ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη Ευφυούς Συστήματος Αναζήτησης Ταξινομικής Πληροφορίας για Πατέντες»

Του φοιτητή
Φώτη Αλεξίου
Αρ. Μητρώου: 2020007

Επιβλέπων
Δρ Μιχαήλ Σαλαμπάσης
Καθηγητής

1 Σεπτεμβρίου 2025

Ανάπτυξη Ευφυούς Συστήματος Αναζήτησης Ταξινομικής Πληροφορίας για Πατέντες.
Κωδικός Δ.Ε. 25108
Ονοματεπώνυμο φοιτητή Αλεξίου Φώτιος
Ονοματεπώνυμο εισηγητή Μιχαήλ Σαλαμπάσης
Ημερομηνία ανάληψης Δ.Ε. 30/1/2025
Ημερομηνία περάτωσης Δ.Ε. 1/9/2025

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

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Στους ανθρώπους που με στηρίζουν στην πραγματοποίηση των στόχων μου»

Πρόλογος

Η επιλογή της ανάπτυξης ενός ευφυούς συστήματος αναζήτησης ταξινομικής πληροφορίας για πατέντες ως θέμα της διπλωματικής μου εργασίας, προέκυψε έπειτα από το ερευνητικό μου ενδιαφέρον γύρω από τον τομέα των ευρεσιτεχνιών και πιο συγκεκριμένα της προστασίας των πνευματικών δικαιωμάτων κάθε έργου. Επίσης, καθοριστικό κριτήριο αποτέλεσε η τεχνολογική υποδομή της εργασίας που βασίζεται σε σύγχρονες γλώσσες προγραμματισμού με συνεχή εξέλιξη. Θέλοντας λοιπόν να προσφέρω εμπράκτως στην ερευνητική κοινότητα που ασχολείται με τις ευρεσιτεχνίες, ένα νέο ευφυές σύστημα αναζήτησης, αναπτύχθηκε η παρούσα διαδικτυακή εφαρμογή.

Περίληψη

Ο σκοπός της παρούσας διπλωματικής εργασίας ήταν η ανάπτυξη ενός ευφυούς συστήματος αναζήτησης ταξινομικής πληροφορίας για πατέντες. Η εφαρμογή μέσω μια σύγχρονης διαδικτυακής διεπαφής, προσφέρει τη δυνατότητα σε ερευνητές να αναζητήσουν ταξινομικές πληροφορίες, ταυτόχρονα στους τρεις διεθνείς οργανισμούς EPO, USPTO, WIPO και συνολικά σε οκτώ διαφορετικούς ταξινομητές. Μέσα σε θεωρητικό πλαίσιο παρουσιάζεται η δομή των διεθνών συμβόλων IPC και CPC, γίνεται ανάλυση των εφαρμογών που διαθέτουν οι διεθνείς οργανισμοί και ιδιαίτερη έμφαση δίνεται στην εστίαση των προβλημάτων που δημιούργησαν την ανάγκη για την ανάπτυξη αυτής της εφαρμογής.

Στην εφαρμογή έχουν χρησιμοποιηθεί νευρωνικά μοντέλα τύπου Transformer για την ανάθεση συμβόλων σε μεγάλο όγκο κειμένων που έχουν εξαχθεί από ευρεσιτεχνίες. Η ανάπτυξη των νευρωνικών μοντέλων έχει υλοποιηθεί στο πλαίσιο του ερευνητικού έργου της υποψήφιας διδάκτορος Ελένης Καματέρη, χρησιμοποιώντας fine-tuned BERT μοντέλα ειδικά προσαρμοσμένα στις πατέντες, με στόχο την ανάλυση κειμένων με βάση λέξεις-κλειδιά που δίνουν οι χρήστες και τελικά αποδίδοντας τους αντίστοιχους κωδικούς ταξινόμησης. Η ροή εκτέλεσης της εφαρμογής ξεκινάει δίνοντας ένα ή πολλαπλά κείμενα, εκτελείται παράλληλη αναζήτηση σε όλους τους ταξινομητές και ολοκληρώνεται με την επιστροφή συμβόλων IPC/CPC στο επιθυμητό επίπεδο αναζήτησης.

Στην εργασία αποδεικνύεται μέσω πειραματικών δοκιμών η εγκυρότητα των αποτελεσμάτων καθώς κάθε αναζήτηση υπόκειται σε ανάλυση μέσω των μετρικών αξιολόγησης. Επιπλέον, επιβεβαιώνεται η αξιοπιστία της εφαρμογής καθώς εκτελείται απροβλημάτιστα ακόμα και υπό απαιτητικές συνθήκες. Τα παραπάνω αποτελέσματα προέκυψαν έπειτα από το λεπτομερή σχεδιασμό στην αρχιτεκτονική της εφαρμογής και της επιλογής κατάλληλης υποδομής για να την υποστηρίξει. Η δομή έχει στηθεί αξιοποιώντας τη δυναμική του Docker προσφέροντας ευελιξία και επεκτασιμότητα σε μελλοντικές αναβαθμίσεις. Καθώς η εφαρμογή αναπτύχθηκε για διαδικτυακή χρήση, δόθηκε ιδιαίτερη έμφαση στο σχεδιασμό της διεπαφής. Από τη δομή του HTML μέχρι την τελική μορφοποίηση του CSS, βασιστήκαμε σε μελέτες και τεχνικές σχεδιασμού που με έγκυρα δεδομένα προσφέρουν ένα εύχρηστο περιβάλλον χρήσης.

«Development of an Intelligent Patent Classification Information Search System»

«Fotios Alexiou»

Abstract

The purpose of this thesis was to develop an intelligent search system for taxonomic information in patents. The application, through a modern web interface, offers researchers the ability to search for taxonomic information simultaneously across three international organizations EPO, USPTO, WIPO, and across a total of eight different classifiers. Within a theoretical framework, the structure of international IPC and CPC symbols is presented, an analysis of applications available from international organizations is conducted, and particular emphasis is placed on focusing on the problems that created the need for developing this application.

The application utilizes Transformer-type neural models for assigning symbols to large volumes of text extracted from patents. The development of the neural models has been implemented within the framework of the research work of doctoral candidate Eleni Kamateri, using fine-tuned BERT models specifically adapted for patents, with the aim of analyzing texts based on keywords provided by users and ultimately assigning the corresponding classification codes. The application's execution flow begins by inputting one or multiple texts, performs parallel searching across all classifiers, and concludes with the return of IPC/CPC symbols at the desired search level.

The work demonstrates through experimental testing the validity of the results, as each search is subject to analysis through evaluation metrics. Additionally, the reliability of the application is confirmed as it operates smoothly even under demanding conditions. These results emerged following detailed design of the application's architecture and the selection of appropriate infrastructure to support it. The structure has been built utilizing the power of Docker, offering flexibility and scalability for future upgrades. Since the application was developed for web use, particular emphasis was given to interface design. From HTML structure to final CSS formatting, we relied on studies and design techniques that, with valid data, provide a user-friendly environment.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά όλους όσους συνέβαλαν στην περάτωση της διπλωματικής μου εργασίας. Συγκεκριμένα θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Μιχαήλ Σαλαμπάση, ο οποίος καθ' όλη τη διάρκεια της εργασίας συμμετείχε ενεργά με ουσιαστικές συμβουλές και στοχευμένες διορθώσεις, οι οποίες κατέληγαν κάθε φορά στη βελτίωση της εφαρμογής και στη διαμόρφωση πιο δομημένης προγραμματιστικής σκέψης. Επίσης, θέλω να ευχαριστήσω την υποψήφια διδάκτορα κα. Ελένη Καματέρη, που μέσω του ερευνητικού της έργου, της συνεισφοράς της με την παροχή υλικού και τις συνεχείς προτροπές για την εξέλιξη της εργασίας, συντέλεσε στην ολοκλήρωση της διπλωματικής εργασίας.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
CGN	Graph Convolutional Networks
CNN	Convolutional Neural Network
CORS	Cross-Origin Resource Sharing
CPC	Cooperative Patent Classification
FN	False Negatives
FP	False Positives
GIL	Global Interpreter Lock
IPC	International Patent Classification
JSON	JavaScript Object Notation
NLP	Natural Language Processing
RNN	Recurrent Neural Networks
TN	True Negatives
TP	True Positives
TTL	Time to Live
VPS	Virtual Private Server
UI	User Interface
UX	User Experience

Κεφάλαιο 1ο: Εισαγωγή

1.1 Σκοπός και στόχοι της εργασίας

Ο κύριος σκοπός της παρούσας διπλωματικής εργασίας ήταν η ανάπτυξη μιας ευφυούς διαδικτυακής εφαρμογής αναζήτησης ταξινομικής πληροφορίας που αφορά ευρεσιτεχνίες (πατέντες). Η εφαρμογή έχει ως στόχο να αυτοματοποιήσει τη διαδικασία ανάκτησης ταξινομικών πληροφοριών μέσω παράλληλης αναζήτησης σε πολλαπλά συστήματα. Μια από τις τεχνολογικές απαιτήσεις για την υλοποίηση της, ήταν η ανάπτυξη νευρωνικού μοντέλου ικανού να αναγνωρίζει και να κατηγοριοποιεί το περιεχόμενο πατεντών με υψηλή ακρίβεια. Μεγάλη πρόκληση αποτέλεσε η ενσωμάτωση τεχνικών εξόρυξης δεδομένων (web scraping) για τη συλλογή πληροφοριών από επίσημους ιστότοπους πατεντών, σε συνδυασμό με τη διασφάλιση λειτουργικών θεμάτων όπως η υψηλή ταχύτητα εκτέλεσης και η αδιάβλητη αξιοπιστία επιστροφής αποτελεσμάτων. Σε αντίθεση με παρόμοιες υλοποιήσεις αναζήτησης ευρεσιτεχνιών, η συγκεκριμένη εφαρμογή δεν επιτρέπει απλά στο χρήστη την αναζήτηση πατεντών μέσω της εισαγωγής λέξεων-κλειδιών χρησιμοποιώντας τη φυσική του γλώσσα. Ως καινοτομία διακρίνεται το γεγονός ότι μέσω μιας εύχρηστης διεπαφής προσφέρεται η δυνατότητα φόρτωσης κατάλληλα μορφοποιημένου αρχείου για παράλληλη αναζήτηση πολλαπλών ερωτημάτων. Η εφαρμογή υποστηρίζει πληθώρα μηχανισμών προς χρήση, όπως το φιλτράρισμα των κωδικών, τη σύγκριση και εξαγωγή των αποτελεσμάτων αλλά και την αποθήκευση του ιστορικού αναζητήσεων. Επιπλέον, για τη συνεχή αξιολόγηση του συστήματος εφαρμόζουμε και αποτυπώνουμε βασικές μετρικές ακριβείας, προσφέροντας ουσιαστική εικόνα για την ποιότητα της ταξινόμησης. Με τη χρήση σύγχρονων τεχνικών τεχνητής νοημοσύνης (ΤΝ) και μηχανικής μάθησης (ΜΜ), επιτυγχάνεται με ακρίβεια η λήψη αποτελεσμάτων κατηγοριοποιημένα με βάση τα διεθνή πρότυπα ταξινόμησης CPC (Cooperative Patent Classification) και IPC (International Patent Classification).

1.2 Ιστορική αναδρομή

Οι διαδικτυακές εφαρμογές, σε συνδυασμό με τη ραγδαία εξέλιξη της Τεχνητής Νοημοσύνης (ΤΝ), έχουν δημιουργήσει το ιδανικό περιβάλλον για την ανάπτυξη λύσεων, τόσο για τις ανάγκες των επιχειρήσεων όσο και για την υλοποίηση προσωπικών έργων. Ο συγκεκριμένος συνδυασμός που σήμερα αποτελεί αναπόσπαστο μέρος της καθημερινότητας των ανθρώπων, χρειάστηκε να διανύσει πολλές διαφορετικές καταστάσεις στο πέρασμα του χρόνου μέχρι να φτάσει στη τρέχουσα κατάσταση. Κατά τις δεκαετίες 1950 με 1970 αναπτύχθηκαν τα νευρωνικά δίκτυα (Neural Networks), που αποκαλούνται και τεχνητά νευρωνικά δίκτυα (Artificial Neural Networks) και τα οποία προσομοιάζουν τα ανθρώπινα βιολογικά νευρωνικά δίκτυα. Στις αρχές του 1990 ξεκίνησε η ανάπτυξη του web όπου μέχρι και το 1993 το περιεχόμενο ήταν μόνο στατικό χρησιμοποιώντας HTML, ενώ σταδιακά έως το 1995 με τη χρήση CGI πρωτοκόλλου είχαν δημιουργηθεί οι πρώτες δυναμικές Web εφαρμογές. Παράλληλα με την ανάπτυξη του διαδικτύου, από τη δεκαετία του 1980 είχε αρχίσει η άνθιση της μηχανικής μάθησης (Machine Learning), ενός υποσυνόλου της τεχνητής νοημοσύνης το οποίο αποτελούνταν από σύνολα μαθηματικών αλγορίθμων και μπορούσαν να αναλύουν αυτόματα δεδομένα. Η δημιουργία τεχνολογιών κατά την περίοδο 2000 έως το 2010, όπως AJAX, JavaScript frameworks και server-side γλωσσών προγραμματισμού (PHP, ASP.NET, Java) επέτρεψε την ανάπτυξη πιο διαδραστικών και δυναμικών εφαρμογών. Την ίδια περίοδο αναπτύχθηκαν χαρακτηριστικοί αλγόριθμοι της μηχανικής μάθησης τους οποίους χρησιμοποιούμε έντονα σήμερα, παραδείγματα των οποίων αποτελούν οι αλγόριθμοι αναγνώρισης ομιλίας και εικόνας, ομαδοποίησης

Κ-μέσων και το δέντρο αποφάσεων. Στα χρόνια που πέρασαν μέχρι να φτάσουμε στο σήμερα, η εμφάνιση των RESTful APIs και των microservices αρχιτεκτονικών επαναπροσδιόρισαν τον τρόπο ανάπτυξης web εφαρμογών. Η εξέλιξη της μηχανικής μάθησης οδήγησε στην ανάπτυξη της βαθιάς μάθησης (Deep Learning) έναν ειδικό τύπο νευρωνικού δικτύου που έχει ξεπεράσει πολλούς άλλους αλγορίθμους σε μεγάλα σύνολα δεδομένων. Η ενσωμάτωση της τεχνητής νοημοσύνης και της μηχανικής μάθησης σε web εφαρμογές αποτελεί μια από τις σημαντικότερες τάσεις της τελευταίας δεκαετίας. Αυτή η τάση έχει έντονη επίδραση στον τρόπο με τον οποίο αναπτύσσονται εξειδικευμένες εφαρμογές, συμπεριλαμβανομένων των συστημάτων διαχείρισης και αναζήτησης πατεντών.

1.3 Επισκόπηση του προβλήματος αναζήτησης ταξινομικής πληροφορίας σε πατέντες

Με ένα συνεχώς αυξανόμενο αριθμό καταχωρήσεων νέων ευρεσιτεχνιών, η μαζική αναζήτηση και η κατηγοριοποίηση πολλαπλών πατεντών, ακόμα και σήμερα αποτελεί μια πολύπλοκη διαδικασία που αντιμετωπίζει σημαντικές προκλήσεις τόσο σε τεχνικό όσο και σε εννοιολογικό επίπεδο. Οι παραδοσιακές τεχνικές αναζήτησης βασίζονται κυρίως στην προσέγγιση λέξεων-κλειδιών, με περιορισμένες δυνατότητες δυναμικών μηχανισμών, βασικές λειτουργικές ελλείψεις τόσο στη διαχείριση όσο και στην αποτύπωση των συμβόλων που αναζητά ο χρήστης και που τελικά επιστρέφονται στην οθόνη του. Σημαντική πρόκληση αποτελεί η ταξινομική πολυπλοκότητα, διότι τα διεθνή πρότυπα CPC και IPC αποτελούνται από χιλιάδες κατηγορίες και υποκατηγορίες με ιεραρχική δομή. Η επιλογή του κατάλληλου επιπέδου αναζήτησης απαιτεί βαθιά κατανόηση τόσο του τεχνικού περιεχομένου όσο και της δομής του ταξινομικού συστήματος καθιστώντας το δύσχρηστο σε έναν απλό χρήστη. Κύριο πρόβλημα στην αναζήτηση πατεντών είναι ο κατακερματισμός πληροφοριών σε πολλαπλές ανεξάρτητες βάσεις δεδομένων. Διεθνείς οργανισμοί όπως το European Patent Office (EPO), το United States Patent and Trademark Office (USPTO), το World Intellectual Property Organization (WIPO), και άλλες εθνικές βάσεις δεδομένων λειτουργούν ανεξάρτητα. Οι ερευνητές αναγκάζονται να πραγματοποιούν ξεχωριστές αναζητήσεις σε κάθε πλατφόρμα, με πλήρη απουσία μεθόδων σύγκρισης της πληροφορίας, ξοδεύοντας έτσι πολύτιμο χρόνο και λαμβάνοντας αμφίβολα αποτελέσματα. Η συγκεκριμένη απουσία ενός ενοποιημένου συστήματος, με καίριες λειτουργίες, που να επιτρέπει την ταυτόχρονη αναζήτηση σε πολλαπλές βάσεις δεδομένων που αφορούν ευρεσιτεχνίες, είναι ιδιαίτερα προβληματική για ερευνητικά ιδρύματα και εταιρείες που χρειάζονται να αναλύσουν μεγάλο όγκο δεδομένων. Πέρα από τις λύσεις που απαιτούν τα παραπάνω προβλήματα, ο τομέας της αναζήτησης ταξινομικής πληροφορίας σε πατέντες χαρακτηρίζεται από σημαντικούς περιορισμούς των διεθνών οργανισμών. Τα υπάρχοντα συστήματα αναζήτησης παρουσιάζουν μεμονωμένη λειτουργία βάσεων, με τα δεδομένα των πατεντών να προέρχονται από διαφορετικές πηγές και περιόδους και ως αποτέλεσμα αυτού να υπάρχουν ασυνέπειες ως προς το περιεχόμενο. Επίσης, η λειτουργία κάθε βάσης δεδομένων ως ανεξάρτητη οντότητα συνεπάγεται με τη χρήση ξεχωριστών APIs, την ίδια στιγμή που οι περισσότεροι οργανισμοί δεν προσφέρουν δημόσια πρόσβαση στα APIs για την άντληση πληροφοριών. Λαμβάνοντας υπόψη και την περιορισμένη αξιοποίηση σύγχρονων τεχνικών τεχνητής νοημοσύνης στις υπάρχουσες μηχανές αναζήτησης πατεντών, δημιουργείται μια ιδανική συνθήκη ανάπτυξης μιας νέας διαδικτυακής εφαρμογής που να εξασφαλίζει την ακρίβεια και την αποτελεσματικότητα. Η ανάπτυξη της συγκεκριμένης εφαρμογής αναλύεται λεπτομερώς στα επόμενα κεφάλαια και επιλύει όλα τα παραπάνω προβλήματα μέσω προεκπαιδευμένων γλωσσικών μοντέλων και τεχνικών εξόρυξης πληροφορίας.

1.4 Δομή κειμένου

Η παρούσα εργασία οργανώνεται σε έξι κύρια κεφάλαια, τα οποία παρουσιάζουν με συστηματικό τρόπο τη θεωρητική βάση, την ολοκληρωμένη ανάπτυξη της διαδικτυακής εφαρμογής και την αξιολόγηση του προτεινόμενου συστήματος μέσω πειραματικών δοκιμών.

Στο πρώτο εισαγωγικό κεφάλαιο παρουσιάστηκε ο σκοπός και οι στόχοι της εργασίας. Επίσης, έγινε ανασκόπηση στο ιστορικό πλαίσιο μέσω του οποίου εξελίχθηκαν οι web εφαρμογές και αναλύθηκε το πρόβλημα της αναζήτησης ταξινομικής πληροφορίας σε πατέντες.

Το δεύτερο κεφάλαιο εστιάζει στις βασικές έννοιες και τα σύγχρονα ερευνητικά αποτελέσματα που θεμελιώνουν την εργασία. Περιγράφονται λεπτομερώς οι δομές ταξινόμησης CPC και IPC, παρουσιάζονται τα υπάρχοντα συστήματα αναζήτησης ευρεσιτεχνιών και εξετάζονται προηγμένες τεχνικές μηχανικής μάθησης που εφαρμόζονται στην επεξεργασία της φυσικής γλώσσας. Το κεφάλαιο ολοκληρώνεται με κριτική επισκόπηση συναφών εργασιών, ώστε να αποτυπωθεί το ερευνητικό κενό και η προστιθέμενη αξία της προτεινόμενης λύσης.

Το τρίτο κεφάλαιο μεταφέρει τον αναγνώστη από τη θεωρητική θεμελίωση στην υλοποίηση. Περιγράφεται το νευρωνικό μοντέλο που έχει χρησιμοποιηθεί στην εφαρμογή, η διαδικασία προετοιμασίας των δεδομένων (tokenization, fine-tuning) και οι μηχανισμοί διαχείρισης των προβλέψεων, μέσω της εκπαίδευσης που έχει προηγηθεί σε αυτό. Επιπροσθέτως, γίνεται πλήρης παρουσίαση της οργάνωσης και εξόρυξης των δεδομένων μέσω τεχνικών Web Scraping, καθώς και η λειτουργία του μηχανισμού φιλτραρίσματος μεταξύ των διαφορετικών επιπέδων CPC και IPC. Ιδιαίτερη έμφαση δίνεται στην υποστήριξη της λειτουργίας μαζικής επεξεργασίας και αναζήτησης ερωτημάτων (Batch Processing) μέσω του ανεβάσματος κατάλληλα μορφοποιημένου XML αρχείου. Ακόμα, επεξηγούνται οι μετρικές ακρίβειας (Precision – Recall – F1 score) που απεικονίζονται στη διεπαφή και το κεφάλαιο κλείνει με πειραματικές μετρήσεις σύγκρισης, σε πραγματικές συνθήκες εκτέλεσης της εφαρμογής.

Η αρχιτεκτονική του συστήματος συμπεριλαμβανομένης και της υποδομής αναλύεται στο τέταρτο κεφάλαιο. Πιο συγκεκριμένα, αναφέρονται οι ανάγκες και οι πόροι που απαιτούνται από το server ώστε να τρέχει εύρυθμα η εφαρμογή, ενώ παράλληλα γίνεται αναφορά στη χρήση του microframework (Flask) που χρησιμοποιήθηκε. Εφαρμόζεται μεθοδολογική ανάλυση της διασφάλισης της φορητότητας μέσω χρήσης τεχνολογίας Docker, γίνεται παρουσίαση της διάσπασης των υπηρεσιών σε μικρο-υπηρεσίες (API gateway, worker nodes) και η επικοινωνία μεταξύ backend με το frontend. Αναλύονται επίσης οι πρακτικές αποθήκευσης στην προσωρινή μνήμη (caching), η διαχείριση των cookies, οι μηχανισμοί εξαγωγής των αποτελεσμάτων αναζήτησης, καθώς και ζητήματα ασφαλείας που προκύπτουν λόγω του web scraping (authentication, rate-limiting).

Στο πέμπτο κεφάλαιο εκτελούνται πειράματα σε πραγματικά σενάρια χρήσης, αξιολογούνται οι επιδόσεις ταξινόμησης και καταγράφονται οι χρόνοι απόκρισης. Τα αποτελέσματα συγκρίνονται με συμβατικές μεθόδους αναζήτησης, αναδεικνύοντας τα οφέλη αλλά και τους περιορισμούς της προσέγγισής μας. Το κεφάλαιο κλείνει με τη συνολική αποτίμηση των ευρημάτων και τεκμηριωμένες προτάσεις για μελλοντικές βελτιώσεις και επεκτάσεις.

Το έκτο και τελευταίο κεφάλαιο περιλαμβάνει τη βιβλιογραφία με τις αναφορές σε ερευνητικές δημοσιεύσεις, επιστημονικά βιβλία και άλλες πηγές που αξιοποιήθηκαν για την ολοκλήρωση της εργασίας.

Κεφάλαιο 2ο: Θεωρητικό υπόβαθρο και συναφής έρευνα

2.1 Εισαγωγή

Πριν από οποιαδήποτε πρακτική υλοποίηση ενός ευφυούς συστήματος αναζήτησης ταξινομημένων πατεντών, απαιτείται η πλήρης κατανόηση: (α) των ίδιων των ταξινομήσεων, (β) των υφισταμένων εργαλείων που τις διαχειρίζονται, (γ) του πώς η τεχνητή νοημοσύνη (ΤΝ) αναδιαμορφώνει το τοπίο ευρεσιτεχνιών και (δ) των σύγχρονων τεχνικών μηχανικής μάθησης (ΜΜ) στην επεξεργασία φυσικής γλώσσας (NLP). Το κεφάλαιο αυτό συγκεντρώνει τη σχετική βιβλιογραφία, αναδεικνύοντας το ερευνητικό κενό που καλύπτει η παρούσα εργασία.

2.2 Δομή και χρήση στις ταξινομήσεις πατεντών CPC & IPC

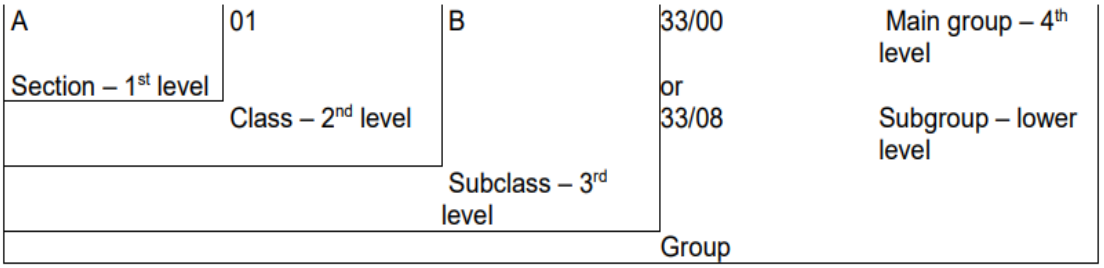
Η ορθή ταξινόμηση των τεχνικών πληροφοριών κρίνεται απαραίτητη για την αποδοτική διαχείριση των δεδομένων που αφορούν τις πατέντες. Πρώιμες μορφές ταξινόμησης για επιστημονικούς λόγους υπάρχουν τουλάχιστον από τον 4ο αιώνα π.Χ., όταν ο Αριστοτέλης χρησιμοποίησε ένα σύστημα για την ταξινόμηση των οργανισμών. Η έννοια της ταξινόμησης επεκτάθηκε με την πάροδο των αιώνων και σε άλλους τομείς της ανθρώπινης δραστηριότητας, συμπεριλαμβανομένων των τεχνικών και επιστημονικών γνώσεων. Στο πεδίο των ευρεσιτεχνιών, η συστηματική κατηγοριοποίηση των τεχνικών πληροφοριών καθιερώθηκε για να διευκολύνει την αναζήτηση, την αξιολόγηση και την ανάλυση των καινοτομιών. Τα δύο επικρατέστερα συστήματα ταξινόμησης διεθνώς είναι το International Patent Classification (IPC) και το Cooperative Patent Classification (CPC). Παρότι εμφανίζουν αρκετά κοινά στοιχεία στη δομή τους, διαφέρουν ουσιωδώς ως προς την εμβέλεια, τη λεπτομέρεια και τη χρήση τους. Προκειμένου να επιτευχθεί η σωστή κατάταξη και αναζήτηση των ευρεσιτεχνιών, είναι απαραίτητο πρώτα να κατανοηθεί πλήρως η εσωτερική αρχιτεκτονική των συμβόλων που αντιστοιχούν στις πατέντες. Παρακάτω αναλύεται η ιεραρχική δομή ταξινόμησης των δύο συστημάτων η οποία αποτελείται από πέντε κύρια επίπεδα.

2.2.1 Ιεραρχική δομή ταξινόμησης

1. **Section (Ενότητα):** Αποτελεί το ανώτατο επίπεδο, στο σύστημα IPC χαρακτηρίζεται από ένα κεφαλαίο γράμμα Α-Η, ενώ στο CPC στο εύρος γραμμάτων Α-Η προστίθεται ένα επιπλέον το Υ. Το κάθε γράμμα αντιστοιχεί σε μία ενότητα που αντιπροσωπεύει έναν ευρύ τεχνικό τομέα. Για παράδειγμα, η ενότητα Α αφορά τις «Ανθρώπινες Ανάγκες».
2. **Class (Κλάση):** Η κλάση δηλώνεται με δύο αριθμούς και συμπληρώνεται ακριβώς μετά το γράμμα της ενότητας. Η κλάση περιορίζει το τεχνικό εύρος της ενότητας σε πιο συγκεκριμένα πεδία. Για παράδειγμα, η κλάση Α01 αφορά τα εξής «Γεωργία, Δασοκομία, Κτηνοτροφία, Κυνήγι, Παγίδευση, Αλιεία».
3. **Subclass (Υποκλάση):** Η υποκλάση χαρακτηρίζεται με την προσθήκη ενός επιπλέον γράμματος στο τέλος της κλάσης. Στην πράξη η υποκλάση κατατμήσει περαιτέρω την τεχνική θεματική. Για παράδειγμα, η υποκλάση Α01C αναφέρεται στη «Φύτευση, Σπορά, Λίπανση».
4. **Main Group (Κύρια ομάδα):** Η κύρια ομάδα είναι η βασική εννοιολογική υποδιαίρεση εντός μιας υποκλάσης στις ταξινομήσεις. Ακολουθεί την υποκλάση, προσδιορίζεται με αριθμό και με το κάθετο σύμβολο ("/"). Ο τίτλος της κύριας

ομάδας ορίζει ένα διακριτό και επαρκώς ευρύ τεχνολογικό πεδίο, χρήσιμο ως σημείο αναφοράς για την αναζήτηση και την ευρετηρίαση των πατεντών. Παράδειγμα για την καλύτερη κατανόηση αποτελεί το σύμβολο A01C 11/00 με περιγραφή «Μηχανές μεταφύτευσης».

5. **Subgroup (Υποομάδα):** Η υποομάδα αποτελεί μια λεπτομερή ταξινομική υποδιαίρεση εντός μιας κύριας ομάδας (main group) στο σύστημα IPC και CPC. Αναγνωρίζεται από τη δεκαδική επέκταση του αριθμού της κύριας ομάδας, επιτρέποντας την περιγραφή εξειδικευμένων τεχνολογικών χαρακτηριστικών ή λειτουργικών πτυχών μιας ευρεσιτεχνίας. Το σύμβολο A01C 11/04 αποτελεί παράδειγμα επιπέδου υποομάδας με περιγραφή «για βαθύτερη φύτευση ή μετακίνηση φυτών» η οποία αναφέρεται στα εξειδικευμένα τεχνικά χαρακτηριστικά της ομάδας A01C 11/00.



Εικόνα 2.1: Πλήρες σύμβολο ταξινόμησης με το σύστημα IPC. Πηγή: [1].

