ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Speech to text Technologies for People with Deafness and hearing Loss

Ησαΐας Κωνσταντίνος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κωνσταντίνος Κολομβάτσος

Λαμία Φεβρουάριος έτος 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

SPEECH TO TEXT TECHNOLOGIES FOR PEOPLE WITH DEAFNESS AND HEARING LOSS

ΗΣΑΙΑΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κωνσταντίνος Κολομβάτσος

Λαμία Φεβρουάριος έτος 2025

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

Speech to text Technologies for People with Deafness and hearing Loss

Isaias Konstantinos

FINAL THESIS

ADVISOR

Konstantinos Kolomvatsos

Lamia February year 2025

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

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

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

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

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

Ημερομηνία: ...../...../20.....

Ο - Η Δηλ.

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

ΠΕΡΙΛΗΨΗ

Η συγκεκριμένη πτυχιακή εργασία αφορά την ανάπτυξη ενός προγράμματος για την “μετάφραση” κειμένου από τον χρήστη είτε από την αναγνώριση της φωνής του είτε (για μεγαλύτερη ευκολία και ακρίβεια) από εισαγωγή μέσο πληκτρολογίου. Ο συγκεκριμένος μεταφραστής θα έχει την δυνατότητα να αντιστοιχεί κάθε γράμμα του χρήστη με ένα συγκεκριμένο σύμβολο στην νοηματική γλώσσα. Ο μεταφραστής έχει την δυνατότητα αναγνώρισης φωνής στα Αγγλικά και Ελληνικά. Η συγκεκριμένη εφαρμογή εκτός από την αναγνώριση και αντιστοίχιση γραμμάτων με σύμβολα νοηματικής, έχει και την δυνατότητα(Μόνο στην αγγλική γλώσσα) να απλοποιεί ως ένα βαθμό και το κείμενο που δίνει ο χρήστης σαν εισαγωγή όταν μιλάει. Το συγκεκριμένο χαρακτηριστικό της εφαρμογής υλοποιείται μέσο μοντέλου μηχανικής μάθησης το οποίο έχει “προπονηθεί” σε ‘ένα μεγάλο σύνολο δεδομένων με απλές προτάσεις και το απλοποιημένο περιεχόμενό τους. Συνοψίζοντας όλα τα παραπάνω υλοποιούνται από την χρήση τεχνολογιών όπως αναγνώριση φωνής, chatbot(μόνο για εκφώνηση έτοιμων διαλόγων) , βιβλιοθήκη επεξεργασίας και προβολής εικόνων, βιβλιοθήκη για την καταγραφή ημερομηνιών (χρησιμοποιείται στο csv για την σωστή και οργανωμένη καταγραφή δεδομένων), βιβλιοθήκη για την συμπερίληψη ANN(Advanced Neural Networks) όπως το t5-small και BART για NLP (Natural Language Processing) που ευθύνονται για την απλοποίηση κειμένου στην εφαρμογή μας.

ABSTRACT

This thesis project involves the development of a program for “translating” text from the user either through voice recognition or (for greater ease and accuracy) through keyboard input. This translator will have the capability to map each letter entered by the user to a specific symbol in sign language. The translator can recognize speech in both English and Greek. In addition to recognizing and mapping letters to sign language symbols, the application also has the capability (only in English) to simplify, to some extent, the text provided by the user during speech input. This specific feature of the application is implemented through a machine learning model that has been “trained” on a large dataset of simple sentences and their simplified content. In summary, all of the above functionalities are achieved using technologies such as voice recognition, a chatbot (only for rendering pre-defined dialogues), a library for image processing and displaying it, a library for date recording (which is used in the CSV for proper and organized data logging), and a library for the inclusion of advanced neural networks (ANN) such as T5-small and BART for natural language processing (NLP) which are responsible for the text simplification.

Table of Contents

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή και κίνητρο

1.1 Λίγα λόγια για τη νοηματική γλώσσα

Η νοηματική γλώσσα είναι μια προσπάθεια του ανθρώπου να εκφραστεί μέσο νοημάτων με τας χέρια του, με την στάση του σώματος του αλλά και τις εκφράσεις του προσώπου του για να μπορέσει να εκφράσει κάποια συναισθήματα ή καταστάσεις χωρίς να μιλήσει. Η νοηματική γλώσσα διαθέτει λεκτικές και συντακτικές δομές για να εκφράσει οποιοδήποτε αφηρημένη έννοια, και καθημερινές ανάγκες μας (καλημέρα, χαίρω πολύ, πεινάω κ.α.). Οι γλωσσολόγοι διακρίνουν τις φυσικές νοηματικές γλώσσες από άλλα συστήματα που προέρχονται από αυτές , όπως είναι οι τεχνητοί ανθρώπινοι κώδικες (αλγόριθμοι, γλώσσες προγραμματισμού κ.α.), η ομιλούμενες γλώσσες, η οικιακή, η "βρεφική" και νοήματα που αποκτούνται από την βρεφική ηλικία. Έτσι όπως υπάρχουν πολλές ομιλούμενες γλώσσες υπάρχουν και πολλές νοηματικές γλώσσες . Για παράδειγμα, η ASL (American sign language) είναι διαφορετική από την BSL (British sign language) και τίς άλλες υπάρχουσες γλώσσες [1]. Σύμφωνα με τον ethnologue υπάρχουν περισσότερες από 130 νοηματικές γλώσσες παγκοσμίως. Αυτές οι γλώσσες χρησιμοποιούνται από διάφορες κοινότητες κωφών και ημί-κωφων και ποικίλλουν από χώρα σε χώρα καθώς και σε διαφορετικές περιοχές εντός των ίδιων χωρών [2]. Σύμφωνα με τον Π.Ο.Υ (παγκόσμιος οργανισμός υγείας) περισσότερο από το 5% του παγκόσμιου πληθυσμού ή περίπου 430 εκατομμύρια άνθρωποι έχουν μόνιμη βλάβη στο ακουστικό τους σύστημα, είτε υπάρχει εκ γενετής ή μετά από ατύχημα. Μέχρι το 2050 εκτιμάται ότι πάνω από 700 εκατομμύρια άνθρωποι θα έχουν προβλήματα ακοής [3]. Να σημειωθεί ότι η νοηματική γλώσσα δεν θα πρέπει να συσχετιστεί με την γλώσσα του σώματος , που θεωρείται ένα είδος μη λεκτικής επικοινωνίας [4]. Πρώτες καταγεγραμμένες αναφορές για την νοηματική γλώσσα μπορούμε να βρούμε στο βιβλίο του Πλάτωνα "Κρατύλο" από συνομιλίες του με τον Σωκράτη [5]. Σε αυτό το σημείο να αναφέρω ότι τα άτομα με προβλήματα ακοής βρίσκονταν στο περιθώριο, διότι εκείνη την εποχή επικρατούσε η άποψη ότι ή γλώσσα μπορεί να μαθευτεί μόνο ακούγοντας τον προφορικό λόγο. Για παράδειγμα στην ρώμη σύμφωνα με το ρωμαϊκό δίκαιο επικρατούσε η άποψη ότι οι άνθρωποι που γεννιούνται κωφοί δεν είχαν ίσα δικαιώματα με τους υπόλοιπους πολίτες καθώς θεωρούταν ότι "δεν καταλάβαιναν τίποτα". Αυτή η προκατάληψη ενάντια στα κωφά άτομα άρχισε να εξαλείφεται την περίοδο της αναγέννησης [6]. Μάλιστα το πρώτο άτομο που δημιούργησε μια "επίσημη" νοηματική γλώσσα είναι ο Ισπανός μοναχός Pedro Ponce de Leon τον 16ο αιώνα μΧ. Στη συνέχεια ένας άλλος Ισπανός κληρικός ο Juan Pablo Bonet κυκλοφόρησε το 1620 το πρώτο έργο για την εκμάθηση νοηματικής γλώσσας σε ανθρώπους με προβλήματα ακοής (Είναι από τα πρώτα έργα που έχουμε καταγεγραμμένα). Παρακάτω είναι μερικές από τις σελίδες του βιβλίου του Pedro [7].

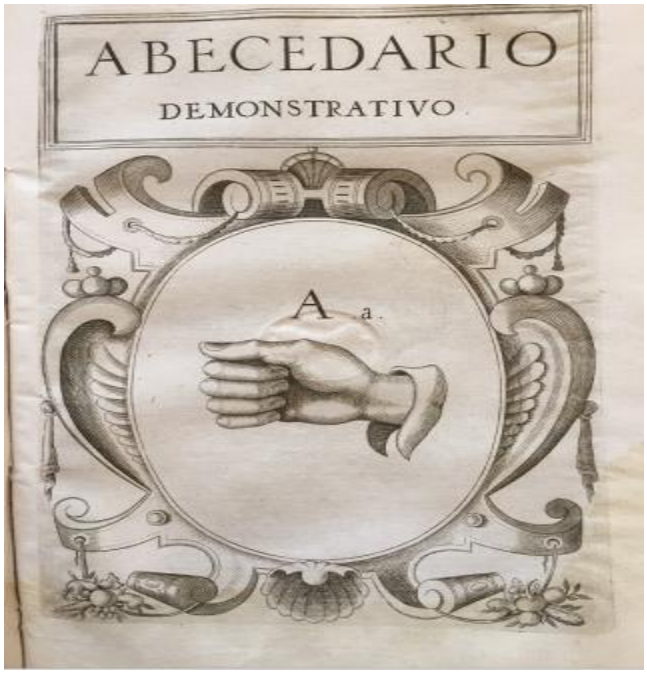

Figure 1 Εικόνα από το βιβλίο του Juan Pablo Bonet https://shorturl.at/qtZB5

Figure 2 Εικόνα από το βιβλίο του Juan Pablo Bonet https://shorturl.at/zxkDn .

Η ASL(American sign language) έχει ρίζες από την Αμερικανική σχολή για κωφούς που ιδρύθηκε στο Hartford, Connecticut το 1817 [8]. Η συγκεκριμένη σχολή ιδρύθηκε από τον απόφοιτο του Yale Thomas Hopkins Gallaudet [9] και εκείνη την εποχή θεωρούταν σχολή για "χαζούς".

Στην νοηματική γλώσσα ASL για να γίνει σωστά η εκτέλεση των λέξεων είναι απαραίτητες η παρακάτω 5 παράμετροι [10].

1. Σχηματισμός χεριού

2. Προσανατολισμός παλάμης
3. κίνηση του χεριού
4. Τοποθεσία χεριού
5. Εκφράσεις σώματος (NMS)

Σχηματισμός χεριού

Σχηματισμοί χεριών αποτελούνται από το αλφάβητο που υπάρχει με της κινήσεις των χεριών σε διάφορες παραλλαγές σχημάτων όπως στην εικόνα παραπάνω.

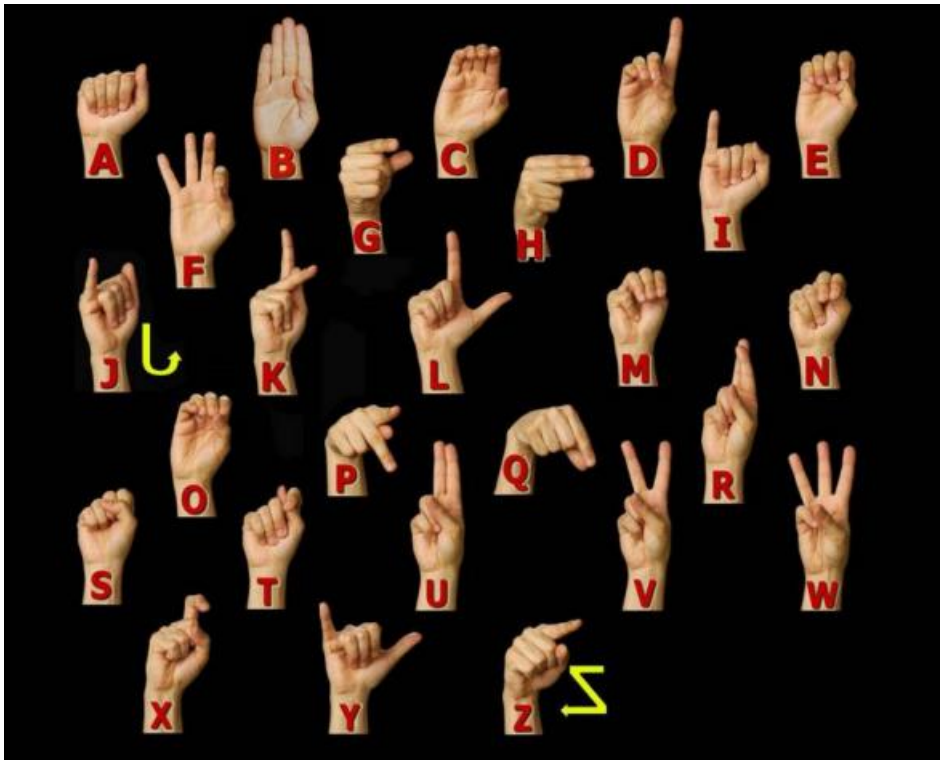

Figure 3 Αλφάβητο της ASL https://shorturl.at/iMmTu.

Προσανατολισμός Παλάμης

Ο σχηματισμός αναφέρεται σε ποια κατεύθυνση βλέπει η παλάμη για ένα συγκεκριμένο νοηματικό σήμα.

- Παλάμη προς τα έξω
- Παλάμη προς τα μέσα
- Παλάμη σε οριζόντια θέση
- Παλάμη βλέπεις προς τα Δεξιά/Αριστερά
- Παλάμη βλέπεις προς παλάμη
- Παλάμη βλέπει προς τα Πάνω/Κάτω

Κίνηση του χεριού

Ένα νόημα μπορεί να διακριθεί με κάποιες από τις παραπάνω αναριθμηζόμενες κινήσεις.

- Κυκλική κίνηση
- Πάνω και κάτω
- Προς τα μπροστά
- Προς τα πίσω
- χτύπημα
- Μπροστά και πίσω
- Διάφορα κουνήματα

Τοποθεσία χεριού

Η τοποθεσία είναι ο χώρος στον οποίο λαμβάνει χώρα το νόημα που κάνει ο άνθρωπος σε σχέση με την τοποθεσία του σώματος του . Παραπάνω είναι όλες οι τοποθεσίες που μπορούν να γίνουν τα νοήματα, να σημειωθεί ότι ένα νόημα μπορεί να αρχίσει για παράδειγμα από το κεφάλι και να ολοκληρωθεί στο στήθος.

- Πιγούνι
- Ώμος
- Μπροστά από το σώμα
- Αριστερά και δεξιά του σώματος
- Μέτωπο

Εκφράσεις σώματος

Η παραπάνω εκφράσεις σώματος αναφέρονται ως συμπληρωματικές για να υπάρχει μεγαλύτερη κατανόηση σε νοήματα.

- Κούνημα κεφαλιού (Καταφατικά/αρνητικά)
- Κούνημα φρυδιών
- Κούνημα μύτης
- Νοήματα με τα μάτια
- Χρήση των χειλιών

1.2 Σκοπός και κίνητρο της πτυχιακής εργασίας

Ο σκοπός της συγκεκριμένης πτυχιακής είναι η εξοικείωση ανθρώπων χωρίς νοηματικά προβλήματα με το αλφάβητο της ASL και GSL. Είναι μια απλή προσπάθεια της τριβής ανθρώπων που δεν έχουν ξαναδεί νοήματα μέσο της ομιλίας τούς ή από την είσοδο στο πληκτρολόγιο στο πρόγραμμα που έχω υλοποιήσει. Περαιτέρω θα ήθελα μελλοντικά να ασχοληθώ με την αλληλεπίδραση προγραμμάτων χρήστη-υπολογιστή αλλά κυρίως με την κατανόηση της Νοηματικής και δημιουργία μιας πιο εξελιγμένης εφαρμογής.
Αυτό που με ώθησε και ήθελα να ασχοληθώ με το συγκεκριμένο θέμα είναι το συναισθηματικό κομμάτι, καθώς με μια απλή εφαρμογή μπορείς να κάνεις πιο εύκολη την ζωή κάποιου με κάποιες γραμμές κώδικα. Πέρα από την ευκολία της μετάφρασης των νοημάτων μπορείς να διευκολύνεις και κάποιον που θέλει να κάνει μια αρχική εισαγωγή στην νοηματική γλώσσα. Η συγκεκριμένη εφαρμογή για εμένα δεν απευθύνεται σε άτομα που γνωρίζουν ήδη την Νοηματική γλώσσα αλλά σε άτομα που σε μια μεγαλύτερη ή και μικρότερη ηλικία θέλουν να βοηθηθούν στην εκμάθηση της . Κανείς δεν ξέρει πότε θα γνωρίσει κάποιον σημαντικό άνθρωπο που θα είναι κωφός και θα χρειαστεί να επικοινωνήσει μαζί του μέσο νοημάτων , οι περισσότεροι από εμάς θα θέλαμε να γνωρίζουμε την γλώσσα του για να μπορούμε να εμβαθύνουμε την σχέση μας με το συγκεκριμένο άτομο. Στην τελική αυτός είναι ο σκοπός της εφαρμογής, μια μικρή εισαγωγή στην νοηματική γλώσσα και το να φέρουμε τους ανθρώπους πιο κοντά.

1.3 Εκμάθηση γραμμάτων στην νοηματική

Η εκμάθηση του αλφαβήτου της Νοηματικής γλώσσας είναι σημαντική για την ανάπτυξη επικοινωνίας με βαρήκοα και κωφά άτομα. Η εκμάθηση είναι πολύ πιο αποδοτική με άτομα που την γνωρίζουν και μιλάνε, αλλά δεν έχουν όλοι την ευκαιρία να έχουν στον κύκλο τους ένα άτομο που να μιλάει και να χρησιμοποιεί την νοηματική γλώσσα. Για αυτό τον λόγο με το συγκεκριμένο πρόγραμμα ο μέσος χρήστης θα έχει την δυνατότητα να μιλάει με το πρόγραμμα ή να γράφει τι θέλει να πει και με τον καιρό να αρχίσει να μαθαίνει το κάθε νόημα σε ποιο γράμμα αντιστοιχεί. Παρακάτω είναι το αλφάβητο της ASL.

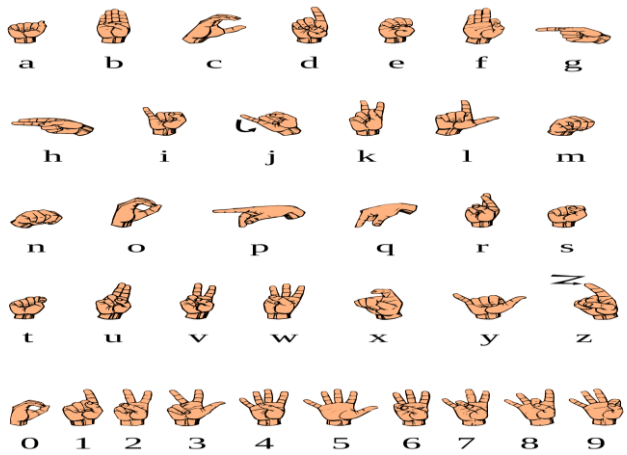

Figure 4 Αλφάβητο της ASL https://shorturl.at/pyVqo .

1.4 Εισαγωγή ατόμων στη Νοηματική γλώσσα(με ή χωρίς ακουστικά προβλήματα)

Πέρα από την βοήθεια στην εκμάθηση της νοηματικής το πρόγραμμα βοηθάει και στην εισαγωγή ατόμων που δεν έχουν καμιά επαφή με την Νοηματική γλώσσα στα σύμβολα που χρησιμοποιούνται. Μπορεί να υπάρξουν περιπτώσεις οι οποίες δεν έχουν ξαναδεί κανένα σύμβολο. Ακόμα η νοηματική προσφέρει μια πολυδιάστατη επικοινωνία στον άνθρωπο. Ενισχύει την ικανότητα επικοινωνίας του είτε έχει κάποιο ακουστικό πρόβλημα ή απλά την χρησιμοποιεί ως εναλλακτικό μέσο επικοινωνίας, επιπλέον συμβάλει και στην κοινωνική του συνοχή με άτομα της κωφής κοινότητας προάγοντας την συμπερίληψη και την κατανόηση διαφορετικών πολιτισμών. Πέρα από όλα αυτά ανοίγονται και επαγγελματικής φύσεως ευκαιρίες διότι η εκμάθηση της νοηματικής γλώσσας μπορεί να προσφέρει εργασία σε όλους τους επαγγελματικούς χώρους όπου υπάρχει η κοινότητα των κωφών [11].
Τέλος έχουν δείξει έρευνες ότι με την εκμάθηση της νοηματικής δίνεται η δυνατότητα ανάπτυξης γλωσσικών αλλά και γνωστικών δεξιοτήτων σε άτομα όλων των ηλικιών και πως αυτά τα άτομα έχουν βελτιωμένη νοητική ευελιξία, επίλυση καθημερινών προβλημάτων και αποκτούν δημιουργική σκέψη [12].

ΚΕΦΑΛΑΙΟ 2 Μοντέλα μηχανικής μάθησης

2.1 Text-summarization

Η περίληψη του κειμένου είναι μια διαδικασία που αποσκοπεί στη μείωση της πολυπλοκότητας του λόγου, ώστε το κείμενο να γίνει μικρό σε μέγεθος και πιο κατανοητό, ιδιαίτερα από άτομα με περιορισμένες αναγνωστικές δεξιότητες ή γλωσσική επάρκεια. Στην περίπτωση του προγράμματος που υλοποιώ θέλω να προσθέσω την απλοποίηση/περίληψη κειμένου για την διευκόλυνση του χρήστη στην μετάφραση της νοηματικής γλώσσας. Με λίγα λόγια το σκεπτικό είναι ότι είτε από είσοδο μικροφώνου ή από το πληκτρολόγιο ο χρήστης θα μπορεί να δίνει την πρόταση που θέλει για μετάφραση και το πρόγραμμα με βάση τα δεδομένα που έχει εκπαιδευθεί θα υλοποιεί μια απλοποίηση/περίληψη του κειμένου που στην συνέχεια θα μεταφράζεται από το πρόγραμμα μου.

Η απλοποίηση/περίληψη (**Summarization)** κειμένου συνδέεται με τον κλάδο του **Natural Language processing (NLP)** όπου είναι ένας τομέας της επιστήμης υπολογιστών και ιδιαίτερα της τεχνητής νοημοσύνης. Η **NLP** ουσιαστικά επιτρέπει στους υπολογιστές να κατανοούν, να επεξεργάζονται και να αναπαράγουν την ανθρώπινη γλώσσα Ουσιαστικά με το **NLP** γίνεται η καταγραφή κειμένων από διαφορετικές πηγές όπως το πληκτρολόγιο η το μικρόφωνο και να εφαρμοστούν αυτόματα μέθοδοι απλοποίησης ή παραπάνω επεξεργασία από εκπαιδευμένα μοντέλα για να γίνει στην τελική η μετάφραση στην νοηματική γλώσσα με τα κατάλληλα νοήματα [13].

Ξεκινώντας η απλοποίηση κειμένου αποτελείται από την **Λεξιλογική**, **Συντακτική** και **Νοηματική** απλοποίηση [14].

- Η **Λεξιλογική** απλοποίηση αφορά την αναγνώριση σύνθετων λέξεων, όπως σπάνιων, τεχνικών ή αφηρημένων, και την αντικατάστασή τους με απλούστερα και πιο κατανοητά συνώνυμα. Εναλλακτικά, το κείμενο μπορεί να εμπλουτιστεί με ορισμούς, εικόνες ή βίντεο για να διευκολύνει την κατανόηση. Κατά τη διαδικασία αυτή, η αποσαφήνιση παίζει σημαντικό ρόλο, καθώς απαιτείται η επιλογή της πιο κατάλληλης έννοιας από μια λίστα συνωνύμων. Ωστόσο, η εξάρτηση από την πιο συχνή έννοια μιας λέξης μπορεί να οδηγήσει σε προβλήματα που χρειάζονται περαιτέρω έρευνα.
- Η **Συντακτική** απλοποίηση στοχεύει στη μείωση της πολυπλοκότητας των προτάσεων, όπως παθητικές φράσεις, μεγάλες προτάσεις και σχετικές προτάσεις. Αυτό επιτυγχάνεται μέσω αναδιάταξης, διαχωρισμού ή απλοποίησης των γραμματικών δομών, με στόχο την εξάλειψη περιττών πληροφοριών.
- Η **Νοηματική** απλοποίηση του λόγου διασφαλίζει ότι καμία ουσιαστική πληροφορία δεν χάνεται κατά τη λεξιλογική ή συντακτική απλοποίηση. Αυτό το βήμα επικεντρώνεται στη διατήρηση της συνοχής του κειμένου, αναλύοντας τις αναφορές και αντικαθιστώντας επαναλαμβανόμενες οντότητες ή απλοποιώντας τις ονοματικές φράσεις [15] [16].

2.2 BART (Bidirectional and Auto-Regressive Transformers)

Το **BART (Bidirectional and Auto-Regressive Transformers)** είναι ένας αυτόματος κωδικοποιητής αποθορυβοποίησής (**Denoising autoencoder**) που έχει υλοποιηθεί με ένα μοντέλο σχέσεων (sequence-to-sequence model) που μπορεί να χρησιμοποιηθεί σε ένα μεγάλο φάσμα εργασιών. Με λίγα λόγια είναι ένας συνδυασμός της αρχιτεκτονικής του **BERT (Bidirectional encoder)** και του **GPT (Left to Right Decoder)**.

Figure 5 Συνδυασμός του BERT+GPT αποτελεί το BART
https://shorturl.at/dNSPc.

Το **BART** έχει σχεδιαστεί από το Facebook AI και είναι αρκετά αποδοτικό, όταν ρυθμιστεί στις κατάλληλες παραμέτρους και δεδομένα, για να επιλύσει εργασίες **NLP (Natural language processing)**, όπως η δημιουργία κειμένου αλλά και εργασίες λογικής. Στην συγκεκριμένη περίπτωση θα το χρειαστούμε για να επιλύσουμε **εργασίες λογικής** δηλαδή περίληψη κειμένου (Text-summarization), αλλά αρχικά ας απεικονίσουμε το πώς είναι η βασική αρχιτεκτονική του μοντέλου. Όπως προαναφέρθηκα το **BART** μπορεί να αναφερθεί και ως ένας συνδυασμός παρόμοιας αρχιτεκτονικής του **BERT+GPT** τα οποία είναι παρόμοια μοντέλα.

- Το μοντέλο **BERT** λειτουργεί με την **αντικατάσταση τυχαίων tokens** με **ειδικές masks ([MASK] Element)** και κωδικοποιεί το κείμενο αμφίδρομα, δηλαδή λαμβάνοντας υπόψη το συμφραζόμενο και πριν και μετά το κάθε token. Ωστόσο τα **tokens** που δεν υπάρχουν προβλέπονται ανεξάρτητα, έτσι το BERT δεν είναι κατάλληλο για δημιουργία νέου κειμένου [18].

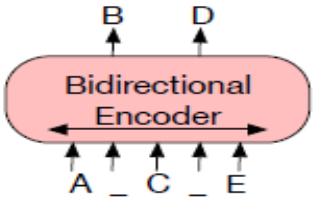

Figure 6 Κωδικοποιητής [17].

- Το μοντέλο **GPT**, τα tokens προβλέπονται **auto-regressively**, κάτι που επιτρέπει τη χρήση του για δημιουργία κειμένου. Ωστόσο το μοντέλο μπορεί να βασιστεί μόνο στα συμφραζόμενα που βρίσκονται στα αριστερά του κάθε token, γεγονός που σημαίνει ότι δεν μπορεί να

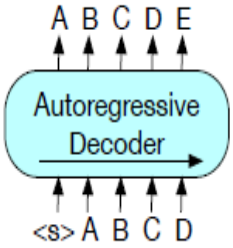

Figure 7 Αποκωδικοποιητής [17].

κατανοήσει πλήρως αμφίδρομες αλληλεπιδράσεις στο κείμενο [19].

- Στο μοντέλο **BART**, τα δεδομένα εισόδου στον κωδικοποιητή (**Encoder**) δεν χρειάζεται να ταιριάζουν με αυτά που παράγει ο αποκωδικοποιητής (**Decoder**), δίνοντας τη δυνατότητα για αυθαίρετες/τυχαίες μετατροπές θορύβου. Για παράδειγμα, ένα έγγραφο μπορεί να αλλοιωθεί αντικαθιστώντας τμήματα κειμένου με **mask symbols**. Αυτό το αλλοιωμένο έγγραφο κωδικοποιείται (**Encoded)** με ένα αμφίδρομο μοντέλο, και στη συνέχεια υπολογίζεται η πιθανότητα του αρχικού κειμένου μέσω ενός (**Auto-regressive Decoder)**. Στη φάση της βελτιστοποίησης, ένα μη αλλοιωμένο έγγραφο δίνεται και στον κωδικοποιητή (**Encoder)** και στον αποκωδικοποιητή (**Decoder)**, και χρησιμοποιούνται οι τελικές αναπαραστάσεις από την κρυφή κατάσταση (**Hidden state)** του αποκωδικοποιητή (**Decoder**) [20].

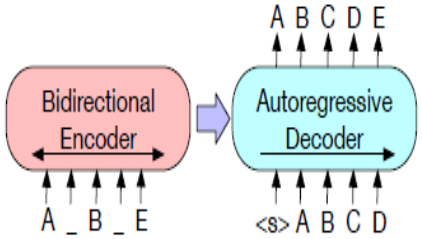

Figure 8 Κωδικοποιητής/Αποκωδικοποιητής[17].

Ας δούμε και λίγο πιο αναλυτικά τον **encoder-decode-transformer** [20] που χρησιμοποιείται και τον κώδικα για αυτόν. Η παρακάτω φωτογραφία είναι το βασικό πλαίσιο για το μοντέλο μας.

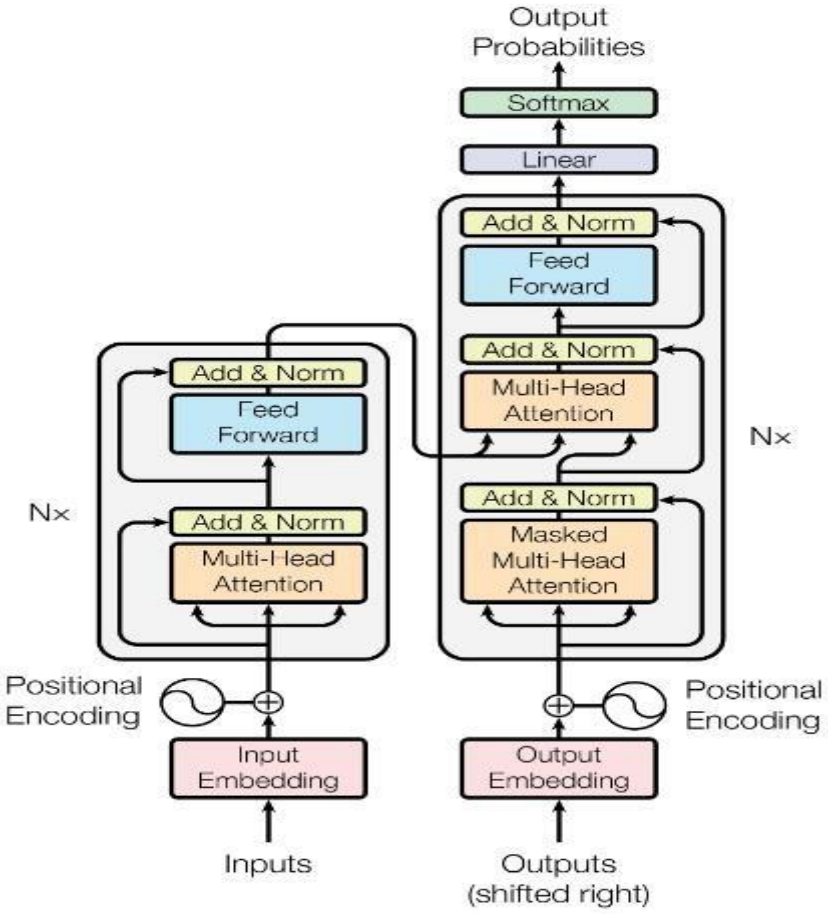

Figure 9 Κωδικοποιητής/Αποκωδικοποιητής Αναλυτικό γράφημα https://nlp.seas.harvard.edu/annotated-transformer/ [23].

Στην παραπάνω εικόνα βλέπουμε την αρχιτεκτονική κωδικοποιητή-αποκωδικοποιητή **(Encoder-Decoder) Transformer** είναι ένα πρωτοποριακό πλαίσιο στον τομέα του **Deep learning**, ειδικά ανεπτυγμένο για επεξεργασία διαδοχικών εργασιών δεδομένων, όπως **automatic translation**, η **text summarization** που χρειαζόμαστε στην περίπτωσή μας [22]. Ας ξεκινήσουμε με μια περιεκτική ανάλυση του αριστερού μέρους του transformer μας τον κωδικοποιητή (**Encoder)** που είναι υπεύθυνος για την επεξεργασία της ακολουθίας (**sequence).**

Multi-Head Self-Attention

Επιτρέπει στον κωδικοποιητή να εστιάζει σε διαφορετικά μέρη της ακολουθίας εισόδου (**input-sequence).** Με λίγα λόγια στα **Self-Attention** layers όλα τα **keys,values** και **queries** προέρχονται από το ίδιο σημείο, οπού σε αυτή την περίπτωση είναι το προηγούμενο **layer** στον κωδικοποιητή (**Encoder)** και έτσι κάθε θέση στον κωδικοποιητή μπορεί να λάβει υπόψιν το προηγούμενο **layer** στον κωδικοποιητή (**Encoder) .** Στην παρακάτω class **"MultiHeadedAttention"** με τα δοθέντα δεδομένα υπολογίζει το **Attention** σε διάφορα **Head** παράλληλα. Ο παρακάτω είναι ο μαθηματικός τύπος και μετά ο κώδικας.

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

where Q, K, and V are the query, key, and value matrices derived from the input.

Figure 10 Μηχανισμός στον Κωδικοποιητή/αποκωδικοποιητή
https://nlp.seas.harvard.edu/annotated-transformer/ [23].

Στην συνέχεια μπορούμε να δούμε και σε κώδικα πώς πραγματοποιείται αλλά και σε ένα απλό διάγραμμα που βλέπουμε που χρησιμοποιούνται οι τιμές αυτές.

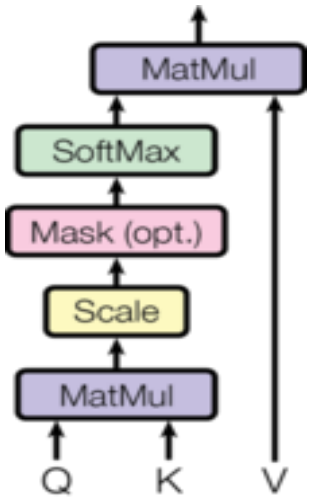

Figure 11 Εικόνα για την εφαρμογή του Attention
https://nlp.seas.harvard.edu/annotated-transformer/ [23].

```python
# %% id="qsoVxS5yTsqG"
def attention(query, key, value, mask=None, dropout=None):
    "Compute 'Scaled Dot Product Attention'"
    d_k = query.size(-1)
    scores = torch.matmul(query, key.transpose(-2, -1)) / math.sqrt(d_k)
    if mask is not None:
        scores = scores.masked_fill(mask == 0, -1e9)
    p_attn = scores.softmax(dim=-1)
    if dropout is not None:
        p_attn = dropout(p_attn)
    return torch.matmul(p_attn, value), p_attn
```

Figure 12 Κώδικας Attention σε python [https://github.com/ikoctac/annotated-transformer[22]](https://github.com/ikoctac/annotated-transformer[22]).

Η **MultiHeadedAttention** class που χρησιμοποιεί και το attention του παραπάνω κώδικα.

```python
# %% id="D2LBMKCQTsqH"
class MultiHeadedAttention(nn.Module):
    def __init__(self, h, d_model, dropout=0.1):
        "Take in model size and number of heads."
        super(MultiHeadedAttention, self).__init__()
        assert d_model % h == 0
        # We assume d_v always equals d_k
        self.d_k = d_model // h
        self.h = h
        self.linears = clones(nn.Linear(d_model, d_model), 4)
        self.attn = None
        self.dropout = nn.Dropout(p=dropout)

    def forward(self, query, key, value, mask=None):
        "Implements Figure 2"
        if mask is not None:
            # Same mask applied to all h heads.
            mask = mask.unsqueeze(1)
        nbatches = query.size(0)

        # 1) Do all the linear projections in batch from d_model => h x d_k
        query, key, value = [
            lin(x).view(nbatches, -1, self.h, self.d_k).transpose(1, 2)
            for lin, x in zip(self.linears, (query, key, value))
        ]

        # 2) Apply attention on all the projected vectors in batch.
        x, self.attn = attention(
            query, key, value, mask=mask, dropout=self.dropout
        )

        # 3) "Concat" using a view and apply a final linear.
        x = (
            x.transpose(1, 2)
            .contiguous()
            .view(nbatches, -1, self.h * self.d_k)
        )
        del query
        del key
        del value
        return self.linears[-1](x)
```

Figure 13 Εικόνα 13 Κώδικας MultiHeadAttention σε python [https://github.com/ikoctac/annotated-transformer[22]](https://github.com/ikoctac/annotated-transformer[22]).

