ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«PROCESSING AND ENHANCING OF TEXTUAL DATA»

Του φοιτητή	**Επιβλέπων**
Παπώτης Χρήστος	**Σαλαμπάσης Μιχαήλ**
Αρ. Μητρώου: 512087	**Διδάσκων Καθηγητής**

Θεσσαλονίκη, Ιανουάριος 2024

Τίτλος Δ.Ε.: Processing and enhancing of textual data.
Ονοματεπώνυμο φοιτητή: Παπώτης Χρήστος
Ονοματεπώνυμο εισηγητή: Σαλαμπάσης Μιχαήλ
Ημερομηνία ανάληψης Δ.Ε.: 16/10/2023
Ημερομηνία περάτωσης Δ.Ε.: 10/01/2024

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 πτυχιακής 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.

Πρόλογος

Η διαδικασία οργάνωσης, επιμέλειας και ενίσχυσης των δεδομένων μιας βάσης παρουσιάζει μοναδικές προκλήσεις ιδιαίτερα όταν έχουμε να κάνουμε με ελλιπείς και αδόμητες πληροφορίες.

Στόχος της παρούσας πτυχιακής εργασίας είναι η αντιμετώπιση αυτών των προκλήσεων για την βάση δεδομένων του Art Domain αλλά και η αυτόματη συστηματική οργάνωση και ο εμπλουτισμός των δεδομένων χωρίς την αναγκαστική παρουσία χρήστη. Η εργασία υπογραμμίζει τη σημασία της αξιοποίησης σύγχρονων τεχνολογιών για να διασφαλιστεί ότι τα δεδομένα πληρούν υψηλά πρότυπα ακρίβειας αλλά και είναι προσαρμόσιμα σε εφαρμογές ανάλυσης και μηχανικής μάθησης.

Περίληψη

Η παρούσα πτυχιακή εργασία έχει ως στόχο τη δημιουργία μιας ολοκληρωμένης και δομημένης βάσης δεδομένων για την εφαρμογή Theatrical Plays. Η βάση θα παρέχεις πληροφορίες για θεατρικές παραγωγές όπως συντελεστές, ρόλους, συνεργάτες, εκδηλώσεις και διοργανωτές.

Η υλοποίηση βασίζεται σε ένα API αναπτυγμένο σε JAVA(Spring Boot framework) το οποίο παρέχει endpoints για την επιμέλεια όλων των πινάκων της βάσης αλλά και ένα frontend admin panel το οποίο μπορεί να το χρησιμοποιεί ένας χρήστης για να στοχεύσει σε συγκεκριμένες στήλες, σειρές ή ολόκληρους πίνακες της βάσης. Παράλληλα αξιοποιούνται βιβλιοθήκες όπως το Stanford NLP για την ανάλυση και επεξεργασία κειμένου, επιτρέποντας εργασίες όπως η αναγνώριση οντοτήτων, η κανονικοποίηση κειμένου και η εξαγωγή σχέσεων.

Για την συμπλήρωση κενών δεδομένων ενσωματώθηκαν Μεγάλα Γλωσσικά Μοντέλα(LLMs) ώστε να ανακτηθούν δεδομένα με προτάσεις όπου παρείχαν πληροφορίες ικανές ώστε τα γλωσσικά μοντέλα να μας δώσουν τα σωστά δεδομένα που χρειαζόμαστε.

Βασικές προκλήσεις που αντιμετωπίστηκαν περιλαμβάνουν την σωστή επικύρωση δεδομένων και την διασφάλιση συνέπειας μεταξύ των πινάκων.

Λέξεις – Κλειδιά: Προγραμματισμός, API , Βάση Δεδομένων, LLMs.

Processing And Enhancing of Textual Data

PAPOTIS CHRISTOS

Abstract

This thesis aims to create a comprehensive and structured database for the Theatrical Plays application. The database will provide information about theatrical productions such as actors, roles, collaborators, events and organizers.

The implementation is based on an API developed in JAVA (Spring Boot framework) which provides endpoints for editing all the database tables and also a frontend admin panel which can be used by a user to target specific columns, rows or whole base tables. In parallel, libraries such as Stanford NLP are leveraged for text analysis and processing, enabling tasks such as entity recognition and text normalization.

To fill data gaps, Large Language Models (LLMs) were incorporated to retrieve data with sentences where they provided enough information for the language models to give us the right data we need. Key challenges faced include proper data validation and ensuring consistency between tables.

Key – Words: Programming, API, Database, LLMs.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, τον κύριο Σαλαμπάση Μιχαήλ, για τη πολύτιμη βοήθεια που μού παρείχε και την εμπιστοσύνη που μού έδειξε, αναθέτοντάς μου την εν λόγω πτυχιακή εργασία.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

Η παρούσα πτυχιακή εργασία εστιάζει στην επιμέλεια και την οργάνωση των δεδομένων της βάσης που αφορά το Art Domain και συγκεκριμένα τα Theatrical Plays. Η εργασία στοχεύει στην υλοποίηση μιας δομημένης και ολοκληρωμένης βάσης δεδομένων που περιλαμβάνει πληροφορίες για διοργανωτές, χώρους, ρόλου, εκδηλώσεις και παραστάσεις.

Η υλοποίηση έγινε με ένα backend service αναπτυγμένο με Spring Boot(Java), το οποίο προσφέρει endpoints για την διαχείριση όλων των πινάκων της βάσης και είναι συνδεδεμένο με ένα frontend το οποίο ανατύχθηκε με Next.js προσφέροντας μια εύχρηστη διεπαφή για την διαχείριση των δεδομένων.

1.1 Αντικείμενο Εργασίας

Η παρούσα πτυχιακή εργασία έχει ως αντικείμενο της την δημιουργία ενός ολοκληρωμένου συστήματος για την διαχείριση δεδομένων και στοχεύει στην βελτίωση της ποιότητάς των. Το σύστημα θα παρέχει δυνατότητες όπως:

1. Προσθήκη, επεξεργασία και διαγραφή δεδομένων που αφορούν παραστάσεις, συντελεστές, ρόλους, εκδηλώσεις, χώρους και διοργανωτές.

2. Ανάλυση συσχετίσεων μεταξύ των οντοτήτων (π.χ., συντελεστές και ρόλοι).

3. Εύχρηστο γραφικό περιβάλλον για την επεξεργασία των δεδομένων.

4. Συμπλήρωση ελλιπών δεδομένων.

5. Αυτόματη επικύρωση και διατήρηση συνέπειας μεταξύ των δεδομένων.

6. Συμπλήρωση ελλιπών δεδομένων χρησιμοποιώντας LLMs.

1.2 Στόχος Εργασίας

Ο στόχος της εργασίας είναι η ανάπτυξη ενός ολοκληρωμένου συστήματος end to end επιμέλειας και οργάνωσης των δεδομένων στον τομέα της τέχνης με επίκεντρο τις θεατρικές παραστάσεις. Το σύστημα αυτό σκοπεύει στην βελτίωση της ποιότητας αλλά και της πληρότητας των δεδομένων που παρέχει η βάση χρησιμοποιώντας σύγχρονες τεχνολογίες όπως Natural Language Processing (NLP) και Μεγάλα Γλωσσικά Μοντέλα(LLMs). Επιπλέον, στοχεύει στη δημιουργία μιας δυναμικής και εύχρηστης πλατφόρμας που επιτρέπει την αποθήκευση, επεξεργασία, ανάλυση και παρουσίαση δεδομένων.

1.3 Δομή Εργασίας

Η πτυχιακή εργασία αποτελείται από έξι(6) κύρια κεφάλαια όπου το καθένα αναλύει και μια διαφορετίκη πτυχή του έργου. Τα κεφάλαια είναι τα εξής παρακάτω:

Κεφάλαιο 1ο: Εισαγωγή
Παρουσιάζεται ο σκοπός και ο στόχος της εργασίας καθώς και η δομή που χρησιμοποιήθηκε.

Κεφάλαιο 2ο: Data Curation
Ανάλυση της θεωρίας και πρακτικων επιμέλειας δεδομένων

Κεφάλαιο 3ο: Ανάπτυξη backend με Spring Boot
Ανάλυση σχεδιασμού και υλοποίησης του συστήματος αλλά και των αλγορίθμων.

Κεφάλαιο 4ο: Χρήση Μεγάλων Γλωσσικών Μοντέλων και NLP
Ανάλυση χρήσης βιβλιοθηκών NLP αλλά και χρήσης LLm για τον εμπλουτισμό των δεδομένων.

Κεφάλαιο 5ο: Αναπτυξη frontend με Next.js
Παρουσίαση ανάπτυξης του frontend αλλά και της διασύνδεσης με το backend.

Κεφάλαιο 6ο: Αποτελέσματα και Συμπεράσματα
Σύνοψη των αποτελεσμάτων του συστήματος, αξιολόγηση της απόδοσης του αλλά και προτάσεις για βελτιώσεις.

Κεφάλαιο 2ο:Data curation(επιμέλεια δεδομένων)

2.1 Εισαγωγή στο data curation

Η επιμέλεια δεδομένων είναι η οργάνωση και η ενοποίηση δεδομένων που συλλέγονται από διάφορες πηγές. Περιλαμβάνει σχολιασμό, δημοσίευση και παρουσίαση των δεδομένων, έτσι ώστε η αξία των δεδομένων να διατηρείται με την πάροδο του χρόνου και τα δεδομένα να παραμένουν διαθέσιμα για επαναχρησιμοποίηση και διατήρηση.

Η κατανόηση και η ανάλυση μεγάλων δεδομένων αναγνωρίζεται σταθερά ως ισχυρή και στρατηγική προτεραιότητα. Για βαθύτερη ερμηνεία και καλύτερη ευφυΐα με μεγάλα δεδομένα, είναι σημαντικό να μετατραπούν τα ανεπεξέργαστα δεδομένα (μη δομημένες, ημιδομημένες και δομημένες πηγές δεδομένων, π.χ. σύνολα δεδομένων κειμένου, βίντεο, εικόνων) σε επιμελημένα δεδομένα: δεδομένα και γνώσεις που διατηρούνται και δημιουργούνται διαθέσιμα για χρήση από τελικούς χρήστες και εφαρμογές όπως στην δική μας περίπτωση. Συγκεκριμένα, η επιμέλεια δεδομένων λειτουργεί ως κόλλα μεταξύ των ακατέργαστων δεδομένων και των αναλυτικών στοιχείων, παρέχοντας ένα επίπεδο αφαίρεσης(abstraction layer) που απαλλάσσει τους χρήστες από χρονοβόρες, κουραστικές και επιρρεπείς σε σφάλματα εργασίες επιμέλειας.

Ο απώτερος στόχος της επιμέλειας δεδομένων είναι να μειωθεί ο χρόνος από τα δεδομένα έως τις πληροφορίες. Με τον αυξανόμενο όγκο δεδομένων στους οργανισμούς σήμερα, η επιμέλεια δεδομένων γίνεται απαραίτητη.

2.1.1 Ιστορική Αναδρομή του Data Curation

Ο χρήστης, και όχι η ίδια η βάση δεδομένων, συνήθως ξεκινά την επεξεργασία δεδομένων και διατηρεί μεταδεδομένα. Σύμφωνα με το Graduate School of Library and Information Science του University of Illinois, "Η επιμέλεια δεδομένων είναι η ενεργός και συνεχής διαχείριση δεδομένων μέσω του κύκλου ζωής του ενδιαφέροντος και της χρησιμότητας για την υποτροφία, την επιστήμη και την εκπαίδευση· οι δραστηριότητες επιμέλειας επιτρέπουν την ανακάλυψη και ανάκτηση δεδομένων, διατηρούν την ποιότητα, προσθέτουν αξία και παρέχουν επαναχρησιμοποίηση με την πάροδο του χρόνου."

Η ροή εργασιών επιμέλειας δεδομένων διαφέρει από τη διαχείριση ποιότητας δεδομένων, την προστασία δεδομένων, τη διαχείριση του κύκλου ζωής και τη μετακίνηση δεδομένων.

Τα δεδομένα απογραφής ήταν διαθέσιμα σε μορφή πινακοποιημένης κάρτας διάτρησης(tabulated punch card) από τις αρχές του 20ου αιώνα και ήταν ηλεκτρονικά από τη δεκαετία του 1960. Ο ιστότοπος της Διαπανεπιστημιακής Κοινοπραξίας για Πολιτική και Κοινωνική Έρευνα (ICPSR) σηματοδοτεί το 1962 ως την ημερομηνία του πρώτου τους Αρχείου Δεδομένων Έρευνας. Το υπόβαθρο για τις βιβλιοθήκες δεδομένων εμφανίστηκε σε ένα τεύχος του 1982 του περιοδικού του Ιλινόις, Library Trends.

Η εμφάνιση των υπολογιστών και των βάσεων δεδομένων στα μέσα του 20ού αιώνα μετέβαλε ριζικά την επιμέλεια δεδομένων. Στη δεκαετία του 1960, αναπτύχθηκαν οι πρώτες σχεσιακές βάσεις δεδομένων, όπως το IBM System R και το Oracle Database, που προσέφεραν μεθόδους αποθήκευσης και ανάκτησης δεδομένων. Με την ανάπτυξη του Big Data και του Cloud Computing, η επιμέλεια δεδομένων επεκτάθηκε πέρα από τη βασική αρχειοθέτηση και εστιάστηκε στη συλλογή, ανάλυση και επεξεργασία μεγάλων ποσοτήτων δεδομένων. Η χρήση μηχανικής μάθησης προσέφερε νέες δυνατότητες για την αυτοματοποίηση διαδικασιών, όπως η αναγνώριση μοτίβων και η συμπλήρωση ελλιπών πληροφοριών. Σήμερα, η επιμέλεια δεδομένων συνδυάζει τεχνικές Natural Language Processing (NLP), Μεγάλων Γλωσσικών Μοντέλων (LLMs) και συστημάτων διαχείρισης περιεχομένου. Εφαρμόζεται σε πολλούς τομείς, από την επιστημονική έρευνα και τη βιοϊατρική έως την πολιτιστική κληρονομιά και την τεχνητή νοημοσύνη.

2.1.2 Διαφορά μεταξύ ακατέργαστων δεδομένων και επιμελημένων δεδομένων

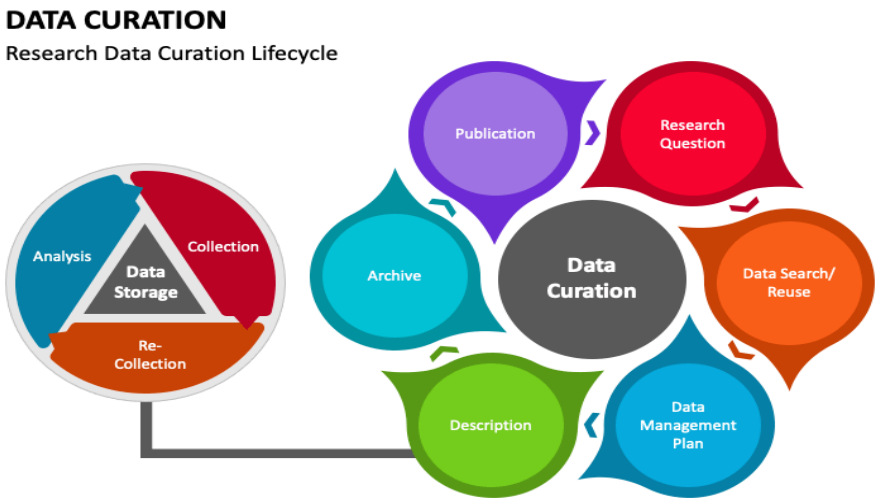
Τα ακατέργαστα δεδομένα είναι οι μη επεξεργασμένες πληροφορίες που συλλέγονται από διάφορες πηγές. Συχνά είναι ελλιπές και δύσκολο να ερμηνευθούν. Για παράδειγμα, τα δεδομένα που συλλέγονται από web scraping όπως στην περίπτωση της βασης δεδομέων των Theatrical Plays ενδέχεται να περιέχουν διπλές καταχωρίσεις, ασυνεπείς μορφές και άσχετες πληροφορίες. Ενώ τα ακατέργαστα δεδομένα έχουν πιθανή αξία, απαιτούν σημαντική επεξεργασία για να είναι χρήσιμα. Τα επιμελημένα δεδομένα, από την άλλη πλευρά, έχουν υποστεί αυστηρή επεξεργασία για να διασφαλιστεί ότι είναι καθαρά, ακριβή και σχετικά. Πρόκειται για κάτι περισσότερο από απλή οργάνωση πληροφοριών: η διαδικασία επιμέλειας δεδομένων περιλαμβάνει επίσης τη βελτίωση του συνόλου δεδομένων ώστε να ευθυγραμμιστεί με συγκεκριμένους στόχους. Για παράδειγμα, τα επιμελημένα σχόλια χρηστών θα κατηγοριοποιηθούν, θα συνοψιστούν και θα σχολιαστούν για να τονιστούν βασικές πληροφορίες, τάσεις και συναισθήματα. Συγκεκριμένα, η διαδικασία επιμέλειας δεδομένων συνήθως περιλαμβάνει τα ακόλουθα βήματα:

Καθαρισμός:Αφαίρεση διπλότυπων, διόρθωση σφαλμάτων και τυποποίηση μορφών.
Επικύρωση: Διασφάλιση της ακρίβειας και αξιοπιστίας των δεδομένων.
Σχολιασμός: Προσθήκη πλαισίου στα δεδομένα, διευκολύνοντας την κατανόηση και τη χρήση τους.

Η κύρια διαφορά μεταξύ της επιμέλειας δεδομένων και της διαχείρισης δεδομένων έγκειται στο πεδίο εφαρμογής τους. Η διαχείριση δεδομένων εστιάζει στον συνολικό κύκλο ζωής των δεδομένων, συμπεριλαμβανομένης της αποθήκευσης, της ασφάλειας και της προσβασιμότητας. Η επιμέλεια δεδομένων, ωστόσο, είναι μια εξειδικευμένη διαδικασία στο πλαίσιο της διαχείρισης δεδομένων που

στοχεύει στη βελτίωση της ποιότητας και της χρησιμότητας των δεδομένων.

2.2 Κύκλος διαδικασιών data curation



Σχήμα 2.2: Σχήμα διαδικασίας επιμέλειας δεδομένων [12].

Η διαδικασία επιμέλειας δεδομένων περιλαμβάνει διάφορα βήματα για να διασφαλιστεί ότι τα δεδομένα που συλλέγονται είναι οργανωμένα, δομημένα και εύκολα προσβάσιμα. Το πρώτο βήμα στη διαδικασία επιμέλειας δεδομένων είναι ο εντοπισμός των πηγών δεδομένων και η αξιολόγηση της ποιότητάς τους. Αυτό περιλαμβάνει την αξιολόγηση της αξιοπιστίας, της ακρίβειας, της πληρότητας και της συνάφειας των διαθέσιμων συνόλων δεδομένων.

Αφού προσδιορίσουμε τις πηγές μας και αξιολογήσουμε την ποιότητά τους εξαγάγουμε τις σχετικές πληροφορίες από αυτές. Αυτό περιλαμβάνει τον καθαρισμό τυχόν σφαλμάτων ή ασυνεπειών στο σύνολο δεδομένων, αφαιρώντας διπλότυπα ή τιμές που λείπουν.

Αφού καθαρίσουμε το σύνολο δεδομένων μας, πρέπει να το μετατρέψουμε σε μια τυποποιημένη μορφή που μπορεί εύκολα να αναζητηθεί και να αναλυθεί. Αυτό μπορεί να περιλαμβάνει τυποποίηση συμβάσεων ονομασίας για μεταβλητές ή πεδία στο σύνολο δεδομένων μας.

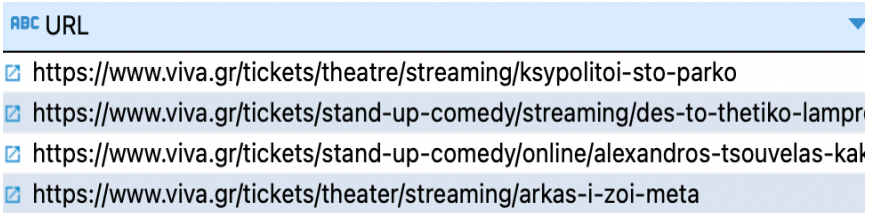
Μόλις τα δεδομένα μας καθαριστούν και μετατραπούν σε τυποποιημένη μορφή, μπορούμε στη συνέχεια να τα αποθηκεύσουμε σε ένα ασφαλές repository(βάση δεδομένων, csv ή excel αρχεία) όπου θα είναι εύκολα προσβάσιμα από αναλυτές, προγραμματιστές ή ερευνητές. Η σωστή αποθήκευση των επιμελημένων δεδομένων μας διασφαλίζει ότι η μελλοντική ανάλυση θα παραμείνει συνεπής με τα προηγούμενα ευρήματα. Συνολικά, η διαδικασία επιμέλειας πολύτιμων συνόλων δεδομένων απαιτεί χρόνο, αλλά παρέχει μακροπρόθεσμα οφέλη όταν γίνεται σωστά. Επιτρέπει στους οργανισμούς να λαμβάνουν πιο ενημερωμένες αποφάσεις με βάση πληροφορίες υψηλής ποιότητας, αποφεύγοντας κοινές παγίδες που σχετίζονται με σύνολα δεδομένων κακής ποιότητας, όπως ανακριβή συμπεράσματα ή χαμένες ευκαιρίες.

2.2.1 Συλλογή των δεδομένων

Όταν πρόκειται για την ακρίβεια των μεγάλων δεδομένων, δεν μπορεί να πραγματοποιηθεί χωρίς προγραμματισμό. Η ύπαρξη ενός σαφούς οράματος και στρατηγικής δεδομένων διασφαλίζει ότι τα δεδομένα μας ευθυγραμμίζονται με τους στόχους του έργου μας και το ίδιο το μοντέλο. Γι' αυτό τον λόγο πριν την συλλογή των δεδομένων έχουμε δημιουργήσει του πίνακες στην βάση δεδομένων μας που θα χρειαστούμε να αποθυκεύσουμε τα δεδομένα μας με την σωστή μοντελοποίηση.

Σχήμα 2.2.1: Οι πίνακες που θα χρειαστούμε για τα δεδομένα [13].

Η διαδικασία συλλογής δεδομένων μπορεί να γίνει απο διάφορες πηγές όπως από αρεία παραστάσεων, άρθρα, ιστοσελίδες, άλλες βάσεις δεδομένων, στην δική μας περίπτωση η συλλογή έγινε χρησιμοποιώντας τεχνικές web scraping αντλώντας πληροφορίες απο δημοφιλείς ιστοσελίδες όπως το www.viva.gr/tickets το οποίο παρέχει πληθώρα δεδομένων για τις περισσότερες θεατρικές παραστάσεις.



Σχήμα 2.2.1: Παραδείγματα από url στην βάση μας που χρησιμοποιήθηκαν [14].

Τα βήματα για την συλλογή δεδομένων που χρησιμοποιήθηκαν είναι:

1. Ενδελεχής αναγνώριση των πηγών δεδομένων ώστε να είματε σίγουροι ότι είναι σχετικές με το έργο μας.
2. Εντοπισμός δεδομένων από ιστοσελίδες μέσω web scraping.

3. Αποθήκευση δεδομένων σε κατάλληλη μορφή για περαιτέρω επεξεργασία.

2.2.2 Καθαρισμός δεδομένων(data cleaning)

Μετά την ολοκλήρωση της συλλογής δεδομένων πρέπει να ελέγξουμε όλα τα δεδομένα για ελλείψεις και ασυνέπειες. Για να εξαλειφθούν αυτά τα προβλήματα και να διασφαλιστεί η αξιοπιστία των δεδομένων προχωρήσουμε σε καθαρισμό δεδομένων(data cleaning).

