ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΦΙΛΟΣΟΦΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΦΙΛΟΛΟΓΙΑΣ
ΤΟΜΕΑΣ ΓΛΩΣΣΟΛΟΓΙΑΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ
ΘΕΩΡΗΤΙΚΗ ΚΑΙ ΕΦΑΡΜΟΣΜΕΝΗ ΓΛΩΣΣΟΛΟΓΙΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υπολογιστική Ανάλυση της Εσωτερικής Αύξησης στα Σύνθετα Ρήματα με Προθέσεις

Κωνσταντίνος Λώμης
Α.Μ. 997

Επιβλέπων: Γεώργιος Μαρκόπουλος

Αθήνα, Ιούνιος 2018

Κωνσταντίνος Λώμης

Υπολογιστική Ανάλυση της Εσωτερικής Αύξησης στα Σύνθετα Ρήματα με Προθέσεις

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΠΟΠΤΗΣ:

Γ. Μαρκόπουλος

ΕΠΙΤΡΟΠΗ:

Γ. Μαρκόπουλος
Α. Μόζερ
Ε. Παναρέτου

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω ορισμένα άτομα, τα οποία συνέβαλαν με τον δικό τους τρόπο στην παρούσα διπλωματική εργασία.

Αρχικά θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου για την υποστήριξη που μου παρείχαν καθόλη τη διάρκεια συγγραφής της συγκεκριμένης εργασίας, αλλά και για τη γενικότερη στήριξη που μου παρέχουν σε όλη τη διάρκεια της ζωής μου.

Επιπλέον, θα ήθελα να ευχαριστήσω συγκεκριμένα τους φίλους μου Βεργίνα-Κωνσταντίνα Βυζικίδου και Γιάννη Τερζόπουλο για τα σχόλια και τις επισημάνσεις τους σε διάφορα στάδια της εργασίας. Οι παρατηρήσεις τους βελτίωσαν σε σημαντικό βαθμό την ποιότητα της συγκεκριμένης εργασίας.

Ακόμη, θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Μαρκόπουλο Γεώργιο για τη γενικότερη συμβολή του και τις επισημάνσεις του, με κυριότερη την πρότασή του περί μετασχηματισμού της αρχικής μορφής του προγράμματος, που οδήγησε σε σημαντική βελτίωση του τρόπου λειτουργίας του.

Τέλος, θα ήθελα να ευχαριστήσω θερμά όλους τους συμμετέχοντες που, με μεγάλη προθυμία, αφιέρωσαν χρόνο και έλαβαν μέρος στη διαδικασία των δοκιμών του προγράμματος.

Περίληψη

Η παρούσα διπλωματική εργασία εστιάζει στην εσωτερική αύξηση των σύνθετων ρημάτων με προθέσεις στα Νέα Ελληνικά. Η βασική υπόθεση της εργασίας είναι ότι υψηλό ποσοστό των φυσικών ομιλητών της Ελληνικής συγχέει τις περιπτώσεις στις οποίες η παρουσία της εσωτερικής αύξησης είναι απαραίτητη με τις περιπτώσεις στις οποίες δεν είναι απαραίτητη.

Προκειμένου να ελεγχθεί η εν λόγω υπόθεση, αναπτύχθηκε ένα υπολογιστικό μοντέλο με τη χρήση της γλώσσας προγραμματισμού Python. Το συγκεκριμένο μοντέλο λειτουργεί ως εξής: ζητάει από τον χρήστη να εισαγάγει μία πρόταση που να περιέχει σύνθετο ρήμα με πρόθεση στο τρίτο πρόσωπο ενικού αριθμού οριστικής αορίστου ή στο δεύτερο πρόσωπο ενικού αριθμού προστακτικής αορίστου. Στη συνέχεια, το μοντέλο ενημερώνει τον χρήστη εάν η πρόταση που εισήγαγε είναι γραμματικά αποδεκτή ή όχι. Σε περίπτωση που δεν είναι, το μοντέλο παρέχει στον χρήστη τη σωστή πρόταση, αντικαθιστώντας τον λανθασμένο ρηματικό τύπο με τον σωστό (αφαιρώντας ή προσθέτοντας την εσωτερική αύξηση).

Σε επόμενο στάδιο, πραγματοποιήθηκαν δοκιμές του μοντέλου στις οποίες συμμετείχαν 20 φυσικοί ομιλητές της Ελληνικής. Κάθε ομιλητής/ομιλήτρια εισήγαγε 10 προτάσεις στο πρόγραμμα, επομένως, ο συνολικός αριθμός των προτάσεων ανέρχεται στις 200. Το ποσοστό επιτυχίας του μοντέλου ανέρχεται στο 73% και αφορά τις περιπτώσεις που παρουσίασε τις γραμματικά αποδεκτές προτάσεις ως ορθές και τις γραμματικά μη αποδεκτές προτάσεις ως λανθασμένες. Όσον αφορά την αρχική υπόθεση, τα αποτελέσματα των δοκιμών την επιβεβαιώνουν ως έναν βαθμό, διότι το 34,5% των προτάσεων που εισήγαγαν οι ομιλητές δεν ήταν γραμματικά αποδεκτό. Το ποσοστό αυτό δείχνει ότι ένας σημαντικός αριθμός φυσικών ομιλητών της Ελληνικής συγχέει τις περιπτώσεις στις οποίες η εσωτερική αύξηση είναι απαραίτητη με τις περιπτώσεις στις οποίες δεν είναι απαραίτητη.

Πέρα από την εξέταση της αρχικής υπόθεσης, το εν λόγω υπολογιστικό μοντέλο αναπτύχθηκε και για εκπαιδευτικούς σκοπούς. Για την ακρίβεια, θα μπορούσε να χρησιμοποιηθεί από μαθητές σε εκπαιδευτικά πλαίσια ή από ενήλικες που μαθαίνουν την Ελληνική ως ξένη γλώσσα, προκειμένου να τους βοηθήσει να διακρίνουν τις περιπτώσεις στις οποίες η εσωτερική αύξηση θα πρέπει να περιληφθεί και τις περιπτώσεις στις οποίες θα πρέπει να παραλειφθεί.

Λέξεις Κλειδιά: Υπολογιστική Γλωσσολογία, Επεξεργασία Φυσικής Γλώσσας, Υπολογιστικό Μοντέλο, Εσωτερική Αύξηση, Σύνθετα Ρήματα, Προθέσεις, Προστακτική, Εκπαίδευση, Python

Abstract

This dissertation focuses on the internal augment of Greek verbs with prepositional prefixes. The main hypothesis is that native speakers of Greek confuse, to a great extent, the cases in which the internal augment is necessary with the cases in which it is not.

In order to test the above hypothesis, a computational model was developed using the Python programming language. This model asks the user to input a sentence which contains a verb with prepositional prefix in one of the following verb forms: either the third person singular of the aorist indicative or the second person singular of the aorist imperative. Afterwards, the model informs the user whether their sentence is grammatically acceptable or not. If it is not, the model provides the user with the correct sentence by replacing the verb form with the correct one (it either removes or adds the internal augment).

Following its development, the model was tested with 20 native speakers of Greek, each of which input 10 sentences to the program giving a total of 200 sentences. The model was successful at a percentage of 73% by presenting the grammatically acceptable sentences as correct and by presenting the grammatically unacceptable sentences as wrong. As for the initial hypothesis, the results support it to some extent since 34,5% of the sentences that the users input was grammatically unacceptable, which shows that there is a significant number of native speakers of Greek who do confuse the cases in which the internal augment is necessary with the cases in which it is not.

Apart from testing the initial hypothesis, the computational model was also developed as a tool to be used for educational purposes. More specifically, it can be used by children in the school environment or by adults who are learning Greek as a foreign language in order to help them distinguish the cases in which the internal augment should be included and the cases in which it should not.

Keywords: Computational Linguistics, Natural Language Processing, Computational Model, Internal Augment, Prepositional Prefixes, Imperative, Education, Python

Περιεχόμενα

1. Εισαγωγή

Η παρούσα διπλωματική εργασία αφορά την εσωτερική αύξηση σε προθηματοποιημένους ρηματικούς τύπους (σύνθετα ρήματα με προθέσεις), εστιάζοντας αφενός στο τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου της ενεργητικής φωνής και αφετέρου στο δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου της ενεργητικής φωνής.

Βασική αφορμή για την πραγματοποίηση της συγκεκριμένης μελέτης αποτέλεσε η υπόθεση ότι υψηλό ποσοστό των φυσικών ομιλητών της Ελληνικής συγχέει τους παραπάνω τύπους, διότι λανθασμένα παραλείπουν την εσωτερική αύξηση στο τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου της ενεργητικής φωνής είτε περιλαμβάνουν την εσωτερική αύξηση στο δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου της ενεργητικής φωνής.

Για να ελεγχθεί η παραπάνω υπόθεση αναπτύχθηκε ένα υπολογιστικό μοντέλο, με τη χρήση της γλώσσας προγραμματισμού Python, το οποίο αφορά την εσωτερική αύξηση των σύνθετων ρημάτων με προθέσεις και το οποίο εστιάζει αποκλειστικά στο τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου στην ενεργητική φωνή και στο δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή.

Στη συνέχεια, πραγματοποιήθηκαν δοκιμές του προγράμματος με φυσικούς ομιλητές της Ελληνικής, ώστε να ελεγχθεί η ορθότητα της αρχικής υπόθεσης, καθώς και να υπολογιστούν τα ποσοστά επιτυχίας και αποτυχίας του προγράμματος.

Το βασικό κομμάτι λοιπόν της εργασίας είναι το εν λόγω υπολογιστικό μοντέλο, το οποίο αναπτύχθηκε με δύο βασικούς στόχους. Ο πρώτος στόχος είναι η συμβολή στη μοντελοποίηση της γλώσσας, στην προσπάθεια δηλαδή ανάπτυξης συστημάτων που δίνουν στον υπολογιστή τη δυνατότητα να επιδεικνύει (αν και σε περιορισμένο βαθμό στην προκειμένη περίπτωση) γλωσσική συμπεριφορά παρόμοια με του ανθρώπου. Ο δεύτερος στόχος είναι η αξιοποίηση του συγκεκριμένου μοντέλου για εκπαιδευτικούς λόγους. Συγκεκριμένα, το μοντέλο αυτό θα μπορούσε να χρησιμοποιηθεί αφενός σε σχολικό πλαίσιο και αφετέρου από φυσικούς ομιλητές της Ελληνικής ή άτομα που μαθαίνουν την Ελληνική ως ξένη γλώσσα, προκειμένου να ελέγξουν εάν σε κάθε περίπτωση είναι απαραίτητη η παρουσία της εσωτερικής αύξησης ή όχι.

1.1 Θεωρητικό πλαίσιο

1.1.1 Αύξηση

«Η αύξηση στη νέα ελληνική αποτελεί ένα μόρφημα που δηλώνει την έννοια του παρελθόντος» (Χατζησαββίδης και Χατζησαββίδου, 2011: 97) και συναντάται αποκλειστικά στα ρήματα. Συγκεκριμένα, σύμφωνα με τους Χατζησαββίδη και Χατζησαββίδου (2011: 77):

> **Αύξηση** ονομάζεται ένα μέρος του μορφολογικού τύπου του ρήματος, που μπαίνει πριν από το θέμα στον παρατατικό και στον αόριστο οριστικής και με το οποίο δηλώνεται ότι η πράξη που εκφράζει το ρήμα έγινε στο παρελθόν, π.χ. *λέω* → ***έ****λεγα*, *καταγράφω* → *κατ****έ****γραψα*. Το μόρφημα της αύξησης μπορεί να πραγματώνεται με το *ε* ή, σπάνια, με το *η*, π.χ. *έ-παιζα, ή-ξερα*.

Επομένως, βλέπουμε ότι η αύξηση συναντάται αποκλειστικά στον παρατατικό και στον αόριστο και μόνο στην οριστική έγκλιση. Ακόμη, τα ρήματα που διαθέτουν αύξηση χαρακτηρίζονται ως αυξημένοι τύποι (βλ. Τριανταφυλλίδης, 1941: 154).

Ακόμη, στον παραπάνω ορισμό αναφέρεται ότι το μόρφημα της αύξησης δεν πραγματώνεται πάντοτε με τον ίδιο τρόπο. Συγκεκριμένα, όταν το θέμα του ρήματος ξεκινάει από σύμφωνο, πρόκειται για την περίπτωση της *συλλαβικής αύξησης*, επομένως η ρηματική αύξηση πραγματώνεται με το *ε*. Αντιθέτως, όταν το θέμα του ρήματος ξεκινάει από φωνήεν, πρόκειται για την περίπτωση της *φωνηεντικής αύξησης* οπότε και πραγματώνεται με το *η* (βλ. Mackridge, 1985: 182-183).

Επιπλέον, «η αύξηση διακρίνεται σε δύο είδη: σε εξωτερική και εσωτερική. Εξωτερική είναι η αύξηση που μπαίνει στην αρχή του ρηματικού τύπου, ενώ εσωτερική στο εσωτερικό του ρηματικού τύπου των σύνθετων ρημάτων και πάντα πριν από το δεύτερο συνθετικό» (Χατζησαββίδης και Χατζησαββίδου, 2011: 77). Η πλειοψηφία των περιπτώσεων εσωτερικής αύξησης αφορά ρήματα που διαθέτουν ως πρόθημα[1] κάποια από τις προθέσεις της Αρχαίας Ελληνικής. Μάλιστα, σε ορισμένα από τα ρήματα αυτά μπορεί να εντοπίζονται περισσότερες από μία προθέσεις (Mackridge, 1985: 184-185). Στην περίπτωση αυτή, η

[1] Στην πραγματικότητα, όσον αφορά τις προθέσεις της Αρχαίας Ελληνικής δεν είναι ξεκάθαρο εάν αποτελούν πρόθημα ή μέρος του θέματος (βλ. Κουτσούκος, 2010: 179).

αύξηση εντοπίζεται ανάμεσα στη δεύτερη πρόθεση και το ρήμα, παραδείγματος χάρη, συμ-περι-λαμβάνω → συμ-περι-**έ**-λαβα (Φιλιππάκη-Warburton κ.ά., 2011: 137).

Για να γίνει πιο ξεκάθαρη η έννοια της εσωτερικής αύξησης, αξίζει να αναφέρουμε και το ακόλουθο: «Στα ρήματα που έχουν ως πρώτο συνθετικό τους μια πρόθεση, βάζουμε την αύξηση (όταν χρειάζεται) ανάμεσα στην πρόθεση και το ρήμα. Αυτή η αύξηση λέγεται **εσωτερική αύξηση**» (Φιλιππάκη-Warburton κ.ά., 2011). Ως παραδείγματα εξωτερικής και εσωτερικής αύξησης, θα μπορούσαμε να παραθέσουμε τα ακόλουθα:

1) λύνω → **έ**-λυνα (εξωτερική)
2) επιλέγω → επ**έ**λεγα (εσωτερική)
(Φιλιππάκη-Warburton κ.ά., 2011)

Η εσωτερική αύξηση με τη σειρά της μπορεί να είναι συλλαβική ή φωνηεντική εάν το θέμα του ρήματος ξεκινάει από σύμφωνο ή φωνήεν αντίστοιχα (Mackridge, 1985: 185).

Στο σημείο αυτό, αξίζει να αναφερθεί ότι ο Κουτσούκος (2010), εξετάζοντας τη σχέση μεταξύ των διαδικασιών της Κλίσης και της Παραγωγής, καταλήγει σε ενδιαφέροντα συμπεράσματα σχετικά με την εσωτερική αύξηση. Για την ακρίβεια, εξετάζοντας προθηματοποιημένους ρηματικούς τύπους και εστιάζοντας στα σημασιολογικά χαρακτηριστικά της ρηματικής βάσης και των προθημάτων (προθέσεις της Αρχαίας Ελληνικής) συμπεραίνει ότι η εσωτερική αύξηση συναντάται σε σχηματισμούς που εμφανίζεται η εσωτερική πλευρά του προθήματος ή σε σχηματισμούς που η ρηματική βάση διαθέτει το χαρακτηριστικό [+ΛΟΓΙΟ] (βλ. και στην ενότητα 1.1.2). Στα εν λόγω συμπεράσματα καταλήγει εξετάζοντας τα προθήματα *απο-* και *παρα-*.

1.1.2 Θεωρητικές ερμηνείες σχετικά με την αύξηση

Όσον αφορά τη γραμματική υπόσταση της αύξησης, έχουν διατυπωθεί διαφορετικές θεωρητικές προσεγγίσεις. Αρχικά, οι Joseph & Janda (1988: 198-199), βασιζόμενοι στην Kaisse (1982), αναφέρουν ότι η αύξηση αποτελεί φωνολογικό στοιχείο που εντοπίζεται μόνο όταν τονίζεται. Για την ακρίβεια, πρόκειται για σύμβολο υποκατάστασης (placeholder) σε παρελθοντικούς ρηματικούς τύπους στους οποίους είναι αναγκαίος ο τονισμός στην προπαραλήγουσα (antepenultimate). Αφορά δηλαδή ρηματικούς τύπους των οποίων το θέμα δε διαθέτει επαρκή αριθμό συλλαβών για να τονιστεί η προπαραλήγουσα. Το κενό αυτό λοιπόν καλύπτει η αύξηση υπό τη μορφή επένθεσης. Ως παράδειγμα αναφέρουν τους τύπους

«έλαβα» και «λάβαμε»· στον τύπο «έλαβα» τοποθετείται η αύξηση για να τονιστεί η προπαραλήγουσα, ενώ στον τύπο «λάβαμε» δεν είναι αναγκαίο, διότι η προπαραλήγουσα μπορεί να τονιστεί χωρίς την παρουσία αύξησης. Οι Joseph & Janda καταλήγουν λοιπόν στο συμπέρασμα ότι πρόκειται για περίπτωση φωνημικοποίησης (ή απομορφολογικοποίησης) καθώς το, κατά τα άλλα, μορφολογικό φαινόμενο της αύξησης γίνεται φωνολογικό.

Εντούτοις, στη συνέχεια παρουσιάζουν παραδείγματα που έρχονται σε αντίθεση με την παραπάνω θέση που έχουν υποστηρίξει (1988: 201). Συγκεκριμένα, αναφέρουν ότι υπάρχουν περιπτώσεις ρημάτων με αύξηση που δεν τονίζεται («επρόκειτο»), ή περιπτώσεις ρημάτων με επαρκή αριθμό συλλαβών για να τονιστεί η προπαραλήγουσα, που όμως εξακολουθούν να εμφανίζονται με αύξηση («μετέφρασα») και, συγκεκριμένα, με εσωτερική αύξηση. Σύμφωνα με τους Joseph & Janda, αιτία για τα παραπάνω αποτελεί η καθαρεύουσα και το «υψηλό» επίπεδό της, το οποίο μάλιστα υπαγορεύει τη χρήση των εν λόγω τύπων, οι οποίοι παρουσιάζονται ως συνειδητή κοινωνιογλωσσική επιλογή. Η άποψη αυτή ενισχύεται σε συνδυασμό με τα λεγόμενα του Κουτσούκου (2010: 182-184), ο οποίος υποστηρίζει ότι, όταν η ρηματική βάση διαθέτει το σημασιολογικό χαρακτηριστικό [+ΛΟΓΙΟ], η εσωτερική αύξηση κάνει την εμφάνισή της. Με βάση τα παραπάνω λοιπόν, οι Joseph & Janda καταλήγουν στο συμπέρασμα ότι η αύξηση δεν μπορεί να χαρακτηριστεί εξ ολοκλήρου ως φωνολογικό φαινόμενο.

Η Ράλλη (2005) φαίνεται να συμφωνεί με την αρχική θέση των Joseph & Janda, αφού υποστηρίζει ότι η αύξηση στα Νέα Ελληνικά δεν έχει προθηματικό χαρακτήρα και δε δηλώνει την έννοια του παρελθόντος, καθώς δεν εμφανίζεται υποχρεωτικά σε όλα τα πρόσωπα κατά την κλίση των παρωχημένων χρόνων. Σύμφωνα με τη Ράλλη, η έκφραση του παρελθόντος πραγματώνεται στα ληκτικά κλιτικά επιθήματα που εμφανίζουν συστηματικότητα στον παρατατικό και τον αόριστο. Καταλήγει ότι η αύξηση αποτελεί μορφοφωνολογική επένθεση που συνδέεται με την παρουσία τόνου στην προπαραλήγουσα.

Η άποψη που διατυπώνουν οι Spyropoulos & Revithiadou (2009) για την αύξηση είναι ότι αποτελεί ένα κενό πρόθημα που πραγματώνεται κάτω από ορισμένες συνθήκες ως επένθεση, για να τονιστεί η προπαραλήγουσα σε ρηματικούς τύπους που το θέμα τους αποτελείται από μία μόνο συλλαβή. Επιπλέον, αναφέρουν ότι το κενό αυτό πρόθημα χρησιμοποιείται ως δείκτης του παρελθόντος. Μάλιστα, υποστηρίζουν ότι το κενό πρόθημα της αύξησης που φέρει τόνο βρίσκεται σε συμπληρωματική κατανομή με το υποκατάστατο θέμα *πηρ-* («πήρα») του ρήματος *παίρνω* και με το σχηματιστικό στοιχείο *-ηκ* («μπήκα») του ρήματος *μπαίνω*, διότι επιτελούν την ίδια λειτουργία. Με τον τρόπο αυτό δικαιολογούν την αντιγραμματικότητα των τύπων **έπηρα* και **έμπηκα*. Ακόμη, αναφορικά με ορισμένες

διαλέκτους της Ελληνικής, υποστηρίζουν ότι η χρήση της αύξησης είναι υποχρεωτική όταν τονίζεται και προαιρετική όταν δεν τονίζεται (2009: 12).

Οι Μαλικούτη-Drachman & Drachman απορρίπτουν την υπόθεση ότι η αύξηση δεν έχει μορφολογικό χαρακτήρα και προστίθεται μόνο για να καλύψει την οπισθοχωρητική κίνηση του τόνου στους παρελθοντικούς χρόνους της οριστικής, διότι υπάρχουν ρηματικοί τύποι με παρουσία ατόνου αυξήσεως. Για τον λόγο αυτό, καταλήγουν ότι η αύξηση αποτελεί μορφολογικό στοιχείο του ρήματος στους παρελθοντικούς χρόνους, η παρουσία του οποίου είναι υποχρεωτική όταν τονίζεται και προαιρετική όταν δεν τονίζεται (1992: 86-88, 1993: 149). Επιπλέον επιχείρημα που, αφενός, απορρίπτει την άποψη ότι η αύξηση είναι φορέας τόνου και, αφετέρου, την παρουσιάζει ως μορφολογικό στοιχείο των παρελθοντικών χρόνων της οριστικής είναι το γεγονός ότι, όσον αφορά την προστακτική, ενώ σε ορισμένα ρήματα παρατηρείται οπισθοχωρητική κίνηση του τόνου (καθαρίζω: καθάριζε), στα ρήματα που ο αριθμός των συλλαβών δεν επαρκεί για κίνηση του τόνου, δεν παρατηρείται προσθήκη αύξησης που θα έδινε τον απαραίτητο αριθμό συλλαβών (γράφω: γράφε) (1992: 86-87).

Σε διαφορετική μελέτη (2001: 62-63), οι Drachman & Malikouti-Drachman αναφέρουν για την αύξηση ότι έχει χάσει πλέον την υπόστασή της ως μόρφημα που δηλώνει το παρελθόν, με σημαντική ένδειξη το γεγονός ότι έχει σχεδόν εκλείψει η χρήση της φωνηεντικής ή χρονικής αύξησης. Σύμφωνα με τους Drachman & Malikouti-Drachman, βασικός δείκτης του παρελθόντος είναι η αλλαγή θέσης του τόνου και, συγκεκριμένα, η οπισθοχωρητική κίνησή του. Ακόμη, αναφέρουν ότι, πλέον, η αύξηση είναι άμεσα συνυφασμένη με την προσωδία.

1.1.3 Εσωτερική αύξηση και λανθασμένη χρήση

Στην παρούσα εργασία θα μας απασχολήσει μόνο η περίπτωση της εσωτερικής αύξησης και συγκεκριμένα το τρίτο πρόσωπο ενικού αριθμού της οριστικής αορίστου, στην ενεργητική φωνή (π.χ. *επανέλαβε*). Ο συγκεκριμένος τύπος παρουσιάζει ενδιαφέρον διότι, σύμφωνα με την υπόθεση του γράφοντος, συγχέεται σε υψηλό βαθμό από τους φυσικούς ομιλητές με το δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου, στην ενεργητική φωνή (π.χ. *επανάλαβε*), όταν πρόκειται για σύνθετα ρήματα με προθέσεις.

Συγκεκριμένα, αξίζει να παραθέσουμε τα ακόλουθα παραδείγματα για να γίνει ξεκάθαρο το ζήτημα:

3) Επανάλαβε την πρόταση αργά και καθαρά, αν μπορείς.

4) * Επανέλαβε την πρόταση αργά και καθαρά, αν μπορείς.

5) Η Μαρία επανέλαβε την πρόταση, επειδή της το ζήτησε ο καθηγητής.

6) * Η Μαρία επανάλαβε την πρόταση, επειδή της το ζήτησε ο καθηγητής.

Στο παράδειγμα 3, έχουμε δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου, στην ενεργητική φωνή. Επομένως, ο τύπος *επανάλαβε* δε διαθέτει εσωτερική αύξηση, επειδή βρίσκεται σε έγκλιση προστακτική και, όπως αναφέρθηκε παραπάνω, η αύξηση είναι χαρακτηριστικό της οριστικής.

Στο παράδειγμα 4, ο τύπος της προστακτικής διαθέτει λανθασμένα εσωτερική αύξηση, γεγονός που τον ταυτίζει με το τρίτο πρόσωπο ενικού αριθμού της οριστικής αορίστου, στην ενεργητική φωνή. Όμως η αύξηση δεν αποτελεί χαρακτηριστικό της προστακτικής, επομένως θα πρέπει να απουσιάζει σε κάθε περίπτωση.

Στο παράδειγμα 5, εντοπίζεται ορθή χρήση της εσωτερικής αύξησης, καθώς έχουμε τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου, στην ενεργητική φωνή.

Τέλος, στο παράδειγμα 6 ο τύπος της οριστικής λανθασμένα δεν διαθέτει εσωτερική αύξηση, γεγονός που τον ταυτίζει με το δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου, στην ενεργητική φωνή. Η εσωτερική αύξηση θα έπρεπε όμως να είναι παρούσα στην προκειμένη περίπτωση, διότι έχουμε οριστική έγκλιση.

Επομένως, θα μπορούσαμε να πούμε ότι η (λανθασμένη) παρουσία της εσωτερικής αύξησης στην προστακτική, σε ορισμένες περιπτώσεις, και η (λανθασμένη) απουσία της από την οριστική, σε άλλες περιπτώσεις, προκαλούν τη σύγχυση των δύο τύπων από τους ομιλητές.

