Available: https://encord.com/blog/complete-guide-to-embeddings-in-2026/.

[51] Instaclustr, “How vector embeddings work, common applications, and best practices,” Instaclustr Data Infrastructure Education, 2026. [Online]. Available: https://www.instaclustr.com/education/vector-database/how-vector-embeddings-work-common-applications-and-best-practices/.

[52] Couchbase, “What is Semantic Search? The Definitive Guide,” The Couchbase Blog, 2026. [Online]. Available: https://www.couchbase.com/blog/what-is-semantic-search/.

[53] V. Oliinyk and P. Ponochovnyy, “EXPERIMENTAL EVALUATION OF RAG COMPONENTS AND THEIR IMPACT ON THE PERFORMANCE OF CUSTOMER SUPPORT CHATBOTS,” Adaptive Systems of Automatic Control, no. 1, pp. 20-29, 2026.

[54] Θ. Κατσίκας, “Design and Implementation of a Dynamic E-Learning Platform for Personalized Education Using Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs),” Μεταπτυχιακή Διατριβή, Τμήμα Πληροφορικής, Πανεπιστήμιο Πειραιώς, Ιούλιος 2025.

[55] P. Lewis et al., “Retrieval-Augmented Generation for knowledge-intensive NLP tasks,” NeurIPS, vol. 33, pp. 9459-9474, 2020.

[56] V. Oliinyk, “Development of a Retrieval-Augmented Generation (RAG) Chatbot,” in Proc. International Conference on Security, Fault Tolerance, Intelligence (ICSFTI 2024), Kyiv, Ukraine, 2024, pp. 1-13.

[57] J. Hladěna et al., “Reconstructing Context: Evaluating Advanced Chunking Strategies for Retrieval-Augmented Generation,” in Proceedings of the 14th Computer Science Online Conference 2025, Volume 2, Springer, 2025, pp. 317-325.

[58] Chitika, “Best AI Chatbot for Government Agencies in 2026: A Buyer's Guide,” Chitika, 2026. [Online]. Available: https://www.chitika.com/best-ai-chatbot-for-government-agencies-in-2026-a-buyers-guide/.

[59] Dataforest, “Changes to RAG in 2026 – Better LLM Integration,” Dataforest Blog, 2026. [Online]. Available: https://dataforest.ai/blog/rag-in-2025-smarter-retrieval-and-real-time-responses.

[60] N. Ots and P. Taremaa, “Chunking an unfamiliar language: Results from a perception study of German listeners,” in Prosodic Boundary Phenomena, F. Schubö, S. Zerbian, S. Hanne, and I. Wartenburger, Eds. Berlin, Germany: Language Science Press, 2023, ch. 3, pp. 87-117.

[61] S. T. Ngandoh, R. Riandi, A. Rahmat, M. Muslim, E. Candrawati, and M. Dirham, “Chunking Techniques to Enhance Learning Outcomes in the Human Body System,” International Journal of Recent Educational Research (IJORER), vol. 6, no. 1, pp. 1-12, Jan. 2025.

[62] M. Stäbler, S. Turnbull, T. Müller, C. S. Langdon, J. Marx-Gómez, and F. Köster, “The Impact of Chunking Strategies on Domain-Specific Information Retrieval in RAG Systems,” in Proc. 2025 IEEE International Conference on Omni-layer Intelligent Systems (COINS), 2025, pp. 1-6.

[63] A. Afzal, J. Vladika, G. Fazlija, A. Staradubets, and F. Matthes, “Evaluating Advanced Chunking Strategies for Retrieval-Augmented Generation,” arXiv preprint arXiv:2501.11401, 2025.

[64] N. Mohapatra, N. Sarraf, and S. S. Sahu, “ENSEMBLE MODEL FOR CHUNKING,” *Computer Science & Information Technology*, vol. 11, no. 8, pp. 113-119, 2021.

[65] K. Kimura, “Chunking as a function of sequence length,” in Proc. IEEE International Conference on Wireless Communications '24, 2024, pp. 45-48.

[66] M. J. Ruttanakorn, “Digitally Enhanced Chunk & Check Learning: An Innovative, Instructor-Friendly Approach Powered by an Open-Source Tool for Effective Laboratory Instruction and Formative Assessment,” International Journal of Learning, Teaching and Educational Research, vol. 24, no. 12, pp. 1-18, Dec. 2025.

[67] S. Sreeni, “RAG Chunking Strategies,” DEV Community, 2026. [Online]. Available: https://dev.to/sreeni5018/rag-chunking-strategies-4i3a.

[68] AtlasSC, “Text Chunking Strategies for RAG: A Comprehensive Guide,” AtlasSC Analytics, Mar. 30, 2026. [Online]. Available: https://atlassc.net/2026/03/30/text-chunking-strategies-for-rag.

[69] M. Palhares, “Chunking Strategies for RAG: Fixed, Recursive, Semantic, Language-Based, and Context-Aware Approaches,” Medium, 2026. [Online]. Available: https://matheusjerico.medium.com/chunking-strategies-for-rag-fixed-recursive-semantic-language-based-and-context-aware-4ab476aea7d1.

[70] Tech-Japan, “Comparative Study of Text Chunking Techniques — Method Selection and Chunk Size Optimization,” Tech-Japan Blog, 2026. [Online]. Available: https://www.tech-japan.jp/en/blog/chunking-research/.

[71] Firecrawl, “Best Chunking Strategies for RAG (and LLMs) in 2026,” Firecrawl Blog, 2026. [Online]. Available: https://www.firecrawl.dev/blog/best-chunking-strategies-rag.

[72] S. Haldar, “PII Extraction for RAG: Building Secure Document Pipelines,” Sandipan Haldar Blog, 2026. [Online]. Available: https://sandipanhaldar.com/blog/pii-extraction-for-rag/.

[73] S. Bisser, *Microsoft Conversational AI Platform for Developers: End-to-End Chatbot Development from Planning to Deployment*. Berkeley, CA: Apress, 2021.

[74] CMARIX, “RAG & AI Trust Statistics 2026: Beating Hallucinations,” CMARIX Blog, 2026. [Online]. Available: https://www.cmarix.com/blog/rag-ai-statistics/.

[75] Master of Code, “AI Evaluation Metrics 2026: Tested by Conversation Experts,” Master of Code Blog, 2026. [Online]. Available: https://masterofcode.com/blog/ai-agent-evaluation.

[76] Rasa, “Best Conversational AI Platforms for Enterprise in 2026,” Rasa Blog, 2026. [Online]. Available: https://rasa.com/blog/best-conversational-ai.

[77] Confident AI, “Multi-Turn LLM Evaluation in 2026: What You Need to Know,” Confident AI Blog, 2026. [Online]. Available: https://www.confident-ai.com/blog/multi-turn-llm-evaluation-in-2026.

[78] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, “RAGAS: Automated Evaluation of Retrieval Augmented Generation,” in *Proc. 18th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2024): System Demonstrations*, 2024, pp. 1-8.

[79] Python Software Foundation, “Welcome to Python.org,” Python.org, 2026. [Online]. Available: https://www.python.org/.

[80] Microsoft, “Visual Studio Code - Code Editing. Redefined,” Visual Studio, 2026. [Online]. Available: https://code.visualstudio.com/.

[81] n8n, “n8n - Fair-code Workflow Automation,” n8n.io, 2026. [Online]. Available: https://n8n.io/.

[82] Chainlit, “Welcome to Chainlit,” Chainlit Docs, 2026. [Online]. Available: https://docs.chainlit.io/.

[83] Ragas, “Ragas Documentation,” Ragas Docs, 2026. [Online]. Available: https://docs.ragas.io/en/stable/.

[84] OpenAI, “OpenAI API Documentation,” OpenAI Platform, 2026. [Online]. Available: https://platform.openai.com/docs/.

[85] European School Radio, “Οδηγός Συμμετοχής: Εγγραφή Χρηστών,” European School Radio Community, 2026. [Online]. Available: https://community.europeanschoolradio.eu/el/participation-guide-old.

[86] Ragas, “Migrate from v0.3 to v0.4,” Ragas Documentation, 2026. [Online]. Available: https://docs.ragas.io/en/stable/howtos/migrations/migrate_from_v03_to_v04/.

[87] S. Wang, L. Hou, J. Lee, and X. Bu, “Evaluating wheel loader operating conditions based on radar chart,” Automation in Construction, vol. 84, pp. 42-49, Dec. 2017, doi: 10.1016/j.autcon.2017.08.020.

# ΠΑΡΑΡΤΗΜΑ Α : ΚΩΔΙΚΑΣ PYTHON

Σε αυτό το παράρτημα παρατίθεται ο πλήρης πηγαίος κώδικας (σε γλώσσα Python) που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας για την προετοιμασία και τον τεμαχισμό του οδηγού χρήσης, τη διεξαγωγή των πειραμάτων και τέλος, την αξιολόγηση των αποτελεσμάτων.

## Α.1: Μετατροπή Word σε Markdown (convert_to_md.py)

```
# ============================================================
#
# Μετατροπή 4 εγγράφων Word (.docx) σε Markdown (.md) μέσω της MarkItDown.
# Είσοδος: Φάκελος 'user_guide' (αρχεία tab1.docx έως tab4.docx).
# Έξοδος: Αποθήκευση των νέων .md αρχείων στον φάκελο
'user_guide_converted'.
#
# ============================================================
import os
from markitdown import MarkItDown

# Αρχικοποίηση του εργαλείου της μετατροπής
md = MarkItDown()

# Ορισμός του ονόματος του φακέλου προορισμού
output_dir = "user_guide_converted"

# Δημιουργία του φακέλου εξόδου (εάν δεν προϋπάρχει)
os.makedirs(output_dir, exist_ok=True)

# Επεξεργασία των τεσσάρων εγγράφων (tab1 - tab4)
for i in range(1, 5):

    # Φτιάχνουμε τα ονόματα των αρχείων βάζοντας τον αριθμό {i}
    input_file = f"user_guide/user_guide_tab{i}.docx"
    output_file = f"{output_dir}/user_guide_tab{i}_converted.md"

    # Μετατροπή του αρχείου (.docx) σε Markdown
    result = md.convert(input_file)

    # Φτιάχνει ένα νέο αρχείο (.md) και αποθηκεύει το αποτέλεσμα
    with open(output_file, "w", encoding='utf-8') as file:
        file.write(result.text_content)

    print(f"Το αρχείο user_guide_tab{i} μετατράπηκε!")

print("Η μετατροπή όλων των αρχείων ολοκληρώθηκε!")
```

## Α.2 Δοκιμή 11 Μεθόδων Τεμαχισμού (chunking_11_methods.py)

```
import os
import json
import tiktoken
import pandas as pd
from langchain_text_splitters import CharacterTextSplitter,
RecursiveCharacterTextSplitter, MarkdownHeaderTextSplitter
from langchain_huggingface import HuggingFaceEmbeddings
from langchain_experimental.text_splitter import SemanticChunker
```

```
# Ορισμός των ονομάτων των φακέλων προορισμού
output_dir_excel = "chunks_excel"
output_dir_json = "chunks_json"

# Δημιουργία των φακέλων εξόδου (εάν δεν προϋπάρχει)
os.makedirs(output_dir_excel, exist_ok=True)
os.makedirs(output_dir_json, exist_ok=True)

# Βοηθητική λίστα για να αποθηκεύσουμε τα στατιστικά
results = []
# Βοηθητικό λεξικό (dictionary) για να αποθηκεύσουμε τα chunks κάθε μεθόδου
για τα Excel
all_chunks_dict = {}

# 1. Φόρτωση του κειμένου Markdown
with open("user_guide_converted/user_guide_tab2_converted.md", "r",
encoding="utf-8") as f:
    text = f.read()

for num in range(1, 20):
    text = text.replace(f"{num}. ", f"{num}) ")

# 2. Εργαλείο καταμέτρησης tokens
enc = tiktoken.get_encoding("cl100k_base")

# 3. Βοηθητική συνάρτηση για τον υπολογισμό των στατιστικών
def get_stats(chunks, method_name):
    num_chunks = len(chunks)
    if num_chunks == 0:
        return {"Μέθοδος": method_name, "Σύνολο Chunks": 0, "Μ.Ο.
Χαρακτήρων": 0, "Μ.Ο. Tokens": 0}

    # Εξαγωγή του κειμένου από κάθε chunk
    chunk_texts = [c.page_content if hasattr(c, 'page_content') else c for
c in chunks]
    avg_chars = sum(len(c) for c in chunk_texts) / num_chunks
    avg_tokens = sum(len(enc.encode(c)) for c in chunk_texts) / num_chunks

    return {
        "Μέθοδος": method_name,
        "Σύνολο Chunks": num_chunks,
        "Μ.Ο. Χαρακτήρων": round(avg_chars),
        "Μ.Ο. Tokens": round(avg_tokens)
    }

print("\nΥπολογισμός διαχωρισμών βάσει Tokens... Παρακαλώ περιμένετε.\n")

#Μετατρέπουμε όλα τα Enter (\n) σε απλά κενά διαστήματα (" ")
clean_text = text.replace('\n', ' ')

# --- Μέθοδος 1) Fixed size 256 tokens (32 tokens overlap) ---
# Χρησιμοποιούμε το CharacterTextSplitter με separator το space για να μην
κόβει τις λέξεις στην μέση
fixed_256 = CharacterTextSplitter.from_tiktoken_encoder(chunk_size=256,
chunk_overlap=32, separator=" ")
chunks_1 = fixed_256.create_documents([clean_text])
results.append(get_stats(chunks_1, "1. Fixed 256t (32t overlap)"))
all_chunks_dict["1_Fixed_256_32"] = chunks_1
```

```
# --- Μέθοδος 2) Fixed size 512 tokens (64 tokens overlap) ---
fixed_512 = CharacterTextSplitter.from_tiktoken_encoder(chunk_size=512,
chunk_overlap=64, separator=" ")
chunks_2 = fixed_512.create_documents([clean_text])
results.append(get_stats(chunks_2, "2. Fixed 512t (64t overlap)"))
all_chunks_dict["2_Fixed_512_64"] = chunks_2

# --- Μέθοδος 3) Recursive 256 tokens (32 tokens overlap) ---
rec_256 =
RecursiveCharacterTextSplitter.from_tiktoken_encoder(chunk_size=256,
chunk_overlap=32)
chunks_3 = rec_256.create_documents([text])
results.append(get_stats(chunks_3, "3. Recursive 256t (32t overlap)"))
all_chunks_dict["3_Recursive_256_32"] = chunks_3

# --- Μέθοδος 4) Recursive 512 tokens (64 tokens overlap) ---
rec_512 =
RecursiveCharacterTextSplitter.from_tiktoken_encoder(chunk_size=512,
chunk_overlap=64)
chunks_4 = rec_512.create_documents([text])
results.append(get_stats(chunks_4, "4. Recursive 512t (64t overlap)"))
all_chunks_dict["4_Recursive_512_64"] = chunks_4

# --- Μέθοδος 5) Recursive 1024 tokens (128 tokens overlap) ---
rec_1024 =
RecursiveCharacterTextSplitter.from_tiktoken_encoder(chunk_size=1024,
chunk_overlap=128)
chunks_5 = rec_1024.create_documents([text])
results.append(get_stats(chunks_5, "5. Recursive 1024t (128t overlap)"))
all_chunks_dict["5_Recursive_1024_128"] = chunks_5

# --- Κοινό Βήμα Προετοιμασίας για τις Μεθόδους 6 & 7 ---
# Κόβουμε πρώτα το κείμενο δομικά, βάσει των τίτλων του Markdown.
headers_to_split_on = [
    ("#", "Καρτέλα"),
    ("##", "Υποκαρτέλα"),
    ("###", "Ενότητα"),
    ("####", "Υποενότητα")
]
md_splitter =
MarkdownHeaderTextSplitter(headers_to_split_on=headers_to_split_on)
md_splits = md_splitter.split_text(text)

# --- Μέθοδος 6) Markdown Headers + Recursive (Όριο: 256 tokens) ---
chunks_6 = rec_256.split_documents(md_splits)
results.append(get_stats(chunks_6, "6. Markdown Headers (max 256t)"))
all_chunks_dict["6_Markdown_Headers_256"] = chunks_6

# --- Μέθοδος 7) Markdown Headers + Recursive (Όριο: 512 tokens) ---
chunks_7 = rec_512.split_documents(md_splits)
results.append(get_stats(chunks_7, "7. Markdown Headers (max 512t)"))
all_chunks_dict["7_Markdown_Headers_512"] = chunks_7

# --- Μέθοδος 8) Markdown Headers + Recursive (Όριο: 1024 tokens) ---
chunks_8 = rec_1024.split_documents(md_splits)
results.append(get_stats(chunks_8, "8. Markdown Headers (max 1024t)"))
all_chunks_dict["8_Markdown_Headers_1024"] = chunks_8
```

```
print("Φόρτωση τοπικού μοντέλου Embeddings... Παρακαλώ περιμένετε.\n")

# Κατεβάζει (μόνο την πρώτη φορά) και φορτώνει το δωρεάν μοντέλο τοπικά
hf_embeddings = HuggingFaceEmbeddings(model_name="lighteternal/stsb-xlm-r-
greek-transfer")

# --- Μέθοδος 9) Markdown Headers + Semantic Chunker (Percentile: 90) ---
# Διαχωρίζει το κείμενο νοηματικά, κόβοντας μόνο στο top 10% των πιο
απότομων αλλαγών θέματος.
semantic_percentile = SemanticChunker(
    hf_embeddings,
    breakpoint_threshold_type="percentile",
    breakpoint_threshold_amount=90
)
chunks_9 = semantic_percentile.split_documents(md_splits)
results.append(get_stats(chunks_9, "9. Semantic (Percentile 90%)"))
all_chunks_dict["9_Semantic_Percentile_90"] = chunks_9

# --- Μέθοδος 10) Markdown Headers + Semantic Chunker (Standard Deviation:
1.5) ---
# Διαχωρίζει το κείμενο όταν η αλλαγή νοήματος ξεπερνάει τον μέσο όρο κατά
1.5 τυπική απόκλιση.
semantic_std = SemanticChunker(
    hf_embeddings,
    breakpoint_threshold_type="standard_deviation",
    breakpoint_threshold_amount=1.5
)
chunks_10 = semantic_std.split_documents(md_splits)
results.append(get_stats(chunks_10, "10. Semantic (Std Dev 1.5)"))
all_chunks_dict["10_Semantic_Std_1_5"] = chunks_10

# --- Μέθοδος 11) Markdown Headers + Semantic Chunker (Interquartile Range:
1.5) ---
# Διαχωρίζει το κείμενο υπολογίζοντας τις αλλαγές με βάση το
ενδοτεταρτημοριακό εύρος (IQR), αγνοώντας τα outliers.
semantic_iqr = SemanticChunker(
    hf_embeddings,
    breakpoint_threshold_type="interquartile",
    breakpoint_threshold_amount=1.5
)
chunks_11 = semantic_iqr.split_documents(md_splits)
results.append(get stats(chunks 11, "11. Semantic (IQR 1.5)"))
all_chunks_dict["11_Semantic_IQR_1_5"] = chunks_11

# 4. Εκτύπωση των στατιστικών σε μορφή πίνακα
df = pd.DataFrame(results)
print(df.to_string(index=False))
print("\n----------------------------------------------------------------\n")

# 5. Εκτύπωση των ονομάτων των καρτελών (Metadata)
print("Οι Καρτέλες/Ενότητες που εντόπισε η Μέθοδος Markdown είναι οι
εξής:\n")

found_tabs = []
for chunk in chunks_6:
    # Αν το metadata του chunk δεν υπάρχει ήδη στη λίστα μας, το
προσθέτουμε
    if chunk.metadata not in found_tabs and chunk.metadata:
        found_tabs.append(chunk.metadata)
```

```
for tab in found_tabs:
    hierarchy_path = " > ".join(tab.values())
    print(f"• {hierarchy_path}")

print("\n-----------------------------------------------------------\n")

# 6. Εξαγωγή όλων των chunks σε ξεχωριστά Excel και JSON
print("Εκκινάει η εξαγωγή των chunks στους αντίστοιχους φακέλους (Excel &
JSON)...\n")

for method_name, chunks_list in all_chunks_dict.items():
    chunk_texts = []
    chunk_metadatas_str = []
    chunk_tokens = []

    # Λίστα που θα κρατήσει τα δεδομένα για το JSON
    json_records = []

    # Διατρέχουμε το κάθε κομμάτι κειμένου που έφτιαξε η μέθοδος
    for i, c in enumerate(chunks_list):
        content = c.page_content if hasattr(c, 'page_content') else str(c)

        # Παίρνουμε το metadata ως κανονικό λεξικό (dict) - Ιδανικό για το
JSON
        metadata_dict = c.metadata if hasattr(c, 'metadata') and c.metadata
else {}
        # Το μετατρέπουμε και σε απλό κείμενο (string) για να φαίνεται
ωραία στο Excel
        metadata_str = str(metadata_dict) if metadata_dict else "Χωρίς
Ετικέτα"

        tokens_count = len(enc.encode(content))

        # Προσθήκη στις λίστες του Excel
        chunk_texts.append(content)
        chunk_metadatas_str.append(metadata_str)
        chunk_tokens.append(tokens_count)

        # Προσθήκη στη δομή του JSON
        json_records.append({
            "chunk_id": i + 1,
            "metadata": metadata_dict, # Εδώ το περνάμε ως δομημένο λεξικό,
όχι χύμα κείμενο
            "tokens": tokens_count,
            "text": content
        })

    # --- ΕΞΑΓΩΓΗ ΣΕ EXCEL ---
    df_export = pd.DataFrame({
        "A/A Chunk": range(1, len(chunks_list) + 1),
        "Καρτέλα (Metadata)": chunk_metadatas_str,
        "Αριθμός Tokens": chunk_tokens,
        "Κείμενο (Chunk)": chunk_texts
    })

    # Αποθήκευση στον φάκελο chunks_excel
    excel_filename = f"{output_dir_excel}/Chunks_{method_name}.xlsx"
    df_export.to_excel(excel_filename, index=False)

    # --- ΕΞΑΓΩΓΗ ΣΕ JSON ---
```

```
    # Αποθήκευση στον ΝΕΟ φάκελο chunks_json
    json_filename = f"{output_dir_json}/Chunks_{method_name}.json"
    with open(json_filename, "w", encoding="utf-8") as json_file:
        json.dump(json_records, json_file, ensure_ascii=False, indent=4)

    print(f"- Δημιουργήθηκαν: {excel_filename} & {json_filename}")

print("\nΗ διαδικασία ολοκληρώθηκε με επιτυχία!")
print("Μπορείς να βρεις τα αρχεία σου στους φακέλους 'chunks_excel' και
'chunks_json'.\n")
```