Ο καθαρισμός δεδομένων ουσιαστικά είναι η διαδικασία διόρθωσης ή αφαίρεσης εσφαλμένων, κατεστραμμένων, εσφαλμένα μορφοποιημένων, διπλότυπων ή ελλιπών δεδομένων σε ένα σύνολο δεδομένων. Όταν συνδυάζονται πολλαπλές πηγές δεδομένων, υπάρχουν πολλές ευκαιρίες για αντιγραφή ή εσφαλμένη επισήμανση δεδομένων. Εάν τα δεδομένα είναι λανθασμένα, τα αποτελέσματα, οι αλγόριθμοι και τα δεδομένα είναι αναξιόπιστα, παρόλο που μπορεί να φαίνονται σωστά. Δεν υπάρχει ένας απόλυτος τρόπος για να συνταγογραφηθούν τα ακριβή βήματα στη διαδικασία καθαρισμού δεδομένων, επειδή οι διαδικασίες θα διαφέρουν από σύνολο δεδομένων σε σύνολο δεδομένων γι' αυτό και εμείς επικεντρωνόμαστε μόνο στο σύνολο δεδομένων των Theatrical Plays. Τα βασικά βήματα που ακολουθουθήσαμε για τον καθαρισμό των δεδομένων της βάσης μας είναι οι παρακάτω:

<u>1. Αφαίρεση Διπλότυπων</u>

Το πρώτο βήμα στο data cleaning, είναι να αφαιρέσουμε δεδομένα που εντοπίζονται παραπάνω από μία φορά. Το να εμφανίζονται δεδομένα δύο ή περισσότερες φορές, είναι κάτι σύνηθες και φυσιολογικό, αφού δεν έχουμε τον πλήρη έλεγχο των πηγών από όπου προέρχονται.

organizer | Enter a SQL expression to filter results (use Ctrl+Space)

Grid

	id	name	address
290	654	BLOOM ΑΣΤΙΚΗ ΜΗ ΚΕΡΔΟΣΚΟΠΙΚΗ ΕΤΑΙΡΕΙΑ	Φωκίωνος Νέγρη 61
291	655	ΕΛΛΗΝΙΚΟ ΦΕΣΤΙΒΑΛ ΑΕ	Αλκμήνης 5
292	656	ΛΥΚΟΦΩΣΓΛυκιαρδόπουλος ΣΙΑ ΕΕ	Βουλής 38

Σχήμα 2.2.2: Δεδομένα πίνακα organizer [15].

organizer | Enter a SQL expression to filter results (use Ctrl+Space)

Grid

	id	name	address
393	759	ΘΕΑΜΑ ΑΚΡΟΠΟΛ ΔΕΜΕΣΤΙΧΑΣ ΠΙΕΡΡΟΣ ΚΑΙ ΣΙΑ ΕΙ	Ιπποκράτους 9-11
394	760	ΕΛΛΗΝΙΚΟ ΦΕΣΤΙΒΑΛ ΑΕ	Αλκμήνης 5

Σχήμα 2.2.2: Διπλά δεδομένα πίνακα organizer [16].

Στις παραπάνω εικόνες [15][16] παραθέτουμε διπλότυπα δεδομένα στον πίνακα organizer στα οποία θα πρέπει να αφαιρεθεί ένα από τα δύο.

<u>2. Διόρθωση Λαθών</u>

Το επόμενο βήμα είναι να διορθώσουμε δεδομένα, που έχουν καταχωρηθεί λανθασμένα.Αυτό σημαίνει ότι μπορεί να έχουν καταχωρηθεί σε μορφή η οποία δεν είναι σύμφωνη με τη μορφή που θα έπρεπε να έχουν.

Για παράδειγμα, αν εντός ενός αριθμού τηλεφώνου προστεθεί κατά λάθος κάποιο γράμμα, τότε η συγκεκριμένη πληροφορία δεν θα αναγνωρίζεται.

venue | Enter a SQL expression to filter results (use Ctrl+Space)

Grid

	id	title	address
588	918	Κινηματοθέατρο Όνειρο	Αγ. Ιωάννης Ρέντης,
589	919	Εύπολις	Art
590	920	Πολυχώρος Αλεξάνδρεια	Αθήνα, Αττική

Σχήμα 2.2.2: Δεδομένα με λάθος διεύθυνση στον πίνακα venue [17].

Στην παραπάνω εικόνα διακρίνουμε ότι στην στήλη address η διεύθυνση είναι λάθος στην σειρα με id 589 και θα έπρεπε να είναι Λεωφ. Λαυρίου 158Α, Παιανία αντί για Art.

3. Εντοπισμός και αφαίρεση ανεπιθύμητων σημείων

Κάποια δεδομένα παρόλο που έχουν καταχωρηθεί εκεί που πρέπει και όπως πρέπει, μπορεί αυτό που εκφράζουν να μην είναι λογικό. Για παράδειγμα, συνεχίζοντας με το παράδειγμα των κινητών τηλεφώνων, μπορεί η καταχώρησή μας να μοιάζει με κινητό και να έχει όσα ψηφία χρειάζεται, αλλά σε περίπτωση τηλεφώνων Ελλάδας να ξεκινάει με "30" αντί για "+30" και το τελικό αποτέλεσματα να μπερδεύει τον χρήστη.

Αυτό αποδεικνύει ότι η καταχώρηση αυτή ίσως να είναι λανθασμένη και χρειάζεται έλεγχο και πιθανή αφαίρεση.

town	postcode	phone
ΑΘΗΝΑ	11851	306943224305
Αθήνα	11744	2155451110
ΑΘΗΝΑ	17124	6947239091
Athens	11254	306984869891
ΑΘΗΝΑ	10671	2103611206
ATHENS ILIOUP	16346	+30 6970012618
Αθήνα	11741	6974673798

Σχήμα 2.2.2: Δεδομένα με λάθος τηλέφωνο στον πίνακα organizer [18].

Τα δεδομένα μας θα πρέπει να είναι σταθερά σε όλο το εύρως του πίνακα και σε περιπτώσεις όπως αύτη της εικόνας [18] θα πρέπει να αποφασίσουμε ότι αν θέλουμε να διατηρήσουμε την μορφή του τηλεφώνου με +30, τότε και τα υπόλοιπα δεδομένα που αφορούν τηλέφωνα θα πρέπει να ακολουθούν την ίδια μορφή.

4. Επίλυση Ελλιπών Δεδομένων

Κάποιες φορές είναι λογικό να υπάρξουν περιπτώσεις όπου λείπουν δεδομένα. Σε αυτές τις περιπτώσεις μπορούμε είτε να συμπληρώσουμε εμείς τα δεδομένα, με βάση την εικόνα που έχουμε από το σύνολο όλων των υπόλοιπων καταχωρήσεων, είτε να αλλάξουμε τον τρόπο με τον οποίο ο αλγόριθμος διαχειρίζεται τα δεδομένα, ώστε ακόμη και αν δεν υπάρχουν να μπορεί να λειτουργήσει.

Για παράδειγμα, αν σε αριθμούς, σε κάποιες περιπτώσεις που λείπουν δεδομένα θα μπορούσαν να αντικατασταθούν από το σύμβολο "-" ή από την λέξη "άγνωστο".

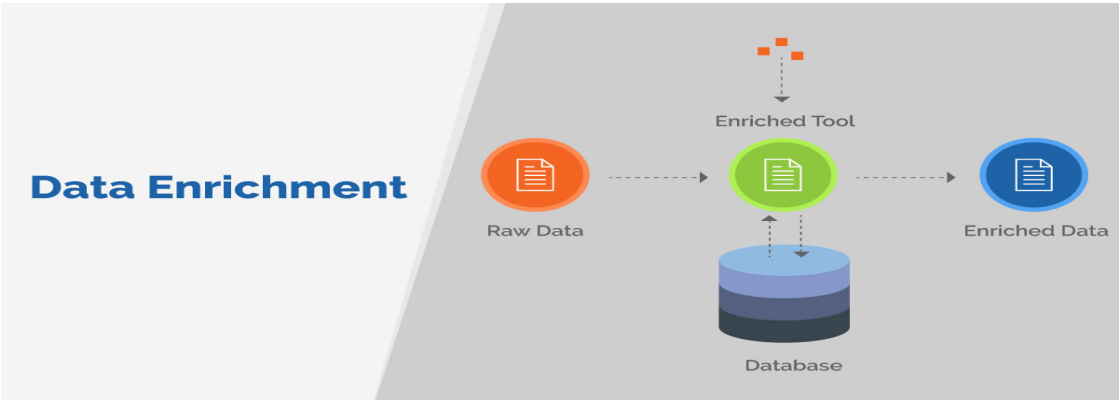
```java
// Validate and clean other fields
if (organizer.getEmail() != null && !validateEmail(organizer.getEmail())) {
    organizer.setEmail(email:"unknown");
}
if (organizer.getPhone() != null && !validatePhoneNumber(organizer.getPhone())) {
    organizer.setPhone(phone:"unknown");
}
```

Σχήμα 2.2.2: Java κώδικας για default value σε λάθος δεδομένα [19].

Από τα παραπάνω παραδείγματα μπορούμε να καταλήξουμε στο συμπέρασματα ότι το data cleaning μπορεί να προσφέρει πολλά οφέλη όπως ότι διορθώνει όσα πιθανά λάθη υπάρχουν και δίνει ένα πιο αξιόπιστο αποτέλεσμα αλλά και αφαιρεί επιπρόσθετη δουλειά που πιθανόν να χρειαζόταν να κάνουν οι administrators της βάσης δεδομένων.

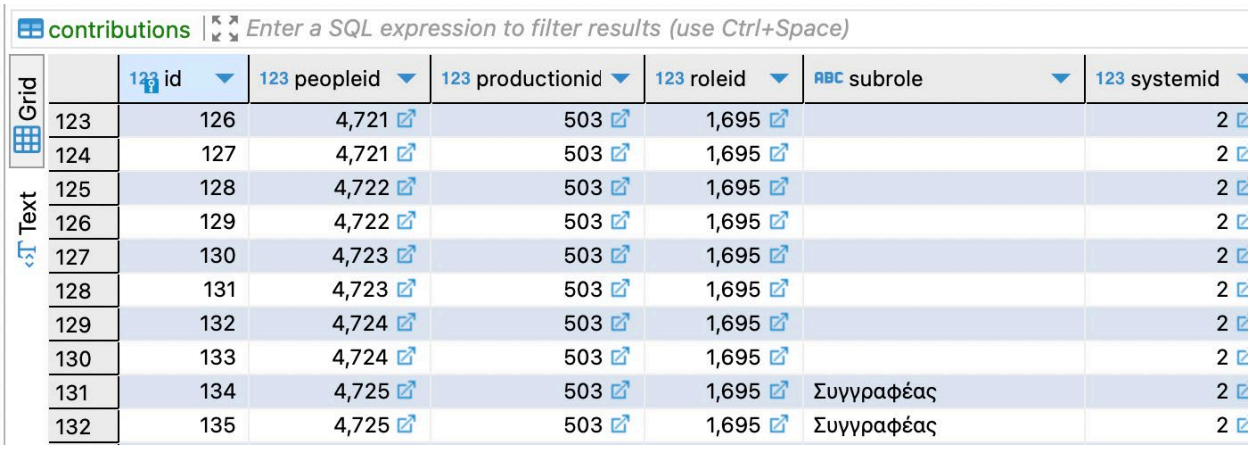
2.2.3 Εμπλουτισμός δεδομένων(Data enrichment)

Ο εμπλουτισμός δεδομένων είναι η διαδικασία αύξησης των υπάρχων δεδομένων με πρόσθετες λεπτομέρειες από διάφορες πηγές, μετατρέποντας τα ακατέργαστα δεδομένα σε ένα ολοκληρωμένα για καλύτερη λήψη αποφάσεων, ανάλυση και χρήση.



Σχήμα 2.2.3: Διάγραμμα εμπλουτισμού δεδομένων [20].

Στην βάση δεδομένων για τις θεατρικές παραστάσεις είχαμε πολλά δεδομένα τα οποία ήταν συχνά ελλιπείς, για παράδειγμα στην παρκάτω εικόνα παραθέτουμε δεδομένα από τον πίνακα contribution στον όποιο δεν υπάρχουν αναφορές για τους υπορόλους(subroles).



contributions | Enter a SQL expression to filter results (use Ctrl+Space)

Grid | Text

	id	peopleid	productionid	roleid	subrole	systemid
123	126	4,721	503	1,695		2
124	127	4,721	503	1,695		2
125	128	4,722	503	1,695		2
126	129	4,722	503	1,695		2
127	130	4,723	503	1,695		2
128	131	4,723	503	1,695		2
129	132	4,724	503	1,695		2
130	133	4,724	503	1,695		2
131	134	4,725	503	1,695	Συγγραφέας	2
132	135	4,725	503	1,695	Συγγραφέας	2

Σχήμα 2.2.3: Ελλιπείς δεδομένα στον πίνακα contributions [21].

Οι βασικά στοιχεία του εμπλουτισμού δεδομένων περιλαμβάνουν:

1. Επαλήθευση

Διασφάλιση ότι τα δεδομένα είναι ενημερωμένα και ακριβή.

2. Συμπλήρωση

Προσθήκη δεδομένων που λείπουν.

3. Ενσωμάτωση

Συγχώνευση δεδομένων από διάφορες πηγές για τη δημιουργία ενός πιο ολοκληρωμένου συνόλου δεδομένων.

2.2.3.1 Τεχνικές εμπλουτισμού δεδομένων

Υπάρχουν 6 κοινές τεχνικές που εμπλέκονται στον εμπλουτισμό δεδομένων:

1.Προσάρτηση δεδομένων(appending data)

Προσθέτοντας δεδομένα στο σύνολο της βάσης, συγκεντρώνουμε πολλές πηγές δεδομένων για να δημιουργήσουμε ένα πιο ακριβές και ολοκληρωμένο σύνολο από αυτό που παράγεται από οποιαδήποτε πηγή δεδομένων. Για παράδειγμα, η εξαγωγή δεδομένων θεατρικών παραστάσεων από βάσεις δεδομένων πολιτιστικής κληρονομιάς, όπως το **Europeana**(https://www.europeana.eu/el) και η συγκέντρωση αυτών θα μας δώσει μια καλύτερη και συνολική εικόνα για το πως να τα παρουσιάσουμε στους χρήστες από οποιοδήποτε μεμονωμένο σύστημα web scraping στο οποίο βασιζόμαστε.

2. Τμηματοποίηση δεδομένων(data segmentation)

Η τμηματοποίηση δεδομένων είναι μια διαδικασία με την οποία διαιρείτε ένα αντικείμενο δεδομένων(data object) όπως contributor, role, venue σε ομάδες με βάση ένα κοινό σύνολο προκαθορισμένων μεταβλητών. Αυτή η τμηματοποίηση χρησιμοποιείται στη συνέχεια ως τρόπος καλύτερης κατηγοριοποίησης και περιγραφής της οντότητας.

3. Παράγωγα χαρακτηριστηκά(derived attributes)

Τα παράγωγα χαρακτηριστικά είναι πεδία που δεν αποθηκεύονται στο αρχικό σύνολο δεδομένων, αλλά μπορούν να προκύψουν από ένα ή περισσότερα πεδία. Για παράδειγμα, η "ηλικία" αποθηκεύεται πολύ σπάνια, αλλά μπορούμε να την εξαγάγετε με βάση το πεδίο "ημερομηνία γέννησης". Τα παράγωγα χαρακτηριστικά είναι πολύ χρήσιμα επειδή συχνά περιέχουν λογική που χρησιμοποιείται επανειλημμένα για ανάλυση. Για παράδειγμα στον πίνακα persons της βάσης δεδομένων μας για τα Theatrical Plays έχουμε το πεδίο birthdate(ημερομηνία γέννησης) το οποίο μπορούμε να το χρησιμοποιήσουμε αν θέλουμε να παρέχουμ την ηλικία του κάθε ατόμου από τον πίνακα persons στους χρήστες.

4. Χειρισμός δεδομένων(data manipulation)

Ο χειρισμός δεδομένων είναι η διαδικασία αντικατάστασης τιμών για δεδομένα που λείπουν ή είναι ασυνεπ ή μέσα σε πεδία. Αντί να αντιμετωπίζεται η τιμή που λείπει ως μηδέν, κάτι που θα παραμορφώνει τα αποτελέσματα, η εκτιμώμενη τιμή βοηθά στη διευκόλυνση μιας πιο ακριβούς ανάλυσης των δεδομένων σας.

5. Εξαγωγή οντοτήτων(entity extraction)

Η εξαγωγή οντοτήτων είναι η διαδικασία λήψης μη δομημένων δεδομένων ή ημιδομημένων δεδομένων και εξαγωγής σημαντικών δομημένων δεδομένων από αυτό το στοιχείο. Όταν εφαρμόζεται η εξαγωγή οντοτήτων, μπορούμε να προσδιορίσουμε οντότητες, μέρη, οργανισμούς και έννοιες, αριθμητικές εκφράσεις (ημερομηνίες, ώρες, ποσά νομισμάτων, αριθμοί τηλεφώνου) καθώς και χρονικές εκφράσεις (ημερομηνίες, ώρα, διάρκεια, συχνότητα).

6. Κατηγοριοποίηση Δεδομένων

Η κατηγοριοποίηση δεδομένων είναι η διαδικασία επισήμανσης(labelling) μη δομημένων δεδομένων έτσι ώστε να είναι δομημένα και ικανά να αναλυθούν. Αυτό εμπίπτει σε δύο διακριτές κατηγορίες:

Εξόρυξη γνώμης(sentiment analysis) – η εξόρυξη γνώμης και συναισθημάτων από το κείμενο. Για παράδειγμα, τα σχόλια των πελατών ήταν απογοητευμένα ή ευχαριστημένα, θετικά ή ουδέτερα. **Θεματολογία** – προσδιορισμός του «θέματος» του κειμένου. Το κείμενο αφορούσε την πολιτική, τον αθλητισμό ή κάποιο άλλο θέμα;

Και οι δύο αυτές τεχνικές μάς δίνουν τη δυνατότητα να αναλύσουμε μη δομημένο κείμενο για να κατανοήσουμε καλύτερα αυτά τα δεδομένα.

2.2.4 Συντήρηση δεδομένων (Data archiving)

Η αρχειοθέτηση δεδομένων είναι η πρακτική του εντοπισμού δεδομένων που δεν είναι πλέον ενεργά και της μετακίνησής τους από τα συστήματα παραγωγής σε συστήματα μακροπρόθεσμης αποθήκευσης. Τα αρχειακά δεδομένα αποθηκεύονται έτσι ώστε ανά πάσα στιγμή να μπορούν να τεθούν ξανά σε λειτουργία.Τα πλεονεκτήματα της αρχειοθέτησης δεδομένων περιλαμβάνουν τη διασφάλιση ότι

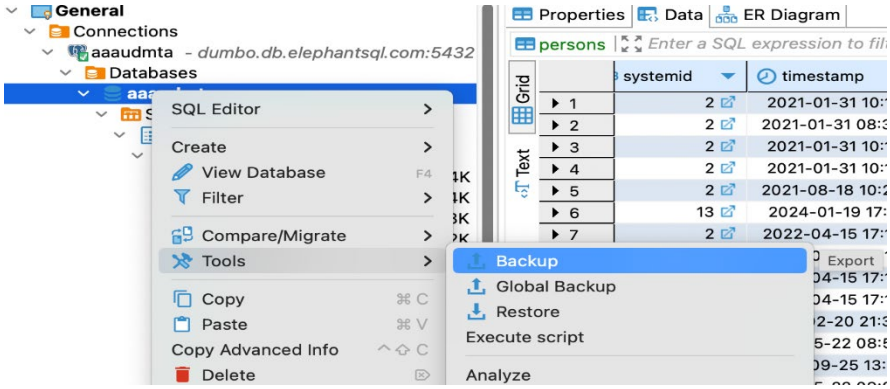
τα συστήματα παραγωγής χρησιμοποιούν λιγότερους πόρους, λειτουργούν πιο αποτελεσματικά και μειώνουν συνολικά το κόστος αποθήκευσης.

Το service που εφαρμόζει το data curation στην βάσης μας θα μπορούσε να προκάλεσει οποιαδήποτε στιγμή καταστροφή ή παραμόρφωση δεδομένων λόγω σφαλμάτων, γι' αυτό τον λόγο η συντήρηση δεδομένων είναι πολύ σημαντική.

Για την αντιμετώπιση αυτών των κινδύνων, χρησιμοποιήσαμε στρατηγικές αρχειοθέτησης δεδομένων με χρήση του εργαλείου **DBeaver**. Το DBeaver παρέχει λειτουργίες δημιουργίας αντιγράφων ασφαλείας (backups), εισαγωγής (import) και εξαγωγής (export) δεδομένων, καθώς και δυνατότητα ανάκτησης χαμένων πληροφοριών, εξασφαλίζοντας τη συνέχεια και την ακεραιότητα των δεδομένων.

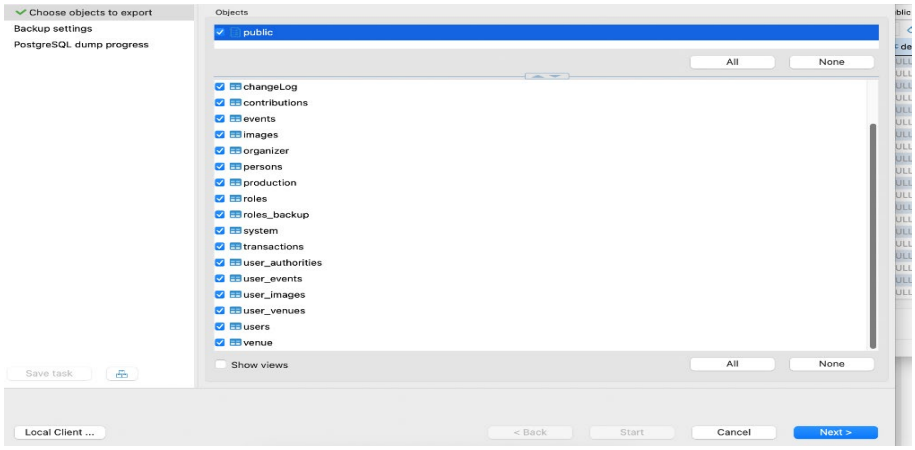
Η βάση που χρησιμοποιούμε είναι PostgreSQL και η διαδικασία για την συντήρηση δεδομένων που εφαρμόσαμε είναι πολυ συγκεκριμένη. Για να δημιουργήσουμε μια εφεδρική βάση δεδομένων ακολουθήσαμε τα παρακάτω βήματα:

1. Επιλέξαμε την επιθυμητή βάση δεδομένων.
2. Κάναμε δεξί κλικ στη βάση δεδομένων και επιλέξαμε Εργαλεία -> Δημιουργία αντιγράφων ασφαλείας.



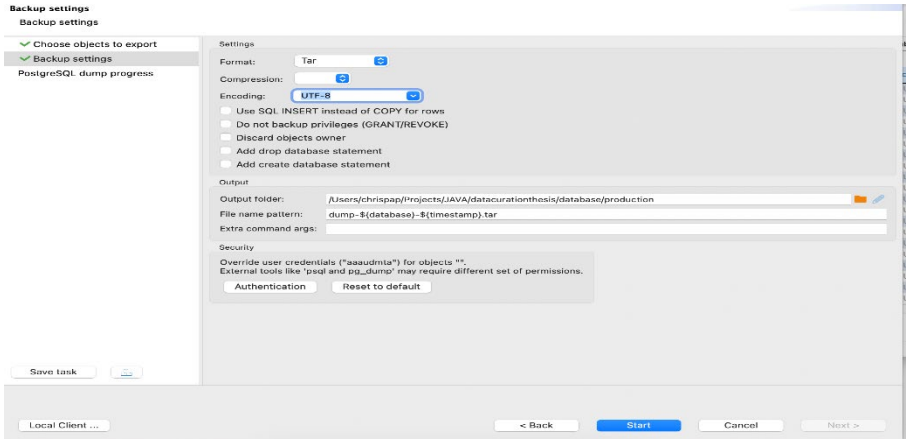
Σχήμα 2.2.4: Backup database [22].

3. Στο παράθυρο Dump επιλέξαμε τα απαραίτητα αντικείμενα και κάναμε κλικ στο Επόμενο.



Σχήμα 2.2.4: Απαραίτητα αντικείμενα για backup [23].

4. Αφού διαμορφώσαμε τις ρυθμίσεις, κάναμε κλικ στο Έναρξη.

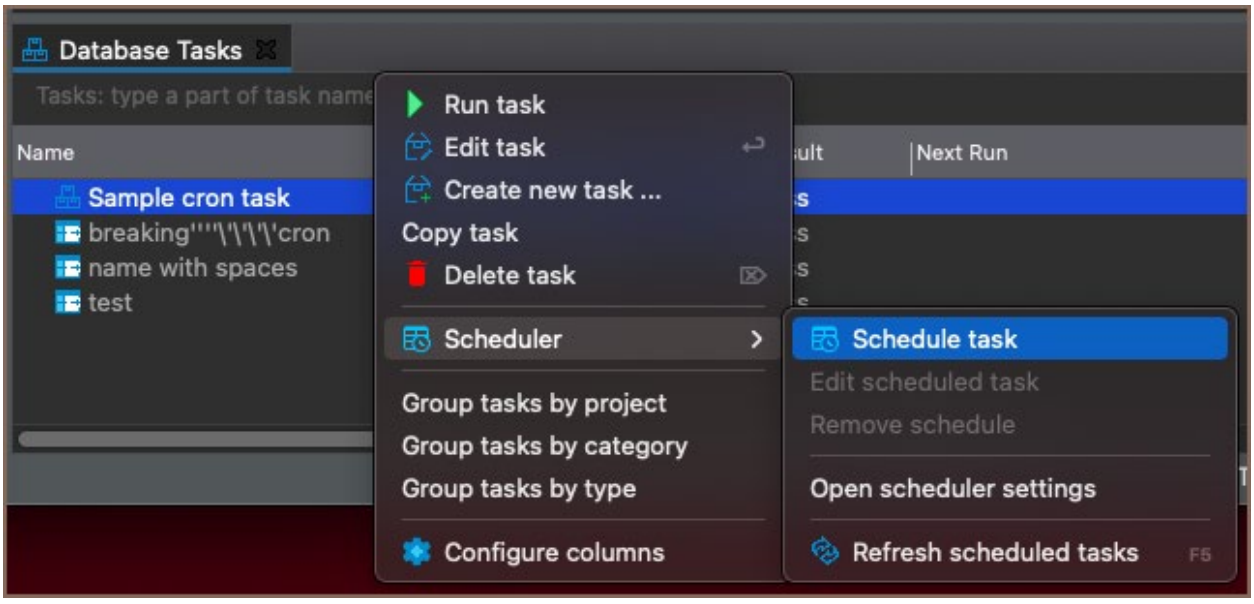


Σχήμα 2.2.4: Backup start [24].