Άλλωστε, το συγκεκριμένο ζήτημα έχει ήδη επισημανθεί. Σύμφωνα με τον Mackridge (1985: 186-187), αρκετοί ομιλητές της Ελληνικής αντιμετωπίζουν δυσκολίες αναφορικά με τη χρήση της εσωτερικής αύξησης και, ως εκ τούτου, προσπαθούν να την αποφύγουν με διάφορους τρόπους (λόγου χάρη επιλέγοντας ένα συνώνυμο ρήμα που δεν είναι σύνθετο και, επομένως, δεν τίθεται ζήτημα εσωτερικής αύξησης). Ακόμη, αναφέρει ότι οι περιπτώσεις παράλειψης της εσωτερικής αύξησης, παραδείγματος χάρη η χρήση του τύπου «μετάφρασα» αντί για τον τύπο «μετέφρασα», αντιμετωπίζονται ως λανθασμένες και μη φυσικές από αρκετούς ομιλητές[2]. Μάλιστα, σύμφωνα με τον Mackridge, η επέκταση χρήσης της

[2] Το συγκεκριμένο παράδειγμα αναφέρεται σε συνειδητή επιλογή υποστηρικτών της δημοτικής, την περίοδο που συνυπήρχαν η δημοτική και η καθαρεύουσα.

εσωτερικής αύξησης ακόμη και στην Προστακτική[3], (αν και λανθασμένη) αποτελεί ένδειξη του βαθμού στον οποίο η εσωτερική αύξηση είναι ριζωμένη στη συνείδηση των ομιλητών της Ελληνικής. Για τον λόγο αυτό, αναφέρει ότι σε παραδείγματα όπως: «επέστρεψε τα μπουκάλια παρακαλώ», είναι πιο συχνός ο τύπος με αύξηση (*επέστρεψε*) από τον τύπο χωρίς αύξηση (*επίστρεψε*) που είναι στην πραγματικότητα ο ορθός τύπος.

1.2 Υπολογιστικό πλαίσιο

1.2.1 Υπολογιστικά μοντέλα και Υπολογιστική Γλωσσολογία

Με αφετηρία λοιπόν όσα αναφέρθηκαν παραπάνω, κρίθηκε χρήσιμη η ανάπτυξη ενός υπολογιστικού μοντέλου[4] σχετικά με το μορφολογικό ζήτημα της εσωτερικής αύξησης, καθώς και τη λανθασμένη περίληψη ή παράλειψή της.

Αρχικά, αξίζει να παραθέσουμε τον ορισμό των Hill κ.ά. για τα υπολογιστικά μοντέλα, για τα οποία μας λένε ότι αποτελούν:

> ένα σύνολο υπολογιστικών κωδίκων, οι οποίοι εκτελούνται σε ορισμένο περιβάλλον λογισμικού/υλισμικού και μετατρέπουν ένα σύνολο εισαγόμενων δεδομένων σε ένα σύνολο εξαγόμενων δεδομένων, ενώ, παράλληλα, το εισαγόμενο, το εξαγόμενο και η μετατροπή διαθέτουν κάποια ερμηνεία σε σχέση με φαινόμενα του πραγματικού κόσμου.
> (2001: 3)

Ο παραπάνω ορισμός, διαθέτει αρκετά κοινά στοιχεία με τον ορισμό που μας δίνει ο Μαρκόπουλος (2006: 25) για την υπολογιστική πράξη, συγκεκριμένα λέγοντας ότι: «ταυτίζεται με τη λειτουργία μιας μηχανής, η οποία δέχεται δεδομένα, τα οποία επεξεργάζεται σύμφωνα με συγκεκριμένες εντολές και παράγει ένα αποτέλεσμα». Θα μπορούσαμε να πούμε ότι οι δύο ορισμοί σχετίζονται εφόσον, αυτό που κάνει ένα υπολογιστικό μοντέλο, είναι να εκτελεί υπολογιστικές πράξεις. Επιπλέον, ο Μαρκόπουλος

[3] Για παραδείγματα παρουσίας της εσωτερικής αύξησης στην Προστακτική βλ. και Μαλικούτη-Drachman & Drachman (2001: 63).
[4] Στην παρούσα εργασία, εκτός από τον όρο «υπολογιστικό μοντέλο», χρησιμοποιείται και ο πιο γενικός όρος «πρόγραμμα», όταν αναφερόμαστε στο υπολογιστικό μοντέλο που αναπτύχθηκε για το φαινόμενο της εσωτερικής αύξησης.

επισημαίνει ότι τα δεδομένα που αναλύονται είναι το εισαγόμενο (input), ενώ το αποτέλεσμα της υπολογιστικής πράξης αποτελεί το εξαγόμενο (output) (2006: 25), έννοιες που εντοπίζουμε και στον ορισμό των Hill κ.ά. (2001).

Ως επιπλέον χαρακτηριστικό των υπολογιστικών μοντέλων, σύμφωνα με τον Sun (2008: 2), μπορούμε να αναφέρουμε ότι περιγράφουν τις λεπτομέρειες μιας διαδικασίας μέσω αλγορίθμων (για την έννοια του αλγόριθμου και πως σχετίζεται με την παρούσα εργασία βλ. Ανάλυση).

Κατά την ανάπτυξη υπολογιστικών μοντέλων υπάρχουν δύο προσεγγίσεις. Η πρώτη είναι η προσέγγιση που βασίζεται σε κανόνες (rule-based) και αφορά αλγοριθμική συμβολική επεξεργασία, ενώ η δεύτερη είναι η εμπειρική-στατιστική προσέγγιση (data-oriented) που αφορά αλγοριθμική στατιστική επεξεργασία (βλ. Bangalore & Johnston, 2003). Και οι δύο προσεγγίσεις, παρά τις διαφορές τους, βασίζονται σε χρήση αλγορίθμων. Το υπολογιστικό μοντέλο της παρούσας εργασίας αναπτύχθηκε με βάση την πρώτη προσέγγιση, διότι η επεξεργασία των γλωσσικών δεδομένων γίνεται αποκλειστικά με βάση τις λίστες που έχουν εισαχθεί στο πρόγραμμα. Βέβαια, η πλειοψηφία των γλωσσικών μοντέλων (language models) στον χώρο της επεξεργασίας φυσικής γλώσσας υιοθετούν την εμπειρική-στατιστική προσέγγιση (βλ. Jurafsky & Martin, 2009 και Jozefowicz κ.ά., 2016). Στην παρούσα μελέτη λοιπόν, έχουμε να κάνουμε με ένα υπολογιστικό μοντέλο, το οποίο αναπτύχθηκε έχοντας ως σημείο αναφοράς ένα συγκεκριμένο γλωσσικό φαινόμενο.

Θα μπορούσαμε λοιπόν να πούμε ότι το συγκεκριμένο υπολογιστικό μοντέλο εντάσσεται στο πεδίο της Υπολογιστικής Γλωσσολογίας, η οποία «αποτελεί τη μελέτη των υπολογιστικών συστημάτων για την κατανόηση και την παραγωγή φυσικής γλώσσας» (Grishman, 1986: 4). Επιπλέον, σύμφωνα με τον Mitkov, «η Υπολογιστική Γλωσσολογία είναι ένας διεπιστημονικός κλάδος που ασχολείται με την επεξεργασία της γλώσσας από τους υπολογιστές». (2003: ix).

Με βάση λοιπόν τους δύο παραπάνω ορισμούς, συμπεραίνουμε ότι η φυσική γλώσσα βρίσκεται στο επίκεντρο του πεδίου της Υπολογιστικής Γλωσσολογίας με τη «μοντελοποίηση της ανθρώπινης γλωσσικής χρήσης» (Μαρκόπουλος, 2006) να παίζει πρωταρχικό ρόλο. Συγκεκριμένα, αυτό που επιδιώκεται είναι η «αναπαραγωγή της φυσικής μετάδοσης πληροφοριών μέσω της μοντελοποίησης της παραγωγής, από τη μεριά του ομιλητή, και της κατανόησης, από τη μεριά του ακροατή, σε ένα κατάλληλο είδος υπολογιστή» (Hausser, 2014: xix). Το εγχείρημα αυτό απέχει αρκετά από την επιτυχή ολοκλήρωσή του, ωστόσο εκτιμάται ότι κάθε εφαρμογή που αναπτύσσεται μας φέρνει ολοένα και πιο κοντά στην επίτευξη αυτού του στόχου.

Συγκεκριμένα, για το υπολογιστικό μοντέλο της παρούσας εργασίας, θα μπορούσαμε να πούμε ότι επιδεικνύει γλωσσική συμπεριφορά παρόμοια με του ανθρώπου (σε πολύ περιορισμένο βαθμό, καθώς αφορά ένα συγκεκριμένο γλωσσικό φαινόμενο), εφόσον είναι σε θέση να διακρίνει με υψηλό ποσοστό επιτυχίας (βλ. 3.6.2) σε ποιες περιπτώσεις είναι απαραίτητη η εσωτερική αύξηση στα σύνθετα ρήματα με προθέσεις και σε ποιες όχι. Επομένως, η συμβολή στη μοντελοποίηση της γλώσσας αποτελεί τον πρώτο από τους δύο βασικούς σκοπούς που επιτελεί το συγκεκριμένο υπολογιστικό μοντέλο (για τον δεύτερο σκοπό βλ. 1.2.3).

1.2.2 Επεξεργασία φυσικής γλώσσας με τη χρήση γλωσσών προγραμματισμού

Όπως αναφέρθηκε ήδη, για τους σκοπούς της παρούσας εργασίας αναπτύχθηκε ένα πρόγραμμα με τη χρήσης της γλώσσας προγραμματισμού Python, το οποίο εστιάζει σε ένα συγκεκριμένο γλωσσικό φαινόμενο της Ελληνικής. Παρομοίως, στον χώρο της Υπολογιστικής Γλωσσολογίας έχει αναπτυχθεί πληθώρα συστημάτων επεξεργασίας φυσικής γλώσσας με χρήση διαφόρων γλωσσών προγραμματισμού, είτε για εμπορικούς είτε για πειραματικούς σκοπούς. Στο σημείο αυτό παραθέτουμε ορισμένα παραδείγματα.

Η Κεραμιτζή (2016) ανέπτυξε έναν μορφολογικό αναλυτή, ο οποίος λειτουργεί ως εξής: λαμβάνει ως εισαγόμενο ένα αρχείο κειμένου και συνδέεται με τον μορφολογικό επισημειωτή του ΙΕΛ (Ινστιτούτο Επεξεργασίας του Λόγου). Ακολούθως, ο επισημειωτής του ΙΕΛ χωρίζει το κείμενο σε λεκτικές μονάδες (tokenization), αποδίδει σε κάθε μονάδα ετικέτα μορφολογικής πληροφορίας (PoS tagging) και προσδιορίζει το λήμμα στο οποίο αντιστοιχεί (lemmatization). Στη συνέχεια, το πρόγραμμα λαμβάνει τις επισημειωμένες λεκτικές μονάδες (σε αρχείο XML) από τον επισημειωτή του ΙΕΛ και δημιουργεί ένα λεξικό στο οποίο οι λέξεις ταξινομούνται σε διαφορετικές κατηγορίες με βάση το μέρος του λόγου στο οποίο ανήκουν. Ακόμη, η ταξινόμηση των λέξεων εντός της κάθε κατηγορίας γίνεται αλφαβητικά. Το λεξικό που παράγεται βασίζεται στην πλατφόρμα XLE (Xerox Linguistic Environment), η οποία λειτουργεί έχοντας ως βάση τον γραμματικό φορμαλισμό της Λεξικής Λειτουργικής Γραμματικής (Lexical Functional Grammar). Το εν λόγω πρόγραμμα αναπτύχθηκε με τη χρήση της γλώσσας προγραμματισμού Java.

Οι Παπακίτσος κ.ά. (2016) παρουσιάζουν τη σημασία που έχουν για τον χώρο της Υπολογιστικής Γλωσσολογίας τα συστήματα Παραγωγής Φυσικής Γλώσσας, τα οποία έχουν ως βασική λειτουργία τη μετατροπή ψηφιακά κωδικοποιημένων πληροφοριών σε κείμενα φυσικής γλώσσας που επιτελούν συγκεκριμένους επικοινωνιακούς σκοπούς. Στο επίκεντρο

των συστημάτων αυτών αναφέρουν ότι βρίσκεται η μορφολογική γεννήτρια, η οποία είναι υπεύθυνη για την επεξεργασία των μορφοφωνολογικών στοιχείων της γλώσσας. Υπογραμμίζουν μάλιστα ότι η μορφολογική γεννήτρια είναι ιδιαίτερα σημαντική όταν το σύστημα Παραγωγής Φυσικής Γλώσσας έχει αναπτυχθεί για γλώσσες με πλούσια μορφολογία, όπως είναι η Νέα Ελληνική. Ακόμη, όσον αφορά την αρχιτεκτονική της μορφολογικής γεννήτριας επισημαίνουν ότι «εξαρτάται από τη μορφολογική τυπολογία και τη μορφοτακτική της υποκείμενης γλώσσας» (2016: 96). Ως βασικό παράδειγμα υπολογιστικού συστήματος που ενσωματώνει μορφολογική γεννήτρια παρουσιάζουν το ΣΟΜΕ (Σύστημα Ολοκληρωμένης Μορφολογικής Επεξεργασίας, Παπακίτσος, 2000) το οποίο διαθέτει λεξικό μορφημάτων και έχει αναπτυχθεί σε γλώσσες προγραμματισμού Pascal και C. Κατά τις δοκιμές του σημείωσε ποσοστό επιτυχίας 98% σε κείμενα 1.880.000 λέξεων.

Οι Kotsanis & Maistros (1985) ανέπτυξαν ένα σύστημα μορφολογικής ανάλυσης για τα Νέα Ελληνικά, που περιέχει συνολικά 250 κανόνες. Ονομάζεται «Λεξιφάνης» και η ανάπτυξή του έγινε με τη χρήση της γλώσσας προγραμματισμού Pascal και βασίστηκε στα πεπερασμένα αυτόματα. Σημαντικό πλεονέκτημα είναι το γεγονός ότι μπορεί να χρησιμοποιηθεί ανεξάρτητα ή σε συνδυασμό με άλλα συστήματα επεξεργασίας φυσικής γλώσσας. Η λειτουργία του είναι η ακόλουθη: δέχεται ένα κείμενο ως εισαγόμενο και προσδιορίζει το μέρος του λόγου στο οποίο ανήκει η κάθε λέξη του κειμένου. Ακόμη, μπορεί να ελεγχθεί το συγκείμενο, ώστε να επιλυθούν αρκετές περιπτώσεις αμφισημίας. Τα ποσοστά επιτυχίας του κυμαίνονται μεταξύ 95% και 98%, ανάλογα με το είδος του κειμένου (παραδείγματος χάρη, εντοπίζονται χαμηλότερα ποσοστά επιτυχίας σε λογοτεχνικά κείμενα και υψηλότερα ποσοστά επιτυχίας σε επιστημονικά κείμενα). Μια από τις πιθανές χρήσεις του «Λεξιφάνη» είναι στο πεδίο της Λεξικογραφίας για την παραγωγή λημμάτων.

Το σύστημα «Quicklem» (βλ. Kotsanis κ.ά., 1987) έχει αναπτυχθεί με τη χρήση της γλώσσας προγραμματισμού Lisp και διαθέτει αλγόριθμο που μπορεί να επεξεργαστεί οποιοδήποτε κείμενο, αφού πρώτα έχει μετατραπεί σε λίστα με καθεμία από τις λέξεις του. Βασική λειτουργία του «Quicklem» είναι να προσδιορίζει τη γραμματική κατηγορία κάθε λέξης και να παρέχει επίσης τα κλιτικά επιθήματα της κάθε λέξης. Ακόμη, για κάθε κλιτικό επίθημα δίνει πληροφορία σχετικά με τη συλλαβή που τονίζεται (λήγουσα, παραλήγουσα, προπαραλήγουσα) όταν αυτό συνδυάζεται με το θέμα. Το «Quicklem» υιοθετεί κάθετη ανάλυση για την επίλυση αμφισημιών και όχι τη συμβατική οριζόντια ανάλυση (βάσει προσφύματος ή μορφήματος) που χρησιμοποιείται συχνά για γλώσσες με πλούσια κλιτική μορφολογία. Η εν λόγω κάθετη ανάλυση ομαδοποιεί τις λέξεις που εισάγονται στο σύστημα σε διαφορετικές κλάσεις ανάλογα με το θέμα τους και για κάθε κλάση συλλέγει όλες τις

διαφορετικές καταλήξεις. Στη συνέχεια, συγκρίνει τις καταλήξεις αυτές με τις πιθανές καταλήξεις που διαθέτει στη βάση δεδομένων του για κάθε κλάση. Με τον τρόπο αυτό αποδίδεται η γραμματική κατηγορία. Ακόμη, το σύστημα διαθέτει και λίστα με λειτουργικές λέξεις και άκλιτα μέρη του λόγου (*stopwords*) που αποτελείται από 700 συνολικά λέξεις. Κατά την επεξεργασία του κειμένου, όσες από αυτές τις λέξεις εντοπίζονται, αφαιρούνται από το κείμενο. Το λεξικό του προγράμματος αποτελείται από συνολικά 20.000 λέξεις. Όσον αφορά τα ποσοστά επιτυχίας, στο 77% των λέξεων αποδίδεται σωστή γραμματική κατηγορία, στο 3% λανθασμένη, ενώ ένα ποσοστό της τάξης του 20% δεν λαμβάνει γραμματική κατηγορία. Να σημειωθεί ότι το «Quicklem» έχει σχεδιαστεί ώστε το εξαγόμενό του να αξιοποιείται από έναν άλλο μορφολογικό αναλυτή της Νέας Ελληνικής που έχουν αναπτύξει οι Kotsanis & Maistros (1991), ο οποίος βασίζεται σε φορμαλισμό ενοποιητικής γραμματικής.

1.2.3 Υπολογιστικά μοντέλα και εκπαίδευση

Εκτός από τους σκοπούς μοντελοποίησης της φυσικής γλώσσας που μπορεί να εξυπηρετεί ένα υπολογιστικό μοντέλο γλωσσικής φύσεως, σημαντική μπορεί να είναι και η χρήση του για εκπαιδευτικούς λόγους. Σύμφωνα με τους Hill κ.ά. (2001: 2) τα «υπολογιστικά μοντέλα αναπτύσσονται για να υποστηρίξουν την έρευνα, την ανάπτυξη και την εκπαίδευση σε όλους τους τομείς».

Συγκεκριμένα, είτε πρόκειται για φυσικούς ομιλητές της Ελληνικής, είτε για άτομα που μαθαίνουν την Ελληνική ως ξένη γλώσσα, όσον αφορά το φαινόμενο της εσωτερικής αύξησης στα σύνθετα ρήματα με προθέσεις, είναι δυνατή η χρήση του εν λόγω υπολογιστικού μοντέλου για εισαγωγή προτάσεων από τον χρήστη, ώστε να γίνει έλεγχος των περιπτώσεων στις οποίες η εσωτερική αύξηση είναι απαραίτητη και των περιπτώσεων στις οποίες δεν είναι απαραίτητη[5]. Η χρήση για εκπαιδευτικούς σκοπούς που περιγράφεται στο σημείο αυτό, θα μπορούσε να λάβει χώρα είτε στο πλαίσιο σχολικής τάξης, είτε ατομικά από έναν χρήστη που αντιμετωπίζει δυσκολίες σχετικά με το φαινόμενο της εσωτερικής αύξησης.

Στο σημείο αυτό, αξίζει να δούμε ορισμένα παραδείγματα προγραμμάτων που έχουν αναπτυχθεί κυρίως για εκπαιδευτικούς σκοπούς. Οι Μπαλτζής κ.ά. (2006) ανέπτυξαν ένα υπολογιστικό λεξικό για τα Νέα Ελληνικά, το οποίο αποτελείται από 4 επιμέρους λεξικά που

[5] Για να γίνει πιο κατανοητή η διαδικασία που περιγράφεται, βλ. 3.3.2 όπου αναλύεται ο τρόπος λειτουργίας του προγράμματος.

το καθένα εστιάζει σε διαφορετικό επίπεδο ανάλυσης της γλώσσας. Το πρώτο μέρος είναι το μορφολογικό λεξικό, του οποίου βασική λειτουργία είναι η αναγνώριση και παραγωγή μορφολογικών τύπων (είτε μονολεκτικών, είτε περιφραστικών) της Νέας Ελληνικής, χωρίς να υπάρχει περιορισμός στην πολυπλοκότητα των δεδομένων που του παρέχονται. Μάλιστα, οι Μπαλτζής κ.ά. αναφέρουν ότι το μορφολογικό λεξικό σημειώνει ποσοστό επιτυχίας 100%. Το μορφολογικό λεξικό μπορεί να χρησιμοποιηθεί ξεχωριστά ή σε συνδυασμό με τα υπόλοιπα υπολογιστικά λεξικά που έχουν αναπτυχθεί. Ορισμένες ενδεικτικές χρήσεις του είναι ο έλεγχος συμφωνίας μεταξύ λέξεων (παραδείγματος χάρη άρθρου και ουσιαστικού), καθώς και η αναγνώριση ενός τύπου και στη συνέχεια, εάν το επιθυμεί ο χρήστης, η κλίση του εν λόγω τύπου. Αξίζει να σημειωθεί ότι οι συγκεκριμένες διαδικασίες πραγματοποιούνται σε πολύ μικρό χρονικό διάστημα. Σημαντικό πλεονέκτημα του λεξικού είναι ότι, σε περίπτωση που εισαχθεί ανορθόγραφη λέξη, έχει τη δυνατότητα να εντοπίσει και να παρουσιάσει πιθανές σωστές εκδοχές της λέξης που επιθυμούσε να εισαγάγει ο χρήστης. Ακόμη, οι Μπαλτζής κ.ά. δίνουν έμφαση στον τρόπο με τον οποίο έχει αναπτυχθεί το μορφολογικό λεξικό, χάρη στον οποίο οι αλγόριθμοι είναι ανεξάρτητοι από τα δεδομένα και, ως εκ τούτου, οι μορφολογικοί τύποι των λέξεων παράγονται σε πραγματικό χρόνο και δεν ανακαλούνται απλώς από μια βάση δεδομένων στην οποία είναι ήδη σχηματισμένοι. Επιπλέον, ο συγκεκριμένος τρόπος ανάπτυξης καθιστά ακόμη πιο εύκολη τη διαδικασία εμπλουτισμού του λεξικού με νέα δεδομένα.

Η Σώρρα (2014) ανέπτυξε ένα σύστημα μορφολογικής ανάλυσης για τα ονόματα της Αρχαίας Ελληνικής. Η λειτουργία του προγράμματος είναι η ακόλουθη: ο χρήστης εισάγει ουσιαστικό των Αρχαίων Ελληνικών σε ονομαστική ενικού και, στη συνέχεια, το πρόγραμμα αναγνωρίζει την κλίση του (Α, Β, Γ) και την επιστρέφει στον χρήστη μαζί με το ουσιαστικό το οποίο κλίνει σε ενικό και πληθυντικό. Ο συγκεκριμένος μορφολογικός αναλυτής έχει αναπτυχθεί σε γλώσσα προγραμματισμού Python και αποτελείται από γραμματικούς κανόνες, αλλά και κανόνες τονισμού για να μπορεί να αντεπεξέλθει στο πολυτονικό σύστημα της Αρχαίας Ελληνικής. Η χρήση του μορφολογικού αναλυτή καθίσταται ακόμη πιο εύκολη διότι έχει αναπτυχθεί γραφικό περιβάλλον χρήστη (graphical user interface), το οποίο καθιστά το πρόγραμμα πιο φιλικό προς τον χρήστη και του δίνει τη δυνατότητα να το χρησιμοποιήσει χωρίς να απαιτείται εξοικείωση με γλώσσες προγραμματισμού. Ένας από τους βασικούς λόγους ανάπτυξης του μορφολογικού αναλυτή είναι η ενίσχυση του μαθήματος των Αρχαίων Ελληνικών στο πλαίσιο της σχολικής τάξης. Η Σώρρα αναφέρει ότι, με βάση τον συγκεκριμένο μορφολογικό αναλυτή, μπορεί να αναπτυχθεί ένα σύστημα

ανάλυσης που θα καλύπτει όλα τα μέρη του λόγου της Αρχαίας Ελληνικής και θα είναι σε θέση να ενισχύσει την εκπαιδευτική διαδικασία τόσο στο σχολείο, όσο και στο σπίτι.

Η Tzevelekou (2000) σχεδίασε ένα σύστημα που ενισχύει τη διδασκαλία της Ελληνικής ως δεύτερης γλώσσας σε δημοτικά σχολεία, σε μειονότητες της Θράκης που έχουν ως μητρική γλώσσα την Τουρκική. Το γλωσσικό υλικό του προγράμματος βασίστηκε σε έρευνα που πραγματοποιήθηκε, προκειμένου να εντοπιστούν τα πιο συχνά λάθη που κάνουν τα συγκεκριμένα παιδιά όταν χρησιμοποιούν την Ελληνική. Ακόμη, για την εύρεση υλικού που να ανταποκρίνεται στις ανάγκες της εν λόγω μελέτης αξιοποιήθηκε επισημειωμένο σώμα κειμένων του ΙΕΛ, κυρίως λόγω της πληθώρας κριτηρίων αναζήτησης που προσφέρει. Για να είναι πιο ενδιαφέρον για τα παιδιά, το σύστημα έχει τη μορφή ιστορίας που αποτελείται από 20 επεισόδια. Τα παιδιά πρέπει να ολοκληρώσουν κάθε επίπεδο για να φτάσουν στο τέλος της ιστορίας. Κάθε επίπεδο εστιάζει σε συγκεκριμένα φαινόμενα της Ελληνικής και ενσωματώνει ασκήσεις. Το σύστημα αποτελείται από κείμενα στην Ελληνική, τα οποία συνοδεύονται από τη μετάφραση στην Τουρκική, ασκήσεις που εστιάζουν σε διαφορετικά επίπεδα ανάλυσης της γλώσσας, θεωρία γραμματικής και δίγλωσσο (Ελληνοτουρκικό) λεξικό με 4000 καταχωρήσεις. Ως σημαντικό πλεονέκτημα παρουσιάζεται το γεγονός ότι, με τη χρήση του συστήματος αυτού, τα παιδιά δουλεύουν με τους δικούς τους ρυθμούς, χωρίς πίεση και αντιμετωπίζουν το κάθε γλωσσικό φαινόμενο με τη σειρά που επιθυμούν, σε αντίθεση με το έντυπο βιβλίο που διαθέτει γραμμική πορεία.

2. Δεδομένα - Μεθοδολογία

Προκειμένου να ελεγχθεί η αρχική υπόθεση αναφορικά με τη λανθασμένη χρήση της εσωτερικής αύξησης, πραγματοποιήθηκε αναζήτηση στο Σώμα Ελληνικών Κειμένων[6] (Γούτσος, 2003) για τους τύπους «μετέφερε» και «μετάφερε» του ρήματος «μεταφέρω».