## A.3 Τελικός Τεμαχισμός με 6 Μεθόδους (chunking_6_methods.py)

```
# ============================================================
#
# Εφαρμογή 6 μεθόδων Chunking (3 Recursive & 3 Semantic) σε 4 .md αρχεία.
# Υπολογισμός στατιστικών (χαρακτήρες, tokens) και συλλογή metadata
επικεφαλίδων.
# Εξαγωγή αποτελεσμάτων σε 24 ξεχωριστά αρχεία JSON και 24 αρχεία Excel.
#
# ============================================================

import os
import json
import tiktoken
import pandas as pd
from langchain_text_splitters import RecursiveCharacterTextSplitter,
MarkdownHeaderTextSplitter
from langchain_huggingface import HuggingFaceEmbeddings
from langchain_experimental.text_splitter import SemanticChunker

# --- 1. Αρχικοποίηση Φακέλων ---
output_dir_excel = "chunks_excel"
os.makedirs(output_dir_excel, exist_ok=True)

# --- 2. Εργαλείο καταμέτρησης tokens & Στατιστικά ---
enc = tiktoken.get_encoding("cl100k_base")

def get_stats(chunks, method_name, tab_num):
    num_chunks = len(chunks)
    if num_chunks == 0:
        return {"Tab": f"Tab {tab_num}", "Μέθοδος": method_name, "Σύνολο
Chunks": 0, "Μ.Ο. Χαρακτήρων": 0, "Μ.Ο. Tokens": 0}

    chunk_texts = [c.page_content if hasattr(c, 'page_content') else c for
c in chunks]
    avg_chars = sum(len(c) for c in chunk_texts) / num_chunks
    avg_tokens = sum(len(enc.encode(c)) for c in chunk_texts) / num_chunks

    return {
        "Tab": f"Tab {tab_num}",
        "Μέθοδος": method_name,
        "Σύνολο Chunks": num_chunks,
        "Μ.Ο. Χαρακτήρων": round(avg_chars),
        "Μ.Ο. Tokens": round(avg_tokens)
    }
```

```
# --- 3. Αρχικοποίηση Μοντέλων & Splitters ---
# Recursive splitters που είναι απαραίτητα για τις μεθόδους 1, 2 και 3
rec_256 =
RecursiveCharacterTextSplitter.from_tiktoken_encoder(chunk_size=256,
chunk_overlap=32)
rec_512 =
RecursiveCharacterTextSplitter.from_tiktoken_encoder(chunk_size=512,
chunk overlap=64)
rec_1024 =
RecursiveCharacterTextSplitter.from_tiktoken_encoder(chunk_size=1024,
chunk_overlap=128)

headers_to_split_on = [
    ("#", "Καρτέλα"),
    ("##", "Υποκαρτέλα"),
    ("###", "Ενότητα"),
    ("####", "Υποενότητα")
]
md_splitter =
MarkdownHeaderTextSplitter(headers_to_split_on=headers_to_split_on)

print("\nΦόρτωση τοπικού μοντέλου Embeddings... Παρακαλώ περιμένετε.\n")
hf_embeddings = HuggingFaceEmbeddings(model_name="lighteternal/stsb-xlm-r-
greek-transfer")

# Αρχικοποίηση Semantic Chunkers (Μεθόδοι 4, 5 και 6)
semantic_percentile = SemanticChunker(hf_embeddings,
breakpoint_threshold_type="percentile", breakpoint_threshold_amount=90)
semantic_std = SemanticChunker(hf_embeddings,
breakpoint_threshold_type="standard_deviation",
breakpoint_threshold_amount=1.5)
semantic_iqr = SemanticChunker(hf_embeddings,
breakpoint_threshold_type="interquartile", breakpoint_threshold_amount=1.5)

# Λεξικό με όλες τις ενεργές μεθόδους (1 έως 6) για εύκολη διαχείριση στο
loop
methods_info = {
    1: {"name": "1_MD_Recursive_256", "splitter": lambda docs:
rec_256.split_documents(docs)},
    2: {"name": "2_MD_Recursive_512", "splitter": lambda docs:
rec_512.split_documents(docs)},
    3: {"name": "3 MD Recursive 1024", "splitter": lambda docs:
rec_1024.split_documents(docs)},
    4: {"name": "4_MD_Semantic_Percentile_90", "splitter": lambda docs:
semantic_percentile.split_documents(docs)},
    5: {"name": "5_MD_Semantic_Std_1_5", "splitter": lambda docs:
semantic_std.split_documents(docs)},
    6: {"name": "6_MD_Semantic_IQR_1_5", "splitter": lambda docs:
semantic_iqr.split_documents(docs)}
}

# Δημιουργία των 6 ξεχωριστών φακέλων για τα JSON αρχεία κάθε μεθόδου
for method_id, info in methods_info.items():
    json_dir = f"chunks_json_{info['name']}"
    os.makedirs(json_dir, exist_ok=True)
    methods_info[method_id]["json_dir"] = json_dir

results = []
found_tabs_all = set()
```

```
print("\nΈναρξη διαχωρισμών ανά Tab... Παρακαλώ περιμένετε.\n")

# --- 4. Main Loop για κάθε αρχείο Tab (1 έως 4) ---
for tab_num in range(1, 5):
    file_path =
f"user_guide_converted/user_guide_tab{tab_num}_converted.md"
    print(f"--- Επεξεργασία: {file_path} ---")

    # Έλεγχος αν υπάρχει το αρχείο για να μη χτυπήσει error
    if not os.path.exists(file_path):
        print(f"Το αρχείο {file_path} δεν βρέθηκε! Προχωράμε στο
επόμενο.\n")
        continue

    with open(file_path, "r", encoding="utf-8") as f:
        text = f.read()

    # Καθαρισμός κειμένου (Αντικατάσταση λιστών του τύπου '1.' με '1)')
    for num in range(1, 20):
        text = text.replace(f"{num}. ", f"{num}) ")

    # Αρχικός δομικός διαχωρισμός με βάση το Markdown (Κοινό βήμα)
    md_splits = md_splitter.split_text(text)

    # Εκτέλεση και των 6 μεθόδων για το συγκεκριμένο Tab
    for method_id, info in methods_info.items():
        method_name = info["name"]
        json_dir = info["json_dir"]

        print(f"  -> Εκτέλεση μεθόδου: {method_name}...")

        # Εφαρμογή της μεθόδου (καλείται δυναμικά από το λεξικό)
        chunks = info["splitter"](md_splits)

        if not chunks:
            print(f"     ΠΡΟΣΟΧΗ: Η μέθοδος {method_name} δεν παρήγαγε
κανένα chunk για το Tab {tab_num}.")
            continue # Πάμε στην επόμενη μέθοδο

        # Αποθήκευση Στατιστικών
        results.append(get_stats(chunks, method_name, tab_num))

        # Συλλογή Metadata (για εκτύπωση στο τέλος) - Έλεγχος πλέον στη
μέθοδο 1
        if method_id == 1:
            for chunk in chunks:
                if chunk.metadata:
                    hierarchy_path = " > ".join(chunk.metadata.values())
                    found_tabs_all.add(hierarchy_path)

        # --- 5. Προετοιμασία Δεδομένων για Εξαγωγή ---
        chunk_texts = []
        chunk_metadatas_str = []
        chunk_tokens = []
        json_records = []

        for i, c in enumerate(chunks):
            content = c.page_content if hasattr(c, 'page_content') else
str(c)
```

```
            metadata_dict = c.metadata if hasattr(c, 'metadata') and
c.metadata else {}
            metadata_str = str(metadata_dict) if metadata_dict else "Χωρίς
Ετικέτα"
            tokens_count = len(enc.encode(content))

            chunk_texts.append(content)
            chunk metadatas str.append(metadata str)
            chunk_tokens.append(tokens_count)

            json_records.append({
                "chunk_id": i + 1,
                "metadata": metadata_dict,
                "tokens": tokens_count,
                "text": content
            })

        # --- ΕΞΑΓΩΓΗ ΣΕ EXCEL ---
        df_export = pd.DataFrame({
            "Α/Α Chunk": range(1, len(chunks) + 1),
            "Καρτέλα (Metadata)": chunk_metadatas_str,
            "Αριθμός Tokens": chunk_tokens,
            "Κείμενο (Chunk)": chunk_texts
        })
        excel filename =
f"{output_dir_excel}/Chunks_Tab{tab_num}_{method_name}.xlsx"
        df_export.to_excel(excel_filename, index=False)

        # --- ΕΞΑΓΩΓΗ ΣΕ JSON ---
        json_filename =
f"{json_dir}/Chunks_Tab{tab_num}_{method_name}.json"
        with open(json_filename, "w", encoding="utf-8") as json_file:
            json.dump(json_records, json_file, ensure_ascii=False,
indent=4)

# --- 6. Τελικές Εκτυπώσεις Στατιστικών ---
print("\n=============================================================
========================\n")
df_results = pd.DataFrame(results)

# 6.α. Πίνακας 1: Αναλυτικός (24 γραμμές), Ταξινομημένος ανά Μέθοδο και Tab
df_results_sorted = df_results.sort_values(by=['Μέθοδος', 'Tab'])
print("ΑΝΑΛΥΤΙΚΑ ΣΤΑΤΙΣΤΙΚΑ ΔΙΑΧΩΡΙΣΜΟΥ (Ταξινομημένα ανά Μέθοδο &
Tab):\n")
print(df_results_sorted.to_string(index=False))
print("\n=============================================================
========================\n")

# 6.β. Πίνακας 2: Συνοπτικός (6 γραμμές), Ομαδοποιημένος ανά Μέθοδο
df_results['Συνολικοί Χαρακτήρες'] = df_results['Σύνολο Chunks'] *
df_results['Μ.Ο. Χαρακτήρων']
df_results['Συνολικά Tokens'] = df_results['Σύνολο Chunks'] *
df_results['Μ.Ο. Tokens']

df_grouped = df_results.groupby('Μέθοδος').agg({
    'Σύνολο Chunks': 'sum',
    'Συνολικοί Χαρακτήρες': 'sum',
    'Συνολικά Tokens': 'sum'
}).reset_index()
```

```
df_grouped['Μ.Ο. Χαρακτήρων'] = (df_grouped['Συνολικοί Χαρακτήρες'] /
df_grouped['Σύνολο Chunks']).round().fillna(0).astype(int)
df_grouped['Μ.Ο. Tokens'] = (df_grouped['Συνολικά Tokens'] /
df_grouped['Σύνολο Chunks']).round().fillna(0).astype(int)

df_final_stats = df_grouped[['Μέθοδος', 'Σύνολο Chunks', 'Μ.Ο. Χαρακτήρων',
'Μ.Ο. Tokens']]

print("ΣΥΓΚΕΝΤΡΩΤΙΚΑ ΣΤΑΤΙΣΤΙΚΑ ΔΙΑΧΩΡΙΣΜΟΥ (Συνολικά για όλο το User Guide
ανά Μέθοδο):\n")
print(df_final_stats.to_string(index=False))

print("\n----------------------------------------------------------------------
------------------------\n")
print("Οι Καρτέλες/Ενότητες που εντοπίστηκαν συνολικά είναι οι εξής:\n")
for tab in sorted(list(found_tabs_all)):
    print(f"• {tab}")

print("\n----------------------------------------------------------------------
------------------------\n")
print("Η διαδικασία ολοκληρώθηκε με επιτυχία!")
print(f"- Δημιουργήθηκαν 24 αρχεία Excel στον φάκελο:
'{output_dir_excel}'")
print("- Δημιουργήθηκαν 24 αρχεία JSON, ταξινομημένα στους αντίστοιχους
φακέλους της κάθε μεθόδου.\n")
```