5. Μετά την επιτυχή ολοκλήρωση, θα εμφανιστεί μια ειδοποίηση με πληροφορίες σχετικά με τη διαδικασία.

6. Μπορούμε να βρούμε το αρχείο αντιγράφου ασφαλείας στον φάκελο που καθορίστηκε κατά το βήμα ρυθμίσεων δημιουργίας αντιγράφων ασφαλείας.

Μετα την ολοκλήρωση της διαδικασίας θα ανοίξουμε το αρχείο SQL για να επαληθεύσουμε ότι περιέχει όλα τα δεδομένα που αναμέναμε και θα χρησιμοποιήσουμε χρονοπρογραμματισμός (Scheduling) για προγραμματισμένες εφεδρικές λήψεις της βάσης μας.



Σχήμα 2.2.4: Backup scheduler [25].

2.3 Εργαλεία και Τεχνολογίες για Data Curation

Η επιμέλεια δεδομένων συνεπάγεται με τη χρήση εργαλείων και τεχνικών φιλτραρίσματος για να εντοπιστούν ποια δεδομένα λειτουργούν και ποια δεδομένα όχι. Αυτά τα εργαλεία είναι συνήθως λογισμικά, βιβλιοθήκες ή πλατφόρμες που έχουν σχεδιαστεί για να διευκολύνουν την οργάνωση, τον καθαρισμό, την επικύρωση και τη διαχείριση των δεδομένων που έχουν επιμεληθεί, διασφαλίζοντας την ποιότητα και την προσβασιμότητά τους για ανάλυση και χρήση.

2.3.1 Python και Pandas

Η δημοτικότητα της Python βρίσκεται σε δύο βασικούς πυλώνες. Το ένα είναι ότι είναι μια εύκολη στην εκμάθηση γλώσσα προγραμματισμού, σχεδιασμένη να είναι ευανάγνωστη, με σύνταξη αρκετά σαφή και διαισθητική. Και ο δεύτερος λόγος είναι ότι η φιλικότητα προς το χρήστη δεν αφαιρεί από τη δύναμή τη;. Η Python μπορεί να εκτελέσει μια ποικιλία σύνθετων υπολογισμών και είναι μια από τις πιο ισχυρές γλώσσες προγραμματισμού που προτιμούν οι ειδικοί. Η εφαρμογή Jupyter Notebook είναι μια εφαρμογή διακομιστή-πελάτη που επιτρέπει να επεξεργαστουμε τον κώδικά μας μέσω ενός προγράμματος περιήγησης ιστού. Οι πυρήνες γλωσσών είναι προγράμματα σχεδιασμένα να διαβάζουν και να εκτελούν κώδικα σε μια συγκεκριμένη γλώσσα προγραμματισμού, όπως η Python, η R ή η Julia. Η εγκατάσταση του Jupyter συνοδεύεται πάντα με έναν εγκατεστημένο πυρήνα(kernel) Python και οι άλλοι πυρήνες μπορούν να εγκατασταθούν επιπλέον. Οι διεπαφές, όπου μπορούμε να γράψουμε κώδικα, αντιπροσωπεύουν τους clients. Ένα παράδειγμα τέτοιου client είναι το πρόγραμμα περιήγησης ιστού. Ο διακομιστής Jupyter παρέχει το περιβάλλον όπου ένας πελάτης αντιστοιχίζεται με έναν αντίστοιχο πυρήνα γλωσσών. Στην περίπτωσή μας, θα επικεντρωθούμε στην Python και σε ένα πρόγραμμα περιήγησης ιστού ως πελάτη.

Το Pandas είναι μια βιβλιοθήκη χειρισμού και ανάλυσης δεδομένων ανοιχτού κώδικα για την Python. Έχει σχεδιαστεί για να χειρίζεται, να καθαρίζει και να αναλύει δεδομένα με τρόπο που είναι ταυτόχρονα ισχυρός και διαισθητικός. Με το Panda, μπορούμε να φορτώσουμε δεδομένα από διάφορες πηγές, να τα μετατρέψουμε και να εκτελέσουμε πολύπλοκες λειτουργίες με ευκολία. Το Pandas επίσης παρέχει μια ευέλικτη δομή δεδομένων, τα DataFrames, που διευκολύνει την πλοήγηση, την αναζήτηση και τη μετατροπή δεδομένων, κάνοντας τη γλώσσα ιδανική για εργασίες επιμέλειας δεδομένων. Ένα από τα βασικά χαρακτηριστικά της Pandas είναι η ικανότητα διαχείρισης δεδομένων σε διάφορες μορφές, όπως CSV, JSON, Excel, και SQL. Τα δεδομένα μπορούν εύκολα να αναγνωστούν και να αποθηκευτούν σε μορφές κατάλληλες για ανάλυση ή εισαγωγή σε βάση δεδομένων. Η δημιουργία και η τροποποίηση των DataFrames είναι εξαιρετικά απλή, επιτρέποντας την εύκολη εκτέλεση πολύπλοκων λειτουργιών με ελάχιστο κώδικα.

Στον τομέα του καθαρισμού και του μετασχηματισμού δεδομένων, το Pandas παρέχει λειτουργίες για τον εντοπισμό και την αφαίρεση διπλότυπων ή λανθασμένων εγγραφών, καθώς και για τη διαχείριση ελλιπών τιμών. Τιμές που λείπουν μπορούν να αντικατασταθούν με προκαθορισμένες τιμές, ενώ οι λανθασμένες εγγραφές μπορούν να αφαιρεθούν. Παράλληλα υποστηρίζει την κανονικοποίηση των τιμών, όπως η μορφοποίηση ημερομηνιών και η μετατροπή των δεδομένων σε κατάλληλους τύπους,

προσαρμόζοντας τα δεδομένα στις απαιτήσεις μιας βάσης δεδομένων, όπως η PostgreSQL στην περιπτωσή μας.

```python
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import re
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

df = pd.read_csv('/kaggle/input/theatrical-plays-dataset/production_202405121936.csv')

# Display the first few rows
df.head()
# General dataset information
df.info()
# Check for missing values
print(df.isnull().sum())
# Display unique values for key columns
print("Unique Titles:", df['Title'].nunique())
print("Unique Producers:", df['Producer'].nunique())
# Fill missing values
df['Producer'].fillna('Unknown', inplace=True)
df['Duration'].fillna(df['Duration'].median(), inplace=True)
# Standardize text columns
df['Title'] = df['Title'].str.strip().str.title()
df['Producer'] = df['Producer'].str.strip().str.title()
# Drop unnecessary columns
df.drop(columns=['SystemID'], inplace=True)
```

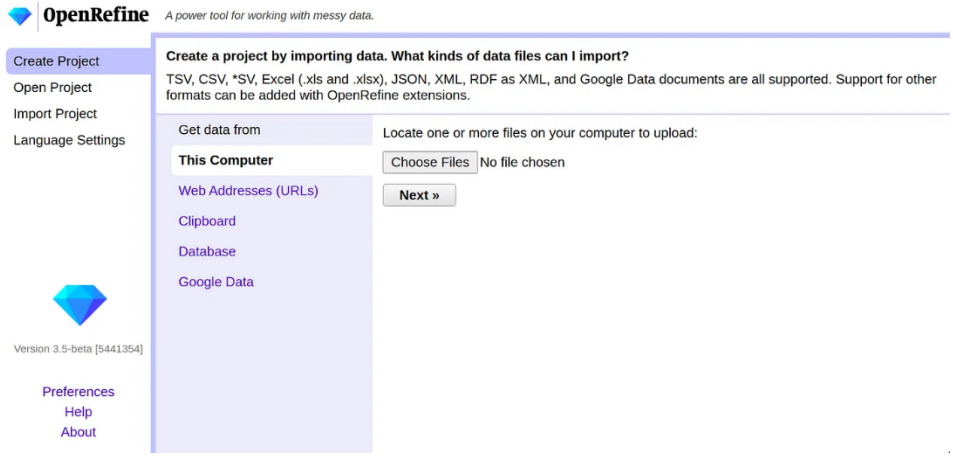
Σχήμα 2.3.1: Pandas DataFrames ανάλυση για τον πίνακα production[26].

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1377 entries, 0 to 1376
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ID           1377 non-null   int64
 1   OrganizerID  1377 non-null   int64
 2   Title        1368 non-null   object
 3   Description  1364 non-null   object
 4   URL          1377 non-null   object
 5   Producer     1368 non-null   object
 6   MediaURL     1367 non-null   object
 7   Duration     12 non-null     object
 8   SystemID     1377 non-null   int64
 9   timestamp    1377 non-null   object
dtypes: int64(3), object(7)
memory usage: 107.7+ KB
ID               0
OrganizerID      0
Title            9
Description     13
URL              0
Producer         9
MediaURL        10
Duration      1365
SystemID         0
timestamp        0
dtype: int64
Unique Titles: 1295
Unique Producers: 371
```

Σχήμα 2.3.1: Αποτελέσματα pandas script για τον πίνακα production[27].

2.3.2 OpenRefine

Το OpenRefine, παλαιότερα γνωστό και ως Google Refine, είναι ένα λογισμικό ανοιχτού κώδικα που χρησιμοποιείται για εργασία με ακατάστατα δεδομένα και παρέχει πολλές λειτουργίες για τελειοποίηση δεδομένων, επεξεργασία δεδομένων, χειρισμό δεδομένων και εισαγωγή δεδομένων στα wikidata. Χρησιμοποιείται ευρέως για την ανίχνευση και διόρθωση σφαλμάτων, την ομαλοποίηση τιμών και τον εμπλουτισμό δεδομένων μέσω σύνδεσης με εξωτερικές πηγές. Το εργαλείο προσφέρει ένα φιλικό περιβάλλον χρήσης που επιτρέπει στους διαχειριστές δεδομένων να αναγνωρίζουν ασυνέπειες και να εφαρμόζουν αυτόματες διορθώσεις, εξασφαλίζοντας τη συνοχή και τη χρηστικότητα των δεδομένων.



Σχήμα 2.3.2: OpenRefine Dashboard[28].

Το OpenRefine παρέχει μια διεπαφή μέσω της οποίας μπορούμε να δημιουργήσουμε ένα project χρησιμοποιώντας οποιοδήποτε CSV, XML, Excel (.xls και .xlsx), JSON και άλλα έγγραφα δεδομένων της Google που μπορούμε πρώτα να δημιουργήσουμε το project μας για να εκτελέσουμε τις λειτουργίες μας. Μπορουμε ακόμη να δημιουργήσουμε project παρέχοντας έγκυρες διευθύνσεις URL για τη λήψη των δεδομένων. Αφού δημιουργήσουμε το project χρησιμοποιώντας διάφορες λειτουργίες όψεων(facet operations), μπορούμε να επιθεωρήσουμε τη φύση των δεδομένων, μπορούμε να μάθουμε αν υπάρχουν μηδενικές τιμές, πανομοιότυπες τιμές κ.λπ. Χρησιμοποιώντας την λειτουργία όψη κειμένου(face text), μπορούμε ακόμη και να ομαδοποιήσουμε τα δεδομένα και να τα παρέχουμε ακριβώς την ίδια τιμή σε όλες τις ίδιες τιμές κελιών, γεγονός που θα διευκολύνει την περαιτέρω επεξεργασία.

Επιπλέον, το OpenRefine υποστηρίζει την αυτοματοποίηση εργασιών χρησιμοποιώντας κανόνες και φίλτρα για την ομοιομορφία των δεδομένων. Μπορούμε να ρυθμίσουμε επαναλαμβανόμενες ενέργειες όπως ο καθαρισμός και η μορφοποίηση δεδομένων εξοικονομώντας χρόνο. Στην παρούσα εργασία, το OpenRefine χρησιμοποιήθηκε εκτενώς για τον καθαρισμό εγγραφών. Τέλος τα καθαρισμένα και εμπλουτισμένα δεδομένα εξήχθησαν σ εμορφές CSV και εισήχθησαν στη βάση δεδομένων PostgreSQL για περαιτέρω επεξεργασία και αποθήκευση.

2.3.3 Stanford NLP: Εργαλεία Επεξεργασίας Φυσικής Γλώσσας (NLP)

Τα εργαλεία επεξεργασίας φυσικής γλώσσας (NLP) διαδραματίζουν κρίσιμο ρόλο στην ανάλυση και εξαγωγή πληροφοριών από μη δομημένα κείμενα. Επιτρέπουν την κατανόηση, την αναγνώριση και την επεξεργασία πληροφοριών, όπως ονόματα, ημερομηνίες και σχέσεις μεταξύ λέξεων, συμβάλλοντας έτσι στη δομή και στον εμπλουτισμό των δεδομένων.

Το Stanford NLP, ένα από τα πιο ισχυρά και δημοφιλή εργαλεία NLP, είναι γραμμένο σε Java και προσφέρει πληθώρα λειτουργιών για την ανάλυση κειμένων. Μέσω αλγορίθμων επεξεργασίας φυσικής γλώσσας μπορεί να επεξεργάζεται κείμενα και να εξάγει πολύτιμες πληροφορίες, οι οποίες ενσωματώνονται στη βάση δεδομένων.

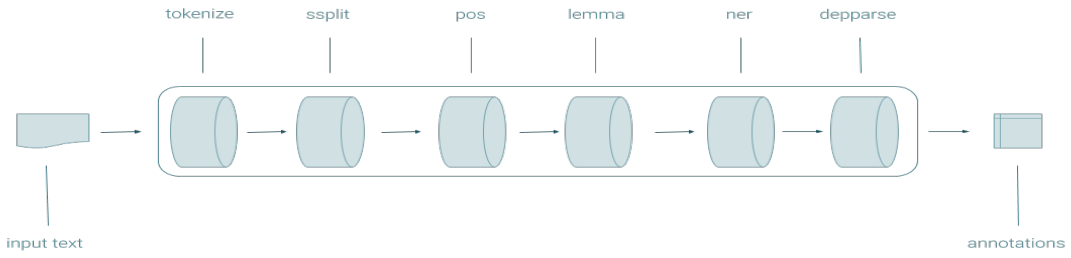
Ένα από τα βασικά χαρακτηριστικά του Stanford NLP είναι η αναγνώριση οντοτήτων (Named Entity Recognition - NER). Αυτή η λειτουργία επιτρέπει τον εντοπισμό και την κατηγοριοποίηση κρίσιμων στοιχείων από το κείμενο όπως ονόματα προσώπων, τοποθεσίες και ημερομηνίες. Η δυνατότητα αυτή είναι ιδιαίτερα χρήσιμη όταν πρόκειται για δεδομένα που σχετίζονται με θεατρικές παραστάσεις καθώς μπορεί να ανιχνεύσει συντελεστές, χώρους και χρονολογίες από περιγραφές παραστάσεων.

Η διαδικασία Tokenization και POS Tagging προσφέρει τη δυνατότητα διάσπασης του κειμένου σε λέξεις ή φράσεις και την ταυτοποίηση της γραμματικής κατηγορίας κάθε στοιχείου (όπως ουσιαστικά, ρήματα, επίθετα). Αυτή η ανάλυση συμβάλλει στη δομική κατανόηση του κειμένου, καθιστώντας δυνατή την εξαγωγή συγκεκριμένων πληροφοριών με ακρίβεια.

Η αξιοποίηση του Stanford NLP στην παρούσα εργασία ανέδειξε το πόσο σημαντικά είναι τα εργαλεία NLP για την ανάλυση μη δομημένων δεδομένων, παρέχοντας ένα ισχυρό εργαλείο για την επιμέλεια και τον εμπλουτισμό πληροφοριών στη βάση δεδομένων. Οι κυριότερες λειτουργίες του Stanford NLP είναι:

Pipeline

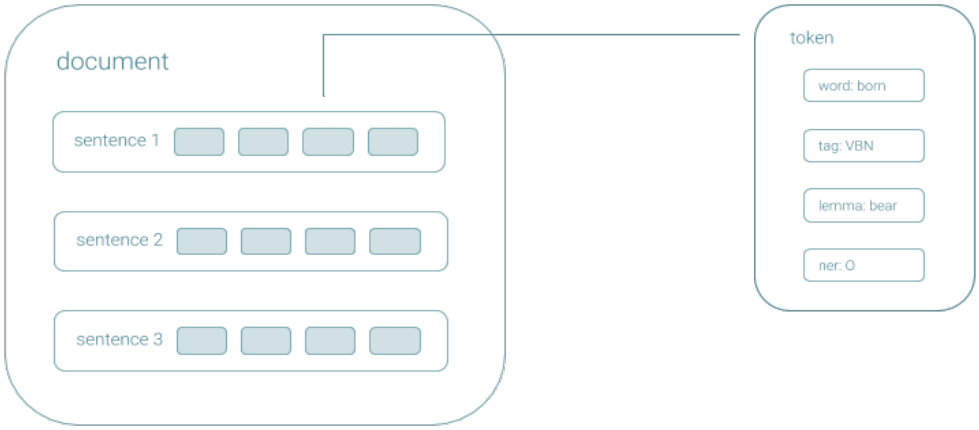
Το κεντρικό στοιχείο του CoreNLP είναι ο αγωγός(pipeline). Οι αγωγοί λαμβάνουν ακατέργαστο κείμενο, εκτελούν μια σειρά σχολιαστών NLP στο κείμενο και παράγουν ένα τελικό σύνολο σχολιασμών.



Σχήμα 2.3.3: Stangord NLP Pipeline[29].

CoreDocument

Οι αγωγοί παράγουν CoreDocuments, αντικείμενα δεδομένων που περιέχουν όλες τις πληροφορίες σχολιασμού, προσβάσιμα με ένα απλό API και σειριοποιήσιμα(serializable) σε μια προσωρινή μνήμη πρωτοκόλλου Google(Google Protocol Buffer).

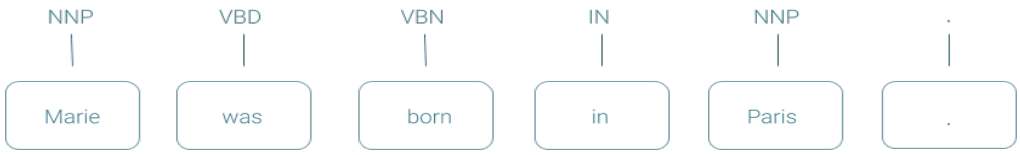


Σχήμα 2.3.3: Stangord NLP CoreDocument[30].

Annotations

Το CoreNLP δημιουργεί μια ποικιλία γλωσσικών σχολιασμών, όπως:

PARTS OF SPEECH

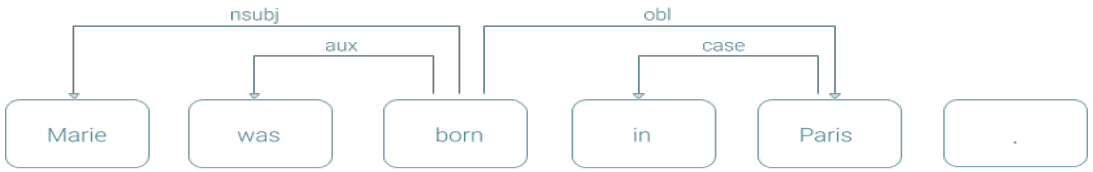


Σχήμα 2.3.3: Stanford NLP Parts of speech annotation[31].

NAMED ENTITIES

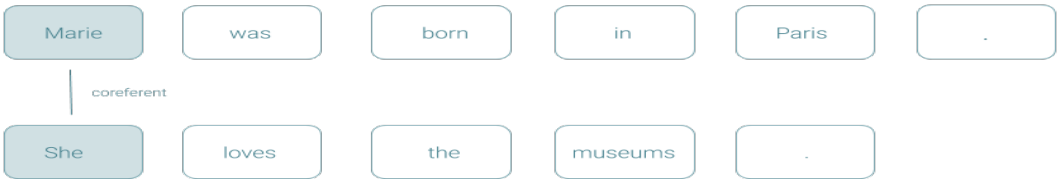
Σχήμα 2.3.3: Stanford NLP named entities annotation[32].

DEPENDENCY PARSES



Σχήμα 2.3.3: Stanford NLP dependency parses annotation[33].

COREFERENCE



Σχήμα 2.3.3: Stanford NLP coreference annotation[34].

Κεφάλαιο 3ο: Υλοποίηση Backend με Spring Boot(Java)

3.1 Εισαγωγή στο Spring Boot

Η ανάπτυξη του backend αποτελεί έναν από τους κεντρικούς πυλώνες της παρούσας εργασίας, καθώς καθορίζει τη λειτουργικότητα και τη διασύνδεση του συστήματος με τη βάση δεδομένων, τις εξωτερικές υπηρεσίες και τα εργαλεία επεξεργασίας δεδομένων. Για την υλοποίηση του backend χρησιμοποιήθηκε το Spring Boot, ένα ευέλικτο και δημοφιλές framework της γλώσσας Java.

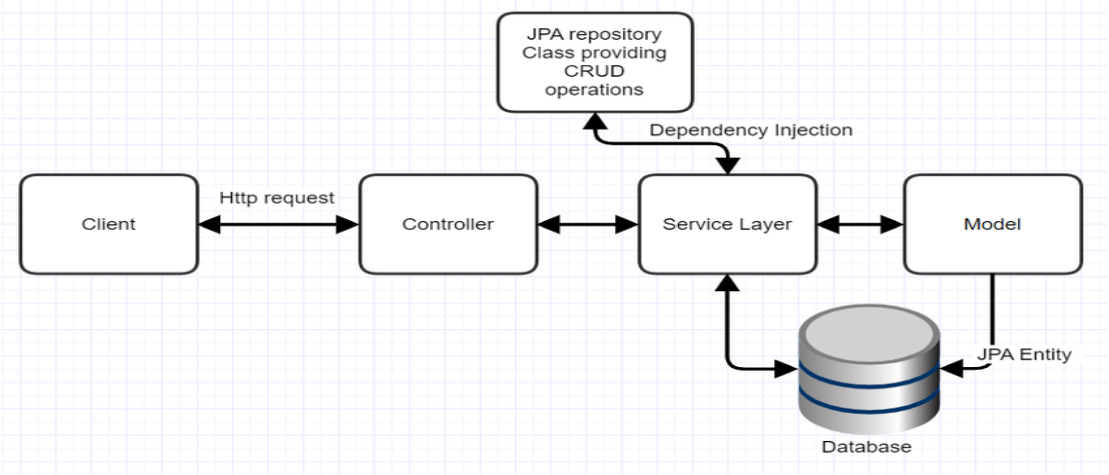
Το Spring Boot επιλέχθηκε λόγω της δυνατότητάς του να προσφέρει μια εύχρηστη και παραμετροποιήσιμη πλατφόρμα για την ανάπτυξη εφαρμογών με υψηλή απόδοση και σταθερότητα. Παρέχει έτοιμες λύσεις για τη διαχείριση των RESTful APIs, τη σύνδεση με βάσεις δεδομένων και την ενσωμάτωση εξωτερικών υπηρεσιών, ενώ μειώνει τον χρόνο ανάπτυξης μέσω αυτοματοποιημένων ρυθμίσεων. Επιπλέον, υποστηρίζει πλήρως τη διασύνδεση με σύγχρονα εργαλεία τεχνητής νοημοσύνης και επεξεργασίας φυσικής γλώσσας, όπως το Stanford NLP και τα LLMs APIs της OpenAI και Ollama της Meta που χρησιμοποιήσαμε.

Ο βασικός στόχος της ανάπτυξης του backend ήταν να εξασφαλιστεί η ομαλή διαχείριση και οργάνωση των δεδομένων, όπως οι θεατρικές παραστάσεις, οι συντελεστές, οι ρόλοι και οι τοποθεσίες. Το σύστημα σχεδιάστηκε ώστε να είναι επεκτάσιμο, ασφαλές και διαλειτουργικό, επιτρέποντας την εύκολη πρόσβαση στα δεδομένα μέσω RESTful APIs. Παράλληλα, δόθηκε έμφαση στη διατήρηση της ακεραιότητας και της ακρίβειας των δεδομένων, μέσω κατάλληλου σχεδιασμού και εφαρμογής μηχανισμών επικύρωσης.

Το Spring Boot, μέσω του modular σχεδιασμού του, διευκόλυνε την υλοποίηση βασικών λειτουργιών, όπως η επικοινωνία με τη βάση PostgreSQL, η δημιουργία και η διαχείριση πινάκων, καθώς και η ενσωμάτωση εξωτερικών εργαλείων ανάλυσης και επεξεργασίας. Το παρόν κεφάλαιο εξετάζει τη δομή και την αρχιτεκτονική του backend συστήματος, περιγράφει τη σχεδίαση και την υλοποίηση της βάσης δεδομένων, και παρουσιάζει τις βασικές υπηρεσίες που υλοποιήθηκαν μέσω RESTful APIs, αναδεικνύοντας τη σημασία του Spring Boot στην ανάπτυξη του συστήματος.

3.2 Αρχιτεκτονική και Δομή του Backend Συστήματος

Η αρχιτεκτονική του backend συστήματος ακολουθεί το πρότυπο Model-View-Controller (MVC), το οποίο διαχωρίζει τη λογική της εφαρμογής, τη διαχείριση δεδομένων και την παρουσίαση. Αυτό το μοντέλο προσφέρει ξεκάθαρη διάκριση των ρόλων κάθε επιπέδου, διευκολύνοντας την ανάπτυξη και τη συντήρηση του συστήματος.



Σχήμα 3.2: Backend architecture[35].

3.2.1 Πακέτα

Το backend σύστημα που αναπτύχθηκε με Spring Boot βασίζεται σε ένα σύνολο πακέτων και βιβλιοθηκών που εξασφαλίζουν την ομαλή λειτουργία, την επεκτασιμότητα, και την αποτελεσματική διαχείριση δεδομένων. Η επιλογή των πακέτων έγινε με γνώμονα τη βέλτιστη υποστήριξη της επεξεργασίας δεδομένων, τη διασύνδεση με εξωτερικές υπηρεσίες και την ασφάλεια της εφαρμογής.