Feed-Forward Network (FFN)

Εφόσον ολοκληρωθεί το πρώτο κομμάτι και υπολογισθεί το **Attention,** τα δεδομένα περνάνε μέσω ενός διπλού **Fully-Connected Network Layer.** Συγκεκριμένα το μοντέλο χρησιμοποιεί **Learned embeddings[24]** για την μετατροπή των **input-tokens** και **output-tokes** σε **Vectors χ** διαστάσεων. Επίσης χρησιμοποιούνται οι συναρτήσεις **Learned-linear-transformation** και **SoftMax** για να με μετατραπεί η έξοδος του αποκωδικοποιητή στα επόμενα **Predicted-Token-Probabilities.** Στο παραπάνω μοντέλο τα **Two-embedding-layers** μοιράζονται τα ίδια **weight-matrix** και την **pre-SoftMax linear transformation[24].** Τέλος στα **embedding layers** πολλαπλασιάζουμε τα **weights** με $\sqrt{d_{model}}$

```python
# %% id="6HHCemCxTsqH"
class PositionwiseFeedForward(nn.Module):
    "Implements FFN equation."

    def __init__(self, d_model, d_ff, dropout=0.1):
        super(PositionwiseFeedForward, self).__init__()
        self.w_1 = nn.Linear(d_model, d_ff)
        self.w_2 = nn.Linear(d_ff, d_model)
        self.dropout = nn.Dropout(dropout)

    def forward(self, x):
        return self.w_2(self.dropout(self.w_1(x).relu()))
```

Figure 14 PositionwiseFeedForward κώδικας
https://github.com/ikoctac/annotated-transformer[22].

Ο παραπάνω κώδικας υπολογίζει την μαθηματική φόρμουλα για το **FFN (Feed-Forward-Network).**

$$\mathrm{FFN}(x) = \phi(xW_1 + b_1)W_2 + b_2$$

where W_1 and W_2 are weight matrices, b_1 and b_2 are biases, and ϕ is an activation function (often ReLU).

Figure 15 Μαθηματική φόρμουλα για τον υπολογισμό στην Python
https://nlp.seas.harvard.edu/annotated-transformer/ [22].

Add & Norm

Οι υπολειπόμενες συνδέσεις μεταφέρουν την είσοδο με την έξοδο του **sublayer** και στην συνέχεια εφαρμόζουν **Layer-Normalization** στην αντίστοιχη class **LayerNorm.**

```python
class SublayerConnection(nn.Module):
    """
    A residual connection followed by a layer norm.
    Note for code simplicity the norm is first as opposed to last.
    """

    def __init__(self, size, dropout):
        super(SublayerConnection, self).__init__()
        self.norm = LayerNorm(size)
        self.dropout = nn.Dropout(dropout)

    def forward(self, x, sublayer):
        "Apply residual connection to any sublayer with the same size."
        return x + self.dropout(sublayer(self.norm(x)))
```

Figure 16 Figure 16 Κώδικας για SublayerConnection με Python https://github.com/ikoctac/annotated-transformer[22].

```python
# %% id="3jKa_prZTsqE"
class LayerNorm(nn.Module):
    "Construct a layernorm module (See citation for details)."

    def __init__(self, features, eps=1e-6):
        super(LayerNorm, self).__init__()
        self.a_2 = nn.Parameter(torch.ones(features))
        self.b_2 = nn.Parameter(torch.zeros(features))
        self.eps = eps

    def forward(self, x):
        mean = x.mean(-1, keepdim=True)
        std = x.std(-1, keepdim=True)
        return self.a_2 * (x - mean) / (std + self.eps) + self.b_2
```

Figure 17 Κώδικας για LayerNorm σε Python https://github.com/ikoctac/annotated-transformer [22].

Positional Encoding

Για να ενσωματωθούν οι πληροφορίες για την τοποθεσία στο μοντέλο προστίθενται επιπλέον κωδικοποιήσεις (**Encodings)** τοποθεσίας για την ενσωμάτωση του **embedding-input,** αυτό βοηθάει το μοντέλο να καταλαβαίνει την σειρά τον **tokens in sequences.**

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

Figure 18 Κώδικας για τοποθεσία των tokens για υπολογισμό στο paper https://nlp.seas.harvard.edu/annotated-transformer/ [22].

Encoder Stacking

Τέλος έχουμε το **Encoder stacking** το οποίο επαναλαμβάνει τα βήματα για το **Attention, FFN (Feed-Forward-Network)** και **Normalization** μέσα από πολλαπλά στρώματα (στο μοντέλο μας **BART-base** είναι 6 ενώ στο **BART-Large** είναι 12). Το **Stacking** επιτυγχάνεται με την class **"EncoderLayer"** η οποία βρίσκεται στην class **"Encoder".**

```
# %% id="qYkUFr6GTsqE"
class EncoderLayer(nn.Module):
    "Encoder is made up of self-attn and feed forward (defined below)"

    def __init__(self, size, self_attn, feed_forward, dropout):
        super(EncoderLayer, self).__init__()
        self.self_attn = self_attn
        self.feed_forward = feed_forward
        self.sublayer = clones(SublayerConnection(size, dropout), 2)
        self.size = size

    def forward(self, x, mask):
        "Follow Figure 1 (left) for connections."
        x = self.sublayer[0](x, lambda x: self.self_attn(x, x, x, mask))
        return self.sublayer[1](x, self.feed_forward)
```

Figure 19 Κώδικας για το EncoderLayer στην python https://github.com/ikoctac/annotated-transformer [22].

```python
# %% id="xqVTz9MkTsqD"
class Encoder(nn.Module):
    "Core encoder is a stack of N layers"

    def __init__(self, layer, N):
        super(Encoder, self).__init__()
        self.layers = clones(layer, N)
        self.norm = LayerNorm(layer.size)

    def forward(self, x, mask):
        "Pass the input (and mask) through each layer in turn."
        for layer in self.layers:
            x = layer(x, mask)
        return self.norm(x)
```

Figure 20 Τελικός κώδικας του κωδικοποιητή στην python https://github.com/ikoctac/annotated-transformer **[22].**

Αυτά όσον αφορά το αριστερό κομμάτι του **Transformer,** στη συνέχεια έχουμε στο δεξιό κομμάτι τον αποκωδικοποιητή (**Decoder**) ο οποίος δημιουργεί μια ακολουθία εξόδου βασισμένη στην κωδικοποιημένη **(Encoded)** είσοδο και τα **tokens** που δημιουργήθηκαν προηγούμενος και αποτελείται από τα παρακάτω κομμάτια.

Masked Multi-Head Self-Attention

Όπως και στον κωδικοποιητή (**Encoder**) έτσι και εδώ έχουμε παρόμοιο πρώτο στρώμα είναι ίδιο με το **"**MultiHeadSelfAttention**"** μόνο που χρησιμοποιεί **masking** για να αποτραπεί η εστίαση σε μελλοντικά **tokens.** Έτσι στο **"DecodeLayer"** το πρώτα στρώμα που χρησιμοποιεί την **"**MultiHeadSelfAttention**"** μαζί με την **"subsequent_mask".**

```python
# %% id="QN98O2l3TsqF"
def subsequent_mask(size):
    "Mask out subsequent positions."
    attn_shape = (1, size, size)
    subsequent_mask = torch.triu(torch.ones(attn_shape), diagonal=1).type(
        torch.uint8
    )
    return subsequent_mask == 0
```

Figure 21 Κώδικας για το Masked MHA (MutliHeadAttention) στην Python https://github.com/ikoctac/annotated-transformer **[22].**

Encoder-Decoder Attention

Ο αποκωδικοποιητής **(Decoder)** ελέγχει και τα αποτελέσματα του κωδικοποιητή **(Encoder)** και ελέγχει και μαθαίνει από τις σχέσεις της εισόδου/εξόδου, και αυτό είναι το δεύτερο υπόστρωμα στον **"DecoderLayer"** και χρησιμοποιεί και αυτό την "**MultiHeadSelfAttention**".

```python
# %% id="M2hA1xFQTsqF"
class DecoderLayer(nn.Module):
    "Decoder is made of self-attn, src-attn, and feed forward (defined below)"

    def __init__(self, size, self_attn, src_attn, feed_forward, dropout):
        super(DecoderLayer, self).__init__()
        self.size = size
        self.self_attn = self_attn
        self.src_attn = src_attn
        self.feed_forward = feed_forward
        self.sublayer = clones(SublayerConnection(size, dropout), 3)

    def forward(self, x, memory, src_mask, tgt_mask):
        "Follow Figure 1 (right) for connections."
        m = memory
        x = self.sublayer[0](x, lambda x: self.self_attn(x, x, x, tgt_mask))
        x = self.sublayer[1](x, lambda x: self.src_attn(x, m, m, src_mask))
        return self.sublayer[2](x, self.feed_forward)
```

Figure 22 Κώδικας για το DecoderLayer σε Python
https://github.com/ikoctac/annotated-transformer [22].

Feed-Forward Network (FFN)

Ακολουθεί την ίδια λογική με τον κωδικοποιητή **(Encoder)** και χρησιμοποιεί την class **PositionwiseFeedForward (FeedForwardNetwork).**

Add & Norm

Παρόμοια με τον κωδικοποιητή **(Encoder)** έτσι και ο αποκωδικοποιητής **(Decoder)** **χρησιμοποιεί το ίδιο "AddNorm".**

Decoder Stacking

Πάλι όπως και στον κωδικοποιητή **(Encoder)** παρόμοια και στον αποκωδικοποιητή **(Decoder)** αποτελείται από 6 στρώματα επαναλαμβάνοντας τα παραπάνω βήματα για την εξαγωγή αποτελεσμάτων μόνο που αυτή την φορά χρησιμοποιούμε το **"DecoderLayer"** το **"Decoder"** class.

```python
# %% id="M2hA1xFQTsqF"
class DecoderLayer(nn.Module):
    "Decoder is made of self-attn, src-attn, and feed forward (defined below)"

    def __init__(self, size, self_attn, src_attn, feed_forward, dropout):
        super(DecoderLayer, self).__init__()
        self.size = size
        self.self_attn = self_attn
        self.src_attn = src_attn
        self.feed_forward = feed_forward
        self.sublayer = clones(SublayerConnection(size, dropout), 3)

    def forward(self, x, memory, src_mask, tgt_mask):
        "Follow Figure 1 (right) for connections."
        m = memory
        x = self.sublayer[0](x, lambda x: self.self_attn(x, x, x, tgt_mask))
        x = self.sublayer[1](x, lambda x: self.src_attn(x, m, m, src_mask))
        return self.sublayer[2](x, self.feed_forward)
```

Figure 23 Κώδικας για το DecoderLayer σε Python
https://github.com/ikoctac/annotated-transformer [22].

```python
# %%
class Decoder(nn.Module):
    "Generic N layer decoder with masking."

    def __init__(self, layer, N):
        super(Decoder, self).__init__()
        self.layers = clones(layer, N)
        self.norm = LayerNorm(layer.size)

    def forward(self, x, memory, src_mask, tgt_mask):
        for layer in self.layers:
            x = layer(x, memory, src_mask, tgt_mask)
        return self.norm(x)
```

Figure 24 Κώδικας για τον Decoder στην Python
https://github.com/ikoctac/annotated-transformer [22].

Final Linear & SoftMax

Τέλος η τελική έξοδος από τον αποκωδικοποιητή μετατρέπεται σε μια πιθανότητα πάνω στα δεδομένα που έχει προπονηθεί το μοντέλο και εξάγει ένα αποτέλεσμα με την class **"Generator".**

```python
# %% id="NKGoH2RsTsqC"
class Generator(nn.Module):
    "Define standard linear + softmax generation step."

    def __init__(self, d_model, vocab):
        super(Generator, self).__init__()
        self.proj = nn.Linear(d_model, vocab)

    def forward(self, x):
        return log_softmax(self.proj(x), dim=-1)
```

Figure 25 Κώδικας για Generator στην Python
https://github.com/ikoctac/annotated-transformer [22].

Και τέλος η συνάρτηση που δημιουργεί το μοντέλο μας από μια συνάρτηση με όλες της Hyperparameters.

```python
# %% id="mPe1ES0UTsqI"
def make_model(
    src_vocab, tgt_vocab, N=6, d_model=512, d_ff=2048, h=8, dropout=0.1
):
    "Helper: Construct a model from hyperparameters."
    c = copy.deepcopy
    attn = MultiHeadedAttention(h, d_model)
    ff = PositionwiseFeedForward(d_model, d_ff, dropout)
    position = PositionalEncoding(d_model, dropout)
    model = EncoderDecoder(
        Encoder(EncoderLayer(d_model, c(attn), c(ff), dropout), N),
        Decoder(DecoderLayer(d_model, c(attn), c(attn), c(ff), dropout), N),
        nn.Sequential(Embeddings(d_model, src_vocab), c(position)),
        nn.Sequential(Embeddings(d_model, tgt_vocab), c(position)),
        Generator(d_model, tgt_vocab),
    )

    # This was important from their code.
    # Initialize parameters with Glorot / fan_avg.
    for p in model.parameters():
        if p.dim() > 1:
            nn.init.xavier_uniform_(p)
    return model
```

Figure 26 Κώδικας συνάρτησης για την δημιουργία μοντέλου στην Python
https://github.com/ikoctac/annotated-transformer [22].

Pre-training BART

Εφόσον έχουμε τον κώδικα για το πρόγραμμα μας έχει έρθει η ώρα για το **pre-training** του **BART.** Η εκπαίδευση του μοντέλου γίνεται μέσο του **Document corrupting** και της ανακατασκευής αυτών προσπαθώντας να βελτιστοποιήσει την ανακατασκευή αυτών. Το μοντέλο σε αντίθεση με άλλες **de-noising** προσεγγίσεις που λειτουργούν σε συγκεκριμένα **noising schemes** μας επιτρέπει να χρησιμοποιήσουμε οτιδήποτε τύπου δεδομένων χωρίς περιορισμούς.
Οι παρακάτω μετασχηματισμοί που υφίστανται στο **BART** ευθύνονται για την δημιουργία θορύβου **(Noising)** στο αρχείο μας [20].

1. Token Masking
2. Token Deletion
3. Text Infilling
4. Sentence permutation
5. Document Rotation

Στο παρακάτω σχήμα μπορούμε να δούμε και τι κάνουν στην εισαγωγή κειμένου.

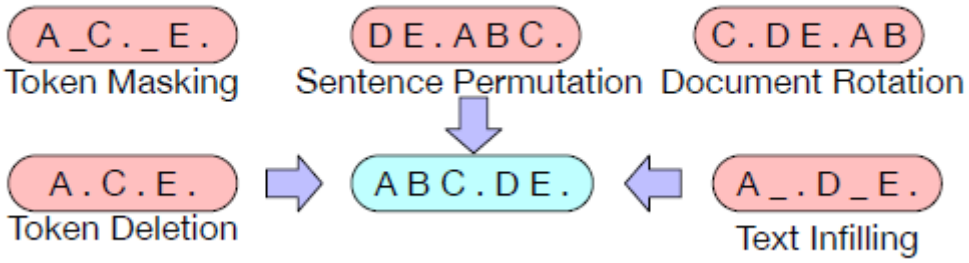

Figure 27 Εισαγωγή κειμένου στο μοντέλο Transformations for noising [20].

Token Masking

Όπως και στο **BERT** γίνεται τυχαίο sampling στα δεδομένα και γίνεται αντικατάσταση με στοιχεία [**MASK] [20].**

Token Deletion

Τυχαία **tokens** διαγράφονται από την εισαγωγή των δεδομένων, σε αντίθεση με το **token masking** σε αυτή την περίπτωση πρέπει να ορίσει το μοντέλο ποιες θα είναι αυτές οι θέσεις που θα διαγραφούν στην εισαγωγή δεδομένων [20].

Text Infilling

Η διαδικασία **Text Infilling (**Συμπλήρωση κειμένου) περιλαμβάνει την επιλογή τμημάτων κειμένου με μήκη που προέρχονται από την κατανομή **Poisson,** όπου κάθε τμήμα αντικαθίσταται από ένα **Token[MASK].** Αυτή η μέθοδος είναι εμπνευσμένη από το **span-BERT.** Εν κατακλείδι ο σκοπός της συμπλήρωσης κενού είναι να εκπαιδευτή το μοντέλο και να μπορεί να βρίσκει πόσα **tokens** λείπουν από κάθε τμήμα των δεδομένων [20].

Sentence Permutation

Η διαδικασία περιλαμβάνει τη διάσπαση ενός εγγράφου σε προτάσεις, που προσδιορίζονται από τις τελείες. Στη συνέχεια ,γίνεται τυχαίο ανακάτεμα στις προτάσεις δημιουργώντας μια νέα σειρά που θα χρησιμοποιηθεί για τον σκοπό της εκπαίδευσης ανάλυσης κειμένου στην περίπτωσή μας [20].

Document Rotation

Η διαδικασία περιλαμβάνει την τυχαία επιλογή ενός **token** από το έγγραφο και γίνεται το **document rotation** ώστε να ξεκινάει με αυτό το **token.** Με αυτή την μέθοδο έχει σαν στόχο να εκπαιδεύσει το μοντέλο να αναγνωρίζει που βρίσκεται η αρχή και με αυτόν τον τρόπο να το κάνει καλύτερο στο να κατανοεί την δομή αλλά και την οργάνωση των δεδομένων [20].

Performance Metrics

Εφόσον ολοκληρωθεί το pre-training το μοντέλο **BART** βελτιώνεται πάνω σε συγκεκριμένες εργασίες που δίνονται από τον χρήστη και χρησιμοποιεί δεδομένα από datasets που περιγράφουν τι εμπεριέχουν έτσι ώστε να γνωρίζει το μοντέλο ποια εργασία εκτελεί. Αυτό συμπεριλαμβάνει ότι ο χρήστης θα παραμετροποίηση της παραμέτρους κατάλληλα για εργασίες όπως **summarization,translation** και **question answering** [26].
Όπως θα δούμε και παρακάτω το **BART** τα πηγαίνει εξαιρετικά καλά σε τεστ δοκιμών για την απόδοσή του για συγκεκριμένες εργασίες. Πιο συγκεκριμένα έχουμε τα: **ROUGE-1, ROUGE-2, ROUGE-L** και **BLEU.**

1. Το **ROUGE-1** μετράει κατά πόσο είναι μοναδικές η λέξεις που παράγει το μοντέλο σε σχέση με το αρχικό κείμενο [32]. Το score μετριέται από 0-1 και οτιδήποτε υψηλότερο από 0.5 θεωρείται αποδεκτό [33].
2. Το **ROUGE-2** μετράει κατά πόσο υπάρχει συνέχεια στον λόγο από το κείμενο που δημιούργησε το μοντέλο μας. Κατά πόσο είναι ίδια η σειρά με της λέξεις από το αρχικό κείμενο [32]. Το score και εδώ μετριέται από 0-1 και οτιδήποτε υψηλότερο από 0.4 θεωρείται αποδεκτό [33].
3. Το **ROUGE-L** ψάχνει κατά πόσο υπάρχουν κοινές ακολουθίες από λέξεις στο παραγόμενο κείμενο και το αρχικό [32].Το score και εδώ μετριέται από 0-1 και οτιδήποτε υψηλότερο από 0.5 θεωρείται αποδεκτό [33].
4. Τέλος το **BLEU** το οποίο μετράει κατά πόσο είναι αποδεκτό το κείμενο που έχει μεταφραστεί από το πρωτότυπο.

   Και εδώ το score είναι σε παρόμοιούς δείκτες που κυμαίνονται από το 0-1. Παρακάτω είναι ο πίνακας με τα score.

a. **<0.1:** Δεν μπορεί να χρησιμοποιηθεί.
b. **0.1 – 0.19:** Δύσκολα μπορεί να κατανοηθεί.
c. **0.2 – 0.29:** Υπάρχει νόημα αλλά με σημαντικές αποκλίσεις.
d. **0.3 – 0.39:** Καλή, απλή κατανοητή μετάφραση.
e. **0.4 – 0.49:** Αρκετά καλή μετάφραση.
f. **0.5 – 0.59:** Πολύ καλής ποιότητας μετάφραση.
g. **>0.6:** Η μετάφραση συνήθως ξεπερνάει και την ανθρώπινη.

Παρακάτω είναι ο κώδικας στην python για την υλοποίηση των μετρήσεων του μοντέλου **BART** πάνω στα δεδομένα που έχει εκπαιδευθεί [34].

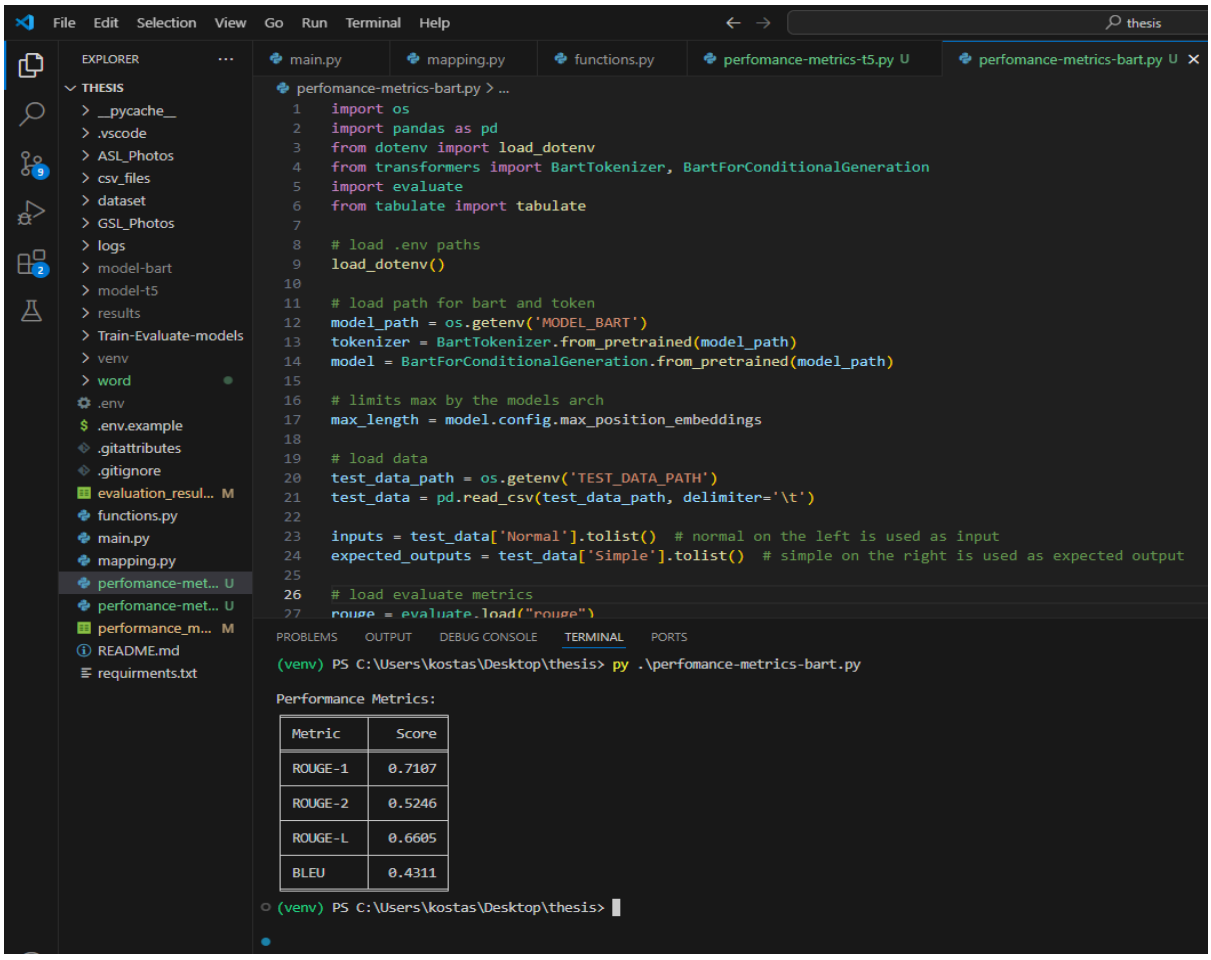

Figure 28 Κώδικας για αξιολόγηση μοντέλου perfomance-metrics-bart.py.

Metric	Score
ROUGE-1	0.7107
ROUGE-2	0.5246
ROUGE-L	0.6605
BLEU	0.4311

Table 1 Δεδομένα που έχουμε συλλέξει από τον κώδικα performance-metrics-bart.py.

2.3 T5 (Text-To-Text Transfer Transformer)

Το μοντέλο T5 (ονομάστηκε έτσι γιατί τα 5 αρχικά του ονόματος του ξεκινάνε από T) της google [25]. είναι ουσιαστικά ένας **text-to-text** transformer που αναπτύχθηκε για να ανταπεξέλθει σε διάφορες **NLP (Natural language processing)** εργασίες όπως η **μετάφραση γλώσσας, απάντηση σε ερωτήσεις ,περίληψη εγγράφων, κατηγοριοποίηση κειμένου κ.α.**
Το μοντέλο ουσιαστικά χειρίζεται όλα τα προβλήματα σαν **Text-to-Text** που σημαίνει ότι οποιαδήποτε και αν είναι η εργασία που θέλουμε να εκτελέσει το πρόγραμμα σαν είσοδο θα έχει **text** και σαν έξοδο θα έχει πάλι **text.** Σε αυτό που καινοτομεί το **T5** μοντέλο είναι η αρχιτεκτονική του που του επιτρέπει να ενσωματώνει πολλές λειτουργίες χωρίς να απαιτεί διαφορετικές ρυθμίσεις στο μοντέλο ή διαφορετικές μονάδες επεξεργασίας στο μοντέλο μας για διαφορετικές λειτουργίες.

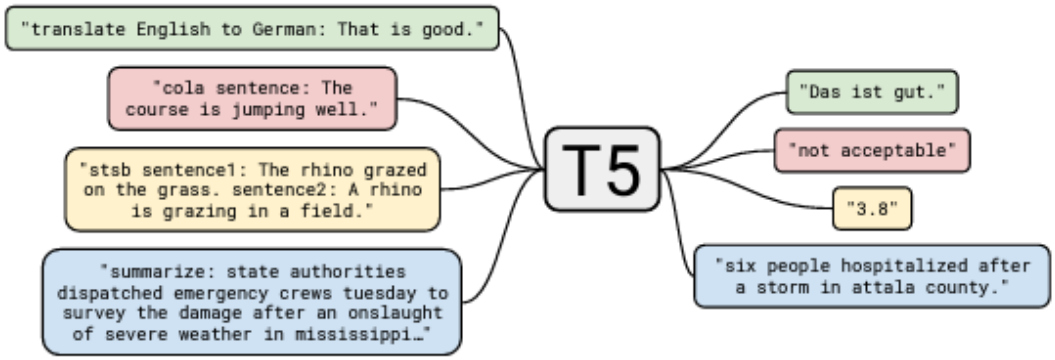

Figure 29 Οπτικοποίηση των λειτουργιών του T5 [25].

Το **T5** έρχεται σε διαφορετικά μεγέθη ανάλογα με τον όγκο δεδομένων αλλά και τον τύπο εργασίας που θες να υλοποιήσεις.

- **T5 SMALL**
- **T5 BASE**
- **T5 LARGE**
- **T5 3B**
- **T5 11B**

Το **T5 Small** χρησιμοποιεί 60 εκατομμύρια παραμέτρους και είναι αρκετά χρήσιμο για την ανάπτυξης ενός πρωτοτύπου μιας ιδέας αλλά και «φθηνό» καθώς δεν χρειάζεται αρκετά μεγάλη υπολογιστική δύναμη [26].
Το **T5 Base** χρησιμοποιεί 220 εκατομμύρια παραμέτρους και χρησιμοποιείται σε μια πληθώρα εργασιών όπως περίληψη κειμένου (**text-summarization),** μετάφραση **(Translation)**, αλλά και περίπλοκές ερωτήσεις του χρήστη [26].
Το **T5 Large** χρησιμοποιεί 770 παραμέτρους και είναι αρκετά αποδοτικό σε εργασίες όπως κατανόηση κειμένου **(Advanced understanding of context)**, ακριβής περίληψη κειμένου **(Detailed text-summarization)** αλλά και περιεκτικότερες απαντήσεις σε ερωτήσεις του

χρήστη **(Comprehensive question-answering)**. Η συγκεκριμένη έκδοση του μοντέλου είναι αρκετά χρήσιμη για έρευνα αλλά και σε εμπορικές χρήσεις οπού αναμένεται μεγάλη ακρίβεια.
Το **T5 3B** χρησιμοποιεί 3 Δισεκατομμύρια παραμέτρους και σε αυτήν την περίπτωση έχουμε να κάνουμε με δημιουργική γραφή από το μοντέλο μας **(Creative writing),** σε περιεκτική περίληψη νομικών εγγράφων συνήθως για περισσότερη ακρίβεια **(Text-summarization),** αλλά και συζητήσεις χρήστη-υπολογιστή που μέσα από πλήθος ερωτήσεων-απαντήσεων το μοντέλο που να δώσεις χρήσιμες πληροφορίες **(Multi-turn conversational agents).**
Και τέλος το **T5 11B** το οποίο χρησιμοποιεί το αστρονομικό ποσό τον 11 Δισεκατομμυρίων παραμέτρων και είναι το μεγαλύτερο σε κυκλοφορία των T5 [26]. Οι χρήσεις του είναι πολλές και συνήθως όχι από απλούς καταναλωτές αλλά μεγάλους κολοσσούς οι οποίοι διαθέτουν χιλιάδες ευρώ/δολάρια για την λειτουργία τους [27]. Μερικές από αυτές της εργασίες είναι πρωτοποριακές εφαρμογές στη φυσική γλώσσα **(cutting-edge NLP applications)**, συμπεριλαμβανομένων πολυδιάστατων εργασιών **(multi-modal tasks),** αρκετά προχωρημένη παραγωγή κειμένου **(context-rich text generation)** και κατανόησης γλώσσας σε μεγάλη κλίμακα **(large-scale language understanding**) [26].

Architecture

Η αρχιτεκτονική του **T5** όπως και του **BART** βασίζονται στον Transformer [23] οπότε έχουν την ίδια αρχιτεκτονική μόνο που στο **T5** υπάρχει η διαφορά ότι οι παράμετροι που χρησιμοποιούνται από τον κωδικοποιητή **(Encoder)** και τον αποκωδικοποιητή **(Decoder)** είναι κοινές μειώνοντας τον αριθμό των παραμέτρων που μπορούν να εκπαιδευτούν κάνοντας το μοντέλο κατάλληλο για μελλοντικές επεκτάσεις χωρίς μεγάλο κόστος [25].
Όπως προανέφερα και πριν το **T5** χρησιμοποιεί έναν **text-to-text framework,** που σημαίνει ότι η είσοδος και έξοδος του προγράμματος θα είναι **strings of text**. Ένα απλό παράδειγμα για να διαφοροποιήσουμε το **BART** από το **T5** είναι να δούμε τα δύο μοντέλα σαν δύο έμπειρους σεφ στην κουζίνα, δηλαδή.

CHEF T5: Πάντα ακολουθάει την συνταγή

Πολύ απλά μπορούμε να φανταστούμε το **T5** σαν έναν σεφ με ένα βιβλίο γεμάτο από συνταγές και οδηγίες στο πώς να παρασκευαστούν. Η κάθε συνταγή απαιτεί συγκεκριμένα υλικά σε συγκεκριμένη σειρά εκτέλεσης.
Σε αυτή την περίπτωση ο πελάτης (Χρήστης) μπορεί να θελήσει να φάει ένα σάντουιτς με **μπαγκέτα/γαλοπούλα/μαγιονέζα/ντομάτα/μαρούλι** και να δώσει τα υλικά με αυτή την σειρά στον σεφ. Αν στο βιβλίο του σεφ **T5** λέει πως η σειρά τον υλικών είναι έτσι καλός, και θα βγάλει σαν αποτέλεσμα **“ Ορίστε η μπαγκέτα με γαλοπούλα μαγιονέζα ντομάτα και μαρούλι”** σε περίπτωση που ο χρήστης δώσει μια συνταγή με ανακατεμένα υλικά ο σεφ **T5** δεν θα κάνει του κεφαλιού του και θα παρακούσει την συνταγή αλλά θα φτιάξει την μπαγκέτα του όπως του λέει το βιβλίο με τις συνταγές.
Δηλαδή έχουμε **μαγιονέζα/μπαγκέτα/ντομάτα/μαρούλι/γαλοπούλα** η απάντηση θα είναι πάλι **“ Ορίστε η μπαγκέτα με γαλοπούλα μαγιονέζα ντομάτα και μαρούλι”.** Αυτό μάς δείχνει πως ο σεφ **T5** χρησιμοποιεί την **text-to-text** τεχνική και δεν είναι τόσο δημιουργικός, παρόλα αυτά δεν καθιστάτε και άχρηστος καθώς ξέρει και εκτελεί πολύ καλά ότι λέει το βιβλίο με τις συνταγές του, στην περίπτωση αυτή οι συνταγές είναι τα δεδομένα που εκπαιδεύεται το μοντέλο μας.

Pretraining Data

Το **T5** κάνει **pretraining** στο C4 Dataset το οποίο είναι ένα μεγάλου μεγέθους web scraping project , το common crawl [28], το οποίο στην συνέχεια φιλτράρετε και γίνεται καθαρισμός δεδομένων για την βέλτιστη απόδοση του μοντέλου.

- Διατηρούνται μόνο γραμμές κειμένου στις οποίες **υπάρχει ένα τέλος** (Άνω τελεία, τελεία κ.α.).
- Σελίδες οι οποίες εμπεριείχαν λιγότερες από **3 προτάσεις** κείμενο δεν συμπεριλήφθηκαν ούτε και προτάσεις με **λιγότερο από 5 λέξεις η καθεμία**.
- Έγινε καθαρισμός **«κακών λέξεων»** [29].
- Οτιδήποτε εμπεριείχε **"lorem ipsum"** απορρίφθηκε.
- Ότι εμπεριείχε **"{"** απορρίφθηκε.
- Σε σελίδες όπως η Wikipedia αφαιρέθηκαν τα σημάδια που έδειχναν σε **citations**.

Επιπλέον χρησιμοποιήθηκε και η **langdetect [30]** για να βεβαιωθούν ότι τα δεδομένα χρησιμοποιούν μόνο την αγγλική γλώσσα σε μεγάλο ποσοστό. Σε αντίθεση με άλλα μοντέλα το **T5** έχει μια πολύ καλή βάση δεδομένων (χρησιμοποιήθηκαν περίπου 750GB) [25] και καθαρή που βοηθάει στην ομαλή λειτουργία του μοντέλου και στον καλών αποτελεσμάτων.

Tokenization

Το **T5** μοντέλο όπως είπαμε και πριν κάνει **pretraining** χρησιμοποιώντας τα δεδομένα από το **C4(Common crawl) [28]** για 524.288 βήματα. Αυτά τα βήματα ουσιαστικά είναι σαν το μοντέλο μας να διαβάζει για κάθε βήμα μια παρτίδα από λέξεις και φράσεις προσπαθώντας να μάθει πως διασυνδέονται μεταξύ τους. Εφόσον έχει τελειώσει με αυτά τα βήματα θα μπορούμε να πούμε ότι έχει καταλάβει αρκετά ώστε να καταλαβαίνει την γενική γλώσσα. Έτσι το μοντέλο μας με μέγιστο μήκος για μια ακολουθία τα 512 token και κάθε μια με μέγιστη 128, έχουμε 65.536 **tokens** για κάθε βήμα , που συνολικά το μοντέλο μας θα έχει κάτι περισσότερο από 34 δισεκατομμύρια **tokens.** Το μοντέλο χρησιμοποιεί **SentencePiece tokenization.** Τα **tokens** ουσιαστικά είναι ένα μικρό unit of text που γλωσσικά μοντέλα σαν το **T5** μπορούν να διαχειριστούν και να επεξεργαστούν κατάλληλα, για παράδειγμα:

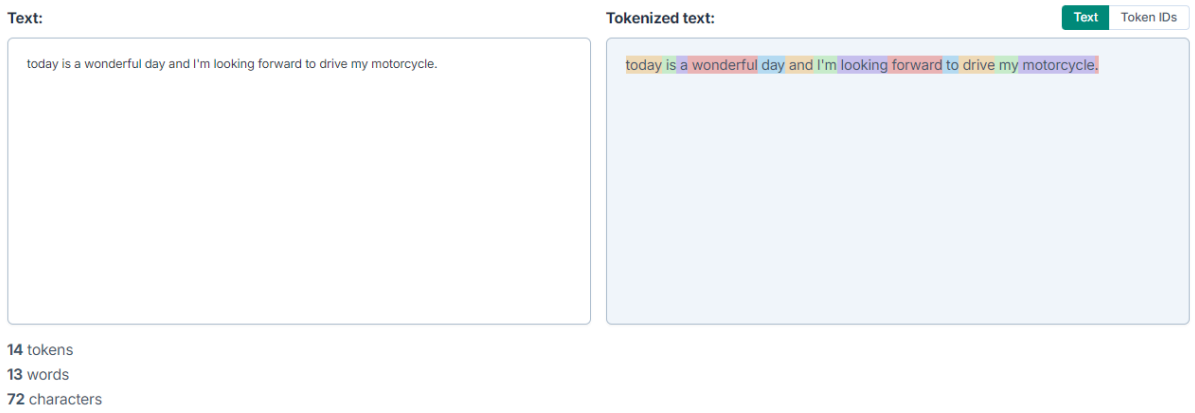

Figure 30 Οπτικοποίηση για το πώς δουλεύουν τα tokens https://gptforwork.com/tools/tokenizer [31].