## A.4 Κεντρικό Σενάριο Εκτέλεσης Διεπαφής (app.py)

```
import os
import chainlit as cl
from agents import SupportLlamaAgent, SupportGeminiAgent,
SupportGemmaAgent, SupportDeepSeekAgent

AGENT_TYPE = os.getenv("AGENT_TYPE", "SUPPORT")
# esr_kb_rag_rec256_new
# esr_kb_rag__Recursive_512_new2
# esr_kb_rag_Recursive_1024_new
# esr_kb_rag_Semantic_Percentile_90_new
# esr_kb_rag_Semantic_Std_new
# esr_kb_rag_Semantic_IQR_new

# greek_rec_256
# greek_rec_512
# greek_rec_1024
# greek_sem_per_90
# greek_sem_std
# greek_sem_iqr

# multi_rec_256
# multi_rec_512
# multi_rec_1024
# multi_per_90
# multi_std
# multi_iqr


@cl.on_chat_start
async def start():
```

```
    if AGENT_TYPE.upper() == "SUPPORT":
        # agent = SupportLlamaAgent(collection_name="multi_iqr")
        # agent =
SupportGemmaAgent(collection_name="esr_kb_rag_Recursive_1024_new")
        # agent = SupportGeminiAgent(collection_name="multi_iqr")
        agent =
SupportDeepSeekAgent(collection_name="esr_kb_rag_rec256_new")
    elif AGENT_TYPE.upper() == "PODCAST_SEARCH":
        agent = PodcastAgent(collection_name="podcast_docs")
    elif AGENT_TYPE.upper() == "LESSON_STRUCTURE":
        agent = LessonStructureAgent(collection_name="lesson_structure")
    else:
        agent = SupportLlamaAgent(collection_name="esr_kb_rag")

    cl.user_session.set("agent", agent)

    # await cl.Message(content=f"Το σύστημα ξεκίνησε ως **{AGENT_TYPE}
Agent**. Πώς μπορώ να βοηθήσω;").send()


@cl.on_message
async def main(message: cl.Message):
    agent = cl.user_session.get("agent")

    # Δημιουργία κενού μηνύματος για το streaming
    msg = cl.Message(content="")

    # Καλούμε το streaming generator του agent
    async for chunk in agent.answer(message.content):
        await msg.stream_token(chunk)

    await msg.send()
```

## A.5 Αφηρημένη Βασική Κλάση Πρακτόρων (base_agent.py)

```
from abc import ABC, abstractmethod

class BaseAgent(ABC):
    def __init__(self, collection_name: str):
        self.collection_name = collection_name

    @abstractmethod
    async def answer(self, prompt: str):
        pass
```

## A.6 Κλάση Πράκτορα Υποστήριξης - Μοντέλο Llama (support_llama.py)

```
import ollama
import os
import uuid
import json
from core.base_agent import BaseAgent
from qdrant_client import QdrantClient
from qdrant_client.models import Filter, FieldCondition, MatchValue
from sentence_transformers import SentenceTransformer
```

```
import logging
from typing import List, Dict, Any, Optional

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Σταθερές για τα E5 prefixes (ΠΟΛΥ ΣΗΜΑΝΤΙΚΟ!)
E5_QUERY_PREFIX = "query: "
E5_PASSAGE_PREFIX = "passage: "


class SupportLlamaAgent(BaseAgent):
    def __init__(self, collection_name="esr_kb_rag"):
        super().__init__(collection_name)
        self.collection_name = collection_name

        self.qdrant = QdrantClient(
            host=os.getenv("QDRANT_HOST", "localhost"),
            port=int(os.getenv("QDRANT_PORT", "6333"))
        )

        self.encoder = SentenceTransformer('intfloat/multilingual-e5-
large')
        # self.encoder = SentenceTransformer('paraphrase-multilingual-
MiniLM-L12-v2')
        # self.encoder = SentenceTransformer('lighteternal/stsb-xlm-r-
greek-transfer')

        # Cache για expansions και classifications
        # self.query_cache = {}

        # Έλεγχος σύνδεσης με Qdrant
        try:
            self.qdrant.get_collections()
            logger.info("Successfully connected to Qdrant")
        except Exception as e:
            logger.error(f"Failed to connect to Qdrant: {e}")
            raise

    async def get_context(self, query: str) -> str:
        """Βελτιωμένη ανάκτηση context με E5 prefixes"""

        # # 1. Expand Query
        # search_query = await self.expand_query(query)
        #
        # # 2. Classification
        # category = await self.get_tab_subtab(query)
        #
        # # 3. Προετοιμασία φίλτρων
        # must_conditions = []
        # if category:
        #     # Προσοχή: Το key πρέπει να ταιριάζει με το payload από το
ingestion script
        #     # Αν χρησιμοποιήσατε "category" ή "metadata.category"
        #     must_conditions.append(
        #         FieldCondition(
        #             key="category",  # ή "metadata.category" ανάλογα με
το payload σας
        #             match=MatchValue(value=category)
```

```
        #         )
        #     )

        # 4. Vector Search - ΣΗΜΑΝΤΙΚΟ: Προσθήκη του E5 QUERY PREFIX!
        query_with_prefix = f"{E5_QUERY_PREFIX}{query}"

        query_vector = self.encoder.encode(
            query_with_prefix,
            convert_to_numpy=True,
            normalize_embeddings=True  # Κρίσιμο για cosine similarity
        ).astype("float32").tolist()

        # Search με βελτιωμένες παραμέτρους
        search_result = self.qdrant.query_points(
            collection_name=self.collection_name,
            query=query_vector,
            # query_filter=Filter(must=must_conditions) if must_conditions
else None,
            limit=4,  # Λίγο μεγαλύτερο για καλύτερο context
            # score_threshold=0.6,  # Αγνοούμε irrelevant results
        )

        logger.info(f"Retrieved {len(search_result.points)} chunks for
query")

        # 5. Κατασκευή του Context με ranking και scoring
        context_parts = []
        for idx, res in enumerate(search_result.points):
            # Παίρνουμε το text από το payload
            text = res.payload.get("text", "")
            if not text:
                text = res.payload.get("full_text", "")

            source = res.payload.get("source_file", "Οδηγός Χρήσης")
            category_info = res.payload.get("category", "")
            subcategory = res.payload.get("subcategory", "")
            score = res.score

            # Προσθήκη metadata στο context για καλύτερη κατανόηση
            context_header = f"[Πηγή: {source}"
            if category_info:
                context_header += f" | Ενότητα: {category_info}"
            if subcategory:
                context_header += f" | Υποενότητα: {subcategory}"
            context_header += f" | Σχετικότητα: {score:.2f}]"

            context_parts.append(f"{context_header}\n{text}")

        context = "\n\n---\n\n".join(context_parts)

        if not context_parts:
            logger.warning("No relevant context found")
            return "Δεν βρέθηκαν σχετικές πληροφορίες για αυτή την
ερώτηση."

        return context, context_parts

    async def answer(self, prompt: str):
        """Βελτιωμένη απάντηση με καλύτερο prompt engineering"""
```

```
        try:
            # Retrieve context
            context, context_parts = await self.get_context(prompt)

            if "Δεν βρέθηκαν σχετικές πληροφορίες" in context:
                yield "Δεν μπόρεσα να βρω σχετικές πληροφορίες για την
ερώτησή σας. Παρακαλώ επικοινωνήστε με την υποστήριξη στο
support@europeanschoolradio.eu"
                return

            # Βελτιωμένο σύστημα prompt για Gemma
            system_prompt = """Είσαι ο επίσημος βοηθός υποστήριξης του
European School Radio (ESR). Η δουλειά σου είναι να βοηθάς τους χρήστες να
πλοηγηθούν στην πλατφόρμα χρησιμοποιώντας ΑΠΟΚΛΕΙΣΤΙΚΑ τις πληροφορίες που
σου δίνονται.

                        ΣΤΟΧΟΣ: Να βοηθάς συντονιστές, μαθητές και χρήστες
με ακριβείς, σαφείς απαντήσεις.

                        ΟΔΗΓΙΕΣ:
                        1. ΠΑΝΤΑ χρησιμοποιείς και απαντάς χρησιμοποιώντας
ΜΟΝΟ τις πληροφορίες από το Context.
                        2. ΜΗΝ χρησιμοποιείς γενικές γνώσεις. Αν η
πληροφορία δεν υπάρχει στο Context, μην προσθέτεις αυθαίρετες πληροφορίες.
                        3. Αν η πληροφορία δεν υπάρχει στο Context,
απάντησε: "Λυπάμαι, δεν βρέθηκαν οδηγίες στα εγχειρίδια της πλατφόρμας για
αυτό το θέμα. Δοκιμάστε να ρωτήσετε με άλλο τρόπο αυτό που ψάχνετε."
                        4. Χρησιμοποίησε bullets για τα βήματα.
                        5. Μετέφερε τις οδηγίες ακριβώς όπως περιγράφονται,
ειδικά ονόματα κουμπιών και τοποθεσίες στην οθόνη.
                        6. Η απάντηση σου να είναι στην ίδια γλώσσα με αυτή
που φαίνεται στο τέλος της ερώτησης μέσα στις αγκύλες .
                        7. ΠΡΟΣΟΧΗ ΣΤΗΝ ΟΡΟΛΟΓΙΑ: Δώσε την απάντηση που
αντιστοιχεί αποκλειστικά στον όρο που χρησιμοποίησε ο χρήστης.

                        ΤΟΝΟΣ: Βοηθητικός, υπομονετικός, σαφής."""

            # User prompt με structured format
            user_prompt = f"""Context:
            {context}

            Question:
            {prompt}

            Instructions:
            Απάντησε στην ερώτηση χρησιμοποιώντας ΜΟΝΟ το παραπάνω context.
            Αν η απάντηση δεν υπάρχει στο context, πες το ευγενικά.
            """

            # Stream από το Gemma
            stream = ollama.chat(
                model='llama3.1:8b',
                messages=[
                    {'role': 'system', 'content': system_prompt},
                    {'role': 'user', 'content': user_prompt}
                ],
                stream=True,
                # options={
                #     'temperature': 0,
```

```
                #     'top_p': 0.9,
                #     'repeat_penalty': 1.1,
                # }
            )

            full_response = ""
            for chunk in stream:
                if 'message' in chunk and 'content' in chunk['message']:
                    content = chunk['message']['content']
                    if content:
                        full_response += content
                        yield content

            # Logging για debugging
            logger.info(f"Generated response of {len(full_response)}
chars")

            filename = "data.json"
            new_data = {
                "user_question": prompt,
                "chunks": context_parts,
                "llm_answer": full_response,
            }

            if os.path.exists(filename) and os.path.getsize(filename) > 0:
                with open(filename, "r", encoding="utf-8") as file:
                    try:
                        # Φόρτωση των υπαρχόντων δεδομένων (αναμένεται
list)
                        data_list = json.load(file)
                    except json.JSONDecodeError:
                        # Αν το αρχείο είναι κατεστραμμένο, ξεκινάμε νέα
λίστα
                        data_list = []
            else:
                data_list = []

            # 2. Append το νέο αντικείμενο στη λίστα
            if isinstance(data_list, list):
                data_list.append(new_data)
            else:
                # Αν το αρχείο δεν είχε λίστα αλλά ένα μεμονωμένο object
                data_list = [data_list, new_data]

            # 3. Εγγραφή (Replace όλο το αρχείο με την ενημερωμένη λίστα)
            with open(filename, "w", encoding="utf-8") as file:
                json.dump(data_list, file, indent=4, ensure_ascii=False)

        except Exception as e:
            error_msg = f"Σφάλμα κατά την επεξεργασία: {str(e)}"
            logger.error(error_msg)
            yield error_msg
```

## A.7 Κλάση Πράκτορα Υποστήριξης - Μοντέλο Gemma (support_gemma.py)

```
import ollama
import os
```

```
import uuid
import json
from core.base_agent import BaseAgent
from qdrant_client import QdrantClient
from qdrant_client.models import Filter, FieldCondition, MatchValue
from sentence_transformers import SentenceTransformer
import logging
from typing import List, Dict, Any, Optional

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Σταθερές για τα E5 prefixes (ΠΟΛΥ ΣΗΜΑΝΤΙΚΟ!)
E5_QUERY_PREFIX = "query: "
E5_PASSAGE_PREFIX = "passage: "


class SupportGemmaAgent(BaseAgent):
    def __init__(self, collection_name="esr_kb_rag"):
        super().__init__(collection_name)
        self.collection_name = collection_name

        self.qdrant = QdrantClient(
            host=os.getenv("QDRANT_HOST", "localhost"),
            port=int(os.getenv("QDRANT_PORT", "6333"))
        )

        # self.encoder = SentenceTransformer('intfloat/multilingual-e5-
large')
        self.encoder = SentenceTransformer('paraphrase-multilingual-MiniLM-
L12-v2')
        # self.encoder = SentenceTransformer('lighteternal/stsb-xlm-r-
greek-transfer')

        # Cache για expansions και classifications
        # self.query_cache = {}

        # Έλεγχος σύνδεσης με Qdrant
        try:
            self.qdrant.get_collections()
            logger.info("Successfully connected to Qdrant")
        except Exception as e:
            logger.error(f"Failed to connect to Qdrant: {e}")
            raise

    async def get_context(self, query: str) -> str:
        """Βελτιωμένη ανάκτηση context με E5 prefixes"""

        # # 1. Expand Query
        # search_query = await self.expand_query(query)
        #
        # # 2. Classification
        # category = await self.get_tab_subtab(query)
        #
        # # 3. Προετοιμασία φίλτρων
        # must_conditions = []
        # if category:
        #     # Προσοχή: Το key πρέπει να ταιριάζει με το payload από το
ingestion script
```

```
        #     # Αν χρησιμοποιήσατε "category" ή "metadata.category"
        #     must_conditions.append(
        #         FieldCondition(
        #             key="category",  # ή "metadata.category" ανάλογα με
το payload σας
        #             match=MatchValue(value=category)
        #         )
        #     )

        # 4. Vector Search - ΣΗΜΑΝΤΙΚΟ: Προσθήκη του E5 QUERY PREFIX!
        query_with_prefix = f"{E5_QUERY_PREFIX}{query}"

        query_vector = self.encoder.encode(
            query_with_prefix,
            convert_to_numpy=True,
            normalize_embeddings=True  # Κρίσιμο για cosine similarity
        ).astype("float32").tolist()

        # Search με βελτιωμένες παραμέτρους
        search_result = self.qdrant.query_points(
            collection_name=self.collection_name,
            query=query_vector,
            # query_filter=Filter(must=must_conditions) if must_conditions
else None,
            limit=4,  # Λίγο μεγαλύτερο για καλύτερο context
            # score_threshold=0.6,  # Αγνοούμε irrelevant results
        )

        logger.info(f"Retrieved {len(search_result.points)} chunks for
query")

        # 5. Κατασκευή του Context με ranking και scoring
        context_parts = []
        for idx, res in enumerate(search_result.points):
            # Παίρνουμε το text από το payload
            text = res.payload.get("text", "")
            if not text:
                text = res.payload.get("full_text", "")

            source = res.payload.get("source_file", "Οδηγός Χρήσης")
            category_info = res.payload.get("category", "")
            subcategory = res.payload.get("subcategory", "")
            score = res.score

            # Προσθήκη metadata στο context για καλύτερη κατανόηση
            context_header = f"[Πηγή: {source}"
            if category_info:
                context_header += f" | Ενότητα: {category_info}"
            if subcategory:
                context_header += f" | Υποενότητα: {subcategory}"
            context_header += f" | Σχετικότητα: {score:.2f}]"

            context_parts.append(f"{context_header}\n{text}")

        context = "\n\n---\n\n".join(context_parts)

        if not context_parts:
            logger.warning("No relevant context found")
```

```
            return "Δεν βρέθηκαν σχετικές πληροφορίες για αυτή την
ερώτηση."

        return context, context_parts

    async def answer(self, prompt: str):
        """Βελτιωμένη απάντηση με καλύτερο prompt engineering"""
        try:
            # Retrieve context
            context, context_parts = await self.get_context(prompt)

            if "Δεν βρέθηκαν σχετικές πληροφορίες" in context:
                yield "Δεν μπόρεσα να βρω σχετικές πληροφορίες για την
ερώτησή σας. Παρακαλώ επικοινωνήστε με την υποστήριξη στο
support@europeanschoolradio.eu"
                return

            # Βελτιωμένο σύστημα prompt για Gemma
            system_prompt = """Είσαι ο επίσημος βοηθός υποστήριξης του
European School Radio (ESR). Η δουλειά σου είναι να βοηθάς τους χρήστες να
πλοηγηθούν στην πλατφόρμα χρησιμοποιώντας ΑΠΟΚΛΕΙΣΤΙΚΑ τις πληροφορίες που
σου δίνονται.

                        ΣΤΟΧΟΣ: Να βοηθάς συντονιστές, μαθητές και χρήστες
με ακριβείς, σαφείς απαντήσεις.

                        ΟΔΗΓΙΕΣ:
                        1. ΠΑΝΤΑ χρησιμοποιείς και απαντάς χρησιμοποιώντας
ΜΟΝΟ τις πληροφορίες από το Context.
                        2. ΜΗΝ χρησιμοποιείς γενικές γνώσεις. Αν η
πληροφορία δεν υπάρχει στο Context, μην προσθέτεις αυθαίρετες πληροφορίες.
                        3. Αν η πληροφορία δεν υπάρχει στο Context,
απάντησε: "Λυπάμαι, δεν βρέθηκαν οδηγίες στα εγχειρίδια της πλατφόρμας για
αυτό το θέμα. Δοκιμάστε να ρωτήσετε με άλλο τρόπο αυτό που ψάχνετε."
                        4. Χρησιμοποίησε bullets για τα βήματα.
                        5. Μετέφερε τις οδηγίες ακριβώς όπως περιγράφονται,
ειδικά ονόματα κουμπιών και τοποθεσίες στην οθόνη.
                        6. Η απάντηση σου να είναι στην ίδια γλώσσα με αυτή
που φαίνεται στο τέλος της ερώτησης μέσα στις αγκύλες .
                        7. ΠΡΟΣΟΧΗ ΣΤΗΝ ΟΡΟΛΟΓΙΑ: Δώσε την απάντηση που
αντιστοιχεί αποκλειστικά στον όρο που χρησιμοποίησε ο χρήστης.

                        ΤΟΝΟΣ: Βοηθητικός, υπομονετικός, σαφής."""

            # User prompt με structured format
            user_prompt = f"""<context>
            {context}
            </context>

            <question>
            {prompt}
            </question>

            <instructions>
            Απάντησε στην ερώτηση χρησιμοποιώντας ΜΟΝΟ το παραπάνω context.
            Αν η απάντηση δεν υπάρχει στο context, πες το ευγενικά.
            </instructions>
            """
```

```
            # Stream από το Gemma
            stream = ollama.chat(
                model='gemma4:26b',
                messages=[
                    {'role': 'system', 'content': system_prompt},
                    {'role': 'user', 'content': user_prompt}
                ],
                stream=True,
                options={
                    'temperature': 0,
                    'top_p': 0.9,
                    'repeat_penalty': 1.1,
                }
            )

            full_response = ""
            for chunk in stream:
                if 'message' in chunk and 'content' in chunk['message']:
                    content = chunk['message']['content']
                    if content:
                        full_response += content
                        yield content

            # Logging για debugging
            logger.info(f"Generated response of {len(full response)}
chars")

            filename = "data.json"
            new_data = {
                "user_question": prompt,
                "chunks": context_parts,
                "llm_answer": full_response,
            }

            if os.path.exists(filename) and os.path.getsize(filename) > 0:
                with open(filename, "r", encoding="utf-8") as file:
                    try:
                        # Φόρτωση των υπαρχόντων δεδομένων (αναμένεται
list)
                        data_list = json.load(file)
                    except json.JSONDecodeError:
                        # Αν το αρχείο είναι κατεστραμμένο, ξεκινάμε νέα
λίστα
                        data_list = []
            else:
                data_list = []

            # 2. Append το νέο αντικείμενο στη λίστα
            if isinstance(data_list, list):
                data_list.append(new_data)
            else:
                # Αν το αρχείο δεν είχε λίστα αλλά ένα μεμονωμένο object
                data_list = [data_list, new_data]

            # 3. Εγγραφή (Replace όλο το αρχείο με την ενημερωμένη λίστα)
            with open(filename, "w", encoding="utf-8") as file:
                json.dump(data_list, file, indent=4, ensure_ascii=False)

        except Exception as e:
            error_msg = f"Σφάλμα κατά την επεξεργασία: {str(e)}"
```

```
            logger.error(error_msg)
            yield error_msg
```