Στη συνέχεια, κατασκευάστηκαν λίστες που περιέχουν τα υπό εξέταση ρήματα σε τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου στην ενεργητική φωνή (παραδείγματος χάρη «επανέλαβε») καθώς και σε δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή (παραδείγματος χάρη «επανάλαβε»), ώστε να είναι διαθέσιμος τόσο ο τύπος με εσωτερική αύξηση όσο και ο τύπος χωρίς εσωτερική αύξηση που συγχέονται από τους ομιλητές.

Για τη συλλογή των ρημάτων χρησιμοποιήθηκε το Λεξικό της Νέας Ελληνικής Γλώσσας (Μπαμπινιώτης, 1998), όμως αρκετοί τύποι προστίθεντο με την πάροδο του χρόνου, καθώς εντοπίζονταν κατά τη χρήση του προγράμματος και τις δοκιμές του με συμμετέχοντες. Επιπλέον, για τον εμπλουτισμό των λιστών, κάθε φορά που εντοπίζονταν νέα ρήματα, γινόταν έλεγχος συμβατότητας του δεύτερου συνθετικού τους με κάποια άλλη πρόθεση. Λόγου χάρη, κατά τον εντοπισμό του ρήματος «απορρίπτω», γινόταν έλεγχος συμβατότητας του «ρίπτω» με προθέσεις πέρα από το «από», γεγονός που επέτρεπε τον εντοπισμό των ρημάτων «επιρρίπτω» και «καταρρίπτω».[7]

Οι λίστες ρημάτων που αξιοποιήθηκαν για την παρούσα εργασία δεν είναι εξαντλητικές. Ωστόσο, όπως αποδείχτηκε στις δοκιμές του προγράμματος με συμμετέχοντες (βλ. Ανάλυση), οι λίστες περιέχουν σημαντικό αριθμό σύνθετων ρημάτων με προθέσεις, ώστε να επιτρέπουν στο πρόγραμμα να αντεπεξέλθει στην πλειοψηφία των προτάσεων που εισάγουν οι χρήστες.

Επίσης, όπως αναφέρθηκε και παραπάνω, έπειτα από τις δοκιμές του προγράμματος με συμμετέχοντες, οι λίστες εμπλουτίστηκαν καθώς, όσα ρήματα των προτάσεων που εισήχθησαν από τους συμμετέχοντες απουσίαζαν από τις λίστες του προγράμματος, εντοπίστηκαν και εισήχθησαν στο πρόγραμμα.

[6] http://www.sek.edu.gr/.
[7] Ακόμη, να σημειωθεί ότι έγινε εκτενής χρήση της ιστοσελίδας https://www.lexigram.gr/ για να ελεγχθεί η ορθή διατύπωση ορισμένων τύπων ρημάτων.

Το πρόγραμμα της παρούσας εργασίας αναπτύχθηκε σε γλώσσα προγραμματισμού Python (έκδοση 3.5) που διανέμεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation (PSF).[8]

Έπειτα από την ολοκλήρωση του προγράμματος[9] κρίθηκε αναγκαία η δοκιμή του, προκειμένου να εξεταστούν τα ποσοστά επιτυχίας του αλλά και ο βαθμός στον οποίο γίνεται ορθή χρήση της εσωτερικής αύξησης από τους ομιλητές της Ελληνικής. Οι δοκιμές πραγματοποιήθηκαν με 20 συμμετέχοντες συνολικά, 10 άνδρες και 10 γυναίκες. Οι ηλικίες των συμμετεχόντων κυμαίνονται από τα 23 μέχρι και τα 29 έτη και όλοι είναι φυσικοί ομιλητές της ελληνικής γλώσσας. Ορισμένοι από τους συμμετέχοντες είναι απόφοιτοι της Τριτοβάθμιας εκπαίδευσης, ενώ ορισμένοι φοιτούν την τρέχουσα περίοδο σε κάποιο τμήμα Τριτοβάθμιας εκπαίδευσης. Κανένας από τους συμμετέχοντες δεν είναι απόφοιτος ή φοιτά την τρέχουσα περίοδο σε κάποιο τμήμα Φιλολογίας. Η συγκεκριμένη επιλογή ήταν σκόπιμη και όχι τυχαία και οφείλεται στο γεγονός ότι, για τις δοκιμές του προγράμματος, κρίθηκε αναγκαία η επιλογή ατόμων που είναι φυσικοί ομιλητές της Ελληνικής, αλλά δεν έχουν τη γλώσσα ως βασικό αντικείμενο των σπουδών τους. Εκτιμήθηκε λοιπόν ότι με τον τρόπο αυτό, θα οδηγούμασταν σε πιο ασφαλή συμπεράσματα με τις δοκιμές του προγράμματος (βλ. Ανάλυση και Συμπεράσματα-Συζήτηση), εφόσον τα εν λόγω συμπεράσματα θα αφορούσαν τον μέσο ομιλητή και όχι άτομα με εξειδίκευση στη γλώσσα.

Στην πειραματική φάση εισήχθησαν στο πρόγραμμα 10 προτάσεις κατά ομιλητή. Επομένως ο συνολικός αριθμός προτάσεων ανέρχεται στις 200 (20 συμμετέχοντες, 10 προτάσεις ο καθένας). Με βάση αυτές τις 200 προτάσεις, προέκυψαν αφενός τα ποσοστά επιτυχίας και αποτυχίας του προγράμματος και αφετέρου τα ποσοστά ορθής και λανθασμένης χρήσης της εσωτερικής αύξησης από τους ομιλητές.

Κατά το στάδιο των δοκιμών, κάθε συμμετέχων εισήγαγε τις προτάσεις στο πρόγραμμα χωρίς να γνωρίζει ότι εξετάζεται και η ορθότητα των επιλογών του, πέρα από τα ποσοστά επιτυχίας του προγράμματος. Η προαναφερθείσα απόκρυψη ήταν αναγκαία ώστε να διασφαλιστεί ότι οι συμμετέχοντες δεν θα επηρεάζονταν και δεν θα μετέβαλαν τις επιλογές τους, αλλά ότι θα εισήγαγαν προτάσεις διατυπωμένες με τον τρόπο που θα τις εκφωνούσαν και στον καθημερινό λόγο τους. Μόλις ολοκληρωνόταν η εισαγωγή των προτάσεων, κάθε

[8] Διαθέσιμη στην ιστοσελίδα https://www.python.org/. Επιπλέον, χρησιμοποιήθηκε το ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment) PyCharm που διανέμεται από την εταιρεία JetBrains στην ιστοσελίδα https://www.jetbrains.com/pycharm/.

[9] Αξίζει να σημειωθεί ότι η πραγματοποίηση της παρούσας εργασίας θα ήταν αναμφίβολα αδύνατη χωρίς την ιστοσελίδα https://www.codecademy.com/, η οποία προσφέρει τη δυνατότητα εκμάθησης μιας πληθώρας γλωσσών προγραμματισμού μέσω διαδικτύου. Συγκεκριμένα, για τους σκοπούς της παρούσας εργασίας, η εκμάθηση της γλώσσας προγραμματισμού Python έγινε εξ ολοκλήρου μέσω της εν λόγω ιστοσελίδας.

συμμετέχων ενημερωνόταν σχετικά με τους σκοπούς των δοκιμών και, σε περίπτωση που είχε οποιαδήποτε αντίρρηση, οι προτάσεις του αφαιρούνταν από τα δεδομένα της παρούσας εργασίας.

Οι δοκιμές του προγράμματος έλαβαν χώρα στην Αθήνα από τον Δεκέμβριο του 2016 μέχρι και τον Ιούλιο του 2017.

3. Ανάλυση

3.1 Έλεγχος σε σώματα κειμένων

Όπως αναφέρθηκε και στο κομμάτι της μεθοδολογίας, πραγματοποιήθηκε έλεγχος σχετικά με την αρχική υπόθεση (συγκεκριμένα ότι υψηλό ποσοστό των φυσικών ομιλητών της Ελληνικής κάνουν συστηματικά λάθη αναφορικά με τη χρήση της εσωτερικής αύξησης) στο Σώμα Ελληνικών Κειμένων. Βασικός σκοπός ήταν να εξεταστεί εάν όντως θα εντοπίζονταν παραδείγματα λανθασμένης χρήσης της εσωτερικής αύξησης, γεγονός που θα ενίσχυε την αρχική υπόθεση. Ο έλεγχος που πραγματοποιήθηκε δεν ήταν εκτενής, καθώς βασικός άξονας της παρούσας εργασίας δεν είναι η εξέταση του φαινομένου της εσωτερικής αύξησης σε σώματα κειμένων, αλλά η ανάπτυξη ενός υπολογιστικού μοντέλου για το φαινόμενο αυτό. Στην πραγματικότητα, ο έλεγχος σε σώματα κειμένων ήταν προκαταρκτικός, προκειμένου να εξαχθούν ορισμένα βασικά συμπεράσματα πριν από την ανάπτυξη του υπολογιστικού μοντέλου.

Το ρήμα που χρησιμοποιήθηκε για τον έλεγχο στο ΣΕΚ ήταν το «μεταφέρω». Συγκεκριμένα, πραγματοποιήθηκε αναζήτηση για τους τύπους «μετέφερε» (τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου, ενεργητική φωνή) και «μετάφερε» (δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου, ενεργητική φωνή), προκειμένου να ελεγχθεί εάν χρησιμοποιούνται λανθασμένα.

Η αναζήτηση για τον τύπο «μετέφερε» επέστρεψε 266 αποτελέσματα, εκ των οποίων όλα ήταν σωστά. Για την ακρίβεια, σε όλες τις περιπτώσεις γινόταν χρήση της οριστικής, επομένως η παρουσία της εσωτερικής αύξησης ήταν ορθή. Σε κανένα από τα αποτελέσματα λοιπόν δεν εντοπίστηκε χρήση προστακτικής, η οποία θα καθιστούσε την περίληψη της εσωτερικής αύξησης λανθασμένη.

Από την άλλη μεριά, η αναζήτηση για τον τύπο «μετάφερε» επέστρεψε 4 αποτελέσματα, εκ των οποίων όλα ήταν λανθασμένα. Συγκεκριμένα, ο τύπος «μετάφερε» βρίσκεται σε προστακτική, γεγονός που δικαιολογεί την απουσία της εσωτερικής αύξησης. Όμως, και τα 4 παραδείγματα που επέστρεψε η αναζήτηση στο ΣΕΚ είναι περιπτώσεις οριστικής, επομένως η απουσία εσωτερικής αύξησης είναι λανθασμένη. Ακολουθεί ένα από τα αποτελέσματα του ΣΕΚ, το οποίο συνοδεύεται από πρόταση που δείχνει πως θα έπρεπε να είναι διατυπωμένο:

7) *...γι' αυτό και ο Μητροπολίτης της Αίνου *μετάφερε* εδώ την έδρα του από το 1890...

8) ...γι' αυτό και ο Μητροπολίτης της Αίνου *μετέφερε* εδώ την έδρα του από το 1890... (ορθή διατύπωση)

Με βάση τα παραπάνω λοιπόν, γίνεται εμφανές ότι εντοπίστηκε μόνο λανθασμένη απουσία της εσωτερικής αύξησης και όχι λανθασμένη περίληψή της. Τα δεδομένα του ΣΕΚ λοιπόν δεν υποστηρίζουν πλήρως την αρχική υπόθεση, μας δείχνουν όμως ότι ισχύει ως έναν βαθμό.

3.2 Ανάπτυξη προγράμματος και αλγόριθμοι

Για τους σκοπούς της παρούσας εργασίας, κρίθηκε αναγκαία η ανάπτυξη ενός προγράμματος. Ένα υπολογιστικό πρόγραμμα είναι προϊόν της διαδικασίας του προγραμματισμού. Ήδη από το 1950, ο Alan Turing μας λέει ότι «προγραμματίζω μια μηχανή για να εκτελέσει μια ενέργεια Α σημαίνει ότι παρέχω στη μηχανή την κατάλληλη σειρά οδηγιών για να κάνει το Α» (Turing, 1950: 5).

Ουσιαστικά, το πρόγραμμα της παρούσας εργασίας βασίζεται σε έναν αλγόριθμο. Σύμφωνα με τους Cormen κ.ά., ως αλγόριθμος ορίζεται «οποιαδήποτε ικανοποιητικά διατυπωμένη υπολογιστική διαδικασία, η οποία λαμβάνει μια τιμή, ή ένα σύνολο τιμών, ως ***εισαγόμενο*** και παράγει κάποια τιμή, ή ένα σύνολο τιμών, ως ***εξαγόμενο***»[10] (2009: 5) και συμπληρώνουν ότι, ως εκ τούτου ο αλγόριθμος αποτελεί «μια ακολουθία υπολογιστικών βημάτων που μετατρέπουν το εισαγόμενο στο εξαγόμενο» (2009: 5). Διαπιστώνουμε λοιπόν ότι οι Cormen κ.ά. δίνουν έμφαση στο γεγονός ότι, μέσω του αλγόριθμου, οδηγούμαστε από το εισαγόμενο στο εξαγόμενο.

Επιπλέον, για να γίνει ακόμη πιο ξεκάθαρη η έννοια του αλγόριθμου, αξίζει να παραθέσουμε ορισμένα βασικά χαρακτηριστικά του και συγκεκριμένα ότι αποτελεί:

- μια πεπερασμένη περιγραφή ενός υπολογισμού στα πλαίσια ικανοποιητικά διατυπωμένων και βασικών διαδικασιών (ή οδηγιών)·

- μια ντετερμινιστική διαδικασία: το επόμενο βήμα (εάν υπάρχει) είναι ρητά διατυπωμένο·

[10] Για τις έννοιες «εισαγόμενο» και «εξαγόμενο» βλ. 1.2.1.

- μια μέθοδο που παράγει ένα αποτέλεσμα σε κάθε περίπτωση, οποιοδήποτε και αν είναι το εισαγόμενο (δηλαδή ο υπολογισμός που περιγράφεται από τον αλγόριθμο ολοκληρώνεται πάντοτε).
(Fernández, 2009: 1-2)

Το πρόγραμμα της παρούσας εργασίας λοιπόν βασίζεται σε έναν αλγόριθμο ο οποίος δέχεται προτάσεις (εισαγόμενο) που περιέχουν σύνθετα ρήματα με προθέσεις σε έναν από τους δύο τύπους που εξετάζονται (τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου στην ενεργητική φωνή ή δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή) και στη συνέχεια, αφού λάβουν χώρα οι καθορισμένες διαδικασίες (βλ. 3.3 για αναλυτική περιγραφή των διαδικασιών του προγράμματος), παρέχει στον χρήστη είτε την αντίστοιχη σωστή πρόταση, σε περίπτωση που ο χρήστης έχει εισαγάγει πρόταση με λανθασμένο τύπο, είτε το μήνυμα ότι η πρόταση που εισήγαγε είναι σωστή (εξαγόμενο).

Όπως αναφέρθηκε και στην εισαγωγή, το μοντέλο της εργασίας υιοθετεί την προσέγγιση που βασίζεται σε κανόνες και όχι τη στατιστική προσέγγιση (για περισσότερες λεπτομέρειες βλ. 1.2.1).

3.3 Αναλυτική περιγραφή του προγράμματος

Προτού προβούμε σε ανάλυση του προγράμματος, αξίζει να παρουσιαστεί ο τρόπος με τον οποίο λειτουργεί. Αρχικά να αναφερθεί ότι ένα πρόγραμμα μπορεί να επεξεργάζεται δεδομένα φυσικής γλώσσας, όμως ο υπολογιστής δεν είναι σε θέση να κατανοήσει τη φυσική γλώσσα καθαυτή (Μαρκόπουλος, 2006: 17). Επομένως, ήταν αναγκαία η εύρεση ενός κριτηρίου, βάσει του οποίου το πρόγραμμα θα διακρίνει σε ποιες περιπτώσεις είναι αναγκαία η παρουσία της εσωτερικής αύξησης και σε ποιες περιπτώσεις δεν είναι. Το κριτήριο που επιλέχθηκε είναι η θέση στην πρόταση.

Συγκεκριμένα, η Ελληνική είναι μια γλώσσα με ελεύθερη σειρά όρων μέσα στην πρόταση. Επομένως, οι θέσεις των λέξεων μέσα στην πρόταση δεν είναι πάντα σταθερές και μπορούν να μεταβάλλονται. Ωστόσο, σύμφωνα με την υπόθεση του γράφοντος, τις περισσότερες φορές που γίνεται χρήση προστακτικής, το ρήμα βρίσκεται στην πρώτη θέση της πρότασης. Αντίστοιχα, τις περισσότερες φορές που γίνεται χρήση οριστικής, το ρήμα εντοπίζεται σε κάποια άλλη θέση πλην της πρώτης, εφόσον συνήθως προηγείται το υποκείμενο. Παραδείγματος χάρη:

9) Μετάδωσε την είδηση σε όλη τη χώρα, εάν θες να είσαι δίκαιος.

10) Σε παρακαλώ, μετάδωσε την εκπομπή την Κυριακή.

11) Ο Αντώνης μετέδωσε το μήνυμα σε όλη την παρέα.

12) Μετέδωσε την ασθένεια και σε άλλα άτομα, καθώς δεν ήταν καθόλου προσεκτικός.

Τα παραδείγματα 9 και 10 περιέχουν το ρήμα «μεταδίδω» στο δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή. Στο παράδειγμα 9 το ρήμα βρίσκεται στην πρώτη θέση της πρότασης, ενώ στο παράδειγμα 10 βρίσκεται στην τρίτη θέση, καθώς προηγείται η φράση «Σε παρακαλώ».

Στα παραδείγματα 11 και 12 εντοπίζεται το ρήμα «μεταδίδω» στο τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου στην ενεργητική φωνή. Στο παράδειγμα 11 το ρήμα βρίσκεται στην τρίτη θέση της πρότασης, μετά από το υποκείμενο και το άρθρο του, ενώ στο παράδειγμα 12 εντοπίζεται στην πρώτη θέση.

Στα παραπάνω παραδείγματα λοιπόν, έχουμε περιπτώσεις που τόσο η προστακτική όσο και η οριστική εντοπίζονται είτε στην πρώτη θέση της πρότασης, είτε σε κάποια άλλη θέση πλην της πρώτης. Ωστόσο, η υπόθεση με βάση την οποία αναπτύχθηκε το πρόγραμμα είναι ότι η προστακτική εμφανίζεται συχνότερα στην αρχή της πρότασης, ενώ η οριστική εμφανίζεται συχνότερα σε θέση διαφορετική από την πρώτη (βλ. 4.1.1.1 για επιβεβαίωση της συγκεκριμένης υπόθεσης).

Ακολουθεί παρουσίαση της αρχικής μορφής του προγράμματος της παρούσας εργασίας, καθώς και της τελικής μορφής που έλαβε στη συνέχεια.

3.3.1 Πρώτη μορφή του προγράμματος

Η αρχική ιδέα αναφορικά με τη λειτουργία του προγράμματος, ήταν η ακόλουθη: Το πρόγραμμα ζητάει από τον χρήστη να εισαγάγει 2 προτάσεις, οι οποίες πρέπει να είναι σχεδόν πανομοιότυπες, με μοναδική διαφορά ότι η μία πρόταση θα περιέχει το ρήμα στον έναν από τους δύο τύπους που εξετάζονται (τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου στην ενεργητική φωνή) και η άλλη πρόταση θα περιέχει το ρήμα στον άλλον τύπο (δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή), χωρίς να υπάρχει περιορισμός αναφορικά με την πρόταση (πρώτη ή δεύτερη) στην οποία θα περιέχεται ο κάθε τύπος. Στη συνέχεια, το πρόγραμμα εκτελεί στο υπόβαθρο τις

καθορισμένες διαδικασίες και πληροφορεί τον χρήστη αναφορικά με το ποια από τις δύο προτάσεις είναι σωστή. Παραδείγματος χάρη:

13) Παρακαλώ εισαγάγετε την πρώτη πρόταση: Η Μαρία απέτρεψε την καταστροφή.
Παρακαλώ εισαγάγετε τη δεύτερη πρόταση: Η Μαρία απότρεψε την καταστροφή.
Η σωστή πρόταση είναι: Η Μαρία απέτρεψε την καταστροφή.

Το πρόγραμμα αναπτύχθηκε αρχικά διαθέτοντας την παραπάνω μορφή. Ωστόσο, μετά από συζήτηση με τον επιβλέποντα, κρίθηκε καταλληλότερος ο μετασχηματισμός του προγράμματος και του τρόπου λειτουργίας του (βλ. 3.3.2).

3.3.1.1 Συμβολοσειρές και Λίστες

Όσον αφορά την ανάπτυξη του προγράμματος λοιπόν, όπως αναφέρθηκε και στην ενότητα Δεδομένα-Μεθοδολογία, πραγματοποιήθηκε συλλογή μεγάλου αριθμού σύνθετων ρημάτων με προθέσεις που, σε ορισμένα περιβάλλοντα, διαθέτουν εσωτερική αύξηση. Τα ρήματα αυτά και συγκεκριμένα, οι δύο τύποι που μας ενδιαφέρουν, εισήχθησαν σε 2 διαφορετικές λίστες στο πρόγραμμα με τη μορφή συμβολοσειρών (strings)[11]. Οι συμβολοσειρές αποτελούν τον βασικό τύπο δεδομένων που μας απασχολεί στην παρούσα εργασία.

Η έννοια της «λίστας» στον προγραμματισμό είναι αρκετά συναφής με τη συμβατική έννοια που διαθέτει η λέξη. Όπως μας λέει ο Αγγελιδάκης (2015), «μία **λίστα (list)** είναι μια διατεταγμένη συλλογή τιμών, οι οποίες αντιστοιχίζονται σε δείκτες». Συγκεκριμένα, μια λίστα σε ένα πρόγραμμα μπορεί να περιέχει διαφορετικά είδη δεδομένων (αριθμούς, συμβολοσειρές), τα οποία διαθέτουν συγκεκριμένο δείκτη (index) ανάλογα με τη θέση τους μέσα στη λίστα (βλ. παρακάτω). Επίσης, κάθε λίστα διαθέτει τη δική της ονομασία, προκειμένου να μπορούμε να αναφερθούμε σε αυτή, σε άλλα σημεία του προγράμματος. Παραδείγματος χάρη, η ακόλουθη είναι μια λίστα που διαθέτει κάποιους αριθμούς:

14) σύντομη_λίστα = [3, 5, 9, 24]

[11] Μια συμβολοσειρά «είναι μια ακολουθία από χαρακτήρες που περικλείονται σε μονά ή διπλά εισαγωγικά» (Αγγελιδάκης, 2015: 18).

Η λίστα του παραδείγματος 14 διαθέτει την ονομασία «σύντομη_λίστα»[12] και περιέχει τους αριθμούς 3, 5, 9 και 24 με ορισμένη σειρά. Επίσης, κάθε αριθμός διαθέτει και τον αντίστοιχο δείκτη μέσα στη λίστα. Η αρίθμηση των δεικτών σε μια λίστα ξεκινάει πάντοτε από το 0 και όχι από το 1. Επομένως, στην παραπάνω λίστα υπάρχουν οι θέσεις δείκτη 0, 1, 2 και 3 που αντιστοιχούν στους αριθμούς 3, 5, 9 και 24. Ο αριθμός 5 λοιπόν διαθέτει δείκτη 1 στη σύντομη_λίστα. Οι δείκτες παίζουν πολύ σημαντικό ρόλο στη λειτουργία του προγράμματος της παρούσας εργασίας, όπως αναλύεται και παρακάτω.

Οι δύο λίστες που εισήχθησαν στο πρόγραμμα είναι η list_a και η list_b. Η list_a περιλαμβάνει τα υπό εξέταση ρήματα στο τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου στην ενεργητική φωνή, ενώ η list_b περιλαμβάνει τα υπό εξέταση ρήματα στο δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή.

Να σημειωθεί ότι, στις παραπάνω λίστες, δεν έχουν προστεθεί ορισμένα σύνθετα ρήματα με προθέσεις, από τα οποία απουσιάζει η εσωτερική αύξηση στην οριστική αορίστου, με αποτέλεσμα να έχουν κοινό τύπο τόσο για το τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου στην ενεργητική φωνή όσο και για το δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή[13]. Ως παραδείγματα αυτής της κατηγορίας, μπορούμε να αναφέρουμε τα ρήματα «αναμερίζω», «διαλέγω», και «ανασαίνω».

3.3.1.2 Συνάρτηση

3.3.1.2.1 Η έννοια της συνάρτησης

Στη συνέχεια, κάτω από τις λίστες, το πρόγραμμα διαθέτει μια συνάρτηση, η οποία περιλαμβάνει τις εντολές που επιτρέπουν στο πρόγραμμα να επεξεργάζεται τις προτάσεις του χρήστη και να καθορίζει ποια από τις δύο είναι η σωστή. Στον προγραμματισμό, ως συνάρτηση ορίζεται μια σειρά εντολών που διαθέτει δική της ονομασία, μέσω της οποίας μπορούμε να καλούμε τη συνάρτηση όσες φορές επιθυμούμε και σε οποιοδήποτε σημείο του προγράμματός μας (Αγγελιδάκης, 2015: 52). Μέσω των συναρτήσεων αποφεύγεται η επανάληψη των ίδιων εντολών σε διαφορετικά σημεία του προγράμματος. Συγκεκριμένα, οι

[12] Η χρήση του underscore (κάτω παύλα) αντί για κενό είναι συμβατική στον προγραμματισμό όταν ορίζονται ονομασίες μεταβλητών και χρησιμοποιείται για μια πληθώρα λόγων όπως, παραδείγματος χάρη, την αποφυγή ανεπιθύμητης αναφοράς σε δεσμευμένες λέξεις που διαθέτουν συγκεκριμένες λειτουργίες στην Python (list, function κτλ).

[13] Όπως αναφέρουν χαρακτηριστικά οι Φιλιππάκη-Warburton κ.ά. (2011: 137) «Σε μερικά τέτοια ρήματα η εσωτερική αύξηση δε χρησιμοποιείται ποτέ...».

εντολές διατυπώνονται μόνο μία φορά μέσα στη συνάρτηση και, κάθε φορά που είναι αναγκαίο, καλούμε τη συνάρτηση μέσω της ονομασίας της[14].

Στο σημείο αυτό, αξίζει να παραθέσουμε ένα παράδειγμα συνάρτησης, προκειμένου να παρουσιαστούν τα μέρη από τα οποία αποτελείται:

```
15) def καλησπέρα_φίλε():
        """Τυπώνεται η φράση "Καλησπέρα φίλε!" στην κονσόλα."""
        print ("Καλησπέρα φίλε!")
```

Στην πρώτη σειρά εντοπίζεται η δεσμευμένη λέξη def (αποκαλείται δεσμευμένη διότι επιτελεί συγκεκριμένη λειτουργία στην Python), η οποία είναι πάντοτε παρούσα αριστερά από την ονομασία της συνάρτησης. Επιπλέον, εντοπίζουμε την ονομασία που ορίζουμε εμείς για τη συνάρτηση καθώς και παρενθέσεις οι οποίες, στην προκειμένη περίπτωση, δεν περιέχουν τίποτα. Οι παρενθέσεις μπορεί να περιέχουν ορίσματα που επιθυμούμε να λαμβάνει η συνάρτησή μας κάθε φορά που την καλούμε. Όταν ορίζουμε μια συνάρτηση, οι παρενθέσεις πρέπει να τοποθετούνται ακόμη και σε περιπτώσεις όπως η συγκεκριμένη, που δεν επιθυμούμε να συμπεριλάβουμε ορίσματα. Ακόμη, στο τέλος της πρώτης σειράς πρέπει να υπάρχει πάντοτε άνω και κάτω τελεία.