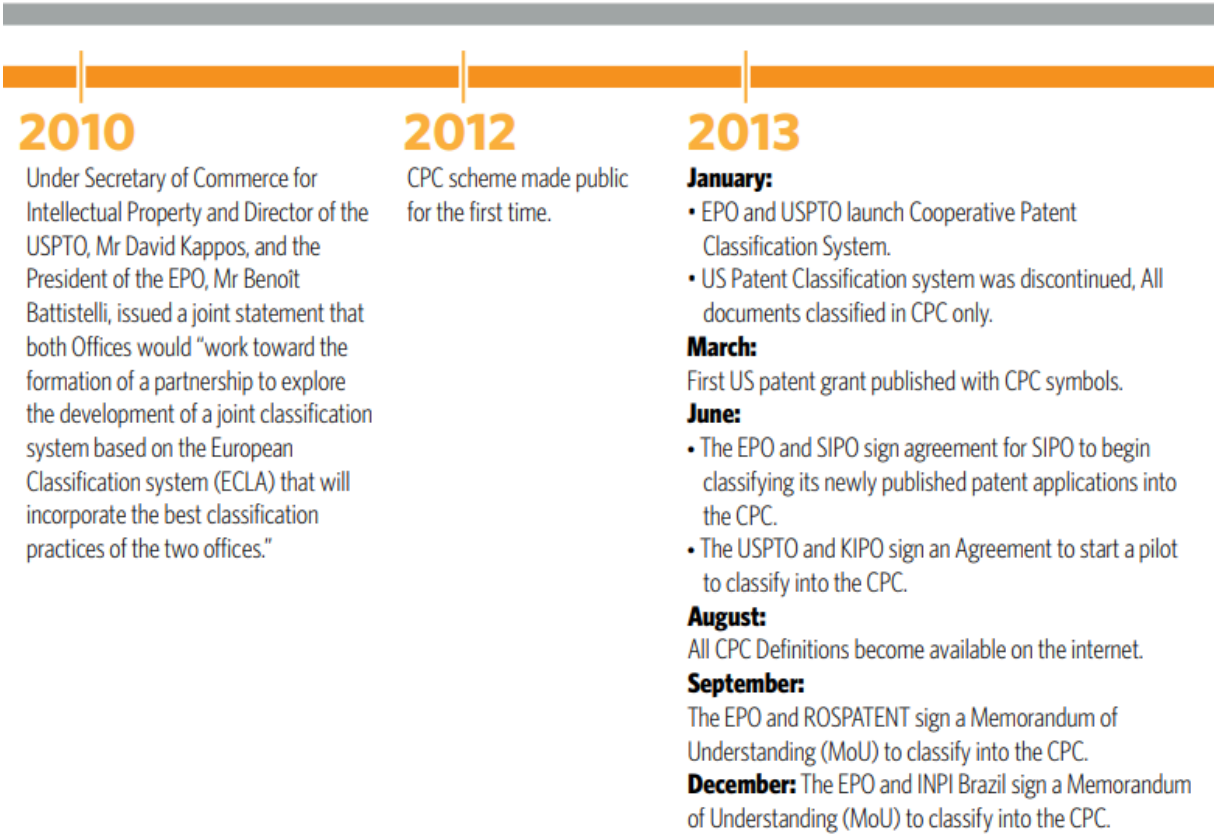
Στο σύστημα CPC, υπάρχουν ακόμα πιο λεπτομερείς υποομάδες ή ακόμα και εξαιρετικά εξειδικευμένες, γεγονός που σημαίνει ότι διαμορφώνονται περίπου 250.000 μοναδικές θέσεις ταξινόμησης, σε σύγκριση με τις περίπου 70.000 του συστήματος IPC.

2.2.2 Ανάλυση της χρήσης των συστημάτων IPC και CPC

Ο Παγκόσμιος Οργανισμός Πνευματικής Ιδιοκτησίας (WIPO) διαχειρίζεται από το 1967 το διεθνές πρότυπο International Patent Classification (IPC). Το πρότυπο IPC καθιερώθηκε με τη Συμφωνία του Στρασβούργου το 1971 και έκτοτε αποτελεί ένα ιεραρχικό σύστημα συμβόλων ανεξάρτητων από τη γλώσσα, για την ταξινόμηση διπλωμάτων ευρεσιτεχνίας και υποδειγμάτων χρησιμότητας σύμφωνα με τα διάφορα τεχνικά πεδία στα οποία ανήκουν [2]. Τα δεδομένα του IPC υπόκεινται σε συνεχή επικαιροποίηση, με νέες εκδόσεις να δημοσιεύονται περιοδικά. Η πιο πρόσφατη έκδοση, IPC 2025.01, τέθηκε σε ισχύ την 1η Ιανουαρίου 2025, σύμφωνα με επίσημη ανακοίνωση της WIPO. Παρά τις ενημερώσεις, η δομή της IPC παραμένει σχετικά σταθερή, διατηρώντας τη δυνατότητα ιστορικής αναδρομής και διεθνούς συμβατότητας.

Το Ευρωπαϊκό Γραφείο Διπλωμάτων Ευρεσιτεχνίας (EPO) σε συνεργασία με το Γραφείο Ευρεσιτεχνιών και Εμπορικών Σημάτων των Ηνωμένων Πολιτειών (USPTO) δημιούργησαν το 2013 το σύστημα Cooperative Patent Classification (CPC). Πρόκειται για επέκταση της IPC, εμπλουτισμένη με πρόσθετες υποκατηγορίες και λεπτομέρειες, ώστε να ανταποκρίνεται στις αυξημένες ανάγκες των εθνικών γραφείων διατηρώντας παράλληλα τη βασική ιεραρχική δομή. Ιδιαίτερο χαρακτηριστικό της CPC είναι η ύπαρξη της ενότητας Y, η οποία έχει σχεδιαστεί για την ταξινόμηση αναδυόμενων και διατομεακών τεχνολογιών, όπως η τεχνητή νοημοσύνη, οι τεχνολογίες κλιματικής προσαρμογής και τα ψηφιακά υπολογιστικά συστήματα. Το σύστημα CPC ενημερώνεται

διαρκώς, σχεδόν σε τριμηνιαία βάση διατηρώντας έτσι υψηλό επίπεδο ακρίβειας και αξιοπιστίας στην αναζήτηση.



Εικόνα 2.2: Patent Classification Timeline. Πηγή: [3].

2.3 Ανασκόπηση υπαρχόντων συστημάτων

Η ανάπτυξη ευφυών εφαρμογών αναζήτησης ταξινομικής πληροφορίας για ευρεσιτεχνίες, προϋποθέτει την πλήρη κατανόηση της λειτουργίας των υπαρχόντων συστημάτων αναζήτησης, που έχουν αναπτυχθεί από τους κύριους διεθνείς οργανισμούς, τόσο σε επίπεδο χρήσης όσο και της τεχνολογικής υποδομής τους. Οι τρεις κυριότεροι οργανισμοί με τα μεγαλύτερα datasets και τις πιο ευρέως χρησιμοποιούμενες διαδικτυακές εφαρμογές στον ερευνητικό χώρο είναι το Ευρωπαϊκό Γραφείο Διπλωμάτων Ευρεσιτεχνίας (EPO), το Γραφείο Ευρεσιτεχνιών και Εμπορικών Σημάτων των Ηνωμένων Πολιτειών (USPTO) και ο Παγκόσμιος Οργανισμός Πνευματικής Ιδιοκτησίας (WIPO). Κάθε ένας από αυτούς τους οργανισμούς έχει αναπτύξει ποικίλες μεθοδολογίες και τεχνολογικές λύσεις, παρέχοντας εργαλεία που επιτρέπουν την αναζήτηση σε τεράστιο όγκο δεδομένων που συνολικά ξεπερνάει τα 260 εκατομμύρια καταγεγραμμένα έγγραφα ευρεσιτεχνιών ταξινομικής πληροφορίας. Η μελέτη και η ανάλυση κάθε μιας από τις προσφερόμενες εφαρμογές αναδεικνύει πρωτίστως τις λύσεις που παρέχουν, ωστόσο οι πολλοί και διαφορετικοί περιορισμοί που υπάρχουν στα συγκεκριμένα συστήματα γίνονται άμεσα αντιληπτοί μέσω της χρήση τους. Στις επόμενες ενότητες αναλύονται ξεχωριστά τα προσφερόμενα συστήματα των τριών διεθνών οργανισμών, θέτοντας τα θεμέλια για την ανάπτυξη μιας εφαρμογής που εστιάζει κυρίως στη λύση των

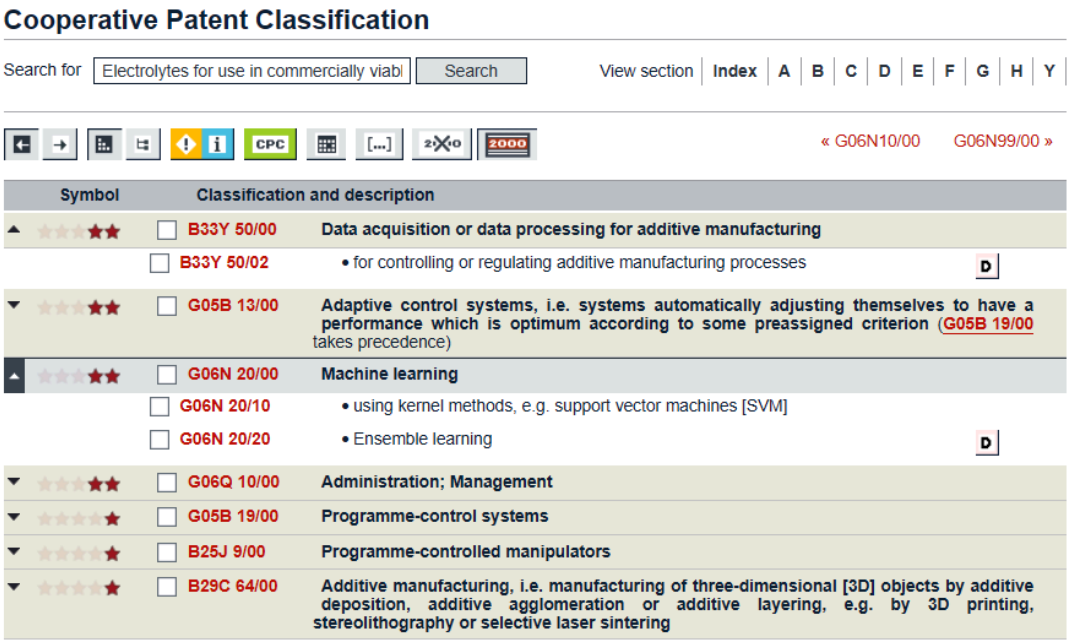
περιορισμών και στην επέκταση των δυνατοτήτων μέσω ενός ευφυούς συστήματος αναζήτησης ταξινομικής πληροφορίας βασισμένο στις ανάγκες των χρηστών.

2.3.1 Espacenet Patent Classification Search

Μέσω της πλατφόρμας Espacenet του Ευρωπαϊκού Γραφείου Διπλωμάτων Ευρεσιτεχνίας (EPO), οι χρήστες έχουν πρόσβαση σε περισσότερα από 160 εκατομμύρια καταγεγραμμένα έγγραφα ευρεσιτεχνιών, γεγονός που καθιστά το σύστημα ένα από τα πλέον δημοφιλή και πολύτιμα εργαλεία για ερευνητές και ακαδημαϊκούς. Το Espacenet λειτουργεί σε μια σύνθετη αρχιτεκτονική πολλαπλών βάσεων δεδομένων που λειτουργούν από κοινού για την παροχή ολοκληρωμένων πληροφοριών για τα διπλώματα ευρεσιτεχνίας. Η κύρια βάση δεδομένων είναι η **DOCDB,** περιέχει παγκόσμια βιβλιογραφικά δεδομένα από περισσότερες από 100 χώρες που χρονολογούνται από τη δεκαετία του 1830 [4]. Η αναζήτηση διπλωμάτων ευρεσιτεχνίας στην πλατφόρμα Espacenet αντλεί δεδομένα νομικής κατάστασης από τη βάση δεδομένων **INPADOC,** η οποία περιέχει μια συλλογή βιβλιογραφικών δεδομένων από έγγραφα διπλωμάτων ευρεσιτεχνίας και διαχειρίζεται κυρίως τις πληροφορίες του νομικού τους καθεστώτος.

Το Espacenet Patent Classification Search αποτελεί μια εξειδικευμένη πλατφόρμα πλοήγησης και αναζήτησης μέσα στο σύστημα ταξινόμησης πατεντών, βασισμένο στις διεθνείς δομές CPC (Cooperative Patent Classification). Το εργαλείο προσφέρει τη δυνατότητα αναζήτησης με λέξεις-κλειδιά, απεικονίζοντας τα αποτελέσματα σε ιεραρχική δομή δέντρου. Τα αποτελέσματα εμφανίζονται σε έναν πίνακα με την πρώτη στήλη να περιλαμβάνει τη συνάφεια του αποτελέσματος απεικονισμένη σε αστέρια, στη δεύτερη στήλη υπάρχει το CPC σύμβολο σε επίπεδο GROUP που επέστρεψε η αναζήτηση και στην τρίτη στήλη η περιγραφή του συμβόλου. Επίσης, για την καλύτερη πλοήγηση στα αποτελέσματα εφαρμόζεται η τεχνική επέκτασης και σύμπτυξης, πατώντας το εικονίδιο βέλους το οποίο βρίσκεται πριν από κάθε σύμβολο, εμφανίζονται οι υποομάδες (subgroup) του συμβόλου.

Ωστόσο, παρά τα πλεονεκτήματά του, το σύστημα έχει και ορισμένους περιορισμούς. Η εισαγωγή λέξεων-κλειδιών περιορίζεται αυστηρά σε έως 20 όρους αναζήτησης και 19 λογικούς τελεστές. Αξιολογήθηκε επίσης ως σημαντική η έλλειψη δυνατότητας μαζικών ερωτημάτων και επιστροφής αποτελεσμάτων σε μια αναζήτηση, περιορίζοντας έτσι τους χρήστες σε αποκλειστικά μεμονωμένες αναζητήσεις.

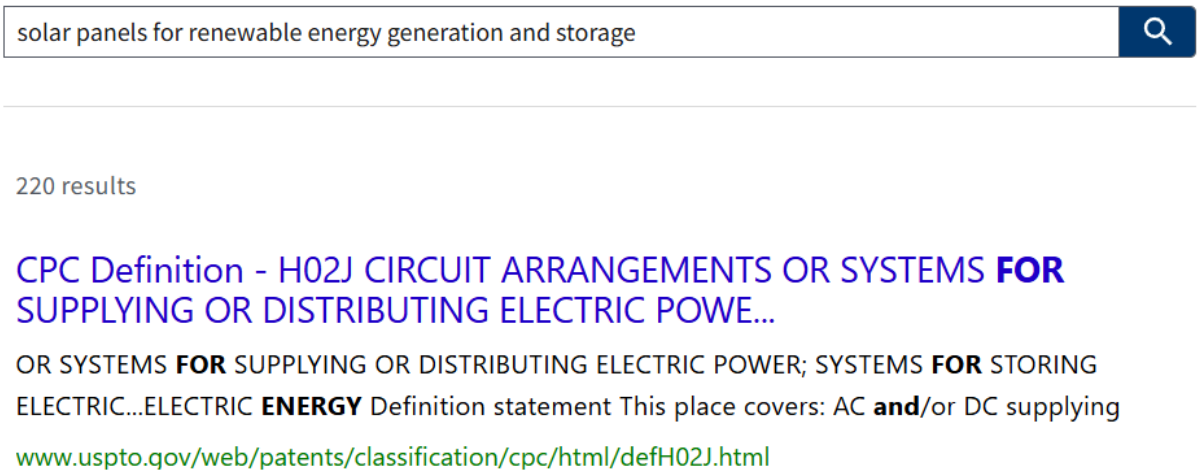


Εικόνα 2.3: Πίνακας αποτελεσμάτων αναζήτησης στο Espacenet

2.3.2 United States Patent and Trademark Office

Το Γραφείο Ευρεσιτεχνιών και Εμπορικών Σημάτων των Ηνωμένων Πολιτειών μέσω της εφαρμογής Search Patents Classification, για την αναζήτηση ευρεσιτεχνιών προσφέρει πρόσβαση στους χρήστες σε πάνω από 13 εκατομμύρια καταγεγραμμένα έγγραφα ευρεσιτεχνιών και δημοσιευμένων αιτήσεων, γεγονός που καθιστά το σύστημα ένα από τα πλέον αξιόπιστα. Η εφαρμογή αναζήτησης και τα επιστρεφόμενα από αυτή αποτελέσματα βασίζονται στο πλαίσιο Cooperative Patent Classification (CPC). Το σύστημα υλοποιείται κυρίως μέσω στατικών HTML σελίδων, οι οποίες περιλαμβάνουν CPC σύμβολα και περιγραφές με δυνατότητα πλοήγησης μέσω υπερσυνδέσμων. Ο χρήστης μπορεί να πραγματοποιεί αναζήτηση με λέξεις-κλειδιά μέσω του USPTO Search Affiliate engine που επιστρέφει αντίστοιχους CPC κωδικούς και οδηγεί στις επίσημες σελίδες ορισμών. Το βασικό πλεονέκτημα του συστήματος έγκειται στην αξιοπιστία και εγκυρότητα των δεδομένων, αφού οι περιγραφές προέρχονται απευθείας από το USPTO με τον ίδιο τον οργανισμό να έχει προσθέσει σύμβολα CPC σε όλα τα δημοσιευμένα έγγραφά του που χρονολογούνται από το 1836 και αυτό επιτεύχθηκε μέσω ενός συστήματος ηλεκτρονικής αντιστοιχίας [5].

Ωστόσο, η εφαρμογή εμφανίζει και ορισμένα μειονεκτήματα, όπως η περιορισμένη διαδραστικότητα λόγω της στατικής HTML. Ο χρήστης μέσω μιας φόρμας αναζήτησης, έχει τη δυνατότητα να αναζητήσει ευρεσιτεχνίες χρησιμοποιώντας λέξεις-κλειδιά. Βασικός περιορισμός στη συγκεκριμένη εφαρμογή είναι το αυστηρό μήκος χαρακτήρων προς αναζήτηση, που έχει δηλωθεί σε 128 χαρακτήρες το μέγιστο μέσω της ιδιότητας maxlength="128" στην HTML. Τα αποτελέσματα που επιστρέφονται είναι επιπέδου υποκλάσης (subclass), δημιουργώντας έναν ακόμα περιορισμό στο χρήστη καθώς δεν μπορεί να επιλέξει επίπεδο αναζήτησης αποτελεσμάτων. Η αναζήτηση και στη συγκεκριμένη πλατφόρμα περιορίζεται σε μεμονωμένα ερωτήματα, καθώς απουσιάζει η δυνατότητα ταυτόχρονης αναζήτησης πολλαπλών ερωτημάτων. Σε πιο τεχνική ανάλυση και συγκρίνοντάς το με παρόμοιες εφαρμογές όπως του Espacenet, παρατηρείται η απουσία APIs για την απευθείας άντληση CPC συμβόλων και περιγραφών που οδηγεί στην ανάγκη χρήσης τεχνικών scraping, καταλήγοντας σε μεγαλύτερους χρόνους φόρτωσης των αποτελεσμάτων και σε λιγότερο φιλική εμπειρία χρήστη.



Εικόνα 2.4: Πίνακας αποτελεσμάτων αναζήτησης στο USPTO

2.3.3 World Intellectual Property Organization

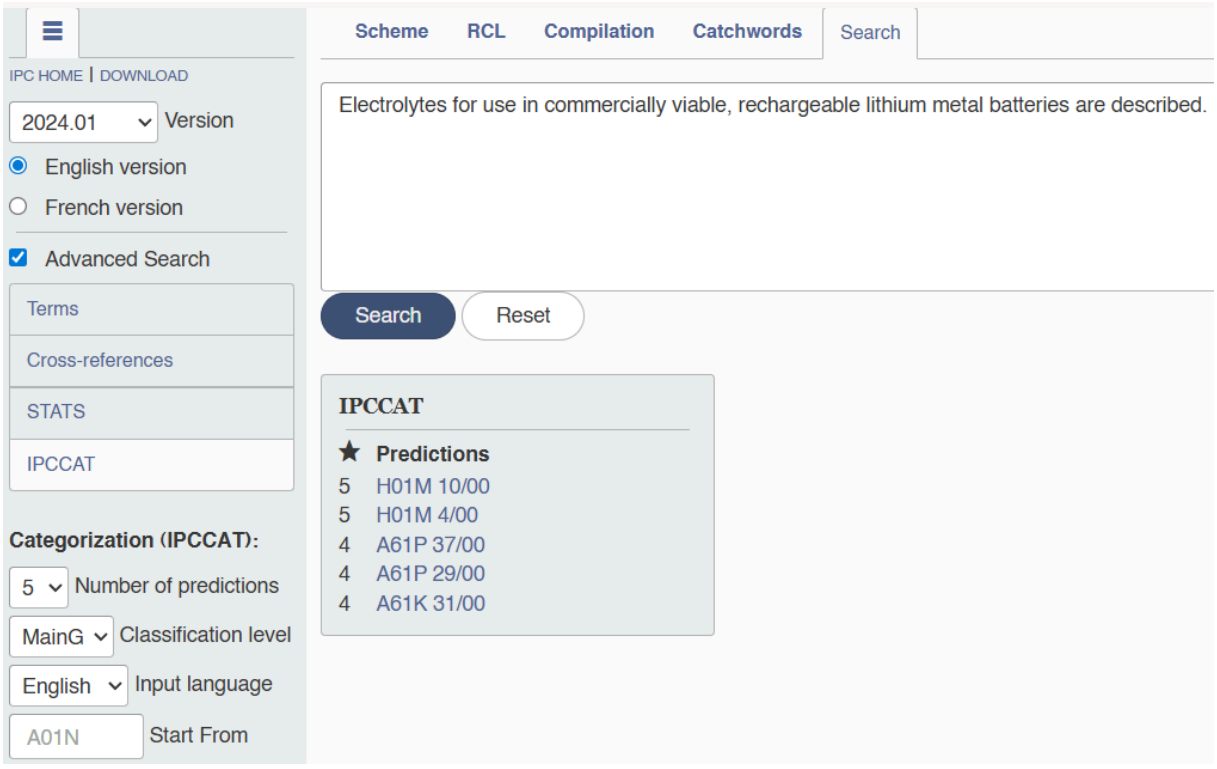
Ο Παγκόσμιος Οργανισμός Πνευματικής Ιδιοκτησίας (WIPO) προσφέρει μια πιο σύγχρονη εφαρμογή για την περιήγηση και αναζήτηση στις ταξινομήσεις ευρεσιτεχνιών, διαθέσιμο μέσω της πλατφόρμας ipcpub.wipo.int. Η συγκεκριμένη εφαρμογή βασίζεται στο πρότυπο International Patent Classification (IPC), η πλοήγηση γίνεται μέσω ενός σχήματος με δενδρική ιεραρχία ξεκινώντας από το επίπεδο section με τις πιο ευρείες θεματικές ενότητες και καταλήγοντας σε επιμέρους ομάδες (subgroups) με περισσότερη εξειδίκευση. Ο χρήστης έχει τη δυνατότητα να πλοηγηθεί στη δενδρική δομή ταξινόμησης είτε με αλφαβητική έναρξη είτε μέσω του IPCCAT smart search. Το IPCCAT smart search αποτελεί μια έξυπνη μηχανή αναζήτησης που δέχεται λέξεις-κλειδιά και επιστρέφει έως και πέντε συναφή IPC σύμβολα με συνδέσμους στις αντίστοιχες σελίδες περιγραφών και με υψηλό ποσοστό επιτυχημένης πρόβλεψης. Επιπλέον, οι περιγραφές και τα metadata παρέχονται σε πραγματικό χρόνο από αρχεία JSON, γεγονός που εξασφαλίζει ταχύτητα και συνέπεια στη διάθεση της πληροφορίας.

Η εφαρμογή προσφέρει δυνατότητες που δεν είχαν εντοπιστεί στα προηγούμενα συστήματα αναζήτησης. Πέρα από το εύχρηστο προς το χρήστη περιβάλλον και τη διαδραστική πλοήγηση με αναλυτικές περιγραφές, υπάρχει η δυνατότητα επιλογής επιπέδου αναζήτησης από class έως subgroup, η επιλογή γλώσσας εισαγωγής των λέξεων-κλειδιών αλλά και η δυνατότητα φιλτραρίσματος ώστε η εφαρμογή να ψάχνει πιο στοχευμένα σύμβολα, καθιστώντας το εργαλείο ιδιαίτερα χρήσιμο και πρωτοποριακό. Ο οργανισμός ενημερώνει συστηματικά τη βάση δεδομένων και ενσωματώνει κάθε νεότερη έκδοση που ανακοινώνεται για το IPC με την πιο πρόσφατη διαθέσιμη στην εφαρμογή της WIPO να είναι η 2026.01. Το σύστημα εφαρμόζει θεμελιώδη διαχωρισμό βάσει του μεγέθους του ερωτήματος αναζήτησης. Όταν ένα ερώτημα περιέχει τουλάχιστον 10 όρους, η εφαρμογή αναγνωρίζει την πολυπλοκότητα του περιεχομένου και ενεργοποιεί την αναζήτηση με τη χρήση του εργαλείου IPCCAT. Για ερωτήματα που αποτελούνται από λιγότερες από 10 λέξεις, το σύστημα εφαρμόζει μια εξαιρετικά οργανωμένη ιεραρχική διαδικασία αναζήτησης πέντε σταδίων που στοχεύει στη μεγιστοποίηση της ακρίβειας και της πληρότητας των αποτελεσμάτων.

Η πρώτη φάση της αναζήτησης επικεντρώνεται στους όρους του Σχήματος (Terms in Scheme), εξαιρώντας τις παραπομπές. Εάν η πρώτη φάση δεν επιστρέψει αποτελέσματα, το σύστημα μεταβαίνει

στην αναζήτηση στο Ευρετήριο Λέξεων-Κλειδιών (Terms in Catchwords). Το εργαλείο αναζήτησης επιστρέφει όλες τις θέσεις του IPC που αναφέρονται μέσω συνωνύμων και συναφών όρων που μπορεί να μην εμφανίζονται άμεσα στο κύριο σχήμα ταξινόμησης. Στο τρίτο στάδιο ενεργοποιείται το εργαλείο STATS (Statistical Analysis), το οποίο αναγνωρίζει τις θέσεις IPC που αναφέρονται συχνότερα στη βάση δεδομένων PATENTSCOPE όταν αναζητούνται οι συγκεκριμένοι όροι. Κομβικό σημείο σε αυτό το στάδιο αποτελεί το γεγονός πως αντί ο αλγόριθμος να εξετάζει όλες τις κατηγορίες IPC, αναγνωρίζει την πιο σχετική υποκλάση και περιορίζει την αναζήτηση μόνο στις ομάδες της συγκεκριμένης υποκλάσης, γεγονός που βελτιστοποιεί την ακρίβεια των αποτελεσμάτων. Το τέταρτο στάδιο εισάγει την τεχνολογία τεχνητής νοημοσύνης μέσω του συστήματος IPCCAT (IPC Computer-Assisted Categorization), το οποίο αποτελείται από χιλιάδες νευρωνικά δίκτυα που λειτουργούν παράλληλα. Το σύστημα έχει εκπαιδευτεί με εκτενής βάση δεδομένων 48.699.609 εγγράφων διπλωμάτων ευρεσιτεχνίας από τη συλλογή WIPO-delta 2024, χρησιμοποιώντας τον αλγόριθμο myClass του ιδρύματος Olanto [6]. Το πέμπτο και τελικό στάδιο αφορά την αναζήτηση στους Ορισμούς (Terms in Definitions). Η αναζήτηση στους ορισμούς παρέχει πρόσβαση σε εξειδικευμένες τεχνικές διευκρινίσεις που πιθανόν να περιέχουν τους όρους που ζητήθηκαν προς αναζήτηση. Η απόδοση και η αξιοπιστία του συστήματος αποδεικνύεται σε κάθε αναζήτηση μέσω βαθμονόμησης από ένα έως πέντε αστέρια πριν από κάθε IPC σύμβολο.

Παρόλο που η εφαρμογή διαθέτει αρκετές καινοτομίες συγκριτικά με όσες έχουν αναφερθεί έως τώρα, έχουν υπογραμμιστεί και εδώ οι δύο κοινές ελλείψεις. Η απουσία εισαγωγής μαζικών ερωτημάτων προς αναζήτηση δυσχεραίνει τη διαδικασία και καθυστερεί σημαντικά ερευνητές και απλούς χρήστες που θέλουν να θέσουν πολλαπλά ερωτήματα προς εύρεση. Εξίσου σημαντικό μειονέκτημα, αποτελεί και εδώ η απουσία δημοσίων προς πρόσβαση APIs ώστε να δίνεται πρόσβαση σε εξωτερικές εφαρμογές μέσω κλήσεων και να επιτυγχάνεται η άντληση των ζητούμενων συμβόλων και περιγραφών για τα IPC σύμβολα.



Εικόνα 2.5: Πίνακας αποτελεσμάτων αναζήτησης στο WIPO

2.4 Ευρεσιτεχνίες και τεχνητή νοημοσύνη

Η ενσωμάτωση της τεχνητής νοημοσύνης στον τομέα των ευρεσιτεχνιών αποτελεί σήμερα μια αναπόφευκτη πραγματικότητα που μεταμορφώνει τον τρόπο με τον οποίο γίνεται η διαχείριση και η ταξινόμηση των πατεντών. Παραδοσιακά, η αντιστοίχιση των ταξινομικών συμβόλων, όπως οι IPC και CPC κωδικοί, γινόταν από εξειδικευμένες εξεταστικές επιτροπές με βάση την εμπειρία τους και την προσεκτική ανάγνωση των κειμένων που είχαν κατατεθεί. Η πρακτική αυτή αν και είναι ακόμα ιδιαίτερα αξιόπιστη και αποτελεσματική, εντούτοις καθιστά τη διαδικασία ιδιαίτερα χρονοβόρα και δυσχεραίνει την έγκαιρη επεξεργασία του μεγάλου όγκου αιτήσεων που κατατίθενται διεθνώς. Η αξιοποίηση της τεχνητής νοημοσύνης προσφέρει σημαντικές νέες δυνατότητες κυρίως στην αυτοματοποίηση των χειροκίνητων διαδικασιών. Διεθνείς οργανισμοί όπως το USPTO, WIPO και το EPO έχουν ενσωματώσει μοντέλα μηχανικής μάθησης για τη διαδικασία απόδοσης κατάλληλων ταξινομικών συμβόλων, όπως επίσης και στις εφαρμογές αναζήτησης ευρεσιτεχνιών για τη βελτιστοποίηση των προβλέψεων [7].

Η εξέλιξη των μεθόδων deep learning έχει αναδιαμορφώσει το ερευνητικό πεδίο. Οι αρχικές προσεγγίσεις βασίστηκαν σε συνελικτικά νευρωνικά δίκτυα (CNN) και σε επαναληπτικά νευρωνικά δίκτυα (RNN), με τη χρήση στατικών ενσωματωμένων λέξεων (word embeddings). Στη συνέχεια, η εμφάνιση μοντέλων τύπου Transformer και ειδικότερα του BERT έφερε σημαντική βελτίωση. Η βασική ιδέα βασίζεται στη χρήση προ-εκπαιδευμένων νευρωνικών γλωσσικών μοντέλων, ώστε να αναλύουν τα κείμενα των πατεντών εστιάζοντας σε τίτλους, περιλήψεις ακόμα και στις αξιώσεις και να προβλέπουν με υψηλή ακρίβεια τους αντίστοιχους ταξινομικούς κωδικούς. Χαρακτηριστικό παράδειγμα αποτελεί η ανάπτυξη γλωσσικών μοντέλων, προσαρμοσμένα για χρήση σε συγκεκριμένα γλωσσικά περιβάλλοντα, όπως το ParsBERT για την περσική γλώσσα, το οποίο χρησιμοποιήθηκε επιτυχώς στην ταξινόμηση IPC κωδικών με επίτευξη validation accuracy 56% και test accuracy 40% [7]. Παράλληλα, η ερευνητική κοινότητα πρότεινε νέες μεθόδους που προχωρούν πέρα από τη βελτίωση της αναπαράστασης του κειμένου, ενσωματώνοντας και τις σχέσεις ανάμεσα στα ίδια τα ταξινομημένα σύμβολα.

Συστήματα όπως το PatentALL δεν περιορίζονται μόνο στην ανάγνωση του κειμένου μιας πατέντας, αλλά αξιοποιούν και τις σχέσεις που υπάρχουν ανάμεσα στα ίδια τα σύμβολα. Αυτό γίνεται με τη χρήση Graph Convolutional Networks (GCN), τα οποία μπορούν να εντοπίσουν όχι μόνο τις βασικές ιεραρχικές σχέσεις, αλλά και πιο σύνθετες διασυνδέσεις ανάμεσα σε διαφορετικές κατηγορίες [8]. Όταν η πληροφορία αυτή συνδυάζεται με γλωσσικά embeddings που προκύπτουν από το BERT, μέσω μηχανισμών attention, το σύστημα αποκτά μια πολυδιάστατη αναπαράσταση. Η ολοκληρωμένη αυτή προσέγγιση οδηγεί σε σημαντική βελτίωση της ακρίβειας του συστήματος, ιδίως σε long-tail labels δηλαδή σε ταξινομικές κατηγορίες που έχουν λίγες διαθέσιμες εγγραφές, όπου τα κλασικά μοντέλα εμφανίζουν συνήθως μειωμένες επιδόσεις.