## A.8 Κλάση Πράκτορα Υποστήριξης - Μοντέλο Gemini (support_gemini.py)

```
import os
import uuid
import json
import asyncio

from google import genai
from google.genai import types
from google.oauth2 import service_account
from core.base_agent import BaseAgent
from qdrant_client import QdrantClient
from qdrant_client.models import Filter, FieldCondition, MatchValue
from sentence_transformers import SentenceTransformer
import logging
from typing import Optional

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Σταθερές για τα E5 prefixes (αν τα χρησιμοποιήσετε αργότερα)
E5_QUERY_PREFIX = "query: "
E5_PASSAGE_PREFIX = "passage: "

# Ρύθμιση Service Account
SERVICE_ACCOUNT_FILE = "ai-pipeline-467810-4ebcd195ce93.json"
PROJECT_ID = "ai-pipeline-467810"
LOCATION = "global"  # ή "us-central1", "europe-west4" κλπ.

# Βεβαιωθείτε ότι το αρχείο service account υπάρχει
if not os.path.exists(SERVICE_ACCOUNT_FILE):
    logger.error(f"Service account file not found: {SERVICE_ACCOUNT_FILE}")
    raise FileNotFoundError(f"Service account file not found:
{SERVICE_ACCOUNT_FILE}")


class SupportGeminiAgent(BaseAgent):
    def __init__(self, collection_name="esr_kb_rag", model_name="gemini-
2.5-flash"):
        super().__init__(collection_name)
        self.collection_name = collection_name

        # Gemini Model with service account
        self.model_name = model_name

        # Δημιουργία client με service account
        # Φορτώνουμε τα credentials από το αρχείο
        credentials =
service_account.Credentials.from_service_account_file(
            SERVICE_ACCOUNT_FILE,
            scopes=["https://www.googleapis.com/auth/cloud-platform"]
        )
```

```
        self.client = genai.Client(
            vertexai=True,
            project=PROJECT_ID,
            location=LOCATION,
            credentials=credentials  # Προσθέτουμε τα credentials εδώ
        )

        logger.info(f"Successfully initialized Gemini client with model:
{model_name}")

        # Qdrant connection
        self.qdrant = QdrantClient(
            host=os.getenv("QDRANT_HOST", "localhost"),
            port=int(os.getenv("QDRANT_PORT", "6333"))
        )

        # Ενσωμάτωση embeddings
        # self.encoder = SentenceTransformer('paraphrase-multilingual-
MiniLM-L12-v2')
        # self.encoder = SentenceTransformer('lighteternal/stsb-xlm-r-
greek-transfer')
        self.encoder = SentenceTransformer('intfloat/multilingual-e5-
large')

        # Έλεγχος σύνδεσης με Qdrant
        try:
            self.qdrant.get_collections()
            logger.info("Successfully connected to Qdrant")
        except Exception as e:
            logger.error(f"Failed to connect to Qdrant: {e}")
            raise

    async def get_context(self, query: str):
        """Ανάκτηση context (ίδιο με το αρχικό)"""

        query_with_prefix = f"{E5_QUERY_PREFIX}{query}"

        query_vector = self.encoder.encode(
            query_with_prefix,
            convert_to_numpy=True,
            normalize_embeddings=True
        ).astype("float32").tolist()

        search_result = self.qdrant.query_points(
            collection_name=self.collection_name,
            query=query_vector,
            limit=4,
        )

        logger.info(f"Retrieved {len(search_result.points)} chunks for
query")

        context_parts = []
        for idx, res in enumerate(search_result.points):
            # Παίρνουμε το text από το payload
            text = res.payload.get("text", "")
            if not text:
                text = res.payload.get("full_text", "")
```

```
            source = res.payload.get("source_file", "Οδηγός Χρήσης")
            category_info = res.payload.get("category", "")
            subcategory = res.payload.get("subcategory", "")
            score = res.score

            # Προσθήκη metadata στο context για καλύτερη κατανόηση
            context_header = f"[Πηγή: {source}"
            if category_info:
                context_header += f" | Ενότητα: {category_info}"
            if subcategory:
                context_header += f" | Υποενότητα: {subcategory}"
            context_header += f" | Σχετικότητα: {score:.2f}]"

            context_parts.append(f"{context_header}\n{text}")

        context = "\n\n---\n\n".join(context_parts)

        if not context_parts:
            logger.warning("No relevant context found")
            return "Δεν βρέθηκαν σχετικές πληροφορίες για αυτή την
ερώτηση.", []

        return context, context_parts

    async def answer_gemini_stream(self, prompt: str):
        """Streaming απάντηση με Gemini"""

        try:
            context, context_parts = await self.get_context(prompt)

            if "Δεν βρέθηκαν σχετικές πληροφορίες" in context:
                yield "Δεν μπόρεσα να βρω σχετικές πληροφορίες για την
ερώτησή σας. Παρακαλώ επικοινωνήστε με την υποστήριξη στο
support@europeanschoolradio.eu"
                return

            system_prompt = """Είσαι ο επίσημος βοηθός υποστήριξης του
European School Radio (ESR). Η δουλειά σου είναι να βοηθάς τους χρήστες να
πλοηγηθούν στην πλατφόρμα χρησιμοποιώντας ΑΠΟΚΛΕΙΣΤΙΚΑ τις πληροφορίες που
σου δίνονται.

ΣΤΟΧΟΣ: Να βοηθάς συντονιστές, μαθητές και χρήστες με ακριβείς, σαφείς
απαντήσεις.

ΟΔΗΓΙΕΣ:
1. ΠΑΝΤΑ χρησιμοποιείς και απαντάς χρησιμοποιώντας ΜΟΝΟ τις πληροφορίες από
το Context.
2. ΜΗΝ χρησιμοποιείς γενικές γνώσεις. Αν η πληροφορία δεν υπάρχει στο
Context, μην προσθέτεις αυθαίρετες πληροφορίες.
3. Αν η πληροφορία δεν υπάρχει στο Context, απάντησε: "Λυπάμαι, δεν
βρέθηκαν οδηγίες στα εγχειρίδια της πλατφόρμας για αυτό το θέμα. Δοκιμάστε
να ρωτήσετε με άλλο τρόπο αυτό που ψάχνετε."
4. Χρησιμοποίησε bullets για τα βήματα.
5. Μετέφερε τις οδηγίες ακριβώς όπως περιγράφονται, ειδικά ονόματα κουμπιών
και τοποθεσίες στην οθόνη.
6. Η απάντηση σου να είναι στην ίδια γλώσσα με αυτή που φαίνεται στο τέλος
της ερώτησης μέσα στις αγκύλες .
7. ΠΡΟΣΟΧΗ ΣΤΗΝ ΟΡΟΛΟΓΙΑ: Δώσε την απάντηση που αντιστοιχεί αποκλειστικά
στον όρο που χρησιμοποίησε ο χρήστης.
```

```
ΤΟΝΟΣ: Βοηθητικός, υπομονετικός, σαφής."""

            user_prompt = f"""<context>
{context}
</context>

<question>
{prompt}
</question>

<instructions>
Απάντησε στην ερώτηση χρησιμοποιώντας ΜΟΝΟ το παραπάνω context.
Αν η απάντηση δεν υπάρχει στο context, πες το ευγενικά.
</instructions>
"""

            # Συνδυασμός system και user prompt
            full_prompt = f"{system_prompt}\n\n{user_prompt}"

            full_response = ""

            # Streaming από Gemini - ΣΩΣΤΟΣ ΤΡΟΠΟΣ
            response = self.client.models.generate_content_stream(
                model=self.model name,
                contents=full_prompt,
                config=types.GenerateContentConfig(
                    temperature=0,
                    top_p=0.9,
                    safety_settings=[
                        types.SafetySetting(
                            category=types.HarmCategory.HARM_CATEGORY_HATE_
SPEECH,
                            threshold=types.HarmBlockThreshold.BLOCK_NONE
                        ),
                        types.SafetySetting(
                            category=types.HarmCategory.HARM_CATEGORY_DANGE
ROUS_CONTENT,
                            threshold=types.HarmBlockThreshold.BLOCK_NONE
                        ),
                        types.SafetySetting(
                            category=types.HarmCategory.HARM_CATEGORY_SEXUA
LLY_EXPLICIT,
                            threshold=types.HarmBlockThreshold.BLOCK_NONE
                        ),
                        types.SafetySetting(
                            category=types.HarmCategory.HARM_CATEGORY_HARAS
SMENT,
                            threshold=types.HarmBlockThreshold.BLOCK_NONE
                        ),
                    ]
                )
            )

            # Iterate over the stream
            for chunk in response:
                if chunk.text:
                    full_response += chunk.text
                    yield chunk.text
```

```
            logger.info(f"Generated response of {len(full_response)}
chars")

            # Save interaction
            filename = "data.json"
            new_data = {
                "user_question": prompt,
                "chunks": context_parts,
                "llm_answer": full_response,
            }

            if os.path.exists(filename) and os.path.getsize(filename) > 0:
                with open(filename, "r", encoding="utf-8") as file:
                    try:
                        data_list = json.load(file)
                    except json.JSONDecodeError:
                        data_list = []
            else:
                data_list = []

            if isinstance(data_list, list):
                data_list.append(new_data)
            else:
                data_list = [data_list, new_data]

            with open(filename, "w", encoding="utf-8") as file:
                json.dump(data_list, file, indent=4, ensure_ascii=False)

        except Exception as e:
            error_msg = f"Σφάλμα κατά την επεξεργασία: {str(e)}"
            logger.error(error_msg)
            yield error_msg

    # Legacy alias (για συμβατότητα με το chainlit)
    async def answer(self, prompt: str):
        async for chunk in self.answer_gemini_stream(prompt):
            yield chunk
```