```
<dependencies>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- JAXB API -->
    <dependency>
        <groupId>javax.xml.bind</groupId>
        <artifactId>jaxb-api</artifactId>
        <version>2.3.1</version>
    </dependency>
    <!-- JAXB Runtime -->
    <dependency>
        <groupId>com.sun.xml.bind</groupId>
        <artifactId>jaxb-core</artifactId>
        <version>2.3.0.1</version>
    </dependency>
    <dependency>
        <groupId>com.sun.xml.bind</groupId>
        <artifactId>jaxb-impl</artifactId>
        <version>2.3.3</version>
    </dependency>
```

```xml
<dependency>
    <groupId>dev.langchain4j</groupId>
    <artifactId>langchain4j-ollama-spring-boot-starter</artifactId>
    <version>0.33.0</version>
</dependency>

<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-ollama-spring-boot-starter</artifactId>
    <version>1.0.0-M4</version>
</dependency>

<dependency>
    <groupId>org.springframework.retry</groupId>
    <artifactId>spring-retry</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-aop</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-openai-spring-boot-starter</artifactId>
</dependency>
```

Σχήμα 3.2.1: po.xml παράδειγμα[36].

Spring Boot Starter Libraries

- spring-boot-starter-actuator: Επιτρέπει την παρακολούθηση της εφαρμογής μετρήσεις και άλλα χαρακτηριστικά.
- spring-boot-starter-data-jpa: Παρέχει λειτουργίες για διαχείριση δεδομένων με χρήση του JPA (Java Persistence API).
- spring-boot-starter-web: Υποστηρίζει την ανάπτυξη RESTful APIs και web εφαρμογών.
- spring-boot-starter-thymeleaf: Προσφέρει δυνατότητες για δυναμική απόδοση HTML περιεχομένου.

Βάση Δεδομένων

- org.postgresql:postgresql: Παρέχει τη σύνδεση και διαχείριση δεδομένων στη βάση PostgreSQL.

Επεξεργασία Φυσικής Γλώσσας

- edu.stanford.nlp:stanford-corenlp: Εργαλείο για ανάλυση φυσικής γλώσσας, όπως αναγνώριση οντοτήτων και ανάλυση συντακτικών σχέσεων.
- org.apache.opennlp:opennlp-tools: Παρέχει λειτουργίες για επεξεργασία γλώσσας, όπως tokenization και parsing.
- org.languagetool:language-all: Χρησιμοποιείται για γραμματική και συντακτικό έλεγχο.

Εξωτερικές Διασυνδέσεις και APIs

- spring-ai-openai-spring-boot-starter: Επιτρέπει τη διασύνδεση με OpenAI APIs για τη χρήση μεγάλων γλωσσικών μοντέλων (LLMs).
- langchain4j-ollama-spring-boot-starter: Παρέχει λειτουργίες για την ανάπτυξη εφαρμογών με ενσωμάτωση γλωσσικών μοντέλων και εργαλείων NLP.
- spring-retry: Χρησιμοποιείται για τη διαχείριση επαναληπτικών προσπαθειών σε αποτυχημένα αιτήματα APIs.

Εργαλεία Διαχείρισης και Ανάπτυξης

- spring-boot-starter-log4j2: Υποστηρίζει καταγραφή (logging) για παρακολούθηση και ανάλυση της εφαρμογής.
- spring-boot-devtools: Ενισχύει την εμπειρία ανάπτυξης με λειτουργίες όπως αυτόματη επανεκκίνηση της εφαρμογής κατά την αλλαγή του κώδικα.

Επεξεργασία και Διαχείριση Δεδομένων

- org.apache.commons:commons-lang3: Παρέχει χρήσιμα βοηθητικά εργαλεία για τη διαχείριση κειμένων και δεδομένων.
- javax.xml.bind:jaxb-api: Χρησιμοποιείται για επεξεργασία και μετατροπή δεδομένων XML.
- dk.dren.hunspell:hunspelljna: Υποστηρίζει λειτουργίες ελέγχου ορθογραφίας.

Batch Processing

- spring-batch-core: Ενισχύει τη διαχείριση μεγάλων όγκων δεδομένων μέσω batch επεξεργασίας.

Η επιλογή των παραπάνω πακέτων και βιβλιοθηκών διαμορφώνει ένα ολοκληρωμένο σύστημα backend, ικανό να ανταποκριθεί στις απαιτήσεις της διαχείρισης δεδομένων και της ενσωμάτωσης σύγχρονων τεχνολογιών επεξεργασίας γλώσσας και τεχνητής νοημοσύνης.

3.3 Λίστα data curations και υλοποίηση με κώδικα Java

Σε αυτό το κεφάλαιο θα αναλύσουμε την υλοποίηση της επιμέλειας δεδομένων των πινάκων της βάσης μας χρησιμοποιώντας κώδικα Java.

3.3.1 General Data Normalization

Το General Data Normalization αναφέρεται στη διαδικασία καθαρισμού και τυποποίησης δεδομένων ώστε να διασφαλιστεί η ομοιομορφία, η ποιότητα και η αξιοπιστία τους. Περιλαμβάνει ενέργειες όπως η αφαίρεση ανεπιθύμητων χαρακτήρων, η τυποποίηση της χρήσης κενού χώρου και η

ορθογραφική διόρθωση. Αυτές οι ενέργειες βοηθούν στην αποτροπή σφαλμάτων κατά την επεξεργασία, την αναζήτηση και την ανάλυση δεδομένων.

Οι κύριες λειτουργίες περιλαμβάνουν:

- Trimming Extra Spaces: Αφαιρεί τα περιττά κενά από την αρχή, το τέλος ή και ενδιάμεσα στις τιμές κειμένου.
- Special Character Removal: Αφαιρεί χαρακτήρες που δεν είναι αλφαριθμητικοί ή δεν είναι απαραίτητοι.
- Whitespace Standardization: Αντικαθιστά πολλαπλά συνεχόμενα κενά με ένα μόνο.

```
private static final Pattern MULTIPLE_SPACES_PATTERN = Pattern.compile(regex:"\\s{2,}");
private static final Pattern SPECIAL_CHARACTERS_PATTERN = Pattern.compile(regex:"[^\\p{L}\\p{N}\\s]");
private static final Pattern EMAIL_PATTERN = Pattern.compile(regex:"^[A-Za-z0-9+_.-]+@(.+)$");
private static final Pattern PHONE_PATTERN = Pattern.compile(regex:"^(\\d{10}|\\d{12})$");
private static final Pattern INVALID_LETTER_PATTERN = Pattern.compile(regex:"\\s+[a-zA-Zα-ωA-Ω]\\s+");
@Autowired
private OrganizerRepository organizerRepository;
@Autowired
private VenueRepository venueRepository;

public String normalizeString(String str) {
    return str.trim().replaceAll(SPECIAL_CHARACTERS_PATTERN.pattern(), replacement:"")
            .replaceAll(MULTIPLE_SPACES_PATTERN.pattern(), replacement:" ");
}

public String normalizeAddressString(String str) {
    // Trim the string to remove leading and trailing spaces
    str = str.trim();
    // Convert to uppercase
    str = str.toUpperCase();
    // Replace multiple spaces with a single space
    str = str.replaceAll(regex:"\\s+", replacement:" ");
    // Return the normalized address string
    return str;
}
```

Σχήμα 3.3.1: Regex for General data normalization[37].

Ο παραπάνω κώδικας περιλαμβάνει δύο βασικές μεθόδους (normalizeString και normalizeAddressString) και σταθερές εκφράσεις (Regular Expressions - Pattern) για την κανονικοποίηση δεδομένων που εισάγονται στη βάση δεδομένων μας.

Μέθοδος normalizeString

Η μέθοδος αυτή εκτελεί τις εξής λειτουργίες κανονικοποίησης σε μια συμβολοσειρά:

- Αφαίρεση Κενών: Χρησιμοποιώντας την ενσωματωμένη μέθοδο trim, αφαιρεί τα κενά στην αρχή και το τέλος της συμβολοσειράς.
- Αφαίρεση Ειδικών Χαρακτήρων: Χρησιμοποιώντας το SPECIAL_CHARACTERS_PATTERN, αφαιρεί οποιονδήποτε χαρακτήρα δεν είναι γράμμα, αριθμός ή διάστημα.
- Κανονικοποίηση Κενών: Με το MULTIPLE_SPACES_PATTERN, αντικαθιστά τα πολλαπλά συνεχόμενα κενά με ένα μόνο.

Μέθοδος normalizeAddressString

Αυτή η μέθοδος είναι εξειδικευμένη για την κανονικοποίηση διευθύνσεων, εκτελώντας τις παρακάτω ενέργειες:

- Αφαίρεση Κενών: Χρησιμοποιεί το trim για να αφαιρέσει τα περιττά κενά στην αρχή και το τέλος.
- Μετατροπή σε Κεφαλαία Γράμματα: Χρησιμοποιώντας τη μέθοδο toUpperCase, όλα τα γράμματα μετατρέπονται σε κεφαλαία για συνέπεια.
- Κανονικοποίηση Κενών: Αντικαθιστά όλα τα πολλαπλά συνεχόμενα κενά με ένα μόνο.

3.3.2 Entity-Specific Curation

Η Ειδική Επιμέλεια Δεδομένων για Οντότητες (Entity-Specific Curation) αποτελεί μια εξειδικευμένη διαδικασία που εφαρμόζεται σε συγκεκριμένους πίνακες ή οντότητες μιας βάσης δεδομένων, προκειμένου να βελτιστοποιηθεί η ποιότητα των δεδομένων τους με τρόπους που αντικατοπτρίζουν τις ιδιαιτερότητες και τις ανάγκες της κάθε οντότητας.

Σε αντίθεση με τη Γενική Κανονικοποίηση Δεδομένων (General Data Normalization), η οποία εφαρμόζεται οριζόντια σε όλα τα πεδία ανεξάρτητα από το περιεχόμενο, η Ειδική Επιμέλεια εστιάζει στη σημασιολογική και λειτουργική βελτίωση των δεδομένων μιας οντότητας.

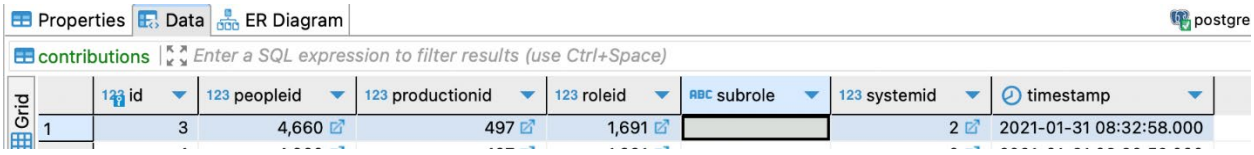
Contributions Table

- SubRole Normalization: Αφαιρεί ειδικούς χαρακτήρες και περιττά κενά.
  Αν το subRole λείπει, δημιουργείται αυτόματα με βάση τις σχέσεις μεταξύ Contribution και Person μέσω NLP και LLM API για την αντιστοίχιση ρόλων.
- Duplicate Merging: Συνδυάζει διπλές εγγραφές Contributions με βάση κοινά personId, productionId και roleId.

```java
public Contribution cleanContribution(Contribution contribution) {
    if (contribution.getSubrole() != null && !contribution.getSubrole().isEmpty()) {
        contribution.setSubrole(normalizeString(contribution.getSubrole()));
    }

    if (contribution.getSubrole() == null || contribution.getSubrole().isEmpty()) {
        Production production = contribution.getProduction();
        String url = production.getUrl();
        Person person = contribution.getPerson();
        String fullname = person.getFullname();

        System.out.println("Production: " + production.toString());
        System.out.println("Production URL: " + url);
        System.out.println("Person: " + person.toString());
        System.out.println("Person fullname: " + fullname);

        // Fetch subroles from LLM
        Map<String, List<String>> subrolesMap = ollamaController.getSubrolesFromLLM(url);
        StringBuilder newSubrole = new StringBuilder();
        subrolesMap.forEach((role, names) -> {
            if (names.contains(fullname)) {
                if (newSubrole.length() > 0) {
                    newSubrole.append(str:" , ");
                }
                newSubrole.append(role);
            }
        });

        if (newSubrole.length() > 0) {
            contribution.setSubrole(newSubrole.toString());
            System.out.println("Updated subRole with roles: " + newSubrole.toString());
        } else {
            System.out.println(x:"The fullname does not exist in the subroles JSON response.");

        }
    }

    return contribution;
}
```

Σχήμα 3.3.2: Clean contribution method[38].

Παράδειγμα από τον πίνακα cotributions εφαρμόζοντας text preprocessing curation:



Σχήμα 3.3.2: Εικόνα πριν το curation από τον πίνακα contribution[39].

contributions | Enter a SQL expression to filter results (use Ctrl+Space)

Grid

	id	peopleid	productionid	roleid	subrole	systemid	timestamp
1	3	4,660	497	1,691	Σκηνικά	2	2021-01-31 08:32:58.000

Σχήμα 3.3.2: Εικόνα μετά το curation από τον πίνακα contribution[40].

Roles Table

Case-Sensitive Duplicate Merging:

- Διασφαλίζει ότι δεν υπάρχουν διπλές εγγραφές ρόλων με ταυτόσημες τιμές πεδίων.
- Προτεραιότητα δίνεται σε ρόλους με ενεργές σχέσεις (συνδέσεις με άλλες οντότητες όπως οι Contributions) κατά τη συγχώνευση διπλών εγγραφών.
- Αν υπάρχουν πολλαπλοί ρόλοι με το ίδιο όνομα αλλά διαφορετικές σχέσεις, διατηρούνται όλες οι ενεργές σχέσεις.

```
@Service
public class RoleCurationService {
@Autowired
    private RoleRepository roleRepository;

    /**
     * Συγχωνεύει διπλές εγγραφές στον πίνακα Roles.
     */
    public void mergeDuplicateRoles() {
        List<Role> allRoles = roleRepository.findAll();

        Map<String, List<Role>> rolesByName = allRoles.stream()
                .collect(Collectors.groupingBy(role -> role.getRole1().toLowerCase()));

        rolesByName.forEach((roleName, roles) -> {
            if (roles.size() > 1) {
                Role primaryRole = roles.stream()
                        .max(Comparator.comparingInt(this::getActiveRelationsCount))
                        .orElse(roles.get(index:0));

                System.out.printf(format:"Πρωτεύων ρόλος για '%s': %s%n", roleName, primaryRole.getId());

                roles.stream()
                        .filter(role -> role != primaryRole)
                        .forEach(duplicateRole -> {
                            mergeRelations(primaryRole, duplicateRole);
                            roleRepository.delete(duplicateRole);
                            System.out.printf(format:"Διαγραφή διπλού ρόλου: %s%n", duplicateRole.getId());
                        });

                roleRepository.save(primaryRole);
            }
        });
    }
```

Σχήμα 3.3.2: Clean Role table[90].

```java
    /**
     * Υπολογίζει τον αριθμό ενεργών σχέσεων ενός ρόλου.
     */
    private int getActiveRelationsCount(Role role) {
        return role.getContributions().size();
    }

    /**
     * Συγχωνεύει τις σχέσεις ενός διπλού ρόλου στον πρωτεύοντα ρόλο.
     */
    private void mergeRelations(Role primaryRole, Role duplicateRole) {
        duplicateRole.getContributions().forEach(contribution -> {
            contribution.setRole(primaryRole);
        });
        if (duplicateRole.getContributions() != null) {
            if (primaryRole.getContributions() == null) {
                primaryRole.setContributions(new ArrayList<>());
            }
            primaryRole.getContributions().addAll(duplicateRole.getContributions());
        }
    }
}
```

Σχήμα 3.3.2: Clean Role table 2[95].

1. Φόρτωση όλων των ρόλων:
   - Όλοι οι ρόλοι φορτώνονται από τη βάση δεδομένων μέσω του RoleRepository.
2. Ομαδοποίηση ανά όνομα:
   - Οι ρόλοι ομαδοποιούνται με βάση το όνομά τους, ανεξαρτήτως πεζών-κεφαλαίων (case-insensitive).
3. Εντοπισμός πρωτεύοντα ρόλου:
   - Ο πρωτεύων ρόλος είναι αυτός με τις περισσότερες ενεργές σχέσεις (π.χ., συνδέσεις με συνεισφορές).
4. Συγχώνευση σχέσεων:
   - Όλες οι σχέσεις του διπλού ρόλου (π.χ., από τον πίνακα Contributions) ανατίθενται στον πρωτεύοντα ρόλο.
   - Αν υπάρχουν άλλες σχέσεις (π.χ., διαφορετικοί τύποι δεδομένων), συγχωνεύονται ανάλογα.
5. Διαγραφή διπλών:
   - Οι διπλοί ρόλοι διαγράφονται αφού οι σχέσεις τους έχουν συγχωνευθεί.
6. Αποθήκευση του πρωτεύοντα ρόλου:
   - Ο ενημερωμένος πρωτεύων ρόλος αποθηκεύεται στη βάση δεδομένων.

1566	3,255	βοηθοί σκηνοθέτη	3	2023-02-20 22:10:05.000
1567	3,256	ΒΟΗΘΟΙ ΣΚΗΝΟΘΕΤΗ	3	2023-02-20 22:10:05.000

Σχήμα 3.3.2: Εικόνα πριν το curation από τον πίνακα roles[41].

1566	3,255	Βοηθοί Σκηνοθέτη	3	2023-02-20 22:10:05.000

Σχήμα 3.3.2: Εικόνα μετά το curation από τον πίνακα roles[42].

Persons Table

Fullname Curation:

- Τυποποιεί τα ονόματα αφαιρώντας περιττά κενά και ομαλοποιώντας την εμφάνιση κεφαλαίων-πεζών.
- Διορθώνει ονόματα με μη έγκυρη απόσταση γραμμάτων ή ειδικούς χαρακτήρες χρησιμοποιώντας καθαρισμό βάσει regex.
- Εντοπίζει και διορθώνει τυπογραφικά λάθη, εφαρμόζοντας έλεγχο ορθογραφίας για συνέπεια.

```java
@Service
public class PersonCurationService {

    @Autowired
    private SpellCheckService spellCheckService;

    // Regex patterns για καθαρισμό
    private static final Pattern MULTIPLE_SPACES_PATTERN = Pattern.compile(regex:"\\s{2,}");
    private static final Pattern INVALID_CHARACTERS_PATTERN = Pattern.compile(regex:"[^\\p{L}\\s'-]");
    private static final Pattern INVALID_SPACING_PATTERN = Pattern.compile(regex:"\\b[a-zA-Zα-ωA-Ω]\\b");

    /**
     * Τυποποίηση και καθαρισμός των ονομάτων στον πίνακα Persons.
     */
    public Person curatePerson(Person person) {
        if (person.getFullname() != null) {
            String cleanedName = cleanAndNormalizeName(person.getFullname());
            person.setFullname(cleanedName);
        }
        return person;
    }

    /**
     * Καθαρίζει και τυποποιεί το όνομα.
     */
    private String cleanAndNormalizeName(String name) {
        // Αφαίρεση ειδικών χαρακτήρων
        name = INVALID_CHARACTERS_PATTERN.matcher(name).replaceAll(replacement:"");
        // Καθαρισμός περιττών κενών
        name = MULTIPLE_SPACES_PATTERN.matcher(name).replaceAll(replacement:" ");
        // Αφαίρεση ακατάλληλων αποστάσεων γραμμάτων
        name = INVALID_SPACING_PATTERN.matcher(name).replaceAll(replacement:"");
        // Τυποποίηση κεφαλαίων-πεζών
        name = capitalizeWords(name);
```

Σχήμα 3.3.2: Clean Persons table[89].

```
        // Έλεγχος ορθογραφίας
        if (!spellCheckService.isValidWord(name)) {
            String correctedName = spellCheckService.autoCorrect(name);
            System.out.printf(format:"Διόρθωση ονόματος από '%s' σε '%s'%n", name, correctedName);
            name = correctedName;
        }

        return name.trim();
    }

    /**
     * Κεφαλαιοποιεί τις πρώτες λέξεις κάθε λέξης.
     */
    private String capitalizeWords(String input) {
        return Arrays.stream(input.split(regex:"\\s+"))
                .map(word -> word.substring(beginIndex:0, endIndex:1).toUpperCase() + word.substring(beginIndex:1).toLowerCase())
                .collect(Collectors.joining(delimiter:" "));
    }

    /**
     * Επεξεργασία όλων των εγγραφών του πίνακα Persons.
     */
    public void curateAllPersons(List<Person> persons) {
        persons.forEach(this::curatePerson);
    }
}
```

Σχήμα 3.3.2: Clean Persons table 2[43].

Καθαρισμός Ονομάτων:

- Αφαιρούνται ειδικοί χαρακτήρες με το INVALID_CHARACTERS_PATTERN.
- Εντοπίζονται και διορθώνονται τυπογραφικά λάθη, π.χ., περιττά κενά ή γράμματα με λάθος απόσταση.

Τυποποίηση Κεφαλαίων-Πεζών:

- Η μέθοδος capitalizeWords κεφαλαιοποιεί κάθε λέξη στο όνομα.

Έλεγχος Ορθογραφίας:

- Χρησιμοποιείται η υπηρεσία SpellCheckService για να διορθωθούν λάθη ορθογραφίας.
- Αν το όνομα δεν είναι έγκυρο, γίνεται αυτόματη διόρθωση.

Εφαρμογή σε Πολλαπλές Εγγραφές:

- Η μέθοδος curateAllPersons εφαρμόζει την επεξεργασία σε όλες τις εγγραφές του πίνακα.

Venue Table

Title Cleaning:

- Αντικαθιστά ειδικούς συμβολισμούς (& με and, , κ.λπ.).
- Αφαιρεί διπλά κενά και περιττά διαστήματα.

Address Normalization:

- Εφαρμόζει κεφαλαία και διορθώνει περιττά κενά.
- Χρησιμοποιεί LLM APIs για να συμπληρώσει αυτόματα ελλιπείς διευθύνσεις.

Duplicate Merging: Συγχωνεύει διπλές εγγραφές βάσει των πεδίων τίτλου και διεύθυνσης, διατηρώντας το κύριο Venue με ενεργές σχέσεις.

```
public Venue cleanVenue(Venue venue) {
    Map<String, List<String>> entities = new HashMap<>();

    if (venue.getTitle() != null) {
        mergeDuplicates(venue);
        // Normalize the string
        if (venue.getTitle().contains(s:"&")) {
            venue.setTitle(venue.getTitle().replace(target:"&", replacement:"και"));
        }
        String title = normalizeString(venue.getTitle());
        title = StringUtils.capitalizeWords(title);
        LoggerController.formattedInfo(format:"START: Normalized title: %s", title);
        String possibleTitleReplacement = null;

        // Spell check
        LoggerController.formattedInfo(format:"Spell checked title: %s",
                title + " - " + spellCheckService.isValidWord(title));
        // Remove all special characters
        title = SPECIAL_CHARACTERS_PATTERN.matcher(title).replaceAll(replacement:"");
        // Remove random letter with spaces around it
        title = INVALID_LETTER_PATTERN.matcher(title).replaceAll(replacement:" ");
        // Remove any multiple spaces that might have been introduced
        title = MULTIPLE_SPACES_PATTERN.matcher(title).replaceAll(replacement:" ");

        // Entity extraction
        entities = nlpService.extractEntities(title);
        LoggerController.formattedInfo(format:"Venue Extracted entities: %s", entities);

        // Spell check and correction
        if (!spellCheckService.isValidWord(title)) {
            logInvalidWord(title, entity:"Venue", field:"title");
            possibleTitleReplacement = spellCheckService.autoCorrect(title);
            LoggerController.formattedInfo(format:"Venue title corrected by spell checking service to: %s",
                    possibleTitleReplacement);
            title = possibleTitleReplacement;
            venue.setTitle(title.trim());
        }
        if (spellCheckService.isEnglishText(title)) {
            LoggerController.formattedInfo(format:"English word detected: %s", title);
            possibleTitleReplacement = spellCheckService.autoCorrectEnglish(title);
            LoggerController.formattedInfo(format:"Venue title corrected by spell checking service to: %s",
                    possibleTitleReplacement);
            title = possibleTitleReplacement;
            venue.setTitle(title.trim());
        }
```

Σχήμα 3.3.2: Clean Venue table[88].