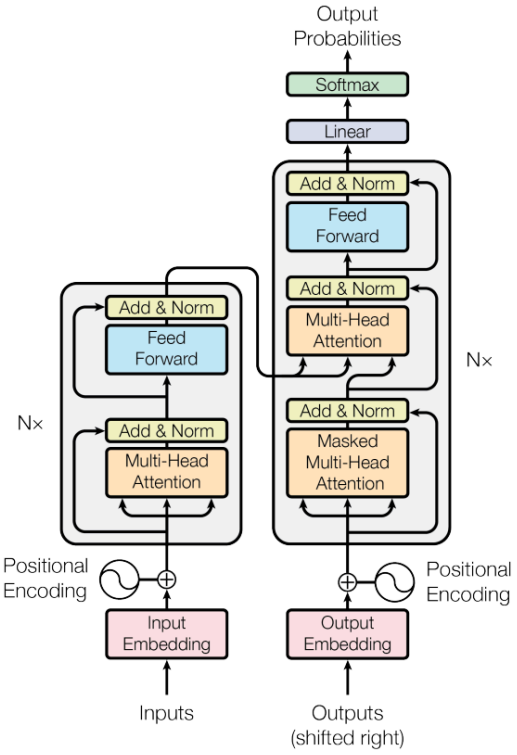
Συνολικά, η εφαρμογή της τεχνητής νοημοσύνης στον χώρο των ευρεσιτεχνιών έχει μετατραπεί από συμπληρωματικό εργαλείο σε βασικό μηχανισμό υποστήριξης της ταξινόμησης και της αναζήτησης. Από τις πρώτες εφαρμογές με κλασικά νευρωνικά δίκτυα έως τα σύγχρονα μοντέλα τύπου Transformer, η πρόοδος είναι εντυπωσιακή και συνεχής. Η σύγκλιση τεχνικών επεξεργασίας της φυσικής γλώσσας, γραφημάτων και μηχανισμών contrastive learning δείχνει ότι η μελλοντική πορεία θα στοχεύει σε ολοκληρωμένα συστήματα που δεν περιορίζονται απλώς στην απόδοση IPC/CPC συμβόλων, αλλά ενισχύουν τη στρατηγική ανάλυση, την πρόβλεψη τεχνολογικών τάσεων και τη λήψη αποφάσεων στον τομέα της βιομηχανικής ιδιοκτησίας.

2.5 Αρχιτεκτονική Μετασχηματιστών και Μεταφορά Μάθησης

2.5.1 Αρχιτεκτονική Transformer και Μηχανισμός Attention

Οι μετασχηματιστές (transformers) ως αρχιτεκτονική παρουσιάστηκαν πρώτη φορά από την ερευνητική ομάδα της Google Brain στο άρθρο "Attention Is All You Need" το 2017 και χρησιμοποιούνται κυρίως στον τομέα της επεξεργασίας φυσικής γλώσσας (NLP). Οι μετασχηματιστές εισήγαγαν την έννοια της πλήρους παραλληλοποίησης μέσω του μηχανισμού attention και σχεδιάστηκαν για το χειρισμό ακολουθιακών δεδομένων. Σε αντίθεση όμως με τα επαναλαμβανόμενα νευρωνικά δίκτυα, οι μετασχηματιστές δεν χρειάζεται να επεξεργάζονται τα δεδομένα με τη σωστή σειρά. Για παράδειγμα, αν ορίσουμε ως είσοδο μια πρόταση φυσικής γλώσσας, οι transformers δύνανται να επεξεργαστούν το τέλος της πρότασης παράλληλα με την αρχή της. Εξαιτίας αυτής της παραλληλοποίησης κατέστη δυνατή η αποδοτικότερη εκμάθηση μακροχρόνιων εξαρτήσεων και συνεπώς μικρότερος χρόνος εκπαίδευσης σε μεγάλο όγκο δεδομένων.

Η αρχιτεκτονική των μετασχηματιστών συγκροτείται από στοίβες κωδικοποιητών (encoders) και αποκωδικοποιητών (decoders). Ο κωδικοποιητής (encoder) αποτελείται από έξι ίδια στρώματα που στοιβάζονται το ένα πάνω στο άλλο, ενώ αντίστοιχα και ο αποκωδικοποιητής (decoder) περιλαμβάνει έξι παρόμοια στρώματα. Κάθε στρώμα του κωδικοποιητή ενσωματώνει δύο βασικά υποσυστήματα, ένα μηχανισμό πολυκεφαλικής αυτοπροσοχής (multi-head self-attention) και ένα πλήρως συνδεδεμένο δίκτυο προώθησης (feed-forward network) που εφαρμόζεται ξεχωριστά σε κάθε θέση της ακολουθίας. Ο αποκωδικοποιητής προσθέτει σε αυτή τη δομή ένα τρίτο υποσύστημα που υλοποιεί την αλληλεπίδραση μεταξύ κωδικοποιητή και αποκωδικοποιητή μέσω του μηχανισμού cross-attention. Όλα αυτά τα υποσυστήματα ενισχύονται με υπολειπόμενες συνδέσεις (residual connections) και κανονικοποίηση στρώματος για τη διασφάλιση σταθερότητας κατά την εκπαίδευση.

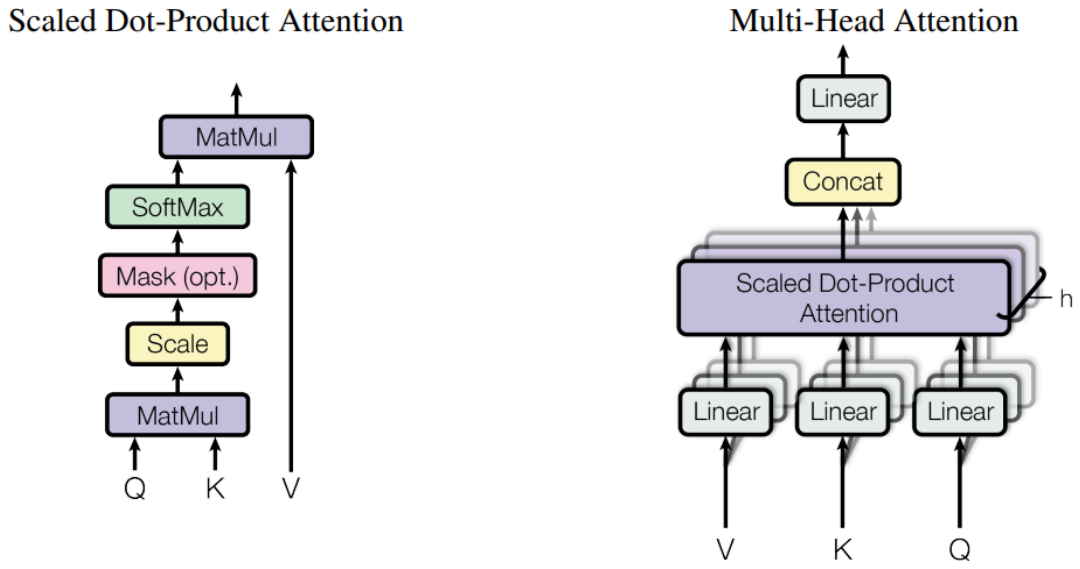


Εικόνα 2.6: Η αρχιτεκτονική των μετασχηματιστών. Πηγή: [9]

Ο μηχανισμός attention αποτελεί τον πυρήνα της λειτουργίας του μετασχηματιστή. Βασίζεται στον υπολογισμό τριών διανυσματικών αναπαραστάσεων για κάθε στοιχείο της εισόδου: ερωτήσεων (Queries), κλειδιών (Keys) και τιμών (Values). Η έξοδος προκύπτει ως στάθμιση των τιμών, με βάρη που υπολογίζονται από την ομοιότητα μεταξύ ερωτήσεων και κλειδιών, κανονικοποιημένη μέσω της συνάρτησης softmax. Η μαθηματική διατύπωση είναι:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.1)$$

όπου d_k είναι η διάσταση των διανυσμάτων κλειδιών. Η χρήση του παράγοντα κλίμακας $\frac{1}{\sqrt{d_k}}$ αποτρέπει την υπερβολική μεγέθυνση των εσωτερικών γινομένων, η οποία θα οδηγούσε τη συνάρτηση softmax σε περιοχές με εξαιρετικά μικρές κλίσεις και εξασφαλίζει πιο σταθερή εκπαίδευση. Επιπλέον, μέσω του μηχανισμού πολυκεφαλικής προσοχής (multi-head attention), οι υπολογισμοί εκτελούνται ταυτόχρονα σε πολλαπλούς υποχώρους αναπαράστασης, επιτρέποντας στο δίκτυο να συλλάβει διαφορετικού τύπου συσχετίσεις και εξαρτήσεις μέσα στην ίδια ακολουθία. Η διαδικασία αυτή συνοψίζεται σχηματικά στην εικόνα 2.7, όπου παρουσιάζεται τόσο ο μηχανισμός Scaled Dot-Product Attention όσο και η επέκτασή του σε Multi-Head Attention.



Εικόνα 2.7: Αρχιτεκτονική των μηχανισμών attention. Πηγή: [9]

Στα κείμενα ευρεσιτεχνιών συναντάμε συχνά εκτενείς και περίπλοκες τεχνικές περιγραφές, με έννοιες που αλληλοεξαρτώνται ακόμη και σε απομακρυσμένες θέσεις μέσα στο κείμενο. Ο μηχανισμός attention των μετασχηματιστών υπερτερεί, διότι επιτρέπει την άμεση σύνδεση δύο θέσεων της ακολουθίας χωρίς να απαιτείται σειριακή διάσχιση όλων των ενδιάμεσων στοιχείων. Τεχνικά, αυτό σημαίνει ότι η πολυπλοκότητα της πρόσβασης σε οποιαδήποτε θέση είναι O(1), δηλαδή σταθερός αριθμός πράξεων ανεξαρτήτως της απόστασης, σε αντίθεση με τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN) όπου η ίδια διαδικασία απαιτεί O(n) βήματα, δηλαδή έναν αριθμό πράξεων που αυξάνεται γραμμικά με το μήκος της ακολουθίας. Χάρη σε αυτή την ιδιότητα, οι μετασχηματιστές μπορούν να μοντελοποιούν αποδοτικά τις μακρές εξαρτήσεις που χαρακτηρίζουν τα κείμενα των

πατεντών, προσφέροντας μεγαλύτερη ακρίβεια και ταχύτερη εκπαίδευση στα συστήματα ταξινόμησης.

2.5.2 Προ-εκπαιδευμένα Γλωσσικά Μοντέλα και Μεταφορά Μάθησης

Τα προ-εκπαιδευμένα γλωσσικά μοντέλα άλλαξαν ραγδαία τον τρόπο που αντιμετωπίζονται τα προβλήματα στην επεξεργασία φυσικής γλώσσας. Μέχρι πρόσφατα, για κάθε νέα εργασία τα περισσότερα συστήματα εκπαιδεύονταν από την αρχή. Το BERT (Bidirectional Encoder Representations from Transformers), άλλαξε αυτή την προσέγγιση εισάγοντας την έννοια της διπλής κατεύθυνσης στην προ-εκπαίδευση. Ενώ τα παλαιότερα μοντέλα "διάβαζαν" το κείμενο από αριστερά προς τα δεξιά και το αντίστροφο, το BERT εκμεταλλεύεται τη δυνατότητα των Transformers να εξετάζουν ταυτόχρονα ολόκληρη την ακολουθία, δημιουργώντας πιο πλούσιες και συνεκτικές αναπαραστάσεις.

Η φιλοσοφία της μεταφοράς μάθησης (transfer learning) στα γλωσσικά μοντέλα βασίζεται σε δύο διακριτά στάδια. Πρώτα, στην προ-εκπαίδευση (pre-training), το μοντέλο εκπαιδεύεται σε πολύ μεγάλα σώματα κειμένων (corpora) χωρίς επιβλεπόμενη μάθηση (unsupervised learning), μαθαίνοντας γενικές γλωσσικές αναπαραστάσεις. Ένας τυπικός στόχος σε αυτή τη φάση είναι το Masked Language Modeling, όπου το μοντέλο καλείται να προβλέψει κρυμμένες λέξεις με βάση τα συμφραζόμενα. Στη συνέχεια ακολουθεί η φάση του fine-tuning, κατά την οποία το ήδη προ-εκπαιδευμένο μοντέλο προσαρμόζεται σε μικρότερα και πιο εξειδικευμένα σύνολα δεδομένων με την προσθήκη επιπλέον στρωμάτων πάνω από την προ-εκπαιδευμένη βάση, ανάλογα με την εκάστοτε εργασία. Με αυτόν τον τρόπο, αξιοποιούνται οι εμπλουτισμένες γλωσσικές αναπαραστάσεις που ενσωματώνουν συντακτικές και σημασιολογικές γνώσεις από την προ-εκπαίδευση, μειώνοντας σημαντικά το χρόνο και τους υπολογιστικούς πόρους που απαιτούνται για την εκπαίδευση.

Στον τομέα των ευρεσιτεχνιών, η αξία που προσδίδουν τα εξειδικευμένα μοντέλα γίνεται ιδιαίτερα εμφανής. Μοντέλα όπως το PatentBERT και το "anferico/bert-for-patents" που χρησιμοποιήθηκαν στην παρούσα εργασία, έχουν προ-εκπαιδευτεί σε εκτεταμένα κείμενα ευρεσιτεχνιών, αποκτώντας εξειδικευμένη γνώση πάνω στην τεχνική ορολογία και στις γλωσσικές ιδιαιτερότητες που διατυπώνονται στις πατέντες. Αυτό τα καθιστά κατάλληλα για εργασίες όπως η ταξινόμηση ευρεσιτεχνιών σε κατηγορίες CPC ή IPC, όπου η ακριβής κατανόηση των όρων και των συσχετίσεών τους είναι κρίσιμη.

Κεφάλαιο 3ο: Λειτουργικότητα Εφαρμογής

3.1 Εισαγωγή

Η ανάπτυξη της παρούσας ευφυούς διαδικτυακής εφαρμογής αναζήτησης ταξινομικής πληροφορίας, απαιτούσε ένα σύνολο λειτουργιών και τεχνολογιών ώστε να ξεπεράσει την πειραματική υλοποίηση και να μπορεί να υποστηρίξει παραγωγική χρήση. Σε αυτό το κεφάλαιο αναλύεται εκτενώς η χρήση των νευρωνικών μοντέλων τύπου transformers που χρησιμοποιήθηκαν στην εφαρμογή, οι τεχνικές web scraping και η δομή των APIs για την άμεση άντληση πληροφοριών. Ακολουθεί η ανάλυση του μηχανισμού φιλτραρίσματος επιπέδων CPC και IPC, ο οποίος επιτρέπει την ακριβή αναζήτηση ανάλογα με το επιλεγμένο επίπεδο ταξινόμησης. Επιπρόσθετα, περιγράφεται η λειτουργία batch mode, η οποία δίνει τη δυνατότητα επεξεργασίας πολλαπλών ερωτημάτων με τη μορφή αρχείου XML, ενώ αναλύεται τόσο η backend υποδομή όσο και το frontend περιβάλλον που υποστηρίζει τη λειτουργία αυτή. Τέλος, παρουσιάζονται οι μετρικές αξιολόγησης του συστήματος, οι οποίες χρησιμοποιούνται για την αποτίμηση της απόδοσης της εφαρμογής και των επιμέρους μηχανισμών συλλογής δεδομένων. Μέσω αυτής της ανάλυση καθίσταται σαφές πως η εφαρμογή συνδυάζει τεχνικές μηχανικής μάθησης, μεθοδολογίες web scraping και δομές αξιολόγησης, δημιουργώντας ένα ισχυρό εργαλείο για την αναζήτηση και ανάλυση ταξινομικής πληροφορίας των πατεντών σε διεθνές επίπεδο.

3.2 Σύστημα Ταξινόμησης Ευρεσιτεχνιών με Χρήση Fine-Tuned BERT

Για την αντιμετώπιση του προβλήματος της ταξινόμησης ευρεσιτεχνιών/πατεντών σε πολλές κατηγορίες βάσει των κειμενικών τους περιγραφών, αναπτύξαμε δύο νευρωνικά μοντέλα που βασίζονται σε αρχιτεκτονικές τύπου transformer. Συγκεκριμένα, χρησιμοποιήσαμε την παραλλαγή anferico/bert-for-patents [10] του BERT, η οποία είναι προεκπαιδευμένη σε έγγραφα ευρεσιτεχνιών, και τη βελτιστοποιήσαμε (fine-tuning) για να ταξινομεί ευρωπαϊκά κείμενα ευρεσιτεχνιών σε κωδικούς IPC (International Patent Classification) και CPC (Cooperative Patent Classification).

Οι δύο ταξινομητές αναθέτουν πιθανότητες σε ένα προκαθορισμένο σύνολο ετικετών σε επίπεδο ομάδας (4ο επίπεδο της ιεραρχίας IPC/CPC):

- Ο ταξινομητής CPC προβλέπει μεταξύ 9,135 κωδικών CPC.
- Ο ταξινομητής IPC προβλέπει μεταξύ 6,399 κωδικών IPC.

Αυτά τα μοντέλα υποστηρίζουν ταξινόμηση σε διαφορετικά επίπεδα λεπτομέρειας, από ολόκληρα κείμενα έως μεμονωμένες λέξεις ή τμήματα κειμένου, επιτρέποντας ευέλικτη κατηγοριοποίηση πατεντών. Οι ακόλουθες ενότητες περιγράφουν την αρχιτεκτονική του μοντέλου, τη διαδικασία επεξεργασίας δεδομένων και τη μεθοδολογία εκπαίδευσης.

3.2.1 Φόρτωση και Προεπεξεργασία Δεδομένων

Τα δεδομένα εκπαίδευσης προήλθαν από τη συλλογή WPI [11]. Συγκεκριμένα, χρησιμοποιήσαμε τις #EP πατέντες (core vertical) και τις ετικέτες IPC και CPC για την εκπαίδευση των ταξινομητών. Από τα διαθέσιμα έγγραφα ευρεσιτεχνιών του #EP core vertical, επιλέχθηκαν μόνο όσα διέθεταν πλήρη κείμενα στα πεδία περίληψης (abstract), περιγραφής (description) και αξιώσεων (claims). Από τις διαθέσιμες πατέντες στην #EP core vertical, διατηρήθηκαν μόνο αυτές που:

- Περιείχαν τους κωδικούς ταξινόμησης ενδιαφέροντος,

- Είχαν όλα τα κειμενικά πεδία πλήρη (περίληψη, περιγραφή και αξιώσεις), και
- Εάν υπήρχαν πολλαπλά έγγραφα για την ίδια πατέντα, διατηρήθηκε μόνο το τελευταίο έγγραφο.

Επίσης, οι κωδικοί IPC και CPC μετατράπηκαν σε μορφή ομάδας (π.χ. "G06F 17/30" → "G06F17") και τέλος, από τις ενότητες Περιγραφή και Αξιώσεις διατηρήθηκαν μόνο οι πρώτες 300 λέξεις.

Για την ανάπτυξη των ταξινομητών CPC και IPC, ο τίτλος και η περίληψη κάθε πατέντας συγχωνεύτηκαν για να σχηματίσουν την κειμενική είσοδο. Το συνδυασμένο κείμενο tokenize χρησιμοποιώντας BERT tokenizer με μέγιστο μήκος ακολουθίας 256 tokens, παράγοντας padded input_ids και attention_mask tensors κατάλληλα ως είσοδο για το μοντέλο BERT.

3.2.2 Κωδικοποίηση Ετικετών

Εφόσον η ταξινόμηση είναι πολυετικετική (κάθε ευρεσιτεχνία μπορεί να αντιστοιχεί σε πολλούς κωδικούς), οι ετικέτες κωδικοποιήθηκαν με χρήση του MultiLabelBinarizer. Ο κωδικοποιητής εκπαιδεύτηκε στο συνδυασμένο σύνολο ετικετών των splits εκπαίδευσης και δοκιμής για να εξασφαλιστεί η συνέπεια του χώρου ετικετών. Οι τελικοί στόχοι αναπαριστώνται ως δυαδικοί πίνακες, με κάθε θέση να δηλώνει την παρουσία ή απουσία ενός συγκεκριμένου κωδικού IPC/CPC.

3.2.3 Αρχιτεκτονική Μοντέλου

Έπειτα από πειραματισμούς με διάφορα μοντέλα, καταλήξαμε στην παρακάτω αρχιτεκτονική:

- **Επιλεγμένο μοντέλο:** Εξάγει τη [CLS] αναπαράσταση από την τελευταία κρυφή κατάσταση του BERT, τη διέρχεται από ένα dropout layer και στη συνέχεια από ένα πυκνό layer με sigmoid ενεργοποίηση για την πολυετικετική ταξινόμηση.

Ο ταξινομητής υλοποιήθηκε με χρήση του TensorFlow Keras API, με κατανεμημένη στρατηγική εκπαίδευσης (tf.distribute.MirroredStrategy) σε δύο 2 Nvidia A16 GPUs με χωρητικότητα 16 GB η κάθε μια.

3.2.4 Διαδικασία Εκπαίδευσης

Το μοντέλο εκπαιδεύτηκε με τον Adam optimizer, με ρυθμό εκμάθησης 1×10^{-5} και συνάρτηση κόστους binary cross-entropy. Η εκπαίδευση διήρκησε για 4 εποχές με μέγεθος παρτίδας (batch size) ίσο με 6. Τα δεδομένα εκπαίδευσης και επικύρωσης διαχειρίστηκαν μέσω pipelines TensorFlow tf.data.Dataset, συμπεριλαμβανομένου shuffling, batching και prefetching για βελτιστοποίηση της χρήσης GPU.

3.2.5 Αποθήκευση και Επαναχρησιμοποίηση Μοντέλου

Μετά την ολοκλήρωση της εκπαίδευσης, το τελικό μοντέλο αποθηκεύτηκε σε μορφή Keras .h5, διευκολύνοντας την επαναχρησιμοποίησή του ή την ενσωμάτωσή του σε εφαρμογές.

3.3 Εξόρυξης Πληροφορίας με Web Scraping

3.3.1 Εισαγωγή

Το web scraping αποτελεί μία από τις πιο σημαντικές τεχνικές αυτοματοποιημένης εξόρυξης δεδομένων. Συγκεκριμένα συλλέγει δεδομένα από διαδικτυακές πηγές, επιτρέποντας την επεξεργασία των πληροφοριών που διατίθενται μέσω HTML σελίδων, δυναμικών web εφαρμογών αλλά και σύνθετων διαδικτυακών πλατφορμών. Η τεχνική αυτή έχει εξελιχθεί από απλές HTTP αιτήσεις και

parsing HTML περιεχομένου σε σύνθετα συστήματα, τα οποία μπορούν να αλληλεπιδρούν με δυναμικές JavaScript-based εφαρμογές, να διαχειρίζονται cookie sessions και να προσομοιώνουν ανθρώπινη συμπεριφορά παρακάμπτοντας έτσι συστήματα ελέγχου εικονικών χρηστών (bot). Στα πλαίσια της συγκεκριμένης εργασίας στόχος ήταν η αποστολή του ερωτήματος στις υπάρχουσες μηχανές αναζήτησης πατεντών που δεν διέθεταν APIs και μέσω τεχνικών web scraping να επιτευχθεί η ανάκτηση κωδικών CPC / IPC και των περιγραφών τους.

Η αρχιτεκτονική που υιοθετήθηκε διαχωρίζεται λειτουργικά σε δύο κατηγορίες τεχνικών:

1) τις request–parse μεθόδους, όπου η δομή HTML ανακτάται μέσω HTTP GET και αναλύεται in-process με parsers τύπου BeautifulSoup4.
2) τις τεχνικές headless browser automation, στις οποίες εργαλεία όπως το Puppeteer εκκινούν μη ορατούς φυλλομετρητές (Chromium/Firefox/WebKit) για την πλήρη εκτέλεση της JavaScript, εξασφαλίζοντας πιστή αναδόμηση του τελικού DOM και διατήρηση του συνόλου της συνεδρίας όπως session cookies και local storage για επαναχρησιμοποίηση έως τη λήξη τους.

3.3.2 Τεχνική Ανάλυση Web Scraping Εργαλείων

Η λειτουργία της εφαρμογής στηρίζεται σε δύο κύριες τεχνολογίες web scraping, στη βιβλιοθήκη της Python που ονομάζεται BeautifulSoup4 και χρησιμοποιείται για την ανάλυση HTML και XML εγγράφων και στη βιβλιοθήκη Puppeteer της γλώσσας προγραμματισμού JavaScript. Κάθε εργαλείο εξυπηρετεί συγκεκριμένες λειτουργικές απαιτήσεις και διαχειρίζεται διαφορετικούς τύπους δεδομένων από τις βάσεις ταξινόμησης πατεντών. Για την άντληση πληροφοριών από κάθε classifier, έχουν αναπτυχθεί στην εφαρμογή αυτόνομες υπηρεσίες για την καλύτερη διαχείριση της εργασίας, τον ευκολότερο εντοπισμό λαθών και την ευελιξία σε πιθανή επέκτασή της.

3.3.3 Βιβλιοθήκη BeautifulSoup4

Η βιβλιοθήκη BeautifulSoup4 της Python αποτελεί την πιο διαδεδομένη επιλογή στην επεξεργασία στατικού περιεχομένου HTML. Η παρούσα εφαρμογή αξιοποιεί τη βιβλιοθήκη BeautifulSoup4 για την εξαγωγή CPC συμβόλων από την υπηρεσία USPTO στο αρχείο uspto_service.py. Συγκεκριμένα, μέσω της συνάρτησης fetch_group_level_from_subclass() για την εξαγωγή των CPC συμβόλων επιπέδου group μέσω των σελίδων ορισμού. Η δομή HTML αναλύεται με την BeautifulSoup4, η οποία τη μετατρέπει σε δενδρική δομή εγγράφου μέσω html.parser, επιτρέποντας την πλοήγηση στα στοιχεία μέσω στόχευσης από CSS selectors. Ενδεικτικά, ο επιλογέας div#definition-content > div.row χρησιμοποιείται για τον εντοπισμό των κύριων τμημάτων της σελίδας, ενώ για κάθε γραμμή εφαρμόζονται κλήσεις στη μέθοδο find() προκειμένου να ταυτοποιηθούν τα στοιχεία με κλάσεις classificationSymbol και defTitle.

Ένα ιδιαίτερο πλεονέκτημα της βιβλιοθήκης είναι η ευελιξία στη διαχείριση μικτού περιεχομένου τόσο κειμένου όσο και ετικετών HTML. Μέσω της επεξεργασίας του αντικειμένου contents κάθε κόμβου, είναι δυνατός ο διαχωρισμός απλού κειμένου από στοιχεία tags. Η ενσωματωμένη συνάρτηση process_element() αξιοποιεί αυτή τη δυνατότητα ώστε να αναγνωρίζει τον τύπο του στοιχείου (el.name), να εξάγει χαρακτηριστικά (el.has_attr()) και να αποδίδει καθαρό κείμενο μέσω της μεθόδου get_text(strip=True). Παράλληλα, παρέχεται η δυνατότητα δυναμικής τροποποίησης των υπερσυνδέσμων, ώστε οι σχετικές διευθύνσεις URL να μετασχηματίζονται από σχετικές σε απόλυτες και να καταλήγουν ορθά στις αντίστοιχες σελίδες ορισμών του USPTO. Η συνδυαστική επεξεργασία όλων των στοιχείων οδηγεί στη δημιουργία μιας ολοκληρωμένης και ενιαίας περιγραφής, η οποία

διατηρεί την αρχική μορφοποίηση HTML, εξασφαλίζοντας τόσο την ακρίβεια όσο και την πληρότητα της πληροφορίας που παρουσιάζεται στον χρήστη.

```python
@lru_cache(maxsize=256)
def fetch_group_level_from_subclass(subclass_symbol: str, max_groups: int = 1):
    import requests
    url = f"https://www.uspto.gov/web/patents/classification/cpc/html/def{subclass_symbol}.html"
    print(f"[USPTO] Fetching group-level from subclass: {subclass_symbol} → {url}")
    try:
        resp = requests.get(url, timeout=10)
        resp.raise_for_status()
    except Exception as e:
        print(f"[Group Fetch Error] {subclass_symbol}: {e}")
        return []

    soup = BeautifulSoup(resp.text, "html.parser")
    rows = soup.select("div#definition-content > div.row")

    results = []
    for row in rows:
        code_div = row.find("div", class_="classificationSymbol")
        title_div = row.find("div", class_="defTitle")
        if not code_div or not title_div:
            continue

        symbol_raw = code_div.get_text()
        symbol = re.sub(r"\s+", "", symbol_raw)
        if "/" not in symbol:
            continue

        anchor_symbol = symbol.replace("/", "-")
        link = f"https://www.uspto.gov/web/patents/classification/cpc/html/def{subclass_symbol}.html#{anchor_symbol}"

        def process_element(el):
            if isinstance(el, str):
                return el
            elif el.name == "a" and el.has_attr("href"):
                text = el.get_text(strip=True)
                match = re.search(r'([A-HY][0-9]{2}[A-Z])\s?([0-9]+/[0-9]+)', text)
                if match:
                    section = match.group(1).replace(" ", "")
                    group = match.group(2).replace(" ", "")
                    cpc_code = section + group
                    href = f"https://www.uspto.gov/web/patents/classification/cpc/html/def{section}.html#{cpc_code.replace('/', '-')}"
                    return f'<a href="{href}">{text}</a>'
                return text
            else:
                return el.get_text(strip=True)

        description_parts = [process_element(e) for e in title_div.contents]
        description = " ".join(description_parts).strip()
```

Εικόνα 3.1: Απόσπασμα κώδικα με τη χρήση της BeautifulSoup στο USPTO

3.3.4 Βιβλιοθήκη Puppeteer μέσω Node.js

Η πλατφόρμα της υπηρεσίας Espacenet αποτελεί Single Page Application (SPA) καθιστώντας τη διαδικασία της εξαγωγής συμβόλων IPC πολύπλοκη λόγω της εκτεταμένης χρήσης JavaScript για τη φόρτωση του περιεχομένου. Αυτή η αρχιτεκτονική καθιστά αδύνατη την εξαγωγή των απαιτούμενων πληροφοριών μέσω απλών τεχνικών ανάκτησης HTML όπως οι requests και BeautifulSoup, καθώς οι πληροφορίες δεν ενσωματώνονται στο αρχικό payload του διακομιστή, αλλά καθίστανται διαθέσιμες ύστερα από την πλήρη εκτέλεση JavaScript στον φυλλομετρητή του χρήστη (client-side rendering).

Για την αντιμετώπιση των δυσκολιών επιλέχθηκε η χρήση της JavaScript βιβλιοθήκης Puppeteer, ενός headless browser automation framework, μέσω ενός παράλληλου Node.js server. Η βιβλιοθήκη Puppeteer έχει τη δυνατότητα να εκκινεί, να αλληλεπιδρά και να εξάγει δεδομένα από ιστοσελίδες όπως ακριβώς θα το έκανε ένας πραγματικός χρήστης. Εξομοιώνει πλήρως το περιβάλλον ενός

κανονικού browser, συμπεριλαμβανομένης της εκτέλεσης JavaScript, του χειρισμού cookies καθώς και της διαχείρισης DOM events.

Η υλοποίηση βασίστηκε στη δημιουργία ενός παράλληλου διακομιστή Node.js, όπως αυτός ορίζεται στο αρχείο server.js, ο οποίος εκκινεί και διαχειρίζεται μία μόνιμη headless συνεδρία Puppeteer και παρέχει πρόσβαση σε αυτή μέσω ενός απλού REST API. Με αυτό τον τρόπο αποδεσμεύεται πλήρως το κύριο backend της εφαρμογής (Python/Flask) από την υπολογιστική επιβάρυνση της δυναμικής απόδοσης του DOM αποφεύγοντας επιπλέον χρόνους φόρτωσης. Το Node.js, ως server-side πλαίσιο βασισμένο στη μηχανή V8 της Google, παρέχει ένα ελαφρύ, event-driven μοντέλο εκτέλεσης, κατάλληλο για εφαρμογές I/O-bound όπως η συγκεκριμένη.

```
const puppeteer = require("puppeteer-extra");
const StealthPlugin = require("puppeteer-extra-plugin-stealth");

puppeteer.use(StealthPlugin());

let browser;
let page;

async function initBrowser() {
    browser = await puppeteer.launch({
        headless: true,
        executablePath: '/usr/bin/chromium',
        args: ['--no-sandbox', '--disable-setuid-sandbox']
    });
    page = await browser.newPage();
    await page.setUserAgent('Mozilla/5.0 (Windows NT 10.0; Win64; x64)
}
```

Εικόνα 3.2: Ρύθμιση του περιβάλλοντος εκτέλεσης της βιβλιοθήκης Puppeteer

Στο αρχείο server.js, διαπιστώνεται ότι το σύστημα εκκινεί έναν headless Chromium browser, ενώ εξίσου σημαντική είναι η χρήση του stealth plugin καθώς τροποποιεί τα χαρακτηριστικά του προφίλ χρήστη χωρίς να προδίδεται η αυτοματοποιημένη χρήση περιηγητή. Αμέσως μετά, εκκινείται μια νέα σελίδα (page = await browser.newPage()), η οποία διατηρείται ενεργή μειώνοντας δραματικά τον χρόνο απόκρισης στις επόμενες αναζητήσεις.

