

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



SPEECH TO TEXT TECHNOLOGIES FOR PEOPLE WITH DEAFNESS AND HEARING LOSS

ΗΣΑΙΑΣ ΚΩΝΣΤΑΝΤΊΝΟΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κωνσταντίνος Κολομβάτσος

Λαμία Φεβρουάριος έτος 2025



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

[ΠΕΡΙΛΗΨΗ i](#_Toc24624054)

[ABSTRACT iii](#_Toc24624055)

[ΚΕΦΑΛΑΙΟ 1 ΕΙΣΑΓΩΓΉ ΚΑΙ ΚΊΝΗΤΡΟ 1](#_Toc24624056)

**1.1** Λιγα λόγια για την νοηματική γλώσσα [1](#_Toc24624057)

**1.2** σκοπος και κινητρο της πτυχιακης εργασιας [1](#_Toc24624058)

**1.3** Εκμαθηση γραμμάτων στη νοηματική

**1.4** Εισαγωγή ατόμων στη Νοηματική γλωσσα(Με ή χωρις ακουστικά προβληματα)

[ΚΕΦΑΛΑΙΟ 2 Μοντελα μηχανικης μαθησης 2](#_Toc24624059)

**2.1** bart-base [2](#_Toc24624060)

**2.2** t5-small [2](#_Toc24624061)

**2.3** 3ο μοντέλο μηχανικής μαθησης

**2.4** Συγκριτική αξιολόγηση μεταξύ των μοντέλων

[ΚΕΦΑΛΑΙΟ 3 SPEECH RECOGNITION……… 3](#_Toc24624062)

3.1 speech recognition βιβλιοθηκη [3](#_Toc24624063)

3.2 Αναγνωριση ομιλιας με recognize\_google

3.3 λειτουργια recognizer  [3](#_Toc24624064)

[ΚΕΦΑΛΑΙΟ 4 προτεινομενο συστημα……… 4](#_Toc24624065)

4.1 γλωσσα προγραμματισμου και πακετα [4](#_Toc24624066)

4.2 αναλυση λειτουργιας του προγραμματος

4.3 διαρθωση κωδικα

4.4 εκπαιδευση μοντελων bart και t5-small [4](#_Toc24624067)

[ΚΕΦΑΛΑΙΟ 5 πειραματικη αποτιμηση 5](#_Toc24624068)

5.1 ελεγχος αποθηκευση στο csv

5.2 δοκιμες speech\_recognition σε δυο γλωσσες

5.3 Ελεγχος λειτουργιας switch/stop

5.4

5.5 Δοκιμή αντιστοιχησης εικόνων με γραμματα

5.6 Ελεγχος αναγνωρισης χαρακτηρων, αριθμων και συμβολων

5.7 Δοκιμη προβολης εικονων (ταχυτητα και ομαλοτητα)

5.8 Προτασεις απο χρηστη ASL για βελτιωση εφαρμογης

[κεφαλαιο 6 συμπερασματα και μελλοντικες προεκτασεις……………….](#_Toc24624069)

6.1 δυσκολιες στην υλοποιηση του προγραμματος

6.2 προβληματα ευρεσης λεξιλογίου νοηματικης γλωσσας

6.3 αναπτυξη της νοηματικης γλωσσας

6.4 Μελλοντικες προεκτασεις: χρηση νεων τεχνολογιων(animation bot)

6.5 προοπτικοη για προγραμμα αναγνωρισης νοηματων απο βιντεο

6.6 Δημιουργια μοντελου που συγκρινει βιντεο με κειμενα

[ΒΙΒΛΙΟΓΡΑΦΙΑ 6](#_Toc24624069)

# 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]

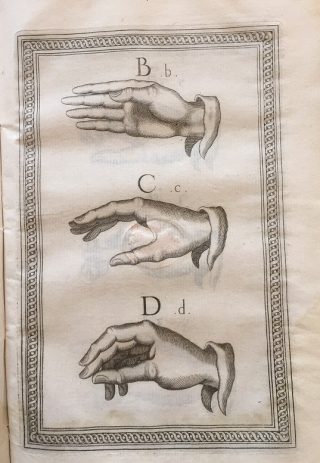
Σύμφωνα με τον Π.Ο.Υ (παγκόσμιος οργανισμός υγείας) περισσότερο από το 5% του παγκόσμιου πληθυσμού ή περίπου 430 εκατομμύρια άνθρωποι έχουν μόνιμη βλάβη στο ακουστικό τους σύστημα, είτε υπάρχει εκ γενετής ή μετά από ατύχημα. Μέχρι το 2050 εκτιμάται ότι πάνω από 700 εκατομμύρια άνθρωποι θα έχουν προβλήματα 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]



**Εικόνα 1 Εικόνα από το βιβλίο του Juan Pablo Bonet** [**https://shorturl.at/qtZB5**](https://shorturl.at/qtZB5)

**Εικόνα 2 Εικόνα από το βιβλίο του Juan Pablo Bonet** [**https://shorturl.at/zxkDn**](https://shorturl.at/zxkDn)

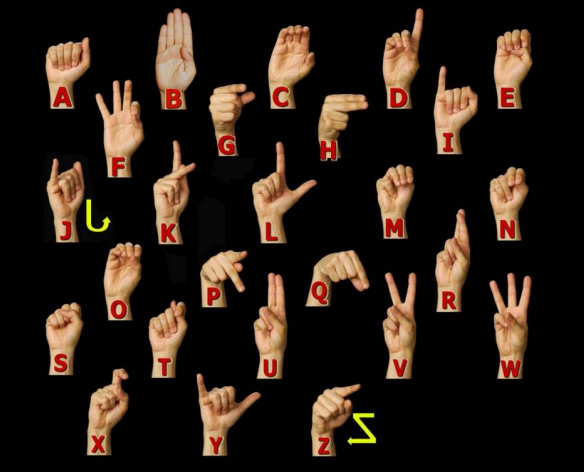
Η ASL( American sign language ) έχει ρίζες από την Αμερικανική σχολή για κωφούς που ιδρύθηκε στο Hartford, Connecticut το 1817[8]. Η συγκεκριμένη σχολή ιδρύθηκε από τον απόφοιτο του Yale Thomas Hopkins Gallaudet[9] και εκείνη την εποχή θεωρούταν σχολή για “χαζούς”.

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.

****

**Εικόνα 3 Αλφάβητο της ASL** [**https://shorturl.at/iMmTu**](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.

Εικόνα που περιέχει πεταλούδα

Περιγραφή που δημιουργήθηκε αυτόματα

**Εικόνα 4 Αλφάβητο της ASL** [**https://shorturl.at/pyVqo**](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)**.

**Εικόνα 5 Συνδυασμός του BERT+GPT αποτελεί το BART** [**https://shorturl.at/dNSPc**](https://shorturl.at/dNSPc)

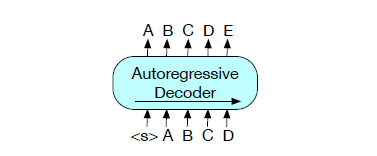
Το **BART** έχει σχεδιαστεί από το Facebook AI καιείναι αρκετά αποδοτικό, όταν ρυθμιστεί στις κατάλληλες παραμέτρους και δεδομένα, για να επιλύσει εργασίες **NLP (Natural language processing)**, όπως η δημιουργία κειμένου αλλά και εργασίες λογικής. Στην συγκεκριμένη περίπτωση θα το χρειαστούμε για να 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.

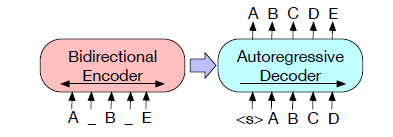
* Εικόνα που περιέχει κείμενο, γραμματοσειρά, σχεδίαση

  Περιγραφή που δημιουργήθηκε αυτόματαΤο 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]

**Εικόνα 7 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 κατανοήσει πλήρως αμφίδρομες αλληλεπιδράσεις στο κείμενο.[19] **Εικόνα 8**

**Αποκωδικοποιητής[17].**

* Στο μοντέλο **BART**, τα δεδομένα εισόδου στον κωδικοποιητή (**Encoder**) δεν χρειάζεται να ταιριάζουν με αυτά που παράγει ο αποκωδικοποιητής (**Decoder**), δίνοντας τη δυνατότητα για

**Εικόνα 9 Κωδικοποιητής/Αποκωδικοποιητής[17].**

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]

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.

Εικόνα που περιέχει κείμενο, διάγραμμα, στιγμιότυπο οθόνης, Σχέδιο

Περιγραφή που δημιουργήθηκε αυτόματα

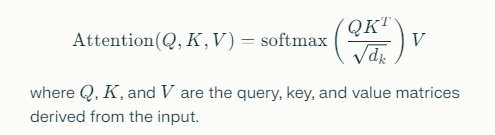
**Εικόνα 10 Κωδικοποιητής/Αποκωδικοποιητής Αναλυτικό γράφημα** [**https://nlp.seas.harvard.edu/annotated-transformer/**](https://nlp.seas.harvard.edu/annotated-transformer/)**[23]**

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.



**Εικόνα 11 Μηχανισμός στον Κωδικοποιητή/αποκωδικοποιητή** [**https://nlp.seas.harvard.edu/annotated-transformer/**](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.

**Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα**

**Εικόνα 12 Εικόνα για την εφαρμογή του Attention** [**https://nlp.seas.harvard.edu/annotated-transformer/[23**](https://nlp.seas.harvard.edu/annotated-transformer/%5b23)**]**

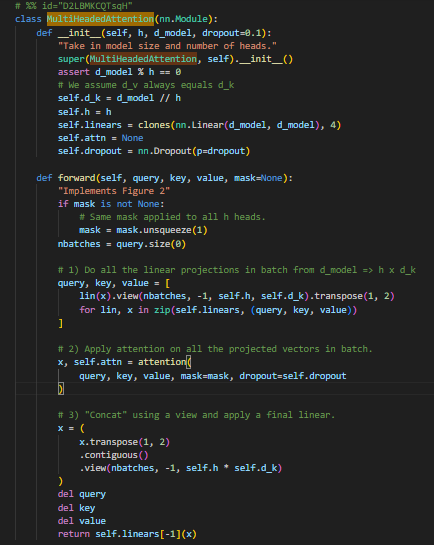
**Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα**

**Εικόνα 12.1 Κώδικας Attention σε python**

[**https://github.com/ikoctac/annotated-transformer/[22**](https://github.com/ikoctac/annotated-transformer/%5b22)**]**

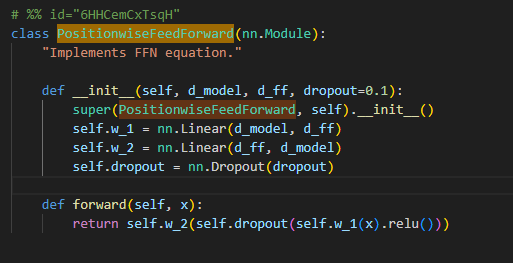
Η **MultiHeadedAttention** class που χρησιμοποιεί και το attention του παραπάνω κώδικα.

****

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

**Feed-Forward N****etwork (FFN)**

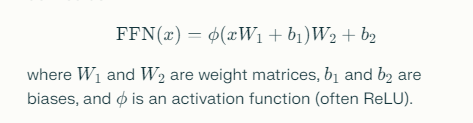
Εφόσον ολοκληρωθεί το πρώτο κομμάτι και υπολογισθεί το **Attention,** τα δεδομένα περνάνε μέσω ενός διπλού **Fully-Connected Network Layer.** Συγκεκριμένα το μοντέλο χρησιμοποιεί **Learned embeddings[24]** για την μετατροπή των **input-tokens** και **output-tokes** σε **Vectors χ** διαστάσεων. Επίσης 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

****

**Εικόνα 13 PositionwiseFeedForward κώδικας**

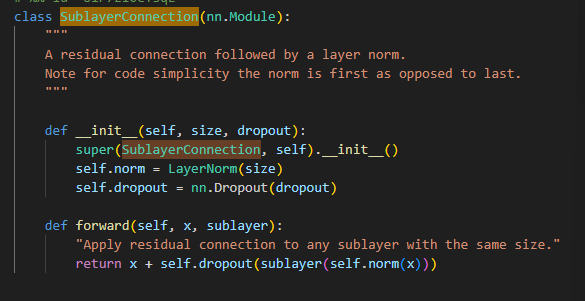
[**https://github.com/ikoctac/annotated-transformer [22**](https://github.com/ikoctac/annotated-transformer%20%5b22)**]**

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

****

**Εικόνα 14 Μαθηματική φόρμουλα για τον υπολογισμό στην Python** [**https://nlp.seas.harvard.edu/annotated-transformer/**](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.**

**Εικόνα 15.1 Κώδικας για SublayerConnection σε Python**

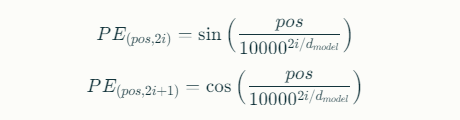
[**https://github.com/ikoctac/annotated-transformer [22**](https://github.com/ikoctac/annotated-transformer%20%5b22)**]**

**Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές, στιγμιότυπο οθόνης, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα 15.2 Κώδικας LayerNorm σε Python**

[**https://github.com/ikoctac/annotated-transformer**](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.**

**Εικόνα 16 Κώδικας για τοποθεσία των tokens στο paper για να υπολογισθούν**

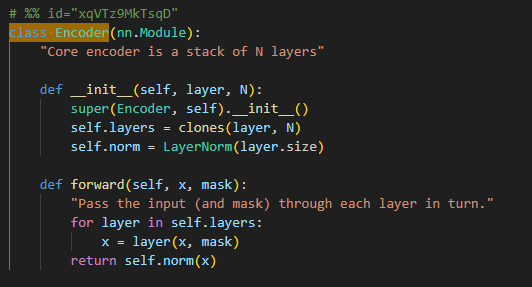
[**https://nlp.seas.harvard.edu/annotated-transformer/**](https://nlp.seas.harvard.edu/annotated-transformer/) **[22]**

**Encoder Stacking**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, οθόνη, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματαΤέλος 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”.**

**Εικόνα 17 Κώδικας για το EncoderLayer στην Python** [**https://github.com/ikoctac/annotated-transformer [22**](https://github.com/ikoctac/annotated-transformer%20%5b22)**]**

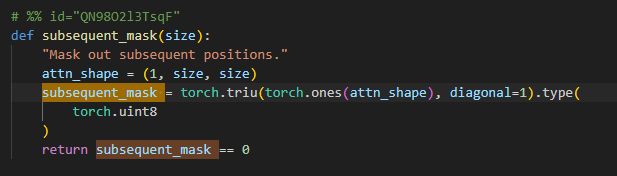
****

**Εικόνα 18 Τελικός κώδικας του κωδικοποιητή στην Python** [**https://github.com/ikoctac/annotated-transformer [22**](https://github.com/ikoctac/annotated-transformer%20%5b22)**]**

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**

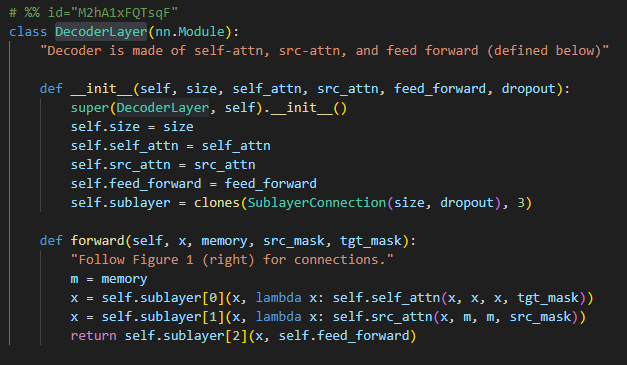
Όπως και στον κωδικοποιητή (**Encoder**) έτσι και εδώ έχουμε παρόμοιο πρώτο στρώμα είναι ίδιο με το **”[MultiHeadSelfAttention](#MultiHeadSelfAttention)”** μόνο που χρησιμοποιεί **masking** για να αποτραπεί η εστίαση σε μελλοντικά **tokens.** Έτσι στο **“DecodeLayer”** το πρώτα στρώμα που χρησιμοποιεί την **“[MultiHeadSelfAttention](#MultiHeadSelfAttention)”** μαζί με την **“subsequent\_mask”.**



**Εικόνα 19 Κώδικας για το Masked MHA (MutliHeadAttention) στην Python** [**https://github.com/ikoctac/annotated-transformer [22**](https://github.com/ikoctac/annotated-transformer%20%5b22)**]**

**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](#MultiHeadSelfAttention)”.**

****

**Εικόνα 20 Κώδικας για το DecoderLayer σε Python** [**https://github.com/ikoctac/annotated-transformer [22**](https://github.com/ikoctac/annotated-transformer%20%5b22)**]**

**Feed-Forward Network (FFN)**

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

**Add & Norm**

Παρόμοια με τον κωδικοποιητή **(Encoder)** έτσι και ο αποκωδικοποιητής **(Decoder) χρησιμοποιεί το ίδιο “[AddNorm](#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.

Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές, στιγμιότυπο οθόνης, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

**Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα 21 Κώδικας για το DecoderLayer σε Python** [**https://github.com/ikoctac/annotated-transformer [22**](https://github.com/ikoctac/annotated-transformer%20%5b22)**]**

**Εικόνα 22 Κώδικας για τον Decoder στην Python**

[**https://github.com/ikoctac/annotated-transformer [22**](https://github.com/ikoctac/annotated-transformer%20%5b22)**]**

**Final Linear & SoftMax**

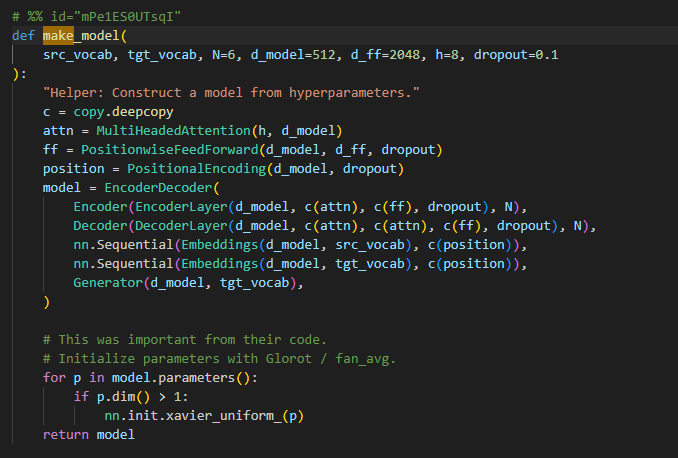
Τέλος η τελική έξοδος από τον αποκωδικοποιητή μετατρέπεται σε μια πιθανότητα πάνω στα δεδομένα που έχει προπονηθεί το μοντέλο και εξάγει ένα αποτέλεσμα με την class **“Generator”.**

**Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα**

**Εικόνα 23 Κώδικας για Generator στην Python**

[**https://github.com/ikoctac/annotated-transformer [22**](https://github.com/ikoctac/annotated-transformer%20%5b22)**]**

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

**Εικόνα 24 Κώδικας συνάρτησης για την δημιουργία μοντέλου στην Python** [**https://github.com/ikoctac/annotated-transformer [22**](https://github.com/ikoctac/annotated-transformer%20%5b22)**]**

**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.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

**Εικόνα 25 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**

H 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]

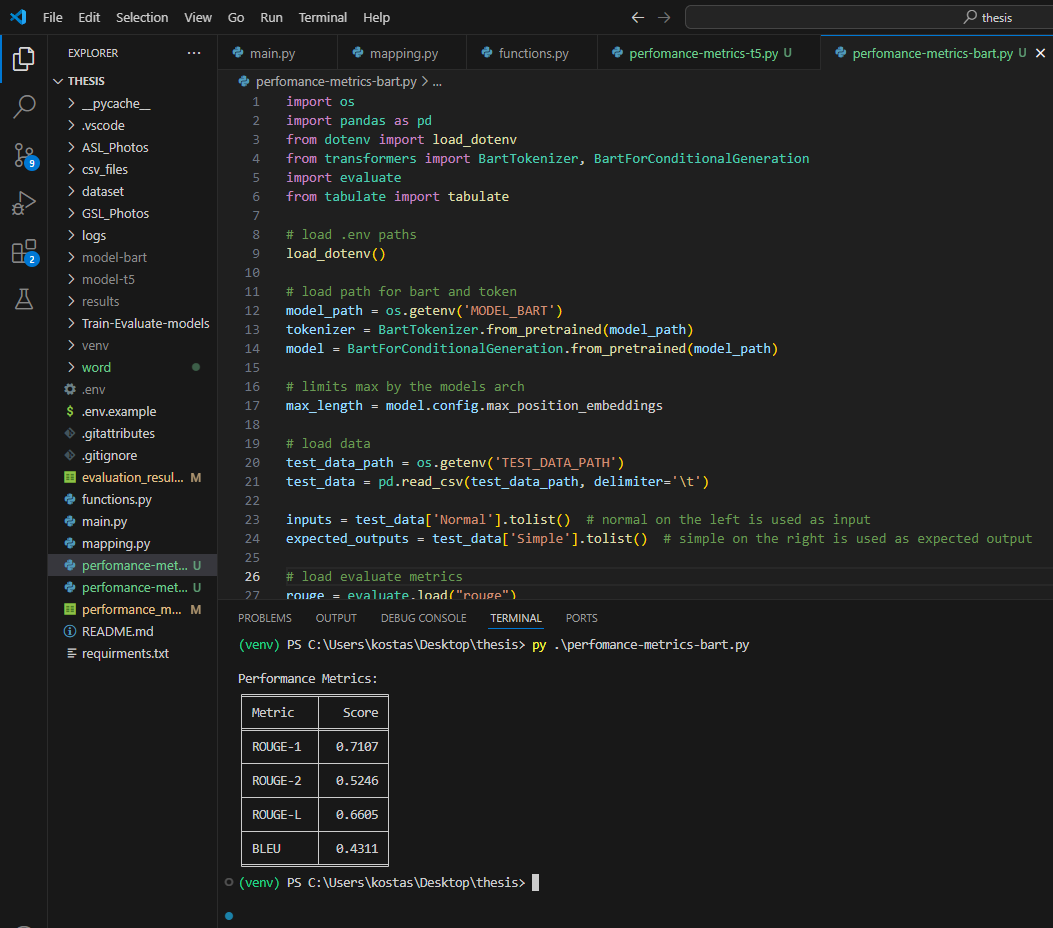
Όπως θα δούμε και παρακάτω το **BART** τα πηγαίνει εξαιρετικά καλά σε τεστ δοκιμών για την απόδοσή του για συγκεκριμένες εργασίες. Πιο συγκεκριμένα έχουμε τα: **ROUGE-1, ROUGE-2, ROUGE-L** και **BLEU.**

