ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Αυτοματοποιημένη Εξαγωγή Κειμένου Μετά από Ομιλία

ΤΣΟΥΜΑΝΗΣ ΘΕΟΦΑΝΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κωνσταντίνος Κολομβάτσος
Επίκουρος Καθηγητής
Τμήματος Πληροφορικής & Τηλεπικοινωνιών Πανεπιστημίου Θεσσαλίας

Ημερομηνία

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Αυτοματοποιημένη Εξαγωγή Κειμένου Μετά από Ομιλία

ΤΣΟΥΜΑΝΗΣ ΘΕΟΦΑΝΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κωνσταντίνος Κολομβάτσος
Επίκουρος Καθηγητής
Τμήματος Πληροφορικής & Τηλεπικοινωνιών Πανεπιστημίου Θεσσαλίας

Ημερομηνία

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

Automatic Text Extraction After Speech

TSOUMANIS THEOFANIS

FINAL THESIS

ADVISOR

Konstantinos Kolomvatsos
Assistant Professor
Department of Informatics and Telecommunications University of Thessaly
CO ADVISOR

The Date

«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.»

Περίληψη

Η συνεχής ανάπτυξη της τεχνολογίας στην εποχή μας εξελίσεται ραγδαία, χωρίς πυλώνες να εμποδίζουν την υλοποίηση της φαντασίας του ανθρώπου. Τα τελευταία 100 χρόνια έχει παρουσιαστεί πολύ μεγάλη εξέλιξη στο hardware και το software. Η αναβάθμιση αυτών των δύο παραγόντων επέτρεψε τη δημιουργία και την ανάπτυξη για την Αναγνώριση Ομιλίας και των Chatbot. Η Αναγνώριση Ομιλίας είναι ένα πρόγραμμα που μετατρέπει τη φωνή ενός ομιλητή μέσα από ένα μικρόφωνο ή κάποιο ηχογραφημένο αρχείο σε μορφή κειμένου. Τα Chatbot είναι και αυτά προγράμματα, αλλά η δημιουργία τους στοχεύει στην επικοινωνία με τον άνθρωπο. Η χρήση της Αναγνώρισης Ομιλίας σε συνδυασμό με τα Chatbot μας κατευθύνουν στο μονοπάτι της αυτοματοποίησης. Μέρα με τη μέρα η εποπτεία του ανθρώπινου δυναμικού σε κάποιο τομέα γίνεται περιττή, αφού μπορούν οι υπολογιστές να αναλάβουν να πάρουν τον φόρτο εργασίας σε απλά ή και σε πολύπλοκα θέματα. Πολλές εταιρείες έχουν δημιουργήσει δικούς τους εικονικούς βοηθούς, οι οποίοι προσαρμόζονται με τις ανάγκες των ανθρώπων σε επίπεδο καθημερινότητας όπως είναι η Siri της Apple ή η Cortana της Microsoft και η Alexa της Amazon. Οι εικονικοί βοηθοί εντάσσονται στην καθημερινότητά μας αλλά και στον τομέα εργασίας διευκολύνοντάς μας. Εκπαίδευση, τηλεφωνικά κέντρα, υγεία ακόμα και τα μεταφορικά μέσα έχουν επηρεαστεί από τις παροχές που η τεχνολογία επιτρέπει να έχουμε. Αυτό αλλάζει με γοργούς ρυθμούς και επηρεάζει και εμάς τους ίδιους, διότι στον χρόνο που θα αφιερώναμε για να καλύψουμε μια ανάγκη κατά τη διάρκεια της μέρας ή κάποιο ρόλο στον εργασιακό τομέα, τον διοχετεύουμε κάπου αλλού. Δεν υπάρχει λόγος να αφιερώνουμε τον χρόνο μας για να εκπληρώσουμε διεργασίες που επαναλαμβάνονται συνέχεια, ενώ μπορεί να αναλάβει ο υπολογιστής να διεκπερεώσει το ζήτημα αυτό το ίδιο σωστά σε λιγότερο χρόνο. Η εργασία περιέχει τη βασική δομή για τη αυτόματη συμπλήρωση ορισμένων εγγράφων που μπορεί ένας φοιτητής να λάβει από τη γραμματεία χρησιμοποιώντας τη φωνή του. Η διευκόλυση αυτή έχει στόχο να αυτοματοποιήσει τη διαδικασία και να αφαιρέσει λίγο από το χρόνο που χρειάζεται για να την ολοκλήρωσή της, τόσο για τους φοιτητές όσο και στη γραμματεία.

Abstract

The constant development of technology now days evolve too fast, without pillars getting in the way of human fantasy. In the last 100 years it has been noticed a big evolution in hardware and software. The upgrade of these two factors lead to the creation and development of Speech Recognition and Chatbots. Speech Recognition is a program that converts the voice of a person by using a microphone or a recorded voice file in a text form. Chatbots are programs too, but their creation targets to communicate with people. Combination of Speech Recognition and Chatbots bring us closer to the automatic path. Day after day human staff is needed less to fill positions, because computers can take the workload from simple or more demanding tasks. Many enterprises have created their own virtual assistants that adapt to people's needs in the daily level like Siri from Apple, Cortana from Microsoft and Alexa from Amazon. Virtual assistants come into our daily life or in the domain of work making it easier. Education, call centers, health and even transportation have been affected by the benefits that technology allows us to have. This is changing at a rapid rate, and it affects us too, because the time we would spend covering a need during the day or a role in the work sector, we channel it somewhere else. There is no reason to spend our time fulfilling repetitive processes repeatedly when the computer can oversee this issue itself properly in less time. The task holds the basic structure for auto-completing some documents that a student can receive from the secretariat using his voice. This facilitation aims to automate the process and to remove some of the time needed to complete it, both for the students and the secretariat.

Contents

ΠΡΟΛΟΓΟΣ

Στην εποχή που ζούμε η τεχνολογία και η επιστήμη προοδεύουν με ταχύτατους ρυθμούς ως προς το hardware αλλά και το software. Νέοι και πιο σύγχρονοι υπολογιστές και άλλες συσκευές τεχνολογίας έρχονται στην αγορά και προσφέρουν περισσότερες δυνατότητες χάρη στα ανώτερα χαρακτηριστικά που προσφέρουν σε σχέση με τις προηγούμενες γενιές τους. Κάθε μέρα ένας νέος προγραμματιστής υλοποιεί την ιδέα του, επειδή η βελτίωση της τεχνολογίας του το επιτρέπει. Ένα εργαλείο που υλοποιήθηκε και έχει ξεκινήσει να χρησιμοποιείται τα τελευταία χρόνια είναι η Αναγνώριση Ομιλίας, η οποία χρησιμοποιεί την ομιλία ως τρόπο επικοινωνίας του υπολογιστή με τον άνθρωπο αντί για την χρήση του πληκτρολογίου. Η χρήση του εργαλείου αυτού μπορεί να είναι ιδιαίτερα χρήσιμη τόσο για έναν πολυάσχολο εργαζόμενο όσο και για μια εταιρία, αφού διευκολύνει κάποιες διαδικασίες, ειδικά αν συνδυαστεί με κάποιο AI Assistant κάποιας εταιρείας, όπως η Apple ή η Microsoft. Οι εταιρείες αυτές έχουν τα δικά τους chatbot ,στα οποία χρησιμοποιείται το εργαλείο της Αναγνώρισης Ομιλίας και πολλά άλλα για να υπάρχει μια ομαλή επικοινωνία με έναν χρήστη.

ΚΕΦΑΛΑΙΟ 1 ΕΙΣΑΓΩΓΗ

1.1 ΣΚΟΠΟΣ ΤΗΣ ΠΤΥΧΙΑΚΗΣ

Η πτυχιακή εργασία έχει ως γνώμονα να φέρει ένα βήμα πιο κοντά τους φοιτητές προς την αυτοματοποίηση της εξυπηρέτησής τους, για να υπάρξει ένα κέρδος ως προς τον χρόνο και την παραγωγικότητα. Εκπαιδευτικά ιδρύματα που ασχολούνται με την εκμάθηση των τομέων της τεχνολογίας, θα ήταν πολύ ενδιαφέρον να έχουν την ίδια την τεχνολογία σε μια πιο καθημερινή χρήση. Αυτό θα είχε σαν αποτέλεσμα ένας φοιτητής να εμπνευστεί και να αποδώσει καλύτερα στην πορεία της εκμάθησής του.

1.2 ΔΟΜΗ ΤΗΣ ΠΤΥΧΙΑΚΗΣ

Η εργασία αυτή αποτελείται από πέντε κεφάλαια, στα οποία έχει δημιουργηθεί κατάλληλη ροή ώστε να επιτευχθεί η βαθιά κατανόησή της. Ακολουθούν τα κεφάλαια και σύντομη αναφορά των περιεχομένων τους.

- ΚΕΦΑΛΑΙΟ 2:

  Στο κεφάλαιο δύο, γίνεται επεξήγηση για το τι είναι η αναγνώριση ομιλίας, που μπορεί να χρησιμοποιηθεί, πως γίνεται η λειτουργία της, και αναφορά με σύντομη επεξήγηση κάποιων μοντέλων. Θα γίνει μια ιστορική αναδρομή για το πως ξεκίνησε και πως εξελίχθηκε κατά τη διάρκεια των χρόνων.

- ΚΕΦΑΛΑΙΟ 3:

  Στο κεφάλαιο 3 γίνεται περιγραφή των chatbot. Τι είναι, πως λειτουργούν, ποιος είναι ο ρόλος τους και πως χρησιμοποιούνται. Γίνεται επίσης μια αναδρομή στο παρελθόν σχετικά με την εξέλιξή των chatbot, και επισήμανση των πιο σπουδαίων από αυτά.

- ΚΕΦΑΛΑΙΟ 4:

Στο κεφάλαιο 4 θα μιλήσουμε για τα περιεχόμενα του κώδικα και όλων των παραγόντων που χρειάστηκαν, για να μπορέσει να λειτουργήσει. Θα γίνει περιγραφή των εργαλείων και των βιβλιοθηκών που χρησιμοποιήθηκαν, καθώς περιγραφή για το ρόλο τους και την εγκατάστασή τους.

- ΚΕΦΑΛΑΙΟ 5:

  Στο κεφάλαιο 5, θα αναλύσουμε τον κώδικα αναλυτικά και θα εξηγήσουμε κάποιες διαφορές που υπάρχουν μεταξύ των αρχείων του, προβλήματα που παρουσιάστηκαν κατά την υλοποίησή του, καθώς και συμπεράσματα που έβγαλα με την ολοκλήρωσή του.

- ΚΕΦΑΛΑΙΟ 6:

  Στο κεφάλαιο 6, θα αναφέρουμε ενέργειες που μπορούν να γίνουν για να βελτιωθεί το πρόγραμμα, καθώς και προεκτάσεις που θα το έκαναν πολύ ενδιαφέρον, και θα προσέφεραν μια πολύ σημαντική βοήθεια σε συγκεκριμένες περιπτώσεις.

ΚΕΦΑΛΑΙΟ 2 ΑΝΑΓΝΩΡΙΣΗ ΟΜΙΛΙΑΣ

2.1 ΤΙ ΕΙΝΑΙ Η ΑΝΑΓΝΩΡΙΣΗ ΟΜΙΛΙΑΣ

Η αναγνώριση ομιλίας αναφέρεται στην τεχνολογία που επιτρέπει στους υπολογιστές να ερμηνεύουν και να επεξεργάζονται την ανθρώπινη ομιλία, μετατρέποντας τον προφορικό λόγο σε κείμενο. Η τεχνολογία αυτή χρησιμοποιεί διάφορους αλγόριθμους και μοντέλα για τον εντοπισμό λέξεων και φράσεων, καθιστώντας δυνατή την κατανόηση και την ανταπόκριση των συσκευών στη φωνητική εισαγωγή. Η διαδικασία περιλαμβάνει την ανάλυση των ηχητικών κυμάτων για την αντιστοίχιση μοτίβων ομιλίας με προϋπάρχοντα γλωσσικά μοντέλα. Με την πάροδο των ετών, οι εξελίξεις στη μηχανική μάθηση, την επεξεργασία φυσικής γλώσσας και τη βαθιά μάθηση, έχουν βελτιώσει σημαντικά την ακρίβεια και την αποτελεσματικότητα των συστημάτων αναγνώρισης ομιλίας, έχοντας έτσι τη δυνατότητα να χειριστούν προφορές, γλώσσες ακόμα και σε θορυβώδη περιβάλλοντα.

2.2 ΕΙΔΗ ΑΝΑΓΝΩΡΙΣΗΣ ΟΜΙΛΙΑΣ

Οι τεχνολογίες φωνής[1][6], όπως η Αυτόματη Αναγνώριση Ομιλίας (ASR), η Επεξεργασία Φυσικής Γλώσσας (NLP) και η Μετατροπή Ομιλίας σε Κείμενο (STT), έχουν φέρει επανάσταση στον τρόπο με τον οποίο αλληλεπιδρούμε με τις μηχανές. Από φωνητικούς βοηθούς μέχρι έξυπνα συστήματα μεταγραφής, αυτές οι τεχνολογίες βελτιώνουν την επικοινωνία και την προσβασιμότητα. Παρακάτω αναλύονται οι βασικές αρχές και οι εφαρμογές κάθε τεχνολογίας ξεχωριστά.

Η αυτόματη αναγνώριση ομιλίας (ASR) είναι μια τεχνολογία που μετατρέπει τον προφορικό λόγο σε γραπτό κείμενο. Χρησιμοποιείται ευρέως σε εφαρμογές όπως οι φωνητικοί βοηθοί (π.χ. Siri, Alexa), οι υπηρεσίες μεταγραφής και τα συστήματα φωνητικού ελέγχου. Τα

συστήματα ASR βασίζονται σε μοντέλα μηχανικής μάθησης, ιδίως βαθιάς μάθησης, για την επεξεργασία ηχητικών σημάτων, τον εντοπισμό φωνημάτων και την αντιστοίχισή τους σε λέξεις. Οι προκλήσεις περιλαμβάνουν το χειρισμό των προφορών, του θορύβου υποβάθρου και των παραλλαγών στα μοτίβα ομιλίας. Τα σύγχρονα συστήματα ASR, όπως αυτά που βασίζονται σε επαναλαμβανόμενα νευρωνικά δίκτυα (RNN) ή μετασχηματιστές, επιτυγχάνουν υψηλή ακρίβεια με εκπαίδευση σε μεγάλα σύνολα δεδομένων επισημασμένης ομιλίας. Το ASR αποτελεί κρίσιμο συστατικό της αλληλεπίδρασης μεταξύ ανθρώπου και υπολογιστή, επιτρέποντας την ελεύθερη επικοινωνία μεταξύ χρηστών και μηχανών μέσω της φυσικής γλώσσας.

Η Επεξεργασία Φυσικής Γλώσσας (NLP) είναι ένας τομέας της τεχνητής νοημοσύνης που επικεντρώνεται στο να επιτρέπει στους υπολογιστές να κατανοούν, να ερμηνεύουν και να παράγουν ανθρώπινη γλώσσα. Περιλαμβάνει εργασίες όπως η ανάλυση συναισθήματος, η μηχανική μετάφραση, η περίληψη κειμένου και η απάντηση ερωτήσεων. Η NLP βασίζεται σε τεχνικές όπως η επισημείωση, η επισήμανση μέρους του λόγου και η σημασιολογική ανάλυση, που συχνά τροφοδοτούνται από μοντέλα βαθιάς μάθησης όπως οι μετασχηματιστές (π.χ. BERT, GPT). Αυτά τα μοντέλα εκπαιδεύονται σε τεράστια σώματα κειμένων για την καταγραφή γλωσσικών προτύπων και συμφραζομένων. Οι εφαρμογές NLP περιλαμβάνουν chatbots, μηχανές αναζήτησης και εργαλεία γλωσσικής μετάφρασης. Παρά τις προόδους, προκλήσεις όπως η ασάφεια, ο σαρκασμός και οι πολιτισμικές αποχρώσεις παραμένουν. Το NLP συνεχίζει να εξελίσσεται, γεφυρώνοντας το χάσμα μεταξύ της ανθρώπινης επικοινωνίας και της μηχανικής κατανόησης.

Η τεχνολογία Speech-to-Text (STT) επιτρέπει τη μετατροπή της προφορικής ομιλίας σε γραπτό κείμενο, βελτιώνοντας την πρόσβαση σε πληροφορίες και την αλληλεπίδραση με τις μηχανές. Χρησιμοποιείται σε εφαρμογές όπως η απομαγνητοφώνηση συνομιλιών, οι βοηθητικές τεχνολογίες για άτομα με προβλήματα ακοής και οι πλατφόρμες δημιουργίας υποτίτλων σε πραγματικό χρόνο. Οι σύγχρονες τεχνικές STT βασίζονται σε βαθιά νευρωνικά δίκτυα, όπως τα Convolutional Neural Networks (CNNs) και τα Transformer-based μοντέλα, που επιτυγχάνουν υψηλή ακρίβεια αναγνώρισης. Οι προκλήσεις περιλαμβάνουν την αναγνώριση διαλέκτων, την αντιμετώπιση θορύβου και τη σωστή κατανόηση συμφραζομένων. Με τις συνεχείς βελτιώσεις στη μηχανική μάθηση και τη συλλογή δεδομένων, τα συστήματα STT γίνονται όλο και πιο ακριβή και προσαρμόσιμα, ενισχύοντας τη φυσική αλληλεπίδραση μεταξύ ανθρώπων και τεχνολογίας.

2.3 ΠΩΣ ΛΕΙΤΟΥΡΓΕΙ Η ΑΝΑΓΝΩΡΙΣΗ ΟΜΙΛΙΑΣ

Η τεχνολογία αναγνώρισης ομιλίας λειτουργεί μέσω ενός συνδυασμού front-end επεξεργασίας και back-end υπολογισμού και δίνουν τη δυνατότητα στους υπολογιστές να ‘κατανοήσουν’ την ανθρώπινη ομιλία. Στο front-end, η διαδικασία ξεκινά όταν ο χρήστης μιλάει σε ένα μικρόφωνο. Το ηχητικό σήμα συλλαμβάνεται και μετατρέπεται σε ψηφιακή μορφή με τη χρήση αναλογικής και ψηφιακής μετατροπής (ADC)[7]. Αυτός ο μετασχηματισμός επιτρέπει στο σύστημα να αναλύσει τα ηχητικά κύματα και να προσδιορίσει βασικά χαρακτηριστικά, όπως η συχνότητα, το ύψος και ο τόνος. Στη συνέχεια, το σύστημα εφαρμόζει τεχνικές μείωσης του θορύβου και φιλτραρίσματος για να βελτιώσει τη σαφήνεια της εισερχόμενης ομιλίας και να αφαιρέσει το θόρυβο του περιβάλλοντος, εξασφαλίζοντας μια πιο ακριβή μεταγραφή.

Μόλις γίνει η επεξεργασία του ηχητικού σήματος, αναλαμβάνουν οι αλγόριθμοι του back-end. Το ψηφιακό σήμα αναλύεται σε μικρότερα τμήματα που ονομάζονται φωνήματα, τις βασικές μονάδες του ήχου στη γλώσσα. Αυτά τα φωνήματα συγκρίνονται με ένα προϋπάρχον γλωσσικό μοντέλο, το οποίο βοηθά το σύστημα να αναγνωρίσει λέξεις και φράσεις. Τα συστήματα αναγνώρισης ομιλίας συχνά βασίζονται σε μοντέλα κρυμμένου μαρκόβ (HMM)[2] ή νευρωνικά δίκτυα που βασίζονται σε βαθιά μάθηση για να προβλέψουν την πιο πιθανή ακολουθία λέξεων με βάση την είσοδο. Η επεξεργασία φυσικής γλώσσας (NLP) στη συνέχεια ερμηνεύει το νόημα των αναγνωρισμένων λέξεων, λαμβάνοντας υπόψη τη γραμματική, τη σύνταξη και τα συμφραζόμενα.

Στα συστήματα αναγνώρισης ομιλίας που βασίζονται στο νέφος, τα ψηφιοποιημένα και επεξεργασμένα δεδομένα αποστέλλονται σε ισχυρούς διακομιστές που αναλύουν μεγάλα σύνολα δεδομένων σε πραγματικό χρόνο. Αυτά τα συστήματα χρησιμοποιούν μοντέλα μηχανικής μάθησης που εκπαιδεύονται σε τεράστιες ποσότητες προφορικού λόγου, επιτρέποντάς τους να βελτιώνουν την ακρίβεια μαθαίνοντας συνεχώς από νέες αλληλεπιδράσεις. Το τελικό στάδιο περιλαμβάνει τη μετατροπή της αναγνωρισμένης ομιλίας σε κείμενο ή εντολές, επιτρέποντας στο σύστημα να ανταποκριθεί κατάλληλα, είτε εκτελώντας μια ενέργεια, είτε μεταγράφοντας κείμενο, είτε αλληλεπιδρώντας με έναν χρήστη.

Οι σύγχρονες εφαρμογές αναγνώρισης ομιλίας χρησιμοποιούν ένα μείγμα επεξεργασίας στη συσκευή και υπολογιστικού νέφους, εξισορροπώντας την αποτελεσματικότητα και την ιδιωτικότητα. Ενώ η τοπική επεξεργασία εξασφαλίζει γρήγορες απαντήσεις για απλές εντολές, η ανάλυση στο νέφος επιτρέπει πιο σύνθετες και διαφοροποιημένες εργασίες αναγνώρισης. Καθώς η τεχνολογία εξελίσσεται, νέες μέθοδοι όπως οι αρχιτεκτονικές που βασίζονται σε μετασχηματιστές (π.χ. Whisper της OpenAI και BERT της Google) βελτιώνουν περαιτέρω τις δυνατότητες αναγνώρισης ομιλίας, καθιστώντας τις φωνητικές αλληλεπιδράσεις πιο απρόσκοπτες και φυσικές από ποτέ.

2.4 ΘΕΩΡΗΤΙΚΗ ΚΑΙ ΠΡΑΚΤΙΚΗ ΠΡΟΣΕΓΓΙΣΗ

Η ανάπτυξη των συστημάτων αναγνώρισης ομιλίας βασίζεται τόσο σε θεωρητικά πλαίσια όσο και σε πρακτικές εφαρμογές. Ένα θεμελιώδες θεωρητικό μοντέλο που χρησιμοποιείται σε αυτόν τον τομέα είναι το μοντέλο κρυμμένου Μαρκόφ (Hidden Markov Model - HMM). Τα HMM είναι στατιστικά μοντέλα που αναπαριστούν συστήματα τα οποία μεταβαίνουν μεταξύ διαφόρων καταστάσεων, καθεμία από τις οποίες συνδέεται με μια κατανομή πιθανότητας. Στο πλαίσιο της αναγνώρισης ομιλίας, οι καταστάσεις αυτές αντιστοιχούν σε διαφορετικά φωνήματα - τις διακριτές μονάδες ήχου σε μια γλώσσα. Το μοντέλο αναλύει ακολουθίες ακουστικών σημάτων και εκτιμά την πιο πιθανή ακολουθία φωνημάτων, τα οποία στη συνέχεια αντιστοιχίζονται σε λέξεις. Πρακτικά, αυτό περιλαμβάνει την κατάτμηση του συνεχούς σήματος ομιλίας σε πλαίσια, την εξαγωγή σχετικών χαρακτηριστικών (όπως συνιστώσες συχνότητας) και την εφαρμογή αλγορίθμων για την αποκωδικοποίηση της πιο πιθανής ακολουθίας λέξεων. Οι εξελίξεις στη μηχανική μάθηση, ιδίως στη βαθιά μάθηση, έχουν βελτιώσει περαιτέρω την ακρίβεια αυτών των συστημάτων, επιτρέποντας τη μοντελοποίηση πολύπλοκων μοτίβων στα δεδομένα ομιλίας.

Στις πρακτικές εφαρμογές, τα συστήματα αναγνώρισης ομιλίας υποβάλλονται σε μια σειρά βημάτων για τη μετατροπή του προφορικού λόγου σε κείμενο. Αρχικά, το αναλογικό σήμα ομιλίας συλλαμβάνεται και ψηφιοποιείται, ακολουθούμενο από προεπεξεργασία για την απομάκρυνση του θορύβου και τη βελτίωση της ποιότητας του σήματος. Στη συνέχεια πραγματοποιείται εξαγωγή χαρακτηριστικών, όπου προκύπτουν χαρακτηριστικά όπως οι συντελεστές Mel-Frequency Cepstral Coefficients (MFCC)[10] για την αναπαράσταση του φωνητικού περιεχομένου της ομιλίας. Αυτά τα χαρακτηριστικά χρησιμεύουν ως είσοδοι στο μοντέλο αναγνώρισης, το οποίο αποκωδικοποιεί την ακολουθία για να παράγει το αντίστοιχο κείμενο. Η ενσωμάτωση γλωσσικών μοντέλων βελτιώνει περαιτέρω αυτή τη διαδικασία, λαμβάνοντας υπόψη το πλαίσιο και την πιθανότητα των ακολουθιών λέξεων, βελτιώνοντας έτσι την ακρίβεια της μεταγραφής.

