ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Ανάπτυξη Εφαρμογής Αυτόματης Παραγγελιοληψίας για Εμπορικά Καταστήματα με Χρήση Λογισμικού Ανοικτού Κώδικα για την Αναγνώριση Ομιλίας

Παναγιώτης Μαρινόπουλος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Τζιάλλας Γρηγόριος
Καθηγητής Α Βαθμίδας

Λαμία Νοέμβριος 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Ανάπτυξη Εφαρμογής Αυτόματης Παραγγελιοληψίας για Εμπορικά Καταστήματα με Χρήση Λογισμικού Ανοικτού Κώδικα για την Αναγνώριση Ομιλίας

Παναγιώτης Μαρινόπουλος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Τζιάλλας Γρηγόριος
Καθηγητής Α Βαθμίδας

Λαμία Νοέμβριος 2021

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

Development of an Automatic Ordering Application for Commercial Stores using Open Source Software for Speech Recognition

Panagiotis Marinopoulos

FINAL THESIS

ADVISOR

Tziallas Grigorios
Professor

Lamia November 2021

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: ……/…../20……

Ο – Η Δηλ.

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση
του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων
σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Το φωνητικό εμπόριο είναι μια σχετικά νέα μορφή του ηλεκτρονικού εμπορίου που δίνει τη δυνατότητα στον καταναλωτή να χρησιμοποιεί τη φωνή του για να πραγματοποιεί αγορές προϊόντων και υπηρεσιών. Σε αυτή τη πτυχιακή εργασία, θα αναπτυχθεί μια εφαρμογή που αξιοποιώντας την τεχνολογία αυτόματης αναγνώρισης ομιλίας και πιο συγκεκριμένα του συστήματος ανοιχτού κώδικα DeepSpeech, θα επιτρέπει στους καταναλωτές να εκτελούν φωνητικές εντολές για την διαχείριση ενός καλαθιού και την αγορά των προϊόντων που αυτό περιέχει, παρέχοντας στην ουσία μια φωνητική διεπαφή χρήστη (Voice-User Interface). Η εφαρμογή αποτελείται από τρία μέρη, την επικοινωνία του καταναλωτή με ένα κατάστημα που προσφέρει προϊόντα, την αναγνώρισης της ομιλίας του και την επεξεργασία των φωνητικών εντολών μετά το προηγούμενο στάδιο.

ABSTRACT

Voice commerce is a relatively new trend in electronic commerce which gives consumers the ability to use their voice to carry out purchases of products or services. The purpose of this thesis is to develop an application that utilizes the technology of automatic speech recognition, specifically the open-source engine DeepSpeech, to allow consumers to manage a shopping cart using voice commands and eventually make a purchase of products, effectively providing a voice user interface. The said application consists of three parts: communication between the consumer and a store, the recognition and translation of their voice commands into text and the processing of those into performing certain actions.

Table of Contents

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

Η αγορά προϊόντων και υπηρεσιών μέσω του Διαδικτύου έχει γίνει πλέον καθημερινότητα. Πολλά καταστήματα, μικρά και μεγάλα, ανεξάρτητα του τι προσφέρουν παρέχουν, τη δυνατότητα στους πελάτες τους να πραγματοποιούν τις συναλλαγές τους από την άνεση του σπιτιού τους, μέσω της συσκευής προτίμησής τους. Παράλληλα με τη ραγδαία εξέλιξη του ηλεκτρονικού εμπορίου, έχει εμφανιστεί μια νέα μορφή του, το φωνητικό εμπόριο, επιτρέποντας στους καταναλωτές να κάνουν συναλλαγές μονάχα με τη φωνή τους. Αυτή η καινοτομία αναμένεται από ειδικούς να συμβάλει καθοριστικά στην ανάπτυξη του ηλεκτρονικού εμπορίου τα επόμενα χρόνια [23] .

Σε αυτό, έχει συμβάλει και η συνεχής ανάπτυξη και εξέλιξη των τεχνολογιών αναγνώρισης ομιλίας, που καθιστούν δυνατή τη μετατροπή του προφορικού λόγου σε κείμενο αναγνωρίσιμο και επεξεργάσιμο από υπολογιστές. Επίσης σημαντικός παράγοντας είναι οι εικονικοί βοηθοί που έχουν δει άνοδο τα τελευταία χρόνια. Στα επόμενα υποκεφάλαια θα παρουσιαστεί μια σύντομη περιγραφή και ιστορική αναδρομή του ηλεκτρονικού εμπορίου, καθώς και μια περιγραφή του φωνητικού εμπορίου με μερικά από τα πλεονεκτήματα και μειονεκτήματά του.

Έχοντας υπόψη τα παραπάνω, θα αναπτύξουμε μια εφαρμογή αυτόματης παραγγελιοληψίας, που θα προσφέρει τη δυνατότητα στους καταναλωτές ενός καταστήματος να κάνουν παραγγελίες μόνο με τη φωνή τους. Η εφαρμογή αυτή είναι σχεδιασμένη ώστε να είναι έως ένα βαθμό παραμετροποιήσιμη, εύκολη στη χρήση από τον καταναλωτή και στην εξατομίκευση από τη μεριά του διαχειριστή και χωρίζεται σε τρία διακριτά μέρη:

1. Επικοινωνία με το κατάστημα
2. Αναγνώριση Ομιλίας
3. Επεξεργασία εντολής

Τα Κεφάλαια 2 και 3 αφορούν στο δεύτερο μέρος, και περιγράφουν τη διαδικασία αυτόματης αναγνώρισης ομιλίας (Automatic Speech Recognition, ASR) επίσης παρουσιάζεται μια σύγκριση μεταξύ τριών συστημάτων αναγνώρισης ομιλίας ανοιχτού κώδικα σύμφωνα με έρευνες. Τέλος, στο Κεφάλαιο 3 γίνεται η εκπαίδευση ενός ακουστικού και γλωσσικού μοντέλου αναγνώρισης ομιλίας.

Στο Κεφάλαιο 4 αναπτύσσεται μια εφαρμογή για την επικοινωνία του πελάτη με το κατάστημα μέσω ενός φυλλομετρητή (Web Browser) στη γλώσσα προγραμματισμού JavaScript, με τη χρήση των framework Node.js και Express.

Στο Κεφάλαιο 5 αναπτύσσεται η βασική εφαρμογή αυτής της πτυχιακής εργασίας που επεξεργάζεται τις εντολές του πελάτη και εν τέλει καταλήγει σε μια παραγγελία. Επιπλέον, γίνεται εισαγωγή στην γλώσσα προγραμματισμού Python, που χρησιμοποιήθηκε για την εφαρμογή και δημιουργείται μια βάση δεδομένων για να αποθηκεύει όλα τα προϊόντα του καταστήματος αλλά και τις πληροφορίες των χρηστών και τις παραγγελίες που αυτοί κάνουν.

Τέλος, στο Κεφάλαιο 6 παρουσιάζεται ενδεικτικά μια περίπτωση χρήσης της εφαρμογής, ρυθμίζοντάς την και δημιουργώντας μια βάση δεδομένων με μερικά προϊόντα για πώληση.

Η εφαρμογή που θα αναπτυχθεί στο Κεφάλαιο 5 θα δίνει τη δυνατότητα στους πελάτες να προσθέτουν και να αφαιρούν προϊόντα από το καλάθι τους, να το αδειάσουν και να δουν ή ακούσουν τα περιεχόμενά του και να ολοκληρώσουν την παραγγελία τους εφόσον έχουν προσθέσει προϊόντα σε αυτό μέσω φωνητικών ή γραπτών εντολών.

Ο χρήστης (ή διαχειριστής) της εφαρμογής, θα μπορεί να προσαρμόσει τα μηνύματα με τα οποία επικοινωνεί η εφαρμογή με τον πελάτη, να ζητάει επιβεβαίωση από αυτόν πριν προχωρήσει σε μια ενέργεια όπου κρίνει ο χρήστης απαραίτητη, να ζητάει επίσης την ποσότητα του προϊόντος που προστίθεται ή αφαιρείται από το καλάθι ,να ρυθμίσει πολλές παραμέτρους στους αλγόριθμους αναζήτησης, να τροποποιήσει τα λεξικά που χρησιμοποιούνται κατά την επεξεργασία των εντολών και να ορίσει τις λέξεις-κλειδιά στις οποίες βασίζεται η εφαρμογή για την αναγνώριση των εντολών.

Η βάση δεδομένων είναι σχεδιασμένη με τέτοιο τρόπο ώστε να μπορεί να διακρίνει προϊόντα σε κατηγορίες και υποκατηγορίες αλλά και να περιλαμβάνει όλα τα ειδικά χαρακτηριστικά που μπορεί να έχει ένα προϊόν. Αποθηκεύει τα βασικά στοιχεία τους όπως τιμή, κατασκευαστής, μάρκα, όνομα και περιγραφή μαζί με αυτά που προαναφέρθηκαν.

1.1 Ηλεκτρονικό Εμπόριο

Ηλεκτρονικό εμπόριο (Electronic Commerce ή e-commerce) είναι «η χρήση ηλεκτρονικών επικοινωνιών και τεχνολογιών επεξεργασίας ψηφιακής πληροφορίας σε επιχειρηματικές συναλλαγές για δημιουργία, μετατροπή και επαναπροσδιόριση σχέσεων μεταξύ οργανισμών ή (και) οργανισμών και ατόμων» [1] .

Ειδικότερα, αφορά στην αγορά ή πώληση προϊόντων ή υπηρεσιών μέσω του Διαδικτύου ή άλλης μορφής ψηφιακής επικοινωνίας, χωρίς να είναι απαραίτητη η φυσική παρουσία των εμπλεκόμενων μερών.

Μερικές συχνές εφαρμογές του ηλεκτρονικού εμπορίου είναι οι εξής:

- Business to Business (B2B): η πώληση προϊόντων ή υπηρεσιών μεταξύ επιχειρήσεων
- Business to Consumer (B2C): η πώληση προϊόντων ή υπηρεσιών μεταξύ επιχείρησης και καταναλωτή
- Business to Government (B2G): συναλλαγές που πραγματοποιούνται μεταξύ επιχείρησης και κράτους
- Consumer to Consumer (C2C): συναλλαγές που πραγματοποιούνται μεταξύ καταναλωτών
- Mobile commerce(m-commerce): η χρήση κινητών συσκευών, όπως Smartphones και Tablets, για τη διεξαγωγή εμπορικών συναλλαγών στο Διαδίκτυο

1.1.1 Ιστορική Αναδρομή του Ηλεκτρονικού Εμπορίου

Σύμφωνα με τη Wikipedia [2]

- Δεκαετία του 70

  Κυκλοφορούν πολύ βασικά συστήματα ηλεκτρονικού εμπορίου χρησιμοποιώντας τις τεχνολογίες Electronics Funds Transfer (EFT) και Electronic Data Interchange (EDI)

- Δεκαετία του 80

  Ιδρύθηκαν τα Boston Computer Exchange, μια πλατφόρμα για πώληση χρησιμοποιημένων υπολογιστών, και το Minitel, μια Videotex (σύστημα που προσφέρει διαδραστικό περιεχόμενο και το εμφανίζει σε μια οθόνη) διαδικτυακή υπηρεσία, δύο από τις πρώτες πιο αξιοσημείωτες πλατφόρμες ηλεκτρονικού εμπορίου στον κόσμο

- Δεκαετία του 90

  Με την κυκλοφορία του πρώτου φυλλομετρητή WorldWideWeb το 1990, οι χρήστες με πρόσβαση στο διαδίκτυο μπορούν να περιηγηθούν πιο εύκολα σε πλατφόρμες ηλεκτρονικού εμπορίου. Κυκλοφορούν, μέσω άλλων, οι υπηρεσίες Amazon.com και eBay, δύο από τις πιο σπουδαίες πλατφόρμες ηλεκτρονικού εμπορίου μέχρι και σήμερα

1.2 Φωνητικό Εμπόριο

Το φωνητικό εμπόριο είναι ένα υποσύνολο του ηλεκτρονικού εμπορίου που προσφέρει στους καταναλωτές τη δυνατότητα να χρησιμοποιούν τη φωνή τους για να πραγματοποιήσουν συναλλαγές δίνοντας φωνητικές εντολές, μέσω εικονικών βοηθών ή ζωντανής υποστήριξης καθώς επίσης με online chat ή chatbot. Μπορούν να χρησιμοποιηθούν τερματικές συσκευές όπως για παράδειγμα τα Amazon Echo και Google Home και υπολογιστές ή κινητές συσκευές με ενσωματωμένο λογισμικό εικονικού βοηθού όπως η Alexa της Amazon, Google Assistant της Google και Siri της Apple.

Εικονικός βοηθός (Virtual Assistant, VA) ή προσωπικός βοηθός (Personal Assistant, PA) είναι ένα λογισμικό που μπορεί να ερμηνεύσει την ανθρώπινη ομιλία και να απαντήσει με συνθετική φωνή, συχνά χρησιμοποιώντας τεχνικές τεχνίτης νοημοσύνης και επεξεργασίας φυσικής γλώσσας (Natural Language Processing, NLP). Οι χρήστες μπορούν να κάνουν ερωτήσεις στους VAs, «να ελέγξουν τον αυτοματισμό του σπιτιού τους και την

αναπαραγωγή πολυμέσων μέσω της φωνής τους, και να διαχειριστούν βασικές εργασίες όπως email, λίστες υποχρεώσεων και ημερολόγια μέσω φωνητικών εντολών» [3] .

Chatbot (bot) είναι ένα πρόγραμμα υπολογιστή που ανταποκρίνεται σαν μια έξυπνη οντότητα όταν κάποιος συνομιλεί μαζί του είτε με γραπτά είτε με ηχητικά μηνύματα [4] . Είναι ένας πολύ γενικός όρος, στον οποίο μπορεί να συμπεριληφθεί και ο εικονικός βοηθός και μπορεί να είναι απλό σε άποψη πολυπλοκότητας όπως οι ενέργειές του να είναι προκαθορισμένες βάση κάποιων κανόνων (transactional chatbot) αλλά και πιο σύνθετο βασισμένο σε μηχανική μάθηση και τεχνίτη νοημοσύνη (conversational chatbot) [24] .

1.2.1 Πλεονεκτήματα του φωνητικού εμπορίου

- Ευκολία χρήσης και άνεση

  Εφόσον ο καταναλωτής έχει στη διάθεσή του συσκευή που μπορεί να επικοινωνήσει με ένα κατάστημα είτε μέσω VA ή άλλο τρόπο (εφαρμογή, web browser, κ.α.), να κάνει αγορές ενώ ασχολείται ταυτόχρονα με άλλη εργασία, όπως μαγείρεμα, περπάτημα κ.α. χρησιμοποιώντας μόνο τη φωνή του. Επίσης, οι αγορές με φωνή μπορούν να είναι πιο γρήγορες από τις παραδοσιακές ηλεκτρονικές αγορές αφού δεν γίνεται χειροκίνητη αναζήτηση προϊόντων.

- Εξατομίκευση αγορών

  Με την κατάλληλη τεχνολογία, η επιχείρηση μπορεί να προσφέρει στον καταναλωτή εξατομικευμένες προτάσεις ανάλογα με τα προϊόντα που αγοράζει και τον τρόπο αλληλεπίδρασης με την εφαρμογή, βελτιώνοντας σημαντικά την ικανοποίηση του πελάτη.

- 24/7 Εξυπηρέτηση

  Ο καταναλωτής μπορεί να κάνει αγορές όλες τις ώρες της μέρας, όπως θα μπορούσε με μια ηλεκτρονική σελίδα κάποιου καταστήματος, χρησιμοποιώντας μόνο τη φωνή του.

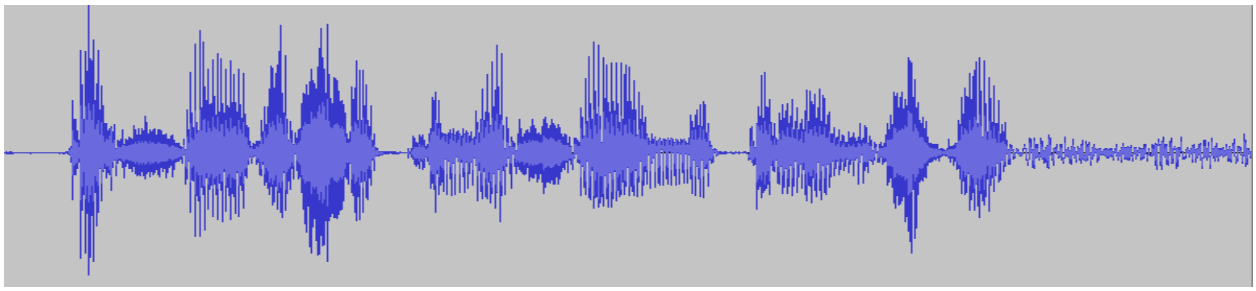
ΚΕΦΑΛΑΙΟ 2 Αυτόματη Αναγνώριση Ομιλίας

Στο παρόν κεφάλαιο θα ασχοληθούμε με τη διαδικασία της αυτόματης αναγνώρισης ομιλίας, θα γίνει περιγραφή της λειτουργίας τριών συστημάτων ανοιχτού κώδικα που είναι ελευθέρα, δωρεάν και διαθέσιμα στο Διαδίκτυο και σύμφωνα με έρευνες θα παρουσιαστούν οι αποδόσεις τους στην αναγνώριση λέξεων σε διαφορετικές γλώσσες.

2.1 Βασικές Έννοιες

2.1.1 Ηχητικό Σήμα

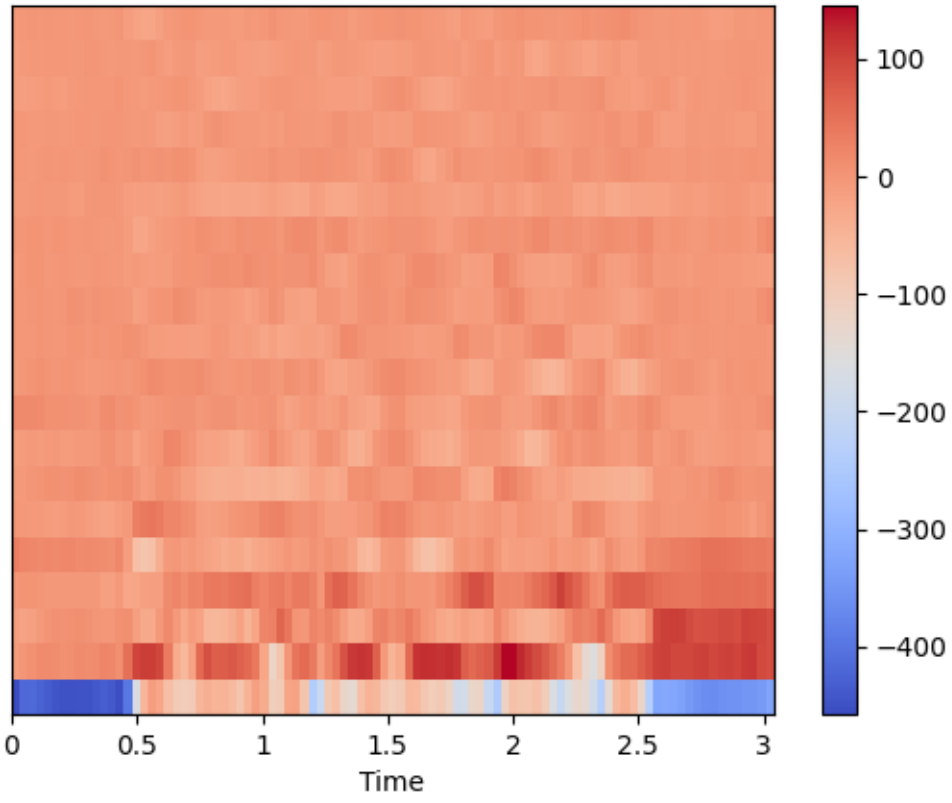
Το ηχητικό σήμα (Εικόνα 1) είναι μια ψηφιακή αναπαράσταση του ήχου που μεταδίδεται σε κάποιο μέσο, όπως ο αέρας, το νερό κ.λπ.. Ένας τρόπος για τη μετατροπή του ήχου σε ψηφιακή μορφή είναι το μικρόφωνο.



Εικόνα 1

2.1.2 Mel-Frequency Cepstral Coefficients

Τα Mel-Frequency Cepstral Coefficients (MFCCs) (Εικόνα 2) αναπαριστούν ένα ηχητικό σήμα μετά την εξαγωγή των χαρακτηριστικών του ήχου που ανήκει στο ακουστό φάσμα (το διάστημα μεταξύ της μικρότερης και της μεγαλύτερης συχνότητας ήχου που μπορεί να ακούσει ένας άνθρωπος) μαζί με μερικές ιδιότητες που έχουν σημασία για την ανθρώπινη ομιλία [13] .Η τεχνική αυτή χρησιμοποιείται από πολλά συστήματα αναγνώρισης ομιλίας, καθώς και από συστήματα αναγνώρισης ομιλητή.



Εικόνα 2

2.1.3 Hidden Markov Model

Το κρυφό μαρκοβιανό μοντέλο (Hidden Markov Model, HMM) [14] αποτελείται από μια ακολουθία κρυφών καταστάσεων και ένα σύνολο ορατών μεταβλητών που προκύπτουν από τις κρυφές καταστάσεις. Το HMM μελετώντας τις μεταβλητές μαθαίνει για τις κρυφές καταστάσεις. Για να γίνει πιο κατανοητό, ας εξετάσουμε ένα υποθετικό σενάριο:

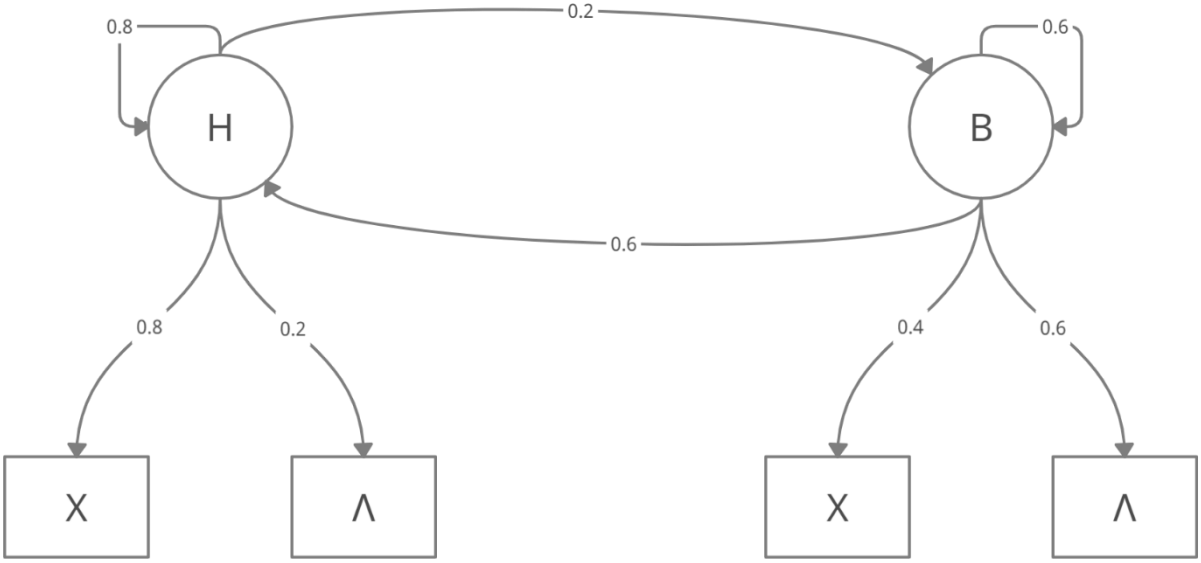
Βρισκόμαστε σε ένα σπίτι χωρίς παράθυρα παρέα με έναν σκύλο. Η διάθεση του σκύλου αλλάζει ανάλογα με τον καιρό έξω από το σπίτι και μπορεί να είναι είτε χαρούμενος (Χ) ή λυπημένος (Λ). Όταν έξω έχει ήλιο (Η), ο σκύλος είναι πιο πιθανό να είναι χαρούμενος παρά λυπημένος, από την άλλη όταν έξω βρέχει (Β) πιθανότατα να είναι λυπημένος παρά χαρούμενος. Ο στόχος μας είναι να μάθουμε τι καιρό έχει έξω (ακολουθία κρυφών καταστάσεων), παρατηρώντας τη διάθεση του σκύλου κάθε μέρα (ορατές μεταβλητές), χωρίς να έχουμε τη δυνατότητα να κοιτάξουμε έξω από το σπίτι.

Στην Εικόνα 3 φαίνεται το HMM με τις κρυφές καταστάσεις Η, Β και τις ορατές μεταβλητές Χ, Λ. Τα βέλη μεταξύ των κύκλων δείχνουν τις μεταβάσεις από την μια κρυφή κατάσταση στην άλλη.

Βασίζοντας στον γράφο της Εικόνα 3, μπορούμε να συμπεράνουμε πως αν ο σκύλος είναι χαρούμενος τότε η πιθανότητα ο καιρός να είναι ηλιόλουστος είναι 0.8 ή 80%.

Μπορούμε επίσης να υπολογίσουμε τι καιρό είχε έξω με βάση μια ακολουθία της διάθεσης του σκύλου, πχ αν ήταν χαρούμενος τη Δευτέρα, χαρούμενος τη Τρίτη και λυπημένος τη Τετάρτη, να βρούμε το καιρό για αυτές τις τρις μέρες.

Τα HMM εφαρμόζονται και στην διαδικασία της αναγνώρισης ομιλίας για να βρουν την πιο πιθανή ακολουθία φωνημάτων (ακολουθία κρυφών καταστάσεων), τις διακριτές φωνητικές μονάδες του φυσικού ήχου που μπορούν να διαφοροποιούν τις λέξεις, για ένα μέρος του ηχητικού σήματος (ορατές μεταβλητές).



Εικόνα 3

2.1.4 Συνάρτηση Softmax

Η συνάρτηση Softmax δέχεται ένα διάνυσμα z από K πραγματικούς αριθμούς και το κανονικοποιεί σε ένα διάνυσμα με K πραγματικούς αριθμούς που ανήκουν στο πεδίο [0,1]. Το άθροισμα όλων των αριθμών του διανύσματος ισούται με τη μονάδα. Συχνά αυτοί οι αριθμοί ερμηνεύονται σαν πιθανότητες.

Η συνάρτηση ορίζεται ως εξής:

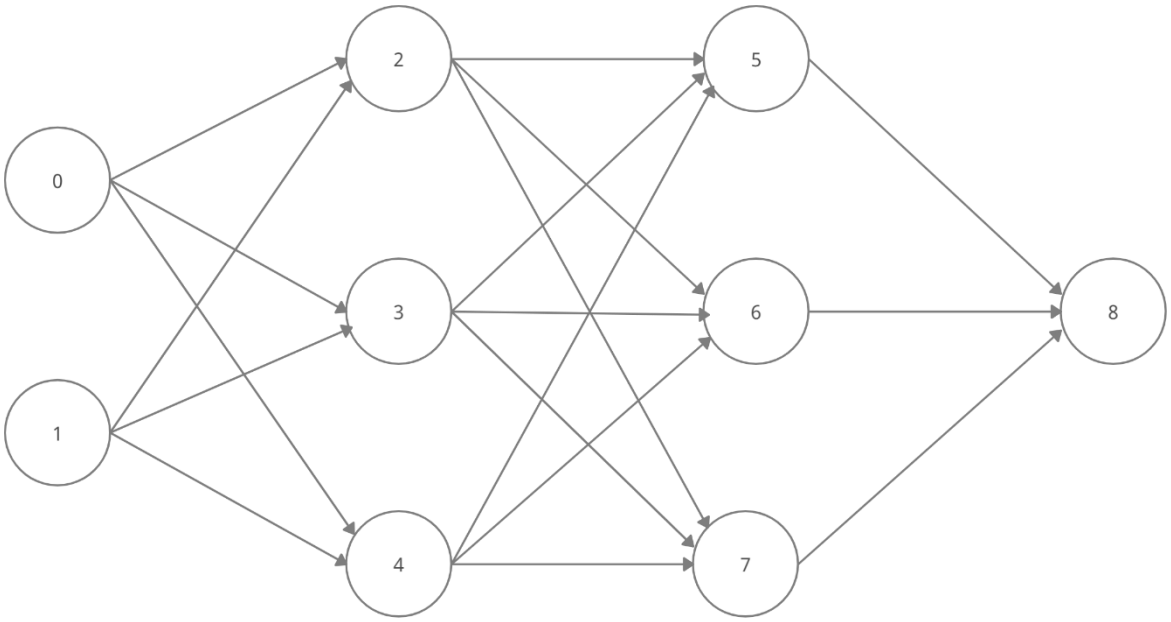
$$\sigma\left(\vec{z}\right)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

για i = 1,2,...,K

Όπου $\vec{z}$ το διάνυσμα εισόδου της συνάρτησης και z_i όλες οι τιμές του διανύσματος z

2.2 Artificial Neural Network

Τα τεχνητά νευρωνικά δίκτυα (Artificial Neural Network, ANN) είναι μια προσπάθεια στην ψηφιακή αναπαράσταση του πραγματικού ανθρώπινου εγκεφάλου. Είναι ένα σύστημα που αποτελείται από νευρώνες (κόμβους) οι οποίοι διασυνδέονται και ομαδοποιούνται σε επίπεδα (layers). Κάθε ANN αποτελείται από ένα επίπεδο εισόδου (input layer) στο οποίο εισέρχεται μια ακολουθία τιμών $[x_1,x_2,\ldots,x_n]$, ένα κρυφό επίπεδο (hidden layer) το οποίο δεν είναι πάντα απαραίτητο να υπάρχει και ένα επίπεδο εξόδου (output layer) το οποίο παράγει μια ακολουθία τιμών $[y_1,y_2,\ldots,y_m]$. Ένα απλό ANN με ένα επίπεδο εισόδου, δύο κρυφά επίπεδα και ένα επίπεδο εξόδου φαίνεται στην εικόνα ANN 1



ANN 1

Όταν μια ακολουθία τιμών φτάνει στο επίπεδο εισόδου, μεταδίδεται από επίπεδο σε επίπεδο μέχρι να φτάσει το επίπεδο εξόδου. Στις συνδέσεις (συνάψεις) μεταξύ των κόμβων υπάρχουν βάρη που στην ουσία δηλώνουν πόσο σημαντική είναι η τιμή που έρχεται από αυτή τη σύνδεση. Όταν κάποιες τιμές φτάνουν σε έναν κόμβο (εκτός του επιπέδου εισόδου), πολλαπλασιάζονται με τα ανάλογα βάρη τους και στη συνέχεια προσθέτονται όλες μαζί. Έπειτα αυτό το άθροισμα γινομένων τροφοδοτείται σε μια συνάρτηση ενεργοποίησης (activation function), η οποία καθορίζει την τιμή που θα μεταδώσει ο κόμβος αυτός στους επόμενους. Υπάρχουν πολλές και διάφορες συναρτήσεις ενεργοποίησης όπως η γραμμική, Rectified Linear Unit (ReLU) κ.α., η οποία επιλέγεται ανάλογα τον σκοπό και τη λειτουργία του ANN.

Το ANN της εικόνας ANN 1 χαρακτηρίζεται και ως ANN πρόσθιας τροφοδότησης (feed-forward), καθώς κάθε επίπεδο τροφοδοτεί το επόμενο μέχρι να καταλήξει στο επίπεδο εξόδου. Το Deep Neural Network (DNN) είναι ένα ANN που περιέχει πολλαπλά κρυφά επίπεδα.

Τα Recurrent Neural Networks (RNNs) είναι μια διαφορετική μορφή των ANN τα οποία μπορούν να «θυμούνται» τις προηγούμενες εισόδους τους κάνοντας έτσι μια καλύτερη εκτίμηση εξόδου για την τωρινή είσοδο.

2.2.6 Εκπαίδευση ANN

Ο σκοπός κάθε ANN είναι να βελτιστοποιήσει τα βάρη έτσι ώστε η έξοδός του να είναι όσο πιο κοντά γίνεται στην επιθυμητή τιμή(ες). Αυτό γίνεται κατά τη διαδικασία της εκπαίδευσης όπου στην έξοδο του δικτύου εφαρμόζεται μια συνάρτηση απώλειας. Η απώλεια είναι ένας τρόπος μέτρησης του πόσο κοντά ήταν η έξοδός του στο αναμενόμενο αποτέλεσμα. Υπάρχουν πάλι πολλές τέτοιες συναρτήσεις και η επιλογή μιας εξαρτάται αποκλειστικά από το σκοπό του ANN. Ωστόσο, μια από αυτές που θα συναντήσουμε και αργότερα είναι η Connectionist Temporal Classification [25] (CTC). Έπειτα, αυτή η απώλεια τροφοδοτείται σε έναν αλγόριθμο που είναι υπεύθυνος για την προσαρμογή των βαρών του ANN. Αυτοί οι αλγόριθμοι ονομάζονται αλγόριθμοι βελτιστοποίησης (optimization algorithms) και ένας που χρησιμοποιείται συχνά είναι ο αλγόριθμος Adam [26] .