## A.9 Κλάση Πράκτορα Υποστήριξης - Μοντέλο DeepSeek (support_deepseek.py)

```
import os
import uuid
import json
import asyncio
from openai import AsyncOpenAI
from core.base_agent import BaseAgent
from qdrant_client import QdrantClient
from qdrant_client.models import Filter, FieldCondition, MatchValue
from sentence_transformers import SentenceTransformer
import logging
from typing import Optional

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)
```

```
# Σταθερές για τα E5 prefixes (ΠΟΛΥ ΣΗΜΑΝΤΙΚΟ!)
E5_QUERY_PREFIX = "query: "
E5_PASSAGE_PREFIX = "passage: "

# DeepSeek Configuration
DEEPSEEK_API_KEY = os.getenv("DEEPSEEK_API_KEY", "")
DEEPSEEK_BASE_URL = os.getenv("DEEPSEEK_BASE_URL",
"https://api.deepseek.com")
DEEPSEEK_MODEL = os.getenv("DEEPSEEK_MODEL", "deepseek-chat")


class SupportDeepSeekAgent(BaseAgent):
    def __init__(self, collection_name="esr_kb_rag",
model_name=DEEPSEEK_MODEL):
        super().__init__(collection_name)
        self.collection_name = collection_name

        # DeepSeek Model with OpenAI-compatible client
        self.model_name = model_name

        # Δημιουργία AsyncOpenAI client για DeepSeek
        self.client = AsyncOpenAI(
            api_key=DEEPSEEK_API_KEY,
            base_url=DEEPSEEK_BASE_URL,
        )

        logger.info(f"Successfully initialized DeepSeek client with model:
{model_name}")

        # Qdrant connection
        self.qdrant = QdrantClient(
            host=os.getenv("QDRANT_HOST", "localhost"),
            port=int(os.getenv("QDRANT_PORT", "6333"))
        )

        # Ενσωμάτωση embeddings
        # self.encoder = SentenceTransformer('intfloat/multilingual-e5-
large')
        # self.encoder = SentenceTransformer('lighteternal/stsb-xlm-r-
greek-transfer')
        self.encoder = SentenceTransformer('paraphrase-multilingual-MiniLM-
L12-v2')

        # Έλεγχος σύνδεσης με Qdrant
        try:
            self.qdrant.get_collections()
            logger.info("Successfully connected to Qdrant")
        except Exception as e:
            logger.error(f"Failed to connect to Qdrant: {e}")
            raise

    async def get_context(self, query: str):
        """Ανάκτηση context (ίδιο με το αρχικό)"""

        query_with_prefix = f"{E5_QUERY_PREFIX}{query}"

        query_vector = self.encoder.encode(
            query,
            convert_to_numpy=True,
```

```
            normalize_embeddings=True
        ).astype("float32").tolist()

        search_result = self.qdrant.query_points(
            collection_name=self.collection_name,
            query=query_vector,
            limit=16,
        )

        logger.info(f"Retrieved {len(search_result.points)} chunks for
query")

        context_parts = []
        for idx, res in enumerate(search_result.points):
            # Παίρνουμε το text από το payload
            text = res.payload.get("text", "")
            # if not text:
            #     text = res.payload.get("full_text", "")
            #
            # source = res.payload.get("source_file", "Οδηγός Χρήσης")
            # category_info = res.payload.get("category", "")
            # subcategory = res.payload.get("subcategory", "")
            # score = res.score
            #
            # # Προσθήκη metadata στο context για καλύτερη κατανόηση
            # context_header = f"[Πηγή: {source}"
            # if category_info:
            #     context_header += f" | Ενότητα: {category_info}"
            # if subcategory:
            #     context_header += f" | Υποενότητα: {subcategory}"
            # context_header += f" | Σχετικότητα: {score:.2f}]"

            context_parts.append(f"{text}")

        context = "\n\n---\n\n".join(context_parts)

        if not context_parts:
            logger.warning("No relevant context found")
            return "Δεν βρέθηκαν σχετικές πληροφορίες για αυτή την
ερώτηση.", []

        return context, context_parts

    async def answer_deepseek_stream(self, prompt: str):
        """Streaming απάντηση με DeepSeek"""

        try:
            context, context_parts = await self.get_context(prompt)

            if "Δεν βρέθηκαν σχετικές πληροφορίες" in context:
                yield "Δεν μπόρεσα να βρω σχετικές πληροφορίες για την
ερώτησή σας. Παρακαλώ επικοινωνήστε με την υποστήριξη στο
support@europeanschoolradio.eu"
                return

            system_prompt = """Είσαι ο επίσημος βοηθός υποστήριξης του
European School Radio (ESR). Η δουλειά σου είναι να βοηθάς τους χρήστες να
πλοηγηθούν στην πλατφόρμα χρησιμοποιώντας ΑΠΟΚΛΕΙΣΤΙΚΑ τις πληροφορίες που
σου δίνονται.
```

```
ΣΤΟΧΟΣ: Να βοηθάς συντονιστές, μαθητές και χρήστες με ακριβείς, σαφείς
απαντήσεις.

ΟΔΗΓΙΕΣ:
1. ΠΑΝΤΑ χρησιμοποιείς και απαντάς χρησιμοποιώντας ΜΟΝΟ τις πληροφορίες από
το Context.
2. ΜΗΝ χρησιμοποιείς γενικές γνώσεις. Αν η πληροφορία δεν υπάρχει στο
Context, μην προσθέτεις αυθαίρετες πληροφορίες.
3. Αν η πληροφορία δεν υπάρχει στο Context, απάντησε: "Λυπάμαι, δεν
βρέθηκαν οδηγίες στα εγχειρίδια της πλατφόρμας για αυτό το θέμα. Δοκιμάστε
να ρωτήσετε με άλλο τρόπο αυτό που ψάχνετε."
4. Χρησιμοποίησε bullets για τα βήματα.
5. Μετέφερε τις οδηγίες ακριβώς όπως περιγράφονται, ειδικά ονόματα κουμπιών
και τοποθεσίες στην οθόνη.
6. Η απάντηση σου να είναι στην ίδια γλώσσα με αυτή που φαίνεται στο τέλος
της ερώτησης μέσα στις αγκύλες .
7. ΠΡΟΣΟΧΗ ΣΤΗΝ ΟΡΟΛΟΓΙΑ: Δώσε την απάντηση που αντιστοιχεί αποκλειστικά
στον όρο που χρησιμοποίησε ο χρήστης.

ΤΟΝΟΣ: Βοηθητικός, υπομονετικός, σαφής."""

            user_prompt = f"""<context>
{context}
</context>

<question>
{prompt}
</question>

<instructions>
Απάντησε στην ερώτηση χρησιμοποιώντας ΜΟΝΟ το παραπάνω context.
Αν η απάντηση δεν υπάρχει στο context, πες το ευγενικά.
</instructions>
"""

            full_response = ""

            # Streaming από DeepSeek
            stream = await self.client.chat.completions.create(
                model=self.model_name,
                messages=[
                    {"role": "system", "content": system_prompt},
                    {"role": "user", "content": user_prompt}
                ],
                temperature=0,
                top_p=0.9,
                stream=True,
            )

            # Iterate over the stream
            async for chunk in stream:
                if chunk.choices[0].delta.content:
                    content = chunk.choices[0].delta.content
                    full_response += content
                    yield content

            logger.info(f"Generated response of {len(full_response)}
chars")
```

```
            # Save interaction
            filename = "data.json"
            new_data = {
                "user_question": prompt,
                "chunks": context_parts,
                "llm_answer": full_response,
                "model": "deepseek"
            }

            if os.path.exists(filename) and os.path.getsize(filename) > 0:
                with open(filename, "r", encoding="utf-8") as file:
                    try:
                        data_list = json.load(file)
                    except json.JSONDecodeError:
                        data_list = []
            else:
                data_list = []

            if isinstance(data_list, list):
                data_list.append(new_data)
            else:
                data_list = [data_list, new_data]

            with open(filename, "w", encoding="utf-8") as file:
                json.dump(data_list, file, indent=4, ensure_ascii=False)

        except Exception as e:
            error_msg = f"Σφάλμα κατά την επεξεργασία: {str(e)}"
            logger.error(error_msg)
            yield error_msg

    # Legacy alias (για συμβατότητα με το chainlit)
    async def answer(self, prompt: str):
        async for chunk in self.answer_deepseek_stream(prompt):
            yield chunk
```

## A.10 Αξιολόγηση Πειραμάτων με Ragas (ragas_evaluation.py)

```
# ============================================================
# RAGAS v0.4 EXPERIMENT-BASED EVALUATION
# Για 1 φάκελο / 1 συνδυασμό μοντέλων
# πχ. gemma_greek
#
# 6 JSON μέσα στον φάκελο = 6 chunking methods
# π.χ. re_256
#
# Για κάθε JSON αρχείο μέσα στον φάκελο:
# - διαβάζει 40 ερωτήσεις
# - παίρνει user_question, chunks, llm_answer
# - ενώνει με golden answers από Excel
# - τρέχει 4 Ragas metrics
# - φτιάχνει 1 Excel ανά μέθοδο
#
# Στο τέλος:
# - φτιάχνει 1 summary Excel με μέσους όρους για τις 6 μεθόδους
# ============================================================

import re
```

```python
import json
import asyncio
import pandas as pd
from pathlib import Path
from typing import List, Optional
from pydantic import BaseModel
from openai import AsyncOpenAI
from ragas import experiment
from ragas.llms import llm_factory
from ragas.embeddings.base import embedding_factory
from ragas.metrics.collections import (
    ContextRecall,
    ContextPrecision,
    Faithfulness,
    AnswerCorrectness,
)

# ============================================================
# 1. ΡΥΘΜΙΣΕΙΣ
# ============================================================

COMBINATION_FOLDER = Path("JSON FILES/llama_e5")
GOLDEN_EXCEL = Path("GOLDEN DATASET.xlsx")
OUTPUT_ROOT = Path("RAGAS_RESULTS")
OUTPUT ROOT.mkdir(parents=True, exist ok=True)
COMBINATION_NAME = COMBINATION_FOLDER.name

# ============================================================
# 2. LLM INITIALIZATION
# ============================================================

client = AsyncOpenAI(api_key="...")
llm = llm_factory("gpt-4o-mini", client=client, max_tokens=8192)
embeddings = embedding_factory(provider="openai", model="text-embedding-3-
small", client=client)

# ============================================================
# 3. ΜΕΤΡΙΚΕΣ RAGAS
# ============================================================

context_recall_metric = ContextRecall(llm=llm)
context_precision_metric = ContextPrecision(llm=llm)
faithfulness_metric = Faithfulness(llm=llm)
answer_correctness_metric = AnswerCorrectness(llm=llm,
embeddings=embeddings)

# ============================================================
# 4. Pydantic models για experiment()
# ============================================================

class RagasInputRow(BaseModel):
    question_id: int
    user_input: str
    retrieved_contexts: List[str]
    response: str
    reference: str

class RagasExperimentResult(BaseModel):
    context_recall: Optional[float]
```

```
    context_precision: Optional[float]
    faithfulness: Optional[float]
    answer_correctness: Optional[float]

# ============================================================
# 5. RAGAS v0.4 experiment function
# ============================================================

@experiment(RagasExperimentResult)
async def run_ragas_experiment(row: RagasInputRow) ->
RagasExperimentResult:
    context_recall_result = await context_recall_metric.ascore(
        user_input=row.user_input,
        retrieved_contexts=row.retrieved_contexts,
        reference=row.reference
    )
    context_precision_result = await context_precision_metric.ascore(
        user_input=row.user_input,
        reference=row.reference,
        retrieved_contexts=row.retrieved_contexts
    )
    faithfulness_result = await faithfulness_metric.ascore(
        user_input=row.user_input,
        response=row.response,
        retrieved contexts=row.retrieved contexts
    )
    answer_correctness_result = await answer_correctness_metric.ascore(
        user_input=row.user_input,
        response=row.response,
        reference=row.reference
    )
    return RagasExperimentResult(
        context_recall=float(context_recall_result.value),
        context_precision=float(context_precision_result.value),
        faithfulness=float(faithfulness_result.value),
        answer_correctness=float(answer_correctness_result.value),
    )

# ============================================================
# 6. Helper functions
# ============================================================

# Παίρνει ένα όνομα και το μετατρέπει για ονομασία αρχείου πχ. RE 256 ->
RE_256
def safe_filename(name: str) -> str:
    name = str(name).replace(" ", "_")
    name = re.sub(r"[^a-zA-Z0-9_\-\.]", "_", name)
    return name

# Κρατάει την μέθοδο και την μετατρέπει για το excel πχ. re_256_gemma_e5 ->
RE 256
def extract_chunking_method(json_path: Path, combination_name: str) -> str:
    stem = json_path.stem
    suffix = f"_{combination_name}"

    if stem.endswith(suffix):
        method = stem[:-len(suffix)]
    else:
        method = stem
```

```
    return method.upper().replace("_", " ")

# Κάνει την αντιστοίχιση από τα πεδία στα .json αρχεία σε πεδία Ragas:
# user_question    ->      user_input
# chunks           ->      retrieved_contexts
# llm_answer       ->      response
def load_json_results(json_path: Path) -> pd.DataFrame:
    with open(json_path, "r", encoding="utf-8") as f:
        data = json.load(f)

    rows = []

    for i, item in enumerate(data, start=1):
        question = item.get("user_question", "")
        chunks = item.get("chunks", [])
        answer = item.get("llm_answer", "")

        if chunks is None:
            chunks = []

        if isinstance(chunks, str):
            chunks = [chunks]

        rows.append({
            "question_id": i,   # Κρατάει τον αριθμό της ερώτησης
            "user_input": str(question).strip(),
            "retrieved_contexts": [str(chunk) for chunk in chunks],
            "response": str(answer).strip(),
        })

    return pd.DataFrame(rows)

# Μετατρέπει τις στήλες από το GOLDEN DATASET σε πεδία Ragas:
# Ερώτηση          ->      user_input
# Golden Απάντηση  ->      reference
# A/A              ->      question_id
# Κείμενο Αναφοράς ->      reference_context
def load_golden_answers(golden_excel: Path) -> pd.DataFrame:
    golden_df = pd.read_excel(golden_excel)

    golden_df.columns = [str(c).strip() for c in golden_df.columns]

    required_columns = ["Ερώτηση", "Golden Απάντηση"]

    for col in required_columns:
        if col not in golden_df.columns:
            raise ValueError(
                f"Λείπει η στήλη '{col}' από το golden Excel. "
                f"Βρέθηκαν οι στήλες: {golden_df.columns.tolist()}"
            )

    golden_df = golden_df.rename(columns={
        "Ερώτηση": "user_input",
        "Golden Απάντηση": "reference",
        "Κείμενο Αναφοράς": "reference_context",
    })
```

```
    golden_df["user_input"] =
golden_df["user_input"].astype(str).str.strip()
    golden_df["reference"] = golden_df["reference"].astype(str).str.strip()

    return golden_df[["user_input", "reference"]].copy()

# Μετατρέπει το DataFrame σε μία λίστα από RagasInputRow (κάθε ερώτηση
γίνεται structured object για να περάσει στο @experiment)
def build_experiment_rows(eval_df: pd.DataFrame) -> List[RagasInputRow]:
    rows = []

    for _, row in eval_df.iterrows():
        rows.append(
            RagasInputRow(
                question_id=int(row["question_id"]),
                user_input=str(row["user_input"]),
                retrieved_contexts=list(row["retrieved_contexts"]),
                response=str(row["response"]),
                reference=str(row["reference"]),
            )
        )

    return rows

# Μετατρέπει τα structured results του experiment σε pandas DataFrame
def experiment_results_to_dataframe(exp_results) -> pd.DataFrame:
    return pd.DataFrame([result.model_dump() for result in exp_results])

# ============================================================
# 7. Αξιολόγηση ενός JSON αρχείου / μίας chunking μεθόδου
# ============================================================

async def evaluate_one_json_with_experiment(
    json_path: Path,
    golden_df: pd.DataFrame,
    output_folder: Path,
    combination_name: str
) -> dict:

    print("\n" + "=" * 80)
    print(f"\nΕπεξεργασία: {json_path.name}")
    chunking_method = extract_chunking_method(json_path, combination_name)
    print(f"Chunking μέθοδος: {chunking_method}")

    # 1. Load JSON
    df = load_json_results(json_path)

    # 2. Ενώνει το JSON με το GOLDEN DATASET
    eval_df = df.merge(
        golden_df,
        on="user_input",
        how="left"
    )

    # 3. Έλεγχος για missing golden answers
    missing_refs = eval_df["reference"].isna().sum()

    if missing_refs > 0:
```

```
        missing_questions =
eval_df[eval_df["reference"].isna()]["user_input"].tolist()

        raise ValueError(
            f"Στο αρχείο {json_path.name} λείπουν {missing_refs} golden
answers.\n"
            f"Πρώτες ερωτήσεις χωρίς reference:\n{missing_questions[:5]}"
        )

    # 4. Μετατροπή σε rows για experiment()
    experiment_rows = build_experiment_rows(eval_df)

    # 5. Τρέξιμο Ragas experiment

    CONCURRENCY_LIMIT = 4 #Τρέξε 4 ερωτήσεις παράλληλα
    semaphore = asyncio.Semaphore(CONCURRENCY_LIMIT)

    async def run_one_row(row):
        async with semaphore:
            return await run_ragas_experiment(row)

    exp_results = await asyncio.gather(
        *[run_one_row(row) for row in experiment_rows]
    )

    # 6. Results σε DataFrame
    scores_df = experiment_results_to_dataframe(exp_results)

    metric_cols = [
        "context_recall",
        "context_precision",
        "faithfulness",
        "answer_correctness",
    ]

    for col in metric_cols:
        if col not in scores_df.columns:
            raise ValueError(
                f"Το experiment δεν επέστρεψε τη στήλη '{col}'. "
                f"Στήλες που βρέθηκαν: {scores_df.columns.tolist()}"
            )

    # 7. Ένωση δεδομένων + scores
    final_df = pd.concat(
        [
            eval_df.reset_index(drop=True),
            scores_df[metric_cols].reset_index(drop=True)
        ],
        axis=1
    )

    # 8. Overall score ανά ερώτηση
    final_df["overall_score"] = final_df[metric_cols].mean(axis=1)

    # 9. Διάταξη στηλών αναλυτικού Excel για κάθε JSON
    output_columns = [
        "user_input",
        "retrieved_contexts",
        "response",
```

```
        "context_recall",
        "context_precision",
        "faithfulness",
        "answer_correctness",
    ]
    detailed_df = final_df[output_columns].copy()

    # 10. Αποθήκευση αναλυτικού Excel για κάθε JSON
    output_file = output_folder /
f"EVALUATION_{safe_filename(json_path.stem)}.xlsx"
    detailed_df.to_excel(output_file, index=False)
    print(f"\nΑποθηκέυτηκε το αναλυτικό Excel: {output_file}")

    # 11. Summary row για αυτή τη μέθοδο
    summary_row = {
        "Μοντέλα LLM & Embedding": combination_name.upper().replace("_", "
"),
        "Μέθοδος Chunking": chunking_method,
        "Context Recall": final_df["context_recall"].mean(),
        "Context Precision": final_df["context_precision"].mean(),
        "Faithfulness": final_df["faithfulness"].mean(),
        "Answer Correctness": final_df["answer_correctness"].mean(),
    }
    return summary_row

# ============================================================
# 8. Τρέξιμο όλων των JSON ενός φακέλου
# ============================================================

async def run_ragas_for_combination_folder_v04(
    combination_folder: Path,
    golden_excel: Path,
    output_root: Path
) -> pd.DataFrame:

    combination_name = combination_folder.name
    output_folder = output_root / combination_name  # Φτιάχνει φάκελο για
να βάλει τα excel πχ. RAGAS_RESULTS/gemma_greek/
    output_folder.mkdir(parents=True, exist_ok=True)
    golden_df = load_golden_answers(golden_excel)
    json_files = sorted(combination_folder.glob("*.json"))
    if not json_files:
        raise FileNotFoundError(
            f"\nΔεν βρέθηκαν JSON αρχεία στον φάκελο: {combination_folder}"
        )
    print(f"\nΦάκελος: {combination_folder}")
    print(f"Βρέθηκαν {len(json_files)} JSON αρχεία:")
    for f in json_files:
        print(f" - {f.name}")

    summary_rows = []

    # Τρέχει κάθε JSON
    for json_path in json_files:
        summary_row = await evaluate_one_json_with_experiment(
            json_path=json_path,
            golden_df=golden_df,
            output_folder=output_folder,
            combination_name=combination_name
        )
```

```
            summary_rows.append(summary_row)

        summary_df = pd.DataFrame(summary_rows)

        # Συνολικός μέσος όρος = απλός μέσος όρος των 4 metrics
        summary_df["ΣΥΝΟΛΙΚΟΣ ΜΕΣΟΣ ΟΡΟΣ"] = summary_df[
            [
                "Context Recall",
                "Context Precision",
                "Faithfulness",
                "Answer Correctness",
            ]
        ].mean(axis=1) * 100

        # Ταξινόμηση από το καλύτερο προς το χειρότερο
        summary_df = summary_df.sort_values(by="ΣΥΝΟΛΙΚΟΣ ΜΕΣΟΣ
ΟΡΟΣ",ascending=False).reset_index(drop=True)

        # Προσθήκη στήλης "Κατάταξη" στο συνολικό Excel
        summary_df.insert(0, "Κατάταξη", range(1, len(summary_df) + 1))

        # Στρογγυλοποίηση
        summary_df["Context Recall"] = summary_df["Context Recall"].round(2)
        summary_df["Context Precision"] = summary_df["Context
Precision"].round(2)
        summary_df["Faithfulness"] = summary_df["Faithfulness"].round(2)
        summary_df["Answer Correctness"] = summary_df["Answer
Correctness"].round(2)
        summary_df["ΣΥΝΟΛΙΚΟΣ ΜΕΣΟΣ ΟΡΟΣ"] = summary_df["ΣΥΝΟΛΙΚΟΣ ΜΕΣΟΣ
ΟΡΟΣ"].round(2)

        # Σειρά στηλών στο συνολικό Excel
        summary_df = summary_df[
            [
                "Κατάταξη",
                "Μοντέλα LLM & Embedding",
                "Μέθοδος Chunking",
                "Context Recall",
                "Context Precision",
                "Faithfulness",
                "Answer Correctness",
                "ΣΥΝΟΛΙΚΟΣ ΜΕΣΟΣ ΟΡΟΣ",
            ]
        ]

        summary_file = output_folder /
f"{safe_filename(combination_name).upper()}_EVALUATION.xlsx"
        summary_df.to_excel(summary_file, index=False)
        print("\n" + "=" * 80)
        print(f"\nΑποθηκεύτηκε το συνολικό Excel: {summary_file}\n")
        print(summary_df.to_string(index=False))
        print("\n" + "=" * 80 + "\n")
        return summary_df

# ============================================================
# 9. Run
# ============================================================

if __name__ == "__main__":
```

```
asyncio.run(
    run_ragas_for_combination_folder_v04(
        combination_folder=COMBINATION_FOLDER,
        golden_excel=GOLDEN_EXCEL,
        output_root=OUTPUT_ROOT
    )
)
```

# ΠΑΡΑΡΤΗΜΑ Β : ΣΥΝΟΛΟ ΔΕΔΟΜΕΝΩΝ ΑΞΙΟΛΟΓΗΣΗΣ (GOLDEN DATASET)

| Α/Α | Ερώτηση | Golden Απάντηση |
|---|---|---|
| 1 | Πώς ακούω ραδιόφωνο; | Για να ακούσετε ραδιόφωνο στην πλατφόρμα του European School Radio:<br><br>*Πατήστε το κυκλικό κουμπί με το εικονίδιο του ραδιοφώνου, που βρίσκεται στην κάτω δεξιά πλευρά της οθόνης.<br>*Το κουμπί αυτό εμφανίζεται σε όλες τις σελίδες της πλατφόρμας ως Floating Action Button.<br>*Μόλις το πατήσετε, ανοίγει ένα μικρό παράθυρο δίπλα από το κουμπί με τη ζωντανή ραδιοφωνική ροή.<br>*Η αναπαραγωγή της ζωντανής μετάδοσης ξεκινάει αυτόματα μόλις ανοίξει το παράθυρο, αφού ολοκληρωθεί η φόρτωση της ροής.<br>*Στο παράθυρο αναπαραγωγής μπορείτε να χρησιμοποιήσετε:<br>**Το κουμπί Παύσης/Αναπαραγωγής.<br>**Τον επιλογέα έντασης.<br>**Το κουμπί Ζωντανής συζήτησης.<br>**Το κουμπί Προτάσεις μουσικής.<br><br>Επιπλέον, στο παράθυρο εμφανίζονται πληροφορίες για το τι παίζει εκείνη τη στιγμή, όπως:<br><br>*Τίτλος τραγουδιού και καλλιτέχνης.<br>*Όνομα ραδιοφωνικής παραγωγής και εκπαιδευτικής μονάδας.<br>*Η ένδειξη “ESR Live” όταν υπάρχει ζωντανή μετάδοση από την ομάδα του European School Radio. |
| 2 | Αν αλλάξω σελίδα θα σταματήσει να παίζει το podcast; | Όχι, αν αλλάξετε σε άλλη σελίδα, η ακρόαση του podcast συνεχίζεται χωρίς διακοπή.<br><br>Συγκεκριμένα:<br><br>*Όταν επιλέξετε την αναπαραγωγή ενός επεισοδίου, εμφανίζεται ένα μικρό παράθυρο player στο κάτω αριστερό μέρος της σελίδας.<br>*Μπορείτε να συνεχίσετε να περιηγείστε στην πλατφόρμα ενώ το επεισόδιο παίζει.<br>*Σε περίπτωση που αλλάξετε σε άλλη σελίδα, η ακρόαση συνεχίζεται κανονικά χωρίς να σταματήσει η αναπαραγωγή. |
| 3 | Πώς βρίσκω το σχολείο μου; | Για να βρείτε το σχολείο σας:<br><br>*Επιλέξτε από το οριζόντιο μενού την επιλογή «Κοινότητα».<br>*Στη συνέχεια επιλέξτε «Εκπαιδευτικές Μονάδες».<br>*Στη σελίδα που θα ανοίξει, πληκτρολογήστε τον όρο αναζήτησης που επιθυμείτε στην μπάρα αναζήτησης.<br>*Θα εμφανιστεί ένας χάρτης στο πάνω μέρος της σελίδας με τις τοποθεσίες όλων των ενεργών εκπαιδευτικών μονάδων.<br>*Κάτω από τον χάρτη θα εμφανιστούν τα πιο κοντινά αποτελέσματα.<br>*Στη πλαϊνή στήλη μπορείτε να εφαρμόσετε φίλτρα όπως:<br>**τύπος εκπαιδευτικής μονάδας,<br>**ηλικιακή κατηγορία, |

| | | |
|---|---|---|
| | | **χώρα,<br>**πόλη.<br>*Μόλις βρείτε το σχολείο που αναζητάτε, κάντε κλικ στο όνομά του για να μεταφερθείτε στη σελίδα του σχολείου. |
| 4 | Πώς βλέπω τι εκπομπή παίζει τώρα στο live; | Για να δείτε τι εκπομπή παίζει εκείνη τη στιγμή στο live, θα πρέπει να ανοίξετε το μικρό παράθυρο της ζωντανής ραδιοφωνικής ροής:<br><br>*Πατήστε το κυκλικό κουμπί με το εικονίδιο του ραδιοφώνου, που βρίσκεται στην κάτω δεξιά πλευρά της οθόνης.<br>*Θα ανοίξει ένα μικρό παράθυρο δίπλα από το Floating Action Button και θα ξεκινήσει η αναπαραγωγή του live ραδιοφώνου.<br>*Μέσα σε αυτό το παράθυρο εμφανίζονται πληροφορίες για το τι αναπαράγεται εκείνη τη στιγμή.<br><br>Υπάρχουν 3 περιπτώσεις:<br><br>*Αν παίζει τραγούδι, εμφανίζεται ο τίτλος του τραγουδιού και ο καλλιτέχνης.<br>*Αν παίζει επεισόδιο ραδιοφωνικής παραγωγής, εμφανίζεται:<br>**το όνομα της παραγωγής<br>**το όνομα της εκπαιδευτικής μονάδας κάτω από την οποία έχει δημιουργηθεί<br>*Αν εμφανίζεται η ένδειξη “ESR Live”, σημαίνει ότι ακούγεται ζωντανή μετάδοση από την ομάδα του European School Radio. |
| 5 | Γίνεται να βάλω υπότιτλους ή μετάφραση στα podcast; | Ναι, η πλατφόρμα προσφέρει εργαλεία προσβασιμότητας όπως υπότιτλους και μετάφραση για τα podcast και τα επεισόδια.<br><br>*Κάτω από κάθε επεισόδιο ή podcast θα βρείτε διαθέσιμα εργαλεία όπως απομαγνητοφώνηση (Transcript), υπότιτλους και μετάφραση.<br>*Αυτά εμφανίζονται στις παραγωγές όταν τις ακούτε εκτός του Radio Player.<br><br>Για τους υπότιτλους:<br><br>*Πατήστε το εικονίδιο «Υπότιτλοι» (Subtitles).<br>*Θα ενεργοποιηθεί η προβολή του κειμένου συγχρονισμένου με τον ήχο.<br>*Μπορείτε να ενεργοποιήσετε ή να απενεργοποιήσετε τη λειτουργία χωρίς να σταματήσει η αναπαραγωγή.<br><br>Για τη μετάφραση:<br><br>*Πατήστε το εικονίδιο «Μετάφραση» (Translate).<br>*Το κείμενο της απομαγνητοφώνησης μεταφράζεται αυτόματα στη γλώσσα σας, ώστε να κατανοείτε εύκολα το περιεχόμενο του επεισοδίου. |
| 6 | Πώς βρίσκω ένα επεισόδιο για να το ακούσω; | Για να ακούσετε ένα επεισόδιο, θα πρέπει πρώτα να το εντοπίσετε με έναν από τους ακόλουθους τρόπους:<br><br>*Στην Βιβλιοθήκη (Library) του δημιουργού.<br>*Στο Προφίλ της Ομάδας - Εκπαιδευτικής Μονάδας που ανήκει ο δημιουργός.<br>*Στην Εξερεύνηση Podcast αξιοποιώντας την μπάρα αναζήτησης.<br>*Στα Floating Buttons στην κάτω δεξιά γωνία της σελίδας:<br>**είτε σε αυτό με τον μεγεθυντικό φακό για απλή αναζήτηση, |

| | | |
|---|---|---|
| | | **είτε σε αυτό με τα αστέρια για αναζήτηση με χρήση AI.<br><br>Αφού επιλέξετε το επεισόδιο:<br><br>*Μεταφέρεστε στη σελίδα του επεισοδίου.<br>*Πατήστε το κουμπί αναπαραγωγής στο αριστερό μέρος κάτω από το banner.<br>*Στο κάτω αριστερό μέρος της σελίδας θα εμφανιστεί ένα μικρό παράθυρο player.<br>*Από εκεί μπορείτε:<br>**να κάνετε Παύση (Pause),<br>**να συνεχίσετε την αναπαραγωγή (Resume),<br>**ή να μετακινηθείτε σε άλλο σημείο του επεισοδίου μέσα από τα κουμπιά ελέγχου του player.<br><br>Η ακρόαση συνεχίζεται χωρίς διακοπή ακόμα και αν αλλάξετε σε άλλη σελίδα. |
| 7 | Μπορώ να δω πότε θα παίξει η επόμενη εκπομπή; | Ναι, μπορείτε να δείτε πότε θα παίξει η επόμενη εκπομπή.<br><br>Οι «Επόμενες Εκπομπές (Next Shows)» εμφανίζονται σε 2 σημεία στην πλατφόρμα:<br><br>*Στην αρχική σελίδα της πλατφόρμας.<br>*Στη σελίδα του ραδιοφωνικού προγράμματος που βρίσκεται στο κεντρικό μενού κάτω από το «Podcast/Ραδιόφωνο».<br><br>Επιπλέον:<br><br>*Στην ενότητα «Επόμενες Μεταδόσεις» της σελίδας ανακάλυψης Podcast, μπορείτε να δείτε παραγωγές που δεν έχουν ακόμη δημοσιευθεί και θα γίνουν διαθέσιμες αφού ακουστούν στην προγραμματισμένη ώρα στη ροή του ραδιοφώνου.<br>*Αν έχετε εγκατεστημένη την εφαρμογή του European School Radio ή του Youth Radio, μπορείτε να λαμβάνετε αυτοματοποιημένες ειδοποιήσεις:<br>**15 λεπτά πριν την προγραμματισμένη αναπαραγωγή.<br>**5 λεπτά πριν την προγραμματισμένη αναπαραγωγή. |
| 8 | Μπορώ να ψάξω κάτι χωρίς να ξέρω τον τίτλο; | Ναι. Η πλατφόρμα υποστηρίζει αναζήτηση ακόμα και αν δεν γνωρίζετε τον ακριβή τίτλο.<br><br>Μπορείτε να χρησιμοποιήσετε την «Έξυπνη Αναζήτηση (AI Search)» ως εξής:<br><br>*Στη κεντρική μπάρα, επιλέξτε «Podcast/Radio» και μετά «Ανακαλύψτε Podcast».<br>*Στη σελίδα αναζήτησης, μπορείτε να πληκτρολογήσετε:<br>**λέξεις,<br>**φράσεις,<br>**ή ακόμα και κάτι που ειπώθηκε μέσα στο ηχητικό.<br>*Η αναζήτηση εντοπίζει περιεχόμενο σχετικό με:<br>**τον τίτλο,<br>**την περιγραφή,<br>**αλλά και τα λεγόμενα του επεισοδίου.<br>*Για παράδειγμα, αν γράψετε «ασφάλεια στο διαδίκτυο», θα εμφανιστούν επεισόδια όπου υπάρχει σχετική αναφορά.<br>*Η αναζήτηση προσαρμόζεται στα συμφραζόμενα και προτείνει σχετικό περιεχόμενο. |

| | | |
|---|---|---|
| | | *Μπορείτε επίσης να ανοίξετε γρήγορα την AI αναζήτηση από το κυκλικό κουμπί (Floating Action Button) με το σύμβολο των αστεριών, στο κάτω δεξί μέρος της οθόνης. |
| 9 | Πώς μπορώ να κάνω αναζήτηση χωρίς τη χρήση AI; | Για να κάνετε αναζήτηση χωρίς τη χρήση AI, ακολουθήστε τα παρακάτω βήματα:<br><br>*Μεταβείτε στη σελίδα "Εξερεύνηση Podcast".<br>*Στο δεξί μέρος της μπάρας αναζήτησης, απενεργοποιήστε την επιλογή "Αναζήτηση με AI".<br>*Πληκτρολογήστε τον όρο αναζήτησης που επιθυμείτε.<br>*Θα εμφανιστούν τα αποτελέσματα της αναζήτησής σας.<br>*Στη πλαϊνή στήλη μπορείτε να εφαρμόσετε φίλτρα όπως:<br>**θεματική,<br>**ηλικιακή ομάδα,<br>**διάρκεια,<br>**εκπαιδευτική μονάδα,<br>**γλώσσα,<br>**τύπο παραγωγής,<br>**ημερομηνία δημοσίευσης.<br>*Μόλις βρείτε το Podcast ή/και την παραγωγή που αναζητάτε, κάντε κλικ στον τίτλο της για να μεταφερθείτε στη σελίδα της παραγωγής ή του Podcast. |
| 10 | Εκτός από το να μιλάω με άλλους, τι άλλο μπορώ να κάνω ή να βρω μέσα στο community; | Μέσα στο Community, εκτός από το να μιλάτε με άλλους χρήστες, μπορείτε να κάνετε ή να βρείτε τα εξής:<br><br>*Κοινότητα: Να βρείτε άλλους χρήστες, ομάδες ή σχολεία καθώς και τη δραστηριότητά τους μέσα στη ραδιοφωνική κοινότητα.<br>*Νέα: Να ανακοινώνετε στους υπόλοιπους χρήστες δράσεις που πραγματοποιήσατε ή να ενημερώνεστε για δραστηριότητες της κοινότητας στις οποίες μπορείτε να συμμετέχετε.<br>*Εκπαιδευτικό Υλικό & Οδηγούς: Να βρείτε άρθρα, οδηγούς και παραδείγματα καλών πρακτικών για:<br>**τη δημιουργία podcasts,<br>**την ηχογράφηση,<br>**την παιδαγωγική αξιοποίηση του ραδιοφώνου,<br>**καθώς και αποθετήριο πόρων για έμπνευση δημιουργών.<br>*Προφίλ: Να προσαρμόσετε το προσωπικό ή ομαδικό προφίλ σας με:<br>**φωτογραφία,<br>**περιγραφή,<br>**στοιχεία επικοινωνίας,<br>ώστε άλλοι δημιουργοί να μπορούν να σας βρουν και να συνεργαστούν μαζί σας.<br>*Podcast/Ραδιόφωνο: Να επιστρέψετε πίσω στη σελίδα του ραδιοφώνου.<br>*Σχετικά: Να δείτε συγκεντρωμένες όλες τις πληροφορίες της εταιρείας. |
| 11 | Τι επιπλέον δυνατότητες έχω αν κάνω εγγραφή στην πλατφόρμα; | Με την εγγραφή σας στην πλατφόρμα αποκτάτε επιπλέον δυνατότητες για να αξιοποιήσετε τα εξελιγμένα χαρακτηριστικά δημιουργίας, ακρόασης ραδιοφωνικών εκπομπών και Podcasts, αλλά και κοινωνικής αλληλεπίδρασης γύρω από αυτά.<br><br>Ανάλογα με τον ρόλο σας στην πλατφόρμα, έχετε τις εξής δυνατότητες:<br><br>*Εκπαιδευτικοί – Συντονιστές Ομάδων:<br>**Κράτηση ραδιοφωνικών εκπομπών. |

| | | |
|---|---|---|
| | | **Δημιουργία σειρών podcast.<br>**Ανέβασμα ηχητικών επεισοδίων.<br>**Έγκριση εγγραφών μαθητευόμενων και περιεχομένου.<br>**Έγκριση νέων εκπαιδευτικών – συντονιστών.<br>**Διαχείριση αναφορών μη κατάλληλου περιεχομένου.<br>**Δημιουργία blog posts, εκπαιδευτικού περιεχομένου και ομάδων.<br>**Δυνατότητες κοινωνικής αλληλεπίδρασης (Like, follow, share, comment, rate).<br>*Μαθητευόμενοι:<br>**Συμμετοχή σε ραδιοφωνικές εκπομπές.<br>**Δημιουργία podcasts.<br>**Δυνατότητες κοινωνικής αλληλεπίδρασης (Like, follow, share, comment, rate).<br>*Επισκέπτες:<br>**Ακρόαση εκπομπών.<br>**Περιήγηση στην πλατφόρμα.<br>**Προβολή δημόσιου περιεχομένου.<br>**Περιορισμένες δυνατότητες κοινωνικής αλληλεπίδρασης (Like, follow, share). |
| 12 | Είμαι δάσκαλος σε σχολείο στην Ελλάδα, πώς κάνω εγγραφή; | Αν είστε δάσκαλος σε σχολείο στην Ελλάδα, η εγγραφή γίνεται αυτόματα ως Εκπαιδευτικός – Συντονιστής ακολουθώντας τα παρακάτω βήματα:<br><br>*Πατήστε την επιλογή «Σύνδεση με Πανελλήνιο Σχολικό Δίκτυο».<br>*Θα ανακατευθυνθείτε στο ΠΣΔ.<br>*Συνδεθείτε χρησιμοποιώντας τους κωδικούς που έχετε στο Πανελλήνιο Σχολικό Δίκτυο.<br>*Με την ολοκλήρωση της σύνδεσης, η εγγραφή σας πραγματοποιείται αυτόματα.<br>*Με αυτόν τον τρόπο εντάσσεστε αυτόματα και ως συντονιστής στην ή στις εκπαιδευτικές μονάδες σας. |
| 13 | Πώς μπορώ να αλλάξω τη φωτογραφία στο προφίλ μου; | Για να αλλάξετε τη φωτογραφία στο προφίλ σας, ακολουθήστε τα παρακάτω βήματα:<br><br>*Από το αναπτυσσόμενο μενού προφίλ στην αρχική σελίδα, πατήστε την επιλογή «Βιβλιοθήκη».<br>*Από το αριστερό μενού πλοήγησης στο προφίλ σας, πατήστε την επιλογή «ΚΟΙΝΟΤΗΤΑ».<br>*Στην πάνω αριστερή γωνία της μαύρης μπάρας πλοήγησης, πατήστε «European School Radio Radio Community».<br>*Στη σελίδα Προφίλ, πατήστε την καρτέλα «Πλήρες προφίλ».<br>*Πατήστε την επιλογή «Επεξεργασία φωτογραφίας».<br>*Στο παράθυρο που εμφανίζεται, μπορείτε:<br>**είτε να σύρετε το αρχείο φωτογραφίας,<br>**είτε να πατήσετε το κουμπί «Επιλέξτε το Αρχείο σας».<br>*Αφού επιλέξετε τη φωτογραφία, μπορείτε να την περικόψετε.<br>*Πατήστε το κουμπί «Ψαλίδισμα Εικόνας» για οριστικοποίηση.<br>*Όταν εμφανιστεί το μήνυμα «Η νέα σας φωτογραφία προφίλ ανέβηκε επιτυχώς», πατήστε το κουμπί «Χ» για κλείσιμο. |
| 14 | Τι είναι ο Επίσημος Αριθμός Σχολείου; | Το πεδίο «Επίσημος Αριθμός Σχολείου» είναι ο αριθμός του σχολείου που χρησιμοποιείται στο υπουργείο. Για τα σχολεία της Κύπρου, είναι ο αριθμός με τον οποίο γίνεται η παραγγελία βιβλίων στο υπουργείο. |

| | | |
|---|---|---|
| | | Αν δεν γνωρίζετε ποιος είναι αυτός ο αριθμός, θα πρέπει να ρωτήσετε τον διευθυντή του σχολείου σας. |
| 15 | Μπορούν οι μαθητές να ανεβάζουν τα δικά τους podcast κατευθείαν στην πλατφόρμα; | Ναι, οι μαθητές μπορούν να ανεβάζουν τα δικά τους podcast στην πλατφόρμα, αλλά μόνο εφόσον έχουν αναβαθμιστεί στην κατηγορία παραγωγών. Συγκεκριμένα:<br><br>*Οι μαθητευόμενοι, εάν βρίσκονται κάτω από την κατηγορία παραγωγών, μπορούν να ανεβάσουν τα δικά τους podcasts.<br>*Οι συντονιστές θα πρέπει να τα αποδεχτούν προκειμένου να δημοσιευτούν. |
| 16 | Υπάρχει κάπου να αποθηκεύω τα επεισόδια που μου αρέσουν για να τα ακούσω άλλη στιγμή; | Ναι, η πλατφόρμα προσφέρει δυνατότητα αποθήκευσης αγαπημένων επεισοδίων σε «Λίστα Αναπαραγωγής».<br>Συγκεκριμένα:<br><br>*Από τη «Βιβλιοθήκη» σας, υπάρχει η καρτέλα «Λίστα Αναπαραγωγής».<br>*Η πλατφόρμα προσφέρει τη δυνατότητα αποθήκευσης των αγαπημένων σας επεισοδίων σε λίστες αναπαραγωγής (Playlists), ώστε να μπορείτε να τα ακούσετε εύκολα αργότερα.<br>*Επίσης, στην καρτέλα «Αρέσει» εμφανίζονται τα επεισόδια στα οποία έχετε πατήσει τον χαρακτηρισμό «Μου αρέσει». |
| 17 | Τι είναι το περιεχόμενο H5P; | Το περιεχόμενο H5P (HTML5 Package) είναι ένας απλός τρόπος για τη δημιουργία και την κοινή χρήση πλούσιου και διαδραστικού περιεχομένου στον ιστό. Το H5P είναι αρθρωτό και αποτελείται από διάφορους τύπους περιεχομένου και εφαρμογές, ειδικά σχεδιασμένες για χρήση στην εκπαίδευση.<br><br>Στη δημοσίευση εκπαιδευτικού περιεχομένου H5P μπορείτε:<br><br>*Να επιλέξετε τύπο διαδραστικής δραστηριότητας από το πλαίσιο «Select content type».<br>*Να αναζητήσετε ανάμεσα σε 52 διαφορετικούς τύπους περιεχομένου, όπως:<br>**Interactive Video<br>**Course Presentation<br>*Να δημιουργήσετε νέο υλικό επιλέγοντας «Create Content».<br>*Να ανεβάσετε ένα ήδη υπάρχον αρχείο H5P από τον υπολογιστή σας μέσω της επιλογής «Upload». |
| 18 | Τι είναι το Podfolio; | Το Podfolio είναι μία επιλογή στη Βιβλιοθήκη του χρήστη που εμφανίζει όλες τις παραγωγές του χρήστη συγκεντρωμένες και οργανωμένες. |
| 19 | Είμαι εκπαιδευτικός. Από πού μπορώ να αποδεχτώ τους νέους μαθητές μου και τα επεισόδια που ανεβάζουν; | Ως εκπαιδευτικός, μπορείτε να αποδεχτείτε νέους μαθητές και τα επεισόδια που ανεβάζουν από τον «Πίνακα Ελέγχου» της εκπαιδευτικής μονάδας σας.<br>Ακολουθήστε τα παρακάτω βήματα:<br><br>*Από το μενού, επιλέξτε το όνομα χρήστη πάνω δεξιά.<br>*Στο πλαίσιο που εμφανίζεται, επιλέξτε «Διαχειριστές».<br>*Θα εμφανιστεί ο «Πίνακας Ελέγχου» της εκπαιδευτικής μονάδας σας.<br><br>Εκεί θα βρείτε τις επιλογές:<br><br>*«Εγκρίσεις Συντονιστών»: Εγκρίνετε έναν Χρήστη να πάρει τον ρόλο του Συντονιστή Ομάδας ή του Μαθητευόμενου της σχολικής μονάδας σας.<br>*«Εγκρίσεις Επεισοδίων»: Εγκρίνετε επεισόδια που ανεβάζουν οι Μαθητευόμενοι. |

| | | |
|---|---|---|
| | | *«Αναβάθμιση Μαθητευόμενων»: Δώστε σε κάποιον Μαθητευόμενο τη δυνατότητα να γίνει παραγωγός και να ανεβάζει παραγωγές στο προφίλ του σχολείου.<br>*«Εγκρίσεις Παραγωγών»: Επιβεβαιώστε την ανάρτηση κάποιας παραγωγής ενός αναβαθμισμένου Μαθητευόμενου. |
| 20 | Τι διαφορά έχουν τα Podcasts και οι Ραδιοφωνικές Εκπομπές; | Η διαφορά μεταξύ των Podcasts και των Ραδιοφωνικών Εκπομπών στην πλατφόρμα είναι η εξής:<br><br>*Podcasts:<br>**Υπάρχει δυνατότητα να μεταφορτώσετε τα δικά σας Podcasts.<br>**Τα Podcasts ΔΕΝ αναμεταδίδονται στο ραδιοφωνικό πρόγραμμα του σταθμού.<br>**Μπορείτε να τα διαχειριστείτε μέσα από την καρτέλα «Podcasts» στη Βιβλιοθήκη σας.<br>*Ραδιοφωνικές Εκπομπές:<br>**Μπορείτε να κάνετε κράτηση για live εκπομπή.<br>**Μπορείτε να διαχειριστείτε τις εκπομπές σας μέσα από την καρτέλα «Ραδιοφωνικές Εκπομπές».<br>**Οι Ραδιοφωνικές Εκπομπές σχετίζονται με το ραδιοφωνικό πρόγραμμα του σταθμού. |
| 21 | Πού μπορώ να βρω το έντυπο συγκατάθεσης γονέων; | Για να βρείτε το έντυπο συγκατάθεσης γονέων, από την αρχική οθόνη ακολουθήστε τα εξής βήματα:<br><br>*Πατήστε “Εκπαιδευτικό Υλικό” (πάνω δεξιά).<br>*Πηγαίνετε το ποντίκι πάνω από τον “Οδηγό Συμμετοχής” και επιλέξτε “Έντυπα - Δηλώσεις”.<br>*Στη λίστα αριστερά βεβαιωθείτε ότι είναι επιλεγμένο το “Δήλωση Ενημέρωσης - Συγκατάθεσης Γονέα”.<br>Κάντε κλικ στο σύνδεσμο “ΔΗΛΩΣΗ-ΕΝΗΜΕΡΩΣΗΣ-ΣΥΓΚΑΤΑΘΕΣΗΣ-2025-26” για να το κατεβάσετε. |
| 22 | Θέλω να βάλω γνωστά τραγούδια στην εκπομπή μου, θα έχω πρόβλημα με τα πνευματικά δικαιώματα; | Όχι, δεν θα έχετε πρόβλημα με τα πνευματικά δικαιώματα μουσικής και τραγουδιών.<br><br>*Το θέμα των ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ μουσικής και τραγουδιών έχει διευθετηθεί κεντρικά.<br>*Μπορείτε να χρησιμοποιείτε όποια μουσική επιθυμείτε στις εκπομπές σας. |
| 23 | Πώς κάνω τις ρυθμίσεις για ζωντανή εκπομπή; | Για να κάνετε τις ρυθμίσεις για μια ζωντανή εκπομπή, θα πρέπει να ακολουθήσετε τα παρακάτω:<br><br>*Αφού έχετε δηλώσει και προγραμματίσει σε προηγούμενο βήμα την ζωντανή σας εκπομπή, επικοινωνήστε μαζί μας μέσω του support@europeanschoolradio.eu για τις απαραίτητες ρυθμίσεις.<br>*Μπορείτε να κάνετε ζωντανή μετάδοση με οποιοδήποτε λογισμικό ψηφιακού DJ υποστηρίζει broadcasting (π.χ. Virtual DJ) ή με ειδικά λογισμικά broadcasting (π.χ. Max Broadcaster).<br>*Εμείς προτείνουμε το ελεύθερο λογισμικό Mixxx. |
| 24 | Γιατί δεν με αφήνει να διαγράψω ένα podcast; | Δεν σας αφήνει να διαγράψετε ένα podcast επειδή πρώτα πρέπει να διαγράψετε όλα τα επεισόδια που περιέχει.<br>Ακολουθήστε τα παρακάτω βήματα: |

| | | |
|---|---|---|
| | | *Μεταβείτε στη σελίδα του Podcast σας στην Βιβλιοθήκη σας.<br>*Επιλέξτε το Podcast από τη λίστα.<br>*Εντοπίστε το επεισόδιο που θέλετε να διαγράψετε και κάντε κλικ στο εικονίδιο διαγραφής του επεισοδίου.<br>*Αφού διαγράψετε όλα τα επεισόδια, επιστρέψτε στη σελίδα του Podcast.<br>*Εντοπίστε το Podcast που θέλετε να διαγράψετε και κάντε κλικ στις τρεις τελείες στην πάνω δεξιά γωνία του Podcast.<br>*Επιλέξτε «Διαγραφή». |
| 25 | Τι μικρόφωνο να πάρω και πόσο κοστίζει περίπου; | *Χρησιμοποιήστε ένα απλό μικρόφωνο που να συνδέεται εύκολα με τον υπολογιστή σας.<br>*Δοκιμάστε και ακούστε το αποτέλεσμα.<br>*Αν ακούγεται φύσημα, προσαρμόστε στο μικρόφωνο ένα υλικό που θα απορροφά τους θορύβους (ένα σφουγγαράκι).<br>*Αν έχετε διάθεση να ξοδέψετε κάποια χρήματα τότε μπορείτε να προμηθευτείτε ένα μικρόφωνο USB που έχουν εξαιρετικές επιδόσεις.<br>*Τα μικρόφωνα USB «κυμαίνονται περίπου από 40*150 ευρώ». |
| 26 | Γίνεται να βάλουμε το ραδιόφωνο να παίζει μέσα στο site του σχολείου μας; | Ναι, γίνεται να ακούγεται το ραδιόφωνο μέσα στο site του σχολείου σας. Σύμφωνα με τις οδηγίες της πλατφόρμας, μπορείτε να ενσωματώσετε έναν player στο site σας ώστε να παίζει το European School Radio απευθείας από εκεί.<br><br>Ακολουθήστε τα παρακάτω βήματα:<br><br>*Ενσωματώστε στο site σας τον παρακάτω κώδικα για να προσθέσετε τον player του ραδιοφώνου.<br>*Αν θέλετε διαφορετική εμφάνιση του player, αλλάξτε την τιμή:<br>'skin':'xavi3'<br>*Για περισσότερα skins, αναζητήστε στο διαδίκτυο:<br>muses official skins<br><br>Ο κώδικας που δίνεται στις οδηγίες είναι: <script type="text/javascript"><br>MRP.insert({<br>'url':'http://europeanschoolradio.eu:1351/esradio.mp3',<br>'lang':'auto',<br>'codec':'mp3',<br>'volume':60,<br>'introurl':'',<br>'fallback':'',<br>'radiotype':'icecast',<br>'autoplay':true,<br>'jsevents':false,<br>'buffering':1,<br>'title':' ',<br>'welcome':'European School Radio',<br>'bgcolor':'#343434',<br>'wmode':'transparent',<br>'skin':'xavi3',<br>'width':250,<br>'height':90<br>}); </script> Επιπλέον, αν το site σας φιλοξενείται στα Blogs του ΠΣΔ ή στο |