Μια μαθηματική συνάρτηση που χρησιμοποιείται συνήθως σε αυτό το πλαίσιο είναι ο αλγόριθμος Viterbi[9], ο οποίος βρίσκει την πιο πιθανή ακολουθία κρυφών καταστάσεων (φωνήματα) που οδηγεί στην παρατηρούμενη ακολουθία ακουστικών σημάτων. Ο αλγόριθμος υπολογίζει την πιθανότητα κάθε πιθανής ακολουθίας καταστάσεων και επιλέγει εκείνη με τη μεγαλύτερη πιθανότητα, αποκωδικοποιώντας ουσιαστικά την προφορική είσοδο σε κείμενο. Αυτή

η πιθανολογική προσέγγιση επιτρέπει στο σύστημα να χειρίζεται διακυμάνσεις στα πρότυπα ομιλίας και στο θόρυβο υποβάθρου, ενισχύοντας την ανθεκτικότητα και την αξιοπιστία του.

Η πρακτική εφαρμογή αυτών των θεωρητικών μοντέλων έχει οδηγήσει στην ανάπτυξη διαφόρων εφαρμογών αναγνώρισης ομιλίας, που κυμαίνονται από εικονικούς βοηθούς που ελέγχονται με τη φωνή μέχρι αυτόματες υπηρεσίες μεταγραφής. Η συνεχής έρευνα και ανάπτυξη σε αυτόν τον τομέα αποσκοπεί στη βελτίωση της ακρίβειας και της αποτελεσματικότητας αυτών των συστημάτων, καθιστώντας τα πιο προσαρμόσιμα σε διαφορετικές γλώσσες, προφορές και στυλ ομιλίας. Η ενσωμάτωση προηγμένων τεχνικών μηχανικής μάθησης, όπως τα βαθιά νευρωνικά δίκτυα, έχει προωθήσει περαιτέρω τις δυνατότητες της αναγνώρισης ομιλίας, επιτρέποντας πιο φυσικές και διαισθητικές αλληλεπιδράσεις ανθρώπου-υπολογιστή.

Τα θεωρητικά θεμέλια της αναγνώρισης ομιλίας, σε συνδυασμό με πρακτικές στρατηγικές υλοποίησης, έχουν καταλήξει σε εξελιγμένα συστήματα ικανά να κατανοούν και να μεταγράφουν την ανθρώπινη ομιλία με αξιοσημείωτη ακρίβεια. Καθώς η τεχνολογία συνεχίζει να εξελίσσεται, τα συστήματα αυτά αναμένεται να γίνουν ακόμη πιο αναπόσπαστο μέρος διαφόρων πτυχών της καθημερινής ζωής, γεφυρώνοντας περαιτέρω το χάσμα μεταξύ ανθρώπων και υπολογιστών.

2.5 ΕΦΑΡΜΟΓΗ ΤΗΣ ΑΝΑΓΝΩΡΙΣΗΣ ΟΜΙΛΙΑΣ ΣΤΗΝ ΚΑΘΗΜΕΡΙΝΗ ΖΩΗ

Η τεχνολογία αναγνώρισης ομιλίας έχει γίνει αναπόσπαστο μέρος της καθημερινής μας ρουτίνας και μπορεί να χρησιμοποιηθεί με διάφορους τρόπους και σε αρκετούς τομείς. Η βοήθεια που προσφέρει αφορά απλά πράγματα και καταστάσεις όπως ένα search σε κάποιο browser με τη χρήση της ομιλίας για διευκόλυνση ή ακόμα και πιο σοβαρά ζητήματα όπως η βοήθεια προς μια ανθρώπινη ζωή.

Η ύπαρξη των εικονικών βοηθών στις μέρες μας είναι ένα εντυπωσιακό χαρακτηριστικό, το οποίο μας λύνει τα χέρια βοηθώντας μας στην καθημερινότητα κερδίζοντας μας χρόνο και κόπο. Με την τωρινή τεχνολογία και τη συνεχή εξέλιξή της το απλό speech recognition έγινε ένας εικονικός βοηθός ο οποίος «καταλαβαίνει» φωνητικές εντολές σε αρκετούς διαφορετικούς τομείς. Ξεκίνησε από ένα «κομμάτι» της αναζήτησης της μπάρας εργασίας ενός browser και πλέον είναι ένα έξυπνο σύστημα που συναναστρέφεται μαζί μας σε καθημερινές ανάγκες που προκύπτουν. Αυτό μπορεί να βρίσκεται σε έναν σταθερό υπολογιστή, σε ένα λάπτοπ, σε ένα κινητό, στο αμάξι μας, κλπ. Με την εμφάνιση των smartphones παρουσιάστηκαν στοιχεία του speech recognition

εντυπωσιακά για τα δεδομένα του τότε. Πλέον όμως τα smartphone λόγω των υψηλών επιδόσεων και των υπολογιστικών δυνατοτήτων έχουν λάβει πολλά νέα χαρακτηριστικά που κάποτε φαίνονταν απρόσιτα. Η δυνατότητα των smartphone να συνδέονται με όλες σχεδόν τις συσκευές και στο σπίτι και με αρκετές εκτός του σπιτιού «ανοίγει την πόρτα» για το speech recognition να αξιοποιηθεί στο έπακρον. Ορισμένες από αυτές μας επιτρέπουν να τις ενεργοποιούμε, απενεργοποιούμε ή και να τις ρυθμίζουμε με τη φωνή μας. Παραδείγματα συσκευών που το επιτρέπουν αυτό είναι λάμπες, ηχεία, κλιματιστικά, θερμοσίφωνες και πολλά άλλα.

Στον εργασιακό χώρο, οι επαγγελματίες αξιοποιούν το λογισμικό ομιλίας σε κείμενο για την απομαγνητοφώνηση συναντήσεων, την υπαγόρευση μηνυμάτων ηλεκτρονικού ταχυδρομείου και τη δημιουργία εγγράφων, βελτιώνοντας έτσι τις ροές εργασίας και μειώνοντας τον χρόνο που χρειάζεται για την εισαγωγή δεδομένων πληκτρολογώντας. Για παράδειγμα, εφαρμογές όπως η Otter.ai παρέχουν υπηρεσίες απομαγνητοφώνησης σε πραγματικό χρόνο, επιτρέποντας στους χρήστες να επικεντρωθούν στις συζητήσεις χωρίς την απόσπαση της προσοχής από την καταγραφή σημειώσεων. Αυτή η δυνατότητα είναι ιδιαίτερα επωφελής σε περιβάλλοντα με γρήγορους ρυθμούς, όπου η ακριβής και γρήγορη καταγραφή πληροφοριών είναι ζωτικής σημασίας.

Στην αυτοκινητοβιομηχανία, τα συστήματα αναγνώρισης φωνής ενσωματώνονται στα οχήματα, επιτρέποντας στους οδηγούς να ελέγχουν την πλοήγηση, να πραγματοποιούν τηλεφωνικές κλήσεις και να προσαρμόζουν τις ρυθμίσεις ψυχαγωγίας χωρίς να παίρνουν τα χέρια τους από το τιμόνι. Αυτή η προσέγγιση hands-free ενισχύει την ασφάλεια ελαχιστοποιώντας την απόσπαση της προσοχής και επιτρέποντας στους οδηγούς να επικεντρωθούν στο δρόμο. Μάρκες όπως η Ford και η BMW έχουν ενσωματώσει προηγμένα χαρακτηριστικά φωνητικού ελέγχου στα συστήματα infotainment τους, αντανακλώντας την αυξανόμενη ζήτηση για τέτοια τεχνολογία στα σύγχρονα οχήματα.

Ο τομέας της υγειονομικής περίθαλψης έχει επίσης σημειώσει σημαντικές εξελίξεις λόγω της αναγνώρισης ομιλίας. Οι γιατροί χρησιμοποιούν εφαρμογές με φωνητική υποστήριξη για να ενημερώνουν τους φακέλους των ασθενών, να συνταγογραφούν φάρμακα και να ανακτούν πληροφορίες από ιατρικές βάσεις δεδομένων χωρίς χειροκίνητη εισαγωγή. Αυτό όχι μόνο βελτιώνει την ακρίβεια της τεκμηρίωσης, αλλά επιτρέπει επίσης στους παρόχους υγειονομικής περίθαλψης να διατηρούν οπτική επαφή και να δημιουργούν καλύτερη σχέση με τους ασθενείς κατά τη διάρκεια των διαβουλεύσεων. Επιπλέον, οι ασθενείς με προβλήματα ομιλίας επωφελούνται από τις υποστηρικτικές τεχνολογίες που μετατρέπουν τον προφορικό τους λόγο σε κείμενο, διευκολύνοντας την σαφέστερη επικοινωνία με τους ιατρούς.

Επιπλέον, τα άτομα με σωματικές αναπηρίες βρίσκουν τις τεχνολογίες με φωνητικό έλεγχο ενθαρρυντικές, καθώς μπορούν να χειρίζονται υπολογιστές, smartphones και άλλες συσκευές χωρίς να βασίζονται σε παραδοσιακές μεθόδους εισαγωγής, όπως πληκτρολόγια ή οθόνες αφής. Η ενσωμάτωση της τεχνολογίας επιτρέπει στους χρήστες να εκτελούν ορισμένες εργασίες όπως να συμπληρώνουν κείμενα ή και να ελέγχουν συσκευές που ενδέχεται για εκείνους να ήταν κάτι δύσκολο. Με την πάροδο του χρόνου η χρήση της τεχνολογίας γίνεται πιο φιλική προς το χρήστη βοηθόντας τον καθημερινά.

Στον τομέα της εκπαίδευσης βηθάει σε πολύ μεγάλο βαθμό μαθητές και φοιτητές που δυσκολεύονται να κατανοήσουν το μάθημα είτε λόγω μαθησιακών δυσκολιών είτε λόγω γλώσσας. Υπάρχουν πλέον εργαλεία τα οποία εντάσονται σε εικονικές πλατφόρμες όπως το Zoom, ομάδες Microsoft, Google Meet και δημοφιλείς πλατφόρμες ηλεκτρονικής μάθησης όπως το Moodle, Blackboard και Canvas. Η ένταξη του εργαλείου στις πλατφόρμες, προσφέρει δύνανότητα μετάφρασης σε πραγματικό χρόνο, καταγραφή κάποιας διάλεξης ή σεμιναρίου μεταφρασμένη ώστε ένας μαθητής να μπορεί να επιστρέψει σε προηγούμενο μάθημα για καλύτερη κατανόηση. Η καταγραφή των μαθημάτων βοηθάει τους μαθητές να βελτιώσουν τις γνώσεις τους με το δικό τους ρυθμό αφού η επιστροφή σε προηγούμενο μάθημα είναι εφικτή.Οι μεταφράσεις γίνονται σε πολλές γλώσσες ώστε να υπάρχει μια ισοτιμία κατά την διάρκεια της διάλεξης και να μπορούν όλοι να παρακολουθήσουν το μάθημα στην μητρική τους γλώσσα για περισσότερη διευκόλυνση.

Τα εργαλεία που κάνουν μεταφράσεις σε πραγματικό χρόνο, οι οποίες έιναι ακριβείς και αξιόπιστες τροφοδοτούνται από προηγμενη τεχνητή νοημοσύνη και μηχανική μάθηση. Ένα από τα εργαλεία αυτά είναι το Lingvanex το οποίο μπορεί να πραγματοποιήσει μεταφράσεις με μεγάλη ακρίβεια και σε δύσκολες συνθήκες με θόρυβο. Μια λειτουργία ακόμα που διαθέτει και είναι ιδιαίτερα χρήσιμη είναι να βελτιώνει τον τρόπο ομιλίας κάποιου μαθητή κάνοντας σύγκριση με διαφορετικές εγγενείς προφορές και προτείνοντας τρόπους για να βελτιώσουν τις δεξιότητές τους. Επίσης οι καθηγητές μπορούν να παρακολουθήσουν την πρόοδο των μαθητών μέσω εργαλείων, κερδίζοντας έτσι χρόνο και εστιάζοντας στις πραγματικές ανάγκες τους ώστε να επικεντρωθούν σε πιο σημαντικές πτυχές της μαθησιακής διαδικασίας. Αυτά τα εργαλεία δίνουν τη δυνανότητα σε άτομα με ορισμένες μαθησιακές δυσκολίες να κατανοούν καλύτερα τις διαλέξεις εφόσον γίνεται καταγραφή αυτών και ο ήχος παιρνάει σε μορφή κειμένου.

2.6 ΙΣΤΟΡΙΑ ΚΑΙ ΕΞΕΛΙΞΗ ΤΗΣ ΑΝΑΓΝΩΡΙΣΗΣ ΟΜΙΛΙΑΣ

Ας ξεκινήσουμε κάνοντας αναφορά στον Λούντιμαρ Χέρμαν[12] έναν γερμανό επιστήμονα ομιλίας και φυσιολόγο ο οποίος έκανε χρήση φωνητικής καταγραφής το 1894. Το «δώρο» του για την εξέλιξη της έρευνάς του, ήταν η λύση του προβλήματος για την αναγνώριση φωνιέντων κατά την έντονη αλλαγή της συχνότητας φωνής, κατά την ομιλία. Για μια συνηθισμένη περίπτωση ομιλίας δεν παρουσίαζε κάποιο πρόβλημα, όταν όμως χρησιμοποιήθηκε για να γίνει καταγραφή ενός ομιλητή με διαφορετικά μήκη φωνητικών οδών, όπως για παράδειγμα τραγουδιαστές μπάσο και σοπράνο, οι οποίοι μπορούσαν να παράγουν ήχους που θεωρούνται ότι ανήκουν στην ίδια φωνητική κατηγορία, εκεί δεν μπορούσε να διακρίνει σωστά τις φασματικές πληροφορίες που στηρίζουν την ταυτότητα των φωνηέντων. Το κάθε φωνήεν σύμφωνα με τον Χέρμαν έχει τη δικία του μοναδική συχνότητα που εμφανίζεται να ποικίλλει χωρίς να αλλοιώνεται η ταυτότητα του φωνήεν αυτού. Έλυσε το πρόβλημα αυτό χωρίζοντας σε κατηγορίες τα φωνήεντα με τη συχνότητα τους για να μπορεί να γίνει η καταγραφή χωρίς λάθη.

Το πρώτο μοντέλο ASR που κατασκευάστηκε ήταν το Audrey από τρεις ερευνητές της Bell Labs, τον Stephen Balashek, τον R. Biddulph και τον KH Davis και ο σκοπός του ήταν να αναγνωρίζει τα ψηφία από το ένα έως το εννέα υπολογίζοντας τους σχηματιστές στο φάσμα ισχύος κάθε εκφοράς. Για την εποχή του, αυτό που έκανε ήταν πολύ εντυπωσιακό. Αξίζει να αναφερθεί το επίτευγμα του Gunnar Fant[11] το 1952 που προσδιόρισε την ομιλία ως συνδυασμό μιας πηγής ήχου και ενός γραμμικού ακουστικού φίλτρου. Η συνεισφορά από άλλους επιστήμονες αναγνωρίζεται για τη συνεχή εξέλιξη του μοντέλου αλλά η ανάπτυξή του οφείλεται κυρίως σε Fant. Το αποτέλεσμα του μεντέλου πηγής-φίλτρου παρόλο που ήταν μια απλή προσέγγιση, η απλότητά του, επέτρεψε τη χρήση του σε ορισμένες εφαρμογές όπως είναι η σύνθεση ομιλίας και η ανάλυση ομιλίας. Το μοντέλο αυτό συσχετίζεται με τη γραμμική πρόβλεψη που είναι μια μαθηματική πράξη που υπολογίζει μελλοντικές τιμές ενός σήματος διακριτού χρόνου ως γραμμική συνάρτηση προηγούμενων δειγμάτων.

Δέκα χρόνια μετά η IBM παρουσίασε στην Παγκόσμια Έκθεση του 1962 το «Shoebox». Το Shoebox[3] ήταν μια μηχανή που πραγματοποιούσε μαθηματικές πράξεις με τους αριθμούς μηδέν έως εννέα αναγνώριζε την ομιλία έως 16 λέξεις. Ήταν μια μια βελτιωμένη έκδοση του μοντέλου Audrey της Bell Labs. Όλα τα μοντέλα που δημιουργήθηκαν έως τώρα, απαιτούσαν από τους χρήστες να κάνουν παύση μετά από κάθε λέξη. Ο Raj Reddy, μεταπτυχιακός φοιτητής του Πανεπιστημίου Stanford, ήταν το πρώτο άτομο που ανέλαβε την συνεχή ομιλία και το αποτέλεσμα του ήταν το σύστημα Reddy που εξέδωσε εντολές για το παιχνίδι σκάκι. Το διάστημα εκείνο, επινοήθηκε ο αλγόριθμος δυναμικής στρέβλωσης χρόνου (DTW) από Σοβιετικούς ερευνητές και η χρήση του στόχευε στην δημιουργία ενός αναγνωριστικού ικανό να λειτουργήσει σε λεξιλόγιο

200 λέξεων. Η λειτουργία του ήταν να διαιρεί την ομιλία σε μικρά καρέ και να τα επεξεργάζεται ξεχωριστά ως μια ενιαία μονάδα.

Το 1971 έγινε μια πενταετή χρηματοδότηση για την κατανόηση ομιλίας που αναζητούσε ελάχιστο μέγεθος λεξιλογίου 1.000 λέξεων. Θεώρησαν πως η κατανόηση ομιλίας θα ήταν το κλειδί για την πρόοδο στην αναγνώριση ομιλίας, κάτι που αποδείχτηκε αργότερα αναληθές. Η Εταιρία Επεξεργασίας Σήματος IEEE διοργάνωσε το ICASSP ένα κορυφαίο διεθνές ετήσιο συνέδριο στη Φιλαδέλφεια για την Ακουστική, την Ομιλία και την Επεξεργασία Σήματος, που υπήρξε σημαντικός χώρος για τη δημοσίευση της έρευνας για την αναγνώριση ομιλίας.

Στα τέλη της δεκαετίας του 1960 , ο Leonard Baum ανέπτυξε τα μαθηματικά αλυσίδων Markov στο Ινστιτούτο Αμυντικής Ανάλυσης. Μια δεκαετία αργότερα, στο CMU, οι μαθητές Raj Reddy, James Baker και Janet M. Baker, χρησιμοποίησαν το κρυφό μοντέλο (HMM) για την αναγνώριση ομιλίας. Το μοντέλο αυτό επέτρεψε στους ερευνητές να συνδυάσουν διαφορετικές πηγές γνώσης, όπως η ακουστική, η γλώσσα και η σύνταξη, σε ένα ενοποιημένο πιθανολογικό μοντέλο. Το 1980 υλοποιήθηκε το Tangora μια γραφομηχανή με ενεργοποίηση φωνής από την ομάδα του Fred Jelinek της IBM, που χειριζόταν μέγιστο λεξιλόγιο 20.000 λέξεων. Η στατιστική προσέγγιση του Jelinek έδινε λιγότερη έμφαση στην μίμιση του τρόπου με τον οποίο ο ανθρώπινος επεξεργάζεται και κατανοεί την ομιλία υπέρ της χρήσης στατιστικών τεχνικών μοντελοποίησης ομιλίας όπως τα HMM. Το HMM αποδείχτηκε ότι ήταν ένας πιο χρήσιμος τρόπος μοντελοποίησης της ομιλίας και πήρε τη θέση του ως κυρίαρχος αλγόριθμος αναγνώρισης ομιλίας για την δεκαετία του 1980 αντικαθιστώντας τη δυναμική χρονική στρέβλωση. Οι James και Janet M. Baker ίδρυσαν την Dragon Systems το 1982, όπου ήταν ένας από τους λίγους ανταγωνιστές της IBM.

Το 1980 έκανε την εμφάνισή του το μοντέλο γλώσσας n-gram το οποίο ήταν μια ακολουθία από n γειτονικά σύμβολα με συγκεκριμένη σειρά που μπορεί να ήταν γράμματα, σημεία στίξης, συλλαβές ή και σπάνια ολόκληρες λέξεις που βρίσκονταν σε ένα σύνολο δεδομένων γλώσσας αλλά και φωνήματα που εξάγονται από ένα σύνολο δεδομένων καταγραφής ομιλίας ή γειτονικά ζεύγη βάσεων που εξάγονται από ένα γονιδίωμα. Το 1987 παρουσιάστηκε το μοντέλο back-off. Το back-off δημιουργήθηκε από τον Slava Katz και επέτρεπε στα μοντέλα γλώσσας να χρησιμοποιούν n-γραμμάρια πολλαπλού μήκους. Πρέπει να αναφερθεί ότι και η ανάπτυξη της τεχνολογίας έπαιξε πολύ σημαντικό ρόλο στην συνεχή πρόοδο του τομέα, αφού οι υπολογιστές πρόσφεραν περισσότερες δυνατότητες για βελτίωση των ήδη υπαρχόντων μοντέλων ή και δημιουργία νέων που απαιτούσαν περισσότερους πόρους για να μπορούν τα λειτουργήσουν και να προσφέρουν περισσότερα και καλύτερα αποτελέσματα. Κατά τη λήξη του προγράμματος DARPA το 1976, ο πιο σύγχρονος υπολογιστής της περιόδου που ήταν διαθέσιμος στους

ερευνητές ήταν το PDP-10 με 4 MB ram. Ήταν ικανός να αποκωδικοποιήσει 30 δευτερόλεπτα ομιλίας σε 100 λεπτά.

Το 1984, η εταιρεία ACT Ltd κατασκεύασε έναν φορητό υπολογιστή, τον Apricot Portable. Ο Apricot Portable, υποστήριζε έως και 4096 λέξεις και 64 από αυτές μπορούσαν να κρατηθούν στη μνήμη RAM κάθε φορά. Σε αντίθεση με άλλους φορητούς υπολογιστές όπως ο Compaq Portable και ο Commodore SX-64, ο Apricot Portable διέθετε οθόνη LCD 80 στηλών και 25 γραμμών η οποία ήταν πολύ μεγαλύτερη. Απέκτησε γρήγορα δημοτικότητα από το ευρύ κοινό και το αναγνώριση από την Kurzweil Applied Intelligence το 1987. Το 1990 η AT&T έφερε στο φως το dragon dictate το οποίο είναι ένα λογισμικό επεξεργασίας κλήσεων και αναγνώρισης φωνής χωρίς την απαίτηση ανθρώπινου χειριστή.

Ο Xuedong Huang ανέπτυξε το σύστημα Sphinx-II στο CMU. Το Sphinx-II (ήταν το πρώτο που έκανε ανεξάρτητο από τον ομιλητή) διέθετε μεγάλο λεξιλόγιο και έκανε συνεχή αναγνώριση ομιλίας έχοντας την καλύτερη απόδοση στην αξιολόγηση της DARPA το 1992. Ο χειρισμός συνεχούς ομιλίας με μεγάλο λεξιλόγιο ήταν ένα σημαντικό ορόσημο(κάτι θαυμαστό) στην ιστορία της αναγνώρισης ομιλίας. Ο Huang συνέχισε να ιδρύει την ομάδα αναγνώρισης ομιλίας στη Microsoft το 1993. Ο Kai-Fu0 Lee, εντάχθηκε στην Apple όπου, το 1992, βοήθησε στην ανάπτυξη ενός πρωτότυπου διασύνδεσης ομιλίας για τον υπολογιστή Apple γνωστό ως Casper. Αργότερα η Apple έδωσε άδεια χρήσης λογισμικού από τη Nuance για να παρέχει δυνατότητα αναγνώρισης ομιλίας στον ψηφιακό βοηθό της Siri. Σε αυτό το σημείο αξίζει να αναφερθεί πως το λεξιλόγιο ενός τυπικού εμπορικού συστήματος αναγνώρισης ομιλίας ήταν ανώτερο από το μέσο ανθρώπινο λεξιλόγιο.

Με τη χρηματοδότηση της DARPA στην δεκαετία του 2000 έγινε εκκίνηση δύο νέων προγραμμάτων αναγνώρισης ομιλίας. Το 2000 το EARS (Επαναχρησιμοποιήσιμη Ομιλία σε Κείμενο) και το 2002 το GALE (Παγκόσμια Αυτόνομη Εκμετάλευση Γλωσσών. Η EARS χρηματοδότησε τη συλλογή του σώματος τηλεφωνικής ομιλίας του Switchboard που περιείχε 260 ώρες ηχογραφημένων συνομιλιών από πάνω από 500 ομιλητές. Το πρόγραμμα GALE επικεντρώθηκε στην ομιλία ειδήσεων μετάδοσης των Αραβικών και Μανδαρινικών. Το 2007 η GOOGLE έκανε την πρώτη προσπάθεια για την αναγνώριση ομιλίας και το προϊόν της ήταν το GOOG-411, μια τηλεφωνική υπηρεσία καταλόγου. Η Google κατάφερε να βελτιώσει τα συστήματα αναγνώρισης της μέσα από τα πολύτιμα δεδομένα που προήλθαν από τις ηχογραφήσεις του GOOG-411. Πλέον η φωνητική αναζήτηση της google υποστηρίζεται σε περισσότερες από 30 γλώσσες.

Στις Ηνωμένες Πολιτείες, η Εθνική Υπηρεσία Ασφαλείας χρησιμοποίησε έναν τύπο αναγνώρισης ομιλίας για τον εντοπισμό λέξεων-κλειδιών τουλάχιστον από το 2006 και η

τεχνολογία αυτή επέτρεπε στους αναλυτές να αναζητούν μεγάλους όγκους ηχογραφημένων συνομιλιών και να απομονώνουν αναφορές λέξεων-κλειδιών. Η δομή των εγγραφών θύμιζε ευρετήριο και οι αναλυτές μπορούσαν να βρουν συνομιλίες που τους ενδιέφεραν εκτελώντας ερωτήματα και αναφορές στη βάση δεδομένων. Προγράμματα που το χρησιμοποίησαν ήταν το EARS της DARPA και το Babel της IARPA.

Στις αρχές του 2000 επικρατούσε ακόμα η χρήση των κρυφών μοντέλων Markov τα οποία τα λειτουργούσαν συνδυάζοντας τεχνικά νευρωνικά δίκτυα. Αρκετά μοντέλα χρησιμοποιούν τη μέθοδο βαθιάς μάθησης Long-term memory (LSTM), ένα επαναλαμβανόμενο νευρωνικό δίκτυο το οποίο δημοσιεύτηκε από τους Sepp Hochreiter & Jürgen Schmidhuber το 1997. Αυτό βοήθησε στην εκμάθηση εργασιών «Πολύ βαθιάς μάθησης» που απαιτούσαν αναμνήσεις γεγονότων από χιλιάδες διακριτά χρονικά βήματα. Περίπου το 2007, το Connectionist Temporal Classification (CTC)[8] εκπαίδευσε το LSTM και άρχισε να ξεπερνά την παραδοσιακή αναγνώριση ομιλίας σε ορισμένες εφαρμογές. Η αναγνώριση ομιλίας της Google το 2015 με την εκπαίδευση του LSTM από το CTC κατάφερε να βελτιώσει την απόδοση του κατά 49% και να εφαρμοστεί μέσω του Google Voice για όλους τις χρήστες smartphone. Οι μετασχηματιστές, είναι ένας τύπος νευρωνικού δικτύου και σε ορισμένες εργασίες παρατηρήθηκε πως πρόσφεραν υψηλότερα επίπεδα απόδοσης αλλά απαιτούσαν μεγάλης κλίμακας σύνολα δεδομένων εκπαίδευσης.

Το 2009 η συνεργασία της Microsoft με το Πανεπιστήμιο του Τορόντο συμπερέλαβε την IBM και την Google και όλες μαζί κατάφεραν να επιτύχουν μεγάλη αλλαγή στην ακρίβεια των μοντέλων. Μετά από πολλές σταθερές βελτιώσεις που έγιναν τις προηγούμενες δεκαετίες και την εφαρμογή της βαθιάς μάθησης το ποσοστό μείωσης λάθους λέξης άγγιξε το 30%. Οι ερευνητές υιοθέτησαν την ιδέα και άρχισαν να εντάσουν στη μοντελοποίηση της γλώσσας τη βαθιά μάθηση. Η διερεύνηση που έγινε από το 1980 μέχρι τις αρχές τους 2000 σε απλά μοντέλα αναγνώρισης ομιλίας αλλά και σε μοντέλα βαθιάς μορφής τεχνητών νευρωνικών δικτύων έδειξε ότι το μη ομοιόμορφο εσωτερικό-χειροποίητο μοντέλο Gaussian μείγματος / κρυφό μοντέλο Markov (GMM-HMM) υπερτερούσε σε απόδοση και αποτέλεσμα κάνοντας χρήση τεχνολογίας που βασίζεται σε παραγωγικά μοντέλα ομιλίας που εκπαιδεύονται διακριτικά.

Στη δεκαετία του 2010, η αναγνώριση ομιλίας, γνωστή και ως αναγνώριση φωνής διαφοροποιήθηκε σαφώς από την αναγνώριση ομιλητή και η ανεξαρτησία του ομιλητή θεωρήθηκε σημαντική ανακάλυψη. Μέχρι τότε, τα συστήματα απαιτούσαν περίοδο «προπόνησης». Το 2017 ερευνητές της Microsoft έφτασαν σε ένα ιστορικό ορόσημο ανθρώπινης ισοτιμίας της μεταγραφής ομιλίας τηλεφωνικής συνομιλίας στην ευρέως συγκριτική εργασία του Switchboard. Χρησιμοποιήθηκαν πολλαπλά μοντέλα βαθιάς μάθησης για τη βελτιστοποίηση της ακρίβειας αναγνώρισης ομιλίας. Το ποσοστό σφάλματος λήξης αναγνώρισης ομιλίας αναφέρθηκε ότι ήταν

τόσο χαμηλό όσο 4 επαγγελματίες άνθρωποι μεταγραφείς που εργάζονται μαζί στο ίδιο σημείο αναφοράς.

Πιο πρόσφατα, η εξέλιξη της βαθιάς μάθησης και των νευρωνικών δικτύων έφερε επανάσταση στην αναγνώριση ομιλίας. Τεχνολογίες όπως τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN) και τα μοντέλα Transformer έχουν επιτρέψει στα συστήματα αναγνώρισης ομιλίας να επιτύχουν σχεδόν ανθρώπινα επίπεδα ακρίβειας. Η εισαγωγή μοντέλων όπως το WaveNet της Google και το Whisper της OpenAI έχει βελτιώσει σημαντικά την ικανότητα των μηχανών να αναγνωρίζουν διάφορες προφορές, γλώσσες, ακόμη και θόρυβο υποβάθρου. Από σήμερα, η τεχνολογία αναγνώρισης ομιλίας ενσωματώνεται σε αμέτρητες εφαρμογές, από smartphones και εικονικούς βοηθούς μέχρι αυτοματοποίηση υπηρεσιών εξυπηρέτησης πελατών και ιατρική μεταγραφή.

Κοιτάζοντας μπροστά, το μέλλον της αναγνώρισης ομιλίας φαίνεται ακόμη πιο ελπιδοφόρο. Οι ερευνητές συνεχίζουν να εργάζονται για τη βελτίωση της αναγνώρισης ομιλίας σε θορυβώδη περιβάλλοντα, την ανάπτυξη συστημάτων που απαιτούν λιγότερη εκπαίδευση και τη βελτίωση της φωνητικής τεχνολογίας για ομιλητές με διαφορετικές προφορές ή προβλήματα ομιλίας. Επιπλέον, η σύνθεση ομιλίας και οι εξελίξεις στην επεξεργασία φυσικής γλώσσας (NLP) ανοίγουν το δρόμο για πιο ανθρώπινες αλληλεπιδράσεις μεταξύ ανθρώπων και μηχανών. Ενώ τα πρώιμα συστήματα αναγνώρισης ομιλίας ήταν αργά και άκαμπτα, τα σημερινά μοντέλα με βάση την τεχνητή νοημοσύνη προσφέρουν δυνατότητες μεταγραφής και αλληλεπίδρασης σε πραγματικό χρόνο, με μεγάλη ακρίβεια. Η πορεία της αναγνώρισης ομιλίας αντικατοπτρίζει την ευρύτερη εξέλιξη της τεχνητής νοημοσύνης, αναδεικνύοντας πόσο μακριά έχει φτάσει η τεχνολογία - και υποδηλώνοντας ακόμη μεγαλύτερες δυνατότητες για τα επόμενα χρόνια.

ΚΕΦΑΛΑΙΟ 3 CHATBOT

3.1 ΤΙ ΕΙΝΑΙ ΤΑ CHATBOT

Ένα chatbot είναι μια εφαρμογή λογισμικού που έχει σχεδιαστεί για να προσομοιώνει συνομιλίες που μοιάζουν με τις ανθρώπινες συνομιλίες με τους χρήστες, μέσω αλληλεπιδράσεων κειμένου ή φωνής[13][14][15]. Με βάση την τεχνητή νοημοσύνη (AI) και την επεξεργασία φυσικής γλώσσας (NLP), τα chatbot μπορούν να κατανοούν τα ερωτήματα των χρηστών, να τα επεξεργάζονται και να παρέχουν σχετικές απαντήσεις σε πραγματικό χρόνο. Χρησιμοποιούνται ευρέως στην εξυπηρέτηση πελατών, την εκπαίδευση, την υγειονομική περίθαλψη και το ηλεκτρονικό εμπόριο για την αυτοματοποίηση εργασιών, την απάντηση συχνών ερωτήσεων και την ενίσχυση της δέσμευσης των χρηστών. Τα σύγχρονα chatbot αξιοποιούν μοντέλα μηχανικής μάθησης, όπως οι μετασχηματιστές, για να βελτιώσουν την ικανότητά τους να κατανοούν το πλαίσιο και να παρέχουν ακριβείς απαντήσεις. Ενώ τα chatbot που βασίζονται σε κανόνες ακολουθούν προκαθορισμένα σενάρια, τα chatbot με τεχνητή νοημοσύνη μαθαίνουν από τις αλληλεπιδράσεις, καθιστώντας τα πιο προσαρμοστικά και έξυπνα. Παρά τις προόδους τους, παραμένουν προκλήσεις, όπως ο χειρισμός σύνθετων ερωτημάτων και η διατήρηση της ροής της συνομιλίας. Τα chatbot αντιπροσωπεύουν ένα σημαντικό βήμα προς την αλληλεπίδραση του ανθρώπου με τον υπολογιστή, προσφέροντας αποτελεσματικότητα σε πολλούς κλάδους αλλά και μια ιδιαιτερότητα.

3.2 ΠΩΣ ΣΥΝΔΕΕΤΑΙ ΤΟ CHATBOT ΜΕ ΤΗΝ ΑΝΑΓΝΩΡΙΣΗ ΟΜΙΛΙΑΣ

Τα chatbots και η αναγνώριση ομιλίας είναι δύο αλληλένδετες τεχνολογίες που συνεργάζονται για την απρόσκοπτη αλληλεπίδραση του ανθρώπου με τον υπολογιστή. Ενώ τα

chatbots έχουν σχεδιαστεί για να προσομοιώνουν συνομιλίες μέσω κειμένου ή φωνής, η αναγνώριση ομιλίας επιτρέπει στους χρήστες να επικοινωνούν με τα chatbots χρησιμοποιώντας την ομιλία. Αυτή η ενσωμάτωση παρατηρείται σε συστήματα με φωνητική ενεργοποίηση, όπως οι εικονικοί βοηθοί.

Στον πυρήνα αυτής της σύνδεσης βρίσκεται η διαδικασία μετατροπής της ομιλίας σε κείμενο. Όταν ένας χρήστης μιλάει σε ένα chatbot, το σύστημα αναγνώρισης ομιλίας καταγράφει πρώτα την ηχητική είσοδο, και την επεξεργάζεται χρησιμοποιώντας προηγμένους αλγορίθμους για τον εντοπισμό φωνημάτων, λέξεων και προτάσεων. Αυτό το μεταγραμμένο κείμενο διαβιβάζεται στη συνέχεια στο chatbot, το οποίο χρησιμοποιεί επεξεργασία φυσικής γλώσσας (NLP) για να κατανοήσει το ερώτημα του χρήστη. Μόλις το chatbot παράγει μια απάντηση, μπορεί είτε να την εμφανίσει ως κείμενο είτε να την μετατρέψει ξανά σε ομιλία, ολοκληρώνοντας έτσι τη διαδικασία.

Η συνέργεια μεταξύ των chatbot και της αναγνώρισης ομιλίας βελτιώνει την εμπειρία του χρήστη, καθιστώντας τις αλληλεπιδράσεις πιο φυσικές και προσιτές. Για παράδειγμα, στην εξυπηρέτηση πελατών, τα chatbot με δυνατότητα φωνής μπορούν να χειριστούν ερωτήματα χωρίς να απαιτείται από τους χρήστες να πληκτρολογούν, κάτι που είναι ιδιαίτερα χρήσιμο για σενάρια ώπου η χρήση των χεριων δεν είναι εφικτή ή για άτομα με αναπηρίες. Ωστόσο, προκλήσεις όπως η ακριβής αναγνώριση διαφορετικών προφορών, ο χειρισμός του θορύβου του περιβάλλοντος και η κατανόηση σύνθετων ερωτημάτων παραμένουν τομείς συνεχιζόμενης έρευνας.

Η ενσωμάτωση των chatbot και της αναγνώρισης ομιλίας αποτελεί σημαντική πρόοδο στην επικοινωνία με βάση την τεχνητή νοημοσύνη, επιτρέποντας πιο αποτελεσματικές αλληλεπιδράσεις μεταξύ ανθρώπου και υπολογιστή. Καθώς οι δύο τεχνολογίες συνεχίζουν να εξελίσσονται, οι συνδυασμένες δυνατότητές τους πιθανότατα θα επεκταθούν, ανοίγοντας το δρόμο για εξυπνότερα και πιο ευέλικτα συστήματα.

3.3 ΤΑ ΠΙΟ ΓΝΩΣΤΑ, ΣΥΓΧΡΟΝΑ ΚΑΙ ΙΚΑΝΑ CHATBOT

Η Siri[16], που αναπτύχθηκε από την Apple Inc., παρουσιάστηκε το 2011 ως ένας εικονικός βοηθός με φωνητική ενεργοποίηση. Δημιουργήθηκε από μια ομάδα με επικεφαλής τους Dag Kittlaus, Tom Gruber και Adam Cheyer και σκοπός της Siri είναι να βοηθά τους χρήστες σε

εργασίες όπως ο καθορισμός υπενθυμίσεων, η αποστολή μηνυμάτων και η παροχή πληροφοριών μέσω αλληλεπιδράσεων φυσικής γλώσσας. Ενσωματωμένη σε συσκευές της Apple, όπως iPhones, iPads και Mac, η Siri χρησιμοποιείται ευρέως σε προσωπικά και επαγγελματικά περιβάλλοντα για ευκολία hands-free. Η ικανότητά της να συνδέεται με άλλες υπηρεσίες της Apple την καθιστά ένα ευέλικτο εργαλείο για τη διαχείριση καθημερινών δραστηριοτήτων, την απάντηση ερωτημάτων και τον έλεγχο έξυπνων οικιακών συσκευών.

Η Amazon λάνσαρε την Alexa[18] το 2014, η οποία αναπτύχθηκε από την ομάδα Lab126. Η Alexa είναι ένα φωνητικά ελεγχόμενο chatbot που έχει σχεδιαστεί για να εκτελεί εργασίες όπως η αναπαραγωγή μουσικής, η παροχή ενημερώσεων για τον καιρό και ο έλεγχος έξυπνων οικιακών συσκευών. Βασισμένη στην τεχνητή νοημοσύνη της Amazon, η Alexa χρησιμοποιείται κυρίως στα έξυπνα ηχεία Echo, αλλά έχει επεκταθεί και σε άλλες συσκευές, όπως αυτοκίνητα και wearables. Η ενσωμάτωσή της με εφαρμογές και υπηρεσίες τρίτων την καθιστά κεντρικό κόμβο για τον οικιακό αυτοματισμό, την ψυχαγωγία και την παραγωγικότητα, καλύπτοντας τόσο προσωπικές όσο και εμπορικές ανάγκες.

Η OpenAI[17], που ιδρύθηκε από τον Elon Musk, τον Sam Altman και άλλους, κυκλοφόρησε το ChatGPT το 2022. Αυτό το chatbot με τεχνητή νοημοσύνη έχει σχεδιαστεί για να παράγει απαντήσεις κειμένου που μοιάζουν με ανθρώπινο κείμενο, να βοηθά στη δημιουργία περιεχομένου και να παρέχει πληροφορίες σε διάφορα θέματα. Το ChatGPT χρησιμοποιείται στην εκπαίδευση, την υποστήριξη πελατών και τις δημιουργικές βιομηχανίες, προσφέροντας λύσεις όπως η διδασκαλία, η σύνταξη μηνυμάτων ηλεκτρονικού ταχυδρομείου και ο καταιγισμός ιδεών. Η ευελιξία του και η ικανότητά του να προσαρμόζεται σε διάφορα πλαίσια το καθιστούν πολύτιμο εργαλείο για άτομα και επιχειρήσεις που αναζητούν αποτελεσματική επικοινωνία με τεχνητή νοημοσύνη.

Το Google Assistant[19], που λανσαρίστηκε το 2016 από την Google, είναι ένα chatbot με τεχνητή νοημοσύνη που έχει σχεδιαστεί για να παρέχει εξατομικευμένη βοήθεια μέσω φωνής και κειμένου. Δημιουργήθηκε από την ομάδα μηχανικών της Google και βοηθά τους χρήστες να διαχειρίζονται προγράμματα, να κάνουν αναζήτηση στο διαδίκτυο και να ελέγχουν έξυπνες συσκευές. Διαθέσιμο σε τηλέφωνα Android, έξυπνα ηχεία και wearables, το Google Assistant χρησιμοποιείται ευρέως σε σπίτια, γραφεία και οχήματα. Η ενσωμάτωσή του στο οικοσύστημα της Google, συμπεριλαμβανομένων των Χαρτών και του Ημερολογίου, το καθιστά ένα ισχυρό εργαλείο για την ενίσχυση της παραγωγικότητας και την απλούστευση των καθημερινών εργασιών.

3.4 Η ΙΣΤΟΡΙΑ ΚΑΙ ΕΞΕΛΙΞΗ ΤΩΝ CHATBOT

Η έννοια των chatbots χρονολογείται από τα μέσα του 20ου αιώνα, σηματοδοτώντας την αρχή ενός συναρπαστικού ταξιδιού στην τεχνητή νοημοσύνη (AI) και την αλληλεπίδραση ανθρώπου-υπολογιστή. Το πρώτο chatbot που δημιουργήθηκε ποτέ ήταν το ELIZA[20], το οποίο αναπτύχθηκε το 1966 από τον Joseph Weizenbaum, έναν επιστήμονα πληροφορικής στο Τεχνολογικό Ινστιτούτο της Μασαχουσέτης (MIT). Το ELIZA σχεδιάστηκε για να προσομοιώνει έναν ψυχοθεραπευτή, χρησιμοποιώντας μια τεχνική που ονομάζεται αντιστοίχιση προτύπων για να ανταποκρίνεται στις εισόδους του χρήστη. Αν και υποτυπώδες για τα σημερινά δεδομένα, το ELIZA μπορούσε να εμπλέκει τους χρήστες σε φαινομενικά ουσιαστικές συζητήσεις, αναδιατυπώνοντας τις δηλώσεις τους σε ερωτήσεις. Για παράδειγμα, αν ένας χρήστης πληκτρολογούσε: "Νιώθω λυπημένος", το ELIZA θα μπορούσε να απαντήσει: "Γιατί νιώθεις λυπημένος;". Αυτή η απλότητα κατέστησε το ELIZA ένα πρωτοποριακό επίτευγμα, αποδεικνύοντας τη δυνατότητα των μηχανών να μιμούνται την ανθρώπινη συνομιλία.

Μετά το ELIZA, στις δεκαετίες του 1970 και 1980 αναπτύχθηκαν τα chatbots που βασίζονται σε κανόνες, τα οποία βασίζονταν σε προκαθορισμένα σενάρια και δέντρα αποφάσεων. Ένα αξιοσημείωτο παράδειγμα ήταν το PARRY[20], που δημιουργήθηκε το 1972 από τον ψυχίατρο Kenneth Colby. Το PARRY σχεδιάστηκε για να προσομοιώνει ένα άτομο με παρανοϊκή σχιζοφρένεια και χρησιμοποιούσε ένα πιο σύνθετο σύνολο κανόνων από το ELIZA για τη δημιουργία απαντήσεων. Παρόλο που αυτά τα πρώιμα chatbots είχαν περιορισμένες δυνατότητες, έθεσαν τα θεμέλια για μελλοντικές εξελίξεις εξερευνώντας τις δυνατότητες της επεξεργασίας φυσικής γλώσσας (NLP) και της τεχνητής νοημοσύνης.

Κατά τη διάρκεια αυτής της περιόδου, τα chatbots χρησιμοποιούνταν κυρίως για ακαδημαϊκή έρευνα και πειραματισμό. Ωστόσο, άρχισαν να αναδεικνύονται οι δυνατότητές τους για πρακτικές εφαρμογές, ιδίως σε τομείς όπως η εξυπηρέτηση πελατών και η εκπαίδευση. Παρά τους περιορισμούς τους, αυτά τα πρώιμα συστήματα έδειξαν ότι οι μηχανές μπορούσαν να συμμετέχουν σε βασικές συνομιλίες, προκαλώντας το ενδιαφέρον για περαιτέρω ανάπτυξη.

Η δεκαετία του 1990 σηματοδότησε ένα σημείο καμπής στην εξέλιξη των chatbots, καθώς οι εξελίξεις στην υπολογιστική ισχύ και την τεχνητή νοημοσύνη οδήγησαν σε πιο εξελιγμένα συστήματα. Μια από τις σημαντικότερες εξελίξεις ήταν το ALICE[21] (Artificial Linguistic Internet Computer Entity), που δημιουργήθηκε το 1995 από τον Richard Wallace. Η ALICE χρησιμοποιούσε μια πιο προηγμένη μορφή αντιστοίχισης προτύπων που ονομαζόταν AIML (Artificial Intelligence Markup Language), επιτρέποντάς της να χειρίζεται ένα ευρύτερο φάσμα

θεμάτων συνομιλίας. Αν και το ALICE εξακολουθούσε να βασίζεται σε προκαθορισμένους κανόνες, αντιπροσώπευε ένα σημαντικό βήμα προς τα εμπρός στην τεχνολογία chatbot.

Κατά τη διάρκεια αυτής της δεκαετίας, τα chatbots άρχισαν να βρίσκουν εμπορικές εφαρμογές, ιδίως στην υποστήριξη πελατών και στις διαδικτυακές υπηρεσίες. Για παράδειγμα, το SmarterChild, που ξεκίνησε το 2001 από την ActiveBuddy, έγινε ένα δημοφιλές chatbot σε πλατφόρμες άμεσων μηνυμάτων όπως το AOL Instant Messenger και το MSN Messenger. Το SmarterChild μπορούσε να παρέχει ενημερώσεις για τον καιρό, αποτελέσματα αθλητικών αγώνων και ακόμη και να συμμετέχει σε περιστασιακές συνομιλίες, καθιστώντας το ένα από τα πρώτα chatbots που πέτυχαν ευρεία δημοτικότητα.

Τη δεκαετία του 2000 εμφανίστηκαν οι εικονικοί βοηθοί, οι οποίοι συνδύαζαν την τεχνολογία chatbot με την αναγνώριση ομιλίας και άλλες δυνατότητες τεχνητής νοημοσύνης. Ένα από τα πιο αξιοσημείωτα παραδείγματα ήταν η Siri, που αναπτύχθηκε από την Apple και παρουσιάστηκε το 2011. Η Siri σχεδιάστηκε για να εκτελεί εργασίες όπως ο καθορισμός υπενθυμίσεων, η αποστολή μηνυμάτων και η απάντηση σε ερωτήσεις μέσω αλληλεπιδράσεων φυσικής γλώσσας. Η ενσωμάτωσή της στο οικοσύστημα συσκευών της Apple την κατέστησε ένα πρωτοποριακό εργαλείο για την προσωπική παραγωγικότητα.

Περίπου την ίδια εποχή, ο Google Assistant και η Alexa της Amazon εισήλθαν στην αγορά, προωθώντας περαιτέρω τις δυνατότητες των εικονικών βοηθών. Αυτά τα συστήματα αξιοποίησαν τη μηχανική μάθηση και μεγάλα σύνολα δεδομένων για να βελτιώσουν την κατανόηση της πρόθεσης και του πλαισίου του χρήστη. Εισήγαγαν επίσης την έννοια της ενσωμάτωσης του έξυπνου σπιτιού, επιτρέποντας στους χρήστες να ελέγχουν συσκευές όπως φώτα, θερμοστάτες και συστήματα ασφαλείας μέσω φωνητικών εντολών.

Η δεκαετία του 2010 υπήρξε μάρτυρας μιας επανάστασης στην τεχνολογία chatbot, η οποία καθοδηγήθηκε από τις εξελίξεις στη μηχανική μάθηση, τη βαθιά μάθηση και την επεξεργασία φυσικής γλώσσας. Μια από τις σημαντικότερες ανακαλύψεις ήταν η ανάπτυξη μοντέλων βασισμένων σε μετασχηματιστές, όπως η σειρά GPT (Generative Pre-trained Transformer) της OpenAI. Αυτά τα μοντέλα, που εκπαιδεύτηκαν σε τεράστιες ποσότητες δεδομένων κειμένου, μπορούσαν να παράγουν απαντήσεις που έμοιαζαν με ανθρώπινες και να χειρίζονται πολύπλοκες συνομιλίες με αξιοσημείωτη ακρίβεια.

Το 2022, η OpenAI κυκλοφόρησε το ChatGPT, ένα chatbot βασισμένο στην αρχιτεκτονική GPT-3.5. Το ChatGPT απέκτησε γρήγορα δημοτικότητα για την ικανότητά του να βοηθά σε εργασίες όπως η δημιουργία περιεχομένου, η κωδικοποίηση και η διδασκαλία. Σε αντίθεση με τα προηγούμενα chatbots, το ChatGPT μπορούσε να κατανοεί τα συμφραζόμενα, να παράγει συνεκτικές παραγράφους και να προσαρμόζεται σε διάφορα στυλ συνομιλίας. Αυτό

σηματοδότησε μια νέα εποχή στην τεχνολογία chatbot, όπου τα συστήματα με τεχνητή νοημοσύνη μπορούσαν να παρέχουν ουσιαστικές και συνειδητοποιημένες στο πλαίσιο αλληλεπιδράσεις.

Κατά τη διάρκεια αυτής της περιόδου, τα chatbots έγιναν επίσης αναπόσπαστο στοιχείο σε κλάδους όπως η υγειονομική περίθαλψη, η χρηματοδότηση και το ηλεκτρονικό εμπόριο. Για παράδειγμα, το Woebot, ένα chatbot ψυχικής υγείας που αναπτύχθηκε το 2017, χρησιμοποίησε τεχνικές γνωστικής-συμπεριφορικής θεραπείας για την παροχή συναισθηματικής υποστήριξης. Ομοίως, το Erica της Bank of America βοήθησε τους πελάτες να διαχειριστούν τα οικονομικά τους μέσω εξατομικευμένων συστάσεων και πληροφοριών.

Σήμερα, τα chatbots είναι πανταχού παρόντα, τροφοδοτώντας τα πάντα, από πλατφόρμες εξυπηρέτησης πελατών έως εκπαιδευτικά εργαλεία. Έχουν εξελιχθεί από απλά συστήματα βασισμένα σε κανόνες σε εξελιγμένους βοηθούς με τεχνητή νοημοσύνη, ικανούς να χειρίζονται σύνθετες εργασίες. Τα σύγχρονα chatbots αξιοποιούν τεχνολογίες όπως η ανάλυση συναισθήματος, η ανίχνευση συναισθημάτων και η πολύγλωσση υποστήριξη για τη βελτίωση των εμπειριών των χρηστών.

Κοιτάζοντας μπροστά, το μέλλον των chatbots είναι πιθανό να διαμορφωθεί από την πρόοδο της γενικής τεχνητής νοημοσύνης, της κατανόησης του πλαισίου και της συναισθηματικής νοημοσύνης. Οι ερευνητές διερευνούν τρόπους για να καταστήσουν τα chatbots πιο ενσυναισθητικά και ικανά να οικοδομήσουν μακροχρόνιες σχέσεις με τους χρήστες. Επιπλέον, η ενσωμάτωση των chatbots με την επαυξημένη πραγματικότητα (AR) και την εικονική πραγματικότητα (VR) θα μπορούσε να ανοίξει νέες δυνατότητες για καθηλωτικές αλληλεπιδράσεις.

Η ιστορία και η εξέλιξη των chatbots αντικατοπτρίζουν την αξιοσημείωτη πρόοδο της τεχνητής νοημοσύνης και της αλληλεπίδρασης ανθρώπου-υπολογιστή. Από την απλή αντιστοίχιση μοτίβων του ELIZA στην προηγμένη γλωσσική παραγωγή του ChatGPT, τα chatbots έχουν διανύσει πολύ δρόμο σε σχετικά σύντομο χρονικό διάστημα. Καθώς η τεχνολογία συνεχίζει να εξελίσσεται, τα chatbots είναι έτοιμα να γίνουν ακόμη πιο έξυπνα, ευέλικτα και ενσωματωμένα στην καθημερινή μας ζωή. Η πορεία τους από ακαδημαϊκά πειράματα σε απαραίτητα εργαλεία αναδεικνύει τις μετασχηματιστικές δυνατότητες της τεχνητής νοημοσύνης και την ικανότητά της να αναδιαμορφώνει τον τρόπο με τον οποίο επικοινωνούμε με τις μηχανές.

ΚΕΦΑΛΑΙΟ 4 ΠΡΟΕΤΟΙΜΑΣΙΑ

Όταν ξεκίνησα την εργασία, έπρεπε να αποφασίσω το λειτουργικό, τη γλώσσα και τα εργαλεία που θα χρησιμοποιήσω. Ήταν πολύ σημαντικό να ξέρω τι ακριβώς θέλω να υλοποιήσω για να υπάρχει μια ροή από την εκκίνηση της εργασίας μέχρι το τέλος.

4.1 ΕΞΟΠΛΙΣΜΟΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΕ

Ο εξοπλισμός που χρησιμοποίθηκε αποτελείτε από ένα σταθερό υπολογιστή με τα περιφεριακά του, οθόνη, πληκτρολόγιο, ποντίκι και μικρόφωνο με ένα κανάλι. Ο υπολογοστής μου τρέχει το λειτουργικό Windows 11, κατέβασα το Oracle VM Virtualbox και εγκατέστησα τα Linux Ubuntu 22.04 για να υλοποιήσω τον κώδικα. Στην πορία έτρεξα τον κώδικα και σε λάπτοπ με εγκατεστημένο το λειτουργικό των Linux. Οπότε για να μπορέσει κάποιος να υλοποιήσει την εργασία πρέπει να έχει έναν σταθερό υπολογιστή και τα περιφεριακά του με ένα καλό σχετικά μικρόφωνο ή ένα λάπτοπ με ξεχωριστό μικρόφωνο στην καλύτερη περίπτωση.

4.2 ΥΛΟΠΟΙΗΣΗ

Το λειτουργικό σύστημα που επέλεξα ήταν τα Linux (Ubuntu 24.04) ένα ανοιχτού τύπου λειτουργικό σύστημα. Είναι πιο ελαφρύ και γρήγορο από τα Windows ώστε να μπορούν να το τρέξουν και υπολογιστές που δεν διαθέτουν δυνατά χαρακτηριστικά. Διατίθεται δωρεάν και είναι πολύ ασφαλές αφού προσφέρει διορθώσεις και βελτιώσεις ως προς το security, καλύπτοντας τις περισσότερες περιπτώσεις απειλών που μπορεί να παρουσιαστούν, χωρίς να χρειαστεί να χρησιμοποιηθεί κάποιο anti-virus. Έχει ένα καλό community στο οποίο συζητούνται και λύνονται προβλήματα που συναντώνται σε διάφορες περιπτώσεις. Μαζί με την εγκατάσταση των Linux

μπορεί να γίνει λήψη και του Libre Office ένα πολύ καλό word processing program για το άνοιγμα και την αποστολή αρχείων. Ο μεγάλος όγκος δωρεάν εργαλείων και η μεγάλη συμβασιμότητα στο hardware είναι ένας πολύ δυνατός λόγος για να επιλέξει κάποιος το λειτουργικό αυτό.

Ο λόγος που προτίμησα να χρησιμοποιήσω τα Linux ως λειτουργικό σύστημα ήταν για να μάθω καλύτερα το λειτουργικό καθώς δεν το είχα χρησιμοποιήσει για κάτι περισσότερο από τα βασικά. Ήθελα να βελτιώσω τις γνώσεις μου, μέσα από την πρόκληση υλοποίησης της εργασίας. Παρόλο που είναι πιο ελαφρύ, γρήγορο και περιέχει πολλά εργαλεία απευθύνεται περισσότερο σε προγραμματιστές.

Η γλώσσα που επιλέχθηκε είναι η python. Μια πιο σχετικά νέα γλώσσα πολύ φιλική ως προς το χρήστη. Η python είναι μια υψηλού επιπέδου γλώσσα και η δομή των εντολών θυμίζουν την ανάγνωση προτάσεων από ένα βιβλίο. Για κάποιον που δεν γνωρίζει από προγραμματισμό η φυσική κατανόηση της την καθιστά ιδανική γλώσσα για να ξεκινήσει το ταξίδι του εκεί. Είναι γλώσσα υψηλού επιπέδου, γρήγορη και χρησιμοποιείτε σχεδόν σε όλους τους τομείς. Διακρίνεται κυρίως στο data science αλλά και στη δημιουργία και εφαρμογή μοντέλων AI.

4.3 ΕΡΓΑΛΕΙΑ ΚΑΙ ΒΙΒΛΙΟΘΗΚΕΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

Για να ανοίξουμε το αρχείο και να το τρέξουμε πρέπει να γίνει εγκατάσταση ορισμένων βιβλιοθηκών μέσα από το terminal. Το terminal είναι ο τρόπος επικοινωνίας του χρήστη με τον υπολογιστή. Το λειτουργικό από μόνο του μπορεί να έχει εγκατεστημένες από πριν κάποιες από τις βιβλιοθήκες που θα χρησιμοποιηθούν. Παρακάτω θα γίνει αναφορά και επεξήγηση όλων των εργαλείων και βιβλιοθηκών των οποίων θα γίνει χρήση καθώς και ο τρόπος εγκατάστασης τους μέσα από το terminal. Η δομή τους θα αποτελείτε από το όνομα, τις εντολές για την εγκατάσταση και την επεξήγηση τους.

- Speech Recognition
- Py audio (ξεχωριστά)
- Unicodedata
- Docxtpl
- Smptlib

➉ Speech Recognition

Εγκατάσταση της βιβλιοθήκης SpeechRecognition:

< pip install SpeechRecognition >

Η βιβλιοθήκη SpeechRecognition είναι ένα ισχυρό εργαλείο για την αναγνώριση ομιλίας στη γλώσσα Python. Παρέχει μια απλή διεπαφή που επιτρέπει στους προγραμματιστές να χρησιμοποιούν διάφορες μηχανές ASR (Automatic Speech Recognition), όπως τα Google Speech Recognition API, IBM Watson, Microsoft Bing Voice Recognition και Sphinx. Η βιβλιοθήκη υποστηρίζει πολλαπλές πηγές ήχου και επιτρέπει τη μετατροπή ηχητικών δεδομένων σε κείμενο, καθιστώντας την ιδανική για εφαρμογές φωνητικών βοηθών, μεταγραφής και φωνητικού ελέγχου. Διαθέτει αρκετές γλώσσες προ εγκατεστημένες και σε συνδυασμό με την απλότητα του το έκανε την τέλεια επιλογή. Η έκδοση της python πρέπει να είναι 3.8+ για να λειτουργήσει αλλά καλό θα ήταν να είναι στην τελευταία έκδοση της.

➉ PyAudio

Εγκατάσταση εξαρτήσεων για το PyAudio:

< sudo apt update >

< sudo apt install portaudio19-dev python3-pyaudio >

Εγκατάσταση της βιβλιοθήκης PyAudio μέσω pip:

< pip install pyaudio >

Η PyAudio είναι μια βιβλιοθήκη που παρέχει μια Pythonic διεπαφή για τη χρήση της πλατφόρμας PortAudio, η οποία επιτρέπει την εγγραφή και αναπαραγωγή ήχου σε πραγματικό χρόνο. Η PyAudio χρησιμοποιείται κυρίως για τη λήψη ήχου από μικρόφωνο και την επεξεργασία του για εφαρμογές όπως αναγνώριση ομιλίας, ηχητική ανάλυση και φωνητικός έλεγχος. Επειδή βασίζεται στο PortAudio, απαιτεί την εγκατάσταση εξαρτήσεων για να λειτουργήσει σωστά.

- **Unicodedata**

Εγκατάσταση της βιβλιοθήκης Unicodedata:

<pip install unicode>

Η συγκεκριμένη βιβλιοθήκη έχει πρόσβαση στα χαρακτηριστικά των χαρατκήρων και την αλλαγή τους.Εδώ χρησιμοποιείται για να αφαιρέσει τους τόνους που έχουν οι λέξεις. Η χρήση της βιβλιοθήκες έγινε για την αποφυγή ορισμένων λαθών που προέκυπταν κατά τη μετατροπή του ήχου σε κείμενο από κακό πιθανότατα μικρόφωνο.

- **Docxtpl**

Εγκατάσταση της βιβλιοθήκης Docxtpl:

<pip install docxtpl>

Το Docxtpl χρειάζεται για να γίνει η αντικατάσταση των λέξεων που θα λάβει το Speech Recognition από τον χρήστη μέσω του μικροφώνου και θα τις αντικαταστήσει στο αρχείο word που έχω δημιουργήσει για τα έγγραφα ώστε να συμπληρωθούν αυτόματα.

- **Smtplib**

Εγκατάσταση της βιβλιοθήκης smtplib:

<sudo apt install smtplib>

Η βιβλιοθήκη smtplib χρησιμοποιείται για την αποστολή email μέσω του πρωτοκόλλου SMPT (simple mail transfer Protocol). Επιτρέπει σε ένα πρόγραμμα να συνδεθεί σε έναν mail server, να στείλει μηνύματα και να κλείσει τη σύνδεση. Η παρουσία του παρατηρείται στις υπηρεσίες Gmail και Outlook και η χρήση του απαιτεί ταυτοποίηση.

Για να μπορέσω να πραγματοποιήσω αποστολή του αρχείου με την ολοκλήρωση της διαδικασίας, χρησιμοποίησα την βιβλιοθήκη smtplib. Η βιβλιοθήκη αυτή χρησιμοποιεί την τυπική θύρα email. Η ενεργοποίηση της λειτουργίας 2 Factor Authentication (2FA) είναι ένα απαραίτητο βήμα για τη λειτουργία του. Στη συνέχεια πρέπει να γίνει δημιουργία κωδικού εφαρμογής (app password). Αυτός ο κωδικός θα χρησιμοποιηθεί για να γίνει η ταυτοποίηση.

ΚΕΦΑΛΑΙΟ 5 ΠΕΡΙΓΡΑΦΗ, ΕΚΤΕΛΕΣΗ ΚΑΙ ΠΡΟΒΛΗΜΑΤΑ

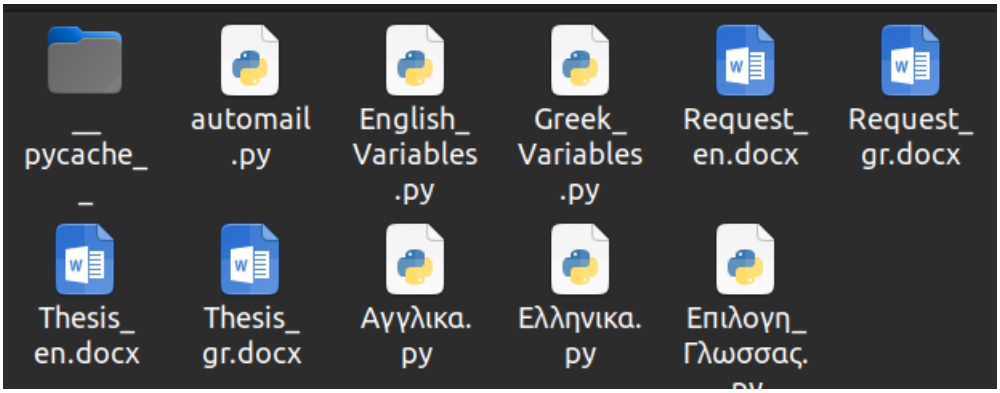
5.1 Γενική περιγραφή

Το πρόγραμμα αυτό δημιουργήθηκε για να εισάγουμε πληροφορίες στον υπολογιστή, για τη συμπλήρωση ενός εντύπου, χρησιμοποιώντας τη φωνή μας αντί του παραδοσιακού τρόπου εισαγωγής πληροφοριών, δηλαδή το πληκτρολόγιο. Με το πέρας της διαδικασίας, ο χρήστης θα έχει αποθηκεύσει και στείλει το συμπληρωμένο αρχείο σε κάποιο email. Για να λειτουργήσει, δημιουργήθηκαν έξι αρχεία με κατάληξη .py και άλλα τέσσερα με κατάληξη .docx.

Στο προηγούμενο κεφάλαιο έγινε περιγραφή των βιβλιοθηκών που χρησιμοποιήθηκαν για την λειτουργία του προγράμματος, αλλά και του τρόπου εγκατάστασης τους. Στο κεφάλαιο αυτό θα ασχοληθούμε με τον ίδιο τον κώδικα εξονυχιστικά για την πλήρη κατανόησή του. Έχουν δημιουργηθεί έξι αρχεία σε γλώσσα Python που καλύπτουν την εκκίνηση του προγράμματος, την επιλογή της γλώσσας, την επιλογή του έντυπου αντικατάστασης και την αποστολή του τελικού αρχείου σε email. Στο πρώτο αρχείο <Επιλογή_Γλώσσας.py> πραγματοποιείτε η επιλογή γλώσσας. Το δεύτερο και το τρίτο < Αγγλικά.py > και < Ελληνικά.py > είναι υπεύθυνα για το μεγαλύτερο μέρος των εργασιών του κώδικα και είναι παρόμοια μεταξύ τους. Τα αρχεία τέσσερα και πέντε < Greek_Variables.py > και < English_Variables.py > περιέχουν τις μεταβλητές και τις λίστες που θα χρησιμοποιήσει το αρχείο τρια και τέσσερα. Τέλος, το αρχείο έξι < automail.py > είναι υπεύθυνο για την αποστολή του αρχείου σε κάποιο mail. Τα έντυπα που επιθυμεί ο φοιτητής να συμπληρώσει βρίσκονται στον ίδιο φάκελο με αυτά τα έξι αρχεία ώστε να μπορέσουν να τα ‘δουν’ και να πραγματοποιήσουν την αυτόματη συμπλήρωση. Η κατάληξη τους είναι διαφορετική εφόσον είναι διαφορετικού τύπου αρχεία.

Το αρχείο < __pycache__ > δημιουργείτε αυτόματα από το διερμηνευτή όταν εκτελούμε ή εισάγουμε κάποιο αρχείο για τη μεταγλώττιση του. Το αρχείο αλλάζει κατάληξη από .py (πηγαίος κώδικας) σε .pyc (bytecode) που είναι η δυαδική μορφή του αρχείου την οποία καταλαβαίνει ο υπολογιστής και πραγματοποιείται η εκτέλεση του κώδικα. Επειδή το μεταγλωττισμένο αρχείο

αποθηκεύεται, την επόμενη φορά που θα εκτελεστεί το αρχείο αυτό, θα εκτελεστεί γρηγορότερα διότι η μεταγλώττιση θα είναι πλέον περιττή. Αυτός είναι και ένας λόγος που η Python είναι μια γρήγορη γλώσσα προγραμματισμού.



Εικόνα 1. Τα αρχεία που χρειάζονται.

5.2 Σύντομη επεξήγηση περιεχομένων

Κατά την εκτέλεση του κώδικα το πρώτο αρχείο που θα ανοίξει είναι το Επιλογή_Γλώσσας.py στο οποίο γίνεται η επιλογή της γλώσσας από το χρήστη με εκτύπωση μηνυμάτων για να τον κατευθύνει. Η επιλογή της γλώσσας θα γίνει μέσω της ομιλίας και θα μας οδηγήσει στο επόμενο κατάλληλο αρχείο. Σε περίπτωση που ο χρήστης δεν μιλήσει, επιστρέφει μια μεταβλητή NONE και με μια συνθήκη ελέγχου επιλέγεται η ελληνική γλώσσα, σε αντίθετη περίπτωση που ο χρήστης επιθυμεί τα αγγλικά θα πρέπει να πει την λέξη English. Αυτό γίνεται επειδή η συνάρτηση που έχει τοποθετηθεί λειτουργεί μόνο για τα αγγλικά ως γλώσσα και τα ελληνικά μπορεί να οδηγήσουν σε εσφαλμένη λήψη από το μικρόφωνο. Υπάρχει παράμετρος που δέχεται τη λέξη “English” για τα αγγλικά μόνο ως επιλογή ή το “NONE” για τα ελληνικά. Για οποιαδήποτε άλλη απάντηση εμφανίζεται μήνυμα και επαναλαμβάνεται η διαδικασία. Για κάθε απάντηση που λαμβάνει, μετατρέπεται η γραμματοσειρά σε κεφαλαία και στη συνέχεια γίνεται έλεγχος. Ο χρήστης έχει τη δυνατότητα να τερματήσει το πρόγραμμα οποιαδήποτε στιγμή με τη λέξη “τερματισμός” ή “close” αντίστοιχα.

Όταν λάβει κάποια αποδεκτή απάντηση, περνάει στο δεύτερο στάδιο που είναι η κλήση των δύο αρχείων Ελληνικά.py και Αγγλικά.py. Ως παράδειγμα θα πάρω την περίπτωση των ελληνικών. Το αρχείο Ελληνικά.py εκτελείτε με το κάλεσμα μιας συνάρτησης από το αρχείο

Επιλογή_Γλώσσας.py. Στη συνάρτηση αυτή αρχικοποιούνται κάποιες μεταβλητές που θα χρησιμοποιηθούν στη συνέχεια και τις δηλώνω με τέτοιο τρόπο ώστε όλες οι συναρτήσεις να τις βλέπουν και να τις χρησιμοποιούν. Σε αυτό το σημείο ο χρήστης θα ερωτηθεί για τα διαθέσιμα έντυπα που μπορεί να συμπληρώσει. Ο χρήστης θα εκφωνήσει την επιλογή του και εφόσον ταιριάζει με τις μεταβλητές στις συνθήκες, θα γίνει αποδεκτή. Θα δωθεί μια λίστα με αντικείμενα προς αντικατάσταση στο word αρχείο και μια μεταβλητή θα την κρατήσει αποθηκευμένη. Η λίστα αυτή εμπεριέχεται μαζί με άλλες λίστες που είναι αρχικοποιημένες και έτοιμες για χρήση στο αρχείο Greek_Variables.py και English_Variables.py για τα αγγλικά. Η ίδια συνάρτηση διαθέτει και μια ακόμα συνθήκη η οποία επαναλαμβάνει όλη την διαδικασία ξανά από την αρχή με την ίδια γλώσσα.

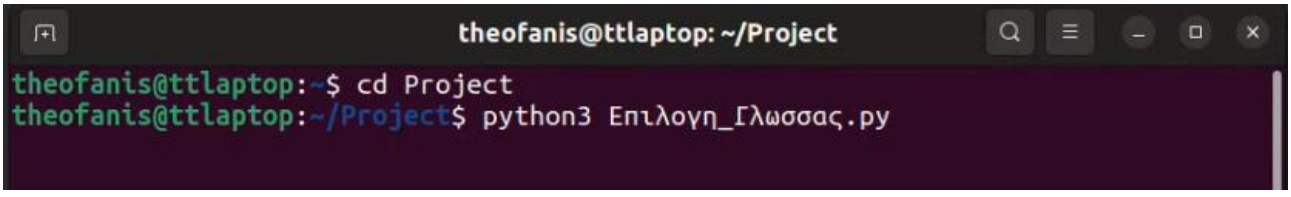
Η επόμενη συνάρτηση είναι αυτή που αλληλεπιδρά με το χρήστη πιο πολύ. Μέσω μιας δομής επανάληψης κάθε ένα από τα στοιχεία της λίστας περνάνε από μια επεξεργασία. Παρακάτω θα εξηγήσω την συνάρτηση που επεξεργάζεται τα αντικείμενα ένα ένα με τη φορά της λίστας. Στη συνέχεια με την όλοκλήρωση του βρόχου, θα συμπληρωθεί αυτόματα η ημερομηνία που είναι ρυθμισμένη στο σύστημα με κατάλληλη βιβλιοθήκη που δηλώνεται στο πάνω μέρος του αρχείου που περιέχει την εντολή ή θα τη δώσει ο χρήστης φωνητικά. Αφού ολοκληρωθεί και αυτό στη συνέχεια θα γίνει επεξεργασία με μια εμφολευμένη δομή επανάληψης καθώς και συνθήκης if-else για τον τρόπο που θα εκτυπώνονται ορισμένα από τα αντικείμενα στην περίπτωση μόνο που ο χρήστης επιλέξει την αίτηση. Σε αυτό το σημείο έχουν εμφανιστεί στο χρήστη όλα τα αντικείμενα από όλες τις λίστες που χρειάζονται για την αντικατάσταση με αυτών στο αρχείο word. Ο χρήστης βλέπει εκτυπωμένες τις απαντήσεις του και μπορεί να επιλέξει να κάνει κάποια διόρθωση σε οποιαδηποτε από αυτές. Αυτή τη φορά γίνεται με την χρήση πληκτρολογίου για να αποφύγουμε κάποιο πιθανό σφάλμα που μπορεί να εμφανίστηκε προηγουμένως, λόγω κακής ποιότητας μικροφώνου ή άλλων ήχων που εμπόδισαν την ομαλή και καθαρή λήψη της απάντησης. Ο χρήστης ονοματίζει το αρχείο όπως επιθυμεί και αποθηκεύεται στον ίδιο φάκελο με τα υπόλοιπα αρχεία. Αφού γίνει και αυτό, ο χρήστης στέλνει το αρχείο στο email της ηλεκτρονικής γραμματείας ή σε κάποιο της επιλογής του. Αυτό γίνεται σε ένα άλλο αρχείο που έχει ονομαστεί automail.py που θα εξηγήσω πιο αναλυτικά στο επόμενο υποκεφάλαιο. Τέλος γίνεται μια διαγραφή των λιστών που έχουν δημιουργηθεί από το χρήστη ώστε αν επιλέξει να επαναλάβει τη διαδικασία να μην δημιουργηθεί κάποιο σφάλμα με προηγούμενες απαντήσεις. Η επανάληψη της διαδικασίας πραγματοποιείται στο τέλος και γίνονται αποδεκτές μόνο οι λέξεις ΝΑΙ και YES.

Μια συνάρτηση είναι υπεύθυνη για την αποκωδικοποίηση του ήχου και τη μετατροπή του σε μορφή κειμένου. Κάθε φορά που εκτυπώνεται το αντικείμενο που ζητείται από το χρήστη θα

γίνεται μέσω εκείνης. Όταν λάβει την απάντηση του χρήστη, την περνάει από μια σειρά παραμέτρων και ξεχωριστών συναρτήσεων που έχω δημιουργήσει για να μπορέσω να αποτρέψω λέξεις και φράσεις που δεν αντιπροσωπεύουν κατάλληλη απάντηση για το αντικείμενο εκείνο. Για παράδειγμα αν ζητάει ένα νούμερο δεν επιτρέπει εισαγωγή κάποιου γράμματος ή λέξης και ζητάει άλλη απάντηση. Η πρώτη παράμετρος που έχει τοποθετηθεί, είναι για την απενεργοποίηση του προγράμματος, ακολουθούμενη από εύρεση μη επιτρεπτών λέξεων, έλεγχο για όνομα κάποιου μήνα και μετατροπή του σε αριθμό, απαγόρευση απαντήσεων που δεν είναι ορθά για ορισμένα αντικείμενα, αλλά και αλλαγή στην ίδια την απάντηση όπως αφαίρεση κενών, κεφαλαιοποίηση, αφαίρεση τόνων ώστε να επιτευχθεί αρμονία στις απαντήσεις.

5.3 Αναλυτική Περιγραφή και Εκτέλεση

Αρχικά, στο λειτουργικό σύστημα Linux η επικοινωνία του χρήστη με τον υπολογιστή, γίνεται μέσου του τερματικό (terminal). Για να γίνει η εκκίνηση του προγράμματος πρέπει να πληκτρολογήσουμε κάποιες εντολές στο terminal. Επειδή ο υπολογιστής δεν μπορεί να καταλάβει πως να ξεκινήσει το πρόγραμμα ή να βρει που βρίσκεται στη μνήμη, η εντολή <cd Project> είναι απαραίτητη. Το cd (Change Directory) χρησιμοποιείται για να μας εισάγει στον φάκελο Project που περιέχει τα αρχεία προς εκτέλεση. Επειδή δεν γνωρίζει ούτε την γλώσσα που πρέπει να χρησιμοποιήσει ώστε να μπορεί να “διαβάσει” τι λέει το αρχείο η εντολή <python3 Επιλογή_Γλωσσας.py> χρησιμοποιείται. Το python3 υποδηλώνει τη γλώσσα που θα χρησιμοποιήσει και το Επιλογη_Γλωσσας.py την ονομασία του αρχείου που πρέπει να δεί καθώς και την κατάληξη του.

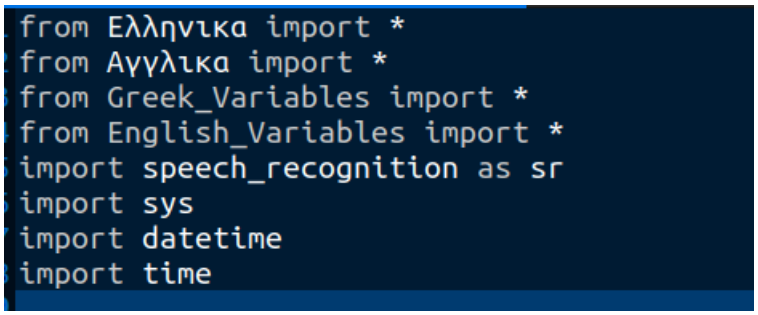


Εικόνα 2. Οι εντολές για να τρέξει ο κώδικας.

Στο πάνω μέρος κάθε αρχείου γίνεται εισαγωγή άλλων αρχείων που συνεργάζονται μεταξύ τους ή και εργαλείων που εκτελούν κάποιο σκοπό. Το αρχείο Επιλογη_Γλωσσας.py περιέχει όλα τα αρχεία εκτός από το automail.py που θα δούμε παρακάτω καθώς και ορισμένα εργαλεία που

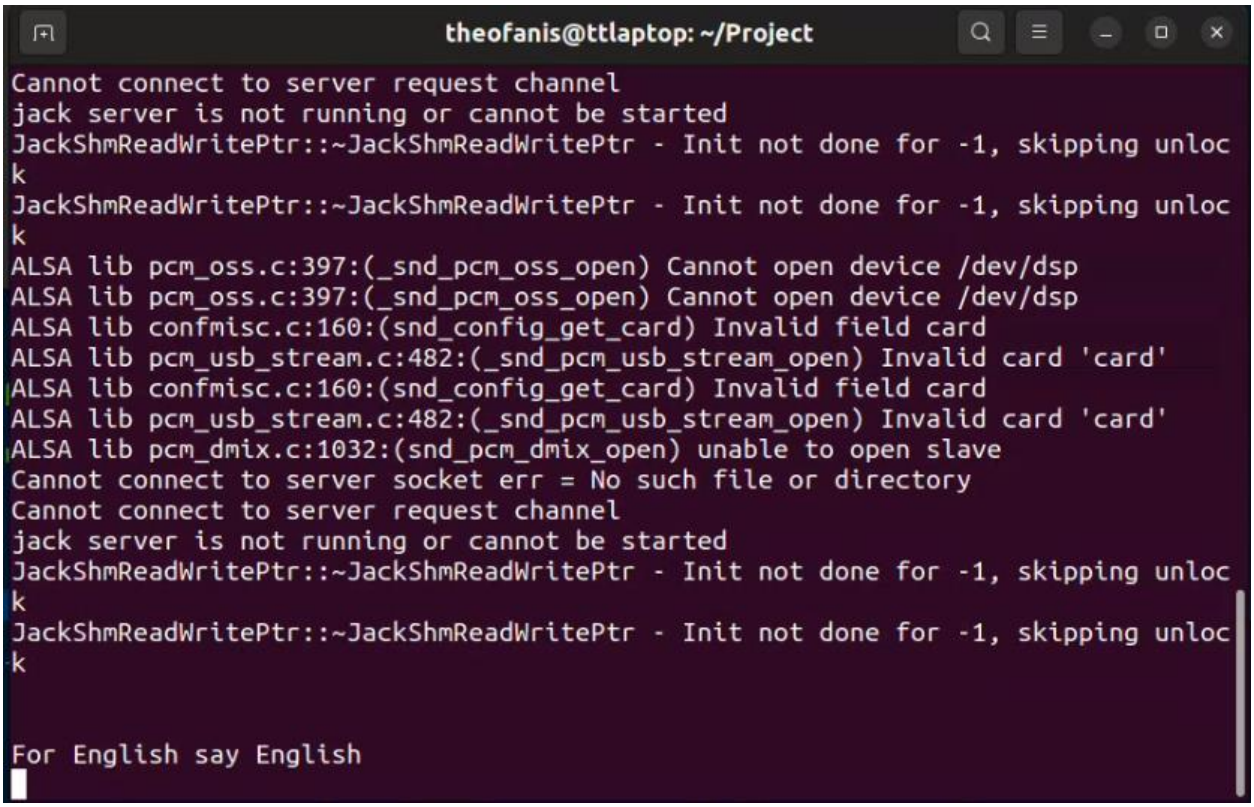
αναλύσαμε πριν. Το σύμβολο (*) χρησιμοποιείται για να συμπεριληφθούν όλα τα περιεχόμενα του αρχείου. Σε περιπτώσεις μαγάλων προγραμμάτων μπορεί να υπάρχουν δεκάδες συναρτήσεις και δεν χρειάζεται να υπερφορτώνουμε το πρόγραμμα με πιο πολλά από ότι χρειάζονται. Για να επιλέξουμε κάποια συγκεκριμένη βιβλιοθήκη από ένα αρχείο η δομή είναι η εξής:

from <το όνομα του αρχείου> **as** <το όνομα της συνάρτηση>



Εικόνα 3. Εισαγωγή των εξωτερικών εργαλείων/αρχείων.

Μόλις γίνει η εκτέλεση των δύο εντολών θα ανοίξει το αρχείο Επιλογη_Γλωσσας.py και θα ξεκινήσει να εκτελεί τις εντολές που βρίσκονται μέσα σε αυτό με τη σειρά. Το πρώτο πράγμα που θα εμφανιστεί είναι πληροφορίες της βιβλιοθήκης από επεξεργασίες που έχουν γίνει στο παρασκήνιο και μια πρόταση ‘For English say English’ που την εμφανίζω με την εντολή print().



Εικόνα 4. Μήνυμα κατά την εκτέλεση.

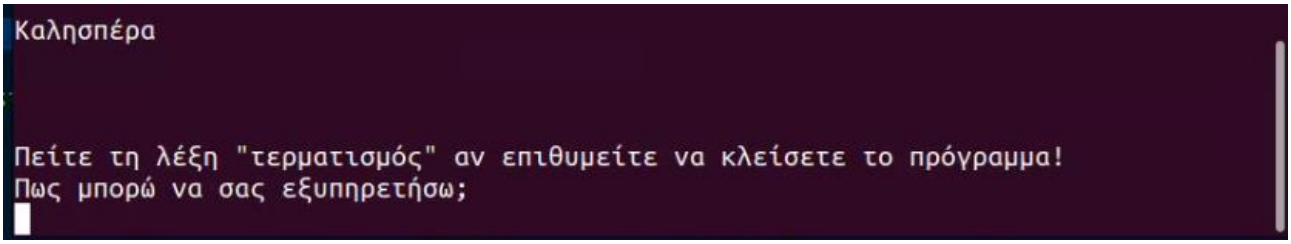
Εδώ ο χρήστης θα πρέπει να αποφασίσει τη γλώσσα για την επικοινωνία του με το τον υπολογιστή. Έχει δύο επιλογές Ελληνικά που είναι το προκαθορισμένο σενάριο ώπου δεν χρειάζεται να πει κάτι για 2-3 δευτερόλεπτα και την λέξει english για την αγγλική γλωσσα. Η εισαγωγή κάποιας άλλης έκφρασης ή αριθμού θα οδηγήσει στην επανάληψη της διαδικασίας εμφανίζοντας ενημερωτικό μήνυμα. Το μήνυμα που εμφανίζεται σε περίπτωση διαφορετικής απάντησης βρίσκεται στο κάτω μέρος της εικόνα 6. Η επιλογή της γλώσσας θα αποθηκευτεί στην μεταβλητή language η οποία θα χρησιμοποιηθεί αργότερα από άλλο αρχείο. Ορίζω κάποιες μεταβλητές για να κρατήσουν τιμές και αποτελέσματα από βιβλιοθήκες όπως είναι το present_time που βοηθάει στο μήνυμα καλωσορίσματος που θα δει ο χρήστης ανάλογα την ώρα. Για παράδειγμα αν είναι πρωί πριν τις 12 θα πει καλημέρα και good morning αντίστοιχα και μετά τις 12 καλησπέρα και good evening. Χρησιμοποιώ την βιβλιοθήκη time για να μπορώ να σταματάω την εκτέλεση του προγράμματος για το χρονικό διάστημα που θέλω. Τα μηνύματα που ο χρήστης μπορεί να δει είναι επειδή το σύστημα έχει ‘παγώσει’ για ένα μικρό χρονικό διάστημα, ειδάλλως ο υπολογιστης θα τα εμφάνιζε μόνο για κλάσματα του δευτερολέπτου.

Αμέσως μετά καλείτε η συνάρτηση ‘Function_Greek(Say_hi, language) η οποία βρίσκεται στο αρχείο <Ελληνικα.py>. Η συνάρτηση αυτή θα λάβει δύο μεταβλητές κατά τη μεταφορά του, το

Say_hi που έχει την τιμή της καλημέρα ή καλησπέρα και το language που περιλαμβάνει τη γλώσσα και θα γίνει χρήση τους στη συνέχεια. Ο λόγος που δηλώνονται και μεταφέρονται είναι επειδή δεν μπορούν τα προγράμματα να δουν προς τα 'πίσω' διότι δεν γίνεται. Έχω παραλείψει μια συνάρτηση και λίγες σειρές εντολών που αφορούν το τέλος της διαδικασίας και θα επιστρέψουμε για την ανάλυση τους όταν φτάσουμε εκεί. Η διαδικασία για τα αγγλικά είναι ακριβώς η ίδια με τη μόνη διαφορά την αλλαγή του ονόματος κάποιας μεταβλητής. Στην φωτογραφία 6 γίνεται καλωσόρισμα με τη λέξη καλημέρα ή καλησπέρα και στη συνέχεια ενημέρωση του χρήστη για τον τερματισμό του προγράμματος που γίνεται με λέξη κλειδί τερματισμός που είναι εντολή του αρχείου <Ελληνικα.py>. Το τερματισμός σταματάει το πρόγραμμα σε όποιο στάδιο και να βρίσκεται αρκεί το μικρόφωνο να είναι ανοιχτό.

```
while True:
        present_time = datetime.datetime.now()
        text = Speech()
        Close_program(text)
        if text is None :
                language = 'GREEK'
                if present_time.hour < 12:
                        print('\n\nΚαλημέρα\n')
                        Say_hi = 'Καλημέρα'
                        time.sleep(1)
                else:
                        print('\n\nΚαλησπέρα\n')
                        Say_hi = 'Καλησπέρα'
                        time.sleep(1)
                while True:
                        Repeat = Function_Greek(Say_hi, language)
                        if Repeat == 'ΝΑΙ':
                                pass
                        else:
                                print('Καλή συνέχεια !!!')
                                sys.exit(0)
        else:
                if text == 'ENGLISH':
                        language = 'ENGLISH'
                        if present_time.hour < 12:
                                print('Good morning\n')
                                Say_hi = 'Good morning'
                                time.sleep(1)
                        else:
                                print('Good evening\n')
                                Say_hi = 'Good evening'
                                time.sleep(1)
                        while True:
                                Repeat = Function_English(Say_hi, language)
                                if Repeat == 'YES':
                                        pass
                                else:
                                        print('Ok, have a nice day !!!')
                                        sys.exit(0)
                else:
                        print('\nΠρέπει να επιλέξετε γλώσσα !\nYou need to choose language !\nΕλληνικά , English')
                        time.sleep(2)
```

Εικόνα 5. Επιλογή γλώσσας και συνθήκη για λάθος.



Εικόνα 6. Εμφάνιση μηνύματος για καλωσόρισμα.

Η συνάρτηση Function_Greek() στην εικόνα 7 ξεκινάει με τη δήλωση τριών μεταβλητών και τις ορίζει ως global. Αυτό σημαίνει πως όλες οι συναρτήσεις που καλούνται από τη συνάρτηση αυτή θα μπορούν να δουν τις μεταβλητές αυτές. Οι δυο μεταβλητές που πήρε μαζί της η συνάρτηση κατά τη μεταφορά, αποθηκεύονται σε μεταβλητές με άλλο όνομα αλλά το περιεχόμενο παραμένει το ίδιο. Στη συνέχεια εμφανίζεται μήνυμα που καλείτε ο χρήστης να επιλέξει ανάμεσα σε δύο επιλογές, την 'ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ' και την 'ΑΙΤΗΣΗ'. Υπάρχει μια δομή επανάληψης που εμποδίζει το χρήστη να προχωρήσει στο επόμενο βήμα με λέξεις διαφορετικές από εκείνες που ζητάει. Ο χρήστης ενημερώνεται εάν κάνει κάποιο λάθος. Στην εικόνα 8, η συνάρτηση File_doc() περιέχει τα αντικείμενα που θα εμφανίζονται στο χρήστη ώστε να δώσει απαντήσεις για αυτά. Ως είσοδο χρειάζεται μόνο τη μεταβλητή Choice που περιέχει την επιλογή για το έντυπο. Η λίστα με μια συνθήκη if-elif-else, καταχωρείται στη μεταβλητή Item_List η οποία και επιστρέφεται στην μεταβλητή που τυχάνει να έχει το ίδιο όνομα στην εικόνα 7. Καλείται στην επόμενη γραμμή η συνάρτηση Function_Choice() και παίρνει μαζί της τις μεταβλητές Choice και Item_List.

```
def Function_Greek(say_hi, language):
        global Choice
        global Say_hi
        global Language
        Language = language
        Say_hi = say_hi
        print('\n\nΠείτε τη λέξη "τερματισμός" αν επιθυμείτε να κλείσετε το πρόγραμμα!\nΠως μπορώ να σας εξυπηρετήσω;')
        while True:
                k = 4
                time.sleep(3)
                Choice = Greek_Speech(k,'\nΕπιλέξτε ένα από τα παρακάτω: \n1. ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ\n2. ΑΙΤΗΣΗ')
                Close_program(Choice)
                if Choice == 'ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ' or Choice == 'ΑΙΤΗΣΗ':
                        Item_List = File_doc(Choice)
                        Function_Choice(Choice,Item_List)
                        k = 5
                        Repeat = Greek_Speech(k,'\nΕαν επιθυμείτε να επαναλάβετε την διαδικασία πείτε ναι !')
                        return Repeat
                else:
                        print('\nΠαρακαλώ δώστε μια έγκυρη απάντηση !')
                        time.sleep(2)
```

Εικόνα 7. Επιλογή αρχείου από το χρήστη με συνθήκη.

```
def File_doc(Choice):

        if Choice == 'ΑΙΤΗΣΗ':
                Item_List = list(['επώνυμο','όνομα','όνομα_πατρός','όνομα_μητρός','Α_Μ_Τ','εξάμηνο','διεύθυνση_οδό',
                'διεύθυνση_αριθμό','ταχυδρομικός_κώδικας','πόλη','τηλέφωνο','κινητό'])

        elif Choice == 'ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ':
                Item_List = list(['επώνυμο', 'όνομα', 'όνομα_πατρός', 'αριθμό_μητρώου','διεύθυνση_και_αριθμό',
                'ταχυδρομικός_κώδικας', 'πόλη', 'αριθμός_τηλεφώνου', 'email', 'όνομα_και_επώνυμο_καθηγητή', 'θέμα_πτυχιακής'])

        return Item_List
```

Εικόνα 8. Λίστες των ονομάτων που θα εμφανιστούν στην οθόνη.

Εδώ με μια εντολή επανάληψης πραγματοποιείται προσπέλαση όλων των αντικειμένων της Item_list και το κάθε αντικείμενο τοποθετείτε και στην μεταβλητή item μέχρι το επόμενο αντικείμενο να λάβει την θέση του. Το item το δέχεται ως είσοδο η συνάρτηση Function_Voice() και προχωράει στην επεξεργασία του (εικόνα 9). Όταν ολοκληρωθεί η διαδικασία, το πλέον αλλαγμένο item θα τοποθετηθεί στη λίστα Answer_List με την εντολή .append. Η εισαγωγή των στοιχείων γίνεται με τη σειρά. Όταν όλα τα στοιχεία τοποθετηθούν μέσα στην λίστα, συνεχίζει στο επόμενο στάδιο που θα δούμε παρακάτω.

```
def Function_Choice(Choice,Item_List):

        print('\nΦορτώνει παρακαλώ περιμένετε !')
        time.sleep(1)
        for item in Item_List :
                time.sleep(0)
                text = Function_Voice(item)
                Answer_List.append(text)
```

Εικόνα 9. Προσπέλαση των αντικεινένων της λίστας.

Πριν συνεχίσουμε, πρέπει να γίνει επεξήγηση της συνάρτησης Greek_Speech() που δέχεται δύο εισόδους το k που αρχικοποιείται αυτόματα καλώντας τη συνάρτηση Function_print() και το item. Η συνάρτηση Function_print θα δώσει μια τιμή στο k ανάλογα το αντικείμενο που έχουμε στην κατοχή μας. Αυτό μπορεί να παρατηρηθεί στην εικόνα 9 και 10 πιο κάτω. Για κάθε item οι τιμές που παίρνει το k είναι διαφορετικές μεταξύ τους. Επειδή δεν μπορεί να δει η συνάρτηση στο αρχείο αυτό πίσω, υπάρχουν ομαδοποιημένα τα item που θέλουν συγκεκριμένη τιμή στο k για να βγάζει νόημα το αποτέλεσμα στην οθόνη. Με την αρχικοποίηση του k στην συνάρτηση Greek_Speech() όλα τα αντικείμενα εμφανίζονται με ωραίο τρόπο.

Στην εικόνα 11 παρατηρούμε την συνάρτηση που είναι υπεύθυνη για την αποδοχή του ήχου και επεξεργασία του. Αρχικά με την εντολή .Recognizer() δημιουργεί έναν αναγνωριστή

φωνής, με το sr.Microphone() αξιοποιεί το μικρόφωνο ως πηγή ήχου με το rec.adjust_for_ambient_noise(mic, duration=0.2 πραγματοποιείται ρύθμιση του μικροφώνου για την προσαρμογή ανάλογα με τον θόρυβο που επικρατεί, η rec.listen(mic) κάνει καταγραφή του ήχου από το μικρόφωνο και την αποθηκεύει στην μεταβλητή audio και η rec.recognize_google(audio, language="el-GR") μετατρέπει τον ήχο σε μορφή κειμένου με τη χρήση της μεθόδου recognize_google της βιβλιοθήκης speech_recognition και ορίζει την γλώσσα στα Ελληνικά. Σε περίπτωση που δεν γινόταν τοποθέτηση του language θα λειτουργούσε σε διάφορες γλώσσες αλλά όχι το ίδιο καλά. Τέλος οι επόμενες δύο εντολές πραγματοποιούν διάσπαση στους τονισμένους χαρακτήρες σε ξεχωριστούς απλούς χαρακτήρες και τόνους (για παράδειγμα το 'ά' γίνεται [α] + [']) και στη συνέχεια αφαιρεί τους τόνους. Όταν λάβει την απάντηση την επιστρέφει πίσω στην μεταβλητή text και σε περίπτωση που δεν λάβει κάποια απάντηση επιστρέφει πάλι πίσω και επαναλαμβάνεται.

```
def Function_print(item):

        # για διαφορετικές απαντήσεις στα items

        Yes_or_No_List_What_For = list(['βεβαίωση_σπουδών','αναλυτική_βαθμολογία','κάτι_άλλο'])
        Yes_or_No_List_What_Need = list(['κάθε_χρήση','εφορία','στρατολογία'])
        Auto_Date = '\nΕπιθυμείτε να ορίσω τη σημερινή ημερομηνία ;\nΑπαντήστε με ΝΑΙ η ΟΧΙ ;'
        Kati_Allo = '\nΤι θα επιθυμούσατε να σας στείλουμε ;'
        Change_Answer = '\nΑν επιθυμείτε να πραγματοποιήσετε κάποια αλλαγή πληκτρολογήστε ναι! '
        Ask_for_Email = '\nΕπιθυμείτε το αρχείο να σταλεί στο email της γραμματείας;'
        Ask_for_Email_2 = '\nΕπιθυμείτε το αρχείο να σταλεί σε κάποιο άλλο email;'

        if item in Yes_or_No_List_What_For :
                k = 2
        elif item in Yes_or_No_List_What_Need :
                k = 3
        elif item == Auto_Date or item == Kati_Allo or item == Change_Answer or item == Ask_for_Email or item == Ask_for_Email_2:
                k = 4
        else:
                k = 1

        return k
```

Εικόνα 10. Έτοιμα μηνύματα για το K και ορισμός τιμής.

```
def Greek_Speech(k,item):

        rec = sr.Recognizer()
        try:
                with sr.Microphone() as mic:

                        if k == 1:
                                print(f'\nΠαρακαλώ πείτε μου {item.replace("_"," ").capitalize()} !')
                        elif k == 2:
                                print(f'\nΕπιθυμείτε να σας στείλουμε {item.replace("_"," ").capitalize()} ;')
                        elif k == 3:
                                print(f'\nΤο χρειάζεστε για {item.replace("_"," ").capitalize()} ;')
                        else:
                                print(f'\n{item}')
                        rec.adjust_for_ambient_noise(mic, duration = 0.2)
                        audio = rec.listen(mic)

                        text= str(rec.recognize_google(audio, language = "el-GR"))
                        text_2 = {ord('\N{COMBINING ACUTE ACCENT}'):None}
                        text = ud.normalize('NFD',text).translate(text_2)

                        text = text.upper()

                        return text
        except:
                if k == 5:
                        pass
                else:
                        print('\nΔεν σας άκουσα παρακαλώ επαναλάβετε !')
                        time.sleep(1)
```

Εικόνα 11. Συνάρτηση αναγνώρισης ομιλίας.

Στη συνέχεια, (εικόνα 12) βλέπουμε μια μορφοποιημένη εντολή print η οποία περιέχει και το λατινικό γράμμα f μέσα στην παρένθεση. Αυτό μου επιτρέπει να εκτυπώσω το περιεχόμενο κάποιας μεταβλητής, στην περίπτωση μας το item, και να προσθέσω πληροφορίες πριν και μετά από αυτό. Χρησιμοποιήθηκε η βιβλιοθήκη .capitalize() που γυρνάει όλα τα γράμματα του περιεχομένου του item σε μικρά, εκτός από το πρώτο που το εμφανίζει ως κεφαλαίο. Έτσι ώστε να ξεχωρίζει το περιεχόμενο την στιγμή που βρίσκεται στην εκτυπωμένη πρόταση. Η συνθήκη if-else διαχωρίζει όλα τα αντικείμενα με το αντικείμενο που θα έχει ως τιμή το email. Όλες οι απαντήσεις που θα δοθούν με τη βοήθεια του εργαλείου Speech Recognition θα είναι στα ελληνικά. η απάντηση που αντιστοιχεί στο email θα είναι μια λέξη που θα αποτελείτε από λατινικό αλφάβητο κάτι που αποτρέπει την κατανόηση του εργαλείου να την αποκωδικοποιήσει και θα την εκλάβει λάθος. Για αυτό το λόγω η εισαγωγή της πραγματοποιείται με το πληκτρολόγιο, Μία δομή επανάληψης τοποθετήθηκε πριν γίνει η κλήση της συνάρτησης που περιέχει το εργαλείο αναγνώρισης ομιλίας και μηδενίζει το περιθώριο σφάλματος και λήψης ήχου από το μικρόφωνο

που μπορεί να ‘κολλήσει’ το πρόγραμμα. Λειτουργεί επίσης και ως δικλείδα τερματισμού του προγράμματος. Η μεταβλητή times αρχικοποιείται με την τιμή μηδέν αυξάνεται κατά ένα και όταν

γίνει ίσο με το δύο το πρόγραμμα τερματίζει. Η πρόσθεση αυτής της ασφάλειας έγινε για την περίπτωση που κολλήσει το μικρόφωνο και δέχεται συνέχεια ήχους και τους επιστρέφει ως NONE.

```
def Function_Voice(item):

        k = Function_print(item)

        if item == 'email':
                text = input(f'\nΠαρακαλώ πληκτρολογήστε το {item.capitalize()} σε λατινικό λεξιλόγιο !\n')
        else:
                times = 0
                while True:
                        text = Greek_Speech(k,item)
                        if text is not None:
                                break
                        else:
                                times += 1
                                if times == 2:
                                        sys.exit(0)
                                pass

        Close_program(text)
        text = Check_For_Bad_words(k,text,item)
```

Εικόνα 12. Έλεγχος για επιστροφής τιμής NONE.

Με την ολοκλήρωση της δομής αυτής, γίνεται έλεγχος για τη λέξη τερματισμού και καλείται η επόμενη συνάρτηση Check_For_Bad_words() με είσοδο το item. Θα γίνει έλεγχος, αν το item βρίσκεται στην λίστα Bad_Words() (εικόνα 13.α,13.β). Αν η λέξη βρίσκεται μέσα σε αυτή, με δομή επανάληψης καλείται πάλι η συνάρτηση για να ενεργοποιηθεί το μικρόφωνο και ζητάει άλλη απάντηση εμφανίζοντας κατάλληλο μήνυμα. Αν όμως δεν υπάρχει εκεί, απλά επιστρέφει το item πίσω.

```
def Check_For_Bad_words(k,text,item):

        while True:
                # έλεγχος για μη επιτρεπτές λέξεις
                if text in Bad_Words:
                        print(f'Η λέξη {text} δεν είναι αποδεκτή, επαναλάβετε το {item} παρακαλώ !')
                        time.sleep(1)
                        text = Greek_Speech(k,item)
                else:
                        return text
```

Εικόνα 13.α. Μη επιτρεπτές λέξεις, έλεγχος και εκτύπωση για λάθος.

```
Παρακαλώ πείτε μου Επώνυμο !
Ελεγχος για τερματισμό
Η λέξη ΚΟΥΤΟΣ δεν είναι αποδεκτή, επαναλάβετε το επώνυμο παρακαλώ !
```

Εικόνα 13.β. Μη επιτρεπτές λέξεις, εκτύπωση στο terminal.

Η συνέχεια της Function_Voice() στην εικόνα 14 δείχνει εντολές που πραγματοποιούν αλλαγές στο item ή το ελέγχουν για κάποιο λόγο. Υπάρχουν τέσσερεις συνθήκες if, μια για την ημερομηνία, μια για τους αριθμούς σε μορφή σειμβολοσειράς, μια για παραμετροποίηση των αριθμών και μια για το email. Η ημερομηνία όταν γίνεται αντικατάσταση του template που βρίσκεται στο αρχείο .docx, δεν γίνεται να δέχεται κάτι άλλο πέρα από αριθμό. Για αυτό το λόγο υπάρχει μια εμφολευμένη συνθήκη if η οποία πραγματοποιεί δύο ελέγχους. Ο πρώτος είναι αν το αντικείμενο βρίσκεται στη λίστα Date_List και ο δεύτερος για το αν το text που είναι η απάντηση του χρήστη βρίσκετε στη λίστα Month_List. Αν και οι δύο συνθήκες ισχύουν, τότε η εκτέλεση δύο εντολών παίρνουν μέρος. Η Date_List περιέχει τρεις μεταβλητές την ημέρα, τον μήνα και το έτος. Θα τρέξουν οι εντολές για όλα τα στοιχεία της λίστας, αλλά μόνο για το μήνα θα κάνει διαφορά. Αυτό γίνεται διότι ο χρήστης μπορεί να δώσει τιμή στο μήνα και αριθμό και συμβολοσειρά. Η λίστα Month_List περιέχει όλους τους μήνες με τη σειρά για την αντικατάσταση τους με αριθμό. Εφόσον τα στοιχεία της λίστας είναι σε σειρά ο κάθε αριθμός αντιστοιχεί σε κάποιο από τα αντικείμενα. Όμως επειδή το πρώτο κελί της λίστας ξεκινάει από το μηδέν και δεν υπάρχει μήνας μηδέν, κάνω μια πρόσθεση του αριθμού του κελιού της λίστας με το ένα και διορθώνεται. Η Words_Number είναι λίστα που κρατάει τα νούμερα από το μηδέν έως το δέκα σε μορφή συμβολοσειράς. Υπάρχει περίπτωση το μικρόφωνο να ακούσει τον αριθμό σωστά αλλά αντί να δώσει ψηφίο κατά τη μετατροπή του σε κείμενο, δίνει συμβολοσειρά. Οπότε, πάλι γίνεται έλεγχος αν κάποιος από τους αριθμούς από το μηδέν έως το δέκα βρίσκεται στη λίστα και τον αντικαθιστά με το ανάλογο ψηφίο του. Εφόσον το πρώτο κελί της λίστας και ο πρώτος αριθμός που περιέχει η λίστα ταυτίζονται με το μηδέν, δεν χρειάζεται να προσθέσω κάποιο αριθμό για να γίνει διόρθωση.

Στην τρίτη if, ελέγχει όλα τα στοιχεία της λίστας Number_List και αν είναι μέσα τότε η απάντηση πρέπει να είναι οπωσδήποτε αριθμός. Σε αρκετές περιπτώσεις το μικρόφωνο άκουγε έναν αριθμό που τον αποτελούσαν τουλάχιστον 7 ψηφία, εκ των οποίων κάποια εμφανίζονταν ως συμβολοσειρές, άλλα με κενό και άλλα με κόμμα ή πλάγια κάθετο. Αυτό συμβαίνει διότι ο τρόπος και οι παύσεις κατά την εκφώνηση του χρήστη για είσοδο κάποιας τιμής, μπορεί να είναι αυτός που χρειάζεται ώστε η βιβλιοθήκη να το καταλάβει ως ημερομηνία ή χρονολογία. Οπότε αποφεύγουμε αρκετά λάθη χρησιμοποιόντας την βιλιοθήκη .replace που θα προέκυπταν στην πορεία με αυτόν τον τρόπο. Επίσης τοποθετήθηκε η βιβλιοθήκη .isdigit() που μπορεί να καταλλάβει εαν η είσοδος του χρήστη είναι αριθμός. Για να λειτουργήσει πρέπει ένας αριθμός να είναι σε μορφή συμβολοσειράς. Η μεταβλητή b θα λάβει την απάντηση της βιβλιοθήκης και θα την κρατήσει. Εάν είναι True τότε προχωράμε στο επόμενο βήμα ενώ στην περίπτωση που είναι False, εκτυπώνει μήνυμα ότι δεν είναι αριθμός. Η συνάρτηση Function_Voice() είναι μια

αναδρομική συνάρτηση, δηλαδή η ίδια η συνάρτηση μπορεί να καλέσει τον εαυτό της. Επειδή η σειρά των συνθηκών if δεν γίνεται να αλλάξει, ούτε το στοιχείο στην περίπτωση False να περάσει από τις παραμετροποιήσεις αν δεν γίνει επανάληψη όλης της συνάρτησης, σε αυτό το σημείο καλείται η συνάρτηση από τον εαυτό εκτελώντας την από την αρχή για το αντικείμενο που χρειάζεται εναλλακτική απάντηση. Τέλος, γίνεται ένας έλεγχος για το αν το στοιχείο είναι το email. Αν ναι, τότε προστίθεται στην απάντηση του χρήστη η κατάληξη '@uth.gr' που ολοκληρώνει το email σαν έκφραση.

```
    if item in Date_List:
            if text in Month_List:
                    index = Month_List.index(text)
                    text = str(index + 1)

    if text in Words_Number:
            index = Words_Number.index(text)
            text = str(index)

    if item in Number_List:
            text = text.replace(" ","").replace(",","").replace("-","").replace("/","")
            a = text.replace(" ","").replace("-","").replace(".","").replace("/","")
            b = a.isdigit()
            if b == True:
                    print(f'Το {text} είναι αριθμός!')
                    time.sleep(1)
            else:
                    print(f'Η λέξη {text} δεν είναι αριθμός, παρακαλώ δώστε μου μόνο αριθμό!')
                    time.sleep(1)
                    text = Function_Voice(item)
    if item == 'email':
            text = text + '@uth.gr'

    return text
```

Εικόνα 14. Έλεγχος για ημερομηνία, νούμερα, γραμματοσειρά και αντικείμενα.

Το επόμενο βήμα αντιστοιχεί στην εισαγωγή της ημερομηνίας. Η δομή επανάληψης φροντίζει η απάντηση του χρήστη να περιορίζεται σε ΝΑΙ ή ΟΧΙ ώστε να ισχύει μια από τις δύο if-elif συνθήκες. Το chatbot, έχει προγραμματιστεί να ρωτήσει τον χρήστη να του ορίσει την σημερινή ημερομηνία με ένα απλό 'ΝΑΙ' ή 'n', δεκτό σε ελληνικό ή λατινικό αλφάβητο. Λειτουργεί όπως στην συνάρτηση που συναντήσαμε στην εικόνα 5 και 6. Σε μια μεταβλητή την present_date, δίνουμε σαν όρισμα τη γραμμή εκτέλεσης datetime.datetime.now() ώπου αντλεί από το σύστημα την ημερομηνία και την ώρα. Για να μπορέσουμε να λάβουμε αυτό που χρειαζόμαστε δηλαδή την ημέρα, τον μήνα και το έτος μεμονωμένα, χρησιμοποιούμε την εντολή '.day', '.month' και '.year' μετά την μεταβλητή present_date όπως φαίνεται στην εικόνα 15 και την φορτώνουμε με τη βοήθεια του .append στην λίστα με τις απαντήσεις του ομιλητή. Όμως για να γίνει σωστή αντικατάσταση στο αρχείο docx που περιέχει το έντυπο, πρέπει να κάνω το ίδιο και στην λίστα Item_List. Τα αντικείμενα της Item_List είναι ίδια με τα template στο docx αρχείο. Τον λόγο θα τον εξηγήσουμε

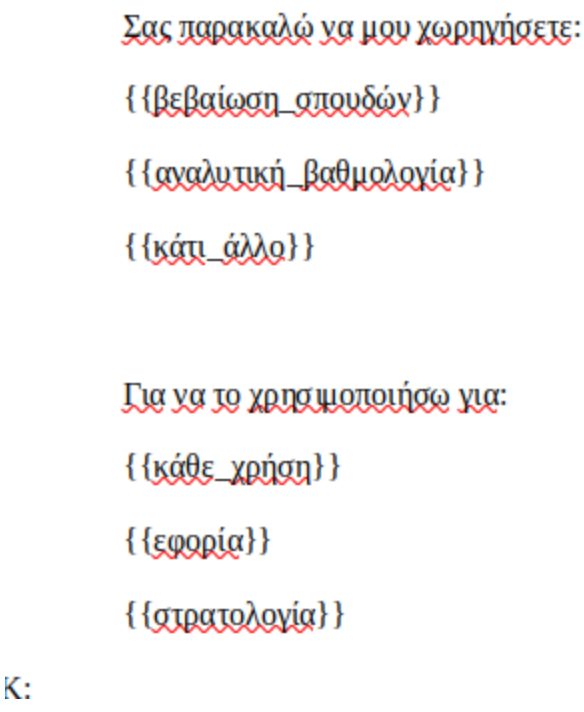
παρακάτω. Σε διαφορετική περίπτωση που ο χρήστης επιθυμεί να ορίσει δικιά του ημερομηνία, το ΟΧΙ είναι η απάντηση που χρειάζεται να δώσει. Έχει δημιουργηθεί μια λίστα η Date_List και περιέχει την ημέρα, τον μήνα και το έτος. Για κάθε ένα από τα αντικείμενα καλείται η συνάρτηση Function_Voice ώστε να γινουν οι απαραίτητοι έλεγχοι και μετά στη συνέχεια γίνεται προσθήκη τους στις δύο αυτές λίστες. Όταν οποιαδήποτε από τις δύο συνθήκες ολοκληρωθεί, η εντολή break θα τερματίσει την δομή επανάληψης και θα συνεχίσει στο επόμενο βήμα.

```
while True:

        text = Function_Voice('\nΕπιθυμείτε να ορίσω τη σημερινή ημερομηνία;\nΑπαντήστε με ΝΑΙ η ΟΧΙ!')
        present_date = datetime.datetime.now()
        if text == 'ΝΑΙ':
                Item_List.append('ημέρα')
                Answer_List.append(present_date.day)
                Item_List.append('μήνα')
                Answer_List.append(present_date.month)
                Item_List.append('έτος')
                Answer_List.append(present_date.year)
                break
        elif text == 'ΟΧΙ':
                for item in Date_List:
                        text = Function_Voice(item)
                        Item_List.append(item)
                        Answer_List.append(text)
                break
        else:
                print('Πρέπει να απαντήσετε με "ΝΑΙ" ή "ΟΧΙ" !')
                time.sleep(2)
```

Εικόνα 15. Επιλογή για την εισαγωγή ημερομηνίας.

Έχουμε φτάσει σε ένα σημαντικό κομμάτι που αποτελεί το μοναδικό σημείο το οποίο κάνει τα δύο αρχεία docx να διαφέρουν μεταξύ τους. Ενώ το έντυπο για τη δήλωση πτυχιακής εργασίας ακολουθεί μια σταθερή δομή από την αρχή εως το τέλος, η αίτηση διαφέρει σε ένα σημείο. Στην εικόνα 16 φαίνεται το σημείο αυτό. Πρωτού ξεκινήσω να αναλύω την εικόνα 17, είναι μια καλή στιγμή να αναφερθούμε λίγο στα δύο αρχεία docx. Η μορφή που πρέπει να έχει ένα template είναι {{όνομα}}. Το όνομα είναι μια τυχαία λέξη ενώ τα άγγιστρα είναι υποχρεωτικά για να λειτουργήσει η βιβλιοθήκη και θα δούμε τις εντολές αργότερα. Το αντικείμενο της λίστας Item_List πρέπει να είναι το ίδιο με την έκφραση που περιέχεται μέσα στα άγγιστρα για να είναι εφικτή η αλλαγή που θα πραγματοποιηθεί.



Εικόνα 16. Διαφορά στη δομή το έντυπο αίτηση.

Οι λίστες What_Need και What_For περιέχουν από τρεις μεταβλητές. Η What_Need περιέχει τα τρία πρώτα στοιχεία και η What_For τα υπόλοιπα της εικόνας 16 . Η Request_List στην εικόνα 16 είναι μια λίστα που περιέχει τις δύο λίστες που ανέφερα μόλις ως αντικείμενα της, χρησιμοποιώντας τις αγγύλες ‘ [ ] ’. Θα προσπελαστεί και για κάθε μια θα μπει σε μια δομή επανάληψης while η οποία θα τρέχει όσο ο χρήστης δεν έχει επιλέξει έστω και ένα από τα αντικείμενα της κάθε λίστας. Επειδή με την αντικατάσταση των στοιχείων τα αποτελέσματα δεν ήταν όπως θα ήθελα, χρειάστηκε μια παραμετροποίηση για να γίνει πιο ομοιόμορφο. Φανταστείτε ότι κάθε φορά που γίνεται αντικατάσταση, η απάντηση του χρήστη θα πάει στην ανάλογη θέση που την αντιπροσωπεύει. Αυτό όμως δημιουργούσε κενά αν παραλείπαμε μια από τις δύο πρώτες επιλογές. Ας το δούμε μέσω ενός παραδείγματος με την εικονα 16. Αν ζητήσω μόνο αναλυτική βαθμολογία, η βεβαίωση σπουδών και το κάτι άλλο απλά θα εξαφανιστούν και θα αφήσουν κενό. Αυτό κάνει το αποτέλεσμα να φαίνεται άσχημο. Το ίδιο ισχύει και με τη δεύτερη τριάδα. Για το λόγο αυτό βρήκα ένα τρόπο για να λύσω αυτό το μορφολογικό πρόβλημα. Για κάθε ένα αντικείμενο που επιλέγω, δεν χρειάζεται να ταιριάζει στις λίστες που δεν εμφανίζονται στον χρήστη ή στο τελικό αποτέλεσμα. Δηλαδή αν επιθυμώ την αναλυτική βαθμολογία, θα την κάνω .append στην λίστα Answer_List που περιέχει τις απαντήσεις του χρήστη και θα κάνω .append μόνο το πρώτο στοιχείο που ταιριάζει σε αυτά της εικόνας 16. Αυτό πραγματοποιείται με μια μεταβλητή που

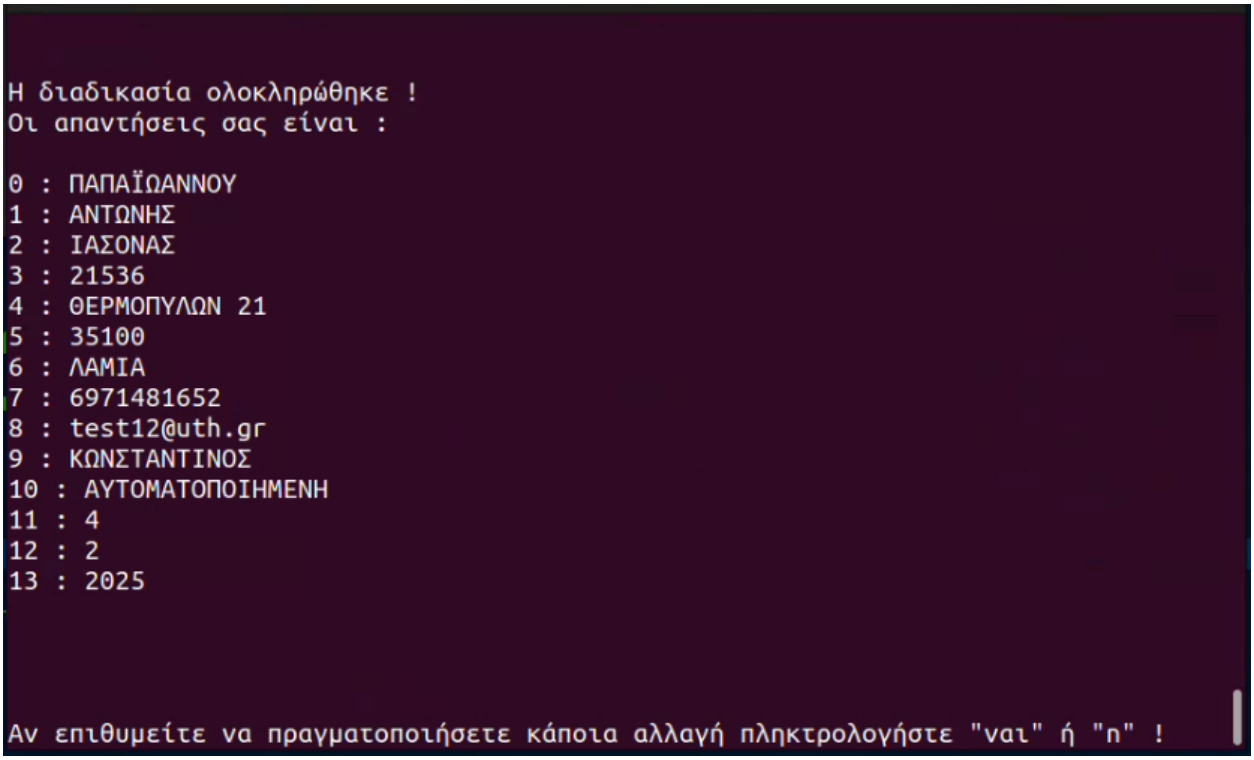
μετράει τα αντικείμενα που πρόσθεσε ο χρήστης και πιο απλοϊκά ξεκλειδώνουμε τις ‘θέσεις’ με τη σειρά ώστε η αντικατάσταση να μην αφήσει κενά ανάμεσα στις προτάσεις. Με την εισαγωγή των δύο επιλογών, η εντολή break θα ‘σπάσει’ την επαναληπτική δομή και θα προχωρίσουμε στην εμφάνιση, τροποποίηση και αποθήκευση του αρχείου.

```
if Choice == 'ΑΙΤΗΣΗ':
        Request_List = list([What_Need , What_For])
        for what_list in Request_List:
                while True:
                        count_yes = 0
                        for item in what_list:
                                text = Function_Voice(item)
                                if text == 'ΝΑΙ':
                                        if item == 'κάτι_άλλο':
                                                text = Function_Voice('\nΤι θα επιθυμούσατε να σας στείλουμε ?')
                                                Answer_List.append(text.replace("_"," ").capitalize())
                                        else:
                                                Answer_List.append(item.replace("_"," ").capitalize())
                                        count_yes += 1
                                else:
                                        pass
                        for item in range(count_yes):
                                Item_List.append(what_list[item])

                        if count_yes == 0:
                                print('Πρέπει να απαντήσετε τουλάχιστον μια φορά με ναι !')
                                time.sleep(1)
                        else:
                                break
```

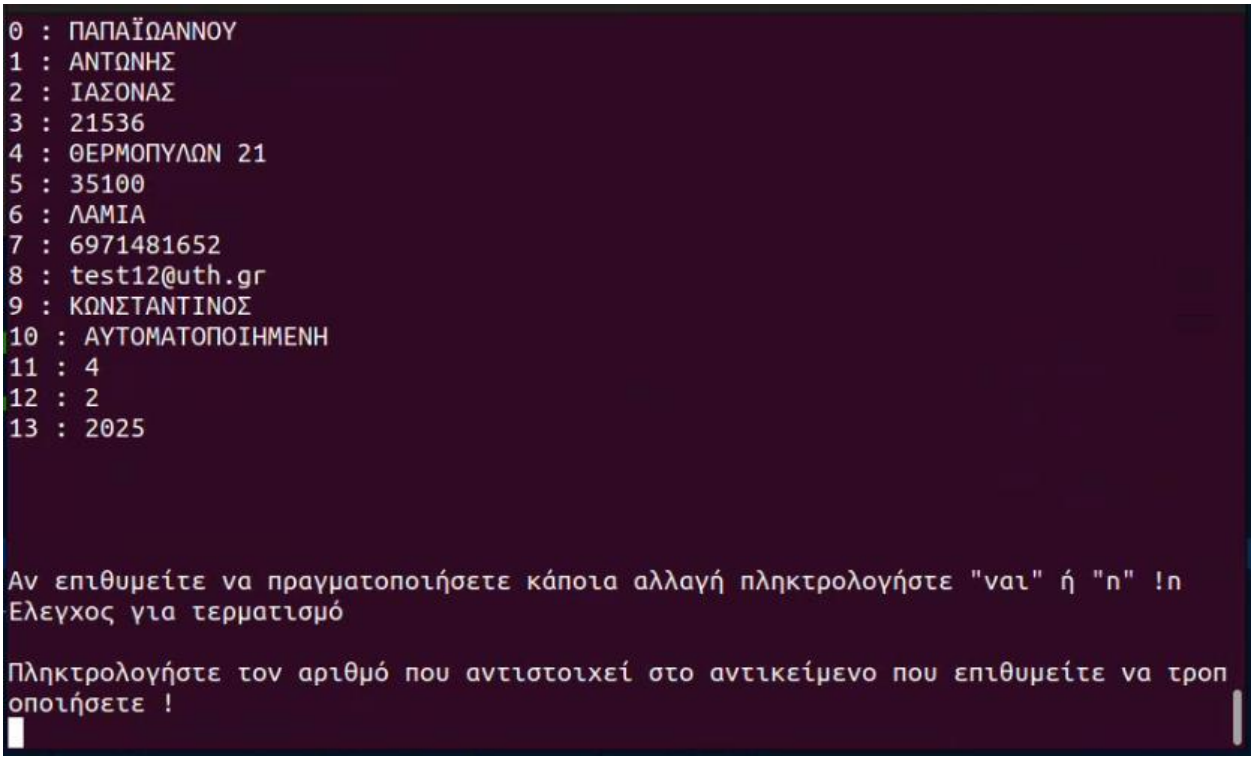
Εικόνα 17. Κώδικας για το έντυπο αίτηση.

Όλα τα αντικείμενα έχουν ζητηθεί και βρίσκονται στη λίστα Answer_List. Ξεκινάμε με την εμφάνιση τους που πραγματοποιείται με την δημιουργία ενός λεξικού (dictionary). Για την εκτύπωση ενός dictionary στην οθόνη χρειαζόμαστε να αριθμίσουμε τη λίστα Answer_List και σε μια επαναληπτική δομή for να γίνει προσπέλαση όλων των κελιών της λίστας αλλά θα έχει δύο μεταβλητές, δηλαδή δύο πληροφορίες για κάθε κελί. Με την εκτύπωση θα βλέπουμε στην οθόνη τον αριθμό του κελιού καθώς και το περιεχόμενο του όπως παρατηρείται στην εικόνα 18. Η εντολή για τη δημιουργία του λεξικού είναι <data = dict(zip(Item_List,Answer_List))> ώπου data η μεταβλητή που θα χρησιμοποιηθεί αργότερα για την αντικατάσταση, και το υπόλοιπο η δημιουργία του λεξικού.



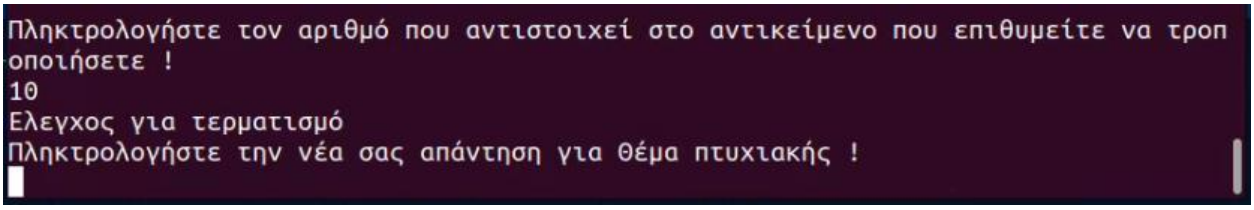
Εικόνα 18. Αποτελέσματα στο terminal.

Πλέον ο χρήστης μπορεί να δει όλες τις απαντήσεις του και εφόσον επιθυμεί να πραγματοποιήσει κάποια αλλαγή μπορεί να το κάνει. Σκόπιμα στο νούμερο 10 βρίσκεται μια λανθασμένη απάντηση για το θέμα της πτυχιακής εργασίας. Θα αλλάξουμε την απάντηση με μια άλλη κάνοντας χρήση του πληκτρολογίου. Στην εικόνα 19 έχουμε πληκτρολογήσει το λατινικό γράμμα ‘n’ και στη συνέχεια εμφανίστηκε μήνυμα με το επόμενο βήμα που πρέπει να εκτελέσουμε.

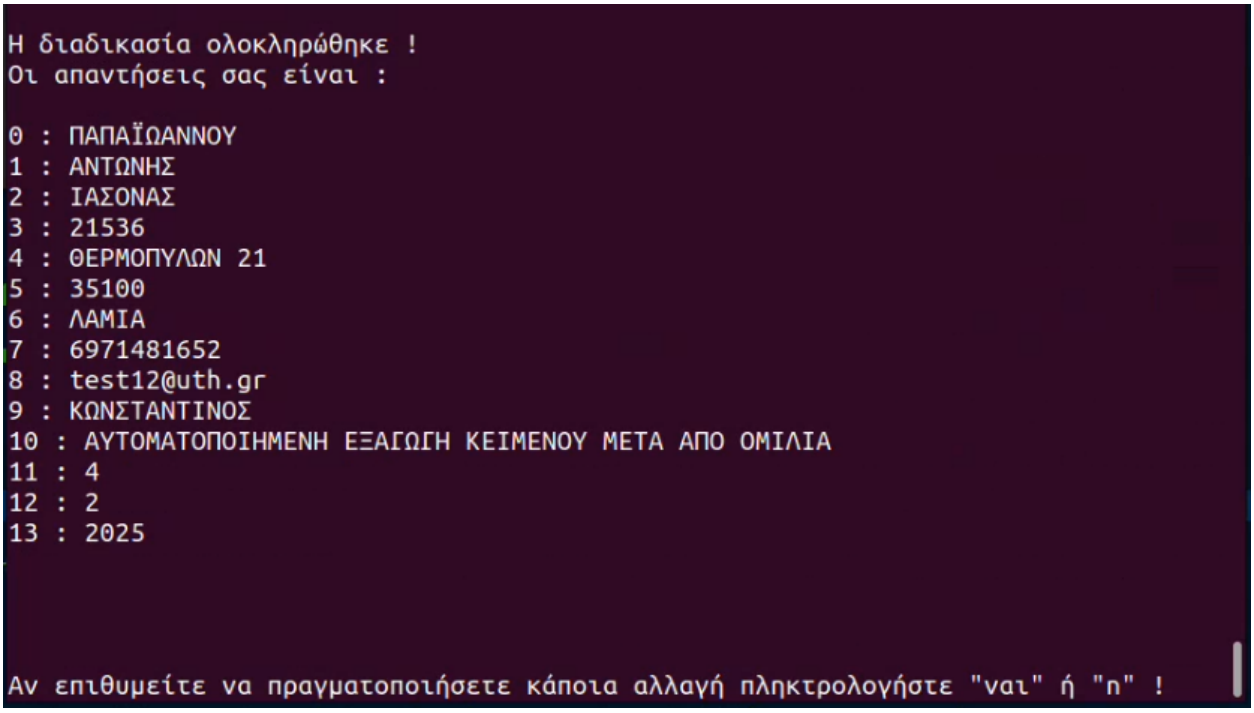


Εικόνα 19. Κατάλληλο μήνυμα για την αλλαγή απάντησης.

Το επόμενο βήμα είναι να πληκτρολογήσουμε τον αριθμό που αντιστοιχεί στην πτυχιακή εργασία που είναι το 10 όπως φαίνεται στην εικόνα 20. Μόλις πληκτρολογήσουμε την νέα απάντηση η εικόνα 21 δείχνει μια άλλη λίστα που έχει αντικαταστήσει τον αριθμό 10 με τη νέα απάντηση.



Εικόνα 20. Εισαγωγή νέας απάντησης.



Εικόνα 21. Ολοκλήρωση αλλαγής της νέας απάντησης.

Η αποθήκευση του αρχείου πραγματοποιείται με την εισαγωγή ενός ονόματος που διαλέγει ο χρήστης. Η ονομασία του αρχείου πρέπει να αποτελείται από λατινικούς χαρακτήρες. Η ονομασία σπάει σε δύο μεταβλητές την a και b. Η a έχει το όνομα που θα εισαχθεί από το πληκτρολόγιο ενώ το b έχει την κατάληξη του αρχείου. Η εντολή ord(char)<128 περιέχει τους χαρακτήρες ASCII Αμερικανικού τύπου.

```
data = dict(zip(Item_List,Answer_List))

status = 'INVALID'
while status == 'INVALID':
        a = input('\nΔώστε όνομα για την αποθήκευση του αρχείου!\nΤο αρχείο πρέπει να περιέχει λατινικούς χαρακτήρες : ')
        if all(ord(char) < 128 for char in a) :
                print(f"Εγκυρη απάντηση : {a}")
                status = 'VALID'
                break
        else:
                print("Μη έγκυρη απάντηση δοκιμάστε ξανά!")
```

Εικόνα 22. Δημιουργία λεξικού και αποθήκευση του αρχείου.

Γίνεται πρόσθεση της μεταβλητής a και b και αποθηκεύονται στην μεταβλητή File_Name η οποία ανάλογα με την επιλογή του χρήστη που βρίσκεται στο Choice που είδαμε στην αρχή θα ανοίξει το κατάλληλο αρχείο docx και θα πραγματοποιήσει την αντικατάσταση για όλες τις μεταβλητές που χρειάζεται. Θα εμφανιστεί μήνυμα ότι το αρχείο αποθηκεύτηκε με επιτυχία. Στην

παρακάτω εικόνα η πρώτη εντολή DocxTemplate χρησιμεύει για να μπορέσει να δει το αρχείο που θα επεξεργαστεί και αποθηκεύεται στην μεταβλητή doc, η δεύτερη doc.render(data) ολοκληρώνει την αντικατάσταση με το λεξικό που δημιουργήσαμε νωρίτερα και η τρίτη, doc.save(File_Name) αποθηκεύει το αρχείο με το όνομα που έχει η μεταβλητή στην παρένθεση.

```
b = '.docx'
# Τελικό όνομα αρχείου
File_Name = a + b
if Choice == 'ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ':

        doc = DocxTemplate("Thesis_gr.docx")
        doc.render(data)
        doc.save(File_Name)

else:
        doc = DocxTemplate("Request_gr.docx")
        doc.render(data)
        doc.save(File_Name)

print(f'Το αρχείο αποθηκεύτηκε στη συσκευή σας επιτυχώς με όνομα {File_Name}.')
time.sleep(1.5)
```

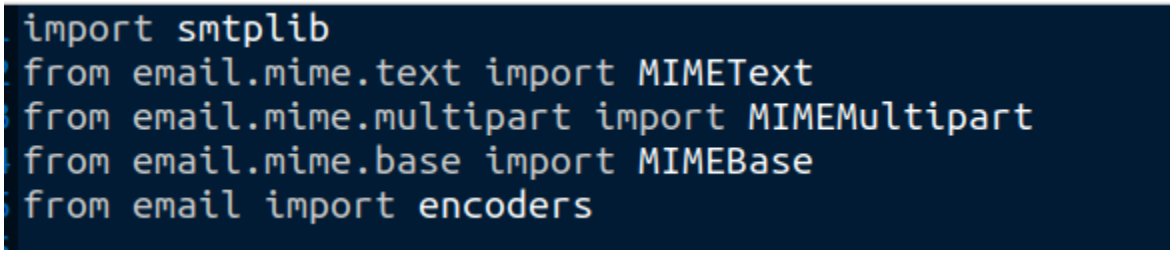
Εικόνα 23. Αντικατάσταση των template και αποθήκευση.

Το τελευταίο στάδιο της διαδικασίας είναι η αποστολή του πλέον ολοκληρωμένου και αποθηκευμένου αρχείου, σε κάποιο ηλεκτρονικό ταχυδρομίο (email). Πριν ξεκινήσουμε στη επεξήγηση των εντολών για το συγκεκριμένο σημείο, πρέπει να παραμετροποιήσουμε τις ρυθμίσεις στο google λογαριασμό. Εφόσον είμαστε συνδεδεμένοι στον λογαριασμό google ως προεπιλεγμένη γλώσσα τα αγγλικά, πηγαίνουμε στις ρυθμίσεις, επιλέγουμε το Security και πρέπει να ενεργοποιήσουμε το 2-Step Verification. Στη συνέχεια πληκτρολογούμε app passwords στην μπάρα αναζήτησης θα μας ζητήσει να δώσουμε όνομα για την εφαρμογή και με την δημιουργία του, θα λάβουμε έναν 16-ψήφιο κωδικό τον οποίο πρέπει να σημειωθεί και να αποθηκευτεί διότι δεν ξαναεμφανίζεται.

Έγινε ρύθμιση του λογαριασμού Gmail και πλέον είναι εφικτό το επόμενο βήμα. Για την αποστολή του αρχείου έχουμε κάποιες παραμέτρους στο αρχείο Ελληνικα.py και κάποιες στο automail.py. Ορίζουμε δύο μεταβλητές, την ονομασία του email καθώς και τον κωδικό που λάβαμε από την ρύθμιση του λογαριασμού google που τον κρατάει η μεταβλητή pswd σε μορφή συμβολοσειράς. Ο χρήστης θα ερωτηθεί μια φορά εάν επιθυμεί να σταλεί στο email της γραμματείας ή σε κάποιο της επιλογής του. Το email της γραμματείας στέλνετε αυτομάτα μόνο με την απάντηση ΝΑΙ, ενώ για επιλογή κάποιου άλλου, χρειάζεται να εκχωρηθεί μέσω του πληκτρολογίου. Όποια από τις δύο περιπτώσεις και αν επιλέξουμε μεταφερόμαστε στο ίδιο

αρχείο, το automail.py. Η συνάρτηση send_mails θα μας μεταφέρει στο επόμενο αρχείο και θα πάρει και επτά μεταβλητές ως είσοδο της.

Για την αποστολή των αρχείων (εικόνα 24), κάνω εισαγωγή κάποιον επεκτάσεων της βιβλιοθήκης smtplib. Η κάθε μια εκτελεί μια διαφορετική διαδικασία και ο συνδυασμός τους μας δίνουν το τελικό αποτέλεσμα. Η επέκταση “from email.mime.text import MIMEText χρησιμοποιείται για τη δημιουργία ενός MIME αντικειμένου το οποίο θα περιέχει ένα κείμενο που θα αποτελεί το “σώμα” του email. Το “from email.mime.multipart import MIMEMultipart” περιέχει διάφορα στοιχεία για την μεταβλητή smg, όπως είναι το email που αποστέλνεται το μήνυμα, το που θα πάει καθώς και πιο θα είναι το θέμα του. Η επόμενη επέκταση είναι η “from email.mime.base import MIMEBase” επιτρέπει την προσθήκη αρχείων που προορίζονται για αποστολή. Τέλος τα encoders που εξάγονται από τη βιβλιοθήκη email και παρέχουν κατάλληλα εργαλεία για την αποκωδικοποίηση των αρχείων. Το Base64 είναι η συνήθης μορφή που παίρνει, μετά τη μετατροπή από τους αποκωδικοποιητές.



Εικόνα 24. Smtp εισαγωγή βιβλιοθηκών.

Έχουν οριστεί κάποιες μεταβλητές στο πάνω μέρος του automail.py όπως είναι η smtp_port που περιέχει την τιμή 587 που είναι κατάλληλη για την αποστολή με το ηλεκτρονικό ταχυδρομείο. Η μεταβλητή smtp_server περιέχει την τιμή smtp.gmail.com και η χρήση της μπορεί να ρυθμίσει οποιαδήποτε εξωτερική εφαρμογή ηλεκτρονικού ταχυδρομείου για την αποστολή μηνυμάτων. Οι μεταβλητές που έλαβε το αρχείο με το κάλεσμα της συνάρτησης, θα χρησιμοποιηθούν για να γίνει εκτύπωση χαιρετισμού (καλημέρα-καλησπέρα), τη γλώσσα που τα τα εμφανίσει, το κείμενο που θα εκτυπώσει για το docx που επιλέχθηκε, το όνομα του αρχείου, το email που θα στείλουμε το μήνυμα καθώς και το δικο μας email μαζί με τον κωδικό που αποθηκεύσαμε.

Χρησιμοποιώ τη βιβλιοθήκη MIMEMultypart() για να δημιουργήσω ένα αντικείμενο το οποίο θα έχει κάποια χαρακτηριστικά. Τα χαρακτηριστικά του θα είναι από πιο email στέλνεται το μήνυμα, ποιο το λαμβάνει και ποιο είναι το θέμα του. Η δομή κάθε χαρακτηριστικού περιγράφεται

ως εξής: msg[‘From’] που είνα η μεταβλητή που θα περιέχει το περιεχόμενο της μεταβλητής email_from που έγινε εισαγωγή με τη συνάρτηση send emails και θα έχει την ετικέτα “From” που αντιστοιχεί στο email από το οποίο απεστάλει το μήνυμα. Το ίδιο γίνεται και για το email που θα το λάβει “to” όπως και με το περιεχόμενο του μηνύματος “Subject”. Όπως μπορεί να παρατηρηθεί πιο κάτω (εικόνα 25), η εντολή msg.attach(MIMEText(body, ‘plain’)) προσθέτει τα σχόλια στο μήνυμα ως απλή μορφή κειμένου και η επόμενη εντολή attachment = open(File_Name), ‘rb’) μετατρέπει το αρχείο σε δυαδικό σύστημα για την κατανόηση από τον υπολογιστή. Τα αρχικά ‘rb’ σημαίνουν ανάγνωση (read) και διαδικό (binary).

```
msg = MIMEMultipart()
msg['From'] = email_from
msg['To'] = email_to
msg['Subject'] = subject
msg.attach(MIMEText(body, 'plain'))
attachment = open(File_Name, 'rb')
```

Εικόνα 25. Ετικέτες.

Οι εντολές που ακολουθούν (εικόνα 26) θα παραμετροποιήσουν το αρχείο ώστε να είναι έτοιμο για την αποστολή του. Ξεκινάμε με την πρώτη η οποία θα αποθηκευτεί στην μεταβλητή attachment_package. Πραγματοποιείται δημιουργία ενός αντικειμένου MIMEBase για την αποστολή όμοιου με Word αρχείου (docx), δηλώνοντας με το ‘application’ ότι πρόκειται για αρχείο εφαρμογής και το ‘vnd.openxmlformats-officedocument.wordprocessingml.document’ για να προσδιορίσει ότι πρόκειται για ένα word αρχείο. Με την επόμενη εντολή, ‘attachment_package.set_payload((attachment).read()), γίνεται ανάγνωση του περιεχομένου του αρχείου (attachment).read αλλά και μεταφορά του στο attachment_package ως περιεχόμενο. Η εντολή encoders.encode_base64(attachment_package) κάνει μετατροπή όπως αναφέρθηκε και πρωτύτερα κάνει μετατροπή του αρχείου σε μορφή base64 για να μπορέσει να λάβει μέρος η αποστολη του. Η εκτέλεση της αμέσως επόμενης εντολής κάνει προσθήκη επικεφαλίδας HTTP για να δηλώσει πως το αρχείο είναι για αποστολή στο email, και με τη μεταβλητή File_Name η οποία ήταν και αυτή είσοδος με τη συνάρτηση send_emails εμφανίζεται το όνομα του αρχείου στον παραλήπτη. Η είσοδος του ονόματος, όπως ανέφερα σε προηγούμενο κεφάλαιο πρέπει να είναι σε λατινικό αλφάβητο για την μεταφορά του ονόματος. Η χρήση της βιβλιοθήκης ‘msg.as_string()’ μετατρέπει το μήνυμα που περιέχει η μεταβλητή msg σε μια μορφοποιημένη συμβολοσειρά και αποθηκεύεται στην μεταβλητή text.

```
attachment_package = MIMEBase('application', 'vnd.openxmlformats-officedocument.wordprocessingml.document')
attachment_package.set_payload((attachment).read())
encoders.encode_base64(attachment_package)
attachment_package.add_header('Content-Disposition', "attachment; filename = " + File_Name)
msg.attach(attachment_package)
text = msg.as_string()
```

Εικόνα 27. Αναγνώριση και επεξεργασία αρχείου.

Τέλος, εμφανίζεται μήνυμα ότι πραγματοποιείται σύνδεση με server για την αποστολή του αρχείου (εικόνα Start Process). Η εντολή smtplib, λαμβάνει τις δύο μεταβλητές που ορίσαμε πρωτύτερα smtp_server και smtp_port, τις τοποθετεί στην μεταβλητή TIE_server και με την βιβλιοθήκη '.starttls()' ξεκινάει η διαδικασία αποστολής. Οι χρήση των μεταβλητών email_from και pswd, είναι και αυτά είσοδοι της συνάρτησης send_emails και αξιοποιούνται για τον έλεγχο ταυτοποίησης και αυτοματοποιημένης σύνδεσης στον λογαριασμο από τον οποίο θα γίνει η αποστολή του email. Η 'TIE_server.sendmail()' θα λάβει τρεις μεταβλητές, το όνομα του λογαριασμού που θα στείλει το μήνυμα, το κείμενο και το όνομα εκείνου που θα το λάβει (εικόνα End Process). Η διαδικασία ολοκληρώνεται με την εντολή 'TIE_server.quit' που τερματίζει την επικοινωνία με τον server.

```
print("Σύνδεση με το διακομιστή...")
TIE_server = smtplib.SMTP(smtp_server, smtp_port)
TIE_server.starttls()
TIE_server.login(email_from, pswd)
print("Επιτυχής σύνδεση με το διακομιστή\n")
```

Εικόνα 28. Έναρξη διαδικασίας σύνδεσης με τον server.

```
print(f"Αποστολή email στο: {email_to}...\n")
TIE_server.sendmail(email_from, email_to, text)
print(f"Το email στάλθηκε στο: {email_to} με επιτυχία\n")

TIE_server.quit()
```

Εικόνα 29. Αποστολή του αρχείου και διακοπη επικοινωνίας με τον server.

Με το τέλος της διαδικασίας της αποστολής του email, πάμε πίσω στο αρχείο Ελληνικά.py χωρίς να επιστρέφει κάποια μεταβλητή η συνάρτηση send_mails(). Μετά γίνεται εκκαθάριση των απαντήσεων του χρήστη καθώς και όλων των μεταβλητών που προστέθηκαν στη λίστα Item_List μέσω μιας δομής επανάληψης for (εικόνα Delete_Lists). Η αφαίρεση των στοιχείων από τις λίστες

παίρνει μέρος διότι ο χρήστης θα ερωτηθεί αν επιθυμεί να επαναλάβει τη διαδικασία που είναι το τελευταίο κομμάτι εντολών και συνδέονται με την αρχή του προγράμματος αφού εάν επιθυμεί να την επαναλάβει, οι λίστες θα είναι κενές και δεν θα προκύψει πρόβλημα σύγκρουσης με τις προηγούμενες απαντήσεις. Ο χρήστης μπορεί να επαναλάβει τη διαδικασία μόνο με τη λέξη ναι, διαφορετικά το πρόγραμμα θα τερματιστεί.

```
Answer_List.clear()
Delete_List = list(['ημέρα','μήνα','έτος','βεβαίωση_σπουδών', 'αναλυτική_βαθμολογία',
                    'κάτι_άλλο','κάθε_χρήση', 'εφορία', 'στρατολογία'])
for item in Delete_List:
        if item in Item_List:
                Item_List.remove(item)
```

Εικόνα 30. Εκαθάριση των λιστών.

5.4 Διαφορές

Τα δύο αρχεία , Ελληνικα.py και Αγγλικα.py έχουν μια μεγάλη διαφορά που τα ξεχωρίζει. Το εκτελέσιμο αρχείο Αγγλικά.py χρησιμοποιεί την βιβλιοθήκη Speech Recognition όπως χωρίς αλλαγές. Δέχεται λέξεις στην Αγγλική γλώσσα και τις μετατρέπει σε μορφή κειμένου. Οι λατινικοί χαρακτήρες δεν έχουν τόνους όπως οι Ελληνικές λέξεις. Αυτή είναι η βασική διαφορά που τα κάνει ξεχωριστά μεταξύ τους. Παρατηρήθηκε κατά την προσομοίωση της εργασίας ότι υπήρξαν φορές που το μικρόφωνο δεν ‘άκουσε’ τη λέξη σωστά από το χρήστη με αποτέλεσμα να τονιστεί σε διαφορετική συλλαβή. Αυτό το σφάλμα που προκύπτει, εξαφανίζεται με τη χρήση της βιβλιοθήκης που αναφέραμε νωρίτερα στο ίδιο και σε προηγούμενο κεφάλαιο. Η χρήση της ‘εξαφανίζει’ τους τόνους από τις λέξεις εξαλείφοντας την περίπτωση δημιουργίας τέτοιου λάθους.