Ένα φαινόμενο που μπορεί να εμφανιστεί στη διαδικασία της εκπαίδευσης ενός ANN είναι η υπερ-προσαρμογή (overfitting). Αυτό συμβαίνει όταν το δίκτυο προσαρμόζεται υπερβολικά (ή τέλεια) σε ένα συγκεκριμένο σύνολο δεδομένων και δεν είναι ικανό μετέπειτα να προσαρμοστεί σε νέα σύνολα, μειώνοντας έτσι την ακρίβεια στις προβλέψεις του. Δηλαδή, αν το μοντέλο προσαρμοστεί ακριβώς στα δεδομένα με τα οποία εκπαιδεύτηκε, μαθαίνοντας ακόμα και τον θόρυβο ή γενικά τις αχρείαστες πληροφορίες που αυτά πιθανώς περιέχουν, τότε δεν θα είναι ικανό να κάνει προβλέψεις για δεδομένα που δεν έχει ξαναδεί.

Για να αποφευχθεί η υπερπροσαρμογή έχουν αναπτυχθεί πολλές τεχνικές, δύο από αυτές θα συναντήσουμε και στο επόμενο Κεφάλαιο. Η πρώτη είναι η πρόωρη διακοπή (early stopping) της εκπαίδευσης. Αυτό γίνεται όταν το μοντέλο παύει να βελτιώνεται με τον χρόνο (δεν μειώνεται η απώλεια). Η δεύτερη είναι η «Dropout» [27] , η οποία προσωρινά απενεργοποιεί κάποιους κόμβους εντός του ANN με μια πιθανότητα p.

2.3 Αυτόματη Αναγνώριση Ομιλίας

Αυτόματη αναγνώριση ομιλίας (Automatic Speech Recognition, ASR) είναι μία τεχνολογία που επιτρέπει την επικοινωνία μεταξύ ανθρώπου-υπολογιστή και είναι ενεργό πεδίο έρευνας εδώ και πέντε δεκαετίες [5] .Εναλλακτική ονομασία είναι η ομιλία-σε-κείμενο (Speech-to-Text, STT). Κύριος στόχος της ASR είναι να μεταφράσει την ομιλία ενός ανθρώπου σε κείμενο είτε από ένα αρχείο ήχου ή σε “πραγματικό” χρόνο από μια ροή (stream).

Το πρώτο σύστημα αναγνώρισης ομιλίας δημιουργήθηκε το 1952 από τους Davis, Biddulph και Balashek στο Bell Telephone Laboratories [6] και ήταν ικανό να αναγνωρίσει τα ψηφία 0-9 με ακρίβεια 97-99% ενός μόνο ομιλητή αρσενικού φύλλου. Η αναγνώριση της ομιλίας άλλων ομιλητών απαιτούσε την προσαρμογή του συστήματος για τον συγκεκριμένο ομιλητή.

Το σύστημα συνέκρινε το εισερχόμενο σήμα, την φωνή του ομιλητή όταν έλεγε έναν αριθμό, με γνωστά πρότυπα σήματα για να καθορίσει με ποιο από αυτά μοιάζει πιο πολύ, αναγνωρίζοντας έτσι τον αριθμό που είπε ο ομιλητής. Αυτή η διαδικασία ορίζεται ως pattern matching [6] .

2.3.1 Γενική Αρχιτεκτονική Συστημάτων ASR

Ένα τυπικό σύστημα ASR αποτελείται από τέσσερα βασικά μέρη [5] :

Επεξεργασία σήματος και εξαγωγή χαρακτηριστικών
Ακουστικό μοντέλο (Acoustic Model, AM)
Γλωσσικό μοντέλο (Language Model, LM)
Διαδικασία εύρεσης (Hypothesis Search)

Το μέρος της επεξεργασίας σήματος και εξαγωγής χαρακτηριστικών λαμβάνει ένα ηχητικό σήμα, το βελτιώνει αφαιρώντας τον θόρυβο και τις παραμορφώσεις που τυχόν έχουν να προκύψει από το κανάλι μετάδοσης του ήχου, μετατρέπει το σήμα από τον χώρο του χρόνου στον χώρο της συχνότητας και εξάγει διανύσματα σημαντικών χαρακτηριστικών (πχ MFCC) κατάλληλα για χρήση στο ακουστικό μοντέλο.

Στο ακουστικό μοντέλο, εισέρχονται τα χαρακτηριστικά από το προηγούμενο μέρος, και παράγεται μια βαθμολογία (AM Score) για την ακολουθία χαρακτηριστικών μεταβλητού μεγέθους.

Το γλωσσικό μοντέλο δημιουργεί και αυτό με τη σειρά του μια βαθμολογία (LM Score) υπολογίζοντας την πιθανότητα μιας υποθετικής ακολουθίας λέξεων. Εκπαιδεύεται πάνω σε ένα κείμενο για να μάθει τη συσχέτιση μεταξύ των λέξεων, για παράδειγμα μαθαίνει ποια είναι η πιθανότητα η λέξη w_2 να ακολουθεί τη λέξη w_1 .

$$P(w_2 \mid w_1)$$

Ένα τέτοιο μοντέλο ονομάζεται 2-gram, αλλά μπορεί να γενικευτεί σε n-gram [12] .

Το μέρος της διαδικασίας εύρεσης συνδυάζει τις βαθμολογίες AM Score και LM Score και δίνει την ακολουθία λέξεων με τη μεγαλύτερη βαθμολογία σαν το αποτέλεσμα της αναγνώρισης ομιλίας.

2.4 Λογισμικά Αναγνώρισης Φωνής Ανοιχτού Κώδικα

Λογισμικό ανοιχτού κώδικα σημαίνει ότι ο πηγαίος κώδικας του είναι διαθέσιμος (πχ στο Διαδίκτυο) για τροποποίηση και αναδιανομή από όλους με μια άδεια (license) που συνήθως το συνοδεύει. Υπάρχουν πολλά συστήματα ASR ανοιχτού κώδικα διαθέσιμα μερικά από αυτά είναι τα Kaldi, CMU Sphinx και Mozilla's DeepSpeech που θα δούμε παρακάτω.

2.4.1 Kaldi

Το Kaldi πρωτοεμφανίστηκε το 2009 στο Johns Hopkins University σε ένα εργαστήριο με όνομα "Low Development Cost, High Quality Speech Recognition for New Languages and Domains", με κύριο στόχο τη μοντελοποίηση χρησιμοποιώντας τον αλγόριθμο Subspace Gaussian Mixture Model (SGMM) και βασιζόταν στο Hidden Markov Model Toolkit (HTK), ένα toolkit που χειρίζεται Hidden Markov Models (HMM). Έχει σχεδιαστεί και προτείνεται για χρήση από ερευνητές στο πεδίο της αναγνώρισης ομιλίας.

Η δομή ενός Kaldi μοντέλου αποτελείται από τα παρακάτω μέρη: [7]

Feature Extraction
Acoustic Model
Phonetic Decision Trees
Language Model (LM)
Decoder

Για την εξαγωγή χαρακτηριστικών από ένα ηχητικό σήμα, το Kaldi υποστηρίζει τα Mel-Frequency Cepstral Coefficients (MFCC), Cepstral Mean and Variance Normalization (CMVN), i-Vectors [8] και Perceptual Linear Prediction (PLP) χαρακτηριστικά, με τη δυνατότητα παραμετροποίησης ορισμένων μεταβλητών στην διαδικασία εξαγωγής τους.

Για το ακουστικό μοντέλο, υποστηρίζει μοντέλα GMMs και SGMMs, αλλά είναι σχεδιασμένο έτσι ώστε να μπορεί να υποστηρίξει και νέα είδη μοντέλων. Εκτός αυτών, έχουν κυκλοφορήσει κώδικες που λειτουργούν σαν frameworks για DNN μοντέλα, τα "nnet1", "nnet2" και το πιο πρόσφατο "nnet3".Βασισμένο στο "nnet3", τα "chain" μοντέλα, όπως τα ονομάζει η επίσημη σελίδα του Kaldi, είναι στην ουσία σαν τα υβριδικά DNN-HMM με μία διαφορά στη διαδικασία εκπαίδευσής τους. Υπάρχουν έτοιμα εκπαιδευμένα "chain" μοντέλα στη σελίδα του Kaldi.

2.4.2 CMU Sphinx

Το CMU Sphinx αναπτύχθηκε στο Carnegie Mellon University, ξεκινώντας από την πρώτη έκδοση Sphinx και φτάνοντας στην πιο πρόσφατη, την Sphinx 4.Τα εργαλεία που παρέχει το CMU Sphinx σήμερα είναι τα εξής:

Sphinx4: ένα ευέλικτο και τροποποιήσιμο framework γραμμένο στη γλώσσα προγραμματισμού Java

Pocketsphinx: μια πιο “ελαφριά” έκδοση του Sphinx4 γραμμένη στη C για χρήση σε συσκευές με μικρότερη υπολογιστική ισχύ, όπως κινητά τηλέφωνα

Sphinxbase: μια βιβλιοθήκη που απαιτείται από το Pocketsphinx

Sphinxtrain: εργαλεία για την εκπαίδευση ακουστικών μοντέλων

Το Sphinx4 αποτελείται από τρία βασικά μέρη [9] :

Το FrontEnd λαμβάνει ένα ή περισσότερα ηχητικά σήματα και τα επεξεργάζεται ώστε να εξάγει χαρακτηριστικά, όπως MFCC και PLP.

Το Linguist παράγει δεδομένα σε μορφή γράφου αναζήτησης (“SearchGraph”) για τον αποκωδικοποιητή. Χρησιμοποιεί γλωσσικό μοντέλο (Language Model), ακουστικό μοντέλο (Acoustic Model) και λεξικό (Dictionary) για το σκοπό αυτό.

Για το Language Model μπορούν να χρησιμοποιηθούν N-Gram μοντέλα

Το Acoustic Model κάνει μια αντιστοίχιση μεταξύ μιας μονάδας ομιλίας και ενός HMM.

Το Dictionary παρέχει τις προφορές των λέξεων που βρίσκονται στο Language Model και η χρήση του είναι προαιρετική.

Ο Decoder χρησιμοποιώντας τα χαρακτηριστικά από το FrontEnd και τον γράφο αναζήτησης από το Linguist παράγει αποτελέσματα “Results”.

2.4.3 Mozilla’s DeepSpeech

Το DeepSpeech [10] ανακοινώθηκε το 2014 από ερευνητές του Baidu Research – Silicon Valley AI Lab, και έκτοτε έχει αλλάξει από πολλές απόψεις σε σχέση με το σύστημα που παρουσιάστηκε αρχικά.

Ο πυρήνας ενός μοντέλου DeepSpeech είναι ένα αναδρομικό νευρωνικό δίκτυο (Recurrent Neural Network, RNN) που δέχεται σαν είσοδο ένα ηχητικό σήμα το οποίο χωρίζεται σε μικρότερα κομμάτια (της τάξης των millisecond), εξάγει τα MFCC χαρακτηριστικά, και σαν τελικό αποτέλεσμα παράγει την μετάφραση του ήχου σε κείμενο.

Το RNN μοντέλο αποτελείται από 6 επίπεδα, τα πρώτα τρία είναι δίκτυα πρόσθιας τροφοδότησης (feed-forward). Το τέταρτο επίπεδο είναι ένα αμφίδρομο RNN επίπεδο και το πέμπτο επίπεδο είναι πάλι δίκτυο πρόσθιας τροφοδότησης. Το έκτο και τελευταίο είναι το επίπεδο εξόδου. Σαν είσοδος στο μοντέλο είναι τα MFCC χαρακτηριστικά κάθε κομματιού του αρχικού σήματος.

Το επίπεδο εξόδου αποτελείται από τόσους κόμβους όσο τα γράμματα της γλώσσας που αναγνωρίζει το μοντέλο. Για παράδειγμα, αν το μοντέλο αναγνώριζε την ελληνική γλώσσα, το επίπεδο εξόδου θα περιείχε 24 κόμβους έναν για κάθε γράμμα από το α-ω. Ο κάθε κόμβος είναι μια αριθμητική τιμή logit, μια τιμή που αντιπροσωπεύει τη πιθανότητα να εμφανιστεί ο χαρακτήρας που αντιστοιχεί στον συγκεκριμένο κόμβο. Υπολογίζεται ως εξής:

$$logit = \log\left(\frac{p}{1-p}\right)$$

όπου p η πιθανότητα εμφάνισης ενός χαρακτήρα.

Κατά τη διάρκεια της εκπαίδευσης, χρησιμοποιείται η συνάρτηση απώλειας (loss function) Connectionist Temporal Classification (CTC) [25] μαζί με τον Adam optimizer [26] .

Όταν ένα μοντέλο χρησιμοποιείται για την αναγνώριση ομιλίας, εφαρμόζεται η συνάρτηση softmax σε κάθε κόμβο του επιπέδου εξόδου, και επιλέγεται το γράμμα με την μεγαλύτερη πιθανότητα. Αυτή η διαδικασία επαναλαμβάνεται για κάθε κομμάτι του αρχικού σήματος με αποτέλεσμα να έχουμε μια σειρά από γράμματα, δηλαδή λέξεις ή προτάσεις. Ωστόσο το αποτέλεσμα δεν είναι τίποτα άλλο πέρα από μια σειρά από γράμματα χωρίς γνώσεις των συμφραζομένων. Ένα παράδειγμα [10] είναι η πρόταση «are there any tickets for the game» που μεταφράζεται ως «arther n tickets for the game». Σαν δεύτερο παράδειγμα, η πρόταση «I read a book» μπορεί να μεταφραστεί ως «I red a book» καθώς οι λέξεις “read” και “red” προφέρονται με τον ίδιο τρόπο.

Για αυτούς τους λόγους, το DeepSpeech έχει ενσωματώσει έναν αλγόριθμο Beam Search για να βελτιώσει τις μεταφράσεις, αλλά υποστηρίζει και προαιρετικά εξωτερικά γλωσσικά μοντέλα που ονομάζει “scorer”. Αυτοί οι “scorer” υλοποιούνται με τη βοήθεια της βιβλιοθήκης KenLM [11] και μερικών εργαλείων που παρέχει το ίδιο.

2.5 Σύγκριση Kaldi – CMU Sphinx – Mozilla’s DeepSpeech

Σε αυτό το υποκεφάλαιο θα παρουσιαστούν μερικά αποτελέσματα ερευνών που συγκρίνουν τα τρία λογισμικά. Για τις συγκρίσεις των μοντέλων χρησιμοποιείται η μετρική Word Error Rate (WER) που δηλώνει με τι ακρίβεια το μοντέλο μπόρεσε να αναγνωρίσει μια λέξη (η μικρότερη τιμή είναι καλύτερη) και η Character Error Rate (CER) που δηλώνει με τι ακρίβεια το μοντέλο μπόρεσε να αναγνωρίσει έναν χαρακτήρα (η μικρότερη τιμή είναι καλύτερη). Από τις παρακάτω έρευνες, θα αγνοήσουμε τα αποτελέσματα των συστημάτων που δεν έχουν περιγραφτεί στο υποκεφάλαιο 2.4.

2.5.1 Περιγραφη Αποτελεσματων Ερευνων

Έρευνα 1: [15]

Η εργασία [15] γράφτηκε από τον συγγραφέα Basem Rizk με τίτλο «Evaluation of State Of Art Open-source ASR Engines with Local Inferencing»

Σε αυτή την εργασία ο συγγραφέας συγκρίνει τα λογισμικά Kaldi, Mozilla's DeepSpeech και Facebook 's Wav2letter στα Αγγλικά. Στην περίπτωση του DeepSpeech, χρησιμοποιεί ένα μοντέλο για την πρώτη έκδοση [10] και ένα για την δεύτερη [16] (DeepSpeech 2) (με όνομα PaddlePaddle DeepSpeech).

Να σημειωθεί ότι η έκδοση του πρώτου μοντέλου DeepSpeech είναι η v0.5.1.

Σε ένα σετ δεδομένων, το Kaldi πέτυχε 19% WER, ενώ το DeepSpeech 29-31%.
Σε δεύτερο σετ, το Kaldi πέτυχε 17% WER ενώ το DeepSpeech 40%.

Συμπέρανε λοιπόν ότι το Kaldi είχε την καλύτερη ακρίβεια ενώ το DeepSpeech τη χειρότερη. Από την άλλη το DeepSpeech ήταν το πιο εύκολο στη χρήση.

Έρευνα 2: [17]

Αυτή η έρευνα [17] από τους συγγραφείς Balenko M.V. και Balakshin P.V. με τίτλο «COMPARATIVE ANALYSIS OF SPEECH RECOGNITION SYSTEMS WITH OPEN CODE» συγκρίνει τα συστήματα CMU Sphinx, Kaldi, HTK, Julius, iAtros, RWTH ASR στα Αγγλικά.

Το Kaldi έδειξε την καλύτερη ακρίβεια με 6.5% WER με ελάχιστα πιο αργή ταχύτητα αναγνώρισης από το CMU Sphinx.
Για τη πλατφόρμα CMU Sphinx, εξετάστηκαν ένα μοντέλο Pocketsphinx με 21.4% WER και ένα Sphinx4 μοντέλο με 22.7% WER, και τα δύο έδειξαν μέτρια ακρίβεια αλλά με την γρηγορότερη ταχύτητα αναγνώρισης για το μοντέλο Pocketsphinx.

Έρευνα 3: [18]

Στην έρευνα [18] με τίτλο «Comparing Open-Source Speech Recognition Toolkits ⋆» οι συγγραφείς Christian Gaida , Patrick Lange , Rico Petrick , Patrick Proba , Ahmed Malatawy , και David Suendermann-Oeft συγκρίνουν τα HDecode, Julius, Pocketsphinx, Sphinx4 και Kaldi στα Αγγλικά και Γερμανικά.

Στα Αγγλικά, το Kaldi είχε το καλύτερο WER 6.5%, το Pocketsphinx είχε 21.4% WER και το Sphinx4 22.7% WER.

Στα Γερμανικά, το Kaldi έρχεται πάλι πρώτο με 12.7% WER και το Pocketsphinx με 23.9% WER ξεπερνά ξανά το Sphinx4 που πέτυχε 26.9% WER.

Έρευνα 4: [19]

Στην έρευνα [19] με τίτλο «Automatic Speech Recognition in Taxi Call Service Systems» οι συγγραφείς Samir Rustamov, Natavan Akhundova και Alakbar Valizada συγκρίνουν τα Kaldi και CMU Sphinx στα Αζέρικα.

Το Kaldi έδειξε ακρίβεια μεταξύ 97.3 και 99.6% ενώ το CMU Sphinx ήταν μεταξύ 95.6 και 97.8 %, συμπεραίνοντας ότι το μοντέλο Kaldi ήταν το καλύτερο, αλλά το CMU Sphinx ήταν πιο γρήγορο στη διαδικασία εκπαίδευσης.

2.5.2 Παρατηρήσεις

Η πρώτη παρατήρηση, όχι απαραίτητα αποκλειστικά από τις προηγούμενες τέσσερις έρευνες, είναι ότι δεν υπάρχουν πολλές έρευνες που συγκρίνουν το DeepSpeech με άλλα συστήματα/μοντέλα.

Η δεύτερη παρατήρηση, βασισμένος στην Έρευνα 1 [15] αλλά και στην προσωπική μου εμπειρία κατά την αναζήτηση ενός συστήματος που θα χρησιμοποιηθεί σε αυτή την εργασία για την αναγνώριση ομιλίας, είναι ότι το DeepSpeech ήταν το πιο εύκολο στην χρήση του, στην εγκατάσταση, εκπαίδευση και χρήση στο τελικό πρόγραμμα. Στην τελευταία έκδοσή του, 0.9.3, σύμφωνα με τη σελίδα του DeepSpeech στο github[1] το μοντέλο έχει WER 7.06%, σημαντικά καλύτερο από την Έρευνα 1 [15] ,βέβαια η εκπαίδευση του μοντέλου έγινε σε διαφορετικά δεδομένα άρα η σύγκριση μπορεί να μην είναι απόλυτα δίκαιη. Συνεπώς επιλέχθηκε το DeepSpeech για την αναγνώριση ομιλίας στην εφαρμογή που αναπτύσσουμε σε αυτή την εργασία.

1 https://github.com/mozilla/DeepSpeech/releases/tag/v0.9.3

ΚΕΦΑΛΑΙΟ 3 Εκπαίδευση ενός μοντέλου DeepSpeech

Σε αυτό το κεφάλαιο θα πειραματιστούμε με την εκπαίδευση των ακουστικών μοντέλων του DeepSpeech και θα δημιουργήσουμε ένα γλωσσικό μοντέλο που θα συνεργάζεται με το τελικό ακουστικό για αποτελεσματική και με ακρίβεια αναγνώριση ομιλίας σε ένα μικρό σύνολο δεδομένων.

3.1 Δημιουργώντας το περιβάλλον εκπαίδευσης

Τα προαπαιτούμενα για την εκπαίδευση του μοντέλου είναι το λειτουργικό σύστημα Ubuntu 18.04 και η γλώσσα προγραμματισμού Python 3.6. Για την εγκατάσταση του Ubuntu 18.04 μπορεί να χρησιμοποιηθεί μια εικονική μηχανή (Virtual Machine, VM).

Οι εικονικές μηχανές είναι στην ουσία ένας (εικονικός) υπολογιστής μέσα σε έναν άλλο. Υπάρχουν πολλοί τρόποι να δημιουργήσει κανείς ένα VM, ο πιο απλός ενδεχομένως είναι με τη χρήση δωρεάν λογισμικών που επιτρέπουν την εύκολη δημιουργία, επεξεργασία και διαχείριση αυτών. Δύο από αυτά είναι τα Oracle's VM VirtualBox[2] (ανοιχτού κώδικα) και το VMware[3]. Σε αυτή την εργασία χρησιμοποιήθηκε το VirtualBox. Το VM με Ubuntu 18.04 έχει πρόσβαση σε 8GB RAM και τέσσερις πυρήνες του επεξεργαστή i7-6700K στα 4.0GHz.

Είναι δυνατόν να αξιοποιηθεί μια κάρτα γραφικών (ή περισσότερες) που επιταχύνει σημαντικά τον χρόνο εκπαίδευσης του μοντέλου.

Το Ubuntu 18.04 έρχεται με προ εγκατεστημένη τη γλώσσα προγραμματισμού Python 3.6.9.

Ακολουθώντας το επίσημο εγχειρίδιο (documentation) του DeepSpeech για την εκπαίδευση μοντέλων [20] μερικές βιβλιοθήκες που έλειπαν από την καθαρή εγκατάσταση του Ubuntu 18.04 ήταν οι python3-venv και git. Μπορούν να εγκατασταθούν με τις εντολές:

```
sudo apt install python3-venv

sudo apt install git
```

Πιο συγκεκριμένα, τα πεδία της σελίδας[4] [20] που αφορούν την εγκατάσταση είναι τα παρακάτω:

«Prerequisites for training a model»

«Getting the training code»

[2] https://www.virtualbox.org/

[3] https://www.vmware.com/

[4] https://deepspeech.readthedocs.io/en/r0.9/TRAINING.html

«Creating a virtual environment»
«Activating the environment»
«Installing DeepSpeech Training Code and its dependencies»

Επειδή δεν υπάρχει διαθέσιμη κάρτα γραφικών στο VM μας, οι εντολές από τα πεδία «Recommendations» και «Basic Dockerfile for training» της σελίδας δεν εκτελέστηκαν.

3.2 Συλλογή Δεδομένων

Το Common Voice [21] είναι ένα έργο της Mozilla που προσκαλεί χρήστες του Διαδικτύου να δωρίσουν τις φωνές τους ηχογραφώντας μικρές προτάσεις και να ελέγξουν τις ηχογραφήσεις άλλων χρηστών για την ορθότητά τους. Τα δεδομένα αυτά γίνονται διαθέσιμα για όλους και μέχρι στιγμής (Common Voice Corpus 7.0, 21-07-2021) για την αγγλική γλώσσα έχουν μαζευτεί 2637 ώρες ηχογραφήσεων και για την ελληνική 23 ώρες.

Τα ελληνικά δεδομένα ήχου δεν θα χρησιμοποιηθούν καθώς λόγω της περιορισμένης υπολογιστικής ισχύς του VM καθυστερεί σημαντικά η εκπαίδευση του μοντέλου.

Τα συνολικά ηχητικά δεδομένα που συλλέχθηκαν κατά τη διάρκεια αυτής της πτυχιακής εργασίας έχουν συνολική διάρκεια 00:12:48, αποτελούνται από κυρίως μικρές προτάσεις 2-5 δευτερολέπτων διάρκειας η κάθε μια και περιέχουν (3) μόνο αρσενικές φωνές. Το μοντέλο μετά την εκπαίδευση θα είναι ικανό να αναγνωρίσει ένα μικρό αριθμό μοναδικών λέξεων, συγκεκριμένα 59 λέξεις.

Το DeepSpeech υποστηρίζει .wav αρχεία με μονό κανάλι και 16000 (16kHz) ρυθμό δειγματοληψίας (Sample Rate) με κωδικοποίηση (encoding) 16 bit PCM.

Όλα τα δεδομένα ηχογραφήθηκαν, σύμφωνα με τις απαιτήσεις του DeepSpeech παραπάνω, από μέτριο μικρόφωνο σε περιβάλλον χωρίς θόρυβο και με καθαρή ομιλία.

3.3 Διαχείριση των Δεδομένων

Το DeepSpeech για την εκπαίδευση μοντέλων απαιτεί την ύπαρξη τριών .csv αρχείων. Τα train.csv (training), test.csv (testing), dev.csv (development). Επομένως από τα ηχητικά δεδομένα μας πρέπει να παράγουμε αυτά τα τρία αρχεία.

Το csv (comma-separated values) είναι ένα αρχείο κειμένου που περιέχει τιμές που χωρίζονται από **κόμμα**. Οι τιμές ονομάζονται πεδία (fields) και κάθε γραμμή του αρχείου μπορεί να έχει ένα ή περισσότερα πεδία, αλλά όλες οι γραμμές αναγκαστικά έχουν τον ίδιο αριθμό πεδίων και με την ίδια σειρά. Η πρώτη γραμμή μπορεί να είναι τα ονόματα των πεδίων που ακολουθούν και σε αυτή τη περίπτωση τα ονόματα δεν διαβάζονται σαν τιμές.

Παράδειγμα csv αρχείου:

```
field1,field2,field3,field4
row1value1,row1value2,row1value3,row1value4
row2value1,row2value2,row2value3,row2value4
```

Τα αρχεία που πρέπει να παράγουμε έχουν τα πεδία με τη σειρά (στη πρώτη γραμμή):

wav_filename, το όνομα του αρχείου
wav_filesize, το μέγεθος του αρχείου σε bytes
transcript, τη μετάφραση της ομιλίας που αντιστοιχεί στο συγκεκριμένο αρχείο

Για την δημιουργία των αρχείων, γράφτηκε ένα script (create_csv_files.py) στην Python 3.7. Δέχεται σαν προαιρετικές παραμέτρους από την γραμμή εντολών (command line arguments) τα εξής

-p ή --path : Το μονοπάτι στο οποίο βρίσκονται τα αρχεία των δεδομένων ήχου. Το προεπιλεγμένο μονοπάτι είναι “audio_data/”.
-ro : Τυχαιοποιεί τη σειρά των αρχείων. Απενεργοποιημένο από προεπιλογή.
-r : Τυχαιοποιεί ποια αρχεία θα χρησιμοποιηθούν για τα training, testing και development με πιθανότητες 70/10/20 % αντίστοιχα. Απενεργοποιημένο από προεπιλογή.
-ts ή –transcripts : Το μονοπάτι στο οποίο βρίσκεται το .txt αρχείο με τις μεταφράσεις των ηχογραφήσεων. Το προεπιλεγμένο μονοπάτι είναι “audio_data/transcripts.txt”

και δημιουργεί τα αρχεία training.csv, testing.csv, dev.csv, με τη σωστή δομή και δεδομένα, στο μονοπάτι που ορίζεται με τη παράμετρο -p ή --path.

Το script βασίζεται σε δύο προϋποθέσεις:

Υπάρχει ένα αρχείο κειμένου, έστω transcripts.txt, του οποίου κάθε γραμμή έχει τη μορφή [αριθμός] [κενό] [πρόταση]. Η πρόταση συμβολίζει την μετάφραση ενός .wav αρχείου. Ο αριθμός είναι το αναγνωριστικό της πρότασης.
Τα ονόματα των .wav αρχείων πρέπει να έχουν την μορφή [χαρακτήρες][αριθμός]. Οι χαρακτήρες μπορούν να είναι μόνο γράμματα (πχ a-z A-Z). Ο αριθμός αντιστοιχεί σε μια μετάφραση από το αρχείο transcripts.txt.
Για παράδειγμα, αν η πρώτη γραμμή του αρχείου transcripts.txt είναι “1 σήμερα έχει καλό καιρό” τότε για ένα αρχείο abcd1.wav το πρόγραμμα γνωρίζει ότι μεταφράζεται ως “σήμερα έχει καλό καιρό”.

Όλοι οι χαρακτήρες που περιέχονται στις μεταφράσεις των ηχογραφήσεων (transcripts.txt) αποτελούν το αλφάβητο του μοντέλου μας. Έτσι, ορίζεται το αλφάβητο με τα ελληνικά γράμματα α-ω καθώς και τα επτά φωνήεντα με τόνους ά-έ-ή-ί-ό-ύ-ώ. Στην περίπτωση που οι μεταφράσεις περιείχαν χαρακτήρες εκτός του αλφαβήτου θα παρουσιαζόταν σφάλμα κατά την εκπαίδευση του μοντέλου. Επίσης, τα κεφαλαία γράμματα αποτελούν διαφορετικούς χαρακτήρες από τα μικρά και δεν συμπεριλαμβάνονται στην αλφάβητο. Τέλος, στην αλφάβητο πρέπει να συμπεριλαμβάνεται και ο κενός χαρακτήρας.

Η εφαρμογή που θα αναπτυχθεί στο Κεφάλαιο 4 θα αναγνωρίζει ναι μεν ελληνικά, αλλά πολλά ονόματα προϊόντων στην πραγματικότητα είναι πιθανό να περιέχουν αγγλικές

λέξεις. Για αυτό το λόγο, στο μοντέλο συμπεριλαμβάνουμε τις ελληνικές προφορές αγγλικών λέξεων (με ελληνική γραμματοσειρά). Για παράδειγμα, την αγγλική λέξη “road” μπορούμε να την γράψουμε σαν “ρόουντ” ή την “Gigabyte” σαν τις δύο λέξεις “γκίγκα μπάιτ”. Έτσι το λεξιλόγιο αποτελείται από ελληνικές λέξεις και από τις ελληνικές προφορές αγγλικών λέξεων γραμμένες με ελληνικά γράμματα.

Για την αλλαγή του αλφάβητου, αρκεί να αντικατασταθεί το αρχείο “alphabet.txt” στο μονοπάτι “DeepSpeech/data/” εφόσον έχει εγκατασταθεί ο φάκελος “DeepSpeech” σύμφωνα με το εγχειρίδιο χρήσης του DeepSpeech [20] .

3.4 Εκπαίδευση Ακουστικού Μοντέλου

Κατά τη διάρκεια αυτής της εργασίας, εκπαιδεύτηκαν συνολικά 18 μοντέλα, δοκιμάζοντας κάθε φορά νέες παραμέτρους και νέα δεδομένα. Καθοριστικής σημασίας ήταν η χρήστη του script του υποκεφαλαίου 3.3, που έκανε εφικτή την γρήγορη προσθήκη νέων δεδομένων στα τρία .csv αρχεία, και την τυχαιοποίηση αυτών. Μετά από όλες τις δοκιμές, παρατηρήθηκε ότι η καλύτερη κατανομή των δεδομένων για τα training,testing και development δεδομένα ήταν 70/10/20 % αντίστοιχα και ταυτόχρονα χωρίς την τυχαιοποίηση της σειρά εμφάνισής τους.

Οι παράμετροι που χρησιμοποιήθηκαν φαίνονται στον πίνακα:

Παράμετρος	**Τιμή**
--n_hidden	1024
--epochs	200
--early_stop	
--es_epochs	40
-dropout_rate	0.4
--train_batch_size	32
--test_batch_size	32
--dev_batch_size	32

Η παράμετρος --n_hidden προσδιορίζει τον αριθμό των κρυφών κόμβων κάθε επιπέδου του RNN. Έγιναν δοκιμές με 2048, 1024 και 512 κόμβους και παρατηρήθηκε ότι η εκπαίδευση του μοντέλου με 2048 είναι αρκετά αργή, ακόμα και για το μικρό πλήθος δεδομένων που χρησιμοποιήθηκαν, με 512 κόμβους η ακρίβεια των μοντέλων μειώνεται καθώς τα δεδομένα αυξάνονται ενώ με 1024 κόμβους η ακρίβεια είναι πολύ καλή και ο χρόνος της εκπαίδευσης σχετικά μικρός.

Μια αναλυτική λίστα παραμέτρων παρουσιάζεται στο επίσημο εγχειρίδιο[5], και ένας οδηγός για τη διαδικασία της εκπαίδευσης στο “playbook” του DeepSpeech[6].

Στον παρακάτω πίνακα παρουσιάζονται τα αποτελέσματα των τεσσάρων μοντέλων:

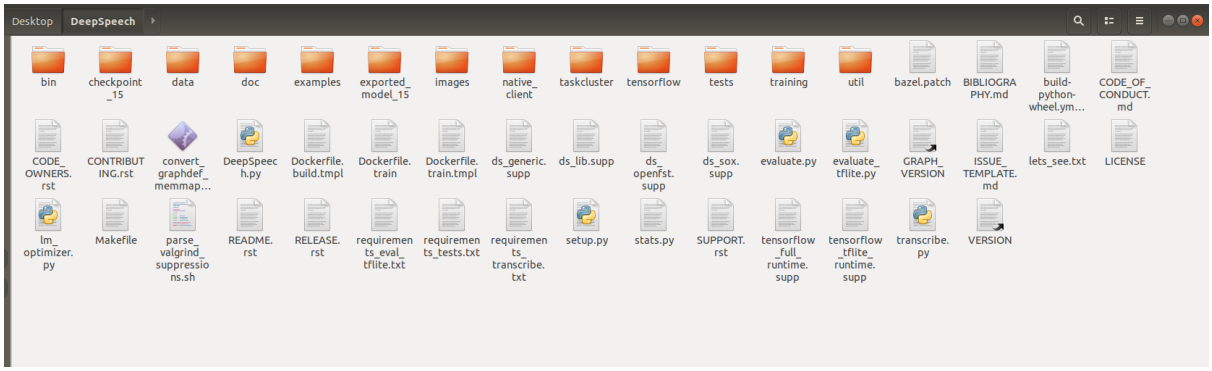
m1024-r: 1024 κόμβοι, με 70/10/20 κατανομή των δεδομένων (-r)
m512-r: 512 κόμβοι με 70/10/20 κατανομή των δεδομένων (-r)
m1024-r-ro: 1024 κόμβοι με 70/10/20 κατανομή των δεδομένων (-r) και τυχαία σειρά (-ro)
m512-r-ro: 512 κόμβοι με 70/10/20 κατανομή των δεδομένων (-r) και τυχαία σειρά (-ro)

Name	WER (%)	CER (%)	Loss
m1024-r	0.75	0.32	60.76
m512-r	0.79	0.31	59.44
m1024-r-ro	0.95	0.47	101.45
m512-r-ro	0.89	0.44	96.02

Τα WER και CER είναι εξαιρετικά χαμηλά λόγω του μικρού λεξιλογίου και του μικρού πλήθους ομιλητών στα ηχητικά δεδομένα.

Αν έχουν ακολουθηθεί οι οδηγίες της σελίδας του DeepSpeech και βρισκόμαστε στο φάκελο DeepSpeech(Εκπαίδευση 1), τότε για την εκπαίδευση του μοντέλου m1024-r για παράδειγμα η εντολή είναι:

```
python3 DeepSpeech.py --train_files audio_data/train.csv --
dev_files audio_data/dev.csv --test_files audio_data/test.csv --
checkpoint_dir checkpoint_1024r --export_dir exported_model_1024r
--epochs 200 --n_hidden 1024 --early_stop --es_epochs 40 -
dropout_rate 0.4 --train_batch_size 32 --test_batch_size 32 --
dev_batch_size 32
```



Εκπαίδευση 1

[5] https://deepspeech.readthedocs.io/en/latest/Flags.html#training-flags
[6] https://mozilla.github.io/deepspeech-playbook/TRAINING.html

Όταν ολοκληρωθεί η εκπαίδευση (Εκπαίδευση 2), στον φάκελο “DeepSpeech” (με την προϋπόθεση ότι η εντολή εκτελέστηκε εντός του μονοπατιού του φακέλου “DeepSpeech”) θα έχουν δημιουργηθεί δύο επιπλέον φάκελοι, οι “checkpoint_1024r” και “exported_model_1024r”, ο πρώτος μπορεί να χρησιμοποιηθεί μελλοντικά για την περαιτέρω εκπαίδευση του μοντέλου και για τη δημιουργία του γλωσσικού μοντέλου, ενώ ο δεύτερος περιέχει το αρχείο “output_graph.pb” το οποίο πρέπει να μετατρέψουμε σε αρχείο .pbmm . Αυτό θα το κάνουμε με τη χρήση του αρχείου “convert_graphdef_memmapped_format” το οποίο μπορούμε να εγκαταστήσουμε με την εντολή

```
python3 util/taskcluster.py --source tensorflow --artifact
convert_graphdef_memmapped_format --branch r1.15 --target .
```

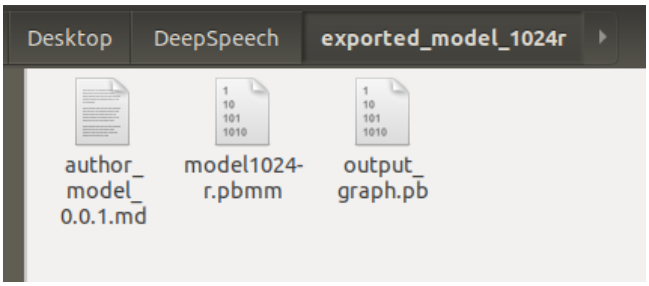
Έπειτα με την ακόλουθη εντολή θα γίνει η μετατροπή:

```
./convert_graphdef_memmapped_format --
in_graph=exported_model_1024r/output_graph.pb --
out_graph=exported_model_1024r/model1024-r.pbmm
```

```
I Early stop triggered as the loss did not improve the last 40 epochs
I FINISHED optimization in 0:59:09.987921
I Loading best validating checkpoint from checkpoint_16/best_dev-300
I Loading variable from checkpoint: cudnn_lstm/rnn/multi_rnn_cell/cell_0/cudnn_compatible_lstm_cell/bias
I Loading variable from checkpoint: cudnn_lstm/rnn/multi_rnn_cell/cell_0/cudnn_compatible_lstm_cell/kernel
I Loading variable from checkpoint: global_step
I Loading variable from checkpoint: layer_1/bias
I Loading variable from checkpoint: layer_1/weights
I Loading variable from checkpoint: layer_2/bias
I Loading variable from checkpoint: layer_2/weights
I Loading variable from checkpoint: layer_3/bias
I Loading variable from checkpoint: layer_3/weights
I Loading variable from checkpoint: layer_5/bias
I Loading variable from checkpoint: layer_5/weights
I Loading variable from checkpoint: layer_6/bias
I Loading variable from checkpoint: layer_6/weights
Testing model on /home/   /Desktop/          /test.csv
Test epoch | Steps: 1 | Elapsed Time: 0:00:21
Test on /home/   /Desktop/           /test.csv - WER: 0.755396, CER: 0.320646, loss: 60.759857
--------------------------------------------------------------------------------
```

Εκπαίδευση 2

Έτσι, στον φάκελο “exported_model_1024r” έχουμε το ακουστικό μοντέλο μας έτοιμο για χρήση (Εκπαίδευση 3).



Εκπαίδευση 3

3.5 Γλωσσικό Μοντέλο

Το γλωσσικό μοντέλο ή "scorer" συσχετίζει τις λέξεις μεταξύ τους, προβλέποντας ποια λέξη είναι πιθανό να ακολουθήσει μετά από κάποια άλλη, βελτιώνοντας έτσι την ακρίβεια του ακουστικού μοντέλου. Το DeepSpeech υποστηρίζει n-gram γλωσσικά μοντέλα που δημιουργούνται με την KenLM βιβλιοθήκη.

Στο σύστημά μας, έπρεπε να χτίσουμε τη KenLM χειροκίνητα τα βήματα που ακολουθήθηκαν είναι τα εξής:

1. Διαγραφή του φακέλου "kenlm" στο μονοπάτι "DeepSpeech/native_client"
2. Εκτέλεση της εντολής [7]

```
git clone https://github.com/kpu/kenlm.git
```

   Και μετακίνηση του φακέλου που δημιουργεί αυτή στον φάκελο "DeepSpeech/native_client"
3. Εκτέλεση των ακόλουθων εντολών στο μονοπάτι "DeepSpeech/native_client/kenlm":

```
sudo apt install build-essential cmake libboost-system-dev
libboost-thread-dev libboost-program-options-dev libboost-
test-dev libeigen3-dev zlib1g-dev libbz2-dev liblzma-dev

mkdir -p build && cd build
cmake ..
make -j 4
```

Οι εντολές μπορούν επίσης να βρεθούν στο αρχείο "BUILDING.txt" στο φάκελο kenlm αμέσως μετά το πρώτο βήμα

Έπειτα θα χρειαστούμε ένα αρχείο κειμένου .txt με λέξεις ή προτάσεις για να βρεθούν οι συσχετίσεις μεταξύ των λέξεων. Κάθε λέξη ή πρόταση πρέπει να βρίσκεται σε ξεχωριστή γραμμή. Το αρχείο μας περιέχει συνολικά 31 προτάσεις που αποτελούνται από τις 59 μοναδικές λέξεις που μπορεί να αναγνωρίσει το ακουστικό μοντέλο. Οι λέξεις στο αρχείο αυτό αποτελούν το λεξιλόγιο του μοντέλου (δεν μπορεί δηλαδή να αναγνωρίσει λέξεις εκτός αυτού).

Ακολουθώντας τα βήματα της σελίδας "playbook"[8] του DeepSpeech που αναλύει τη διαδικασία δημιουργίας "scorer" φτιάχνουμε το αρχείο m1024-r.scorer. Στο Table 3 φαίνονται οι παράμετροι lm_alpha και lm_beta που χρησιμοποιήθηκαν.

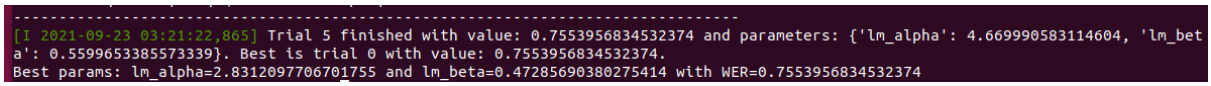
[7] https://github.com/kpu/kenlm
[8] https://mozilla.github.io/deepspeech-playbook/SCORER.html

Παράμετρος	Τιμή
--default_alpha	2.8312097706701755
--default_beta	0.47285690380275414

Εντός του φακέλου “DeepSpeech” εκτελούμε την εντολή

```
python3 lm_optimizer.py --test_files audio_data/test.csv --
checkpoint_dir checkpoint_1024r --n_hidden 1024 --n_trials 6
```

Για να βρούμε τα lm_alpha και lm_beta, το αποτέλεσμα φαίνεται παρακάτω



Εκπαίδευση 4

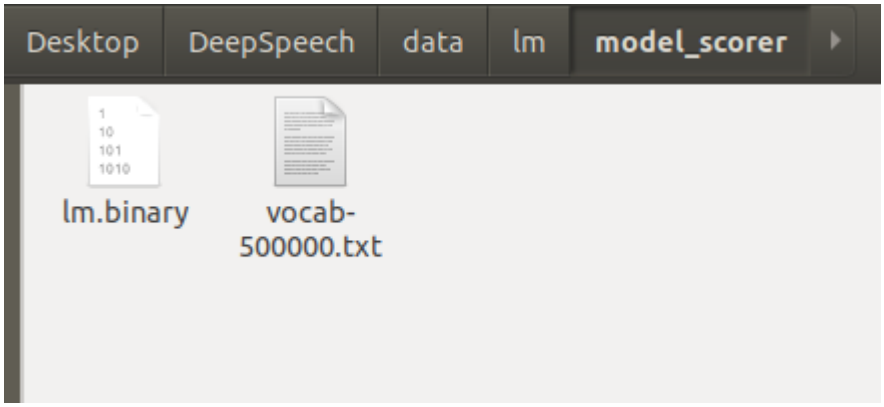
Τώρα, εντός του φακέλου “DeepSpeech/data/lm” δημιουργούμε τον φάκελο “model_scorer” και εκτελούμε την εντολή

```
python3 generate_lm.py --input_txt /home/[όνομα χρήστη]
/Desktop/DeepSpeech/sentences.txt --output_dir model_scorer --
top_k 500000 --kenlm_bins /home/[όνομα χρήστη]
/Desktop/DeepSpeech/native_client/kenlm/build/bin --arpa_order 5
--max_arpa_memory "85%" --arpa_prune "0|0|1" --binary_a_bits 255
--binary_q_bits 8 --binary_type trie --discount_fallback
```

Όπου

- --input_txt = το μονοπάτι στο αρχείο με τις προτάσεις
- --output_dir = το μονοπάτι στο οποίο θα αποθηκεύσει η εντολή την έξοδό της
- --kenlm_bins = το μονοπάτι στο φάκελο “build/bin” εντός του φακέλου KenLM που εγκαταστήσαμε προηγούμενος

Αν όλα πήγαν καλά, το αποτέλεσμα φαίνεται στην εικόνα Εκπαίδευση 5



Εκπαίδευση 5

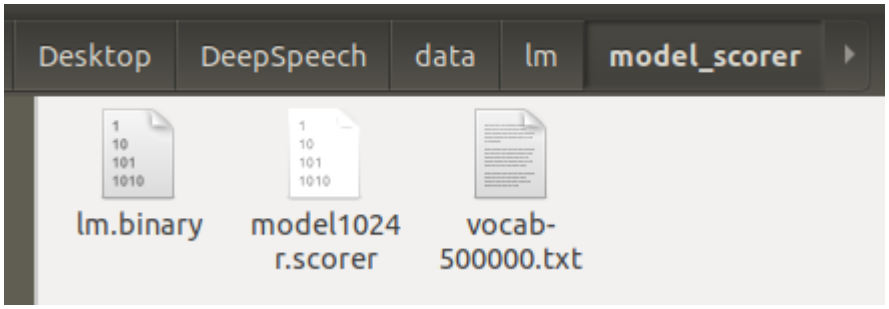
Τελευταίο βήμα είναι να χρησιμοποιήσουμε τα lm_alpha και lm_beta που βρήκαμε προηγουμένως καθώς και τα αρχεία που μόλις δημιουργήθηκαν. Παραμένοντας στον ίδιο φάκελο (“DeepSpeech/data/lm”) εκτελούμε την εντολή

```
./generate_scorer_package --alphabet ../alphabet.txt --lm
model_scorer/lm.binary --vocab model_scorer/vocab-500000.txt --
package model_scorer/model1024r.scorer --default_alpha
2.8312097706701755--default_beta 0.47285690380275414
```

Όπου

- --alphabet = το μονοπάτι για το αρχείο που περιέχει την αλφάβητο του μοντέλου, στην περίπτωσή μας βρίσκεται στο “DeepSpeech/data”
- --lm = το μονοπάτι για το αρχείο “lm.binary” που δημιουργήθηκε προηγουμένως (εικόνα Εκπαίδευση 5)
- --vocab = το μονοπάτι για το αρχείο “vocab-500000.txt” που δημιουργήθηκε προηγουμένως (εικόνα Εκπαίδευση 5)
- --package = το μονοπάτι και όνομα για το τελικό γλωσσικό μοντέλο που θα παράγει η εντολή
- --default_alpha = η τιμή lm_alpha από την εικόνα Εκπαίδευση 4
- --default_beta = η τιμή lm_beta από την εικόνα Εκπαίδευση 4

Τέλος, παράγεται το .scorer αρχείο (εικόνα Εκπαίδευση 6) που θα χρησιμοποιήσουμε ως γλωσσικό μοντέλο.



Εκπαίδευση 6

ΚΕΦΑΛΑΙΟ 4 Επικοινωνία με το Κατάστημα

Το πρώτο μέρος της εφαρμογή μας αφορά τον τρόπο με τον οποίο ο πελάτης επικοινωνεί με το κατάστημα για να κάνει μια παραγγελία. Σε αυτό το πλαίσιο εξετάστηκαν δύο προσεγγίσεις, η επικοινωνία μέσω τηλεφώνου και μέσω ενός φυλλομετρητή (web browser). Η πρώτη μπορεί να υλοποιηθεί με τη χρήση της υπηρεσίας Twilio[9], μια υπηρεσία που μέσα μιας πληθώρας δυνατοτήτων, επιτρέπει σε ένα πρόγραμμα να δέχεται και να πραγματοποιεί τηλεφωνικές κλήσεις καθώς και να λαμβάνει ή να στέλνει μηνύματα όπως SMS. Ωστόσο η χρήση της υπηρεσίας δεν έρχεται χωρίς χρηματικό κόστος.

Έτσι, αποφασίστηκε να υλοποιηθεί η δεύτερη προσέγγιση για την οποία αναπτύχθηκε μια βασική εφαρμογή με την οποία ένας χρήστης (πελάτης - client) επικοινωνεί με έναν διακομιστή (server) στο υποκεφάλαιο 4.2.

4.1 Περιγραφή

Ο καταναλωτής συνδέεται σε μια ιστοσελίδα και του δίνεται η δυνατότητα να πατήσει ένα κουμπί για δώσει εντολές στην εφαρμογή του Κεφαλαίου 5, ή εναλλακτικά να πληκτρολογήσει ένα μήνυμα το οποίο θα σταλθεί σε αυτή. Κάτι τέτοιο γίνεται δυνατό με την αρχιτεκτονική Client-Server (πελάτης-διακομιστής).

Ο πελάτης (client) (μια συσκευή) ζητά μια υπηρεσία από μια άλλη συσκευή, τον διακομιστή (server). Ο διακομιστής (ή εξυπηρετητής) περιμένει μέχρι να λάβει κάποιο αίτημα, το οποίο έπειτα επεξεργάζεται και απαντά αναλόγως στον αποστολέα (πελάτη), με τη δυνατότητα να δέχεται ταυτόχρονα πολλαπλά αιτήματα και να εξυπηρετεί ταυτόχρονα πολλαπλούς πελάτες . Οι δύο αυτές οντότητες συνήθως βρίσκονται σε διαφορετικό δίκτυο χωρίς αυτό να σημαίνει ότι ένας πελάτης και διακομιστής δεν μπορούν να είναι όχι μόνο στο ίδιο δίκτυο αλλά και στην ίδια συσκευή.

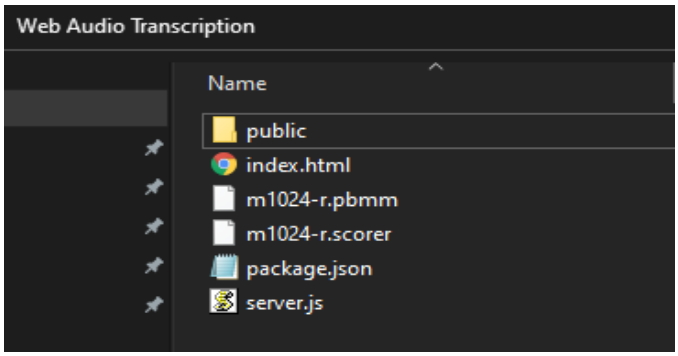
Η εφαρμογή που θα αναπτυχθεί σε αυτό το Κεφάλαιο απαρτίζεται από έναν διακομιστή (server.js) και έναν πελάτη (client.js στον φάκελο Public). Για το σκοπό αυτό θα χρησιμοποιηθούν τα framework Node.js και Express και η γλώσσα προγραμματισμού JavaScript μαζί με τη γλώσσα σήμανσης υπερκειμένου HTML[10] και για να δώσουμε στυλ στη σελίδα τη CSS[11] (Cascading Style Sheets). Επίσης για την ανίχνευση γεγονότων στη φωνή όπως ομιλία ή ησυχία θα χρησιμοποιηθεί το πακέτο node-vad[12].

JavaScript (JS) είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου που χρησιμοποιείται από το 97.6%[13] των ιστοσελίδων του Διαδικτύου και ο κώδικάς της

[9] https://www.twilio.com/
[10] https://www.w3schools.com/html/
[11] https://www.w3schools.com/css/
[12] https://www.npmjs.com/package/node-vad
[13] https://w3techs.com/technologies/details/cp-javascript/

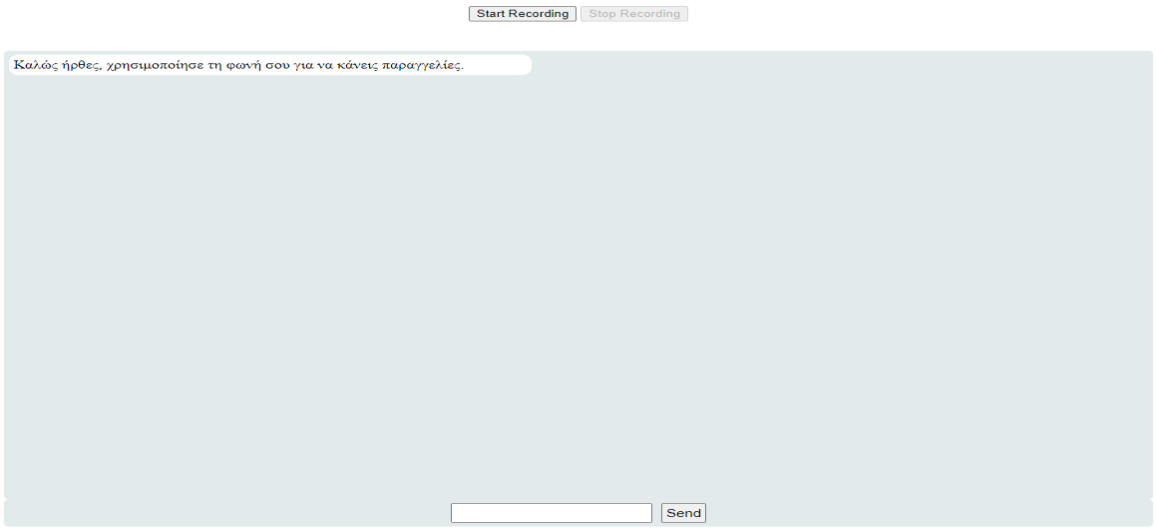
εκτελείται στη συσκευή του πελάτη (client-side). Το Node.js από την άλλη, είναι ένα περιβάλλον εκτέλεσης JavaScript βασισμένο στη μηχανή V8 της Google[14] που εκτελείται στη μεριά του διακομιστή (server-side) η εγκατάσταση του οποίου είναι απλή και έρχεται μαζί με τον διαχειριστή πακέτων npm. Η έκδοση Node που χρησιμοποιήθηκε για την εφαρμογή είναι η 14.17.6 LTS.

Έστω ότι βρισκόμαστε στον φάκελο “Web Audio Transcription” ο οποίος περιέχει τα αρχεία όπως φαίνονται στην Εικόνα 4.



Εικόνα 4

Ανοίγοντας τη γραμμή εντολών και εκτελώντας την εντολή `npm install` θα εγκατασταθούν όλα τα απαραίτητα πακέτα για τη λειτουργία της εφαρμογής σε έναν νέο φάκελο “node_modules”, έπειτα με την εντολή `node server.js` ξεκινάμε τον διακομιστή και μέσω ενός web browser, πηγαίνοντας στη διεύθυνση “localhost:8888” αποκτούμε πρόσβαση στη σελίδα. Η σελίδα φαίνεται στην παρακάτω Εικόνα 5.



Εικόνα 5

[14] https://nodejs.dev/learn/the-v8-javascript-engine

4.2 HTML

Παρακάτω φαίνεται ο κώδικας του αρχείου “index.html” που περιέχει τη δομή της ιστοσελίδας μας (Εικόνα 6).

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <link rel="stylesheet" href="style.css">
    <script src="/socket.io/socket.io.js"></script>
    <script src="web-voice-processor/dist/iife/index.js"></script>
    <script src='client.js'></script>
    <title>Web Voice Tanscription</title>
  </head>

  <body>
    <div id="root">
      <div class='record-buttons'>
        <button id="start-recording" type="button">Start Recording</button>
        <button id="stop-recording" type="button">Stop Recording</button>
        </div>
        <div id="message-container" class="msg-con"></div>
        <form id="send-container">
          <input type="text" id="message-input" autoComplete="off"></input>
          <button type="submit" id="send-button" type="button">Send</button>
        </form>
        <br></br>
    </div>
  </body>
</html>
```

Εικόνα 6

4.3 Διακομιστής

Ας περάσουμε στη δημιουργία του server.js . Αρχικά, καλούμε όλα τα απαραίτητα πακέτα, και ξεκινάμε την εφαρμογή express δίνοντας στον πελάτη πρόσβαση στους φακέλους “public” και “node_modules/@picovoice/” ο οποίος θα βοηθήσει στην επεξεργασία του ήχου του μικρόφωνου του. Έπειτα δημιουργούμε τον HTTP server μας και αρχικοποιούμε τη μεταβλητή `socket` που θα χειρίζεται τη σύνδεση με τον πελάτη. Στη συνέχεια ορίζουμε μερικές σταθερές μεταβλητές (θα τις συναντήσουμε αργότερα), δημιουργούμε το μοντέλο DeepSpeech που θα

κάνει την αναγνώριση ομιλίας (γραμμή 34) και δηλώνουμε το γλωσσικό μοντέλο που θα χρησιμοποιήσουμε (Εικόνα 7).

```
1  const http = require('http');
2  const socketIO = require('socket.io');
3  const express = require('express');
4  const DEEPSPEECH = require('deepspeech');
5  const nodeVAD = require('node-vad');
6  const sanitizeHtml = require('sanitize-html');
7
8
9  const ex = express();
10 ex.get('/', (req, res) =>{
11   res.sendFile(__dirname + '/index.html');
12 })
13
14 ex.use(express.static(__dirname + '/public'));
15 ex.use(express.static(__dirname + '/node_modules/@picovoice/'));
16
17 const app = http.createServer(ex);
18 const io = socketIO(app, {});
19
20 const SERVER_PORT = 8888;
21 const SERVER_IP = "localhost";
22
23 let AM_PATH = "m1024-r.pbmm"; // Path to DeepSpeech Acoustic Model file
24 let LM_PATH = "m1024-r.scorer"; // Path to DeepSpeech Language Model file
25
26 let SILENCE_THRESHOLD = 400;//How much time of silence
27                             //will have to pass to transcribe chunks
28
29 let recording = false;
30 let silenceStartTime = null;
31
32 const vad = new nodeVAD(nodeVAD.Mode.AGGRESSIVE);
33
34 let dsModel = new deepspeech.Model(AM_PATH);
35 dsModel.enableExternalScorer(LM_PATH);
```

Εικόνα 7

```
90  io.on('connection', function(socket){
91
92    console.log('Client ' + socket.handshake.address + ' connected' );
93
94    // Start the python script to process incoming commands
95    const spawn = require('child_process').spawn;
96
97    const scriptArguments = `
98      --ip ` + socket.handshake.address + `
99      --states_db
100     --settings C:/Users/[redacted]/Desktop/Ptuxiaki/other_data/settings.ini
101     --dictionary_file C:/Users/[redacted]/Desktop/Ptuxiaki/other_data/dictionary.txt
102     -a C:/Users/[redacted]/Desktop/Ptuxiaki/other_data/aliases.txt
103   `;
104
105   const scriptExecution = spawn("python.exe", ["C:/Users/[redacted]/Desktop/Ptuxiaki/main_v6.py", scriptArguments]);
106
107   scriptExecution.stdout.on('data', (data) =>{
108     let encodedString = Buffer.from(data, 'utf-8');
109     let strData = encodedString.toString();
110     socket.emit('message', strData);
111   });
112
113   scriptExecution.on("error", (err) =>{
114     console.log("Script failed to start");
115   });
116
117   scriptExecution.stderr.on('data', (data) => {
118     console.error(`Script stderr error: ${data}`);
119   });
120
121   socket.once('disconnect', () => {
122     console.log('Client ' + socket.handshake.address + ' disconnected');
123   });
```

Εικόνα 8

Στην Εικόνα 8 αρχικά περιμένουμε να γίνει μια σύνδεση και εκτυπώνεται ένα μήνυμα όταν γίνει αυτή. Έπειτα καλείται η εφαρμογή του Κεφαλαίου 5 που θα επεξεργαστεί τις εντολές του χρήστη, δίνοντας σαν παραμέτρους τη διεύθυνση IP του χρήστη, την εντολή ότι θέλουμε να φορτώσει τις καταστάσεις από τη βάση δεδομένων και τα μονοπάτια για δύο αρχεία που θα περιγραφούν στο Κεφάλαιο 5. Στη γραμμή 107 περιμένουμε απάντηση από το πρόγραμμα και όταν απαντάει στέλνεται το μήνυμα αυτό στον χρήστη (110). Επιπλέον, διαχειριζόμαστε τυχών λάθη που μπορεί να προκύψουν από το πρόγραμμα εκτυπώνοντας μηνύματα στην κονσόλα και τέλος εκτυπώνεται ένα μήνυμα στη κονσόλα όταν αποσυνδέεται ο πελάτης.

```
125     stream = dsModel.createStream();
126
127     socket.on('audio_stream', function(data){
128       // User is streaming audio, listen for voice or silence and feed to model
129       processChunks(data, (results) => {
130         scriptExecution.stdin.write(data.concat('\r\n'));
131         socket.emit('recognition_result', data);
132       });
133     });
134
135     socket.on('audio_stream_reset', function() {
136       if(stream){
137         let results = stream.finishStream();
138         stream = null;
139         if(results){
140           scriptExecution.stdin.write(data.concat('\r\n'));
141           socket.emit('recognition_result', data);
142         }
143       }
144       stream = dsModel.createStream();
145     });
```

Εικόνα 9

Στην Εικόνα 9, περιμένουμε τον χρήστη να πατήσει το κουμπί “Start Recording” και να ξεκινήσει το audio stream (`audio_stream`), το οποίο υποδεχόμαστε, επεξεργαζόμαστε (Εικόνα 12) και στέλνεται το αποτέλεσμα στο πρόγραμμα Python για επεξεργασία και ταυτόχρονα στον χρήστη (135). Όταν κλείσει το μικρόφωνο (139) μεταφράζονται ό,τι δεδομένα έχουν μαζευτεί μέχρι στιγμής στο μοντέλο και αν ήταν επιτυχής η μετάφραση στέλνεται το αποτέλεσμα πάλι στο πρόγραμμα Python και τον χρήστη (140-141). Την συνάρτηση που επεξεργάζεται το audio stream θα τη δούμε λίγο αργότερα.

Στη συνέχεια (Εικόνα 10) περιμένουμε γραπτές εντολές του χρήστη, γίνεται αντικατάσταση των ειδικών χαρακτήρων για να αποφευχθούν οι κακόβουλες εντολές, στέλνεται η εντολή στην εφαρμογή του Κεφαλαίου 5 και εκτυπώνεται μήνυμα στην κονσόλα με το μήνυμα του χρήστη.

```
147     socket.on('client-command', message =>{
148       // Handle the chat command
149       message = message.replace(/[\!&@\^\/\\#,+()$~%.'":*;?<>{}]/g, "");
150       command = message.concat('\r\n');
151       scriptExecution.stdin.write(command);
152       console.log("Client " + socket.handshake.address + " typed: " + message);
153     });
154   });
```

Εικόνα 10

Τέλος, κλείνει η εντολή της γραμμής 90, και ξεκινάει η εφαρμογή να περιμένει εισερχόμενες συνδέσεις (Εικόνα 11).

```
154   });
155
156   app.listen(SERVER_PORT, SERVER_IP,() => {
157     console.log('Listening on:', SERVER_PORT);
158   });
```

Εικόνα 11

Μέχρι τώρα, δεν έχουμε αναφέρει τη συνάρτηση που επεξεργάζεται το audio stream όταν ο χρήστης ανοίγει το μικρόφωνο. Η συνάρτηση `processChunks` (Εικόνα 12) δέχεται μια παράμετρο `chunk` που περιέχει ένα Buffer με τα δεδομένα του audio stream, και μια συνάρτηση `callback` η οποία χρησιμοποιείται για να επιστραφεί το κείμενο που μεταφράζεται και έπειτα να συνεχιστεί η συνάρτηση.

Αν παρατηρήσουμε στη γραμμή 129 (Εικόνα 9) καλείται η συνάρτηση `processChunks` με παράμετρο `data` και μια ακόμη παράμετρο η οποία είναι μια ανώνυμη callback[15] συνάρτηση που θα εκτελεστεί σε αργότερο στάδιο και αυτή με τη σειρά της δέχεται την παράμετρο `result`. Το βελάκι "=>"[16] επιτρέπει τη δημιουργία συναρτήσεων με λιγότερο συντακτικό. Ο επόμενος κώδικας στις γραμμές 130,131 και 132 ανήκουν σε αυτό το callback.

Επιστρέφοντας στην `processChunks`, το αντικείμενο `vad` ανιχνεύει πότε μιλάει ο χρήστης και πότε όχι δέχοντας σαν παράμετρο το `chunk` και το Sample Rate που στην περίπτωσή μας είναι 16000hz (16kHz). Έτσι, όταν ανιχνευθεί σφάλμα ή θόρυβος στον ήχο δεν κάνουμε τίποτα, ενώ όταν ανιχνευθεί ησυχία αρχικά το κομμάτι ήχου "ταΐζεται" στο μοντέλο DeepSpeech και αν υπάρχει ησυχία για περισσότερο από `SILENCE_THRESHOLD` μιλιδευτερόλεπτα (millisecond, ms) τότε σταματάει η τροφοδότηση του μοντέλου με δεδομένα και μεταφράζονται όσα δεδομένα έχουν τροφοδοτηθεί σε αυτό μέχρι αυτή τη στιγμή. Αν η μετάφραση είναι επιτυχής, τότε καλείται η συνάρτηση `callback` και

[15] https://www.w3schools.com/js/js_callback.asp
[16] https://www.w3schools.com/js/js_arrow_function.asp

επιστρέφεται το μεταφρασμένο κείμενο στην ανώνυμη συνάρτηση που δηλώθηκε ως δεύτερη παράμετρος για τη συνάρτηση `processChunks`. Συγκεκριμένα, καλείται στη γραμμή 129 (Εικόνα 9) οπότε θα κληθεί αυτή η ανώνυμη συνάρτηση με παράμετρο `text` και θα εκτελέσει τις εντολές στις γραμμές 130 και 131 και ύστερα θα συνεχίζει η `processChunks`.

Αν ανιχνευτεί φωνή (Εικόνα 13), τότε επαναφέρεται η αρχή του χρόνου ησυχίας, αν χρειαστεί ξεκινάει το stream και τέλος τροφοδοτείται το κομμάτι ήχου στο μοντέλο.

```
37  function processChunks(chunk, callback){
38    vad.processAudio(chunk, 16000).then(res => {
39          switch (res) {
40              case nodeVAD.Event.ERROR:
41                  break;
42              case nodeVAD.Event.NOISE:
43                  break;
44              case nodeVAD.Event.SILENCE:
45                  if(recording){
46                    if(stream){
47                      // Feed the chunk of audio to the stream
48                        stream.feedAudioContent(chunk);
49                    }
50                    // Based on how much time has passed since the silence began,
51                    // determine if its time to stop feeding chunks
52                    // If it is translate the chunks and reset,
53                    // else continue feeding them
54                    if (silenceStartTime === null){
55                      silenceStartTime = new Date().getTime();
56                    }
57                    else{
58                      let currentTime = new Date().getTime();
59                      if (currentTime - silenceStartTime > SILENCE_THRESHOLD){
60                        silenceStartTime = null;
61                        let text = null;
62                        if(stream){
63                          text = stream.finishStream();
64                        }
65                        if(text){
66                          // Transcription is complete return the text
67                          stream = dsModel.createStream();
68                          recording = false;
69                          callback(text);
70                        }
71                        stream = null
72                      }
73                    }
74                  }
75                  break;
```

Εικόνα 12

```
76                case nodeVAD.Event.VOICE:
77                    // Voice is detected
78                    silenceStartTime = null
79                    recording = true
80                    if(!stream){
81                      stream = dsModel.createStream();
82                    }
83                    // Feed the chunk of audio to the stream
84                    stream.feedAudioContent(chunk);
85                    break;
86            }
87        }).catch(console.error);
88    }
```

Εικόνα 13

4.4 Πελάτης

Εντός του φακέλου “Public” βρίσκεται ο κώδικας για τον πελάτη (client.js). Χρησιμοποιεί τα script που δηλώνονται στο index.html (Εικόνα 6) συγκεκριμένα τις γραμμές 6 και 7 ενώ η 8 καλεί το ίδιο στη σελίδα. Η σελίδα περιέχει στο σύνολο τρία κουμπιά, τα Start Recording, Stop Recording και Send. Όταν πατιέται το Start Recording, απενεργοποιείται αυτό και ενεργοποιείται το Stop Recording και το αντίθετο. Το κάθε κουμπί καλεί αντίστοιχα τις συναρτήσεις `startRecording` και `stopRecording` για ξεκινήσει και να σταματήσει το audio stream (Εικόνα 14)

```
29  window.addEventListener('load', (event) =>{
30    const startButton = document.getElementById("start-recording");
31    const stopButton = document.getElementById("stop-recording");
32    const sendButton = document.getElementById("send-button");
33
34    startButton.disabled = false;
35    stopButton.disabled = true;
36
37    startButton.addEventListener("click", function(event){
38      stopButton.disabled = false;
39      startButton.disabled = true;
40      startRecording();
41    });
42
43    stopButton.addEventListener("click", function(event){
44      stopButton.disabled = true;
45      startButton.disabled = false;
46      stopRecording();
47    });
48
```

Εικόνα 14

Όταν ο χρήστης στέλνει ένα μήνυμα και πατάει το κουμπί Send (Εικόνα 15), τότε το μήνυμα που έχει γράψει στο text box πάνω από το κουμπί εμφανίζεται στη συνομιλία και ταυτόχρονα στέλνεται στον διακομιστή (server.js) για να σταλθεί από αυτόν στο πρόγραμμα Python για να το επεξεργαστεί.

```
49  sendButton.addEventListener("click", function(e){
50    const messageInput = document.getElementById('message-input');
51    e.preventDefault();
52    const message = messageInput.value;
53    if(message === ""){
54      return;
55    }
56    const msgBox = document.getElementById('message-container');
57    let newHtml = "</div><div class='client-message'><span class='client-message-style'>" + message + "</span></div><div class='separator'>";
58    msgBox.innerHTML += newHtml;
59    msgBox.scrollTop = msgBox.scrollHeight; // Auto Scroll to the bottom of the chat
60    socket.emit('client-command', message);
61    messageInput.value="";
62    });
```

Εικόνα 15

Όταν δέχεται μήνυμα `recognition_result` στον πελάτη (Εικόνα 16), δηλαδή όταν αναγνωρίστηκε η ομιλία του, εμφανίζεται το κείμενο αυτό το στη συνομιλία.

```
64  socket.on('recognition_result', (results) => {
65    const msgBox = document.getElementById('message-container');
66    let newHtml = "<div class='client-message'><span class='client-message-style'>" + results + "</span></div><div class='separator'></div>";
67    msgBox.innerHTML += newHtml;
68    msgBox.scrollTop = msgBox.scrollHeight; // Auto Scroll to the bottom of the chat
69  });
```

Εικόνα 16

Τέλος, όταν στέλνει μήνυμα ο διακομιστής μέσω του socket (Εικόνα 17) εμφανίζεται αυτό στη συνομιλία και αν είναι απενεργοποιημένο το κουμπί Start Recording (δηλαδή ο χρήστης χρησιμοποιεί τη φωνή του) τότε διαβάζεται αυτό το μήνυμα από μια συνθετική φωνή στα ελληνικά (75-85).

```
71    socket.on('message', (msg) =>{
72      const msgBox = document.getElementById('message-container');
73      if (startButton.disabled){ // If the start button is disabled enable Text To Speech
74        var voices = window.speechSynthesis.getVoices();
75        if(typeof speechSynthesis === 'undefined') {
76          console.log("Error when using tts");
77          return;
78        }
79        let tts = new SpeechSynthesisUtterance(msg);
80        tts.lang = "el-GR";
81        window.speechSynthesis.speak(tts);
82      };
83      msgBox.innerHTML += "</div><div class='server-message'><span>" + msg + "</span></div><div class='separator'></div>";
84      msgBox.scrollTop = msgBox.scrollHeight; // Auto Scroll to the bottom of the chat
85    });
86
87  });
```

Εικόνα 17

Όταν αρχίζει το stream (Εικόνα 18), δημιουργείται ένα αντικείμενο Web Worker (worker.js) και ξεκινά η ηχογράφηση του μικροφώνου με τη βοήθεια του πακέτου Web Voice Processor[17] το οποίο μετατρέπει το Sample Rate σε 16kHz που απαιτείται από το

[17] https://www.npmjs.com/package/@picovoice/web-voice-processor

μοντέλο DeepSpeech. Ο ήχος αυτός περνά από τον πολύ απλό Web Worker μας, που απλά επιστρέφει τον ήχο και κατευθείαν στέλνεται στον διακομιστή (server.js) για την αναγνώριση ομιλίας.

```
const socket = io();

let handle;
let worker;

async function startRecording(){

  worker = new Worker("worker.js");

  handle = await WebVoiceProcessor.WebVoiceProcessor.init({
    engines: [worker],
    start : true,
  });
  worker.onmessage = (e) => {
    if(socket){
      socket.emit("audio_stream", e.data.inputFrame.buffer);
    }
  };
}
```

Εικόνα 18

Όταν πατιέται το κουμπί Stop Recording (Εικόνα 19), τερματίζεται ο Web Worker, σταματά η ηχογράφηση και στέλνει μήνυμα στο διακομιστή ότι σταμάτησε η ηχογράφηση.

```

function stopRecording(){
  worker.terminate();
  handle.release();
  if(socket){
    socket.emit("audio_stream_reset");
  }
```

Εικόνα 19

ΚΕΦΑΛΑΙΟ 5 Επεξεργασία Εντολών

Προτού περάσουμε στην ανάπτυξη της εφαρμογής μας, οφείλουμε να κάνουμε μια σύντομη περιγραφή της γλώσσας προγραμματισμού Python στο υποκεφάλαιο 5.1, φυσικά μια αναλυτική περιγραφή της γλώσσας είναι μη εφικτή στο πλαίσιο αυτής της πτυχιακής εργασίας, για αυτό θα παρουσιαστούν βασικές έννοιες που θα συναντήσουμε αργότερα. Έπειτα θα παρουσιαστεί η βάση δεδομένων της εφαρμογής.

5.1 Εισαγωγή στην Python

Η Python[18] είναι μια διερμηνευόμενη γλώσσα προγραμματισμού υψηλού επιπέδου και γενικού σκοπού, είναι αντικειμενοστραφής και ταυτόχρονα υποστηρίζει επιπλέον προσεγγίσεις «όπως ο διαδικαστικός και συναρτησιακός προγραμματισμός» [22] . Πέρα των πολλών δυνατοτήτων που προσφέρει, είναι απλή τόσο στη χρήση όσο και στην εκμάθηση και παρέχει από μόνη της με ένα μεγάλο πλήθος βιβλιοθηκών που παρέχουν έτοιμες λειτουργίες, αλλά και τη δυνατότητα επέκτασης αυτών.

Τα module[19] είναι αρχεία Python (με την κατάληξη .py) που περιέχουν ορισμούς και δηλώσεις (πχ μεταβλητών και συναρτήσεων), αλλά μπορούν να περιέχουν και εκτελέσιμες δηλώσεις και συναρτήσεις. Τα modules μπορούν να μειώσουν το μέγεθος ενός προγράμματος χωρίζοντάς το σε μικρότερα μέρη για να είναι πιο ευανάγνωστο και διαχειρίσιμο. Συνεπώς κάθε πρόγραμμα μπορεί να χρησιμοποιεί τα modules, καλώντας τα με την εντολή `import [όνομα]` (χωρίς την κατάληξη), αλλά επίσης τα modules μπορούν να εκτελεστούν και από μόνα τους. Μια συλλογή από modules μπορούν να δημιουργήσουν πακέτα (packages)[20].

Ο όρος συνάρτηση αναφέρεται σε ένα κομμάτι κώδικα που εκτελείται μόνο όταν καλείται η συνάρτηση. Σε αυτές, μπορούν να περαστούν δεδομένα (παράμετροι) και να επιστρέψουν (στον κώδικα που την κάλεσε) δικά της δεδομένα. Δηλώνονται με τη λέξη `def`, μια απλή συνάρτηση φαίνεται στην Εικόνα 20 που εκτυπώνει στην οθόνη τη πρόταση “Hello World”, και επιστρέφει με τη λέξη `return` την πρόταση “Returning data”.

```
2    def hello():
3        print("Hello World")
4
5        return "Returning Data"
6
```

Εικόνα 20

[18] https://docs.python.org/3/faq/general.html#what-is-python
[19] https://docs.python.org/3/tutorial/modules.html
[20] https://realpython.com/python-modules-packages/#python-packages

Οι δύο αυτές προτάσεις της Εικόνα 20 ανήκουν σε ένα τύπο δεδομένων που αποκαλούνται Strings. Η Python υποστηρίζει πολλούς τύπους δεδομένων, μεταξύ άλλων, τα int (Integer) και float (Floating point).

Ένας ακόμη τύπος δεδομένων είναι οι λίστες (lists) που μπορούν να αποθηκεύσουν πολλαπλές τιμές ακόμα και διαφορετικού τύπου δεδομένων. Μπορεί επίσης να περιέχει άλλες λίστες σαν τιμή. Για να προσπελάσουμε μια τιμή της λίστας, χρησιμοποιούμε το όνομά της και τις αγκύλες [ ] εισάγοντας σε αυτές τον δείκτη μιας τιμής της λίστας που μας ενδιαφέρει. Μια λίστα (name) φαίνεται στην Εικόνα 21 που περιέχει το ονοματεπώνυμο ενός ατόμου ως μια εσωτερική λίστα με strings, καθώς και την ηλικία (int) και το ύψος (float) του εκτός αυτής. Επίσης, τα εκτυπώνει αυτά στην οθόνη με τρις διαφορετικούς τρόπους με τη συνάρτηση `print()`.

```
name = [["John", "Smith"], 25, 1.80]

print("My name is", name[0][0], name[0][1], ",my age is", name[1], "and i am", name[2], "m tall")

print("My name is " + name[0][0] + " " + name[0][1] + ", my age is " + str(name[1])
      + " and i am " + str(name[2]) + "m tall")

print(f"My name is {name[0][0]} {name[0][1]}, my age is {name[1]} and i am {name[2]}m tall")
```

```
My name is John Smith ,my age is 25 and i am 1.8 m tall
My name is John Smith, my age is 25 and i am 1.8m tall
My name is John Smith, my age is 25 and i am 1.8m tall

Process finished with exit code 0
```

Εικόνα 21

Σημείωση, η συνάρτηση `str()` (μετατρέπει ένα αντικείμενο σε string) στη δεύτερη `print()` είναι απαραίτητη, καθώς σε αυτή τη περίπτωση προσθέτει όλα τα strings σε ένα, και δεν επιτρέπεται να προσθέσει int και float τιμές με string.

Ακόμη ένας τύπος δεδομένων που θα συναντήσουμε στην εφαρμογή μας είναι τα λεξικά. Παρόμοια με τις λίστες, αποθηκεύουν μια συλλογή δεδομένων αλλά με τη μορφή κλειδιού-τιμής. Οι τιμές των κλειδιών μπορεί να είναι οποιοσδήποτε τύπος δεδομένων, και για να αποκτήσουμε πρόσβαση σε αυτή αναφερόμαστε στο αντίστοιχο κλειδί εντός αγκύλων [ ]. Δεν μπορούν να υπάρξουν δύο κλειδιά με το ίδιο όνομα εντός του λεξικού. Θα μετατρέψουμε τη λίστα από την Εικόνα 21 σε ένα λεξικό που φαίνεται στην Εικόνα 22.

```
dict = {
    "name": ["John", "Smith"],
    "age": 25,
    "height": 1.80
}
print(f"My name is {dict['name'][0]} {dict['name'][1]}, my age is {dict['age']} and i am {dict['height']}m tall")
```

Εικόνα 22

Έπειτα, έχουμε τις πλειάδες (tuples) που μπορεί να μοιάζουν αρκετά στις λίστες, καθώς η μόνη οπτική αλλαγή είναι οι παρενθέσεις ( ) έναντι των αγκύλων [ ]. Ωστόσο, οι τιμές που περιέχει το tuple δεν μπορούν να αλλάξουν από τη στιγμή που εισαχθούν σε αυτό (ούτε να προστεθούν ή να αφαιρεθούν τιμές σε/από αυτό) και σε αντίθεση με τα λεξικά,

επιτρέπονται διπλές τιμές. Η πλειάδα για το παράδειγμα των Εικόνων 18 και 19 γίνεται:
`name = (“John”,”Smith”], 25, 18)`

Τέλος, έχουμε τα set, τα οποία έχουν την ιδιαιτερότητα ότι οι τιμές τους δεν ακολουθούν κάποια σειρά όπως τα τρία προηγούμενα (οι τιμές εντός των λεξικών ακολουθούν μια σειρά από την Python 3.7 και έπειτα, ενώ των λιστών και πλειάδων είναι σε σειρά) με αποτέλεσμα να εμφανίζεται ένα διαφορετικό αποτέλεσμα κάθε φορά που χρησιμοποιείται. Μπορεί να περιέχει σχεδόν όλους τους τύπους δεδομένων για τιμές, αλλά δεν μπορούμε να τις προσπελάσουμε και δεν επιτρέπονται διπλές τιμές. Επίσης, παρομοίως με τις πλειάδες οι τιμές δεν αλλάζουν αλλά γίνεται να προστεθούν νέες. Το set του παραδείγματός μας γίνεται:

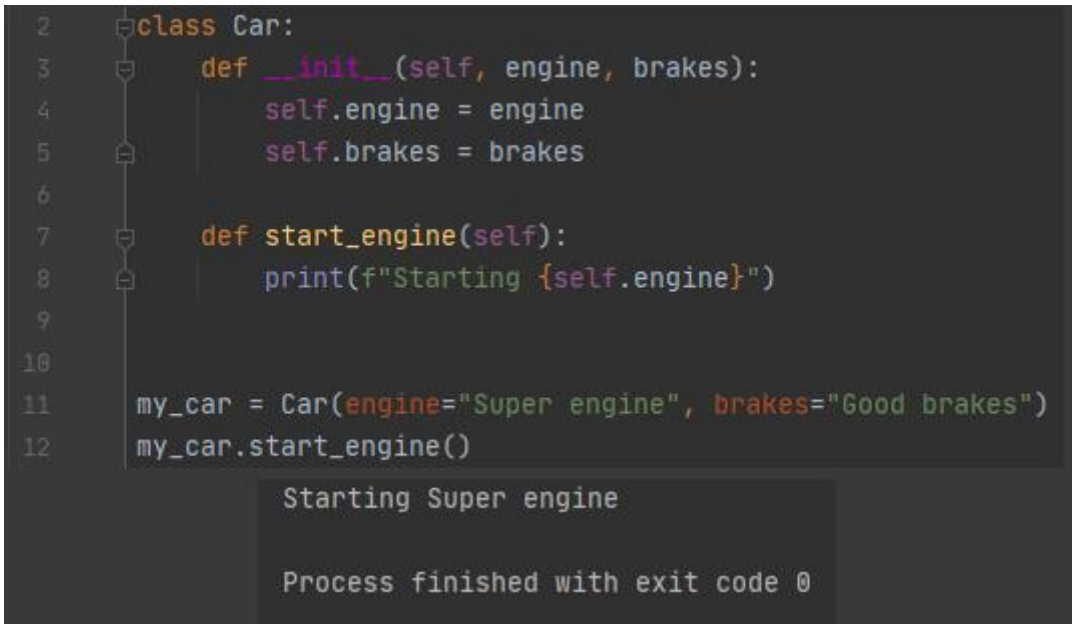
```
name = {(“John”, “Smith”), 25, 18}
```

Παρατηρούμε ότι το όνομα βρίσκεται εντός μιας πλειάδας αντί για λίστας, αυτό οφείλεται στο γεγονός ότι τα sets δεν αναγνωρίζουν λίστες καθώς οι τιμές τους μεταβάλλονται. Αν χρησιμοποιούσαμε λίστα θα αντιμετωπίζαμε το σφάλμα που φαίνεται στην Εικόνα 23 , παρομοίως και για τα λεξικά.

```
    name = {["John", "Smith"], 25, 1.80}
TypeError: unhashable type: 'list'
```

Εικόνα 23

Ένα αντικείμενο είναι μια συλλογή δεδομένων και συναρτήσεων που ενεργούν πάνω σε αυτά τα δεδομένα. Τα δεδομένα ονομάζονται attributes και μια συνάρτηση ενός αντικειμένου ονομάζεται μέθοδος η οποία περιγράφει την συμπεριφορά αυτού και επεξεργάζεται τα attributes του. Κάθε μέθοδος δέχεται σαν παράμετρο το “self”, δηλώνεται εντός της κλάσης και μπορούμε να τη καλέσουμε μέσω του αντικειμένου χρησιμοποιώντας τον τελεστή **τελεία**. Η κλάση, είναι ένας τρόπος να κατασκευάσουμε πολλαπλά αντικείμενα του ίδιου τύπου. Ένα παράδειγμα μιας κλάσης “Car” φαίνεται στην Εικόνα 24, το οποίο έχει τα attributes “engine” και “brakes” καθώς και τη μέθοδο “start_engine” που ξεκινάει τη μηχανή εκτυπώνοντας ένα μήνυμα. Επίσης εκτός της κλάσης δημιουργείται ένα αντικείμενο Car και καλείται η συνάρτηση `start_engine()`



Εικόνα 24

Η `__innit__` είναι μια ειδική μέθοδος για τις κλάσεις στη Python. Λειτουργεί σαν κατασκευαστής (Constructor) και καλείται όταν δημιουργείται το αντικείμενο (γραμμή 11 στην Εικόνα 24) για να αρχικοποιήσει το αντικείμενο. Η λέξη “self” αναφέρεται στο ίδιο το αντικείμενο και είναι υποχρεωτική παράμετρος για όλες τις μεθόδους του. Στη γραμμή 8 γράφοντας `self.engine` αναφερόμαστε στο attribute “engine” του αντικειμένου μας.

5.2 Βάση Δεδομένων

Η σχεδίαση της εφαρμογής ξεκινά από τη σχεδίαση μιας βάση δεδομένων (Database, DB).

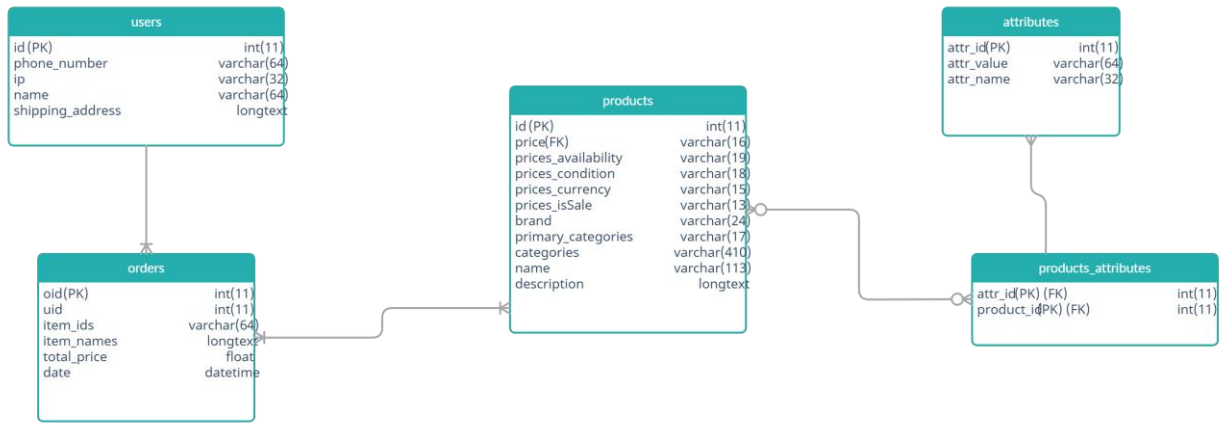
Ως βάση δεδομένων χαρακτηρίζεται μια συλλογή οργανωμένης πληροφορίας αποθηκευμένη σε ηλεκτρονική μορφή. Συχνές ενέργειες πάνω σε αυτές είναι η αναζήτηση, εισαγωγή, τροποποίηση και διαγραφή δεδομένων. Για την διαχείριση τους υπάρχουν τα Συστήματα Διαχείρισης Βάσεων Δεδομένων (Database Management System, DBMS). Τα δύο πιο διαδεδομένα είδη μοντέλων βάσεων δεδομένων είναι τα σχεσιακά και μη-σχεσιακά μοντέλα. Η γλώσσα προγραμματισμού SQL χρησιμοποιείται συχνά από τα σχεσιακά DBMS (relational DBMS, RDBMS) για τη διαχείριση των δεδομένων.

Τα σχεσιακά μοντέλα αποθηκεύουν τα δεδομένα σε γραμμές (εγγραφές, rows, tuples) και στήλες (columns, attributes) οι οποίες περιέχουν τιμές που σχετίζονται με τη συγκεκριμένη γραμμή, ταυτόχρονα, ένα σύνολο από γραμμές και τις αντίστοιχες στήλες τους οργανώνονται σε πίνακες (tables). Η κάθε στήλη μπορεί να αποθηκεύει διαφορετικό τύπο πληροφορίας (πχ ακέραια τιμή, ή κείμενο) και ακόμα να μην πάρει καμία τιμή (null). Όλες οι γραμμές ενός πίνακα έχουν τις ίδιες στήλες.

Τα μη σχεσιακά μοντέλα από την άλλη, δεν χρησιμοποιούν την ίδια δομή αποθήκευσης πληροφοριών με τα σχεσιακά μοντέλα, αλλά μπορούν να προσαρμοστούν ως κάποιο βαθμό για τα δεδομένα που σκοπεύουν να αποθηκεύσουν.

Θα χρησιμοποιήσουμε το MariaDB[21], ένα σύστημα RDBMS, για τη βάση δεδομένων μας, η εγκατάσταση του οποίου θα γίνει μέσω του xampp[22], ένα ανοιχτού κώδικα περιβάλλον ανάπτυξης PHP του οποίου η εγκατάσταση είναι αρκετά εύκολη και περιέχει το MariaDB μαζί με άλλα "εξαρτήματα". Από αυτά, θα χρειαστούμε τα "MySQL" και "phpMyAdmin"[23]. Το πρώτο είναι το RDBMS και το δεύτερο είναι ένα εργαλείο με το οποίο θα μπορούμε να διαχειριστούμε τη βάση δεδομένων μέσω ενός φυλλομετρητή (web browser).

Η δομή της βάσης δεδομένων μας φαίνεται στην Εικόνα 25. Αποτελείται από έξι πίνακες (tables).



Εικόνα 25

Το table "users" περιέχει τις πληροφορίες των χρηστών, πιο συγκεκριμένα το μοναδικό αναγνωριστικό "id" που δίνεται αυτόματα σε κάθε νέο χρήστη, το κινητό του (αν υπάρχει) "phone_number", τη διεύθυνση "ip" αν υπάρχει και μια διεύθυνση στην οποία αποστέλνονται οι παραγγελίες "shipping_address".

Το "products" περιέχει τις πληροφορίες των προϊόντων που προσφέρει το κατάστημα. Κάθε προϊόν έχει μια τιμή "price", το νόμισμα αυτής της τιμής "prices_currency" (πχ EUR-ευρώ) τον κατασκευαστή του "manufacturer", τη μάρκα του προϊόντος "brand", την (τις, χωρισμένες με κόμμα) βασική κατηγορία στην οποία ανήκει "primary_categories", τις υποκατηγορίες (χωρισμένες με κόμμα) "categories", το όνομα του "name" και τέλος μια περιγραφή για αυτό "description".

Επίσης, κάθε προϊόν μπορεί να έχει συγκεκριμένα χαρακτηριστικά (πχ ένα ρούχο μπορεί να έχει χρώμα, μέγεθος και ύφασμα). Αυτά δηλώνονται από το table "products_attributes", η δεύτερη στήλη του, "product_id" αναφέρεται στο "id" του προϊόντος από το πίνακα "products" που έχει το χαρακτηριστικό που δηλώνεται από τη

[21] https://mariadb.org/
[22] https://www.apachefriends.org/index.html
[23] https://www.phpmyadmin.net/

στήλη “attr_id” η οποία αναφέρεται στο πίνακα “attributes” και πιο συγκεκριμένα στη στήλη “attr_id” αυτού του πίνακα.

Το “attributes” αποτελείται, εκτός του “attr_id”, και από τη στήλη “attr_name” που δηλώνει το όνομα ενός χαρακτηριστικού και τη στήλη “attr_value” που περιέχει μια συγκεκριμένη τιμή του “attr_name”.

Ένα παράδειγμα που θα ξεκαθαρίσει τη δομή των χαρακτηριστικών είναι το εξής. Ας υποθέσουμε ότι στον πίνακα “products” έχουμε τα δύο προϊόντα:

id	**primary_categories**	**categories**	**Name**	**description**
1	Ρούχα	Μπλούζες	Μαύρη μπλούζα Medium	Μια μαύρη μπλούζα μεγέθους Medium
2	Ρούχα	Μπλούζες	Κόκκινη μπλούζα Large	Μια κόκκινη μπλούζα Large

Τότε, για να προσδιορίσουμε ότι το προϊόν με id 1 έχει χρώμα Μαύρο και μέγεθος Medium (και αντίστοιχα για το προϊόν 2), πρέπει να εισάγουμε στο πίνακα “attributes” τα παρακάτω:

attr_id	**attr_value**	**attr_name**
10	Μαύρο	Χρώμα
11	Κόκκινο	Χρώμα
12	Medium	Μέγεθος
13	Large	Μέγεθος

Επομένως, τώρα μπορούμε να αναθέσουμε τις γραμμές με attr_id 10 και 12 στο προϊόν 1, και αντίστοιχα τις 11 και 13 στο προϊόν 2, ο πίνακας “product_attributes” διαμορφώνεται ως εξής:

attr_id	**product_id**
10	1
12	1
11	2
13	2

Ένα προϊόν μπορεί να έχει κανένα ή “άπειρα” χαρακτηριστικά, ενώ προφανώς δεν είναι δυνατόν να έχει το ίδιο χαρακτηριστικό πάνω από μια φορά.

Ο πίνακας που αποθηκεύει τα στοιχεία των παραγγελιών που κάνουν οι χρήστες είναι το “orders” και αποτελείται από το μοναδικό αναγνωριστικό “oid” το οποίο δίνεται αυτόματα σε κάθε νέα παραγγελία, το “uid” που αναφέρεται στο μοναδικό αναγνωριστικό

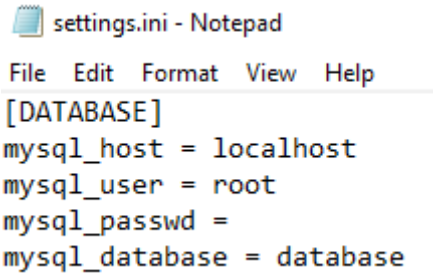
του χρήστη που πραγματοποίησε τη συγκεκριμένη παραγγελία, το “item_ids” στο οποίο αποθηκεύονται τα αναγνωριστικά των προϊόντων που αγοράστηκαν χωρισμένα με κόμμα, στο “quantities” είναι οι ποσότητες των προϊόντων χωρισμένα με **κόμμα**, στο “prices_at_date” οι τιμές αυτών των προϊόντων τη συγκεκριμένη ημερομηνία , το “oship_addr” αποθηκεύει τη διεύθυνση αποστολής της παραγγελίας, “total_price” η συνολική τιμή των προϊόντων και “date” η ημερομηνία (και ώρα) που πραγματοποιήθηκε η παραγγελία.

Για τη σύνδεση στη βάση δεδομένων μέσω της εφαρμογής μας, χρησιμοποιείται το πακέτο “pymysql”. Εντός της εφαρμογής, η συνάρτηση `connect_to_db` (στο αρχείο additional_functions.py) δέχεται σαν ορίσματα τα “host” (η διεύθυνση του διακομιστή που παρέχει τη βάση δεδομένων, από προεπιλογή είναι η localhost), “user”, “passwd”, “database” το όνομα της βάσης δεδομένων στην οποία θέλουμε να συνδεθούμε και “client_flag”. Η εκτέλεση εντολών στη βάση γίνεται με τη χρήση του αντικειμένου σύνδεσης που επιστρέφει η προαναφερθείσα συνάρτηση. Στην Εικόνα 26 φαίνεται ένα παράδειγμα όπου εκτελείται η εντολή `SELECT * FROM products WHERE `name` LIKE `%παράδειγμα%`` η οποία επιστρέφει όλα τα προϊόντα από τον πίνακα “products” τα οποία στο πεδίο “name” περιέχουν τη λέξη “παράδειγμα”

```
connection = connect_to_db(host="localhost", user="root", passwd="", database="database")
sql = "SELECT * FROM `states` WHERE `name` LIKE `%παράδειγμα%` "
with connection.cursor() as cursor:
    cursor.execute(sql, ())
    result = cursor.fetchall()
connection.commit()
connection.close()
```

Εικόνα 26

Οι μεταβλητές host, user, passwd, database δηλώνονται στο αρχείο ρυθμίσεων “other_data/settings.ini” (Εικόνα 27) και αποθηκεύονται στις μεταβλητές της Εικόνα 28

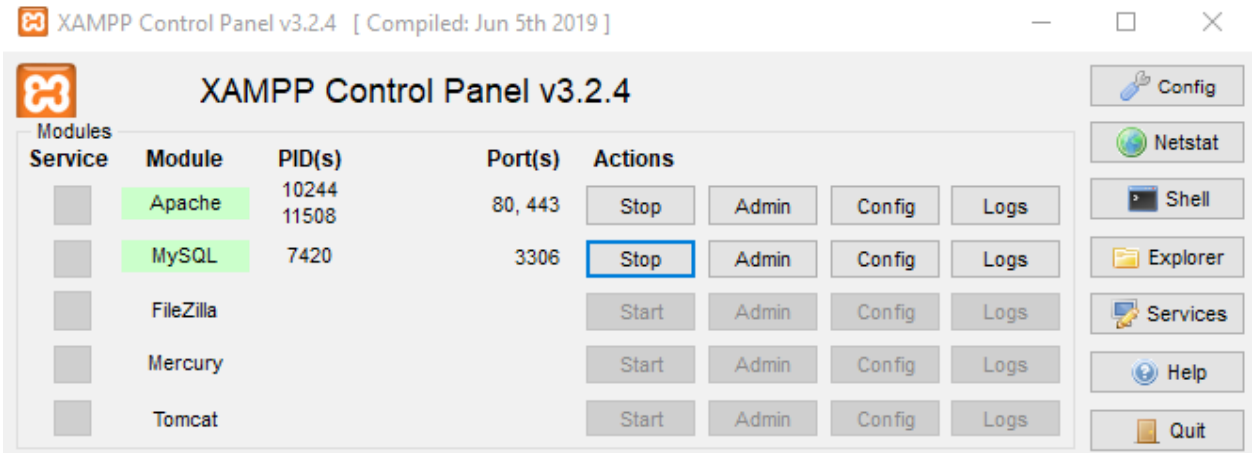


Εικόνα 27

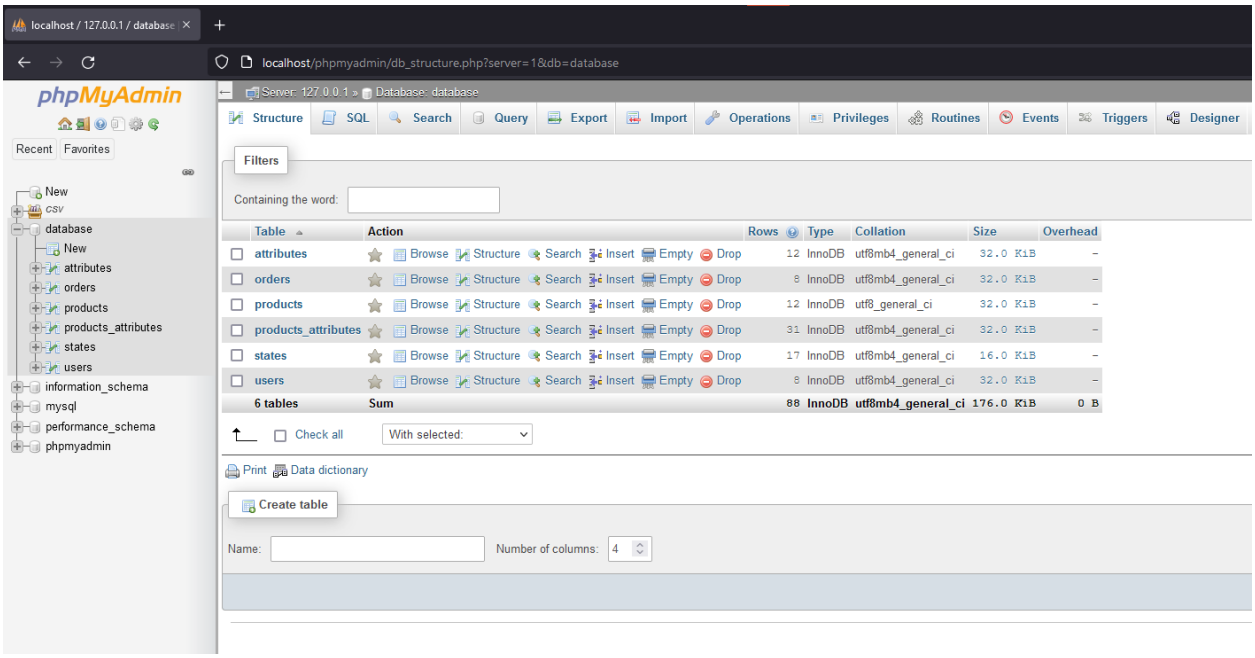
```
# MySQL connection parameters
self.MYSQL_HOST = config["DATABASE"]["mysql_host"]
self.MYSQL_USER = config["DATABASE"]["mysql_user"]
self.MYSQL_PASSWD = config["DATABASE"]["mysql_passwd"]
self.MYSQL_DATABASE = config["DATABASE"]["mysql_database"]
```

Εικόνα 28

Η πρόσβαση στη βάση δεδομένων μπορεί να γίνει και μέσω της σελίδας “http://localhost/phpmyadmin/” (Εικόνα 29Εικόνα 29) ενώ είναι ενεργοποιημένα ο διακομιστής Apache και ο διακομιστής MySQL εντός του xampp(Εικόνα 30).



Εικόνα 30



Εικόνα 29

5.3 Εφαρμογή Επεξεργασίας Εντολών

Έχοντας στη διάθεσή μας το μεταφρασμένο κείμενο μιας φωνητικής εντολής ή ενός απλού γραπτού μηνύματος, θα το επεξεργαστούμε δίνοντας στον χρήστη την δυνατότητα εκτέλεσης πέντε εντολών για την προσθήκη και αφαίρεση αντικειμένων στο και από το καλάθι του, για να αδειάσει όλα τα αντικείμενα από το καλάθι, να δει (ή ακούσει) τα περιεχόμενά του και να ολοκληρώσει την αγορά του. Ωστόσο , σε αυτό το σημείο να αναφέρουμε, ότι η εφαρμογή δεν υποστηρίζει την πληρωμή αυτών των αγορών μέσω κάρτας (πχ χρεωστική, πιστωτική κ.α.) ή με κάποια μορφή ηλεκτρονικής πληρωμής.

Τα απαιτούμενα για την λειτουργία της εφαρμογής είναι η γλώσσα προγραμματισμού Python (χρησιμοποιήθηκε η έκδοση 3.7.9) και η εφαρμογή xampp. Η κύρια κλάση CommandProcessor καθώς και το εκτελέσιμο πρόγραμμα βρίσκεται στο αρχείο main.py, στο additional_functions.py βρίσκονται πρόσθετες συναρτήσεις τις οποίες αξιοποιεί το βασικό αρχείο και το user.py περιέχει την κλάση User (υποκεφάλαιο 5.3.8)

5.3.1 Καταστάσεις

Η εφαρμογή στο πυρήνα της αποτελείται από καταστάσεις οι οποίες διαμορφώνουν τη ροή του προγράμματος. Η ιδέα είναι ότι ο διαχειριστής της εφαρμογής μπορεί να τροποποιήσει τις καταστάσεις για να εξατομικεύσει την εφαρμογή, όσον αυτό είναι επιτρεπτό και δυνατό. Για αυτό το λόγο, πολλές λειτουργίες είναι σχεδιασμένες έτσι ώστε να μπορούν να παραμετροποιηθούν. Ξεκινώντας από τη κατάσταση μηδέν (0) εκτελείται μια εντολή και γίνεται μετάβαση σε μια άλλη κατάσταση. Αυτό επαναλαμβάνεται επ’ άπειρον ή μέχρι να φτάσει το πρόγραμμα σε μια κατάσταση εξόδου.

Η ροή των καταστάσεων δίνεται από τον διαχειριστή, ο οποίος μπορεί να τις δημιουργήσει χειροκίνητα, ή να χρησιμοποιήσει ένα script που γράφτηκε για αυτό το σκοπό (edit_and_import.py). Το script καθοδηγεί το διαχειριστή και του δίνει τις επιλογές που φαίνονται στην Εικόνα 31, έπειτα γράφοντας “1” μπορεί να δημιουργήσει μια κατάσταση όπως φαίνεται στην Εικόνα 32. Κάθε κατάσταση καλεί μια μέθοδο, οι οποίες θα παρουσιαστούν αργότερα σε αυτό το Κεφάλαιο. Επίσης μπορούν να περιέχουν μηνύματα για τους χρήστες, αν είναι πάνω από ένα θα επιλεχθεί κάποιο τυχαία. Επιπλέον, μετάβαση σε μια κατάσταση μπορεί να γίνει όταν ο χρήστης πει ή γράψει κάποιες λέξεις κλειδιά. Τέλος στις καταστάσεις μπορούν να δοθούν παράμετροι (όπως όταν καλείται ένα πρόγραμμα από τη γραμμή εντολών) για περαιτέρω παραμετροποίηση, αν τις υποστηρίζει η καλούμενη μέθοδος.

```
1. Edit the states
2. Import a csv file with products to the database
3. Import a csv file with states to the database
4. Import a csv file with users to the database
0. Exit
> 1
1. Create a new state
2.Delete a state
3. Save states to .csv
4. Load states from .csv file
5. Save states to the database
6. Load states from the database
7. Print states
0. Exit
Type help to get this message
```

Εικόνα 31

```
1 > 1
Enter the details to create a new state in the following order, seperated by ;
 State number, State name, Function to be called by this state, Next state
1/1  > 0;Welcome State;message_once;1
Type the message(s) of this state, separated by ;
1/1  > Τώρα βρισκόμαστε στην κατάσταση 0 και θα πάμε στην 1;Αυτό είναι ένα δεύτερο μήνυμα
Type the keys to search for in the user's command seperated by ;
Note: ALL of the keys need to exist in order to enter this state.
Optional.
1/1  >
Type the arguments of this state as a String
1/1  >
```

Εικόνα 32

Οι καταστάσεις αποθηκεύονται σαν .csv αρχεία, και μπορούν να διαβαστούν από την εφαρμογή σαν αυτά, αλλά υπάρχει και η δυνατότητα να αποθηκευτούν στη βάση δεδομένων στο πίνακα “states”. Και οι δύο τρόποι αποθήκευσης παρέχονται από το script (επιλογές 3-6 ενώ έχει γίνει η επιλογή “Edit the states”), να σημειωθεί ότι αποθηκεύονται μόνο οι καταστάσεις που είναι διαθέσιμες σε αυτό (δηλαδή είτε έχουν δημιουργηθεί ή φορτωθεί από αρχείο ή τη βάση δεδομένων). Και με τους δύο τρόπους αποθήκευσης, έχουν τις ίδιες στήλες (βάση δεδομένων) και πεδία (csv αρχείο), πιο συγκεκριμένα τα:

- state: ο αριθμός της κατάστασης
- name: ένα όνομα που περιγράφει τη κατάσταση
- function_name: η μέθοδος την οποία καλεί η κατάσταση
- next_state: η επόμενη (ή επόμενες) κατάσταση που ακολουθεί την τωρινή
- messages: ένα ή περισσότερα μηνύματα για τον χρήστη

- keys: ένα ή περισσότερα κλειδιά με τα οποία μπορεί να γίνει είσοδος στην κατάσταση (μέσω της get_input)
- arguments: επιπλέον στοιχεία που μπορεί να χρησιμοποιούνται για παραμετροποίηση από τις μεθόδους

Σαν τελευταία λειτουργία, αυτό το script παρέχει τη δυνατότητα μεταφοράς .csv αρχείων στη βάση δεδομένων, εφόσον αυτά έχουν τη σωστή δομή.

Η περιγραφή των μεταβάσεων στις καταστάσεις στην αρχή του υποκεφαλαίου δεν ήταν πλήρης, επεκτείνοντας τη σε αυτό το σημείο προσθέτουμε ότι μια μετάβαση μπορεί να είναι αποτέλεσμα κάποιας μεθόδου, συγκεκριμένα των `get_input`, `more_than_one_result` ή `confirmation` αλλά και πρόωρης διακοπής κάποιας άλλης. Οι τρις αυτές μέθοδοι αναλύονται στο υποκεφάλαιο 5.3.4, εν ολίγης η `get_input` δέχεται τις εντολές από το χρήστη, τις επεξεργάζεται και ανάλογα με το αν έχει βρει κλειδί ή όχι μεταβαίνει στη κατάσταση που αντιστοιχεί στο(-α) κλειδί(-α) αλλιώς παραμένει στην ίδια μέχρι να βρεθεί κλειδί. Η μέθοδος `more_than_one_result` ελέγχει αν υπάρχουν διαθέσιμα περισσότερα από ένα αποτελέσματα (από την αναζήτηση του χρήστη ή το καλάθι του) ή όχι, και μπορεί να μεταβεί στην ανάλογη κατάσταση. Τέλος, η `confirmation` ζητάει επιβεβαίωση από το χρήστη για το αν μπορεί να συνεχίσει μια ενέργεια ή όχι και μεταβαίνει στις ανάλογες καταστάσεις. Η πραγματική μετάβαση από την τωρινή κατάσταση (self.current_state) σε μια άλλη γίνεται με την κλήση της μεθόδου `move_to_state` που δέχεται όρισμα μια κατάσταση `i` και έπειτα μεταβαίνει σε αυτή.

Τελευταίος τρόπος με τον οποίο γίνονται μεταβάσεις είναι η χρήση σημείων ελέγχων (checkpoint). Μια τέτοια μετάβαση ενεργοποιείται όταν έχει παρουσιαστεί κάποιο απρόβλεπτο σφάλμα στη ροή της εφαρμογής. Ως checkpoint θέτονται καταστάσεις που εκτελούν την μέθοδο `get_input`, έτσι όταν κάποια άλλη δεν μπορεί να συνεχίσει τη λειτουργία της μεταβαίνει σε αυτό.

Οι καταστάσεις φορτώνονται όταν κατασκευάζεται το αντικείμενο `CommandProcessor`, από τη συνάρτηση `load_states` (Εικόνα 33) του αρχείου “additional_functions.py”. Σαν ορίσματα δέχεται τα στοιχεία της βάσης στην οποία θα γίνει η σύνδεση (host,user,passwd,database) και τη boolean μεταβλητή `states_db` που αν είναι True τότε οι καταστάσεις φορτώνονται από τη βάση δεδομένων ενώ αν είναι False από ένα αρχείο csv. Στην πρώτη περίπτωση απαιτούνται τα στοιχεία της βάσης δεδομένων, ενώ στη δεύτερη χρειάζεται να δοθεί το μονοπάτι path στο οποίο βρίσκεται το .csv αρχείο (Εικόνα 34). Και στις δύο περιπτώσεις η συνάρτηση επιστρέφει μια λίστα αποτελούμενη από λεξικά (ένα λεξικό για κάθε κατάσταση με κλειδιά “state”, “name”, “function_name”, “next_state”, “messages”, “keys”, “arguments”) από καταστάσεις.

```
self.states = load_states(ARGS.states_db, self.MYSQL_HOST, self.MYSQL_USER, self.MYSQL_PASSWD,
                          self.MYSQL_DATABASE)
```

Εικόνα 33

Στην παρακάτω Εικόνα 35 φαίνεται η μορφή μιας κατάστασης της εφαρμογής, η οποία δέχεται εντολές από το χρήστη και περιμένει μέχρι να αναγνωρίσει τα κλειδιά “keys” για να μεταβεί στις αντίστοιχες καταστάσεις “next_state”

```
state : 1
name : Main Input
function_name : get_input
next_state : 2,8,13,14,16
messages : []
keys : [['πρόσθεσε', 'καλάθι'], ['αφαίρεσε', 'καλάθι'], ['ολοκλήρωση', 'παραγγελίας'], ['άδειασε', 'καλάθι'], ['δείξε', 'καλάθι']]
arguments :
```

Εικόνα 35

Η εφαρμογή ξεκινά με την κλήση της μεθόδου start η οποία εκτελεί μια συνεχής επανάληψη μέχρι να βρεθεί στην τελική κατάσταση `FINAL_STATE`. Αρχικά εκτελεί τη μέθοδο της τωρινής κατάστασης (η μέθοδος δηλώνεται από το κλειδί “function_name”) και αν το αποτέλεσμα της είναι False, τότε συνεχίζει η επανάληψη και δεν εκτελείται ο υπόλοιπος κώδικας. Έπειτα, βρίσκει σε ποια κατάσταση πρέπει να μεταβεί η εφαρμογή (δηλώνεται από το κλειδί “next_state” της τωρινής κατάστασης) και στη συνέχεια ψάχνει το δείκτη της επόμενης κατάστασης στη λίστα των καταστάσεων και καλεί τη `move_to_state` για να πραγματοποιήσει τη μετάβαση.

Ο λόγος που αναζητεί στη λίστα των καταστάσεων (states) είναι επειδή οι καταστάσεις δεν είναι ταξινομημένες με βάση τον αριθμό τους επομένως ο αριθμός (κατάσταση) του κλειδιού “next_state” μιας κατάστασης διαφέρει από τον δείκτη που δείχνει σε αυτή. Για παράδειγμα, αν η λίστα states περιέχει τις καταστάσεις:

state	name	function_name	next_state	message	...	...
0	First state	message_once	2	[“Hello World”]	...	...
2	Second state	empty_cart	...	[]	...	...
10	Cant reach this	get_input	0,2	[]	...	...

Table 4 Πίνακας 4

Τότε

states[0][“state”] = 0
states[1][“state”] = 2
states[2][“state”] = 10

και αν βρισκόμαστε στη κατάσταση 0, η μετάβαση στη states[2] είναι λάθος καθώς δεν θα γίνει μετάβαση στην κατάσταση με state=2 αλλά με state=10. Άρα πρέπει να βρεθεί ποιος δείκτης δείχνει στη κατάσταση με “state” το 2, στο παράδειγμα αυτό είναι ο δείκτης 1 (states[1])

5.3.2 Tags

Οι ετικέτες (tags) είναι ένας τρόπος για να μεταφέρονται τα δεδομένα που μπορεί να έχει παράξει μια κατάσταση στην επόμενη. Όπως αναλύθηκε στο υποκεφάλαιο 5.3.1, οι καταστάσεις στην εφαρμογή αποθηκεύονται σε μια λίστα, η οποία περιέχει ένα λεξικό για κάθε κατάσταση που αποτελείται με τη σειρά του από τα ανάλογα κλειδιά που προσδιορίζουν αυτή τη κατάσταση. Τα tags λοιπόν, προστίθενται σε αυτά τα λεξικά (κατά την εκτέλεση μιας μεθόδου) για να περάσουν κάποια πληροφορία στις καταστάσεις που ακολουθούν που ίσως να βασίζονται από αυτή.

Χαρακτηριστικό παράδειγμα είναι όταν εκτελείται η μέθοδος `get_input` και δεχτεί μια αποδεκτή εντολή από το χρήστη, τότε μετά την επεξεργασία της εντολής την αποθηκεύει στο tag “input” (στο λεξικό της τωρινής κατάστασης). Στην Εικόνα 36 φαίνεται ότι στην κατάσταση 1 (από την Εικόνα 35 παραπάνω) έχει προστεθεί ένα ακόμα κλειδί στο λεξικό, το “input” που θα αποκαλούμε tag αντί για κλειδί με τιμή “1 προϊόν” που προέκυψε από την επεξεργασία της εντολής “αγόρασε ένα προϊόν”.

```
αγόρασε ένα προϊόν

state : 1
name : Main Input
function_name : get_input
next_state : 2,8,13,14,16
messages : []
keys : [['πρόσθεσε', 'καλάθι'], ['αφαίρεσε', 'καλάθι'], ['ολοκλήρωση', 'παραγγελίας'], ['άδειασε', 'καλάθι'], ['δείξε', 'καλάθι']]
arguments :
input :  1 προϊόν
```

Εικόνα 36

Τα tags μεταφέρονται μεταξύ των καταστάσεων από τη μέθοδο `move_to_state` η οποία και πραγματοποιεί τη μετάβαση. Η μέθοδος ψάχνει να βρει ποια tags (κλειδιά του λεξικού της τωρινής κατάστασης) δεν υπάρχουν σε μια προκαθορισμένη λίστα από κλειδιά (τα κλειδιά που προσδιορίζουν την κατάσταση) και όποιο δεν ανήκει σε αυτή, το μεταφέρει στην επόμενη κατάσταση. Ο κώδικας της φαίνεται στην Εικόνα 37, η πρώτη επανάληψη ψάχνει στη λίστα από λεξικά (καταστάσεις) και αν βρει ότι το κλειδί “state” (αριθμός της κατάστασης) ταιριάζει με την κατάσταση στην οποία θέλουμε να μεταβούμε (`i`) συνεχίζει στη δεύτερη επανάληψη η οποία μεταφέρει τα κλειδιά στην επόμενη κατάσταση (`self.states[int(next_index)]`) και έπειτα διαγράφει τα tags από την τωρινή κατάσταση. Τέλος, αλλάζει την τωρινή κατάσταση στην επόμενη (`self.current_state = int(next_index)`) ολοκληρώνοντας έτσι τη μετάβαση.

```
def move_to_state(self, i):
    for next_index, next_state in enumerate(self.states):
        if int(next_state['state']) == i:
            # Pass all the tags in the next state
            for key in self.states[self.current_state]:
                if key not in ["state", "name", "function_name", "next_state", "messages", "keys", "arguments"]:
                    self.states[int(next_index)][key] = self.states[self.current_state][key]
            # Remove all the non-default keys from the current state
            self.states[self.current_state] = {k: v for k, v in self.states[self.current_state].items() if
                                               k in ["state", "name", "function_name", "next_state",
                                                     "messages", "keys", "arguments"]}
            self.current_state = int(next_index)
            break
    return True
```

Εικόνα 37

Εκτός από την `get_input`,τα tags δημιουργούνται και από άλλες μεθόδους που θα δούμε σε αργότερα υποκεφάλαια, όπου και θα αναφέρουμε τα απαραίτητα tag που χρειάζεται μια μέθοδος για τη λειτουργία της.

Ορισμένες ετικέτες μπορούν να χρησιμοποιηθούν από το διαχειριστή της εφαρμογής στα μηνύματα για να δείξουν στοιχεία όπως αριθμούς, λέξεις ή προτάσεις στον χρήστη. Πιο συγκεκριμένα, αυτά τα tag είναι τα εξής:

- {input} η εντολή που έδωσε ο χρήστης
- {quantity} η ποσότητα ενός προϊόντος που πρόκειται να αγοράσει
- {price} η τιμή ενός προϊόντος
- {item} το αναγνωριστικό του προϊόντος
- {item_name} το όνομα ενός προϊόντος που επρόκειτο να προσθέσει ή να αφαιρέσει στο/από το καλάθι του ο χρήστης
- {quantity_price} η συνολική τιμή μιας ποσότητας προϊόντος (quantity*price)

Για παράδειγμα, ένα μήνυμα θα μπορούσε να είναι το ακόλουθο:

“Θα θέλατε να προσθέσετε {quantity} {item_name} στο καλάθι σας; Η συνολική τιμή ανέρχεται στα {quantity_price}€ .“

Επιπλέον, δύο ετικέτες χρησιμοποιούνται για να ορίσουν το χρονικό διάστημα μεταξύ επαναλαμβανόμενων μηνυμάτων καθώς και τον μέγιστο χρόνο που αυτά επαναλαμβάνονται. Οι ετικέτες είναι οι εξής:

- {interval}:x διάστημα μεταξύ των επαναλαμβανόμενων μηνυμάτων, όπου x ένας αριθμός που συμβολίζει δευτερόλεπτα
- {max_time}:x μέγιστος χρόνος επανάληψης των παραπάνω μηνυμάτων, το x πάλι συμβολίζει δευτερόλεπτα

Για παράδειγμα, το μήνυμα:

“Καλώς ήρθατε {interval}:30 {max_time}:120”

θα επαναλαμβάνεται κάθε 30 δευτερόλεπτα, για συνολικά 120 δευτερόλεπτα (2 λεπτά)

5.3.3 Λεξικό

Η εφαρμογή χρησιμοποιεί ένα λεξικό (να μην μπερδεύεται με τα λεξικά – τύπο δεδομένων της Python) για να μεταφράσει λέξεις που δίνει ο χρήστης. Το λεξικό αυτό βρίσκεται στο φάκελο “other_data/dictionary.txt” και σε αυτό το υποκεφάλαιο θα αναλύσουμε τα περιεχόμενά του και τους τρόπους που αξιοποιούνται.

Υπάρχουν τέσσερις υποκατηγορίες εντός του λεξικού. Ξεκινώντας από την πρώτη, η οποία δεν έχει όνομα, κάθε γραμμή αποτελείται από δύο λέξεις και ένα “=” (ισούται με), ας ονομάσουμε τώρα την γραμμή ως “κανόνα”. Ο κανόνας δηλώνει ότι η λέξη στα δεξιά του “=” θα αντικατασταθεί από τη λέξη στα αριστερά, στα σημεία της εφαρμογής που δέχεται εντολή ή δεδομένα από το χρήστη. Για παράδειγμα, στην Εικόνα 36 του προηγουμένου υποκεφαλαίου, ο χρήστης έδωσε την εντολή “αγόρασε ένα προϊόν”, αλλά σαν “input” αποθηκεύτηκε η πρόταση “1 προϊόν”. Αυτό συνέβη επειδή στο λεξικό υπάρχει ο κανόνας “1=ένα” που δηλώνει ότι η λέξη “ένα” αντικαθίσταται από τον αριθμό 1. Ακόμη μια περίπτωση χρήσης για αυτούς τους κανόνες είναι η μορφοποίηση των λέξεων που δίνει ο χρήστης, είτε πληκτρολογώντας ή μέσω της αναγνώρισης ομιλίας. Επειδή το μοντέλο αναγνώρισης ομιλίας δεν γνωρίζει τα κεφαλαία γράμματα, μπορεί εμείς να θέλουμε να αντικαταστήσουμε μερικές λέξεις με τις αντίστοιχες που περιέχουν κεφαλαία. Οι αντικαταστάσεις αυτές βοηθούν επίσης στην αναζήτηση προϊόντων.

Έπειτα, έχουμε την δεύτερη υποκατηγορία με όνομα [DONT_REPLACE] της οποίας οι κανόνες είναι ίδιοι με την πρώτη. Η διαφορά με την πρώτη, είναι πως οι λέξεις που αντικαθίστανται δεν αλλάζουν την αρχική εντολή. Δηλαδή, αν υπήρχε εδώ ο κανόνας “1=ένα” (και όχι στην πρώτη κατηγορία) τότε η εντολή “αγόρασε ένα προϊόν” θα γινόταν “ένα προϊόν”, όταν όμως γινόταν η αναζήτηση για αυτό στη βάση δεδομένων η πραγματική εντολή θα ήταν “1 προϊόν”. Μια τέτοια λειτουργία είναι χρήσιμη όταν δεν θέλουμε να “πειράξουμε” την αρχική εντολή του χρήστη.

Ένα ακόμη παράδειγμα, η εντολή “θέλω να αγοράσω ένα κινητό από την κατηγορία Κινητά”, εσωτερικά θα γίνει “ένα κινητό από την κατηγορία Smartphones” (λόγω του κανόνα “Smartphones=Κινητά”). Ο λόγος για την αντικατάσταση είναι πως στη βάση δεδομένων μπορεί να μην υπάρχει η κατηγορία “Κινητά”, συνεπώς δεν θα λαμβανόταν υπόψη στην εύρεση προϊόντων.

Η τρίτη κατηγορία είναι τα [ALT_KEYS]. Εδώ, η μορφή των κανόνων αλλάζουν, πλέον στα αριστερά υπάρχουν λέξεις χωρισμένες με κόμμα, στη μέση πάλι το “=” και στα δεξιά προτάσεις χωρισμένες με κόμμα. Η κάθε πρόταση περιέχει λέξεις χωρισμένες με κενό. Οι λέξεις στα αριστερά αντιπροσωπεύουν τις λέξεις κλειδιά των καταστάσεων ενώ η κάθε πρόταση στα δεξιά αντιπροσωπεύει μια ή περισσότερες εναλλακτικές λέξεις για τα κλειδιά αυτά.

Ας δούμε πάλι ένα παράδειγμα, έστω ότι βρισκόμαστε στη κατάσταση 1 (Εικόνα 35) τότε για να μεταβούμε στην κατάσταση 2 πρέπει η εντολή του χρήστη να περιέχει τις λέξεις “πρόσθεσε” και “καλάθι”. Εναλλακτικά, τώρα, μπορούμε να ορίσουμε και άλλες λέξεις για να γίνει η ίδια μετάβαση. Έστω ότι υπάρχει ο κανόνας “πρόσθεσε,καλάθι=θέλω να

αγοράσω,αγόρασε” τότε αν η εντολή του χρήστη περιέχει μια από τις ακολουθίες λέξεων (σε οποιαδήποτε σειρά):

1. πρόσθεσε **ΚΑΙ** καλάθι
2. θέλω **ΚΑΙ** να **ΚΑΙ** αγοράσω
3. αγόρασε

και στις 3 περιπτώσεις θα γίνει η μετάβαση στην κατάσταση 2.

Τέλος, η τέταρτη κατηγορία με όνομα [FRIENDLY_NAMES] ακολουθεί σχεδόν τους κανόνες τις πρώτης και δεύτερης κατηγορίας, αντικαθιστώντας τις λέξεις στα αριστερά με τις λέξεις στα δεξιά. Οι περιπτώσεις χρήσης αυτής αφορά κυρίως στήλες (της βάσης δεδομένων) ή χαρακτηριστικά προϊόντων. Όταν, για παράδειγμα, θέλουμε να στείλουμε το μήνυμα “Θα ήθελες το προϊόν να έχει Μαύρο σαν [ΧρώμαΜπλούζας];” τότε με τον κανόνα “[ΧρώμαΜπλούζας]=χρώμα μπλούζας” το μήνυμα γίνεται “Θα ήθελες το προϊόν να έχει Μαύρο σαν χρώμα μπλούζας;”. Γενικά όμως, αυτοί οι κανόνες αντικαθιστούν συγκεκριμένες λέξεις (αριστερά) με τις λέξεις από τα δεξιά όταν πρόκειται να σταλθεί μήνυμα στο χρήστη.

5.3.4 Μέθοδοι Ελέγχου

Έχουμε αναφερθεί προηγουμένως στις μεθόδους `get_input`, `more_than_one_result` και `confirmation`, πλέον τις ονομάζουμε συναρτήσεις ελέγχου για να τις διαφοροποιήσουμε κυρίως από τις υπόλοιπες καθώς η λειτουργία τους είναι παρόμοια. Αυτές οι τρις έχουν την ιδιότητα να μπορούν να επηρεάσουν την ροή των καταστάσεων η κάθε μια με το δικό της τρόπο.

Ξεκινώντας από την `get_input`, είναι η πιο βασική μέθοδος της εφαρμογής, αφού δέχεται μια εντολή και είναι ικανή να αναγνωρίσει λέξεις κλειδιά εντός αυτής. Ανάλογα με ποιο κλειδί εντόπισε, θα κάνει μετάβαση στην κατάσταση που αντιστοιχεί σε αυτό. Επιπλέον, αποτελεί ένα “στρώμα ασφαλείας” επειδή ορίζεται σαν `checkpoint` έτσι ώστε αν υπάρξει σφάλμα στη ροή του προγράμματος, να επιστρέψει αυτό εκεί.

Δέχεται συνεχώς εντολές (μέσω του stdin στην Εικόνα 9) μέχρι να βρεθεί ένα (ή περισσότερα) κλειδί. Κάθε φορά που μια εντολή είναι διαθέσιμη για επεξεργασία, ψάχνει να βρει αν υπάρχει κάποιο από τα κλειδιά (λίστα κλειδιών) που δίνονται κατευθείαν στην κατάσταση (πχ Εικόνα 35). Αν το συγκεκριμένο κλειδί (ή κλειδιά) δεν βρεθεί, τότε αναζητά την ύπαρξη κλειδιών με εναλλακτική ονομασία ([ALT_KEYS], υποκεφάλαιο 5.3.3), αν και πάλι δεν βρεθεί, ελέγχει το επόμενο κλειδί (ή λίστα κλειδιών) διαθέσιμο σε αυτή τη κατάσταση και επαναλαμβάνεται η διαδικασία. Όταν βρεθεί είτε βασικό ή εναλλακτικό κλειδί, αντικαθιστά τις λέξεις των κλειδιών στην εντολή του χρήστη με κενό ώστε να μείνει μια πρόταση χωρίς τα κλειδιά που χρησιμοποιήθηκαν (εξού και η μετατροπή της εντολής “αγόρασε ένα προϊόν” σε “ένα προϊόν” ή “1 προϊόν” στα παραδείγματα προηγούμενων υποκεφαλαίων) και έπειτα δημιουργεί το tag “input” με τιμή την υπολειπόμενη εντολή του χρήστη. Τέλος, καλεί τη μέθοδο `move_to_state` με όρισμα την επόμενη κατάσταση (

`int(next_states[i])`) που πρέπει να μεταβεί το πρόγραμμα και σταματάει η επανάληψη.

Η μέθοδος `more_than_one_result` έχει πιο απλή λειτουργία. Αρχικά κοιτάει να δει αν υπάρχει το tag “results” (θα δούμε αργότερα πως δημιουργείται), αν δεν υπάρχει τότε επιστρέφει στο τελευταίο `checkpoint`. Όπως βλέπουμε στην Εικόνα 38, έχει δύο πιθανές επόμενες καταστάσεις τις 4 και 5 και δύο κλειδιά, τα True και False (δεν μπορεί να πάρει άλλες τιμές για αυτά τα κλειδιά). Αν στη λίστα που παραπέμπει το tag “results” υπάρχουν πάνω(ή ακριβώς) από δύο αντικείμενα, τότε ισχύει το κλειδί “True” και σταματά τη λειτουργία της καλώντας τη μέθοδο `move_to_state` για να μεταβεί στην κατάσταση που αντιστοιχεί στο True (4). Αν όμως δεν ισχύει η συνθήκη, τότε δημιουργεί τα tags “item_name” και “price”, “item” που συναντήσαμε στο υποκεφάλαιο 5.3.2, καθώς και το “column_names” που θα δούμε αργότερα.

```
state : 3
name : More than 2 results
function_name : more_than_one_result
next_state : 4,5
messages : []
keys : [['True'], ['False']]
arguments :
```

Εικόνα 38

Τελευταία μέθοδος ελέγχου είναι η `confirmation` η οποία ζητά την επιβεβαίωση του χρήστη για να πραγματοποιήσει μια μετάβαση. Δέχεται σαν “messages” ένα μήνυμα(πχ [“Θα ήθελες να συνεχίσεις;”]), το στέλνει στον χρήστη μέσω της μεθόδου `message_once` και περιμένει για απάντηση. Αν ο χρήστης απαντήσει θετικά, και ως θετικά εννοούμε η απάντησή του να περιέχει μια λέξη που βρίσκεται στη λίστα `confirm_pos_strs` τότε μεταβαίνει στη κατάσταση που αντιστοιχεί στη θετική απάντηση. Παρομοίως, αν η απάντησή του περιέχεται στη λίστα με τις αρνητικές απαντήσεις `confirm_neg_strs` τότε μεταβαίνει στην κατάσταση της αρνητικής απάντησης. Μια κατάσταση `confirmation` φαίνεται στην Εικόνα 39 που ζητάει την άδεια του χρήστη για να αδειάσει το καλάθι του.

```
state: 14
name: Ask for confirmation to empty the cart
function_name: confirmation
next_state: 15,1
messages: ['θέλεις να αφαιρέσεις όλα τα προιόντα απο το καλάθι σου;']
arguments:
```

Εικόνα 39

5.3.5 Μέθοδοι Μηνυμάτων

Οι μέθοδοι μηνυμάτων στέλνουν μηνύματα στο χρήστη (μέσω stdout γραμμή 107 Εικόνα 8) και είναι τρις στο σύνολο. Πριν περάσουμε στην ανάλυσή τους, τα μηνύματα που

στέλνουν βρίσκονται συνήθως στο “messages” μιας κατάστασης. Η δομή αυτής της τιμής είναι [“μήνυμα 1”,”μήνυμα 2”], δηλαδή είναι μια λίστα από προτάσεις (strings). Στην περίπτωση που γράφονται χειροκίνητα οι καταστάσεις, είναι απαραίτητες οι αγκύλες [ ] και τα εισαγωγικά “ ”. Αν μια κατάσταση δεν περιέχει μήνυμα, τότε το “message” πρέπει να είναι οι κενές αγκύλες [ ]. Αν όμως οι καταστάσεις δημιουργούνται από το script “edit_and_import.py” τότε δεν υπάρχει τέτοια ανησυχία.

Ας περάσουμε λοιπόν στη μέθοδο `message_once`. Σαν ορίσματα δέχεται τα `msg` και `get_msg`, το πρώτο της δίνει ένα συγκεκριμένο μήνυμα για να εκτυπώσει, ενώ με το δεύτερο δεν εκτυπώνει κανένα μήνυμα, αλλά το επεξεργάζεται και το επιστρέφει στο σημείο του προγράμματος που την κάλεσε. Τα ορίσματα αυτά χρησιμοποιούνται μόνο από άλλες μεθόδους της εφαρμογής.

Αν υπάρχουν περισσότερα από ένα μηνύματα (στο “messages” της τωρινής κατάστασης) τότε επιλέγεται ένα στη τύχη. Στη συνέχεια, αντικαθιστά όλα τα φιλικά ονόματα ([FRIENDLY_NAMES], υποκεφάλαιο 5.3.3) εντός του μηνύματος, έπειτα όλα τα tags (υποκεφάλαιο 5.3.2) και εκτυπώνει το μήνυμα με την εντολή print.

Η εφαρμογή δίνει επίσης τη δυνατότητα επαναλαμβανόμενων μηνυμάτων χωρίς να διακόπτεται η ροή της. Αυτό επιτυγχάνεται με τη χρήση threads (νήματα) με τη βοήθεια της βιβλιοθήκης “threading”[24]. Όταν κατασκευάζεται η κλάση `CommandProcessor` αρχικοποιείται η μεταβλητή `ERM`, η οποία είναι υπεύθυνη για την αναστολή λειτουργίας του thread. Η μέθοδος `message_repeating` αρχικά επαναφέρει την μεταβλητή `ERM` από προηγούμενη χρήση, και καλεί τη `message_once` με το όρισμα `get_msg=True` για να της επιστρέψει το μήνυμα που θα επαναληφθεί. Έπειτα, αναζητεί τα tags “{interval}:” και “{max_time}:” στο μήνυμα, και με τη βοήθεια μιας συνάρτησης (`get_number_msg` από το αρχείο “additional_functions.py”) ανακτά τις αντίστοιχες τιμές (τα tags είναι προαιρετικά, καθώς οι τιμές αρχικοποιούνται από ένα αρχείο με τις ρυθμίσεις της εφαρμογής). Αμέσως μετά, ξεκινά το thread `message_thread` το οποίο κρατάει τον χρόνο λειτουργίας του, και κάθε “{interval}” δευτερόλεπτα καλεί την `message_once` με παράμετρο `msg=message_to_repeat` το οποίο είναι το μήνυμα που επιλέχθηκε προηγουμένως στη μέθοδο `message_repeating`.Η λειτουργία του σταματά όταν φτάσει στον χρόνο {max_time} ή γίνει True το `ERM` (από τη συνάρτηση `get_input`).

5.3.6 Απόκτηση Δεδομένων

Η απόκτηση των δεδομένων από τη βάση δεδομένων για την εύρεση ενός μόνο προϊόντος, είναι μια διαδικασία με δύο βασικά και ένα προαιρετικό μέρος. Το πρώτο βασικό μέρος αποτελείται από την άντληση δεδομένων από τη βάση και το δεύτερο από το φιλτράρισμα αυτών για να γίνουν διαθέσιμα τα πιο σχετικά προϊόντα με αυτό που αναζήτησε ο χρήστης. Το προαιρετικό μέρος μπορεί να μειώσει τον συνολικό αριθμό των προϊόντων στα “results”, αλλά πιθανώς με κάποιο κόστος.

[24] https://docs.python.org/3/library/threading.html

Το πρώτο μέρος υλοποιείται με τη μέθοδο `load_items_from_db`. Αρχικά, χωρίζει την εντολή του χρήστη σε μικρότερα κομμάτια (tokens) με τη βοήθεια της συνάρτησης `get_tokens` του αρχείου “additional_functions.py”, έπειτα πραγματοποιεί μια αναζήτηση (query) και ψάχνει κάθε προϊόν που περιέχει σε ορισμένες στήλες (δηλώνονται από τη μεταβλητή `new_searching_priorities`) κάποιο από τα token και επιστρέφει τα προϊόντα (“results”) αλλά και τις στήλες στις οποίες έγινε η αναζήτηση μαζί με τις στήλες των χαρακτηριστικών των προϊόντων αυτών (“column_names”). Παράδειγμα φαίνεται στην Εικόνα 40, όπου με την αναζήτηση “υπολογιστή” έχουν επιστραφεί πέντε υπολογιστές από τη βάση δεδομένων μαζί με τα χαρακτηριστικά τους (Graphics έως Type).

```
Βάση υπολογιστής
id, price, name, primary_categories, categories, brand, description, Graphics, RAM, ReleaseDate, Resolution, Storage, StorageType, Type,
1 599.99 Υπολογιστής Ι Κάρτα Γραφικών 1, 5GB RAM και 128GB SSD Electronics Computers,Gaming,Desktop Μάρκα Α Περιγραφή Υπολογιστή Ι Κάρτα Γραφικών 1 5GB None None 128GB SSD None
2 699.99 Υπολογιστής ΙΙ Κάρτα Γραφικών 1, 4GB RAM και 128GB SSD Electronics Computers,Gaming,Desktop Μάρκα Α Περιγραφή Υπολογιστή ΙΙ Κάρτα Γραφικών 1 4GB None None 128GB SSD None
3 699.99 Υπολογιστής ΙΙΙ Κάρτα Γραφικών 2, 5GB RAM Electronics Computers,Gaming,Desktop Μάρκα Α Περιγραφή Υπολογιστή ΙΙΙ Κάρτα Γραφικών 2 5GB None None None None None
4 899.99 Υπολογιστής IV Κάρτα Γραφικών 2, 4GB RAM και 64GB HDD Electronics Computers,Gaming,Desktop Μάρκα Β Περιγραφή Υπολογιστή IV Κάρτα Γραφικών 2 4GB None None 64GB HDD None
5 499.99 Υπολογιστής IV Κάρτα Γραφικών 1, 6GB RAM και 64GB SSD Electronics Computers,Gaming,Desktop Μάρκα Β Περιγραφή Υπολογιστή V Κάρτα Γραφικών 1 6GB None None 64GB SSD None
```

Εικόνα 40

Η `get_tokens` δέχεται μια πρόταση, μια λίστα για λέξεις που πρέπει να αγνοήσει, το λεξικό [DONT_REPLACE] από το υποκεφάλαιο 5.3.3 και έναν αριθμό που δηλώνει το μήκος των tokens. Αν ο αριθμός `sub_tokens` είναι 0, τότε ένα token είναι μια λέξη της πρότασης (χωρίζεται η πρόταση με κενό), αλλιώς περνάει από κάθε λέξη και τη χωρίζει σε υπό λέξεις μήκους `sub_tokens` κρατώντας και τις αρχικές, δηλαδή αν η πρόταση ήταν “θέλω να αγοράσω αυτό το προϊόν” και το `sub_tokens=3` , τότε, τα τελικά tokens φαίνονται στη δεύτερη γραμμή στην Εικόνα 41(το “θέλω να αγοράσω” αφαιρείται αφού είναι κλειδί).

```
θέλω να αγοράσω αυτό το προϊόν
['αυτό', 'αυτ', 'ό', 'προϊόν', 'προ', 'ιόν']
```

Εικόνα 41

Το δεύτερο μέρος είναι αυτό που καλεί τη πρώτη μέθοδο, και υπολογίζει μια βαθμολογία για κάθε αποτέλεσμα, για να απορρίψει αποτελέσματα που πιθανώς να μην ταιριάζουν πολύ σε αυτό που αναζήτησε ο χρήστης. Αυτή είναι η `find_items`, η οποία μπορεί να κάνει αναζήτηση είτε σε προϊόντα στη βάση δεδομένων, ή στο καλάθι του χρήστη αν στο “arguments” της κατάστασης δοθεί η παράμετρος “--type cart”. Στην πρώτη περίπτωση, καλεί τη `load_items_from_db` για να αποκτήσει τα δεδομένα και δημιουργεί τα tag “results” και “column_names”. Στη δεύτερη περίπτωση, αν δεν υπάρχει το “column_names” επιστρέφει στο τελευταίο checkpoint ενώ αν υπάρχει δημιουργεί ένα αντίγραφο του καλαθιού του χρήστη για να εργαστεί σε αυτό.

Η συνέχεια είναι η ίδια και για τις δύο περιπτώσεις. Αρχικά, καλεί την συνάρτηση `calculate_weights` που υπολογίζει το βάρος κάθε στήλης που δηλώνεται από τη μεταβλητή `new_searching_priorities` και περιέχει στήλες χωρισμένες με κόμμα, από τις οποίες η στήλη στα αριστερά της δήλωσης έχει μεγαλύτερη προτεραιότητα από τη στήλη στα δεξιά. Η διαφορά μεταξύ των προτεραιοτήτων των στηλών είναι 0.2, για παράδειγμα αν `new_searching_priorities=primary_categories,categories,brand,name,description` τότε αντίστοιχα η προτεραιότητα των στηλών αυτών είναι 2,1.8,1.6,1.4,1.2.

Έπειτα, καλεί τη συνάρτηση `calculate_scores` (additional_functions.py) που υπολογίζει τη διαφορά κάθε στήλης (`new_searching_priorities`) με κάθε token και το αποτέλεσμά της είναι το σφάλμα της αναζήτησης (tag “input”) με κάθε προϊόν στα tag “results”. Σαν παραμέτρους δέχεται μια λίστα από προϊόντα (“items”), τα “tokens” και τους πολλαπλασιαστές (βάρη) “multipliers” που βρέθηκαν προηγουμένως. Υπολογίζει μετά για κάθε στήλη κάθε προϊόντος τη διαφορά που έχει με κάθε token με τη βοήθεια της συνάρτησης `SequenceMatcher` του πακέτου “difflib” η οποία παράγει το ποσοστό ομοιότητας μεταξύ δύο λέξεων (της τιμής της στήλης και του token). Στη συνέχεια βρίσκει τη συνολική βαθμολογία αυτού του προϊόντος πολλαπλασιάζοντας την ατομική βαθμολογία κάθε στήλης με το αντίστοιχο βάρος της (multiplier) και προσθέτοντας το αποτέλεσμα κάθε στήλης στο συνολικό. Σαν τελικό βήμα, κανονικοποιεί τις βαθμολογίες όλων των προϊόντων ώστε να είναι μεταξύ 0 και 1.

Τέλος, εξετάζει αν το σφάλμα κάθε προϊόντος (`normalized_scores`) είναι επιτρεπτό (`score_error`) και εν τέλει μένουν μόνο τα προϊόντα με επιτρεπτό σφάλμα. Η τιμή του επιτρεπτού σφάλματος (`score_error`) ορίζεται στο αρχείο ρυθμίσεων.

Το τρίτο μέρος επεξεργάζεται περαιτέρω τα δεδομένα που έχουν επιστραφεί από τη πρώτη μέθοδο (ή και την δεύτερη). Η `get_majority_results` ψάχνει όλες τις τιμές κάθε στήλης που δηλώνονται από τη μεταβλητή `product_path` και υπολογίζει το ποσοστό συχνότητας κάθε μοναδικής τιμής αυτής της στήλης για κάθε αποτέλεσμα (“results”). Έπειτα, ελέγχει να δει αν αυτό το ποσοστό ξεπερνά ένα κατώτατο όριο (threshold) και αναλόγως αν το ξεπερνά τότε κρατάει τα αποτελέσματα με αυτή τη τιμή, ενώ αν όχι, τα διαγράφει. Ένα παράδειγμα που εξηγεί τη λειτουργία της είναι το εξής:

Έστω ότι το κατάστημά μας έχει 7 προϊόντα όπως φαίνεται στον παρακάτω πίνακα και ο χρήστης αναζητά μόνο με τη λέξη “Μπλούζα”

id	**name**	**color**	**size**
1	Μπλούζα 1	Μαύρο	Medium
2	Μπλούζα 2	Μπλε	Medium
3	Μπλούζα 3	Γκρι	Medium
4	Μπλούζα 4	Γαλάζιο	Large
5	Μπλούζα 5	Γαλάζιο	Medium
6	Μπλούζα 6	Γαλάζιο	Large
7	Μπλούζα 7	Μπλε	Medium

Και επίσης η μεταβλητή product_path=color,size

Επιπλέον, το κατώτατο όριο (threshold) της στήλης “color” είναι 30%, ενώ της “size” είναι 50%.

Τότε η μέθοδος θα υπολογίσει αρχικά τη συχνότητα κάθε μοναδικής τιμής στη στήλη “color”, έτσι έχουμε

Γαλάζιο: 3 ~ 42.85%
Μπλε: 2 ~ 28.57%
Μαύρο: 1 ~ 14.28%
Γκρι: 1 ~ 14.28%

Άρα, ξεκινώντας πρώτα από τη στήλη color (αφού είναι πρώτη σε σειρά στη μεταβλητή `product_path`) θα κρατήσει μόνο τις μπλούζες με χρώμα “Γαλάζιο” επειδή το threshold είναι 30% και καμία άλλη τιμή δεν το φτάνει. Ο πίνακας των αποτελεσμάτων διαμορφώνεται ως:

id	name	color	size
4	Μπλούζα 4	Γαλάζιο	Large
5	Μπλούζα 5	Γαλάζιο	Medium
6	Μπλούζα 6	Γαλάζιο	Large

Περνάει στη στήλη “size”, ξανά υπολογίζει τα ποσοστά συχνότητας εμφάνισης των μοναδικών τιμών:

Large: 2 ~66.7%
Medium: 1 ~33.3%

Και τελικά, αφού το threshold για το size είναι 50%, θα διαγραφτεί η Μπλούζα 5 που έχει Medium και θα μείνουν οι μπλούζες 4 και 6.

Τα πλεονεκτήματα αυτής της μεθόδου είναι ότι μπορεί να μειώσει πολύ τον όγκο των δεδομένων που επιστρέφονται από τη βάση δεδομένων με τη `load_items_from_db`, και να προσαρμοστεί καθώς και τα threshold και τα `product_path` είναι παραμετροποιήσιμα στο αρχείο “other_data/settings.ini”. Από την άλλη, μπορεί να διαγράψει δεδομένα που πιθανώς να μην έπρεπε, γι’ αυτό το λόγο προτείνεται το threshold να είναι σχετικά μικρό κοντά στο 10-15%, αλλά φυσικά μπορεί να απενεργοποιηθεί εντελώς θέτοντας τη μεταβλητή `get_majority=0` στο αρχείο ρυθμίσεων και απλά να μην δημιουργηθεί κατάσταση με αυτή τη μέθοδο. Τα threshold ορίζονται στο αρχείο ρυθμίσεων και συγκεκριμένα στο πεδίο [attribute_thresholds].

5.3.7 Αναζήτηση Χαρακτηριστικών

Μετά την απόκτηση των σχετικών δεδομένων από τη βάση με τις μεθόδους του υποκεφαλαίου 5.3.6, μπορούμε να κάνουμε αναζήτηση στα συγκεκριμένα χαρακτηριστικά κάθε προϊόντος για να καταλήξουμε μόνο σε ένα. Αυτό μπορεί να γίνει με τις μεθόδους `find_specific_product` και `search_one_value` . Η πρώτη, είναι μια πιο αυτοματοποιημένη έκδοση της δεύτερης καθώς μπορεί από μόνη της να βρει ένα συγκεκριμένο προϊόν (φυσικά συμβουλευόμενη τον πελάτη για τις προτιμήσεις του) και να χρησιμοποιήσει τη δεύτερη όταν χρειάζεται. Η δεύτερη, μπορεί να καθοδηγηθεί από τον

διαχειριστή για να ψάχνει για συγκεκριμένες τιμές σε συγκεκριμένα χαρακτηριστικά (στήλες), βέβαια κάτι τέτοιο δεν θα ήταν αποτελεσματικό για μεγάλες βάσεις δεδομένων.

Ας ξεκινήσουμε λοιπόν από τη δεύτερη μέθοδο, την `search_one_value`. Δέχεται παραμέτρους (όταν καλείται από κάποιο σημείο του κώδικα) και "arguments" (όταν θέλει ο διαχειριστής να δώσει συγκεκριμένες τιμές), οι παράμετροι απαρτίζονται από "value", μια τιμή την οποία θα αναζητήσει, "col" η στήλη στην οποία θα γίνει η αναζήτηση (προϋποθέτει λοιπόν την ύπαρξη του tag "column_name"), "message" ένα μήνυμα που θα στείλει στο χρήστη και "results" μια λίστα από προϊόντα (αποτελέσματα). Τα "arguments" από την άλλη δίνονται όταν καλείται απευθείας από μια κατάσταση και όχι από τη μέθοδο "find_specific_product", συγκεκριμένα είναι τα "--value[κενό][τιμή]" που αναλογεί στο όρισμα "value", "--col[κενό][τιμή]" που αναλογεί στο όρισμα "col", και ένα μήνυμα "--message[κενό][μήνυμα]". Αν κληθεί από κατάσταση και δεν έχουν δοθεί τα "--value" και "--col" επιστρέφει στο τελευταίο checkpoint.

Τα μηνύματα μπορούν να περιέχουν τη λέξη "[col]" η οποία αντικαθίσταται με τη τιμή της παραμέτρου "--col" έτσι ώστε να εμφανιστεί το [FRIENDLY_NAMES] που της αναλογεί. Επιπλέον, ένα καινούριο tag που χρησιμοποιείται και από τις δύο μεθόδους είναι το "{possible_values}" το οποίο αν υπάρχει στο μήνυμα θα αντικατασταθεί με τη τιμή της παραμέτρου "--value".

Έπειτα, δέχεται εντολή από το χρήστη (απάντηση στο μήνυμα) και εξετάζει αν αυτό είναι θετικό (ανήκει στη λίστα `confirm_pos_strs`) ή αρνητικό (`confirm_neg_strs`). Αν είναι θετικό, τότε αφαιρεί όλα τα προϊόντα (αποτελέσματα) που **δεν** έχουν τη τιμή "value" (ή "--value") σαν "col" (ή "--col"), ενώ αν είναι αρνητικό, αφαιρεί όλα τα προϊόντα που **έχουν** αυτή τη τιμή στο "col" από το tag "results".

Στην ουσία, αυτή η μέθοδος μπορεί να αξιοποιηθεί για να ρωτήσει το χρήστη αν θέλει μια συγκεκριμένη τιμή σε μια συγκεκριμένη στήλη (πχ χαρακτηριστικό) και να αφαιρέσει τα ανάλογα προϊόντα από τα "results", ώστε να καταλήξει σιγά σιγά σε ένα μόνο προϊόν.

Η `find_specific_product` δέχεται σαν "arguments" τα "--multiple_msg[κενό][μήνυμα]" για να αλλάξει το μήνυμα που θα σταλθεί στο χρήστη όταν έχει να επιλέξει μεταξύ πολλών τιμών για μια μόνο στήλη, "--single_msg" για να αλλάξει το μήνυμα όταν ο χρήστης πρέπει να επιλέξει αν θέλει μια τιμή σε μια στήλη ή όχι και "not_found" το μήνυμα που θα εμφανίζεται όταν δεν κατάφερε η μέθοδος να βρει κάποιο προϊόν.

Παρομοίως με την `search_one_value`, το tag "{possible_values}" μπορεί να περιλαμβάνεται στο μήνυμα όπου ο χρήστης πρέπει να επιλέξει μεταξύ πολλών τιμών, για να του δείξει ποιες είναι αυτές (χωρισμένες με κόμμα).

Μετά την αρχικοποίηση των μεταβλητών column_names και results, ξεκινά η επανάληψη σε κάθε στήλη για να βρει μόνο μια τιμή για αυτή. Αν υπάρχει μόνο ένα αποτέλεσμα τότε διακόπτεται η επανάληψη, ενώ συνεχίζοντας βρίσκει όλες τις μοναδικές τιμές (`temp_unique_values`) στην τρέχουσα στήλη και έπειτα τις **πιθανές** τιμές(`possible_values`) για αυτή τη στήλη. Η διαφορά των μοναδικών με των πιθανών

τιμών είναι ότι οι πρώτες περιλαμβάνουν τις κενές τιμές (null ή None). Αν υπάρχει περισσότερη από μια πιθανή τιμή και δεν υπάρχει καμία πιθανή τιμή στην αναζήτηση που έκανε ο χρήστης, τότε ρωτάει το χρήστη ποια επιθυμεί, περιμένοντας μια απάντηση που ανήκει στη λίστα με τις πιθανές τιμές (στην είσοδο του χρήστη γίνεται αντικατάσταση πάλι από το λεξικό). Αν η είσοδος του χρήστη ανήκει στις τιμές της λίστας `dont_want_strs` τότε δηλώνει ότι δεν θέλει καμία από τις πιθανές τιμές, και διαγράφονται όλα τα προϊόντα που τις περιέχουν (στη τρέχουσα στήλη). Αν διαλέξει μια τιμή, τότε διαγράφονται όλα τα προϊόντα που δεν την έχουν στην τρέχουσα στήλη. Στον χρήστη δίνεται επίσης η δυνατότητα να ακυρώσει την αναζήτηση αν η είσοδός του ανήκει στη λίστα `confirm_neg_strs`. Επιστρέφοντας λίγο πίσω στον κώδικα, αν υπάρχει μόνο μια πιθανή τιμή, τότε καλείται η μέθοδος `search_one_value` που αναλύθηκε προηγουμένως για να τον ρωτήσει αν την θέλει ή όχι.

Τελευταίο βήμα είναι να δημιουργήσει τα tag "item_name" και "price" που είδαμε στο υποκεφάλαιο 5.3.2 και να ανανεώσει το tag αποτελεσμάτων "results".

5.3.8 Κλάση User

Η κλάση User (αρχείο user.py) δημιουργεί ένα αντικείμενο χρήστη, που αναπαριστά έναν χρήστη που κάνει μια παραγγελία. Ο κάθε χρήστης της κλάσης User αποτελείται από τα attributes:

- id: το μοναδικό αναγνωριστικό του
- phone: το τηλέφωνό του
- name: το όνομά του
- addr: η διεύθυνσή αποστολής των προϊόντων που παραγγέλνει
- shopping_cart: το καλάθι του που περιέχει τα προϊόντα που θέλει να αγοράσει
- ip: η διεύθυνση πρωτοκόλλου διαδικτύου του (Internet Protocol Address, IP)
- confirm_pos_strs: λίστα που περιέχει θετικές απαντήσεις
- confirm_neg_strs: λίστα που περιέχει αρνητικές απαντήσεις

Κατά τη δημιουργία ενός αντικειμένου User, είναι απαραίτητο να δοθεί είτε η IP του είτε το κινητό του ώστε να μπορεί να γίνει αμέσως η ταυτοποίησή του με την αντίστοιχη καταχώρηση στη βάση δεδομένων (πίνακας "users"). **Δεν πρέπει να χρησιμοποιούνται και οι δυο τρόποι ταυτόχρονα ή εναλλασσόμενα**.

Η μέθοδος `fetch_user_details` ψάχνει στη βάση δεδομένων να βρει τον χρήση με τη δοσμένη IP ή κινητό, και ορίζει το id (δίνεται αυτόματα σε κάθε χρήστη) του αντικειμένου User, το όνομά του και τη διεύθυνση αποστολής (αν υπάρχουν). Αν η IP ή το κινητό δεν υπάρχει στη βάση δεδομένων, δημιουργεί μια νέα γραμμή με τα δοσμένα στοιχεία για τον νέο χρήστη και ορίζει το id του αντικειμένου.

Η `add_to_cart` δέχεται μια λίστα από ένα προϊόν που περιέχει πληροφορίες όπως το id του, το όνομά του, τα χαρακτηριστικά του κ.α., καθώς και την ποσότητα "quantity" αυτού του προϊόντος που πρόκειται να προστεθεί στο καλάθι. Έπειτα ψάχνει να βρει αν

υπάρχει ήδη στο καλάθι, σε αυτή την περίπτωση ανανεώνει την ποσότητα προσθέτοντας την καινούρια, αλλιώς προσθέτει το ίδιο το προϊόν στη λίστα του καλαθιού. Τέλος, εκτυπώνει ένα μήνυμα επιτυχίας.

Η μέθοδος `remove_from_cart` δέχεται τα ίδια ορίσματα με την `add_to_cart`, αλλά κάνει το αντίθετο, δηλαδή αν υπάρχει ήδη το προϊόν και η ποσότητα "quantity" είναι μικρότερη από τη ποσότητα του προϊόντος τότε αφαιρείται αυτή η ποσότητα, αλλιώς αφαιρείται ολόκληρο το προϊόν και εκτυπώνει ένα μήνυμα επιτυχίας.

Κάποιοι μέθοδοι που υπάρχουν εντός της κλάσης και καλούνται από τη βασική εφαρμογή είναι:

- Η `empty_cart` απλά αδειάζει το καλάθι.
- Η `cart_to_str` μετατρέπει το τα προϊόντα σε πρόταση, με το όνομά τους, την τιμή τους και την ποσότητα τους με κατάλληλα μηνύματα αν το καλάθι είναι άδειο όταν καλείται.

Η `checkout` είναι αυτή που πραγματοποιεί την εικονική παραγγελία. Πρώτα ψάχνει να βρει τη διεύθυνση αποστολής της, ζητώντας από τον χρήστη να την καταχωρήσει αν χρειαστεί εκτυπώνοντας μηνύματα. Αν δεν υπάρχει αποθηκευμένη διεύθυνση στη βάση δεδομένων ή δώσει μια νέα τότε ανανεώνεται η τιμή της υπάρχουσας (που είναι κενή αν δεν υπάρχει ήδη). Ο χρήστης έχει τη δυνατότητα να διακόψει τη διαδικασία ολοκλήρωσης παραγγελίας κατά τη διάρκεια επιλογής διεύθυνσης. Έπειτα συλλέγει όλες τις απαραίτητες πληροφορίες για τη παραγγελία και δημιουργεί μια νέα καταχώρηση στο πίνακα "orders" της βάσης δεδομένων.

5.3.9 Άλλες Μέθοδοι

Ας συνεχίσουμε με τις μεθόδους της βασικής εφαρμογής (κλάσης CommandProcessor, main.py) που καλούν αυτές του προηγούμενου υποκεφαλαίου.

- `add_to_cart`: Καλεί τη αντίστοιχη μέθοδο `add_to_cart` του αντικειμένου `User` μέσω της μεταβλητής `user` και της δίνει σαν παραμέτρους το όνομα του προϊόντος που θα προστεθεί καθώς και την ποσότητά του. Αν υπάρχουν περισσότερα από ένα προϊόντα στο tag "results" επιλέγει ένα τυχαίο

- `remove_from_cart`: Παρομοίως με την `add_to_cart`, αλλά τα προϊόντα αφαιρούνται

- `checkout`: Καλεί τη μέθοδο `checkout` του αντικειμένου User και της δίνει το tag "column_name"

- `empty_cart`: Καλεί την αντίστοιχη μέθοδο του αντικειμένου `User`

- `print_cart`: Καλεί τη `cart_to_str` του αντικειμένου User, στην περίπτωση που αυτή αποτύχει επιστρέφει στο τελευταίο `checkpoint`. Αν όχι, εκτυπώνει το αποτέλεσμα της καλούμενης μεθόδου με την συνάρτηση `message_once`

- `ask_for_quantity`: η οποία ρωτάει το χρήστη τι ποσότητα θα ήθελε για το προϊόν που θα προσθέσει στο καλάθι του. Ο χρήστης μπορεί να δώσει αριθμό είτε σε ακέραια μορφή ή σε μορφή λέξεων, στην τελευταία περίπτωση τους μετατρέπει σε ακέραιους με τη συνάρτηση `gr_text_to_number`. Στη συνέχεια, δημιουργεί τα tag “quantity” και “quantity_price”

Ακόμη, δύο μέθοδοι του αρχείου “additional_functions.py” που δεν έχουν αναλυθεί προς το παρόν είναι η `remove_last_str` η οποία δέχεται ένα string `str` και ένα άλλο string `rmv` και αφαιρεί το `rmv` από το τέλος του `str`. Επιπλέον, η `gr_text_to_number` δέχεται ένα string `text` από αριθμούς (πχ εκατόν τρία) και το μετατρέπει σε ακέραιους (103), ο μέγιστος αριθμός που μπορεί να μετατρέπει είναι μέχρι και 999.

ΚΕΦΑΛΑΙΟ 6 Παράδειγμα Χρήσης Εφαρμογής

Σε αυτό το κεφάλαιο θα περιγραφτεί η διαδικασία της ρύθμισης της εφαρμογής του Κεφαλαίου 5 για την εξυπηρέτηση πελατών και ενδεικτικά η διαδικασία αγοράς προϊόντων μέσω της εφαρμογής του Κεφαλαίου 4.

Ο κώδικας της εφαρμογής βρίσκεται στον σύνδεσμο https://github.com/dnckl/ptux

6.1 Εισαγωγή Προϊόντων στη Βάση Δεδομένων

Για τη βάση δεδομένων χρησιμοποιούμε το πρόγραμμα xampp, όπως φαίνεται στην Εικόνα 30 και έπειτα το «phpMyAdmin» («http://localhost/phpmyadmin/»).

Αρχικά, πρέπει να προσδιορίσουμε τα προϊόντα του καταστήματος, στην περίπτωσή μας το κατάστημα θα προσφέρει τρις κατηγορίες προϊόντων :

1. Υπολογιστές
2. Κινητά
3. Τηλεοράσεις

Καθένα από τα παραπάνω θα έχει τα ειδικά του χαρακτηριστικά, πιο συγκεκριμένα:

1. Ο κάθε υπολογιστής μπορεί να έχει κανένα, ένα ή περισσότερα από τα ακόλουθα:
    - Κάρτα γραφικών (“Graphics”)
    - Μέγεθος μνήμης RAM (Random Access Memory) (“RAM”), μετράται σε GB (Gigabytes)
    - Μέγεθος αποθηκευτικού χώρου (“Storage”), μετράται σε GB (Gigabytes)
    - Τύπος αποθηκευτικού χώρου (“StorageType”)

Για καθένα από τα τέσσερα ειδικά χαρακτηριστικά των υπολογιστών το κατάστημα έχει διαθέσιμα τα μοντέλα – είδη:

Graphics
Κάρτα Γραφικών 1
Κάρτα Γραφικών 2

RAM
4GB
5GB
6GB

Storage
64GB
128GB

StorageType
SSD
HDD

2. Το κάθε κινητό μπορεί να έχει κανένα, ένα ή περισσότερα από τα ακόλουθα:
    - Μέγεθος μνήμης RAM (“RAM”)
    - Μέγεθος αποθηκευτικού χώρου (“Storage”)
    - Έτος κυκλοφορίας (“ReleaseDate”)

Το κατάστημα παρέχει τα ίδια μοντέλα με τους Υπολογιστές για τα χαρακτηριστικά “RAM” και “Storage”, επιπλέον παρέχει:

ReleaseDate
2020
2021

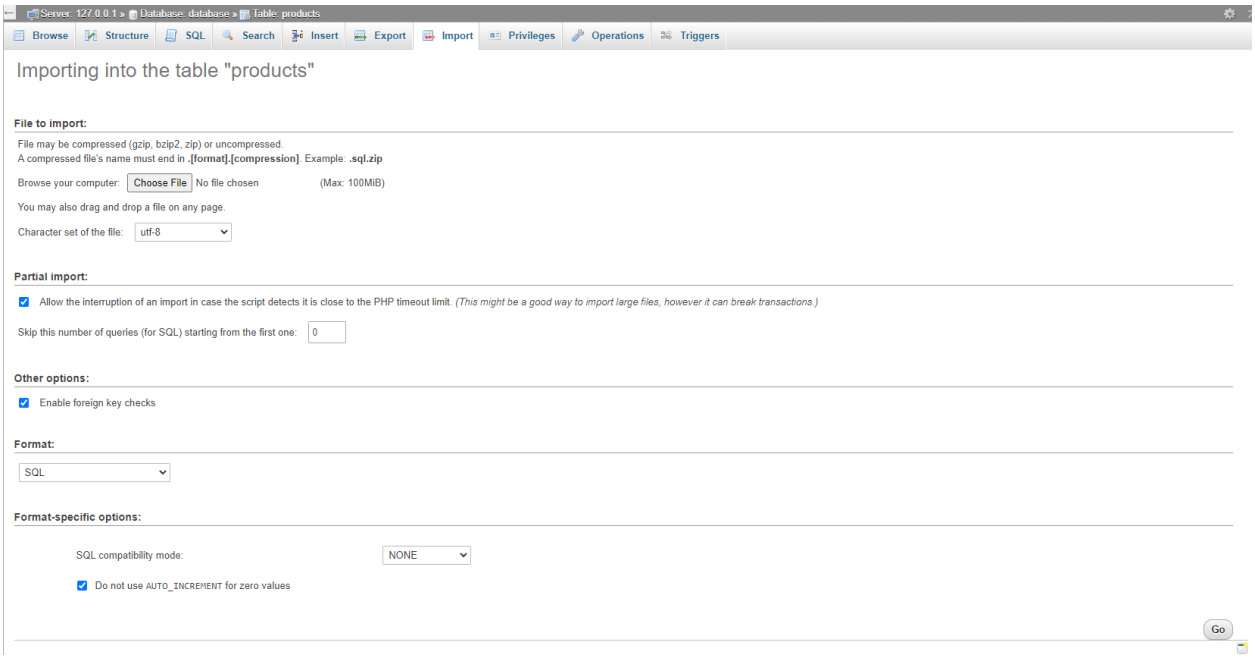
3. Η κάθε τηλεόραση μπορεί να έχει κανένα, ένα ή περισσότερα από τα ακόλουθα:
    - Ανάλυση οθόνης (“Resolution”)
    - Τύπος τηλεόρασης (“Type”)
    - Έτος κυκλοφορίας (“ReleaseDate”)

Τα διαθέσιμα έτη κυκλοφορίας των τηλεοράσεων είναι όμοια με αυτά των κινητών και επιπλέον:

Type
Smart

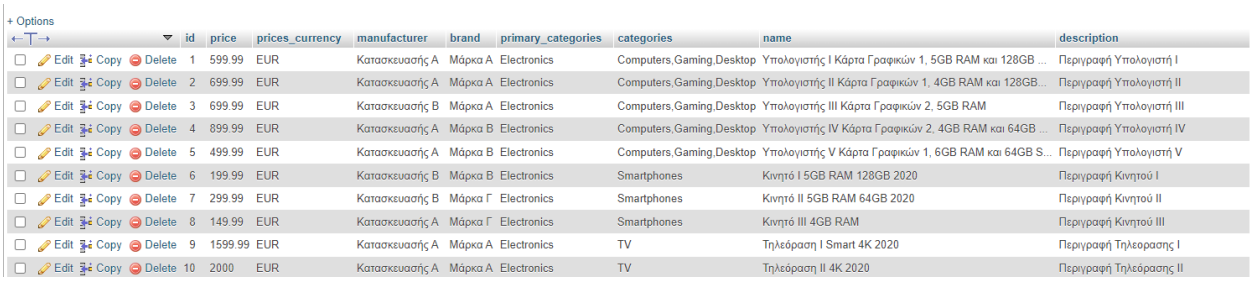
Resolution
4K

Με βάση αυτά τα χαρακτηριστικά, θα εισάγουμε τα προϊόντα στον πίνακα “products”. Αυτό μπορεί να γίνει είτε με το script «edit_and_import.py» είτε απ’ ευθείας από τη σελίδα phpMyAdmin με την επιλογή “Import” (Εικόνα 42), ακόμα και χειροκίνητα εκτελώντας εντολές SQL ή με την επιλογή “Insert”.



Εικόνα 42

Μετά την εισαγωγή τυχαίων προϊόντων, ο πίνακας “products” διαμορφώνεται ως (Εικόνα 43):

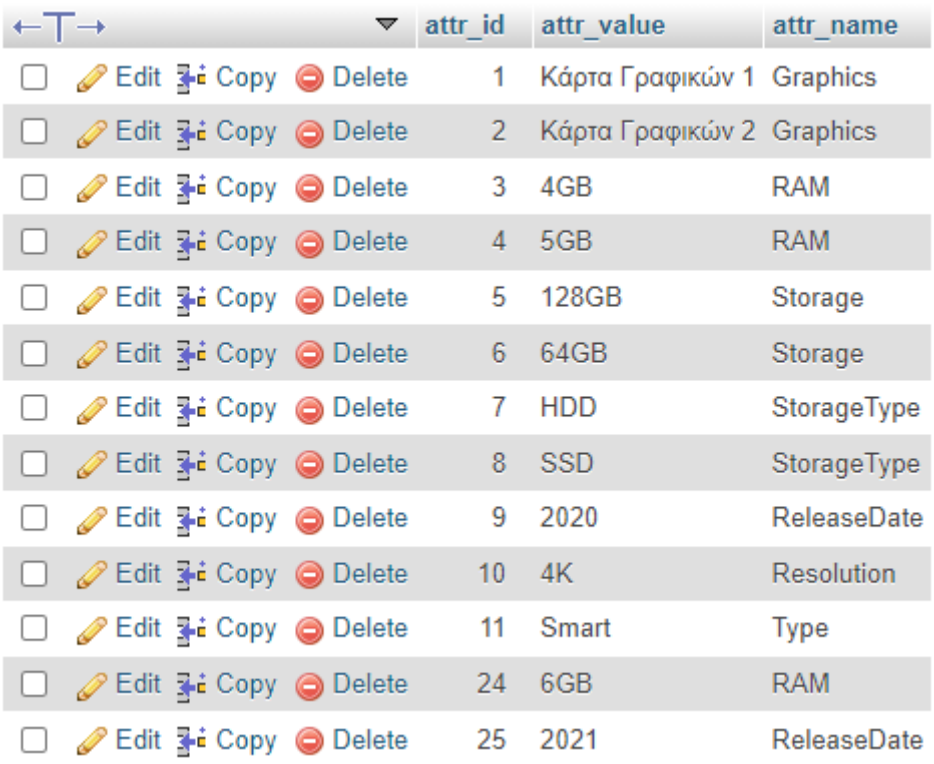


+ Options

	id	price	prices_currency	manufacturer	brand	primary_categories	categories	name	description
Edit Copy Delete	1	599.99	EUR	Κατασκευαστής Α	Μάρκα Α	Electronics	Computers,Gaming,Desktop	Υπολογιστής Ι Κάρτα Γραφικών 1, 5GB RAM και 128GB ...	Περιγραφή Υπολογιστή Ι
Edit Copy Delete	2	699.99	EUR	Κατασκευαστής Α	Μάρκα Α	Electronics	Computers,Gaming,Desktop	Υπολογιστής ΙΙ Κάρτα Γραφικών 1, 4GB RAM και 128GB...	Περιγραφή Υπολογιστή ΙΙ
Edit Copy Delete	3	699.99	EUR	Κατασκευαστής Β	Μάρκα Α	Electronics	Computers,Gaming,Desktop	Υπολογιστής ΙΙΙ Κάρτα Γραφικών 2, 5GB RAM	Περιγραφή Υπολογιστή ΙΙΙ
Edit Copy Delete	4	899.99	EUR	Κατασκευαστής Α	Μάρκα Β	Electronics	Computers,Gaming,Desktop	Υπολογιστής IV Κάρτα Γραφικών 2, 4GB RAM και 64GB ...	Περιγραφή Υπολογιστή IV
Edit Copy Delete	5	499.99	EUR	Κατασκευαστής Α	Μάρκα Β	Electronics	Computers,Gaming,Desktop	Υπολογιστής V Κάρτα Γραφικών 1, 6GB RAM και 64GB S...	Περιγραφή Υπολογιστή V
Edit Copy Delete	6	199.99	EUR	Κατασκευαστής Β	Μάρκα Β	Electronics	Smartphones	Κινητό Ι 5GB RAM 128GB 2020	Περιγραφή Κινητού Ι
Edit Copy Delete	7	299.99	EUR	Κατασκευαστής Β	Μάρκα Γ	Electronics	Smartphones	Κινητό ΙΙ 5GB RAM 64GB 2020	Περιγραφή Κινητού ΙΙ
Edit Copy Delete	8	149.99	EUR	Κατασκευαστής Α	Μάρκα Γ	Electronics	Smartphones	Κινητό ΙΙΙ 4GB RAM	Περιγραφή Κινητού ΙΙΙ
Edit Copy Delete	9	1599.99	EUR	Κατασκευαστής Α	Μάρκα Α	Electronics	TV	Τηλεόραση Ι Smart 4K 2020	Περιγραφή Τηλεόρασης Ι
Edit Copy Delete	10	2000	EUR	Κατασκευαστής Α	Μάρκα Α	Electronics	TV	Τηλεόραση ΙΙ 4K 2020	Περιγραφή Τηλεόρασης ΙΙ

Εικόνα 43

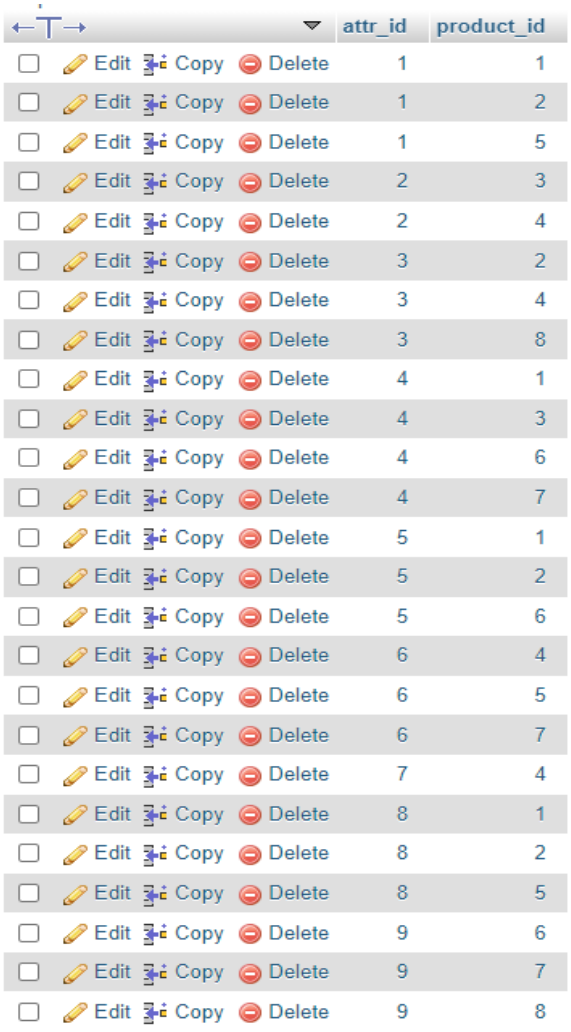
Παρομοίως εισάγουμε όλα τα χαρακτηριστικά και τις τιμές τους στον πίνακα “attributes” και διαμορφώνεται ως (Εικόνα 44):



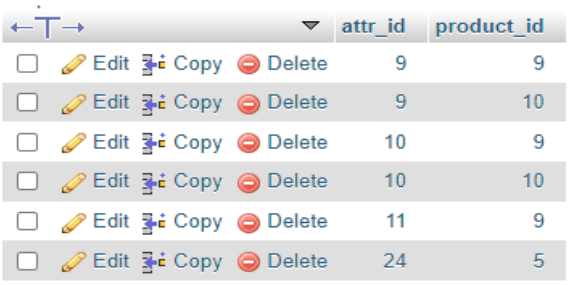
	attr_id	attr_value	attr_name
Edit Copy Delete	1	Κάρτα Γραφικών 1	Graphics
Edit Copy Delete	2	Κάρτα Γραφικών 2	Graphics
Edit Copy Delete	3	4GB	RAM
Edit Copy Delete	4	5GB	RAM
Edit Copy Delete	5	128GB	Storage
Edit Copy Delete	6	64GB	Storage
Edit Copy Delete	7	HDD	StorageType
Edit Copy Delete	8	SSD	StorageType
Edit Copy Delete	9	2020	ReleaseDate
Edit Copy Delete	10	4K	Resolution
Edit Copy Delete	11	Smart	Type
Edit Copy Delete	24	6GB	RAM
Edit Copy Delete	25	2021	ReleaseDate

Εικόνα 44

Τέλος, αναθέτουμε αυτά τα χαρακτηριστικά σε κάθε προϊόν και ο πίνακας “products_attributes” γίνεται (Εικόνα 45):



	attr_id	product_id
Edit Copy Delete	1	1
Edit Copy Delete	1	2
Edit Copy Delete	1	5
Edit Copy Delete	2	3
Edit Copy Delete	2	4
Edit Copy Delete	3	2
Edit Copy Delete	3	4
Edit Copy Delete	3	8
Edit Copy Delete	4	1
Edit Copy Delete	4	3
Edit Copy Delete	4	6
Edit Copy Delete	4	7
Edit Copy Delete	5	1
Edit Copy Delete	5	2
Edit Copy Delete	5	6
Edit Copy Delete	6	4
Edit Copy Delete	6	5
Edit Copy Delete	6	7
Edit Copy Delete	7	4
Edit Copy Delete	8	1
Edit Copy Delete	8	2
Edit Copy Delete	8	5
Edit Copy Delete	9	6
Edit Copy Delete	9	7
Edit Copy Delete	9	8



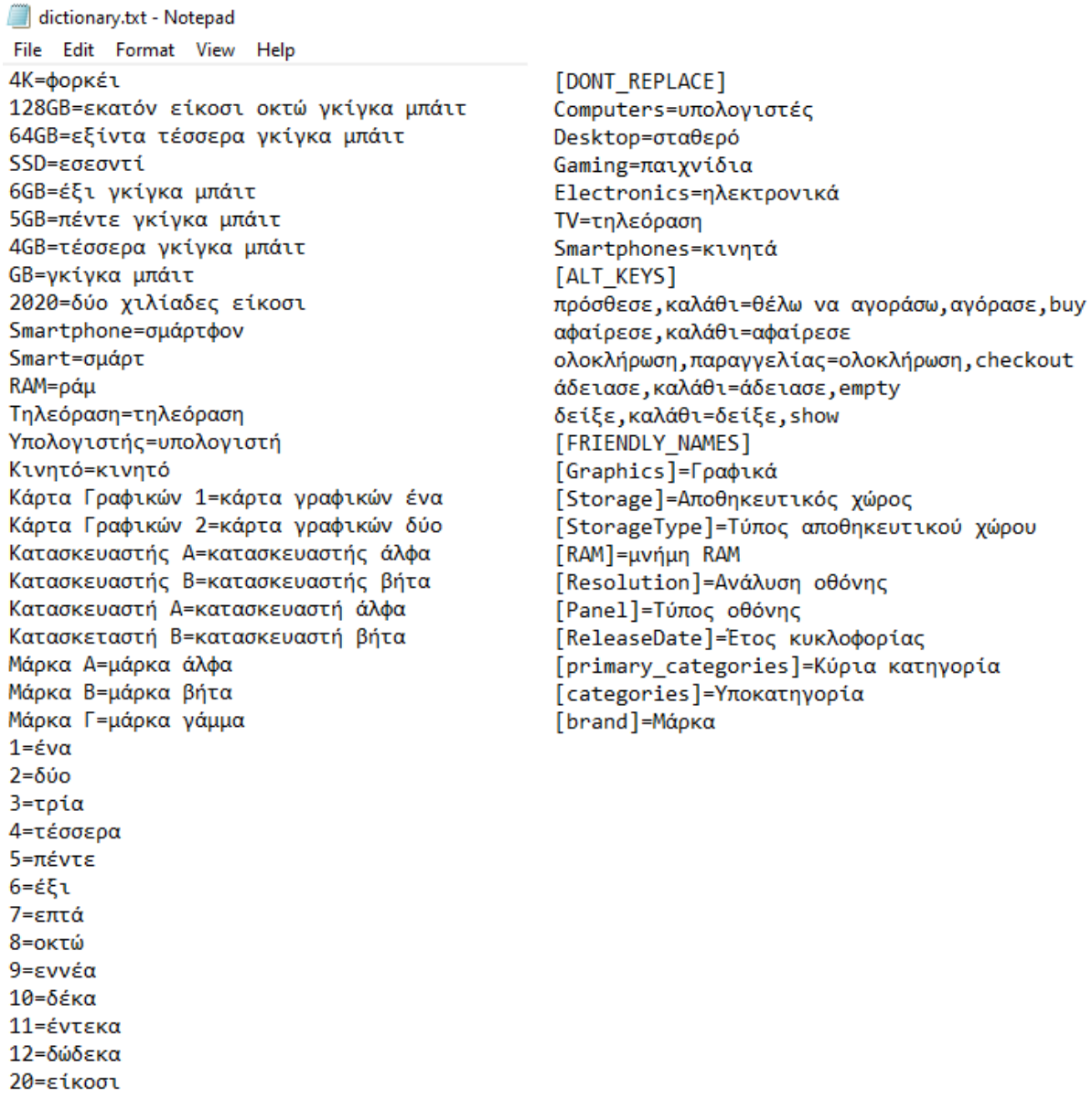
	attr_id	product_id
Edit Copy Delete	9	9
Edit Copy Delete	9	10
Edit Copy Delete	10	9
Edit Copy Delete	10	10
Edit Copy Delete	11	9
Edit Copy Delete	24	5

Εικόνα 45

Επιπλέον, θα μπορούσαν να εισαχθούν στοιχεία ήδη υπάρχοντών πελατών στον πίνακα “users” αλλά δεν είναι απαραίτητο.

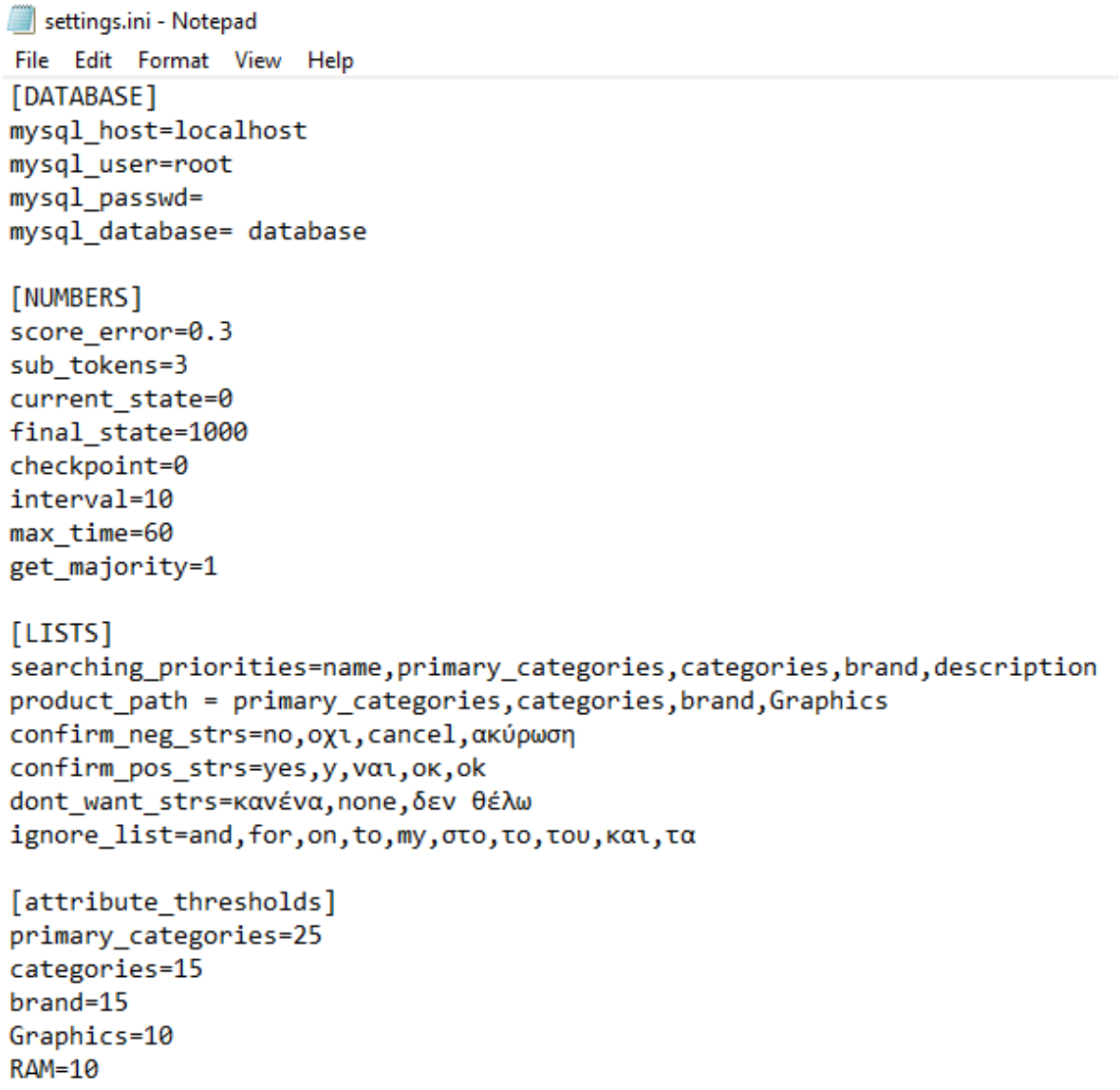
6.2 Λεξικό και Ρυθμίσεις

Το πρώτο κομμάτι του λεξικού αφορά τη μετατροπή των λέξεων που αναγνωρίζει το μοντέλο αναγνώρισης ομιλίας σε λέξεις που πιθανώς υπάρχουν στη βάση δεδομένων. Το δεύτερο αφορά τις λέξεις που θα αντικατασταθούν στην αναζήτηση αλλά όχι στην εντολή του πελάτη, η τρίτη τις εναλλακτικές λέξεις για τα βασικά κλειδιά που ψάχνουμε στην εντολή και το τέταρτο τις φιλικές ονομασίες για τα χαρακτηριστικά των προϊόντων που αναφέρθηκαν και στο 6.1. Επομένως το λεξικό αποτελείται από τους ακόλουθους κανόνες (Εικόνα 46):



Εικόνα 46

Οι ρυθμίσεις (Εικόνα 47) αποτελούνται και αυτές από τέσσερα κομμάτια, το πρώτο περιέχει τα στοιχεία της βάσης δεδομένων στην οποία συνδεόμαστε, το δεύτερο διάφορες αριθμητικές τιμές οι οποίες αναλύονται στο Κεφάλαιο 5, το τρίτο περιέχει τιμές που επεξεργάζονται ως λίστες και το τελευταίο τα κατώτατα όρια των χαρακτηριστικών όπως αναλύονται στο υποκεφάλαιο 5.3.6 (μέθοδος `get_majority_results`).



Εικόνα 47

Συγκεκριμένα, η βάση δεδομένων βρίσκεται στη διεύθυνση “localhost”, τα στοιχεία σύνδεσης είναι τα προεπιλεγμένα (“root” και χωρίς κωδικό) και το όνομα της βάσης είναι “database”. Θα θεωρούμε αποδεκτά αποτελέσματα όσα προϊόντα δεν διαφέρουν περισσότερο από (score_error) 0.3 από την εντολή του πελάτη της οποίας κάθε λέξη χωρίζεται σε (sub_tokens) υπό λέξεις με 3 χαρακτήρες. Επίσης ξεκινάμε από την κατάσταση 0 (current_state) και η εφαρμογή σταματά όταν (και αν) φτάσουμε στη κατάσταση 1000 (final_state), με αρχικό checkpoint τη 0. Από προεπιλογή, η συχνότητα των επαναλαμβανόμενων μηνυμάτων είναι 10 δευτερόλεπτα (interval) με μέγιστο χρόνο τα 60 δευτερόλεπτα (max_time). Επίσης δηλώνουμε (get_majority=1) ότι θέλουμε να χρησιμοποιηθεί η μέθοδος `get_majority_results` στην αναζήτηση προϊόντων.

Επιπλέον, οι στήλες στις οποίες γίνεται η αρχική αναζήτηση των προϊόντων καθώς και το αντίστοιχο βάρος κάθε μιας δηλώνονται από την μεταβλητή `searching_priorities`. Οι στήλες στις οποίες θα εκτελεστεί η `get_majority_results` δηλώνονται από τη μεταβλητή `product_path`. Οι λέξεις (χωρίζονται με κόμμα) που δηλώνουν ότι ο χρήστης απαντάει αρνητικά σε μια ερώτηση δίνονται από τη `confirm_neg_strs`, όταν απαντάει

θετικά από τη `confirm_pos_strs` και όταν δεν θέλει να συνεχίσει με μια ενέργεια από τη `dont_want_strs`. Επίσης στην διαδικασία της αναζήτησης προϊόντων από τη βάση δεδομένων αγνοούνται οι λέξεις (χωρίζονται με κόμμα) της μεταβλητής `ignore_list`.

Τέλος, τα κατώτατα όρια που χρησιμοποιούνται στη μέθοδο `get_majority_results` δηλώνονται στο κομμάτι [attribute_thresholds].

6.3 Δημιουργία Καταστάσεων

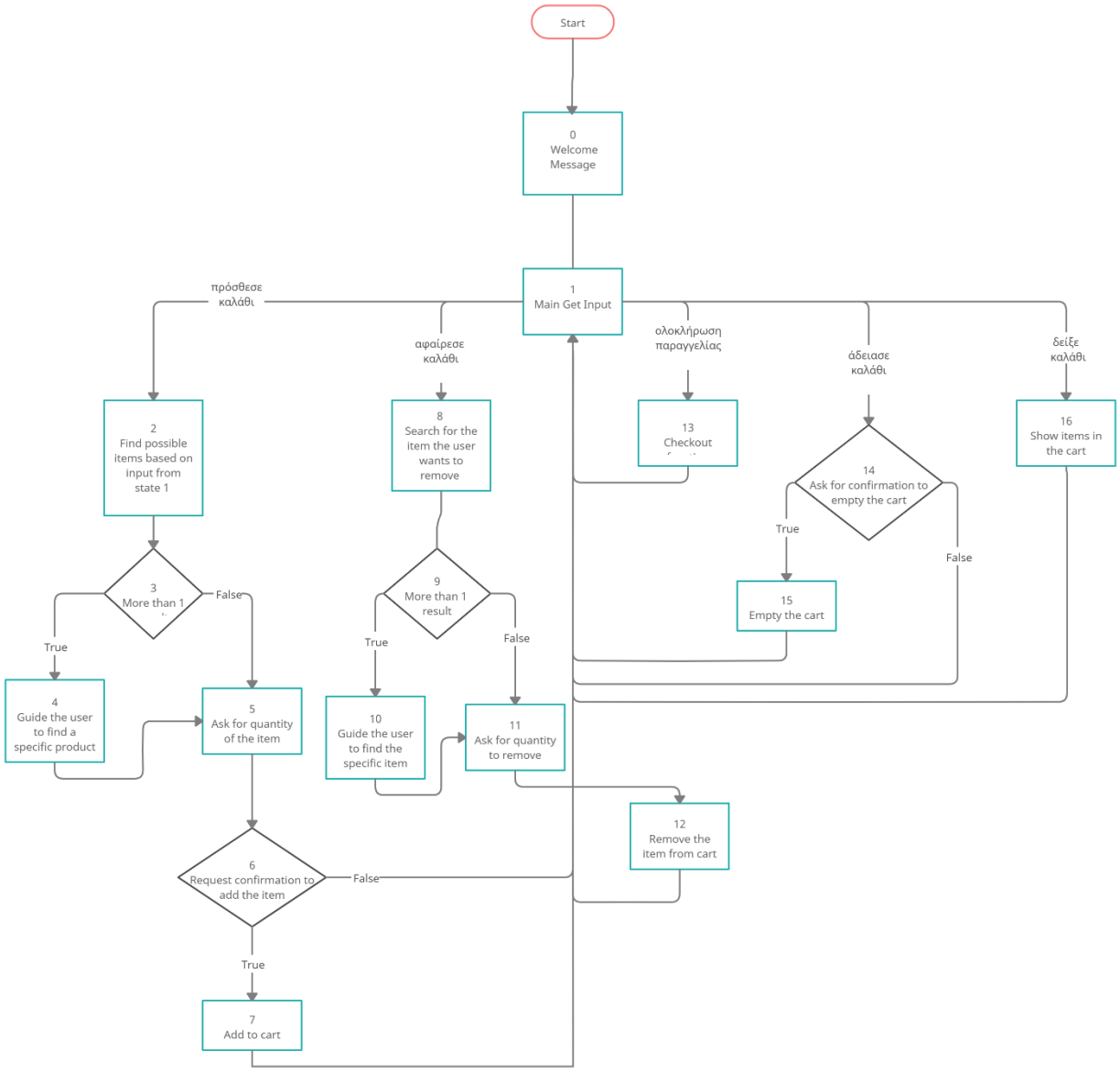
Για τη δημιουργία των καταστάσεων, μπορούμε να οπτικοποιήσουμε τη ροή της εφαρμογής με ένα διάγραμμα ροής όπως φαίνεται στην Εικόνα 48. Αρχικά στέλνεται ένα μήνυμα καλωσορίσματος στον πελάτη (0) και έπειτα αναμένεται να στείλει εντολή που περιέχει μια από τις πέντε λίστες κλειδιών (1).

Όταν θέλει να προσθέσει κάτι στο καλάθι του, γίνεται μετάβαση στην κατάσταση 2 όπου και γίνεται η αναζήτηση στη βάση δεδομένων. Αν τα αποτελέσματα είναι περισσότερα από ένα (3) γίνεται μετάβαση στην 4 όπου και καθοδηγείται ο πελάτης για να διαλέξει ένα συγκεκριμένο προϊόν και αμέσως γίνεται μετάβαση στην 5 που τον ρωτάει τι ποσότητα αυτού του προϊόντων θέλει να προσθέσει. Αν το αποτέλεσμα είναι μόνο ένα (3), γίνεται μετάβαση κατευθείαν στην 5. Έπειτα, ζητείται επιβεβαίωση (6) για να προσθέσει την δοσμένη ποσότητα ενός προϊόντος στο καλάθι και αν δοθεί έγκριση αμέσως προστίθεται (7) και γίνεται μετάβαση στην κατάσταση 1. Αν δεν δοθεί έγκριση γίνεται απ' ευθείας μετάβαση στην 1 χωρίς να προστεθεί τίποτα στο καλάθι.

Η διαδικασία για την αφαίρεση ενός προϊόντος είναι η ίδια, αλλά αλλάζουν τα μηνύματα που του στέλνονται (10, 11) και δεν ζητείται επιβεβαίωση για την διαγραφή. Μετά την διαγραφή γίνεται επιστροφή στην κατάσταση 1.

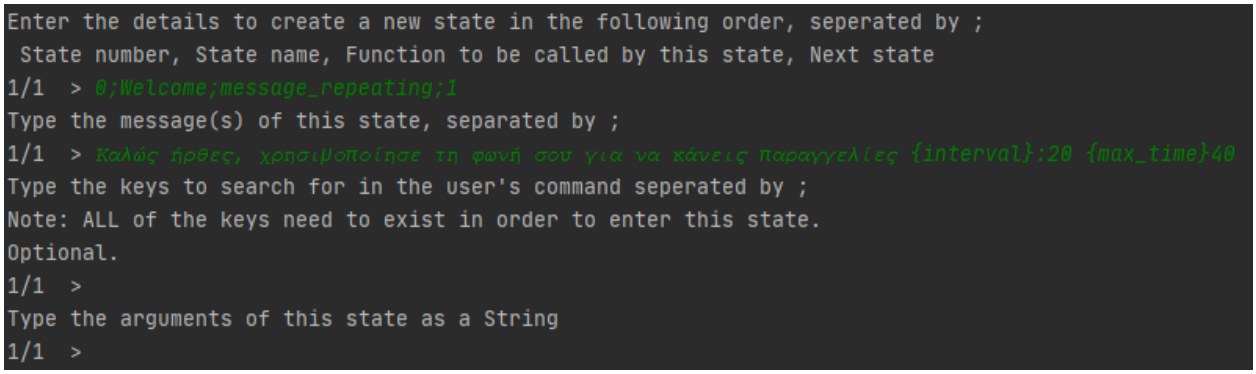
Όταν θέλει να ολοκληρώσει την παραγγελία του, γίνεται μετάβαση στην 13 όπου και καλείται η μέθοδος για την ολοκλήρωση. Παρομοίως όταν επιθυμεί να δει ή ακούσει τα αντικείμενα του καλαθιού γίνεται μετάβαση στην 16. Και στις δύο περιπτώσεις η εφαρμογή επιστρέφει στην κατάσταση 1 μετά το πέρας των μεθόδων των 13 και 16.

Τέλος, όταν θέλει να αδειάσει το καλάθι του, ζητείται επιβεβαίωση από την κατάσταση 14 και αν είναι θετική τότε αδειάζει (15) και γίνεται επιστροφή στην 1, αλλιώς δεν αδειάζει και πάλι επιστρέφει στην 1.



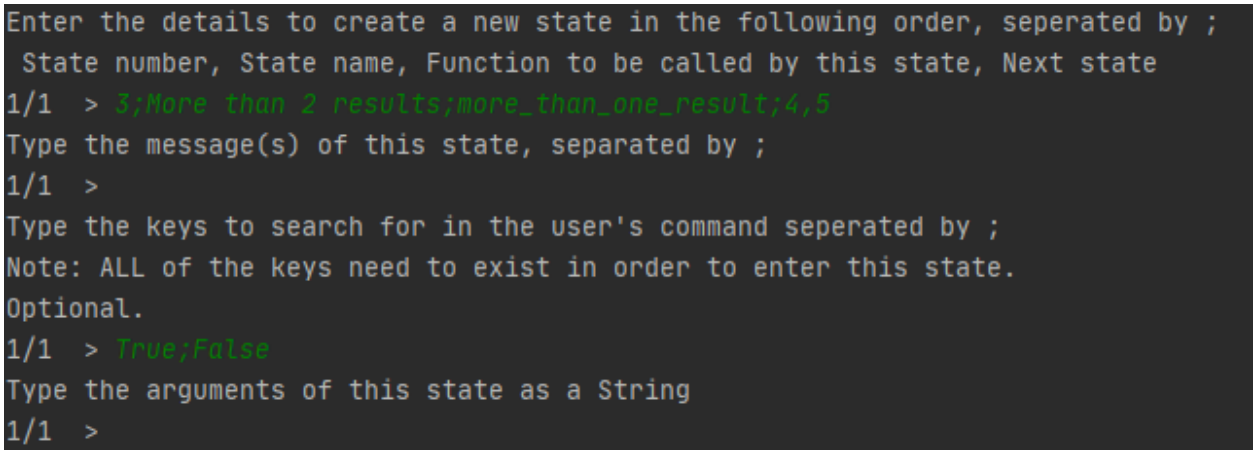
Εικόνα 48

Ενδεικτικά, στις Εικόνες Εικόνα 49, Εικόνα 50, Εικόνα 51, Εικόνα 52, Εικόνα 53, Εικόνα 54 φαίνεται η δημιουργία των καταστάσεων 0, 1, 3, 4, 5, 8 αντίστοιχα χρησιμοποιώντας το πρόγραμμα edit_and_import.py και έπειτα αποθηκεύονται όλες στη βάση δεδομένων (Εικόνα 55)

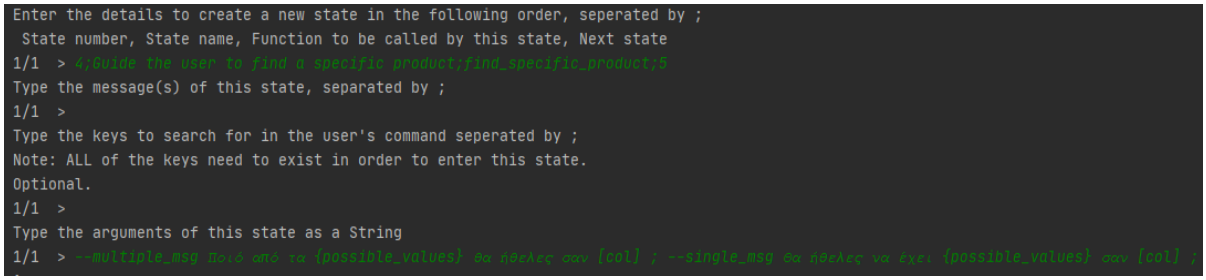


Εικόνα 49

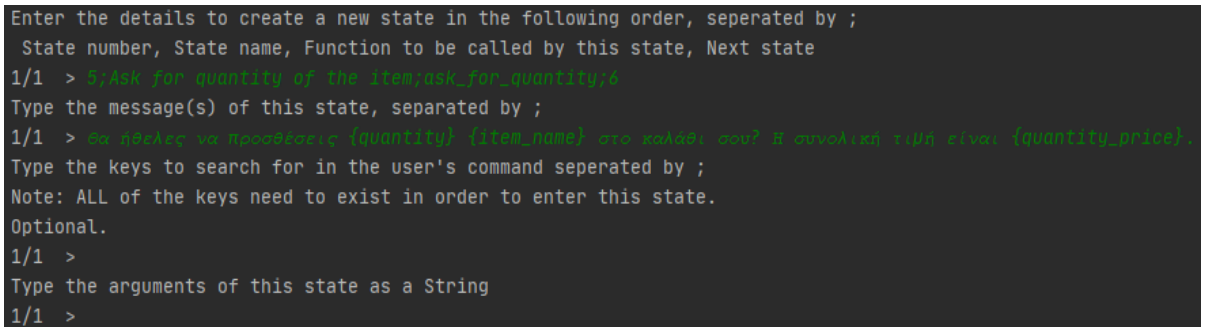
Εικόνα 50



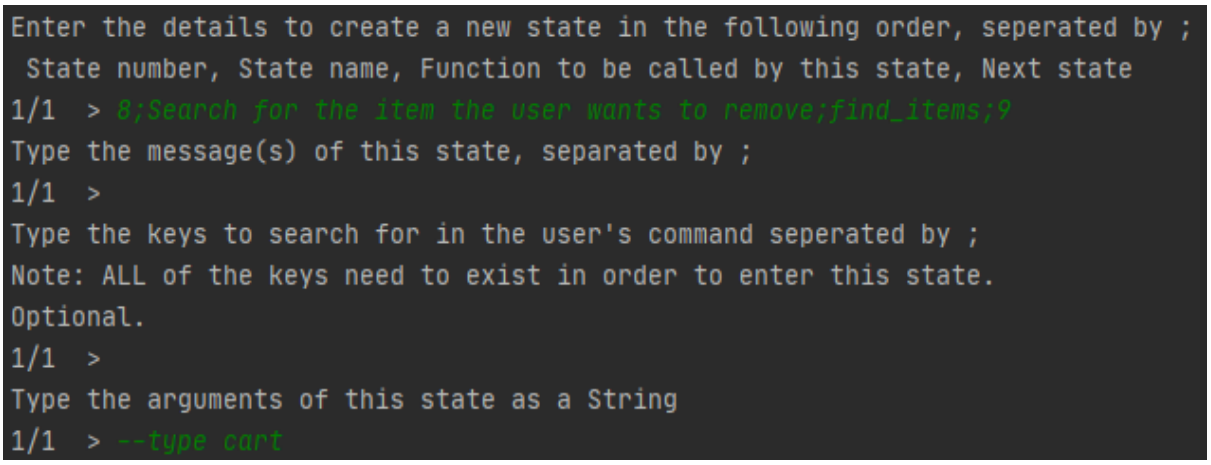
Εικόνα 51



Εικόνα 52



Εικόνα 53



Εικόνα 54

	state	name	function_name	next_state	messages	keys	arguments
Edit Copy Delete	0	Welcome	message_repeating	1	["Καλώς ήρθες, χρησιμοποίησε τη φωνή σου για να κά...	[]	
Edit Copy Delete	10	Guide the user to find the specific item	find_specific_product	11	[]	[]	
Edit Copy Delete	1	Main Input	get_input	2,8,13,14,16	[]	[["πρόσθεσε", "καλάθι"], ["αφαίρεσε", "καλάθι"], [...	
Edit Copy Delete	11	Ask for quantity to remove	ask_for_quantity	12	["Τι ποσότητα {item_name} θα ήθελες να αφαιρέσεις;...	[]	--message Τι ποσότητα {item_name} θα ήθελες να αφα...
Edit Copy Delete	12	Remove the item from cart	remove_from_cart	1	[]	[]	
Edit Copy Delete	13	Checkout function	checkout	1	[]	[]	
Edit Copy Delete	14	Ask for confirmation to empty the cart	confirmation	15,1	["Θέλεις να αφαιρέσεις όλα τα προιόντα απο το καλά...	[["True"], ["False"]]	
Edit Copy Delete	15	Empty the cart	empty_cart	1	[]	[]	
Edit Copy Delete	16	Show items in the cart	print_cart	1	[]	[]	
Edit Copy Delete	2	Find possible items based on input from state 1	find_items	3	[]	[]	
Edit Copy Delete	3	More than 2 results	more_than_one_result	4,5	[]	[["True"], ["False"]]	
Edit Copy Delete	4	Guide the user to find a specific product	find_specific_product	5	[]	[]	--multiple_msg Ποιό από τα {possible_values} θα ήθ...
Edit Copy Delete	5	Ask for quantity of the item	ask_for_quantity	6	["Τι ποσότητα {item_name} θα ήθελες;", "Τι ποσότητ...	[]	
Edit Copy Delete	6	Request confirmation to add the item	confirmation	7,1	["Θα ήθελες να προσθέσεις {quantity} {item_name} σ...	[["True"], ["False"]]	
Edit Copy Delete	7	Add to cart	add_to_cart	1	[]	[]	
Edit Copy Delete	8	Search for the item the user wants to remove	find_items	9	[]	[]	--type cart
Edit Copy Delete	9	More than 2 results	more_than_one_result	10,11	[]	[["True"], ["False"]]	

Εικόνα 55

6.4 Παράδειγμα Αγοράς Προϊόντος

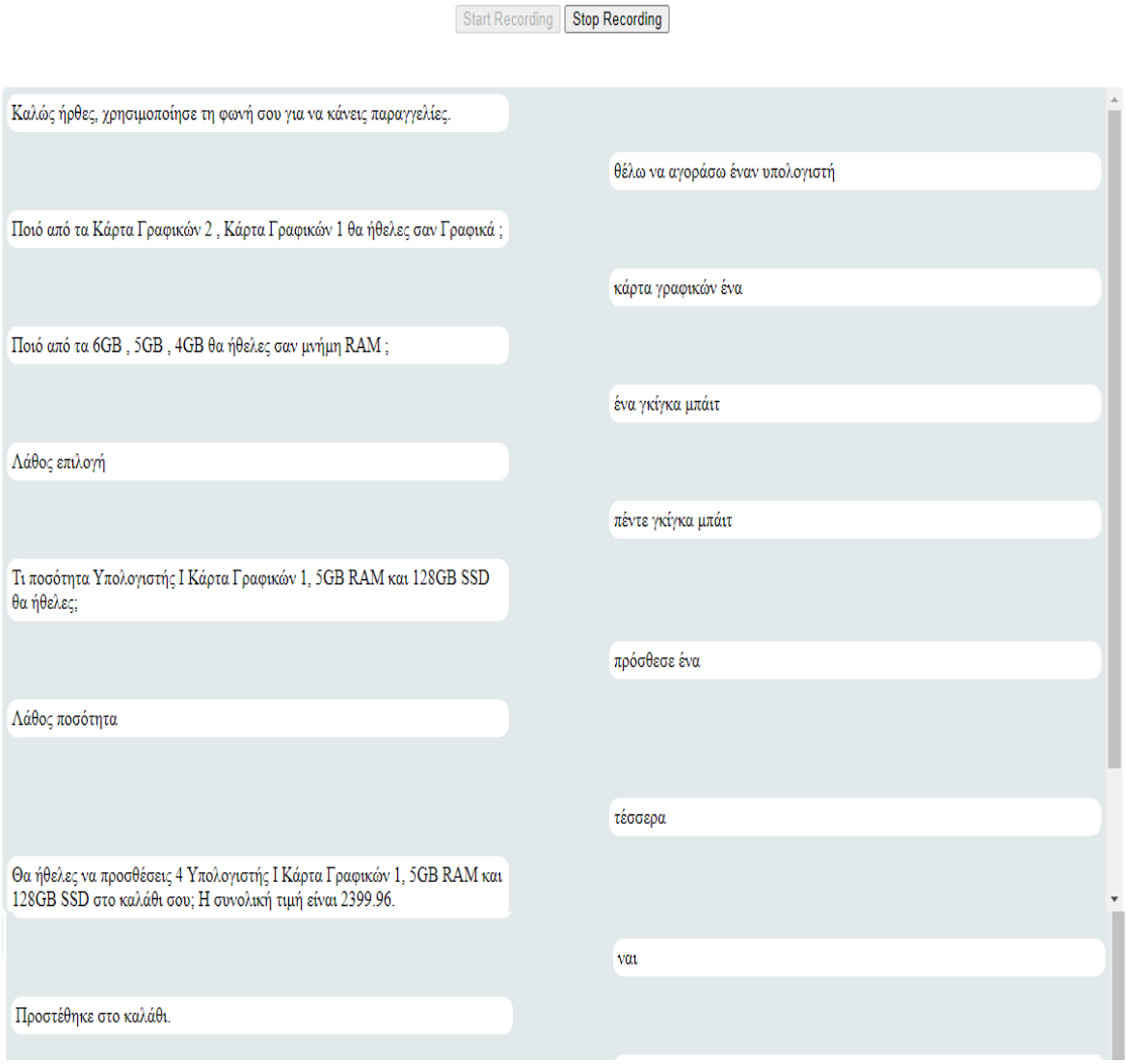
Ας δούμε τώρα ένα παράδειγμα για την αγορά ενός προϊόντος . Ξεκινάμε με την είσοδο στη σελίδα “localhost:8888” και πατώντας το κουμπί “Start Recording” για να δώσουμε φωνητικές εντολές. Η πρώτη εντολή που δίνουμε είναι η «Θέλω να αγοράσω έναν υπολογιστή» η οποία μεταφράζεται σωστά, μια γενική εντολή που δεν προσδιορίζει ειδικά χαρακτηριστικά ενός υπολογιστή και επομένως υποψήφια προϊόντα είναι και οι πέντε υπολογιστές που υπάρχουν στη βάση δεδομένων.

Για αυτό, το πρόγραμμα μας ρωτάει ποια τιμή του ειδικού χαρακτηριστικού «Γραφικά» θα θέλαμε. Το μήνυμα αυτό αποστέλλεται από τη μέθοδο find_specific_product, και αρχικά είναι «Ποιό από τα Κάρτα Γραφικών 2 , Κάρτα Γραφικών 1 θα ήθελες σαν [Graphics] ;» αλλά η μέθοδος message_once αντικαθιστά τη λέξη [Graphics] με τη φιλική ονομασία «Γραφικά» όπως δηλώνεται στους κανόνες του λεξικού. Έπειτα λέμε ότι θέλουμε τη τιμή «κάρτα γραφικών ένα» η οποία μεταφράζεται ως «Κάρτα Γραφικών 1» πάλι λόγω των κανόνων του λεξικού.

Έπειτα το πρόγραμμα ρωτάει για ακόμα ένα χαρακτηριστικό (RAM) όπου δίνουμε την εντολή «πέντε γκίγκα μπάιτ», αλλά την πρώτη φορά δεν την αναγνωρίζει επιτυχώς και επομένως την ξαναδίνουμε. Στη συνέχεια ρωτάει την ποσότητα του προϊόντος που θα θέλαμε, να σημειωθεί ότι το πλήρες όνομα του προϊόντος προέρχεται από τη στήλη «name» της βάσης δεδομένων. Απαντάμε με την εντολή «τέσσερα» η οποία αρχικά δεν αναγνωρίζεται σωστά.

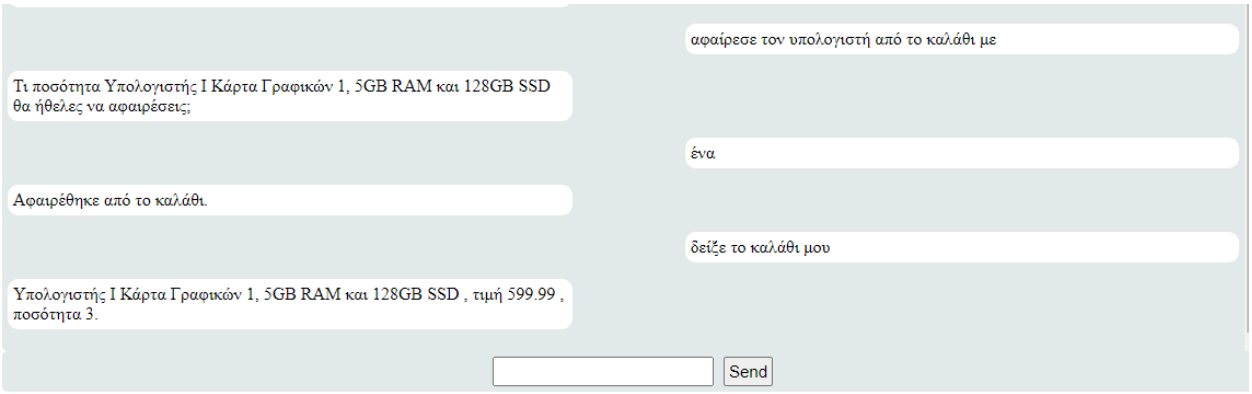
Μετά την ερώτηση επιβεβαίωσης για την προσθήκη του προϊόντος και τη θετική απάντηση γίνεται η προσθήκη στο καλάθι.

Η συζήτηση μέχρι τώρα φαίνεται στην Εικόνα 56



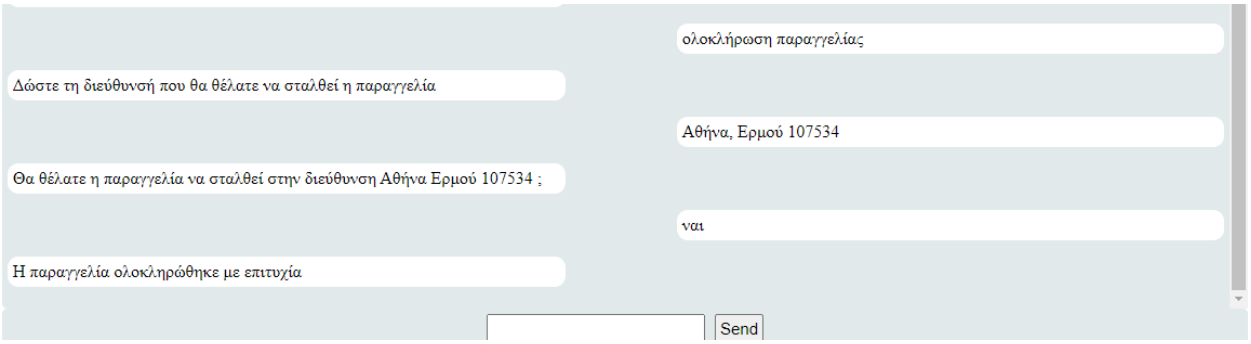
Εικόνα 56

Έπειτα (Εικόνα 57) δίνουμε την εντολή «αφαίρεσε τον υπολογιστή από το καλάθι μου» και αφαιρούμε τον ένα από τους τέσσερις υπολογιστές που προσθέσαμε πριν και ζητάμε να μας δείξει το πρόγραμμα το καλάθι στο οποίο μέχρι στιγμής έχουμε συνολικά τρις υπολογιστές.



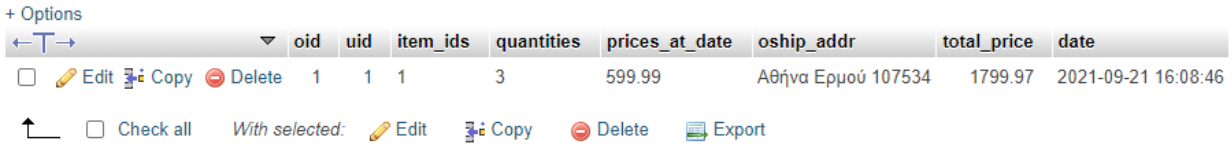
Εικόνα 57

Τέλος ολοκληρώνουμε τη παραγγελία με την ανάλογη εντολή και δίνουμε μια διεύθυνση αποστολής της στην Εικόνα 58



Εικόνα 58

και βλέπουμε στη βάση δεδομένων ότι όντως έχει ολοκληρωθεί η παραγγελία του χρήστη με id 1 (uid).



ΚΕΦΑΛΑΙΟ 7 Συμπεράσματα

Κατά τη διάρκεια αυτής της πτυχιακής εργασίας αναπτύχθηκε μια εφαρμογή που είναι ικανή να δέχεται φωνητικές και γραπτές εντολές και να βοηθά έναν καταναλωτή να πραγματοποιεί παραγγελίες με ευκολία, χωρίς απαραίτητα να κοιτάει την οθόνη του. Για την επίτευξη αυτού, δημιουργήθηκε μια ιστοσελίδα η οποία δέχεται τα αιτήματα του καταναλωτή είτε αυτά είναι ηχητική ροή του μικροφώνου που επεξεργάζεται και τροφοδοτεί σε ένα μοντέλο αναγνώρισης ομιλίας DeepSpeech, που με τη σειρά του μετατρέπει την ομιλία σε κείμενο, είτε εντολές σε μορφή γραπτών μηνυμάτων. Έπειτα καλεί την εφαρμογή επεξεργασίας εντολών η οποία είναι υπεύθυνη για την αναγνώριση και εκτέλεση των εντολών που έδωσε ο καταναλωτής αλλά και την επικοινωνία μαζί του όταν είναι απαραίτητο.

Μια βάση δεδομένων αξιοποιείται από το πρόγραμμα επεξεργασίας εντολών η οποία αποθηκεύει όλα τα προϊόντα που μπορεί να έχει διαθέσιμα ένα κατάστημα, καθώς και τα στοιχεία των καταναλωτών και τις παραγγελίες που αυτοί πραγματοποιούν.

Στην διαδικασία εκπαίδευσης ενός μοντέλου DeepSpeech αναπτύχθηκε ένα απλό πρόγραμμα που μπορεί να βοηθήσει σε αυτή, αυτοματοποιώντας τη δημιουργία τριών αρχείων που απαιτούνται για τη διαδικασία.

Επιπλέον, αναπτύχθηκε ένα πρόγραμμα που βοηθά τον χρήστη στη δημιουργία καταστάσεων για την εφαρμογή και την αποθήκευση αυτών είτε σε αρχείο .csv είτε κατευθείαν στη βάση δεδομένων. Επίσης μπορεί να αποθηκεύσει αρχεία .csv απ’ ευθείας σε έναν πίνακα της βάσης δεδομένων.

7.1 Μελλοντικές Επεκτάσεις

Οι επεκτάσεις που μπορούν να γίνουν στο μέλλον για μια τέτοια εφαρμογή είναι σχεδόν ατελείωτες όχι μόνο για την λειτουργική βελτίωση αυτής αλλά και την ακολουθία των τάσεων που συνέχεια αλλάζουν. Εφόσον υπάρχουν τρία μέρη της εφαρμογής, οφείλουμε να αναφέρουμε ιδέες περαιτέρω επέκτασης κάθε μέρους.

Αρχικά, το πρώτο μέρος όντας ο τρόπος επικοινωνίας του καταναλωτή με την εφαρμογή επεξεργασίας εντολών (και αντίστροφα) μπορεί να επεκταθεί σε άλλο μέσο πέρα από ιστοσελίδα όπως θα ήταν οι τηλεφωνικές κλήσεις και SMS. Η εφαρμογή ήδη υποστηρίζει την ταυτοποίηση των καταναλωτών με τα τηλέφωνά τους, επομένως αυτό που πρέπει να γίνει είναι να αναπτυχθεί ένα πρόγραμμα που με χρήση κάποιας υπηρεσίας όπως το Twilio δέχεται τηλεφωνικές κλήσεις, επεξεργάζεται την εισερχόμενη ροή ήχου και την τροφοδοτεί πάλι σε ένα μοντέλο αναγνώρισης ομιλίας και απαντά με text-to-speech όταν η εφαρμογή στέλνει κάποιο μήνυμα στον καταναλωτή. Έτσι ουσιαστικά αφού το τρίτο μέρος (Κεφάλαιο 5) παραμένει ίδιο, έχει προστεθεί στη συνολική εφαρμογή η δυνατότητα επικοινωνίας μέσω τηλεφωνικών κλήσεων.

Ένας ακόμη τρόπος επέκτασης του πρώτου μέρους είναι η δυνατότητα ενσωμάτωσής του σε ήδη υπάρχουσες ιστοσελίδες καταστημάτων έτσι ώστε να μην είναι από μόνο του μια ιστοσελίδα, αλλά μια λειτουργία εντός κάποιας άλλης. Θα μπορούσε για παράδειγμα να γίνει ένα μικρό παραθυράκι που θα φαίνεται στη γωνία μιας ιστοσελίδας ή να είναι προσβάσιμο με το πάτημα ενός κουμπιού.

Στο δεύτερο μέρος που αφορά την αναγνώριση ομιλίας η προφανής μελλοντική δουλεία σε είναι η βελτίωση του ακουστικού και γλωσσικού μοντέλου, ώστε να είναι ικανό να αναγνωρίζει ένα μεγάλο πλήθος λέξεων κάνοντάς το πιο «ελκυστικό» σε πραγματικά καταστήματα και επιχειρήσεις αλλά και βελτιώνοντας δραματικά την εμπειρία των καταναλωτών όταν δίνουν φωνητικές εντολές. Αυτό γίνεται με τη μαζική συλλογή δεδομένων τόσο ηχογραφήσεων για την εκπαίδευση του ακουστικού μοντέλου όσο και λεξιλογίου για την επέκταση του γλωσσικού μοντέλου.

Μια ακόμη ιδέα για αυτό, είναι η δυναμική και αυτόματη πρόσθεση των λέξεων μιας βάσης δεδομένων στο γλωσσικό μοντέλο, επεκτείνοντάς έτσι το λεξιλόγιο για όλα τα προϊόντα που παρέχει μια επιχείρηση και πιθανώς αυτά να μην υπάρχουν ήδη στο μοντέλο.

Το τρίτο μέρος είναι ίσως αυτό με τη μεγαλύτερη επεκτασιμότητα. Προσθήκες και βελτιώσεις μπορούν να γίνουν σε πολλαπλά μέτωπα, τόσο για να βελτιώσουν την εμπειρία του καταναλωτή όσο και τη λειτουργικότητα της ίδιας της εφαρμογής. Μερικές ιδέες αναφέρονται παρακάτω:

1. Δημιουργία συστήματος προσωπικού βοηθού που θα μελετά τις δραστηριότητες ενός καταναλωτή όπως τα προϊόντα που αγοράζει, τον τρόπο αλληλεπίδρασης με την εφαρμογή, τις φωνητικές εντολές που δίνει και όλα αυτά για να μπορεί να του προσφέρει μια εξατομικευμένη εμπειρία μοναδική για αυτόν. Για παράδειγμα μπορεί να του προτείνει προϊόντα που ίσως να τον ενδιαφέρουν και να αποφεύγει να του παρουσιάζει άλλα που ίσως να μην του αρέσουν, να επεξεργάζεται καλύτερα τις εντολές του εφόσον έχει γνώσεις για τη συμπεριφορά του κ.α. Κάτι τέτοιο είναι δυνατό με χρήση αλγορίθμων μηχανικής μάθησης.

2. Βελτίωση των αλγορίθμων αναζήτησης προϊόντων από τη βάση δεδομένων. Μια τέτοια αναβάθμιση όχι μόνο θα αυξήσει την ευχαρίστηση των καταναλωτών αφού θα καταλήγει η εφαρμογή πιο γρήγορα στο προϊόν που αυτοί αναζητούν, αλλά και θα αυξήσει την αποδοτικότητα και χρόνο απόκρισης της εφαρμογής ενώ θα μειώσει τον χρόνο αναζήτησης προϊόντων σε μεγάλες βάσεις δεδομένων.

3. Παροχή δυνατότητας να διατηρείται το καλάθι αγορών και μετά την αποσύνδεση του καταναλωτή από την εφαρμογή, καθώς και τη δυνατότητα γρήγορης επανάληψης μιας προηγούμενης παραγγελίας.

4. Επέκταση της μεθόδου ολοκλήρωσης παραγγελίας για να επαληθεύει τη διεύθυνση αποστολής που παρέχει ο καταναλωτής, να μπορεί να ελέγξει για παράδειγμα αν είναι πραγματική και να πράττει αναλόγως.

5. Υποστήριξη ηλεκτρονικών πληρωμών όπως με χρεωστικές και πιστωτικές κάρτες, PayPal κ.α. Αυτό προϋποθέτει και την ασφαλή αποθήκευση και επεξεργασία των προσωπικών δεδομένων των καταναλωτών.

6. Δημιουργία μιας γραφικής διεπαφής (Graphical User Interface, GUI) για τον άμεσο έλεγχο των ρυθμίσεων της εφαρμογής, τις καταστάσεις της και για επεξεργασία της βάσης δεδομένων χωρίς να είναι απαραίτητη κάποια τρίτη υπηρεσία όπως το phpMyAdmin. Επίσης σε αυτή μπορούν να προστεθούν στατιστικά χρήσης της εφαρμογής είτε με αριθμούς είτε με τη μορφή γράφων όπως για παράδειγμα συχνές εντολές, πιο δημοφιλή προϊόντα, συχνότερες ώρες χρήσης της, ποσοστό λανθασμένων αναγνωρίσεων από το μοντέλο αναγνώρισης ομιλίας κ.α.

7. Δημιουργία ενός συστήματος προώθησης προϊόντων. Μπορεί μια επιχείρηση να θέλει να παροτρύνει τους καταναλωτές να αγοράσουν ένα (ή περισσότερα) προϊόν για διάφορους λόγους. Μια τέτοια λειτουργία μπορεί να έχει τη μορφή διαφήμισης όταν το πρόγραμμα φτάσει μια κατάσταση ή την τεχνητή εισαγωγή αυτών των προϊόντων στα πιθανά αποτελέσματα μιας αναζήτησης ανεβάζοντας τη βαθμολογία τους στη διαδικασία αυτή.

8. Δημιουργία συστήματος εκπτώσεων, όπου ο διαχειριστής θα μπορεί να θέτει όσα προϊόντα επιθυμεί σε έκπτωση, πιθανώς με τη βοήθεια ενός GUI και σε συνεργασία με το σύστημα προώθησης προϊόντων να προτείνονται πρώτα αυτά στις αναζητήσεις.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Chanana, N., & Goele, S. (2012). FUTURE OF E-COMMERCE IN INDIA.

[2] Wikimedia Foundation. (2021, July 21). *Timeline of e-commerce*. Wikipedia. Retrieved September 5, 2021, from https://en.wikipedia.org/wiki/Timeline_of_e-commerce.

[3] Hoy, M. B. (2018). Alexa, Siri, Cortana, and more: An introduction to voice assistants. Medical Reference Services Quarterly, 37(1), 81–88. https://doi.org/10.1080/02763869.2018.1404391

[4] Khanna, Anirudh & Pandey, Bishwajeet & Vashishta, Kushagra & Kalia, Kartik & Bhale, Pradeepkumar & Das, Teerath. (2015). A Study of Today's A.I. through Chatbots and Rediscovery of Machine Intelligence. International Journal of u- and e-Service, Science and Technology. 8. 277-284. 10.14257/ijunesst.2015.8.7.28.

[5] Yu, D., & Deng, L. (2015). Automatic speech recognition. Signals and Communication Technology. https://doi.org/10.1007/978-1-4471-5779-3

[6] Davis, K. H., Biddulph, R., & Balashek, S. (1952). Automatic recognition of spoken digits. The Journal of the Acoustical Society of America, 24(6), 637–642. https://doi.org/10.1121/1.1906946

[7] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., ... & Vesely, K. (2011). The Kaldi speech recognition toolkit. In IEEE 2011 workshop on automatic speech recognition and understanding (No. CONF). IEEE Signal Processing Society.

[8] Dehak, N., Torres-Carrasquillo, P. A., Reynolds, D., & Dehak, R. (2011). Language recognition via i-vectors and dimensionality reduction. Interspeech 2011. https://doi.org/10.21437/interspeech.2011-328

[9] Walker, Willie & Lamere, Paul & Kwok, Philip & Raj, Bhiksha & Singh, Rita & Gouvea, Evandro & Wolf, Peter & Wölfel, Joe. (2004). Sphinx-4: A flexible open source framework for speech recognition. Sun Microsystems.

[10] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, Deep Speech: Scaling up end-to-end speech recognition, 2014. arXiv: 1412.5567 [cs.CL].

[11] Heafield, K. (2011). KenLM: Faster and Smaller Language Model Queries. WMT@EMNLP.

[12] Brown, P. F., Pietra, V. J. D., deSouza, P. V., Lai, J. C., & Mercer, R. L. (1992). Class-Based n-gram Models of Natural Language. Computational Linguistics, 18(4), 467–480. https://aclanthology.org/J92-4003

[13] Hasan, Md & Jamil, Mustafa & Rabbani, Golam & Rahman, Md. Saifur. (2004). Speaker Identification Using Mel Frequency Cepstral Coefficients. Proceedings of the 3rd International Conference on Electrical and Computer Engineering (ICECE 2004).

[14] Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models. IEEE ASSP Magazine, 3(1), 4–16. https://doi.org/10.1109/massp.1986.1165342

[15] Rizk, Basem. (2019). Evaluation of State Of Art Open-source ASR Engines with Local Inferencing. 10.13140/RG.2.2.34901.37603.

[16] Amodei, Dario, et al. “Deep Speech 2: End-to-End Speech Recognition in English and Mandarin.” ArXiv:1512.02595 [Cs], Dec. 2015. arXiv.org, http://arxiv.org/abs/1512.02595.

[17] Belenko, M. V., & Balakshin, P. V. (2017). Comparative analysis of speech recognition systems with open code. Международный научно-исследовательский журнал, (04 (58) Часть 4), 13-18.

[18] Gaida, C., Lange, P., Petrick, R., Proba, P., Malatawy, A., & Suendermann-Oeft, D. (2014). Comparing Open-Source Speech Recognition Toolkits ⋆.

[19] Rustamov, S., Akhundova, N., & Valizada, A. (2019). Automatic speech recognition in Taxi call service systems. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 243–253. https://doi.org/10.1007/978-3-030-23943-5_18

[20] Training your own model. Training Your Own Model - Mozilla DeepSpeech 0.9.3 documentation. Retrieved September 5, 2021, from https://deepspeech.readthedocs.io/en/r0.9/TRAINING.html.

[21] Common voice by mozilla. Common Voice. Retrieved September 5, 2021, from https://commonvoice.mozilla.org/el.

[22] Αγγελιδάκης, Ν. (2015). Εισαγωγή στον προγραμματισμό με την Python. http://aggelid.mysch.gr/pythonbook/

[23] Kraus, Daniel & Reibenspiess, Victoria & Eckhardt, Andreas. (2019). How Voice Can Change Customer Satisfaction: A Comparative Analysis between E-Commerce and Voice Commerce.

[24] Tuzovic, S., Paluch, S. (2018). Conversational commerce – a new era for service business development? Service Business Development, 81–100. https://doi.org/10.1007/978-3-658-22426-4_4

[25] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd international conference on Machine learning

(ICML '06). Association for Computing Machinery, New York, NY, USA, 369–376. DOI:https://doi.org/10.1145/1143844.1143891

[26] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[27] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.