Στη δεύτερη σειρά εντοπίζεται ένα σχόλιο το οποίο μας πληροφορεί αναφορικά με το τι κάνει η συγκεκριμένη συνάρτηση. Τα σχόλια, τόσο στην περίπτωση της συνάρτησης όσο και στο υπόλοιπο πρόγραμμα, είναι σαφώς προαιρετικά και τοποθετούνται συνήθως για να είναι πιο κατανοητός ο κώδικας από άτομα πέραν του προγραμματιστή. Για να γράψουμε ένα σχόλιο τοποθετούμε είτε μία δίεση στην αρχή της σειράς είτε τριπλά εισαγωγικά (όπως παραπάνω) στην αρχή και το τέλος του σχολίου. Κάθε φορά που εκτελείται ένα πρόγραμμα, ο υπολογιστής αγνοεί πλήρως τα σχόλια. Επομένως, δεν παίζουν κανέναν ρόλο στη λειτουργία του προγράμματος καθαυτή, αλλά, όπως αναφέρθηκε και παραπάνω, χρησιμοποιούνται για παροχή επεξήγησης σε άλλους χρήστες.

Στη συνέχεια, εντοπίζεται το κυρίως μέρος μιας συνάρτησης, το οποίο περιλαμβάνει τις εντολές που εκτελεί η εκάστοτε συνάρτηση. Στην προκειμένη περίπτωση εντοπίζεται η εντολή print, η οποία με τη σειρά της αποτελεί μία συνάρτηση (για τον λόγο αυτό γίνεται και χρήση παρενθέσεων). Η συνάρτηση print τυπώνει στην κονσόλα οτιδήποτε δέχεται ως όρισμα (στην προκειμένη περίπτωση τη συμβολοσειρά "Καλησπέρα φίλε!").

[14] Η φράση «καλούμε τη συνάρτηση» είναι συμβατική στον προγραμματισμό και αναφέρεται στην εντολή που θέτει τη συνάρτηση σε λειτουργία.

Έχοντας εξηγήσει λοιπόν την έννοια της συνάρτησης και έχοντας παραθέσει ένα παράδειγμα, προχωράμε στην παρουσίαση της συνάρτησης του προγράμματός μας.

3.3.1.2.2 Η συνάρτηση της αρχικής μορφής του προγράμματος

Στο σημείο αυτό παραθέτουμε τη συνάρτηση του προγράμματος προκειμένου να παρουσιαστεί κάθε μέρος της αναλυτικά:

```
1. def find_correct_choice():
2.     while 1 < 2:
3.         sentence_1 = input("Παρακαλώ εισαγάγετε την πρώτη πρόταση: ")
4.         sentence_2 = input("Παρακαλώ εισαγάγετε τη δεύτερη πρόταση: ")
5.         lowercase_sentence_1 = sentence_1.lower()
6.         lowercase_sentence_2 = sentence_2.lower()
7.         split_sentence_1 = lowercase_sentence_1.split()
8.         split_sentence_2 = lowercase_sentence_2.split()
9.         if split_sentence_1[0] in list_b:
10.            print ("Η σωστή πρόταση είναι: %s" % sentence_1)
11.        elif split_sentence_2[0] in list_b:
12.            print ("Η σωστή πρόταση είναι: %s" % sentence_2)
13.        else:
14.            for item in list_a:
15.                if item in split_sentence_1:
16.                    print ("Η σωστή πρόταση είναι: %s" % sentence_1)
17.                elif item in split_sentence_2:
18.                    print ("Η σωστή πρόταση είναι: %s" % sentence_2)
19.        print("")
20.        question = input("Επιθυμείτε να συνεχίσετε; ")
21.        if question == 'ΟΧΙ' or question == 'Όχι' or question == 'Οχι' \
22.        or question == 'όχι' or question == 'οχι':
23.            break
24.        print("")
25.
26. find_correct_choice()
```

Αρχικά, αξίζει να αναφερθεί ότι σε ένα ολοκληρωμένο περιβάλλον ανάπτυξης κάθε σειρά είναι αριθμημένη για τη διευκόλυνση του χρήστη. Επομένως, και στην προκειμένη περίπτωση που παρατίθεται η συνάρτηση του προγράμματος, έχει πραγματοποιηθεί αρίθμηση των σειρών προκειμένου να είναι ευκολότερη η αναφορά σε συγκεκριμένα σημεία της συνάρτησης.

Στη σειρά 1 ορίζεται η ονομασία της συνάρτησης, η οποία δεν απαιτεί συγκεκριμένα ορίσματα (για τον λόγο αυτό οι παρενθέσεις είναι κενές).

Προτού συνεχίσουμε με την ανάλυση της συνάρτησης, είναι αναγκαίο να επεξηγηθεί η έννοια του βρόχου (loop) ή αλλιώς της εντολής επανάληψης, που εμφανίζεται σε ορισμένα σημεία του προγράμματος. Ένας βρόχος αποτελεί «μία...ομάδα εντολών που επαναλαμβάνεται» (Αγγελιδάκης, 2015: 43). Βασικό στοιχείο λοιπόν ενός βρόχου είναι το γεγονός ότι οι εντολές που περιλαμβάνει επαναλαμβάνονται για την επιτέλεση ενός σκοπού. Δύο βασικά είδη βρόχου που θα μας απασχολήσουν στην εν λόγω εργασία είναι ο βρόχος for και ο βρόχος while. Ας εξετάσουμε όμως παραδείγματα του καθενός:

```
16) for x in [1, 2, 3]:
        print (x * 2)
```

Στην πρώτη σειρά εντοπίζεται η δεσμευμένη λέξη «for», την οποία ακολουθεί η μεταβλητή x. Δεν υπάρχει περιορισμός αναφορικά με τη μεταβλητή που θα χρησιμοποιήσουμε, καθώς μπορεί να διαθέτει οποιαδήποτε ονομασία, η περίληψή της όμως είναι αναγκαία. Στη συνέχεια ακολουθεί η δεσμευμένη λέξη in (η οποία διαθέτει τη σημασία «περιλαμβάνεται σε») και τέλος μια σύντομη λίστα που περιέχει τους αριθμούς 1, 2 και 3. Ουσιαστικά η πρώτη σειρά του βρόχου δηλώνει: για κάθε στοιχείο x που βρίσκεται στην ακόλουθη λίστα, κάνε τα ακόλουθα. Η μεταβλητή x λοιπόν αναφέρεται σε καθένα από τα στοιχεία της λίστας, για τα οποία μάλιστα λαμβάνουν χώρα οι διαδικασίες που περιγράφονται στο κυρίως μέρος του βρόχου.

Στη δεύτερη σειρά λοιπόν εντοπίζεται η εντολή print διαθέτοντας το όρισμα x * 2 (στην Python το * συμβολίζει τη μαθηματική πράξη του πολλαπλασιασμού). Όπως αναφέραμε ήδη, η μεταβλητή x αναφέρεται σε κάθε στοιχείο της λίστας. Επομένως, η εντολή της δεύτερης σειράς θα εκτελεστεί συνολικά 3 φορές, μία για κάθε στοιχείο της λίστας, δίνοντας το εξαγόμενο 2, 4 και 6 για τα 1, 2 και 3 αντίστοιχα. Εδώ λοιπόν εντοπίζεται το στοιχείο της επανάληψης που αναφέραμε παραπάνω για τον βρόχο.

Ακολουθεί παράδειγμα ενός βρόχου while:

```
17) count = 0

    while count < 3:
        print ("Καλημέρα")
        count += 1
```

Ένας βρόχος while διαθέτει επίσης το στοιχείο της επανάληψης, με τη διαφορά ότι εκτελείται για όσο ισχύει μια συγκεκριμένη συνθήκη (εξού και η ονομασία του). Αρχικά

δημιουργούμε μια μεταβλητή με την ονομασία «count» και τη θέτουμε ίση με το μηδέν. Στη συνέχεια, στην πρώτη σειρά του βρόχου εντοπίζουμε τη δεσμευμένη λέξη «while» η οποία ακολουθείται από τη συνθήκη της. Στην προκειμένη περίπτωση, η συνθήκη είναι «για όσο η μεταβλητή count είναι μικρότερη από το 3». Στο κυρίως μέρος του βρόχου εντοπίζεται η εντολή «print» που τυπώνει τη λέξη «Καλημέρα» και, στην επόμενη σειρά, προστίθεται το 1 στη μεταβλητή «count» (το σύμβολο += σημαίνει προσθήκη της τιμής που βρίσκεται στα δεξιά του στη μεταβλητή που βρίσκεται στα αριστερά του).

Κάθε φορά που εκτελείται ο συγκεκριμένος βρόχος λοιπόν, αφενός τυπώνει τη λέξη «Καλημέρα» στην κονσόλα και αφετέρου προσθέτει 1 στο «count». Ο βρόχος λοιπόν εκτελείται 3 φορές συνολικά, καθώς την τέταρτη φορά που πάει να εκτελεστεί, το count ισοδυναμεί με το 3, επομένως η συνθήκη δεν ισχύει πλέον και ο βρόχος διακόπτεται. Το τελικό εξαγόμενο του συγκεκριμένου βρόχου λοιπόν είναι το ακόλουθο:

```
18) Καλημέρα
    Καλημέρα
    Καλημέρα
```

Στην περίπτωση που εξετάσαμε, η συνθήκη του βρόχου παύει να ισχύει έπειτα από 3 επαναλήψεις. Υπάρχουν όμως περιπτώσεις κατά τις οποίες η συνθήκη ενός βρόχου while δεν παύει να ισχύει ποτέ, γεγονός που οδηγεί στην περίπτωση του ατέρμονος βρόχου (Αγγελιδάκης, 2015: 47) ή αλλιώς της ατέρμονης επανάληψης (Μαρκόπουλος, 2006: 48). Στην Αγγλική, το φαινόμενο λέγεται infinite loop. Παραδείγματος χάρη:

```
19) while 6 == 6:
        print ("spam")
```

Ο παραπάνω βρόχος εκτελείται ξανά και ξανά τυπώνοντας τη συμβολοσειρά «spam», διότι η συνθήκη του δεν παύει να ισχύει ποτέ (το 6 πάντοτε ισοδυναμεί με 6).

Επιστρέφοντας λοιπόν στη συνάρτηση του προγράμματος της παρούσας εργασίας, στη σειρά 2 εντοπίζεται ένας ατέρμων βρόχος while, καθώς έχουμε τη συνθήκη 1 < 2 (το 1 είναι πάντοτε μικρότερο του 2). Οι λόγοι περίληψης ενός ατέρμονος βρόχου στη συνάρτησή μας θα γίνουν ξεκάθαροι παρακάτω.

Στην τρίτη σειρά της συνάρτησης εντοπίζεται η εντολή input που επίσης αποτελεί μια συνάρτηση. Η εντολή input τυπώνει στην κονσόλα τη συμβολοσειρά που λαμβάνει ως

όρισμα (στην προκειμένη περίπτωση τη συμβολοσειρά "Παρακαλώ εισαγάγετε την πρώτη πρόταση: ") και επίσης, ζητάει από τον χρήστη να εισαγάγει κάποιον τύπο δεδομένων στο πρόγραμμα. Στη συνέχεια, η εντολή input καταχωρεί το εισαγόμενο του χρήστη στη μεταβλητή την οποία είχαμε ορίσει αρχικά ως ίση με την εντολή input και το όρισμά της. Στη συγκεκριμένη περίπτωση λοιπόν, η πρόταση που θα εισαγάγει ο χρήστης ισοδυναμεί με τη μεταβλητή sentence_1. Με τον τρόπο αυτό, μπορούμε να αναφερόμαστε μέσα στο πρόγραμμα στην πρώτη πρόταση του χρήστη, (μέσω της μεταβλητής sentence_1), αλλά και να την επεξεργαζόμαστε, άσχετα από το ποια είναι η πρόταση αυτή σε κάθε περίπτωση.

Η ίδια διαδικασία λαμβάνει χώρα και στη σειρά 4, με μοναδική διαφορά ότι το όρισμα του input σε αυτή την περίπτωση είναι το "Παρακαλώ εισαγάγετε τη δεύτερη πρόταση: " και ότι το εισαγόμενο του χρήστη καταχωρείται στη μεταβλητή sentence_2.

Προτού μιλήσουμε για τις σειρές 5 και 6, είναι αναγκαίο να αναφερθούμε στην έννοια της μεθόδου. «Οι μέθοδοι είναι όμοιες με τις συναρτήσεις (παίρνουν ορίσματα και επιστρέφουν κάποια τιμή), αλλά διαφέρουν στον τρόπο με τον οποίο συντάσσονται (dot notation, χρήση του συμβόλου της τελείας)» (Αγγελιδάκης, 2015: 80). Σύμφωνα με τον ορισμό λοιπόν, οι μέθοδοι λειτουργούν με παρόμοιο τρόπο με τις συναρτήσεις, με τις βασικές διαφορές να εντοπίζονται στο κομμάτι της σύνταξης, στο οποίο εντοπίζεται πάντοτε η τελεία. Στις σειρές 5 και 6 λοιπόν εντοπίζουμε τη μέθοδο .lower(), η οποία μετατρέπει σε πεζούς τους κεφαλαίους χαρακτήρες του ορίσματος που δέχεται. Βασικός λόγος περίληψης της συγκεκριμένης μεθόδου, είναι ότι επιτρέπει στο πρόγραμμα να εντοπίζει το σύνθετο ρήμα σε μια πρόταση, άσχετα από το εάν ο χρήστης το έχει γράψει με πεζούς ή κεφαλαίους χαρακτήρες ή ακόμη και με συνδυασμό αυτών, καθώς, μέσα στο πρόγραμμα, όλοι οι χαρακτήρες μετατρέπονται σε πεζούς μέσω της συγκεκριμένης μεθόδου. Παραδείγματος χάρη, στην πρόταση του παραδείγματος 9, το ρήμα «Μετάδωσε» διαθέτει κεφαλαίο τον πρώτο χαρακτήρα, διότι βρίσκεται στην αρχή της πρότασης. Επομένως, είναι αναγκαία η χρήση της μεθόδου .lower() προκειμένου ο πρώτος χαρακτήρας να γίνει πεζός. Σε διαφορετική περίπτωση, το πρόγραμμα δεν θα μπορούσε να εντοπίσει το ρήμα κάνοντας ταύτιση με τον αντίστοιχο τύπο της λίστας, ο οποίος έχει εισαχθεί με τη μορφή «μετάδωσε». Επομένως, η πρώτη και η δεύτερη πρόταση του χρήστη καταχωρούνται (με όλους τους χαρακτήρες τους πεζούς) στις μεταβλητές lowercase_sentence_1 και lowercase_sentence_2 αντίστοιχα.

Στο σημείο αυτό, οι προτάσεις του χρήστη έχουν εισαχθεί ως ενιαίες συμβολοσειρές και δεν είναι χωρισμένες σε λέξεις. Για να πραγματοποιηθεί κάτι τέτοιο, χρησιμοποιούμε τη

μέθοδο .split(), η οποία δέχεται μια συμβολοσειρά, τη χωρίζει σε λέξεις και, στη συνέχεια, δημιουργεί μια λίστα η οποία περιλαμβάνει κάθε λέξη ως ξεχωριστό στοιχείο. Λόγου χάρη:

```
20) print ("Κοίτα ψηλά στον ουρανό".split())
```

Το εξαγόμενο της παραπάνω εντολής θα είναι η λίστα: ['Κοίτα', 'ψηλά', 'στον', 'ουρανό']. Επομένως, στις σειρές 7 και 8, η πρώτη και η δεύτερη πρόταση του χρήστη χωρίζονται σε λέξεις με τη μέθοδο .split() και καταχωρούνται στις μεταβλητές split_sentence_1 και split_sentence_2 αντίστοιχα.

Στη συνέχεια, κρίνεται απαραίτητη η αναφορά στην εντολή if και τη λειτουργία της. «Η εντολή if χρησιμοποιείται για έλεγχο της ροής εκτέλεσης ενός προγράμματος. Ελέγχεται μία συνθήκη και ανάλογα με το αποτέλεσμα (Αληθής ή Ψευδής) εκτελείται ή δεν εκτελείται μία ή κάποια άλλη ομάδα (μπλοκ) εντολών» (Αγγελιδάκης, 2015: 38). Παραδείγματος χάρη:

```
21) package = 5

 if package > 6:
     print ("heavy")
elif package == 6:
     print ("ok")
else:
     print ("light")
```

Προτού ξεκινήσουμε την ανάλυση του παραδείγματος 21, αξίζει να αναφέρουμε ότι κάθε φορά που εκτελείται ένα πρόγραμμα Python, ξεκινάει από την πρώτη σειρά, έπειτα συνεχίζει με τη δεύτερη και ούτω καθεξής. Επομένως, η σειρά με την οποία εισάγονται οι εντολές στο πρόγραμμα έχει μεγάλη σημασία.

Στο παράδειγμα 21 λοιπόν, ξεκινάμε με τη μεταβλητή package, την οποία θέτουμε ίση με 5. Στη συνέχεια εντοπίζουμε τη δεσμευμένη λέξη if, η οποία ακολουθείται από μία συνθήκη. Συγκεκριμένα, η συνθήκη είναι «εάν η μεταβλητή package διαθέτει τιμή μεγαλύτερη από το 6». Σε περίπτωση που η συγκεκριμένη συνθήκη ισχύει, το πρόγραμμα θα εκτελέσει την εντολή που βρίσκεται στο κυρίως μέρος της εντολής if. Συγκεκριμένα, θα τυπώσει τη συμβολοσειρά «heavy» στην κονσόλα.

Η συγκεκριμένη συνθήκη όμως δεν ισχύει, καθώς η μεταβλητή package ισούται με 5, το οποίο είναι μικρότερο από το 6. Εφόσον λοιπόν η συνθήκη δεν είναι αληθής και δεν εκτελείται η διαδικασία που περιέχεται στην πρώτη εντολή if, το πρόγραμμα συνεχίζει στην

επόμενη σειρά, όπου εντοπίζουμε τη δεσμευμένη λέξη elif. Κάθε φορά που επιθυμούμε να εισαγάγουμε επιπλέον συνθήκες πέρα από την αρχική, χρησιμοποιούμε την εντολή elif (else + if). Στην προκειμένη περίπτωση, η εντολή elif θα τυπώσει στην κονσόλα τη συμβολοσειρά «ok», εάν η μεταβλητή package είναι ίση με το 6. Στην προκειμένη περίπτωση δεν ισχύει κάτι τέτοιο, επομένως η συνθήκη είναι ψευδής και, ως εκ τούτου, η εντολή δεν εκτελείται.

Στο σημείο αυτό, θα μπορούσαμε να χρησιμοποιήσουμε περαιτέρω εντολές elif, ανάλογα με τον αριθμό των συνθηκών που θα θέλαμε να εξετάσουμε.

Στην επόμενη σειρά, εντοπίζουμε την εντολή else, η οποία όμως δε συνοδεύεται από κάποια συνθήκη. Αυτό συμβαίνει διότι, έχοντας ελέγξει ορισμένες συνθήκες και εφόσον αυτές δεν είναι αληθείς, μπορεί να επιθυμούμε να εκτελεστεί μια συγκεκριμένη εντολή, ανεξάρτητα από το ποια συνθήκη ισχύει τελικά. Συγκεκριμένα, στην προκειμένη περίπτωση, έχει ελεγχθεί εάν η μεταβλητή package είναι ίση ή μεγαλύτερη από το 6 και, επειδή δεν ισχύει τίποτε από τα δύο, έχουμε καταλήξει στην εντολή else, σύμφωνα με την οποία θα τυπωθεί στην κονσόλα η συμβολοσειρά «light».

Θα μπορούσαμε να πούμε ότι σε μια τυπική περίπτωση χρήσης εντολών if, ελέγχονται ορισμένες συνθήκες και, εάν κάποια από αυτές είναι αληθής, εκτελούνται οι διαδικασίες που περιέχονται στην αντίστοιχη εντολή if. Για κάθε άλλη περίπτωση, εκτελείται η εντολή που περιλαμβάνεται στο κυρίως μέρος της εντολής else.

Έχοντας αναλύσει τα παραπάνω λοιπόν, επιστρέφουμε στη συνάρτησή μας και συγκεκριμένα στη σειρά 9. Όπως αναφέρθηκε στο 3.3.1.1, τα στοιχεία μιας λίστας διαθέτουν συγκεκριμένο δείκτη (index) ανάλογα με τη θέση τους μέσα στη λίστα. Επιπλέον, η αρίθμηση των δεικτών ξεκινάει από το μηδέν και όχι από το ένα.

Στη σειρά 9 λοιπόν, η συνθήκη της εντολής if είναι «εάν η πρώτη λέξη της πρότασης 1 περιλαμβάνεται στη list_b». Να επαναλάβουμε στο σημείο αυτό ότι η πρόταση 1 του χρήστη έχει χωριστεί σε λέξεις με τη χρήση της μεθόδου .split() και έχει καταχωρηθεί στη μεταβλητή split_sentence_1. Επιπλέον, η list_b είναι η μία από τις δύο βασικές λίστες του προγράμματος, η οποία περιλαμβάνει τα υπό εξέταση ρήματα στο δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή. Όπως αναφέρθηκε και παραπάνω, η δεσμευμένη λέξη in διαθέτει τη σημασία «περιλαμβάνεται σε». Επομένως, είναι απαραίτητη για να ελέγξουμε εάν η πρώτη λέξη της πρότασης 1 περιλαμβάνεται στη list_b.

Εφόσον λοιπόν, το κριτήριο με το οποίο λειτουργεί το συγκεκριμένο πρόγραμμα είναι η θέση μέσα στην πρόταση και εφόσον η υπόθεση του γράφοντος είναι ότι, τις περισσότερες φορές, η προστακτική εντοπίζεται στην αρχή της πρότασης, η πρόταση 1 χαρακτηρίζεται ως σωστή, εάν η πρώτη λέξη της περιέχεται στη list_b.

Για τον λόγο αυτό, στη σειρά 10 βλέπουμε ότι η εντολή print που περιλαμβάνεται στο κυρίως μέρος της πρώτης εντολής if παρουσιάζει ως σωστή την πρώτη πρόταση. Αρχικά, αξίζει να αναφερθούμε στο σύμβολο %s, το οποίο λειτουργεί ως μεταβλητή, η οποία αντικαθίσταται από ό,τι καθορίσει ο δημιουργός του προγράμματος. Στη συγκεκριμένη περίπτωση έχουμε καθορίσει να αντικαθίσταται από τη μεταβλητή sentence_1, δηλαδή την πρώτη πρόταση που έχει εισαγάγει ο χρήστης. Έχουμε επιλέξει τη μεταβλητή sentence_1 και όχι κάποια από τις άλλες (lowercase_sentence_1 ή split_sentence_1) για να τυπώνεται στην κονσόλα ως σωστή η πρόταση που έχει εισαγάγει ο χρήστης, με τη μορφή που την έχει εισαγάγει (και όχι με επεξεργασμένη μορφή, παραδείγματος χάρη χωρισμένη σε λέξεις ή με όλους τους χαρακτήρες της πεζούς).

Οι διαδικασίες των σειρών 9 και 10 που περιγράψαμε μόλις λαμβάνουν χώρα και στις σειρές 11 και 12, με τη διαφορά ότι αφορούν τη δεύτερη και όχι την πρώτη πρόταση του χρήστη. Συγκεκριμένα, στη σειρά 11, η συνθήκη της εντολής elif είναι «εάν η πρώτη λέξη της πρότασης 2 περιλαμβάνεται στη list_b». Η εντολή λοιπόν, εξετάζει εάν υπάρχει προστακτική στην πρώτη θέση της δεύτερης πρότασης. Σε περίπτωση που η συνθήκη είναι αληθής, η εντολή print της σειράς 12 θα τυπώσει τη δεύτερη πρόταση του χρήστη, παρουσιάζοντάς την ως σωστή.

Στο σημείο αυτό λοιπόν, το πρόγραμμα έχει εξετάσει τις πρώτες θέσεις και των δύο προτάσεων για παρουσία δεύτερου προσώπου ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή. Εφόσον λοιπόν δεν εντοπιστεί προστακτική στις πρώτες θέσεις των δύο προτάσεων, το πρόγραμμα συνεχίζει να εκτελείται, εξετάζοντας τις υπόλοιπες θέσεις των δύο προτάσεων.

Για την ακρίβεια, στη σειρά 13 εντοπίζεται η εντολή else, η οποία εκτελείται μόνο εάν οι συνθήκες των σειρών 9 και 11 δεν είναι αληθείς. Στη σειρά 14, όπου ξεκινάει το κυρίως μέρος της εντολής else, εντοπίζεται ένας βρόχος for που συνοδεύεται από τη μεταβλητή item, η οποία αναφέρεται σε καθένα από τα στοιχεία της list_a. Συγκεκριμένα, ο βρόχος εκτελείται ίσο αριθμό φορών με τον αριθμό των στοιχείων της list_a, η οποία είναι η μία από τις δύο βασικές λίστες του προγράμματος και περιλαμβάνει τα υπό εξέταση ρήματα στο τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου στην ενεργητική φωνή.

Στη σειρά 15, η εντολή if εξετάζει εάν κάποιο από τα στοιχεία της list_a περιλαμβάνεται στην πρώτη πρόταση του χρήστη σε οποιαδήποτε θέση. Εξετάζεται δηλαδή εάν η πρώτη πρόταση περιλαμβάνει σύνθετο ρήμα με πρόθεση στο τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου στην ενεργητική φωνή. Εάν η συνθήκη είναι αληθής, η εντολή

print της σειράς 16 παρουσιάζει την πρώτη πρόταση του χρήστη ως σωστή, τυπώνοντάς την στην κονσόλα.

Οι διαδικασίες των σειρών 15 και 16 λαμβάνουν χώρα και στις σειρές 17 και 18 με τη διαφορά ότι αφορούν τη δεύτερη πρόταση του χρήστη. Συγκεκριμένα, η εντολή elif της σειράς 17 εξετάζει εάν η δεύτερη πρόταση περιλαμβάνει κάποιο ρήμα οριστικής της list_a σε οποιαδήποτε θέση. Σε περίπτωση που ισχύει κάτι τέτοιο, η εντολή print της σειράς 18 τυπώνει τη δεύτερη πρόταση και την παρουσιάζει ως σωστή.