3.3.5 Αξιοποίηση JSON APIs

Η εφαρμογή ενσωματώνει έναν εναλλακτικό μηχανισμό εξαγωγής δεδομένων για την υπηρεσία του Παγκόσμιου Οργανισμού Πνευματικής Ιδιοκτησίας (WIPO). Αντίθετα με την προσέγγιση HTML parsing που εφαρμόζεται για το USPTO, η επικοινωνία με το WIPO πραγματοποιείται μέσω δομημένων JSON APIs που παρέχουν άμεση πρόσβαση στα IPC δεδομένα. Το αρχείο wipo_service.py υλοποιεί έναν υβριδικό μηχανισμό που συνδυάζει το IPCCAT "smart search" API για την αναζήτηση συμβόλων με το WIPO για την ανάκτηση λεπτομερών περιγραφών.

Η κεντρική συνάρτηση wipo_smart_search() δημιουργεί δυναμικά URLs προς το IPCCAT endpoint, ενσωματώνοντας παραμέτρους όπως την έκδοση του IPC (IPC_VERSION), το επίπεδο αναζήτησης μέσω του LEVEL_MAP, και τον αριθμό αποτελεσμάτων. Η επεξεργασία της JSON απόκρισης πραγματοποιείται μέσω της βιβλιοθήκης json της Python, η οποία αναλύει τα δομημένα δεδομένα και εξάγει τα HTML snippets που περιέχουν τα IPC σύμβολα. Η συνάρτηση extract_symbol_from_html() εφαρμόζει regular expressions για την αναγνώριση και εξαγωγή των συμβόλων από τα HTML fragments, ενώ η get_symbol_description() αξιοποιεί το WIPO για την ανάκτηση εμπλουτισμένων περιγραφών από τα αρχεία index.json και {SUBCLASS}.json.

Ιδιαίτερη έμφαση δίνεται στην κανονικοποίηση των αποτελεσμάτων μέσω της επεξεργασίας των συμβόλων ανά επίπεδο ταξινόμησης. Για τα group-level σύμβολα, η εφαρμογή αφαιρεί τα κενά διαστήματα μετασχηματίζοντας μορφές όπως "H02J 7/00" σε "H02J7/00", ενώ για τα section-level εξάγει αποκλειστικά το αρχικό γράμμα. Η διαχείριση των permalinks υλοποιείται μέσω της make_wipo_permalink(), η οποία για τα group σύμβολα δημιουργεί 14-ψήφιους symbolcodes συμβατούς με το σύστημα του WIPO, εξασφαλίζοντας τη σωστή πλοήγηση στις αντίστοιχες σελίδες περιγραφών. Παράδειγμα αυτής της μετατροπής αποτελεί το σύμβολο "H01M10/00" το οποίο με τη μετατροπή του σε 14-ψήφιο symbolcode γίνεται "H01M0010000000".

3.3.6 EPO Auth API

Σε αντίθεση με όλους του υπόλοιπους οργανισμούς, η επικοινωνία με τις υπηρεσίες του Ευρωπαϊκού Οργανισμού Διπλωμάτων Ευρεσιτεχνίας (EPO) και η άντληση πληροφοριών μπορεί να γίνει μέσω API. Το αρχείο epo_service.py ενσωματώνει έναν ολοκληρωμένο μηχανισμό authentication που διασφαλίζει την ασφαλή και αυθεντικοποιημένη πρόσβαση στα EPO OPS (Open Patent Services) APIs. Η διαδικασία ξεκινάει με τη συνάρτηση get_epo_access_token(), η οποία επικοινωνεί με το endpoint https://ops.epo.org/3.2/auth/accesstoken χρησιμοποιώντας μηχανισμό Basic Authentication για την απόκτηση access_token. Το κλειδί αυτό χρησιμοποιείται σε όλες τις επόμενες κλήσεις προς τα υπόλοιπα endpoints, εξασφαλίζοντας την έγκυρη και ασφαλή πρόσβαση.

Η συνάρτηση search_epo_classification(query, level_filter="group") επιτρέπει την αναζήτηση CPC classifications χρησιμοποιώντας έως δέκα λέξεις κλειδιά, αξιοποιώντας το endpoint https://ops.epo.org/3.2/rest-services/classification/cpc/search?q={query}. Η XML απάντηση επεξεργάζεται ώστε να εξαχθούν οι σχετικοί CPC κωδικοί και οι περιγραφές τους, οι οποίες εμπλουτίζονται με επιπλέον πληροφορίες από τα <cpc:explanation> στοιχεία. Μέσω αυτής της λειτουργικότητας, η εφαρμογή παρέχει στον χρήστη ενημερωμένα και επίσημα δεδομένα ταξινόμησης, αξιοποιώντας πλήρως τις δυνατότητες του EPO API σε συνδυασμό με τεχνικές parsing και παράλληλης εκτέλεσης. Στην εικόνα 3.3 που ακολουθεί παρουσιάζεται η επιστρεφόμενη δομή της απάντησης XML από το ερώτημα "fix a car" προς το EPO API.

```
<ops:world-patent-data xmlns:ops="http://ops.epo.org" xmlns:reg="http://www.epo.org/register"
  <ops:classification-search total-result-count="10" scheme-type="CPC">
    <ops:query syntax="">fix a car</ops:query>
    <ops:search-result>
      <ops:classification-statistics classification-symbol="A63H17/00" percentage="6.0150375">
        <cpc:class-title date-revised="2013-01-01">
          <cpc:title-part>
            <cpc:text>Toy vehicles, e.g. with self-drive; </cpc:text>
            <cpc:comment>
              <cpc:explanation>
                <cpc:text>
                  convertible into other toys
                  <cpc:class-ref scheme="cpc">A63H33/003</cpc:class-ref>
                </cpc:text>
              </cpc:explanation>
              <cpc:text>; Cranes, winches or the like;</cpc:text>
            </cpc:comment>
            <cpc:text> Accessories therefor </cpc:text>
            <cpc:explanation>
              <cpc:text>
                traffic games with figures moved by players
                <cpc:class-ref scheme="cpc">A63F9/14</cpc:class-ref>
              </cpc:text>
            </cpc:explanation>
          </cpc:title-part>
        </cpc:class-title>
      </ops:classification-statistics>
      <ops:classification-statistics classification-symbol="A63H33/00" percentage="3.7593985">
        <cpc:class-title date-revised="2013-01-01">
          <cpc:title-part>
            <cpc:text>Other toys</cpc:text>
          </cpc:title-part>
        </cpc:class-title>
      </ops:classification-statistics>
      <ops:classification-statistics classification-symbol="A63H3/00" percentage="3.0075188">
        <cpc:class-title date-revised="2013-01-01">
          <cpc:title-part>
            <cpc:text>Dolls </cpc:text>
            <cpc:comment>
              <cpc:explanation>
                <cpc:text>
                  puppets or marionettes for shows or theatres
                  <cpc:class-ref scheme="cpc">A63J19/006</cpc:class-ref>
                </cpc:text>
              </cpc:explanation>
            </cpc:comment>
          </cpc:title-part>
        </cpc:class-title>
      </ops:classification-statistics>
```

Εικόνα 3.3: XML απόκριση από το EPO API

Για την ανάκτηση των περιγραφών συγκεκριμένων CPC συμβόλων αξιοποιείται η συνάρτηση fetch_description_by_symbol(symbol, token), η οποία αποστέλλει αίτημα στο endpoint https://ops.epo.org/3.2/rest-services/classification/cpc/{symbol} και αναλύει την XML απόκριση με τη χρήση xml.etree.ElementTree. Παράλληλα, η extract_related_class_refs() επεξεργάζεται τους κόμβους <cpc:explanation> προκειμένου να αντλήσει συναφείς CPC κωδικούς, ενώ η enrich_explanation_with_links() μετατρέπει αυτούς τους κωδικούς σε ενεργούς υπερσυνδέσμους μέσω κανονικών εκφράσεων, προσφέροντας άμεση πλοήγηση στο Espacenet. Επιπλέον, η συνάρτηση fetch_multiple_descriptions() υποστηρίζει μαζική ανάκτηση περιγραφών με τη βοήθεια του ThreadPoolExecutor, μειώνοντας σημαντικά τον χρόνο απόκρισης.

3.3.7 REST APIs

Το σύστημα εκθέτει ένα ολοκληρωμένο σύνολο από REST API endpoints που είναι διαχωρισμένα σε δύο κύριες κατηγορίες ανάλογα με τη λειτουργικότητά τους και την υπηρεσία που τα διαχειρίζεται. Ξεκινώντας με την κύρια εφαρμογή Flask και τα διαθέσιμα endpoints υλοποιημένα στην port 5000, μέσω του αρχείου app.py γίνεται κεντρικοποιημένη διαχείριση από το πρώτο στάδιο αναζήτησης ευρεσιτεχνιών έως τις πιο σύνθετες ασύγχρονες επεξεργασίες. Η αρχιτεκτονική των endpoints ακολουθεί τις αρχές σχεδιασμού RESTful και διακρίνονται στα εξής :

- / (GET, POST) - Root endpoint αποτελεί το κεντρικό σημείο εισόδου για την εκτέλεση αναζητήσεων ταξινόμησης. Δέχεται παραμέτρους όπως το query προς αναζήτηση (user_input), το επίπεδο ταξινόμησης (level), και τον τύπο ταξινόμησης (classification_type). Επιπλέον, συντονίζει την παράλληλη εκτέλεση των υπηρεσιών, δημιουργώντας ένα ολοκληρωμένο Flask Response object μέσω make_response(), το οποίο ενσωματώνει τα αποτελέσματα από όλους τους classifiers στο index.html μέσω του προτύπου Jinja2.
- /search-async (POST) - Για την υποστήριξη των ασύγχρονων λειτουργιών, δημιουργείται ένα μοναδικό search_id με χρήστη του uuid.uuid4. Εκτελείται στο παρασκήνιο της εφαρμογής η συνάρτηση run_all_classifiers_parallel() μέσω threading.Thread και γίνεται παράλληλη εκκίνηση των επιλεγμένων ταξινομητών. Επιστρέφει άμεσα JSON response με πληροφορίες για ευκολότερη παρακολούθηση της ροής εκτέλεσης.
- /search-status/<search_id> (GET) - Παρέχει σε πραγματικό χρόνο ενημερώσεις για την κατάσταση εκτέλεσης της κάθε υπηρεσίας. Επίσης ανακτά δεδομένα από το search_store.get_search_status() και υπολογίζει τις μετρικές απόδοσης μέσω της συνάρτησης compute_prf1() για κάθε classifier.
- /cancel-search/<search_id>/<classifier> (POST) - Επιτρέπει τη διακοπή συγκεκριμένου classifier σε ενεργή αναζήτηση μέσω της υλοποιημένης συνάρτησης cancel_search_classifiers().
- /cancel-search-all/<search_id> (POST) - Ακυρώνει όλους τους classifiers για συγκεκριμένη αναζήτηση που εκτελείται και χρησιμοποιείται κυρίως για batch cancellation.
- /process-batch-json (POST) - Έχει υλοποιηθεί για τη διαχείριση της λειτουργίας μαζικής αναζήτησης. Υποστηρίζει την παράλληλη επεξεργασία πολλαπλών αιτημάτων αναζήτησης ταξινομικής πληροφορίας ακόμα και διαφορετικού τύπου IPC ή CPC, με τη χρήση του ThreadPoolExecutor και παρουσιάζει συγκεντρωτικά τα αποτελέσματα.
- /cleanup-search/<search_id> (POST) - Αποτελεί ένα από τα διαχειριστικά endpoints, λειτουργεί ως μηχανισμός garbage collection για την εκκαθάριση των μεταδεδομένων από τις ολοκληρωμένες ασύγχρονες αναζητήσεις από το SearchStore της μνήμης και καλείται αυτόματα από το frontend JavaScript μέσω της μεθόδου cleanupPreviousSearches().
- /search-stats (GET) - Παρέχει σε πραγματικό χρόνο στατιστικά παρακολούθησης του συστήματος σε μορφή json. Σε αυτά συμπεριλαμβάνονται ο αριθμός των ενεργών αναζητήσεων, το μέγιστο όριο των ταυτόχρονων αναζητήσεων, και τις χρονικές παραμέτρους TTL σύμφωνα με τις οποίες μια αναζήτηση παραμένει ενεργή στη μνήμη του SearchStore πριν διαγραφεί αυτόματα.

Συνεχίζοντας, το microservice Puppeteer διαχειρίζεται τις λειτουργίες web scraping, αυθεντικοποίησης και παρακολούθησης των ενεργών sessions προς τις εξωτερικές πηγές. Εκτελείται στην port 4000 όπως αυτό ορίζεται στο τέλος του αρχείου server.js και τα διαθέσιμα endpoints αποτελούνται από τα εξής :

- /search (GET) - Αποτελεί κεντρικό σημείο της αρχιτεκτονικής και χρησιμοποιείται από το κύριο backend της εφαρμογής. Δέχεται αιτήματα GET που περιλαμβάνουν το query κειμένου προς αναζήτηση στην υπηρεσία Espacenet. Μετά τη δυναμική εκτέλεση της σελίδας και την πλήρη φόρτωση των συμβόλων ταξινόμησης, το API επιστρέφει μια δομημένη απάντηση JSON που περιλαμβάνει τα σύμβολα IPC, τις περιγραφές τους, και τους σχετικούς

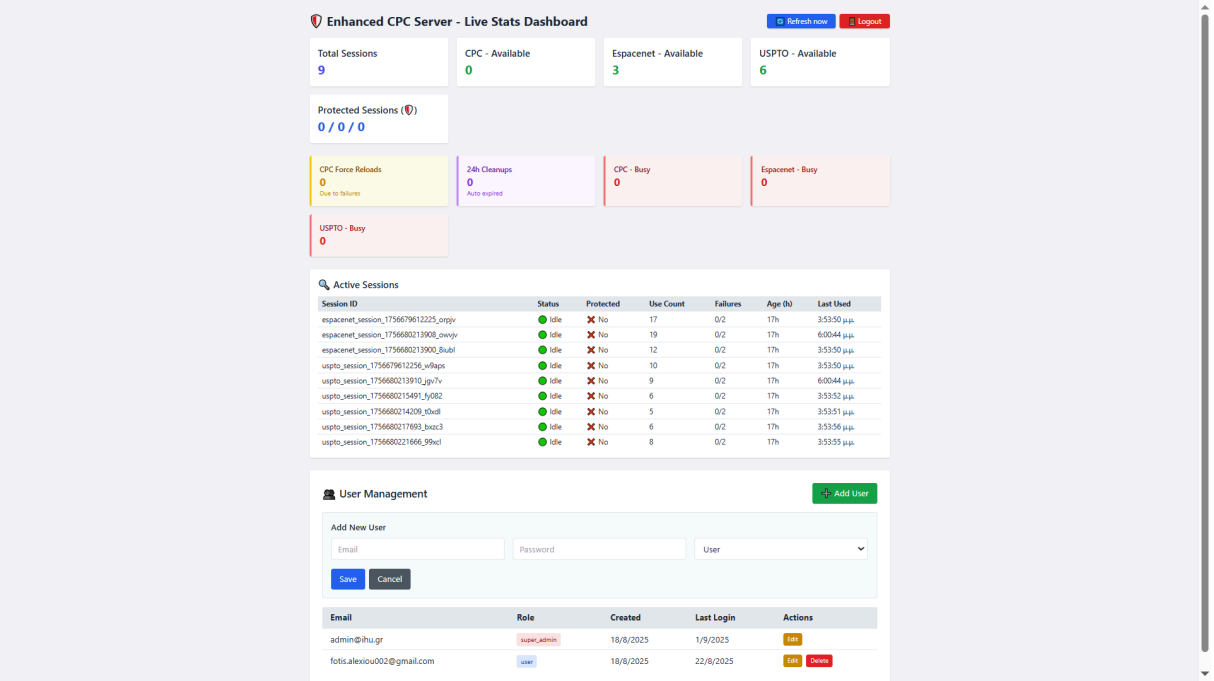
υπερσυνδέσμους. Η πρόσβαση στο endpoint ορίζεται δυναμικά από το Python backend μέσω του αρχείου espacenet_service.py, όπου δηλώνεται μια παραμετροποιήσιμη μεταβλητή περιβάλλοντος PUPPETEER_API_HOST που επιτρέπει την ευέλικτη δρομολόγηση των αιτημάτων, ελέγχοντας αν η εφαρμογή εκτελείται σε απομακρυσμένο host server ή εντός docker localhost.

- /live-stats (GET) - Εκτελεί ασύγχρονη συλλογή δεδομένων της κατάστασης των session Pools (EPO, Espacenet, USPTO) μέσω Promise.all() και παρέχει περιεκτικά στατιστικά σε μορφή JSON μέσω του DualSessionManager.
- /cpc-stats (GET) - Εξειδικευμένα στατιστικά για το CPC Text Categoriser του EPO που καλούνται από τη συνάρτηση get_cpc_queue_status() στο cpc_service.py μέσω HTTP requests.
- /cpc-session-status (GET) - Παρέχει λεπτομερείς πληροφορίες για τη διαθεσιμότητα, την υγεία και την κατάσταση προστασίας του CPC Text Categoriser session.
- /captcha-protection-status (GET) - Χρησιμοποιείται για την παρακολούθηση της κατάστασης των ενεργών sessions pools και παρέχει πληροφορίες σχετικά με τους μηχανισμούς προστασίας από captcha στις αντίστοιχες εξωτερικές πηγές καλώντας τη συνάρτηση get_captcha_protection_status().

Αντίστοιχα για το ολοκληρωμένο σύστημα αυθεντικοποίησης των χρηστών που βασίζεται στη θύρα 4000, έχουν αναπτυχθεί endpoints που εξασφαλίζουν την ασφαλή πρόσβαση στις προηγμένες λειτουργίες της εφαρμογής. Πιο αναλυτικά τα endpoints αυθεντικοποίησης είναι τα εξής :

- /login-page (GET) - Εμφανίζει τη σελίδα σύνδεσης για την αυθεντικοποίηση των χρηστών.
- /login (POST) - Επαληθεύει τα στοιχεία του χρήστη μέσω userDB.authenticateUser(), δημιουργεί session και επιστρέφει JSON success response με user metadata.
- /logout (POST) - Τερματίζει το session σύνδεσης εκτελώντας req.session.destroy() και πραγματοποιεί ασφαλή αποσύνδεση.
- /auth-status (GET) - Επιστρέφει την τρέχουσα κατάσταση αυθεντικοποίησης του χρήστη.
- /logout-page (GET) - Εμφανίζει επιβεβαιωτική μορφοποιημένη σελίδα μετά την επιτυχή αποσύνδεση.
- /api/users (GET) - Εκτελεί τη συνάρτηση userDB.getAllUsers() και επιστρέφει μόνο στον super_admin λίστα όλων των χρηστών χωρίς τους κωδικούς ασφαλείας τους
- /api/users (POST) - Δημιουργεί νέο χρήστη μέσω της συνάρτησης userDB.addUser() με πεδία email, κωδικό πρόσβασης και δικαιώματα ρόλου.
- /api/users/:email (PUT) - Ενημερώνει τα στοιχεία των υπαρχόντων χρηστών για τα επιλεγμένα πεδία κωδικού πρόσβασης και ρόλου μέσω της συνάρτησης userDB.updateUser().
- /api/users/:email (DELETE) - Διαγράφει λογαριασμούς χρηστών μέσω της συνάρτησης userDB.deleteUser() με λογικό έλεγχο προστασίας για τους λογαριασμούς με ρόλο super_admin.

Αξιοποιώντας τα παραπάνω endpoints, για την άμεση παρακολούθηση της κατάστασης των ενεργών sessions και της διαχείρισης των χρηστών έχει υλοποιηθεί στο endpoint /dashboard (GET) μια διαχειριστική διεπαφή για super administrators, η οποία οπτικοποιεί τα αποτελέσματα που επιστρέφουν από τα JSON responses.



Εικόνα 3.4: Πίνακας ελέγχου διαχειριστών στο endpoint /dashboard

3.4 Μηχανισμός φιλτραρίσματος επιπέδων

Όπως αναλύθηκε σε προηγούμενο κεφάλαιο, η ταξινόμηση των ευρεσιτεχνιών σύμφωνα με τα συστήματα CPC και IPC βασίζεται στην ιεραρχική δομή των συμβόλων. Για την ορθολογική χρήση των πληροφοριών που προέρχονται από τις πατέντες, η εφαρμογή δίνει στο χρήστη τη δυνατότητα επιλογής του επιθυμητού βάθους φιλτραρίσματος. Συγκεκριμένα ο χρήστης μπορεί να επιλέξει μεταξύ επιπέδων Section, Class, Subclass και Group με αποτέλεσμα την ενεργοποίηση ενός μηχανισμού, ο οποίος εφαρμόζεται σε όλες τις υπηρεσίες της εφαρμογής, είτε αυτές βασίζονται σε web scraping, είτε σε απευθείας API αναζητήσεις, είτε ακόμα και στις προβλέψεις του νευρωνικού μοντέλου. Ο πυρήνας του μηχανισμού βασίζεται στην έννοια της "αποκοπής" (trimming) του πλήρες συμβόλου, στο επίπεδο επιλογής του χρήστη. Ως προεπιλεγμένο επίπεδο έχει οριστεί το subclass, ενώ για το σκοπό αυτό έχει δημιουργηθεί ένας ενιαίος πίνακας αντιστοίχισης επιπέδου ως προς το μήκος του συμβόλου και ορίζεται ως LEVEL_MAP. Παρακάτω αποτυπώνεται στον κώδικα η εφαρμογή του μηχανισμού μέσω της συνάρτησης trim_symbol_to_level. Η συνάρτηση δέχεται δύο ορίσματα, το symbol που αντιστοιχεί σε έναν πλήρη σύμβολο CPC ή IPC και το length δηλαδή το μήκος χαρακτήρων του συμβόλου που αντιστοιχεί στο επιθυμητό επίπεδο φιλτραρίσματος με βάση την επιλογή του χρήστη. Αξίζει να σημειωθεί ένας επιπλέον σημαντικός έλεγχος που εφαρμόζεται για το επίπεδο Group, στο οποίο πέρα από το μήκος του συμβόλου ελέγχεται και η ύπαρξη του ειδικού χαρακτήρα (/), πριν τελικά η συνάρτηση επιστρέψει τον κατάλληλο “κομμένο” κωδικό, σύμφωνα με τον στόχο του χρήστη. Η συγκεκριμένη προσέγγιση στο μηχανισμό του φιλτραρίσματος προσφέρει ευελιξία καθώς τα επιστρεφόμενα σύμβολα ανεξαρτήτου υπηρεσίας παρουσιάζονται ακριβώς στο ίδιο επίπεδο αλλά και επεκτασιμότητα καθώς το LEVEL_MAP μπορεί να προσαρμοστεί για υποομάδες (subgroup) στο μέλλον.

```python
LEVEL_MAP = {
    "section": 1,
    "class": 3,
    "subclass": 4,
    "group": 5
}

def trim_symbol_to_level(symbol: str, length: int) -> Optional[str]:
    symbol = symbol.strip()
    if length == 1:
        return symbol[0] if len(symbol) >= 1 else None
    elif length == 3:
        return symbol[:3] if len(symbol) >= 3 else None
    elif length == 4:
        return symbol[:4] if len(symbol) >= 4 else None
    elif length == 5:
        return symbol if "/" in symbol else None
    return None
```

Εικόνα 3.5: Mapping αντιστοίχισης επιπέδου

3.5 Batch Mode

3.5.1 Εισαγωγή

Η ανάγκη για την αποστολή πολλών text queries ως είσοδο στην εφαρμογή, με αποδοτικό τρόπο και χωρίς την ανάγκη για χειροκίνητη επαναλαμβανόμενη αναζήτηση οδήγησε στην ενσωμάτωση της λειτουργίας Batch Mode. Η λειτουργία αυτή επιτρέπει στο χρήστη να εισάγει ένα προκαθορισμένο κατάλληλα μορφοποιημένο XML αρχείο, το οποίο περιλαμβάνει ένα σύνολο ερωτημάτων (queries) και να εκτελέσει στην εφαρμογή μαζική πρόβλεψη κωδικών στο ενσωματωμένο νευρωνικό μοντέλο αλλά και στις υπόλοιπες υπηρεσίες μέσω των προαναφερθέντων τεχνικών web scraping και API.

3.5.2 Δομή και Ανάλυση XML

Απαραίτητη προϋπόθεση για την σωστή εκτέλεση της λειτουργίας Batch Mode κρίνεται η ορθή δόμηση του XML με χρήση των προκαθορισμένων tags. Η ρίζα του εγγράφου είναι το στοιχείο <batch>, το οποίο περιλαμβάνει επαναλαμβανόμενα στοιχεία <patent>, καθένα από τα οποία αντιστοιχεί σε μια πατέντα ή τεχνική περιγραφή προς αναζήτηση. Επόμενο εμφωλευμένο στοιχείο εντός του <patent> είναι το <docid> και αποτελεί ένα μοναδικό αναγνωριστικό εγγραφής. Ακολουθεί το tag <query> το οποίο περιέχει το κείμενο προς αναζήτηση, ενώ στην ίδια ιεραρχία η επόμενη ετικέτα είναι το <level> όπου διευκρινίζεται το επίπεδο ταξινόμησης. Για την ορθή αντιστοίχιση των queries με τους classifiers έχει προστεθεί το tag <classification_type> στο οποίο ορίζεται αν πρόκειται για CPC ή IPC αναζήτηση. Ιδιαίτερα σημαντική είναι η ετικέτα <codes> η οποία περιλαμβάνει επαναλαμβανόμενα στοιχεία <code> με τους σωστούς IPC και CPC κωδικούς που αναμένεται να επιστρέψουν από τις προβλέψεις των ταξινομητών.

```
<batch>
  <patent>
    <docid>Q001</docid>
    <query>Batteries designed for use in modern electric vehicles including thermal management</query>
    <level>group</level>
    <classification_type>cpc</classification_type>
    <codes>
      <code>H01M10/0525</code>
      <code>B60L11/00</code>
    </codes>
  </patent>
  <patent>
    <docid>Q002</docid>
    <query>Advanced medical compositions used specifically for targeted cancer treatment therapy</query>
    <level>subclass</level>
    <classification_type>cpc</classification_type>
    <codes>
      <code>A61K</code>
    </codes>
  </patent>
  <patent>
    <docid>Q003</docid>
    <query>Applications of machine learning algorithms for complex visual image recognition tasks</query>
    <level>class</level>
    <classification_type>ipc</classification_type>
    <codes>
      <code>G06</code>
      <code>A01</code>
    </codes>
  </patent>
  <patent>
    <docid>Q004</docid>
    <query>Solar cell technology for renewable energy applications</query>
    <level>group</level>
    <classification_type>ipc</classification_type>
    <codes>
      <code>H01L31/04</code>
      <code>H02N6/00</code>
    </codes>
  </patent>
</batch>
```

Εικόνα 3.6: Κατάλληλα μορφοποιημένο XML για batch mode

Η διαδικασία της εκτέλεσης ξεκινάει με την επιλογή του αρχείου και την αυθεντικοποίηση του χρήστη προκειμένου να καθοριστούν οι διαθέσιμοι ταξινομητές που έχουν οριστεί ανάλογα με τα δικαιώματα του ρόλου. Ακολουθεί η διαδικασία ανάλυσης και επικύρωσης του XML αρχείου που γίνεται μέσω της συνάρτησης parseXMLBatchData(), χρησιμοποιώντας το DOMParser() για τη μετατροπή του XML σε δομή DOM και τον παράλληλο συντακτικό έλεγχο. Εφόσον το αρχείο είναι έγκυρο, εντοπίζονται τα tags <patent> και για κάθε εγγραφή εφαρμόζεται κανονικοποίηση των δεδομένων και εξάγονται όλα τα πεδία όπως παρουσιάζεται στην εικόνα 3.6. Προκειμένου να διασφαλιστεί η σωστή εκτέλεση του αρχείου και η αποφυγή λαθών, η εφαρμογή φιλτράρει τις κενές εγγραφές και στην περίπτωση απουσίας query προσπερνάει το συγκεκριμένο patent block, ενώ για τα υπόλοιπα tags έχουν οριστεί fallback προεπιλεγμένες τιμές.

3.5.3 Batch Mode Backend

Η backend λειτουργία του Batch Mode στην εφαρμογή έχει υλοποιηθεί με τρόπο που υποστηρίζει την παράλληλη επεξεργασία πολλών ερωτημάτων, επιτρέποντας την ταχεία αξιολόγηση αποτελεσμάτων ταξινόμησης έναντι προσδοκώμενων CPC και IPC κωδικών. Το αρχείο batch_upload.js περιέχει όλη την client-side λογική και υλοποιεί τη διαδικασία εκτέλεσης μαζικών queries, ξεκινώντας από την επιλογή του XML αρχείου και καταλήγοντας στην τελική παρουσίαση των αποτελεσμάτων με δυνατότητες αξιολόγησης και εξαγωγής σε διαφορετικούς τύπους αρχείων. Η συνάρτηση

startAsyncItemFromBatch(item) αποτελεί τον κεντρικό μηχανισμό αρχικοποίησης της επεξεργασίας. Εκτελεί HTTP POST αίτημα προς το endpoint /search-async του Flask server ως εξής:

```
const response = await fetch('/search-async', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    credentials: 'include',
    body: JSON.stringify({
        user_input: item.query,
        level: item.level,
        classification_type: item.classification_type,
        query_index: 0,
        user_codes: item.expected_codes || [],
        exclude_classifiers: [],
        auth_confirmed: await checkAuthStatus()
    })
});
```

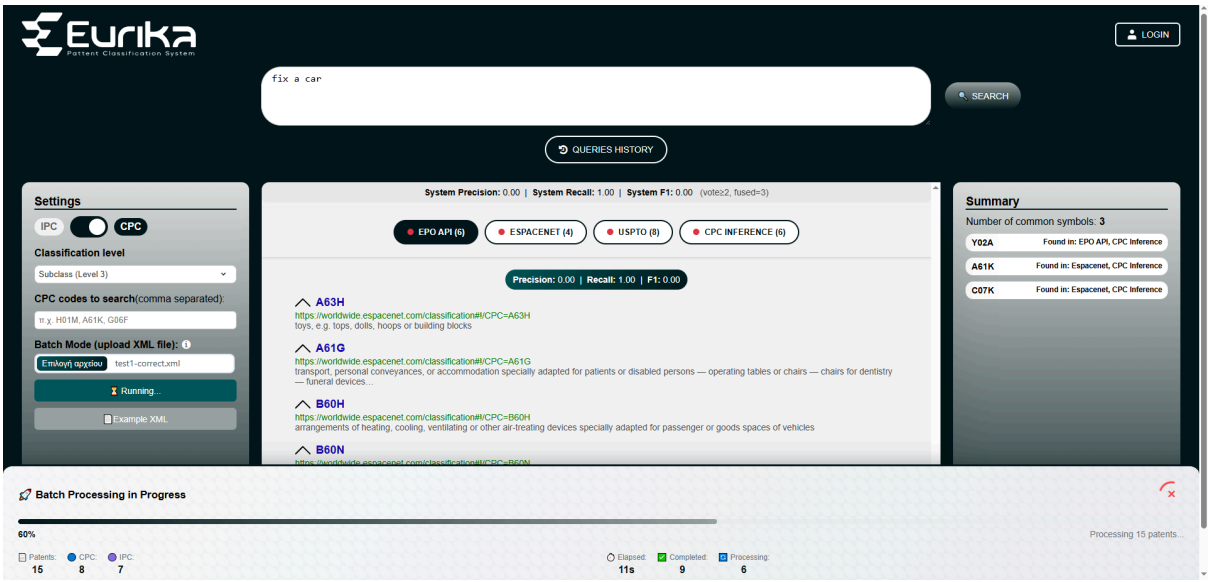
Εικόνα 3.7: Endpoint για την αποστολή αιτημάτων σε batch mode

Κάθε αίτημα λαμβάνει έναν μοναδικό αναγνωριστικό search_id από τον server και είναι απαραίτητο για την παρακολούθηση της εξέλιξης. Με αυτόν τον τρόπο, κάθε query συνδέεται με ξεχωριστή διεργασία στο backend, η οποία εκτελείται παράλληλα με τις υπόλοιπες. Ο μηχανισμός παρακολούθησης υλοποιείται στη pollSearchStatus(searchId, onPartial, itemIndex). Η συνάρτηση αυτή εφαρμόζει χρονικό όριο polling, ανιχνεύει διαδοχικά σφάλματα και τερματίζει όταν όλοι οι ταξινομητές δηλώσουν completed/error/cancelled, είτε όταν παρέλθει το χρονικό όριο MAX_POLLING_TIME για την αποφυγή ατέρμονης εκτέλεσης. Σε κάθε επιτυχή επανάληψη καλεί την onPartial ανανέωση προόδου, ώστε να ενημερώνεται η διεπαφή του χρήστη. Ο μηχανισμός αυτός επαναλαμβάνει την αποστολή αιτημάτων προς το endpoint /search-status/<search_id> μέχρι όλοι οι ταξινομητές να ολοκληρώσουν τις εκτελέσεις. Επιπλέον περιλαμβάνει μια τελική ανάκτηση κατάστασης πριν την έξοδό της, ώστε να μην χαθεί το τελευταίο έγκυρο status σε περίπτωση παροδικών αποτυχιών δικτύου.