Επίσης το κάθε αρχείο αντλεί λέξεις από ένα δικό του άλλο αρχείο. Το αρχείο Ελληνικά.py τις δέχεται από το Greek_Variables.py ενώ το αρχείο Αγγλικά.py από το English_Variables.py. Τα μηνύματα που εκτυπώνονται από το χρήστη είναι και αυτά στην ανάλογη γλώσσα του κάθε αρχείου. Ο λόγος που γράφτηκαν έτσι τα δύο αρχεία, είναι για να μην δημιουργείται ‘μπέρδεμα’ με τις μεταβλητές και να είναι πιο καθαρές και κατανοητές οι εντολές που περιέχουν.

5.5 Αποτελέσματα

Τα αποτελέσματα της εργασίας είναι ένα αποθηκευμένο αρχείο με κατάληξη ‘.docx’ στον υπολογιστή καθώς και η αποστολή του στο email που επιλέχτηκε. Παρακάτω υπάρχουν εικόνες με τις αλλαγές που έγιναν πριν και μετά την διεργασία καθώς και το κείμενο που εμφανίζεται στα email που στάλθηκαν. Οι εικόνες 31, 32 και 33 περιέχουν τα αποτελέσματα για την αίτηση ενώ οι 34,35 και 36 για την πτυχιακή.

ΑΙΤΗΣΗ

Επώνυμο: {{επώνυμο}}

Όνομα: {{όνομα}}

Όν.Πατέρα: {{όνομα_πατρός}}

Όν.Μητέρας: {{όνομα_μητρός}}

Α.Μ.Τ: {{A_M_T}}

Εξάμηνο: {{εξάμηνο}}

Διεύθυνση κατοικίας:

Οδός: {{διεύθυνση_οδό}}

Αριθμός: {{διεύθυνση_αριθμό}} Τ.Κ: {{ταχυδρομικός_κώδικας}}

Πόλη: {{πόλη}}

Τηλέφωνο: {{τηλέφωνο}}

Κινητό: {{κινητό}}

Σας παρακαλώ να μου χορηγήσετε:

{{βεβαίωση_σπουδών}}

{{αναλυτική_βαθμολογία}}

{{κάτι_άλλο}}

Για να το χρησιμοποιήσω για:

{{κάθε_χρήση}}

{{εφορία}}

{{στρατολογία}}

Ο ΑΙΤΩΝ / Η ΑΙΤΟΥΣΑ

Λαμία {{ημέρα}}/{{μήνα}}/{{έτος}}

_ _ _ _ _ _ _ _ _ _ _ _ _

(υπογραφή)

Εικόνα 31. Αίτηση πριν την αλλαγή.

ΑΙΤΗΣΗ

Επώνυμο: ΠΑΠΑΪΩΑΝΝΟΥ

Όνομα: ΑΝΤΩΝΗΣ

Όν.Πατέρα: ΠΕΤΡΟΣ

Όν.Μητέρας: ΣΤΑΥΡΟΥΛΑ

Α.Μ.Τ: 21536

Εξάμηνο: 15

Διεύθυνση κατοικίας:

Οδός: ΘΕΡΜΟΠΥΛΩΝ

Αριθμός: 21 Τ.Κ: 35100

Πόλη: ΛΑΜΙΑ

Τηλέφωνο: 2104872329

Κινητό: 6971481652

Σας παρακαλώ να μου χορηγήσετε:

Αναλυτική βαθμολογία

Για να το χρησιμοποιήσω για:

Κάθε χρήση

Στρατολογία

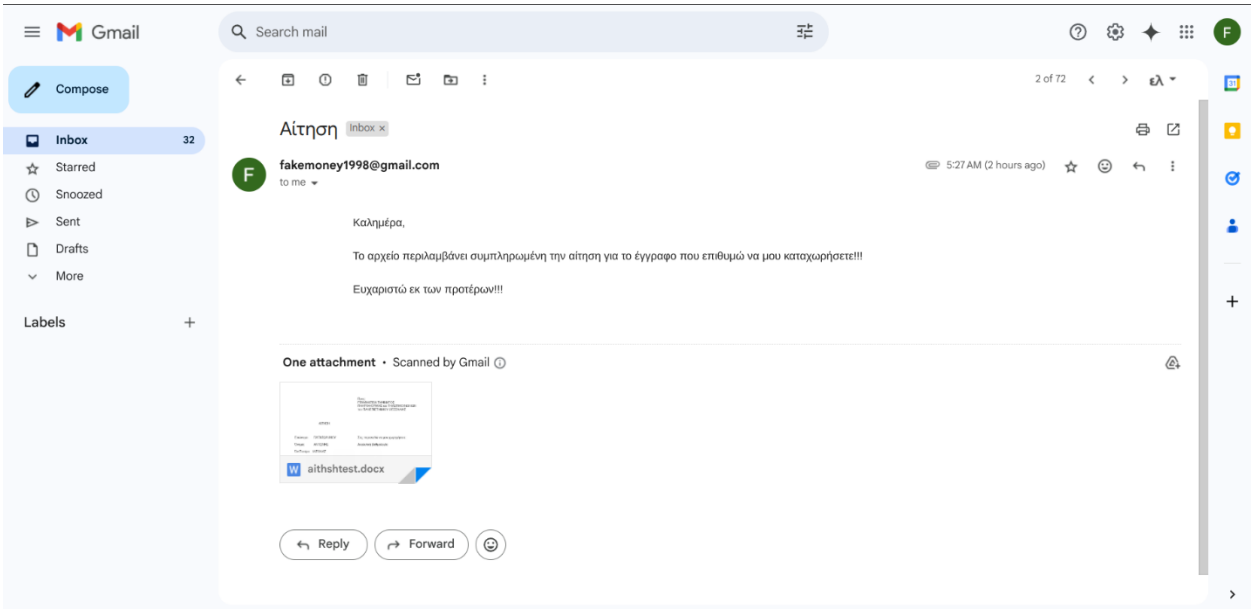
Λαμία 12/2/2025

Ο ΑΙΤΩΝ / Η ΑΙΤΟΥΣΑ

_ _ _ _ _ _ _ _ _ _ _ _ _

(υπογραφή)

Εικόνα 32. Αίτηση μετά την αλλαγή.



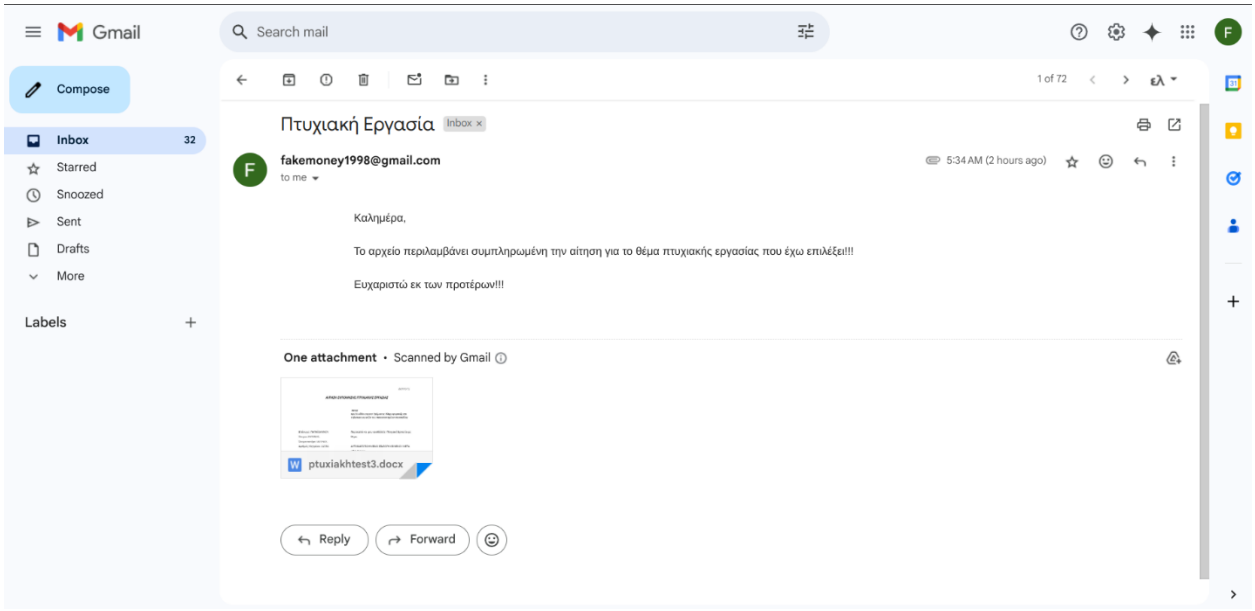
Εικόνα 33, Μήνυμα στο email αίτηση.

Επώνυμο: {{επώνυμο}}. Όνομα: {{όνομα}}. Όνομα πατέρα: {{όνομα_πατρός}}. Αριθμός Μητρώου: {{αριθμό_μητρώου}}. Δ/νση Κατοικίας Οδός: {{διεύθυνση_και_αριθμό}}. Τ.Κ. : {{ταχυδρομικός_κώδικας}}. Πόλη: {{πόλη}}. Τηλέφωνο: {{αριθμός_τηλεφώνου}}. e-mail.: {{email}}@uth.gr **Ο/Η ΕΠΙΒΛΕΠΩΝ/ΟΥΣΑ ΚΑΘΗΓΗΤΗΣ/ΤΡΙΑ** {{όνομα_και_επώνυμο_καθηγητή}}.	Παρακαλώ να μου αναθέσετε Πτυχιακή Εργασία με θέμα: {{θέμα_πτυχιακής}} με Επιβλέποντα/ουσα Καθηγητή/ήτρια: {{όνομα_και_επώνυμο_καθηγητή}}.
	Λαμία {{ημέρα}}/{{μήνα}}/{{έτος}}. Ο/Η Αιτών/ούσα

Εικόνα 34. Πτυχιακή εργασία πριν την αλλαγή.

Επώνυμο: ΠΑΠΑΪΩΑΝΝΟΥ. Όνομα: ΑΝΤΩΝΗΣ. Όνομα πατέρα: ΠΕΤΡΟΣ. Αριθμός Μητρώου: 21536. Δ/νση Κατοικίας Οδός: ΘΕΡΜΟΠΥΛΩΝ 21. Τ.Κ. : 35100. Πόλη: ΛΑΜΙΑ. Τηλέφωνο: 6971481652. e-mail.: emailtestv2@uth.gr@uth.gr **Ο/Η ΕΠΙΒΛΕΠΩΝ/ΟΥΣΑ ΚΑΘΗΓΗΤΗΣ/ΤΡΙΑ** ΚΩΝΣΤΑΝΤΙΝΟΣ.	Παρακαλώ να μου αναθέσετε Πτυχιακή Εργασία με θέμα: ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΗ ΕΞΑΓΩΓΗ ΚΕΙΜΕΝΟΥ ΜΕΤΑ ΑΠΟ ΟΜΙΛΙΑ με Επιβλέποντα/ουσα Καθηγητή/ήτρια: ΚΩΝΣΤΑΝΤΙΝΟΣ.
	Λαμία 12/2/2025. Ο/Η Αιτών/ούσα

Εικόνα 35. Πτυχιακή εργασία μετά την αλλαγή.



Εικόνα 36. Μήνυμα στο email πτυχιακή εργασία.

ΚΕΦΑΛΑΙΟ 6 ΠΡΟΒΛΗΜΑΤΑ, ΣΥΜΠΕΡΑΣΜΑΤΑ, ΒΕΛΤΙΩΣΕΙΣ ΚΑΙ ΕΠΕΚΤΑΣΕΙΣ

6.1 ΠΡΟΒΛΗΜΑΤΑ ΠΟΥ ΠΑΡΟΥΣΙΑΣΤΗΚΑΝ

Για τα προβλήματα που συνάντησα βρήκα λύση σε πολύ σύντομο διάστημα. Ας ξεκινήσω με το μικρόφωνο. Όταν εγκατέστησα την βιβλιοθήκη για το speech recognition στον σταθερό υπολογιστή στο Virtualbox, συνέδεσα το μικρόφωνο και δεν μπορούσε να λάβει κάποιο ήχο. Το πρόβλημα ήταν ότι έπρεπε το μικρόφωνο να είναι συνδεδεμένο πριν την ενεργοποίηση του Virtualbox αλλιώς δεν το αναγνώριζε. Ακόμα και στο Laptop συνάντησα πρόβλημα με το μικρόφωνο. Τα laptop έχουν δικό τους ενσωματομένο μικρόφωνο το οποίο ενώ φαινόταν ότι δέχεται ήχο, δεν μπορούσε να το εκτυπώσει κατά την διάρκεια της εκτέλεσης του κώδικα. Η λύση ήταν να συνδέσω το ίδιο μικρόφωνο που χρησιμοποίησα στον σταθερό υπολογιστή. Υπήρχαν ορισμένα προβλήματα με την ένταση του ήχου που με κατάλληλες ρυθμίσεις διορθώθηκαν.

Η χρήση της βιβλιοθήκης του docxtpl για να λειτουργήσει σωστά ήθελε τις εξής προυποθέσεις. Την ύπαξη δύο ίδιων αντικειμένων, ένα σε μια λίστα στο πρόγραμμα και το άλλο στα άγγιστρα στο έντυπο. Για την αντικατάσταση απαιτήθηκε η δημιουργία λεξικού μεταξύ των λιστών αυτών. Όταν ξεκίνησα να κάνω τεστ και να δίνω διάφορες λέξεις μου εκτύπωνε xml error. Αυτό το error έφυγε όταν αφαίρεσα τα κενά ανάμεσα σε λέξεις των έτοιμων αντικειμένων (π.χ., Όνομα πατέρα). Η ύπαρξη του κενού ανάμεσα στη λέξη όνομα και πατέρα αντικαταστήθηκε με μια κάτω παύλα η οποία και έλυσε αυτό το πρόβλημα.

Δεν δεχόταν να καλέσω τη συνάρτηση για την συνθήκη τερματισμού μέσα από την συνάρτηση του speech recognition, αλλά ούτε και την δυνατότητα να κάνω ένα απλό έλεγχο για την λέξη κλειδί που τερμαρίζει το πρόγραμμα. Για το πρόβλημα αυτό δεν κατάφερα να βρω τη λύση απλώς δημιούργησα μια συνάρτηση η οποία έτρεχε αμέσως μόλις επέστρεφε την τιμή μετά την ομιλία του χρήστη.

Στο Laptop εγκατέστησα μια έκδοση των linux τα Ubuntu 22.10. Μετά από πολλές προσπάθειες ακολουθώντας οδηγίες εγκατάστασης, δεν κατέστη εφικτό να γίνει εγκατάσταση των βιβλιοθηκών και εργαλείων γιατί απλά το λειτουργικό δεν τα υποστήριζε. Επέστρεψα στα Ubuntu

22.04.

Το αποτέλεσμα του αρχείου που αποθηκευόταν, είχε την ίδια ονομασία την οποία έδινα από προεπιλογή μέσα από τις εντολές του κώδικα. Για αποφυγή προβλημάτων χρησιμοποίησα την πρόσθεση μεταξύ δύο συμβολοσειρών οι οποίες ένωναν το επίθετο του χρήστη που είναι το πρώτο στοιχείων σε κάθε λίστα με το είδος από το έντυπο που επέλεγαν για την αυτόματη συμπλήρωση. Ο διαχωρισμός του επιθέτου και ονόματος από το έντυπο (αίτηση ή πτυχιακή) πραγματοποιήθηκε με τη χρήση της κάτω παύλας. Είναι εφικτό με κατάλληλες τροποποιήσεις να μπορέσει ο χρήστης να δώσει το όνομα που επιθυμεί στο αρχείο και απλά να προσθέσω την κατάληξη του αρχείου στο τέλος.

6.2 ΣΥΜΠΕΡΑΣΜΑΤΑ

Μέσα από την έρευνα, την υλοποίηση και των εμποδίων που συνάντησα σε αυτή την εργασία, συνειδητοποίησα την πρόοδο που έχει επιτύχει ο άνθρωπος μέσα στα χρόνια, στον τομέα της τεχνολογίας αλλά και στον συνεχή καινοτόμο τρόπο χρήσης της. Μέσα από αυτήν έμαθα από την αρχή την εξέλιξη μιας λειτουργίας που συνδέεται σε αμέτρητες περιπτώσεις χρήσης της, βοηθώντας σε πολλούς τομείς. Τα εμπόδια που συνάντησα στο κομμάτι του hardware και software με έφεραν πιο κοντά στο ρόλο του προγραμματιστή αλλά ταυτόχρονα απαίτησαν και φαντασία για να μετατροπή ορισμένων εντολών ώστε να είναι εφικτό ένα σωστό αποτέλεσμα. Έχει γίνει πλέον πολύ αισθητό ότι η εξέλιξη της τεχνολογίας δεν έχει ταβάνι και τα περιθώρια της είναι απεριόριστα. Με κάθε βήμα που γίνεται μια νέα εποχή ανθίζει για πολλούς τομείς αλλά και για τους ίδιους τους ανθρώπους που τη δέχονται ως προέκταση του εαυτού τους.

Η ένταξη αυτής της ιδέας για ένα εκπαιδευτικό ίδρυμα είναι μια καλή κίνηση για να έλξει τους μαθητευόμενους πιο κοντά στην τεχνολογία και να αποτελέσει πηγή έναρξης για την υλοποίηση κάποιου ανάλογου μοντέλου που θα στοχεύει στην καλυτέρευση και διευκόλυνσή τους.

6.3 ΒΕΛΤΙΩΣΕΙΣ

- Χρήση κάποιου μοντέλου αναγνώρισης τόνων στις λέξεις:

  Μετά την ολοκλήρωση της εργασίας, σκέφτηκα τι μπορεί να ήταν αυτό που θα την βελτίωνε, έτσι ώστε να μπορεί να προσφέρει πιο σωστές απαντήσεις. Στην περίπτωση της ελληνικής γλώσσας, θα μπορούσε να χρησιμοποιηθεί ένα μοντέλο μηχανικής μάθησης το οποίο θα αναγνώριζε καλύτερα το σημείο όπου η ένταση της φωνής του φοιτητή είναι μεγαλύτερη ώστε να τοποθετούνται οι τόνοι και να μην γίνονται πιθανά λάθη σε λέξεις που ακούγονται σχεδόν το ίδιο. Ας πάρουμε το όνομα Φάνης ως παράδειγμα για να κατανοήσουμε καλύτερα τη συμαντικότητα αυτής της βελτιωμένης λειτουργίας. Σε περίπτωση που το όνομα αυτό χρησιμοποιηθεί ενδέχεται το μικρόφωνο να το λάβει σαν Φανή αντί για Φάνη, κάτι που θα οδηγούσε σε λάθος.

- Μεγαλυτερο εύρος από μη επιτρεπτές λέξεις:

  Μια καλύτερη λίστα με περισσότερες λέξεις κλειδιά για την απαγόρευση μη επιτρεπτών λέξεων από τους φοιτητές κατά τη χρήση του chatbot, ώστε να μην δημιουργούνται ηθικά ζητήματα και να γίνεται αποφυγή σκόπιμης ή μη χρήσης τους.

6.4 ΕΠΕΚΤΑΣΕΙΣ

- Αναγνώριση γλώσσας:

  Θα ήταν πιο εντυπωσιακό αν καλούσαμε μέσα στο πρόγραμμα κάποια βιβλιοθήκη που θα αναγνώριζε τη γλώσσα μετά από την ομιλία. Για παράδειγμα η κλήση της βιβλιοθήκης αναγνώρισης ομιλίας χρησιμοποιεί από προεπιλογή την αγγλική γλώσσα, οπότε η χρήση μιας ελληνικής γλώσσας θα μας έβγαζε ένα λάθος αποτέλεσμα. Η ‘κατασκευή’ του προγράμματος αυτού έγινε με γνώμονα τη χρήση ενός μόνο μικροφώνου όπου θα ζητηθεί από το φοιτητή να επιλέξει τη γλώσσα που θα χρησιμοποιήσει κάνοντας την επιλογή του με τις λέξεις ‘Greek και English’. Η χρήση μιας βιβλιοθήκης για την αναγώριση της ελληνικής γλώσσας ακόμα και αν έχουμε την βιβλιοθήκη του speech recognition στα αγγλικά ή και το ανάστροφο. Έτσι θα μπορούσε οποιοσδήποτε φοιτητής να μίλαγε και να γινόταν η επιλογή της γλώσσας αυτόματα με βάση την απάντηση του. Για παράδειγμα ‘Καλημέρα’ η ‘ελληνικά’ που θα σήμαινε ότι έχουμε τη γλώσσα ελληνικά.

- Υποστήριξη περισσότερων γλωσσών:

  Η εργασία αυτή υλοποιήθηκε για την Ελληνική και Αγγλική γλώσσα. Φανταστείτε να μπορούσαν όλοι να επικοινωνήσουν στη μητρική τους γλώσσα και να γίνεται παραμετροποίηση της γραμματοσειράς στην ελληνική γλώσσα, ώστε να γίνεται κατανοητή από το ανθρώπινο δυναμικό που θα την λάβει, είτε πρόκειτε για κάτι απλό όπως η προσκόμιση ενός εγγράφου ή για κάτι πιο περίπλοκο όπως η δήλωση μιας πτυχιακής, ώπου χρειάζεται περισσότερα στοιχεία και επικοινωνία. Αυτό θα εξηπηρετούσε όλους όσους έπαιρναν μέρος στη διαδικασία.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] Wikipedia “Speech recognition” (https://en.wikipedia.org/wiki/Speech_recognition)
- [2] Sean R Eddy, Nature Biotechnology “What is a hidden Markov model?” (https://www.nature.com/articles/nbt1004-1315)
- [3] IBM “IBM Shoebox” (https://en.wikipedia.org/wiki/IBM_Shoebox)
- [4] David Amos “The Ultimate Guide To Speech Recognition With Python” (https://storage.googleapis.com/kaggle-forum-message-attachments/882826/15829/Ultimate%20Guide%20To%20Speech%20Recognition%20With%20Python.pdf?)
- [5] Sonix “A short history of speech recognition” (https://sonix.ai/history-of-speech-recognition)
- [6] Transkriptor “12 Types of speech recognition” (https://transkriptor.com/speech-recognition-types/)
- [7] Analog “ADC Input Noise: The Good, The Bad, and The Ugly. Is No Noise Good Noise?” (https://www.analog.com/en/resources/analog-dialogue/articles/adc-input-noise.html)
- [8] Springer Nature “Connectionist Temporal Classification” (https://link.springer.com/chapter/10.1007/978-3-642-24797-2_7)
- [9] IEEE Xplore “The Viterbi algorithm” (https://ieeexplore.ieee.org/document/1450960/metrics#metrics)
- [10] Journal Of Computer Science and Technology “Comparison of different implementation of MFCC” (https://link.springer.com/article/10.1007/bf02943243)
- [11] Wikipedia “Gunnar Fant” (https://en.wikipedia.org/wiki/Gunnar_Fant)
- [12] Wikipedia “Ludimar Hermann” (https://en.wikipedia.org/wiki/Ludimar_Hermann)

- [13] Wikipedia “Chatbot” (https://en.wikipedia.org/wiki/Chatbot)
- [14] IBM “What is a chatbot” (https://www.ibm.com/think/topics/chatbots)
- [15] ScienceDirect, Adamopoulou, E., & Moussiades, L. “Chatbots: History, Technology, and Applications” (https://www.sciencedirect.com/science/article/pii/S2666827020300062)
- [16] Wikipedia “Siri” (https://en.wikipedia.org/wiki/Siri)
- [17] Wikipedia “Chatgpt” (https://en.wikipedia.org/wiki/ChatGPT)
- [18] Wikipedia “Amazon Alexa” (https://en.wikipedia.org/wiki/Amazon_Alexa)
- [19] Wikipedia “Google Assistant” (https://en.wikipedia.org/wiki/Google_Assistant)
- [20] MT Zemcik “A Brief History of Chatbots” [pdf] (https://www.researchgate.net/profile/Tomas-Zemcik/publication/336734161_A_Brief_History_of_Chatbots/links/5dc1bc51a6fdcc21280872a3/A-Brief-History-of-Chatbots.pdf)\
- [21] Neliti “An Insterlligent Behaviour Shown by Chatbot System” (https://www.neliti.com/publications/263312/an-intelligent-behaviour-shown-by-chatbot-system)