Στο σημείο αυτό, ο αναγνώστης μπορεί να διαφωνεί αναφορικά με τον τρόπο που είναι διατυπωμένες οι συνθήκες των σειρών 15 και 17. Για την ακρίβεια, σύμφωνα με τον τρόπο που έχει αναπτυχθεί το συγκεκριμένο πρόγραμμα, οι προτάσεις που διαθέτουν σύνθετο ρήμα με πρόθεση σε προστακτική στην πρώτη θέση πρέπει να παρουσιάζονται ως σωστές, ενώ παράλληλα, οι προτάσεις που διαθέτουν σύνθετο ρήμα με πρόθεση σε οριστική σε οποιαδήποτε θέση πλην της πρώτης πρέπει να παρουσιάζονται ως σωστές. Στις σειρές 15 και 17 όμως, δεν εντοπίζεται ο περιορισμός που μόλις αναφέρθηκε, ότι δηλαδή η οριστική πρέπει να εντοπίζεται σε οποιαδήποτε θέση πλην της πρώτης, καθώς οι εντολές if και elif ψάχνουν για στοιχείο της list_a σε οποιαδήποτε θέση των προτάσεων 1 και 2. Δικαιολογημένα λοιπόν, ο αναγνώστης μπορεί να υποθέσει ότι σε περίπτωση που η πρώτη πρόταση περιλαμβάνει σύνθετο ρήμα με πρόθεση σε οριστική στην πρώτη θέση και η δεύτερη πρόταση περιλαμβάνει σύνθετο ρήμα με πρόθεση σε προστακτική στην πρώτη θέση το πρόγραμμα ενδέχεται να παρουσιάσει λανθασμένα την πρώτη πρόταση ως σωστή.

Μάλιστα, στην Python είναι δυνατή η χρήση περιορισμού ο οποίος, στις σειρές 15 και 17 θα εξέταζε εάν υπάρχουν στοιχεία της list_a στις δύο προτάσεις σε οποιαδήποτε θέση πλην της πρώτης. Ωστόσο, όπως αναφέρθηκε ήδη παραπάνω, κάθε φορά που εκτελείται το πρόγραμμα, εξετάζει τις σειρές από πάνω προς τα κάτω. Ακόμη, όταν υπάρχουν εντολές if, το πρόγραμμα μεταβαίνει από τη μία στην άλλη μόνο εάν η συνθήκη της πρώτης είναι ψευδής. Στην προκειμένη περίπτωση λοιπόν, το πρόγραμμα θα έχει φτάσει στις συνθήκες των σειρών 15 και 17 (οι οποίες περιέχονται στην εντολή else της σειράς 13) μόνο εάν οι συνθήκες των σειρών 9 και 11 είναι ψευδείς. Επομένως, η περίπτωση παρουσίας σύνθετου ρήματος με πρόθεση σε προστακτική στην πρώτη θέση της πρώτης ή της δεύτερης πρότασης έχει ήδη αποκλειστεί και επομένως, η περίληψη περιορισμού στις συνθήκες των σειρών 15 και 17 είναι περιττή.

Στη συνέχεια, βλέπουμε ότι το όρισμα της εντολής print στη σειρά 19 είναι το ακόλουθο: (""), το οποίο ουσιαστικά αποτελεί μια κενή συμβολοσειρά. Κάθε φορά που η

εντολή print λαμβάνει το συγκεκριμένο όρισμα, τυπώνει μια κενή σειρά στην κονσόλα. Η συγκεκριμένη εντολή έχει περιληφθεί καθαρά για αισθητικούς λόγους.

Στη σειρά 20, εντοπίζουμε ξανά τη συνάρτηση input, την οποία συναντήσαμε αρχικά στην τρίτη σειρά του προγράμματος. Στην περίπτωση αυτή, η εντολή input τυπώνει τη συμβολοσειρά "Επιθυμείτε να συνεχίσετε; " και καταχωρεί την απάντηση του χρήστη στη μεταβλητή question. Στο συγκεκριμένο σημείο λοιπόν, οι βασικές διαδικασίες του προγράμματος έχουν ολοκληρωθεί και η συγκεκριμένη εντολή ουσιαστικά «ρωτάει» τον χρήστη εάν επιθυμεί να εισαγάγει επιπλέον ζεύγη προτάσεων.

Στη σειρά 21 γίνεται χρήση μιας εντολής if προκειμένου να εξεταστεί η απάντηση του χρήστη, η οποία, όπως αναφέρθηκε, έχει καταχωρηθεί στη μεταβλητή question. Εάν η απάντηση του χρήστη ισοδυναμεί με «όχι» (στις σειρές 21 και 22 εντοπίζουμε όλους τους πιθανούς τρόπους διατύπωσης του «όχι» αναφορικά με την παρουσία ή απουσία τόνου, τη χρήση κεφαλαίων ή πεζών χαρακτήρων κτλ) εκτελείται η εντολή break της σειράς 23. Το σύμβολο « \ » χρησιμοποιείται στο τέλος της σειράς 21 διότι η εντολή συνεχίζεται στη σειρά 22. Το συγκεκριμένο σύμβολο λοιπόν, επιτρέπει στο πρόγραμμα να αντιμετωπίζει ως ενιαία εντολή το περιεχόμενο της σειράς 21 και το περιεχόμενο της σειράς 22.

Όπως αναφέρθηκε και παραπάνω, ένας βρόχος while παύει να εκτελείται όταν η συνθήκη του παύει να ισχύει. Στην περίπτωση όμως του ατέρμονος βρόχου, η συνθήκη ισχύει πάντοτε, επομένως ο βρόχος δεν διακόπτεται ποτέ. Για τον λόγο αυτό είναι, ορισμένες φορές, αναγκαία η περίληψη της εντολής break, η οποία προκαλεί διακοπή της εκτέλεσης ενός βρόχου. Συγκεκριμένα, στη σειρά 2 της συνάρτησης, η συνθήκη του βρόχου while δεν παύει να ισχύει ποτέ (1 < 2) μετατρέποντάς τον σε ατέρμονα. Με τον τρόπο αυτό, η συνάρτηση εκτελείται ξανά και ξανά, δίνοντας στον χρήστη τη δυνατότητα να εισαγάγει περαιτέρω ζεύγη προτάσεων. Ωστόσο, είναι αναγκαία η ύπαρξη της επιλογής διακοπής του βρόχου και, ως εκ τούτου, της συνάρτησης. Για αυτό άλλωστε, η εντολή input στη σειρά 20 «ρωτάει» τον χρήστη εάν επιθυμεί να συνεχίσει. Σε περίπτωση λοιπόν που ο χρήστης δεν επιθυμεί να συνεχίσει, ο βρόχος διακόπτεται άμεσα με την εντολή break (η οποία περιλαμβάνεται στο κυρίως μέρος της εντολής if της σειράς 21), ανεξάρτητα από το γεγονός ότι η συνθήκη του εξακολουθεί να ισχύει.

Η εντολή print της σειράς 24 τυπώνει μια κενή σειρά στην κονσόλα, εφόσον διαθέτει μια κενή συμβολοσειρά ως όρισμα.

Τέλος, στο σημείο αυτό, η συνάρτηση είναι ολοκληρωμένη. Βέβαια, όπως αναφέρθηκε και παραπάνω στον ορισμό των συναρτήσεων, για να τεθεί σε λειτουργία μια συνάρτηση, πρέπει να την καλέσουμε. Σε ένα πρόγραμμα καλούμε μια συνάρτηση μέσω της

ονομασίας της. Στη σειρά 26 λοιπόν, καλούμε τη συνάρτηση find_correct_choice() προκειμένου να τη θέσουμε σε λειτουργία.

3.3.2 Δεύτερη μορφή του προγράμματος

Όπως αναφέρθηκε και στο κομμάτι 3.3.1, μετά από συζήτηση με τον επιβλέποντα, κρίθηκε καταλληλότερος ο μετασχηματισμός του προγράμματος και του τρόπου λειτουργίας του.

Το πρόγραμμα εξακολουθεί βέβαια να διακρίνει εάν η πρόταση είναι σωστή ή λανθασμένη με βάση τη θέση του ρήματος μέσα στην πρόταση. Συγκεκριμένα, οι προτάσεις που διαθέτουν σύνθετο ρήμα με πρόθεση σε προστακτική στην πρώτη θέση θεωρούνται σωστές, ενώ παράλληλα οι προτάσεις που διαθέτουν σύνθετο ρήμα με πρόθεση σε οριστική σε οποιαδήποτε θέση της πρότασης πλην της πρώτης, θεωρούνται σωστές. Η βασικότερη διαφορά στον τρόπο λειτουργίας (σε σχέση με την αρχική μορφή του προγράμματος), είναι ότι, με την τωρινή του μορφή, το πρόγραμμα ζητάει από τον χρήστη να εισαγάγει μία πρόταση (και όχι δύο), την οποία επεξεργάζεται με βάση τις εντολές που έχουν εισαχθεί στο πρόγραμμα και, στη συνέχεια, ενημερώνει τον χρήστη σχετικά με το εάν η πρόταση που εισήγαγε είναι σωστή ή λανθασμένη. Σε περίπτωση που η πρόταση είναι λανθασμένη, το πρόγραμμα παρέχει στον χρήστη την αντίστοιχη σωστή πρόταση. Η διάκριση ανάμεσα σε σωστή και λανθασμένη πρόταση σχετίζεται φυσικά με την ανάγκη παρουσίας ή απουσίας της εσωτερικής αύξησης, δηλαδή με το εάν η πρόταση χρειάζεται τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου στην ενεργητική φωνή ή δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή. Παραδείγματος χάρη:

22) Ο διευθυντής απέβαλε τον μαθητή από το σχολείο.
23) * Απέβαλε τον μαθητή αμέσως, αυτή είναι η γνώμη μου.

Εάν εισαγάγουμε στο πρόγραμμα την πρόταση του παραδείγματος 22, το εξαγόμενο θα είναι το ακόλουθο: «Η πρότασή σας είναι σωστή». Στην περίπτωση αυτή, το ρήμα «αποβάλλω» βρίσκεται στο τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου στην ενεργητική φωνή, επομένως η παρουσία της εσωτερικής αύξησης είναι σωστή. Ακόμη, το ρήμα εντοπίζεται στην τρίτη θέση της πρότασης, επομένως το πρόγραμμα είναι σε θέση να πραγματοποιήσει ορθή διάκριση.

Στη συνέχεια, εάν εισαγάγουμε στο πρόγραμμα την πρόταση του παραδείγματος 23, το εξαγόμενο θα είναι το ακόλουθο «Λάθος. Η σωστή πρόταση είναι: Απόβαλε τον μαθητή αμέσως, αυτή είναι η γνώμη μου.» Σε αυτή την περίπτωση, χρειαζόμαστε δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή, δηλαδή τον τύπο «απόβαλε» του ρήματος «αποβάλλω». Επομένως, η παρουσία της εσωτερικής αύξησης δεν είναι επιθυμητή και, ως εκ τούτου, ο τύπος «απέβαλε» είναι λανθασμένος. Επιπλέον, το πρόγραμμα επιτυγχάνει ορθή διάκριση, καθώς το ρήμα εντοπίζεται στην πρώτη θέση της πρότασης.

Οι περιορισμοί του προγράμματος και οι περιπτώσεις λανθασμένων διακρίσεων παραμένουν ίδιες, καθώς εξακολουθεί να λειτουργεί με κριτήριο τη θέση του ρήματος στην πρόταση. Επομένως, έχουμε τις ακόλουθες προβληματικές περιπτώσεις:

24) Απέβαλε τον μαθητή γιατί απηύδησε με τη συμπεριφορά του.
25) Λάθος. Η σωστή πρόταση είναι: Απόβαλε τον μαθητή γιατί απηύδησε με τη συμπεριφορά του.
26) Σε παρακαλώ, απόβαλε αυτές τις κακές συνήθειες.
27) Λάθος. Η σωστή πρόταση είναι: Σε παρακαλώ, απέβαλε αυτές τις κακές συνήθειες.

Η πρόταση του παραδείγματος 24 είναι σωστή, καθώς χρειαζόμαστε το ρήμα «αποβάλλω» σε οριστική, επομένως είναι αναγκαία η παρουσία της εσωτερικής αύξησης. Ωστόσο, επειδή το ρήμα εντοπίζεται στην πρώτη θέση της πρότασης, το πρόγραμμα κρίνει λανθασμένα ότι χρειαζόμαστε προστακτική και, ως εκ τούτου, παρέχει το εξαγόμενο του παραδείγματος 25.

Παρομοίως, η πρόταση του παραδείγματος 26 είναι σωστή, καθώς το ρήμα «αποβάλλω» βρίσκεται σε προστακτική, επομένως η παρουσία εσωτερικής αύξησης δεν είναι επιθυμητή. Εντούτοις, επειδή το ρήμα εντοπίζεται σε θέση της πρότασης διαφορετική από την πρώτη, το πρόγραμμα κρίνει λανθασμένα ότι είναι απαραίτητη η παρουσία οριστικής, γεγονός που οδηγεί στο εξαγόμενο του παραδείγματος 27.

3.3.2.1 Πλεονεκτήματα του μετασχηματισμού

Από τον εν λόγω μετασχηματισμό του προγράμματος, προέκυψαν δύο βασικά οφέλη. Αρχικά, η χρήση του προγράμματος καθίσταται ευκολότερη, καθώς ο ομιλητής εισαγάγει

μόνο μία πρόταση, η οποία περιέχει μόνο τον ρηματικό τύπο που χρησιμοποιεί γενικά και, στη συνέχεια, το πρόγραμμα παρέχει ενημέρωση αναφορικά με την ορθότητα της πρότασής του. Ο ομιλητής λοιπόν δεν καλείται να γνωρίζει και τους δύο υπό εξέταση τύπους (όπως συνέβαινε με την πρώτη μορφή του προγράμματος) προκειμένου να χρησιμοποιήσει το πρόγραμμα.

Ακόμη, εφόσον οι χρήστες εισάγουν μόνο μία πρόταση, είναι δυνατός ο έλεγχος της ορθότητας των επιλογών τους και, κατ' επέκταση, ο έλεγχος της ερευνητικής υπόθεσης της παρούσας εργασίας. Συγκεκριμένα, το πρόγραμμα αναπτύχθηκε με βάση την υπόθεση ότι υψηλό ποσοστό των φυσικών ομιλητών της Ελληνικής πραγματοποιούν λανθασμένη χρήση της εσωτερικής αύξησης, γεγονός που οδηγεί στη σύγχυση ανάμεσα στο τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου στην ενεργητική φωνή και το δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή. Επομένως, κατά το στάδιο των δοκιμών του προγράμματος, αφενός προέκυψαν ποσοστά επιτυχίας του προγράμματος και αφετέρου προέκυψαν ποσοστά ορθής και λανθασμένης χρήσης της εσωτερικής αύξησης από τους ομιλητές της Ελληνικής (για τις δοκιμές του προγράμματος και τα ποσοστά επιτυχίας βλ. ενότητες 3.6, 4.1 και 4.2).

3.3.2.2 Παρουσίαση της τελικής μορφής του προγράμματος

Όπως αναφέρθηκε και στο 3.3.1.1, οι δείκτες (indices) που αναφέρονται στη θέση που διαθέτει σε μια λίστα καθένα από τα στοιχεία, παίζουν πολύ σημαντικό ρόλο στη λειτουργία του προγράμματος της παρούσας εργασίας και κυρίως (όπως θα γίνει εμφανές και παρακάτω) στη λειτουργία της δεύτερης και τελικής μορφής του προγράμματος.

Αρχικά, κατά τον μετασχηματισμό του προγράμματος δεν πραγματοποιήθηκαν αλλαγές αναφορικά με τις λίστες στις οποίες βασίζεται το πρόγραμμα για να εντοπίσει τα υπό εξέταση ρήματα στην πρόταση. Επομένως, στην αρχή του προγράμματος εντοπίζονται η list_a (η οποία περιλαμβάνει τα υπό εξέταση ρήματα στο τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου στην ενεργητική φωνή) και η list_b (η οποία περιλαμβάνει τα υπό εξέταση ρήματα στο δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή). Στις δύο λίστες, τα ρήματα έχουν εισαχθεί με τη μορφή συμβολοσειρών.

Στο σημείο αυτό λοιπόν, αυτό που θα μας απασχολήσει είναι η συνάρτηση του προγράμματος, στην οποία εντοπίζονται σημαντικές διαφορές σε σχέση με τη συνάρτηση που διέθετε το πρόγραμμα στην αρχική του μορφή.

3.3.2.2.1 Η συνάρτηση της τελικής μορφής του προγράμματος

Στο σημείο αυτό παρατίθεται η συνάρτηση του προγράμματος, προκειμένου να παρουσιάσουμε κάθε μέρος της αναλυτικά:

```
1. def correct_choice():
2.     while 1 < 2:
3.         sentence_in_question = input("Παρακαλώ εισαγάγετε πρόταση: ")
4.         split_sentence_in_question = sentence_in_question.split()
5.         lowercase_sentence_in_question = sentence_in_question.lower()
6.         split_sentence_in_question_2 = lowercase_sentence_in_question.split()
7.         if split_sentence_in_question_2[0] in list_b:
8.             print ("Η πρότασή σας είναι σωστή")
9.         elif split_sentence_in_question_2[0] in list_a:
10.            index_in_question = list_a.index(split_sentence_in_question_2[0])
11.            split_sentence_in_question_2[0] = list_b[index_in_question]
12.            final_sentence = " ".join(split_sentence_in_question_2)
13.            final_sentence = final_sentence[0].upper() + final_sentence[1:]
14.            print ("Λάθος. Η σωστή πρόταση είναι: %s" % final_sentence)
15.        else:
16.            for x in split_sentence_in_question_2:
17.                if x in list_a:
18.                    print ("Η πρότασή σας είναι σωστή")
19.                elif x in list_b:
20.                    index_in_list = list_b.index(x)
21.                    index_in_sentence = \
22.                    split_sentence_in_question_2.index(x)
23.                    split_sentence_in_question[index_in_sentence] = \
24.                    list_a[index_in_list]
25.                    final_sentence = " ".join(split_sentence_in_question)
26.                    print ("Λάθος. Η σωστή πρόταση είναι: %s" \
27.                    % final_sentence)
28.        print ("")
29.        answer = input("Επιθυμείτε να συνεχίσετε; ")
30.        if answer == 'OXI' or answer == 'Όχι' or answer == 'Οχι' or answer == 'όχι' \
31.        or answer == 'οχι':
32.            break
33.        print ("")
34.
35. correct_choice()
```

Όπως και στο 3.3.1.2.2, η συνάρτηση παρατίθεται με αριθμημένες σειρές, ώστε να είναι ευκολότερη η αναφορά σε συγκεκριμένα σημεία της.

Αρχικά, στην πρώτη σειρά εντοπίζεται η δεσμευμένη λέξη def και η ονομασία της συνάρτησης (correct_choice). Η εν λόγω συνάρτηση δεν απαιτεί συγκεκριμένα ορίσματα και, για τον λόγο αυτό, οι παρενθέσεις είναι κενές.

Στη δεύτερη σειρά εντοπίζεται ένας βρόχος while με τη συνθήκη 1 < 2, η οποία ισχύει πάντοτε και, ως εκ τούτου, τον καθιστά ατέρμονα. Η περίληψη της συγκεκριμένης συνθήκης έχει πραγματοποιηθεί, όπως θα δούμε και παρακάτω, για να είναι δυνατή η επανεκτέλεση του προγράμματος και, συνεπώς, η εισαγωγή περαιτέρω προτάσεων από τον χρήστη.

Στην τρίτη σειρά βλέπουμε ότι η εντολή input τυπώνει τη συμβολοσειρά "Παρακαλώ εισαγάγετε πρόταση: " και, στη συνέχεια, καταχωρεί την πρόταση του χρήστη στη μεταβλητή sentence_in_question.

Στη σειρά 4, η πρόταση του χρήστη χωρίζεται σε λέξεις, εφόσον, από ενιαία συμβολοσειρά, μετατρέπεται σε λίστα με κάθε λέξη να αποτελεί ξεχωριστό στοιχείο. Στη συνέχεια, η λίστα αυτή καταχωρείται στη μεταβλητή split_sentence_in_question. Η συγκεκριμένη διαδικασία καθίσταται δυνατή με τη χρήση της μεθόδου .split().

Στην πέμπτη σειρά, μέσω της μεθόδου .lower(), οι χαρακτήρες της αρχικής πρότασης του χρήστη μετατρέπονται σε πεζούς και το εξαγόμενο καταχωρείται στη μεταβλητή lowercase_sentence_in_question.

Στη σειρά 6, η μεταβλητή lowercase_sentence_in_question, που αποτελεί την πρόταση του χρήστη με μορφή ενιαίας συμβολοσειράς και όλους τους χαρακτήρες πεζούς, χωρίζεται σε λέξεις με τη μέθοδο .split() και καταχωρείται στη μεταβλητή split_sentence_in_question_2.

Στο σημείο αυτό, η εντολή της σειράς 4 ενδέχεται να φαίνεται πλεοναστική, καθώς η εντολή της σειράς 5 μετατρέπει τους χαρακτήρες της πρότασης του χρήστη σε πεζούς, ενώ η εντολή της σειράς 6 χωρίζει την πρόταση σε λέξεις. Επομένως, η μεταβλητή split_sentence_in_question_2 ισοδυναμεί με την πρόταση του χρήστη χωρισμένη σε λέξεις και με όλους τους χαρακτήρες πεζούς. Παράλληλα, η μεταβλητή split_sentence_in_question ισοδυναμεί με την πρόταση του χρήστη χωρισμένη σε λέξεις, χωρίς να έχει πραγματοποιηθεί κάποια αλλαγή αναφορικά με τους κεφαλαίους χαρακτήρες. Όπως θα γίνει εμφανές και παρακάτω, η εντολή της σειράς 4 εξυπηρετεί συγκεκριμένο σκοπό και δεν είναι πλεοναστική σε καμία περίπτωση.

Συνεχίζοντας στη σειρά 7, βλέπουμε ότι ξεκινούν οι έλεγχοι του προγράμματος, προκειμένου να καθοριστεί εάν η πρόταση του χρήστη είναι σωστή ή λανθασμένη. Συγκεκριμένα, η εντολή if ελέγχει εάν η λέξη με δείκτη 0 (δηλαδή η πρώτη λέξη της

πρότασης) στη μεταβλητή split_sentence_in_question_2 περιέχεται στη list_b. Εάν ισχύει κάτι τέτοιο, σημαίνει ότι στην πρώτη θέση της πρότασης υπάρχει σύνθετο ρήμα με πρόθεση σε προστακτική, επομένως η εντολή print της σειράς 8 τυπώνει στην κονσόλα τη συμβολοσειρά "Η πρότασή σας είναι σωστή", ώστε να ενημερώσει τον χρήστη σχετικά με την ορθότητα της πρότασής του.

Συνεχίζοντας με την εξέταση της πρώτης λέξης της πρότασης του χρήστη, η εντολή elif στη σειρά 9 ελέγχει εάν υπάρχει σύνθετο ρήμα με πρόθεση σε οριστική στην πρώτη θέση της πρότασης. Εάν όντως ισχύει κάτι τέτοιο, η πρόταση κρίνεται ως λανθασμένη και είναι αναγκαία η διόρθωσή της από το πρόγραμμα. Συγκεκριμένα, το τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου ενεργητικής φωνής πρέπει να αντικατασταθεί από το δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου ενεργητικής φωνής του ίδιου ρήματος, που θα περιέχεται στη list_b. Τον βασικότερο ρόλο για τη συγκεκριμένη διαδικασία τον διαδραματίζει ο δείκτης που διαθέτει το ρήμα στη λίστα. Για τον λόγο αυτό αναφέρθηκε παραπάνω ότι ο δείκτης λίστας είναι ιδιαίτερα σημαντικός για τη λειτουργία της δεύτερης και τελικής μορφής του προγράμματος.

Για την ακρίβεια, κάθε ρήμα διαθέτει τον ίδιο δείκτη και στη list_a και στη list_b. Παραδείγματος χάρη, το ρήμα «παραδίδω» διαθέτει δείκτη 3 και στις δύο λίστες, επομένως στην τέταρτη θέση της list_a εντοπίζεται ο τύπος «παρέδωσε», ενώ στην τέταρτη θέση της list_b εντοπίζεται ο τύπος «παράδωσε» (υπενθυμίζουμε ότι μιλάμε για δείκτη 3 και τέταρτη θέση, διότι η αρίθμηση των δεικτών ξεκινάει από το 0 και όχι από το 1). Με τον τρόπο αυτό καθίσταται δυνατή η αντικατάσταση του τύπου οριστικής ενός ρήματος από τον τύπο προστακτικής και αντίστροφα.

Ας δούμε αναλυτικά όμως πως πραγματοποιείται η αντικατάσταση και, στην προκειμένη περίπτωση, η αντικατάσταση της οριστικής από την προστακτική. Έχοντας διαπιστώσει λοιπόν ότι το σύνθετο ρήμα με πρόθεση στην πρώτη θέση της πρότασης βρίσκεται σε οριστική και, ως εκ τούτου, περιέχεται στη list_a, η εντολή της σειράς 10 εντοπίζει τον δείκτη του ρήματος στη list_a. Συγκεκριμένα, χρησιμοποιείται η μέθοδος .index() η οποία βρίσκει τον δείκτη του στοιχείου που δέχεται ως όρισμα (στην προκειμένη περίπτωση την πρώτη λέξη της πρότασης του χρήστη) στη λίστα που της παρέχουμε αριστερά από την τελεία (στη συγκεκριμένη περίπτωση στη list_a). Ο συγκεκριμένος δείκτης καταχωρείται στη μεταβλητή index_in_question.

Στο σημείο αυτό λοιπόν, έχει εντοπιστεί ο δείκτης που διαθέτει το ρήμα στη list_a. Αυτό που χρειάζεται να κάνουμε λοιπόν, είναι να το αντικαταστήσουμε με το ρήμα που διαθέτει τον ίδιο δείκτη στη list_b, το οποίο, όπως αναφέρθηκε παραπάνω, θα είναι το ίδιο

ρήμα σε προστακτική. Επομένως, στη σειρά 11 θέτουμε το split_sentence_in_question_2[0] (δηλαδή το ρήμα που βρίσκεται στην πρώτη θέση της πρότασης του χρήστη) ίσο με το ρήμα της list_b που διαθέτει τον υπό συζήτηση δείκτη (index_in_question). Με τον τρόπο αυτό, το ρήμα της list_a αντικαθίσταται από το ρήμα της list_b που διαθέτει τον ίδιο δείκτη. Στο σημείο αυτό, αξίζει να αναφερθεί ότι στην Python, μπορούμε να αναφερόμαστε σε συγκεκριμένο στοιχείο μιας λίστας, παρέχοντας την ονομασία της λίστας και τον δείκτη του στοιχείου μέσα σε αγκύλες. Η δυνατότητα αυτή είναι εμφανής στη σειρά 11 και συγκεκριμένα στην εντολή list_b[index_in_question]. Για την ακρίβεια, στο παράδειγμα 14 του τμήματος 3.3.1.1, η εντολή print (σύντομη_λίστα[3]) θα τυπώσει στην κονσόλα τον αριθμό 24 (που έχει δείκτη 3 στη σύντομη_λίστα).