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.

* 1. **<0.1:** Δεν μπορεί να χρησιμοποιηθεί.
  2. **0.1 – 0.19:** Δύσκολα μπορεί να κατανοηθεί.
  3. **0.2 – 0.29:** Υπάρχει νόημα αλλά με σημαντικές αποκλίσεις.
  4. **0.3 – 0.39:** Καλή, απλή κατανοητή μετάφραση.
  5. **0.4 – 0.49:** Αρκετά καλή μετάφραση.
  6. **0.5 – 0.59:** Πολύ καλής ποιότητας μετάφραση.
  7. **>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].

**Εικόνα 26 Κώδικας για αξιολόγηση μοντέλου**

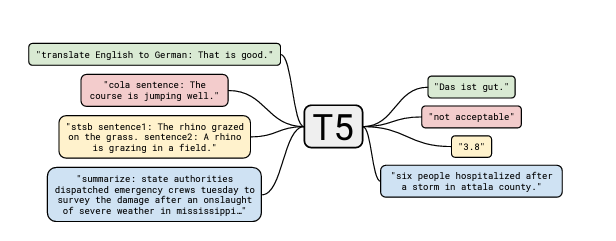
**perfomance-metrics-bart.py**

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

**Πίνακας 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.

**Εικόνα 27 Οπτικοποίηση των λειτουργιών του T5 [25]**

Το **Τ5** 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 Τ5.[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 **Τ5** χρησιμοποιεί την **text-to-text** τεχνικήκαι 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**

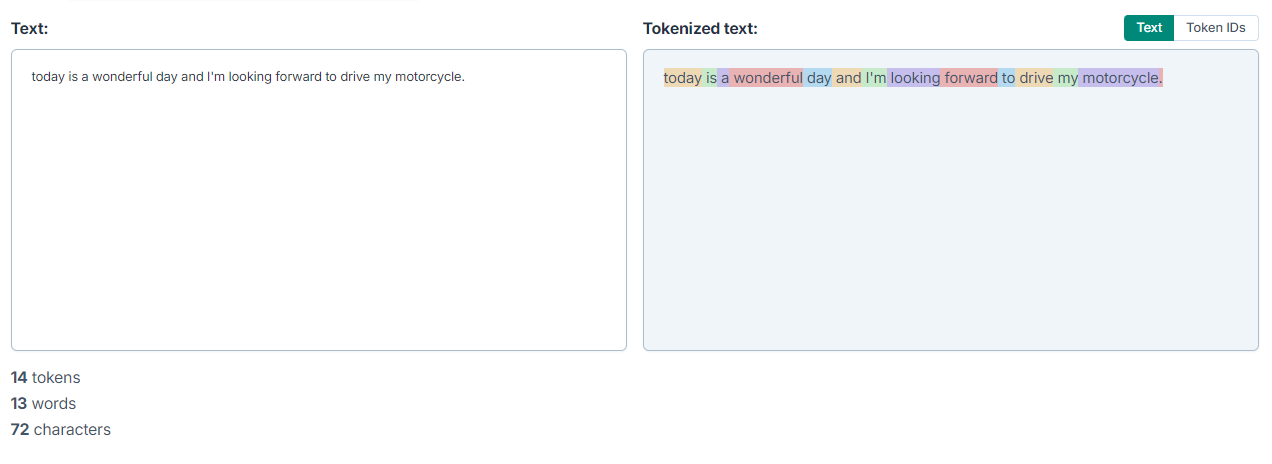
Το **Τ5**  κάνει **pretraining** στο C4 Dataset το οποίο είναι ένα μεγάλου μεγέθους web scraping project , το common crawl[28], το οποίο στην συνέχεια φιλτράρετε και γίνεται καθαρισμός δεδομένων για την βέλτιστη απόδοση του μοντέλου.

* 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**

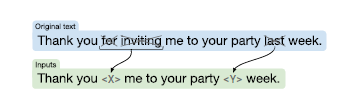
Το **Τ5** μοντέλο όπως είπαμε και πριν κάνει **pretraining** χρησιμοποιώντας τα δεδομένα από το **C4(Common crawl)[28]** για 524.288 βήματα. Αυτά τα βήματα ουσιαστικά είναι σαν το μοντέλο μας να διαβάζει για κάθε βήμα μια παρτίδα από λέξεις και φράσεις προσπαθώντας να μάθει πως διασυνδέονται μεταξύ τους. 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 **Τ5** μπορούν να διαχειριστούν και να επεξεργαστούν κατάλληλα, για παράδειγμα:

**Εικόνα 28 Οπτικοποίηση για το πώς δουλεύουν τα tokens** [**https://gptforwork.com/tools/tokenizer [31**](https://gptforwork.com/tools/tokenizer%20%5b31)**]**

΄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 **Τ5** 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.



**Εικόνα 29 Τυχαίες επιλογές κειμένου για να γίνει το token-masking[26]**

Αυτή η τεχνική δίνει την δυνατότητα στο **Τ5** μοντέλο να ανακατασκευάζει αποδοτικά το κείμενο μας και να καταλαβαίνει τις σχέσεις των λέξεων και προτάσεων μεταξύ τους.

**Fine-Tuning and Task Performance**

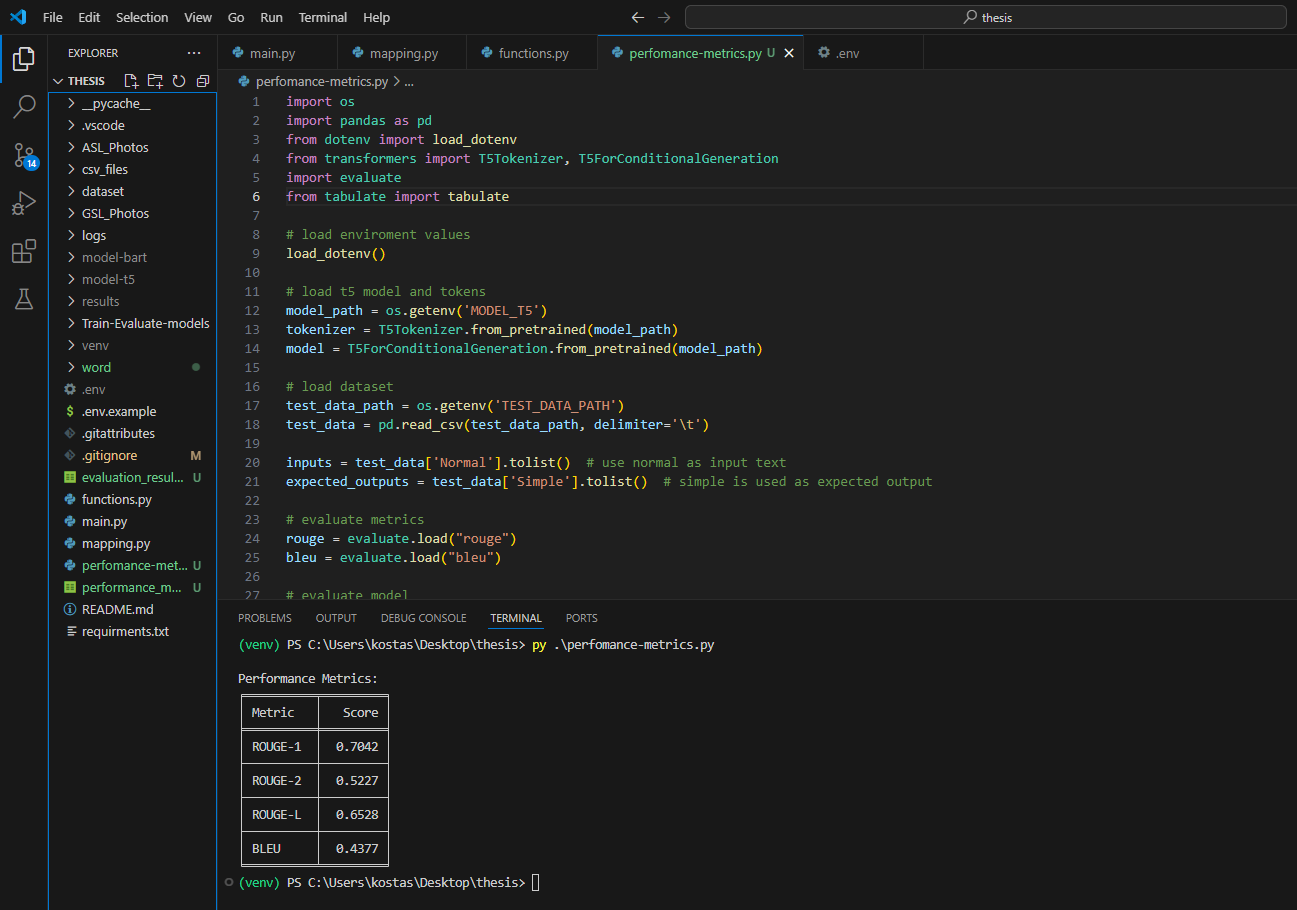
Εφόσον ολοκληρωθεί το pre-training το μοντέλο **Τ5** βελτιώνεται πάνω σε συγκεκριμένες εργασίες που δίνονται από τον χρήστη και χρησιμοποιεί δεδομένα από datasets που περιγράφουν τι εμπεριέχουν έτσι ώστε να γνωρίζει το μοντέλο ποια εργασία εκτελεί. Αυτό συμπεριλαμβάνει ότι ο χρήστης θα παραμετροποίηση της παραμέτρους κατάλληλα για εργασίες όπως **summarization,translation** και **question answering**.[26]

Όπως θα δούμε και παρακάτω το **Τ5** 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.

* 1. **<0.1:** Δεν μπορεί να χρησιμοποιηθεί.
  2. **0.1 – 0.19:** Δύσκολα μπορεί να κατανοηθεί.
  3. **0.2 – 0.29:** Υπάρχει νόημα αλλά με σημαντικές αποκλίσεις.
  4. **0.3 – 0.39:** Καλή, απλή κατανοητή μετάφραση.
  5. **0.4 – 0.49:** Αρκετά καλή μετάφραση.
  6. **0.5 – 0.59:** Πολύ καλής ποιότητας μετάφραση.
  7. **>0.6:** Η μετάφραση συνήθως ξεπερνάει και την ανθρώπινη.

****Παρακάτω είναι ο κώδικας στην python για την υλοποίηση των μετρήσεων του μοντέλου **Τ5** πάνω στα δεδομένα που έχει εκπαιδευθεί[34].

**Εικόνα 30 Κώδικας για την αξιολόγηση του T5 perfomance-metrics-t5.py**

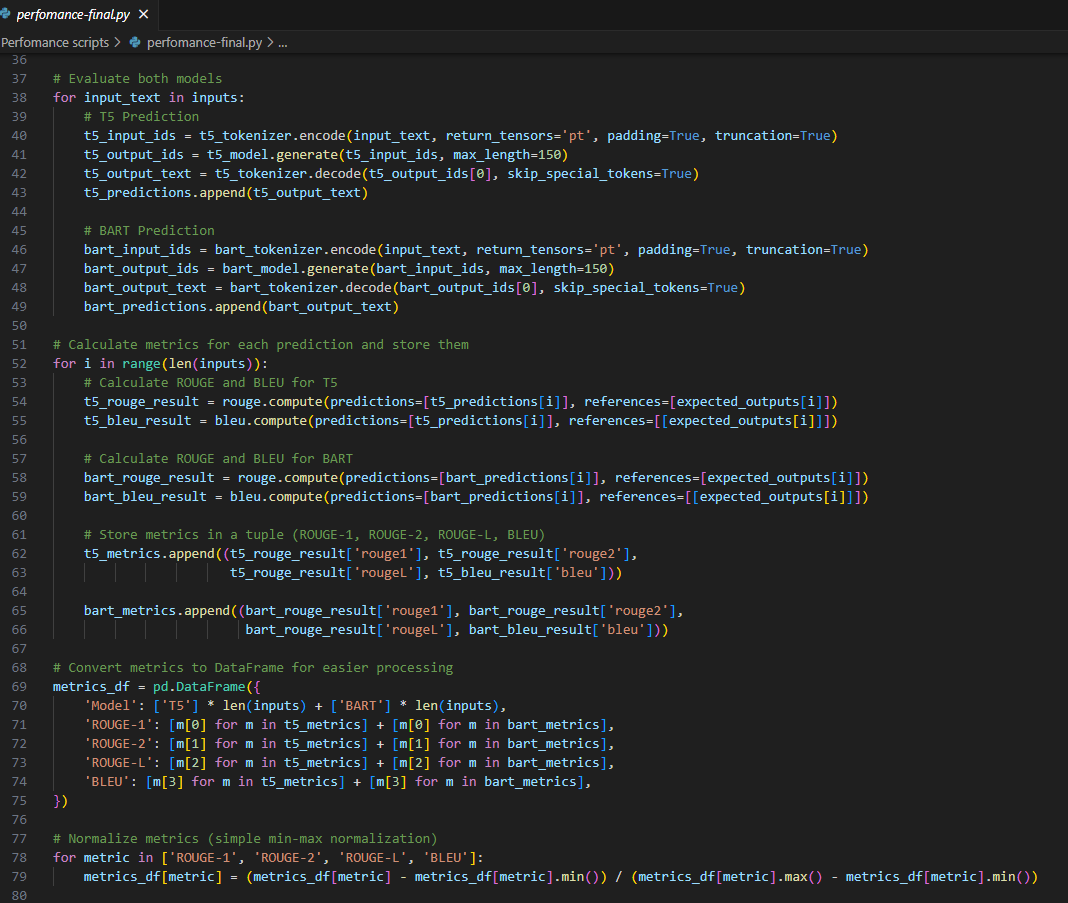
Όπως διακρίνεται και στην εικόνα έχουμε τα δεδομένα για το πώς αποδίδει το μοντέλο πάνω στα δεδομένα που του έχουμε δώσει.

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

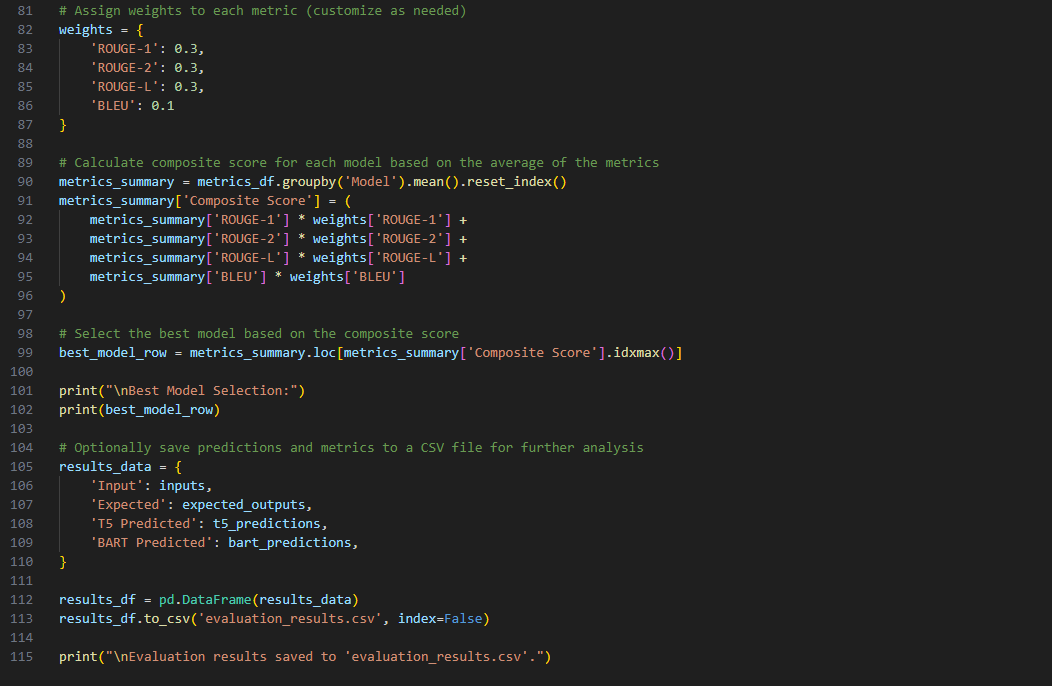
**Πίνακας 2 Αποτελέσματα του performance-metrics-T5.py**

## 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.

**Εικόνα 31 Μέρος πρώτο του script για την αξιολόγηση των 2 μοντέλων performance-final.py**

**Εικόνα 32 Δεύτερο μέρος του script για την αξιολόγηση των 2 μοντέλων performance-final.py**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματαΤέλος 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.

**Εικόνα 33 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.

A screen shot of a computer program

Description automatically generated

**Εικόνα 34 Βιβλιοθήκες για την SpeechRecognition main.py**

Στην συνέχεια ορίζουμε με μια μεταβλητή recognizer = sr.Recognizer() οπου 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.

A screen shot of a computer

Description automatically generated

**Εικόνα 35 Ορισμός μεταβλητής στον recognizer main.py**

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



**Εικόνα 36 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.

A black screen with white text

Description automatically generated

**Εικόνα 37 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.

A screen shot of a computer program

Description automatically generated

**Εικόνα 38 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).

A screenshot of a computer screen

Description automatically generated**Εικόνα 39 Εγγραφή ήχου από το μικρόφωνο main.py**

A screen shot of a computer screen

Description automatically generatedΚαι 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

**Εικόνα 40 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](#srmicrophone) και να ξεκινήσει η “εγγραφή” ήχου με την μέθοδο [**listen()**](#listenmethod)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.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

**Εικόνα 41 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.

Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

**Εικόνα 42 Class για την σταθεροποίηση λευκού θορύβου από το περιβάλλον του χρήστη**

**Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα** 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.

**Εικόνα 43 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)tqdm==4.66.5

1. **SpeechRecognition**
   1. **Version: 3.10.4**
   2. **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.
   3. **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**
   1. **Version:** 4.45.2
   2. **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.
   3. **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**
   1. **Version:** 2.4.1+cu121
   2. **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.
   3. **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**
   1. **Version:** 1.01
   2. **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.
   3. **Use case:** Στο δικό μας πρόγραμμα χρησιμοποιήθηκαν για την ευκολότερη διαχείριση μεταβλητών που αλλάζουν συχνά και βρίσκονται μέσα στο πρόγραμμα παραπάνω από μία φορές , αλλα και όταν γίνεται μεταφορά σε διαφορετικό χρήστη για το τρέξιμο του προγράμματος χρειάζονται να αλλάζουν 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**
   1. **Version:**2.98
   2. **Περιγραφή**: Η Pyttsx3 είναι μια βιβλιοθήκη text-to-speech που μπορεί να χρησιμοποιηθεί και χωρίς σύνδεση στο διαδίκτυο και έχει αρκετά speech engines.
   3. **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**
   1. **Version:** 10.4.0
   2. **Περιγραφή:** Η βιβλιοθήκη pillow χρησιμοποιείται για την μετατροπή και επεξεργασία εικόνων μέσα σε εφαρμογές python.
   3. **Use case :** Στο πρόγραμμα μου χρησιμοποιήθηκε για την προβολή εικόνων[47].
7. **Sentencepiece**
   1. **Version : 0.2.0**
   2. **Περιγραφή:** Η Sentencepiece βιβλιοθήκη είναι μια γλωσσικός ανεξάρτητη subword tokenizer και detokenizer σχεδιασμένη για neural-based δημιουργίες κειμένου.
   3. **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**
   1. **Version:** 0.1.0
   2. **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.
   3. **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

# ΚΕΦΑΛΑΙΟ 5 Συμπεράσματα

# ΒΙΒΛΙΟΓΡΑΦΙΑ

Εδώ θα προστεθεί όλη η βιβλιογραφία