```
        if (spellCheckService.isGreekText(title)) {
            LoggerController.info(message:"Applying Levenshtein distance");
            SpellCheckResponse levenshteinResponse = greekSpellCkeckerService.checkAndSuggestSentence(title);
            String levenshteinSuggestion = levenshteinResponse.getLevenshteinSuggestion();
            LoggerController.formattedInfo(format:"Venue title corrected by Levenshtein distance to: %s",
                    levenshteinSuggestion);

            // Applying Hamming distance
            LoggerController.info(message:"Applying Hamming distance algorithm!");
            String hammingDistanceSuggestion = levenshteinService.hammingDistanceForSentence(title);
            LoggerController.formattedInfo(format:"Venue title corrected by Hamming distance to: %s",
                    hammingDistanceSuggestion);

            // Compare suggestions and ask the user to choose
            System.out.println(x:"Choose a correction for the venue title:");
            System.out.printf(format:"1. Levenshtein suggestion: %s\n", levenshteinSuggestion);
            System.out.printf(format:"2. Hamming distance suggestion: %s\n", hammingDistanceSuggestion);
            System.out.print(
                    s:"Enter the number of the suggestion you want to apply (or press enter to keep the original): ");

            Scanner scanner = new Scanner(System.in);    Resource leak: 'scanner' is never closed
            String input = scanner.nextLine().trim();

            if (!input.isEmpty()) {
                if (input.equals(anObject:"1")) {
                    title = levenshteinSuggestion;
                    LoggerController.formattedInfo(format:"Venue title corrected by Levenshtein distance to: %s", title);
                } else if (input.equals(anObject:"2")) {
                    title = hammingDistanceSuggestion;
                    LoggerController.formattedInfo(format:"Venue title corrected by Hamming distance to: %s", title);
                } else {
                    System.out.print(s:"Enter your custom correction (or press enter to keep the original): ");
                    String customCorrection = scanner.nextLine().trim();
                    if (!customCorrection.isEmpty()) {
                        title = customCorrection;
                        LoggerController.formattedInfo(format:"Venue title manually corrected to: %s", title);
                    } else {
                        LoggerController.formattedInfo(format:"No corrections applied. Keeping original title: %s", title);
                    }
                }
            } else {
                LoggerController.formattedInfo(format:"No corrections applied. Keeping original title: %s", title);
            }
        }
```

Σχήμα 3.3.2: Clean Venue table 2[44].

```
            LoggerController.formattedInfo(format:"Finally curated title: %s", title);
            // Set the title
add a breakpoint    .setTitle(title.trim());
        }

        if (!venue.getAddress().isEmpty()) {
            // Normalize the string
            String address = venue.getAddress();
            address = StringUtils.capitalizeWords(address);
            LoggerController.formattedInfo(format:"START: Normalized address: %s", address);
            String possibleAddressReplacement = null;

            // Spell check
            LoggerController.formattedInfo(format:"Spell checked address: %s",
                    address + " - " + spellCheckService.isValidWord(address));
            // Remove random letter with spaces around it
            address = INVALID_LETTER_PATTERN.matcher(address).replaceAll(replacement:" ");
            // Remove any multiple spaces that might have been introduced
            address = MULTIPLE_SPACES_PATTERN.matcher(address).replaceAll(replacement:" ");

            // Entity extraction
            entities = nlpService.extractEntities(address);
            LoggerController.formattedInfo(format:"Venue Address Extracted entities: %s", entities);

            // Spell check and correction
            if (!spellCheckService.isValidWord(address)) {
                logInvalidWord(address, entity:"Venue", field:"address");
                possibleAddressReplacement = spellCheckService.autoCorrect(address);
                LoggerController.formattedInfo(format:"Venue address corrected by spell checking service to: %s",
                        possibleAddressReplacement);
                address = possibleAddressReplacement;
                venue.setAddress(address.trim());
            }
            if (spellCheckService.isEnglishText(address)) {
                LoggerController.formattedInfo(format:"English word detected: %s", address);
                possibleAddressReplacement = spellCheckService.autoCorrectEnglish(address);
                LoggerController.formattedInfo(format:"Venue address corrected by spell checking service to: %s",
                        possibleAddressReplacement);
                address = possibleAddressReplacement;
                venue.setAddress(address.trim());
            }
            if (spellCheckService.isGreekText(address)) {
                LoggerController.info(message:"Applying Levenshtein distance");
                SpellCheckResponse levenshteinResponse = greekSpellCkeckerService.checkAndSuggestSentence(address);
                String levenshteinSuggestion = levenshteinResponse.getLevenshteinSuggestion();
                LoggerController.formattedInfo(format:"Venue address corrected by Levenshtein distance to: %s",
```

Σχήμα 3.3.2: Clean Venue table 3[45].

Επεξήγηση μεθόδου cleanVenue

1. Κανονικοποίηση Τίτλου
    - Ελέγχει αν ο τίτλος περιέχει το σύμβολο & και το αντικαθιστά με τη λέξη "και".
    - Χρησιμοποιεί τη μέθοδο normalizeString για την αφαίρεση ειδικών χαρακτήρων και περιττών κενών.
    - Εφαρμόζει κεφαλαιοποίηση λέξεων με τη μέθοδο StringUtils.capitalizeWords.

2.Διόρθωση Ορθογραφικών Λαθών
    - Ελέγχει αν ο τίτλος είναι έγκυρος μέσω της υπηρεσίας spellCheckService:
    - Αν δεν είναι έγκυρος, εφαρμόζεται αυτόματη διόρθωση.
    - Ελέγχει αν ο τίτλος είναι Αγγλικός ή Ελληνικός και εφαρμόζει τις αντίστοιχες διορθώσεις μέσω:
        1.Levenshtein distance (απόσταση Levenshtein).
        2.Hamming distance (απόσταση Hamming).

3.Αλληλεπίδραση με Χρήστη
    - Παρέχεται στον χρήστη η δυνατότητα επιλογής μεταξύ των διορθώσεων (Levenshtein ή Hamming) ή χειροκίνητης εισαγωγής.

4.Εξαγωγή Οντοτήτων
    - Χρησιμοποιεί την υπηρεσία nlpService για την εξαγωγή οντοτήτων από τον τίτλο.

```
// Spell check
LoggerController.formattedInfo(format:"Spell checked address: %s",
        address + " - " + spellCheckService.isValidWord(address));
// Remove random letter with spaces around it
address = INVALID_LETTER_PATTERN.matcher(address).replaceAll(replacement:" ");
// Remove any multiple spaces that might have been introduced
address = MULTIPLE_SPACES_PATTERN.matcher(address).replaceAll(replacement:" ");

// Entity extraction
entities = nlpService.extractEntities(address);
LoggerController.formattedInfo(format:"Venue Address Extracted entities: %s", entities);
```

Σχήμα 3.3.2: NLP Entity extraction[46].

1627	310	θατρο κατω	Ηράκλειο	2	2021-01-31 10:21:26.000	[ ]

Σχήμα 3.3.2: Εικόνα πριν το curation από τον πίνακα venue[98].

1627	310	Θέατρο Κάτω	Ηράκλειο, Κρήτη	2	2021-01-31 10:21:26.000	[ ]

Σχήμα 3.3.2: Εικόνα μετα το curation από τον πίνακα venue[47].

Productions Table

1. Τυποποίηση Τίτλου (Title Standardization)
    - Αφαίρεση περιττών συμβόλων, όπως εισαγωγικά (", '), κόμματα, κ.λπ.
    - Κανονικοποίηση του τίτλου με την αφαίρεση διπλών κενών.
    - Διατήρηση μιας συνεπούς μορφής γραφής (π.χ., κεφαλαίο πρώτο γράμμα κάθε λέξης).

2. Καθαρισμός Περιγραφής (Description Cleaning)
    - Αφαίρεση ανεπιθύμητων χαρακτήρων, όπως αστέρια (*) και περιττών αλλαγών γραμμών.
    - Εντοπισμός και αφαίρεση διπλών κενών και περιττών κενού χώρου.
    - Εξασφάλιση ομαλής εμφάνισης του κειμένου για ανάγνωση.
3. Κανονικοποίηση Παραγωγής (Production Normalization)
    - Αφαίρεση ειδικών χαρακτήρων από το όνομα του production.
    - Εφαρμογή τυποποίησης για ομοιομορφία στις εγγραφές.

```java
public class ProductionCurationService {

    private static final Pattern SPECIAL_CHARACTERS_PATTERN = Pattern.compile(regex:"[^\\p{L}\\p{N}\\s]");
    private static final Pattern MULTIPLE_SPACES_PATTERN = Pattern.compile(regex:"\\s{2,}");
    private static final Pattern STAR_PATTERN = Pattern.compile(regex:"\\*");
    private static final Pattern NEWLINE_PATTERN = Pattern.compile(regex:"\\r?\\n");

    public Production cleanProduction(Production production) {
        // Clean and standardize the title
        if (production.getTitle() != null) {
            String title = production.getTitle();
            title = SPECIAL_CHARACTERS_PATTERN.matcher(title).replaceAll(replacement:""); // Remove special characters
            title = MULTIPLE_SPACES_PATTERN.matcher(title).replaceAll(replacement:" ");   // Replace multiple spaces
            title = capitalizeWords(title);                                    // Capitalize each word
            production.setTitle(title.trim());
        }

        // Clean and format the description
        if (production.getDescription() != null) {
            String description = production.getDescription();
            description = STAR_PATTERN.matcher(description).replaceAll(replacement:"");  // Remove stars (*)
            description = NEWLINE_PATTERN.matcher(description).replaceAll(replacement:" "); // Remove newlines
            description = MULTIPLE_SPACES_PATTERN.matcher(description).replaceAll(replacement:" "); // Remove extra spaces
            production.setDescription(description.trim());
        }

        // Normalize producer name
        if (production.getProducer() != null) {
            String producer = production.getProducer();
            producer = SPECIAL_CHARACTERS_PATTERN.matcher(producer).replaceAll(replacement:""); // Remove special characters
            producer = capitalizeWords(producer);                                  // Capitalize each word
            production.setProducer(producer.trim());
        }

        return production;
    }

    // Helper method to capitalize the first letter of each word
    private String capitalizeWords(String str) {
        String[] words = str.split(regex:"\\s+");
        StringBuilder capitalized = new StringBuilder();
        for (String word : words) {
            if (!word.isEmpty()) {
                capitalized.append(Character.toUpperCase(word.charAt(index:0)))
                           .append(word.substring(beginIndex:1).toLowerCase())
                           .append(str:" ");
            }
        }
```

Σχήμα 3.3.2: Clean Production table[48].

Επεξήγηση μεθόδου cleanProduction

1.Τίτλος (Title):

- Αφαιρέθηκαν εισαγωγικά και περιττά κενά.
- Τυποποιήθηκε η μορφή με κεφαλαίο πρώτο γράμμα σε κάθε λέξη.

2.Περιγραφή (Description):

- Αφαιρέθηκαν τα αστέρια (*).
- Αφαιρέθηκαν οι αλλαγές γραμμών.
- Εξασφαλίστηκε ομοιομορφία στη διάταξη των λέξεων.

3.Παραγωγή (Production):

- Αφαιρέθηκαν ειδικοί χαρακτήρες.
- Κανονικοποιήθηκε η εμφάνιση των λέξεων.

Events Table

1. Κανονικοποίηση Εύρους Τιμών (Price Range Normalization)

- Αφαίρεση μη αριθμητικών χαρακτήρων, όπως $, €, κ.λπ.
- Αφαίρεση περιττών κενών.
- Διασφάλιση συνεπούς μορφής για εύρη τιμών, π.χ., 10-20 ή 15 - 30.

2. Τυποποίηση Πεδίων Ημερομηνιών (Standardizing Date-Time Fields)

- Μετατροπή των τιμών ημερομηνίας σε ένα ομοιόμορφο πρότυπο (π.χ., yyyy-MM-dd HH:mm:ss).
- Εξασφάλιση ορθής αναγνώρισης της έναρξης και λήξης εκδήλωσης.

```java
public class EventCurationService {

    private static final Pattern NON_NUMERIC_CHARACTERS_PATTERN = Pattern.compile(regex:"[^\\d\\-\\s]");
    private static final Pattern MULTIPLE_SPACES_PATTERN = Pattern.compile(regex:"\\s{2,}");
    private static final SimpleDateFormat INPUT_DATE_FORMAT = new SimpleDateFormat(pattern:"MM/dd/yyyy HH:mm");
    private static final SimpleDateFormat STANDARD_DATE_FORMAT = new SimpleDateFormat(pattern:"yyyy-MM-dd HH:mm:ss");

    public Event cleanEvent(Event event) {
        // Normalize the price range
        if (event.getPriceRange() != null) {
            String priceRange = event.getPriceRange();
            priceRange = NON_NUMERIC_CHARACTERS_PATTERN.matcher(priceRange).replaceAll(replacement:""); // Remove non-numeric characters
            priceRange = MULTIPLE_SPACES_PATTERN.matcher(priceRange).replaceAll(replacement:" ");      // Remove extra spaces
            event.setPriceRange(priceRange.trim());
        }

        // Standardize the start date
        if (event.getDateEvent() != null) {
            String standardizedStartDate = formatDate(event.getDateEvent());
            event.setStartDate(standardizedStartDate);
        }

        // Standardize the end date
        if (event.getEndDate() != null) {
            String standardizedEndDate = formatDate(event.getEndDate());
            event.setEndDate(standardizedEndDate);
        }

        return event;
    }

    // Helper method to standardize date format
    private String formatDate(String date) {
        try {
            return STANDARD_DATE_FORMAT.format(INPUT_DATE_FORMAT.parse(date));
        } catch (ParseException e) {
            System.err.println("Error parsing date: " + date + ". Using original value.");
            return date; // Return the original value if parsing fails
        }
    }
}
```

Σχήμα 3.3.2: Clean Event table[75].

Επεξήγηση μεθόδου cleanEvent

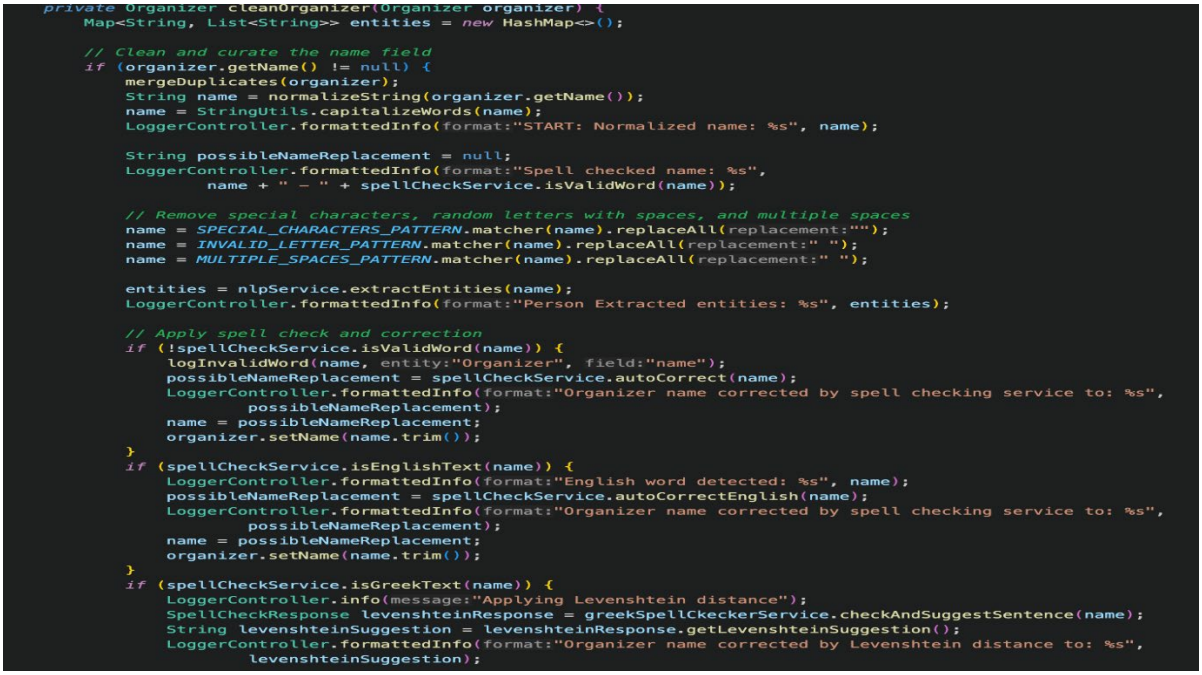
1. Εύρος Τιμών (Price Range):
    - Αφαιρέθηκαν μη αριθμητικοί χαρακτήρες (€, $).
    - Διορθώθηκαν περιττά κενά.

2. Ημερομηνίες (Date-Time Fields):
    - Μετατράπηκαν από την αρχική μορφή (MM/dd/yyyy HH:mm) στην τυποποιημένη μορφή (yyyy-MM-dd HH:mm:ss).
    - Αν αποτύχει η ανάλυση της ημερομηνίας, διατηρείται η αρχική τιμή.

Organizers Table

1. Τυποποίηση Ονομάτων (Name Standardization)
    - Μετατροπή του ονόματος σε κεφαλαία και πεζά με βάση κανόνες σωστής μορφοποίησης.
    - Αφαίρεση ειδικών χαρακτήρων και περιττών κενών.
    - Εντοπισμός και διόρθωση τυπογραφικών σφαλμάτων μέσω ελέγχου ορθογραφίας.

2. Κανονικοποίηση Διεύθυνσης και Πόλης (Address and Town Normalization)
    - Αφαίρεση περιττών χαρακτήρων, όπως εισαγωγικών (") και τόνων.
    - Χρήση regex για την εξάλειψη πολλαπλών κενών και μη έγκυρων χαρακτήρων.
    - Αν το πεδίο είναι κενό ή ελλιπές, χρησιμοποιούνται APIs NLP/LLM για την αυτόματη πρόταση διεύθυνσης ή πόλης.
3. Επαλήθευση Email και Τηλεφώνου (Email and Phone Validation)
    - Έλεγχος εγκυρότητας email μέσω προτύπων regex.
    - Έλεγχος τηλεφωνικών αριθμών για αποδοχή έγκυρης μορφής (π.χ., 10ψήφιοι ή 12ψήφιοι αριθμοί).
    - Αν κάποιο πεδίο είναι μη έγκυρο, αποθηκεύεται η τιμή "unknown".

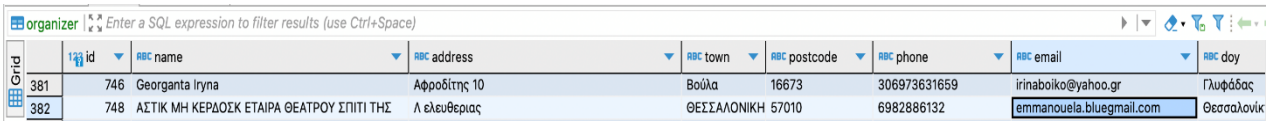


```
private Organizer cleanOrganizer(Organizer organizer) {
    Map<String, List<String>> entities = new HashMap<>();

    // Clean and curate the name field
    if (organizer.getName() != null) {
        mergeDuplicates(organizer);
        String name = normalizeString(organizer.getName());
        name = StringUtils.capitalizeWords(name);
        LoggerController.formattedInfo(format:"START: Normalized name: %s", name);

        String possibleNameReplacement = null;
        LoggerController.formattedInfo(format:"Spell checked name: %s",
                name + " - " + spellCheckService.isValidWord(name));

        // Remove special characters, random letters with spaces, and multiple spaces
        name = SPECIAL_CHARACTERS_PATTERN.matcher(name).replaceAll(replacement:"");
        name = INVALID_LETTER_PATTERN.matcher(name).replaceAll(replacement:" ");
        name = MULTIPLE_SPACES_PATTERN.matcher(name).replaceAll(replacement:" ");

        entities = nlpService.extractEntities(name);
        LoggerController.formattedInfo(format:"Person Extracted entities: %s", entities);

        // Apply spell check and correction
        if (!spellCheckService.isValidWord(name)) {
            logInvalidWord(name, entity:"Organizer", field:"name");
            possibleNameReplacement = spellCheckService.autoCorrect(name);
            LoggerController.formattedInfo(format:"Organizer name corrected by spell checking service to: %s",
                    possibleNameReplacement);
            name = possibleNameReplacement;
            organizer.setName(name.trim());
        }
        if (spellCheckService.isEnglishText(name)) {
            LoggerController.formattedInfo(format:"English word detected: %s", name);
            possibleNameReplacement = spellCheckService.autoCorrectEnglish(name);
            LoggerController.formattedInfo(format:"Organizer name corrected by spell checking service to: %s",
                    possibleNameReplacement);
            name = possibleNameReplacement;
            organizer.setName(name.trim());
        }
        if (spellCheckService.isGreekText(name)) {
            LoggerController.info(message:"Applying Levenshtein distance");
            SpellCheckResponse levenshteinResponse = greekSpellCkeckerService.checkAndSuggestSentence(name);
            String levenshteinSuggestion = levenshteinResponse.getLevenshteinSuggestion();
            LoggerController.formattedInfo(format:"Organizer name corrected by Levenshtein distance to: %s",
                    levenshteinSuggestion);
```

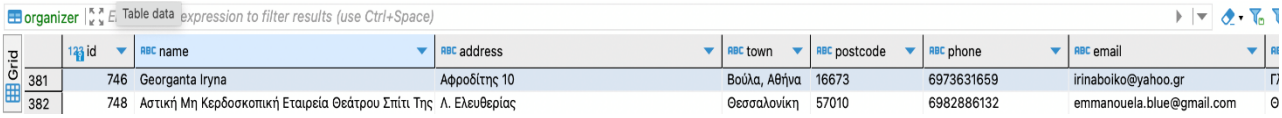
Σχήμα 3.3.2: Clean Organizer table[49].

```
    // Clean and curate the town field
    if (organizer.getTown() != null) {
        organizer.setTown(normalizeString(organizer.getTown()));
        String town = normalizeString(organizer.getTown());
        entities = nlpService.extractEntities(town);
        LoggerController.formattedInfo(format:"Organizer Town Extracted entities: %s", entities);

        String possibleTownReplacement = null;
        if (!spellCheckService.isValidWord(town)) {
            logInvalidWord(town, entity:"Organizer", field:"town");
            possibleTownReplacement = spellCheckService.autoCorrect(town);
            LoggerController.formattedInfo(format:"Organizer town corrected by spell checking service to: %s",
                    possibleTownReplacement);
            town = possibleTownReplacement;
            organizer.setTown(town.trim());
        }

        if (spellCheckService.isGreekText(town)) {
            LoggerController.info(message:"Applying Levenshtein distance");
            SpellCheckResponse levenshteinResponse = greekSpellCkeckerService.checkAndSuggestSentence(town);
            String levenshteinSuggestion = levenshteinResponse.getLevenshteinSuggestion();
            LoggerController.formattedInfo(format:"Organizer town corrected by Levenshtein distance to: %s",
                    levenshteinSuggestion);
            town = levenshteinSuggestion;
        }
        organizer.setTown(StringUtils.capitalizeWords(town.trim()));
    }

    // Validate and clean other fields
    if (organizer.getEmail() != null && !validateEmail(organizer.getEmail())) {
        organizer.setEmail(email:"unknown");
    }
    if (organizer.getPhone() != null && !validatePhoneNumber(organizer.getPhone())) {
        organizer.setPhone(phone:"unknown");
    }
    if (organizer.getDoy() != null) {
        organizer.setDoy(normalizeAddressString(organizer.getDoy()).trim().toUpperCase());
    }

    return organizer;
}
```

Σχήμα 3.3.2: Clean Organizer table town field[50].



Σχήμα 3.3.2: Εικόνα πριν το curation από τον πίνακα organizer[51].



Σχήμα 3.3.2: Εικόνα μετά το curation από τον πίνακα organizer[52].

Επεξήγηση μεθόδου cleanOrganizer

Ο κύριος στόχος της μεθόδου cleanOrganizer είναι να επιμεληθεί και να καθαρίσει κρίσιμα πεδία της οντότητας Organizer, συμπεριλαμβανομένων των name, address, town, email, phone και doy. Αυτά τα πεδία συχνά περιέχουν ασυνέπειες, τυπογραφικά λάθη ή ελλιπή δεδομένα. Με την εφαρμογή πολλαπλών επιπέδων επεξεργασίας, η μέθοδος διασφαλίζει ότι τα δεδομένα είναι σωστά δομημένα, χωρίς σφάλματα και έτοιμα για ενσωμάτωση σε συστήματα downstream.

1. Κανονικοποίηση:

- Το πεδίο name καθαρίζεται μέσω κανονικοποίησης συμβολοσειρών, αφαιρώντας ανεπιθύμητους χαρακτήρες και διαμορφώνοντας το κείμενο με συνεπή μορφοποίηση. Μέσω κανονικών εκφράσεων (regex), απομακρύνονται ειδικοί χαρακτήρες και άκυρα γράμματα.
- Το όνομα επαληθεύεται μέσω υπηρεσίας ορθογραφικού ελέγχου. Εάν εντοπιστούν λάθη, εφαρμόζονται προτάσεις αυτόματης διόρθωσης, όπως ορθογραφικές διορθώσεις με βάση τους αλγορίθμους Levenshtein και Hamming. Η επαλήθευση γίνεται διασταυρώνοντας το text input με αρχεία dictionary(el_GR.dic) τα οποία λειτουργούν ως λεξικές βιβλιοθήκες της ελληνικής γλώσσας.
- Η μέθοδος χρησιμοποιεί υπηρεσίες NLP για να εξάγει ουσιαστικές πληροφορίες (π.χ. ονόματα προσώπων ή ρόλων) από το πεδίο name.

2. Καθαρισμός και Σημασιολογική Ανάλυση Διεύθυνσης

- Το πεδίο address κανονικοποιείται, κεφαλαιοποιείται και επαληθεύεται ως προς την ορθογραφία του. Εφαρμόζονται τεχνικές NLP για την εξαγωγή γεωγραφικών ή οργανωτικών οντοτήτων από το κείμενο, ενώ παράλληλα γίνεται αυτόματη διόρθωση λαθών με βάση προτεινόμενες τροποποιήσεις.

3. Επεξεργασία Πεδίου Πόλης

- Το πεδίο town υποβάλλεται σε κανονικοποίηση και εξαγωγή οντοτήτων. Σε περίπτωση λαθών, εφαρμόζονται ορθογραφικές διορθώσεις με βάση τον Levenshtein.

4. Επικύρωση Επικοινωνιακών Πληροφοριών

- Email: Εάν το email αποδειχθεί άκυρο, αντικαθίσταται από την τιμή unknown.
- Τηλέφωνο: Μη έγκυροι αριθμοί τηλεφώνου αντικαθίστανται με προκαθορισμένη τιμή.

5. Κανονικοποίηση Πεδίου DOY (Εφορία)

- Το πεδίο doy κανονικοποιείται και μετατρέπεται σε κεφαλαία γράμματα, διασφαλίζοντας τη συνέπεια και τη συμμόρφωση με τα πρότυπα μορφοποίησης.