Έχοντας αντικαταστήσει το σύνθετο ρήμα σε οριστική με το ίδιο ρήμα σε προστακτική, το πρόγραμμα πρέπει να τυπώσει τη νέα πρόταση στην κονσόλα, προκειμένου να είναι διαθέσιμη στον χρήστη. Αρχικά, να αναφερθεί ότι, στο σημείο αυτό, η πρόταση είναι χωρισμένη σε λέξεις και δεν αποτελεί ενιαία συμβολοσειρά. Για να ενώσουμε τις λέξεις, χρησιμοποιούμε τη μέθοδο .join(), η οποία ενώνει τα στοιχεία της λίστας που δέχεται ως όρισμα, τοποθετώντας μεταξύ τους οτιδήποτε της παρέχεται αριστερά από την τελεία. Συγκεκριμένα, στη σειρά 12 χρησιμοποιούμε τη μέθοδο .join() με όρισμα τη μεταβλητή split_sentence_in_question_2 (η οποία, στο σημείο αυτό, διαθέτει ρήμα με προστακτική έπειτα από τη διαδικασία της σειράς 11). Ακόμη, αριστερά από την τελεία της μεθόδου .join() τοποθετούμε τη συμβολοσειρά " ", προκειμένου να τοποθετηθεί ένα κενό ανάμεσα στις λέξεις της μεταβλητής split_sentence_in_question_2 κατά την ένωσή τους και τη δημιουργία ενιαίας συμβολοσειράς. Το τελικό εξαγόμενο καταχωρείται στη μεταβλητή final_sentence.

Τα υπό εξέταση ρήματα έχουν εισαχθεί στις list_a και list_b με όλους τους χαρακτήρες πεζούς. Επομένως, το σύνθετο ρήμα με πρόθεση σε προστακτική που τοποθετήσαμε στην πρώτη θέση της final_sentence, δεν διαθέτει κεφαλαίο το πρώτο γράμμα. Για τον λόγο αυτό, στη σειρά 13 γίνεται χρήση της μεθόδου .upper(), η οποία μετατρέπει σε κεφαλαία τη συμβολοσειρά που δέχεται αριστερά από την τελεία. Επομένως, με την εντολή final_sentence[0].upper(), το πρώτο γράμμα της πρότασης μετατρέπεται σε κεφαλαίο. Όμως είναι αναγκαίο να προσθέσουμε το κεφαλαίο γράμμα στην υπόλοιπη πρόταση. Η διαδικασία αυτή πραγματοποιείται με τη χρήση δεικτών. Οι δείκτες, που έχουμε αναφέρει σε αρκετά σημεία της εργασίας, μπορούν να αξιοποιηθούν όχι μόνο σε λίστες, αλλά και σε συμβολοσειρές και λειτουργούν με όμοιο τρόπο. Στη σειρά 13 λοιπόν, το πρώτο γράμμα της πρότασης που έχει μετατραπεί σε κεφαλαίο ενώνεται με τη final_sentence και συγκεκριμένα

με το τμήμα της από τον δεύτερο χαρακτήρα μέχρι το τέλος της. Συγκεκριμένα, ο δείκτης 1 αναφέρεται στον δεύτερο χαρακτήρα της συμβολοσειράς (εφόσον η αρίθμηση των δεικτών ξεκινάει από το 0 και στην περίπτωση των συμβολοσειρών), ενώ η χρήση του συμβόλου « : » διαθέτει την έννοια «μέχρι το τέλος της συμβολοσειράς», χωρίς να είναι αναγκαία η περίληψη συγκεκριμένου δείκτη. Με τον τρόπο αυτό, μπορούμε να αναφερόμαστε στο τέλος της συμβολοσειράς, χωρίς να γνωρίζουμε την ακριβή έκταση της εκάστοτε συμβολοσειράς. Με τον τρόπο αυτό, ο πρώτος χαρακτήρας της final_sentence μετατρέπεται σε κεφαλαίο.

Στη σειρά 14, η εντολή print αναλαμβάνει να τυπώσει στην κονσόλα το τελικό εξαγόμενο, καθώς δέχεται ως όρισμα το ακόλουθο: "Λάθος. Η σωστή πρόταση είναι: %s" % final_sentence. Συγκεκριμένα, το πρόγραμμα επισημαίνει στον χρήστη ότι η πρόταση που εισήγαγε είναι λανθασμένη και του παρέχει τη σωστή πρόταση. Όπως αναφέρθηκε και στο κομμάτι 3.3.1.2.2, το σύμβολο %s αντικαθίσταται από ό,τι καθορίσει ο δημιουργός του προγράμματος. Στην προκειμένη περίπτωση, αντικαθίσταται από τη μεταβλητή final_sentence, που αποτελεί την αρχική πρόταση του χρήστη, η οποία όμως έχει υποστεί τις καθορισμένες διαδικασίες του προγράμματος.

Στο σημείο αυτό, έχει ολοκληρωθεί ο έλεγχος της πρώτης θέσης της πρότασης του χρήστη. Συγκεκριμένα, εάν εντοπιστεί σύνθετο ρήμα με πρόθεση σε προστακτική, το πρόγραμμα δηλώνει στον χρήστη ότι η πρότασή του είναι σωστή. Αντιθέτως, εάν εντοπιστεί σύνθετο ρήμα με πρόθεση σε οριστική στην πρώτη θέση, το πρόγραμμα κρίνει την πρόταση του χρήστη ως λανθασμένη και του παρέχει την αντίστοιχη σωστή. Στη συνέχεια λοιπόν, το πρόγραμμα συνεχίζει με τον έλεγχο των υπόλοιπων θέσεων της πρότασης του χρήστη, προκειμένου να εντοπίσει κάποιο σύνθετο ρήμα με πρόθεση.

Στη σειρά 15 εντοπίζεται η εντολή else, η οποία εκτελείται μόνο σε περίπτωση που δεν ισχύουν οι συνθήκες των εντολών if και elif των σειρών 7 και 9 αντίστοιχα.

Στη σειρά 16 παρατηρούμε έναν βρόχο for, ο οποίος θα εκτελέσει ορισμένες διαδικασίες για κάθε στοιχείο x που περιλαμβάνεται στη μεταβλητή split_sentence_in_question_2, δηλαδή για κάθε λέξη της πρότασης του χρήστη.

Στη σειρά 17, η εντολή if ελέγχει εάν κάποια από τις λέξεις της πρότασης του χρήστη περιέχεται στη list_a. Δηλαδή η εντολή if ελέγχει εάν η πρόταση του χρήστη περιλαμβάνει σύνθετο ρήμα με πρόθεση σε οριστική σε οποιαδήποτε θέση. Εάν η συνθήκη της εντολής if ισχύει, εκτελείται η εντολή της σειράς 18.

Συγκεκριμένα, η πρόταση του χρήστη χαρακτηρίζεται ως σωστή και η εντολή print τυπώνει στην κονσόλα τη συμβολοσειρά "Η πρότασή σας είναι σωστή", εφόσον το

πρόγραμμα κρίνει ως σωστές τις προτάσεις που περιλαμβάνουν σύνθετο ρήμα με πρόθεση σε οριστική σε οποιαδήποτε θέση της πρότασης πλην της πρώτης.

Σε περίπτωση που η συνθήκη της σειράς 17 δεν ισχύει, εκτελείται η εντολή elif της σειράς 19, η οποία ελέγχει εάν κάποια από τις λέξεις της πρότασης του χρήστη περιέχεται στη list_b, δηλαδή εάν αποτελεί σύνθετο ρήμα με πρόθεση σε δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου στην ενεργητική φωνή. Εάν ισχύει κάτι τέτοιο, η πρόταση χαρακτηρίζεται ως λανθασμένη (εφόσον, με βάση τον τρόπο λειτουργίας του προγράμματος, η προστακτική είναι αποδεκτή μόνο στην πρώτη θέση της πρότασης) και ξεκινούν οι διαδικασίες αντικατάστασης του σύνθετου ρήματος σε προστακτική με σύνθετο ρήμα σε οριστική.

Στη σειρά 20 πραγματοποιείται χρήση της μεθόδου .index(), η οποία εντοπίζει τον δείκτη που διαθέτει το x (το σύνθετο ρήμα με πρόθεση της πρότασης του χρήστη) στη list_b και τον καταχωρεί στη μεταβλητή index_in_list.

Στη συνέχεια, η εντολή των σειρών 21 και 22 εντοπίζει τον δείκτη που διαθέτει το x στη μεταβλητή split_sentence_in_question_2, δηλαδή στην επεξεργασμένη πρόταση του χρήστη, και τον καταχωρεί στη μεταβλητή index_in_sentence. Όπως έχει αναφερθεί και παραπάνω, το σύμβολο « \ » επιτρέπει στο πρόγραμμα να αντιμετωπίζει το περιεχόμενο της σειράς 21 και το περιεχόμενο της σειράς 22 ως ενιαία εντολή.

Στο σημείο αυτό έχουμε λοιπόν τον δείκτη που διαθέτει το ρήμα στη list_b (index_in_list) και τον δείκτη που διαθέτει το ρήμα στην πρόταση (index_in_sentence), με τους οποίους καθίσταται δυνατή η αντικατάσταση του σύνθετου ρήματος σε προστακτική από το αντίστοιχο σύνθετο ρήμα σε οριστική.

Συγκεκριμένα, στις σειρές 23 και 24 βλέπουμε ότι το σύνθετο ρήμα σε προστακτική που βρίσκεται στην πρόταση στον δείκτη που έχουμε ήδη εντοπίσει (index_in_sentence) αντικαθίσταται από το στοιχείο της list_a που διαθέτει δείκτη ίδιο με τον δείκτη που διαθέτει το εν λόγω ρήμα στη list_b. Επομένως, εφόσον, όπως έχουμε ήδη αναφέρει, κάθε ρήμα διαθέτει τον ίδιο δείκτη και στις δύο λίστες, μέσω της εντολής των σειρών 23 και 24, το σύνθετο ρήμα με πρόθεση σε προστακτική αντικαθίσταται από το ίδιο ρήμα σε οριστική, που περιέχεται στη list_a.

Ακόμη, στη σειρά 23 παρατηρούμε ότι δεν αξιοποιείται η μεταβλητή split_sentence_in_question_2 αλλά η μεταβλητή split_sentence_in_question που κάνει αρχικά την εμφάνισή της στη σειρά 4 της συνάρτησης. Αυτό συμβαίνει διότι, σε αντίθεση με τις εντολές των σειρών 10 έως 14, η αντικατάσταση που λαμβάνει χώρα στις σειρές 20 έως 27 αφορά λέξη της πρότασης που εντοπίζεται σε οποιαδήποτε θέση πλην της πρώτης.

Επομένως, δεν είναι απαραίτητη η μετατροπή του πρώτου χαρακτήρα του ρήματος σε κεφαλαίο (όπως συμβαίνει στη σειρά 13). Για τον λόγο αυτό, αξιοποιείται η μεταβλητή split_sentence_in_question, η οποία αποτελεί την πρόταση που έχει εισαγάγει ο χρήστης χωρισμένη σε λέξεις, χωρίς κάποια μετατροπή των χαρακτήρων σε πεζούς. Με τον τρόπο αυτό, ο κεφαλαίος πρώτος χαρακτήρας που έχει χρησιμοποιήσει ο χρήστης, κατά πάσα πιθανότητα, διατηρείται και έτσι, στο σημείο αυτό, δεν είναι αναγκαία κάποια διαδικασία αντίστοιχη με την εντολή της σειράς 13. Αυτός λοιπόν είναι ο σκοπός που εξυπηρετεί η εντολή της σειράς 4 και για αυτόν τον λόγο επισημάνθηκε παραπάνω ότι δεν είναι πλεοναστική.

Επομένως, έχοντας αντικαταστήσει το σύνθετο ρήμα με πρόθεση σε προστακτική με το αντίστοιχο σύνθετο ρήμα σε οριστική, είναι αναγκαία η ένωση των στοιχείων της μεταβλητής split_sentence_in_question, προκειμένου να αποτελούν ενιαία συμβολοσειρά. Στη σειρά 25 λοιπόν, θέτουμε τη μεταβλητή final_sentence ίση με τη μέθοδο .join(), η οποία ενώνει τις λέξεις της λίστας split_sentence_in_question, τοποθετώντας παράλληλα ένα κενό ανάμεσά τους (" ").

Στις σειρές 26 και 27, η εντολή print τυπώνει τη συμβολοσειρά "Λάθος. Η σωστή πρόταση είναι: " που ακολουθείται από τη μεταβλητή final_sentence (όπως έχει καθοριστεί) η οποία, στο σημείο αυτό, αποτελεί την αρχική πρόταση του χρήστη, από την οποία έχει αφαιρεθεί το ρήμα σε προστακτική και έχει εισαχθεί το αντίστοιχο ρήμα σε οριστική. Με τον τρόπο αυτό παρέχεται στον χρήστη η πρόταση που είχε εισαγάγει αρχικά, αλλά διορθωμένη.

Στο σημείο αυτό αξίζει να σημειωθεί ο λόγος για τον οποίο ο βρόχος for στη σειρά 16 ελέγχει όλες τις λέξεις της πρότασης του χρήστη και όχι μόνο όσες βρίσκονται σε θέση διαφορετική από την πρώτη. Συγκεκριμένα, όσον αφορά τα σύνθετα ρήματα σε οριστική, το πρόγραμμα τα θεωρεί αποδεκτά όταν εντοπίζονται σε οποιαδήποτε θέση πλην της πρώτης, ένας περιορισμός που θα μπορούσε να έχει προστεθεί, όμως απουσιάζει από τον βρόχο for. Όπως όμως αναφέρθηκε και στο τμήμα 3.3.1.2.2 (στην παρουσίαση της πρώτης μορφής του προγράμματος), η περίληψη ενός περιορισμού δεν είναι απαραίτητη, καθώς το πρόγραμμα εξετάζει τις εντολές με τη σειρά που έχουν εισαχθεί. Για την ακρίβεια, στο σημείο αυτό έχει ήδη πραγματοποιηθεί έλεγχος της πρώτης θέσης της πρότασης για παρουσία σύνθετου ρήματος με πρόθεση και έχουμε φτάσει στη σειρά 16 μόνο εάν δεν ισχύουν οι συνθήκες των σειρών 7 ή 9. Για τον λόγο αυτό, η περίληψη περιορισμού στον βρόχο for της σειράς 16 θα ήταν περιττή.

Η εντολή print της σειράς 28 διαθέτει ως όρισμα μια κενή συμβολοσειρά, επομένως τυπώνει μια κενή σειρά στην κονσόλα και έχει περιληφθεί για αισθητικούς λόγους.

Όπως αναφέρθηκε και παραπάνω, οι εντολές της συνάρτησης correct_choice περιέχονται σε έναν ατέρμονα βρόχο while, εφόσον η συνθήκη του ισχύει πάντοτε (1 < 2). Η συγκεκριμένη επιλογή, επιτρέπει στη συνάρτηση να επανεκτελείται, δίνοντας στον χρήστη τη δυνατότητα να εισαγάγει περαιτέρω προτάσεις στο πρόγραμμα για έλεγχο. Ωστόσο, είναι αναγκαία η παροχή επιλογής στον χρήστη, αναφορικά με το εάν επιθυμεί να εκτελεστεί εκ νέου το πρόγραμμα, κάθε φορά που ολοκληρώνεται η εκτέλεσή του. Για αυτό, στη σειρά 29 χρησιμοποιείται η συνάρτηση input με όρισμα τη συμβολοσειρά "Επιθυμείτε να συνεχίσετε; ", την οποία και τυπώνει στην κονσόλα. Η απάντηση του χρήστη στην προαναφερθείσα ερώτηση καταχωρείται στη μεταβλητή answer.

Στη συνέχεια, στις σειρές 30 και 31, το πρόγραμμα ελέγχει εάν η απάντηση του χρήστη ισοδυναμεί με «όχι», ελέγχοντας κάθε πιθανή διατύπωση αναφορικά με τη χρήση κεφαλαίων ή πεζών χαρακτήρων και την απουσία ή παρουσία τόνου. Εάν η απάντηση του χρήστη ισοδυναμεί με «όχι», εκτελείται η εντολή break της σειράς 32, η οποία, όπως έχει αναφερθεί και παραπάνω, διακόπτει την εκτέλεση του βρόχου while (παρόλο που είναι ατέρμων) και, ως εκ τούτου, την εκτέλεση της συνάρτησης correct_choice.

Στη σειρά 33, η εντολή print τυπώνει ξανά μια κενή σειρά ώστε, σε περίπτωση επανεκτέλεσης της συνάρτησης, να μεσολαβεί μια κενή σειρά ανάμεσα στη συμβολοσειρά "Επιθυμείτε να συνεχίσετε; " και τη συμβολοσειρά "Παρακαλώ εισαγάγετε πρόταση: ".

Τέλος, στη σειρά 35 καλούμε τη συνάρτηση correct_choice() με τον συμβατικό τρόπο που έχουμε αναφέρει, ώστε να τη θέσουμε σε λειτουργία.

3.4 Εκτιμώμενα ποσοστά επιτυχίας του προγράμματος

Έχοντας ολοκληρώσει την ανάπτυξη του προγράμματος, ήταν αναγκαίο να εκτιμηθούν τα ποσοστά επιτυχίας του, δηλαδή ο βαθμός στον οποίο πραγματοποιεί ορθές διακρίσεις αναφορικά με τις προτάσεις που παρέχουν οι χρήστες.

Όπως αναφέρθηκε και παραπάνω, το πρόγραμμα, σύμφωνα με τον τρόπο λειτουργίας του, αναγνωρίζει ως σωστές τις προτάσεις που διαθέτουν σύνθετο ρήμα με πρόθεση σε προστακτική στην πρώτη θέση της πρότασης και τις προτάσεις που διαθέτουν σύνθετο ρήμα με πρόθεση σε οριστική σε οποιαδήποτε θέση πλην της πρώτης. Αντίθετα, το πρόγραμμα αναγνωρίζει ως λανθασμένες τις προτάσεις που διαθέτουν σύνθετο ρήμα με πρόθεση σε προστακτική σε οποιαδήποτε θέση πλην της πρώτης και τις προτάσεις που διαθέτουν σύνθετο ρήμα με πρόθεση σε οριστική στην πρώτη θέση της πρότασης.

Ανατρέχοντας λοιπόν στα παραδείγματα 9-12 του τμήματος 3.3, τα οποία καλύπτουν και τις 4 περιπτώσεις που μόλις αναφέραμε (ρήμα σε προστακτική στην πρώτη θέση και σε θέση πλην της πρώτης, ρήμα σε οριστική στην πρώτη θέση και σε θέση διαφορετική από την πρώτη) θα μπορούσαμε να πούμε ότι το ποσοστό επιτυχίας του προγράμματος ανέρχεται στο 50%, καθώς πραγματοποιεί ορθή διάκριση σε μόλις 2 από τις 4 πιθανές περιπτώσεις (στην προκειμένη περίπτωση πραγματοποιεί ορθή διάκριση για τα παραδείγματα 9 και 11 και λανθασμένη διάκριση για τα παραδείγματα 10 και 12).

Ωστόσο, ένας πολύ σημαντικός παράγοντας είναι η συχνότητα με την οποία προκύπτει καθεμία από τις 4 πιθανές περιπτώσεις. Όπως έχει ήδη αναφερθεί, σύμφωνα με την υπόθεση του γράφοντος, τις περισσότερες φορές που γίνεται χρήση προστακτικής, το ρήμα εντοπίζεται στην πρώτη θέση της πρότασης, ενώ παράλληλα, τις περισσότερες φορές που γίνεται χρήση οριστικής το ρήμα δεν εντοπίζεται στην πρώτη θέση, στην οποία συνήθως υπάρχει το υποκείμενο. Εκτιμάται λοιπόν ότι οι περιπτώσεις των παραδειγμάτων 9 και 11 είναι συχνότερες από τις περιπτώσεις των παραδειγμάτων 10 και 12, γεγονός που, αν ισχύει, αυξάνει το ποσοστό επιτυχίας του συγκεκριμένου προγράμματος.

3.5 Δοκιμές του προγράμματος με συμμετέχοντες

Προκειμένου να ελεγχθεί η παραπάνω υπόθεση, κρίθηκε αναγκαία η δοκιμή του προγράμματος με συμμετέχοντες. Όπως αναφέρθηκε και παραπάνω, οι δοκιμές επιτρέπουν αφενός τον έλεγχο του βαθμού στον οποίο το πρόγραμμα πραγματοποιεί ορθές διακρίσεις και αφετέρου, χάρη στη νέα μορφή που απέκτησε το πρόγραμμα, τον έλεγχο της ορθότητας των επιλογών των ομιλητών αναφορικά με την περίληψη ή παράλειψη της εσωτερικής αύξησης.

Συγκεκριμένα, εφόσον οι χρήστες εισάγουν μόνο μία πρόταση στο πρόγραμμα (σε αντίθεση με την αρχική μορφή του προγράμματος που ζητούσε δύο προτάσεις από τους χρήστες), είναι δυνατό να ελεγχθεί το ποσοστό στο οποίο χρησιμοποιούν ορθά την εσωτερική αύξηση στα σύνθετα ρήματα με προθέσεις.

3.5.1 Τροποποίηση του προγράμματος για τις δοκιμές

Όπως αναφέρθηκε και παραπάνω, σκοπός των δοκιμών ήταν τόσο ο έλεγχος των ποσοστών επιτυχίας του προγράμματος, όσο και ο έλεγχος των ποσοστών ορθής και λανθασμένης χρήσης της εσωτερικής αύξησης από τους ομιλητές.

Για να είναι δυνατός ο έλεγχος της ορθότητας των επιλογών των χρηστών, ήταν αναγκαίος ο μετασχηματισμός του προγράμματος για το στάδιο των δοκιμών. Συγκεκριμένα, όπως είδαμε παραπάνω, κάθε φορά που εισάγει μία πρόταση ο χρήστης, το πρόγραμμα τον ενημερώνει αναφορικά με την ορθότητά της. Αυτό το εξαγόμενο όμως, δεν είναι επιθυμητό κατά το στάδιο των δοκιμών, διότι θα επηρεάζει τους χρήστες και τις επιλογές τους αναφορικά με την εσωτερική αύξηση.

Για τον λόγο αυτό, το πρόγραμμα μετασχηματίστηκε με τέτοιο τρόπο ώστε να ζητάει από τον χρήστη να εισαγάγει 10 προτάσεις συνολικά, χωρίς να παρέχει εξαγόμενο σε οποιοδήποτε στάδιο των δοκιμών.

Οι αλλαγές που πραγματοποιήθηκαν στο πρόγραμμα αφορούν μόνο τη συνάρτηση correct_choice. Αρχικά αφαιρέθηκαν οι εντολές print που εντοπίζονται στο κυρίως μέρος των εντολών if και elif των σειρών 7, 9, 17 και 19 ώστε το πρόγραμμα να μην παρέχει στον χρήστη εξαγόμενο σε κανένα στάδιο της εισαγωγής προτάσεων.

Επίσης, πραγματοποιήθηκε αλλαγή στον βρόχο while της σειράς 2, ώστε να μην είναι πλέον ατέρμων. Συγκεκριμένα, επειδή είναι αναγκαίο να εισαγάγει 10 προτάσεις ο κάθε συμμετέχων, η συνθήκη 1 < 2 αντικαταστάθηκε από μία συνθήκη χάρη στην οποία, το πρόγραμμα εκτελείται 10 φορές.

Αρχικά, πριν από τον βρόχο while προστέθηκε η μεταβλητή count, η οποία τέθηκε ίση με το μηδέν. Στη συνέχεια, στον βρόχο while προστέθηκε η συνθήκη «count < 10». Ο συγκεκριμένος βρόχος δηλαδή εκτελείται όσο η μεταβλητή count είναι μικρότερη από το 10.

Ακόμη, στο τέλος της συνάρτησης correct_choice προστέθηκε η εντολή count += 1, η οποία προσθέτει το 1 στη μεταβλητή count. Κάθε φορά που εκτελείται η συνάρτηση λοιπόν, προστίθεται το 1 στη μεταβλητή count, την οποία έχουμε θέσει αρχικά ίση με 0. Η συνάρτηση correct_choice θα εκτελεστεί λοιπόν 10 φορές συνολικά, καθώς την ενδέκατη φορά που θα ξεκινήσει να εκτελείται, η μεταβλητή count θα είναι ίση με το 10, επομένως η συνθήκη του βρόχου while δεν θα ισχύει και, ως εκ τούτου, η εκτέλεση θα διακοπεί.

Τέλος, έχουν αφαιρεθεί και οι εντολές των σειρών 29-33 διότι, όπως αναφέρθηκε παραπάνω, στο στάδιο των δοκιμών κάθε χρήστης πρέπει να εισαγάγει 10 προτάσεις συνολικά. Επομένως, δεν παρέχεται η επιλογή εκτέλεσης του προγράμματος για όσες φορές επιθυμεί ο χρήστης.

3.6 Αποτελέσματα δοκιμών

3.6.1 Πιθανά σενάρια

Όπως έχει ήδη αναφερθεί, κάθε συμμετέχων εισήγαγε 10 προτάσεις στο πρόγραμμα κατά το στάδιο των δοκιμών. Επομένως, ο συνολικός αριθμός των προτάσεων ανέρχεται στις 200 (20 ομιλητές, από 10 προτάσεις ο καθένας).

Κάθε φορά που ο χρήστης εισήγαγε μία πρόταση, προέκυπτε ένα από τα ακόλουθα σενάρια, ανάλογα με την επιτυχία ή αποτυχία του προγράμματος και την εισαγωγή γραμματικά αποδεκτής ή μη αποδεκτής πρότασης από τον χρήστη:

Σενάριο 1 Χρήστης: 0 Πρόγραμμα: 0	**Σενάριο 2** Χρήστης: 0 Πρόγραμμα: 1
Σενάριο 3 Χρήστης: 1 Πρόγραμμα: 0	**Σενάριο 4** Χρήστης: 1 Πρόγραμμα: 1

Πίνακας 1.

Στον παραπάνω πίνακα, παρουσιάζονται τα 4 διαφορετικά σενάρια που προκύπτουν κάθε φορά που ο χρήστης εισάγει μια πρόταση στο πρόγραμμα. Όσον αφορά τον χρήστη, το 0 αναφέρεται σε περιπτώσεις που γίνεται λανθασμένη χρήση της εσωτερικής αύξησης, ενώ το 1 αναφέρεται σε περιπτώσεις που γίνεται ορθή χρήση της εσωτερικής αύξησης. Όσον αφορά το πρόγραμμα, το 0 αναφέρεται σε περιπτώσεις που το πρόγραμμα πραγματοποιεί λανθασμένη διάκριση αναφορικά με την πρόταση του χρήστη, ενώ το 1 αναφέρεται σε περιπτώσεις που το πρόγραμμα πραγματοποιεί σωστή διάκριση αναφορικά με την πρόταση του χρήστη.