| | | |
|---|---|---|
| | | Schoolpress, μπορείτε να προσθέσετε τη μονάδα – widget “European School Radio” στο ιστολόγιό σας και να κάνετε τις ρυθμίσεις που σας ταιριάζουν. |
| 27 | Αν φτιάξω μια ομάδα για το σχολείο μου στην κοινότητα, μπορούν να τη βλέπουν όλοι; | Ναι, αυτό εξαρτάται από την επιλογή ιδιωτικότητας που θα ορίσετε κατά τη δημιουργία της ομάδας στην Κοινότητα. Συγκεκριμένα, στην καρτέλα «2. Settings» υπάρχουν οι εξής επιλογές:<br><br>*Public group: Οποιοδήποτε μέλος μπορεί να δει το περιεχόμενο και να συμμετάσχει ελεύθερα.<br>*Private group: Η συμμετοχή απαιτεί αίτημα και έγκριση, ενώ το περιεχόμενο είναι ορατό μόνο στα μέλη.<br>*Hidden group: Η ομάδα δεν εμφανίζεται σε καταλόγους αναζήτησης και η πρόσβαση γίνεται μόνο μέσω πρόσκλησης. |
| 28 | Σε τι μορφή πρέπει να είναι το αρχείο ήχου που θα ανεβάσω για την εκπομπή; | Το αρχείο ήχου που θα ανεβάσετε για την εκπομπή πρέπει να είναι σε μορφή mp3. Επίσης:<br><br>*Οι εκπομπές που μεταδίδονται είναι αρχεία ήχου μορφής mp3.<br>*Κάθε εκπομπή που υποβάλλεται για μετάδοση πρέπει να είναι ένα (1) ενιαίο αρχείο ήχου (mp3) που να περιέχει όλο το περιεχόμενο της εκπομπής. |
| 29 | Ποια είναι η διαφορά ανάμεσα στις προγραμματισμένες και τις μη προγραμματισμένες ραδιοφωνικές εκπομπές; | Η διαφορά ανάμεσα στις προγραμματισμένες και τις μη προγραμματισμένες ραδιοφωνικές εκπομπές είναι η εξής:<br><br>*Οι προγραμματισμένες εκπομπές είναι μισάωρες, μονόωρες ή δίωρες.<br>**Πρέπει να κάνετε «κράτηση» στο ημερολόγιο.<br>**Μπορείτε προαιρετικά να προσθέσετε αφίσα (banner) και σποτ (έως 1 λεπτό).<br>**Στη συνέχεια, από τη «Βιβλιοθήκη» σας κάνετε την ανάρτηση των ηχητικών αρχείων του κάθε επεισοδίου μέσω της επιλογής «Προσθήκη επεισοδίου».<br>**Οι εκπομπές αυτές παίζουν στη ζωντανή ροή του ραδιοφώνου την καθορισμένη ώρα και μέρα.<br>*Οι μη προγραμματισμένες εκπομπές έχουν διάρκεια από μερικά λεπτά έως ένα τέταρτο.<br>**Αναρτάτε απευθείας την εκπομπή σας χωρίς κράτηση στο ημερολόγιο.<br>**Ανάλογα με τον τύπο που θα επιλέξετε, αναπαράγονται στο ραδιόφωνο διαφορετικές ώρες και μέρες.<br>**Μπορεί να είναι κάποιο ραδιοφωνικό μήνυμα, κάποιο jingle, μια εκπομπή συγκεκριμένου θέματος ή οτιδήποτε άλλο επιθυμείτε. |
| 30 | Δεν έχω ξανακάνει εκπομπή, υπάρχει κάποιο μάθημα για το πώς να προετοιμαστώ και ποια προγράμματα να χρησιμοποιήσω; | Ναι, υπάρχουν διαθέσιμα διαδικτυακά μαθήματα για να σας βοηθήσουν να προετοιμαστείτε για την πρώτη σας εκπομπή και να μάθετε ποια προγράμματα μπορείτε να χρησιμοποιήσετε.<br><br>Για την προετοιμασία της εκπομπής σας:<br><br>*Πατήστε “Εκπαιδευτικό Υλικό” (πάνω δεξιά).<br>*Από το μενού, επιλέξτε “Διαδικτυακά Μαθήματα”.<br>*Βρείτε και επιλέξτε το μάθημα «Προετοιμάζοντας μια ραδιοφωνική εκπομπή».<br><br>Για τα προγράμματα ηχογράφησης και μοντάζ:<br><br>*Προτείνονται τα ελεύθερα λογισμικά Mixxx και Audacity. |

| | | |
|---|---|---|
| | | *Για αναλυτικές οδηγίες:<br>**Πατήστε “Εκπαιδευτικό Υλικό” (πάνω δεξιά).<br>**Από το μενού, επιλέξτε “Διαδικτυακά Μαθήματα”.<br>**Βρείτε και επιλέξτε το μάθημα «Λογισμικά ραδιοφωνικής παραγωγής & ραδιοφωνικής μετάδοσης».<br><br>Επιπλέον, υπάρχουν τρία αναλυτικά διαδικτυακά μαθήματα που μπορούν να σας βοηθήσουν:<br><br>*«Λογισμικά ραδιοφωνικής παραγωγής & ραδιοφωνικής μετάδοσης»<br>*«Ακουστική χώρου ηχογράφησης- μετάδοσης»<br>*«Υλικοτεχνικός εξοπλισμός για διαδικτυακή ραδιοφωνική παραγωγή» |
| 31 | Πού μπορώ να βρω πληροφορίες για το φεστιβάλ μαθητικού ραδιοφώνου; | Μπορείτε να βρείτε πληροφορίες για το Φεστιβάλ Μαθητικού Ραδιοφώνου στις ενότητες “Διαγωνισμοί / Δράσεις” και “Άλλες Δράσεις του European School Radio”. Συγκεκριμένα:<br><br>*Από το 2014 διοργανώνεται το Φεστιβάλ Μαθητικού Ραδιοφώνου, στο οποίο μπορούν να λάβουν μέρος όλες οι μαθητικές ραδιοφωνικές ομάδες, δηλώνοντας έγκαιρα συμμετοχή.<br>*Το Πανελλήνιο Φεστιβάλ Μαθητικού Ραδιοφώνου διοργανώνεται κάθε χρόνο σε διαφορετική πόλη της Ελλάδας.<br>*Σκοπός του είναι η γνωριμία, η επικοινωνία και η αλληλεπίδραση των μαθητών των ραδιοφωνικών σχολείων του European School Radio.<br>*Το Φεστιβάλ περιλαμβάνει βιωματικά εκπαιδευτικού χαρακτήρα για τη ραδιοφωνική παραγωγή, αλλά και καλλιτεχνικές εκδηλώσεις ψυχαγωγικού χαρακτήρα.<br>*Ενημερώνεστε τακτικά από τα “Νέα” στην Κοινότητα, αλλά και από τα Μέσα Κοινωνικής Δικτύωσης. |
| 32 | Τι είναι το «Κάν’το ν’ακουστεί»; | Το «Κάν’το ν’ακουστεί» είναι ο Μαθητικός Διαγωνισμός Podcast & Τραγουδιού του European School Radio, που διοργανώνεται από το 2015 και δίνει τη δυνατότητα στα παιδιά να εκφραστούν καλλιτεχνικά και δημοσιογραφικά μέσω podcast και τραγουδιού.<br><br>Σύμφωνα με το εγχειρίδιο της πλατφόρμας:<br><br>*Ο διαγωνισμός καλεί τη μαθητική κοινότητα να συνεργαστεί και να εκφραστεί δημιουργικά μέσω του podcast και του τραγουδιού σε θεματική της επιλογής τους.<br>*Στόχος του είναι να επιβραβεύονται οι προσπάθειες που ξεχωρίζουν, αναδεικνύοντας καλές πρακτικές για την ποιότητα περιεχομένου, την πρωτοτυπία και την έμπνευση.<br>*Ο διαγωνισμός περιλαμβάνει κατηγορίες για:<br>**Τραγούδι<br>**Podcast<br>*Η συμμετοχή γίνεται μέσω ειδικής φόρμας στην εφαρμογή του European School Radio.<br>*Ο διαγωνισμός περιλαμβάνει στάδια όπως:<br>**Υποβολές Συμμετοχών<br>**Ψηφοφορία Κριτών |

| | | |
|---|---|---|
| | | **Αξιολόγηση Συμμετοχών<br>**Ψηφοφορία Ακροατών |
| 33 | Πρέπει να γραφτώ κάπου πριν στείλω συμμετοχή στον διαγωνισμό; | Όχι, δεν απαιτείται προεγγραφή στον Διαγωνισμό. Αρκεί η υποβολή της συμμετοχής μέσω της ειδικής φόρμας στην εφαρμογή του European School Radio.<br><br>Ακολουθείτε τα εξής βήματα:<br><br>*Συνδέεστε στην ιστοσελίδα του "European School Radio".<br>*Επιλέγετε το σχολείο σας ως "Ενεργό Σχολείο".<br>*Επιλέγετε "Βιβλιοθήκη".<br>*Από τις "Συμμετοχές Διαγωνισμού", επιλέγετε "Υποβολή συμμετοχής".<br>*Συμπληρώνετε τη φόρμα και υποβάλλετε τη συμμετοχή σας εντός της προθεσμίας. |
| 34 | Είμαι δάσκαλος σε ειδικό σχολείο, με ποια σχολεία θα διαγωνιστούν οι μαθητές μου; | Εάν είστε εκπαιδευτικός σε ειδικό σχολείο, τότε μπορείτε να επιλέξετε μία από τις παρακάτω επιλογές για τη συμμετοχή σας στον Διαγωνισμό:<br><br>*Να διαγωνιστείτε ανάμεσα στα υπόλοιπα ειδικά σχολεία του Διαγωνισμού.<br>*Να διαγωνιστείτε στην κατηγορία που αντιστοιχεί στην ηλικιακή ομάδα των μαθητών/τριών σας από τα γενικά σχολεία. |
| 35 | Σε τι γλώσσα πρέπει να συμπληρώσω τη φόρμα για τον διαγωνισμό; | *Σχετικά με τη γλώσσα, θα πρέπει να συμπληρώνετε υποχρεωτικά όλα τα στοιχεία στα αγγλικά (όχι greeklish), ακόμα κι αν η συμμετοχή σας, δηλαδή το Podcast ή το τραγούδι, είναι στα ελληνικά.<br>*Αν η συμμετοχή είναι στην ελληνική γλώσσα, τότε συμπληρώνετε και μία ακόμα καρτέλα στα ελληνικά.<br>*Αν η συμμετοχή είναι σε κάποια άλλη γλώσσα, πατάτε το "+" και προσθέτετε τη γλώσσα αντίστοιχα. |
| 36 | Πώς μπορούν οι μαθητές μου να ακούσουν και να αξιολογήσουν τις άλλες συμμετοχές από το σπίτι τους; | Οι μαθητές μπορούν να ακούσουν και να αξιολογήσουν τις άλλες συμμετοχές από το σπίτι τους ως εξής:<br><br>*Ο/Η υπεύθυνος/η εκπαιδευτικός πρέπει να μπει στο "Σύστημα αξιολόγησης" με το ίδιο username που χρησιμοποιήθηκε για την υποβολή της συμμετοχής.<br>*Για κάθε συμμετοχή εμφανίζεται ένας μοναδικός "κωδικός ακρόασης", τον οποίο ο/η εκπαιδευτικός δίνει στους μαθητές και στις μαθήτριες της ομάδας.<br>*Οι μαθητές/τριες συνδέονται στη σελίδα "Αξιολόγηση Σχολείων".<br>*Πληκτρολογούν τον κωδικό ακρόασης που τους δόθηκε και πατούν "Δείτε τις συμμετοχές των άλλων σχολείων".<br>*Στη συνέχεια μπορούν να ακούσουν τις συμμετοχές της κατηγορίας τους.<br>*Για να δηλώσουν ποιες συμμετοχές τους αρέσουν, πατούν το σύμβολο της καρδιάς δίπλα σε κάθε τραγούδι ή Podcast.<br>*Ο/Η υπεύθυνος/η εκπαιδευτικός μπορεί να δει ποιες συμμετοχές επέλεξαν οι μαθητές/τριές του και να καταχωρήσει την τελική κατάταξη. |
| 37 | Πάτησα υποβολή στον διαγωνισμό αλλά βρήκα λάθος στα λόγια, πώς τα αλλάζω; | *Μετά την υποβολή, δε θα μπορείτε να επεξεργαστείτε τα ακριβή λόγια του podcast ή στίχο τραγουδιού (transcript), αλλά και το ηχητικό σας αρχείο.<br>*Για αυτό, δεν υπάρχει δυνατότητα αλλαγής στα λόγια αφού πατήσετε το κουμπί "Υποβολή". |
| 38 | Πώς βλέπω ποιές συμμετοχές άρεσαν | Για να δείτε ποιές συμμετοχές άρεσαν περισσότερο στους μαθητές και στις μαθήτριές σας, ακολουθήστε τα παρακάτω βήματα: |

| | | |
|---|---|---|
| | περισσότερο στους μαθητές μου; | *Συνδεθείτε στο “Σύστημα αξιολόγησης” με το ίδιο username που χρησιμοποιήσατε για την υποβολή της συμμετοχής.<br>*Από το Μενού “Κάν’το ν’ακουστεί” επιλέξτε “Σύστημα αξιολόγησης”.<br>*Επιλέξτε τη συμμετοχή για την οποία θέλετε να δείτε τις αξιολογήσεις των μαθητών/τριών σας.<br>*Δώστε στους μαθητές και στις μαθήτριές σας τον “κωδικό ακρόασης” ώστε να ακούσουν τις συμμετοχές και να επιλέξουν αυτές που τους αρέσουν πατώντας το σύμβολο της καρδιάς.<br>*Ως υπεύθυνος/η εκπαιδευτικός μπορείτε:<br>**να δείτε πόσοι μαθητές και μαθήτριες συμμετείχαν στη διαδικασία ακρόασης.<br>**να δείτε ποιές συμμετοχές έχουν επιλέξει οι μαθητές και οι μαθήτριές σας ως αγαπημένες.<br>*Μπορείτε να ακούσετε τι έχουν δηλώσει οι μαθητές/οι μαθήτριές σας ότι τους αρέσει και να το λάβετε υπόψη σας στη δημιουργία της τελικής λίστας κατάταξης. |
| 39 | Πρέπει να φτιάξουμε βίντεο για το τραγούδι μας; | Όχι, δεν είναι υποχρεωτικό να φτιάξετε βίντεο για το τραγούδι σας, καθώς η υποβολή του είναι προαιρετική.<br>Σύμφωνα με τις οδηγίες της πλατφόρμας, ισχύουν τα εξής:<br><br>*Το βίντεο - βίντεο κλιπ είναι προαιρετικό και αποτελεί την οπτικοποίηση του ηχητικού σας, είτε ήταν podcast είτε τραγούδι.<br>*Προτείνεται προαιρετικά για λόγους προώθησης και καλύτερης προβολής των συμμετοχών στο στάδιο της Ψηφοφορίας Ακροατών.<br>*Μπορείτε να συμπεριλάβετε ό,τι θέλετε στο βίντεο, όπως τα ονόματα των μαθητών/τριών σας και το σχολείο σας.<br>*Προτείνεται η μορφή του βίντεο να είναι mp4. |
| 40 | Είμαι κριτής. Πρέπει να αποθηκεύω την αξιολόγησή μου κάθε φορά που αλλάζω κάτι για να μην τη χάσω; | Όχι, δεν χρειάζεται να την αποθηκεύετε χειροκίνητα. Σύμφωνα με τις οδηγίες για την αξιολόγηση από κριτές:<br><br>*Κάθε φορά που αλλάζετε κάτι στην αξιολόγησή σας γίνεται αυτόματα αποθήκευση και ενημέρωση του μέσου όρου.<br>*Μπορείτε να επιβεβαιώσετε την προσωρινή αποθήκευση από την ημέρα και ώρα που εμφανίζονται στο πάνω μέρος.<br>*Μόλις ολοκληρώσετε την αξιολόγηση κάντε κλικ στο “Οριστικοποίηση Αξιολόγησης”. |