```java
public class DoyValidator {
    private static final String DOY_LIST_PATH = "./doy/doy_list.csv"; // Path to the file containing DOY li
    private static Set<String> validDoySet;

    // Load the DOY list into a Set for validation.
    public static void loadDoyList() {
        validDoySet = new HashSet<>();
        try (BufferedReader br = new BufferedReader(new FileReader(DOY_LIST_PATH))) {
            validDoySet = br.lines()
                    .skip(n:1)
                    .map(String::trim)
                    .map(String::toUpperCase)
                    .collect(Collectors.toSet());
        } catch (IOException e) {
            throw new RuntimeException("Failed to load DOY list from file: " + DOY_LIST_PATH, e);
        }
    }

    // Validate the DOY field.
    public static boolean isValidDoy(String doy) {
        if (validDoySet == null) {
            loadDoyList();
        }
        return doy != null && validDoySet.contains(doy.trim().toUpperCase());
    }
}
```

Σχήμα 3.3.2: Επικύρωση πεδίου Δ.Ο.Υ για τον πίνακα organizer[53].

Συλλέγοντας τα δεδομένα Δ.Ο.Υ από την Ανεξάρτητη Αρχή Δημοσίων Εσόδων χρησιμοποιώντας το αρχείο ΕπικΥπηρ1d.pdf (aade.gr) αποθηκεύουμε τη λίστα σε ένα αρχείο CSV. Η μέθοδος loadDoyList διαβάζει τη λίστα στη μνήμη όταν χρειάζεται. Αυτό εξασφαλίζει αποτελεσματική επικύρωση(validation) χωρίς περιττές λειτουργίες I/O.

Επικύρωση Δεδομένων (Data Validation)

Η διαδικασία επικύρωσης δεδομένων (Data Validation) διασφαλίζει την ακρίβεια, την ποιότητα και τη συνέπεια των πληροφοριών που αποθηκεύονται στο σύστημα. Σε εφαρμογές όπου τα δεδομένα προέρχονται από πολλαπλές πηγές ή καταχωρούνται χειροκίνητα, η επικύρωση είναι απαραίτητη για την αποτροπή σφαλμάτων, την εξάλειψη διπλότυπων και τη διατήρηση της συνοχής. Στο πλαίσιο της παρούσας εργασίας, η διαδικασία επικύρωσης περιλαμβάνει τρεις βασικές πτυχές: τον έλεγχο ορθογραφίας και τη διόρθωση κειμένων, την αντιστοίχιση οντοτήτων και την αφαίρεση διπλότυπων εγγραφών.

Ο έλεγχος ορθογραφίας αποτελεί ένα από τα πρώτα στάδια στην επικύρωση δεδομένων που περιέχουν κείμενα, όπως ονόματα, διευθύνσεις και περιγραφές. Για την εφαρμογή αυτής της λειτουργίας, αξιοποιούνται μέθοδοι(functions) ελέγχου ορθογραφίας, τα οποία χρησιμοποιούν γλωσσικά μοντέλα για την αναγνώριση και διόρθωση τυπογραφικών λαθών ή λανθασμένων τιμών. Στη συγκεκριμένη εργασία, οι λέξεις που εντοπίζονται ως εσφαλμένες συγκρίνονται με λεξικά της ελληνικής και αγγλικής γλώσσας. Εάν μια λέξη θεωρηθεί άκυρη, εφαρμόζονται διορθωτικοί αλγόριθμοι όπως η αυτόματη πρόβλεψη και η διόρθωση με βάση την απόσταση Levenshtein.

Η διαδικασία ελέγχου ορθογραφίας είναι ιδιαίτερα σημαντική για πεδία όπως το όνομα του διοργανωτή ή οι τοποθεσίες. Για παράδειγμα, μια τυπογραφική απόκλιση σε ένα όνομα ("Γιωργος" αντί "Γιώργος") μπορεί να οδηγήσει σε εσφαλμένες συνδέσεις μεταξύ οντοτήτων. Παράλληλα, η χρήση εργαλείων επεξεργασίας φυσικής γλώσσας (NLP) βοηθά στην αναγνώριση οντοτήτων από μη δομημένα δεδομένα, επιτρέποντας την περαιτέρω διόρθωση και την εμπλουτισμό.

```
// Levenshtein distance algorithm
private int calculateLevenshteinDistance(String word1, String word2) {
    int[][] dp = new int[word1.length() + 1][word2.length() + 1];

    for (int i = 0; i <= word1.length(); i++) {
        for (int j = 0; j <= word2.length(); j++) {
            if (i == 0) {
                dp[i][j] = j;
            } else if (j == 0) {
                dp[i][j] = i;
            } else {
                dp[i][j] = min(
                        dp[i - 1][j - 1] + costOfSubstitution(word1.charAt(i - 1), word2.charAt(j - 1)),
                        dp[i - 1][j] + 1,
                        dp[i][j - 1] + 1
                );
            }
        }
    }
    return dp[word1.length()][word2.length()];
}
```

Σχήμα 3.3.2: Levenshtein algorithm[54].

Η αντιστοίχιση οντοτήτων είναι ένα σημαντικό βήμα για τη διατήρηση της συνέπειας μεταξύ διαφορετικών πινάκων στη βάση δεδομένων. Στο σύστημα που αναπτύχθηκε, οι οντότητες όπως οι συνεισφορές ατόμων, οι ρόλοι και οι παραγωγές συνδέονται μεταξύ τους μέσω μοναδικών αναγνωριστικών. Ωστόσο, σφάλματα στην καταχώρηση δεδομένων, όπως μικρές αποκλίσεις στα ονόματα ή οι πολλαπλές εγγραφές για την ίδια οντότητα, μπορούν να οδηγήσουν σε εσφαλμένες αντιστοιχίσεις.

Για την αντιμετώπιση αυτών των ζητημάτων, εφαρμόζονται αλγόριθμοι Entity Matching, οι οποίοι συγκρίνουν εγγραφές μεταξύ πινάκων με βάση συγκεκριμένα κριτήρια (π.χ., ονόματα, ημερομηνίες, ρόλους). Ένα βασικό παράδειγμα είναι η αντιστοίχιση των ονομάτων συντελεστών με τους ρόλους τους. Εάν δύο εγγραφές έχουν παρόμοια χαρακτηριστικά αλλά μικρές διαφορές, το σύστημα μπορεί να χρησιμοποιήσει αλγόριθμους σύγκρισης κειμένου, όπως η απόσταση Hamming ή η απόσταση Jaccard, για να αποφασίσει αν πρόκειται για την ίδια οντότητα. Αυτό διασφαλίζει ότι όλες οι συνεισφορές ατόμων συνδέονται σωστά με τις αντίστοιχες παραγωγές.

```java
// Hamming distance algorithm
public int calculateHammingDistance(String word1, String word2) {
    if (word1.length() != word2.length()) {
        throw new IllegalArgumentException(s:"Word lengths must be equal");
    }
    int distance = 0;
    for (int i = 0; i < word1.length(); i++) {
        if (word1.charAt(i) != word2.charAt(i)) {
            distance++;
        }
    }
    return distance;
}

// Method to find the closest matching word in the dictionary using Hamming distance
public String hammingDistanceCalculate(String word) {
    String closestWord = null;
    int minDistance = Integer.MAX_VALUE;

    for (String dictWord : dictionaryWords) {
        if (dictWord.length() == word.length()) {
            int distance = calculateHammingDistance(word, dictWord);
            if (distance < minDistance) {
                minDistance = distance;
                closestWord = dictWord;
            }
        }
    }
    return closestWord;
}
```

Σχήμα 3.3.2: Hamming algorithm[64].

Η ύπαρξη διπλότυπων εγγραφών σε πίνακες δεδομένων είναι ένα συχνό πρόβλημα που μπορεί να προκαλέσει ασυνέπειες και να επηρεάσει αρνητικά την ανάλυση και την επεξεργασία των δεδομένων. Στο σύστημα που αναπτύχθηκε, εφαρμόζεται μια διαδικασία ανίχνευσης και συγχώνευσης διπλότυπων εγγραφών, η οποία βασίζεται σε κανονικές εκφράσεις, συγκρίσεις τιμών και κανόνες προτεραιότητας.
Για παράδειγμα, στον πίνακα Roles, δύο εγγραφές που περιγράφουν τον ίδιο ρόλο ("Σκηνοθέτης" και "Σκηνοθέτης ") ενδέχεται να θεωρηθούν διαφορετικές λόγω μικρών αποκλίσεων στη μορφοποίηση. Ο καθαρισμός αυτών των εγγραφών πραγματοποιείται μέσω κανονικοποίησης συμβολοσειρών, αφαίρεσης περιττών χαρακτήρων και συγκρίσεων βάσει αλγορίθμων. Εάν δύο εγγραφές θεωρηθούν διπλότυπες, συγχωνεύονται σε μία, και τα δεδομένα τους ενοποιούνται. Επιπλέον, στους πίνακες Organizers και Venues, τα διπλότυπα εντοπίζονται και διαγράφονται με τη χρήση μοναδικών ταυτοποιητικών (unique identifiers). Αυτή η διαδικασία μειώνει τον όγκο των δεδομένων και βελτιώνει την ακρίβεια των συνδέσεων μεταξύ πινάκων.

Η διαδικασία επικύρωσης δεδομένων είναι θεμελιώδης για τη διασφάλιση της ποιότητας και της αξιοπιστίας μιας βάσης δεδομένων. Οι τεχνικές που περιγράφονται, όπως ο έλεγχος ορθογραφίας, η αντιστοίχιση οντοτήτων και η αφαίρεση διπλότυπων, επιτρέπουν τη βελτίωση της συνέπειας, την αποφυγή σφαλμάτων και την εξάλειψη περιττών πληροφοριών. Σε ένα σύστημα που βασίζεται στην επιμέλεια

δεδομένων, η επικύρωση συμβάλλει καθοριστικά στην αποτελεσματική διαχείριση και αξιοποίηση των δεδομένων.

Συμπλήρωση Διευθύνσεων με AI (LLM-Based Address Autocomplete)

Η συμπλήρωση διευθύνσεων για τον πίνακα Venues είναι μια κοινή πρόκληση στη διαχείριση δεδομένων, καθώς οι καταχωρήσεις διευθύνσεων συχνά περιέχουν ελλείψεις ή τυπογραφικά λάθη. Η χρήση Μεγάλων Γλωσσικών Μοντέλων (LLMs), όπως αυτά της OpenAI και της Meta(Ollama model) επιτρέπει την αυτόματη συμπλήρωση ελλιπών πεδίων, παρέχοντας ακριβείς και πλήρεις διευθύνσεις.

Η διαδικασία βασίζεται στη χρήση APIs που αναλύουν το υπάρχον κείμενο και προτείνουν την πιο πιθανή ολοκλήρωση. Για παράδειγμα, μια ελλιπής καταχώρηση όπως "Ερμού 15, Α" μπορεί να ολοκληρωθεί ως "Ερμού 15, Αθήνα, 10563". Το LLM λαμβάνει υπόψη το πλαίσιο, όπως την πόλη ή τον ταχυδρομικό κώδικα, για να προτείνει μια κατάλληλη συμπλήρωση. Επίσης, το μοντέλο μπορεί να διορθώσει ορθογραφικά λάθη ή να αντικαταστήσει μη τυποποιημένες λέξεις με τυπικούς όρους. Αυτή η λειτουργικότητα βελτιώνει την ποιότητα των δεδομένων, εξασφαλίζοντας ότι οι διευθύνσεις είναι έτοιμες για χρήση σε downstream εφαρμογές, όπως η απεικόνιση σε χάρτες ή η αποστολή ειδοποιήσεων.

```java
@GetMapping("/openai")
@Retryable(value = { TimeoutException.class }, maxAttempts = 3, backoff = @Backoff(delay = 2000))
public String getOpenAIresponse(@RequestParam String title) {
    try {
        String message = """
                Ποιά είναι η ακριβής διεύθυνση του θεάτρου ή Κινηματογράφου {title} στην Ελλάδα;
                Παρακαλώ δώστε την απάντηση στη μορφή: "Η ακριβής διεύθυνση του {title} είναι [ΔΙΕΥΘΥΝΣΗ]."
                """;
        PromptTemplate template = new PromptTemplate(message);
        Prompt prompt = template.create(Map.of(k1:"title", title));
        LoggerController.formattedInfo("Prompt sent to OpenAI API: " + prompt.toString());

        ChatResponse response = openAiChatClient.prompt(prompt).call().chatResponse();
        String fullResponse = response.getResult().getOutput().getContent();
        LoggerController.formattedInfo("Response received from OpenAI API: " + fullResponse);

        String extractedAddress = extractAddress(fullResponse);
        LoggerController.formattedInfo("Extracted Address: " + extractedAddress);
        return extractedAddress;
    } catch (TimeoutException e) {
        LoggerController.formattedError(format:"Timeout Exception: ", e.getMessage());
        String errorResponse = "Response timed out. Please try again later.";
        return errorResponse;
    }
}
```

Σχήμα 3.3.2: Χρήση LLM για συμπλήρωση διεύθυνσης[55].

Ανάθεση Υπο-Ρόλων σε Contirbutions(SubRole Assignment)

Η ανάθεση subRole σε συνεισφορές (Contributions) αποτελεί μια σύνθετη εργασία, καθώς συχνά απαιτείται η ανάλυση πολλών πηγών δεδομένων για να εντοπιστεί ο ακριβής ρόλος ενός ατόμου σε μια παραγωγή. Με τη χρήση AI, το σύστημα μπορεί να αναλύσει τα υπάρχοντα δεδομένα για το Person και την Production και να προτείνει αυτόματα subRoles. Για παράδειγμα, αν ένας συντελεστής καταχωρηθεί ως "Μουσικός" σε μια παραγωγή, το AI μπορεί να αναγνωρίσει το μουσικό όργανο που παίζει (π.χ., "Βιολιστής") από άλλες πηγές δεδομένων, όπως περιγραφές παραστάσεων ή το βιογραφικό του. Αυτό επιτυγχάνεται με τη χρήση αλγορίθμων NLP που αναλύουν το σχετικό κείμενο, εντοπίζουν μοτίβα και δημιουργούν υποθέσεις για την ανάθεση ρόλων. Η δυνατότητα αυτή μειώνει τον χρόνο χειροκίνητης επεξεργασίας, ενώ ταυτόχρονα ενισχύει την ακρίβεια και την κατηγοριοποίηση των δεδομένων συνεισφορών.

```java
public class ContributionRoleAssigner {
    public String identifySpecificRole(String contributorName, String productionDescription, String contributorBio) {
        // Combine relevant text for analysis
        StringBuilder combinedText = new StringBuilder();
        if (productionDescription != null && !productionDescription.isEmpty()) {
            combinedText.append(productionDescription).append(str:" ");
        }
        if (contributorBio != null && !contributorBio.isEmpty()) {
            combinedText.append(contributorBio).append(str:" ");
        }

        // Analyze the text using the NLP service
        Map<String, List<String>> extractedEntities = nlpService.extractEntities(combinedText.toString());

        // Search for relevant roles in the extracted entities
        String specificRole = null;
        if (extractedEntities.containsKey(key:"Role")) {
            List<String> roles = extractedEntities.get(key:"Role");
            for (String role : roles) {
                if (role.toLowerCase().contains(contributorName.toLowerCase())) {
                    specificRole = role;
                    break;
                }
            }
        }

        // Fallback if no specific role is identified
        if (specificRole == null) {
            specificRole = "General " + (extractedEntities.containsKey(key:"Role") ? extractedEntities.get(key:"Role").get(index:0) : "Performer");
        }

        return specificRole;
```

Σχήμα 3.3.2: Χρήση AI για συμπλήρωση διεύθυνσης[56].

Ανάθεση Υπο-Ρόλων σε Contirbutions(SubRole Assignment)

Η αναγνώριση και επαλήθευση οντοτήτων είναι κρίσιμη για την εξασφάλιση της συνέπειας και της ποιότητας των δεδομένων. Η ενσωμάτωση τεχνικών NLP επιτρέπει στο σύστημα να εξάγει αυτόματα οντότητες, όπως ονόματα, τοποθεσίες και ημερομηνίες, από μη δομημένο κείμενο.
Για παράδειγμα, σε περιγραφές παραστάσεων, το σύστημα μπορεί να αναγνωρίσει ονόματα συντελεστών, τοποθεσίες των παραστάσεων και ημερομηνίες διεξαγωγής. Οι εξαγόμενες οντότητες συγκρίνονται με τα υπάρχοντα δεδομένα για να εντοπιστούν τυχόν ασυμφωνίες ή λείποντα στοιχεία. Εάν το σύστημα

ανιχνεύσει μια ανακολουθία, όπως δύο διαφορετικά ονόματα για τον ίδιο συντελεστή, προτείνει διορθώσεις ή επισημαίνει τις εγγραφές για περαιτέρω έλεγχο.
Η τεχνολογία NLP υποστηρίζει επίσης την επαλήθευση δεδομένων με τη χρήση εξωτερικών πηγών, όπως online λεξικά ή γνωστές βάσεις δεδομένων (π.χ., Wikidata). Αυτή η λειτουργία επιτρέπει την επαλήθευση της ορθότητας των ονομάτων και την ανίχνευση πιθανών λαθών ή ψευδών στοιχείων.

Κεφάλαιο 4ο: Χρήση Μεγάλων Γλωσσικών Μοντέλων και NLP

4.1 Εισαγωγή

Από τη δεκαετία του 1980, τα γλωσσικά μοντέλα (Language Models - LMs) υπάρχουν για περισσότερες από τέσσερις δεκαετίες ως μέσο στατιστικής μοντελοποίησης των ιδιοτήτων που παρατηρούνται στη φυσική γλώσσα. Δεδομένης μιας συλλογής κειμένων ως είσοδο, ένα γλωσσικό μοντέλο υπολογίζει στατιστικές ιδιότητες της γλώσσας από αυτά τα κείμενα, όπως συχνότητες και πιθανότητες λέξεων και του περιβάλλοντος τους, οι οποίες μπορούν στη συνέχεια να χρησιμοποιηθούν για διάφορους σκοπούς, όπως κατανόηση της φυσικής γλώσσας (Natural Language Understanding - NLU), παραγωγή (Natural Language Generation - NLG), συλλογιστική (Natural Language Reasoning - NLR) και, γενικότερα, επεξεργασία (Natural Language Processing - NLP).

Αυτή η στατιστική προσέγγιση για τη μοντελοποίηση της φυσικής γλώσσας έχει προκαλέσει για δεκαετίες συζητήσεις μεταξύ αυτών που υποστηρίζουν ότι η γλώσσα μπορεί να μοντελοποιηθεί μέσω της παρατήρησης και της αναπαράστασης μοτίβων, και αυτών που υποστηρίζουν ότι μια τέτοια προσέγγιση είναι στοιχειώδης και ότι η σωστή κατανόηση της γλώσσας χρειάζεται θεμελίωση στις γλωσσολογικές θεωρίες.

Μόνο πρόσφατα, ως συνέπεια της αυξημένης διαθεσιμότητας συλλογών κειμένων και της πρόσβασης σε βελτιωμένους υπολογιστικούς πόρους, τα μεγάλα γλωσσικά μοντέλα (Large Language Models - LLMs) εισήχθησαν στην επιστημονική κοινότητα, φέρνοντας επανάσταση στον τομέα της επεξεργασίας φυσικής γλώσσας (NLP). Ακολουθώντας την ίδια θεμελιώδη διαίσθηση με τα παραδοσιακά γλωσσικά μοντέλα (LMs) που εισήχθησαν τη δεκαετία του 1980, τα LLMs κλιμακώνουν τις στατιστικές γλωσσικές ιδιότητες που προέρχονται από μεγάλες συλλογές κειμένων.

Σύμφωνα με την ίδια λογική της στατιστικής μοντελοποίησης των γλωσσικών ιδιοτήτων όπως στα παραδοσιακά LMs, οι ερευνητές έχουν αποδείξει ότι, με τους σύγχρονους υπολογιστικούς πόρους, είναι εφικτό να εκπαιδευτούν πολύ μεγαλύτερα LLMs, τα οποία βασίζονται σε τεράστιες συλλογές κειμένων, που σε ορισμένες περιπτώσεις μπορεί να περιλαμβάνουν σχεδόν ολόκληρο τον ιστό(web). Ωστόσο, αυτή η προσέγγιση δεν είναι χωρίς αντιπαραθέσεις, ιδίως επειδή η χρήση τέτοιων μεγάλης κλίμακας συλλογών κειμένων δίνει προτεραιότητα στην ποσότητα έναντι της ποιότητας. Πράγματι, κάποιος χάνει τον έλεγχο

των δεδομένων που τροφοδοτούνται στο μοντέλο όταν χρησιμοποιείται ολόκληρος ο ιστός, ο οποίος, εκτός από πολύτιμες πληροφορίες, περιέχει επίσης προσβλητικό περιεχόμενο και παραπληροφόρηση.

Η άνοδος των μεγάλων γλωσσικών μοντέλων (LLMs) υπήρξε σταδιακή από τα τέλη της δεκαετίας του 2010 και εξελίχθηκε σε κύματα. Αρχικά, ένα κύμα εισήγαγε μοντέλα ενσωμάτωσης λέξεων, όπως το word2vec και το GloVe, για τη συμπαγή αναπαράσταση λέξεων υπό τη μορφή ενσωματώσεων (embeddings). Το πρώτο μεγάλο κύμα ήρθε με την εμφάνιση LLMs που βασίζονται στην αρχιτεκτονική Transformer, όπως τα BERT, RoBERTa και T5.

Ένα πιο πρόσφατο κύμα οδήγησε σε έκρηξη μοντέλων για γενετική τεχνητή νοημοσύνη, όπως τα chatbots ChatGPT, Google Bard, καθώς και εναλλακτικές λύσεις ανοικτού κώδικα(open source), όπως τα LLaMa, Alpaca και Lemur. Αυτά, με τη σειρά τους, ενέπνευσαν τη δημιουργία διαφορετικών τρόπων αξιοποίησης αυτών των LLMs, συμπεριλαμβανομένων μεθόδων προτροπής (prompting methods), όπως η Pattern Exploiting Training (PET) για ταξινόμηση κειμένου με λίγα παραδείγματα (few-shot learning), καθώς και μεθόδων για την παραγωγή φυσικής γλώσσας (NLG).

Ένα LLM είναι συνήθως ένα μοντέλο που έχει προεκπαιδευτεί σε υπάρχοντα μεγάλης κλίμακας σύνολα δεδομένων, μια διαδικασία που απαιτεί σημαντική υπολογιστική ισχύ και χρόνο. Ωστόσο, αυτά τα μοντέλα μπορούν στη συνέχεια να υποβληθούν σε fine-tuning για συγκεκριμένους τομείς με μικρότερη προσπάθεια, επιτρέποντας την προσαρμογή τους σε εξειδικευμένες εφαρμογές.

4.2 Περιορισμοί και προκλήσεις

Η επιτυχία των μεγάλων γλωσσικών μοντέλων (LLMs) δεν είναι χωρίς προβλήματα, γεγονός που διαμορφώνει τη συνεχιζόμενη έρευνα στον τομέα της επεξεργασίας φυσικής γλώσσας (NLP) και ανοίγει νέους δρόμους για περαιτέρω έρευνα με στόχο τη βελτίωση αυτών των μοντέλων. Ακολουθούν ορισμένοι από τους βασικούς περιορισμούς των LLMs που απαιτούν περαιτέρω διερεύνηση.

4.2.1 Black box models

Μετά την κυκλοφορία του πρώτου μεγάλου chatbot συστήματος που βασίζεται σε LLM και απέκτησε ευρεία δημοτικότητα, του ChatGPT από την OpenAI, προέκυψαν ανησυχίες σχετικά με τη φύση του συστήματος. Πράγματι, δεν υπάρχει διαθέσιμη δημόσια πληροφόρηση για το πώς υλοποιήθηκε το ChatGPT, καθώς και για τα δεδομένα που χρησιμοποιήθηκαν για την εκπαίδευση του μοντέλου. Από την οπτική γωνία των ερευνητών NLP, αυτό εγείρει σοβαρές ανησυχίες σχετικά με τη διαφάνεια και την αναπαραγωγιμότητα ενός τέτοιου μοντέλου, όχι μόνο επειδή δεν είναι γνωστό τι συμβαίνει μέσα στο μοντέλο, αλλά και επειδή αυτό εμποδίζει την αναπαραγωγή των αποτελεσμάτων.

Εάν κάποιος διεξάγει πειράματα χρησιμοποιώντας το ChatGPT σε μια συγκεκριμένη ημερομηνία, δεν υπάρχει καμία εγγύηση ότι κάποιος άλλος θα μπορέσει να αναπαράγει αυτά τα αποτελέσματα σε μεταγενέστερη ημερομηνία ενδεχομένως ακόμα και την ίδια ημερομηνία, γεγονός που μειώνει την εγκυρότητα, την πιθανή επίδραση και τη δυνατότητα γενίκευσης της έρευνας που βασίζεται στο ChatGPT.

Για να μετριαστεί αυτός ο αντίκτυπος και να αυξηθεί η κατανόησή μας για τα μοντέλα "μαύρα κουτιά" όπως το ChatGPT, οι ερευνητές έχουν αρχίσει να διερευνούν μεθόδους για την αντίστροφη μηχανική αυτών των μοντέλων, για παράδειγμα, προσπαθώντας να ανακαλύψουν ποια δεδομένα μπορεί να έχουν χρησιμοποιηθεί για την εκπαίδευση του μοντέλου.

Ευτυχώς, ωστόσο, υπάρχει μια πρόσφατη άνοδος ανοιχτού κώδικα μοντέλων στην επιστημονική κοινότητα του NLP, η οποία έχει οδηγήσει στην κυκλοφορία μοντέλων όπως το LLaMa 2 του Facebook και το Alpaca του Stanford, καθώς και πολυγλωσσικών μοντέλων όπως το BLOOM. Πρόσφατες μελέτες έχουν επίσης δείξει ότι η απόδοση αυτών των εναλλακτικών μοντέλων ανοιχτού κώδικα είναι συχνά συγκρίσιμη με εκείνη κλειστών μοντέλων, όπως το ChatGPT.

4.2.2 Κίνδυνος “μόλυνσης” των δεδομέων

Η μόλυνση δεδομένων συμβαίνει όταν τα σύνολα δεδομένων δοκιμής downstream καταλήγουν στο προ-εκπαιδευτικό σύνολο. Όταν ένα LLM που έχει εκπαιδευτεί σε μεγάλες συλλογές κειμένων έχει ήδη δει τα δεδομένα που του δίνονται κατά τη φάση της δοκιμής για αξιολόγηση, το μοντέλο ενδέχεται να παρουσιάσει εντυπωσιακή αλλά μη ρεαλιστική απόδοση. Οι έρευνες έχουν δείξει ότι η μόλυνση δεδομένων μπορεί να είναι συχνή και να έχει σημαντικό αντίκτυπο. Είναι επομένως κρίσιμο οι ερευνητές να διασφαλίζουν ότι τα δεδομένα δοκιμής δεν έχουν ήδη χρησιμοποιηθεί για την εκπαίδευση ενός LLM, προκειμένου να επιτευχθεί δίκαιη και ρεαλιστική αξιολόγηση. Αυτό ωστόσο είναι ιδιαίτερα δύσκολο αν όχι σχεδόν αδύνατο να επιβεβαιωθεί με μοντέλα "μαύρα κουτιά". Αυτό το γεγονός ενισχύει περαιτέρω την ανάγκη χρήσης ανοιχτού κώδικα και διαφανών LLMs.

4.2.3 Δημιουργία προσβλητικού περιεχομένου

Οι προκαταλήψεις στα LLMs μερικές φορές εντείνονται σε σημείο που να παράγεται περιεχόμενο που μπορεί να θεωρηθεί προσβλητικό. Η έρευνα προς αυτή την κατεύθυνση εξετάζει πώς μπορεί να γίνει καλύτερη επιμέλεια των δεδομένων εκπαίδευσης που χρησιμοποιούνται στα LLMs, ώστε να αποφευχθεί η μάθηση προσβλητικών παραδειγμάτων, καθώς και πώς να προκαλείται η παραγωγή επιβλαβών κειμένων για να κατανοηθεί η προέλευσή τους. Αυτή η έρευνα συνδέεται στενά με το ζήτημα των προκαταλήψεων και της δικαιοσύνης στα LLMs και ως εκ τούτου και τα δύο μπορούν να μελετηθούν από κοινού, με στόχο τη μείωση των προκαταλήψεων και των επιβλαβών αποτελεσμάτων.

4.3 Ιδιοτικότητα

Τα LLMs μπορούν επίσης να αποθηκεύουν ευαίσθητες πληροφορίες που προέρχονται από τα δεδομένα εκπαίδευσής τους. Παρόλο που αυτές οι πληροφορίες κωδικοποιούνται σε ενσωματώσεις (embeddings) που δεν μπορεί να γίνει ανάγνωση από ανθρώπους έχει βρεθεί ότι ένας κακόβουλος χρήστης μπορεί να εφαρμόσει τεχνικές αντίστροφης μηχανικής (reverse engineering) για να ανακτήσει τις ευαίσθητες πληροφορίες κάτι που μπορεί να έχει καταστροφικές συνέπειες για τα αντίστοιχα άτομα. Αν και η έρευνα που εξετάζει αυτές τις ευπάθειες των LLMs βρίσκεται ακόμη σε αρχικό στάδιο, υπάρχει

αυξανόμενη συνειδητοποίηση της επείγουσας ανάγκης για τέτοιου είδους έρευνα, ώστε τα LLMs να γίνουν ανθεκτικά σε επιθέσεις κατά της ιδιωτικότητας.

4.4 Model hallucinations

Οι αποκρίσεις και τα παραγόμενα αποτελέσματα από τα LLMs συχνά αποκλίνουν από την κοινή λογική. Για παράδειγμα ένα παραγόμενο κείμενο μπορεί να ξεκινά συζητώντας ένα συγκεκριμένο θέμα και στη συνέχεια να μεταβαίνει σε ένα άσχετο θέμα το οποίο μπορεί ακόμη και να αναφέρει λανθασμένα γεγονότα. Η παραίσθηση των LLMs έχει οριστεί ως «η παραγωγή περιεχομένου που αποκλίνει από τα πραγματικά γεγονότα, οδηγώντας σε μη αξιόπιστα αποτελέσματα» (Maynez et al., 2020; Rawte et al., 2023). Οι προσπάθειες για την καλύτερη κατανόηση της παραίσθησης των μοντέλων εστιάζουν σε διάφορες εργασίες, όπως η ανίχνευση, η εξήγηση και η μείωση του φαινομένου. Μέχρι σήμερα, έχουν προταθεί κάποιες αρχικές λύσεις, όπως η Παραγωγή με Υποστήριξη Ανάκτησης (Retrieval-Augmented Generation - RAG).

4.5 NLP Τεχνικές

Χάρη στις βελτιώσεις στην NLP έχει σημειωθεί μια τεράστια αλλαγή από τη συμβατική χειροκίνητη κωδικοποίηση προς την αυτοματοποίηση της συλλογής δεδομένων, του καθαρισμού δεδομένων και της στατιστικής ανάλυσης. Ουσιαστικά, η NLP καλύπτει μια ποικιλία εργαλείων και τεχνικών όπως η αναγνώριση οντοτήτων(named entity recognition), η εξαγωγή πληροφοριών(information extraction), η ταξινόμηση θεμάτων(topic classification), η ανάλυση κειμένου(text analysis), η ανάλυση συναισθημάτων(sentiment analysis) και οι ενσωματώσεις λέξεων(word embeddings), μεταξύ πολλών άλλων.

4.5.1 Language Detection

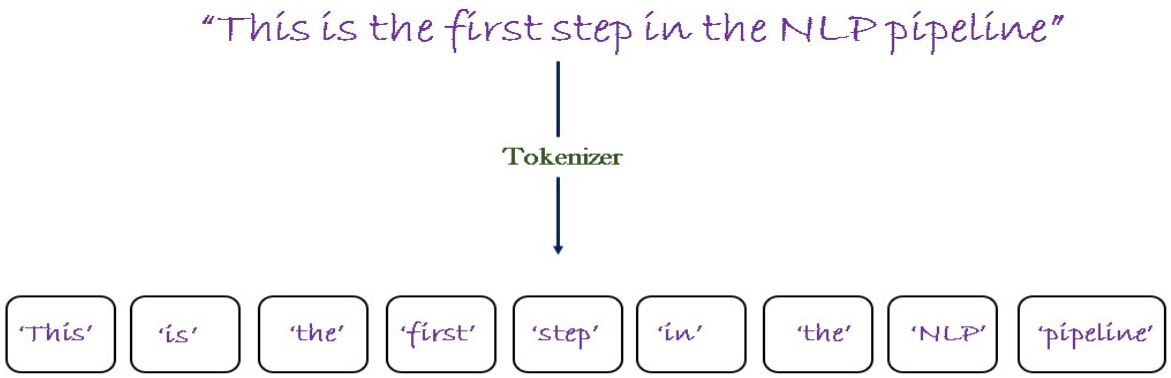
Όσον αφορά τα κείμενα το σώμα των κειμένων συχνά αποτελείται από έγγραφα σε πολλές γλώσσες. Μέχρι πρόσφατα οι περισσότεροι αλγόριθμοι απαιτούσαν την επεξεργασία μίας συγκεκριμένης γλώσσας ωστόσο οι πολυγλωσσικοί αλγόριθμοι έχουν πλέον γίνει πιο συνηθισμένοι. Ταυτόχρονα, η έρευνα συχνά επικεντρώνεται στην ανάλυση κειμένων σε μια συγκεκριμένη γλώσσα. Έτσι, αλγόριθμοι ανίχνευσης γλώσσας όπως το spacy-langdetect επιτρέπουν την αναγνώριση της γλώσσας των εγγράφων. Έχει λογική να πραγματοποιείται η ανίχνευση γλώσσας στην αρχή της διαδικασίας επεξεργασίας κειμένων (text wrangling) ώστε να φιλτράρονται μόνο τα έγγραφα από ολόκληρο το σώμα κειμένων που αντιστοιχούν στη γλώσσα που στοχεύουμε και χρειάζεται να υποστούν περαιτέρω επεξεργασία.

4.5.2 Tokenization

Για να εκτελεστούν εργασίες επεξεργασίας φυσικής γλώσσας (NLP) χρειάζεται να δημιουργηθεί ένα ισχυρό λεξιλόγιο. Αυτό επιτυγχάνεται με τη διάσπαση ενός εγγράφου σε μικρότερα κομμάτια που ονομάζονται tokens, τα οποία είναι λέξεις ή νοηματικές μονάδες. Γλώσσες όπως η Αγγλική ή η Γερμανική διαχωρίζουν τις λέξεις με κενά γεγονός που κάνει την τοκενοποίηση σχετικά εύκολη. Ωστόσο, σε γλώσσες

όπως η Κινέζικη όπου δεν υπάρχουν σαφή κενά μεταξύ των λέξεων, η διαδικασία γίνεται πιο δύσκολη και απαιτεί εξειδικευμένα εργαλεία.

Η τοκενοποίηση είναι συνήθως το πρώτο βήμα στις περισσότερες διαδικασίες NLP καθώς βοηθά στη μετατροπή κειμένων σε δομήσιμα δεδομένα. Εκτός από μεμονωμένες λέξεις, είναι επίσης δυνατό να αναγνωριστούν ομάδες διαδοχικών λέξεων, που ονομάζονται n-grams. Για παράδειγμα, ένα bi-gram περιλαμβάνει δύο συνεχόμενες λέξεις, ενώ ένα tri-gram τρεις. Αυτή η μέθοδος επιτρέπει την ανάλυση όχι μόνο μεμονωμένων λέξεων αλλά και συσχετισμών μεταξύ τους, καθιστώντας τα δεδομένα πιο χρήσιμα για την ανάλυση.

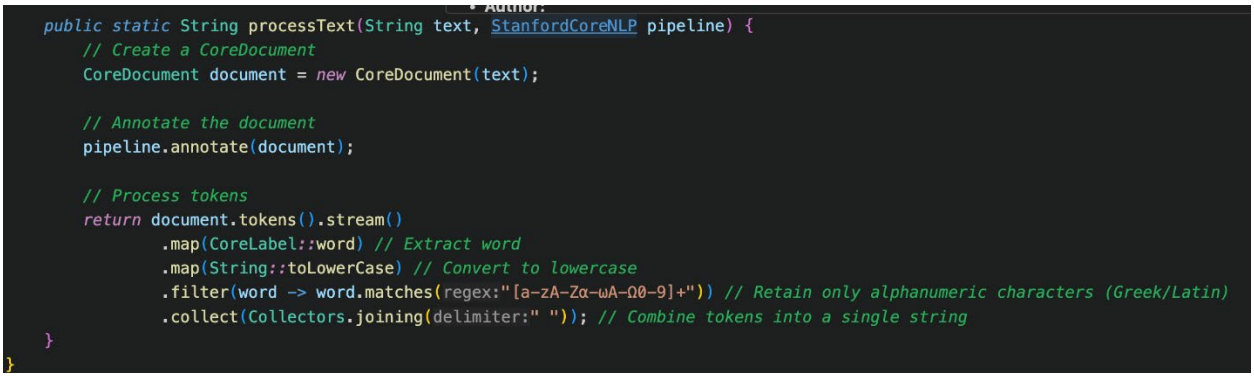


Σχήμα 4.5.2: Tokenizer[57].

4.5.3 Μετατροπή σε πεζά γράμματα και Αφαίρεση Σημείων Στίξης

Συνήθως, η μετατροπή του κειμένου σε πεζά γράμματα και η αφαίρεση σημείων στίξης θεωρούνται τα πρώτα βήματα προεπεξεργασίας(text preprocessing). Τα δεδομένα κειμένου σε πεζά γράμματα είναι ιδιαίτερα σημαντικά για την αποφυγή περιττών επαναλήψεων λέξεων. Για παράδειγμα, κατά την καταμέτρηση λέξεων, οι όροι «Venue» και «venue» θα μετρούνταν ως δύο διαφορετικές λέξεις οδηγώντας σε ανεπιθύμητη αύξηση των διαστάσεων των δεδομένων.

Τα σημεία στίξης, από την άλλη πλευρά, δημιουργούν «θόρυβο» στα δεδομένα και δεν προσφέρουν αξία στην ανάλυση, κάτι που εξηγεί γιατί πρέπει να αφαιρούνται. Ωστόσο, σε ορισμένες περιπτώσεις, έχει νόημα να αναλύονται μεμονωμένες προτάσεις. Σε τέτοιες περιπτώσεις, το σώμα κειμένων πρέπει πρώτα να διαχωριστεί σε μεμονωμένες προτάσεις, πριν αφαιρεθούν τα σημεία στίξης.



```
    public static String processText(String text, StanfordCoreNLP pipeline) {
        // Create a CoreDocument
        CoreDocument document = new CoreDocument(text);

        // Annotate the document
        pipeline.annotate(document);

        // Process tokens
        return document.tokens().stream()
                .map(CoreLabel::word) // Extract word
                .map(String::toLowerCase) // Convert to lowercase
                .filter(word -> word.matches(regex:"[a-zA-Zα-ωA-Ω0-9]+")) // Retain only alphanumeric characters (Greek/Latin)
                .collect(Collectors.joining(delimiter:" ")); // Combine tokens into a single string
    }
}
```

Σχήμα 4.5.3: Μέθοδος μετατροπής σε πεζά γράμματα [58].

4.5.4 Stemming and Lemmatisation

Το Stemming είναι η διαδικασία μείωσης μιας λέξης στη βασική ή ριζική της μορφή, αφαιρώντας προθέματα ή επιθήματα. Αυτή η προσέγγιση βασίζεται σε απλούς κανόνες και συχνά οδηγεί σε μη γλωσσολογικά σωστές ρίζες. Για παράδειγμα οι λέξεις "running", "runner" και "runs" μετατρέπονται όλες σε "run". Η λέξη "studies" μετατρέπεται σε "studi" (το οποίο δεν είναι γλωσσολογικά σωστό, αλλά είναι υπολογιστικά αποδοτικό). Το Stemming δεν λαμβάνει υπόψη το νόημα ή το πλαίσιο της λέξης αλλά χρησιμοποιεί απλούς κανόνες για την αποκοπή των καταλήξεων. Αυτή η τεχνική είναι χρήσιμη σε περιπτώσεις όπου η ακρίβεια δεν είναι κρίσιμη και το κύριο ζητούμενο είναι η ταχύτητα.

Η Λημματοποίηση είναι μια πιο εξελιγμένη διαδικασία, όπου οι λέξεις μετατρέπονται στη βασική τους μορφή (lemma) λαμβάνοντας υπόψη τη σημασιολογική τους δομή και τη γραμματική τους. Αυτό σημαίνει ότι η λημματοποίηση αναγνωρίζει τη μορφή της λέξης, τη συντακτική της λειτουργία και την έννοιά της για να την επαναφέρει στη σωστή της ρίζα. Για παράδειγμα η λέξη "παιδιά" μπορεί να είναι είτε η βασική μορφή του θηλυκού ουσιαστικού "παιδιά" (παιχνίδι) είτε μορφή του ουδέτερου ουσιαστικού "παιδί" ανάλογα με το περιεχόμενο. Σε αντίθεση με την αποκοπή καταλήξεων, η λημματοποίηση επιχειρεί να επιλέξει το σωστό λήμμα ανάλογα με το περιβάλλον.

Original word	Stemmed	Lemmatised
tourist	tourist	tourist
booking	book	book
rating	rate	rat
itinerary	itinerari	itinerary
recreation	recreat	recreation
amenities	amen	amenities
attractions	attract	attractions
sightseeing	sightse	sightsee
eating	eat	eat

Σχήμα 4.5.4: Σύγκριση original, stemmed, lemmatised λέξεων[59].

4.5.5 Part of speech(POS)

Το Part of Speech (POS) είναι η διαδικασία κατάταξης των λέξεων σε κατηγορίες με βάση τη γραμματική τους λειτουργία και το ρόλο τους μέσα σε μια πρόταση. Παραδείγματα κατηγοριών μέρους του λόγου είναι:

- Ουσιαστικά (Nouns): Αντικείμενα ή ονόματα (π.χ. "σπίτι", "άνθρωπος").
- Ρήματα (Verbs): Ενέργειες ή καταστάσεις (π.χ. "τρέχω", "είμαι").
- Επίθετα (Adjectives): Λέξεις που περιγράφουν ουσιαστικά (π.χ. "όμορφος", "μεγάλος").
- Επιρρήματα (Adverbs): Λέξεις που περιγράφουν ρήματα, επίθετα ή άλλα επιρρήματα (π.χ. "γρήγορα").
- Προθέσεις (Prepositions): Λέξεις που συνδέουν μέρη μιας πρότασης (π.χ. "σε", "μεταξύ").

Η επισήμανση των λέξεων με POS tagging είναι βασικό στάδιο στην επεξεργασία φυσικής γλώσσας (NLP) και βοηθά στην κατανόηση της γραμματικής δομής ενός κειμένου.

4.5.6 Named Entity Recognition (NER)

Η αναγνώριση οντοτήτων (Named Entity Recognition - NER), γνωστή επίσης ως αναγνώριση οντοτήτων ή εντοπισμός οντοτήτων, είναι μια προχωρημένη στατιστική διαδικασία που πλέον δεν μπορεί να χαρακτηριστεί ως μέρος του σταδίου προεπεξεργασίας, αλλά ανήκει στον τομέα της εξαγωγής πληροφοριών (information extraction). Μέσω ενός συστήματος αναγνώρισης οντοτήτων, μπορούν να αποδοθούν ετικέτες (μετα-πληροφορίες) σε συνεχόμενες ακολουθίες από tokens.
Από την βάση μας μπορούμε να πάρουμε την πρόταση “Tennessee Williams Η ΝΥΧΤΑ ΤΗΣ ΙΓΚΟΥΑΝΑ Σκηνοθεσία: Μαρία Μαγκανάρη Από 20 Οκτωβρίου έως 28 Νοεμβρίου 2021” από το field description του πίνακα production και εφαρμόζοντας NER μπορούμε να εντοπίσουμε τις εξής οντότητες:

- Tennessee Williams → Πρόσωπο (Person)
- Η ΝΥΧΤΑ ΤΗΣ ΙΓΚΟΥΑΝΑ → Έργο/Τίτλος (Work/Title)

- Μαρία Μαγκανάρη → Πρόσωπο (Person)
- 20 Οκτωβρίου έως 28 Νοεμβρίου 2021 → Ημερομηνίες (Date)

```
    // Κείμενο προς επεξεργασία
    String text = "Tennessee Williams Η ΝΥΧΤΑ ΤΗΣ ΙΓΚΟΥΑΝΑ Σκηνοθεσία: Μαρία Μαγκανάρη Από 20 Οκτωβρίου έως 28 Νοεμβρίου 2021";

    // Ρυθμίσεις για το Stanford CoreNLP
    Properties props = new Properties();
    props.setProperty("annotators", "tokenize,ssplit,pos,lemma,ner");
    StanfordCoreNLP pipeline = new StanfordCoreNLP(props);

    // Δημιουργία Annotation για το κείμενο
    Annotation document = new Annotation(text);

    // Εκτέλεση του pipeline
    pipeline.annotate(document);

    // Ανάκτηση προτάσεων από το κείμενο
    List<CoreMap> sentences = document.get(CoreAnnotations.SentencesAnnotation.class);

    // Ανάλυση κάθε πρότασης
    System.out.println("Recognized Entities:");
    for (CoreMap sentence : sentences) {
        sentence.get(CoreAnnotations.TokensAnnotation.class).forEach(token -> {
            String word = token.originalText(); // Η λέξη
            String ner = token.get(CoreAnnotations.NamedEntityTagAnnotation.class); // Οντότητα
            if (!"O".equals(ner)) { // Αγνοούμε τις λέξεις που δεν έχουν αναγνωρισμένη οντότητα
                System.out.println(word + " -> " + ner);
            }
        });
    }
}
```

Σχήμα 4.5.5: Παράδειγμα NER [60].

4.5.7 Feature extraction

Η εξαγωγή χαρακτηριστικών (Feature Extraction) αποτελεί ένα ακόμα βασικό μέρος της προεπεξεργασίας κειμένου. Στο πλαίσιο της μηχανικής μάθησης, τα χαρακτηριστικά αντιπροσωπεύουν μεταβλητές, και μέσω της εξαγωγής χαρακτηριστικών, νέες μεταβλητές εξάγονται από τα δεδομένα κειμένου για περαιτέρω ανάλυση. Αυτά τα χαρακτηριστικά μπορεί να περιλαμβάνουν τον αριθμό των αριθμητικών χαρακτήρων, το μέσο μήκος των λέξεων κ.λπ. Κατά τη διεξαγωγή περαιτέρω αναλύσεων NLP, μπορούν να προκύψουν νέα χαρακτηριστικά που εξάγονται συνεχώς, όπως τα αποτελέσματα της ανάλυσης συναισθήματος (sentiment analysis) υπό τη μορφή βαθμολογιών συναισθήματος, οι βαρύτητες όρων (term weights) στο topic modelling, ή οι οντότητες (entities) που εξάγονται μέσω της αναγνώρισης οντοτήτων (NER) κ.ά.

4.6 Προκλήσεις της εργασίας με κείμενο(text)

Παρόλο που η Επεξεργασία Φυσικής Γλώσσας (NLP) αυξάνει συνεχώς τη δημοτικότητά της και μπορεί να εφαρμοστεί σε πολλά σενάρια συνοδεύεται επίσης από αρκετές προκλήσεις. Λόγω του γεγονότος ότι οι διαφορετικές γλώσσες περιλαμβάνουν ποικίλες γραμματικές δομές, τις οποίες η NLP μπορεί να μην είναι ικανή να αναγνωρίσει και να αποκωδικοποιήσει, η εργασία με δεδομένα κειμένου μπορεί εύκολα να μετατραπεί σε μια πολύπλοκη διαδικασία. Οι προκλήσεις που σχετίζονται με την ανάλυση δεδομένων κειμένου προέρχονται από πέντε βασικούς λόγους:

Συνωνυμία (Synonymy)
Οι περισσότερες προσεγγίσεις NLP επικεντρώνονται στη συχνότητα εμφάνισης λέξεων κατά την ανάλυση τους. Ωστόσο, λόγω της ύπαρξης συνωνύμων, αυτά τα αποτελέσματα μπορεί να είναι προκατειλημμένα, και η ομοιότητα μεταξύ προτάσεων να εκτιμάται λανθασμένα. Οι μέθοδοι NLP που αντιμετωπίζουν τη συνωνυμία συνήθως εστιάζουν σε unigrams (μονές λέξεις), ενώ κάποιες άλλες εργασίες, όπως η μηχανική μετάφραση και η απλοποίηση κειμένου εξετάζουν την ομοιότητα μεταξύ όρων που αποτελούνται από πολλές λέξεις.

Λεξική Ασάφεια (Lexical Ambiguity)
Η λεξική ασάφεια υπάρχει σε περιπτώσεις όπου ένας όρος ή φράση θεωρείται ομώνυμο, δηλαδή κάποιες λέξεις έχουν πολλαπλές σημασίες.

Γλωσσικά Ζητήματα (Language-Related Issues)
Οι διαφορετικές γλώσσες έχουν διαφορετικές γραμματικές δομές και κανόνες. Για παράδειγμα στην Ελληνική γλώσσα, η λέξη "η ώρα" μπορεί να χρησιμοποιηθεί για να εκφράσει διαφορετικές έννοιες:

- Η ώρα ως χρόνος: «Τι ώρα είναι;» (χρονική έννοια).
- Η ώρα ως στιγμή**:** «Ήρθε η ώρα να φύγουμε.» (συγκεκριμένη στιγμή για δράση).

Αυτή η πολυσημία μπορεί να αποτελέσει πρόκληση για την ανάλυση δεδομένων κειμένου, καθώς η έννοια εξαρτάται από το πλαίσιο.

Αναφορική Ασάφεια (Referential Ambiguity)
Σε πολλές περιπτώσεις, δεν είναι ξεκάθαρο σε τι αναφέρεται μια οντότητα. Στο παράδειγμα "Η Jennifer συνάντησε τη φίλη της στο εστιατόριο πριν επιστρέψει στο ξενοδοχείο", το "επιστρέψει" αναφέρεται στη Jennifer ή στη φίλη της; Τα τελευταία χρόνια, πολλά συστήματα επίλυσης αναφορικής ασάφειας έχουν βελτιστοποιηθεί χάρη στη χρήση νευρωνικών δικτύων. Ωστόσο, η αναφορική ασάφεια εξακολουθεί να αποτελεί πρόβλημα στην αυτόματη ανάλυση κειμένου.

Πρόβλημα Λέξεων Εκτός Λεξιλογίου (Out-of-Vocabulary Problem)
Οι υπολογιστές μπορούν να επεξεργαστούν όρους μόνο εάν τους έχουν ήδη δει. Αυτό καθιστά δύσκολη την ορθή επεξεργασία νέων ή άγνωστων λέξεων. Επομένως, οι χειροκίνητα δημιουργημένες λεξικές βάσεις δεδομένων έχουν μεγάλη σημασία για την ανάλυση κειμένου. Μια ευρέως χρησιμοποιούμενη λεξική βάση δεδομένων είναι το WordNet, όπου τα ουσιαστικά, τα ρήματα, τα επίθετα και τα επιρρήματα στα Αγγλικά ομαδοποιούνται σε Sets βάσει σημασιολογικών ομοιοτήτων, δημιουργώντας ήδη καταγεγραμμένες σημασιολογικές σχέσεις μεταξύ των λέξεων. Το WordNet μπορεί επομένως να θεωρηθεί ως ένα σημασιολογικό δίκτυο λέξεων και εννοιών που συνδέονται νοηματικά.

Κεφάλαιο 5ο: Υλοποίηση Frontend με Next.js(Javascript)

Η ανάπτυξη του frontend αποτελεί ένα κρίσιμο κομμάτι της συνολικής αρχιτεκτονικής της εφαρμογής, καθώς παρέχει τη διεπαφή μέσω της οποίας οι χρήστες αλληλοεπιδρούν με το σύστημα. Σε αυτό το

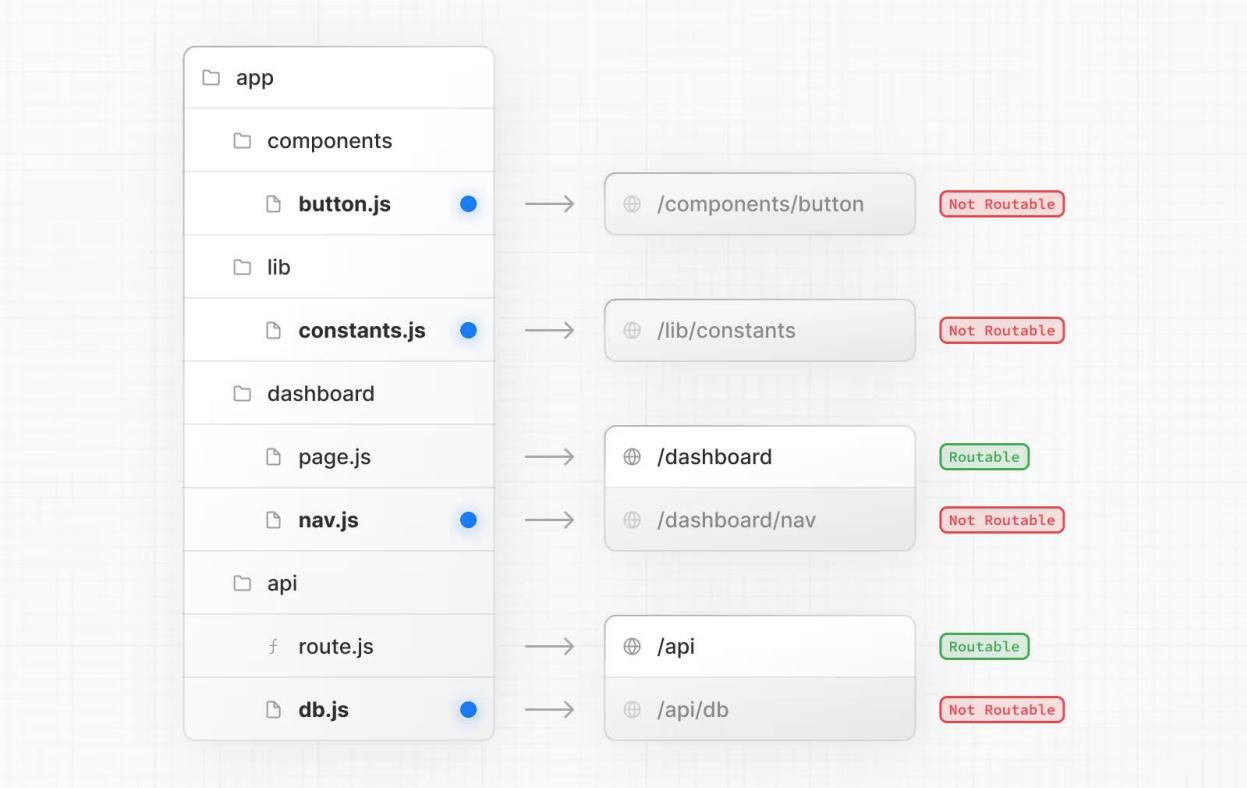
κεφάλαιο, περιγράφεται η χρήση του Next.js για την ανάπτυξη του frontend, εστιάζοντας στη δομή, τη λειτουργικότητα και την επικοινωνία με το backend. Το Next.js, ως ένα σύγχρονο πλαίσιο βασισμένο στο React, προσφέρει υψηλή απόδοση και ευελιξία στην ανάπτυξη εφαρμογών.

5.1 Εισαγωγή στο Next.js

Το Next.js είναι ένα framework της JavaScript που βασίζεται στο React και προορίζεται για τη δημιουργία διακομιστών και στατικών εφαρμογών. Προσφέρει δυνατότητες όπως server-side rendering (SSR) και static site generation (SSG), επιτρέποντας τη δυναμική δημιουργία περιεχομένου και τη βελτίωση της απόδοσης και της εμπειρίας του χρήστη. Επιπλέον, το Next.js παρέχει ενσωματωμένη υποστήριξη για δρομολόγηση (routing), διαχείριση δεδομένων, και βελτιστοποίηση SEO, καθιστώντας το ιδανικό για την ανάπτυξη εφαρμογών που απαιτούν αποδοτικότητα και μοντέρνο σχεδιασμό.

5.2 Δομή και Αρχιτεκτονική του Frontend

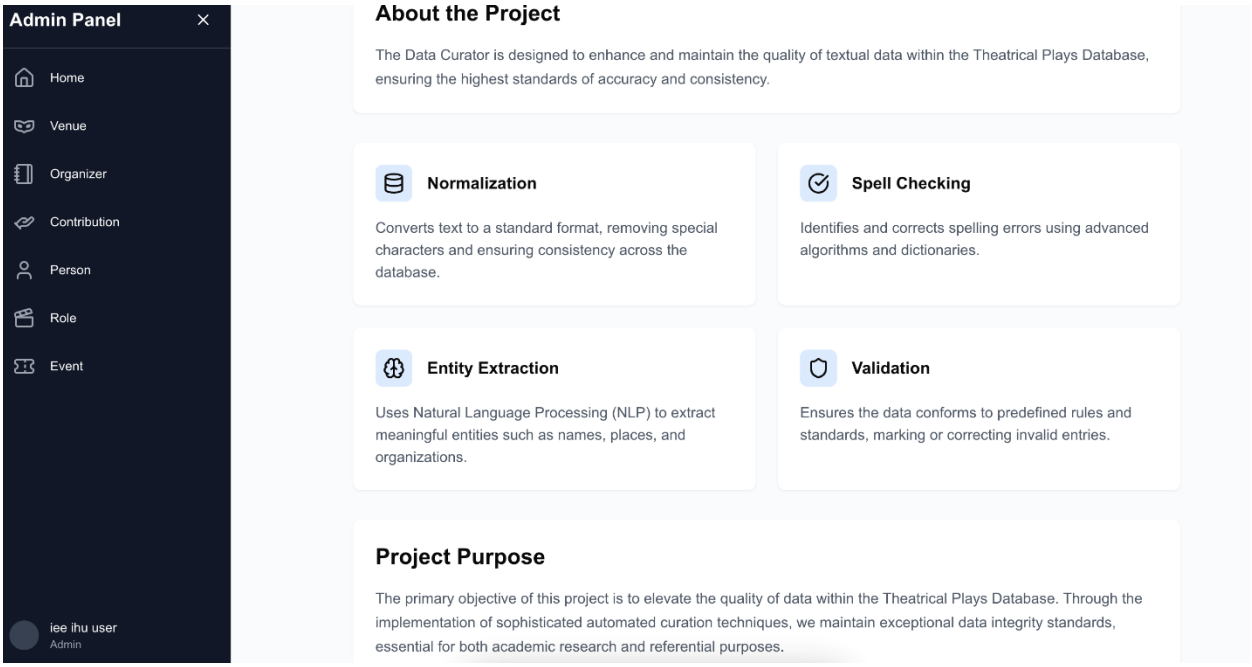
Η δομή του frontend βασίστηκε στις βέλτιστες πρακτικές του Next.js, χωρίζοντας τη λειτουργικότητα σε διακριτά αρχεία και φακέλους. Η ιεραρχία του κώδικα οργανώθηκε έτσι ώστε να περιλαμβάνει φακέλους για σελίδες (pages), συστατικά (components), και ενέργειες (services), οι οποίες περιλαμβάνουν την επικοινωνία με τα REST APIs. Η αρχιτεκτονική σχεδιάστηκε για να είναι επεκτάσιμη, με τα επαναχρησιμοποιήσιμα συστατικά να διευκολύνουν τη συντήρηση και την αναβάθμιση του κώδικα. Η πλοήγηση μεταξύ των σελίδων πραγματοποιήθηκε με το ενσωματωμένο routing του Next.js, ενώ η χρήση των hooks του React βελτίωσε τη διαχείριση της κατάστασης της εφαρμογής.



Σχήμα 5.2: Next.js file architecture[61].

5.3 Διαχείριση Χρήστη και Διεπαφή Εφαρμογής

Η διαχείριση του χρήστη και η σχεδίαση της διεπαφής αποτέλεσαν βασικούς άξονες για την ανάπτυξη του frontend. Ενσωματώθηκε σύστημα ταυτοποίησης χρηστών με χρήση διακριτών API endpoints για εγγραφή, σύνδεση, και διαχείριση προφίλ. Η διεπαφή σχεδιάστηκε με έμφαση στη φιλικότητα προς τον χρήστη, χρησιμοποιώντας σύγχρονα UI συστατικά βασισμένα σε CSS frameworks όπως το Tailwind CSS. Η δυναμική εμφάνιση δεδομένων οι φόρμες αλληλεπίδρασης και οι ειδοποιήσεις υλοποιήθηκαν ώστε να παρέχουν μια εύκολη εμπειρία στον χρήστη.



Σχήμα 5.3: Homepage[62].

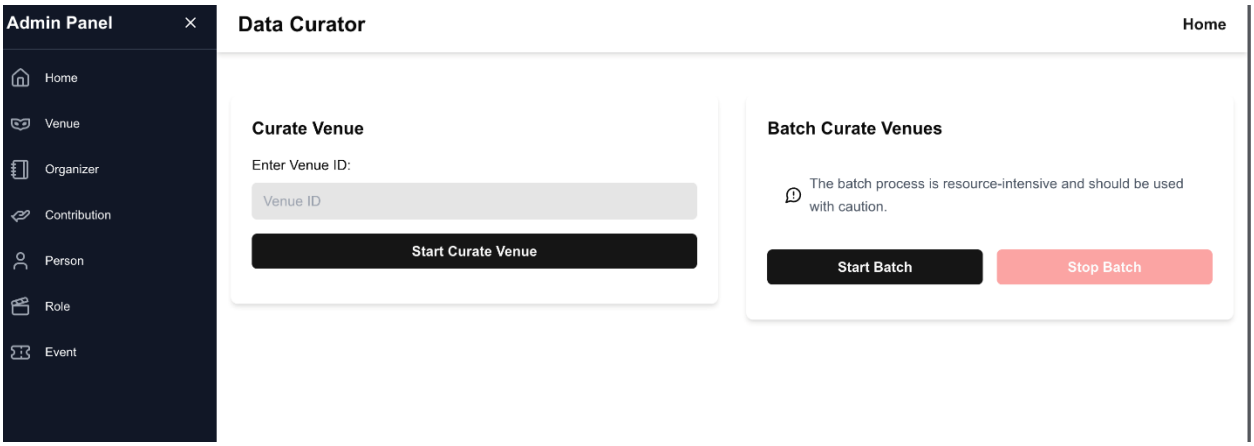
src > components > ui > HomePage.tsx > ...

```tsx
import { Database, CheckCircle, Brain, Shield } from "lucide-react";  2.4k (gzipped: 1.3k)

const HomePage = () => {
  const features = [
    {
      icon: <Database className="w-6 h-6" />,
      title: "Normalization",
      description:
        "Converts text to a standard format, removing special characters and ensuring consistency across the database.",
    },
    {
      icon: <CheckCircle className="w-6 h-6" />,
      title: "Spell Checking",
      description:
        "Identifies and corrects spelling errors using advanced algorithms and dictionaries.",
    },
    {
      icon: <Brain className="w-6 h-6" />,
      title: "Entity Extraction",
      description:
        "Uses Natural Language Processing (NLP) to extract meaningful entities such as names, places, and organizations.",
    },
    {
      icon: <Shield className="w-6 h-6" />,
      title: "Validation",
      description:
        "Ensures the data conforms to predefined rules and standards, marking or correcting invalid entries.",
    },
  ];
```

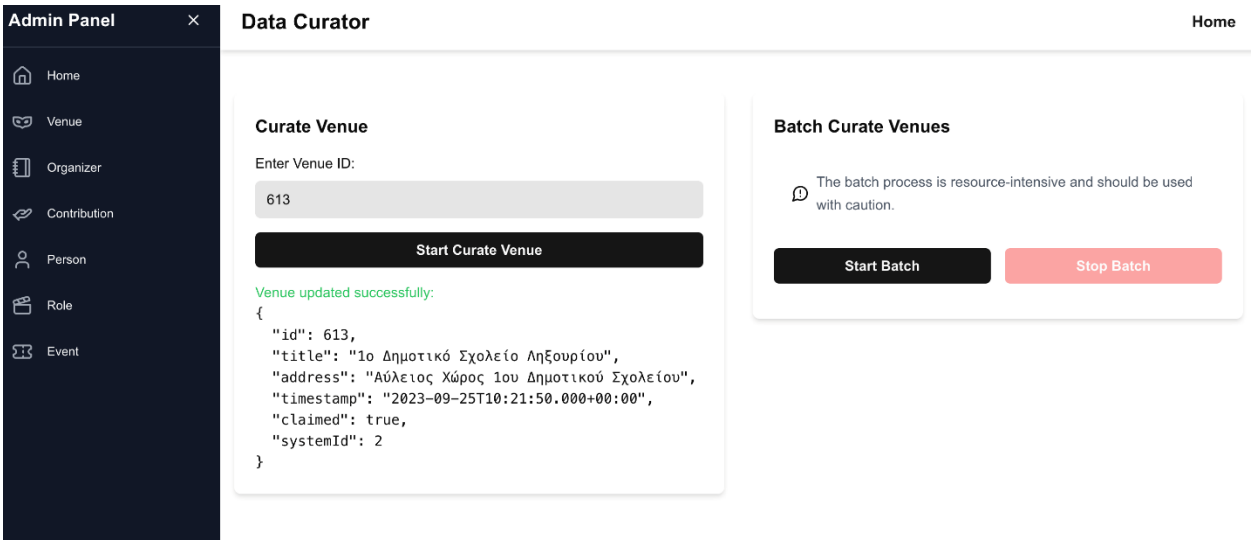
Σχήμα 5.3: Μέρος κώδικα για το Homepage[63].

5.4 Επικοινωνία με το Backend μέσω REST APIs

Η επικοινωνία μεταξύ frontend και backend επιτεύχθηκε μέσω REST APIs που παρέχονται από την εφαρμογή Spring Boot. Το frontend χρησιμοποιεί βιβλιοθήκες όπως το Fetch για την αποστολή αιτημάτων HTTP προς το backend και την επεξεργασία των απαντήσεων. Κάθε API endpoint αντιστοιχεί σε συγκεκριμένες λειτουργίες, όπως η διαχείριση χρηστών, η ανάκτηση δεδομένων για θεατρικές παραστάσεις ή συνεισφορές, και η αποστολή νέων εγγραφών στη βάση δεδομένων. Η ασφάλεια των δεδομένων εξασφαλίστηκε μέσω της χρήσης tokens για αυθεντικοποίηση και εξουσιοδότηση.



Σχήμα 5.4: Curate Venue page[64].



Σχήμα 5.4: Curate Venue με id 613[65].

Κεφάλαιο 6ο: Συμπεράσματα και Προτάσεις Βελτίωσης

Η παρούσα πτυχιακή εργασία είχε ως στόχο την ανάπτυξη και υλοποίηση ενός ολοκληρωμένου συστήματος επεξεργασίας, επιμέλειας και οπτικοποίησης δεδομένων στο πεδίο των θεατρικών παραστάσεων, αξιοποιώντας σύγχρονες τεχνολογίες και μεθόδους όπως το Spring Boot για το backend, το Next.js για το frontend, και βιβλιοθήκες επεξεργασίας φυσικής γλώσσας όπως το Stanford NLP και τα LLMs. Το έργο ανέδειξε τη σημασία της επιμέλειας δεδομένων (data curation) σε συνδυασμό με εργαλεία επεξεργασίας φυσικής γλώσσας για τη βελτίωση της ποιότητας των δεδομένων και την εξαγωγή πολύτιμων πληροφοριών.

Μέσα από τη διαδικασία της υλοποίησης, αντιμετωπίστηκαν προκλήσεις όπως η ασυνέπεια στα δεδομένα, η διαχείριση διπλότυπων εγγραφών, και η ενσωμάτωση τεχνικών ανάλυσης φυσικής γλώσσας για την εξαγωγή οντοτήτων και την κατηγοριοποίηση πληροφοριών. Το σύστημα απέδειξε την ικανότητά του να οργανώνει δεδομένα από πολλαπλές πηγές, να τα καθαρίζει, και να τα παρουσιάζει με τρόπο κατανοητό και χρήσιμο για τον χρήστη. Η χρήση του Next.js για το frontend επέτρεψε τη δημιουργία μιας φιλικής προς τον χρήστη διεπαφής, ενώ η επικοινωνία με το backend μέσω REST APIs εξασφάλισε την αποδοτική ροή δεδομένων μεταξύ των επιπέδων της εφαρμογής.

Παρά τα θετικά αποτελέσματα, υπάρχουν περιθώρια βελτίωσης σε διάφορες πτυχές του συστήματος. Ένα σημαντικό ζήτημα είναι η περαιτέρω βελτίωση της απόδοσης του συστήματος καθώς η επιμέλεια μεγάλου όγκου δεδομένου είναι χρονοβόρα. Επιπλέον η ενσωμάτωση πολυγλωσσικών δυνατοτήτων θα διεύρυνε την εφαρμογή του συστήματος και θα το καθιστούσε πιο προσβάσιμο για διαφορετικά ακροατήρια. Επίσης, προτείνεται η ενίσχυση της ασφάλειας των δεδομένων, ιδίως όσον αφορά την αποθήκευση και επεξεργασία ευαίσθητων πληροφοριών, με τη χρήση τεχνολογιών όπως η κρυπτογράφηση.Τέλος μελλοντικές βελτιώσεις θα μπορούσαν να περιλαμβάνουν την αξιοποίηση μηχανισμών μηχανικής μάθησης για την πρόβλεψη τάσεων, την κατηγοριοποίηση θεατρικών παραστάσεων ή τη σύσταση περιεχομένου.

Συνοψίζοντας, η εργασία αυτή ανέδειξε την αξία της ενσωμάτωσης τεχνολογιών αιχμής για την επεξεργασία και οργάνωση δεδομένων στον πολιτιστικό τομέα, προσφέροντας μια λειτουργική πλατφόρμα με πρακτικές εφαρμογές. Οι προτεινόμενες βελτιώσεις μπορούν να ενισχύσουν την αποτελεσματικότητα και τη χρησιμότητα του συστήματος καθιστώντας το ακόμα πιο καινοτόμο και ικανό να ανταποκριθεί στις ανάγκες του πεδίου.

Βιβλιογραφία

Internet Site

[1] A review: preprocessing techniques and data augmentation for sentiment analysis. [Διαδικτυακό]. Διαθέσιμο: A review: preprocessing techniques and data augmentation for sentiment analysis | Computational Social Networks | Full Text (springeropen.com)

[2] Text Preprocessing in NLP | Basis Steps to Preprocess The Textual Data. [Διαδικτυακό]. Διαθέσιμο: Text Preprocessing | NLP | Steps to Process Text (kaggle.com).

[3] Preprocessing Steps for Natural Language Processing (NLP): A Beginner's Guide. Διαθέσιμο: Preprocessing Steps for Natural Language Processing (NLP): A Beginner's Guide | by Maleesha De Silva | Medium.

[4] Data Curation: Definition, Importance & Examples. [Διαδικτυακό]. Διαθέσιμο: Data Curation: Definition, Importance & Examples (atlan.com).

[5] Technical Deep-Dive: Curating Our Way to a State-of-the-Art Text Dataset. [Διαδικτυακό]. Διαθέσιμο: Technical Deep-Dive: Curating Our Way to a State-of-the-Art Text Dataset (datologyai.com).

[6] What is NLP (natural language processing)?. [Διαδικτυακό]. Διαθέσιμο: What Is NLP (Natural Language Processing)? | IBM.

[7] The Practical Guide to Unleashing the Power of the CORENLP Library. [Διαδικτυακό]. Διαθέσιμο: The Practical Guide to Unleashing the Power of the CORENLP Library | by Tushar Aggarwal | Medium.

[8] Unraveling the Power of Stanford CoreNLP in NLP. [Διαδικτυακό]. Διαθέσιμο: Exploring Stanford CoreNLP: Unleashing the Power of NLP (softobotics.com).

Paper

[9] Roman Egger, "Natural Language Processing (NLP): An Introduction: Making Sense of Textual Data ", January 2022.

[10] Arkaitz Zubiaga, "Natural language processing in the era of large language models", January 2024.

[11] Madhusudhan Margam, "Text Pre-Processing", April 2022.

[12] Christopher D. Manning, "The Stanford CoreNLP Natural Language Processing Toolkit", January 2014.

[12] Abhinav Kathuria, "A Review of Tools and Techniques for Preprocessing of Textual Data", January 2021.