Όπως βλέπουμε το μοντέλο μας σπάει σε της λέξεις σε **tokens** ώστε να μπορεί να αναλύει καλύτερα, να μετατρέπει κείμενα, να μεταφράζει ανεξάρτητα από την δυσκολία της γλώσσας [26].

Pretraining Objective

Το **T5** χρησιμοποιεί **span-corruption-objective.** Ουσιαστικά γίνονται τυχαίες επιλογές του κειμένου και αντικαθιστάτε από **token** «φρουρούς» που τα συγκεκριμένα **token** βρίσκονται εκεί και περιμένουν να αντικατασταθούν από τις προβλέψεις του μοντέλου κατά την εκπαίδευση του μοντέλου.

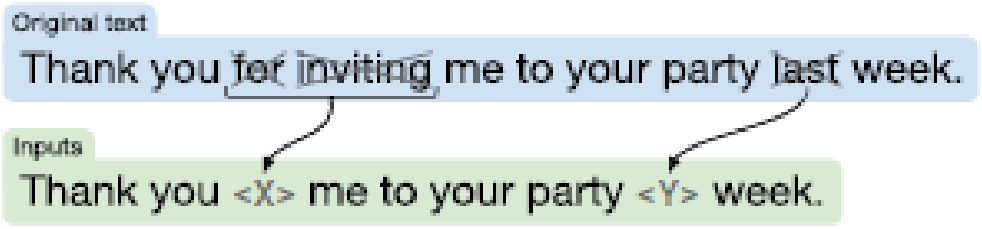

Figure 31 Τυχαίες επιλογές κειμένου για να γίνει το token-masking [26].

Αυτή η τεχνική δίνει την δυνατότητα στο **T5** μοντέλο να ανακατασκευάζει αποδοτικά το κείμενο μας και να καταλαβαίνει τις σχέσεις των λέξεων και προτάσεων μεταξύ τους.

Fine-Tuning and Task Performance

Εφόσον ολοκληρωθεί το pre-training το μοντέλο **T5** βελτιώνεται πάνω σε συγκεκριμένες εργασίες που δίνονται από τον χρήστη και χρησιμοποιεί δεδομένα από datasets που περιγράφουν τι εμπεριέχουν έτσι ώστε να γνωρίζει το μοντέλο ποια εργασία εκτελεί. Αυτό συμπεριλαμβάνει ότι ο χρήστης θα παραμετροποίηση της παραμέτρους κατάλληλα για εργασίες όπως **summarization,translation** και **question answering** [26].
Όπως θα δούμε και παρακάτω το **T5** τα πηγαίνει εξαιρετικά καλά σε τεστ δοκιμών για την απόδοσή του για συγκεκριμένες εργασίες. Πιο συγκεκριμένα έχουμε τα: **ROUGE-1, ROUGE-2, ROUGE-L** και **BLEU.**

1. Το **ROUGE-1** μετράει κατά πόσο είναι μοναδικές η λέξεις που παράγει το μοντέλο σε σχέση με το αρχικό κείμενο [32]. Το score μετριέται από 0-1 και οτιδήποτε υψηλότερο από 0.5 θεωρείται αποδεκτό [33].
2. Το **ROUGE-2** μετράει κατά πόσο υπάρχει συνέχεια στον λόγο από το κείμενο που δημιούργησε το μοντέλο μας. Κατά πόσο είναι ίδια η σειρά με της λέξεις από το αρχικό κείμενο [32]. Το score και εδώ μετριέται από 0-1 και οτιδήποτε υψηλότερο από 0.4 θεωρείται αποδεκτό [33].
3. Το **ROUGE-L** ψάχνει κατά πόσο υπάρχουν κοινές ακολουθίες από λέξεις στο παραγόμενο κείμενο και το αρχικό [32].Το score και εδώ μετριέται από 0-1 και οτιδήποτε υψηλότερο από 0.5 θεωρείται αποδεκτό [33].
4. Τέλος το **BLEU** το οποίο μετράει κατά πόσο είναι αποδεκτό το κείμενο που έχει μεταφραστεί από το πρωτότυπο.

Και εδώ το score είναι σε παρόμοιούς δείκτες που κυμαίνονται από το 0-1. Παρακάτω είναι ο πίνακας με τα score.

h. **<0.1:** Δεν μπορεί να χρησιμοποιηθεί.
i. **0.1 – 0.19:** Δύσκολα μπορεί να κατανοηθεί.
j. **0.2 – 0.29:** Υπάρχει νόημα αλλά με σημαντικές αποκλίσεις.
k. **0.3 – 0.39:** Καλή, απλή κατανοητή μετάφραση.
l. **0.4 – 0.49:** Αρκετά καλή μετάφραση.
m. **0.5 – 0.59:** Πολύ καλής ποιότητας μετάφραση.
n. **>0.6:** Η μετάφραση συνήθως ξεπερνάει και την ανθρώπινη.

Παρακάτω είναι ο κώδικας στην python για την υλοποίηση των μετρήσεων του μοντέλου **T5** πάνω στα δεδομένα που έχει εκπαιδευθεί [34].

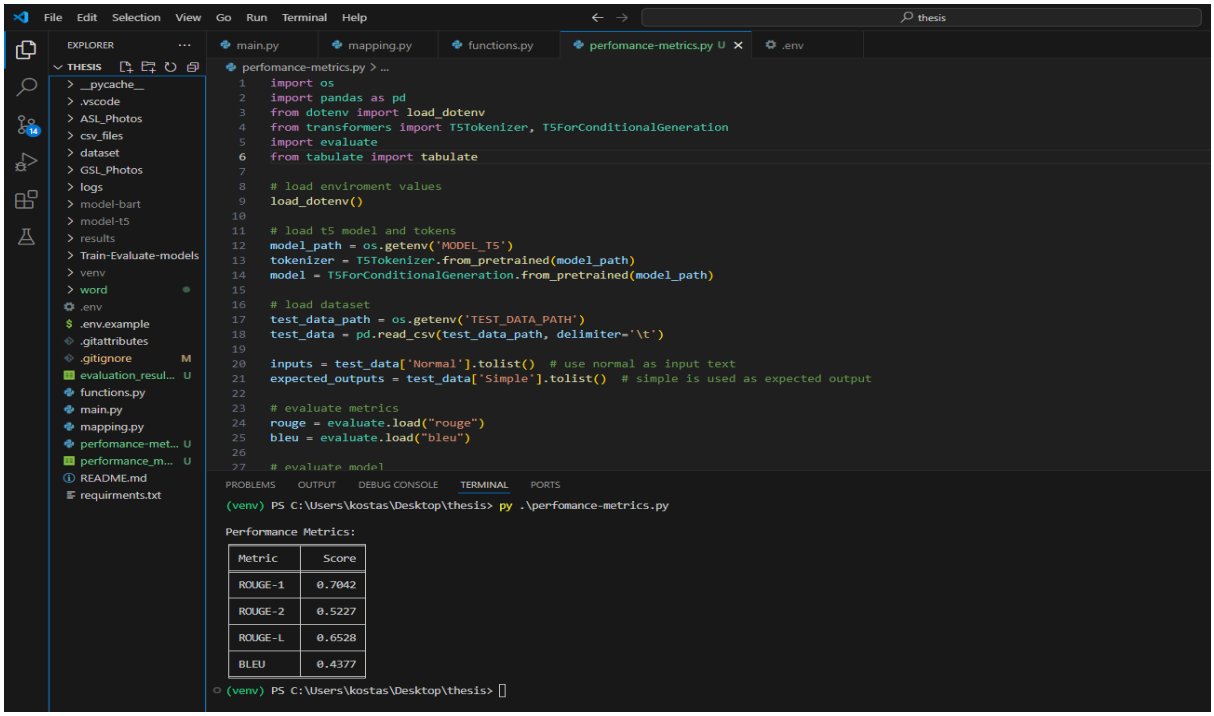

Figure 32 Κώδικας για την αξιολόγηση του T5 perfomance-metrics-t5.py.

Όπως διακρίνεται και στην εικόνα έχουμε τα δεδομένα για το πώς αποδίδει το μοντέλο πάνω στα δεδομένα που του έχουμε δώσει.

Metric	Score
ROUGE-1	0.7042
ROUGE-2	0.5227
ROUGE-L	0.6528
BLEU	0.4377

Table 2 Αποτελέσματα του performance-metrics-T5.py.

2.4 Συγκριτική αξιολόγηση μοντέλων στο εκπαιδευμένο dataset

Και τα δύο μοντέλα είναι εξίσου ικανά για την εργασία της περίληψης στην συγκεκριμένη περίπτωση. Εφόσον έχω τα δεδομένα μου και έχω εκπαιδεύσει και τα δύο μοντέλα πάνω στην εργασία της περίληψης θα δούμε πώς ανταπεξέρχονται.
Η όλη ιδέα με την αξιοποίηση των μοντέλων αυτόν ήταν τα κείμενα που δίνει ο χρήστης να γίνεται μικρότερα και συμπυκνωμένα χωρίς να χάνετε το αρχικό νόημα στο κείμενο αλλά να κάνει ευκολότερη την εκμάθηση με λιγότερο φόρτο στην αρχή. Έτσι εφόσον έγινε η εκπαίδευση των μοντέλων στο παρακάτω script έχουμε και τα δύο μοντέλα που δοκιμάζουμε το test.csv μας με τις απλοποιημένες προτάσεις για να βγάλουμε ένα συμπέρασμα για το ποιο πρέπει να χρησιμοποιήσουμε.

perfomance-final.py

Perfomance scripts > perfomance-final.py > ...

```
36
37  # Evaluate both models
38  for input_text in inputs:
39      # T5 Prediction
40      t5_input_ids = t5_tokenizer.encode(input_text, return_tensors='pt', padding=True, truncation=True)
41      t5_output_ids = t5_model.generate(t5_input_ids, max_length=150)
42      t5_output_text = t5_tokenizer.decode(t5_output_ids[0], skip_special_tokens=True)
43      t5_predictions.append(t5_output_text)
44
45      # BART Prediction
46      bart_input_ids = bart_tokenizer.encode(input_text, return_tensors='pt', padding=True, truncation=True)
47      bart_output_ids = bart_model.generate(bart_input_ids, max_length=150)
48      bart_output_text = bart_tokenizer.decode(bart_output_ids[0], skip_special_tokens=True)
49      bart_predictions.append(bart_output_text)
50
51  # Calculate metrics for each prediction and store them
52  for i in range(len(inputs)):
53      # Calculate ROUGE and BLEU for T5
54      t5_rouge_result = rouge.compute(predictions=[t5_predictions[i]], references=[expected_outputs[i]])
55      t5_bleu_result = bleu.compute(predictions=[t5_predictions[i]], references=[[expected_outputs[i]]])
56
57      # Calculate ROUGE and BLEU for BART
58      bart_rouge_result = rouge.compute(predictions=[bart_predictions[i]], references=[expected_outputs[i]])
59      bart_bleu_result = bleu.compute(predictions=[bart_predictions[i]], references=[[expected_outputs[i]]])
60
61      # Store metrics in a tuple (ROUGE-1, ROUGE-2, ROUGE-L, BLEU)
62      t5_metrics.append((t5_rouge_result['rouge1'], t5_rouge_result['rouge2'],
63                         t5_rouge_result['rougeL'], t5_bleu_result['bleu']))
64
65      bart_metrics.append((bart_rouge_result['rouge1'], bart_rouge_result['rouge2'],
66                           bart_rouge_result['rougeL'], bart_bleu_result['bleu']))
67
68  # Convert metrics to DataFrame for easier processing
69  metrics_df = pd.DataFrame({
70      'Model': ['T5'] * len(inputs) + ['BART'] * len(inputs),
71      'ROUGE-1': [m[0] for m in t5_metrics] + [m[0] for m in bart_metrics],
72      'ROUGE-2': [m[1] for m in t5_metrics] + [m[1] for m in bart_metrics],
73      'ROUGE-L': [m[2] for m in t5_metrics] + [m[2] for m in bart_metrics],
74      'BLEU': [m[3] for m in t5_metrics] + [m[3] for m in bart_metrics],
75  })
76
77  # Normalize metrics (simple min-max normalization)
78  for metric in ['ROUGE-1', 'ROUGE-2', 'ROUGE-L', 'BLEU']:
79      metrics_df[metric] = (metrics_df[metric] - metrics_df[metric].min()) / (metrics_df[metric].max() - metrics_df[metric].min())
80
```

Figure 33 Μέρος πρώτο του script για την αξιολόγηση των 2 μοντέλων performance-final.py.

```python
# Assign weights to each metric (customize as needed)
weights = {
    'ROUGE-1': 0.3,
    'ROUGE-2': 0.3,
    'ROUGE-L': 0.3,
    'BLEU': 0.1
}

# Calculate composite score for each model based on the average of the metrics
metrics_summary = metrics_df.groupby('Model').mean().reset_index()
metrics_summary['Composite Score'] = (
    metrics_summary['ROUGE-1'] * weights['ROUGE-1'] +
    metrics_summary['ROUGE-2'] * weights['ROUGE-2'] +
    metrics_summary['ROUGE-L'] * weights['ROUGE-L'] +
    metrics_summary['BLEU'] * weights['BLEU']
)

# Select the best model based on the composite score
best_model_row = metrics_summary.loc[metrics_summary['Composite Score'].idxmax()]

print("\nBest Model Selection:")
print(best_model_row)

# Optionally save predictions and metrics to a CSV file for further analysis
results_data = {
    'Input': inputs,
    'Expected': expected_outputs,
    'T5 Predicted': t5_predictions,
    'BART Predicted': bart_predictions,
}

results_df = pd.DataFrame(results_data)
results_df.to_csv('evaluation_results.csv', index=False)

print("\nEvaluation results saved to 'evaluation_results.csv'.")
```

Figure 34 Δεύτερο μέρος του script για την αξιολόγηση των 2 μοντέλων performance-final.py.

Τέλος έχουμε τον πίνακα με τα αποτελέσματα που θα συγκρίνει ουσιαστικά τα 2 μοντέλα μας και θα μας δίνει την καλύτερη δυνατή επιλογή με βάση τα δεδομένα που έχουμε εκπαιδεύσει το μοντέλο μας.

```
PS C:\Users\kostas\Desktop\thesis> .\venv\Scripts\activate
(venv) PS C:\Users\kostas\Desktop\thesis> cd '.\Perfomance scripts\'
(venv) PS C:\Users\kostas\Desktop\thesis\Perfomance scripts> py .\perfomance-final.py
Asking to truncate to max_length but no maximum length is provided and the model has no

Best Model Selection:
Model                 BART
ROUGE-1           0.608966
ROUGE-2            0.49317
ROUGE-L           0.585296
BLEU               0.39023
Composite Score   0.545253
Name: 0, dtype: object

Evaluation results saved to 'evaluation_results.csv'.
(venv) PS C:\Users\kostas\Desktop\thesis\Perfomance scripts>
```

Figure 35 Πίνακας αποτελεσμάτων από το performance-final.py για την επιλογή μοντέλου στο πρόγραμμα μας.

ΚΕΦΑΛΑΙΟ 3 Speech recognition

3.1 Speech recognition Βιβλιοθήκη

Η λειτουργία του προγράμματος μπορεί να γίνει με δύο τρόπους. Η πρώτη λύση και πιο εύκολη (χωρίς bugs) είναι η είσοδος του χρήστη από το πληκτρολόγιο για την μετάφραση σε νοηματικά σύμβολα. Η δεύτερη λύση και με μεγαλύτερη προσβασιμότητα είναι αυτή της αναγνώρισης φωνής (Speech Recognition). Ουσιαστικά είναι μια βιβλιοθήκη που προσφέρει η **python**. Γίνεται η καταγραφή του ήχου με το **PyAudio,** απαραίτητη προϋπόθεση ότι ο χρήστης διαθέτει μικρόφωνο, μετά με την βιβλιοθήκη **SpeechRecognition** γίνεται η επεξεργασία της εισόδου του χρήστη (ο ήχος) και μετατρέπεται σε δεδομένα που μπορεί να επεξεργαστεί η βιβλιοθήκη. Και τέλος είναι η αναγνώριση της φωνής γίνεται καλώντας την συνάρτηση της **recognize_google()** (στην περίπτωση που προγράμματός μου τουλάχιστον) για να μετατρέψει τα δεδομένα φωνής σε κείμενο [35]. Μέσο της **recognize_google()** έχω και την δυνατότητα να δίνω ποια γλώσσα θέλω το πρόγραμμα να αναγνωρίζει.
Αρχικά για να στήσουμε το **SpeechRecognition** χρειάζεται να εισάγουμε την βιβλιοθήκη και τα πακέτα που μας ενδιαφέρουν που είναι το **sr**. Στην παρακάτω φωτογραφία είναι υπογραμμισμένο το πακέτο.

```
main.py > ...
1    import speech_recognition as sr
2    import os
3    os.environ['TF_ENABLE_ONEDNN_OPTS'] = '0'
4    from datetime import datetime
5    from functions import *
6    from mapping import *  # importing necess
7
```

Figure 36 Βιβλιοθήκες για την SpeechRecognition main.py.

Στην συνέχεια ορίζουμε με μια μεταβλητή recognizer = sr.Recognizer() όπου η συγκεκριμένη κλάση είναι αρκετά βασική για να δημιουργηθεί το text από την φωνή του χρήστη. Με λίγα λόγια εφόσον η φωνή του χρήστη εγγραφεί ο Recognizer το μετατρέπει σε κείμενο με διάφορες μηχανές που κάνουν την μετατροπή όπως το Google_recognizer που χρησιμοποιώ και στο πρόγραμμα μου.

```
# initalize recognizer.
recognizer = sr.Recognizer()
```

Figure 37 Ορισμός μεταβλητής στον recognizer main.py.

Στην συνέχεια χρησιμοποιούμε σαν προκαθορισμένη είσοδο το μικρόφωνο του υπολογιστή με την παρακάτω γραμμή κώδικα.

```
# use the default microphone as source, bot asks what language will t
with sr.Microphone() as source:
```

Figure 38 Χρησιμοποιούμε το μικρόφωνο από τον υπολογιστή main.py.

Παρακάτω ρυθμίζουμε κιόλας για τον λευκό θόρυβο στο περιβάλλον του χρήστη έτσι ώστε να παίρνει καλύτερα την εισαγωγή και να μην υπάρχουν ανακρίβειες στο τελικό κείμενο.

```
# adjust for ambient noise if necessary.
recognizer.adjust_for_ambient_noise(source)
```

Figure 39 Ρύθμιση για τα παράσιτα main.py.

Και τέλος στο πρόγραμμα μου τρέχει σε μια επανάληψη μέχρι να δώσει τερματικό ο χρήστης να τελειώσει. Μέσα στην επανάληψη υπάρχει η μεταβλητή **audio** και **text** όπου το audio αποθηκεύει τον ήχο με **recognizer.listen()** και μετά στο text πάλι με το **recognizer.recognize_google(audio, language=Language).** Με λίγα λόγια αυτή είναι η όλη διαδικασία για την αναγνώριση φωνής.

```
# continuously listen for speech.
while True:
    try:
        # capture users audio
        audio = recognizer.listen(source)

        # converts speech to text using the selected language.
        text = recognizer.recognize_google(audio, language=language)
```

Figure 40 Loop για την εγγραφή από το μικρόφωνο main.py.

Βλέπουμε όπως τρέχει και το πρόγραμμα ότι χρησιμοποιεί το μικρόφωνο το πρόγραμμα μας και περιμένει για εισαγωγή ήχου από τον χρήστη (Αναφέρομαι στην εικόνα στην επόμενη σελίδα).

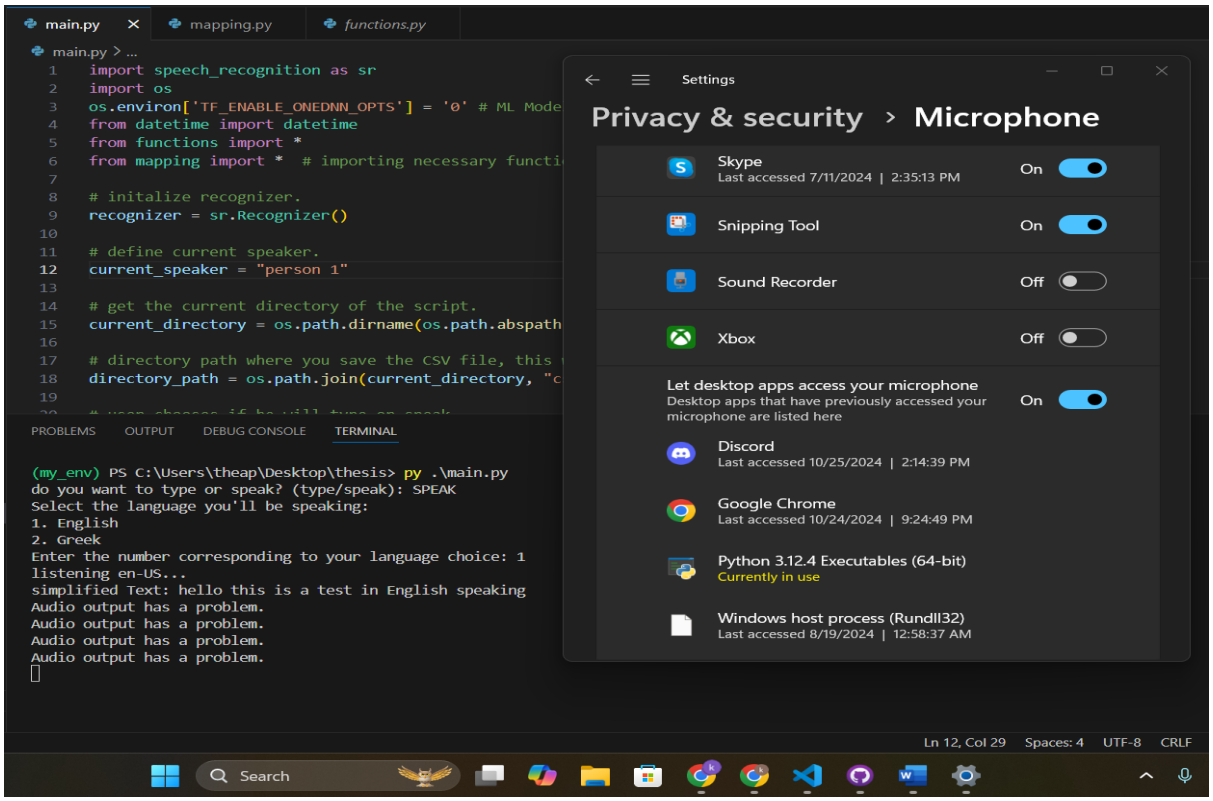

Figure 41 Εγγραφή ήχου από το μικρόφωνο main.py.

Και έτσι είναι το πρόγραμμα εφόσον γίνει εγγραφή του χρήστη και κάνει την μετάφραση. Παίρνει όπως φαίνεται και το κείμενο και το μετατρέπει σε φωτογραφίες της ASL,και βλέπουμε ότι δεν κάνει και η εγγραφή του μικροφώνου.

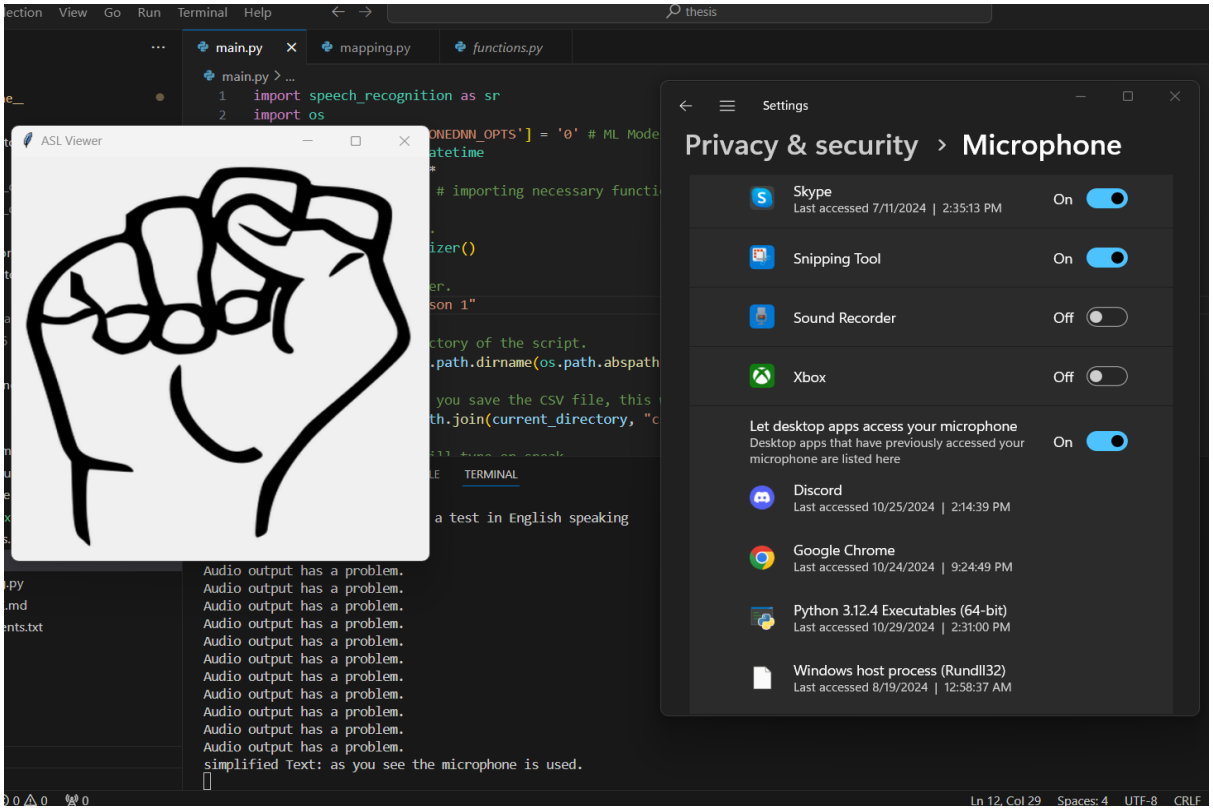

Figure 42 Εγγραφή ήχου από το μικρόφωνο και προβολή του κειμένου main.py.

3.2 Αναγνώριση ομιλίας με recognize_google

Στο πρόγραμμα μου είδαμε στο προηγούμενο υποκεφάλαιο ότι χρησιμοποιώ την speech recognition βιβλιοθήκη για την ενσωμάτωση της εισόδου κειμένου από το μικρόφωνο. Έτσι για μετατρέψουμε την φωνή σε κείμενο χρειαζόμαστε την **class sr.Recognizer()** που έχουμε θέση σε μια μεταβλητή **recognizer** η οποία μπορεί να χρησιμοποιηθεί με τις ακόλουθες μεθόδους.

- recognize_bing(): χρησιμοποιεί το Microsoft Bing Speech API.
- **recognize_google(): χρησιμοποιεί το Google Speech API.**
- recognize_google_cloud(): χρησιμοποίει το Google Cloud Speech API.
- recognize_houndify(): χρησιμοποιεί το Houndify API από την SoundHound.
- recognize_ibm(): χρησιμοποιεί το IBM Text to speech API.
- recognize_sphinx(): χρησιμοποιεί το PocketSphinx API, είναι το μόνο που μπορεί να γίνει και χωρίς σύνδεση στο ίντερνετ.

Στο πρόγραμμα μου επέλεξα να χρησιμοποιήσω την δεύτερη επιλογή το **recognize_google()** σαν μια αξιόπιστη και γρήγορη λύση για την υλοποίηση μου. Για να γίνει η αναγνώριση προφανώς χρειάζεται να υπάρχει και σύνδεση στο διαδίκτυο έτσι ώστε να μπορούμε να χρησιμοποιήσουμε το API. Σε περίπτωση που δεν υπάρχει αυτή η δυνατότητα θα μπορούσαμε να τροποποιήσουμε το πρόγραμμα μας έτσι ώστε να δουλεύει με το

recognize_sphinx() [37].Οπότε ξεκινώντας χρειάζεται να ορίσουμε σαν μικρόφωνο όπως έχουμε δείξει και στο υποκεφάλαιο 3.1 και να ξεκινήσει η "εγγραφή" ήχου με την μέθοδο **listen()** η οποία επιστρέφει την **speech_recognition.AudioData** που αποθηκεύεται στην μεταβλητή audio που έχω δημιουργήσει και στην συνέχεια και περνάει αυτό το **audio** μαζί με την **lang** στην μέθοδο **recognize_google(audio, language=lang)** για να ξέρει ποιόν ήχο θα μετατρέψει και σε τί γλώσσα είναι ο ήχος.

```
# capture user audio
audio = recognizer.listen(source)

# convert speech to text using the selected language
text = recognizer.recognize_google(audio, language=lang)
```

Figure 43 Μεταβλητή audio και μετατροπή audio σε text με την μέθοδο google_recognize().

Έτσι όταν γίνει η εκτέλεση του παραπάνω κώδικα στο πρόγραμμα μου οποιαδήποτε εισαγωγή στο μικρόφωνό θα καταγράφεται μέχρι να γίνει η επόμενη εγγραφή. Συνήθως επειδή τα μικρόφωνα δεν κάνουν και την καλύτερη δυνατή καταγραφή είναι συνετό να χρησιμοποιούμε μια class για την καλύτερη καταγραφή του ήχου χωρίς μεγάλες παρεμβολές από το περιβάλλον του χρήστη.

```
    # adjust for ambient noise if necessary
    recognizer.adjust_for_ambient_noise(source)
```

Figure 44 Class για την σταθεροποίηση λευκού θορύβου από το περιβάλλον του χρήστη.

Και τέλος η μέθοδος **recognize_google()** κάποιες φορές εάν ο χρήστης δεν αρθρώσει σωστά κάποιες λέξεις μπορεί να μην γίνει αντιστοιχία με την αποθήκη λέξεων της μεθόδου, οπότε όπως και στο πρόγραμμα μου χρησιμοποιείται ανάμεσα σε μια **try:** και στο τέλος να έχουμε κάποιες except για ενδεχόμενα errors που μπορεί να προκύψουν.

```
    # used to see if the microphone is still capturing audio
    except sr.UnknownValueError:
        print("Could not understand audio.")
    except sr.RequestError as e:
        print(f"Error: {e}")
```

Figure 45 Except για να βρούμε τυχών errors στην ανίχνευση φωνής.

ΚΕΦΑΛΑΙΟ 4 Προτεινόμενο σύστημα

4.1 Γλώσσα προγραμματισμού και πακέτα

Η υλοποίηση του προγράμματος έγινε με την γλώσσα προγραμματισμού **python.** Η **python** είναι μια γλώσσα προγραμματισμού γενικού σκοπού και αρκετά απλή στην χρήση της. Θεωρείται γλώσσα προγραμματισμού υψηλού επιπέδου και εστιάζει στο πόσο ευανάγνωστος είναι ο κώδικας έτσι ώστε να βοηθάει τον προγραμματιστή στην καλύτερη κατανόηση. Δημιουργήθηκε από τον **Guido van Rossum** και κυκλοφόρησε το 1991 ως έκδοση 0.9.0 και υποστηρίζει αρκετά προγραμματιστικά παραδείγματα όπως διαδικαστικός, αντικειμενοστραφής και λειτουργικός προγραμματισμός [38][39].
Από την αρχή ακόμα απέκτησε μεγάλη υποστήριξη από τους χρήστες και έχει καταφέρει να δημιουργήσει μια τεράστια κοινότητα και αρκετά μεγάλη υποστήριξη από την κοινότητα των χρηστών και έχει επεκταθεί σε τομείς όπως η ανάπτυξη ιστοσελίδων, στην επιστήμη δεδομένων αλλά και στην μηχανική μάθηση/τεχνητή νοημοσύνη [40].

Τα βασικά χαρακτηριστικά της python:

- **Simplicity and Readability:** Το συντακτικό της python είναι σχεδιασμένο έτσι ώστε να είναι αρκετά καθαρό και άμεσο στον προγραμματιστή, κάνοντας έτσι την ανάγνωση του κώδικα ευκολότερη αλλά και την συντήρηση ενός προγράμματος γραμμένο σε **python,**
- **Extensive libraries:** Η **python** περιέχει μια τεράστια γκάμα από βιβλιοθήκες οι οποίες κάνουν πάρα πολύ εύκολη την ζωή ενός προγραμματιστή αλλά και βοηθάνε στην γρήγορη ανάπτυξη προγραμμάτων [40].
- **Cross-Platform Compatibility:** Ένα πλεονέκτημα της **python** είναι η ικανότητα που έχει να λειτουργεί σε πολλαπλές πλατφόρμες, λειτουργικά συστήματα καθιστώντας την ανάπτυξη εφαρμογών αρκετά εύκολη.

Πακέτα που χρησιμοποιήθηκαν για την ανάπτυξη του προγράμματός μου

1) SpeechRecognition == 3.10.4
2) transformers == 4.45.2
3) torch == 2.4.1+cu121
4) python-dotenv == 1.0.1
5) pyttsx3 == 2.98
6) pillow == 10.4.0
7) sentencepiece == 0.2.0
8) Tkinter == 0.1.0

1. SpeechRecognition
    a. **Version: 3.10.4**
    b. **Περιγραφή**: Η SpeechRecognition βιβλιοθήκη επιτρέπει την εύκολη μετατροπή της φωνής από είσοδο ή αρχείο ήχου σε κείμενο χρησιμοποιώντας διάφορες μετατροπές από SpeechRecognition engines and API. Υποστηρίζει

αρκετές γλώσσες και λειτουργεί είτε με φωνή του χρήστη σε πρώτο χρόνο ή από αρχείο με προεγγεγραμμένη φωνή.

c. **Use Case**: Στο συγκεκριμένο πρόγραμμα η SpeechRecognition χρησιμοποιήθηκε για την εγγραφή φωνής από τον χρήστη στη γλώσσα που επιλέγει ο χρήστης αλλά και σε εντολές που μπορεί να δώσει ο χρήστης με συγκεκριμένες λέξεις [42].

2. Transformers
   a. **Version:** 4.45.2
   b. **Περιγραφή:** Οι **Transformers** που έχουν υλοποιηθεί από την HuggingFace είναι μια βιβλιοθήκη που προσφέρει μια τεράστια ποικιλία από μοντέλα **NLP (Natural Language Processing)** αλλά και τροποποιημένες εκδόσεις αυτών των μοντέλων πάνω σε δεδομένα που έχει επιλέξει ο κάθε χρήστης που τα ανέβασε. Με λίγα λόγια υπάρχουν αρκετά διαφοροποιημένα μοντέλα (προπονημένα σε επιπλέον δεδομένα για καλύτερη απόδοση) που έχουν προπονηθεί σε dataset όπως το CNN/Daily Mail [41] στην δικιά μου περίπτωση για text-summarization. Κάποια από αυτά τα μοντέλα είναι το BART και το T5 αλλά και αρκετά ακόμη. Ουσιαστικά με την παραπάνω βιβλιοθήκη έχουμε εύκολη πρόσβαση σε προ-εκπαιδευμένα μοντέλα για εργασίες όπως text classification, translation, και summarization.
   c. **Use case:** Η συγκεκριμένη βιβλιοθήκη χρησιμοποιείται για την αξιοποίηση των μοντέλων που προσφέρει για text-summarization αλλά και την καλύτερη κατανόηση για το πώς δουλεύουν τα συγκεκριμένα μοντέλα μετά από αρκετές πειραματικές χρήσεις [43].
3. Torch
   a. **Version:** 2.4.1+cu121
   b. **Περιγραφή:** Η Torch είναι μια open-source machine learning library που προσφέρει ένα ευέλικτο framework για την δημιουργία deep learning μοντέλων. Είναι ευρέος γνωστό για τις υπολογιστικές δυνατότητες που έχει πάνω στους tensors αλλά και την επιτάχυνση/εγρήγορση στην εκπαίδευση μοντέλων μέσω της GPU.
   c. **Use case:** Η Torch χρησιμοποιήθηκε ως το framework για την εκπαίδευση των 2 μοντέλων μου σε συγκεκριμένο dataset με δεδομένα για την λειτουργία της text-summarization [44].
4. Python-dotenv
   a. **Version:** 1.01
   b. **Περιγραφή:** Η python-dotenv βιβλιοθήκη χρησιμοποιείται για να διαβάζονται τιμές από ένα .env αρχείο και να χρησιμοποιούνται σαν μεταβλητές σε αρχεία python. Με λίγα λόγια αυτό μας βοηθά να κάνουμε καλύτερη διαχείριση του κώδικα μας σε περίπτωση που αλλάξουν αυτές οι μεταβλητές.
   c. **Use case:** Στο δικό μας πρόγραμμα χρησιμοποιήθηκαν για την ευκολότερη διαχείριση μεταβλητών που αλλάζουν συχνά και βρίσκονται μέσα στο πρόγραμμα παραπάνω από μία φορές , αλλά και όταν γίνεται μεταφορά σε διαφορετικό χρήστη για το τρέξιμο του προγράμματος χρειάζονται να αλλάξουν τα paths οπότε και εκεί μπορεί να γίνει ευκολότερη η ζωή του προγραμματιστή, τέλος μπορούμε να κρύψουμε έτσι και τα API keys [45].
5. Pyttsx3
   a. **Version:**2.98

b. **Περιγραφή**: Η Pyttsx3 είναι μια βιβλιοθήκη text-to-speech που μπορεί να χρησιμοποιηθεί και χωρίς σύνδεση στο διαδίκτυο και έχει αρκετά speech engines.
c. **Use case:** Στο πρόγραμμα μου χρησιμοποιείται σαν accessibility feature αλλά και για μεγαλύτερη ευκολία στον χρήστη εφόσον με το να έχεις ήχο στις εντολές και το τί ζητάει το πρόγραμμα εξαλείφονται τα περιθώρια λάθους [46].

6. Pillow
   a. **Version:** 10.4.0
   b. **Περιγραφή:** Η βιβλιοθήκη pillow χρησιμοποιείται για την μετατροπή και επεξεργασία εικόνων μέσα σε εφαρμογές python.
   c. **Use case :** Στο πρόγραμμα μου χρησιμοποιήθηκε για την προβολή εικόνων [47].
7. Sentencepiece
   a. **Version : 0.2.0**
   b. **Περιγραφή:** Η Sentencepiece βιβλιοθήκη είναι μια γλωσσικός ανεξάρτητη subword tokenizer και detokenizer σχεδιασμένη για neural-based δημιουργίες κειμένου.
   c. **Use case:** Στην συγκεκριμένη εργασία χρησιμοποιήθηκε για την προ επεξεργασία δεδομένων κειμένου διαχωρίζοντας τις subword units, οι οποίες αύξησαν την αποδοτικότητα των **NLP (Natural Language processing)** μοντέλων [48].
8. Tkinter
   a. **Version:** 0.1.0
   b. **Περιγραφή:** Η βιβλιοθήκη Tkinter είναι από τις πιο απλές μεθόδους για την προβολή GUI, δίνοντας την δυνατότητα στον προγραμματιστή να έχει ένα εργαλείο με το οποίο μπορεί να φτιάξει εφαρμογές σε παράθυρα με κουμπιά, μενού, και άλλα δια δραστικά στοιχεία που μπορεί να επιλέξει.
   c. **Use case:** Η βιβλιοθήκη Tkinter στην συγκεκριμένη εφαρμογή χρησιμοποιείται για την Οπτικοποίηση της μετάφρασης του κειμένου σε εικόνες νοηματικής, και βοηθάει στην καλύτερη κατανόηση κάθε γράμματος καθώς αφού προβάλει την κάθε φωτογραφία έχει και το αντίστοιχο γράμμα από κάτω [49].

4.2 Ανάλυση λειτουργίας προγράμματος

Στο συγκεκριμένο υποκεφάλαιο θα δούμε αναλυτικά τα περιεχόμενα του project folder μου, που χρησιμοποιείται το καθένα και τι σκοπό έχει. Το project έχει υλοποιηθεί στην γλώσσα προγραμματισμού **python** στο **VSCODE IDE** (Integrated development environment) και περιλαμβάνει τους παρακάτω φακέλους.

- .vscode
- __pycache__
- ASL_Photos
- Csv_files
- Dataset
- Evaluation
- GSL_Photos
- Model-bart

- Model-t5
- Performance scripts
- Venv
- .env
- .gitignore
- Functions.py
- Main.py
- Mapping.py

Τα παραπάνω αποτελούν το τελικό μου project και το καθένα έχει μια συγκεκριμένη χρήση που θα εξηγήσω παρακάτω. Μόνη εξαίρεση θα αποτελέσουν τα αρχεία **functions.py, main.py, mapping.py** που απλά θα αναφέρω αλλά θα εξηγήσω αναλυτικότερα στο επόμενο υποκεφάλαιο που θα δούμε αναλυτικά την κάθε συνάρτηση μέσα στα αρχεία και που συνεισφέρει η καθεμία.

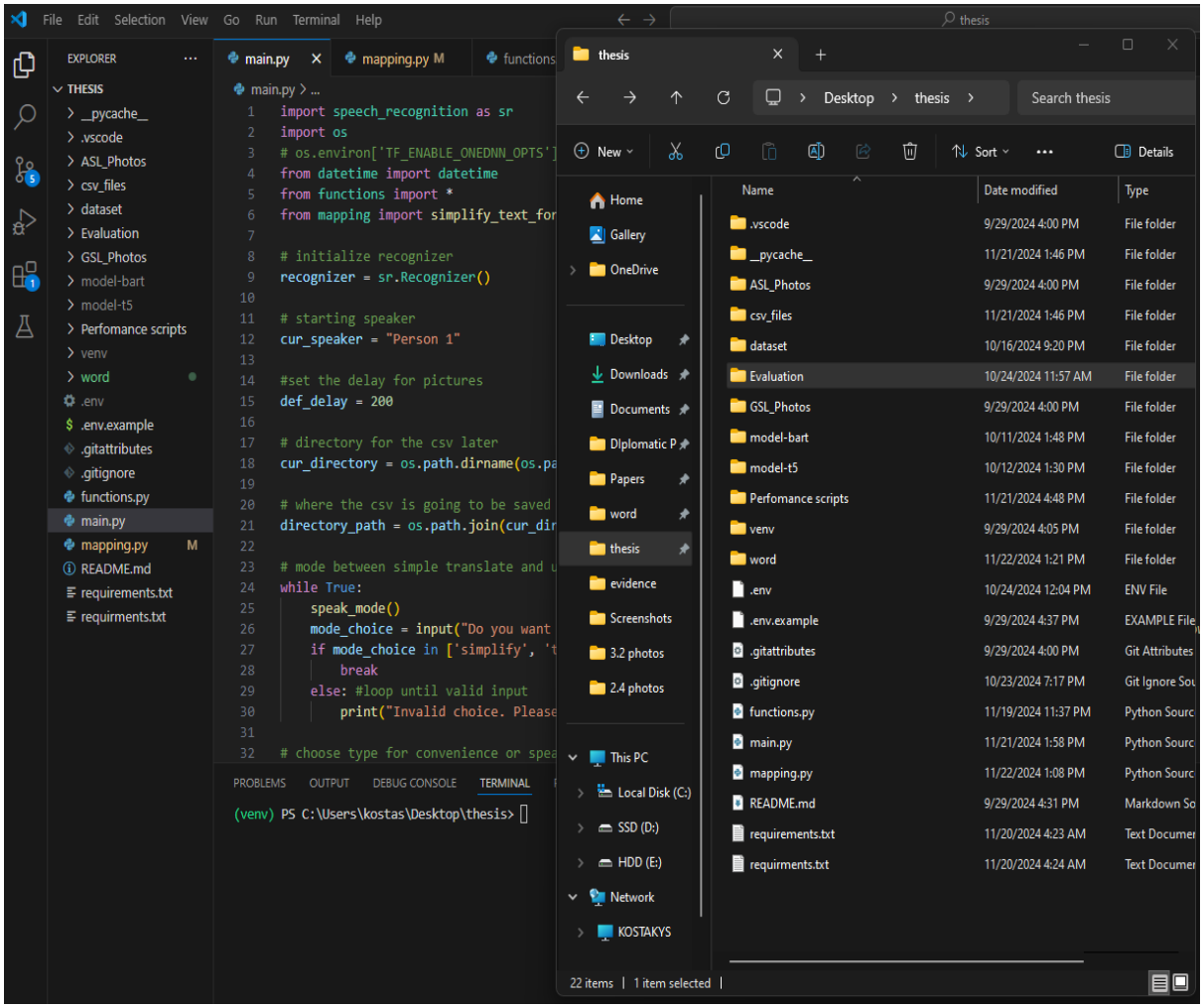

Figure 46 Φάκελος του project μου με όλα τα απαραίτητα αρχεία για την τελική εφαρμογή.

.vscode

- **Use case:** Ο φάκελος **.vscode** περιέχει ρυθμίσεις που σχετίζονται με το συγκεκριμένο project στο VSCODE, και μας επιτρέπει να προσαρμόζουμε τον τρόπο με τον οποίο λειτουργεί το VSCODE για το συγκεκριμένο project.
- **Περιεχόμενα:** περιλαμβάνει διάφορα .json τύπου αρχεία τα οποία περιέχουν πληροφορίες για το project. Στη συγκεκριμένη περίπτωση περιέχει μόνο ένα **settings.json** το οποίο περιέχει μόνο την παρακάτω εντολή για απόκρυψη προειδοποιήσεων για το όριο στο git (συνήθως χρησιμοποιείται σε μεγάλα project).

```
{
    "git.ignoreLimitWarning": true
}
```

- **Χρήση:** Βοηθάει σε περίπτωση που το project εκτελείται/αναπτύσσεται από πολλαπλούς προγραμματιστές σε διαφορετικούς υπολογιστές. Στην περίπτωση μου καθώς δεν είχα συνέχεια πρόσβαση στον σταθερό υπολογιστή μου χρησιμοποιούσα το GitHub σαν ένα τρόπο διαμοιρασμού και συγχρονισμού του Project μου σε διαφορετικούς υπολογιστές ώστε να είμαι όσο το δυνατόν πιο αποδοτικός γίνεται

__pycache__

- **Use case:** Ο φάκελος **__pycache__** δημιουργείται αυτόματα από την python και περιέχει τις προ-μεταγλωττισμένες εκδόσεις αρχείων .py, αυτό βοηθά στο να γίνεται πιο γρήγορα η εκτέλεση του κώδικα καθώς δεν χρειάζεται να μεταγλωττίζονται ξανά και ξανά αρχεία που παραμένουν ίδια.
- **Περιεχόμενα:** περιέχει τα παρακάτω αρχεία
    - **functions.cpython-312.pyc**
    - **image_display.cpython-312.pyc**
    - **mapping.cpython-312.pyc**
    - **mappingtest.cpython-312.pyc**
    - **mapping_testing.cpython-312.pyc**
    - **ui.cpython-312.pyc**
- **Χρήση:** Ακόμα και αν διαγραφεί ο παραπάνω φάκελος τα αρχεία αυτά κάθε φορά που γίνεται εκκίνηση του προγράμματος δημιουργούνται αυτόματα εάν δεν υπάρχουν.

ASL_Photos

- **Use case**: είναι ένας φάκελος που περιέχει εικόνες με νοηματικά σύμβολα για τα γράμματα του αγγλικού αλφάβητου αλλά και εικόνες με αριθμούς από το 0 έως το 9. Κάθε γράμμα έχει συγκεκριμένο format για να μπορεί να το αντιλαμβάνεται σωστά το πρόγραμμα, πχ: **“Sign_language_B.png”**
- **Περιεχόμενα:**
  Sign_language_0.png,Sign_language_1.png,Sign_language_2.png,Sign_language_3.png,Sign_language_4.png,Sign_language_5.png,Sign_language_6.png,Sign_language_7.png,Sign_language_8.png,Sign_language_9.png,Sign_language_a.png,Sign_language_B.png,Sign_language_C.png,Sign_language_D.png,Sign_language_E.png,Sign_language_F.png,Sign_language_G.png,Sign_language_H.png,Sign_language_I.png,Sign_language_J.png,Sign_language_K.png,Sign_language_L.png,Sign_language_M.pn

g,Sign_language_N.png,Sign_language_O.png,Sign_language_P.png,Sign_language_Q.png,Sign_language_R.png,Sign_language_S.png,Sign_language_T.png,Sign_language_U.png,Sign_language_V.png,Sign_language_W.png,Sign_language_X.png,Sign_language_Y.png,Sign_language_Z.png

- **Χρήση:** Το πρόγραμμα χρησιμοποιεί τις παραπάνω φωτογραφίες για την προβολή νοηματικών σημάτων ανάλογα με την εισαγωγή του χρήστη.

csv_files

- **Use case:** Ο συγκεκριμένος φάκελος δημιουργείται από το πρόγραμμα και αποθηκεύει συγκεκριμένα .csv αρχεία
- **Περιεχόμενα:** speech_data_el-GR.csv, speech_data_en-US.gr
- **Χρήση:** Το πρόγραμμα αποθηκεύει την ώρα εισόδου από τον χρήστη, την είσοδο από τον χρήστη, την έξοδο από το πρόγραμμα (αυτό βοηθά περισσότερο όταν χρησιμοποιείται η επιλογή περίληψης), και τέλος ποιος μιλάει.

Dataset

- **Use case:** Εδώ αποθηκεύονται τα dataset που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου ή dataset που θα δημιουργηθούν από την εισαγωγή εξαγωγή δεδομένων κατά την χρήση του προγράμματος.
- **Περιεχόμενα:** wiki.full.aner.ori.test.95.tsv, wiki.full.aner.ori.train.95.tsv
- **Χρήση:** είναι το dataset από το HuggingFace στο οποίο εκπαίδευσα τα 2 μοντέλα που χρησιμοποίησα για να δω ποιο είχε καλύτερο αποτέλεσμα.

Evaluation

- **Use case:** Ο συγκεκριμένος φάκελος περιέχει τα αποτελέσματα από την σύγκριση των 2 μοντέλων σε αρχεία .csv
- **Περιεχόμενα:** evaluation_results.csv, perfomance_metrics.csv
- **Χρήση:** Τα συγκεκριμένα .csv χρησιμοποιούνται για την αποθήκευση δεδομένων από τα 2 μοντέλα κατά τον έλεγχο τους και για να χρησιμοποιηθούν σαν πίνακες στο paper μου.

GSL_Photos

- **Use case:** είναι ένας φάκελος που περιέχει εικόνες με νοηματικά σύμβολα για τα γράμματα του Ελληνικού αλφάβητου αλλά και εικόνες με αριθμούς από το 0 έως το 9. Κάθε γράμμα έχει συγκεκριμένο format για να μπορεί να το αντιλαμβάνεται σωστά το πρόγραμμα, πχ: **“Sign_language_Γ.png”**
- **Περιεχόμενα:** Sign_language_0.png, Sign_language_1.png, Sign_language_10.png, Sign_language_2.png, Sign_language_3.png, Sign_language_4.png, Sign_language_5.png, Sign_language_6.png, Sign_language_7.png, Sign_language_8.png, Sign_language_9.png, Sign_language_Ά.png, Sign_language_Έ.png, Sign_language_Ή.png, Sign_language_Ί.png, Sign_language_Ό.png, Sign_language_Ύ.png, Sign_language_Ώ.png, Sign_language_A.png, Sign_language_B.png, Sign_language_Γ.png, Sign_language_Δ.png, Sign_language_E.png, Sign_language_Z.png, Sign_language_H.png, Sign_language_Θ.png, Sign_language_I.png, Sign_language_K.png, Sign_language_Λ.png, Sign_language_M.png, Sign_language_N.png, Sign_language_Ξ.png, Sign_language_O.png, Sign_language_Π.png, Sign_language_P.png, Sign_language_Σ.png,

Sign_language_T.png, Sign_language_Y.png, Sign_language_Φ.png, Sign_language_X.png, Sign_language_Ψ.png, Sign_language_Ω.png

- **Χρήση:** Το πρόγραμμα χρησιμοποιεί τις παραπάνω φωτογραφίες για την προβολή νοηματικών σημάτων ανάλογα με την εισαγωγή του χρήστη.

Model-bart

- **Use case:** Ο συγκεκριμένος φάκελος περιέχει το προπονημένο μοντέλο BART πάνω στο dataset που προαναφέραμε.
- **Περιεχόμενα:** config.json, generation_config.json, merges.txt, model.safetensors, special_tokens_map.json, tokenizer_config.json, vocab.json
- **Χρήση:** Στον συγκεκριμένο φάκελο έχουμε το μοντέλο που θα χρησιμοποιήσουμε για την περίληψη στο πρόγραμμά μας.

Model-t5

- **Use case:** Ο συγκεκριμένος φάκελος περιέχει το προπονημένο μοντέλο T5 πάνω στο dataset που προαναφέραμε.
- **Περιεχόμενα**: model_epochs2, model_epochs4, model_epochs6, model_epochs8
- **Χρήση:** στο συγκεκριμένο φάκελο βλέπουμε την διαφοροποιημένη προπόνηση των μοντέλων και έπειτα συγκρίνουμε σε ποιο επίπεδο έχουμε την μέγιστη απόδοση.

Perfomance scripts

- **Use case:** Ο συγκεκριμένος φάκελος περιέχει συναρτήσεις για την ανάλυση αποδοτικότητας του κάθε μοντέλου διαφορετικά, και στο τέλος ένα μεγάλο πρόγραμμα που αναλύει και τα δύο και βγάζει συμπέρασμα για το ποιο από τα δύο πρέπει να χρησιμοποιηθούν στο πρόγραμμά μας με βάση το dataset που έχουμε.
- **Περιεχόμενα:** evaluation_results.csv, perfomance-final.py, perfomance-metrics-bart.py, perfomance-metrics-t5.py
- **Χρήση:** Τα παραπάνω δημιουργήθηκαν για τον σκοπό της αξιολόγησης έτσι ώστε να μην κάνουμε μια τυχαία επιλογή ανάμεσα στα 2 μοντέλα, αλλά με βάση συναρτήσεις αξιολόγησης.

Venv

- **Use case:** Ο συγκεκριμένος φάκελος περιέχει τις βιβλιοθήκες του προγράμματος μας αλλά και στοιχεία για το ποια έκδοση της python χρησιμοποιούμε, συνήθως εκτός από την απομόνωση των πακέτων με τις εκδόσεις τους έχουμε και διαφορετική έκδοση και για την python.
- **Περιεχόμενα:** include, Lib, Scripts, Share, pyenv.cfg
- **Χρήση:** Αυτό που θέλω να επιτύχω με το virtual environment είναι απομόνωση από άλλα project (εκδόσεις βιβλιοθηκών και έκδοση python) για να αποφύγω τριβές μεταξύ των προγραμμάτων που υλοποιούνται και να εξαλείψω παράγοντες σφάλματος. Τέλος βοηθάει αρκετά στην φορητότητα του project σε διαφορετικούς υπολογιστές, για να ενεργοποιηθεί το virtual environment χρησιμοποιούμε την παρακάτω εντολή **.\venv\scripts\activate**

.env

- **Use case:** Στον συγκεκριμένο φάκελο έχουμε την διαδρομή για συγκεκριμένες μεταβλητές που αλλάζουν μέσα στο πρόγραμμά μας.
- **Περιεχόμενα:** ASL, GSL, MODEL_BART,MODEL_T5,TEST_DATA_PATH

- **Χρήση:** Σε περίπτωση εναλλαγής του προγράμματος από πολλούς χρήστες, ταυτόχρονος προγραμματισμός, να μην χρειάζεται η εναλλαγή διαδρομών σε συγκεκριμένες μεταβλητές για να μην υπάρχει τριβή.

.gitignore

- **Use case**: Το συγκεκριμένο αρχείο περιέχει τα ονόματα των αρχείων που δεν θα συμπεριληφθούν όταν κάνουμε push/pull από το GitHub repository μας διότι περιλαμβάνουν στοιχεία που σε κάθε τοπικό υπολογιστή είναι διαφορετικά.
- **Περιεχόμενα:** .gitignore
- **Χρήση:** Κάθε χρήστης έχει διαφορετικές τιμές για συγκεκριμένες μεταβλητές, έτσι για να μην γίνεται κάθε φορά αυτό το μπέρδεμα και να υπάρχει μια συνοχή και συνεργασία στους προγραμματιστές.

Functions.py

- **Use case:** Το συγκεκριμένο αρχείο περιέχει συναρτήσεις που χρησιμοποιούνται στο main.py κάνοντας απλά **from functions import *** ώστε να μπορέσουμε να χρησιμοποιήσουμε της συναρτήσεις.

Main.py

- **Use case:** Το συγκεκριμένο αρχείο είναι η βάση του προγράμματος μας που καλεί όλες τις συναρτήσεις που χρειάζονται από τα 2 υπολειπόμενα αρχεία και συγκροτεί το πρόγραμμά μας.

Mapping.py

- **Use case:** Το συγκεκριμένο αρχείο αποτελεί την «δομή» του προγράμματος καθώς εδώ μέσα είναι οι συναρτήσεις που κάνουν αντικατάσταση την είσοδο του χρήστη είτε από το πληκτρολόγιο ή από το μικρόφωνο με τα σύμβολα της νοηματικής, επίσης περιλαμβάνει και την συνάρτηση που κάνει την περίληψη του κειμένου και τέλος η συνάρτηση που οπτικοποιεί το πρόγραμμα μας.

Ας δούμε τώρα πώς λειτουργεί το πρόγραμμα μας από την αρχή και ποιες είναι οι επιλογές που έχει ο χρήστης κατά την εκτέλεση. Αρχικά πρέπει να ενεργοποιήσουμε το virtual environment μας και ύστερά με την εντολή **py .\main.py** να εκτελέσουμε το πρόγραμμα μας.

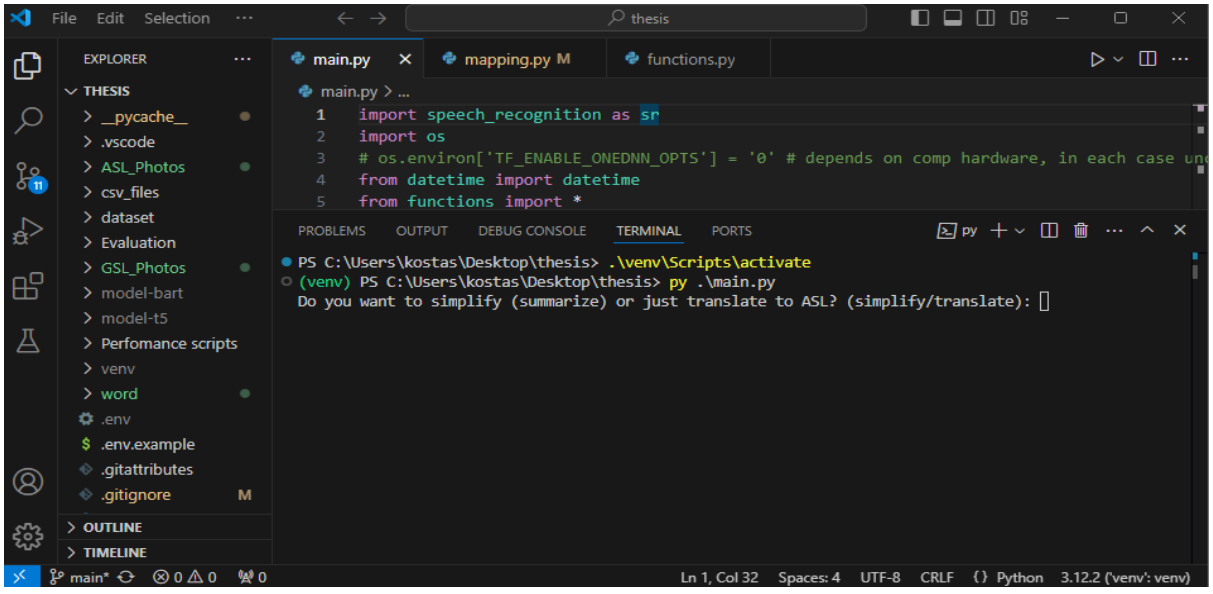

Figure 47 Στην συγκεκριμένη εικόνα βλέπουμε την ενεργοποίηση του περιβάλλοντος και την εντολή για να τρέξουμε το πρόγραμμα.

Επίσης βλέπουμε ότι το πρόγραμμα μας δίνει 2 επιλογές. Είτε να κάνουμε απλοποίηση/περίληψη (simplify) ή απλή μετάφραση (translate). Σε περίπτωση που έχουμε λάθος εισαγωγή υπάρχει κατάλληλος έλεγχος, και σε περίπτωση λάθος εισαγωγής το πρόγραμμα ρωτάει ξανά μέχρι να δοθεί η σωστή εισαγωγή.

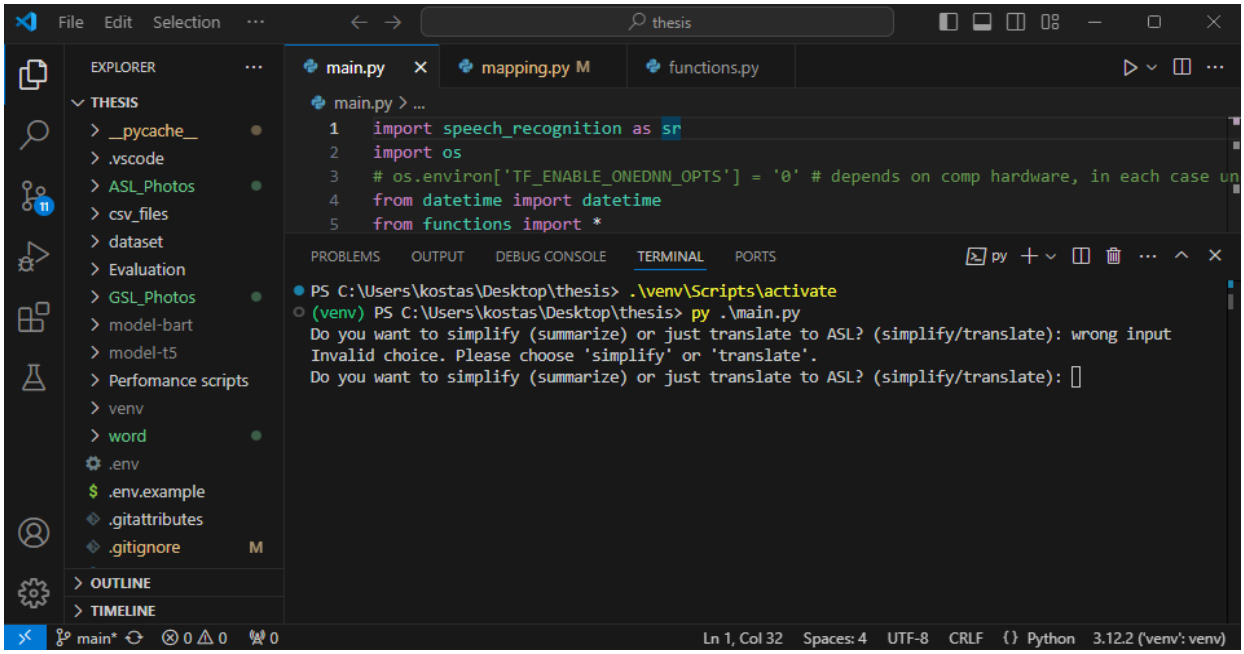

Figure 48 Βλέπουμε τί γίνεται σε περίπτωση λάθος εισαγωγής.

Είτε διαλέξουμε την περίληψη ή την απλή μετάφραση η διαδικασία στην συνέχεια δεν αλλάζει, οπότε θα βάλουμε περίληψη και θα συνεχίσουμε το πρόγραμμα μας για να δούμε την συνέχεια του προγράμματος και ποια είναι η ροή του. Εφόσον διαλέξουμε την παραπάνω επιλογή στην συνέχεια έχουμε επιλογή για να γράψουμε ή να έχουμε το μικρόφωνο σαν είσοδο, και εδώ δουλεύει η λάθος εισαγωγή όπως και στο προηγούμενο μέρος, αλλά θα επιλέξουμε σαν είσοδο το πληκτρολόγιο για την δικιά μας ευκολία.

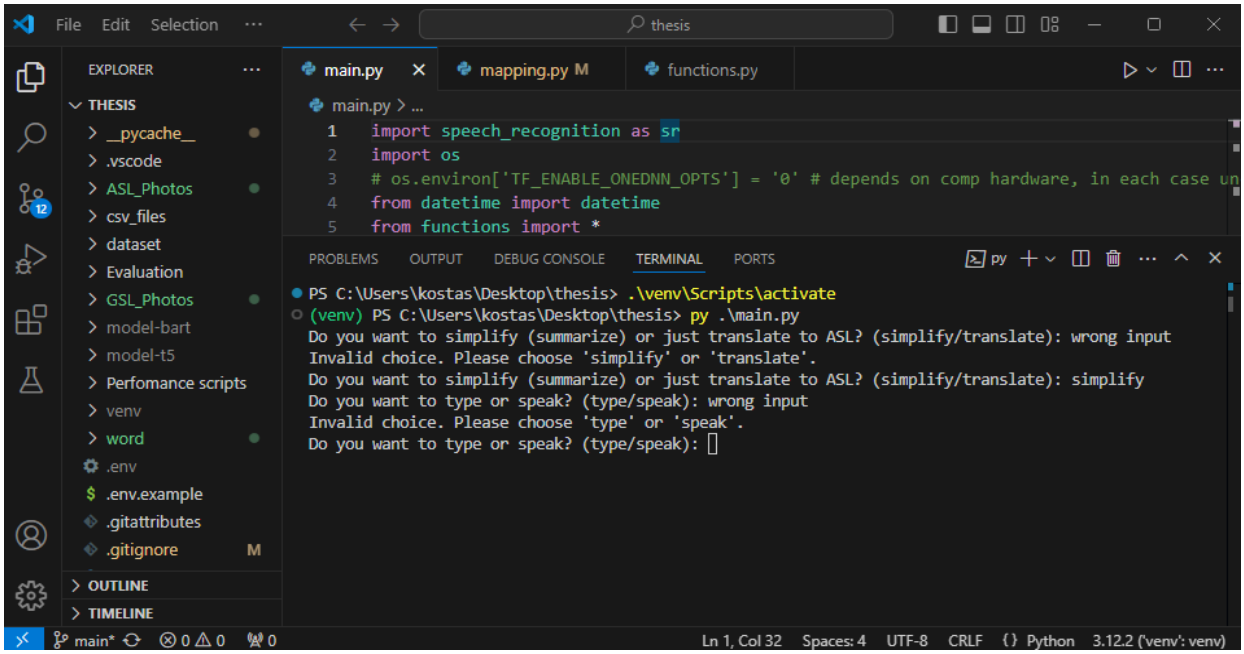

Figure 49 Λάθος εισαγωγή στην εισαγωγή κειμένου.

Τώρα προχωράμε στην επιλογή γλώσσας που είναι είτε Ελληνικά ή Αγγλικά και η επιλογή γίνεται με βάση τον αριθμό 1 για Αγγλικά και αριθμό 2 για Ελληνικά. Στην συγκεκριμένη περίπτωση αν επιλέξουμε λάθος εισαγωγή γίνεται κατευθείαν ανάθεση στα αγγλικά και το πρόγραμμα μας περιμένει για εισαγωγή από τον χρήστη εφόσον επιλέξαμε την πληκτρολόγηση για εισαγωγή.

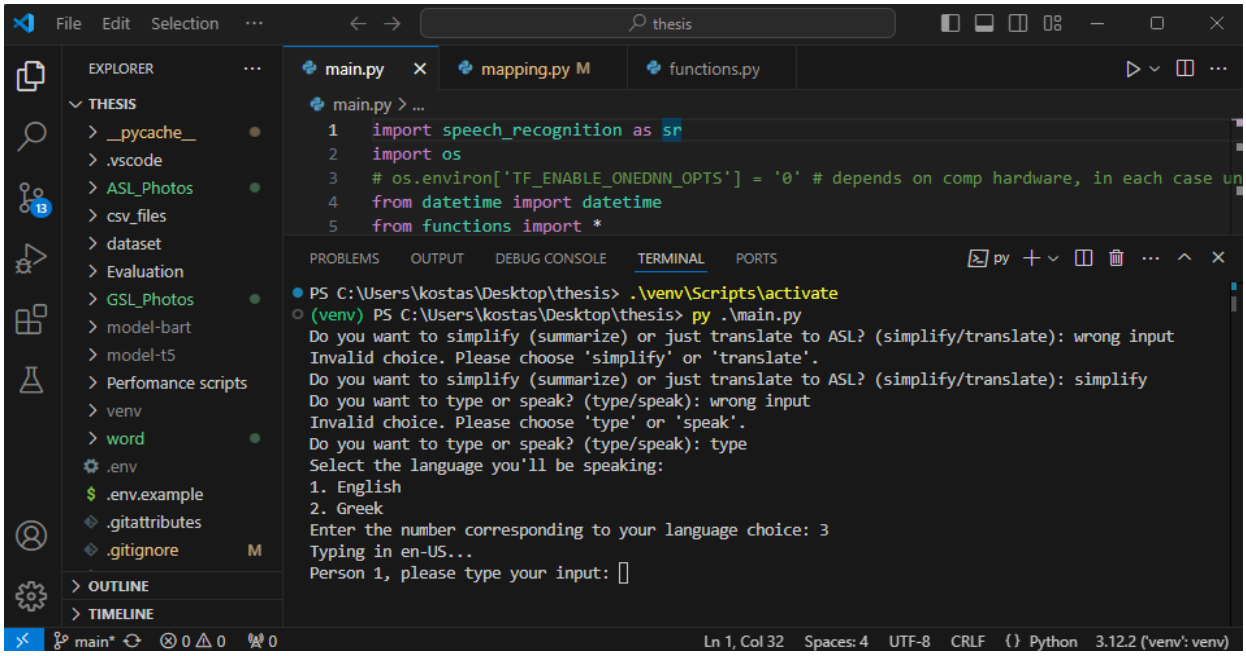

Figure 50 Λάθος εισαγωγή στην επιλογή γλώσσας.

Εφόσον φτάσαμε στο συγκεκριμένο σημείο θα βάλουμε σαν εισαγωγή την παρακάτω πρόταση: **“This is the test for the paper I’m writing, note some specials characters won’t appear. Also simplify works for bigger sentences and might not for this one.”**

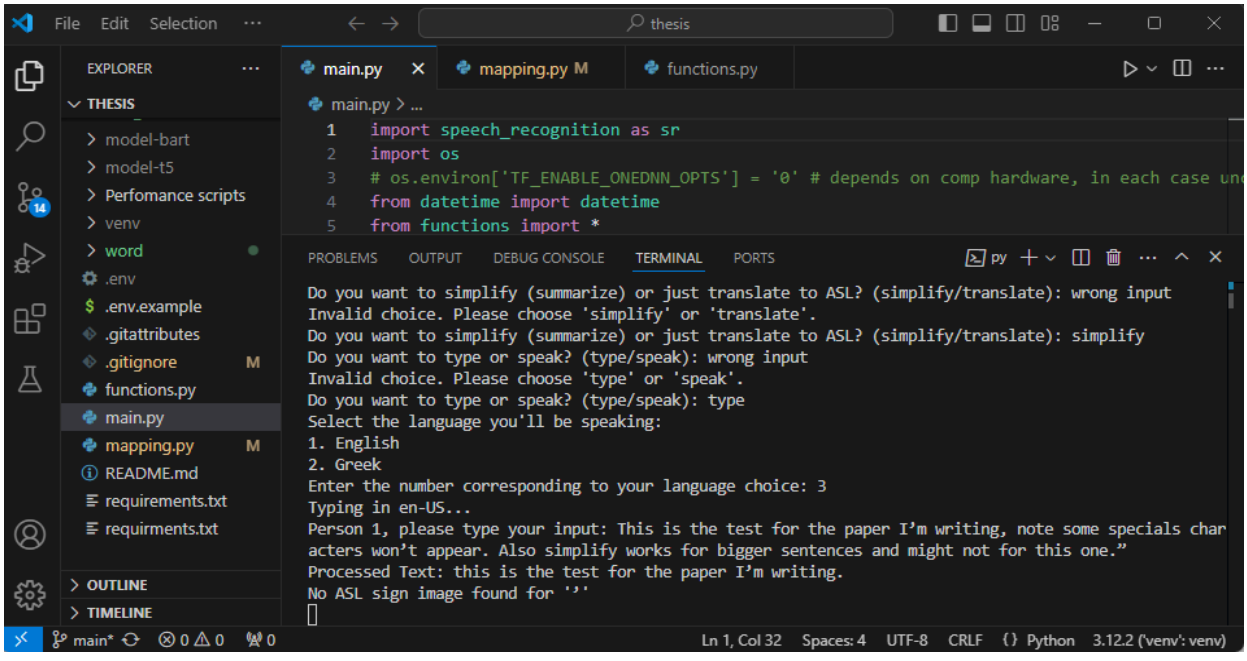

Figure 51 Βλέπουμε την εισαγωγή του χρήστη και μετά την περίληψη που κάνει το μοντέλο μας.

Αυτό είναι το γραφικό περιβάλλον που έχω δημιουργήσει για το πρόγραμμα μου. Επειδή είναι φωτογραφία δεν γίνεται να φανεί ότι κάθε φωτογραφία εμφανίζεται μία προς μία αλλά το τελικό αποτέλεσμα, αλλά γενικά πρώτα σαν main display εμφανίζουμε το νοηματικό εικονίδιο και από κάτω μετά το ίδιο εικονίδιο με το γράμμα που του αντιστοιχεί.

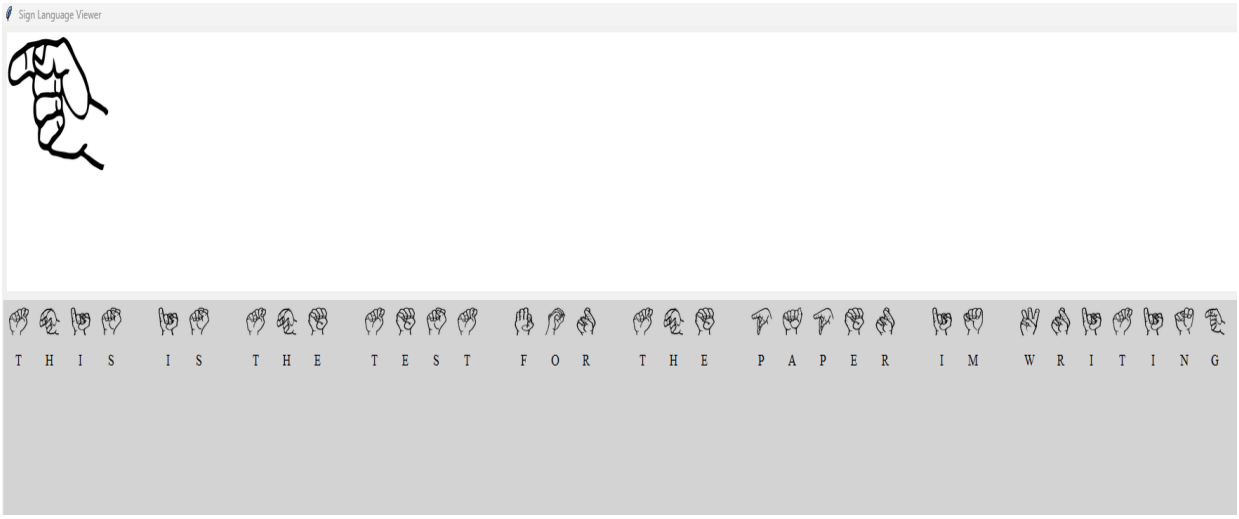

Figure 52 Βλέπουμε το τελικό αποτέλεσμα που εμφανίζει μόνο το processed text.

Τέλος για να γίνει έξοδος από τον χρήστη από το γραφικό περιβάλλον απλά πατάμε ένα escape button στο πληκτρολόγιο μας και μας δίνεται η δυνατότητα να γράψουμε ξανά μέχρι να χρησιμοποιήσουμε την λέξη κλειδί για να τερματιστεί το πρόγραμμα.

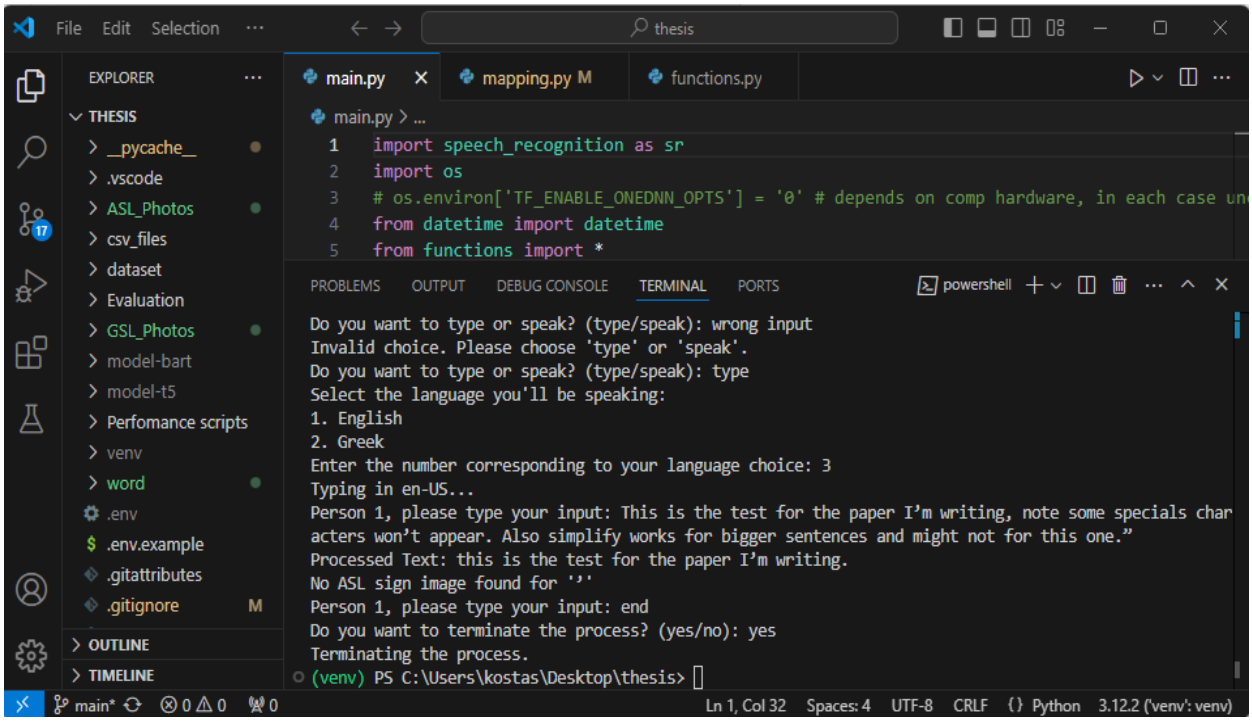

Figure 53 Βλέπουμε ότι με την κατάλληλη λέξη για τερματισμό το πρόγραμμα μας δίνει την επιλογή τερματισμού.

4.3 Διάρθρωση κώδικα (main.py, mapping.py, functions.py)

Όπως έχω προαναφέρει το πρόγραμμά μου αποτελείται από 3 βασικά αρχεία όπου το αρχείο που τρέχει το πρόγραμμά μου ουσιαστικά είναι η **main.py** η οποία καλεί συναρτήσεις από τα άλλα 2 αρχεία **function.py και mapping.py** όποτε χρειάζεται έτσι ώστε να τρέχει το πρόγραμμα. Στο συγκεκριμένο κεφάλαιο θα δούμε από ποιες συναρτήσεις απαρτίζεται το κάθε αρχείο μας και τί προσφέρει η καθεμία στο τελικό μας πρόγραμμα.

4.3.1 Main.py

Αρχικά έχουμε την **main.py** που ξεκινάμε με τις βιβλιοθήκες που θα χρειαστούμε στο συγκεκριμένο αρχείο. Έχουμε την βιβλιοθήκες **SpeechRecognition, os και datetime** αλλά θα δούμε κιόλας ότι κάνουμε import και τα δύο άλλα αρχεία διότι θα χρειαστούμε σημαντικές πληροφορίες από αυτά για την υλοποίηση του προγράμματος μας.

SpeechRecognition

Η **SpeechRecognition** βιβλιοθήκη είναι υπεύθυνη για την επεξεργασία και καταγραφή του ήχου από το μικρόφωνο του χρήστη.

OS

Η **OS** βιβλιοθήκη χρησιμοποιείται για την εύρεση διαδρομών στον υπολογιστή μου τοπικά, για εργασίες όπως η καταγραφή του csv σε σωστή διαδρομή.

Datetime

Παρόμοια και η **Datetime** χρησιμοποιείται για την καταγραφή ώρας για τα δεδομένα μας στο csv έτσι ώστε να έχουμε μια σωστή καταγραφέι των δεδομένων και να μπορέσουμε να τα αξιοποιήσουμε σωστά.

Figure 54 βιβλιοθήκες που φορτώνουμε στο αρχείο main.py.

Προχωρώντας παρακάτω πρωτού συναντήσουμε κάποια συνάρτηση έχουμε κάποες μεταβλητές που δηλώνουμε στην αρχή του προγράμματος. Αρχικά δηλώνουμε την μεταβλητή **recognizer** που θα ορίσουμε μια class Recognizer που θα περιέχει όλες τις δυνατότητες της βιβλιοθήκης speech recognition. Μετά δηλώνουμε την μεταβλητή **cur_speaker** η οποία περιέχει το όνομα του πρώτου ομιλιτή για να εγγραφεί σωστά στο csv που κρατάει τις λεπτομέρειες όταν δίνουμε input και παίρνουμε output. Έχουμε επίσης και την **def_delay** η οποία υπάρχει για να καθορίζουμε πόσο γρήγορα θα γίνεται η προβολή των νοηματικών συμβόλων. Επίσης υπάρχει και η **cur_directory** που ορίζει σε ποιο path θα δημιουργηθεί το csv αρχείο μας αν δεν υπάρχει ήδη, και τέλος υπάρχει το **directory_path** που έχει σε ποια διαδρομή πρέπει να δημιουργηθούν τα αρχεία και πώς θα ονομάζεται ο φάκελος.

```python
# initialize recognizer
recognizer = sr.Recognizer()

# starting speaker
cur_speaker = "Person 1"

#set the delay for pictures
def_delay = 200

# directory for the csv later
cur_directory = os.path.dirname(os.path.abspath(__file__))

# where the csv is going to be saved, if the folder doesnt exist
directory_path = os.path.join(cur_directory, "csv_files")

```

Figure 55 μεταβλητές του προγράμματος.

Στην συνέχεια του προγράμματος έχουμε δύο while loops τα οποία έχουν και αποτροπή σφάλματος κατά την είσοδο του χρήστη έτσι ώστε να λαμβάνουν μόνο μία από τις δύο επιλογές που προτείνει ο προγραμματιστής.

```python
# mode between simple translate and using a ML model for summarize big sentences
while True:
    speak_mode()
    mode_choice = input("Do you want to simplify (summarize) or just translate to ASL? (simplify/translate): ").strip().lower()
    if mode_choice in ['simplify', 'translate']:
        break
    else: #loop until valid input
        print("Invalid choice. Please choose 'simplify' or 'translate'.")

# choose type for convenience or speak to try the speech recognition
while True:
    speak_input()
    input_mode = input("Do you want to type or speak? (type/speak): ").strip().lower() #removes white space and lowers text
    if input_mode in ['speak', 'type']:
        break
    else: #loop until valid input
        print("Invalid choice. Please choose 'type' or 'speak'.")
```

Figure 56 Δύο while loops για την εισαγωγή δεδομένων.

Έπειτα αυτό που ακολουθεί είναι μια **if statement** η οποία ελέγχει το **input_mode** από το δεύτερο while loop και ανάλογα μας βάζει στην σωστή κατεύθυνση του προγράμματος. Εφόσον το **input_mode** είναι **'speak'** μπαίνει κατευθείαν στην **if** και ξεκινάει με **with sr.Microphone() as source:** η οποία ορίζει το μικρόφωνο μου σαν default (εάν υπάρχει) και ξεκινάει να μας ρωτάει για το ποια γλώσσα επιλέγουμε, έπειτα σε μια μεταβλητή **lang** αποθηκεύει την επιλογή μας. Σε αυτό το σημείο να αναφέρω ότι η **speak_language_prompt()** και **select_language()** είναι δύο συναρτήσεις οι οποίες καλούνται από την function.py., έπειτα δημιουργεί το csv file και προσθέτει ένα ambient_noise adjuster για το μικρόφωνο μας που θα βοηθήσει στην καλύτερη καταγραφή του ήχου.

```python
if input_mode == 'speak':
    # use the default microphone as source, bot asks what language will the user use.
    with sr.Microphone() as source:
        speak_language_prompt()  # language selection
        lang = select_language()
        print(f"Listening in {lang}...")

        # csv file creation
        csv_filename = os.path.join(directory_path, f"speech_data_{lang}.csv")

        # adjust for ambient noise if necessary
        recognizer.adjust_for_ambient_noise(source)
```

Figure 57 Αρχή της if statement από το main.py

Εφόσον περάσουμε και από αυτό το στάδιο φτάνουμε στο στάδιο που έχουμε την πρώτη **while loop** με την **try:** εφόσον έχουμε εγγραφή μικροφώνου (για τον εντοπισμό errors) αλλά μέσα στην **while loop** ξεκινάμε και έχουμε μια μεταβλητή **audio** στην οποία αποθηκεύουμε την εγγραφή της φωνής μας και στην συνέχεια την περνάμε σε μια μεταβλητή **text** που απαρτίζεται από την μεταβλητή μας **recognizer** που θέσαμε πρωτύτερα και έχει την μέθοδο **recognize_google(text,lang=language)** για την μετατροπή του ήχου σε text. Στη συνέχεια εφόσον η περίληψη δεν υποστηρίζεται στα Ελληνικά εάν η **lang** είναι 'el-GR' το **simplified_text** μας θα γίνει το κανονικό text, αλλά εάν δεν είναι 'el-GR' το **lang** και το **mode_choice** είναι 'simplify' τότε το **simplified_text** θα δοθεί από την συνάρτηση **simplify_text_for_asl(text)** από το αρχείο **mapping.py** αλλιώς θα γίνει πάλι απλό text.

```
54          # loop for listening to users input
55          while True:
56              try:
57                  # capture user audio
58                  audio = recognizer.listen(source)
59
60                  # convert speech to text using the selected language
61                  text = recognizer.recognize_google(audio, language=lang)
62
63                  # if the language is greek it will only translate, simplify cant work not enough datasets in g
64                  if lang == 'el-GR':
65                      simplified_text = text
66                  else:
67                      # used to choose between translate or simplify
68                      if mode_choice == 'simplify':
69                          simplified_text = simplify_text_for_asl(text)
70                      else:
71                          simplified_text = text  # if anything else than simplify it will just translate the te
72
```

Figure 58 Αρχή της while loop στην main για το input_mode='speak'.

Στην συνέχεια πριν προχωρήσουμε στην εκτύπωση αποτελεσμάτων έχουμε άλλες δύο συναρτήσεις που ελέγχουν για λέξεις κλειδιά, την **check_switch_command(text,lang)** και την **check_termination_phrase(text,lang)** και οι δύο είναι από το **functions.py** αρχείο και η μία ελέγχει για να δει αν άλλαξε ο χρήστης του προγράμματος για να το γράψει στο csv, και η άλλη για τον τερματισμό του προγράμματος, να τονίσω ότι η ενεργοποίηση των συναρτήσεων γίνεται μόνο όταν οι λέξεις κλειδιά είναι μόνες τους σε μια πρόταση.

```
73                  # switch the speaker command
74                  if check_switch_command(text, lang):
75                      print("Who is speaking?")
76                      new_speaker = input("Enter the new speaker (e.g., Person 2): ")
77                      cur_speaker = new_speaker
78                      print(f"Switched to {new_speaker}")
79
80                  # termination function
81                  elif check_termination_phrase(text, lang):
82                      speak_termination_prompt()
83                      decision = input("Do you want to terminate the process? (yes/no): ")
84                      if decision.lower() == 'yes':
85                          print("Terminating the process.")
86                          break
87                      else:
88                          print("Resuming...")
```

Figure 59 Έχουμε τις δύο συναρτήσεις για εκτελέσεις λειτουργιών στον κώδικα του προγράμματος.

Τέλος έχουμε την εκτύπωση του απλοποιημένοι ή όχι **text → simplified_text,** μετά έχουμε την μετατροπή του **simplified_text** σε **list (**καθώς δουλεύει για την προβολή των εικόνων στο thumbnail αργότερα**)** και έχουμε την συνάρτηση **display_images_sequentially(simplified_text, lang, def_delay)** η οποία καλείται από την **mapping.py** και είναι υπεύθυνη για την δημιουργία ενός interface για την προβολή των νοηματικών συμβόλων, και αφού γίνει η προβολή έχουμε και την μεταβλητή timestamp η οποία κρατάει την ώρα για το csv στο οποίο αποθηκεύουμε τα δεδομένα μας στην συνάρτηση **save_to_csv(csv_filename, timestamp, text, simplified_text, cur_speaker)** η οποία είναι μέλος του αρχείου **function.py.** Τέλος έχουμε κάποια **except** για να δούμε τα errors που υπάρχουν και δεν μπορούμε να κάνουμε καταγραφή της φωνής.

```
90                  # print procecced text
91                  print(f"Processed Text: {simplified_text}")
92
93                  # helps when they are going to be displayed in the thumbnail
94                  simplified_text = list(simplified_text)
95
96                  #display images
97                  display_images_sequentially(simplified_text, lang, def_delay)
98
99                  # timestamp for csv
100                 timestamp = datetime.now().strftime("%H:%M:%S")
101
102                 # save data with the timestamp,text,simplified_text and the speaker
103                 save_to_csv(csv_filename, timestamp, text, simplified_text, cur_speaker)
104
105             # used to see if the microphone is still capturing audio
106             except sr.UnknownValueError:
107                 print("Could not understand audio.")
108             except sr.RequestError as e:
109                 print(f"Error: {e}")
110
```

Figure 60 Τέλος της πρώτης if statement για input_mode == 'speak'.

Στην περίπτωση που έχουμε **input_mode == 'type'** το μόνο που θα αλλάξει θα είναι πώς γίνεται η εισαγωγή των δεδομένων στην μεταβλητή **text.** Σε αυτή την περίπτωση θα έχουμε τις ίδιες γραμμές κώδικα εκτός από τα σημεία που αφορούν τον **recognizer().**

```
111 elif input_mode == 'type':
112     speak_language_prompt()  # language selection
113     lang = select_language()
114     print(f"Typing in {lang}...")
115
```

Figure 61 Παρόμοια άρθρωση κώδικα με το πρώτο if statement χωρίς τις εντολές για το speech recognition.

Έπειτα το πρόγραμμα δεν διαφοροποιείται καθόλου εκτός από το γεγονός ότι πλέον δεν υπάρχει **try:** άρα δεν υπάρχουν και except για να πιάσουμε errors κατά την εγγραφή από το μικρόφωνο του χρήστη, απλά υπάρχει μια **while True:** η οποία τρέχει τον κώδικά μας μέχρι να εκτελεστεί κάποια εντολή διαφυγής. Πλέον που δεν υπάρχει η εισαγωγή δεδομένων από το μικρόφωνο του χρήστη η εισαγωγή γίνεται από την συγκεκριμένη εντολή **text = input(f"{cur_speaker}, please type your input: ").strip()** η οποία εξασφαλίζει ότι θα γράψει στον χρήστη ότι πρέπει να πληκτρολογήσει και ότι θα πάρει σαν είσοδο μόνο ότι έγραψε ο χρήστης χωρίς κενά. Παρακάτω είναι ολόκληρος ο κώδικας.

```
    while True:
        # manualy input the user text
        text = input(f"{cur_speaker}, please type your input: ").strip()

        # switch user
        if check_switch_command(text, lang):
            print("Who is speaking?")
            new_speaker = input("Enter the new speaker (e.g., Person 2): ")
            cur_speaker = new_speaker
            print(f"Switched to {new_speaker}")

        # termination phrase
        elif check_termination_phrase(text, lang):
            speak_termination_prompt()
            decision = input("Do you want to terminate the process? (yes/no): ")
            if decision.lower() == 'yes':
                print("Terminating the process.")
                break
            else:
                print("Resuming...")

        # if the language is greek it will only translate, simplify cant work not enough datasets in greek t
        if lang == 'el-GR':
            simplified_text = text
        else:
            # used to choose between translate or simplify
            if mode_choice == 'simplify':
                simplified_text = simplify_text_for_asl(text)
            else:
                simplified_text = text  # if anything else than simplify it will just translate the text

        #print text to ASL
        print(f"Processed Text: {simplified_text}")

        # display the text to ASL images
        display_images_sequentially(simplified_text, lang, def_delay)

        # timestamps for csv
        timestamp = datetime.now().strftime("%H:%M:%S")

        # save data with the timestamp,text,simplified_text and the speaker
        save_to_csv(csv_filename, timestamp, text, simplified_text, cur_speaker)

```

Figure 62 Τελευταίο μέρος του κώδικα main.py όταν ο χρήστης χρησιμοποιεί input_mode ==’type’.

4.3.2 Mapping.py

Η επόμενη συνάρτηση που είναι μέρος του προγράμματος μας είναι η **mapping.py** η οποία περιέχει τις βιβλιοθήκες **os, PIL, Tkinter, transformers** και **dot-env.**

OS

Η **OS** βιβλιοθήκη χρησιμοποιείται για την εύρεση διαδρομών στον υπολογιστή μου τοπικά, για εργασίες όπως η καταγραφή του csv σε σωστή διαδρομή.

PIL

Η **PIL** βιβλιοθήκη χρησιμοποιείται για την μορφοποίηση φωτογραφιών στην python και από την συγκεκριμένη βιβλιοθήκη εισάγουμε την **Image** και **ImageTk.**

Tkinter

Η **Tkinter** βιβλιοθήκη χρησιμοποιείται για την δημιουργία γραφικού περιβάλλοντος για να μπορούν να προβληθούν οι εικόνες.

Transformers

Η **Transformers** βιβλιοθήκη χρησιμοποιείται για την φόρτωση των προ-εκπαιδευμένων μοντέλων για την δημιουργία περίληψης από την εισαγωγή του χρήστη.

Dot-env

Η **Dot-env** βιβλιοθήκη χρησιμοποιείται για την φόρτωση μεταβλητών από το περιβάλλον του χρήστη μέσα από ένα αρχείο **.env.**

Το συγκεκριμένο αρχείο αποτελείται από τρείς βασικές συναρτήσεις για τον κώδικα μας την **simplify_text_for_asl(input_text), map_text_to_asl_images(text, language) και display_images_sequentially(text, lang, delay, word_gap=1).** Πριν όμως φτάσουμε στις τρείς συναρτήσεις έχουμε και έξι γραμμές κώδικα στις οποίες αρχικά φορτώνουμε τις μεταβλητές μας από το περιβάλλον (Διαδρομή για τις φωτογραφίες GSL,ASL και διαδρομή για το εκπαιδευμένο μοντέλο μας). Οπότε έχουμε αρχικά την **load_dotenv()** που παίρνει τις μεταβλητές από το αρχείο .env και τις διαθέτει στο πρόγραμμα, μετά έχουμε τις δύο μεταβλητές **ASL_DIR** και **GSL_DIR** που περιέχουν την διαδρομή για τον φάκελο με τα σύμβολα τις νοηματικής γλώσσας, και τέλος τρείς μεταβλητές που περιέχουν την διαδρομή για το μοντέλο μας **model_name,** μετά το **model** που φορτώνει τα **safetensor** δεδομένα από το μοντέλο μας και τέλος το **token** που φορτώνει τα tokens από τον φάκελο του μοντέλου μας.

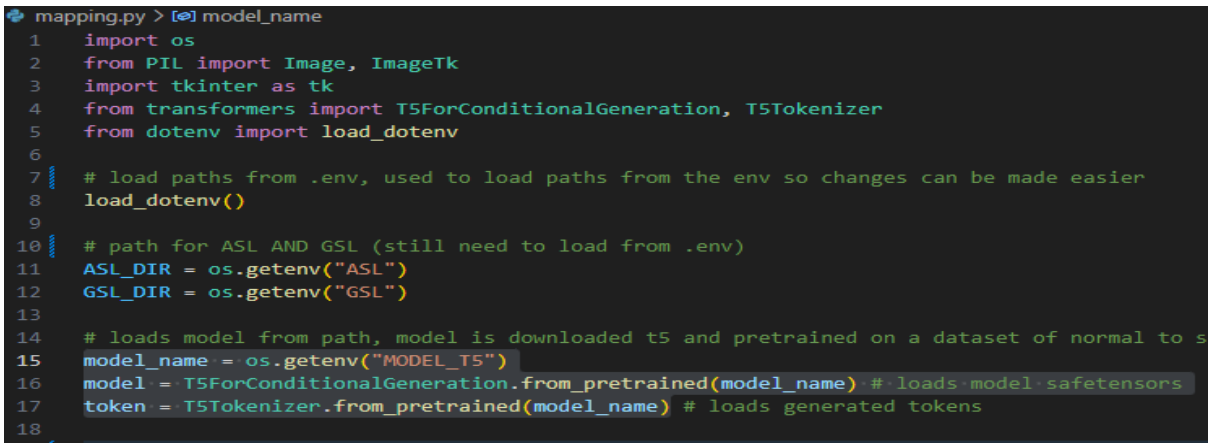

Figure 63 Αρχή του προγράμματος mapping.py.

Αρχικά έχουμε την πρώτη μας συνάρτηση **simplify_text_for_asl(text) (**Figure 64**)**η οποία είναι αυτή που θα χρησιμοποιηθεί για την περίληψη του κειμένου μας εφόσον ζητηθεί από τον χρήστη. Ξεκινάμε με μια **try:** που έχουμε σε περίπτωση κάποιου error από την έξοδο του προγράμματος μας, στην συνέχεια μορφοποιούμε το **text** μας σε **"simplify" + text** έτσι ώστε να "πούμε" στο μοντέλο μας τι ακριβώς θέλουμε να κάνει αν και το έχουμε εκπαιδεύσει επιπλέον σε dataset που κάνει ακριβώς αυτό αλλά σε κάποιες περιπτώσεις δούλευε καλύτερα η περίληψη. Έπειτα έχουμε την μεταβλητή **input_ids** η οποία είναι υπεύθυνη για την μετατροπή του κειμένου από τον χρήση σε token ids για την είσοδο στο μοντέλο μας, και επίσης θέλουμε να γίνει σε ένα **pytorch format** και έπειτα έχουμε το μέγιστο αριθμό tokens για την εισαγωγή και αν τον ξεπεράσει έχουμε **truncation=true** οπού απλά μειώνει τα tokens. Στην συνέχεια σε μια μεταβλητή **summary_ids** θα αποθηκεύσουμε τα tokens μας από το μοντέλο μας, οπότε κάνουμε **model.generate()** με τις παραμέτρους που επιλέγουμε για το πρόγραμμα μας, εγώ έχω βάλει τις παρακάτω:

- Input_ids: Είναι το text μας σε tokens.
- Num_beams: Χρησιμοποιείται για να έχουμε καλύτερα αποτελέσματα με νόημα και να αποφύγουμε το χάσιμο σημαντικών πληροφοριών που μπορεί να είναι κρυμμένες σε μια ακολουθία. Ανάλογα με το νούμερο που βάζουμε έχουμε και διαφορετικά αποτελέσματα, με **num_beams=6** ας φανταστούμε ότι ο υπολογιστής θα έχει από 6 πιθανές προτάσεις να διαλέξει και θα πάρει αυτή με τις μεγαλύτερες πιθανότητες.
- No_repeat_ngrams_size: Αποφυγή της επανάληψής λέξεων σε μια συγκεκριμένη πρόταση.
- Length_penalty: Επηρεάζει το μέγεθος της πρότασης κατά την περίληψη που δημιουργείται, τιμές > 1 κάνουν τον κώδικα να παράγει μικρότερες και πιο συμπυκνωμένες προτάσεις, ενώ οτιδήποτε κάτω από το ένα και ίσο με το ένα παράγει περιλήψεις πιο κοντά στο αρχικό κείμενο.
- Min_length: Είναι το μικρότερο μέγεθος που μπορεί να έχει μια πρόταση.
- Max_length: Είναι η μέγιστη έξοδος που μπορεί να έχει μια πρόταση του μοντέλου.
- Early_stopping: Εφόσον όλες οι υποθέσεις για το num-beams έχουν ολοκληρωθεί και έρχεται το EOS token σταματάει η περίληψη.

Έπειτα έχουμε την μεταβλητή **simplified_text** στην οποία κάνουμε αποκωδικοποίηση τα tokens μας που βρίσκονται στο **summary_ids** (δημιουργήθηκαν από το μοντέλο μας πιο πάνω) και του λέμε να ξεκινήσει να κάνει την αποκωδικοποίηση από το πρώτο token και να μην συμπεριλάβει ειδικούς χαρακτήρες (**skip_special_tokens=True).** Αφού αφαιρούμε και το "simplify:" με κενό (μερικές φορές το έβγαζε στην μετάφραση), και γυρνάμε σαν αποτέλεσμα της συνάρτησης το **simplified_text** μένει στο τέλος το except Exception που έχει μείνει για τα errors όπου σε περίπτωση που δεν μπορέσει να βγάλει αποτέλεσμα το μοντέλο μας απλά γυρνάει το πρωτότυπο κείμενο.

```python
def simplify_text_for_asl(text):
    try:
        text = "simplify: " + text # used to prepare model to summarize given input text
        input_ids = token.encode(text, return_tensors="pt", max_length=512, truncation=True)

        # used to generate output from our model via given input
        summary_ids = model.generate(
            input_ids,
            num_beams=6, # reduces risk to miss on hidden high probability from the word sequ
            no_repeat_ngram_size=2, # no sequence of >2 words is repeated, usefull for genera
            length_penalty=1.0, # affects sentence length, a greater value will generate smal
            min_length=0, # avoid error output for smaller sentences
            max_length=100, # will max out at 100 tokens long
            early_stopping=True #stop early once all beam hypotheses have produced an EOS tok
        )

        simplified_text = token.decode(summary_ids[0], skip_special_tokens=True) # decodes pr
        simplified_text = simplified_text.replace("simplify:", "").strip() # incase program i
        return simplified_text

    except Exception as e:
        print(f"Error: {e}")
        return text  # Return original text if simplification fails

```

Figure 64 simplify_text_for_asl(text) συνάρτηση.

Ακόμα μία συνάρτηση που υπάρχει στο αρχείο **mapping.py** είναι και η **map_text_to_asl_images(text,lang) (**Figure 65, Figure 66**)** η οποία κάνει ακριβώς αυτό που λέει, παίρνει το κείμενο που δίνει ο χρήστης ή το κείμενο σε περίληψη, το μετατρέπει σε κεφαλαία για την αποφυγή σφαλμάτων με τους τόνους ή κάποιο διαφοροποιημένο γράμμα που δεν υπάρχει στο αρχείο με τα νοηματικά σύμβολα και κάνει το **text = text.upper()** και επίσης έχουμε και μια λίστα **image_paths = []** στην οποία θα αποθηκεύσουμε τα αντίστοιχα νοηματικά μας σύμβολα. Μετά υπάρχει η **if statement** στην οποία ελέγχουμε τί γλώσσα επέλεγε ο χρήστης στην μεταβλητή **lang** και βάζει σαν διαδρομή το ανάλογο φάκελο στην μεταβλητή **directory**, στην περίπτωση λανθασμένης επιλογές βάζει ως προκαθορισμένη επιλογή την Αγγλική εκδοχή. Ακόμα όμως υπάρχει και μια **for loop** στην οποία ελέγχουμε για **letters** στο **text** μας και αν υπάρχει κάποιο σύμβολο που μας ενοχλεί/δεν μπορούμε να μεταφράσουμε το αντικαθιστούμε με το κενό **image_paths.append(None)** αλλιώς προχωράει και στο **image_name** βάζει το όνομα του γράμματος αντίστοιχο με αυτό στα νοηματικά σύμβολα (υπάρχει ήδη ένα preset f'Sign_Language_{**letter}.**png οπού αλλάζει το **letter** κάθε φορά) και μετά στο **image_path** προσθέτουμε την φωτογραφία με το αντίστοιχο όνομα στα νοηματικά σύμβολα, έτσι αν δεν υπάρχει βγάζουμε απλά ότι η φωτογραφία για το **letter** δεν βρέθηκε. Τέλος επιστρέφει την λίστα με τις φωτογραφίες.

```python
def map_text_to_asl_images(text, lang):
    text = text.upper()
    image_paths = []  # creates a list of image paths based on the given input

    if lang == 'en-US': # choose ASL if language is english
        directory = ASL_DIR
    elif lang == 'el-GR': # choose GSL if language is greek
        directory = GSL_DIR
    else:
        print("Invalid choice, setting ASL as default.") # defaults ASL cause it
        directory = ASL_DIR

```

Figure 65 Πρώτο μέρος της συνάρτησης map_text_to_asl_images(text,lang).

```
    for letter in text:
        if letter in [' ', "'", '"', ':', '[', ']', '.', ',', '-', '_','(',')']:
            # Treat spaces as None and skip adding an image path for them, used t
            image_paths.append(None)
            continue

        image_name = f"Sign_language_{letter}.png" # used to load image names
        image_path = os.path.join(directory, image_name) # get image path

        if os.path.exists(image_path): # searches image path
            image_paths.append(image_path)
        else:
            print(f"No ASL sign image found for '{letter}'")
            image_paths.append(None)  # Append None if the image is not found

    return image_paths

```

Figure 66 Δεύτερο μέρος της συνάρτησης map_text_to_asl_images(text,lang).

Η τελευταία συνάρτηση αφορά την προβολή των εικόνων μέσο ενός γραφικού περιβάλλοντος με την βοήθεια της βιβλιοθήκης του **Tkinter.** Όμως για να γίνει η προβολή των εικόνων πρέπει πρώτα να έχουμε την λίστα με τα νοηματικά σύμβολα, έτσι καλούμε την συνάρτηση **map_text_to_asl_images(text,lang) (**Figure 67Figure 68**)** μέσα στην συνάρτηση **display_images_sequentially(text, lang, delay, word_gap=1)** και βάζουμε το αποτέλεσμά της στην μεταβλητή **image_paths**, πριν όμως συνεχίσουμε έχουμε και έναν έλεγχο σε περίπτωση που η λίστα μας είναι άδεια να βγάλει το κατάλληλο μήνυμα. Έπειτα ξεκινάμε με την δημιουργία του παράθυρου βάζοντας σε μια μεταβλητή **window = tk.Tk()** και πάνω στην συγκεκριμένη μεταβλητή μπορούμε να ορίσουμε μερικές παραμέτρους από την βιβλιοθήκη **Tkinter** όπως **window.title("Sign Language viewer")** για τον τίτλο του παραθύρου της εφαρμογής μας και μετά να ορίσουμε και την διάσταση από το παράθυρο με **window.geometry("1920x1080").** Στη συνέχεια δημιουργώ μια μεταβλητή **canvas = tk.Canvas(window, width=150, height=150, bg="white")** στην οποία θα γίνεται η προβολή από τα νοηματικά σύμβολά μου και ορίζω το μέγεθος αλλά και το χρώμα από το φόντο, και μετά όσο αφορά την θέση της κεντρικής οθόνης προβολής για τα νοηματικά σύμβολα έχω την μεταβλητή **canvas.grid(row=0, column=0, sticky="snow", padx=5, pady=5)** που θα καθορίζει την θέση του κύριου "προβολέα" στο παράθυρό μου. Στην συνέχεια δημιουργώ την θέση στην οποία θα μένουν τα νοηματικά σύμβολα μετά την προβολή τους στο κύριο προβολέα και από κάτω τους θα έχουν και το αντίστοιχο γράμμα που αντιπροσωπεύουν, οπότε δημιουργούμε το **thumbnail_frame = tk.Frame(window)** και το βάζουμε σε κατάλληλη θέση στο παράθυρό μας με την εντολή **thumbnail_frame.grid(row=1, column=0, sticky="nsew "),** ακόμα όμως πρέπει να βάλουμε σε σωστή θέση για προβολή των εικόνων με τις παρακάτω εντολές **window.rowconfigure(0, weight=1), window.rowconfigure(1, weight=4)** και **window.columnconfigure(0, weight=1).** Μετά ακολουθεί η εντολή **thumbnail_canvas = tk.Canvas(thumbnail_frame, bg="lightgrey")** και η εντολή **thumbnail_canvas.pack(side="left", fill="both", expand=True)** με τις οποίες δημιουργούμε έναν καμβά στον υπολειπόμενο χώρο από εκεί που προβάλουμε τις νοηματικές εικόνες και στην συνέχεια της στοιχίζουμε κατάλληλα για να ξεκινάει η προβολή τους από τα αριστερά προς τα δεξιά,

```
72  def display_images_sequentially(text, lang, delay, word_gap=1):
73
74      # generate image_paths by getting input to according function
75      image_paths = map_text_to_asl_images(text, lang)
76
77      if not image_paths: # if image paths is empty or doesnt exist output er
78          print("No images to display. Check your text or language input.")
79          return
80
81      window = tk.Tk() # create a tkinter window to display ASL or GSL images
82      window.title("Sign Language Viewer") # window title
83      window.geometry("1920x1080") # windows geometry
84
85      canvas = tk.Canvas(window, width=150, height=150, bg="white") # create
86      canvas.grid(row=0, column=0, sticky="nsew", padx=5, pady=5) # position
87
88      thumbnail_frame = tk.Frame(window) # creates a frame widget which will
89      thumbnail_frame.grid(row=1, column=0, sticky="nsew") # places frame in
90
91      # Configure grid weights
92      window.rowconfigure(0, weight=1) # setup a small space for displaying m
93      window.rowconfigure(1, weight=4) # setupt the rest of the space for dis
94      window.columnconfigure(0, weight=1) # column to set images
95
96      # Create a canvas inside the thumbnail frame
97      thumbnail_canvas = tk.Canvas(thumbnail_frame, bg="lightgray") # used to
98      thumbnail_canvas.pack(side="left", fill="both", expand=True) # used to
99
```

Figure 67 Μέρος πρώτο της συνάρτησης display_images_sequentially(text, lang, delay, word_gap=1).

Συνεχίζουμε προσθέτοντάς μια μεταβλητή για να κάνουμε το **scrollbar** για να έχουμε παραπάνω χώρο όταν γεμίσει ο canvas του thumbnail για να δούμε όλα τα νοηματικά σύμβολα, οπότε κάνουμε **scrollbar = tk.scrollbar(thumbnail_frame, orient=”vertical”, command=thumbnail_canvas.yview)** που την προσθέτει στα δεξιά του παραθύρου και μετά **scrollbar.pack(side=”right”, fill=”y”)** για την συγκεκριμένη θέση. Πηγαίνοντας παρακάτω στο πρόγραμμα βάζουμε κάποιες παραπάνω εντολές για να γίνεται η ενημέρωση του scrollbar κατά την διάρκεια λειτουργίας του παραθύρου με τις παρακάτω εντολές, **thumbnail_canvas.configure(yscrollcommand=scrollbar.set)** μετά **thumbnail_canvas.bind(“configure”, lambda e: thumbnail_canvas.configure(scrollregion=thumbnail_canvas.bbox(“all”)),)** και προσθέτουμε μια εντολή **window.update_idletasks()** για να ανανεώνεται το παράθυρο της εφαρμογής και να μεγαλώνει η μπάρα όποτε χρειάζεται. Μετά δημιουργούμε ένα **inner_frame** στο οποίο θα υπάρχουν τα **thumbnails (**κρατάει τα νοηματικά σύμβολα που προβάλλονταν στην κύρια οθόνη) και έχουμε την μεταβλητή **inner_frame = tk.Frame(thumbnail_canvas, bg=”lightgray”)** και **thumbnail_canvas.create_window((0,0), window=inner_frame, anchor=”nw”)** και θέτουμε κάποιες μεταβλητές για το πόσες φωτογραφίες μπορούμε να έχουμε στο παράθυρό μας, **thumbnail_size = 30, max_thumbnail_per_row = 40, current_row = 0, current_column = 0** οπού ορίζουμε το μέγεθος των νοηματικών συμβόλων , πόσα σύμβολα θα χωράνε σε κάθε σειρά και από που θα ξεκινάει η προβολή σε σειρά και στήλη, και τέλος έχουμε μια μεταβλητή για να αποθηκεύουμε πρόχειρα νοηματικά σύμβολα σε cache list.

```
    # Add a scrollbar to the thumbnail canvas
    scrollbar = tk.Scrollbar(thumbnail_frame, orient="vertical", command=thumbnail_canvas.yview) #
    scrollbar.pack(side="right", fill="y") # scrollbar positions

    thumbnail_canvas.configure(yscrollcommand=scrollbar.set)
    thumbnail_canvas.bind(
        "<Configure>",
        lambda e: thumbnail_canvas.configure(scrollregion=thumbnail_canvas.bbox("all")),
    ) # update scrollbar based on the canvas content and when the canvas is resized the scrollbar

    window.update_idletasks() # used to update window for scrollbar

    # Create an inner frame to hold thumbnails
    inner_frame = tk.Frame(thumbnail_canvas, bg="lightgray")
    thumbnail_canvas.create_window((0, 0), window=inner_frame, anchor="nw")

    thumbnail_size = 30 # thumbnail size to get more thumbnails in a row
    max_thumbnails_per_row = 40 # how many thumbnails in a row
    current_row = 0 # starting row
    current_column = 0 # starting column

    image_refs = [] # cache images

```

Figure 68 Μέρος δεύτερο από την συνάρτηση, που δηλώνει μεταβλητές και φτιάχνει την διάταξη στο παράθυρο προβολής.

Έπειτα έχουμε μια συνάρτηση **load_thumbnail(path)(**Figure 69**)** η οποία είναι υπεύθυνη για την προβολή, οπού βάζουμε μια **try:** για να πιάσουμε τυχόν errors από λάθος μεταβλητές στην λίστα με τα thumbnails και έτσι με την μεταβλητή **thumbnail = Image.open(path).convert("RGBA").resize((thumbnail_size,thumbnail_size),Image.LANCZOS)**που κάνει μετατροπή την εικόνα και επιστρέφει στην **ImageTk.PhotoImage(thumbnail)** όπου επιστρέφει και τέλος πιάνει με το **try και except** τα errors.

```
    def load_thumbnail(path):
        try:
            thumbnail = Image.open(path).convert("RGBA").resize((thumbnail_size, thumbnail_size), Image.LANCZOS)
            return ImageTk.PhotoImage(thumbnail) # gives back image
        except Exception as e:
            print(f"Error loading thumbnail {path}: {e}")
            return None

```

Figure 69 συνάρτηση load_thumbnail(path) που φορτώνει τις εικόνες από το main display για προβολή στο thumbnail.

Τέλος έχουμε την συνάρτηση **display_image(i)(**Figure 70Figure 71**)** η οποία είναι υπεύθυνη για την προβολή όλων των νοηματικών συμβόλων, οπότε αρχικά θέτουμε δύο μεταβλητές **current_row** και **current_column** για να έχουμε τα δεδομένα μας για το πρόγραμμα. Στη συνέχεια έχουμε ένα **if statement** η οποία επεξεργάζεται μόνο έγκυρα νοηματικά σύμβολα. Στην επόμενη if statement διαχειριζόμαστε όταν βρίσκουμε κενό στην λίστα με τα νοηματικά σύμβολα στο **image_paths[i],** έτσι μόνο όταν βρίσκουμε το σύμβολο «κενό» θέλουμε να κάνουμε το σύμβολο του κενού οπότε εάν στην εμφωλευμένη **if statement** το **text[i]** περιέχει κενό σύμβολο δημιουργούμε το **gap_label** το οποίο αποτελείται από **gap_label = tk.Label(inner_frame, bg"lightgrey", width=thumbnail_size// 10, height=1)** και μετά **gap_label.grid(row=current_row, column=current_column, padx=5, pady=5)** και **current_column+=word_gap** το οποίο δημιουργεί το σύμβολο του κενού και βγάζει το κενό ανάλογα το **word_gap** που έχουμε ορίσει, και μετά κάνει **window.after(delay, display_image, i+1)** οπού μπαίνει ξανά στην **display_image(i)** με το επόμενο νοηματικό σύμβολο στην λίστα και τελειώνει η **if statement** επιστρέφοντας στη βασική συνάρτηση.

```
131     def display_image(i):
132         nonlocal current_row, current_column # used to get the curr_row and curr_column to maintain the
133 
134         if i >= len(image_paths): # only processes valid image_paths, stops execution once images are di
135             return
136 
137         # if we get none in the image_paths which indicates that we get space or blank images
138         if image_paths[i] is None:
139             if text[i] == " ":
140                 gap_label = tk.Label(inner_frame, bg="lightgray", width=thumbnail_size // 10, height=1)
141                 gap_label.grid(row=current_row, column=current_column, padx=5, pady=5) # place of the ga
142                 current_column += word_gap  # adds gap
143             window.after(delay, display_image, i + 1) # used to diplay images with a delay given from th
144             return
145 
```

Figure 70 Αρχή της display_image(I) που έχουμε για την προβολή του κενού.

Στη συνέχεια προσθέτουμε την εντολή **try:** ξανά για να τυπώσουμε την **main_image** που είναι οπουδήποτε νοηματικό σύμβολο έχουμε από την συνάρτηση **map_text_to_asl_images(text,lang)** και έχουμε **main_image = Image.open(image_paths[i].convert("RGBA")** για την εφαρμογή του σωστού format για την προβολή των νοηματικών συμβόλων, μετά στην μεταβλητή **resized_image = main_image.resize((150,150), Image.LANCZOS)** κάνουμε προσαρμογή των συμβόλων στα pixels που επιθυμούμε και ξεκινάμε κάνοντας **canvas.delete("all")** για τον καθαρισμό των προηγούμενων νοηματικών συμβόλων (ακόμα και αν εκτελείται για πρώτη φορά το πρόγραμμα μας, και ύστερα έχουμε **canvas.create_image(0,0,anchor=tk.NW, image=tk_image)** που δημιουργεί το σημείο που θα φιλοξενηθεί το νέο νοηματικό σύμβολο και προσθέτουμε ακόμα δύο εντολές που αποθηκεύουν προσωρινά τα νοηματικά σύμβολα που έχουν ήδη χρησιμοποιηθεί με τις παρακάτω **canvas.image = tk_image** και **image_refs.append(tk_image)** και τέλος έχουμε την except για να προβάλουμε τα errors. Στη συνέχεια εφόσον προβάλουμε τα πρώτα νοηματικά σύμβολα ήρθε η ώρα για την προβολή των **thumbnails** όπου ξεκινάμε με την μεταβλητή **thumb = load_thumbnail(image_paths[i])** που φορτώνει τις φωτογραφίες από το **main_display** και στη συνέχεια κάνουμε ένα **window.update_idletasks()** και έχουμε ένα **if statement** το οποίο ελέγχει τα errors στην προβολή νοηματικών συμβόλων όταν υπάρχει **None** και ξεκινάει δημιουργώντας μεταβλητές που θα προβάλουν τις **thumbnails, thumbnails_label = tk.Label(inner_frame, image=thumb, bg="lightgrey")** που δημιουργεί ένα widget στο inner frame για να προβάλει τα thumbnails , **thumbnail_label.grid(row=current_row, column=current_column, padx=5, pady=5)** που βάζει σε σωστή θέση το κάθε thumbnail και τέλος κάνει cache το νοηματικό σύμβολο για επόμενη χρήση με την εντολή **image_refs.append(thumb).** Εφόσον προβάλουμε τα thumbnails θέλουμε και κάτω από κάθε νοηματικό σύμβολο πρέπει να προβάλουμε και το γράμμα το καθενός. Ξεκινάμε με **letter = os.path.splitext(os.path.basename(image_paths[i]))[0].split("_")[-1]** όπου η συγκεκριμένη εντολή αναλαμβάνει να εξάγει το κείμενο για την κάθε εικόνα και μετά με τις δύο μεταβλητές θα φτιάξουμε το πώς θα είναι το γράμμα και σε ποια θέση θα βρίσκεται, **text_label = tk.Label(inner_frame, text=letter, bg+"lightgrey", font="times new roman", 12), fg="black")** και **text_label.grid(row=current_row+1, column=current_column, padx=5, pady=5)** και μετά κάνουμε **current_column+=1** που μεταφέρεται η θέση στην επόμενη στήλη ώστε να μην συμπίπτουν τα νοηματικά σύμβολα. Έτσι στο **if statement** που ακολουθεί βλέπουμε αν έχουμε φτάσει στο όριο των στηλών για την προβολή των νοηματικών συμβόλων έτσι ώστε να πάμε στην επόμενη σειρά. Έτσι έχουμε **if current_column>=max_thumbnails_per_row** τότε **current_column=0** και **current_row+=2.** Μετά ακολουθεί η **window.after(delay, display_images, i+1)** και **windows.update_idletasks()** που προβάλουμε τα επόμενα νοηματικά σύμβολα και

ανανεώνουμε το παράθυρο της εφαρμογής. Εφόσον τελείωσε η συνάρτηση της **display_images[i]** και εφόσον δεν έχουμε άλλες φωτογραφίες να προβάλουμε έχουμε την **window.bind("<Escape>", lambda event: window.destroy())** η οποία κλείνει το παράθυρο της **display_images_sequentially(text, lang, delay, word_gap=1),** και κάπως έτσι καλούμε την συνάρτηση **display_image(0)** που αρχίζει να προβάλει τις φωτογραφίες από την αρχή και η τελευταία εντολή **window.mainloop()** που κρατάει την εφαρμογή ανοιχτή μέχρι να πατήσουμε το **Escape button.**

```
    # start diplaying the main images
    try:
        main_image = Image.open(image_paths[i]).convert("RGBA") # properly format for transparency handling (
        resized_image = main_image.resize((150, 150), Image.LANCZOS) # riseze to 150x150 for display purposes
        tk_image = ImageTk.PhotoImage(resized_image) # convert image to tkinter compatible object to fit the
        canvas.delete("all") # delete previous main image to display the new one
        canvas.create_image(0, 0, anchor=tk.NW, image=tk_image) #creates a new image to be able to handle the
        canvas.image = tk_image  # Keep reference to avoid garbage collection
        image_refs.append(tk_image) # images are stored in mem, avoid erros related to garbage and img stay v

    except Exception as e:
        print(f"Error displaying main image: {e}")
        return

    # Display thumbnail
    thumb = load_thumbnail(image_paths[i]) # load images from image_paths for displaying in thumbnail
    window.update_idletasks() # load the window correctly
    if thumb: #prevents erros displaying none images
        thumbnail_label = tk.Label(inner_frame, image=thumb, bg="lightgray") # creates a widget in the inner
        thumbnail_label.grid(row=current_row, column=current_column, padx=5, pady=5) # places the thumbanil l
        image_refs.append(thumb) # cache the thumbnail image to refer in future

    # adds letters bellow thumbnails
    letter = os.path.splitext(os.path.basename(image_paths[i]))[0].split("_")[-1] # retrieves from each image
    text_label = tk.Label(inner_frame, text=letter, bg="lightgray", font=("Times new roman", 12), fg="black")
    text_label.grid(row=current_row + 1, column=current_column, padx=5, pady=5) # position inside the inner f

    current_column += 1 # goes to the next column so the new thumbnail wont overlap the last one
    if current_column >= max_thumbnails_per_row: # used to see if we reached the max langth of the row so we
        current_column = 0
        current_row += 2

    # the time we will display the next image, taking delay time and display image function
    window.after(delay, display_image, i + 1)

    window.update_idletasks() # load window correctly

 close the window to insert new input text in the program
indow.bind("<Escape>", lambda event: window.destroy())

isplay_image(0) # starts the image display process from the first image in the list
indow.mainloop() # keeps tkinter window open and intercative so the loops end only when escape is pressed
```

Figure 71 Τελευταίο μέρος του κώδικα από το αρχείο mapping.py και την συνάρτηση display_images_sequentially(text, lang, delay, word_gap=1).

4.3.3 Functions.py

Φτάσαμε και στο τελευταίο μας αρχείο το οποίο κατά κύριο λόγο απαρτίζεται από δύο βιβλιοθήκες.

pyttsx3

Η **pyttsx3** βιβλιοθήκη είναι υπεύθυνη για την ομιλία του υπολογιστή στον χρήστη με κείμενο που έχει ορίσει ο προγραμματιστής, κατά κύριο λόγο χρησιμοποιείται για καλύτερη προσβασιμότητα του χρήστη.

csv

Είναι η βιβλιοθήκη που έχει τις μεθόδους χειραγωγήσεις των αρχείων .csv .tsv etc. , για την εξαγωγή δεδομένων αλλά και αποθήκευση αυτών σε αυτά.

Στην συνέχεια έχουμε σε μια μεταβλητή **engine = pyttsx3.init()** που ορίζουμε το bot που θα μιλάει στον χρήστη και έπειτα έρχονται οι δύο λίστες **termination_phrases=[], switch_commands = []** που περιέχουν συγκεκριμένες λέξεις κλειδιά για την εκτέλεση συγκεκριμένων λειτουργιών.

```
1   import pyttsx3
2   import csv
3
4   # setup pyttsx3 for talking bot.
5   engine = pyttsx3.init()
6
7   # termination phrases for each language.
8   termination_phrases = {
9       'en-US': ["stop", "terminate", "end", "quit"],
10      'el-GR': ["σταμάτα", "τερματισμός", "τέλος", "εξοδος"]
11  }
12
13  # switch commands for user
14  switch_commands = {
15          'en-US': ["switch", "change","sweets"],
16          'el-GR': ["αλλαγή", "άλλαξε"]
17      }
18
```

Figure 72 Αρχή του κώδικα functions.py που περιέχει χρήσιμες συναρτήσεις που καλούνται στο main.py

Εφόσον προχωράμε στον κώδικα συναντάμε δύο συναρτήσεις που μπορούμε να πούμε ότι η μία δίνει σκοπό στην άλλη, έχουμε την **speak_language_prompt()** στην οποία έχουμε την μεταβλητή **engine** να πει στον χρήστη το συγκεκριμένο κείμενο που έχουμε βάλει → **“Choose English or Greek “** και η επόμενη συνάρτηση **select_language()** ουσιαστικά δίνει τις δύο επιλογές και το νούμερο με το οποίο ο χρήστης μπορεί να τις επιλέξει, επίσης υπάρχει και μια **while loop** για την επιλογή σωστού νούμερου, αλλιώς αν ο χρήστης δώσει γράμμα βάζει κατευθείαν την επιλογή των αγγλικών, το πρόγραμμά μας περιμένει νούμερο και έχουμε την **try:** στις οποίας την **except** εάν υπάρξει **ValueError** μας ξαναζητάει να δώσουμε το 1 ή το 2 μέχρι να δώσουμε ένα από τα δύο.

```python
# use pyttsx3 to vocalize the entry prompt.
def speak_language_prompt():
    engine.say("Choose English or Greek.")
    engine.runAndWait()

# function to select the language.
def select_language():
    print("Select the language you'll be speaking:")
    print("1. English")
    print("2. Greek")

    while True:
        try:
            language_choice = int(input("Enter the number corresponding to your language choice: "))
            languages = {
                1: 'en-US',
                2: 'el-GR'
            }
            # Return the language if it's valid, otherwise prompt the user again
            return languages.get(language_choice, 'en-US')  # Default to Greek if choice is invalid
        except ValueError:
            print("Invalid input. Please enter a valid number (1 or 2).")

```

Figure 73 συναρτήσεις speak_language_prompt() και select_language().

Έπειτα θα μιλήσουμε για τις επόμενες τέσσερις συναρτήσεις αλλά όχι τελευταίες που έμειναν και είναι η **speak_mode(), speak_input(), speak_termination_prompt()** και **check_termination_phrases()**, η πρώτη κατά σειρά έχει την μεταβλητή **engine** και μιλάει στον αναγνώστη την εξής φράση:**[("Do you want to summarize or just translate to ASL?")],** έπειτα η δεύτερη είναι παρόμοια με την πρώτη συνάρτηση και ρωτάει τον χρήστη την εξής φράση**:[("Do you want to type or speak?")]** και η Τρίτη παρόμοια και τελευταία συνάρτηση ρωτάει το παρακάτω:**[("Do you want to terminate the process? ")],** τέλος έχουμε την **check_termination_phrase()** που βάζουμε τις εκφράσεις μάς στα **phrases** μορφοποιούμε το εισαγόμενο κείμενο από τον χρήστη και μετά συγκρίνουμε αν συμπίπτει το κείμενο μας λέξη προς λέξη με τις εκφράσεις μας.

```python
# use pyttsx3 to vocalize the choosing mode summarize or translate
def speak_mode():
    engine.say("Do you want to summarize or just translate to ASL?")
    engine.runAndWait()

# use pyttsx3 to vocalize type or speak
def speak_input():
    engine.say("Do you want to type or speak?")
    engine.runAndWait()

# use pyttsx3 to vocalize the termination prompt
def speak_termination_prompt():
    engine.say("Do you want to terminate the process? ")
    engine.runAndWait()

# check for termination phrase .
def check_termination_phrase(text, language):
    phrases = termination_phrases.get(language, [])

    # normalize text, convert it to lowercase, strip any surrounding whitespace.
    normalized_text = text.lower().strip()

    # check for match in normalized_text.
    return normalized_text in phrases

```

Figure 74 υπόλοιπες συναρτήσεις speak_mode(), speak_input(), speak_termination_prompt() και check_termination_phrases().

Τέλος έμειναν οι τρείς συναρτήσεις **check_switch_command(), create_csv_file()** και **save_to_csv(),** αρχικά η συνάρτηση για την αλλαγή δουλεύει με τον ίδιο τρόπο όπως αυτή του τερματισμού που ανέφερα προηγουμένως, μετά έχουμε την συνάρτηση που δημιουργεί το αρχείο csv και θέτει την ώρα το κείμενο και τον ομιλητή και μετά έχουμε το **save_to_csv()** το οποίο αποθηκεύει την ώρα, το κείμενο και ποιος ομιλητής μιλάει/γράφει, σε περίπτωση που δεν υπάρχει το αρχείο να βγάλει κάποιο error και να συνεχίσει το πρόγραμμα.

```
67  # switch the person talking, mainly used to define who said what in the csv(Users must
68  def check_switch_command(text, language):
69
70      # check switch commands to terminate program
71      commands = switch_commands.get(language, [])
72
73      # normalize text, convert it to lowercase, strip any surrounding whitespace
74      normalized_text = text.lower().strip()
75
76      # check for match in normalized_text
77      return normalized_text in commands
78
79  # used the first time to create the csv file header for each language
80  def create_csv_file(filename):
81      try:
82          with open(filename, 'x', newline='', encoding='utf-8') as csvfile:
83              fieldnames = ["timestamp", "speech", "speaker"]
84              writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
85              writer.writeheader()
86      except FileExistsError:
87          pass
88
89  # in case there isnt a file and we try to save promp an error
90  def save_to_csv(filename, timestamp, text, simplified_text,  speaker):
91      try:
92          with open(filename, 'a', newline='', encoding='utf-8') as csvfile:
93              writer = csv.writer(csvfile)
94              writer.writerow([timestamp, text, simplified_text , speaker])
95      except FileNotFoundError:
96          print("File not found. Create the CSV file first.")
97
```

Figure 75 Τελευταίο μέρος του αρχείου functions.py και τελευταίες συναρτήσεις, check_switch_command(), create_csv_file() και save_to_csv().

Αυτά όσον αφορά την άρθρωση του κώδικα μου και τί κάνει η κάθε εντολή (ελπίζω να κάλυψα ένα μεγάλο μέρος αυτών) και για το κλείσιμο του κεφαλαίου θεωρώ ότι μια επίδειξή το τί μπορεί να κάνει τελικά το πρόγραμμα μου εάν κάποιος αναγνώστης δεν έχει καταλάβει μέχρι τώρα είναι ο καλύτερος τρόπος για να κλείσει αυτό το κεφάλαιο.

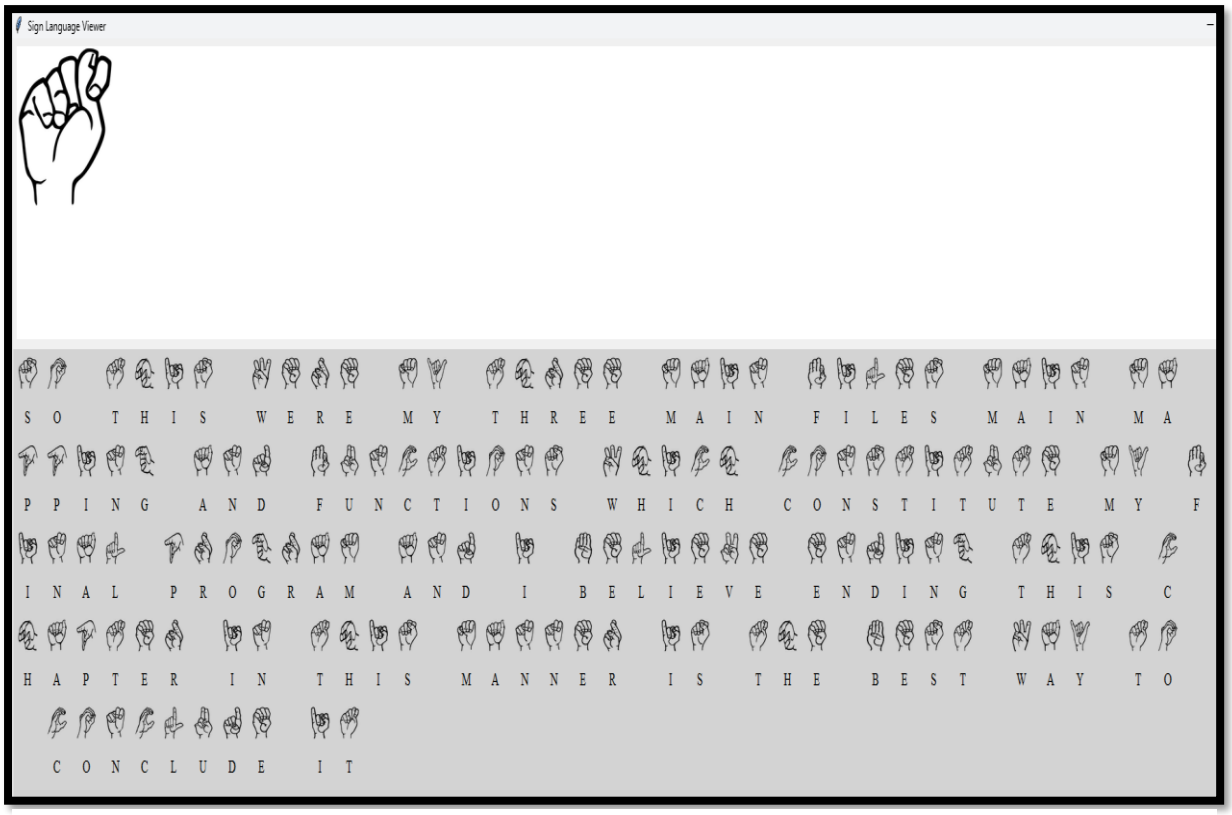

Figure 76 Παράδειγμα προβολής του προγράμματος μου για να κλείσει το κεφάλαιο.

4.4 Εκπαίδευση μοντέλων Bart και T5

Εφόσον είδαμε αναλυτικά το πρόγραμμα μας και τα τρία αρχεία που το απαρτίζουν, είναι ώρα να δούμε και τα δύο αρχεία που είναι υπεύθυνα για την εκπαίδευση των μοντέλων που συγκρίναμε. Με λίγα λόγια παίρνουμε τα συγκεκριμένα μοντέλα από το HuggingFace[50] χρησιμοποιώντας την βιβλιοθήκη transformers για να βελτιώσουμε τα μοντέλα μας πάνω σε συγκεκριμένες λειτουργίες (text-summarization).

4.4.1 Βελτίωση/Εκπαίδευση του μοντέλου BART για συγκεκριμένες λειτουργίες

Αρχικά έχουμε τις δύο βιβλιοθήκες που θα χρησιμοποιήσουμε για να εισάγουμε το μοντέλο μας από την βιβλιοθήκη **transformers** και μετά με την **dataset** κάνουμε εισαγωγή τα δεδομένα μας από τα csv, tsv αρχεία.

Transformers

Εισάγουμε της class **BartForConditionalGeneration** η οποία εισάγει το προ-εκπαιδευμένο μοντέλο **BART** , μετά έχουμε την class **BartTokenizer** για να μετατρέψουμε το κείμενο μας σε **tokens,** και τέλος η **Trainer** και **TrainingArguments** με τις οποίες απλοποιούμε τον τρόπο εκπαίδευσης και έχουμε διάφορες παραμέτρους για να τροποποιήσουμε για να εκπαιδευτεί το μοντέλο.

Datasets

Εισαγωγή δεδομένων από τα .csv .tsv αρχεία για την εκπαίδευση του μοντέλου.

Οπότε ξεκινάμε το πρόγραμμα εκπαίδευσης μας φορτώνουμε τα δεδομένα μας σε μια μεταβλητή **dataset** και στην συνέχεια χωρίζουμε την μεταβλητή μας σε δεδομένα που θα εκπαιδεύσουμε και σε αυτά που θα κάνουμε την δοκιμή επάνω με **dataset['train'].train_test_split(test_size=0.1).** Έπειτα έχουμε τις δύο μεταβλητές **token** και **model** φορτώνουμε το μοντέλο μας από την βιβλιοθήκη και την τροποποίηση για το κείμενο σε **tokens.** Επίσης μπορούμε να δούμε και ποιο μοντέλο χρησιμοποιούμε από τη μεταβλητή **model=BartForConditionalGeneration.from_pretrained('facebook/bart-base').** Στη συνέχεια έχουμε την συνάρτηση **preprocess_function(examples):** που κάνουμε προ επεξεργασία τα δεδομένα μας για να τα δεχθεί το μοντέλο μας στην συνέχεια και να «καταλαβαίνει» πάνω σε τι εκπαιδεύεται. Επίσης το μοντέλο **BART** κάνει ξεχωριστά την διαδικασία tokenization για input και expected output, δηλαδή η εισαγωγή γίνεται κανονικά tokenized ενώ το target που είναι η έξοδος από το dataset και γίνεται tokenized χρησιμοποιώντας την συνάρτηση **as_target_tokenizer():** για την χρήση στον αποκωδικοποιητή [51]. Και μετά έχουμε την μεταβλητή **tokenized_datasets = dataset.map(preprocess_function, batched=True)** στην οποία αποθηκεύουμε το κείμενο.

```python
from transformers import BartForConditionalGeneration, BartTokenizer, Trainer, TrainingArguments
from datasets import load_dataset

# loads my tsv(tab seperated values.)
dataset = load_dataset('csv', data_files=r'C:\Users\kostas\Desktop\thesis\dataset\Data-huggingfac

# uses 90% of my dataset as training data and the rest 10% as testing data.
dataset = dataset['train'].train_test_split(test_size=0.1)

# load the token and model used.
token = BartTokenizer.from_pretrained('facebook/bart-base')
model = BartForConditionalGeneration.from_pretrained('facebook/bart-base')

# preprocess the dataset
def preprocess_function(examples):
    inputs = examples['Normal']
    targets = examples['Simple']
    model_inputs = token(inputs, max_length=512, truncation=True, padding='max_length')# max 512

    # preparing the data from dataset by converting text to tokens to be efficiently processed by
    with token.as_target_tokenizer():
        labels = token(targets, max_length=512, truncation=True, padding='max_length')

    model_inputs['labels'] = labels['input_ids']
    return model_inputs

# use the preprocess function to the dataset and prepare the model for trainning(transforms the e
tokenized_datasets = dataset.map(preprocess_function, batched=True)

```

Figure 77 Αρχή του αρχείου BART_Train.py για την βελτίωση του εκπαιδευμένου μοντέλου BART.

Στη συνέχεια πρέπει να ορίσουμε τις παραμέτρους για την εκπαίδευση στην μεταβλητή **training_args** που θα αποθηκεύσουμε τα αποτελέσματα από την class **TrainingArguments(),** η οποία αποτελείται από **per_device_train_batch_size=8, per_device_eval_batch_size=8** μεταβλητές οι οποίες αναφέρουν πόσα παραδείγματα θα επεξεργάζεται το μοντέλο μας ταυτόχρονα (παίζει ρόλο πόση μνήμη έχει η κάρτα γραφικών μας), μετά έχουμε **output_dir=r"C:\Users\kostas\Desktop\thesis\model\BART"** που αποθηκεύουμε το μοντέλο μας και **overwrite_output_dir=True** που σβήνει το παλαιότερο μοντέλο εάν υπάρχει ήδη ο φάκελος. Στην συνέχεια έχουμε την **num_train_epochs=4** που δίνουμε πόσες φορές θέλουμε να εκπαιδεύσουμε από το dataset**.** Έπειτα βάζουμε και μεταβλητές για να μην αποθηκεύει τα **logs** να μην υπάρχει κάποιο **checkpoint** ούτε evaluation , **logging_dir=None, logging_strategy="no", save_strategy="no", evaluation_strategy="no"**. Τέλος έχουμε τις μεταβλητές για το **Learning_rate=5e-5** για τον **optimizer** μετά το **weight_decay=0.01** για να αποφύγουμε το overfitting και τέλος κάποιες μεταβλητές για την χρήση του επεξεργαστή και κάρτας γραφικών σε συνδυασμό για την καλύτερη χρήση του hardware και την πιο γρήγορη εκπαίδευση.

```
# define training arguments for my model(based on my computer power and having an efficient tr
training_args = TrainingArguments(
    per_device_train_batch_size=8,  # Adjust based on your GPU VRAM.
    per_device_eval_batch_size=8,
    output_dir=r"C:\Users\kostas\Desktop\thesis\model\BART",  # Directory to save the model.
    overwrite_output_dir=True,  # Overwrite the content of the output directory if it exists.
    num_train_epochs=4,  # Number of training epochs.
    logging_dir=None,  # Disable logging by setting to None.
    logging_strategy="no",  # Disable logging.
    save_strategy="no",  # Disable checkpoint saving.
    evaluation_strategy="no",  # Disable evaluation during training.
    learning_rate=5e-5,  # Learning rate for the optimizer.
    weight_decay=0.01,  # Weight decay to prevent overfitting.
    warmup_steps=1000,  # Number of warmup steps for learning rate scheduler.
    fp16=True,  # Enable mixed precision training for compatible GPUs.
    report_to=[],  # Disable reporting to any external services.
)

```

Figure 78 training_args για να περάσουμε τις παραμέτρους στην εκπαίδευση του μοντέλου.

Εφόσον θέσουμε και τις παραμέτρους για το μοντέλο μας ξεκινάμε να παίρναμε στην μεταβλητή **trainer** ην διαδρομή για το μοντέλο, την μεταβλητή με τις παραμέτρους, το **eval_dataset** και **train_dataset** και τον **tokenizer.** Τέλος ξεκινάμε την εκπαίδευση με **trainer.train()** μετά κάνουμε αποθήκευση το μοντέλο μας και τα tokens και προβάλουμε τα αποτελέσματα στην μεταβλητή **evaluation_results = trainer.evaluate().**

```
# initialize the trainer( which model is used, training config, train d
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets['train'],
    eval_dataset=tokenized_datasets['test'],
    tokenizer=token
)

# train the model(Bart)
trainer.train()

# save the model
model.save_pretrained(r"C:\Users\kostas\Desktop\thesis\model-BART")
token.save_pretrained(r"C:\Users\kostas\Desktop\thesis\model-BART")

# evaluate the model based on the dataset.
evaluation_results = trainer.evaluate()
print("Evaluation results:", evaluation_results)

```

Figure 79 Ορισμός του trainer και μετά εκπαίδευση μοντέλου και αποθήκευση μοντέλου, tokens και προβολή αποτελεσμάτων.

Το μοντέλο μας είναι έτοιμα να ξεκινήσει την εκπαίδευση πάνω στα δεδομένα που έχουμε φορτώσει. Ξεκινάμε εφόσον έχουμε ενεργοποιήσει το περιβάλλον μας με την εντολή **.\venv\scripts\activate** και ξεκινάμε με **py.\BART_Train.py** και περιμένουμε να τελειώσει το μοντέλο μας. Όπως θα παρατηρήσετε η συγκεκριμένη εργασία είναι αρκετά χρονοβόρα και ακριβή και προϋποθέτει ότι ο προγραμματιστής διαθέτει ισχυρή υπολογιστική δύναμη. Μια άλλη ενναλακτική είναι να βρούμε το μοντέλο μας ήδη προ εκπαιδευμένο σε συγκεκριμένα δεδομένα που θέλουμε στο HuggingFace [50].

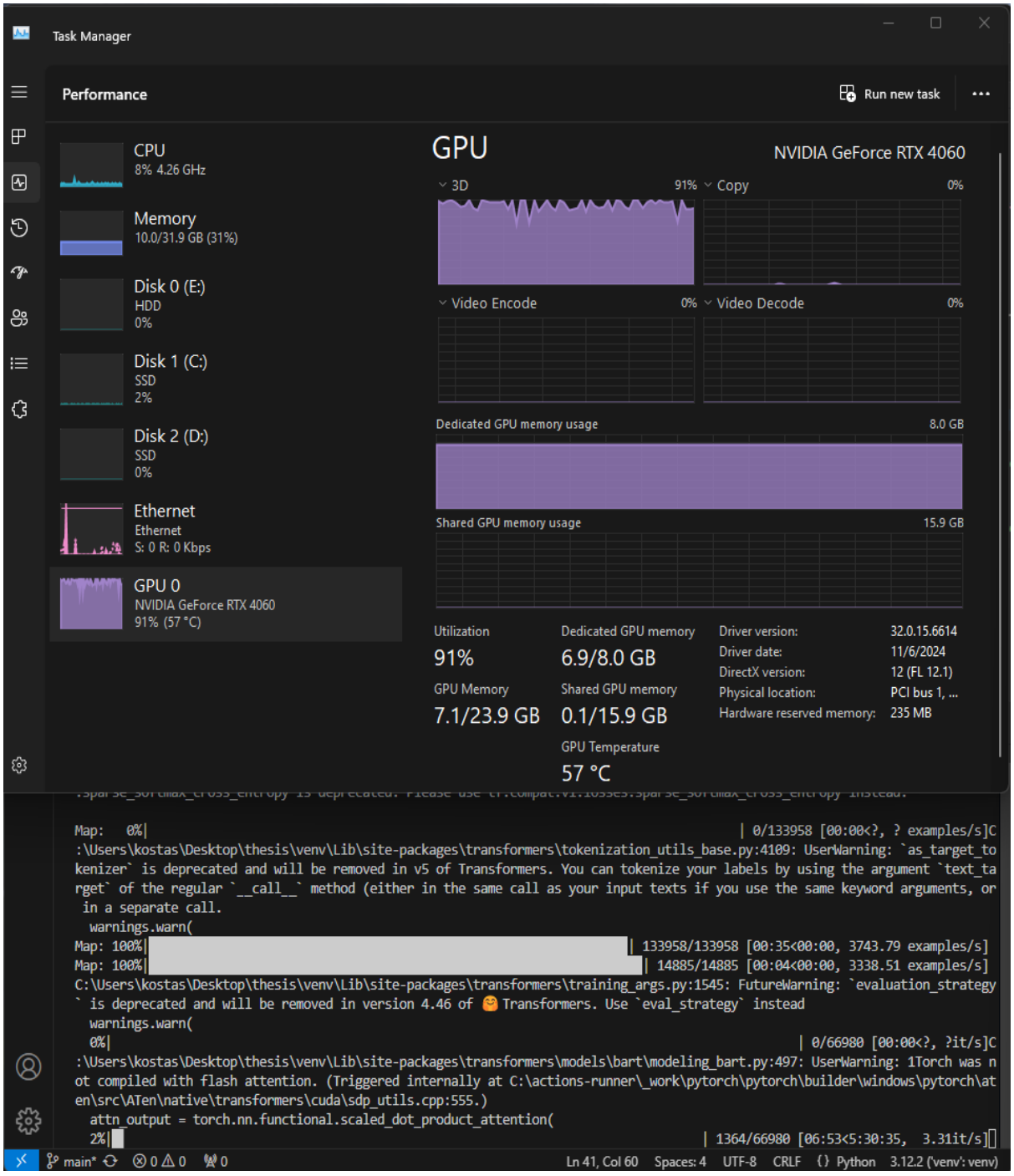

Figure 80 Φωτογραφία με την χρήση της κάρτας γραφικών στο 100% και κάτω δεξιά οι ώρες που χρειάζονται για την εκπαίδευση του μοντέλου.

4.4.2 Βελτίωση/Εκπαίδευση του μοντέλου T5 για συγκεκριμένες λειτουργίες

Αρχικά έχουμε τις δύο βιβλιοθήκες που θα χρησιμοποιήσουμε για να εισάγουμε το μοντέλο μας από την βιβλιοθήκη **transformers** και μετά με την **dataset** κάνουμε εισαγωγή τα δεδομένα μας από τα csv, tsv αρχεία.

Transformers

Εισάγουμε της class **T5ForConditionalGeneration.from_pretrained('t5-small')** η οποία εισάγει το προ-εκπαιδευμένο μοντέλο **T5** , μετά έχουμε την class **T5Tokenizer** για να μετατρέψουμε το κείμενο μας σε **tokens,** και τέλος η **Trainer** και **TrainingArguments** με τις οποίες απλοποιούμε τον τρόπο εκπαίδευσης και έχουμε διάφορες παραμέτρους για να τροποποιήσουμε για να εκπαιδευτεί το μοντέλο.

Datasets

Εισαγωγή δεδομένων από τα .csv .tsv αρχεία για την εκπαίδευση του μοντέλου.

Οπότε ξεκινάμε το πρόγραμμα εκπαίδευσης μας φορτώνουμε τα δεδομένα μας σε μια μεταβλητή **dataset** και στην συνέχεια χωρίζουμε την μεταβλητή μας σε δεδομένα που θα εκπαιδεύσουμε και σε αυτά που θα κάνουμε την δοκιμή επάνω με **dataset['train'].train_test_split(test_size=0.1).** Έπειτα έχουμε τις δύο μεταβλητές **token** και **model** φορτώνουμε το μοντέλο μας από την βιβλιοθήκη και την τροποποίηση για το κείμενο σε **tokens.** Επίσης μπορούμε να δούμε και ποιο μοντέλο χρησιμοποιούμε από τη μεταβλητή **model=T5ForConditionalGeneration.from_pretrained('t5-small').** Στη συνέχεια έχουμε την συνάρτηση **preprocess_function(examples):** που κάνουμε προ επεξεργασία τα δεδομένα μας για να τα δεχθεί το μοντέλο μας στην συνέχεια και να «καταλαβαίνει» πάνω σε τι εκπαιδεύεται, στην συγκεκριμένη περίπτωση είσοδος και έξοδος γίνεται με την ίδια λογική (ίδιο tokenizer) [26]. Και μετά έχουμε την μεταβλητή **tokenized_datasets = dataset.map(preprocess_function, batched=True,remove_columns=dataset['train'].column_names)** στην οποία αποθηκεύουμε το κείμενο.

```python
from transformers import T5ForConditionalGeneration, T5Tokenizer, Trainer, TrainingArguments
from datasets import load_dataset

# Load the dataset (TSV format with tab delimiter).
dataset = load_dataset(
    'csv',  # Use 'csv' as the loader supports custom delimiters.
    data_files=r"C:\Users\kostas\Desktop\thesis\dataset\Data-huggingface\wiki.full.aner.ori.train.95.tsv",
    delimiter='\t'
)

# Split dataset into training and testing sets (90% train, 10% test).
dataset = dataset['train'].train_test_split(test_size=0.1)

# Load tokenizer and model.
tokenizer = T5Tokenizer.from_pretrained('t5-small')
model = T5ForConditionalGeneration.from_pretrained('t5-small')

# Preprocess the dataset.
def preprocess_function(examples):
    inputs = examples['Normal']
    targets = examples['Simple']

    # Tokenize input and target text.
    model_inputs = tokenizer(inputs, max_length=512, truncation=True, padding='max_length')
    labels = tokenizer(targets, max_length=512, truncation=True, padding='max_length')

    # Add labels to the model inputs.
    model_inputs['labels'] = labels['input_ids']
    return model_inputs

# Apply preprocessing to the dataset.
tokenized_datasets = dataset.map(preprocess_function, batched=True, remove_columns=dataset['train'].column_nar
```

Figure 81 Αρχή του αρχείου T5_Train.py για την βελτίωση του εκπαιδευμένου μοντέλου T5.

Στη συνέχεια όπως και στο μοντέλο **BART** πρέπει να θέσουμε τα **training_arguments** και μετά τον **trainer** που θα κάνουμε **train()** για να πάρουμε το τελικό μας μοντέλο αλλά και που θα το αποθηκεύσουμε. Συνοψίζοντας μπορούμε να δούμε ότι και οι δύο κώδικες για το **BART, T5** είναι σχεδόν πανομοιότυποι με μόνες διαφορές πώς διαχειρίζονται το tokenization στο input και Output.

```
# Define training arguments.
training_args = TrainingArguments(
    per_device_train_batch_size=8,  # Adjust based on your GPU VRAM.
    per_device_eval_batch_size=8,
    output_dir=r"C:\Users\kostas\Desktop\thesis\model\model-T5",  # Directory to save the model.
    overwrite_output_dir=True,  # Overwrite the content of the output directory if it exists.
    num_train_epochs=4,  # Number of training epochs.
    logging_dir=None,  # Disable logging by setting to None.
    logging_strategy="no",  # Disable logging.
    save_strategy="no",  # Disable checkpoint saving.
    evaluation_strategy="no",  # Disable evaluation during training.
    learning_rate=5e-5,  # Learning rate for the optimizer.
    weight_decay=0.01,  # Weight decay to prevent overfitting.
    warmup_steps=1000,  # Number of warmup steps for learning rate scheduler.
    fp16=True,  # Enable mixed precision training for compatible GPUs.
    report_to=[],  # Disable reporting to any external services.
)

# Initialize the Trainer.
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets['train'],
    eval_dataset=tokenized_datasets['test'],
    tokenizer=tokenizer,
)

# Train the model.
trainer.train()

# Save the trained model and tokenizer.
model.save_pretrained(r"C:\Users\kostas\Desktop\thesis\model-T5")
tokenizer.save_pretrained(r"C:\Users\kostas\thesis\model-T5")

# Evaluate the model.
evaluation_results = trainer.evaluate()
print("Evaluation results:", evaluation_results)

```

Figure 82 Τέλος του κώδικα T5-Train.py και εκτύπωση αποτελεσμάτων και αποθήκευση του μοντέλου.

Το μοντέλο μας είναι έτοιμα να ξεκινήσει την εκπαίδευση πάνω στα δεδομένα που έχουμε φορτώσει. Ξεκινάμε εφόσον έχουμε ενεργοποιήσει το περιβάλλον μας με την εντολή **.\venv\scripts\activate** και ξεκινάμε με **py.\T5_Train.py** και περιμένουμε να τελειώσει το μοντέλο μας. Όπως θα παρατηρήσετε η συγκεκριμένη εργασία είναι αρκετά χρονοβόρα και

ακριβή και προϋποθέτει ότι ο προγραμματιστής διαθέτει ισχυρή υπολογιστική δύναμη. Μια άλλη ενναλακτική είναι να βρούμε το μοντέλο μας ήδη προ εκπαιδευμένο σε συγκεκριμένα δεδομένα που θέλουμε στο HuggingFace [50] και τέλος είναι η φωτογραφία που δείχνει πώς χρησιμοποιεί το hardware του συστήματός μας (Figure 80).

ΚΕΦΑΛΑΙΟ 5 Πειραματική αποτίμηση κώδικα

Φτάνοντας στο τέλος του paper ήρθε η ώρα να δούμε και πώς δουλεύουν οι λειτουργίες του προγράμματος μου αλλά και πως ανταποκρίνεται σε αυτές ο κώδικάς μου και ποιες είναι οι δυνατότητες του.

5.1 Έλεγχος αποθήκευσης στο csv αρχείο

Αρχικά έχουμε την δυνατότητα αποθήκευσης της εισαγωγής του χρήστη αλλά την έξοδο σε απλοποιημένη μορφή από το πρόγραμμα. Ουσιαστικά επικεντρωνόμαστε στην σωστή εισαγωγή των εισερχόμενων δεδομένων (π.χ. πρωτότυπο κείμενο, απλοποιημένο κείμενο) και στην διατήρηση της ακεραιότητας των δεδομένων κατά την αποθήκευση.

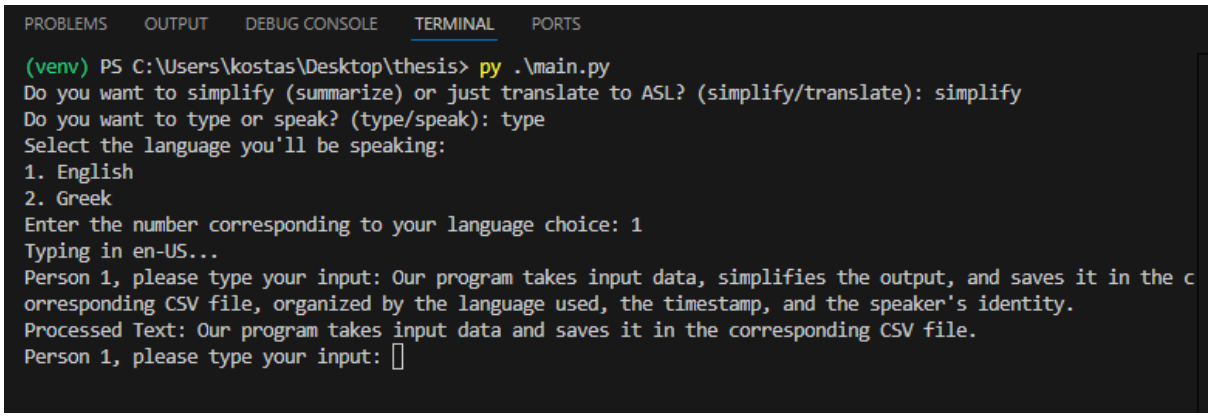

Figure 83 Εισαγωγή δεδομένων για την αποθήκευση στο csv.

Όπως θα δούμε και στην επόμενη φωτογραφία έχουμε ένα δείγμα από το παραγόμενο csv που δείχνει τι ώρα, τι είσοδο είχαμε, τί έξοδο και ποιος χρήστης δίνει δεδομένα.

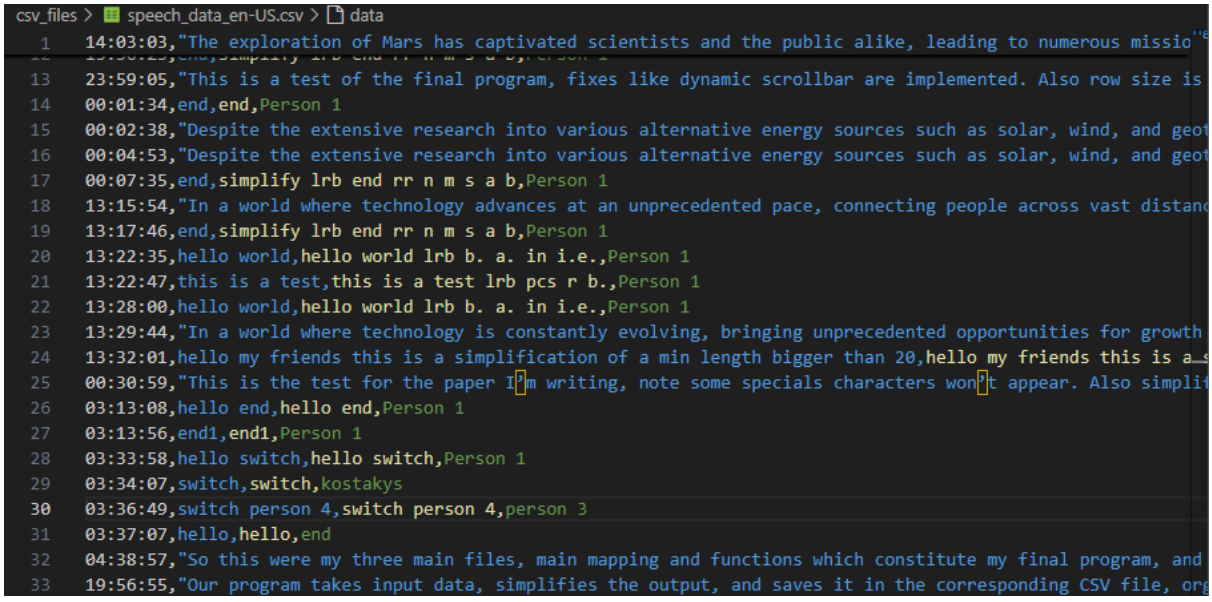

Figure 84 Παραγόμενο csv στα αγγλικά.

Παρόμοια συμπεριφορά έχει και το πρόγραμμα μας όταν ο χρήστης πληκτρολογεί στα Ελληνικά, η μόνη διαφορά στην συγκεκριμένη περίπτωση είναι ότι η έξοδος δεν διαφοροποιείται από την είσοδο διότι δεν έχει γίνει προ εκπαίδευση σε Ελληνικό dataset. Η

αποθήκευση δεδομένων έχει ως απώτερο σκοπό κάποια μελλοντική εκπαίδευση στα δεδομένα με αγγλικό χαρακτήρα, με λίγα λόγια να γίνει εισαγωγή του csv σε ένα από τα μοντέλα μας με τις ετικέτες **[Normal] →[Simple]** για να αυξήσουμε την αποδοτικότητα του μοντέλου. Τέλος από την Ελληνική εισαγωγή θα μπορούσε να δημιουργηθεί μελλοντικά ένα dataset με την απλή εισαγωγή που έχουμε κάποια περίληψη από τον χρήστη.

5.2 Δοκιμές speech_recognition σε δύο γλώσσες

Στο συγκεκριμένο υποκεφάλαιο θα δούμε την ακρίβεια της βιβλιοθήκης **speech_recognition** σε Ελληνικά αλλά και αγγλικά. Η δοκιμή θα γίνει με χρήση προτάσεων με απλή και σύνθετη δομή και η καταγραφή σε ήσυχο περιβάλλον. Παρακάτω οι 10 προτάσεις που έδωσα σαν εισαγωγή και στην συνέχεια τι έκανε αναγνώριση το πρόγραμμα.

1. "The weather today is sunny with a high of 25 degrees."
2. "I would like to order a cup of coffee and a sandwich."
3. "Learning sign language can bridge communication gaps."
4. "The cat jumped over the fence to chase the bird."
5. "Can you tell me the time and the location of the meeting?"
6. "Technology is advancing rapidly, transforming our daily lives."
7. "She visited the library to study for her final exams."
8. "A picture is worth a thousand words."
9. "Please switch off the lights before leaving the room."
10. "The bus arrives at 8:30 in the morning."

Και στην παρακάτω φωτογραφία βλέπουμε τι κατέγραψε το πρόγραμμα και τι αποθήκευσε.

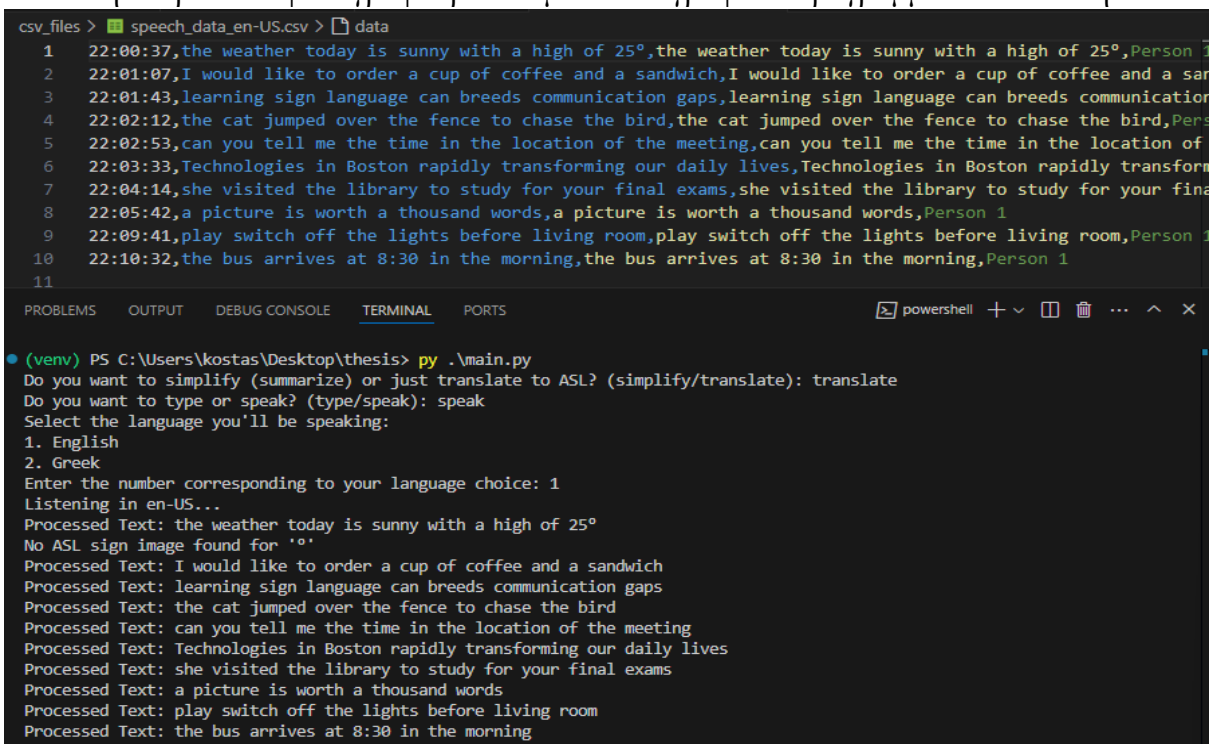

Figure 85 Αναγνώριση φωνής και αποθήκευση στο en-US.csv.

Στην συνέχεια της δοκιμής έχουμε την ίδια λειτουργία αλλά με εισαγωγή στα Ελληνικά από τον χρήστη. Η δοκιμή είναι πάνω στις 10 ίδιες προτάσεις αλλά μεταφρασμένες στα Ελληνικά, το πείραμα γίνεται πάλι σε ήσυχο περιβάλλον χωρίς παρεμβολές.

1. "Ο καιρός σήμερα είναι ηλιόλουστος με μέγιστη θερμοκρασία 25 βαθμούς."
2. "Θα ήθελα να παραγγείλω ένα φλιτζάνι καφέ και ένα σάντουιτς."
3. "Η εκμάθηση της νοηματικής γλώσσας γεφυρώνει τα κενά στην επικοινωνία."
4. "Η γάτα πήδηξε πάνω από τον φράχτη για να κυνηγήσει το πουλί."
5. "Μπορείς να μου πεις την ώρα και την τοποθεσία της συνάντησης;"
6. "Η τεχνολογία προοδεύει γρήγορα, μεταμορφώνοντας την καθημερινότητά μας."
7. "Επισκέφθηκε τη βιβλιοθήκη για να διαβάσει για τις τελικές εξετάσεις της."
8. "Μια εικόνα αξίζει όσο χίλιες λέξεις."
9. "Παρακαλώ σβήστε τα φώτα πριν φύγετε από το δωμάτιο."
10. "Το λεωφορείο φτάνει στις 8:30 το πρωί."

Και στην παρακάτω φωτογραφία βλέπουμε τι κατέγραψε το πρόγραμμα και τι αποθήκευσε. Για να λειτουργήσει σωστά η αναγνώριση φωνής εκτός από το μικρόφωνο που χρειάζεται ο χρήστης, η καλή σύνδεση στο ιντερνέτ είναι απαραίτητη για την σωστή λειτουργία του προγράμματος διότι το πρόγραμμα σταματάει την καταγραφή.

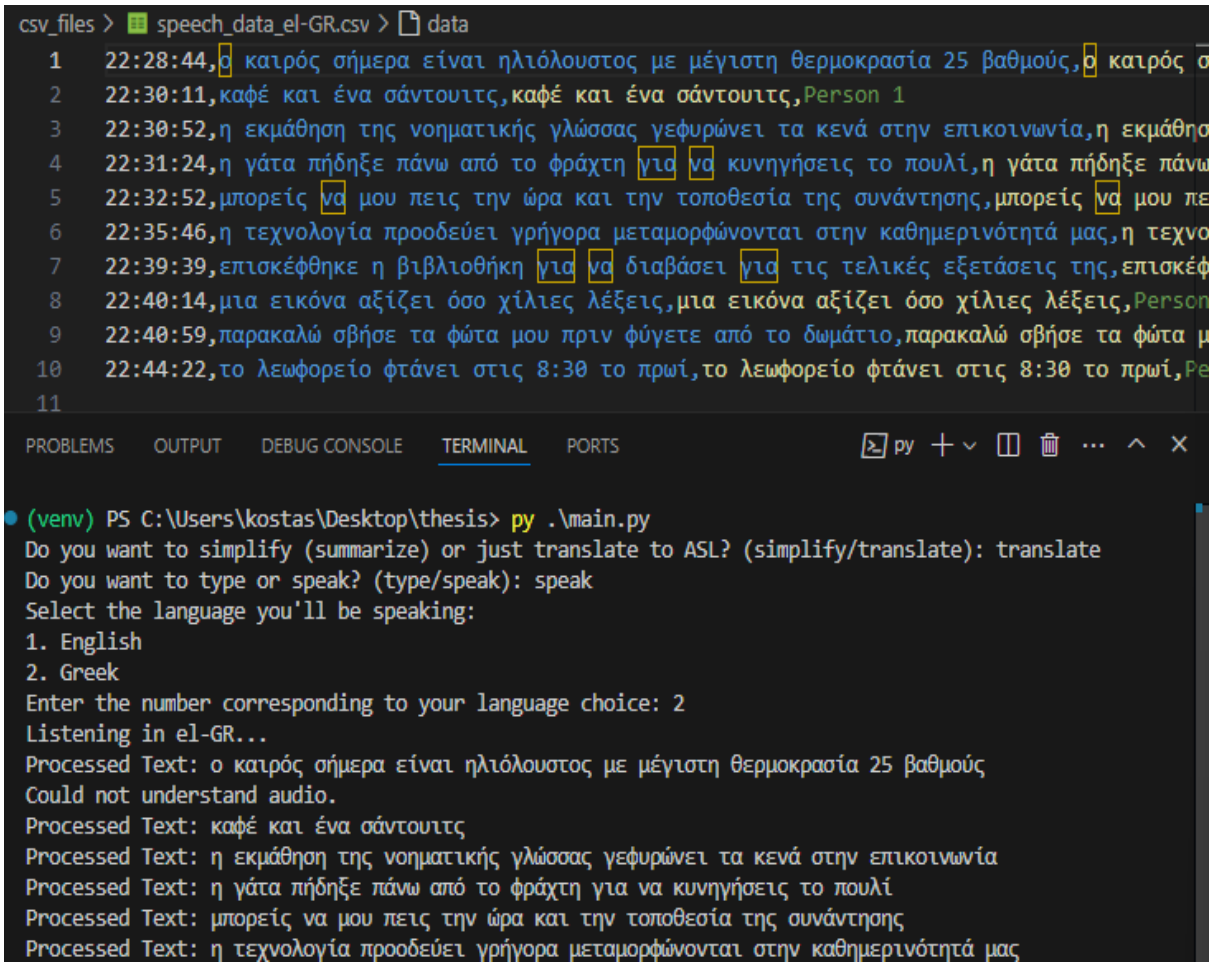

Figure 86 Πρόγραμμα πριν σταματήσει να λειτουργεί λόγο κακής σύνδεσης ίντερνετ.

Και τώρα βλέπουμε την συνέχεια του προγράμματος αφού τερματίστηκε άδοξα την πρώτη φορά η εκτέλεση λόγο κακής σύνδεσης στο ίντερνετ.

```
TimeoutError: [WinError 10060] A connection attempt failed because the connected party did not proper
ly respond after a period of time, or established connection failed because connected host has failed
 to respond
(venv) PS C:\Users\kostas\Desktop\thesis> py .\main.py
Do you want to simplify (summarize) or just translate to ASL? (simplify/translate): translate
Do you want to type or speak? (type/speak): speak
Select the language you'll be speaking:
1. English
2. Greek
Enter the number corresponding to your language choice: 2
Listening in el-GR...
Processed Text: επισκέφθηκε η βιβλιοθήκη για να διαβάσει για τις τελικές εξετάσεις της
Processed Text: μια εικόνα αξίζει όσο χίλιες λέξεις
Processed Text: παρακαλώ σβήσε τα φώτα μου πριν φύγετε από το δωμάτιο
Processed Text: το λεωφορείο φτάνει στις 8:30 το πρωί
```

Figure 87 συνέχεια του προγράμματος μετά το error λόγο σύνδεσης στο ίντερνετ.

Εφόσον έγινε η δοκιμή με τίς απλές προτάσεις ήρθε η σειρά για την λειτουργία simplify που έχουμε στην αγγλική γλώσσα με την αναγνώριση φωνής. Παρακάτω είναι 3 μεγάλες προτάσεις που θα έχουμε σαν είσοδο του προγράμματος για να δούμε την έξοδο μαζί με την περίληψη.

1. **Αρχική:**"Despite the numerous challenges faced by the scientific community in conducting groundbreaking research on renewable energy sources, their tireless efforts have led to significant advancements that are paving the way for a sustainable future."
2. **Αρχική:** "The intricate relationship between economic policies, social dynamics, and environmental sustainability remains a topic of intense debate among scholars, policymakers, and activists, with no clear consensus on the best path forward."
3. **Αρχική:** "While advancements in artificial intelligence have undoubtedly brought about transformative changes in industries ranging from healthcare to finance, concerns regarding ethical considerations, data privacy, and potential job displacement continue to be at the forefront of public discourse."

Και παρακάτω είναι η έξοδος που είχαμε από το μοντέλο μας

1. **Τελική:** "despite the numerous challenges faced by the scientific community in conducting groundbreaking research to renewable energy sources the entire efforts have led to significant advancements that are Paving the way for sustainable future. "
2. **Τελική:** "Enrique relationship between economic policies social dynamics and environmental sustainability Remains the topic of intense debate among Scholars policy makers and activists with nuclear consensus on the best path forward"
3. **Τελική:** "while advancements in artificial intelligence have brought about transformative changes in Industries raising from Healthcare to finance concerns regarding ethical considerations that the privacy and potential and soft displacement continue to be Forefront of public discourse. "

Όπως φαίνεται η περίληψη του κειμένου τα δεδομένα που έχουμε χρησιμοποιήσει (143.589 γραμμές με παραδείγματα) δεν ήταν αρκετά για να γίνει σωστή και πιο ακριβείς περίληψη. Μπορούμε να διακρίνουμε μια μικρή συντόμευση του κειμένου παρά τα ανεπαρκή δεδομένα. Τέλος εάν υπήρχε παραπάνω υπολογιστική δύναμη θα μπορούσαμε να εκπαιδεύσουμε παραπάνω το μοντέλο μας.

5.3 Έλεγχος λειτουργίας switch/stop

Η συγκεκριμένη λειτουργία αφορά στον σταματημό του προγράμματος από τον χρήστη αλλά και στην αλλαγή του χρήστη που κάνει εισαγωγή τα δεδομένα στο csv. Με λίγα λόγια στο αρχείο functions.py υπάρχουν δύο λίστες με λέξεις κλειδιά οι οποίες ενεργοποιούν τις παραπάνω λειτουργίες μόνο όταν υπάρχουν μόνες τους οι λέξεις, διαφορετικά όταν βρίσκονται μέσα σε προτάσεις δεν λειτουργούν.

```
7   # termination phrases for each language.
8   termination_phrases = {
9       'en-US': ["stop", "terminate", "end", "quit"],
10      'el-GR': ["σταμάτα", "τερματισμός", "τέλος", "εξοδος"]
11  }
12
13  # switch commands for user
14  switch_commands = {
15          'en-US': ["switch", "change","sweets"],
16          'el-GR': ["αλλαγή", "άλλαξε"]
17      }
```

Figure 88 λίστες με λέξεις κλειδιά για Ελληνικά και Αγγλικά.

Και η παρακάτω φωτογραφία δείχνει ότι οι λέξεις τερματισμού και αλλαγής δουλεύουν μόνο όταν δίνονται μόνες τους σαν εισαγωγή χωρίς κάποια συνοδευτική πρόταση.

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

Do you want to simplify (summarize) or just translate to ASL? (simplify/translate): translate
Do you want to type or speak? (type/speak): type
Select the language you'll be speaking:
1. English
2. Greek
Enter the number corresponding to your language choice: 1
Typing in en-US...
Person 1, please type your input: hello i want to stop the program
Processed Text: hello i want to stop the program
Person 1, please type your input: hello i want to switch to another user
Processed Text: hello i want to switch to another user
Person 1, please type your input: switch
Who is speaking?
Enter the new speaker (e.g., Person 2): kostas
Switched to kostas
kostas, please type your input: kostas i going to terminate the program or end it or quit it or stop it
Processed Text: kostas i going to terminate the program or end it or quit it or stop it
kostas, please type your input: end
Do you want to terminate the process? (yes/no): yes
Terminating the process.
(venv) PS C:\Users\kostas\Desktop\thesis>
```

Figure 89 εισαγωγή λέξεων για τερματισμό και αλλαγή χρήστη στο πρόγραμμα.

Στην τελευταία εικόνα είναι η παρόμοια διαδικασία με Ελληνικά γράμματα αυτή την φορά.

```
Select the language you'll be speaking:
1. English
2. Greek
Enter the number corresponding to your language choice: 2
Typing in el-GR...
Person 1, please type your input: Θελω να σταμάτα το πρόγραμμα
Processed Text: Θελω να σταμάτα το πρόγραμμα
Person 1, please type your input: Θέλω να αλλαγή σε άλλο χρήστη
Processed Text: Θέλω να αλλαγή σε άλλο χρήστη
Person 1, please type your input: Αλλαγή
Who is speaking?
Enter the new speaker (e.g., Person 2): κώστας
Switched to κώστας
κώστας, please type your input: θέλω τον τερματισμός του προγράμματος
Processed Text: θέλω τον τερματισμός του προγράμματος
κώστας, please type your input: σταμάτα τερματισμός τέλος έξοδος
Processed Text: σταμάτα τερματισμός τέλος έξοδος
κώστας, please type your input: τέλος
Do you want to terminate the process? (yes/no): yes
Terminating the process.
(venv) PS C:\Users\kostas\Desktop\thesis>
```

Figure 90 Εισαγωγή στα Ελληνικά

5.4 Δοκιμή αντιστοίχισης εικόνων με γράμματα

Στο συγκεκριμένο υποκεφάλαιο θα ελέγξουμε εάν όντως γίνεται η αναπαράσταση των γραμμάτων σε νοηματικά σύμβολα στα Αγγλικά, στα Ελληνικά και αν οι αριθμοί από το 0-9 αντιστοιχούν επίσης. Με λίγα λόγια απλά θα βάλουμε σαν είσοδο το αλφάβητο και στις δύο γλώσσες και θα διαπιστώσουμε ότι η αντιστοιχία είναι σωστή.

Figure 91 Αλφάβητο της Αγγλικής γλώσσας.

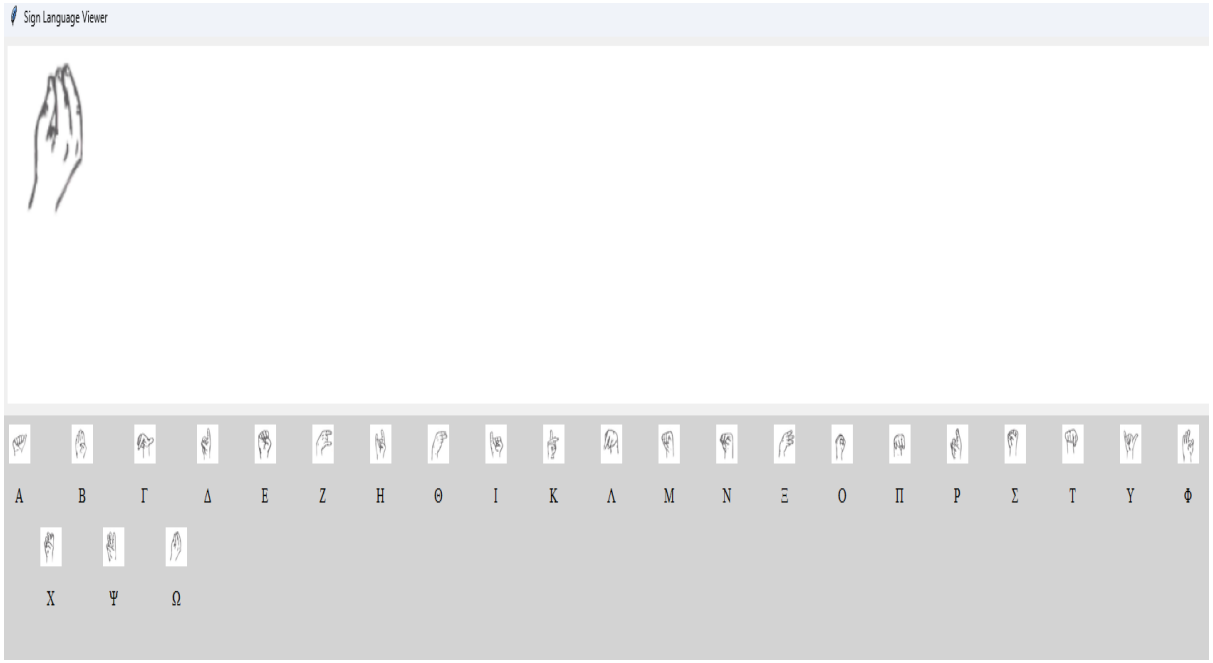

Figure 93 Αλφάβητο της Ελληνικής γλώσσας.

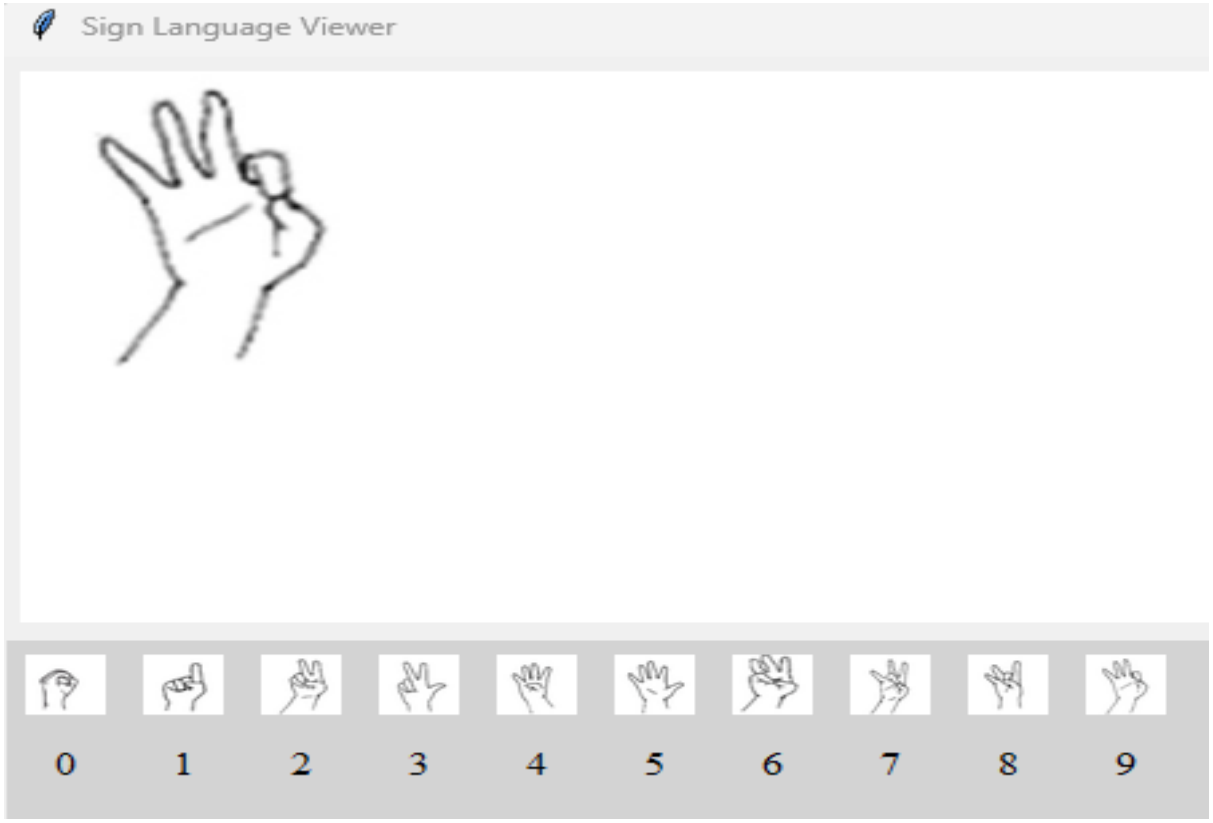

Figure 92 Αριθμοί από το 0 έως το 9.

ΚΕΦΑΛΑΙΟ 6 Συμπεράσματα και μελλοντικές επεκτάσεις

6.1 Δυσκολίες στην υλοποίηση του προγράμματος

Κατά τη διάρκεια της ανάπτυξης και υλοποίησης του προγράμματος, προέκυψαν αρκετές προκλήσεις που έπρεπε να αντιμετωπιστούν προκειμένου να επιτευχθούν τα αποτελέσματα που θέλουμε για την έξοδο του προγράμματος μας. Κάποιες από τις κύριες δυσκολίες που εντοπίστηκαν είναι οι παρακάτω:

Συμβατότητα με εργαλεία και βιβλιοθήκες

Η επιλογή και η ένταξη των κατάλληλων βιβλιοθηκών όπως **speech_recognition, Transformers** ακόμα όμως και το **CUDA Toolkit** και **cuDNN** το οποίο είναι αναγκαίο για την εκπαίδευση του μοντέλου μας χρησιμοποιώντας την κάρτα γραφικών μας, αλλά το μεγαλύτερο θέμα ήταν η σωστή έκδοση που είναι συμβατή με την βιβλιοθήκη **PyTorch.** Επίσης υπήρξαν ασυμβατότητες μεταξύ της έκδοσης της python που χρησιμοποιήθηκαν (βιβλιοθήκες όπως **CUDA cuDNN** και **TensorFlow)** συγκεκριμένες εκδόσεις δεν δουλεύουν. Στην περίπτωση του προγράμματος μας έχουμε την **TensorFlow 2.18.0** η οποία χρειάζεται **CUDA ToolKit == 12.3 && cuDNN == 8.9.7,** έτσι οι συγκεκριμένες ασυμβατότητες που συνάντησα οδήγησαν σε καθυστερήσεις κατά τη διαδικασία υλοποίησης.

Απόδοση μοντέλων μηχανικής μάθησης και επιλογής μοντέλου

Κατά την διαδικασία εκπαίδευσης και δοκιμής των μοντέλων **(BART και T5),** εντοπίστηκαν προβλήματα overfitting, δεν ήταν αρκετά τα δεδομένα για να εκπαιδεύσουμε το μοντέλο μας πάνω από 4 epochs χωρίς να έχουμε αρνητική επίδραση στα score για την αποδοτικότητα του μοντέλου **(ROUGE-1, ROUGE-2, ROUGE-L, BLEU)** οπότε και τα δύο μοντέλα είναι εκπαιδευμένα σε 4 epochs. Τέλος η βελτιστοποίηση των παραμέτρων και η διαχείριση των δεδομένων υπήρξε μια πρόκληση για εμένα καθώς δεν είχα ξανά επιχείρηση κάτι παρόμοιο σε τέτοια κλίμακα και πολλές φορές τα Online documentation δεν φάνηκαν αρκετά χρήσιμα, ειδικά στο σημείο της επιλογής παραμέτρων για το μοντέλο.

Υπολογιστική ισχύς

Παρόλο που υπήρξε μια αξιόλογη υπολογιστική ισχύς από τον προσωπικό μου υπολογιστή η εκπαίδευση οποιουδήποτε μοντέλου πάνω από το βασικό **(“facebook-base”** , **“T5-small”)** έμοιαζε ακατόρθωτη διότι ακόμα και αν χρησιμοποιούσα το αμέσως επόμενο μοντέλο η ώρες που χρειαζόντουσαν κυμαίνονταν από τις **242-270 ώρες** το οποίο μεταφράζεται σε **10-12 μέρες** συνεχόμενης χρήσης του υπολογιστή και με μέσο όρο **145-160watt** μπορούμε να δούμε ότι για την εκπαίδευση θα χρειαστούν (η kWh υπολογίσθηκε στα 0.2108€) περίπου στα 9.20€. Αυτό προϋποθέτει ότι έχει γίνει σωστά και οι παράμετροι που χρησιμοποιήθηκαν δεν χρειάζονται τροποποιήσεις όπως τα **batch_sizes, learning_rate, epochs,** γιατί εφόσον δεν είμαστε ικανοποιημένοι από το αποτέλεσμα το κόστος θα ανέβει κατά πολύ γιατί το μοντέλο θα χρειαστεί αρκετές εκπαιδεύσεις. Στην περίπτωση με τα μικρά μοντέλα χρησιμοποιήθηκε αλγόριθμος οποίος έκανε έλεγχο στα μοντέλα ανά 2 epochs για να εντοπίσει από πιο σημείο και μετά είχαμε overfitting.

Electricity usage	Cost	Time span
1.47 kWh	$0.32	per day
10.3 kWh	$2.23	per week
44.6 kWh	$9.70	per month
536 kWh	$116.39	per year

This calculator assumes there are 30.44 days in a month and 365.25 days in a year on average.

Typical appliance: Desktop computer

Appliance power: 155 watt [W]

Use/run at: 100 % capacity

Usage: 12 days per month

Electricity Price: $0.2173 per kWh

Calculate Clear

Figure 94 Υπολογισμός κόστους εκπαίδευσης μοντέλου.

Και παρακάτω φωτογραφία που δείχνει τις ώρες που χρειάζεται το μοντέλο, την χρήση της ισχύς και πόσα watt χρειάζονται.

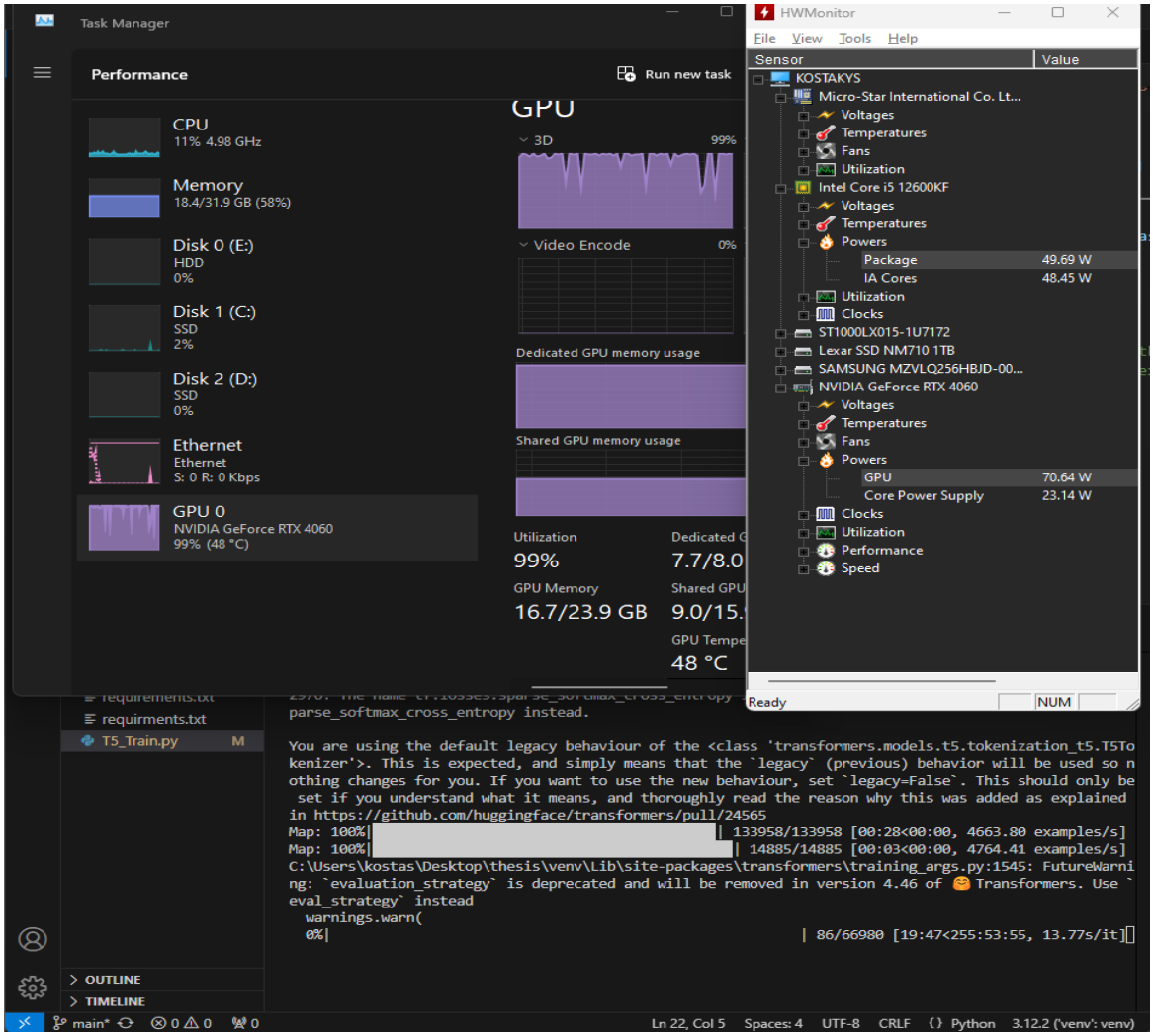

Figure 95 Χρήση υπολογιστή για την εκπαίδευση του μοντέλου.

Τέλος υπάρχουν επιπλέον επιλογές όπως το να ενοικιάζονται κάρτες γραφικών από website όπως το https://www.runpod.io/pricing για την εκπαίδευση μοντέλων απομακρυσμένα.

Δημιουργία Απλουστευμένου Κειμένου

Η διαδικασία απλοποίησης κειμένου για την προβολή της ASL παρουσίασε αρκετές δυσκολίες. Έπρεπε η εκπαίδευση και η επεξεργασία των παραμέτρων να αλλάζει ώστε τα μοντέλα να διατηρούν τις κύριες πληροφορίες που δημιουργούν με προτάσεις σύμφωνα με την είσοδο του χρήστη. Αυτό είχε ως αποτέλεσμα την εκπαίδευση των μοντέλων αρκετές φορές για να βρεθεί η κατάλληλη έξοδος αλλά ακόμα και από την τελική μορφή που είχαν τα μοντέλα μας η έξοδος ακόμα δεν ήταν τέλεια. Με λίγα λόγια ο συνδυασμός των ελλιπών δεδομένων και της απειρίας μου πάνω την εκπαίδευση μοντέλων δεν είχε την καλύτερη απόδοση στα μοντέλα.

Συνδυασμός Φωνητικής Αναγνώρισης και Απλοποίησης Κειμένου

Η ενσωμάτωση της **speech_recognition** με την μετατροπή της φωνής σε κείμενο παρουσίασε αρκετές τεχνικές δυσκολίες και ακόμα παρουσιάζει, διότι σε αρκετές περιπτώσεις η προφορά αλλά και πανομοιότυπες λέξεις μπορούν να συγχιστούν (π.χ. “switch” “sweets”), ακόμα όμως και ο θόρυβος του περιβάλλοντος και οι άρθρωση του χρήστη κατά την ομιλία μπορεί να μην έχει επιθυμητά αποτελέσματα.

Εμφάνιση και Προσαρμογή Εικόνων στη Νοηματική Γλώσσα

Η αντιστοίχιση νοηματικών συμβόλων με λέξεις και φράσεις ήταν μια αρκετά χρονοβόρα και επίπονη διαδικασία διότι έπρεπε να αντιστοιχίσω κάθε γράμμα με λέξεις και να δημιουργήσω μια συνάρτηση η οποία θα αντιστοιχίζει κάθε νόημα με τα γράμματα. Κάτι παρόμοιο σε βιβλιοθήκη δεν υπήρχε ή τουλάχιστον δεν βρήκα οπότε έπρεπε να το φτιάξω. Επιπλέον μια ακόμα ιδέα ήταν η συλλογή βίντεο στην νοηματική με λέξεις τα οποία βίντεο θα αποτελούσαν ένα καινούργιο dataset το οποίο θα εμπεριείχε κάθε λέξη και μετά την αντιστοιχία της σε strings τα οποία θα δημιουργούσαμε από μια νέα συνάρτηση που θα κατέγραφε για κάθε νόημα τα σημεία του χεριού και σε ποιες συντεταγμένες θα βρίσκονται.

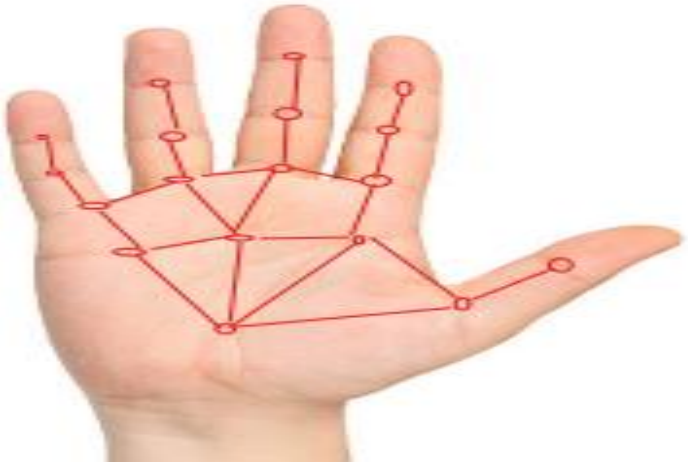

Figure 96 Σημεία που θα αποθηκεύουμε τα σημεία του χεριού.

Έλεγχος και Αξιολόγηση

Η αξιολόγηση της απόδοσης του προγράμματος, τόσο σε επίπεδο φωνητικής αναγνώρισης όσο και σε επίπεδο απλοποίησης κειμένου απαιτούσε τη δημιουργία αποτελεσμάτων για την σύγκριση των μοντέλων με **ROUGE, BLEU** scores αλλά η εξαγωγή των αποτελεσμάτων για την ανάλυση απλουστευμένων κειμένων ήταν αρκετά περίπλοκη και η δημιουργία προγράμματος για την εξαγωγή αποτελέσματος αρκετά περίπλοκη.

Παρά τις παραπάνω δυσκολίες, η υλοποίηση ολοκληρώθηκε (όχι με απόλυτη επιτυχία για τις δικές μου προδιαγραφές και ιδέες) χρησιμοποιώντας διαφορετικές προσεγγίσεις για την αντιμετώπιση των προκλήσεων που συνάντησα κατά την υλοποίηση, κατά την διάρκεια της υλοποίησης η μορφή του τελικού προγράμματος ήταν πολύ διαφορετική από την τελική υλοποίηση, πέρα από το γεγονός ότι άλλαξε αρκετά η δομή του ίδιου προγράμματος από πίσω (μεριά του κώδικα) η τελική μορφή του GUI είναι αρκετά διαφορετική από την αρχική υλοποίηση. Αυτή η τριβή με των κώδικα και η συνεχόμενες ιδέες και δυσκολίες με βοήθησαν να αναπτύξω τις προγραμματιστικές μου ικανότητες αλλά και την επίλυση προβλημάτων.

6.2 Προβλήματα εύρεσης βιβλιοθήκης Νοηματικής γλώσσας

Το πιο σημαντικό πρόβλημα που αντιμετώπισα κατά την ανάπτυξη του συστήματος ήταν η απουσία μια έτοιμης βιβλιοθήκης που παρέχει άμεση αντιστοίχιση λέξεων και φράσεων στην νοηματική γλώσσα. Αυτό είχε ως αποτέλεσμα να χρειαστεί η δημιουργία μια λύσης από το μηδέν, με αποτέλεσμα την δυσκολία ανάπτυξης του προγράμματος και μεγάλη χρονική καθυστέρηση στην υλοποίηση της εφαρμογής. Η απουσία ενός έτοιμου εργαλείου δημιούργησε τις εξής δυσκολίες.

Χειροκίνητη Αντιστοίχιση

Η διαδικασία αντιστοίχισης με κάθε λέξη με το κατάλληλο νοηματικό σύμβολο έγινε χειροκίνητα, κάτι που απαιτούσε αρκετή ώρα και γνώση της νοηματικής γλώσσας (Δεν είχα καμία επαφή με την νοηματική γλώσσα μέχρι την επαφή μου με την εργασία που υλοποίησα και έπρεπε να δω ότι κάθε σύμβολο έχει σωστή αντιστοίχιση).

Εύρεση Υλικού

Χρειάστηκε να αναζητηθούν εικόνες αλλά και βίντεο με νοηματικά σύμβολα από πηγές στο ίντερνετ [53]. Διαδικτυακές πλατφόρμες όπως το **Kaggle [54]** και **signASL [55]** για την άντληση ιδεών ως προς ποιο τρόπο να υλοποιηθεί το πρόγραμμα και η προβολή των συμβόλων.

Προσαρμογή στις Απαιτήσεις του Συστήματος

Το υλικό που βρήκα έπρεπε να προσαρμοστεί κατάλληλα για να λειτουργήσει το πρόγραμμα σωστά. Εφόσον κάποια λύση που να με ικανοποιούσε αλλά και να μπορούσα να την υλοποιήσω δεν υπήρχε έπρεπε να ξεκινήσω και να φτιάξω ένα σύστημα το οποίο θα ικανοποιούσε τις απαιτήσεις μου σε μερικό βαθμό αλλά θα ήταν και έτοιμο στην διορία που είχα θέση.

Με λίγα λόγια η έλλειψη μιας έτοιμης βιβλιοθήκης με ανάγκασε να δημιουργήσω μια συνάρτηση η οποία αντιστοιχεί νοηματικά σύμβολα με κείμενο σε Αγγλική και Ελληνική γλώσσα, καθώς η ανάπτυξη ενός πρώιμου προγράμματος το οποίο θα ανέλυε βίντεο με λέξεις από το **signASL [55]** και θα τις αντιστοιχούσε σε κάποιο πίνακα που είχε θέση σε 2D για την απεικόνιση ακουγόταν αρκετά φιλόδοξο δεν ήταν και τόσο πρακτικό, καθώς η δημιουργία πάνω από 1000+ λέξεων για την ύπαρξή βασικών συζητήσεων ήταν δύσκολη.

6.3 Μελλοντικές προεκτάσεις

Η αναφορά στο προηγούμενο υποκεφάλαιο 6.1 ανέφερα την προοπτική για δημιουργία ενός κώδικα ο οποίος επεξεργάζεται mp4 αρχεία , και ο τρόπος με τον οποίο το επιτυγχάνει είναι η βιβλιοθήκη **mediapipe** που είναι ένα framework για τον εντοπισμό χεριών σε αυτή την περίπτωση. Ουσιαστικά θα έχουμε το βίντεο μας και θα επεξεργαζόμαστε κάθε frame και θα αποθηκεύουμε για κάθε frame τις θέσεις των χεριών του εικονιζόμενου προσώπου. Στην συνέχεια μέχρι να τελειώσει το βίντεο θα αποθηκεύουμε κάθε σημείο στο csv και επειδή έχουμε να κάνουμε με την ASL θέλουμε να έχουμε και τον δείκτη μας σαν ορόσημο οπότε σε σχέση με τα άλλα σημεία κάνουμε τον δείκτη πράσινο.

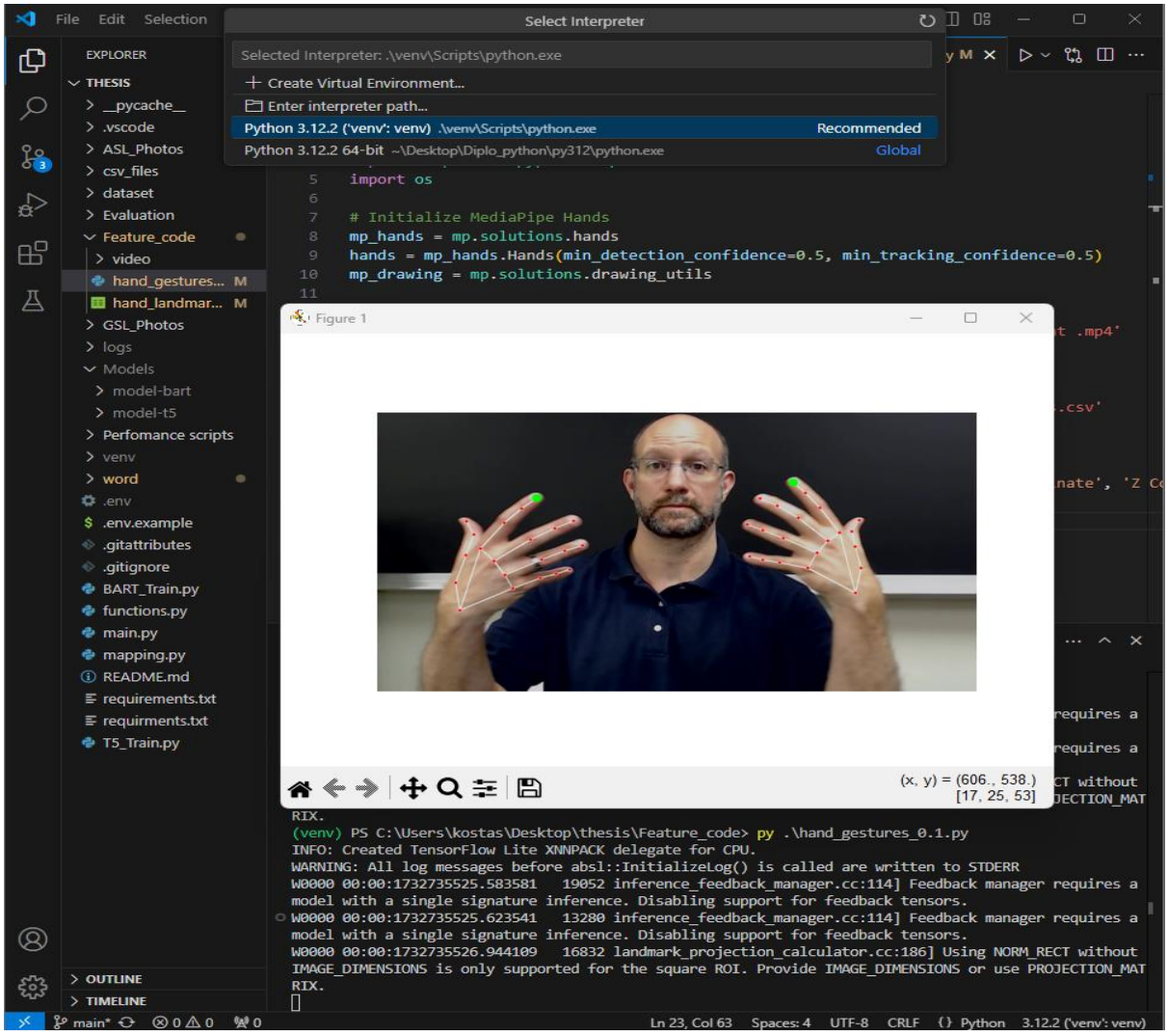

Figure 97 Παράδειγμα εφαρμογής ενός απλού προγράμματος για μελλοντική χρήση.

Η παραπάνω υλοποίηση είναι αρκετά πιο περίπλοκη και χρονοβόρα. Αρχικά θα χρειαζόμασταν ένα αρχείο/συνάρτηση το οποίο θα ήταν υπεύθυνο/η για την μετατροπή και αποθήκευση των δεδομένων για το κάθε βίντεο στο csv. Εφόσον μετά από αυτή την χρονοβόρα διαδικασία έχοντας μετατρέψει τουλάχιστον 500-1000 λέξεις σε συντεταγμένες χεριών, και να μην ξεχνάμε ότι σε αρκετά βίντεο μερικές φορές η παλάμη δεν εμφανίζεται σωστά οπότε μπορεί να εμφανίζει μόνο ένα χέρι το κύριο πρόγραμμα αλλιώς θα έχουμε προβλήματα στην εκτέλεση.
Εφόσον δημιουργήσουμε αυτά τα δεδομένα και ξεπεράσουμε και τα προβλήματα με τις ελλείψεις χεριών ή πότε να καταλαβαίνει το πρόγραμμα ότι θα έχουμε μόνο δεξιό ή αριστερό χέρι, ένα μεγάλο πρόβλημα που θα συναντήσουμε είναι ότι τα περισσότερα βίντεο σε συγκεκριμένες ιστοσελίδες **www.signASL.org**, **https://www.signingsavvy.com/** , **https://lifeprint.com/**, **https://www.spreadthesign.com/en.us/search/** , **https://www.handspeak.com/** είναι και άλλες αλλά αυτές είναι ένα μικρό δείγμα, θα χρειαστεί να πάρουμε άδεια για να χρησιμοποιήσουμε τα συγκεκριμένα βίντεο αλλά ακόμα και αν πάρουμε την συγκεκριμένη άδεια στην καλύτερη περίπτωση η ιστοσελίδα θα διαθέτει ένα API που θα μας δίνει την δυνατότητα να χρησιμοποιούμε έτσι τα βίντεο αλλά στην χειρότερη χρειάζεται να δημιουργήσουμε ένα script για την αυτοματοποίηση εγγραφής των βίντεο από την ιστοσελίδα και πλοήγηση μέσα στην σελίδα για την εγγραφή παραπάνω από ενός βίντεο για τον εμπλουτισμό της βάσης. Πέρα από αυτό εφόσον έχει λυθεί και αυτό το πρόβλημα έπειτα θα πρέπει να διαθέσουμε αρκετές ώρες αδιάκοπης εγγραφής των λέξεων και σωστής αποθήκευσης δεδομένων στο csv.
Οπότε σε έναν ιδανικό κόσμο που δεν υπάρχουν αυτά τα προβλήματα θα μπορούσαμε να είχαμε δημιουργήσει μια βιβλιοθήκη **ASLAnim** θα την ονόμαζα η οποία θα ήταν διαθέσιμη open-source για οποιονδήποτε θέλει να την επεκτείνει και να συνεισφέρει και έπειτα μπορούμε να δημιουργήσουμε το πρόγραμμα μας παρόμοιας δομής με το τωρινό μας απλά η μόνη διαφορά είναι ότι θα έχουμε νοηματικά σύμβολα και “argo” της Νοηματικής (τα οποία μπορούν να χρησιμοποιηθούν με μεγαλύτερη ευκολία από έμπειρους χρήστες).

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] “Noimatikikratilos.” *Εκπαιδευτικό Κέντρο Νοηματικής Γλώσσας Κρατύλος*, 19 Nov. 2019, noimatiki-kratilos.gr/%CE%BD%CE%BF%CE%B7%CE%BC%CE%B1%CF%84%CE%B9%CE%BA%CE%AE-%CE%B3%CE%BB%CF%8E%CF%83%CF%83%CE%B1/. Accessed 30 Sept. 2024.

[2] Eberhard, D. M., Simons, G. F., & Fennig, C. D. (Eds.). (2024). *Ethnologue: Languages of the world* (27th ed.). SIL International. http://www.ethnologue.com

[3] World Health Organization. “Deafness and Hearing Loss.” *WHO*, World Health Organization: WHO, 2 Feb. 2024, www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss.

[4] Wikipedia Contributors. “Sign Language.” *Wikipedia*, Wikimedia Foundation, 6 Aug. 2019, en.wikipedia.org/wiki/Sign_Language.

[5] project gutenberg. “Cratylus.” *Gutenberg.org*, 2022, www.gutenberg.org/cache/epub/1616/pg1616-images.html#chap02. Accessed 30 Sept. 2024.

[6] DAYAS, INÉS. “The History of Sign Language.” *History*, 28 May 2019, www.nationalgeographic.com/history/history-magazine/article/creation-of-sign-language.

[7] “Deaf History - Europe - 1620: Juan Pablo Bonet, the First Book on the Subject of Manual Alphabetic Signs (ES).” *Deafhistory.eu*, deafhistory.eu/index.php/component/zoo/item/1620.

[8] Bahan, Benjamin (1996). *Non-Manual Realization of Agreement in American Sign Language* (PDF). Boston University. Archived (PDF) from the original on October 11, 2017.

[9] *"A Brief History Of The American Asylum, At Hartford, For The Education And Instruction Of The Deaf And Dumb". 1893. Archived from the original on September 5, 2013*

[10] Mt. San Antonio College. ***5 Parameters of ASL***. Mt. San Antonio College Language Learning Center, n.d., https://www.mtsac.edu/llc/passportrewards/languagepartners/5ParametersofASL.pdf.

[11] Valli, Clayton, et al. *American Sign Language: A Look at Its History, Structure, and Community*. 2nd ed., Gallaudet University Press, 2011.Chapter 1,6,7(4,12,103,108,121,136 pages)

[12] Bialystok, E. (2001). *Bilingualism in Development: Language, Literacy, and Cognition*. Cambridge University Press.

[13] Gatt, A., & Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61, 65-170.

[14] Ping Chen, John Rochford, David N Kennedy, Soussan Djamasbi, Peter Fay, and Will Scott. 2017. Automatic text simplification for people with intellectual disabilities. In Artificial

Intelligence Science and Technology: Proceedings of the 2016 International Conference (AIST2016), pages 725–731. World Scientific.

[15] AmaliaTodirascu, Rodrigo Wilkens, Eva Rolin, Thomas Franc¸ois, Delphine Bernhard, and Nuria Gala. 2022.

[16] (n.d.). https://www.semanticscholar.org/paper/A-Review-of-Research-Based-Automatic-Text-Tools-Espinosa-Zaragoza-Salas/5a06ebdb791ec164441e635d1b209ff196da1e60y.org/2023.ranlp-1.36.pdf page 2(section 3.1).

[17] Lewis, M. *et al.* (2019) *Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*, *arXiv.org*. Available at: https://arxiv.org/abs/1910.13461 (Accessed: 17 October 2024).

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI Blog, 1(8), 2019.

[20] “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension” 29 Oct. 2019, arxiv.org/abs/1910.13461. Accessed 17 Oct. 2024.

[21] “Just a moment...” machinelearningmastery.com/the-transformer-model/. Accessed 18 Oct. 2024.

[22] “The Annotated Transformer” nlp.seas.harvard.edu/annotated-transformer/. Accessed 18 Oct. 2024.

[23] “Attention Is All You Need” 6 Dec. 2017, arxiv.org/abs/1706.03762. Accessed 18 Oct. 2024.

[24] Press, Authors:Ofir. “Using the Output Embedding to Improve Language Models” 21 Feb. 2017, arxiv.org/abs/1608.05859. Accessed 18 Oct. 2024.

[25] “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer” 19 Sept. 2023, arxiv.org/abs/1910.10683. Accessed 21 Oct. 2024.

[26] “T5” huggingface.co/docs/transformers/model_doc/t5. Accessed 21 Oct. 2024.

[27] www.phdata.io/blog/what-is-the-cost-to-deploy-and-maintain-a-machine-learning-model/. Accessed 21 Oct. 2024.

[28] “Common Crawl” *Overview*, commoncrawl.org/overview. Accessed 21 Oct. 2024.

[29] “GitHub - LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words: List of Dirty, Naughty, Obscene, and Otherwise Bad Words” *LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwis*, github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words. Accessed 21 Oct. 2024.

[30] “Langdetect” 16 Mar. 2015, pypi.org/project/langdetect/. Accessed 21 Oct. 2024.

[31] https://gptforwork.com/tools/tokenizer

[32] “ROUGE (metric)” *Wikipedia*, 24 Dec. 2008, en.wikipedia.org/wiki/ROUGE_(metric). Accessed 23 Oct. 2024.

[33] “Just a moment...” klu.ai/glossary/rouge-score. Accessed 23 Oct. 2024.

[34] “Bogdancazan/Wikilarge-Text-Simplification At Main” 6 June 2023, huggingface.co/datasets/bogdancazan/wikilarge-text-simplification/tree/main. Accessed 23 Oct. 2024.

[35] “Speech Recognition Module Python” *GeeksforGeeks*, 19 Mar. 2024, www.geeksforgeeks.org/python-speech-recognition-module/. Accessed 29 Oct. 2024.

[36] “Speech Recognition in Python [Learn Easily & Fast]” 27 July 2021, www.simplilearn.com/tutorials/python-tutorial/speech-recognition-in-python. Accessed 30 Oct. 2024.

[37]Malik, Usman. “Introduction to Speech Recognition with Python” 3 Jan. 2020, stackabuse.com/introduction-to-speech-recognition-with-python/. Accessed 21 Nov. 2024.

[38] “Python (programming language)” *Wikipedia*, 29 Oct. 2001, en.wikipedia.org/w/index.php?title=Python_(programming_language)&oldformat=true. Accessed 21 Nov. 2024.

[39]“What Is Python Used For? A Beginner’s Guide” *Coursera*, 17 May 2023, www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python. Accessed 21 Nov. 2024.

[40]“Top 90+ Python Libraries” *Flexiple*, flexiple.com/python/python-libraries. Accessed 21 Nov. 2024.

[41] “Papers with Code - CNN/Daily Mail Dataset” *Papers With Code*, paperswithcode.com/dataset/cnn-daily-mail-1. Accessed 21 Nov. 2024.

[42] “SpeechRecognition” 20 Oct. 2024, pypi.org/project/SpeechRecognition/. Accessed 21 Nov. 2024.

[43]“ Transformers” huggingface.co/docs/transformers/index. Accessed 21 Nov. 2024.

[44]“PyTorch documentation — PyTorch 2.5 documentation” pytorch.org/docs/stable/index.html. Accessed 21 Nov. 2024.

[45]“Python-Dotenv” 23 Jan. 2024, pypi.org/project/python-dotenv/. Accessed 21 Nov. 2024.

[46]“Pyttsx3” 27 Sept. 2024, pypi.org/project/pyttsx3/. Accessed 21 Nov. 2024.

[47]“Pillow” pillow.readthedocs.io/en/stable/. Accessed 21 Nov. 2024.

[48] “GitHub - google/sentencepiece: Unsupervised text tokenizer for Neural Network-based text generation.” Google/sentencepiece: Unsupervised text tokenizer , github.com/google/sentencepiece. Accessed 21 Nov. 2024.

[49]“tkinter — Python interface to Tcl/Tk” docs.python.org/3/library/tkinter.html. Accessed 21 Nov. 2024.

[50]“Hugging Face – The AI community building the future.” 19 Nov. 2024, huggingface.co/. Accessed 25 Nov. 2024.

[51]“Bart” huggingface.co/docs/transformers/en/model_doc/bart. Accessed 26 Nov. 2024.

[52]“Electricity Calculator” www.calculator.net/electricity-calculator.html?appliance=computer%3A150%3AW%3A10%3Ahpd%3A30&power=155&powerunit=W&capacity=100&usage=12&usageunit=dpm&price=0.2173&x=Calculate. Accessed 27 Nov. 2024.

[53]“American Sign Language” *Wikipedia*, 25 Sept. 2001, en.wikipedia.org/wiki/American_Sign_Language. Accessed 27 Nov. 2024.

[54]“ASL Alphabet” *Kaggle*, 22 Apr. 2018, www.kaggle.com/datasets/grassknoted/asl-alphabet?resource=download. Accessed 27 Nov. 2024.

[55]“Just a moment...” www.signasl.org/. Accessed 27 Nov. 2024.