Επομένως, οι πιθανοί συνδυασμοί των περιπτώσεων που μόλις αναφέρθηκαν, οδηγούν σε ένα από τα παραπάνω σενάρια. Συγκεκριμένα, ο πίνακας 2 παρουσιάζει πόσες προτάσεις ανήκουν σε κάθε σενάριο, από τις συνολικά 200 προτάσεις που εισήχθησαν στο πρόγραμμα κατά το στάδιο των δοκιμών:

Σενάριο 1	**Σενάριο 2**
21 προτάσεις	48 προτάσεις
Σενάριο 3	**Σενάριο 4**
33 προτάσεις	98 προτάσεις

Πίνακας 2.

Με βάση τον παραπάνω πίνακα γίνεται εμφανές ότι από τις 200 προτάσεις συνολικά, οι 98 αφορούν το σενάριο κατά το οποίο οι χρήστες χρησιμοποιούν σωστά την εσωτερική αύξηση, ενώ, παράλληλα, το πρόγραμμα κρίνει ορθά τις προτάσεις των χρηστών ως σωστές.

Το δεύτερο πιο συχνό σενάριο προέκυψε 48 φορές. Κατά το σενάριο αυτό, οι συμμετέχοντες πραγματοποίησαν λανθασμένη χρήση της εσωτερικής αύξησης, ενώ το πρόγραμμα ορθά έκρινε τις επιλογές τους ως λανθασμένες.

Στη συνέχεια, βλέπουμε ότι 33 προτάσεις αφορούν το σενάριο 3, το οποίο αναφέρεται σε περιπτώσεις ορθής περίληψης ή παράλειψης της εσωτερικής αύξησης από τους συμμετέχοντες, οι οποίες όμως κρίθηκαν ως λανθασμένες από το πρόγραμμα, παρόλο που ισχύει το αντίθετο.

Τέλος, το σενάριο που προέκυψε τις λιγότερες φορές είναι το σενάριο 1, που αφορά 21 προτάσεις. Πρόκειται για προτάσεις στις οποίες οι χρήστες περιέλαβαν εσωτερική αύξηση δίχως να είναι απαραίτητη ή την παρέλειψαν ενώ ήταν απαραίτητη. Παράλληλα, το πρόγραμμα παρουσίασε τις προτάσεις αυτές ως σωστές, ενώ θα έπρεπε να έχει πραγματοποιήσει αντίθετη διάκριση.

3.6.2 Αναλυτικά ποσοστά

Εξετάζοντας λοιπόν τις φορές που προέκυψε κάθε σενάριο, μπορούμε να εντοπίσουμε τα ποσοστά επιτυχίας του προγράμματος, καθώς και τα ποσοστά ορθής χρήσης της εσωτερικής αύξησης από τους χρήστες. Συγκεκριμένα, το άθροισμα των σεναρίων 1 και 3 μας δίνει το ποσοστό αποτυχίας του προγράμματος, ενώ το άθροισμα των σεναρίων 2 και 4 μας δίνει το ποσοστό επιτυχίας του προγράμματος. Παράλληλα, από το άθροισμα των σεναρίων 1 και 2 προκύπτει το ποσοστό λανθασμένης χρήσης της εσωτερικής αύξησης, ενώ από το άθροισμα των σεναρίων 3 και 4 προκύπτει το ποσοστό ορθής χρήσης της εσωτερικής αύξησης από τους συμμετέχοντες. Αναλυτικά, τα ποσοστά που προέκυψαν είναι τα ακόλουθα:

Επιτυχία προγράμματος	**Αποτυχία προγράμματος**
146 προτάσεις (73%)	54 προτάσεις (27%)

Πίνακας 3. (Ποσοστά επιτυχίας και αποτυχίας του προγράμματος)

Αρχικά, όσον αφορά τις διακρίσεις που πραγματοποίησε το πρόγραμμα σχετικά με τις προτάσεις που εισήγαγαν οι συμμετέχοντες, βλέπουμε ότι το ποσοστό επιτυχίας είναι υψηλό (73%), εφόσον στις 146 από τις 200 προτάσεις πραγματοποίησε ορθή διάκριση. Η ορθή διάκριση αφορά περιπτώσεις κατά τις οποίες ο συμμετέχων έχει χρησιμοποιήσει ορθά την εσωτερική αύξηση και το πρόγραμμα παρουσιάζει την πρότασή του ως σωστή, ή περιπτώσεις που ο συμμετέχων έχει χρησιμοποιήσει λανθασμένα την εσωτερική αύξηση και το πρόγραμμα παρουσιάζει την πρότασή του ως λανθασμένη, παρέχοντας παράλληλα την αντίστοιχη σωστή πρόταση.

Όσον αφορά το ποσοστό αποτυχίας, (το οποίο ανέρχεται στο 27%), βλέπουμε ότι στις 54 προτάσεις από τις 200 που εισήχθησαν συνολικά, το πρόγραμμα πραγματοποίησε λανθασμένες διακρίσεις. Η λανθασμένη διάκριση συναντάται σε περιπτώσεις που ο χρήστης έχει χρησιμοποιήσει ορθά την εσωτερική αύξηση και το πρόγραμμα, εξαιτίας του τρόπου λειτουργίας του, παρουσιάζει την πρόταση του χρήστη ως λανθασμένη και του παρέχει ως σωστή μια εναλλακτική πρόταση, η οποία στην πραγματικότητα είναι λανθασμένη. Ακόμη, η περίπτωση της λανθασμένης διάκρισης συναντάται και σε περιπτώσεις που ο χρήστης έχει χρησιμοποιήσει λανθασμένα την εσωτερική αύξηση και το πρόγραμμα παρουσιάζει την πρότασή του ως σωστή, ενώ θα έπρεπε να την κρίνει ως λανθασμένη και να παρέχει την αντίστοιχη σωστή πρόταση.

Ορθή χρήση εσωτερικής αύξησης	**Λανθασμένη χρήση εσωτερικής αύξησης**
131 προτάσεις (65,5%)	69 προτάσεις (34,5%)

Πίνακας 4. (Ποσοστά ορθής και λανθασμένης χρήσης της εσωτερικής αύξησης από τους συμμετέχοντες)

Στη συνέχεια, στον πίνακα 4 εντοπίζονται τα ποσοστά ορθής και λανθασμένης χρήσης της εσωτερικής αύξησης από τους συμμετέχοντες και τις συμμετέχουσες στο πείραμα. Αρχικά, στις 131 προτάσεις (ποσοστό 65,5%) από τις 200 που εισήχθησαν συνολικά στο πρόγραμμα έχει γίνει ορθή χρήση της εσωτερικής αύξησης. Συγκεκριμένα, η ορθή χρήση αφορά περιπτώσεις όπου οι συμμετέχοντες εισήγαγαν πρόταση η οποία περιελάμβανε σύνθετο ρήμα με πρόθεση στο τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου ενεργητικής φωνής και ορθά χρησιμοποίησαν το εν λόγω ρήμα με εσωτερική

αύξηση. Επιπλέον, η ορθή χρήση της εσωτερικής αύξησης αφορά και περιπτώσεις όπου εισήχθη πρόταση που περιελάμβανε σύνθετο ρήμα με πρόθεση στο δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου ενεργητικής φωνής και ορθά παρελείφθη η εσωτερική αύξηση από τον συγκεκριμένο ρηματικό τύπο.

Αντιθέτως, η λανθασμένη χρήση της εσωτερικής αύξησης αφορά περιπτώσεις παράλειψης της εσωτερικής αύξησης από σύνθετα ρήματα σε οριστική αορίστου και περιπτώσεις περίληψης της εσωτερικής αύξησης σε σύνθετα ρήματα σε προστακτική αορίστου. Οι συγκεκριμένες περιπτώσεις λανθασμένης χρήσης απαντώνται σε ποσοστό 34,5% (69 προτάσεις από τις 200).

	Άνθρωπος	**Υπολογιστής**
Αποτυχία	34,5%	27%
Επιτυχία	65,5%	73%

Πίνακας 5. (Συγκριτικός πίνακας επίδοσης ανθρώπου και υπολογιστή)

Συνδυάζοντας λοιπόν τα ποσοστά από τους πίνακες 3 και 4, προκύπτει ο πίνακας 5, ο οποίος συγκρίνει τις επιδόσεις ανθρώπου και υπολογιστή. Συγκεκριμένα, αντιπαραβάλλονται οι περιπτώσεις ορθής και λανθασμένης διάκρισης του προγράμματος με τις περιπτώσεις ορθής και λανθασμένης χρήσης της εσωτερικής αύξησης στα σύνθετα ρήματα με προθέσεις από τους ομιλητές. Αρκετά ενδιαφέρον είναι το γεγονός ότι, με βάση τον πίνακα 5, ο υπολογιστής σημειώνει καλύτερη επίδοση από τον άνθρωπο σε ποσοστό 7,5%.

4. Συμπεράσματα-Συζήτηση

4.1 Αξιολόγηση του προγράμματος

Στο σημείο αυτό, είναι αναγκαίο να αξιολογηθεί το πρόγραμμα, με βάση τα ποσοστά επιτυχίας και αποτυχίας που προέκυψαν κατά το στάδιο των δοκιμών του με φυσικούς ομιλητές της Ελληνικής. Επιπλέον, είναι σημαντικό να γίνει αναφορά στους περιορισμούς και τις αδυναμίες του προγράμματος, στα οποία οφείλεται το ποσοστό αποτυχίας του.

4.1.1 Αποτελέσματα δοκιμών

4.1.1.1 Ποσοστά επιτυχίας

Όπως είδαμε στον πίνακα 3, το συνολικό ποσοστό επιτυχίας του προγράμματος κατά το στάδιο των δοκιμών ανέρχεται στο 73%. Πρόκειται για ένα ποσοστό σχετικά υψηλό, που δικαιολογεί την ανάπτυξη του συγκεκριμένου υπολογιστικού μοντέλου. Συγκεκριμένα, θα μπορούσαμε να πούμε ότι, με βάση το ποσοστό που προέκυψε, για κάθε 4 προτάσεις που δέχεται το πρόγραμμα είναι σε θέση να πραγματοποιεί ορθή διάκριση για (περίπου) τις 3 από αυτές.

Συγκεκριμένα, στην προκειμένη περίπτωση, το πρόγραμμα πραγματοποίησε ορθή διάκριση για τις 146 από τις συνολικά 200 προτάσεις που εισήγαγαν οι συμμετέχοντες. Όπως αναφέρθηκε και παραπάνω, η ορθή διάκριση αναφέρεται σε περιπτώσεις που οι χρήστες έχουν χρησιμοποιήσει σωστά την εσωτερική αύξηση και το πρόγραμμα έχει κρίνει ως σωστές τις επιλογές τους ή σε περιπτώσεις που οι χρήστες έχουν χρησιμοποιήσει λανθασμένα την εσωτερική αύξηση και το πρόγραμμα έχει κρίνει ως λανθασμένες τις προτάσεις τους, παρέχοντάς τους παράλληλα την αντίστοιχη σωστή πρόταση.

Το συγκεκριμένο ποσοστό επιτυχίας λοιπόν, επιβεβαιώνει την υπόθεση που διατυπώθηκε στο τμήμα 3.3, συγκεκριμένα ότι η προστακτική εντοπίζεται συχνότερα στην πρώτη θέση της πρότασης, ενώ η οριστική εντοπίζεται συχνότερα σε θέση πλην της πρώτης, στην οποία συνήθως βρίσκεται το υποκείμενο της πρότασης. Καταλήγουμε στο συγκεκριμένο συμπέρασμα, διότι η θέση του ρήματος μέσα στην πρόταση είναι το κριτήριο με το οποίο αναπτύχθηκε το πρόγραμμα. Για την ακρίβεια, το πρόγραμμα κρίνει ως σωστές τις προτάσεις που διαθέτουν σύνθετο ρήμα με πρόθεση στο τρίτο πρόσωπο ενικού αριθμού,

οριστικής αορίστου ενεργητικής φωνής σε οποιαδήποτε θέση της πρότασης πλην της πρώτης και τις προτάσεις που διαθέτουν σύνθετο ρήμα με πρόθεση στο δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου ενεργητικής φωνής στην πρώτη θέση. Επομένως, εφόσον το ποσοστό επιτυχίας του προγράμματος ανέρχεται στο 73%, αυτό σημαίνει ότι το 73% των προτάσεων που χρησιμοποιήθηκαν κατά το στάδιο των δοκιμών, ανταποκρίνονται στο παραπάνω κριτήριο (τη θέση του ρήματος μέσα στην πρόταση).

Ακόμη, βλέπουμε ότι ισχύει η άποψη που διατυπώθηκε στο τμήμα 3.4, ότι δηλαδή από τις 4 πιθανές περιπτώσεις που παρουσιάζονται στα παραδείγματα 9-12 του τμήματος 3.3 (ρήμα σε προστακτική στην πρώτη θέση και σε θέση πλην της πρώτης, ρήμα σε οριστική στην πρώτη θέση και σε θέση διαφορετική από την πρώτη), οι περιπτώσεις 9 και 11, στις οποίες το πρόγραμμα πραγματοποιεί ορθή διάκριση, είναι συχνότερες από τις περιπτώσεις 10 και 12, στις οποίες πραγματοποιεί λανθασμένη διάκριση. Με τον τρόπο αυτό, οδηγούμαστε σε ποσοστό επιτυχίας υψηλότερο από το 50% που τυπικά προέκυπτε προτού πραγματοποιηθούν οι δοκιμές (βλ. 3.4).

Συνοψίζοντας λοιπόν, θα μπορούσαμε να αξιολογήσουμε θετικά την επιλογή του κριτηρίου της θέσης του ρήματος μέσα στην πρόταση (βάσει του οποίου αναπτύχθηκε το εν λόγω πρόγραμμα), εφόσον προέκυψε ποσοστό επιτυχίας 73%. Ωστόσο, αξίζει να σημειωθεί ότι είναι αρκετά πιθανό ένα διαφορετικό κριτήριο (από το οποίο θα προέκυπτε και διαφορετικός τρόπος λειτουργίας του προγράμματος) να οδηγούσε σε υψηλότερο ποσοστό επιτυχίας.

4.1.1.2 Ποσοστά αποτυχίας

Σαφώς χαμηλότερο είναι το ποσοστό αποτυχίας του προγράμματος, το οποίο, όπως είναι εμφανές από τον πίνακα 3, ανέρχεται στο 27%. Συγκεκριμένα, στις 54 από τις συνολικά 200 προτάσεις που δέχτηκε, το πρόγραμμα πραγματοποίησε λανθασμένες διακρίσεις, κρίνοντας ως λανθασμένες προτάσεις των συμμετεχόντων στις οποίες είχε γίνει ορθή χρήση της εσωτερικής αύξησης ή κρίνοντας ως σωστές προτάσεις στις οποίες είχε γίνει λανθασμένη χρήση της εσωτερικής αύξησης.[15]

Το ποσοστό αποτυχίας είναι αρκετά χαμηλότερο από το ποσοστό επιτυχίας του προγράμματος, ωστόσο δεν είναι αμελητέο σε καμία περίπτωση και, όπως θα δούμε και

[15] Στην πραγματικότητα, το ποσοστό αποτυχίας του προγράμματος οφείλεται και σε άλλους λόγους, πέρα από αυτούς που αναφέρονται εδώ (βλ. 4.1.1.2.1).

παρακάτω, μπορεί να μειωθεί ακόμη περισσότερο, καθιστώντας με τον τρόπο αυτό ακόμη πιο χρήσιμο το πρόγραμμα.

4.1.1.2.1 Περιπτώσεις αποτυχίας του προγράμματος

Όπως αναφέρθηκε παραπάνω, κατά το στάδιο των δοκιμών το πρόγραμμα σημείωσε ποσοστό αποτυχίας 27%, το οποίο όμως δεν οφείλεται εξ ολοκλήρου στον τρόπο λειτουργίας του. Συγκεκριμένα:

Τρόπος λειτουργίας	Εισαγωγή ρηματικού τύπου που απουσιάζει από το πρόγραμμα	Εισαγωγή ρηματικού τύπου με δύο τόνους	Εισαγωγή ρηματικού τύπου με ορθογραφικό λάθος
31 προτάσεις (57,4%)	14 προτάσεις (25,92%)	3 προτάσεις (5,55%)	6 προτάσεις (11,11%)

Πίνακας 6.

Ο πίνακας 6 παρουσιάζει τις διαφορετικές περιπτώσεις αποτυχίας του προγράμματος, καθώς και σε πόσες προτάσεις συναντάται η κάθε περίπτωση.

Αρχικά, βλέπουμε ότι η πλειοψηφία των περιπτώσεων αποτυχίας του προγράμματος (57,4%) οφείλεται στον τρόπο λειτουργίας του, δηλαδή στο κριτήριο της θέσης του ρήματος μέσα στην πρόταση, με το οποίο κριτήριο έχει αναπτυχθεί το πρόγραμμα. Συγκεκριμένα, όπως έχει αναφερθεί και παραπάνω, το πρόγραμμα δέχεται ως ορθές μόνο τις προτάσεις που διαθέτουν σύνθετο ρήμα με πρόθεση σε προστακτική στην πρώτη θέση της πρότασης και τις προτάσεις που διαθέτουν σύνθετο ρήμα με πρόθεση σε οριστική σε οποιαδήποτε θέση της πρότασης, εκτός από την πρώτη. Επομένως, όσον αφορά τις περιπτώσεις που οι συμμετέχοντες χρησιμοποίησαν ορθά την εσωτερική αύξηση, η προστακτική βρισκόταν σε θέση της πρότασης διαφορετική από την πρώτη και η οριστική βρισκόταν στην πρώτη θέση, με αποτέλεσμα το πρόγραμμα να αναγνωρίσει ως λανθασμένες τις προτάσεις και να παράσχει στους χρήστες λανθασμένο εξαγόμενο. Όσον αφορά τις περιπτώσεις που οι συμμετέχοντες χρησιμοποίησαν λανθασμένα την εσωτερική αύξηση, η προστακτική (που λανθασμένα διέθετε εσωτερική αύξηση και ταυτιζόταν με τον τύπο της οριστικής) βρισκόταν σε θέση της πρότασης διαφορετική από την πρώτη, γεγονός που οδηγούσε το πρόγραμμα να κρίνει την πρόταση ως σωστή. Παράλληλα, η οριστική (που λανθασμένα δεν διέθετε

εσωτερική αύξηση και ταυτιζόταν με τον τύπο της προστακτικής) βρισκόταν στην πρώτη θέση της πρότασης και έτσι, το πρόγραμμα έκρινε την πρόταση ως σωστή.

Στη συνέχεια, όπως επισημάνθηκε στο τμήμα 3.3.1.1, οι τύποι της οριστικής και της προστακτικής των σύνθετων ρημάτων με πρόθεση εισήχθησαν στο πρόγραμμα με τη μορφή λιστών. Αυτό σημαίνει ότι το πρόγραμμα βασίζεται στις συγκεκριμένες λίστες προκειμένου να πραγματοποιήσει ταύτιση και να εντοπίσει το σύνθετο ρήμα μέσα στην πρόταση. Επομένως, σε περιπτώσεις εισαγωγής πρότασης με σύνθετο ρήμα το οποίο απουσιάζει από τις λίστες, το πρόγραμμα δεν ήταν σε θέση να πραγματοποιήσει διάκριση, αλλά ούτε και να παράσχει εξαγόμενο στον χρήστη. Το 25,92% λοιπόν των προτάσεων στις οποίες απέτυχε το πρόγραμμα κατά το στάδιο των δοκιμών, περιείχε ρήματα τα οποία δεν είχαν εισαχθεί στο πρόγραμμα αρχικά. Εντούτοις, αυτό που καθιστά διαφορετική τη συγκεκριμένη περίπτωση αποτυχίας από τις άλλες, είναι ότι τα ρήματα που έλειπαν από το πρόγραμμα κατά το στάδιο των δοκιμών, εισήχθησαν στη συνέχεια, γεγονός που συνεπάγεται ότι, σε περίπτωση εισαγωγής πρότασης με ένα από τα ρήματα αυτά μελλοντικά, το πρόγραμμα θα είναι σε θέση να πραγματοποιήσει διάκριση και να παράσχει εξαγόμενο στον χρήστη.

Η επόμενη περίπτωση αποτυχίας του προγράμματος αφορά τους τύπους με δύο τόνους. Για την ακρίβεια, κατά το στάδιο των δοκιμών χρησιμοποιήθηκαν οι παρακάτω προτάσεις από ορισμένους συμμετέχοντες:

28) Κατέγραψέ μου όλους τους αριθμούς.
29) Εισήγαγέ μου τα ονόματα στην λίστα.
30) Ανάτρεψέ μου το επιχείρημα.

Ανεξάρτητα από την ορθή ή λανθασμένη χρήση της εσωτερικής αύξησης, στις προτάσεις αυτές οι χρήστες έχουν ορθά χρησιμοποιήσει ρηματικούς τύπους με δύο τόνους. Όπως αναφέρθηκε όμως παραπάνω, το πρόγραμμα βασίζεται αποκλειστικά στις list_a και list_b προκειμένου να εντοπίσει το σύνθετο ρήμα σε μια πρόταση. Οι ρηματικοί τύποι όμως έχουν εισαχθεί στις λίστες με έναν τόνο. Επομένως, όταν πρόκειται για τύπους με δύο τόνους, το πρόγραμμα δεν μπορεί να πραγματοποιήσει ταύτιση και, ως εκ τούτου, δεν προσφέρει εξαγόμενο στον χρήστη.

Τέλος, υπήρχαν και περιπτώσεις (και συγκεκριμένα 6 προτάσεις) που ο ρηματικός τύπος που χρησιμοποίησαν οι συμμετέχοντες περιείχε ορθογραφικό λάθος. Το γεγονός αυτό αποτελεί ευθύνη του χρήστη, όμως όπως και να έχει, οδηγεί σε αποτυχία του προγράμματος το οποίο, όπως και στις 2 προηγούμενες περιπτώσεις, δεν μπορεί να πραγματοποιήσει

ταύτιση του ρήματος με βάση τα δεδομένα των λιστών από τις οποίες αντλεί και έτσι, δεν παράσχει γλωσσικό εξαγόμενο στον χρήστη.

4.1.2 Υπολογιστικά μοντέλα και ποσοστά επιτυχίας

Στο σημείο αυτό, έχοντας παρουσιάσει τα αποτελέσματα των δοκιμών, αξίζει να σημειωθεί ότι, γενικά στον χώρο της υπολογιστικής γλωσσολογίας, τα υπολογιστικά μοντέλα και προγράμματα που αναπτύσσονται δεν λειτουργούν με ποσοστό επιτυχίας 100%, καθώς, σε ορισμένες περιπτώσεις, δεν επιτελούν σωστά τον σκοπό για τον οποίο έχουν αναπτυχθεί, κυρίως λόγω περιορισμών στον τρόπο λειτουργίας τους. Επομένως, οι γλωσσολόγοι της υπολογιστικής γλωσσολογίας αποδέχονται ένα μοντέλο το οποίο δεν λειτουργεί με απόλυτη επιτυχία, εφόσον βέβαια, στην πλειοψηφία των περιπτώσεων, επιτελεί σωστά τον σκοπό με βάση τον οποίο έχει αναπτυχθεί, τη στιγμή που δέχεται το γλωσσικό εισαγόμενο (βλ. Grishman, 1986: 6). Για τον λόγο αυτό, το πρόγραμμα της παρούσας εργασίας κρίνεται ως αρκετά ικανοποιητικό, παρουσιάζοντας ποσοστό επιτυχίας 73% και ποσοστό αποτυχίας 27%. Επιπλέον, κρίνεται ικανοποιητικό λαμβάνοντας υπόψη ότι το ανθρώπινο λάθος ανέρχεται σε ποσοστό 34,5%.

4.1.3 Περιορισμοί και αδυναμίες του προγράμματος

Ορισμένες από τις αδυναμίες του προγράμματος έγιναν εμφανείς στην ενότητα 4.1.1.2.1. Συγκεκριμένα, είδαμε ότι η πλειοψηφία των περιπτώσεων αποτυχίας οφείλεται στο κριτήριο της θέσης του ρήματος στην πρόταση, με το οποίο έχει πραγματοποιηθεί η ανάπτυξη του προγράμματος. Μάλιστα, η συγκεκριμένη αδυναμία του προγράμματος δεν έχει περιθώριο βελτίωσης, διότι αποτελεί τον βασικό άξονα γύρω από τον οποίο έχει αναπτυχθεί το πρόγραμμα. Για περαιτέρω βελτίωση στον τομέα αυτό, απαιτείται ανάπτυξη του προγράμματος εκ νέου, με επιλογή διαφορετικού κριτηρίου, το οποίο δεν αποκλείεται να δημιουργήσει νέα προβλήματα που δεν εντοπίζονται με την παρούσα μορφή του προγράμματος.

Ακόμη, είδαμε ότι το πρόγραμμα δεν μπορεί να πραγματοποιήσει ταύτιση εάν γίνει εισαγωγή πρότασης με ρηματικό τύπο που δεν περιέχεται στις λίστες του. Ωστόσο, η συγκεκριμένη αδυναμία δεν κρίνεται ως τόσο σημαντική, καθώς, όπως συνέβη και μετά το στάδιο των δοκιμών, κάθε φορά που εντοπίζεται ρηματικός τύπος που απουσιάζει από το

πρόγραμμα, γίνεται εισαγωγή του με αποτέλεσμα τον εμπλουτισμό του προγράμματος, καθώς και τον αποκλεισμό παρόμοιου ενδεχομένου μελλοντικά.

Επιπλέον αδυναμία του προγράμματος είναι η ταύτιση τύπων με δύο τόνους, που στην προκειμένη περίπτωση αφορά μικρό ποσοστό (5,55%) από το συνολικό ποσοστό αποτυχίας. Ο συγκεκριμένος περιορισμός θα μπορούσε να αντιμετωπιστεί με εισαγωγή όλων των ρηματικών τύπων στις list_a και list_b με δύο τόνους, πέρα από τους τύπους με έναν τόνο που υπάρχουν ήδη. Με τον τρόπο αυτό, το πρόγραμμα θα μπορούσε να εντοπίσει τον ρηματικό τύπο μέσα στην πρόταση, προκειμένου να πραγματοποιήσει, στη συνέχεια, διάκριση σχετικά με την ορθότητα της πρότασης.

Ο τελευταίος περιορισμός που παρουσιάστηκε στην ενότητα 4.1.1.2.1 αφορά την εισαγωγή τύπου με ορθογραφικό λάθος. Εάν ο χρήστης εισαγάγει ρηματικό τύπο με ορθογραφικό λάθος, το πρόγραμμα δεν μπορεί να πραγματοποιήσει ταύτιση με βάση τα δεδομένα που έχει. Επομένως, σε αυτή την περίπτωση, δεν παρέχεται εξαγόμενο στον χρήστη αναφορικά με την ορθότητα της πρότασης που έχει εισαγάγει. Πρόκειται για έναν περιορισμό που δεν μπορεί να αντιμετωπιστεί, με βάση τον τρόπο που έχει αναπτυχθεί το πρόγραμμα. Να σημειωθεί βέβαια ότι, στην περίπτωση αυτή, η αποτυχία του προγράμματος αποτελεί κυρίως ευθύνη του χρήστη, ο οποίος εισαγάγει τύπο με ορθογραφικό λάθος. Όπως και να έχει πάντως, το πρόγραμμα θα ήταν ακόμη πιο αποτελεσματικό εάν μπορούσε να παρέχει εξαγόμενο στον χρήστη, ανεξάρτητα από το εάν ο ρηματικός τύπος είναι σωστός ή περιέχει ορθογραφικά λάθη.