Η χρήση ορισμένων ταξινομητών είναι διαθέσιμη μόνο σε authenticated χρήστες συνεπώς για τον έλεγχο αυθεντικοποίησης κατά τη διάρκεια εκτέλεσης του batch mode υλοποιούνται δύο συναρτήσεις, η ασύγχρονη checkAuthStatus() και η checkAuthStatusSync(). Η checkAuthStatus() είναι ασύγχρονος έλεγχος μέσω fetch στον Puppeteer server (/auth-status) με cookies, με έλεγχο του content-type για να διασφαλιστεί ότι επιστρέφεται JSON, και σε περίπτωση σφάλματος ή ασυμβατότητας τερματίζει με την επιστροφή boolean false. Η checkAuthStatusSync() είναι συμπληρωματικός, συγχρονισμένος έλεγχος μέσω DOM, αναζητώντας ενδείξεις αυθεντικοποίησης του χρήστη μέσω elements με συγκεκριμένες class και id.

3.5.4 Batch Mode Frontend

Για την καλύτερη πληροφόρηση του χρήστη με την απεικόνιση της προόδου κατά τη διάρκεια εκτέλεσης του batch mode, έχει υλοποιηθεί η συνάρτηση showInteractiveLoading() που αποδίδει ένα πλήρες διαδραστικό user interface. Η διεπαφή περιλαμβάνει μια γραμμή προόδου σύμφωνα με το ποσοστό ολοκλήρωσης της διαδικασίας, χρονικούς δείκτες και μετρητές με το πλήθος των πατεντών που αναγνώστηκαν διαχωρισμένες σε IPC και CPC. Η λεπτομερής χρονομέτρηση και προβολή του χρόνου εκτελείται από τη startProgressTracking(), η οποία ανά δευτερόλεπτο ενημερώνει το πεδίο “Elapsed” σε μορφή s/m/h, και συμπληρώνεται από την updateLoadingInfo(patentCount, cpcCount, ipcCount) που ανανεώνει τους μετρητές, ανά τύπο ταξινόμησης, με το πλήθος των πατεντών. Για τον πλήρη έλεγχο της διαδικασίας έχει προστεθεί και σχετικό κουμπί ακύρωσης της εκτέλεσης μέσω της συνάρτησης cancelBatchProcessing(). Σε περίπτωση ακύρωσης, η cancelBatchProcessing() αλλάζει την boolean μεταβλητή isBatchCancelled σε true, ενημερώνει άμεσα το UI, στέλνει σε όλα τα τρέχοντα search IDs αίτημα ακύρωσης (/cancel-search-all/<id>) και τελικά αποδίδει οπτικά την κατάσταση του ακυρωμένου batch mode.

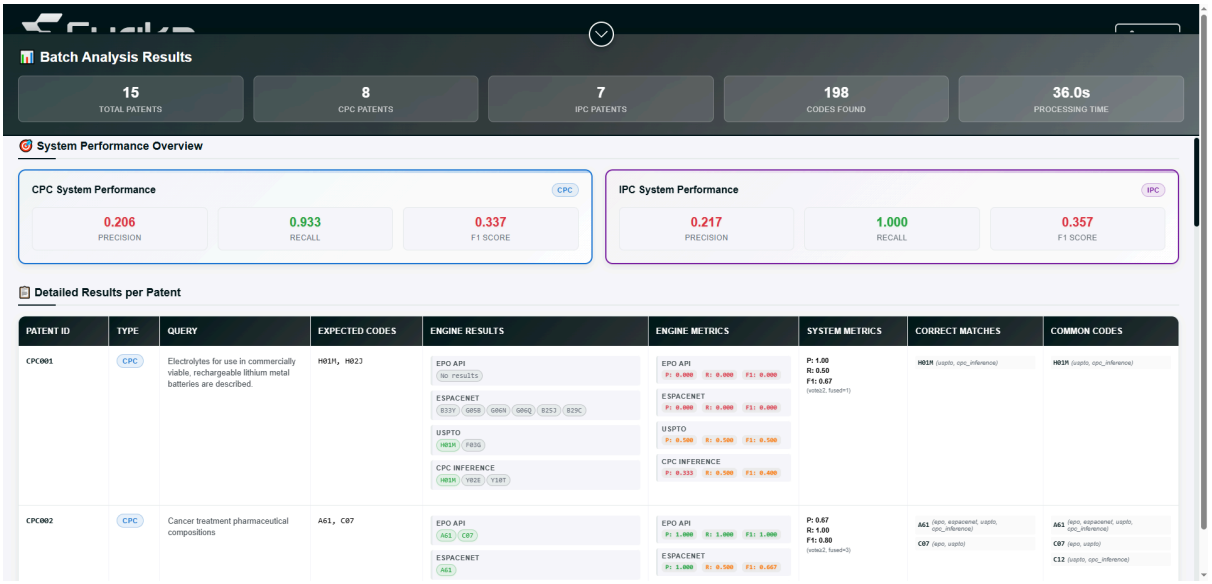


Εικόνα 3.8: Απεικόνιση προόδου κατά την εκτέλεση batch mode

Για την αξιολόγηση της απόδοσης κεντρικό ρόλο έχει η συνάρτηση computePRF1(predictedSymbols, goldSymbols, topK) η οποία υιοθετεί τις μετρικές αξιολόγησης που χρησιμοποιήθηκαν και στις μεμονωμένες αναζητήσεις και αναλύονται εκτενώς στην επόμενη ενότητα. Για την αποφυγή εμφάνισης διπλότυπου συμβόλου σε κάποιον ταξινομητή και την ενίσχυση της συνάφειας, έχει επιλεχθεί η μετατροπή των αποθηκευμένων συμβόλων από arrays σε σύνολα (sets) καθώς έχουν την ιδιότητα να εξασφαλίζουν την εμφάνιση κάθε στοιχείου μόνο μια φορά. Η συνάρτηση calculateEnhancedSystemMetrics() αναλαμβάνει να υπολογίσει τα συνολικά metrics για τα CPC και τα IPC patents σε μια ενιαία μεθοδολογία. Για κάθε πατέντα τύπου CPC, συνδυάζονται τα αποτελέσματα από τους CPC ταξινομητές και αντίστοιχα για κάθε πατέντα τύπου IPC χρησιμοποιούνται οι IPC. Τα αποτελέσματα όλων αυτών συγκεντρώνονται σε δύο σύνολα, το πρώτο fusedFound, περιλαμβάνει όλους τους μοναδικούς κωδικούς που εντοπίστηκαν από τους επιμέρους ταξινομητές και το δεύτερο, fusedMatched, κρατάει μόνο όσους κωδικούς ταιριάζουν με τους

σωστούς (gold), αφού πρώτα υποστούν κανονικοποίηση. Καθώς η διαδικασία επαναλαμβάνεται για κάθε query, ενημερώνονται τρεις βασικοί μετρητές, totalFound (όλοι οι κωδικοί που βρέθηκαν), totalMatched (οι σωστοί κωδικοί) και totalExpected (οι gold κωδικοί). Οι μετρητές αυτοί συσσωρεύονται ξεχωριστά για CPC και IPC, με αποτέλεσμα να προκύπτουν δύο ξεχωριστά σύνολα metrics, τα cpcMetrics και ipcMetrics. Στο τέλος, οι συνολικές τιμές precision, recall και f1-score υπολογίζονται με τους ίδιους μαθηματικούς τύπους χρησιμοποιώντας τα συγκεντρωτικά μεγέθη όλων των πατεντών.

Η συνάρτηση getMetricColorClass(value) χαρακτηρίζει τις τιμές των μετρικών με κλάσεις metric-excellent/metric-good/metric-poor ανάλογα με τις τιμές που υπολογίστηκαν και τις απεικονίζει χρωματικά στο UI. Όταν ολοκληρωθούν όλες οι αναζητήσεις και οι υπολογισμοί, η συνάρτηση renderDetailedResultsSection(results) δημιουργεί τον πίνακα που περιλαμβάνει όλα τα queries, τα σύμβολα που επέστρεψε κάθε classifier, αποτελέσματα των μετρικών και οπτικά badges για πιο ευδιάκριτη πληροφόρηση. Στην εικόνα 3.9 παρουσιάζεται η διεπαφή που εμφανίζεται στο χρήστη μετά την επιτυχή εκτέλεση της μαζικής αναζήτησης.



Εικόνα 3.9: Διεπαφή παρουσίασης των αποτελεσμάτων της εκτέλεσης batch mode

3.5.5 Export Functionalities

Καθώς η εφαρμογή προορίζεται για ερευνητική χρήση τα δεδομένα τα οποία παρουσιάζονται έχουν περισσότερη αξία όταν μπορούν να εξαχθούν για περαιτέρω επεξεργασία. Σε αυτή την κατεύθυνση, υλοποιήθηκε η συνάρτηση renderExportActions(), που προσθέτει στο DOM μέσω Javascript, δύο κουμπιά για την επιλογή τύπου αρχείου εξαγωγής μεταξύ XML και CSV, και ενός τρίτου για άνοιγμα όλων των αποτελεσμάτων σε νέα καρτέλα, με στόχο την ευανάγνωστη παρουσίαση σε μορφή report. Η επιλογή XML καλεί τη συνάρτηση exportToEnhancedXML(), η οποία δημιουργεί ένα δομημένο αρχείο με πλήρη μεταδεδομένα, μετρικές ανά ταξινομητή και συγκεντρωτικά στοιχεία ανά ερώτημα, κατάλληλο για μακροπρόθεσμη επεξεργασία. Αντίστοιχα, η επιλογή CSV καλεί τη exportToEnhancedCSV(), η οποία παράγει έναν πίνακα, εύκολα αναγνώσιμο από υπολογιστικά φύλλα.

Το τρίτο κουμπί ενεργοποιεί τη συνάρτηση openEnhancedReport(), η οποία δημιουργεί ένα ολοκληρωμένο HTML report και το ανοίγει σε νέα καρτέλα του φυλλομετρητή. Το report αυτό παράγεται από τη βοηθητική generateEnhancedHTMLReport(), που συνθέτει το σύνολο των αποτελεσμάτων κατάλληλα μορφοποιημένα, ώστε να δίνει τη δυνατότητα αποθήκευσης και αναπαραγωγής ακόμα και εκτός σύνδεσης. Για την αποφυγή σφαλμάτων κατά την εξαγωγή, οι συναρτήσεις escapeXml() και escapeCsv() εξασφαλίζουν την ορθή κωδικοποίηση ειδικών χαρακτήρων, ενώ η downloadFile() χειρίζεται τη δημιουργία προσωρινών αντικειμένων Blob και την ασφαλή αποθήκευση στον τοπικό δίσκο.


```
▼<batchExport timestamp="2025-09-01T11:20:51.262Z">
  ▼<systemMetrics>
     <overall precision="0.2105" recall="0.9600" f1="0.3453" totalFound="114" totalMatched="24" totalExpected="25"/>
     <cpc precision="0.2059" recall="0.9333" f1="0.3373" totalFound="68" totalMatched="14" totalExpected="15"/>
     <ipc precision="0.2174" recall="1.0000" f1="0.3571" totalFound="46" totalMatched="10" totalExpected="10"/>
   </systemMetrics>
  ▼<classifierMetrics>
     <classifier name="epo" precision="0.4400" recall="0.7333" f1="0.5500" totalFound="25" totalMatched="11" totalExpected="15"/>
     <classifier name="espacenet" precision="0.2143" recall="0.4000" f1="0.2791" totalFound="28" totalMatched="6" totalExpected="15"/>
     <classifier name="uspto" precision="0.2895" recall="0.7333" f1="0.4151" totalFound="38" totalMatched="11" totalExpected="15"/>
     <classifier name="cpc_inference" precision="0.4800" recall="0.8000" f1="0.6000" totalFound="25" totalMatched="12" totalExpected="15"/>
     <classifier name="model" precision="0.4348" recall="1.0000" f1="0.6061" totalFound="23" totalMatched="10" totalExpected="10"/>
     <classifier name="wipo" precision="0.2353" recall="0.8000" f1="0.3636" totalFound="34" totalMatched="8" totalExpected="10"/>
     <classifier name="ipcr_inference" precision="0.4000" recall="1.0000" f1="0.5714" totalFound="25" totalMatched="10" totalExpected="10"/>
   </classifierMetrics>
  ▼<entry docid="CPC002" classification_type="cpc">
     <query>Cancer treatment pharmaceutical compositions</query>
     <level>class</level>
     <expected>A61, C07</expected>
     <systemMetrics precision="0.6667" recall="1.0000" f1="0.8000" voteThreshold="2" fusedCount="3"/>
    ▼<engine name="epo">
       <found>A61, C07</found>
       <matched>A61, C07</matched>
       <precision>1.0000</precision>
       <recall>1.0000</recall>
       <f1>1.0000</f1>
       <truePositives>2</truePositives>
       <falsePositives>0</falsePositives>
       <falseNegatives>0</falseNegatives>
     </engine>
    ▼<engine name="espacenet">
       <found>A61</found>
       <matched>A61</matched>
       <precision>1.0000</precision>
       <recall>0.5000</recall>
       <f1>0.6667</f1>
       <truePositives>1</truePositives>
       <falsePositives>0</falsePositives>
       <falseNegatives>1</falseNegatives>
     </engine>
    ▼<engine name="uspto">
       <found>A61, Y10, C12, G01, C07</found>
       <matched>A61, C07</matched>
       <precision>0.4000</precision>
       <recall>1.0000</recall>
       <f1>0.5714</f1>
       <truePositives>2</truePositives>
       <falsePositives>3</falsePositives>
       <falseNegatives>0</falseNegatives>
     </engine>
    ▼<engine name="cpc_inference">
       <found>A61, C12</found>
       <matched>A61</matched>
       <precision>0.5000</precision>
       <recall>0.5000</recall>
       <f1>0.5000</f1>
       <truePositives>1</truePositives>
       <falsePositives>1</falsePositives>
       <falseNegatives>1</falseNegatives>
     </engine>
     <common_symbols count="3">A61(epo,espacenet,uspto,cpc_inference); C07(epo,uspto); C12(uspto,cpc_inference)</common_symbols>
     <correct_symbols count="2">A61(epo,espacenet,uspto,cpc_inference); C07(epo,uspto)</correct_symbols>
   </entry>
```


Εικόνα 3.10: Δομή του εξαγόμενου xml αρχείου

3.6 Μετρικές αξιολόγησης

Στον τομέα της μηχανικής μάθησης, η μέτρηση και η αξιολόγηση της απόδοσης των αλγορίθμων ταξινόμησης αποτελεί καίρια διαδικασία για τη μέτρηση της αποτελεσματικότητας και της αξιοπιστίας των μοντέλων. Τρεις από τις πιο σημαντικές και ευρέως χρησιμοποιούμενες μετρικές αξιολόγησης

είναι το Precision (Ακρίβεια), το Recall (Ευαισθησία) και το F1-Score (Βαθμολογία F1). Οι μετρικές αυτές προκύπτουν από τον πίνακα σύγχυσης (confusion matrix), ο οποίος καταγράφει τον αριθμό των αληθώς θετικών (True Positives, TP), αληθώς αρνητικών (True Negatives, TN), ψευδώς θετικών (False Positives, FP) και ψευδώς αρνητικών (False Negatives, FN) προβλέψεων. Χρησιμοποιώντας αυτές τις μετρικές μπορούμε να αξιολογήσουμε διαφορετικές καταστάσεις για την ποιότητα των προβλέψεων και να εξασφαλίσουμε την εγκυρότητα των αποτελεσμάτων σε περιπτώσεις δημιουργίας εφαρμογών ταξινομικής πληροφορίας. Το εύρος τιμών που επιστρέφεται ως αποτέλεσμα κυμαίνεται από μηδέν έως ένα.

3.6.1 Precision (Ακρίβεια)

Το Precision εκφράζει το ποσοστό των πραγματικά σωστών θετικών προβλέψεων σε σχέση με όλες τις προβλέψεις που ταξινομήθηκαν ως θετικές και ορίζεται μαθηματικά ως:

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

Το άθροισμα (TP + FP) αντιπροσωπεύει το σύνολο των θετικών προβλέψεων του μοντέλου, ενώ η υψηλή τιμή Precision σημαίνει ότι το μοντέλο έχει χαμηλό ποσοστό ψευδών θετικών αποτελεσμάτων.

3.6.2 Recall (Ευαισθησία)

Το Recall εκφράζει το ποσοστό των πραγματικά σωστών θετικών προβλέψεων σε σχέση με όλα τα πραγματικά αποτελέσματα στο σύνολο των δεδομένων και ορίζεται μαθηματικά ως:

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

Το άθροισμα (TP + FN) αντιπροσωπεύει το σύνολο των πραγματικά θετικών περιπτώσεων στα δεδομένα, ενώ η υψηλή τιμή Recall σημαίνει ότι το μοντέλο έχει χαμηλό ποσοστό ψευδών αρνητικών αποτελεσμάτων.

3.6.3 F1-score

Το F1-Score αποτελεί την αρμονικό μέσο όρο των μετρικών Precision και Recall, θέλοντας να αποτυπώσει μια ισορροπημένη εικόνα της επίδοσης του μοντέλου. Μαθηματικά, το F1-Score ορίζεται ως:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.3}$$

Το F1-Score παρέχει μια ισορροπημένη εκτίμηση της συνολικής απόδοσης του μοντέλου, ιδιαίτερα χρήσιμη όταν υπάρχει ανάγκη για έναν ενιαίο αριθμητικό δείκτη που συνδυάζει τόσο την ακρίβεια όσο και την πληρότητα των προβλέψεων. Είναι ιδιαίτερα αποτελεσματικό σε περιπτώσεις όπου τα δεδομένα είναι ασύμμετρα ή όταν χρειάζεται μια συνολική δίκαιη αξιολόγηση που δεν ευνοεί ούτε το Precision ούτε το Recall.

3.6.4 Αξιοποίηση των μετρικών στην εφαρμογή

Για τον έλεγχο της αξιοπιστίας της διαδικτυακής εφαρμογής αναζήτησης ταξινομικής πληροφορίας αλλά και για την αξιολόγηση της απόδοσης των νευρωνικών μοντέλων υιοθετήθηκαν οι παραπάνω μετρικές και στη λειτουργία της εφαρμογής. Συγκεκριμένα, αναπτύχθηκε η συνάρτηση compute_prf1, η οποία παίρνει ως είσοδο τρεις παραμέτρους, το predicted_symbols που αποτελεί ένα σύνολο προβλέψεων από τους ταξινομητές, τα gold_symbols που είναι τα σωστά σύμβολα IPC ή CPC τα οποία δόθηκαν από το χρήστη και η παράμετρος top_k=10 που λειτουργεί ως όριο για τη χρήση των

πρώτων δέκα επιστρεφόμενων συμβόλων από κάθε ταξινομητή. Προκειμένου η σύγκριση των δεδομένων να είναι σωστή, η συνάρτηση εφαρμόζει κανονικοποίηση των δεδομένων αφαιρώντας τυχόν κενά διαστήματα και μετατρέποντας όλους τους χαρακτήρες σε κεφαλαίους. Εσωτερικά στη συνάρτηση ο αλγόριθμος υπολογίζει τη διασταύρωση των αποτελεσμάτων πρόβλεψης και αυτών που δόθηκαν (TP), τα στοιχεία που προβλέφθηκαν λανθασμένα (FP) και τα στοιχεία που δεν προβλέφθηκαν αλλά υπάρχουν στα σωστά αποτελέσματα (FN). Η υλοποίηση περιλαμβάνει ειδική μεταχείριση για το recall όταν δεν υπάρχουν gold labels, θέτοντάς το σε 1.0. Αυτή η προσέγγιση βασίζεται στη λογική ότι αν δεν υπάρχει τίποτα για να αναζητηθεί, τότε δεν υπάρχει και αστοχία στην ανάκτηση του. Το F1-score υπολογίζεται ως ο αρμονικός μέσος όρος των precision και recall, παρέχοντας μια ενιαία ισορροπημένη μετρική απόδοσης. Για την καλύτερη εμπειρία χρήστη με βάση το ποσοστό του f1 score υπάρχει και μια κουκίδα με χρωματική απόδοση πριν από κάθε ταξινομητή. Ως κακή απόδοση θεωρείται μια τιμή μικρότερη του 0.40 που της αποδίδεται το χρώμα κόκκινο, για τιμές μεταξύ 0.40 και μικρότερες από 0.75 δηλαδή σε περιπτώσεις μέτριας απόδοσης το χρώμα γίνεται πορτοκαλί, ενώ για f1-score μεγαλύτερο από 0.75 θεωρείται μια πολύ υψηλή απόδοση και απεικονίζεται με χρώμα πράσινο.

```python
def compute_prf1(predicted_symbols, gold_symbols, top_k=10):
    """Compute precision/recall/f1 over sets of symbols with strict normalization."""
    norm = lambda s: s.strip().upper().replace(" ", "")
    pred = [norm(s) for s in predicted_symbols if s]
    gold = [norm(s) for s in gold_symbols if s]

    # top-k cutoff
    pred = pred[:top_k]

    pred_set = set(pred)
    gold_set = set(gold)

    tp = len(pred_set & gold_set)
    fp = len(pred_set - gold_set)
    fn = len(gold_set - pred_set)

    precision = tp / (tp + fp) if (tp + fp) > 0 else 0.0

    # If there are no gold labels, treat recall as 1.0 (nothing to find, nothing missed)
    if (tp + fn) > 0:
        recall = tp / (tp + fn)
    else:
        recall = 1.0

    f1_score = (2 * precision * recall / (precision + recall)) if (precision + recall) > 0 else 0.0

    return {
        "true_positives": tp,
        "false_positives": fp,
        "false_negatives": fn,
        "precision": precision,
        "recall": recall,
        "f1_score": f1_score
    }
```

Κώδικας 3.11: Συνάρτηση αξιοποίησης των μετρικών αξιολόγησης

3.7 Πειραματική αξιολόγηση και ανάλυση επιδόσεων

Η αξιολόγηση μιας ευφυούς διαδικτυακής εφαρμογής αναζήτησης ταξινομικής πληροφορίας που αφορά ευρεσιτεχνίες, απαιτεί μια ολοκληρωμένη προσέγγιση που υπερβαίνει την απλή τεχνική υλοποίηση. Στην πράξη, η αποτελεσματικότητα μιας τέτοιας εφαρμογής κρίνεται τόσο από την ακρίβεια των αποτελεσμάτων όσο και από την ικανότητά της να ανταποκρίνεται στις πραγματικές ανάγκες των χρηστών σε λογικούς χρόνους. Για το λόγο αυτό, καθίσταται αναγκαία η διεξαγωγή συστηματικών πειραμάτων που θα αξιολογήσουν τη συμπεριφορά του συστήματος υπό διαφορετικές συνθήκες χρήσης. Η παρούσα ενότητα παρουσιάζει μια λεπτομερή ανάλυση των επιδόσεων της εφαρμογής, εστιάζοντας στη σύγκριση των διαφορετικών υπηρεσιών ταξινόμησης, στη μέτρηση των χρόνων απόκρισης, και στην αξιολόγηση της ακρίβειας των αποτελεσμάτων που επιστρέφονται. Μέσω της χρήσης αντιπροσωπευτικών σεναρίων χρήσης και τυποποιημένων μετρικών αξιολόγησης, επιδιώκεται η εξαγωγή αξιόπιστων συμπερασμάτων σχετικά με την πρακτική αποτελεσματικότητα του συστήματος.

3.7.1 Πειραματική Διαδικασία Μέτρησης Χροχικής Εκτέλεσης

Η αξιολόγηση της απόδοσης του συστήματος πραγματοποιήθηκε μέσω μιας συστηματικής σειράς πειραμάτων που στόχευαν στη μέτρηση τόσο των χρονικών επιδόσεων όσο και της ποιότητας των αποτελεσμάτων. Η πειραματική μεθοδολογία σχεδιάστηκε για να προσομοιώσει ρεαλιστικά σενάρια χρήσης, καλύπτοντας ένα ευρύ φάσμα τεχνολογικών περιοχών και ερωτημάτων διαφορετικών επιπέδων πολυπλοκότητας. Οι δοκιμές διεξήχθησαν χρησιμοποιώντας τόσο μεμονωμένα ερωτήματα όσο και batch επεξεργασία, προκειμένου να αξιολογηθεί η συμπεριφορά του συστήματος υπό διαφορετικές συνθήκες χρήσης και να εκτιμηθεί η επίδραση του παράλληλου μηχανισμού εκτέλεσης στη συνολική απόδοση. Για τη διασφάλιση της εγκυρότητας, κάθε μέτρηση επαναλήφθηκε πολλαπλές φορές και υπολογίστηκαν οι μέσοι όροι των αποτελεσμάτων. Ο παρακάτω πίνακας 3.1 αποτελείται από έξι στήλες, με την πρώτη να περιλαμβάνει την αρίθμηση των σεναρίων, τη δεύτερη τον τύπο ταξινόμησης IPC/CPC, την τρίτη το πλήθος των όρων του ερωτήματος, την τέταρτη το επίπεδο ταξινόμησης, η πέμπτη στήλη περιλαμβάνει το συνολικό αριθμό επαναλήψεων που εκτελέστηκαν σε διαφορετικές χρονικές περιόδους και υπό διαφορετικές συνθήκες και η τελευταία στήλη το μέσο όρο εκτέλεσης όλων των ταξινομητών που καταγράφηκε από την έναρξη έως την ολοκλήρωση των αναζητήσεων. Κάθε σενάριο εκτελέστηκε δέκα φορές για τη διασφάλιση της αξιοπιστίας των μετρήσεων και με διαφορετικό πλήθος διαθέσιμων session pools.

Πίνακας 3.1: Αριθμητικά δεδομένα χρόνων εκτέλεσης

Α/Α	Τύπος ταξινόμησης	Ερώτημα	Επίπεδο	Επαναλήψεις	Χρόνος
S1	CPC	3 λέξεις	class	10	1.15s
S2	IPC	3 λέξεις	class	10	1.14s
S3	CPC	7 λέξεις	subclass	10	1.10s
S4	IPC	7 λέξεις	subclass	10	1.23s
S5	CPC	10 λέξεις	group	10	1.15s
S6	IPC	10 λέξεις	group	10	1.30s
S7	CPC/IPC	4 πατέντες	class-group	10	5.0s
S8	CPC/IPC	15 πατέντες	section-group	10	22.3s

Τα ερωτήματα που εκτελέστηκαν στον πίνακα 3.1 παρουσιάζονται ταξινομημένα κατά τον αύξοντα αριθμό παρακάτω:

- S1-S2: Solar Energy System
- S3-S4: Solar cell technology for renewable energy applications
- S5-S6: Batteries designed for use in electric vehicles including thermal management
- S7: Χρησιμοποιήθηκε το διαθέσιμο example.xml που υπάρχει δημοσιευμένο στην εφαρμογή
- S8: Χρησιμοποιήθηκε διαμορφωμένο xml αρχείο με 15 πολυεπίπεδες αναζητήσεις

Το συμπέρασμα που προκύπτει από τις παραπάνω μετρήσεις είναι πως λόγω της σωστής κατανομής των διαθέσιμων υπολογιστικών πόρων και μέσω της διάσπασης των υπηρεσιών σε ProcessPoolExecutor και ThreadPoolExecutor, που αναλύονται εκτενώς στο κεφάλαιο 4.3.1, ο χρόνος εκτέλεσης παραμένει σχεδόν σταθερός ανεξάρτητα του επιπέδου ή του πλήθους όρων προς αναζήτησης και ο μόνος παράγοντας που αυξάνει την ολοκλήρωση της διαδικασίας είναι το πλήθος των πατεντών που δίνονται προς αναζήτηση στο Batch Mode. Η μεταβλητή του πλήθους θα μπορούσε σχεδόν να εξαλειφθεί σαν παράγοντας που επηρεάζει τον χρόνο αυξάνοντας τους διαθέσιμους πόρους και ακολούθως το πλήθος των workers που μπορούν να εκτελούνται παράλληλα σε κάθε αναζήτηση.

3.7.2 Πειραματική Διαδικασία Αξιολόγησης Μετρικών

Η επικύρωση της ορθότητας των μετρικών αξιολόγησης αποτελεί κρίσιμο στοιχείο για την εξασφάλιση της αξιοπιστίας του συστήματος αναζήτησης. Στο πλαίσιο αυτής της ανάλυσης, εξετάζεται η ακρίβεια των υπολογισμών των μετρικών precision, recall και F1-score που εφαρμόζονται στα αποτελέσματα δύο ταξινομητών της εφαρμογής. Η μεθοδολογία βασίζεται στη χρήση ενός ερωτήματος με προκαθορισμένα αναμενόμενα αποτελέσματα, επιτρέποντας τον χειροκίνητο υπολογισμό και την επαλήθευση των μετρικών. Οι δύο ταξινομητές που επιλέχθηκαν για την ανάλυση είναι τα δύο εκπαιδευμένα νευρωνικά μοντέλα CPC Inference και IPC Inference, παρέχοντας μια ολοκληρωμένη εικόνα της συμπεριφοράς του συστήματος.

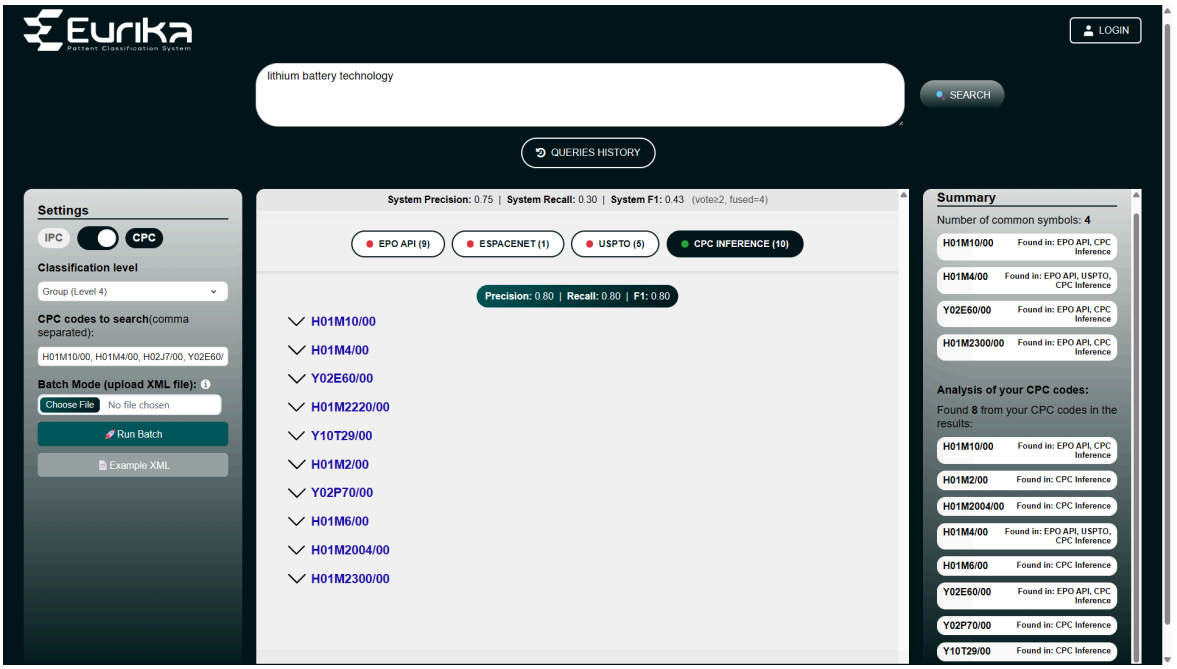
Πίνακας 3.2: Αριθμητικά δεδομένα μετρικών αξιολόγησης CPC ταξινομητή

Classifier	Query	Expected	Predicted	TP	FP	FN	P	R	F1
cpc inference	lithium battery technology	H01M10/00, H01M4/00, H02J7/00, Y02E60/00, H01M2/00, Y02P70/00, H01M6/00, H01M12/00, Y10T29/00, H01M2004/00	H01M10/00, H01M4/00, Y02E60/00, H01M2220/00, Y10T29/00, H01M2/00, Y02P70/00, H01M6/00, H01M2004/00, H01M2300/00	8	2	2	0.80	0.80	0.80

Παρακάτω αναλύονται οι υπολογισμοί των μετρικών του πίνακα 3.2 με βάση τα αποτελέσματα του ταξινομητή:

- True Positives (TP) = Expected ∩ Predicted = 8
- False Positives (FP) = Predicted - Expected = 2
- False Negatives (FN) = Expected - Predicted = 2
- Precision = TP / (TP + FP) = 8 / (8 + 2) = 8/10 = 0.80
- Recall = TP / (TP + FN) = 8 / (8 + 2) = 8/10 = 0.80
- F1-Score = 2 × (Precision × Recall) / (Precision + Recall) = 0.80

Στην εικόνα 3.12 παρουσιάζονται στην εφαρμογή Eurika στο κεντρικό πλαίσιο αποτελεσμάτων, υπολογισμένες ακριβώς οι ίδιες μετρικές αξιολόγησης για τον ταξινομητή CPC INFERENCE. Επιπλέον, του έχει αποδοθεί η πράσινη κουκίδα καθώς το f1 score ήταν μεγαλύτερο του 0.75 δηλώνοντας την αξιοπιστία του στη διαδικασία ταξινόμησης.



Εικόνα 3.12: Αποτελέσματα μετρικών για τον ταξινομητή CPC INFERENCE

Αντίστοιχη διαδικασία πειράματος ακολουθήθηκε και για τον ταξινομητή IPC INFERENCE για να αποτυπωθεί η αξιοπιστία. Τα δεδομένα του πειράματος παρουσιάζονται στον παρακάτω πίνακα:

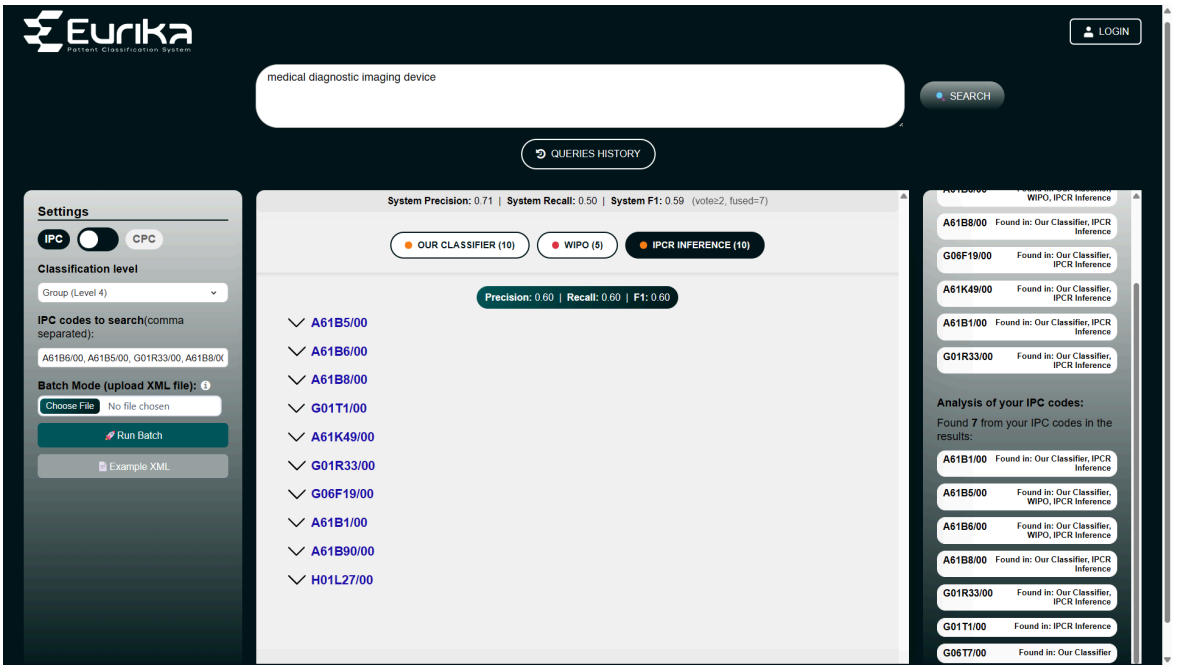
Πίνακας 3.3: Αριθμητικά δεδομένα μετρικών αξιολόγησης IPC ταξινομητή

Classifier	Query	Expected	Predicted	TP	FP	FN	P	R	F1
ipc inference	medical diagnostic imaging device	A61B6/00, A61B5/00, G01R33/00, A61B8/00, H04N5/00, G06T7/00, A61B1/00, G01T1/00, A61B17/00, A61N5/00	A61B5/00, A61B6/00, A61B8/00, G01T1/00, A61K49/00, G01R33/00, G06F19/00, A61B1/00, A61B90/00, H01L27/00	6	4	4	0.60	0.60	0.60

Παρακάτω αναλύονται οι υπολογισμοί των μετρικών του πίνακα 3.3 με βάση τα αποτελέσματα του ταξινομητή:

- True Positives (TP) = Expected ∩ Predicted = 6
- False Positives (FP) = Predicted - Expected = 4
- False Negatives (FN) = Expected - Predicted = 4
- Precision = TP / (TP + FP) = 6 / (6 + 4) = 6/10 = 0.60
- Recall = TP / (TP + FN) = 6 / (6 + 4) = 6/10 = 0.60
- F1-Score = 2 × (Precision × Recall) / (Precision + Recall) = 0.60

Στην εικόνα 3.13 παρουσιάζονται στην εφαρμογή Eurika στο κεντρικό πλαίσιο αποτελεσμάτων, υπολογισμένες ακριβώς οι ίδιες μετρικές αξιολόγησης για τον ταξινομητή IPC INFERENCE. Στο συγκεκριμένο πείραμα, του έχει αποδοθεί η πορτοκαλί κουκίδα καθώς το f1 score ήταν μεγαλύτερο του 0.40 και μικρότερο του 0.75 δηλώνοντας την αξιοπιστία του στη διαδικασία ταξινόμησης.



Εικόνα 3.13: Αποτελέσματα μετρικών για τον ταξινομητή IPC INFERENCE

Πέρα από τους επιμέρους υπολογισμούς των μετρικών precision, recall και f1 score για κάθε ταξινομητή, υπάρχει και σε μεγαλύτερη κλίμακα η συνολική αξιολόγηση της εφαρμογής. Σε κάθε νέα αναζήτηση η εφαρμογή δημιουργεί ένα συνολικό predicted set που περιλαμβάνει όλα τα σύμβολα από όλους τους ενεργούς ταξινομητές. Με βάση αυτό το fused set γίνονται οι ίδιοι υπολογισμοί TP, FP και FN σε σχέση με το gold set. Το αποτέλεσμα αυτής της διαδικασίας είναι οι τιμές System Precision, System Recall και System F1, οι οποίες εκφράζουν την απόδοση του συστήματος συνολικά, δηλαδή ως συνδυασμό όλων των διαθέσιμων ταξινομητών.

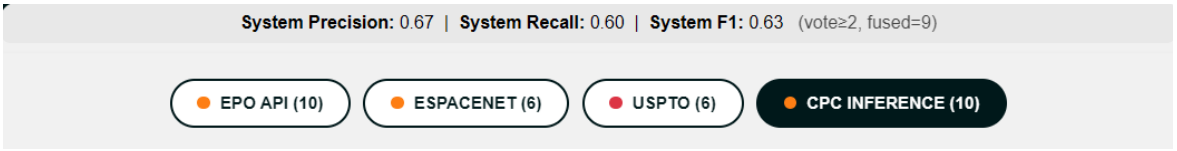
Πίνακας 3.4: Αριθμητικά δεδομένα μετρικών αξιολόγησης συστήματος

Classifier	Query	Expected	Predicted	TP	FP	FN	P	R	F1
EPO API	natural language processing	G06F40/00, G06F17/00, G10L15/00, G06N3/00, G06N20/00, G06F16/00, G06N5/00, C10L1/00, C13K1/00, G06F19/00	G06F40/00, G06F16/00, G06N3/00, G06N20/00, G10L15/00, G06N5/00, G06Q10/00, G06F3/00, G06Q30/00, G06F21/00	6	4	4	0.60	0.60	0.60

ESPACE NET	natural language processing	G06F40/00, G06F17/00, G10L15/00, G06N3/00, G06N20/00, G06F16/00, G06N5/00, C10L1/00, C13K1/00, G06F19/00	G06F40/00, G06N20/00, Y02D10/00, G06V10/00, G06N3/00, G06F16/00	4	2	6	0.67	0.40	0.50
USPTO	natural language processing	G06F40/00, G06F17/00, G10L15/00, G06N3/00, G06N20/00, G06F16/00, G06N5/00, C10L1/00, C13K1/00, G06F19/00	G06F1/00, G10L13/00, G06T1/00, C10L1/00, C13K1/00, G03H1/00	2	4	8	0.33	0.20	0.25
CPC INFERENCE	natural language processing	G06F40/00, G06F17/00, G10L15/00, G06N3/00, G06N20/00, G06F16/00, G06N5/00, C10L1/00, C13K1/00, G06F19/00	G06F17/00, G06F3/00, G10L15/00, G06F19/00, G06N5/00, G06Q30/00, G06K9/00, G06F21/00, G06Q50/00, G06F11/00	4	6	6	0.40	0.40	0.40

Τα System Metrics δεν αντιπροσωπεύουν το άθροισμα των μετρικών από όλους τους ταξινομητές, αλλά υπολογίζονται με βάση έναν μηχανισμό voting και fusion. Συγκεκριμένα, το σύστημα συλλέγει όλα τα expected σύμβολα από όλους τους ταξινομητές και εφαρμόζει ένα voting threshold (≥2 ψήφοι). Μόνο τα σύμβολα που επιστρέφονται από τουλάχιστον δύο διαφορετικούς ταξινομητές διατηρούνται στο τελικό fused αποτέλεσμα. Στη συνέχεια, οι μετρικές precision, recall και F1-score υπολογίζονται συγκρίνοντας αυτά τα fused σύμβολα με τα expected. Αυτή η προσέγγιση στοχεύει στη βελτίωση της ακρίβειας των αποτελεσμάτων μέσω της αντιστοιχίας ίδιων συμβόλων σε τουλάχιστον δύο υπηρεσίες, αποκλείοντας σύμβολα που επιστρέφει μόνο μία υπηρεσία ως πιθανώς λιγότερο αξιόπιστα :

- Fused symbols (≥2 votes): G06F40/00, G06F16/00, G06N3/00, G06N20/00, G10L15/00, G06N5/00, G06F3/00, G06Q30/00, G06F21/00
- True Positives (TP) = Expected ∩ Fused = 6
- False Positives (FP) = Fused - Expected = 3
- False Negatives (FN) = Expected - Fused = 4
- System Precision = TP / (TP + FP) = 6 / (6 + 3) = 6/9 = 0.67
- System Recall = TP / (TP + FN) = 6 / (6 + 4) = 6/10 = 0.60
- System F1-Score = 2 × (Precision × Recall) / (Precision + Recall) = 0.63



Εικόνα 3.14: Αποτελέσματα μετρικών του συστήματος

Η επαλήθευση των System Metrics επιβεβαιώνει την ορθότητα των υπολογισμών του συστήματος μέσω της εικόνας 3.14. Η ταύτιση μεταξύ των θεωρητικών υπολογισμών (Precision: 0.67, Recall: 0.60, F1: 0.63) και των αποτελεσμάτων που επιστρέφει το σύστημα αποδεικνύει την ακρίβεια της υλοποίησης και την αξιοπιστία των μετρικών αξιολόγησης.

Κεφάλαιο 4ο: Αρχιτεκτονική του Συστήματος

4.1 Περιγραφή της υποδομής

Η υποδομή του συστήματος αναπτύχθηκε σε ένα Virtual Private Server (VPS) με λειτουργικό σύστημα Ubuntu 22.04 LTS. Η επιλογή αυτή βασίστηκε κυρίως στη σταθερότητα που προσφέρει η LTS έκδοση του Ubuntu, καθώς και στην ενεργή υποστήριξη με αναβαθμίσεις ασφαλείας έως το 2027. Ο διακομιστής διαθέτει τετραπύρηνο εικονικό επεξεργαστή (vCPU), 16 GB μνήμης RAM και συνολικά 200 GB SSD αποθηκευτικού χώρου. Επιπλέον, στο διακομιστή διαμορφώθηκε swap memory 10GB, ώστε να διασφαλίζεται η ομαλή λειτουργία σε στιγμές αυξημένης ζήτησης και κατανάλωσης πόρων της φυσικής μνήμης RAM, αποτρέποντας έτσι αποτελεσματικά την πιθανότητα αστάθειας και προσωρινής διακοπής λειτουργίας. Οι συγκεκριμένες προδιαγραφές κρίνονται απαραίτητες για την αποτελεσματική λειτουργία της εφαρμογής, την γρήγορη ταυτόχρονη εκτέλεση πολλαπλών υπηρεσιών που συνδυάζουν νευρωνικά μοντέλα μηχανικής μάθησης, web scraping και μαζική επεξεργασία δεδομένων σε πραγματικό χρόνο από πολλούς και διαφορετικούς χρήστες.

4.2 Χρήση Docker και Τεχνολογιών Containerization

Ως καίρια χαρακτηριστικά οποιασδήποτε εφαρμογής από το στάδιο του σχεδιασμού μέχρι την τελική παραγωγική έκδοση ορίζονται η φορητότητα και η επεκτασιμότητα. Έχοντας αυτές τις δύο έννοιες από το αρχικό στάδιο ανάπτυξης της εφαρμογής, η δομή και η λειτουργία βασίστηκε στην προσέγγιση των microservices, δηλαδή την απομόνωση των υπηρεσιών και τη χρήση τους ως ανεξάρτητες οντότητες. Για την επίτευξη αυτού του στόχου επιλέχθηκε η τεχνολογία Docker και πιο συγκεκριμένα η μέθοδος containerization. Αρχικά μέσω των αρχείων Dockerfile ορίζεται το περιβάλλον εκτέλεσης της εφαρμογής, καθορίζοντας τις βιβλιοθήκες, τις εξαρτήσεις και τις παραμέτρους του συστήματος. Στη συνέχεια για τις ανάγκες της δικής μας εφαρμογής επιλέξαμε τη χρήση δύο κύριων containers, το **Patent-app** στο οποίο φιλοξενείται η κύρια εφαρμογή Flask βασισμένη σε Python 3.10.12, ενώ το δεύτερο container **Puppeteer-api** λειτουργεί ανεξάρτητα σε Node.js 20 περιβάλλον και στο οποίο εκτελούνται οι πιο απαιτητικές τεχνικές Web Scraping. Αυτός ο διαχωρισμός επιτρέπει την ανεξάρτητη κλιμάκωση, τον αυτόνομο ορισμό διαθέσιμων πόρων και την καλύτερη συντήρηση κάθε υπηρεσίας.

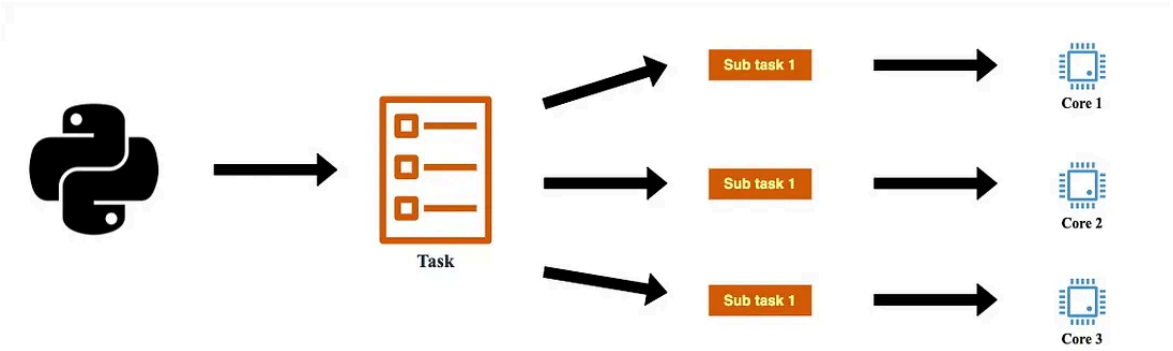
4.3 Τεχνική περιγραφή Backend

Η backend αρχιτεκτονική του συστήματος υλοποιείται κυρίως σε Python με τη χρήση του framework Flask να αποτελεί τον κεντρικό πυρήνα που συντονίζει όλα τα αιτήματα, ενώ για τις πιο εξειδικευμένες διεργασίες που απαιτούν διαδικτυακές αυτοματοποιήσεις αξιοποιείται το Node.js περιβάλλον. Ο συνδυασμός των δύο τεχνολογιών επιτρέπει την αξιοποίηση των πλεονεκτημάτων καθεμιάς ξεχωριστά. Η Python ενδείκνυται για τη γρήγορη επεξεργασία δεδομένων, την ενσωμάτωση βιβλιοθηκών μηχανικής μάθησης και για την ικανότητα διαχείρισης πολύπλοκων ασύγχρονων εργασιών. Από την άλλη πλευρά η επιλογή της χρήσης Node.js και πιο συγκεκριμένα της βιβλιοθήκης Puppeteer έγινε έπειτα από πλήθος πειραμάτων και αποτυχημένων δοκιμών προσπέρασης δικλείδων ασφαλείας από άλλες μεθόδους web scraping, εξασφαλίζοντας υψηλής απόδοσης δυνατότητες headless browser. Η επικοινωνία μεταξύ των δύο υπηρεσιών επιτυγχάνεται μέσω HTTP endpoints.

4.3.1 Flask Framework

Το Flask αποτελεί ένα ελαφρύ framework σε Python, σχεδιασμένο για την ταχεία ανάπτυξη διαδικτυακών εφαρμογών και υπηρεσιών. Το αρχείο app.py αποτελεί το βασικό σημείο εισόδου της Flask εφαρμογής. Στον κώδικα ορίζονται οι απαραίτητες διαδρομές (routes), μέσω των οποίων οι χρήστες και οι εξωτερικές υπηρεσίες αλληλεπιδρούν με το σύστημα. Το Flask εκτελείται στο port 5000 και λειτουργεί ως REST API backend που δέχεται αιτήματα από το frontend και τα δρομολογεί στους κατάλληλους handlers. Η βασική δομή της εφαρμογής ξεκινάει με την τυπική αρχικοποίηση app = Flask(__name__) και εξελίσσεται σε ένα πολυεπίπεδο σύστημα που διαχειρίζεται παράλληλες εργασίες μέσω των ProcessPoolExecutor και ThreadPoolExecutor της βιβλιοθήκης concurrent.futures. Αυτή η αρχιτεκτονική επιλογή μετατρέπει τη συγχρονισμένη εκτέλεση του Flask σε ασύγχρονη επεξεργασία αντιμετωπίζοντας παράλληλα προβλήματα μειωμένης απόδοσης καθώς έχουν διαχωριστεί πλήρως τα νευρωνικά μοντέλα σε εκτέλεση μέσω Process Pool και τα Http αιτήματα εκτελούνται αυτόνομα μέσω Thread Pool. Ως Executor ορίζεται μία αφηρημένη κλάση (abstract base class) του module concurrent.futures, που παρέχει μια κοινή διεπαφή για την ασύγχρονη εκτέλεση των συναρτήσεων. Μέσω των βασικών μεθόδων submit(), map() και shutdown(). Ο Executor διαχειρίζεται την εκχώρηση εργασιών σε υποκείμενα “workers” και επιστρέφει αντικείμενα Future που αντιπροσωπεύουν την κατάσταση και το αποτέλεσμα των εργασιών [12].

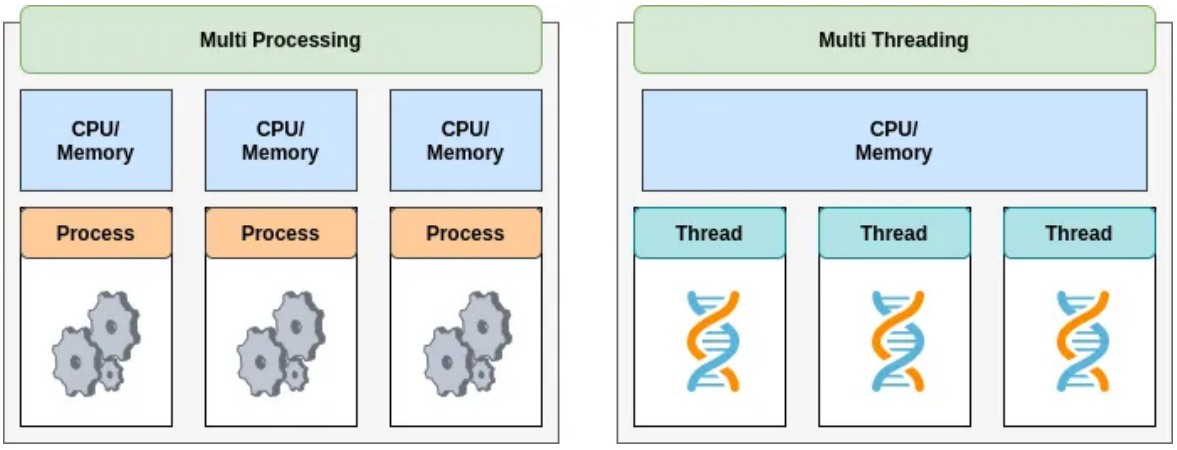
Ο ProcessPoolExecutor είναι υποκλάση του Executor που χρησιμοποιεί ένα pool από διεργασίες (processes) για την ασύγχρονη εκτέλεση εργασιών. Εκμεταλλεύοντας το υποσύστημα multiprocessing, κάθε διεργασία εκτελείται ανεξάρτητα και έχει τον δικό της χώρο μνήμης, ο οποίος επιτρέπει την πραγματική παράλληλη εκτέλεση σε πολλαπλούς πυρήνες CPU. Σημαντικό πλεονέκτημα αποτελεί το γεγονός ότι οι διεργασίες δεν επηρεάζονται από τους περιορισμούς του Global Interpreter Lock (GIL). Για την εκτέλεση απαιτείται υψηλότερο υπολογιστικό κόστος, όλα τα δεδομένα που μεταφέρονται μεταξύ της κύριας διεργασίας και των worker processes πρέπει να υποστηρίζουν σειριοποίηση (serialization) μέσω του pickle module της Python και το κύριο module (__main__) να είναι προσβάσιμο από τους worker subprocesses [12].



Εικόνα 4.1: Multiprocessing μέσω του ProcessPoolExecutor. Πηγή: [13]

Ο ThreadPoolExecutor είναι και αυτό μια υποκλάση του Executor που χρησιμοποιεί έναν pool από threads για να εκτελεί κλήσεις ασύγχρονα μέσα στην ίδια διεργασία. Σε αντίθεση με την υποκλάση ProcessPoolExecutor, τα threads περιορίζονται από το Global Interpreter Lock (GIL), το οποίο εμποδίζει πολλά threads να εκτελούν bytecode Python ταυτόχρονα. Αυτό καθιστά τα threads λιγότερο αποτελεσματικά για εργασίες που συνδέονται με την CPU. Ωστόσο, είναι κατάλληλος για εργασίες που είναι I/O-bound, όπως HTTP αιτήματα ή εγγραφές αρχείων, και επιτρέπει την αποδοτική

διαχείριση πολλαπλών συνδέσεων με χαμηλό υπολογιστικό κόστος, καθώς τα threads μοιράζονται την ίδια μνήμη [12].



Εικόνα 4.2: Σύγκριση μεταξύ Multiprocessing και Multithreading. Πηγή: [14]

Ο μηχανισμός της κλάση Pool βασίζεται στη δημιουργία μιας δεξαμενής από προκαθορισμένες διεργασίες (worker processes) για την επίτευξη παράλληλης επεξεργασίας, στις οποίες ανατίθενται οι προς εκτέλεση εργασίες με τη λογική της ουράς FIFO (First In, First Out). Με αυτόν τον τρόπο, επιτυγχάνεται η βέλτιστη αξιοποίηση των διαθέσιμων επεξεργαστικών πόρων χωρίς την ανάγκη χειροκίνητης δημιουργίας και τερματισμού διεργασιών. Η χρήση της Pool είναι ιδιαίτερα αποτελεσματική σε περιπτώσεις όπου υπάρχει ένα σύνολο ομοιογενών εργασιών που μπορούν να εκτελεστούν παράλληλα καθώς εξασφαλίζει υψηλή απόδοση με ισομερή κατανομή των διαθέσιμων πόρων.

4.3.2 Χρήση Gunicorn

Το Flask αν και αποτελεί ένα ελαφρύ και αποδοτικό framework, διαθέτει ενσωματωμένο server κατάλληλο μόνο για χρήση σε δοκιμαστικό περιβάλλον. Για την παραγωγική έκδοση, η εφαρμογή εκτελείται μέσω του Gunicorn WSGI server, ο οποίος λειτουργεί ως το ενδιάμεσο επίπεδο μεταξύ της Flask εφαρμογής και του συστήματος, διαχειρίζοντας τα εισερχόμενα αιτήματα με τρόπο αποδοτικό και με δυνατότητες γρήγορης κλιμάκωσης σε υψηλές απαιτήσεις. Πιο συγκεκριμένα, στο αρχείο Dockerfile έχει οριστεί να εκτελείται με ένα worker process δηλαδή μια ανεξάρτητη διεργασία που εξυπηρετεί αιτήματα και αξιοποιώντας το gthread worker class δεσμεύει 8 threads εξυπηρέτησης. Επίσης, έχει οριστεί --timeout=300 δευτερολέπτων στο worker, ώστε το Gunicorn να επανεκκινεί workers που δεν ανταποκρίνονται εντός του χρονικού ορίου. Κομβική παράμετρος για τη γρήγορη εκτέλεση είναι η δήλωση --preload, η οποία καθορίζει ότι η Flask εφαρμογή θα φορτωθεί στη μνήμη πριν από τη δημιουργία των worker processes. Με αυτόν τον τρόπο επιτυγχάνεται μικρότερος χρόνος εκκίνησης για κάθε worker, καθώς αποφεύγεται η επαναλαμβανόμενη φόρτωση του κώδικα.

4.3.3 Χρήση Puppeteer

Η χρήση αυτοματοποιημένων εργαλείων για την περιήγηση στους φυλλομετρητές (headless browsers) αποτελεί κρίσιμη μέθοδο για την εξόρυξη πληροφοριών από διαδικτυακές υπηρεσίες που προστατεύονται μέσω captcha και μηχανισμών ελέγχου session. Η εφαρμογή υιοθετεί μια

microservices αρχιτεκτονική όπου το Puppeteer λειτουργεί ως αυτόνομη υπηρεσία που εκτελείται σε ξεχωριστό docker container για την υλοποίηση των αναζητήσεων σε εξωτερικές πηγές (EPO, Espacenet, USPTO). Το σύστημα υλοποιείται ως ανεξάρτητος Node.js server στο αρχείο server.js με χρήση των βιβλιοθηκών puppeteer-extra και puppeteer-extra-plugin-stealth για την προσπέραση των αυτοματοποιημένων ελέγχων captcha. Στο docker-compose.yml ορίζεται το service "puppeteer_api" με ανεξάρτητο resource allocation, το οποίο εκτίθεται στο port 4000, ενώ μέσω του Node.js server παρέχονται REST endpoints στα οποία γίνονται κλήσεις από τα Python αρχεία cpc_service.py, espacenet_service.py, uspto_service.py κατά τις αναζητήσεις.

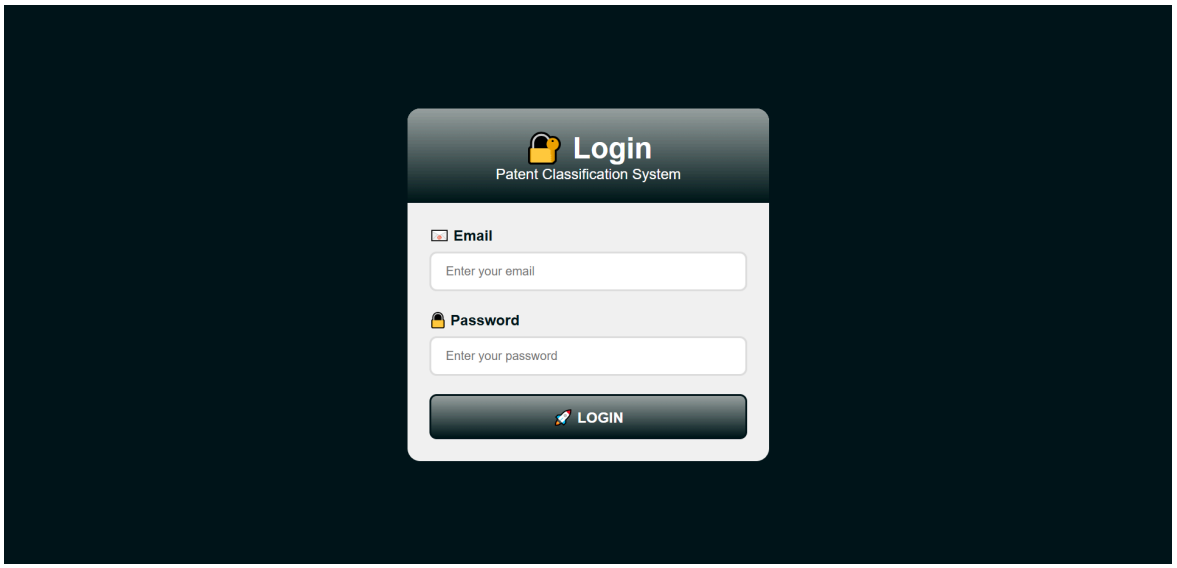
Η κλάση SimpleSessionPool αποτελεί μια σύγχρονη υλοποίηση διαχείρισης των ενεργών session για την καλύτερη παρακολούθηση πολλαπλών ενεργών browser instances στο Puppeteer API. Πιο συγκεκριμένα ελέγχεται η διαθεσιμότητα των sessions, καταμετρείται ο χρόνος λειτουργίας τους και προσμετρούνται οι αποτυχημένες προσπάθειες web scraping που εφάρμοσαν, με αυστηρό φίλτρο επανεκκίνησης νέου session στη δεύτερη αποτυχία. Έχουν υλοποιηθεί τρία ξεχωριστά Pools ένα για κάθε session των τριών υπηρεσιών EPO, Espacenet και USPTO με τη μόνη διαφορά μεταξύ τους να είναι η παράμετρος needsCaptcha η οποία αντικατοπτρίζει την απαίτηση αυτόματης επίλυσης captcha ελέγχου. Επιπλέον, κάθε session μπορεί να χαρακτηριστεί ως protected, οπότε δεν καταστρέφεται από τους αυτόματους μηχανισμούς τερματισμού των διεργασιών. Για το Espacenet και το USPTO η προστασία ισχύει για 30 λεπτά από την τελευταία χρήση που καταγράφηκε, ενώ για το EPO η προστασία παραμένει όσο το captcha παραμένει έγκυρο.

Η εφαρμογή πέρα από τη χρήση του puppeteer-extra-plugin-stealth για τον εμπλουτισμό των headers στα αιτήματα προς τις ιστοσελίδες και για την αποφυγή ανίχνευσης ως bot, ενσωματώνει εξελιγμένη διαχείριση captcha, για την αποτελεσματική του επίλυση σε όποια υπηρεσία αυτό απαιτηθεί. Εφόσον υπάρξει επιτυχής επίλυση αποθηκεύονται τα cookies και το session παραμένει ενεργό για μεμονωμένα ερωτήματα αλλά και μαζικά για 24 ώρες. Έπειτα, εφαρμόζεται αυτόματος τερματισμός του ενεργού session καθώς μετά το πέρας των 24 ωρών απαιτείται εκ νέου επίλυση captcha η οποία γίνεται αυτόματα στην επόμενη αναζήτηση του χρήστη μαζί με την αυτόματη δημιουργία νέου session.

Για την ευκολότερη παρακολούθηση όλων αυτών των ενεργειών έχει δημιουργηθεί ένα διαχειριστικό dashboard για τους διαχειριστές της εφαρμογής προσβάσιμο από το domain:4000/dashboard. Η πρόσβαση στη συγκεκριμένη σελίδα γίνεται μόνο από εξουσιοδοτημένους admin χρήστες μέσω authentication.

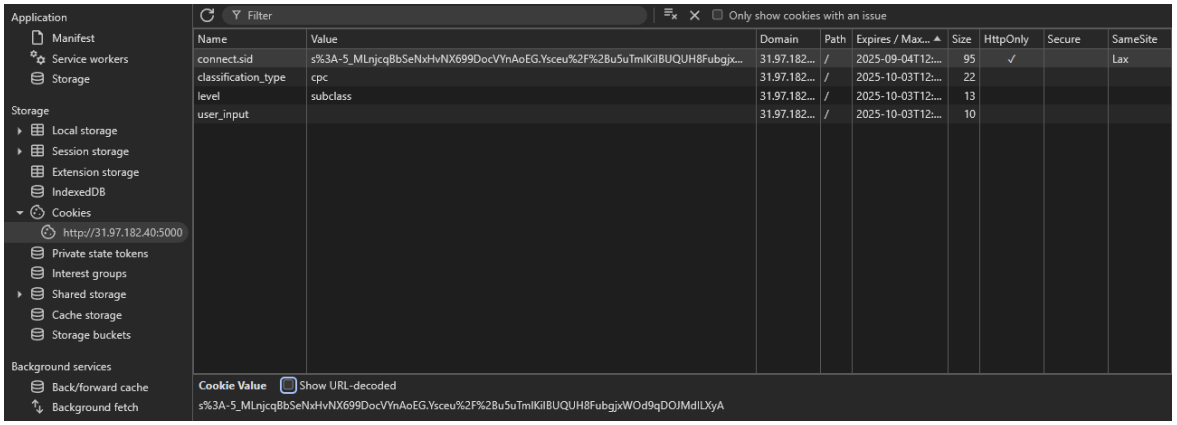
4.3.4 Βάση Δεδομένων και Αυθεντικοποίηση

Θέλοντας σε πρώτο στάδιο μια απλή και αποδοτική λύση για την εκχώρηση πρόσβασης στο διαχειριστικό περιβάλλον αλλά και μελλοντικά σε ορισμένες πρόσθετες υπηρεσίες, υλοποιήθηκε η κλάση UserDatabase που διαχειρίζεται τα δεδομένα χρηστών σε μορφή JSON αρχείου. Κατά την εκτέλεση του Dockerfile δημιουργείται αυτόματα ένας προκαθορισμένος λογαριασμός υπερδιαχειριστή (super admin) με στοιχεία από μεταβλητές περιβάλλοντος. Η προσέγγιση αυτή διασφαλίζει ότι το σύστημα λειτουργεί άμεσα μετά την εγκατάσταση. Ο super admin έχει τη δυνατότητα προσθήκης, ενημέρωσης και διαγραφής δυο τύπων χρήστη αυτών του user και του admin. Για την προσθήκη νέου χρήστη απαιτείται ένα email και ένας κωδικός ασφαλείας στον οποίο εφαρμόζεται κρυπτογράφηση MD5. Στην εφαρμογή υπάρχει διάκριση μεταξύ της ταυτοποίησης και της εξουσιοδότησης ενός χρήστη χρησιμοποιώντας διαφορετικούς κωδικούς σφάλματος. Αν ο χρήστης δεν έχει ταυτοποιηθεί εμφανίζεται κωδικός σφάλματος (401), ενώ για επαληθευμένους χρήστες χωρίς τα κατάλληλα δικαιώματα εμφανίζεται σφάλμα (403) απαγορευμένη πρόσβαση.



Κώδικας 4.3: Σελίδα ταυτοποίησης του χρήστη

Έπειτα από κάθε επιτυχημένη σύνδεση χρήστη δημιουργείται ένα session το οποίο σε JSON μορφή αποθηκεύει τα δεδομένα του αλλά και την τελευταία του χρονικά σύνδεση. Στο browser του χρήστη αποθηκεύεται ένα cookie, με μοναδικό session id και διάρκεια ζωής για 24 ώρες και έπειτα απαιτείται εκ νέου επανασύνδεση του χρήστη. Στην εικόνα 4.4 παρουσιάζεται το cookie που δημιουργείται από την εφαρμογή με το όνομα connect.sid, το οποίο χρησιμοποιείται για να συσχετιστεί το session στον server με τον συγκεκριμένο ταυτοποιημένο χρήστη.



Κώδικας 4.4: Cookie ταυτοποίησης στον browser

Κατά την αποσύνδεση, ο server διαγράφει τα session data από τη μνήμη του και ο browser διατηρεί το cookie μέχρι να λήξει. Η Flask εφαρμογή επικοινωνεί μέσω HTTP αιτήματος με την Puppeteer στη διεύθυνση domain:4000/auth-status για να επαληθεύσει ότι ο χρήστης έχει συνδεθεί. Η παράμετρος cookies=request.cookies διασφαλίζει ότι όλα τα cookies που έστειλε ο browser στην Flask εφαρμογή προωθούνται αυτούσια στην Puppeteer υπηρεσία.

```python
def is_user_authenticated():
    """Check if user is authenticated via the auth server"""
    try:
        import requests
        auth_url = f"http://{PUPPETEER_API_HOST}:4000/auth-status"
        response = requests.get(auth_url, timeout=3, cookies=request.cookies)
        if response.status_code == 200:
            data = response.json()
            return data.get('authenticated', False)
    except Exception as e:
        print(f"Auth check failed: {e}")
    return False
```

Κώδικας 4.5: Συνάρτηση ελέγχου ταυτοποίησης χρήστη

Η CORS διαμόρφωση της Puppeteer υπηρεσίας επιτρέπει αυτή την επικοινωνία μέσω της λίστας επιτρεπόμενων origins, διασφαλίζοντας ότι μόνο εξουσιοδοτημένες υπηρεσίες μπορούν να εκτελέσουν cross-origin αιτήματα προς το authentication API για τη μέγιστη ασφάλεια του συστήματος.

4.4 Δομή φακέλων και διάσπαση υπηρεσιών

Συνολικά στο έργο έχει εφαρμοστεί μια οργάνωση στη δομή των φακέλων και στη διάσπαση των υπηρεσιών που διευκολύνει τη συντήρηση και μακροπρόθεσμα την αναβάθμιση της εφαρμογής. Στη ρίζα του έργου βρίσκονται τα δύο βασικά αρχεία για τη διαμόρφωση του περιβάλλοντος Docker τα οποία είναι το Dockerfile και το docker-compose.yml. Συγκεκριμένα στο τελευταίο αρχείο, γίνεται η κατανομή των διαθέσιμων πόρων του διακομιστή όπου στη δική μας εφαρμογή έχουν κατανεμηθεί δίνοντας βαρύτητα στα τρία νευρωνικά μοντέλα που εκτελούνται μέσω του συστήματός. Για το container patent-app γίνεται μέσω του mem_reservation αρχική δέσμευση 5GB μνήμης RAM, ενώ το όριο τίθεται στα 15 GB με δυνατότητα επέκτασης έως τα 16GB με τη χρήση του διαθέσιμου χώρου swap. Όπως έχει ήδη αναλυθεί σε προηγούμενο κεφάλαιο η αρχιτεκτονική του συγκεκριμένου container βασίζεται στην παράλληλη εκτέλεση πολλαπλών πυρήνων του επεξεργαστή και για το λόγο αυτό γίνεται δέσμευση 3.5 πυρήνων ώστε να διασφαλιστούν η άρτια και αποδοτική λειτουργία χωρίς φαινόμενα bottleneck λόγω της απουσίας διαθέσιμης μνήμης.

Αντίστοιχα, για το container puppeteer_api εφαρμόζεται αρχική δέσμευση 1.5GB μνήμης ram, θέτοντας ως όριο τα 2.5GB με δυνατότητα επέκτασης έως τα 3.5GB με τη χρήση swap, σε περιπτώσεις υψηλού φόρτου και έντονης λειτουργίας της εφαρμογής. Αυτή η μεγάλη απόκλιση στην παραχώρηση χώρου μεταξύ των υπηρεσιών, οφείλεται στο γεγονός ότι το puppeteer_api εκτελεί http ερωτήματα τα οποία δεν απαιτούν αυξημένους πόρους για να εκτελεστούν σωστά και αυτό αποτυπώνεται στον αυστηρό περιορισμό που θέτει όριο χρήσης μόλις 0.8 πυρήνων χωρίς την παρουσίαση κανενός προβλήματος ακόμα και σε απαιτητικά σενάρια χρήσης. Μέσα στο Dockerfile γίνεται αναφορά στο αρχείο requirements.txt το οποίο περιλαμβάνει όλες τις απαραίτητες βιβλιοθήκες μαζί με τις εκδόσεις τους ώστε σε οποιαδήποτε μεταφορά σε νέο διακομιστή να αποφευχθούν προβλήματα ασυμβατότητας λόγω διαφορετικών εκδόσεων σε python dependencies.

Η εκκίνηση της εφαρμογής ορίζεται στο αρχείο app.py το οποίο είναι υπεύθυνο για το συγχρονισμό όλων των υπηρεσιών και βρίσκεται στη ρίζα του ίδιου καταλόγου. Ο φάκελος services αποτελεί το κεντρικό σημείο όπου οργανώνονται όλες οι υπηρεσίες ταξινόμησης σε ξεχωριστά Python modules (epo_service.py, espacenet_service.py, uspto_service.py, wipo_service.py, cpc_service.py,

model_service.py, Inference_cpc.py, Inference_ipcr.py) επιτρέποντας την αυτόνομη διαχείριση κάθε υπηρεσίας και διευκολύνοντας τη διαδικασίας επέκτασης του. Το HTML αρχείο (index.html) που είναι υπεύθυνο για την εμφάνιση του περιεχομένου στο χρήστη βρίσκεται αποθηκευμένο στο φάκελο templates, ενώ το απαραίτητο αρχείο μορφοποίησης της ιστοσελίδας (style.css) έχει τοποθετηθεί στο φάκελο static ακολουθώντας τις αρχές σχεδιασμού ιστοσελίδων. Στον ίδιο φάκελο είναι οργανωμένα τα JavaScript αρχεία που αφορούν τη διαδικασία παράλληλης εκτέλεσης των υπηρεσιών (async_search.js), το μηχανισμό μαζικής αναζήτησης (batch_mode.js), τεχνικές αποθήκευσης στην προσωρινή μνήμη του φυλλομετρητή (client_cache.js) αλλά και τη λειτουργία για την αποθήκευση και εμφάνιση προγενέστεστερων αναζητήσεων μέσω υλοποίησης του ιστορικού αναζήτησης (search_history.js).

Όλη η λειτουργία του container puppeteer_api βρίσκεται οργανωμένη μέσα στον αντίστοιχο φάκελο puppeteer_api στη ρίζα του καταλόγου όπου υπάρχει Dockerfile, package.json για τις εξαρτήσεις Node.js και το κεντρικό αρχείο server.js μέσα στο οποίο έχει υλοποιηθεί όλη η λειτουργία της Puppeteer API. Στον ίδιο φάκελο υπάρχει το αρχείο users.json για τη διαχείριση της βάσης δεδομένων των χρηστών. Για τη γρηγορότερη αποσφαλμάτωση, ο φάκελος logs έχει οριστεί ως κοινόχρηστο σημείο καταγραφής για όλες τις υπηρεσίες του συστήματος, συλλέγοντας logs τόσο από την κύρια Flask εφαρμογή όσο και από τη Puppeteer API μέσω των Docker volumes. Ολοκληρώνοντας τη δομή των αρχείων, αξίζει να γίνει αναφορά και στο φάκελο cache ο οποίος λειτουργεί ως σημείο αποθήκευσης για την υλοποίηση μηχανισμών memoization και lazy loading των μοντέλων μηχανικής μάθησης. Πιο συγκεκριμένα αποθηκεύονται τα προεκπαιδευμένα transformer-based μοντέλα από το HuggingFace Hub (tokenizers, weights, configuration files) μέσω της μεταβλητής περιβάλλοντος που έχει οριστεί σε HUGGINGFACE_HUB_CACHE=/app/cache. Παράλληλα, διατηρείται serialized versions των επεξεργασμένων embeddings και cached αποτελεσμάτων από εξωτερικές κλήσεις API, επιτυγχάνοντας σημαντική μείωση υπολογιστικών πόρων κατά τη φάση inference και ελαχιστοποιώντας τους χρόνους αρχικοποίησης των ProcessPoolExecutors που διαχειρίζονται τα μοντέλα CPC και IPCR classification.

4.5 Περιγραφή της διεπαφής

4.5.1 Εισαγωγή

Η ανάπτυξη σύγχρονων διαδικτυακών εφαρμογών πέρα από την άρτια τεχνική υλοποίηση στο backend, απαιτεί και την ολοκληρωμένη σχεδίαση μιας εύχρηστης προς αλληλεπίδραση διεπαφής για τον χρήστη. Στην παρούσα ενότητα θα αναλυθούν οι αρχές σχεδιασμού πάνω στις οποίες στηρίχθηκε η υλοποίηση, οι τεχνολογίες που χρησιμοποιήθηκαν αλλά και η παρουσίαση του τελικού αποτελέσματος. Η μελέτη και ο σχεδιασμός του frontend ξεκίνησε έχοντας ως γνώμονα την εργονομία για την καλύτερη αξιοποίηση όσο το δυνατόν περισσότερου μέρους της οθόνης που βλέπει ο χρήστης, την προσβασιμότητα αλλά και την απόδοση τόσο σε ταχύτητα φόρτωσης όσο και της συνολικής περιήγησης εσωτερικά σε αυτή. Για το σκοπό αυτό χρησιμοποιήθηκε ένας συνδυασμός vanilla τεχνολογιών συμπεριλαμβανομένων της HTML για τη δομή, της CSS3 για τη μορφοποίηση, JavaScript ES6+ για την προσθήκη διαδραστικότητας και Jinja2 ως template engine για τη μεταφορά των δεδομένων από το backend της Python στο frontend της διεπαφής. Αυτή η αρχιτεκτονική επιλογή εξασφαλίζει την πλήρη συμβατότητα με τα σύγχρονα web standards και παρέχει μια βάση επεκτάσιμη για μελλοντικές βελτιστοποιήσεις.

4.5.2 Αρχιτεκτονική και Δομή της Διεπαφής Χρήστη

Η βασική ιδέα σχεδιασμού για οθόνες μεγαλύτερες των 1024px σε πλάτος και σεβόμενοι τις αρχές του Responsive Design, ήταν η εκτέλεση και απόδοση της εμφάνισης στο πλήρες εύρος της οθόνης του χρήστη τόσο κατά μήκος όσο και κατά πλάτος, ώστε να μην απαιτείται η περιήγηση μέσω scroll. Η αρχιτεκτονική δομή της εφαρμογής βασίζεται στη φιλοσοφία του semantic web design, όπου κάθε HTML element έχει σαφή σημασιολογικό ρόλο. Η προσέγγιση αυτή είναι εναρμονισμένη με το όραμα του Tim Berners-Lee για ένα Σημασιολογικό Ιστό [15], στο οποίο το περιεχόμενο του Web είναι κατανοητό όχι μόνο από ανθρώπους αλλά και από μηχανές. Στο αρχείο index.html έχει υλοποιηθεί η δομή HTML ξεκινώντας με το header το οποίο περιλαμβάνει το λογότυπο της εφαρμογής και το κουμπί login για την αυθεντικοποίηση του χρήστη σε περίπτωση που διαθέτει λογαριασμό. Ακολουθεί μια φόρμα που μέσω μεθόδου POST και πατώντας το search button αποστέλλει το query που έχει συμπληρώσει ο χρήστης στο textarea. Ακριβώς κάτω από τη φόρμα υπάρχει το κουμπί Queries History και πατώντας ανοίγει σε μορφή αναδυόμενου παραθύρου το ιστορικό αναζητήσεων. Ως breakpoint της διεπαφής για το responsive mobile design έχει οριστεί όταν το viewport είναι μικρότερο από 768px και για την καλύτερη αξιοποίηση του χώρου έχει επιλεγεί το Queries History να μετατρέπεται σε εικονίδιο, μετακινώντας το αριστερά του Login button διατηρώντας δομή με την οποία είναι εξοικειωμένοι οι χρήστες.

Το κύριο container main-wrapper υλοποιεί ένα layout τριών στηλών χρησιμοποιώντας CSS Flexbox. Η πρώτη στήλη με κλάση first-wrapper περιέχει τις διαθέσιμες επιλογές ρυθμίσεων για την εφαρμογή μεταξύ των οποίων είναι η επιλογή τύπου ταξινομικής πληροφορίας IPC ή CPC προς αναζήτηση μέσω toggle-switch, το επίπεδο αναζήτησης δομημένο σε dropdown και ένα input πλαίσιο για τη συμπλήρωση πολλαπλών σωστών συμβόλων τα οποία αξιοποιούνται στις μετρικές απόδοσης. Στο τέλος της ίδιας στήλης υπάρχουν τα elements για τη χρήση της λειτουργίας batch mode, τα οποία αποτελούνται από ένα πλαίσιο input που δέχεται μόνο αρχεία τύπου .xml , το κουμπί για την εκτέλεση του Batch το οποίο πατώντας το καλεί τη συνάρτηση handleBatchExecute() και ένα ακόμα κουμπί για τη λήψη ενός δείγματος του προτύπου που πρέπει να ακολουθεί το xml αρχείο. Για την καλύτερη περιήγηση στις κινητές συσκευές έχει προστεθεί η λειτουργία επέκτασης και συρρίκνωσης του πλαισίου πατώντας το αντίστοιχο ενημερωτικό εικονίδιο βέλους.

Το κεντρικό container middle-wrapper έχει υλοποιηθεί με σκοπό της εμφάνιση των αποτελεσμάτων και την περιήγηση μεταξύ των classifiers. Το layout εσωτερικά αποτελείται από τέσσερα containers, ξεκινώντας με το metrics-box στο οποίο απεικονίζονται οι μετρικές απόδοσης συνολικά για όλους τους classifiers και ακολουθούν με μορφή tabs οι classifiers. Στα tabs πέρα από το όνομα του ταξινομητή, εμφανίζεται κατά τη διάρκεια εκτέλεσης της αναζήτησης ένα loading spinner ώστε να πληροφορεί τον χρήστη ότι εκτελείται σωστά ενώ πατώντας το υπάρχει η δυνατότητα ακύρωσης της συγκεκριμένης διεργασίας. Μετά την επιτυχή εκτέλεση, τη θέση του loading spinner παίρνει ο αριθμός των αποτελεσμάτων που έχει επιστρέψει ο ταξινομητής και εμφανίζεται μια χρωματική ένδειξη σε μορφή κύκλου με σκοπό την οπτική μεταφορά πληροφορίας μέσω χρώματος για το αποτέλεσμα του f1 score του κάθε ταξινομητή. Παρακάτω και πριν από τα αποτελέσματα, εμφανίζεται ένα floating element που περιέχει όλες τις μετρικές Precision, Recall και F1 score του συγκεκριμένου classifier. Για τη δομή των αποτελεσμάτων έχει υιοθετηθεί η δοκιμασμένη τεχνική της Google, χρησιμοποιώντας ως hyperlink το σύμβολο ταξινόμησης, ακολουθούμενο από τον υπερσύνδεσμο για την επίσημη ιστοσελίδα και ως meta description προστίθεται το περιγραφικό κείμενο του συμβόλου. Στην παραπάνω δομή και για την καλύτερη οργάνωση, τα αποτελέσματα παρουσιάζονται σε μορφή accordion διατηρώντας εμφανή μόνο το σύμβολο ταξινόμησης. Συνολικά η περιήγηση στο container

γίνεται με εσωτερικό κάθετο scroll ενώ ο μέγιστος αριθμός συμβόλων που μπορούν να εμφανιστούν κάθε φορά έχει οριστεί στα δέκα.

Η τρίτη στήλη third-wrapper χρησιμοποιείται για τη συγκεντρωτική εμφάνιση ενημερωτικών πληροφοριών. Το πρώτο container common-results περιλαμβάνει τα κοινά σύμβολα δηλαδή έναν κωδικό ο οποίος έχει εμφανιστεί σε τουλάχιστον δύο ταξινομητές, συνοδευόμενο από το όνομα των ταξινομητών που εντοπίστηκε για την πλήρη ενημέρωση του χρήστη. Ακολουθώντας τον ίδιο τρόπο απεικόνισης, στην περίπτωση που έχει συμπληρωθεί το input πεδίο της πρώτης στήλης με id="user_codes", τα ενημερωτικά αποτελέσματα θα φανούν με το κατάλληλο μήνυμα στο container matched-user-codes. Η ίδια λειτουργία expand/collapsed κατά την mobile περιήγηση έχει υιοθετηθεί και σε αυτή τη στήλη. Είναι σημαντικό να κατανοηθεί πλήρως πως όλες οι αλλαγές συμβαίνουν δυναμικά μέσω Javascript κατά τη διάρκεια περιήγησης του χρήστη διαφοροποιώντας την εφαρμογή από τις απλές ιστοσελίδες με στατικές απεικονίσεις αποτελεσμάτων.

4.5.3 Jinja2

Το Jinja2 είναι ένα σύγχρονο template engine για τη γλώσσα προγραμματισμού Python, χρησιμοποιείται κυρίως σε διαδικτυακές εφαρμογές και είναι σχεδιασμένο για να αποδίδει δυναμικά HTML περιεχόμενο γεφυρώνοντας το backend με το frontend. Προσφέρει τη δυνατότητα προσθήκης μεταβλητών, βρόχων και λογικές συνθήκες μέσα στο βασικά αρχείο HTML χωρίς να αλλοιώνεται η δομή του κώδικα. Το βασικό του πλεονέκτημα σε αντίθεση με τη στατική HTML που παραμένει αμετάβλητη, είναι το γεγονός πως επιτρέπει τη δυναμική απόδοση δεδομένων ανάλογα με τη λογική συνθήκη της εφαρμογής. Στον τομέα της απόδοσης, ένα βασικό χαρακτηριστικό είναι ότι τα templates μεταγλωτίζονται σε βελτιστοποιημένο Python κώδικα τη στιγμή που χρειάζονται και αποθηκεύονται προσωρινά στη μνήμη cache, ή εναλλακτικά μπορούν να μεταγλωτιστούν εκ των προτέρων, εξασφαλίζοντας υψηλή απόδοση και χαμηλή καθυστέρηση στην παραγωγή δυναμικών σελίδων.

Στην παρούσα εργασία, το Jinja2 χρησιμοποιείται εκτενώς στο αρχείο index.html όπου τα στατικά στοιχεία της HTML συνδυάζονται με δυναμικά blocks. Η ενσωμάτωση του Jinja2 στην Flask εφαρμογή πραγματοποιείται μέσω του function render_template, το οποίο αποτελεί το κεντρικό σημείο επικοινωνίας μεταξύ των Python backend services και του frontend HTML. Βασικό χαρακτηριστικό είναι πως όλες οι εντολές γράφονται μέσα σε αγκύλες {}, όπως εντοπίζεται στη χρήση για την απόδοση των αποτελεσμάτων με λογικό έλεγχο συνθήκης {% if exe_results %} ... {% endif %} και στο βρόχο επανάληψης {% for result in epo_search_results %} ... {% endfor %}, που δημιουργούνται δυναμικές λίστες CPC/IPC συμβόλων. Οι μεταβλητές ενσωματώνονται χρησιμοποιώντας διπλές αγκύλες {{ result.link }}, ενώ ιδιαίτερα σημαντική κρίνεται η χρήση του φίλτρου | safe στις περιγραφές όπως στο {{ result.description | safe}} το οποίο απενεργοποιεί τον αυτόματο μηχανισμό HTML escaping επιτρέποντας στο περιεχόμενο να αποδοθεί αυτούσιο. Η χρήση του φίλτρου έχει γίνει προσεκτικά και στοχευμένα σε μεταβλητές που δεν θα μπορούσαν να αποτελέσουν κίνδυνο σε ευπάθειες XSS.

Η υβριδική αρχιτεκτονική μεταξύ JavaScript και Jinja2 στην εφαρμογή αντιπροσωπεύει μια εξελιγμένη προσέγγιση που συνδυάζει τα πλεονεκτήματα του server-side rendering με τις δυνατότητες του δυναμικού client-side προγραμματισμού. Ένα χαρακτηριστικό παράδειγμα είναι η μέθοδος showSearchingState, η οποία δημιουργεί τα button tabs για τους classifiers μόνο εάν τα αντίστοιχα blocks έχουν αποδοθεί από το Jinja2. Αντίστοιχα, στο αρχείο client_cache.js χρησιμοποιείται το key schema engine::keyword::level και με κλήση της συνάρτησης renderCachedResults ενσωματώνει στο DOM περιεχόμενο μέσα σε <details> tags που ορίζονται από το template. Επιπλέον, στο αρχείο

search_history.js, η υλοποίηση του αναδυόμενου παραθύρου για το ιστορικό αναζητήσεων στηρίζεται επίσης σε στατικό wrapper που αποδίδεται από το Jinja2, αλλά το περιεχόμενο της ενότητας γεμίζει δυναμικά με JavaScript, βάσει των αποθηκευμένων δεδομένων του localStorage.

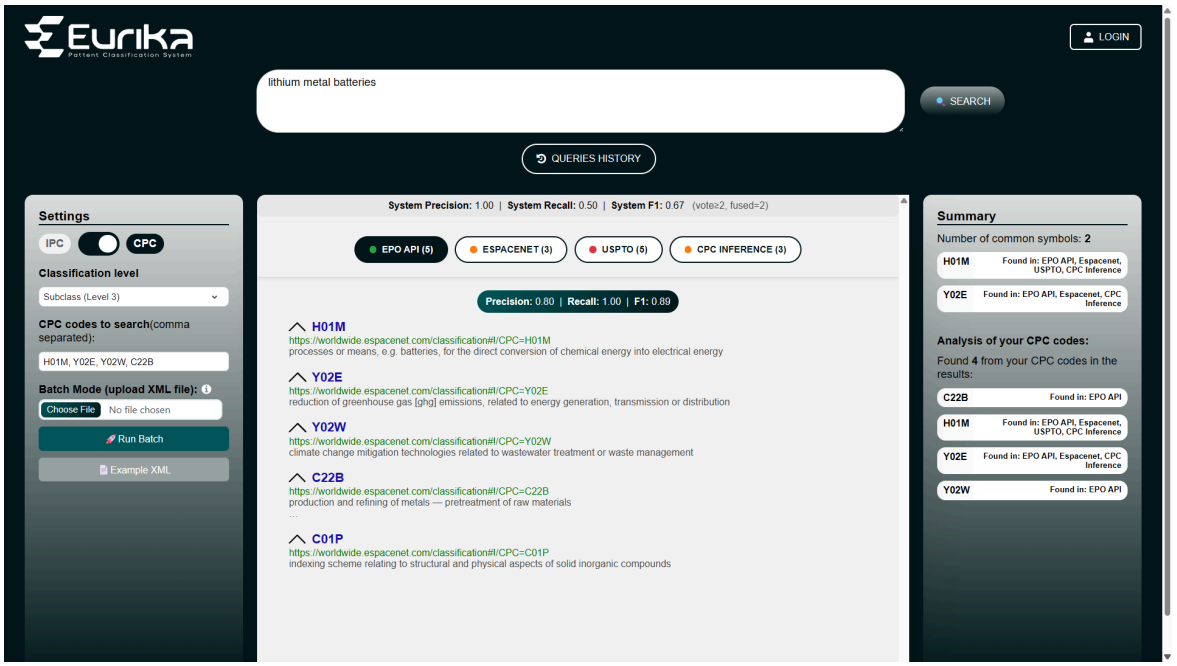
4.5.4 UI/UX Design

Η σχεδίαση της εφαρμογής Eurika στηρίχθηκε στις δύο θεμελιώδεις αρχές του User Experience Design, της προοδευτικής αποκάλυψης (Progressive Disclosure) και της οπτικής ιεραρχίας (Visual Hierarchy). Η προοδευτική αποκάλυψη αποτελεί μια σχεδιαστική στρατηγική κατά την οποία γίνεται σταδιακή εμφάνιση των λειτουργιών στο χρήστη, εμφανίζοντας σε πρώτο πλάνο μόνο το βασικό περιεχόμενο. Για την εμφάνιση των υπολοίπων πληροφοριών επιλέγεται η χρήση των buttons ως την πιο άμεση λύση για την αποφυγή καθυστερήσεων στις αντιδράσεις του χρήστη, αλλά και για την καλύτερη οργάνωση του περιεχομένου. Η προσέγγιση αυτή μειώνει τη γνωστική επιβάρυνση που προκαλείται στο χρήστη όταν έρχεται αντιμέτωπος με υπερβολικό όγκο δεδομένων, ενισχύει την αίσθηση ελέγχου και βελτιώνει την κατανόηση της πληροφορίας [16]. Παράλληλα, η οπτική ιεραρχία καθορίζει τη σειρά με την οποία το ανθρώπινο μάτι επεξεργάζεται τα στοιχεία μιας διεπαφής, χρησιμοποιώντας παράγοντες όπως το μέγεθος, το χρώμα, την αντίθεση, την στοίχιση και τον κενό χώρο για να δημιουργήσει μια ιεραρχημένη παρουσίαση από το πιο σημαντικό στοιχείο του περιεχομένου στο λιγότερο σημαντικό [17].

Στην διεπαφή η αρχή του Progressive Disclosure εφαρμόζεται παρουσιάζοντας σε πρώτο στάδιο μόνο τις πλήρως αναγκαίες ρυθμίσεις για τη λειτουργία της εφαρμογής με αυτές να περιλαμβάνουν τη φόρμα αναζήτησης, τη στήλη των ρυθμίσεων και το κουμπί για τη σύνδεση του χρήστη. Ξεκινώντας μια αναζήτηση, εμφανίζονται μόνο τα αποτελέσματα του πρώτου classifiers ενώ τα υπόλοιπα κρύβονται μέσω της ομαδοποίησης που έχει γίνει σε δομή tabs και παρουσιάζονται πατώντας το αντίστοιχο tab που έχει υλοποιηθεί ως στοιχείο button. Ακολουθώντας την ίδια τεχνική, η χρήση της λειτουργίας batch mode, η εμφάνιση του ιστορικού αναζήτησης και του αναδυόμενου παραθύρου σύνδεσης ενεργοποιούνται μέσω κουμπιού. Οι συμπληρωματικές πληροφορίες της τρίτης στήλης αποκαλύπτονται σταδιακά σε κάθε ολοκληρωμένη αναζήτηση διατηρώντας τη λογική της προοδευτικής αποκάλυψης.

Στη βασική δομή των τριών στηλών έχει υιοθετηθεί και η αρχή του Visual Hierarchy καθώς στο κεντρικό container έχει παραχωρηθεί το 60% του συνολικού πλάτους οθόνης θέλωντας να περάσουμε το μήνυμα στο χρήστη ότι σε αυτό το σημείο βρίσκεται η κύρια πληροφορία, ενώ στις πλευρικές στήλες που λειτουργούν ως συμπληρωματικές έχει παραχωρηθεί από 20% στην καθεμιά διατηρώντας ωστόσο την εύκολη πλοήγηση. Η χρωματική παλέτα αποτελείται από το κυρίαρχο σκούρο χρώμα με hex code #00181a, το δευτερεύον ανοιχτό χρώμα με hex code #99a2a1 και από τη χρήση του λευκού κυρίως για το πλαίσιο αναζήτησης και ως το χρώμα κειμένου. Αυτή η χρωματική επιλογή δεν έγινε τυχαία, αλλά βασίστηκε στο γεγονός ότι τα teal και grey tones συνδέονται ψυχολογικά με την τεχνολογία, την αξιοπιστία και σύμφωνα με μελέτες σημειώθηκε αυξημένος χρόνος παραμονής των χρηστών σε σελίδες που τα χρησιμοποιούσαν ως χρώμα background, χαρακτηριστικά που είναι σημαντικά για την εφαρμογή μας [18]. Χρησιμοποιώντας γραμμικά gradients δόθηκε βάθος, προσδίδοντας οπτική ιεραρχία καθώς τα κείμενα που θέλαμε να κεντρίσουν την προσοχή προστέθηκαν με λευκό χρώμα στη σκούρη πλευρά του gradient δημιουργώντας την αντίθεση που απαιτεί η αρχή Visual Hierarchy. Με τη χρήση των hover states ενισχύθηκε η αλληλεπίδραση μέσω την καλύτερης πληροφόρησης που λαμβάνει ο χρήστης, ενώ ως κύρια γραμματοσειρά επιλέχθηκε η

Arial, sans-serif; λόγω των ευανάγνωστων χαρακτηριστικών της και της ευρείας υποστήριξης από όλους τους περιηγητές.



Εικόνα 4.6: Διεπαφή χρήστη της εφαρμογής Eurika

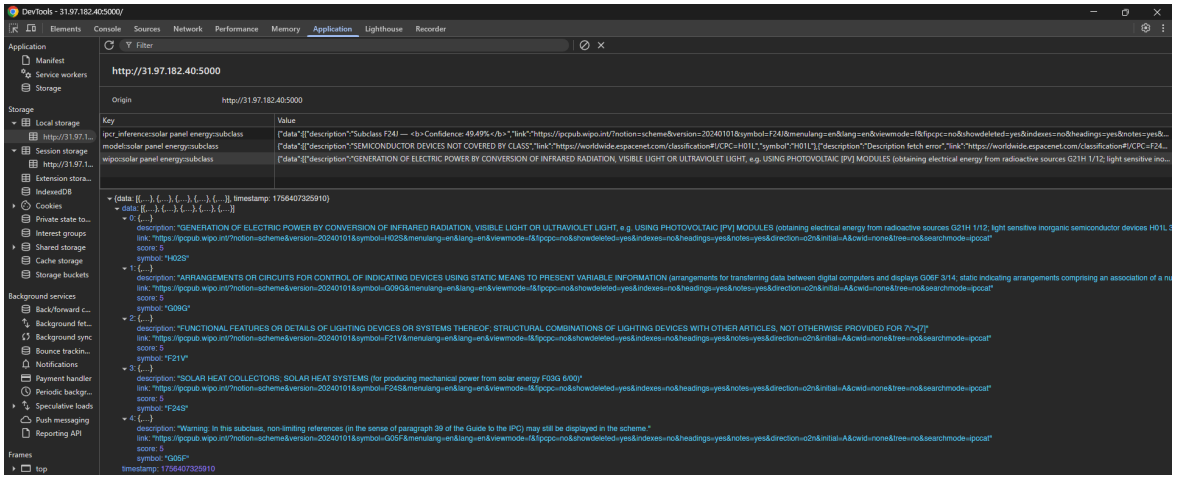
4.6 Τεχνολογίες Client-Side Storage και Data Serialization

4.6.1 Μηχανισμοί Browser Caching

Η αρχιτεκτονική του συστήματος ενσωματώνει πολυεπίπεδους μηχανισμούς caching που εκτελούνται τόσο στο client-side όσο και στο server-side περιβάλλον. Με τον όρο browser caching, δεν αναφέρεται απλά σε ένα τεχνικό χαρακτηριστικό, αλλά ένας από τους πιο άμεσους τρόπους βελτίωσης της εμπειρίας του χρήστη στις διαδικτυακές εφαρμογές. Στην πράξη, το browser caching σημαίνει ότι η εφαρμογή αποθηκεύει προσωρινά τα αποτελέσματα που έχουν ήδη βρεθεί, ώστε να μην χρειάζεται κάθε φορά να ξαναγίνεται επικοινωνία με τον διακομιστή. Αυτό όχι μόνο μειώνει τον χρόνο αναμονής, αλλά και εξοικονομεί σημαντικούς υπολογιστικούς πόρους. Στην εφαρμογή Eurika, η τεχνολογία JavaScript χρησιμοποιείται για την αποθήκευση και την ανάκτηση δεδομένων τοπικά, χρησιμοποιώντας το Web Storage API και συγκεκριμένα το χώρο localStorage. Η επιλογή αυτή διασφαλίζει αποθήκευση στην πλευρά του client, ακόμη και μετά από ανανέωση ή επανεκκίνηση του φυλλομετρητή για όσο χρονικό διάστημα έχει οριστεί.

Ακολουθώντας και σε αυτό το επίπεδο τη λογική των microservices, έχει δημιουργηθεί για τις ανάγκες της client-side caching αρχιτεκτονικής, το ανεξάρτητο αρχείο client_cache.js το οποίο εκτελείται για την αποθήκευση και ανάκτηση των αποτελεσμάτων αναζήτησης. Η συνάρτηση getCacheKey() δημιουργεί μοναδικά κλειδιά με τη μορφή ${engine}::${keyword.toLowerCase()}::${level.toLowerCase()}, εξασφαλίζοντας την ορθή αναγνώριση των cached δεδομένων ανεξάρτητα από τον classifier και το επίπεδο ταξινόμησης (section, class, subclass, group). Ο μηχανισμός αποθήκευσης που υλοποιείται στη συνάρτηση cacheResults(), δημιουργεί δομημένα αντικείμενα της μορφής JSON, που περιλαμβάνουν ως

δεδομένα τα επιστρεφόμενα αποτελέσματα του κάθε ταξινομητή, χρονική σήμανση (timestamp) μέσω της μεθόδο Date.now(), τα κοινά σύμβολα μεταξύ των ταξινομητών και αν βρέθηκαν τα σωστά σύμβολα προς αναζήτηση που εισήγαγε ο χρήστης.



Εικόνα 4.7: Μορφή αποθηκευμένων δεδομένων στο localStorage

Η συνάρτηση getCachedResults() διαχειρίζεται τον έλεγχο και την ανάκτηση των cached αποτελεσμάτων και εφαρμόζει ένα default TTL των 30 ημερών το οποίο ορίζεται σε milliseconds ως εξής maxAgeMs = 30 * 24 * 60 * 60 * 1000. Σε περίπτωση λήξης ή αστοχίας parsing, το σύστημα αυτόματα αφαιρεί τα αντίστοιχα entries από το localStorage με τη χρήση της μεθόδου localStorage.removeItem(key), διατηρώντας έτσι την ακεραιότητα της προσωρινής μνήμης cache.

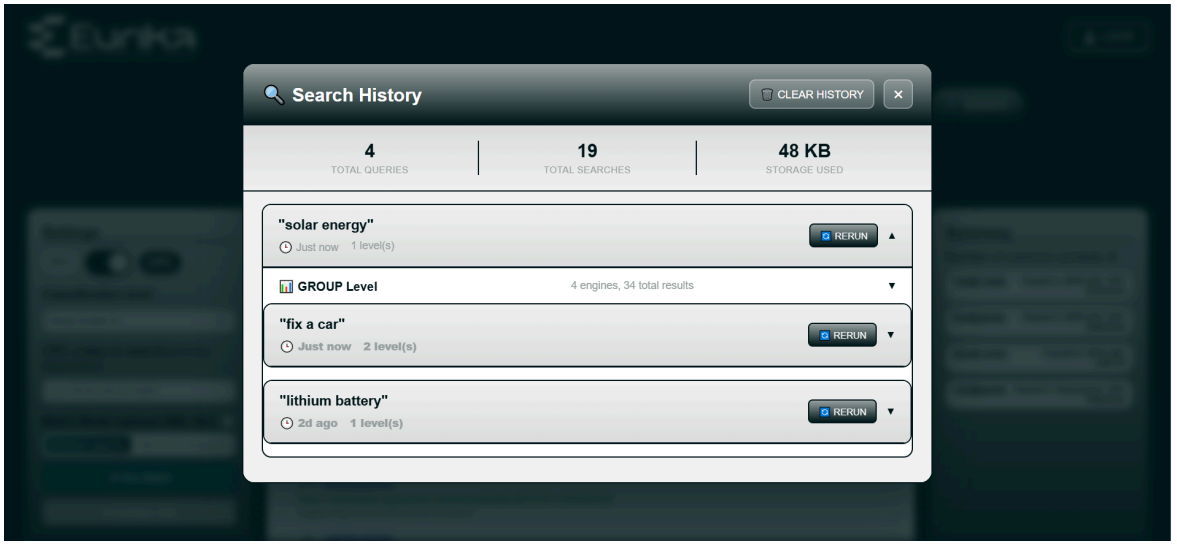
4.6.2 Ιστορικό Αναζητήσεων

Πέρα από τη βελτίωση της εμπειρίας του χρήστη και της εξοικονόμησης πόρων η παραπάνω διαχείριση της cache αξιοποιήθηκε και στη δημιουργία ενός ιστορικού αναζητήσεων του χρήστη και το οποίο υλοποιήθηκε ανεξάρτητα στο αρχείο search_history.js. Συγκεκριμένα ο πυρήνας λειτουργίας βασίζεται στην κλάση SearchHistoryManager που φιλτράρει όλα τα αποθηκευμένα κλειδιά που ακολουθούν το μοτίβο engine::query::level και παρέχει ταξινομημένη χρονικά οργάνωση των αποθηκευμένων αποτελεσμάτων στη μνήμη cache. Η παρουσίασή τους γίνεται μέσω ενός σύγχρονου και εργονομικού αναδυόμενου παραθύρου. Για κάθε καταγεγραμμένο query δημιουργείται μία εγγραφή που περιλαμβάνει όλες τις αναζητήσεις που εκτελέστηκαν, ταξινομημένες με βάση τη χρονοσφραγίδα (timestamp), ενώ διατηρείται και η ένδειξη της πιο πρόσφατης αναζήτησης.

Στο αναδυόμενο παράθυρο που εμφανίζεται στην εικόνα 4.8, πέρα από τα queries αναγράφεται και ο συνολικός χώρος που καταλαμβάνουν τα δεδομένα στο localStorage. Για κάθε εγγραφή υπολογίζεται το συνολικό μέγεθος ως άθροισμα χαρακτήρων κλειδιού και τιμής (totalSize += localStorage[key].length + key.length), με χρονική πολυπλοκότητα O(n) όπου n το πλήθος κλειδιών στο localStorage και χωρική πολυπλοκότητα O(1) αφού χρησιμοποιείται μόνο μία μεταβλητή totalSize για τη συσσώρευση, και το αποτέλεσμα προβάλλεται με προοδευτική μορφοποίηση σε bytes, KB ή MB. Η διαγραφή του ιστορικού αναζήτησης υλοποιείται με τη μέθοδο clearHistory(), σαρώνοντας όλα τα κλειδιά του localStorage και εφαρμόζοντας τρία φίλτρα αναγνώρισης ως εξής:

- Δομικό μοτίβο : έλεγχος ώστε τα keys να περιλαμβάνουν το διαχωριστικό :: που δηλώνει τη μορφή engine::query::level.
- Αναγνώριση του ταξινομητή: ελέγχονται λέξεις κλειδιά που ορίζονται με βάση τα ονόματα που έχουν δοθεί στους ταξινομητές της εφαρμογής όπως espacenet, uspto, wipo ώστε να ταυτοποιηθεί ο μηχανισμός που επέστρεψε την εγγραφή.
- Αναγνώριση επιπέδου ταξινόμησης: εντοπίζονται όροι όπως το section,class,subclass και group

Για κάθε κλειδί που περνά αυτά τα κριτήρια, καλείται η μέθοδος localStorage.removeItem(key), η οποία το αφαιρεί μόνιμα από την αποθήκευση του browser. Η διαδικασία αυτή έχει πολυπλοκότητα $O(n \times k)$, όπου n είναι το πλήθος των εγγραφών και k ο αριθμός των ελέγχων στα τρία μοτίβα που γίνονται ανά κλειδί. Με αυτό τον σχεδιασμό, το σύστημα αποφεύγει την ανεξέλεγκτη χρήση του localStorage.clear(). Επιπλέον, το σύστημα παρέχει λειτουργία επαναφοράς μέσω της rerunSearch() που επιτρέπει την άμεση επανεκτέλεση προηγούμενων αναζητήσεων. Η μέθοδος αυτή συμπληρώνει αυτόματα το search field με το επιλεγμένο query, κλείνει το modal και ενεργοποιεί προγραμματιστικά την υποβολή της φόρμας μέσω event dispatching, με συνολικό χρόνο απόκρισης κάτω από 50ms.



Εικόνα 4.8: Αναδυόμενο παράθυρο ιστορικού αναζήτησης

4.7 Ασφάλεια δικτύου και CORS policies

Η ασφάλεια δικτύου στην εφαρμογή υλοποιείται μέσω μιας συνδυαστικής προσέγγισης που απαρτίζεται από CORS (Cross-Origin Resource Sharing) policies, διαμόρφωση περιβάλλοντος εφαρμογής και δυναμική διαχείριση επιτρεπόμενων πηγών. Η εφαρμογή διαχειρίζεται το πρόβλημα του cross-origin access μεταξύ του κυρίου Flask server (port 5000) και του Puppeteer API server (port 4000), ενώ παράλληλα εξασφαλίζει την προστασία από μη εξουσιοδοτημένες προσβάσεις. Η CORS διαμόρφωση στον Puppeteer server βασίζεται σε μια λίστα προκαθορισμένων επιτρεπόμενων origins που περιλαμβάνει τόσο development όσο και production διευθύνσεις.

```
// CORS Configuration
const cors = require('cors');
const FRONTEND = process.env.FRONTEND_ORIGIN || 'http://localhost:5000';
const ALLOWED_ORIGINS = [
  'http://localhost:5000',
  'http://127.0.0.1:5000',
  'http://localhost:4000',
  'http://127.0.0.1:4000',
  `http://${process.env.SERVER_HOST || 'localhost'}:5000`,
  `https://${process.env.SERVER_HOST || 'localhost'}:5000`,
  `http://${process.env.SERVER_HOST || 'localhost'}:4000`,
  `https://${process.env.SERVER_HOST || 'localhost'}:4000`,
  FRONTEND,
  FRONTEND.replace(':5000', ':4000')
].filter(Boolean);

app.use(cors({
  origin(origin, cb) {
    // allow same-origin (no Origin header) και τα επιτρεπτά origins
    if (!origin || ALLOWED_ORIGINS.includes(origin)) return cb(null, true);
    cb(new Error(`Not allowed by CORS: ${origin}`));
  },
  credentials: true,
  methods: ['GET','POST','PUT','DELETE','OPTIONS'],
  allowedHeaders: ['Content-Type','Authorization']
}));

const useSecureCookies = process.env.COOKIE_SECURE === 'true'; // true σε HTTPS

// Session middleware
app.use(session({
  secret: AUTH_CONFIG.SESSION_SECRET,
  resave: false,
  saveUninitialized: false,
  cookie: {
    secure: useSecureCookies, // αν HTTPS -> true
    sameSite: 'lax',
    maxAge: 24 * 60 * 60 * 1000
  }
}));
```

Εικόνα 4.9: Puppeteer CORS διαμόρφωση

Η παραπάνω διαμόρφωση καλύπτει διαφορετικά σενάρια ανάπτυξης και δημοσίευσης της εφαρμογής, λαμβάνοντας υπόψη περιπτώσεις εκτέλεσης σε τοπικό περιβάλλον (localhost, 127.0.0.1), χρήση προσαρμοσμένων hostnames, καθώς και την υποστήριξη τόσο για HTTP όσο και για HTTPS πρωτόκολλα. Ο μηχανισμός filter(Boolean) συμβάλλει στην αποφυγή ανεπιθύμητων καταχωρήσεων (undefined ή null) από τη λίστα αποδεκτών προελεύσεων (origins). Η επικύρωση των αιτημάτων προέλευσης υλοποιείται μέσω της callback συνάρτησης που συνδυάζεται με τη μέθοδο app.use(cors()). Με αυτόν τον τρόπο υποστηρίζονται τόσο τα αιτήματα της ίδιας προέλευσης

(same-origin) όταν απουσιάζει το header Origin όσο και τα αιτήματα από τα προκαθορισμένα επιτρεπόμενα origins. Η παράμετρος credentials: true επιτρέπει την αποστολή cookies και authentication headers μεταξύ πελάτη και εξυπηρετητή, στοιχείο κρίσιμο για τη διατήρηση του session state της εφαρμογής.

Η διαχείριση των session cookies γίνεται μέσω του middleware session, με διαφορετικές παραμέτρους προσαρμοσμένες στο εκάστοτε παραγωγικό περιβάλλον. Ειδικότερα, η ρύθμιση sameSite: 'lax' επιτυγχάνει ισορροπία μεταξύ ασφάλειας και λειτουργικότητας, παρέχοντας προστασία από επιθέσεις τύπου Cross-Site Request Forgery (CSRF), χωρίς να περιορίζει την εσωτερική πλοήγηση του χρήστη στην εφαρμογή. Η δυνατότητα αποστολής authentication tokens υλοποιείται μέσω της συμπερίληψης του Authorization header στη λίστα των επιτρεπόμενων headers.

Επιπλέον, για την ορθή λειτουργία σε περιβάλλον με reverse proxies ή load balancers, έχει ενεργοποιηθεί η ρύθμιση app.set('trust proxy', true). Η επιλογή αυτή διασφαλίζει ότι αναγνωρίζεται σωστά η πραγματική διεύθυνση IP του πελάτη και υποστηρίζεται η τερματική διαχείριση HTTPS σε επίπεδο proxy, κάτι που βελτιώνει τόσο την ασφάλεια όσο και τη λειτουργικότητα σε υποδομές νέφους. Συνολικά, η προσέγγιση αυτή δημιουργεί έναν ισορροπημένο μηχανισμό ασφαλείας που προστατεύει την εφαρμογή από επιθέσεις cross-origin, χωρίς να θυσιάζει την ευελιξία και την προσαρμοστικότητα σε διαφορετικά σενάρια ανάπτυξης.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

5.1 Χρήση web scraping και APIs

Όπως έχει ήδη αναλυθεί λεπτομερώς η χρήση τόσο επίσημων API όσο και τεχνικών web scraping αποτέλεσαν καίρια σημεία της παρούσας εργασίας. Από τη μία πλευρά, το επίσημο API της EPO προσέφερε μια τυποποιημένη και αξιόπιστη μέθοδο για την ανάκτηση δεδομένων μέσω της χρήσης συγκεκριμένων προσβάσιμων endpoints. Η διαδικασία ήταν αξιόπιστη και αποδοτική σε κάθε νέα κλήση και τα επιστρεφόμενα αποτελέσματα σωστά δομημένα καθιστώντας τα κατάλληλα προς επεξεργασία. Έτσι, η ενσωμάτωσή τους στην εφαρμογή αποτέλεσε τη βάση για μια παραγωγική και σταθερή υποδομή. Από την άλλη πλευρά, η ανάγκη προσπέλασης δεδομένων από συστήματα που δεν διαθέτουν ανοικτά API, όπως το USPTO και το WIPO κατέστησε απαραίτητη την εφαρμογή τεχνικών web scraping.

Η εμπειρία αυτή αποτέλεσε ουσιαστικά την πρώτη πρακτική ενασχόλησή μου με το αντικείμενο, γεγονός που απαίτησε σημαντική μελέτη για την ενημέρωση σχετικά με τα διαθέσιμα εργαλεία scraping μέσω Python και JavaScript αλλά και για την κατανόηση του τρόπου λειτουργίας των επιμέρους βιβλιοθηκών και των ιδιαιτεροτήτων τους. Επίσης, χρειάστηκε η εξοικείωση με τον εντοπισμό μη δημοσιευμένων APIs μέσω των εργαλείων των φυλλομετρητών, και πιο συγκεκριμένα η διαδικασία περιελάμβανε τη λεπτομερή παρακολούθηση δικτυακών αιτημάτων, την αναγνώριση των endpoints που δεν δημοσιοποιούνται επισήμως και την αναπαραγωγή τους μέσω αυτοματοποιημένων scripts. Παρά τις τεχνικές δυσκολίες που προέκυπταν κατά το στάδιο της ανάπτυξης, όπως οι αλλαγές στη δομή των ιστοσελίδων, οι προσθήκη περιορισμών captcha και η ανάγκη για διαχείριση των cookies, η τελική εφαρμογή προσφέρει μέσω web scraping ταχύτατη πρόσβαση σε κρίσιμες πληροφορίες που δεν θα μπορούσαν να ληφθούν αυτοματοποιημένα με διαφορετικό τρόπο.

Η συνολική αποτίμηση καταδεικνύει ότι τα επίσημα API αποτελούν την ιδανική λύση όταν είναι διαθέσιμα, εξαιτίας της σταθερότητας και της τεχνικής τεκμηρίωσης που τα συνοδεύει. Αντιθέτως, το web scraping λειτουργεί ως αναγκαία συμπληρωματική τεχνική, ιδίως σε περιπτώσεις όπου η πληροφόρηση είναι δημόσια αλλά δεν παρέχεται σε κάποια προγραμματιστικά προσβάσιμη μορφή. Μέσα από την εμπειρία αυτή κατέστη σαφές ότι η ανάπτυξη μιας σύγχρονης εφαρμογής συχνά απαιτεί τον συνδυασμό και των δύο μεθοδολογιών, τα API για δομημένη και αξιόπιστη πρόσβαση, και το scraping για την κάλυψη κενών, πάντα με προσοχή στη διατήρηση της λειτουργικότητας και της συμμόρφωσης με το εκάστοτε νομικό πλαίσιο.

5.2 Προτάσεις για βελτίωση

Η παρούσα εφαρμογή αποτελεί μια πρωτότυπη υλοποίηση που συνδυάζει τεχνικές web scraping, RESTful APIs και νευρωνικά μοντέλα μηχανικής μάθησης για την ταξινόμηση και την αναζήτηση ταξινομικών πληροφοριών που αφορούν ευρεσιτεχνίες. Παρόλο που το σύστημα έχει ήδη επιτύχει ένα ικανοποιητικό επίπεδο λειτουργικότητας, υπάρχουν σαφή περιθώρια βελτίωσης που σχετίζονται με την αυτονομία της εφαρμογής και την απόδοση της υποδομής.

Μια ουσιαστική βελτίωση αφορά την ανάπτυξη μιας βάσης δεδομένων ταξινομικών συμβόλων, η οποία θα ενημερώνεται περιοδικά μέσω διαδικασιών crawling προς τις επίσημες υπηρεσίες (EPO, USPTO, WIPO, Espacenet), ώστε να διατηρούνται πάντοτε επικαιροποιημένες οι εγγραφές. Η υλοποίηση μιας τέτοιας υποδομής θα εξασφαλίσει πλήρη ανεξαρτησία από τη διαθεσιμότητα των

τρίτων υπηρεσιών και από πιθανές αλλαγές στο HTML DOM των ιστοσελίδων. Σε αυτή την κατεύθυνση η εφαρμογή θα προσφέρει απόλυτη επιτυχία σε κάθε ερώτημα και σημαντική αύξηση της αξιοπιστίας. Επιπλέον, η τοπική βάση θα επιτρέπει ταχύτερη απόκριση, πιο αποδοτική εκτέλεση στο batch mode και δυνατότητα περαιτέρω εμπλουτισμού με τεχνικές ευρετηρίασης και ανάλυσης όπως το Elasticsearch για full-text αναζητήσεις. Σε συνδυασμό πάντα με την αποθήκευση και τη διαχείριση των δεδομένων εντός της εφαρμογής, το σύστημα αποτελεί μια robust και επεκτάσιμη υλοποίηση που θα μειώσει την πολυπλοκότητα του scraping και θα ενισχύσει την κλιμακωσιμότητα του συστήματος στο μέλλον.

Η υφιστάμενη λειτουργία batch mode παρέχει ήδη τη δυνατότητα ταυτόχρονης επεξεργασίας πολλαπλών ερωτημάτων σε ικανοποιητικό χρόνο, ωστόσο δύναται να ενισχυθεί περαιτέρω μέσω στοχευμένων αναβαθμίσεων στην υποδομή. Συγκεκριμένα, η υιοθέτηση κατανεμημένων container clusters, όπως το Kubernetes, θα προσέφερε κλιμακούμενη διαχείριση πόρων και ευελιξία στην εκτέλεση. Η ενσωμάτωση in-memory βάσεων δεδομένων, όπως η Redis, θα μείωνε δραστικά τους χρόνους απόκρισης μέσω ταχύτερης προσωρινής αποθήκευσης και ανάκτησης αποτελεσμάτων. Παράλληλα, η εφαρμογή υποστηρίζει ήδη προηγούμενους μηχανισμούς παραλληλοποίησης τόσο σε επίπεδο CPU όσο και σε GPU. Συνεπώς για την πλήρης αξιοποίησή τους, προϋπόθεση αποτελεί η ύπαρξη κατάλληλου υλικού από τη μεριά του εξυπηρετητή. Με αυτόν τον τρόπο καθίσταται εφικτή η εκτέλεση πιο σύνθετων σεναρίων σε μικρότερο χρόνο, βελτιώνοντας την εμπειρία του ερευνητή και τη χρηστικότητα της εφαρμογής σε αναλύσεις μεγάλης κλίμακας.

5.3 Μελλοντικές επεκτάσεις

Η υλοποίηση της εφαρμογής αν και ικανοποιεί τους στόχους της παρούσας εργασίας για την ανάπτυξη μιας ευφυούς διαδικτυακής εφαρμογής αναζήτησης ταξινομικής πληροφορίας που αφορά ευρεσιτεχνίες αλλά και την επίλυση ενός κύριου προβλήματος αυτό της μαζικής αναζήτησης, εντούτοις αφήνει ανοιχτά αρκετά περιθώρια για μελλοντική εξέλιξη. Μια πρόταση θα αφορούσε την επέκταση της κάλυψης σε επιπλέον βάσεις δεδομένων και APIs όπως εκείνα του Ιαπωνικού γραφείου ευρεσιτεχνιών (JPO) ή του Κορεάτικου Γραφείου Πνευματικής Ιδιοκτησίας (KIPO). Σε αυτή την κατεύθυνση η εφαρμογή θα μπορούσε να ικανοποιήσει μεγαλύτερο ποσοστό ερευνητών προσφέροντάς τους πρόσβαση σε ένα ευρύτερο φάσμα δεδομένων.

Ένα ιδιαίτερα σημαντικό πεδίο μελλοντικής επέκτασης αφορά την οπτικοποίηση των δεδομένων που προκύπτουν από την ανάλυση και αναζήτηση των πατεντών. Η ύπαρξη μεγάλου όγκου αποτελεσμάτων, ιδίως σε batch αναζητήσεις, καθιστά δύσκολη την άμεση ερμηνεία τους όσο και να βελτιστοποιηθεί η παρουσίασή τους σε μορφή κειμένου. Η ενσωμάτωση μηχανισμών οπτικοποίησης θα μπορούσε να μετατρέψει τα αποτελέσματα από τις εκτελέσεις του συστήματος σε ευανάγνωστες και διαδραστικές αναπαραστάσεις, προσφέροντας στον χρήστη μια συνολική εικόνα. Πιο συγκεκριμένα, η συχνότητα εμφάνισης CPC/IPC κωδικών θα μπορούσε να αποδίδεται μέσω ραβδογραμμάτων ή κυκλικών διαγραμμάτων, επιτρέποντας την άμεση αναγνώριση των πιο σχετικών κατηγοριών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] World Intellectual Property Organization (WIPO), *Guide to the International Patent Classification (IPC), 2023 Edition*, Geneva, Switzerland, 2023. [Online]. Available: https://www.wipo.int/edocs/pubdocs/en/wipo-guide-ipc-2023-en-guide-to-the-international-patent-classification-2023.pdf (Πρόσβαση 28-5-2025)

[2] Eurostat, "Glossary: International patent classification (IPC)," *Statistics Explained*, European Commission, Jul. 2022. [Online]. Available: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Glossary:International_patent_classification_(IPC)

[3] J. Goodbody, *Patent Classification Through the Ages*, U.S. Patent and Trademark Office (USPTO). [Online]. Available: https://www.uspto.gov/sites/default/files/documents/Timeline.pdf (Πρόσβαση 28-5-2025)

[4] European Patent Office, *EPO Worldwide Bibliographic Data (DOCDB) – Bulk Data Sets*. [Online]. Available: https://www.epo.org/en/searching-for-patents/data/bulk-data-sets/docdb (Πρόσβαση 20-6-2025)

[5] TPR International, *Making the Switch from USPC to CPC: What Attorneys and Searchers Need to Know*, 12 October 2015. [Online]. Available: https://www.tprinternational.com/making-the-switch-from-uspc-to-cpc/ (Πρόσβαση 22-7-2025)

[6] World Intellectual Property Organization, *IPC Publication Help*, WIPO, [Online] Available: https://cdn.ipcpub.wipo.int/media/help/en/help.pdf (Πρόσβαση 23-7-2025)

[7] M. Shaghouzi, B. Teimourpour and M. A. Soltanshahi, "Fine-tuning BERT for Persian Patent Classification: A Dataset and Model Exploration," 2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP), Babol, Iran, Islamic Republic of, 2024, pp. 1-3, doi: 10.1109/AISP61396.2024.10475235.

[8] Y. Qiang, G. Sun and H. Liu, "PatentALL: Multi-label Patent Classification Using Adaptive Label Learning," 2024 IEEE 36th International Conference on Tools with Artificial Intelligence (ICTAI), Herndon, VA, USA, 2024, pp. 108-115, doi: 10.1109/ICTAI62512.2024.00024.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention Is All You Need,” arXiv preprint arXiv:1706.03762, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1706.03762.

[10] Anferico, “BERT for patents,” Hugging Face, [Online]. Available: https://huggingface.co/anferico/bert-for-patents

[11] M. Lupu, F. Piroi, J. List, L. Papariello, R. Alentorn, and M. Baycroft, *WPI Test Collection*, Version 1, Feb. 28, 2019. Zenodo. [Online]. Available: https://zenodo.org/records/1489994

[12] Python Software Foundation, *concurrent.futures — Launching parallel tasks*, Python 3.12. Official Documentation. [Online]. Available: https://docs.python.org/3/library/concurrent.futures.html (Πρόσβαση 28-7-2025)

[13] K. Mehta, *Different Methods of Multiprocessing in Python*, Medium, 2021. [Online]. Available: https://medium.com/@mehta.kavisha/different-methods-of-multiprocessing-in-python-70eb4009a990 (Πρόσβαση 1-8-2025)

[14] A. Kunwar, *Multiprocessing vs Multithreading with Use Case and Example*, Medium, 10 August 2024. [Online]. Available: https://ankushkunwar7777.medium.com/multiprocessing-vs-multithreading-with-use-case-and-example-must-known-concept-23dd0201d700 (Πρόσβαση 21-8-2025)

[15] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," Scientific American, vol. 284, no. 5, pp. 34–43, May 2001, [Online] Available: https://www-sop.inria.fr/acacia/cours/essi2006/Scientific%20American_%20Feature%20Article_%20The%20Semantic%20Web_%20May%202001.pdf (Πρόσβαση 27-8-2025)

[16] Interaction Design Foundation, Progressive Disclosure, [Online] Available: https://www.interaction-design.org/literature/topics/progressive-disclosure (Πρόσβαση 28-8-2025)

[17] Interaction Design Foundation, Visual Hierarchy, [Online] Available: https://www.interaction-design.org/literature/topics/visual-hierarchy (Πρόσβαση 28-8-2025)

[18] Fialkowski, B. and Schofield, D. (2024) Considering Color: Applying Psychology to Improve the Use of Color in Digital Interfaces. Art and Design Review, 12, 306-329. doi: 10.4236/adr.2024.124022.