Όπως είδαμε και στην ενότητα 2, για το υπό εξέταση πρόγραμμα αξιοποιήθηκε το ολοκληρωμένο περιβάλλον ανάπτυξης PyCharm. Μάλιστα, μέσω του συγκεκριμένου περιβάλλοντος εκτελέστηκε το πρόγραμμα και κατά το στάδιο των δοκιμών με τους χρήστες. Επομένως, στην παρούσα φάση, ένας περιορισμός είναι ότι, για να εκτελεστεί το πρόγραμμα σε έναν υπολογιστή, πρέπει να είναι εγκατεστημένο το ολοκληρωμένο περιβάλλον ανάπτυξης της Python. Η προαναφερθείσα διαδικασία δεν είναι ιδιαίτερα δύσκολη, ωστόσο, ενδέχεται να δημιουργεί πρόβλημα στον μέσο χρήστη. Επομένως, η δημιουργία ενός εκτελέσιμου αρχείου (.exe) κρίνεται ως πιο φιλική προς τους χρήστες που ενδεχομένως θα επιθυμούν να αξιοποιήσουν το εν λόγω πρόγραμμα για εκπαιδευτικούς σκοπούς. Ένα εκτελέσιμο αρχείο θα μπορεί να εκτελεστεί εύκολα και γρήγορα στον υπολογιστή του χρήστη, χωρίς να είναι απαραίτητη η εγκατάσταση επιπλέον στοιχείων.

Ακόμη μια αδυναμία του προγράμματος αφορά τα ρήματα που στην οριστική αορίστου συναντώνται με αύξηση αλλά και χωρίς αύξηση. Παραδείγματος χάρη:

31) αναιρώ → ανήρεσε/αναίρεσε

32) εκτελώ → εξετέλεσε/εκτέλεσε

Όπως βλέπουμε στα παραπάνω παραδείγματα, τα ρήματα «αναιρώ» και «εκτελώ» στο τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου ενεργητικής φωνής διαθέτουν τύπο με αύξηση, αλλά και τύπο χωρίς αύξηση. Κατά την εισαγωγή τους λοιπόν στις λίστες του προγράμματος, η list_a (που αφορά το τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου ενεργητικής φωνής) θα έπρεπε να περιλαμβάνει και τους δύο τύπους, εφόσον μπορεί να χρησιμοποιηθεί είτε ο ένας, είτε ο άλλος από τον χρήστη κατά την εισαγωγή μιας πρότασης. Ωστόσο, όπως είναι εμφανές, ο τύπος χωρίς αύξηση είναι όμοιος με το δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου ενεργητικής φωνής των συγκεκριμένων ρημάτων, που εντοπίζεται στη list_b. Επομένως, η εισαγωγή τόσο του τύπου με αύξηση όσο και του τύπου χωρίς αύξηση στη list_a θα δημιουργούσε πρόβλημα στο πρόγραμμα, το οποίο ουσιαστικά θα διέθετε όμοιους τύπους και στις δύο λίστες (παραδείγματος χάρη, τύπους όπως «αναίρεσε» και «εκτέλεσε»). Για τον λόγο αυτό, όσον αφορά το τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου ενεργητικής φωνής των εν λόγω ρημάτων, εισήχθη μόνο ο τύπος με αύξηση στη list_a.

Εντούτοις, με τη συγκεκριμένη επιλογή δημιουργείται ένα σημαντικό πρόβλημα. Συγκεκριμένα, έχουμε το ακόλουθο παράδειγμα:

33) Η Μαρία εκτέλεσε τις διαταγές που είχε λάβει.

34) Λάθος. Η σωστή πρόταση είναι: Η Μαρία εξετέλεσε τις διαταγές που είχε λάβει.

Στην πρόταση του παραδείγματος 33, το ρήμα βρίσκεται σε τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου ενεργητικής φωνής, επομένως ο χρήστης μπορεί να επιλέξει να χρησιμοποιήσει είτε τον τύπο με αύξηση («εξετέλεσε»), είτε τον τύπο χωρίς αύξηση («εκτέλεσε»). Εάν χρησιμοποιηθεί ο τύπος με αύξηση, το πρόγραμμα θα κρίνει την πρόταση του χρήστη ως σωστή. Ωστόσο, εάν χρησιμοποιηθεί ο τύπος χωρίς αύξηση, το πρόγραμμα δίνει στον χρήστη το εξαγόμενο του παραδείγματος 34. Όπως αναφέρθηκε ήδη όμως, στην προκειμένη περίπτωση και οι δύο τύποι είναι αποδεκτοί. Επομένως, το πρόγραμμα δεν θα έπρεπε να κρίνει την πρόταση του χρήστη ως λανθασμένη και να του παρέχει τον τύπο με αύξηση, αλλά να δέχεται ως σωστούς και τους δύο τύπους.

Στην πραγματικότητα, αυτό συμβαίνει διότι ο τύπος «εκτέλεσε» έχει εισαχθεί μόνο στη list_b, η οποία περιέχει τα ρήματα σε δεύτερο πρόσωπο ενικού αριθμού, προστακτικής

αορίστου ενεργητικής φωνής, και έτσι γίνεται αποδεκτός από το πρόγραμμα μόνο στην περίπτωση της προστακτικής και όχι της οριστικής.

Συμπεραίνουμε λοιπόν ότι πρόκειται για μια σημαντική αδυναμία του προγράμματος, διότι ο αριθμός των σύνθετων ρημάτων που στην οριστική αορίστου χρησιμοποιούνται είτε με εσωτερική αύξηση, είτε χωρίς εσωτερική αύξηση είναι υψηλός.

4.2 Αξιολόγηση της χρήσης από τους ομιλητές

Στο σημείο αυτό, αξίζει να αναφερθούμε στα ποσοστά ορθής και λανθασμένης χρήσης της εσωτερικής αύξησης από τους συμμετέχοντες, τα οποία προέκυψαν κατά το στάδιο των δοκιμών.

Συγκεκριμένα, οι περιπτώσεις ορθής χρήσης της εσωτερικής αύξησης αφορούν την περίληψή της σε προτάσεις που γίνεται χρήση σύνθετου ρήματος με πρόθεση σε τρίτο πρόσωπο ενικού αριθμού, οριστικής αορίστου ενεργητικής φωνής, καθώς και την παράλειψή της σε προτάσεις που γίνεται χρήση σύνθετου ρήματος με πρόθεση σε δεύτερο πρόσωπο ενικού αριθμού, προστακτικής αορίστου ενεργητικής φωνής. Αντίθετα, οι περιπτώσεις λανθασμένης χρήσης αφορούν προτάσεις που ο συμμετέχων θέλει να χρησιμοποιήσει το ρήμα σε οριστική αορίστου, έχοντας όμως παραλείψει την εσωτερική αύξηση, καθώς και προτάσεις που θέλει να χρησιμοποιήσει το ρήμα σε προστακτική αορίστου, έχοντας όμως περιλάβει την εσωτερική αύξηση.

Με βάση λοιπόν τον πίνακα 4 του τμήματος 3.6.2 βλέπουμε ότι, στην πλειοψηφία των προτάσεων που εισήγαγαν οι ομιλητές, έγινε ορθή χρήση της εσωτερικής αύξησης και, συγκεκριμένα, στις 131 από τις συνολικά 200 προτάσεις (ποσοστό 65,5%). Σε σαφώς μικρότερο ποσοστό (34,5%) εντοπίζεται η λανθασμένη χρήση της εσωτερικής αύξησης από τους ομιλητές, εφόσον αφορά τις 69 από τις 200 προτάσεις που εισήχθησαν. Συμπεραίνουμε λοιπόν ότι δεν επιβεβαιώνεται η υπόθεση που διατυπώθηκε στην ενότητα 1.1.3, ότι δηλαδή οι φυσικοί ομιλητές της Ελληνικής συγχέουν σε υψηλό βαθμό το τρίτο πρόσωπο ενικού αριθμού της οριστικής αορίστου, ενεργητικής φωνής με το δεύτερο πρόσωπο ενικού αριθμού της προστακτικής αορίστου, ενεργητικής φωνής όταν πρόκειται για σύνθετα ρήματα με πρόθεση.

Συγκεκριμένα, η εν λόγω υπόθεση δεν επιβεβαιώνεται διότι το ποσοστό ορθής χρήσης είναι υψηλό και καλύπτει την πλειοψηφία των προτάσεων που χρησιμοποιήθηκαν κατά το στάδιο των δοκιμών. Εντούτοις, το ποσοστό αποτυχίας, αν και χαμηλότερο, δεν είναι αμελητέο και δείχνει ότι το φαινόμενο της εσωτερικής αύξησης αναφορικά με τους δύο υπό

εξέταση τύπους, αποτελεί προβληματική περιοχή για μια μερίδα των φυσικών ομιλητών της Ελληνικής. Επομένως, η ανάπτυξη του υπολογιστικού μοντέλου της παρούσας εργασίας εξακολουθεί να κρίνεται ως χρήσιμη, λαμβάνοντας υπόψη τα τελικά ποσοστά ορθής και λανθασμένης χρήσης που προέκυψαν.

4.3 Μελλοντική έρευνα

Όπως είδαμε στις ενότητες 1.2.1 και 1.2.3, οι βασικοί σκοποί που εξυπηρετεί το πρόγραμμα της παρούσας εργασίας είναι η συμβολή στη μοντελοποίηση της γλώσσας και η χρήση του για εκπαιδευτικούς λόγους. Για να μπορεί να επιτελέσει τους δύο αυτούς σκοπούς καλύτερα, βασική ανάγκη είναι η αύξηση του ποσοστού επιτυχίας του προγράμματος.

Αρχικά, σημαντική αύξηση μπορεί να επέλθει εάν πραγματοποιηθεί αλλαγή του κριτηρίου της θέσης του ρήματος μέσα στην πρόταση, βάσει του οποίου λειτουργεί το πρόγραμμα στην παρούσα φάση. Ένα νέο κριτήριο θα επιφέρει και νέο τρόπο λειτουργίας στο πρόγραμμα, με την αύξηση του ποσοστού επιτυχίας να είναι αρκετά πιθανή (βλ. 4.1.1.1). Εντούτοις, ένας νέος τρόπος λειτουργίας ενδέχεται να προκαλέσει νέα προβλήματα, τα οποία δεν εντοπίζονται με την παρούσα μορφή που διαθέτει το πρόγραμμα (βλ. 4.1.3).

Επιπλέον, αύξηση του ποσοστού επιτυχίας του προγράμματος (αν και μικρότερη) μπορεί να επέλθει με αντιμετώπιση των περιορισμών που διαθέτει το πρόγραμμα, τους οποίους είδαμε στο τμήμα 4.1.3. Αρχικά, πιο εφικτοί στόχοι είναι ο περαιτέρω εμπλουτισμός των λιστών του προγράμματος, ώστε να μπορεί να πραγματοποιεί διακρίσεις για ακόμη περισσότερα σύνθετα ρήματα με προθέσεις, καθώς και η επίλυση της αδυναμίας ταύτισης τύπων με δύο τόνους.

Μια αλλαγή που δεν θα επιφέρει αύξηση στα ποσοστά επιτυχίας του προγράμματος, όμως θα το καταστήσει αρκετά πιο φιλικό προς τον χρήστη, είναι η δημιουργία ενός εκτελέσιμου αρχείου (.exe) για να είναι πιο εύκολη και πιο άμεση η χρήση του προγράμματος από τους χρήστες, ανεξάρτητα από το επίπεδο της γνώσης υπολογιστών που διαθέτουν.

Στη συνέχεια, όσον αφορά τα ρήματα που στην οριστική αορίστου χρησιμοποιούνται είτε με εσωτερική αύξηση, είτε χωρίς εσωτερική αύξηση, κρίνεται αναγκαίος ο μετασχηματισμός του προγράμματος με τέτοιο τρόπο, ώστε να δέχεται ως ορθούς και τους δύο τύπους (παραδείγματος χάρη «εξετέλεσε» και «εκτέλεσε»), κυρίως διότι τα ρήματα της συγκεκριμένης κατηγορίας είναι αρκετά (για περισσότερες λεπτομέρειες βλ. 4.1.3).

Ακόμη, σημαντική βελτίωση του προγράμματος θα αποτελούσε ο μετασχηματισμός του τρόπου λειτουργίας του με τέτοιο τρόπο, ώστε να μη βασίζεται στις list_a και list_b για τον εντοπισμό των ρημάτων που μας ενδιαφέρουν στην παρούσα μελέτη. Ένας τέτοιος μετασχηματισμός μπορεί να έλυνε το πρόβλημα που παρουσιάζεται στις περιπτώσεις εισαγωγής ρηματικού τύπου με ορθογραφικό λάθος από τον χρήστη (βλ. 4.1.3).

Επίσης, θα μπορούσαν να πραγματοποιηθούν δοκιμές με συμμετέχοντες/συμμετέχουσες, μεταβάλλοντας ορισμένες παραμέτρους (λόγου χάρη το εύρος ηλικιών, το μορφωτικό επίπεδο κτλ) και, ίσως, αυξάνοντας τον αριθμό των συμμετεχόντων/συμμετεχουσών. Με τον τρόπο αυτό θα μπορούσε να ελεγχθεί εάν θα προκύψουν παρόμοια ποσοστά, όσον αφορά την ορθή και λανθασμένη χρήση της εσωτερικής αύξησης. Επιπλέον, με τον τρόπο αυτό θα μπορούσαν να ανιχνευθούν και οι λόγοι στους οποίους οφείλεται η λανθασμένη χρήση της εσωτερικής αύξησης.

Σημαντική, επίσης, κρίνεται και η δοκιμή του εν λόγω προγράμματος σε εκπαιδευτικό πλαίσιο. Η πρακτική εφαρμογή του προγράμματος για εκπαιδευτικούς σκοπούς θα δώσει τη δυνατότητα να ελεγχθεί εάν επιτελεί αποτελεσματικά τον έναν από τους δύο σκοπούς, βάσει των οποίων έχει αναπτυχθεί.

Τέλος, θα μπορούσε να αναπτυχθεί ένα υπολογιστικό μοντέλο το οποίο θα εστιάζει σε διαφορετικό γλωσσικό φαινόμενο, ώστε να αξιολογηθεί περαιτέρω η χρήση των υπολογιστικών μοντέλων για τον έλεγχο ερευνητικών υποθέσεων, αλλά και για εκπαιδευτικούς σκοπούς.

Ξενόγλωσση Βιβλιογραφία

Bangalore, S. & M. Johnston (2003). Balancing data-driven and rule-based approaches in the context of a multimodal conversational system. In *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No.03EX721)* (pp. 221–226). https://doi.org/10.1109/ASRU.2003.1318444

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & C. Stein (2009). *Introduction to Algorithms, Third Edition* (3rd ed.). United States: The MIT Press.

Drachman, G. & A. Malikouti-Drachman (2001). Concrete morphology, affix typology and concord chains. In: Ralli, A., Joseph, B. D. & M. Janse (Eds), *Proceedings of the First Conference of Modern Greek Dialects and Linguistic Theory*. Patras: University of Patras, 51–65.

Fernández, M. (2009). *Models of Computation: An Introduction to Computability Theory*. London: Springer Science & Business Media.

Grishman, R. (1986). *Computational Linguistics: An Introduction*. Cambridge: Cambridge University Press.

Hausser, R. (2014). *Foundations of Computational Linguistics* (3rd ed.). Berlin: Springer.

Hill, L. L., Crosier, S. J., Smith, T. R. & M. Goodchild (2001). A content standard for computational models. *D-Lib Magazine*, *7*(6). doi:10.1045/june2001-hill

Joseph, B. D. & R. D. Janda (1988). The how and why of diachronic morphologization and demorphologization. In: Hammond, M. & M. Noonan (Eds), *Theoretical Morphology. Approaches to Modern Linguistics*. New York: Academic Press, 193–210.

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., & Y. Wu (2016). Exploring the limits of language modeling. *ArXiv Preprint ArXiv:1602.02410*.

Jurafsky, D. & J. H. Martin (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (2nd ed.). United States: Prentice Hall.

Kaisse, E. M. (1982). On the preservation of stress in Modern Greek. *Linguistics*, *20*(1–2), 59–82.

Kotsanis, Y. & Y. Maistros (1985). Lexifanis: A lexical analyzer of Modern Greek. In *Proceedings of the Second Conference of the European Chapter of the Association for Computational Linguistics*. Geneva: University of Geneva, 154–158.

Kotsanis, Y. & Y. Maistros (1991). Describing morphological phenomena of Modern Greek using a unification grammar formalism. *Information Systems*, *16*(6), 641–647.

Kotsanis, Y., Maistros, Y. & A. Zavras (1987). Quicklem: A software system for Greek word-class determination. *Literary and Linguistic Computing*, *2*(4), 242–244.

Mackridge, P. (1985). *The Modern Greek Language: A Descriptive Analysis of Standard Modern Greek*. United States: Oxford University Press.

Mitkov, R. (2003). Preface. In: Mitkov, R. (Ed.), *The Oxford handbook of computational linguistics*. United States: Oxford University Press, ix-x.

Spyropoulos, V. & A. Revithiadou (2009). The morphology of past in Greek. *Studies in Greek Linguistics*, *29*, 108–122.

Sun, R. (2008). Introduction to computational cognitive modeling. In: Sun, R. (Ed.), *The Cambridge handbook of computational psychology*, Cambridge: Cambridge University Press, 3–19.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind, 59*(236), 433–460.

Tzevelekou, M. (2000). Integrating language tools into a computer assisted 2nd language learning: Remarks on the methods of teaching Greek in the primary schools of the Turkish-speaking minority of Thrace. In *Second International Conference on Language Resources and*

Evaluation, Workshop: Language Resources and Tools for Educational Applications (pp. 35–39).

Ελληνόγλωσση Βιβλιογραφία

Αγγελιδάκης, Ν. Α. (2015). *Εισαγωγή στον προγραμματισμό με την Python*. Ανακτήθηκε από http://aggelid.mysch.gr/pythonbook/

Γούτσος, Δ. (2003). Σώμα Ελληνικών Κειμένων: Σχεδιασμός και υλοποίηση. Στο *Πρακτικά του 6ου Διεθνούς Συνεδρίου Ελληνικής Γλωσσολογίας, Πανεπιστήμιο Κρήτης, 18-21 Σεπτεμβρίου 2003*. Κρήτη: Πανεπιστήμιο Κρήτης.

Κεραμιτζή, Χ. (2016). *Ανάπτυξη πολυεπίπεδου λεξικού XLE* (Αδημοσίευτη Μεταπτυχιακή Εργασία). Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών, Ελλάδα.

Κουτσούκος, Ν. (2010). Σχέση κλίσης και παραγωγής: Η εμφάνιση εσωτερικής αύξησης στους προθηματοποιημένους ρηματικούς τύπους της ΝΕ. Στο *Πρακτικά 5ης Συνάντησης Εργασίας Μεταπτυχιακών Φοιτητών του Τμήματος Φιλολογίας, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών, 29-31 Μαΐου 2009* (σσ. 177–185). Αθήνα: Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών.

Μαλικούτη-Drachman, Α. & G. Drachman (1992). Θεωρητικά προβλήματα του τονισμού των κλιτικών στα Νέα Ελληνικά. *Μελέτες για την Ελληνική Γλώσσα 12*. Θεσσαλονίκη, 83-104.

Μαλικούτη-Drachman, Α. & G. Drachman (1993). Σύγκριση ρηματικού τονισμού Κοινής και διαλέκτων. *Μελέτες για την Ελληνική Γλώσσα 13*. Θεσσαλονίκη, 143-161.

Μαρκόπουλος, Γ. (2006). *Ζητήματα Υπολογιστικής Γλωσσολογίας: Prolog και Μορφολογική Ανάλυση*. Περιοδικό Παρουσία, Σειρά Αυτοτελών Δημοσιευμάτων, αρ. 69. Αθήνα: Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών.

Μπαλτζής, Σ. Δ., Κολαλάς, Σ., Ευμοιρίδου, Ε., Αλεξάκης, Α. & Δ. Δούκα (2006). *Ένα καινοτόμο ηλεκτρονικό λεξικό της Νέας Ελληνικής Γλώσσας*, στο 3ο Συνέδριο της Ευρωπαϊκής Εταιρείας Νεοελληνικών Σπουδών. Βουκουρέστι, 2-4 Ιουνίου 2006. Αθήνα: Ελληνικά Γράμματα.

Μπαμπινιώτης, Γ. Δ. (1998). *Λεξικό της νέας ελληνικής γλώσσας*. Αθήνα: Κέντρο λεξικολογίας.

Παπακίτσος, Ε. Χ. (2000). *Συμβολή στη μορφολογική επεξεργασία της Νέας Ελληνικής: Λειτουργική αποσύνθεση - Καρτεσιανό ηλεκτρονικό λεξικό*. (Διδακτορική Διατριβή). Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών, Ελλάδα.

Παπακίτσος, Ε. Χ., Παπαθανασοπούλου, Γ. & Α. Αλέξη (2016). Η συμβολή της μορφολογικής γεννήτριας στην υπολογιστική γλωσσολογία. *Γλωσσολογία* 24, 93-101.

Ράλλη, Α. (2005). *Μορφολογία*. Αθήνα: Πατάκης.

Σώρρα, Μ. (2014). *Σχεδιασμός και ανάπτυξη πρότυπου συστήματος μορφολογικής ανάλυσης ονομάτων της Αρχαίας Ελληνικής γλώσσας* (Αδημοσίευτη Μεταπτυχιακή Εργασία). Πανεπιστήμιο Πατρών, Ελλάδα.

Τριανταφυλλίδης, Μ. (1941). *Νεοελληνική γραμματική*. Αθήνα: Οργανισμός Εκδόσεως Σχολικών Βιβλίων.

Φιλιππάκη-Warburton, Ε., Γεωργιαφέντης, Μ., Κοτζόγλου, Γ. & Μ. Λουκά (2011). *Γραμματική Ε' και Στ' Δημοτικού*. Αθήνα: Οργανισμός Εκδόσεως Διδακτικών Βιβλίων.

Χατζησαββίδης, Σ. & Α. Χατζησαββίδου (2011). *Γραμματική Νέας Ελληνικής γλώσσας Α', Β', Γ' γυμνασίου*. Αθήνα: Οργανισμός Εκδόσεως Διδακτικών Βιβλίων.

Ιστότοποι

Codecademy
https://www.codecademy.com/

Lexigram
https://www.lexigram.gr/

PyCharm
https://www.jetbrains.com/pycharm/

Python
https://www.python.org/

Σώμα Ελληνικών Κειμένων
http://www.sek.edu.gr/

Παραρτήματα

Παράρτημα Α

Στο σημείο αυτό παρατίθεται το έντυπο ενημέρωσης που παρεχόταν σε κάθε συμμετέχοντα/συμμετέχουσα, προτού ξεκινήσει η διαδικασία των δοκιμών:

ΠΜΣ "ΚΟΡΑΗΣ" Θεωρητική και Εφαρμοσμένη Γλωσσολογία
Τομέας Γλωσσολογίας
Τμήμα Φιλολογίας
Φιλοσοφική Σχολή
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Διπλωματική Εργασία
Επιβλέπων: Γεώργιος Μαρκόπουλος
Κωνσταντίνος Λώμης

Έντυπο Ενημέρωσης Συμμετεχόντων

Στα πλαίσια της διπλωματικής εργασίας του προγράμματος μεταπτυχιακών σπουδών Θεωρητικής και Εφαρμοσμένης Γλωσσολογίας, αναπτύχθηκε ένα πρόγραμμα σχετικό με την Ελληνική γλώσσα. Η βοήθειά σας κρίνεται απαραίτητη, προκειμένου να δοκιμαστεί το εν λόγω πρόγραμμα και να εξεταστεί ο βαθμός στον οποίο λειτουργεί επιτυχώς.

Συγκεκριμένα, το υπό εξέταση φαινόμενο είναι τα σύνθετα ρήματα με προθέσεις. Αυτό που ζητείται από εσάς είναι να εισαγάγετε συνολικά 10 προτάσεις στο πρόγραμμα, οι οποίες θα περιέχουν ένα από τα ρήματα της προαναφερθείσας κατηγορίας στο τρίτο πρόσωπο ενικού αριθμού, Οριστικής Αορίστου στην Ενεργητική Φωνή ή στο δεύτερο πρόσωπο ενικού αριθμού, Προστακτικής Αορίστου στην Ενεργητική Φωνή. Δεν υπάρχουν περιορισμοί ως προς το μέγεθος της πρότασης, τη σειρά των λέξεων, τα σημεία στίξης κτλ.

Επαναλαμβάνεται ότι σκοπός των δοκιμών δεν είναι να κριθούν οι συμμετέχοντες για τυχόν σωστές ή λανθασμένες προτάσεις που εισάγουν, αλλά να εξεταστεί ο βαθμός στον οποίο λειτουργεί σωστά το πρόγραμμα κατά τη χρήση του από φυσικούς ομιλητές της Ελληνικής γλώσσας. Επομένως, όλες οι προτάσεις, εφόσον σχετίζονται με το υπό εξέταση αντικείμενο, είναι ευπρόσδεκτες.

Η συμμετοχή σας είναι σαφώς εθελοντική και ανώνυμη. Τα προσωπικά σας στοιχεία δεν θα περιληφθούν στην παρούσα εργασία αλλά ούτε και σε μετέπειτα δημοσιεύσεις ή παρουσιάσεις της. Σας ευχαριστώ θερμά για τη συμμετοχή σας.

Παράρτημα Β

Ακολουθεί το έντυπο που αξιοποιήθηκε για την καταγραφή των προτάσεων που χρησιμοποίησε κάθε συμμετέχων/συμμετέχουσα, κατά το στάδιο των δοκιμών:

ΠΜΣ "ΚΟΡΑΗΣ" Θεωρητική και Εφαρμοσμένη Γλωσσολογία
Τομέας Γλωσσολογίας
Τμήμα Φιλολογίας
Φιλοσοφική Σχολή
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Διπλωματική Εργασία
Επιβλέπων: Γεώργιος Μαρκόπουλος
Κωνσταντίνος Λώμης

Ημερομηνία:

Ονοματεπώνυμο:

Ηλικία:

Προτάσεις συμμετεχόντων

Πρόταση 1:

Πρόταση 2:

Πρόταση 3:

Πρόταση 4:

Πρόταση 5:

Πρόταση 6:

Πρόταση 7:

Πρόταση 8:

Πρόταση 9:

Πρόταση 10: