ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
"ΠΡΟΗΓΜΕΝΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ποιος θέλει να γίνει εκατομμυριούχος;» a la Ελληνικά

ALTIN D. CIPI

Επιβλέπων: **Παναγιώτης Σταματόπουλος,** Επίκουρος Καθηγητής

ΑΘΗΝΑ

ΦΕΒΡΟΥΑΡΙΟΣ 2017

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ποιος θέλει να γίνει εκατομμυριούχος;» a la Ελληνικά

ALTIN D. CIPI
A.M.: M1309

ΕΠΙΒΛΕΠΩΝ: **Παναγιώτης Σταματόπουλος,** Επίκουρος Καθηγητής

Φεβρουάριος 2017

ΠΕΡΙΛΗΨΗ

Αυτή η εργασία περιγράφει αναλυτικά τις τεχνικές που χρησιμοποιούνται για την κατασκευή ενός εικονικού παίκτη για το δημοφιλές τηλεοπτικό παιχνίδι «Ποιος θέλει να γίνει εκατομμυριούχος;» και βασίζεται πάνω στο αντίστοιχο άρθρο [1] στο οποίο έχει γίνει υλοποίηση για την αγγλική και την ιταλική έκδοση του παιχνιδιού. Επίσης σε αυτήν την εργασία έγινε μια προσπάθεια εφαρμογής των διαφόρων τεχνικών που περιγράφονται μέσα στο άρθρο. Η υλοποίηση του εικονικού παίκτη για την ελληνική έκδοση του παιχνιδιού έγινε στην γλώσσα προγραμματισμού Java και θα παρουσιαστεί αναλυτικά. Ο εικονικός παίκτης πρέπει να απαντήσει σε μια σειρά από ερωτήσεις πολλαπλής επιλογής που τίθενται σε φυσική γλώσσα, επιλέγοντας τη σωστή απάντηση μεταξύ τεσσάρων διαφορετικών επιλογών. Εάν δεν είναι σίγουρος για κάποια απάντηση μπορεί να χρησιμοποιήσει τις σανίδες σωτηρίας (lifelines) ή να αποχωρήσει από το παιχνίδι. Η αρχιτεκτονική του εικονικού παίκτη αποτελείται από 1) μια μονάδα (module) Απάντησης Ερωτημάτων (Question Answering) (QA), η οποία αξιοποιεί την μηχανή αναζήτησης της Google για να ανακτήσει τα πιο σχετικά χωρία κειμένου που είναι χρήσιμα στο να προσδιοριστεί η σωστή απάντηση σε μία ερώτηση, 2) μια μονάδα Βαθμολόγησης Απαντήσεων (Answer Scoring) (AS), η οποία αποδίδει μια βαθμολογία σε κάθε υποψήφια απάντηση σύμφωνα με διαφορετικά κριτήρια με βάση τα αποσπάσματα των κειμένων που ανακτώνται από την μονάδα QA, και 3) μια μονάδα Λήψης Αποφάσεων (Decision Making) (DM), η οποία επιλέγει τη στρατηγική για το παιχνίδι σύμφωνα με συγκεκριμένους κανόνες, και σύμφωνα με τις βαθμολογίες που αποδίδονται στις υποψήφιες απαντήσεις. Τέλος στην εργασία αξιολογούνται τόσο η ακρίβεια του εικονικού παίκτη να απαντήσει σωστά στις ερωτήσεις του παιχνιδιού, όσο και η ικανότητά του να παίζει πραγματικά παιχνίδια για να κερδίσει χρήματα. Τα πειράματα έχουν διεξαχθεί με ερωτήσεις που προέρχονται από την ελληνική έκδοση του επιτραπέζιου παιχνιδιού. Σε γενικές γραμμές παρατηρείται ότι η μέση ακρίβεια του εικονικού παίκτη είναι σημαντικά καλύτερη από την απόδοση των ανθρώπινων παικτών. Όσον αφορά τη δυνατότητα να παίξει πραγματικά παιχνίδια, το οποίο περιλαμβάνει τον ορισμό μιας κατάλληλης στρατηγικής για τη χρήση των σανίδων σωτηρίας προκειμένου να αποφασίσει είτε να απαντήσει σε μια ερώτηση ακόμη και σε μια κατάσταση αβεβαιότητας ή να αποσυρθεί από το παιχνίδι παίρνοντας τα χρήματα που έχει κερδίσει μέχρι τώρα, ο εικονικός παίκτης κερδίζει κατά μέσο όρο περισσότερα χρήματα από το μέσο ποσό που κέρδισαν οι ανθρώπινοι παίκτες.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνητή Νοημοσύνη

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: γλωσσικά παιχνίδια, απάντηση ερωτήσεων, επεξεργασία φυσικής γλώσσας, βαθμολόγηση απαντήσεων, λήψη απόφασης

ABSTRACT

This work describes in detail the techniques used to build a virtual player for the popular TV game “Who Wants to Be a Millionaire?” and is based on the corresponding article [1] in which the virtual player has been implemented for the English and the Italian versions of the game. Also in this work an attempt was made to apply the various techniques described in the article. The implementation of the virtual player for the Greek version of the game was made using the programming language Java and will be presented in detail. The virtual player must answer a series of multiple-choice questions posed in natural language by selecting the correct answer among four different choices. If he is not sure about an answer he can use the lifelines or quit the game. The architecture of the virtual player consists of 1) a Question Answering (QA) module, which leverages the use of Google search engine to retrieve the most relevant passages of text useful to identify the correct answer to a question, 2) an Answer Scoring (AS) module, which assigns a score to each candidate answer according to different criteria based on the passages of text retrieved by the QA module, and 3) a Decision Making (DM) module, which chooses the strategy for playing the game according to specific rules as well as to the scores assigned to the candidate answers. Finally, in this work both the accuracy of the virtual player to answer correctly the questions of the game, and its ability to play real games in order to earn money are evaluated. The experiments have been conducted with questions derived from the Greek version of the board game. Generally, it is observed that the average accuracy of the virtual player is significantly better that the performance of the human players. Regarding the ability to play real games, which involves the definition of a proper strategy for the usage of lifelines in order to decide whether to answer a question even in a condition of uncertainty or to retire from the game by taking the earned money, the virtual player wins on average more money than the average amount earned by human players.

SUBJECT AREA: Artificial Intelligence

KEYWORDS: language games, question answering, natural language processing, answer scoring, decision making

Αφιερώνω αυτή την πτυχιακή εργασία στους γονείς μου Δώρα και Βασίλη που με στηρίζουν όλα αυτά τα χρόνια.

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας Διπλωματικής εργασίας, θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Παναγιώτη Σταματόπουλο, για τη συνεργασία και την πολύτιμη βοήθειά του. Επίσης θέλω να ευχαριστήσω τον εργοδότη και μέντορά μου Δρ. Νικόλαο Σκαρμέα ο οποίος πλήρωσε τα δίδακτρα του μεταπτυχιακού μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ 12

1. ΕΙΣΑΓΩΓΗ 13

2. ΚΑΝΟΝΕΣ ΤΟΥ ΠΑΙΧΝΙΔΙΟΥ 15

3. ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ 17

4. ΕΠΕΞΕΡΓΑΣΙΑ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ 18

4.1 Υπολογιστική Γλωσσολογία 18

4.2 Μορφές αναπαράστασης γνώσης 19

4.3 Συντακτική Ανάλυση 19

4.4 Σημασιολογική Ανάλυση 21

4.5 Πραγματολογική Ανάλυση 21

4.6 Πεδία εφαρμογής Επεξεργασίας Φυσικής Γλώσσας 22

4.7 Γλωσσικά παιχνίδια 22

5. ΑΠΑΝΤΗΣΗ ΕΡΩΤΗΜΑΤΩΝ 25

5.1 Ιστορική αναδρομή 25

5.2 Αρχιτεκτονική 26

5.3 Μέθοδοι υλοποίησης 27

5.4 Πρόοδος 29

5.5 Εφαρμογή 29

5.5.1 Ερωταποκρίσεις για την μηχανική ανάγνωση και την επικύρωση απάντησης 30

5.5.2 Ερωταποκρίσεις πάνω σε συνδεδεμένα δεδομένα (Linked Data) 32

6. ΛΗΨΗ ΑΠΟΦΑΣΗΣ 34

7. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ΕΙΚΟΝΙΚΟΥ ΠΑΙΚΤΗ 36

7.1 Απάντηση Ερώτησης 37
7.1.1 Η χρησιμοποίηση της DBpedia σαν πηγή γνώσης 41

7.2 Βαθμολόγηση Απάντησης 42

7.3 Λήψη Απόφασης 46

8. ΥΛΟΠΟΙΗΣΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ 50

8.1 Τεχνικές λεπτομέρειες 50

8.2 Ανάλυση κώδικα 50
8.2.1 Κατασκευή του συνόλου ερωτήσεων 52
8.2.2 Υλοποίηση του γραφικού περιβάλλοντος 54
8.2.3 Υλοποίηση του εικονικού παίκτη 64

9. ΠΕΙΡΑΜΑΤΙΚΗ ΑΞΙΟΛΟΓΗΣΗ 72

9.1 Πείραμα 1: αξιολόγηση της απόδοσης των μονάδων QA και AS. 73
9.1.1 Αναπάντητα ερωτήματα και ανάλυση σφαλμάτων 76
9.1.2 Δοκιμές Κατάλυσης (ablation) 77
9.1.3 Ανάλυση απόδοσης συστήματος και ανθρώπινης απόδοσης ανά επίπεδο 79

9.2 Πείραμα 2: αξιολόγηση του εικονικού παίκτη 82

10. ΕΝΔΕΙΚΤΙΚΗ ΕΚΤΕΛΕΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ 88

11. ΣΥΜΠΕΡΑΣΜΑΤΑ 94

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ 95

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ 99

ΠΑΡΑΡΤΗΜΑ Ι 100

ΠΑΡΑΡΤΗΜΑ ΙΙ 103

ΑΝΑΦΟΡΕΣ 112

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

Σχήμα 1: Παράδειγμα ερώτησης του "Ποιος θέλει να γίνει εκατομμυριούχος;" 15

Σχήμα 2: Η αρχιτεκτονική του εικονικού παίκτη. 37

Σχήμα 3: Η αρχιτεκτονική του QuestionCube 38

Σχήμα 4: Το κουτί πληροφοριών (infobox) του Leonardo Da Vinci.....41

Σχήμα 5: Ο αλγόριθμος λήψης απόφασης 47

Σχήμα 6: Το ποσοστό του κοινού που απαντάει σωστά στην ερώτηση όταν χρησιμοποιείται η βοήθεια του κοινού.....49

Σχήμα 7: Κατανομή των αρνητικών ερωτήσεων σε κάθε επίπεδο του παιχνιδιού. 72

Σχήμα 8: Μείωση της ακρίβειας από την κατάλυση ομάδας χαρακτηριστικών. Για κάθε ομάδα, ο αριθμός των χαρακτηριστικών που αφαιρούνται αναφέρεται επίσης.....79

Σχήμα 9: Ακρίβεια σε κάθε επίπεδο της απόδοσης του συστήματος και της ανθρώπινης απόδοσης για τα Ιταλικά. 81

Σχήμα 10: Ακρίβεια της απόδοσης του συστήματος για τα Αγγλικά. 81

Σχήμα 11: Πλοκή του μέσου εισοδήματος για διαφορετικές τιμές του ορίου μέτρησης της διαφοράς μεταξύ της βαθμολογίας της καλύτερης και της δεύτερης καλύτερης υποψήφιας απάντησης. 84

Σχήμα 12: Κατανομή των επιπέδων που επιτεύχθηκαν κατά τη διάρκεια του παιχνιδιού και τα χρήματα που κέρδισαν οι παίκτες (σε λογαριθμική κλίμακα). Τα άνω και κάτω άκρα των κιβωτίων αντιπροσωπεύουν το 3ο και 1ο τεταρτημόριο, αντίστοιχα. Η Διάμεσοι απεικονίζεται με συνεχείς γραμμές, οι μέσες τιμές με σταθερά σημεία. 84

Σχήμα 13: Κατανομή των παιχνιδιών που έφτασαν σε κάποιο συγκεκριμένο επίπεδο..85

Σχήμα 14: Κατανομή των παιχνιδιών που τελείωσαν με το εισόδημα σε ένα συγκεκριμένο διάστημα. 85

Σχήμα 15: Κατανομή των σανίδων που χρησιμοποιούνται κατά την διάρκεια του παιχνιδιού. 86

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Εικόνα 1: Η δομή του Java project όπως φαίνεται στο Eclipse IDE.....51

Εικόνα 2: Το αρχείο Questions.txt που περιέχει όλες τις ερωτήσεις του παιχνιδιού. 53

Εικόνα 3: Αρχική οθόνη του παιχνιδιού. 88

Εικόνα 4: Οθόνη με την πρώτη ερώτηση του παιχνιδιού.....89

Εικόνα 5: Επιβεβαίωση της απάντησης που διάλεξε ο παίκτης.....89

Εικόνα 6: Πάτημα του κουμπιού «Παραιτηθείτε» και επιβεβαίωση για παραίτηση από το παιχνίδι με την λήψη του κερδισμένου έως αυτό το επίπεδο ποσού. 90

Εικόνα 7: Λήψη του κερδισμένου ποσού μετά την παραίτηση από το παιχνίδι. 90

Εικόνα 8: Χρήση της βοήθειας «Κοινό» και εμφάνιση μηνύματος με τα αποτελέσματα του κοινού.....91

Εικόνα 9: Χρήση της βοήθειας «Τηλέφωνο» και εμφάνιση του μηνύματος της απάντησης που έδωσε στο τηλέφωνο ο φίλος του παίκτη.....91

Εικόνα 10: Χρήση της βοήθειας «50/50» και εμφάνιση του μηνύματος με τις δύο λανθασμένες απαντήσεις.....92

Εικόνα 11: Αποχώρηση από το παιχνίδι ύστερα από λάθος απάντηση σε ερώτηση και λήψη του ποσού στο μαξιλαράκι που έχει κλειδώσει ο παίκτης.....92

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

Πίνακας 1: Λίστα από κείμενα που βρέθηκαν από τη μονάδα QA για την ερώτηση "Ποιος σκηνοθέτησε το Blade Runner?". Κάθε κείμενο έχει την σελίδα της Wikipedia στο οποίο βρέθηκε και τον βαθμό της σχέσης του κειμένου με την ερώτηση. 38

Πίνακας 2: Απόδοση των 15 καλύτερων ρυθμίσεων (κατά μέσο όρο για το σύνολο των ερωτήσεων). Ακρωνύμια: P αριθμός των κειμένων, Lex χρήση λεξικοποίησης, KWD λέξεις-κλειδιά, LEM λήμματα, ST μίσχοι, SCORE χρήση της βαθμολογίας του κειμένου, SW χρήση της αφαίρεσης εξαιρουμένων λέξεων, QE χρήση της επέκτασης ερωτήματος. 74

Πίνακας 3: Η καλύτερη και η χειρότερη απόδοση για κάθε κριτήριο με την αντίστοιχη ρύθμισή του (κατά μέσο όρο για το σύνολο των ερωτήσεων). Ακρωνύμια: P αριθμός των κειμένων, Lex χρήση λεξικοποίησης, Score χρήση της βαθμολογίας του κειμένου, SW χρήση της αφαίρεσης εξαιρουμένων λέξεων, QE χρήση της επέκτασης ερωτήματος, ES ακριβές κομμάτι κειμένου, LCS (Longest Common Subsequence) Μέγιστη κοινή συμβολοσειρά, TL τίτλος Levenshtein, KWD λέξεις-κλειδιά, LEM λήμματα, ST μίσχοι..74

Πίνακας 4: Ανάλυση σφάλματος για Ιταλικά και Αγγλικά. 77

ΠΡΟΛΟΓΟΣ

Η εργασία αυτή έγινε στα πλαίσια των μεταπτυχιακών μου σπουδών στο τμήμα Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών. Ο προβλεπόμενος χρόνος φοίτησης είναι δυο χρόνια και ο μέγιστος τρία. Όμως λόγω πλήρους απασχόλησης σε εταιρεία πληροφορικής κατά την διάρκεια της φοίτησης, μου πήρε 2,5 χρόνια να περάσω όλα τα μαθήματα. Στην συνέχεια μου ανατέθηκε το θέμα της εργασίας από τον κ. Σταματόπουλο ο οποίος ήταν επιβλέπων καθηγητής μου και στην πτυχιακή μου εργασία. Δυστυχώς λόγω αυξημένων επαγγελματικών υποχρεώσεων και πολυήμερων επαγγελματικών ταξιδιών στο εξωτερικό δεν μπόρεσα να τελειώσω την εργασία μέσα στα 3 μέγιστα έτη φοίτησης και έτσι αναγκάστηκα να ζητήσω παράταση σπουδών. Με την επίβλεψη και την βοήθεια του κ. Σταματόπουλου κατάφερα να τελειώσω επιτυχώς την εργασία και να ολοκληρώσω τις μεταπτυχιακές μου σπουδές στα 3,5 χρόνια.

Η εργασία αυτή βασίζεται στην τεχνητή νοημοσύνη και πιο συγκεκριμένα στην κατασκευή ενός εικονικού παίκτη που μπορεί να παίξει το λεκτικό παιχνίδι «Ποιος θέλει να γίνει εκατομμυριούχος;». Ο τόπος διεξαγωγής της είναι η Αθήνα και πιο συγκεκριμένα η Νέα Φιλαδέλφεια και ο χρόνος διεξαγωγής της είναι από τον Μάϊο του 2016 μέχρι τον Φεβρουάριο του 2017.

1. ΕΙΣΑΓΩΓΗ

Η δουλειά που έχει γίνει για τα ευφυή παιχνίδια στον υπολογιστή έχει μια μακρά ιστορία και υπήρξε ένας από τους πιο επιτυχημένους και με τα περισσότερα ορατά αποτελέσματα κλάδους έρευνας της Τεχνητής Νοημοσύνης (Artificial Intelligence) (AI) [35]. Πράγματι, σήμερα τα τεχνητά συστήματα είναι σε θέση να ανταγωνιστούν και μερικές φορές να ξεπεράσουν τους ανθρώπινους παίκτες σε διάφορα πολύπλοκα παιχνίδια. Τα περισσότερα από αυτά τα παιχνίδια είναι κλειστού κόσμου (closed world), που σημαίνει ότι έχουν έναν πεπερασμένο αριθμό από πιθανές επιλογές, το οποίο επιτρέπει στους ερευνητές την επίλυσή τους με έναν επίσημο τρόπο, ακόμα κι αν είναι δύσκολο να παιχτούν λόγω της εκθετικής διάστασης του χώρου αναζήτησης. Ένα πιο προκλητικό είδος παιχνιδιών αντιπροσωπεύεται από παιχνίδια ανοικτού κόσμου (open world), όπως τα αθλητικά παιχνίδια ή τα σταυρόλεξα: είναι λιγότερο δομημένα και, επιπλέον, τόσο οι καταστάσεις του παιχνιδιού όσο και οι δράσεις του παίκτη δεν μπορούν εύκολα να απαριθμηθούν, καθιστώντας την αναζήτηση μέσω του χώρου των πιθανών λύσεων πρακτικά ανέφικτη. Ένα από τα πιο πρόσφατα αποτελέσματα στον τομέα αυτό είναι η επιτυχία του Watson, το σύστημα ερωταποκρίσεων ανοιχτού πεδίου που κατασκευάστηκε από την IBM Research, το οποίο τον Φεβρουάριο του 2011 κατάφερε να νικήσει τους δύο παίκτες με την υψηλότερη κατάταξη στο κουίζ σόου Jeopardy! [18][19].

Μας ενδιαφέρουν ιδιαίτερα τα παιχνίδια που σχετίζονται με την ανθρώπινη γλώσσα. Κατατάσσονται σε παιχνίδια με λέξεις (word games), στα οποία οι έννοιες των λέξεων δεν είναι σημαντικές, και στα γλωσσικά παιχνίδια (language games), στα οποία οι σημασίες των λέξεων διαδραματίζουν σημαντικό ρόλο [28]. Τα γλωσσικά παιχνίδια γενικά απαιτούν μια μεγάλη γλωσσική γνώση και κοινή λογική. Το "Ποιος θέλει να γίνει εκατομμυριούχος;» (“Who Wants to Be a Millionaire?”) (WWBM) είναι ένα τέλειο παράδειγμα ενός γλωσσικού παιχνιδιού στο οποίο ο παίκτης δίνει μια απάντηση σε μια ερώτηση που τέθηκε σε φυσική γλώσσα, επιλέγοντας τη σωστή απάντηση από τις τέσσερις πιθανές. Ακόμη και αν ο αριθμός των πιθανών απαντήσεων περιορίζεται σε τέσσερις, για να είναι σε θέση κάποιος να παίξει με επιτυχία αυτό το παιχνίδι εξαρτάται σε μεγάλο βαθμό από τη γνώση του, την κατανόηση των ερωτήσεων και την ικανότητά του να εξισορροπήσει την εμπιστοσύνη στην απάντηση έναντι του κινδύνου που υπάρχει να απαντήσει.

Σε αυτή την εργασία θα περιγράψουμε την αρχιτεκτονική ενός εικονικού παίκτη (Virtual Player) για το παιχνίδι WWBM, η οποία αξιοποιεί τις τεχνικές Απάντησης Ερωτημάτων και την μηχανή αναζήτησης της Google, προκειμένου να ενσωματώσει γνώσεις χρήσιμες για να παιχτεί το παιχνίδι. Η τρέχουσα εργασία βασίζεται στο άρθρο [1] το οποίο δεν χρησιμοποιεί γενικά την μηχανή αναζήτησης της Google αλλά περιορίζεται στην χρήση κειμένων που προέρχονται από τις ανοικτές πηγές πληροφοριών Wikipedia και DBpedia. Το άρθρο [1] επεκτείνει την προηγούμενη εργασία που έχει γίνει σε αυτόν τον τομέα σύμφωνα με τις ακόλουθες κατευθύνσεις: τη χρήση της DBpedia, η στρατηγική λήψης αποφάσεων που έχει ενσωματωθεί για τη διαχείριση των "σανίδων σωτηρίας" που χαρακτηρίζει το παιχνίδι, η δυνατότητα να αποσυρθεί από το παιχνίδι, η χρήση τεχνικών μηχανικής μάθησης για να βελτιώσει τη διαδικασία της βαθμολόγησης των υποψηφίων απαντήσεων σε μια ερώτηση.

Για να αναπτυχθεί ένας αποτελεσματικός εικονικός παίκτης για το παιχνίδι WWBM, αλλά και γενικά για οποιοδήποτε παιχνίδι με ερωτήσεις πολλαπλής επιλογής, πρέπει να αντιμετωπιστούν δύο θέματα, ένα που σχετίζεται με το γενικότερο θέμα του σχεδιασμού αποτελεσματικών συστημάτων διασφάλισης της ποιότητας, ενώ το άλλο αφορά συγκεκριμένες πτυχές του παιχνιδιού. Ως εκ τούτου, παρακάτω θα ερευνηθούν τα ακόλουθα ερευνητικά ερωτήματα:

- RQ1. Σε ποιο βαθμό μπορεί ένα σύστημα QA να σχεδιαστεί με ένα τρόπο ανεξάρτητο γλώσσας, διατηρώντας την αποτελεσματικότητά του;

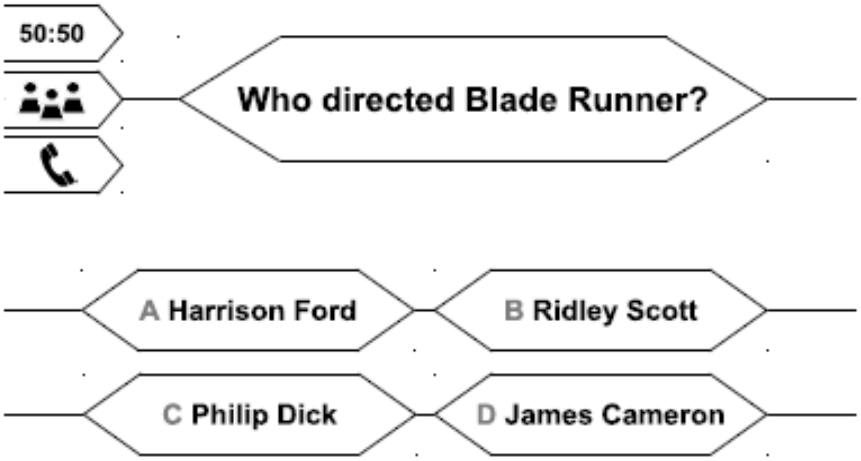
Αυτό το ζήτημα αντιμετωπίζεται προτείνοντας μια γενική αρχιτεκτονική ενός πλαισίου εργασίας (framework) QA και AS που εκμεταλλεύεται πόρους ή αλγορίθμους που έχουν σχεδιαστεί ειδικά για μια δεδομένη γλώσσα αποκλειστικά για τις βασικές λειτουργίες NLP, όπως ταυτοποίηση μέρους ενός λόγου (part-of-speech tagging). Στο άρθρο [1] προκειμένου να αξιολογηθεί η αποτελεσματικότητα του πλαισίου για τουλάχιστον δύο διαφορετικές γλώσσες, οι συγγραφείς πραγματοποίησαν πειράματα σε αγγλικά και ιταλικά. Η εργασία αυτή έρχεται να προσθέσει και τα αντίστοιχα αποτελέσματα για τα ελληνικά.

- RQ2. Μπορεί η Wikipedia και η DBpedia, αλλά και γενικά η μηχανή αναζήτησης της Google να χρησιμεύσουν ως αποτελεσματικές βάσεις γνώσεων για την απάντηση των ερωτήσεων του παιχνιδιού WWBM;

Αυτό το ζήτημα αντιμετωπίζεται στο άρθρο [1] με την ανάπτυξη ενός εικονικού παίκτη με βάση το προτεινόμενο πλαίσιο QA, ο οποίος αξιοποιεί τις πηγές ανοικτής γνώσης Wikipedia και DBpedia για να βρει τις σωστές απαντήσεις. Στην εργασία αυτή χρησιμοποιούνται γενικά τα αποτελέσματα που μας προσφέρει η μηχανή αναζήτησης της Google. Βέβαια θα παραθέσουμε και μερικά αποτελέσματα στα οποία περιορίσαμε την μηχανή αναζήτησης της Google να μας επιστρέψει αποτελέσματα που προέρχονται μόνο από το “el.wikipedia.org” που είναι η αντίστοιχη ελληνική Wikipedia. Εδώ τα αποτελέσματα που μας προσφέρονται δεν είναι πολλά στον αριθμό γι’ αυτό και αποφασίσαμε να μην χρησιμοποιήσουμε αυτό τον περιορισμό αλλά να αφήσουμε τον χρήστη της εφαρμογής να επιλέξει εάν θέλει να το χρησιμοποιήσει όπως θα δούμε στο κεφάλαιο της υλοποίησης. Με την DBpedia δεν ασχοληθήκαμε καθόλου καθώς εκεί η αναζήτηση γίνεται με διαφορετικό τρόπο που απαιτεί την γνώση Τεχνολογίας Γνώσεων. Εκτός από το πλαίσιο της μονάδας αυτής, ο εικονικός παίκτης υιοθετεί μια στρατηγική λήψης αποφάσεων για να παίξει το παιχνίδι με όλους τους κανόνες του, δηλαδή τη χρήση των "σανίδων σωτηρίας", απαντώντας σε μια κατάσταση αβεβαιότητας ή εγκατάλειψης του παιχνιδιού με τη λήψη των κερδισμένων χρημάτων. Τέλος πραγματοποιήθηκαν και μερικά πειράματα για τη σύγκριση της ακρίβειας του εικονικού παίχτη έναντι των ανθρώπινων παικτών. Πιο συγκεκριμένα ζητήθηκε από 20 άτομα να παίξουν το παιχνίδι στο γραφικό περιβάλλον που υλοποιήθηκε, και από την άλλη ο εικονικός παίκτης έπαιξε αρκετές φορές το παιχνίδι και συγκρίθηκαν τα αποτελέσματα μεταξύ τους. Επίσης παρατίθενται και τα αποτελέσματα από το άρθρο [1] και συγκρίνονται με αυτά της παρούσας εργασίας.

2. ΚΑΝΟΝΕΣ ΤΟΥ ΠΑΙΧΝΙΔΙΟΥ

Το WWBM είναι ένα γλωσσικό παιχνίδι, που μεταδόθηκε από πολλούς τηλεοπτικούς σταθμούς σε πολλές χώρες, ανάμεσά τους και η Ελλάδα, στο οποίο ένας παίκτης πρέπει να απαντήσει σωστά σε μια σειρά από 15 ερωτήσεις πολλαπλής επιλογής αυξανόμενης δυσκολίας. Τα ερωτήματα τίθενται σε φυσική γλώσσα και η σωστή απάντηση πρέπει να επιλεγεί μεταξύ των τεσσάρων πιθανών επιλογών.



Σχήμα 1: Παράδειγμα ερώτησης του "Ποιος θέλει να γίνει εκατομμυριούχος;"

Το **Σχήμα 1** δείχνει ένα παράδειγμα με την ερώτηση “Ποιος σκηνοθέτησε το Blade Runner?”, και οι τέσσερις πιθανές απαντήσεις είναι A) Harrison Ford, B) Ridley Scott, C) Philip Dick, D) James Cameron. Δεν υπάρχουν χρονικά όρια για να απαντήσει κάποιος στις ερωτήσεις. Επιπλέον, οι διαγωνιζόμενοι διαβάζουν την ερώτηση εκ των προτέρων, και στη συνέχεια ανά πάσα στιγμή μπορούν να αποφασίσουν αν θα επιχειρήσουν μια απάντηση ή θα εγκαταλείψουν το παιχνίδι κρατώντας τα κερδισμένα χρήματα. Κάθε ερώτηση έχει μια συγκεκριμένη χρηματική αξία σε ευρώ (Επίπεδο 1: 500; Επίπεδο 2: 1.000; Επίπεδο 3: 1.500; Επίπεδο 4: 2.000; Επίπεδο 5: 3.000; Επίπεδο 6: 5.000; Επίπεδο 7: 7.000; Επίπεδο 8: 10.000; Επίπεδο 9: 15.000; Επίπεδο 10: 20.000; Επίπεδο 11: 30.000; Επίπεδο 12: 70.000; Επίπεδο 13: 150.000; Επίπεδο 14: 300.000; Επίπεδο 15: 1.000.000). Εάν η απάντηση είναι σωστή, ο παίκτης κερδίζει το ορισμένο ποσό των χρημάτων και συνεχίζει να παίζει απαντώντας σε ερωτήσεις αυξανόμενης δυσκολίας έως ότου είτε φτάσει στην τελευταία ερώτηση ή αποσυρθεί από το παιχνίδι με τη λήψη του κερδισμένων χρημάτων. Υπάρχουν τρία σημεία εγγύησης (guarantee points) ή αλλιώς «μαξιλαράκια» όπου τα χρήματα είναι ασφαλή και δεν μπορούν να χαθούν ακόμα και αν ο παίκτης δώσει μια λανθασμένη απάντηση σε μία από τις επόμενες ερωτήσεις: 3.000, 20.000 και 1.000.000 ευρώ, που αντιστοιχούν στις ερωτήσεις στα επίπεδα 5, 10, 15, αντίστοιχα. Στο οποιοδήποτε επίπεδο, ο διαγωνιζόμενος μπορεί να χρησιμοποιήσει μία ή περισσότερες από τις τρεις "σανίδες σωτηρίας", που του παρέχουν κάποια μορφή βοήθειας:

- **50:50:** αυτή η σανίδα σωτηρίας αφαιρεί δύο λάθος πιθανές απαντήσεις, αφήνοντας τον παίκτη με μια δυαδική επιλογή μεταξύ της σωστής απάντησης και μίας λανθασμένης
- **Δημοσκόπηση του κοινού (poll the audience):** ο παίκτης ζητά το ακροατήριο που βρίσκεται στο στούντιο να πει ποια είναι η σωστή απάντηση. Τα ποσοστά του κοινού για τις 4 διαφορετικές απαντήσεις δίνονται στον παίκτη, ο οποίος έχει τον τελευταίο λόγο σχετικά με την επιλογή της απάντησης

- **Τηλέφωνο σε ένα φίλο (phone a friend):** ο παίκτης έχει 60 δευτερόλεπτα για να τηλεφωνήσει ένα φίλο, και να διαβάσει την ερώτηση με τις τέσσερις πιθανές επιλογές, για να πάρει μια πρόταση σχετικά με τη σωστή επιλογή.

3. ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

Η ανάκτηση πληροφοριών ή ανάκτηση πληροφορίας (Information Retrieval) είναι το επιστημονικό πεδίο της πληροφορικής που μελετά την αποδοτική αναζήτηση πληροφοριών και κάθε τύπου δεδομένων μέσα σε έγγραφα και μεταδεδομένα (metadata) σχετικά με έγγραφα, όπως επίσης και με την αναζήτηση σε βάσεις δεδομένων και στον Παγκόσμιο Ιστό (World Wide Web) (WWW). Μία εννοιολογική σύγχυση και μία πολύ εκτενής επικάλυψη υφίστανται στη χρήση των όρων ανάκτηση δεδομένων, ανάκτηση εγγράφων, ανάκτηση πληροφοριών και ανάκτηση κειμένου. Ωστόσο, καθένας από αυτούς τους όρους έχει τη δική του θεωρία και τις δικές του τεχνολογίες. Η ανάκτηση πληροφοριών στηρίζεται στη θεωρία των βάσεων δεδομένων, σε κατάλληλα υπολογιστικά συστήματα και σε μαθηματικές μεθόδους της τεχνητής νοημοσύνης, ενώ εφαρμόζεται στην επιστήμη πληροφόρησης και στον Παγκόσμιο Ιστό (μηχανές αναζήτησης). Χάρη στον διεπιστημονικό της χαρακτήρα, αντλεί στοιχεία από την επιστήμη υπολογιστών, από τα μαθηματικά, από τη βιβλιοθηκονομία, από τη γνωστική ψυχολογία, από τη γλωσσολογία και από τη στατιστική.

Μία διαδικασία ανάκτησης πληροφοριών ξεκινά όταν ο χρήστης εισάγει ένα ερώτημα (query) στο σύστημα. Τα ερωτήματα είναι αιτήσεις αναζήτησης σε κάποια βάση δεδομένων, όπως για παράδειγμα η αναζήτηση ενός αλφαριθμητικού σε κάποια μηχανή αναζήτησης του ιστού. Στην ανάκτηση πληροφοριών ένα ερώτημα δεν προσδιορίζει μονοσήμαντα ένα αντικείμενο, αλλά αντ’ αυτού, πολλά αντικείμενα μπορεί να ταιριάζουν στο ερώτημα, ίσως με διαφορετικούς βαθμούς σχετικότητας. Ένα αντικείμενο είναι μια οντότητα που αντιπροσωπεύεται από κάποιες πληροφορίες σε μια βάση δεδομένων. Για να απαντηθούν τα ερωτήματα των χρηστών, γίνεται αναζήτηση ανάμεσα στα στοιχεία της βάσης για πιθανές απαντήσεις. Τα δεδομένα μπορεί να είναι έγγραφα κειμένων, εικόνες, αρχεία ήχου ή βίντεο. Συχνά τα ίδια τα έγγραφα δεν αποθηκεύονται ή διατηρούνται απευθείας στο σύστημα ανάκτησης πληροφοριών, αλλά αντ’ αυτού εκπροσωπούνται στο σύστημα από υποκατάστατα έγγραφα ή μεταδεδομένα. Τα περισσότερα συστήματα ανάκτησης πληροφοριών υπολογίζουν μια αριθμητική βαθμολογία για το πόσο καλά κάθε αντικείμενο στη βάση δεδομένων ταιριάζει με το ερώτημα και κατατάσσει τα αντικείμενα σύμφωνα με αυτή την τιμή. Τα αντικείμενα που βρίσκονται στην κορυφή της κατάταξης εμφανίζονται στο χρήστη. Η διαδικασία μπορεί στη συνέχεια να επαναλαμβάνεται, αν ο χρήστης επιθυμεί να βελτιώσει το ερώτημα.

4. ΕΠΕΞΕΡΓΑΣΙΑ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ

Η επεξεργασία φυσικής γλώσσας (Natural Language Processing) (NLP) είναι μια από τις πιο δημοφιλείς εφαρμογές της Τεχνητής Νοημοσύνης. Η ιδέα του να επικοινωνεί κάποιος με τον υπολογιστή και να τον ελέγχει μιλώντας τη μητρική του γλώσσα ή κάποια ευρύτερα ομιλούμενη, όπως τα αγγλικά, είναι πολύ ελκυστική. Όμως, η φυσική γλώσσα έχει διττή φύση (ως προς τη σύνταξη και ως προς τη σημασιολογία), γεγονός που δεν εμποδίζει μεν την επεξεργασία της, αλλά δημιουργεί προβλήματα στην κατανόησή της, με αποτέλεσμα να καθίσταται το εγχείρημα της επεξεργασίας και παράλληλα της κατανόησής της ιδιαίτερα δύσκολο. Στο κεφάλαιο αυτό εξετάζουμε γενικά την υπολογιστική γλωσσολογία της οποίας η NLP είναι η πρακτική της εφαρμογή, τα στάδια της επεξεργασίας φυσικής γλώσσας (τη συντακτική, τη σημασιολογική και την πραγματολογική ανάλυση) και θίγουμε θέματα σχετικά με την πραγματολογική ανάλυση και τις τεχνικές που χρησιμοποιούνται για διευκρίνιση διφορούμενων προτάσεων και καθιστούν εφικτή την κατανόηση κειμένου. Τέλος παραθέτουμε και διάφορα πεδία εφαρμογής της NLP ένα από τα οποία είναι και τα γλωσσικά παιχνίδια που μας ενδιαφέρουν σε αυτή την εργασία.

4.1 Υπολογιστική Γλωσσολογία

Η μελέτη της γλώσσας αποτελεί αντικείμενο της κλασικής Γλωσσολογίας. Η Υπολογιστική Γλωσσολογία ως διεπιστημονική γνωστική περιοχή βρίσκεται ανάμεσα στην Γλωσσολογία και την Πληροφορική. Ανήκει στις Γνωστικές (Cognitive) Επιστήμες και επικαλύπτεται με την Τεχνητή Νοημοσύνη, κλάδο της Πληροφορικής (Computer Science), που ασχολείται με τα υπολογιστικά μοντέλα της ανθρώπινης γνώσης. Η Υπολογιστική Γλωσσολογία έχει θεωρητικό και πρακτικό κλάδο. Η θεωρητική Υπολογιστική Γλωσσολογία πραγματεύεται τις τυπικές θεωρίες παράστασης της γλωσσικής γνώσης και τους φορμαλισμούς που έχουν ανάγκη οι άνθρωποι για να παράγουν και να κατανοήσουν τη γλώσσα. Οι θεωρίες αυτές σήμερα έχουν φτάσει σε τέτοια πολυπλοκότητα που μόνο με υπολογιστές μπορεί κανείς να τις διαχειριστεί. Οι επιστήμονες του κλάδου αυτού αναπτύσσουν τυπικά μοντέλα που εξομοιώνουν απόψεις της ανθρώπινης γλωσσολογίας και τα υλοποιούν σε υπολογιστικά συστήματα. Αυτά αποτελούν τη βάση αποτίμησης και περεταίρω ανάπτυξης των αντίστοιχων θεωριών. Στην περιοχή αυτή της επιστήμης σημαντικό ρόλο παίζουν ευρήματα από την Γνωστική Ψυχολογία. Επίσης, η Ψυχογλωσσολογία, κλάδος της Ψυχολογίας, εξετάζει τις γνωστικές διαδικασίες που συνιστούν την χρήση της ανθρώπινης γλώσσας. Η εφαρμοσμένη Υπολογιστική Γλωσσολογία εστιάζεται στα πρακτικά αποτελέσματα της μοντελοποίησης της χρήσης της ανθρώπινης γλώσσας. Οι μέθοδοι, οι τεχνικές, τα εργαλεία και οι εφαρμογές αυτής της περιοχής συχνά συμπεριλαμβάνονται στον όρο (ανθρώπινη) γλωσσική μηχανική (language engineering) ή γλωσσική τεχνολογία (language technology). Παρότι τα σημερινά υπολογιστικά συστήματα απέχουν πολύ από τις ανθρώπινες ικανότητες έχουν πολλές πιθανές εφαρμογές. Στόχος είναι η ανάπτυξη συστημάτων με κάποια γνώση της ανθρώπινης γλώσσας. Τα συστήματα αυτά χρησιμοποιούνται στην καλυτέρευση της επικοινωνίας ανθρώπου - μηχανής. Οι σημερινοί υπολογιστές δεν «καταλαβαίνουν» τη γλώσσα μας αλλά, από την άλλη πλευρά, οι (τεχνητές) γλώσσες προγραμματισμού είναι δύσκολο να μαθευτούν και δεν αντιστοιχούν στη δομή της ανθρώπινης σκέψης.

4.2 Μορφές αναπαράστασης γνώσης

Η διαδικαστική (procedural) ήταν η πρώτη μορφή παράστασης γνώσης. Ακολούθησε η δηλωτική (declarative). Η διαδικαστική επιχειρεί να περιγράψει την κίνηση ή τη δράση και φιλοδοξεί να παραστήσει τη γνώση (γλωσσική ή μη) με διαδικασίες, με άλλα λόγια το "πώς" συμβαίνει η γνώση. Η δηλωτική επιχειρεί την παράσταση των συμβολικών δομών καθαυτών, μ' άλλα λόγια προσπαθεί να περιγράψει "τι" συμβαίνει. Ο δηλωτικός προγραμματισμός εξελίχθηκε και συμπεριέλαβε κανόνες συμπερασμού (inference rules). Ο προγραμματισμός αυτός λέγεται Λογικός Προγραμματισμός (Logic Programming) και η σημαντικότερη αντίστοιχη γλώσσα είναι η Prolog.

Τα Αυτόματα Πεπερασμένων Καταστάσεων (Finite State Automata) (FSA) επιστρατεύτηκαν για να αναγνωριστούν γλωσσικές οντότητες (π.χ. λεξικά στοιχεία, μέρη του λόγου) και τα Δίκτυα Μετάπτωσης Πεπερασμένων Καταστάσεων (Finite State Transition Networks) (FSTN) για να δημιουργήσουν γλωσσικές οντότητες. Σημαντικό σημείο αναφοράς στην περιοχή αυτή αποτελεί το πρόγραμμα ELIZA του J. Weizenbaum που βασίστηκε σε αυτά, το 1960. Τα Αναδρομικά Δίκτυα Μετάπτωσης (Recursive Transition Networks) (RTN) που ακολούθησαν ήταν FSTN με αναδρομή. Μπορούσαν να παραστήσουν γραμματικές του τύπου a^nb^n και τις παραστάσεις με παρενθέσεις. Τα Διευρυμένα Δίκτυα Μετάπτωσης (Augmented Transition Networks) (ATN) είχαν επιπλέον και μνήμη για να μην επαναλαμβάνουν διαδρομές. Παριστάνοντας δε μια μορφή δράσης ή ενέργειας (action) κατέστρεφαν τον δηλωτικό χαρακτήρα τους. Οι πρώτοι "τεχνολογητές χάρτη" ή "συντακτικοί αναλυτές χάρτη" (chart parsers) που κατέγραφαν τις πετυχημένες απόπειρες, για να μην επαναλαμβάνουν άσκοπα τους ελέγχους νομιμότητας, επιβιώνουν μέχρι σήμερα ως ιδέα με διαφορετικές μεθόδους υλοποίησης.

Με τη γέννηση της Prolog και του λογικού συμπερασμού (inference) που αυτή επιτρέπει δημιουργήθηκαν οι πρώτοι parsers Γραμματικών Οριστικών Προτάσεων (Definite Clause Grammars) (DCG). Αυτοί είναι parsers Γραμματικών Φραστικής Δομής (Phrase Structure Grammars) (PSG) σε γλώσσα Prolog, και αναμιγνύουν διαδικαστική με δηλωτική μορφή παράστασης. Η αμφισημία ως εγγενές στοιχείο της φυσικής γλώσσας που απαντάται σε όλα τα επίπεδα μελέτης της φυσικής γλώσσας, αποτελεί 'πονοκέφαλο' της μηχανικής επεξεργασίας φυσικής γλώσσας. Έρχεται σε αντίθεση με τον ντετερμινισμό των υπολογιστικών μηχανών και τις ντετερμινιστικές φιλοσοφικές δοξασίες. Η άρση της αμφισημίας αποτέλεσε και αποτελεί πρόκληση στην έρευνα της Γλωσσολογίας και της Υπολογιστικής Γλωσσολογίας.

4.3 Συντακτική Ανάλυση

Η συντακτική ανάλυση (syntactic analysis) εξετάζει τη δομή των προτάσεων προσπαθώντας να διαπιστώσει με διάφορους τρόπους αν αυτή είναι ορθή ή λανθασμένη.

Η πιο απλή προσέγγιση είναι η χρήση κάποιου τύπου ταιριάσματος προτύπου, οπότε απαιτείται να υπάρχουν αποθηκευμένα πρότυπα προτάσεων που περιέχουν μεταβλητές, ώστε να επιτρέπεται το ταίριασμα με συγκεκριμένες προτάσεις, Π.χ. το πρότυπο

<ο ** οδηγεί **>

όπου ** ταιριάζει με οτιδήποτε, επιτρέπει την αναγνώριση ως συντακτικά καλοσχηματισμένων προτάσεων όπως: ο Γιώργος οδηγεί ένα ωραίο αυτοκίνητο, ο Πέτρος οδηγεί το τραίνο, κλπ. Παρά ταύτα, η προσέγγιση δε θεωρείται πολύ αποτελεσματική, γιατί δεν επιτρέπει την αναγνώριση προβληματικών σημασιολογικά

προτάσεων, όπως, π.χ., ο τοίχος οδηγεί ένα σπίτι. Η μέθοδος του ταιριάσματος προτύπων θεωρείται κατάλληλη μόνο για πολύ στοιχειώδη περιβάλλοντα επικοινωνίας, όπου η χρήση της φυσικής γλώσσας πραγματοποιείται υπό περιορισμούς. Χρησιμοποιήθηκε αρχικά από το πρόγραμμα ELIZA (Weizenbaum, 1966) που, όπως αναφέραμε και στην προηγούμενη παράγραφο, δεν κατόρθωσε να εντοπίσει τις διφορούμενες προτάσεις και κατέληξε να αποδέχεται ως καλοσχηματισμένες προτάσεις ασυνάρτητες.

Πιο αποτελεσματική μέθοδος είναι η γραμματική ανάλυση μιας πρότασης, κατά την οποία η πρόταση μετατρέπεται σε μια ιεραρχική δομή που υποδεικνύει τα συστατικά της μέρη. Τα συστήματα γραμματικής ανάλυσης αποτελούνται από:

- μια γραμματική που προσφέρει μια δηλωτική αναπαράσταση των συντακτικών στοιχείων σχετικά με τη γλώσσα,
- έναν συντακτικό αναλυτή που συγκρίνει την πρόταση που εισάγεται με τη γραμματική.

Οι συντακτικοί αναλυτές μπορεί να είναι ανοδικοί, εάν αρχίζουν από τα σύμβολα εισόδου και προσπαθούν να ταιριάξουν κανόνες σε αυτά, ή καθοδικοί, οπότε αρχίζουν από τους γραμματικούς κανόνες και προσπαθούν να ταιριάξουν τα σύμβολα εισόδου σε αυτούς. Ένας συντακτικός αναλυτής (parser) που κάνει καθοδική γραμματική ανάλυση θα αρχίσει από το αρχικό σύμβολο της γραμματικής το οποίο θα αποτελέσει τη ρίζα ενός δένδρου που καλείται συντακτικό δένδρο (parse tree). Ουσιαστικά, ένα συντακτικό δένδρο αντιπροσωπεύει την ιεραρχική διάσπαση μιας πρότασης στα συντακτικά μέρη της. Η ρίζα του είναι το αρχικό σύμβολο και κάθε ενδιάμεσος κόμβος του αντιπροσωπεύει ένα μη-τερματικό σύμβολο της γραμματικής. Τα φύλλα του δένδρου αντιπροσωπεύουν τερματικά σύμβολα. Ο συντακτικός αναλυτής αναπτύσσει το συντακτικό δένδρο βήμα-βήμα με τη βοήθεια των παραγωγών της γραμματικής, έως ότου καταλήξει σε φύλλα, δηλαδή σε τερματικά σύμβολα. Σε κάθε βήμα ανάπτυξης, επιλέγεται ένας κόμβος που εκπροσωπεί ένα αριστερό μη-τερματικό σύμβολο ενός κανόνα και από κάτω διασπάται σε κόμβους που αντιπροσωπεύουν τα σύμβολα του δεξιού μέρους του κανόνα.

Άλλες μέθοδοι συντακτικής ανάλυσης περιλαμβάνουν διαγράμματα μετάβασης (transition networks) και επαυξημένα διαγράμματα μετάβασης (augmented transition networks). Ένα διάγραμμα μετάβασης είναι μια μέθοδος συντακτικής ανάλυσης που αναπαριστά τη γραμματική ως ένα σύνολο μηχανών πεπερασμένων καταστάσεων. Μια μηχανή πεπερασμένων καταστάσεων, που καλείται επίσης αυτόματο πεπερασμένων καταστάσεων, είναι ένα μοντέλο υπολογιστικής συμπεριφοράς, στο οποίο κάθε κόμβος αναπαριστά μια εσωτερική κατάσταση του συστήματος και οι ακμές παρουσιάζουν τον τρόπο μετάβασης από κατάσταση σε κατάσταση. Στην περίπτωση της επεξεργασίας φυσικής γλώσσας, οι ακμές αντιπροσωπεύουν είτε ένα τερματικό είτε ένα μη τερματικό σύμβολο. Οι γραμματικοί κανόνες αντιστοιχούν σε μια διαδρομή μέσα στο διάγραμμα. Για κάθε μη-τερματικό σύμβολο, υπάρχει και ένα διάγραμμα μετάβασης που το αναγνωρίζει.

Αν υπάρχει αμφισημία στις λέξεις, συνήθως δοκιμάζεται η συντακτική ανάλυση διαφόρων ερμηνειών των λέξεων μέσα από το λεξικό και επιλέγονται εκείνες που ταιριάζουν καλύτερα τόσο στη συντακτική όσο και στη σημασιολογική και πραγματολογική ανάλυση μιας πρότασης.

4.4 Σημασιολογική Ανάλυση

Κατά τη σημασιολογική ανάλυση (semantic analysis) πραγματοποιείται μετατροπή των προτάσεων σε εσωτερικές δομές αναπαράστασης γνώσης, χρησιμοποιώντας τη νοηματική σημασία των λέξεων. Για να επιτευχθεί κάτι τέτοιο, απαιτούνται οι λεγόμενες εξελιγμένες Γραμματικές Οριστικών Προτάσεων.

Σημαντικότερο πρόβλημα αποτελεί η ασάφεια σε επίπεδο λεκτικό που προκύπτει από την πολυσημία (ambiguity) των λέξεων. Ένας τρόπος αντιμετώπισης της ασάφειας αυτής είναι το να λαμβάνονται υπόψη χαρακτηριστικά της κλίσης των ουσιαστικών και ρημάτων: π.χ. γένος, αριθμός, πτώση των ουσιαστικών, πρόσωπα ρημάτων. Ακολουθεί ένα παράδειγμα εξελιγμένης γραμματικής που υποστηρίζει την παραπάνω προσέγγιση:

πρόταση → **υποκείμενο** (Γένος, Αριθμός, ονομαστική) **ρήμα** (Αριθμός) **αντικείμενο** (Γένος, Αριθμός, αιτιατική)

πρόταση → **υποκείμενο** (Γένος, Αριθμός, ονομαστική) **ρήμα** (Αριθμός) **κατηγορούμενο** (Γένος, Αριθμός, ονομαστική)

υποκείμενο → **άρθρο** (Γένος, Αριθμός, Πτώση) **ουσιαστικό** (Γένος, Αριθμός, Πτώση)

κατηγορούμενο → **επίθετο** (Γένος ,Αριθμός, Πτώση)

Σε γραμματικές του παραπάνω τύπου, τα χαρακτηριστικά διαδίδονται από τους ειδικότερους στους γενικότερους κανόνες, ώστε να υπάρχει συμφωνία των χαρακτηριστικών αυτών μεταξύ διαφορετικών τμημάτων της πρότασης. Για παράδειγμα:

- Ο αριθμός του υποκειμένου πρέπει να συμφωνεί με την κατάληξη του ρήματος.
- Το γένος, η πτώση και ο αριθμός του άρθρου του υποκειμένου πρέπει να συμφωνεί με τα αντίστοιχα χαρακτηριστικά της κλίσης του ουσιαστικού του υποκειμένου.

4.5 Πραγματολογική Ανάλυση

Σκοπός της πραγματολογικής ανάλυσης είναι η κατανόηση κειμένων και ο χειρισμός διαλόγων. Κατά την πραγματολογική ανάλυση επιχειρείται η ένταξη κάθε πρότασης μέσα στο γενικότερο νοηματικό πλαίσιο των συμφραζόμενων (context) της, λαμβανομένων υπόψη των συνθηκών μέσα στις οποίες αυτή λέχθηκε. Δυσκολίες προκύπτουν, όταν η πρόταση περιέχει αντωνυμίες, οι οποίες αναφέρονται σε ονόματα άλλων προτάσεων.

Για να λυθούν τα παραπάνω προβλήματα κατανόησης, πρέπει το σύστημα επεξεργασίας φυσικής γλώσσας να κατέχει τη γνώση για τον τρόπο με τον οποίο λειτουργεί ο κόσμος (το γενικότερο νοηματικό πλαίσιο) μέσα στον οποίο αναφέρεται η πρόταση, για τις πιθανότητες εμφάνισης κάποιων γεγονότων, για συνήθειες των ενεργούντων, πιθανά σενάρια (scripts) που μπορεί να διαδραματιστούν.

Με βάση αυτήν τη γνώση και το περιεχόμενο των προτάσεων ενός κειμένου ή διαλόγου, το σύστημα μπορεί να κάνει πολλούς εύλογους συμπερασμούς, διευρύνοντας τη γνώση του για την τρέχουσα κατάσταση του κειμένου ή μιας συζήτησης. Χωρίς γενική γνώση του αντικειμένου που πραγματεύεται ένα κείμενο ή διάλογος είναι αδύνατο να γίνει πραγματολογική ανάλυση.

4.6 Πεδία εφαρμογής Επεξεργασίας Φυσικής Γλώσσας

Η επεξεργασία φυσικής γλώσσας έχει ένα πλήθος εφαρμογών σε διάφορους χώρους. Κατά την κρίση μας, σημαντικότεροι για παρουσίαση είναι οι παρακάτω:

- Ανάκτηση πληροφοριών: χαρακτηρισμός εγγράφων με λέξεις- κλειδιά ή, στην ιδανική περίπτωση, νοηματική επεξεργασία των εγγράφων, για να δοθεί απάντηση σε ερώτηση που διατυπώνεται σε φυσική γλώσσα. Αυτός είναι και ο τομέας που μας ενδιαφέρει εμάς στην παρούσα εργασία.
- Ανάκτηση εγγράφων (document retrieval).
- Εξαγωγή πληροφορίας (information extraction).
- Εξόρυξη δεδομένων (Data Mining).
- Φωνητική προσπέλαση βάσεων δεδομένων: Η εφαρμογή αυτή αφορά την εισαγωγή ερωτημάτων σε βάσεις δεδομένων χρησιμοποιώντας φυσική γλώσσα.
- Κατηγοριοποίηση κειμένων (text categorization): ταξινόμηση κειμένων βάσει του περιεχομένου τους.
- Αυτόματη περίληψη (automated synopsis): εξαγωγή από ένα μεγάλο κείμενο ενός μικρότερου, με το βασικό/κεντρικό νόημα του πρώτου. Επιτρέπει την παραγωγή περιλήψεων από τα επιστρεφόμενα έγγραφα μιας μηχανής αναζήτησης.
- Αυτόματη μηχανική μετάφραση/διερμηνεία (machine translation).
- Αναγνώριση μερών του λόγου.
- Συντακτική ανάλυση.
- Εφαρμογές Επεξεργασίας Κειμένου (Word Processing Applications): έλεγχος ορθογραφίας, εισαγωγή κειμένου.
- Εκμάθηση γλώσσας υποβοηθούμενη από Η/Υ.
- Κατασκευή ηλεκτρονικών λεξικών: Επέκταση υπαρχόντων λεξικών π.χ. λεξικό ιατρικής ορολογίας (Wolff, 1984).

Παρά την πληθώρα τους, οι εφαρμογές στις οποίες απαιτείται κατανόηση φυσικής γλώσσας συχνά αποτυγχάνουν στο να κατανοήσουν και να διατυπώσουν σύνθετες ερωτήσεις. Για να υπάρχει ικανοποιητικό αποτέλεσμα, ο χρήστης θα πρέπει να γνωρίζει το υποστηριζόμενο λεξιλόγιο και τις γραμματικές/συντακτικές δυνατότητες του συστήματος. Ίσως και για το λόγο αυτό δε γνώρισαν έως τώρα μεγάλη εμπορική επιτυχία.

4.7 Γλωσσικά παιχνίδια

Τα γλωσσικά παιχνίδια συνήθως απαιτούν ένα μεγάλο ποσό γνώσης και βαθύ συλλογισμό για να ανταγωνιστεί κάποιος σε ανθρώπινο επίπεδο. Οι τεχνητοί παίκτες για γλωσσικά παιχνίδια υιοθετούν τεχνολογίες Επεξεργασίας Φυσικής Γλώσσας, προκειμένου να διαχειριστούν την πολυπλοκότητα και την αμφισημία της γλώσσας, ενώ αποθηκεύουν και χειρίζονται την πολύπλοκη αναπαράσταση της γνώσης που εμπλέκεται στο παιχνίδι.

Ένα δημοφιλές γλωσσικό παιχνίδι είναι η επίλυση σταυρόλεξου. Εκτός από τις γλωσσικές γνώσεις, η επίλυση σταυρόλεξου απαιτεί την ικανοποίηση των περιορισμών για τις πιθανές απαντήσεις. Η πρώτη εμπειρία που αναφέρεται στη βιβλιογραφία είναι το Proverb [29], το οποίο εκμεταλλεύεται μεγάλες βιβλιοθήκες από στοιχεία (clues) και

λύσεις από προηγούμενα σταυρόλεξα, ενώ το WebCrow [16], ο πρώτος λύτης για τα ιταλικά σταυρόλεξα, εκμεταλλεύεται το ίντερνετ ως κύρια πηγή των πληροφοριών, καθώς και μια σειρά από προηγούμενα λυμένα παιχνίδια. Το WebCrow βασίζεται στον διαδοχικό συνδυασμό της "απάντησης στοιχείων" (clue answering) και του "γεμίσματος πλέγματος" (grid filling), το οποίο διαφέρει ριζικά από την ανθρώπινη προσέγγιση. Για να βρει τις καλύτερες υποψήφιες λέξεις, το WebCrow χρησιμοποιεί την μηχανή αναζήτησης Google με ερωτήματα που είναι αναδιατυπώσεις των αρχικών ορισμών που λαμβάνονται εμπλουτίζοντας τις μορφολογικές μορφές των λέξεων-κλειδιών (π.χ., μεταβάλλοντας τον αριθμό και το φύλο για τα ουσιαστικά, ή το χρόνο για τα ρήματα), προσθέτει συνώνυμα από το WordNet, ώστε να κάνει την διαδικασία των επερωτήσεων πιο αποτελεσματική. Το κείμενο στις σελίδες που ανακτήθηκαν αναλύεται με τεχνικές NLP, και ένας ταξινομητής (classifier) επιλέγει το πιθανότερο μέρος του λόγου ανάλογα με τον ορισμό με σκοπό να μειωθεί ο αριθμός των υποψήφιων λέξεων. Τελικά, μια λίστα με τις καλύτερες λέξεις για τον ορισμό δίνεται και συγκρίνεται με τους περιορισμούς των γραμμάτων που αναφέρονται από το δίκτυο. Το WebCrow επιτυγχάνει 68,8% των σωστών λέξεων και 79,9% των σωστών γραμμάτων, δείχνοντας έτσι τις δυνατότητες του Web ως πόρου για πολύπλοκα γλωσσικά παιχνίδια.

Ένα άλλο ενδιαφέρον γλωσσικό παιχνίδι είναι η γκιλοτίνα (Guillotine), ένα παιχνίδι που μεταδόθηκε από την ιταλική εθνική τηλεόραση. Πρόκειται για έναν παίκτη, στον οποίο δίδεται ένα σύνολο από πέντε λέξεις (στοιχεία), το καθένα συνδέεται με κάποιο τρόπο με μια συγκεκριμένη λέξη που αντιπροσωπεύει τη μοναδική λύση του παιχνιδιού. Οι λέξεις είναι άσχετες μεταξύ τους, αλλά η καθεμία από αυτές είναι έντονα σχετική με τη λέξη που εκπροσωπεί την λύση. Για παράδειγμα, δεδομένου των πέντε λέξεων αμαρτία, ο Νεύτωνας, ο γιατρός, η πίτα, η Νέα Υόρκη, η λύση είναι το μήλο, γιατί: το μήλο είναι το σύμβολο του προπατορικού αμαρτήματος στην χριστιανική θεολογία, ο Νεύτωνας ανακάλυψε τη βαρύτητα μέσω ενός μήλου, "Ένα μήλο την ημέρα το γιατρό τον κάνει πέρα" είναι μια διάσημη παροιμία, η μηλόπιτα είναι μια πίτα φρούτων, και η Νέα Υόρκη επίσης ονομάζεται "Το μεγάλο Μήλο". Το OTTHO (On the Tip of my THOught) [4][49], είναι ένας τεχνητός παίκτης για το παιχνίδι της γκιλοτίνας. Η ιδέα πίσω από το OTTHO είναι να καθοριστεί μια διαδικασία έγχυσης γνώσης (knowledge infusion), η οποία αναλύει αδόμητες πληροφορίες που είναι αποθηκευμένες σε ανοικτές πηγές γνώσης στο Web για να δημιουργήσει μια μνήμη γλωσσικών ικανοτήτων (linguistic competencies) και παγκόσμιων γεγονότων που μπορούν να αξιοποιηθούν αποτελεσματικά από το σύστημα για μια βαθύτερη κατανόηση των πληροφοριών που πραγματεύεται. Η διαδικασία έγχυσης γνώσης υιοθετεί NLP τεχνικές για να χτίσει μια βάση γνώσεων και εξάγει πληροφορίες κυρίως από τη Wikipedia. Ένας μηχανισμός συλλογισμού (reasoning) που βασίζεται σε έναν αλγόριθμο ενεργοποίησης εξάπλωσης υιοθετείται για να ανακτηθούν τα πλέον κατάλληλα κομμάτια γνώσεων, χρήσιμων για την εξεύρεση πιθανών λύσεων.

Μια προσέγγιση για την υλοποίηση ενός εικονικού παίκτη για το παιχνίδι «Ποιος θέλει να γίνει εκατομμυριούχος;» έχει ήδη προταθεί [26]. Οι συγγραφείς εκμεταλλεύτηκαν το τεράστιο ποσό των γνώσεων στο Web και χρησιμοποίησαν τεχνικές NLP για να αναδιατυπώσουν τις ερωτήσεις και να δημιουργήσουν διαφορετικά ερωτήματα. Τα ερωτήματα αποστέλλονται στη συνέχεια στη μηχανή αναζήτησης της Google και ο αριθμός των αποτελεσμάτων χρησιμοποιείται ως μηχανισμός κατάταξης που εκμεταλλεύεται το περίσσευμα των πηγών πληροφοριών. Μια μονάδα λήψης αποφάσεων, που συνδυάζει τα αποτελέσματα σύμφωνα με το πνεύμα της μάθησης στο σύνολο χρησιμοποιώντας ένα προσαρμοζόμενο σύστημα στάθμισης, προσπαθεί να μεγιστοποιήσει το κερδισμένο χρηματικό ποσό σε σχέση με τον κίνδυνο της απάντησης. Κάπως έτσι λειτουργεί και εικονικός παίχτης στην παρούσα εργασία. Το σύστημα φτάνει ακρίβεια 75%, δείχνοντας πως αδόμητα δεδομένα μπορούν να είναι χρήσιμα για αυτή

τη δουλειά, αλλά αποτυγχάνει όταν οι ερωτήσεις απαιτούν κοινή συλλογιστική λογική και την πρόσβαση σε δομημένες πληροφορίες. Οι κύριες διαφορές σε σχέση με το άρθρο [1] είναι ότι εκεί οι συγγραφείς έχουν υιοθετήσει επιλεγμένες πηγές πληροφόρησης διαθέσιμες στον παγκόσμιο ιστό, όπως η Wikipedia και DBpedia, και όχι το σύνολο του Web, σε μια προσπάθεια να βελτιώσουν την αξιοπιστία των απαντήσεων. Επιπλέον, έχουν υιοθετήσει ένα QA πλαίσιο αντί της μηχανής αναζήτησης, προκειμένου να βελτιωθεί η διαδικασία επιλογής των πλέον αξιόπιστων κειμένων.

Τον Φεβρουάριο του 2011 ο υπερυπολογιστής της IBM Watson, υιοθέτησε τεχνολογία από το έργο DeepQA [19], και νίκησε δύο πρωταθλητές του Jeopardy! που είναι ένα τηλεοπτικό κουίζ παιχνίδι. Στο Jeopardy!, στον παίκτη δίνεται μια ερώτηση, αλλά εκφράζεται ως απάντηση, και πρέπει να βρει την απάντηση που πρέπει να εκφραστεί ως μια ερώτηση. Το Watson εφάρμοσε πολλές διαφορετικές τεχνικές NLP, Ανάκτησης Πληροφοριών και Μηχανικής Μάθησης (Machine Learning) (ML) που επικεντρώνονταν σε QA ανοικτού πεδίου, απαντώντας σε ερωτήσεις, χωρίς περιορισμούς χώρου (domain constraints). Το Watson ανέλυσε 200 εκατομμύρια στοιχεία περιεχομένου, τόσο δομημένων όσο και αδόμητων, συμπεριλαμβανομένου του πλήρους κειμένου της Wikipedia. Τα βήματα της διαδικασίας που ακολούθησε το Watson για να απαντήσει μπορούν να συνοψισθούν ως εξής:

- αποκτά γνώσεις από διαφορετικές πηγές δεδομένων, δηλαδή εγκυκλοπαίδειες, λεξικά, θησαυρούς, άρθρα σε περιοδικά, βάσεις δεδομένων, ταξινομήσεις και οντολογίες,
- η είσοδος του προβλήματος αντιμετωπίζεται ως ζήτημα, μετά αναλύεται χρησιμοποιώντας αλγόριθμους NLP και τέλος ταξινομείται,
- υποψήφιες απαντήσεις δημιουργούνται από τη γνώση που έχει αποκτηθεί,
- οι υποψήφιες απαντήσεις που δεν περνούν το όριο φιλτράρονται.

Χρησιμοποιούνται διάφοροι αλγόριθμοι βαθμολόγησης για την κατάταξη των υποψηφίων απαντήσεων, προκειμένου να δώσουν στοιχεία για την ποιότητά τους. Τέλος, τα κριτήρια βαθμολόγησης συνδυάζονται για την επιλογή της τελικής υποψήφιας απάντησης. Παρομοίως με την προσέγγιση που εφαρμόζεται στο Watson, οι συγγραφείς του άρθρου [1] υιοθέτησαν τεχνικές QA για την επίλυση του παιχνιδιού WWBM, και χρησιμοποίησαν την ίδια διαδικασία χρησιμοποιώντας διαφορετικά κριτήρια βαθμολόγησης, τα οποία τελικά συνδυάζονται για να επιστρέψουν την καλύτερη υποψήφια απάντηση.

5. ΑΠΑΝΤΗΣΗ ΕΡΩΤΗΜΑΤΩΝ

Η Απάντηση Ερωτημάτων είναι ένα αντικείμενο της επιστήμης των υπολογιστών που συγκαταλέγεται στους τομείς της ανάκτησης πληροφοριών και της Επεξεργασίας Φυσικής Γλώσσας. Ασχολείται με την κατασκευή συστημάτων που απαντάνε αυτόματα σε ερωτήσεις που τίθενται από τον άνθρωπο σε μια φυσική γλώσσα.

Μια εφαρμογή QA, συνήθως ένα πρόγραμμα υπολογιστή, μπορεί να κατασκευάσει τις απαντήσεις του με την αναζήτηση μιας δομημένης βάσης δεδομένων γνώσεων ή πληροφοριών, συνήθως μια βάση γνώσης. Συχνότερα, τα συστήματα QA μπορούν να τραβήξουν απαντήσεις από μια μη δομημένη συλλογή εγγράφων φυσικής γλώσσας. Μερικά παραδείγματα από συλλογές εγγράφων σε φυσική γλώσσα που χρησιμοποιούνται από τα συστήματα QA περιλαμβάνουν:

- μια τοπική συλλογή από κείμενα αναφοράς
- εσωτερικά έγγραφα οργάνωσης και ιστοσελίδες
- εκθέσεις ειδήσεων (newswire reports)
- ένα σύνολο σελίδων Wikipedia
- ένα υποσύνολο από σελίδες του Διαδικτύου με ένα ευρύ φάσμα τύπων ερώτησης όπως: γεγονότος (fact), καταλόγου (list), ορισμού (definition), Πώς (How), Γιατί (Why), υποθετικές (hypothetical), σημασιολογικά περιορισμένες, και διαγλωσσικές (cross-lingual) ερωτήσεις.

Η απάντηση ερωτημάτων χωρίζεται σε δύο κατηγορίες:

- Κλειστού χώρου (Closed-domain) QA. Ασχολείται με ζητήματα κάτω από ένα συγκεκριμένο τομέα (για παράδειγμα, την ιατρική ή τη συντήρηση αυτοκινήτων), και μπορεί να θεωρηθεί ως ένα εύκολο έργο, επειδή τα συστήματα NLP μπορούν να εκμεταλλευτούν συγκεκριμένους τομείς γνώσης που έχουν την μορφή οντολογιών. Εναλλακτικά, κλειστού χώρου μπορεί να αναφέρεται σε μια κατάσταση όπου μόνο ένας περιορισμένος τύπος ερωτήσεων είναι αποδεκτός, όπως ερωτήσεις που ζητάνε περιγραφικές και όχι διαδικαστικές πληροφορίες. Συστήματα QA στο πλαίσιο των εφαρμογών της μηχανικής ανάγνωσης έχουν επίσης κατασκευαστεί στον ιατρικό τομέα, για παράδειγμα που σχετίζονται με τη νόσο του Alzheimer.
- Ανοικτού χώρου (Open-domain) QA. Ασχολείται με ερωτήσεις σχετικές με σχεδόν τα πάντα, και μπορούν να στηρίζονται μόνο σε γενικές οντολογίες και την παγκόσμια γνώση. Από την άλλη πλευρά, αυτά τα συστήματα έχουν συνήθως πολύ περισσότερα δεδομένα διαθέσιμα από τα οποία μπορούν να εξαγάγουν την απάντηση.

5.1 Ιστορική αναδρομή

Δύο αρχικά συστήματα QA ήταν το BASEBALL [53] και το LUNAR [54]. Το BASEBALL απαντούσε ερωτήσεις σχετικά με το πρωτάθλημα μπέιζμπολ των ΗΠΑ για μια περίοδο ενός έτους. Το LUNAR, με τη σειρά του, απαντούσε σε ερωτήσεις σχετικά με τη γεωλογική ανάλυση των βράχων που επιστρέφονται από τις αποστολές των Apollo στο φεγγάρι. Και τα δύο συστήματα QA ήταν πολύ αποτελεσματικά στους επιλεγμένους τομείς τους. Στην πραγματικότητα, το LUNAR παρουσιάστηκε σε ένα σεληνιακό επιστημονικό συνέδριο το 1971 και ήταν σε θέση να απαντήσει στο 90% των ερωτήσεων, στον τομέα του, που τίθενται από τους ανθρώπους ανεκπαίδευτους στο

σύστημα. Περαιτέρω συστήματα QA περιορισμένης περιοχής (restricted-domain) αναπτύχθηκαν κατά τα επόμενα έτη. Το κοινό χαρακτηριστικό όλων αυτών των συστημάτων είναι ότι είχαν μια βάση δεδομένων πυρήνα (core database) ή ένα σύστημα γνώσης που ήταν γραμμένο στο χέρι από τους εμπειρογνώμονες της επιλεγμένης περιοχής. Οι γλωσσικές ικανότητες του BASEBALL και του LUNAR χρησιμοποιούν τεχνικές παρόμοιες με το ELIZA και το DOCTOR, τα πρώτα προγράμματα chatterbot.

Το SHRDLU ήταν ένα ιδιαίτερα επιτυχημένο πρόγραμμα QA που αναπτύχθηκε από τον Terry Winograd στα τέλη της δεκαετίας του '60 και στις αρχές της δεκαετίας του '70. Είναι προσομοίωση της λειτουργίας ενός ρομπότ σε έναν κόσμο παιχνιδιών (το «μπλοκ κόσμο»), και έδινε τη δυνατότητα να κάνει κάποιος ερωτήσεις στο ρομπότ για την κατάσταση του κόσμου. Και πάλι, η δύναμη αυτού του συστήματος ήταν η επιλογή ενός πολύ συγκεκριμένου τομέα και ενός πολύ απλού κόσμου με τους κανόνες της φυσικής που ήταν εύκολο να κωδικοποιήσει κάποιος σε ένα πρόγραμμα υπολογιστή.

Στη δεκαετία του 1970, αναπτύχθηκαν βάσεις γνώσεων που στοχεύουν σε στενότερους τομείς γνώσης. Τα συστήματα QA που αναπτύχθηκαν παρήγαγαν περισσότερες επαναλαμβανόμενες και έγκυρες απαντήσεις στις ερωτήσεις μέσα σε μια περιοχή της γνώσης. Αυτά τα έμπειρα συστήματα έμοιαζαν πολύ με τα σύγχρονα συστήματα QA, εκτός από την εσωτερική αρχιτεκτονική τους. Τα έμπειρα συστήματα βασίζονται σε μεγάλο βαθμό σε βάσεις γνώσεων που είναι έμπειρα οργανωμένες και κατασκευασμένες, ενώ πολλά σύγχρονα συστήματα QA βασίζονται σε στατιστική επεξεργασία ενός μεγάλου, αδόμητου, συνόλου κειμένων φυσικής γλώσσας.

Οι δεκαετίες του 1970 και του 1980 είδαν την ανάπτυξη ολοκληρωμένων θεωριών στην υπολογιστική γλωσσολογία (computational linguistics), η οποία οδήγησε στην ανάπτυξη των φιλόδοξων σχεδίων στην κατανόηση κειμένου και της απάντησης ερωτημάτων. Ένα παράδειγμα ενός τέτοιου συστήματος ήταν ο σύμβουλος Unix (Unix Consultant) (UC), που αναπτύχθηκε από τον Robert Wilensky στο U.C. Berkeley στα τέλη της δεκαετίας του 1980. Το σύστημα απαντούσε σε ερωτήσεις που σχετίζονταν με το λειτουργικό σύστημα Unix. Είχε μια ολοκληρωμένη χειροποίητη βάση γνώσεων του τομέα του, και είχε στόχο την διατύπωση της απάντησης ώστε να φιλοξενήσει διάφορους τύπους χρηστών. Ένα άλλο έργο ήταν το LILOG, ένα σύστημα κατανόησης κειμένου (text understanding) που λειτουργεί στην περιοχή των τουριστικών πληροφοριών σε μια γερμανική πόλη. Τα συστήματα που αναπτύχθηκαν στα έργα UC και LILOG δεν πήγαν ποτέ πέρα από το στάδιο των απλών επιδείξεων, αλλά βοήθησαν την ανάπτυξη των θεωριών για την υπολογιστική γλωσσολογία και την αιτιολογία (reasoning).

Πρόσφατα, έχουν αναπτυχθεί εξειδικευμένα συστήματα QA φυσικής γλώσσας, όπως το EAGLi για τους επιστήμονες της υγείας και της ζωής.

5.2 Αρχιτεκτονική

Τα περισσότερα σύγχρονα συστήματα QA χρησιμοποιούν τα έγγραφα κειμένων φυσικής γλώσσας ως βασική πηγή γνώσης τους. Οι τεχνικές επεξεργασίας φυσικής γλώσσας χρησιμοποιούνται τόσο για την επεξεργασία της ερώτησης όσο και για την αρχειοθέτηση (index) ή την επεξεργασία του σώματος (corpus) των κειμένων από τα οποία εξάγονται απαντήσεις. Ένας αυξανόμενος αριθμός συστημάτων QA χρησιμοποιούν το Παγκόσμιο Ιστό ως σώμα κειμένων και γνώσεων. Ωστόσο, πολλά από αυτά τα εργαλεία δεν παράγουν μια ανθρώπινη απάντηση, αλλά μάλλον χρησιμοποιούν «ρηχές» (shallow) μεθόδους (τεχνικές βασισμένες σε λέξεις, πρότυπα, κλπ.) για να παράγουν μια λίστα εγγράφων ή μια λίστα με αποσπάσματα κειμένων που περιέχουν την πιθανή απάντηση υπογραμμισμένη.

Σε μια εναλλακτική υλοποίηση QA, οι ανθρώπινοι χρήστες συγκεντρώνουν γνώση σε μια δομημένη βάση δεδομένων, που ονομάζεται μια βάση γνώσεων (knowledge base), παρόμοια με αυτές που χρησιμοποιούνται στα συστήματα των εμπειρογνωμόνων της δεκαετίας του 1970. Είναι επίσης δυνατόν να χρησιμοποιούν ένα συνδυασμό των δομημένων βάσεων δεδομένων και των εγγράφων κειμένου φυσικής γλώσσας σε ένα υβριδικό σύστημα QA. Ένα τέτοιο υβριδικό σύστημα μπορεί να χρησιμοποιεί αλγόριθμους εξόρυξης δεδομένων για να συμπληρώσει μια δομημένη βάση γνώσεων που επιμελείται επίσης και από ανθρώπους. Ένα παράδειγμα τέτοιου υβριδικού συστήματος QA είναι το Wolfram Alpha σύστημα QA που χρησιμοποιεί την επεξεργασία φυσικής γλώσσας για να μετατρέψει τα ανθρώπινα ερωτήματα σε μια μορφή που είναι επεξεργάσιμη από μία επιμελημένη βάση γνώσεων.

Από το 2001, τα συστήματα QA περιλαμβάνουν συνήθως μια μονάδα ταξινομητή ερώτησης που καθορίζει τον τύπο της ερώτησης και τον τύπο της απάντησης [55]. Μετά το ερώτημα αναλύεται, το σύστημα χρησιμοποιεί συνήθως διάφορες μονάδες που εφαρμόζουν ολοένα και πιο σύνθετες τεχνικές NLP για να μειώσουν σταδιακά την ποσότητα του κειμένου. Έτσι, μια μονάδα ανάκτησης εγγράφων χρησιμοποιεί μηχανές αναζήτησης για να προσδιορίσει τα έγγραφα ή τις παραγράφους στο σύνολο του εγγράφου που είναι πιθανό να περιέχουν την απάντηση, και ένα φίλτρο (filter) επιλέγει εκ των προτέρων μικρά αποσπάσματα του κειμένου που περιέχουν συμβολοσειρές (strings) του ίδιου τύπου όπως η αναμενόμενη απάντηση. Για παράδειγμα, εάν το ερώτημα είναι "Ποιος ανακάλυψε την πενικιλίνη;", το φίλτρο επιστρέφει κείμενο που περιέχει τα ονόματα των ανθρώπων. Τέλος, μια μονάδα εξαγωγής απάντησης (answer extraction) αναζητά περαιτέρω ενδείξεις στο κείμενο για να καθορίσει εάν η υποψήφια απάντηση μπορεί πράγματι να απαντήσει στην ερώτηση.

Μια πολύ-πρακτορική (multiagent) αρχιτεκτονική QA έχει προταθεί, όπου κάθε περιοχή αντιπροσωπεύεται από έναν πράκτορα (agent) ο οποίος προσπαθεί να απαντήσει σε ερωτήσεις λαμβάνοντας υπόψη τις ιδιαίτερες γνώσεις του. Ένας μετα-πράκτορας (meta-agent) ελέγχει τη συνεργασία μεταξύ των πρακτόρων και επιλέγει την πιο σχετική απάντηση. [56]

5.3 Μέθοδοι υλοποίησης

Η QA εξαρτάται πολύ από ένα καλό σώμα αναζήτησης - γιατί χωρίς έγγραφα που περιέχουν την απάντηση, ένα οποιοδήποτε σύστημα QA δεν μπορεί να κάνει και πολλά. Είναι επομένως λογικό ότι οι συλλογές με μεγαλύτερα μεγέθη γενικά προσφέρονται για καλύτερη απόδοση QA, εκτός και αν ο τομέας του ερωτήματος είναι ορθογώνιος ως προς τη συλλογή. Η έννοια του πλεονασμού δεδομένων (data redundancy) σε μαζικές συλλογές, όπως το διαδίκτυο, σημαίνει ότι ψήγματα (nuggets) πληροφοριών είναι πιθανό να διατυπωθούν με πολλούς διαφορετικούς τρόπους σε διαφορετικά πλαίσια και έγγραφα [57], πράγμα που οδηγεί σε δύο οφέλη:

- Με το να εμφανίζονται οι σωστές πληροφορίες σε πολλές μορφές, το βάρος για το σύστημα QA να εκτελέσει περίπλοκες τεχνικές NLP για να κατανοήσει το κείμενο ελαττώνεται.
- Οι σωστές απαντήσεις μπορούν να φιλτραριστούν από τυχόν ψεύτικα θετικά (false positives) στηριζόμενες στο γεγονός ότι η σωστή απάντηση θα εμφανιστεί περισσότερες φορές στα έγγραφα από τις λανθασμένες.

Η QA σε μεγάλο βαθμό στηρίζεται σε συλλογισμό. Υπάρχει ένας αριθμός συστημάτων QA που είναι σχεδιασμένα σε Prolog [58], μια γλώσσα λογικού προγραμματισμού που σχετίζεται με τεχνητή νοημοσύνη.

Στην ανάκτηση πληροφοριών, ένα σύστημα QA ανοιχτού πεδίου στοχεύει στην επιστροφή μια απάντησης, απαντώντας σε ερώτηση του χρήστη. Η επιστρεφόμενη απάντηση είναι με τη μορφή σύντομων κειμένων, αντί ενός καταλόγου σχετικών εγγράφων. Το σύστημα χρησιμοποιεί ένα συνδυασμό τεχνικών από την υπολογιστική γλωσσολογία, ανάκτησης πληροφοριών και αναπαράστασης της γνώσης για την εύρεση απαντήσεων.

Το σύστημα παίρνει μια ερώτηση σε φυσική γλώσσα ως είσοδο και όχι ένα σύνολο λέξεων-κλειδιών, για παράδειγμα, «Πότε είναι η εθνική εορτή της Κίνας;" Η πρόταση στη συνέχεια μετασχηματίζεται σε ένα ερώτημα μέσω της λογικής της μορφής (logical form). Έχοντας την είσοδο στη μορφή ερωτήματος σε φυσική γλώσσα καθιστά το σύστημα πιο φιλικό προς το χρήστη, αλλά πιο δύσκολο να υλοποιηθεί, καθώς υπάρχουν διάφορα είδη ερωτήσεων και το σύστημα θα πρέπει να προσδιορίσει το σωστό, ώστε να δώσει μια λογική απάντηση. Η ανάθεση ενός τύπου ερώτησης στο ερώτημα είναι ένα κρίσιμο έργο, η όλη διαδικασία εξόρυξης απάντησης βασίζεται στην εύρεση του σωστού τύπου ερώτησης και ως εκ τούτου του σωστού τύπου απάντησης.

Η εξόρυξη λέξεων-κλειδιών (keyword extraction) είναι το πρώτο βήμα για την αναγνώριση του τύπου ερώτησης της εισόδου. Σε ορισμένες περιπτώσεις, υπάρχουν σαφείς λέξεις που δείχνουν το τύπο της ερώτησης απευθείας, δηλαδή «Ποιος», «Πού» ή «Πόσα», αυτές οι λέξεις λένε στο σύστημα ότι οι απαντήσεις θα πρέπει να είναι του τύπου "Πρόσωπο", "Τοποθεσία", "Αριθμός", αντίστοιχα. Στο παραπάνω παράδειγμα, η λέξη «Πότε» δείχνει ότι η απάντηση πρέπει να είναι του τύπου «Ημερομηνία». Η ταυτοποίηση μέρους του λόγου και οι τεχνικές συντακτικής ανάλυσης μπορούν επίσης να χρησιμοποιηθούν για να καθορίσουν τον τύπο της απάντησης. Σε αυτήν την περίπτωση, το θέμα είναι η "Εθνική εορτή της Κίνας", το κατηγόρημα είναι "είναι" και ο επιρρηματικός τροποποιητής (adverbial modifier) είναι το «πότε», ως εκ τούτου, ο τύπος απάντησης είναι «Ημερομηνία». Δυστυχώς, μερικές ερωτηματικές λέξεις όπως «Ποιο», «Τι» ή «Πώς» δεν δίνουν σαφείς τύπους απάντησης. Κάθε μία από αυτές τις λέξεις μπορεί να αντιπροσωπεύει περισσότερους από έναν τύπο. Σε περιπτώσεις όπως αυτή, και άλλες λέξεις μέσα στην ερώτηση πρέπει να εξεταστούν. Το πρώτο πράγμα που πρέπει να κάνουμε είναι να βρούμε τις λέξεις που μπορούν να δείξουν το νόημα της ερώτησης. Ένα λεξικό όπως το WordNet μπορεί στη συνέχεια να χρησιμοποιηθεί για την κατανόηση του πλαισίου.

Μόλις εντοπιστεί το είδος της ερώτησης, ένα σύστημα ανάκτησης πληροφορίας χρησιμοποιείται για να βρεθεί μια σειρά από έγγραφα που περιέχουν τις σωστές λέξεις-κλειδιά. Ένας Tagger και ένας chunker NP/Ομάδας ρήματος μπορούν να χρησιμοποιηθούν για να ελέγξουν αν οι σωστές οντότητες και οι σχέσεις που αναφέρονται βρέθηκαν στα έγγραφα. Για ερωτήσεις όπως "Ποιος" ή "Πού", ένας Αναγνωριστής Επώνυμης Οντότητας (Named Entity Recognizer) χρησιμοποιείται για να βρει σχετικά "Πρόσωπα" και "Τοποθεσίες" από τα ανακτημένα έγγραφα. Μόνο οι σχετικές παράγραφοι επιλέγονται για την κατάταξη.

Ένα μοντέλο διανυσματικού χώρου (vector space model) μπορεί να χρησιμοποιηθεί ως στρατηγική για την ταξινόμηση των υποψήφιων απαντήσεων. Ελέγξτε αν η απάντηση είναι του σωστού τύπου όπως καθορίζεται στο στάδιο της ανάλυσης του τύπου ερώτησης. Τεχνική συμπεράσματος (inference technique) μπορεί επίσης να χρησιμοποιηθεί για την επικύρωση των υποψήφιων απαντήσεων. Η βαθμολογία στη συνέχεια δίνεται σε καθένα από αυτούς τους υποψηφίους ανάλογα με τον αριθμό των λέξεων της ερώτησης που περιέχει και πόσο κοντά αυτές οι λέξεις είναι για τον υποψήφιο, όσο πιο κοντά τόσο το καλύτερο. Η απάντηση στη συνέχεια μεταφράζεται σε μια συμπαγή και ουσιαστική εκπροσώπηση μέσω ανάλυσης. Στο προηγούμενο παράδειγμα, η αναμενόμενη απάντηση είναι "1η Οκτωβρίου"

5.4 Πρόοδος

Τα συστήματα QA επεκτάθηκαν τα τελευταία χρόνια για να συμπεριλάβουν επιπλέον πεδία της γνώσης [59]. Για παράδειγμα, συστήματα έχουν αναπτυχθεί για να απαντήσουν αυτόματα σε χρονικά και γεοχωρικά ερωτήματα, ερωτήματα ορισμού και ορολογίας, βιογραφικές ερωτήσεις, πολύγλωσσες ερωτήσεις και απορίες σχετικά με το περιεχόμενο του ήχου, εικόνων και βίντεο. Τα τρέχοντα θέματα έρευνας της QA περιλαμβάνουν:

- δια δραστικότητα (interactivity) – αποσαφήνιση (clarification) ερωτήσεων ή απαντήσεων
- επαναχρησιμοποίηση ή προσωρινή αποθήκευση (caching) απαντήσεων
- αναπαράστασης γνώσης και συλλογισμός
- ανάλυση των κοινωνικών μέσων μαζικής ενημέρωσης (social media)
- ανάλυσης συναισθήματος (sentiment analysis) [60]
- αξιοποίηση των θεματικών ρόλων (thematic roles) [61]
- σημασιολογική ανάλυση: για να γεφυρώσει το χάσμα μεταξύ συντακτικά διαφορετικών ερωτήσεων και κειμένων που φέρουν την απάντηση [62]
- αξιοποίηση των γλωσσικών πόρων [63], όπως WordNet, FrameNet, και παρόμοια

5.5 Εφαρμογή

Το έργο της Απάντησης Ερώτησης είναι να βρει τις σωστές απαντήσεις στις ερωτήσεις των χρηστών που εκφράζονται σε φυσική γλώσσα. Ο παραδοσιακός αγωγός (pipeline) QA βασίζεται στα ακόλουθα βήματα: Ανάλυση Ερώτησης (Question Analysis), Ανάκτηση Κειμένων (Passage Retrieval), Εξαγωγή Απαντήσεων (Answer Extraction), Επιλογή Απάντησης / Επαλήθευση (Answer Selection / Validation). Το κλειστού χώρου QA αναφέρεται σε εργασίες QA που αφορούν συγκεκριμένες και περιορισμένες περιοχές (όπως η ιατρική). Η ενασχόληση με ερωτήσεις κλειστού χώρου είναι γενικά πιο εύκολο έργο γιατί μπορεί να αξιοποιηθεί η γνώση σε συγκεκριμένους τομείς, και μόνο ένας περιορισμένος τύπος ερωτήσεων είναι αποδεκτός. Το QA ανοιχτού χώρου δεν αναφέρεται σε ένα συγκεκριμένο τομέα και έχει να κάνει με πιο γενικές ερωτήσεις. Αυτό συνήθως απαιτεί τη χρήση της παγκόσμιας γνώσης από την οποία εξάγεται η απάντηση. Το Web χρησιμοποιείται γενικά ως πηγή γνώσης, προκειμένου να εκμεταλλευτεί την περίσσια των πληροφοριών, επιλέγοντας τις απαντήσεις ανάλογα με τη συχνότητα τους, μεταξύ των αποτελεσμάτων αναζήτησης [15][27]. Αυτή η τεχνική συχνά συμπληρώνεται από εξόρυξη μοτίβων κειμένου προκειμένου να βρει τις ακριβείς απαντήσεις και να τις κατατάξει με βάση την σιγουριά [24][40]. Χρησιμοποιούνται NLP τεχνικές για την κατανόηση των ερωτήσεων των χρηστών και την εύρεση αποσπασμάτων που προέρχονται από έγγραφα [23][25]. Όπως φαίνεται και στο [11], η υιοθέτηση του NLP παίζει καθοριστικό ρόλο, δεδομένου ότι υπάρχουν πολύ λίγες πιθανές απαντήσεις στις ερωτήσεις των χρηστών, και ο τρόπος με τον οποίο εκφράζονται ενδέχεται να διαφέρουν σημαντικά από το ερώτημα. Έτσι, η NLP είναι απαραίτητη για την ανακάλυψη λεξικολογικών, συντακτικών και σημασιολογικών σχέσεων μεταξύ των ερωτήσεων και των υποψήφιων απαντήσεων. Τα πιο συχνά βήματα γλωσσικής ανάλυσης περιλαμβάνουν την αφαίρεση εξαιρούμενων λέξεων (stopword removal), τις λέξεις που απορρέουν (stemming), τη λημματοποίηση (lemmatization), την ταυτοποίηση μέρους του λόγου, την ανάλυση, την αναγνώριση

οντότητας, την αποσαφήνιση της αίσθησης της λέξης (word sense disambiguation) και τη σήμανση σημασιολογικού ρόλου (semantic role labeling). Άλλες προσεγγίσεις βασίζονται σε ένα διαφορετικό είδος γνώσης για να εξαγάγουν τις απαντήσεις σε ερωτήσεις. Μια προσέγγιση [17] για QA ανοικτού πεδίου πάνω από βάσεις μαζικής γνώσης πραγματοποιείται από την αποσύνθεση του πλήρους ανοικτού προβλήματος QA σε μικρότερα υπο-προβλήματα, συμπεριλαμβανομένων της παράφρασης της ερώτησης και την αναδιατύπωση του ερωτήματος (query reformulation). Στο πρώτο βήμα, το ερώτημα ξαναγράφεται με τη χρήση ενός φορέα παράφρασης, αφού εξορύσσεται από ένα μεγάλο σώμα ερωτήσεων, προκειμένου να μειωθεί η διακύμανση των ερωτήσεων εισόδου: για παράδειγμα, "Πώς μπορείτε να πείτε εάν έχετε γρίπη?" έχει αναδιατυπωθεί ως “Ποια είναι τα σημάδια της γρίπης?". Το δεύτερο βήμα χρησιμοποιεί χειρόγραφα πρότυπα για να αναλύσει τη παραφρασμένη ερώτηση σε ένα συγκεκριμένο KB ερώτημα, το οποίο στη συνέχεια αναδιατυπώνεται μέσω ενός χειριστή επανεγγραφής ερωτήματος (query-rewrite operator) για να αντιμετωπίσει την λεξικολογική αναντιστοιχία μεταξύ των λέξεων του ερωτήματος και των KB συμβόλων.

Στην εργασία [1] οι συγγραφείς έχουν υιοθετήσει τις πηγές γνώσης Wikipedia και DBpedia για να εξαγάγουμε τις απαντήσεις στις ερωτήσεις, και έχουμε ενσωματώσει κοινά βήματα γλωσσικής ανάλυσης. Η καινοτομία είναι η υιοθέτηση των Σημασιολογικών Μοντέλων Κατανομών (Distributional Semantic Model) (DSM) [48] για τον υπολογισμό της σημασιολογικής ομοιότητας μεταξύ των ερωτήσεων και των εγγράφων. Αυτό είναι μια καινοτομία στο πεδίο QA, ειδικά για την εργασία της εκ νέου κατάταξης απαντήσεων (answer re-ranking).

Κάποιοι συγγραφείς [5] προτείνουν ένα σημασιολογικό αναλυτή που χαρτογραφεί ερωτήσεις σε απαντήσεις μέσω λανθανουσών λογικών μορφών. Πρώτα δημιουργούν μια σειρά από λογικές μορφές εκμεταλλευόμενοι ενός λεξικού που χαρτογραφεί λογικά κατηγορήματα σε φράσεις που χτίστηκε από μια βάση γνώσεων και ένα μεγάλο σώμα κειμένου. Μπορούν στη συνέχεια να υπολογίσουν την πιθανότητα της κάθε υποψήφιας λογικής μορφής με ένα μοντέλο λογαριθμικής πολυπλοκότητας που εκμεταλλεύεται λεξιλογικές και λογικές λειτουργίες, και έχει εκπαιδευτεί σε ζεύγη ερωτήσεων-απαντήσεων. Η προσέγγιση αυτή διαφέρει από τη δική τους, επειδή στηρίζονται κυρίως σε λεξιλογικά χαρακτηριστικά για την ανάκτηση κειμένων παρά τη χαρτογράφηση ερωτήσεων στη λογική μορφή τους, και προσπαθούνε να χαρτογραφήσουνε τις ερωτήσεις σε τριπλέτες της DBpedia αξιοποιώντας την λεξικοποίηση των κατηγορημάτων, η οποία έχει κάποια κοινά στοιχεία με το στάδιο κατασκευής λεξικού.

5.5.1 Ερωταποκρίσεις για την μηχανική ανάγνωση και την επικύρωση απάντησης

Τα QA συστήματα ασχολούνται γενικά με απλές ερωτήσεις που απαιτούν σχεδόν καμία συμπερασματολογία για να βρουν τις σωστές απαντήσεις, μιας και καμία πραγματική κατανόηση των εγγράφων δεν γίνεται. Αυτό ιστορικά οδήγησε σε αρχιτεκτονικές QA που βασίζονται σε τεχνικές ανάκτησης πληροφοριών, στις οποίες οι τελικές απαντήσεις λαμβάνονται εστιάζοντας σε επιλεγμένα τμήματα των ανακτημένων εγγράφων και αντιστοίχισης κομματιών προτάσεων ή μεταγλώττιση δέντρων προτάσεων. Άλλα συστήματα εκτελούν μια βαθύτερη ανάλυση των κειμένων, για να λύσουν προβλήματα που συνεπάγονται κάποιο είδος συλλογισμού. Για παράδειγμα, στο έργο της μηχανικής ανάγνωσης [41], ο στόχος είναι να απαντήσουν σε ερωτήσεις που απαιτούν μια βαθιά γνώση των ατομικών σύντομων κείμενων και αυτά τα συστήματα πρέπει να επιλέξουν μία απάντηση, αναλύοντας τα αντίστοιχα δοκιμαστικά έγγραφα σε συνδυασμό με τις διάφορες συλλογές κειμένων. Άλλες σύνθετες εργασίες περιλαμβάνουν την Αναγνώριση

Κειμενικής Συνεπαγωγής (Recognizing Textual Entailment) (RTE) [13] και η Άσκηση Επικύρωσης Απάντησης (Answer Validation Exercise) (AVE) [46]. Στο RTE, ένα σύστημα πρέπει να αποφασίσει αν η έννοια του κειμένου T συνεπάγεται την έννοια ενός διαφορετικού κειμένου H, δηλαδή η υπόθεση. Διαφορετικά, το AVE συνίσταται στο να αποφασιστεί εάν μια απάντηση σε ένα ερώτημα είναι σωστή ή όχι, σύμφωνα με ένα συγκεκριμένο κείμενο. Τα συστήματα αυτά λαμβάνουν ένα σύνολο τριάδων (ερώτηση, απάντηση και υποστηρικτικό κείμενο) και πρέπει να επιστρέψουν μια τιμή για κάθε τριάδα, η οποία μπορεί να είναι: ΕΠΙΚΥΡΩΜΕΝΟ (VALIDATED), δηλαδή η απάντηση είναι σωστή και υποστηρίζεται, αν και δεν επιλέγεται, ΕΠΙΛΕΓΜΕΝΟ (SELECTED), δηλαδή η απάντηση έχει επικυρωθεί και επιλέχθηκε ως την έξοδο, και ΜΗ ΕΠΙΛΕΓΜΕΝΟ (REJECTED), δηλαδή η απάντηση είναι λανθασμένη ή δεν είναι αρκετή η απόδειξη της ορθότητάς της. Προκειμένου να επιλυθούν αυτά τα καθήκοντα, εγκρίθηκαν διάφορες τεχνικές, κυρίως με βάση τη χρήση της λεξιλογικής επεξεργασίας, της συντακτικής επεξεργασίας, και των ονομαστικών οντοτήτων [45].

Η επικύρωση απάντησης [7] διεξάγεται με τον υπολογισμό της επικάλυψης μεταξύ της απόκρισης του συστήματος σε μια ερώτηση και των λέξεων που προήλθαν από ένα κλειδί απαντήσεων που δημιουργείται από άνθρωπο. Η διαίσθηση πίσω από αυτή τη στρατηγική είναι ότι μια καλή απάντηση αναμένεται να περιέχει συγκεκριμένες λέξεις-κλειδιά, αλλά η ακριβής διατύπωση δεν έχει σημασία. Αυτός είναι ο λόγος για τον οποίο οι συντάκτες χρησιμοποίησαν αφαίρεση εξαιρούμενων λέξεων και λέξεις που απορρέουν. Η επικύρωση απάντησης [33] βασίζεται στην διαίσθηση ότι η γνώση που συνδέει μια απάντηση σε μια ερώτηση μπορεί να εκτιμηθεί από την εκμετάλλευση του πλεονασμού των πληροφοριών στο Web. Πιο συγκεκριμένα, η υπόθεση είναι ότι ο αριθμός των εγγράφων που ανακτώνται από το Web στα οποία το ερώτημα και η απάντηση συνυπάρχουν είναι ένας καλός δείκτης της ισχύος της απάντησης.

Πιο πολύπλοκες στρατηγικές προτείνουν λύσεις για την επικύρωση απαντήσεων με βάση μια ελαφριά διαδικασία απαγωγής ξεκινώντας από τις παραγράφους του κειμένου, όπου οι απαντήσεις μπορούν να βρεθούν [22], ή με βάση τη χρήση της σημασιολογίας. Για παράδειγμα, το Castillo [10] κατασκευάζει ένα μοντέλο χρησιμοποιώντας τα Support Vector Machines για να καθορίσει εάν η επίπτωση υφίσταται, χρησιμοποιώντας ένα σύνολο λεξιλογικών και σημασιολογικών μετρήσεων για να υπολογίσουμε την ομοιότητα μεταξύ ζευγών υπόθεσης - κειμένου. Τα χαρακτηριστικά βασίζονται σε:

- την επικάλυψη μεταξύ κειμένου και υπόθεσης (υπολογίζεται λαμβάνοντας υπόψη μεμονωμένες λέξεις ή φράσεις, δίγραμμα και τρίγραμμα)
- την ομοιότητα συνημίτονου και την απόσταση Levenshtein μεταξύ του κειμένου και της υπόθεσης
- η σημασιολογική ομοιότητα χρησιμοποιώντας Wordnet.

Ο Glöckner [21] χρησιμοποίησε επίσης ρηχή εξόρυξη χαρακτηριστικών (όπως λεξιλογική επικάλυψη) για την επικύρωση απαντήσεων. Ένα τοπικό σκορ υπολογίζεται έπειτα και χρησιμοποιείται συσσωμάτωση για να καθορίσει ένα συνδυασμένο σκορ για κάθε απάντηση που αποτυπώνει την κοινή απόδειξη όλων των αποσπασμάτων που υποστηρίζουν την απάντηση. Ο Ahn [3] αξιοποιεί σημασιολογικές αναπαραστάσεις και τεχνικές εξαγωγής συμπερασμάτων κατά τη διαδικασία της εξόρυξης απάντησης από επιλεγμένα κείμενα. Οι υποψήφιες απαντήσεις βγαίνουν χρησιμοποιώντας μια "χαλαρή" μέθοδο ενοποίησης (unification method) που εκμεταλλεύεται την Prolog ενοποίηση, η οποία επιτρέπει να ορίσουμε υψηλές βαθμολογίες σε τέλειες αντιστοιχίες μεταξύ των όρων της ερώτησης και του κειμένου, και χαμηλές βαθμολογίες για λιγότερο τέλειες αντιστοιχίες που λαμβάνονται με χαλαρή ενοποίηση. Λιγότερο τέλειες αντιστοιχίες χορηγούνται για διάφορους σημασιολογικούς τύπους, κατηγορήματα με διαφορετική

σειρά ορισμάτων, ή όρους με σύμβολα που είναι σημασιολογικά σχετικά (hypernymy) σύμφωνα με το WordNet.

Οι συγγραφείς του άρθρου [1] εμπνεύστηκαν από τις προηγούμενες προσεγγίσεις για την κατασκευή του εικονικού παίκτη για το παιχνίδι WWBM, και δανείστηκαν τις περισσότερες από τις τεχνικές που θεσπίζονται στο έργο AVE, όπως τεχνικές που βασίζονται σε λεξιλογική επεξεργασία και υπολογισμό μιας κατά προσέγγιση αντιστοιχίας μεταξύ χωρίων κειμένου και των υποψήφιων απαντήσεων.

5.5.2 Ερωταποκρίσεις πάνω σε συνδεδεμένα δεδομένα (Linked Data)

Η ταχεία ανάπτυξη των σημασιολογικών πληροφοριών που δημοσιεύονται στο Διαδίκτυο, ιδίως μέσω της πρωτοβουλίας των συνδεδεμένων δεδομένων [6], θέτει νέες προκλήσεις κατά την παροχή υποστήριξης στους χρήστες να αναζητούν αυτά τα μεγάλα ποσά των ετερογενών και δομημένων σημασιολογικών δεδομένων χρησιμοποιώντας διεπαφές φυσικής γλώσσας. Αυτό οδήγησε στην άνοδο του QA με βάση τις οντολογίες, ένα νέο πρότυπο σε θέση να εκμεταλλευτεί την εκφραστική δύναμη των οντολογιών και να προχωρήσουμε πέρα από την αναπαράσταση των αναγκών των χρηστών για πληροφορίες, με ερωτήματα που βασίζονται σε λέξεις-κλειδιά.

Το Freya [14] επιτρέπει στους χρήστες να εισάγουν ερωτήματα σε οποιαδήποτε μορφή. Σε ένα πρώτο στάδιο, παράγει ένα συντακτικό δένδρο για να προσδιορίσει το είδος της απάντησης. Η επεξεργασία στη συνέχεια ξεκινά με μια αναζήτηση, σχολιασμό των όρων του ερωτήματος με τις έννοιες της οντολογίας με τη χρήση του λεξικού OntoRoot. Στη συνέχεια, με βάση τις οντολογικές αντιστοιχίσεις, τριπλέτες παράγονται και, τέλος, συνδυάζονται για να δημιουργήσουν ένα ερώτημα SPARQL. Με παρόμοιο τρόπο, το PowerAqua [31] μετατρέπει το ερώτημα σε ένα σύνολο από τριπλέτες μέσω της γλωσσικής επεξεργασίας, και χαρτογραφεί τις τριπλέτες σε κατάλληλους σημασιολογικούς πόρους σε διάφορες οντολογίες που είναι πιθανό να περιγράψουν τους όρους του ερωτήματος. Δεδομένων αυτών των σημασιολογικών πόρων, ένα σύνολο από τριπλέτες οντολογίας που καλύπτουν από κοινού το ερώτημα προέρχεται και συνδυάζονται σε μια πλήρη απάντηση, με τη συγχώνευση και την κατάταξη των διαφόρων ερμηνειών που παράγονται σε διαφορετικές οντολογίες.

Το QAKiS [9] είναι ένα σύστημα QA πάνω στην DBpedia που επικεντρώνεται στην γεφύρωση του χάσματος μεταξύ των εκφράσεων φυσικής γλώσσας και των εννοιών της οντολογίας μέσω του αποθετηρίου WikiFramework. Αυτό το αποθετήριο έχει κατασκευαστεί από την αυτόματη εξόρυξη σχεσιακών μοτίβων ελεύθερων κειμένων από τη Wikipedia που προσδιορίζει τις πιθανές λεξικοποιήσεις των όρων που υπάρχουν στην DBpedia. Για παράδειγμα, ένα από τα φυσικά γλωσσικά μοτίβα που εκφράζουν την ημερομηνία γέννησης είναι το «γεννήθηκε στις». Η προσέγγιση της χρησιμοποίησης μιας αποθήκης μοτίβων αντιπροσωπεύει μια πολλά υποσχόμενη λύση για τη γεφύρωση του χάσματος μεταξύ λεξιλογικών εκφράσεων της φυσικής γλώσσας και των ετικετών οντολογίας. Ομοίως με το QAKiS, το SemSek [2] επικεντρώνεται επίσης στην αντιστοίχιση των εκφράσεων της φυσικής γλώσσας σε έννοιες της οντολογίας. Αυτό βασίζεται σε τρία βήματα: τη γλωσσολογική ανάλυση (linguistic analysis), το σχολιασμό του ερωτήματος (query annotation) και τη σημασιολογική ομοιότητα (semantic similarity). Ο σχολιασμός του ερωτήματος ψάχνει κυρίως για τους φορείς και τις τάξεις σε ένα πίνακα της DBpedia που ταιριάζουν με τις εκφράσεις που συμβαίνουν στην ερώτηση σε φυσική γλώσσα. Η διαδικασία αυτή καθοδηγείται από το συντακτικό δένδρο που παρέχεται από τη γλωσσική ανάλυση. Ξεκινώντας από τους πιο αληθοφανείς πόρους και κλάσεις, το SemSek ανακτά μια ταξινομημένη λίστα των όρων από το δέντρο εξάρτησης. Για να ταιριάξει αυτούς τους όρους σε έννοιες DBpedia, το SemSek

χρησιμοποιεί δύο μέτρα σημασιολογικής ομοιότητας, ένα με βάση ρητή Σημασιολογική Ανάλυση [20], και ένα με βάση το WordNet.

Στο [32] τα αποτελέσματα είναι ενθαρρυντικά και δείχνουν ότι η QA πάνω σε συνδεδεμένα δεδομένα μπορεί να προσφέρει απαντήσεις σε αρκετά πολύπλοκες ανάγκες πληροφόρησης που εκφράζονται σε φυσική γλώσσα με τη χρήση ετερογενών σημασιολογικών δεδομένων, ακόμη και αν το έργο της χαρτογράφησης της φυσικής γλώσσας σε επίσημα ερωτήματα δεν είναι τετριμμένο και εξακολουθεί να έχει αρκετά προβλήματα [12].

6. ΛΗΨΗ ΑΠΟΦΑΣΗΣ

Πολλές αποφάσεις, ιδιαίτερα οι αποφάσεις στρατηγικής, έχουν μακροπρόθεσμο χαρακτήρα. Δηλαδή, τα αποτελέσματα από την υλοποίηση τους θα διαφανούν στο μέλλον και θα συνεχίσουν να εκδηλώνονται επί μακρόν, ενώ οι όποιες περιορισμένες δυνατότητες αναθεώρησης της απόφασης θα συνοδεύονται κατά κανόνα από σοβαρές – οικονομικές ή άλλες- επιπτώσεις. Θα πρέπει επομένως η λήψη μακροπρόθεσμων αποφάσεων να στηρίζεται στη γνώση των συνθηκών που θα ισχύσουν στο μέλλον έτσι ώστε να ληφθεί η ορθότερη δυνατή απόφαση. Ειδικότερα, θα πρέπει να εκτιμηθούν οι τιμές που θα διαμορφωθούν στο μέλλον σε σχέση με ορισμένες μη ελεγχόμενες μεταβλητές απόφασης, οι οποίες επηρεάζουν σημαντικά την απόδοση ή άλλες επιπτώσεις της απόφασης. Η αβεβαιότητα ως προς τη μελλοντική διαμόρφωση των παραμέτρων που επηρεάζουν την απόφαση εξαρτάται από τα χαρακτηριστικά του εξεταζόμενου συστήματος και το είδος του προβλήματος που επιχειρείται να επιλυθεί. Μπορεί δε να οφείλεται είτε στην ακρίβεια των στοιχείων, η οποία εξαρτάται από την προέλευση τους και την αξιοπιστία των πηγών είτε σε μη προβλέψιμους παράγοντες που θα επηρεάσουν τη διαμόρφωση των τιμών των παραμέτρων της απόφασης στο μέλλον.

Ανάλογα με το επίπεδο γνώσης των μελλοντικών συνθηκών διακρίνουμε τρεις δυνατές καταστάσεις στις οποίες λαμβάνονται οι αποφάσεις:

- **Συνθήκες βεβαιότητας**: υπάρχει σαφής και πλήρης γνώση για τις μελλοντικές εξελίξεις. Παρόμοιες συνθήκες διαμορφώνονται κυρίως σε προβλήματα ρουτίνας που επηρεάζονται από απλές και άμεσα μετρήσιμες παραμέτρους.
- **Συνθήκες ρίσκου (κινδύνου):** Οι διαθέσιμες πληροφορίες για την εξέλιξη των κρίσιμων παραμέτρων της απόφασης δεν επαρκούν για να προσδιορισθεί με βεβαιότητα ποιες συνθήκες θα διαμορφωθούν στο μέλλον, μπορούν όμως να εκτιμηθούν πιθανές εκδοχές συνθηκών και οι αντίστοιχες πιθανότητες εμφάνισης τους.
- **Συνθήκες αβεβαιότητας**: στην περίπτωση αυτή υπάρχει παντελής έλλειψη πληροφόρησης ως προς τις μελλοντικές εξελίξεις, έτσι ώστε να μην είναι εφικτή η εκτίμηση των πιθανοτήτων εμφάνισης κάθε μίας από τις εκτιμώμενες δυνατές εκδοχές των συνθηκών.

Εκτός από τη διάκριση ως προς το βαθμό γνώσης των μελλοντικών συνθηκών, οι τεχνικές διαχείρισης της αβεβαιότητας στη λήψη των αποφάσεων διαφοροποιούνται σε σχέση και με άλλα χαρακτηριστικά του προβλήματος και ειδικότερα ανάλογα με το αν οι εξεταζόμενες εναλλακτικές λύσεις και οι παράμετροι αβεβαιότητας εκφράζονται ως προς διακριτές ή συνεχείς μεταβλητές. Η ανάλυση αβεβαιότητας και ρίσκου εστιάζεται στο πρόβλημα διακριτών εναλλακτικών λύσεων και διακριτών μελλοντικών συνθηκών, ενώ στη δεύτερη περίπτωση γίνεται παράλληλα χρήση εννοιών και τεχνικών της θεωρίας των πιθανοτήτων. Η θεωρία πιθανοτήτων αξιοποιείται και στην περίπτωση συνεχών μεταβλητών, ενώ ευρύτατα εφαρμόζεται και η τεχνική της ανάλυσης ευαισθησίας.

Παρακάτω συνοψίζονται τα κύρια σημεία της παραδοσιακής διαδικασίας λήψης αποφάσεων:

- **Καθορισμός προβλήματος:** Αναφέρεται στη διαδικασία σωστού και ρητού προσδιορισμού ενός θεμελιώδους προβλήματος ή την ευκαιρία (σε αντιδιαστολή με τα προφανή συμπτώματα) προσωπικής επικοινωνίας από τον διοικητικό φορέα. Ο καθορισμός του προβλήματος επιτρέπει στον διοικητικό φορέα να θέσει την εξής πολύ σημαντική ερώτηση: Τι πρέπει να γίνει για να λυθεί αυτό το ιδιαίτερο πρόβλημα; Κατά συνέπεια, ο καθορισμός του προβλήματος στρέφει την

προσοχή στις πιθανές εναλλακτικές λύσεις δράσης που πρέπει σε γενικές γραμμές να οδηγήσουν στην επίτευξη των συγκεκριμένων στόχων

- **Συλλογή πληροφοριών:** Η συλλογή πληροφοριών έχει ως στόχο να εξακριβώσει τα σχετικά γεγονότα που σχετίζονται με το πρόβλημα απόφασης. Συνήθως, αυτό περιορίζεται σε ένα πρόβλημα αναζήτησης. Οι χαρακτηριστικές πηγές πληροφοριών είναι δημοσιευμένα άρθρα και εκθέσεις-αναφορές, εσωτερικά αρχεία της επιχείρησης, έρευνες αγοράς, προσωπικές απόψεις και γνώμες των διάφορων συμμετεχόντων που επιλέγονται μέσα από συνεντεύξεις και ερωτηματολόγια ή ακόμα και άτυπες συνομιλίες, επαγγελματικές διαβουλεύσεις ή άμεση παρατήρηση από τον διοικητικό φορέα των πραγματικών σχετιζόμενων προβλημάτων με τις διαδικασίες μέσα ή έξω από στην οργάνωση.
- **Προσδιορισμός εναλλακτικών ενεργειών:** Η δημιουργικότητα ζητείται σε αυτή την φάση της γενικής διαδικασίας απόφασης. Δεδομένου ότι ο διοικητικός φορέας συγκεντρώνει τις πληροφορίες, κατόπιν εντοπίζει τις πιθανές εναλλακτικές ενέργειες. Η κλασική λήψη απόφασης εστιάζει την προσοχή σε υποκειμενικές πτυχές όπως η διαίσθηση, η εμπειρία και η κρίση προκειμένου να παραχθούν οι υγιείς εναλλακτικές ενέργειες. Επιπλέον, μπορούν να επικαλεστούν ακόμη πιο επίσημες μέθοδοι, όπως το "brainstorming", τις ομάδες εστίασης και τους ποιοτικούς κύκλους. Έμφαση πρέπει σε αυτό το σημείο να δοθεί στην παραγωγή των εναλλακτικών λύσεων, χωρίς αυτές να επικρίνονται.
- **Αξιολόγηση εναλλακτικών λύσεων:** Ο διοικητικός φορέας συγκρίνει τα πλεονεκτήματα και τα μειονεκτήματα «έμφυτα» με κάθε εναλλακτική ενέργεια. Οι δαπάνες και τα κέρδη υπολογίζονται και ο αντίκτυπός τους στους οργανωτικούς στόχους αξιολογείται. Οι αδύνατες εναλλακτικές λύσεις εξαιρούνται και ένα ελάχιστο σύνολο προτιμημένων επιλογών καθορίζεται, συχνά αποτελούμενες από δύο τελικούς υποψηφίους.
- **Επιλογή καλύτερης εναλλακτική ενέργειας**: Πρόκειται για το κλασικό σημείο λήψης αποφάσεων. Μέχρι τώρα, όσον αφορά το παραδοσιακό επιχείρημα, ο διοικητικός φορέας οφείλει να είναι σαφής σε ποιες εναλλακτικές ενέργειες προσφέρεται το καλύτερο σχέδιο δράσης. Συνεπώς, χρησιμοποιώντας την καλύτερη προσωπική κρίση, λαμβάνεται η απόφαση.
- **Εφαρμογή επιλεγμένης εναλλακτικής λύσης:** Ο διοικητικός φορέας θέτει σε εφαρμογή ένα σχέδιο δράσης που περιλαμβάνει τους συνηθισμένους διευθυντικούς στόχους: προγραμματισμός, οργάνωση, ηγεσία και έλεγχος. Είναι το σημείο αυτό όπου οι λειτουργικές επιχειρησιακές ειδικότητες τίθενται σε εφαρμογή: παραγωγή / διαδικασίες, μάρκετινγκ, οικονομική διαχείριση, λογιστική και, στην απαραίτητη έκταση, διαχείριση ανθρώπινου δυναμικού. Δεν υπάρχει τίποτα πραγματικά λανθασμένο με την ανωτέρω διαδικασία λήψης αποφάσεων. Σε γενικές γραμμές, τείνει να παράγει τις εύλογα αποτελεσματικές αποφάσεις μέσα σε σταθερό οικονομικό περιβάλλον. Δυστυχώς, τείνει επίσης να καλύψει τους ενδεχομένως σοβαρούς κινδύνους, μερικοί από τους οποίους αναπτύσσονται εν συνεχεία.

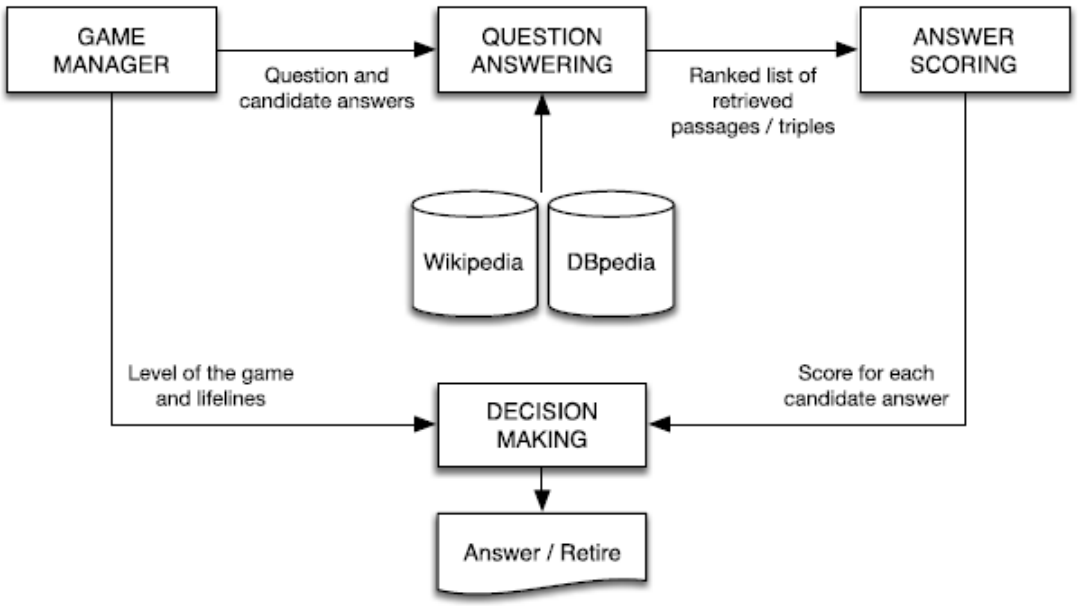
7. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ΕΙΚΟΝΙΚΟΥ ΠΑΙΚΤΗ

Η αρχιτεκτονική του εικονικού παίκτη για το παιχνίδι WWBM όπως αυτή υλοποιήθηκε στο άρθρο [1] παρουσιάζεται στο Σχήμα 2 και αποτελείται από τέσσερις ενότητες (την ίδια αρχιτεκτονική ακολούθησα και εγώ στην δικιά μου υλοποίηση):

- **Διαχειριστής Παιχνιδιού (Game Manager):** διαχειρίζεται το περιβάλλον εργασίας του χρήστη, επιλέγει μια ερώτηση για κάθε επίπεδο του παιχνιδιού, και καταγράφει τις πληροφορίες για κάθε παιχνίδι για τους διάφορους παίκτες. Στην υλοποίηση της παρούσας εργασίας αυτή η ενότητα είναι το κυρίως πρόγραμμα (το γραφικό περιβάλλον και το κυρίως νήμα εκτέλεσης).
- **Απάντηση Ερώτησης (Question Answering):** αξιοποιεί τη Wikipedia και τη DBpedia για την ανάκτηση και την κατάταξη των πιο σχετικών χωρίων κειμένου που είναι χρήσιμα για να προσδιοριστεί η σωστή απάντηση σε μία ερώτηση. Στην παρούσα εργασία, αυτή η ενότητα είναι εκείνο το κομμάτι του κώδικα που χειρίζεται την ερώτηση, κάνει διάφορα φιλτραρίσματα, ρωτάει την μηχανή αναζήτησης της Google και παίρνει τα αποτελέσματα που επιστρέφονται.
- **Βαθμολογία Απαντήσεων (Answer Scoring):** αξιοποιεί τη λίστα των αποσπασμάτων κειμένου που εξάγονται από την ενότητα QA και υιοθετεί αρκετά κριτήρια για να εκχωρήσει μια βαθμολογία για κάθε μία από τις τέσσερις πιθανές απαντήσεις σε μια ερώτηση. Στην παρούσα εργασία, αυτή η ενότητα είναι το κομμάτι κώδικα που διαβάζει τα κείμενα που επέστρεψε η Google και προσπαθεί να βρει μέσα τις πιθανές απαντήσεις δίνοντας μια τελική βαθμολογία σε κάθε απάντηση. Έγινε μια προσπάθεια να υλοποιηθούν τα διάφορα κριτήρια που θα αναφερθούν παρακάτω χωρίς όμως αυτό να γίνει εφικτό λόγω της μεγάλης πολυπλοκότητας.
- **Λήψη Απόφασης (Decision Making):** λαμβάνει την τελική απόφαση για το αν θα απαντήσει σε ερώτηση ή θα φύγει από το παιχνίδι λαμβάνοντας υπόψη το τρέχον επίπεδο του παιχνιδιού, τις διαθέσιμες σανίδες σωτηρίας, και τη βαθμολογία που υπολογίζεται για κάθε πιθανή απάντηση από την ενότητα AS. Αυτή η ενότητα έχει υλοποιηθεί ακριβώς όπως περιγράφεται στο άρθρο [1].

Σύμφωνα με το τρέχον επίπεδο δυσκολίας του παιχνιδιού, το GM επιλέγει μια ερώτηση μαζί με τις τέσσερις πιθανές απαντήσεις, οι οποίες στη συνέχεια περνάνε στην ενότητα QA. Η ενότητα αυτή αξιοποιεί τη γνώση που περιέχεται στη Wikipedia και στη DBpedia για να επιλέξει ένα ιεραρχημένο κατάλογο των αποσπασμάτων κειμένου που πιθανόν να περιέχουν τη σωστή απάντηση στην ερώτηση. Αυτά τα κείμενα υφίστανται επεξεργασία από το AS, το οποίο υλοποιεί μια σειρά από ευρετικά (heuristics) για να καταλήξει σε μια βαθμολογία για κάθε πιθανή απάντηση στο ερώτημα. Τέλος, η ενότητα DM αποφασίζει να παράσχει συγκεκριμένη απάντηση ή να αποχωρήσει από το παιχνίδι, λαμβάνοντας υπόψη τις βαθμολογίες των υποψηφίων απαντήσεων, τις διαθέσιμες σανίδες σωτηρίας και το τρέχον επίπεδο του παιχνιδιού.

Για να εξηγηθεί καλύτερα η διαδικασία της επιλογής απάντησης, θα χρησιμοποιείται το ακόλουθο παράδειγμα σε όλη την εργασία από δω και στο εξής. Ας εξετάσουμε το ζήτημα στην Εικόνα 1: “Ποιος σκηνοθέτησε το Blade Runner?”, στο οποίο η ορθή απάντηση είναι B) Ridley Scott. Ας εξετάσουμε επίσης το ιεραρχημένο κατάλογο των αποσπασμάτων κειμένου που παρέχονται από τη QA στον Πίνακα 1.



Σχήμα 2: Η αρχιτεκτονική του εικονικού παίκτη.

Κάθε κείμενο περιέχει τον τίτλο της σελίδας της Wikipedia από την οποία προήλθε, και το σκορ που υπολογίζεται από τη μονάδα QA. Πιο τυπικά, δεδομένου <q, (A, B, C, D)>, όπου q είναι το ερώτημα και (A, B, C, D) είναι οι τέσσερεις πιθανές απαντήσεις, η μονάδα QA επιστρέφει μια λίστα αποτελεσμάτων R_q = {<t_1, p_1, w_1>,. . . , <T|R_q|, p|R_q|, w |R_q|>}, όπου η τριπλέτα <t_i, p_i, w_i> αντιστοιχεί στο t_i τίτλο της σελίδας Wikipedia που περιέχει το p_i πέρασμα, και w_i είναι το σκορ που δείχνει τη σχέση του αποσπάσματος αυτού σε σχέση με το ερώτημα q. Ο κατάλογος R_q είναι άδειος, αν η μονάδα QA δεν είναι σε θέση να βρει αποσπάσματα σχετικά με το θέμα. Αυτό μπορεί να συμβεί όταν:

- οι πληροφορίες δεν περιλαμβάνονται σε καμία από τις υιοθετημένες πηγές γνώσης, δηλαδή Wikipedia ή DBpedia
- οι πληροφορίες περιέχονται σε μια από τις υιοθετημένες πηγές γνώσης, αλλά δεν είναι δυνατό να βρεθεί μια αντιστοιχία (κειμένου ή σημασιολογική) μεταξύ του ζητήματος και των χωρίων κειμένου που περιέχουν την απάντηση. Για παράδειγμα, ακόμη και αν η απάντηση στο ερώτημα «πότε γεννήθηκε ο Leonardo da Vinci?» περιέχεται στη Wikipedia, είναι δύσκολο να το βρει, αφού αναφέρεται ως "Leonardo da Vinci (15, Απριλίου, 1452-2 Μάη, 1519)". Οι συγγραφείς του άρθρου [1] λύνουν αυτό το πρόβλημα με την εφαρμογή μιας στρατηγικής που βασίζεται στη χρήση της DBpedia.
- το θέμα εμπίπτει σε μία από τις κατηγορίες που παραμένουν αναπάντητες από το σύστημα.

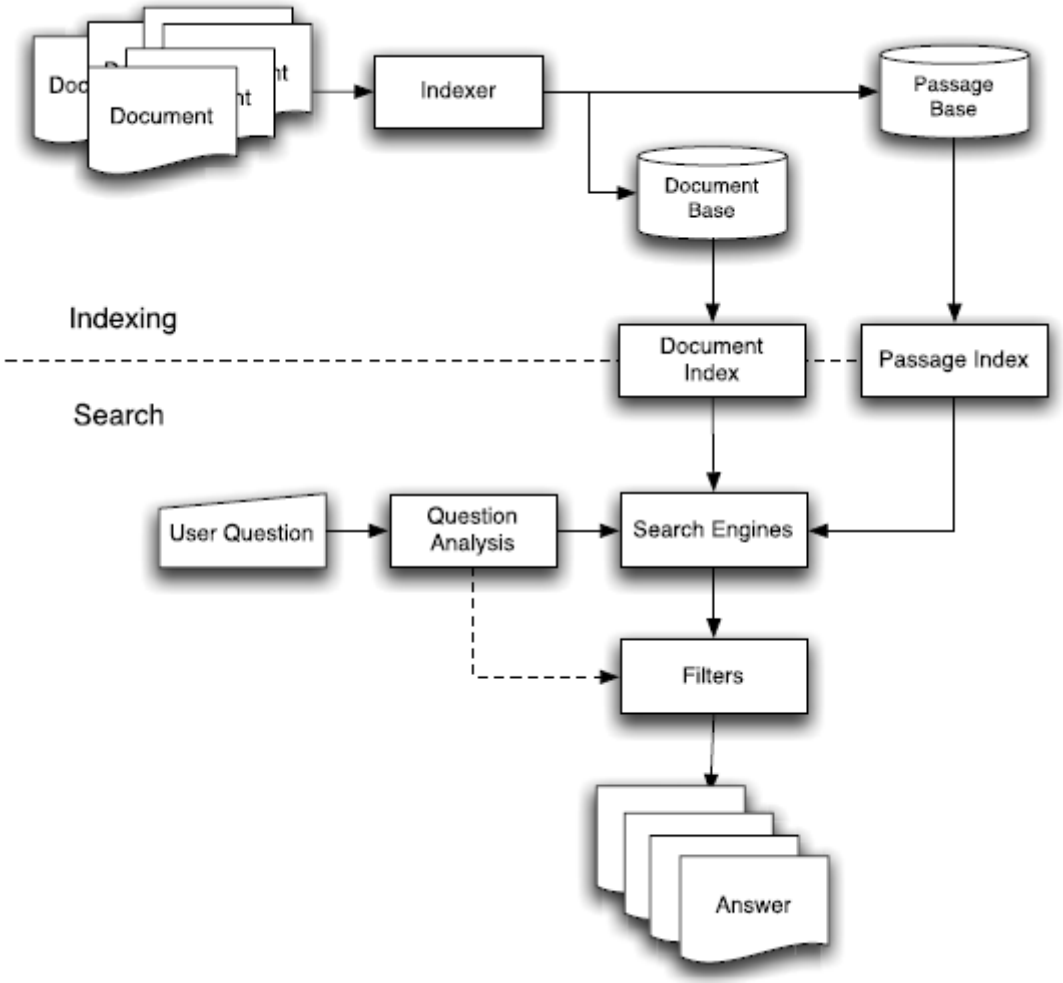
7.1 Απάντηση Ερώτησης

Προκειμένου να δοθεί στον εικονικό παίκτη η χρήσιμη γνώση για την εύρεση των σωστών απαντήσεων στις ερωτήσεις του παιχνιδιού, οι συγγραφείς του άρθρου [1] εκμεταλλεύτηκαν τα δεδομένα που προέρχονται από δύο ανοικτές πηγές γνώσης: Wikipedia και DBpedia. Στην παρούσα εργασία έγινε εκμετάλλευση γενικά των δεδομένων που προέρχονται από την μηχανή αναζήτησης της Google. Οι γνώσεις τους υφίστανται επεξεργασίαι από ένα πολύγλωσσο πλαίσιο QA, που ονομάζεται

QuestionCube [36][37], το οποίο μπορεί να ανακτήσει τα πιο σχετικά χωρία κειμένου που είναι πιθανό να περιέχουν τη σωστή απάντηση στις ερωτήσεις του παιχνιδιού.

Πίνακας 1: Λίστα από κείμενα που βρέθηκαν από τη μονάδα QA για την ερώτηση "Ποιος σκηνοθέτησε το Blade Runner?". Κάθε κείμενο έχει την σελίδα της Wikipedia στο οποίο βρέθηκε και τον βαθμό της σχέσης του κειμένου με την ερώτηση.

Wikipedia page (t_i)	Passage (p_i)	QA score (w_i)
Ridley Scott	Sir Ridley Scott (born 30 November 1937) is an English film director and producer. Following his commercial breakthrough with Alien (1979), his best-known works are the sci-fi classic Blade Runner (1982) and the best picture Oscar-winner Gladiator (2000).	5.32
Blade Runner	Blade Runner is a 1982 American dystopian science fiction action film directed by Ridley Scott and starring Harrison Ford, Rutger Hauer, and Sean Young. The screenplay, written by Hampton Fancher and David Peoples, is loosely based on the novel Do Androids Dream of Electric Sheep? by Philip K. Dick.	5.1
Blade Runner	Director Ridley Scott and the film's producers "spent months" meeting and discussing the role with Dustin Hoffman, who eventually departed over differences in vision. Harrison Ford was ultimately chosen for several reasons.	5
Blade Runner	The screenplay by Hampton Fancher was optioned in 1977. Producer Michael Deeley became interested in Fancher's draft and convinced director Ridley Scott to film it.	4.9
Blade Runner	Interest in adapting Philip K. Dick's novel Do Androids Dream of Electric Sheep? developed shortly after its 1968 publication. Director Martin Scorsese was interested in filming the novel, but never optioned it.	1.2



Σχήμα 3: Η αρχιτεκτονική του QuestionCube

Το Σχήμα 3 απεικονίζει μια αρχιτεκτονική υψηλού επιπέδου ενός QA framework. Λειτουργεί σε δύο χωριστά στάδια: την στιγμή ευρετηρίασης (indexing time) που το σύστημα χτίζει δύο διαφορετικά ευρετήρια για τα έγγραφα και τα κείμενα που ανήκουν σε κάθε έγγραφο, και τη στιγμή του ερωτήματος (query time), όπου το ερώτημα αναλύεται από έναν αγωγό NLP, σχολιασμένο με γλωσσικές ετικέτες, και περνάει σε μια σειρά από μηχανές αναζήτησης. Τέλος, τα κείμενα που ανήκουν στα έγγραφα που ανακτώνται από τις μηχανές αναζήτησης βαθμολογούνται, φιλτράρονται και επιστρέφουν ως πίνακες κατάταξης. Περισσότερες λεπτομέρειες:

Ανάλυση Ερώτησης (Question Analysis): ένας αγωγός από NLP αναλυτές υιοθετείται για να επισημάνει τις ερωτήσεις με γλωσσικούς σχολιασμούς. Υιοθετείτε ο ακόλουθος αγωγός: λέξεις που προκύπτουν (stemming), μέρος του λόγου tagging, λημματοποίηση, και αναγνώριση επώνυμων οντοτήτων (Named Entity Recognition) (NER). Αυτή η παράσταση είναι η είσοδος στις μηχανές αναζήτησης, αλλά και σε κάθε φίλτρο, τα οποία χρειάζονται αυτές τις πληροφορίες για να πραγματοποιήσουν την ανάλυση στο ίδιο γλωσσικό επίπεδο του ζητήματος.

Μηχανές αναζήτησης (Search Engines): η χρήση πολλαπλών μηχανών αναζήτησης επιτρέπει την ενσωμάτωση διαφόρων στρατηγικών ανάκτησης και πηγών δεδομένων. Κάθε μηχανή αναζήτησης έχει τη δική της συνιστώσα γέννησης ερωτημάτων, δεδομένου ότι μπορεί να εκμεταλλευτεί διαφορετικές ετικέτες που προστίθενται στο ερώτημα και μπορεί να υιοθετήσει μια διαφορετική σύνταξη ερωτήματος. Όταν ένα νέο ερώτημα έρχεται, κάθε μηχανή είναι ενεργοποιημένη και οι κατάλογοι των ανακτημένων εγγράφων συγχωνεύονται σε μια ενιαία λίστα, διατηρώντας την αναφορά στην προέλευση του κάθε εγγράφου και την αντίστοιχη βαθμολογία που υπολογίζεται από κάθε μηχανή. Ξεκινώντας από τα ανακτημένα έγγραφα, ο δείκτης κειμένων χρησιμοποιείται για να ληφθεί η λίστα των κειμένων, που είναι η είσοδος στα Φίλτρα. Υιοθετούνται τρεις διαφορετικές μηχανές αναζήτησης:

- δύο μηχανές αναζήτησης με βάση το μοντέλο BM25 [43], που λειτουργούν με λέξεις-κλειδιά και λήμματα που προέρχονται από την Wikipedia, αντίστοιχα.
- μία μηχανή αναζήτησης πάνω στις τριπλέτες της DBpedia.

Φίλτρα (Filters): ένας αγωγός φίλτρων υιοθετείται για να εκχωρήσει μια βαθμολογία σε κάθε κείμενο που ανήκει σε έγγραφα που ανακτώνται στο προηγούμενο βήμα:

- **Zero φίλτρο**: απομακρύνει κείμενα που έχουν βαθμολογία ίση με 0.
- **Φίλτρο Top-N**: επιλέγει τα N κορυφαία κείμενα, δηλαδή εκείνα με την υψηλότερη βαθμολογία.
- **Φίλτρο όρων**: εκχωρεί ένα σκορ σε ένα κείμενο με βάση τη συχνότητα εμφάνισης των όρων του ερωτήματος στο κείμενο.
- **Φίλτρο ακριβής ακολουθίας (exact sequence)**: εκχωρεί ένα σκορ σε ένα κείμενο με βάση τον αριθμό των όρων που εμφανίζονται στη μεγαλύτερη επικαλυπτόμενη ακολουθία λέξεων μεταξύ της ερώτησης και του κειμένου.
- **Φίλτρο Κανονικοποίησης (Normalization)**: αναθέτει μια κανονικοποιημένη βαθμολογία σε ένα κείμενο με βάση το μήκος του κειμένου. Έχουν υλοποιηθεί τόσο ένα φίλτρο που κάνει απλή κανονικοποίηση λαμβάνοντας υπόψη μόνο τον αριθμό των όρων (που ονομάζεται Byte-size κανονικοποίηση) όσο και ένα φίλτρο που βασίζεται στην τεχνική Περιστρεφόμενου Κανονικοποιημένου Μήκους Εγγράφου (Pivoted Normalized Document Length) (PNDL) [34][51].
- **Φίλτρο N-grams**: εκχωρεί ένα σκορ σε ένα κείμενο με βάση την αλληλοεπικάλυψη των n-grams μεταξύ του ερωτήματος και του κειμένου.

- **Φίλτρο Πυκνότητας (Density)**: εκχωρεί ένα σκορ σε ένα κείμενο με βάση την απόσταση των όρων της ερώτησης μέσα σε αυτό το κείμενο. Όσο πιο κοντά είναι μεταξύ τους οι όροι της ερώτησης μέσα στο κείμενο, τόσο υψηλότερη είναι η βαθμολογία. Η πυκνότητα υπολογίζεται χρησιμοποιώντας μια τροποποιημένη έκδοση της Ελάχιστης Στάθμισης Διασποράς (Minimal Span Weighting) [39]: $(|q \cap d|)/(1 + \max(mms) - \min(mms))$ όπου q και d είναι το σύνολο των όρων στο ερώτημα και στο έγγραφο, αντίστοιχα (συγκεκριμένα, το ερώτημα είναι η ερώτηση και το έγγραφο είναι το κείμενο), max(mms) και min(mms) είναι η αρχική και η τελική θέση της αλληλουχίας των όρων του κειμένου που περιέχουν όλους τους όρους του ερωτήματος.
- **Φίλτρο κατανομής (Distributional)**: εκχωρεί ένα σκορ σε ένα απόσπασμα κειμένου που βασίζεται στην ομοιότητα του με το ερώτημα, που υπολογίζεται χρησιμοποιώντας ένα Σημασιολογικό Μοντέλο Κατανομής. Το DSM χτίστηκε εφαρμόζοντας Λανθάνουσα Σημασιολογική Ανάλυση (Latent Semantic Analysis) (LSA) στον πίνακα όρου-όρου των σελίδων της Wikipedia. Κατασκευάζουμε έναν πίνακα που περιέχει τους 100.000 πιο συχνούς όρους της Wikipedia (οι εξαιρούμενες λέξεις αφαιρούνται και οι εκφράσεις με πολλές λέξεις δεν επιτρέπονται). Στα DSMs, δεδομένης της αναπαράστασης δύο λέξεων με διάνυσμα $u = (u_1, u_2,..., u_n)^T$ και $v = (v_1, v_2,..., v_n)^T$ (Χρησιμοποιήσαμε διανύσματα με 400 διαστάσεις), είναι πάντοτε δυνατό να υπολογισθεί η ομοιότητά τους ως το συνημίτονο της γωνίας μεταξύ τους. Ωστόσο, το ερώτημα και τα κείμενα είναι φράσεις που αποτελούνται από διάφορους όρους, γι' αυτό για να υπολογίσουμε την ομοιότητα τους χρειαζόμαστε μια μέθοδο για να συνθέσει τις λέξεις που εμφανίζονται σε αυτές τις προτάσεις. Είναι δυνατόν να συνδυαστούν λόγια μέσω πρόσθεσης διανυσμάτων (+). Ο εν λόγω φορέας είναι παρόμοιος με την υπέρθεση που ορίζεται στα ενωτικά (connectionist) συστήματα [52]. Η πρόσθεση είναι μια αντιμεταθετική πράξη, το οποίο σημαίνει ότι δεν λαμβάνει υπόψη οποιεσδήποτε υποκείμενες δομές που υπάρχουν μεταξύ των λέξεων. Πιο πολύπλοκες μέθοδοι θα μπορούσαν να αξιοποιηθούν για να συνδυάσουμε διανύσματα λέξεων. Επισήμως, αν $q = q_1\ q_2. . .\ q_n$ και $p = p_1\ p_2. . .\ p_m$ είναι το ερώτημα και το κείμενο αντίστοιχα, χτίζουμε δύο διανύσματα q και p, που αντιπροσωπεύουν το ερώτημα και το κείμενο σε ένα σημασιολογικό χώρο αντίστοιχα. Η διανυσματική αναπαράσταση χτίζεται με την εφαρμογή της πρόσθεσης στα διανύσματα των λέξεων που ανήκουν σε αυτά. Η ομοιότητα μεταξύ q και p υπολογίζεται ως η ομοιότητα συνημίτονου μεταξύ τους. Αυτή η ομοιότητα είναι η βαθμολογία που αποδίδεται στο κείμενο.
- **Φίλτρο Z-Score**: εκχωρεί ένα σκορ σε ένα κείμενο με βάση την εξομάλυνση Z-Score [50] των βαθμολογιών που έχουν ανατεθεί στο κείμενο από τις μηχανές αναζήτησης και τα άλλα φίλτρα.
- **Φίλτρο CombSum**: εκχωρεί ένα σκορ σε ένα κείμενο από την άθροιση των βαθμολογιών που έχουν ανατεθεί στο κείμενο από τις μηχανές αναζήτησης και τα άλλα φίλτρα [50].

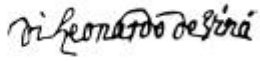
Το φίλτρο όρων και το φίλτρο πυκνότητας έχουν μια βελτιωμένη έκδοση που υιοθετεί το συνδυασμό των λημμάτων και των ετικετών μέρους του λόγου ως χαρακτηριστικά, αντί των όρων. Ένας παράγοντας ενίσχυσης μπορεί να ανατεθεί σε κάθε φίλτρο προκειμένου να αυξήσει (ή να μειώσει) τη δύναμή του.

7.1.1 Η χρησιμοποίηση της DBpedia σαν πηγή γνώσης

Το ερώτημα «Πότε γεννήθηκε ο Leonardo da Vinci;» δείχνει τη δυσκολία να εξαγάγουμε τη σωστή απάντηση, έστω και αν η πληροφορία περιέχεται στη Wikipedia. Στην περίπτωση αυτή, η ημερομηνία γέννησης είναι «15 Απριλίου, 1452», αλλά δεν μπορεί να προσδιοριστεί με την υιοθέτηση της κλασικής διαδικασίας ανάκτησης κειμένου που υλοποιείται από τη μονάδα Απάντησης Ερώτησης. Για τη διαχείριση αυτού του είδους ερωτήσεων, χρησιμοποιήθηκε μια συγκεκριμένη μηχανή αναζήτησης που αξιοποιεί τη γνώση που περιέχεται στην DBpedia. Η DBpedia περιλαμβάνει δομημένες πληροφορίες που ενσωματώνονται στα άρθρα της Wikipedia – το λεγόμενο «κουτί πληροφοριών» (infobox). Στο Σχήμα 4 φαίνεται το κουτί πληροφοριών για τον Leonardo da Vinci, το οποίο περιέχει πολλές χρήσιμες πληροφορίες, όπως η ημερομηνία γέννησης, ημερομηνία θανάτου, εθνικότητα, κλπ. Η DBpedia αναπαριστά τους πόρους, τις ιδιότητες καθώς και τις σχέσεις μεταξύ των πόρων με τη χρήση RDF τριάδων, οι οποίες περιέχουν τρία στοιχεία: το θέμα (subject), το κατηγόρημα (predicate), και το αντικείμενο (object).

Leonardo da Vinci

Portrait of Leonardo by Melzi

Born	Leonardo di ser Piero da Vinci April 15, 1452 Vinci, Republic of Florence (present-day Italy)
Died	May 2, 1519 (aged 67) Amboise, Kingdom of France
Known for	Diverse fields of the arts and sciences
Notable work(s)	*Mona Lisa* *The Last Supper* *The Vitruvian Man* *Lady with an Ermine*
Style	High Renaissance
Signature	

Σχήμα 4: Το κουτί πληροφοριών (infobox) του Leonardo Da Vinci

Για παράδειγμα, η RDF τριπλέτα που αντιπροσωπεύει την ημερομηνία γέννησης του Leonardo da Vinci είναι:

<http://dbpedia.org/resource/Leonardo_da_Vinci, dbpedia-owl:birthDate, 1452-04-15>

Η DBpedia επιτρέπει να διερευνούμε τις σχέσεις και τις ιδιότητες που συνδέονται με τους πόρους της Wikipedia, έτσι την ημερομηνία γέννησης του Leonardo da Vinci μπορούμε να την βρούμε από την πρόσβαση στο κατηγόρημα dbpedia-owl:birthDate. Προκειμένου να αξιοποιήσουμε τις γνώσεις που περιέχονται στην DBpedia, οι συγγραφείς του άρθρου [1] έχουν δημιουργήσει με το χέρι μια χαρτογράφηση μεταξύ των 50 πιο συχνών πεδίων της DBpedia και διαφορετικών λεξικογραφικοποιήσεων, σε μορφή ερώτησης, ζητώντας τα συγκεκριμένα πεδία. Για παράδειγμα, το κατηγόρημα dbpedia-owl:birthDate αντιστοιχίζεται με τις ερωτήσεις «Πότε γεννήθηκε?», ή «Ποια είναι η ημερομηνία γέννησης?», και άλλες παρόμοιες διατυπώσεις. Με αυτό τον τρόπο έχουν φτιάξει δύο σύνολα δεδομένων που περιέχει 347 ερωτήσεις για την ιταλική γλώσσα και 312 για τα αγγλικά, όπου κάθε ερώτηση έχει μαρκαριστεί με το αντίστοιχο κατηγόρημα της DBpedia. Κάθε σύνολο δεδομένων χρησιμοποιείται για να εκπαιδεύσει ένα ταξινομητή Rocchio [44], ο οποίος με δεδομένη μια ερώτηση, είναι σε θέση να προβλέψει το κατηγόρημα της DBpedia που έχει αντιστοιχιστεί σε αυτή. Τα στοιχεία που χρησιμοποιούνται για την εκπαίδευση κάθε ταξινομητή είναι όλες οι λέξεις που εμφανίζονται στις ερωτήσεις. Οι εξαιρούμενες λέξεις δεν καταργούνται, δεδομένου ότι λέξεις όπως Πότε, πώς, πού είναι χρήσιμες για να καθοδηγήσουν τον ταξινομητή για την ορθή κατάταξη.

Για να ανακτήσουν τις σχετικές πληροφορίες από την DBpedia ρώτησαν μια πρόσθετη μηχανή αναζήτησης που περιέχει έγγραφα που είναι η λεξικοποίηση των RDF τριάδων με το ίδιο θέμα, με τη μορφή: <ετικέτα του θέματος, ετικέτα του κατηγορήματος, ετικέτα του αντικειμένου> (τιμή του αντικειμένου, σε περίπτωση literals). Η λεξικοποίηση για το προηγούμενο παράδειγμα είναι: <Leonardo da Vinci, ημερομηνία γέννησης, 04/15/1452>. Μόνο τριπλέτες που σχετίζονται με τα 50 προ-επιλεγμένα κατηγορήματα έχουν λεξικοποιηθεί με αυτόν τον τρόπο. Κάθε έγγραφο έχει ένα επιπλέον πεδίο, την αναφορά στις ιδιότητες της DBpedia που περιέχει, όπως dbpedia-owl:birthDate. Όταν ένα ερώτημα (Ερώτηση) στέλνεται στη μηχανή αναζήτησης DBpedia, πρώτα ταξινομείται χρησιμοποιώντας τον ταξινομητή Rocchio, προκειμένου να προσδιοριστεί το κατηγόρημα στο οποίο αναφέρεται, μετά το επιλεγμένο κατηγόρημα προστίθεται στο ερώτημα, μαζί με τα named entities (αν υπάρχουν) που περιέχονται στην ερώτηση. Το ερώτημα υποβάλλεται στη μηχανή αναζήτησης, η οποία ανακτά το σύνολο των εγγράφων που σχετίζονται με το ερώτημα. Ξεκινώντας από τα έγγραφα, το σύστημα εξάγει την αντίστοιχη λίστα των κειμένων (RDF τριάδες), τα οποία σκοράρονται χρησιμοποιώντας τον αγωγό των φίλτρων που περιγράφεται παραπάνω, και ένα πρόσθετο φίλτρο κατηγορημάτων DBpedia. Αυτό το φίλτρο εκχωρεί ένα σκορ στις τριπλέτες που περιέχουν το κατηγόρημα που επέστρεψε ο ταξινομητής με την εμπιστοσύνη του, και τις τριάδες που δεν το περιέχουν με 0. Αξίζει να σημειωθεί ότι κάθε ερώτηση υποβάλλεται πάντα σε δύο μηχανές αναζήτησης που εργάζονται σε Wikipedia και DBpedia, προκειμένου να ανακτήσουμε αποτελέσματα και από τις δύο πηγές γνώσης.

7.2 Βαθμολόγηση Απάντησης

Ο κύριος στόχος της ενότητας AS είναι να εκχωρήσει μια βαθμολογία για κάθε μία από τις τέσσερις πιθανές απαντήσεις σε μια ερώτηση. Ομοίως με τις προσεγγίσεις που χρησιμοποιούνται στο πλαίσιο της Άσκησης Επικύρωσης Απάντησης [46], υιοθετήθηκαν από τους συγγραφείς πέντε κριτήρια με βάση την ανάλυση των χωρίων που επέστρεψε η μονάδα Απάντησης Ερώτησης. Κάθε κριτήριο επιστρέφει μια βαθμολογία για κάθε πιθανή απάντηση, η οποία είναι κανονικοποιημένη χρησιμοποιώντας το άθροισμα των βαθμολογιών των τεσσάρων πιθανών απαντήσεων. Πιο τυπικά, δεδομένου <q, (A, B, C, D)>, κάθε κριτήριο υπολογίζει <q, (C_A, C_B, C_C, C_D)>, όπου C_x είναι η βαθμολογία που ανατίθεται στην υποψήφια απάντηση X. Στη συνέχεια, κάθε κριτήριο περιγράφεται λαμβάνοντας υπόψη το τρέχον παράδειγμά μας.

Τίτλος Levenshtein (TL). Υπολογίζει την απόσταση Levenshtein μεταξύ μιας υποψήφιας απάντησης X και του τίτλου t_i της Wikipedia σελίδας που επέστρεψε η μονάδα QA. Η απόσταση Levenshtein μετρά τη διαφορά μεταξύ δύο συμβολοσειρών, και ορίζεται ως ο ελάχιστος αριθμός επεξεργασιών ενός χαρακτήρα (π.χ. εισαγωγή, διαγραφή, αντικατάσταση) που απαιτούνται για να αλλάξει μία χορδή στην άλλη. Καθώς η απόσταση Levenshtein είναι ένα μέτρο απόστασης, παρά ένα μέτρο ομοιότητας, υπολογίζεται ως εξής:

$$\frac{max(len(X), len(t_i)) - lev(X, t_i)}{max(len(X), len(t_i))}$$

όπου len (·) είναι η συνάρτηση που υπολογίζει το μήκος της συμβολοσειράς, και lev(X, t_i) είναι η απόσταση Levenshtein μεταξύ X και t_i, που χρησιμοποιείται ως παράγοντας κανονικοποίησης (να έχουν βαθμολογίες στο διάστημα [0, 1]). Στο τρέχον παράδειγμα, λαμβάνοντας υπόψη μόνο το πρώτο κείμενο που επιστρέφεται από τη μονάδα QA (Πίνακας 1), η απάντηση B) Ridley Scott εμφανίζεται στον τίτλο της σελίδας που περιέχει το κείμενο, γι' αυτό παίρνει τη μέγιστη βαθμολογία 1, ενώ η απάντηση A) Harrison Ford παίρνει ένα σκορ ίσο με (13-12)/13 = 0,077, η απάντηση C) Phillip Dick παίρνει μια βαθμολογία ίση με (12-10)/12 = 0,167, και η απάντηση D) James Cameron παίρνει ένα σκορ ίσο με (13-12)/13 = 0.077. Η κανονικοποιημένη βαθμολογία της κάθε απάντησης είναι:

$$c_A = \frac{0.077}{0.077+1+0.167+0.077} = 0.058$$
$$c_B = \frac{1}{0.077+1+0.167+0.077} = 0.757$$
$$c_C = \frac{0.167}{0.077+1+0.167+0.077} = 0.126$$
$$c_D = \frac{0.077}{0.077+1+0.167+0.077} = 0.058.$$

Μεγαλύτερη Κοινή Υποαλληλουχία (Longest Common Subsequence) (LCS). Υπολογίζει τη μακρύτερη κοινή υποαλληλουχία μεταξύ μιας υποψήφιας απάντησης X και ενός κειμένου p_i που επέστρεψε η μονάδα QA, ή απλά το τίτλο t_i της σελίδας Wikipedia που περιέχει το κείμενο. Στο τρέχον παράδειγμα, λαμβάνοντας υπόψη μόνο το δεύτερο κείμενο, η απάντηση A) Harrison Ford παίρνει ένα σκορ ίσο με 13, η απάντηση B) Ridley Scott παίρνει μια βαθμολογία ίση με 12, η απάντηση C) Philip Dick παίρνει μια βαθμολογία ίση με 11, και η απάντηση D) James Cameron παίρνει μια βαθμολογία ίση με 0, δεδομένου ότι δεν υπάρχει στο κείμενο. Η κανονικοποιημένη βαθμολογία της κάθε απάντησης είναι:

$$c_A = \frac{13}{13+12+11+0} = 0.361$$
$$c_B = \frac{12}{13+12+11+0} = 0.333$$
$$c_C = \frac{11}{13+12+11+0} = 0.305$$
$$c_D = \frac{0}{13+12+11+0} = 0.$$

Επικάλυψη (Overlap). Υπολογίζει το δείκτη Jaccard μεταξύ του συνόλου των όρων σε μια υποψήφια απάντηση X και του συνόλου των όρων σε ένα κείμενο p_i που επέστρεψε η μονάδα QA. Ο δείκτης Jaccard μετρά την ομοιότητα μεταξύ των σετ, και ορίζεται ως το μέγεθος της τομής διαιρούμενο με το μέγεθος της ένωσης των σετ. Στο τρέχον παράδειγμά μας, λαμβάνοντας υπόψη μόνο το δεύτερο κείμενο, οι απαντήσεις Α), Β) και Γ) παίρνουν μια βαθμολογία 2/49 = 0,041, ενώ η απάντηση D) παίρνει μια βαθμολογία ίση με 0. Η κανονικοποιημένη βαθμολογία της κάθε απάντησης είναι:

$$c_A = c_B = c_C = \frac{0.041}{0.041+0.041+0.041+0} = 0.333$$
$$c_D = \frac{0}{0.041+0.041+0.041+0} = 0.$$

Ακριβής Υποαλληλουχία (Exact Substring) (ES). Υπολογίζει το μήκος σε χαρακτήρες της μακρύτερης κοινής συμβολοσειράς μεταξύ μιας υποψήφιας απάντησης X και ενός κειμένου p_i που επέστρεψε η μονάδα QA, αφού ομαλοποιήθηκε χρησιμοποιώντας το μήκος της υποψηφίας απάντησης. Στο τρέχον παράδειγμα, λαμβάνοντας υπόψη μόνο το δεύτερο κείμενο, η απάντηση Α) Harrison Ford παίρνει ένα σκορ 13/13 = 1, η απάντηση Β) Ridley Scott παίρνει μια βαθμολογία των 12/12 = 1, η απάντηση Γ) Philip Dick παίρνει μια βαθμολογία 6/11 = 0.55, και η απάντηση D) James Cameron παίρνει ένα σκορ ίσο με 0. Η κανονικοποιημένη βαθμολογία για κάθε απάντηση είναι:

$$c_A = \frac{1}{1+1+0.55+0} = 0.392$$
$$c_B = \frac{1}{1+1+0.55+0} = 0.392$$
$$c_C = \frac{0.55}{1+1+0.55+0} = 0.215$$
$$c_D = \frac{0}{1+1+0.55+0} = 0.$$

Πυκνότητα (Density). Υπολογίζει την πυκνότητα των όρων σε μια υποψήφια απάντηση X μέσα σε ένα κείμενο p_i που επέστρεψε η μονάδα QA, χρησιμοποιώντας την τροποποιημένη έκδοση της μεθόδου ελάχιστου επικαλυπτόμενου διαστήματος που αναφέρθηκε προηγουμένως [39]. Στο τρέχον παράδειγμα, λαμβάνοντας υπόψη μόνο το δεύτερο κείμενο, οι απαντήσεις Α) και Β) παίρνουν ένα σκορ ίσο με 1, η απάντηση Γ) παίρνει ένα σκορ ίσο με 2/3 = 0.66 (καθώς το κείμενο έχει το πλήρες όνομα Philip K. Dick, προσθέτοντας ένα επιπλέον χαρακτήρα μεταξύ των δύο κομματιών της υποψήφιας απάντησης), και η απάντηση Δ) παίρνει 0. Η κανονικοποιημένη βαθμολογία της κάθε απάντησης είναι:

$$c_A = \frac{1}{1+1+0.66+0} = 0.376$$
$$c_B = \frac{1}{1+1+0.66+0} = 0.376$$
$$c_C = \frac{0.66}{1+1+0.66+0} = 0.248$$
$$c_D = \frac{0}{1+1+0.66+0} = 0.$$

Κάθε κριτήριο έχει κάποιες παραμέτρους που μπορούν να ρυθμιστούν:

- Ο αριθμός των επεξεργασμένων κειμένων. Σε αυτή την περίπτωση η βαθμολογία της κάθε απάντησης υπολογίζεται για κάθε ένα από τα top-n κείμενα που επιστρέφονται από την QA μονάδα, και το τελικό σκορ είναι ο μέσος όρος αυτών των τιμών.
- Η χρήση του βάρους w_i των κειμένων που επέστρεψε η μονάδα QA. Σε αυτή την περίπτωση ο μέσος όρος που υπολογίζεται στο προηγούμενο σημείο σταθμίζεται χρησιμοποιώντας το σκορ w_i κάθε κειμένου. Η στρατηγική αυτή επιτρέπει να ορίσουμε υψηλότερα βάρη σε κείμενα που θεωρούνται ως πιο σχετικά με το ερώτημα από τη μονάδα QA.
- Το επίπεδο της γλωσσικής ανάλυσης που πρέπει να υιοθετήσει κανείς για την επεξεργασία των κειμένων που επέστρεψε η μονάδα QA. Τα κείμενα αναπαρίστανται με τη χρήση λέξεων-κλειδιών, λημμάτων, ή μίσχων, με ή χωρίς αφαίρεση εξαιρούμενων λέξεων.
- Η χρήση της επέκτασης ερωτήματος (question expansion). Το σύστημα κάνει τέσσερις διαφορετικές ερωτήσεις που δημιουργούνται με τη συνένωση του αρχικού ερωτήματος με κάθε μία από τις τέσσερις πιθανές υποψήφιες απαντήσεις. Στο τρέχον παράδειγμα μας, ο εικονική παίκτης ρωτάει τη μονάδα QA χρησιμοποιώντας τις ακόλουθες ερωτήσεις: Ποιος σκηνοθέτησε το Blade Runner; Ο Harrison Ford, Ποιος σκηνοθέτησε το Blade Runner; Ο Ridley Scott, Ποιος σκηνοθέτησε το Blade Runner; Ο Philip Dick και Ποιος σκηνοθέτησε το Blade Runner; Ο James Cameron. Αν ένα κριτήριο υιοθετεί την επέκταση ερωτήματος, χρησιμοποιεί τέσσερα διαφορετικά σύνολα κειμένων (ένα για κάθε ερώτηση), αντί να χρησιμοποιεί το ίδιο σύνολο κειμένων.

Εκτός από τα προαναφερθέντα κριτήρια για Βαθμολόγηση Απάντησης, το φίλτρο κατανομής θα μπορούσε επίσης να υιοθετηθεί για τη σύγκριση των απαντήσεων με τα ανακτηθέντα κείμενα. Τελικά οι συγγραφείς αποφάσισαν να αποφύγουν να χρησιμοποιήσουν αυτό το φίλτρο, επειδή, μετά την επιλογή των πιο σχετικών κειμένων για ένα συγκεκριμένο ερώτημα, αναμένουμε ότι η υποψήφια απάντηση ήδη περιέχεται σε ένα από τα κείμενα, και τα φίλτρα που βασίζονται σε αγνά λεξιλογική σύγκριση επαρκούν για να επιλέξουμε σωστά και να βαθμολογήσουμε. Επιπλέον, οι απαντήσεις συνήθως περιέχουν μία ή δύο λέξεις, και τα περισσότερα από αυτά είναι ονόματα, αριθμοί ή ημερομηνίες, που προσδιορίζονται δύσκολα από ένα φίλτρο κατανομής.

Αξίζει να αναφέρουμε πως κατάφεραν οι συγγραφείς να χειριστούν ένα συγκεκριμένο είδος ερωτήσεων, δηλαδή εκείνων που διατυπώθηκαν στην αρνητική μορφή, όπως «Ποιοι από αυτούς τους κακοποιούς δεν είναι εχθρός του Μπάτμαν;». Ας υποθέσουμε ότι οι υποψήφιες απαντήσεις είναι Dottor Destino, Mister Freeze, Pinguino και Τζόκερ. Η μονάδα QA επέστρεψε κείμενα που περιέχουν Τζόκερ, Pinguino και Mister Freeze, αυτά θα ήταν σωστά εάν το ερώτημα είχε τεθεί σε θετική μορφή. Στην πραγματικότητα, όταν τίθεται το ερώτημα σε αρνητική μορφή, τα κορυφαία κείμενα συχνά περιέχουν τις υποψήφιες απαντήσεις που είναι στην πραγματικότητα εκείνα που πρέπει να απορρίψουμε σε αυτή την περίπτωση. Για το λόγο αυτό, όταν το σύστημα εντοπίσει ένα αρνητικό ερώτημα, η στρατηγική για τη βαθμολογία των υποψηφίων απαντήσεων αντιστρέφεται, πράγμα που σημαίνει ότι αυτά με την υψηλότερη βαθμολογία γίνονται τα λιγότερο αληθοφανή και αντίστροφα. Η διαδικασία ανίχνευσης ερωτήσεων που τίθενται σε αρνητική μορφή αξιοποιεί διάφορα ευρετικά που βασίζονται στη χρήση των κανονικών εκφράσεων (regular expressions). Περισσότερες πληροφορίες για την απόδοση της μεθόδου αντίστροφης βαθμολόγησης και τη χρήση των κανονικών εκφράσεων για τη σωστή ανίχνευση των αρνητικών ερωτήσεων αναφέρονται παρακάτω. Κάποια πιο απλή υλοποίηση που χειρίζεται τις αρνητικές ερωτήσεις έχει συμπεριληφθεί και στον κώδικα της παρούσας εργασίας.

7.3 Λήψη Απόφασης

Η μονάδα Λήψης Απόφασης (Decision Making) είναι υπεύθυνη για την απόφαση σχετικά με την απάντηση σε μια συγκεκριμένη ερώτηση, την αποχώρηση από το παιχνίδι ή την χρήση κάποιας από τις διαθέσιμες βοήθειες. Η στρατηγική απόφασης αξιολογεί την αβεβαιότητα των πληροφοριών που παρέχονται από τη μονάδα QA και τη βαθμολόγηση απαντήσεων, τις διαθέσιμες σανίδες σωτηρίας, και το επίπεδο της ερώτησης, προκειμένου να λάβει την τελική απόφαση. Η στρατηγική λήψης αποφάσεων που επινοήθηκε συμπυκνώνει δύο ευρετικά για να διαχειριστεί τις καταστάσεις αβεβαιότητας:

- Η μέγιστη βαθμολογία που υπολογίζεται από την βαθμολόγηση απαντήσεων και για τις τέσσερις υποψήφιες απαντήσεις είναι πολύ χαμηλή. Αυτό σημαίνει ότι είτε η ποιότητα των κειμένων που ανακτώνται από τη μονάδα QA μπορεί να είναι χαμηλή, ως εκ τούτου, αυτά τα κείμενα δεν είναι χρήσιμα για να βρεθεί η σωστή απάντηση στο ερώτημα, ή τα κριτήρια που χρησιμοποιήθηκαν από την βαθμολόγηση απαντήσεων δεν είναι ικανοποιητικά.
- Η διαφορά μεταξύ του σκορ της καλύτερης υποψήφιας απάντησης και της δεύτερης καλύτερης υποψήφιας απάντησης είναι πολύ μικρή. Αυτό σημαίνει ότι ο εικονικός παίκτης δεν είναι σε θέση να παρέχει σαφείς αποδείξεις για την πιο πιθανή υποψήφια απάντηση μεταξύ των δύο.

Σε μια κατάσταση στην οποία ο εικονικός παίκτης έχει αρκετή εμπιστοσύνη σε μία από τις υποψήφιες απαντήσεις, μπορεί να απαντήσει την ερώτηση χωρίς τη χρήση οποιασδήποτε σανίδας σωτηρίας. Σε μια κατάσταση αβεβαιότητας, ο αλγόριθμος λήψης αποφάσεων μπορεί να πάρει μία από τις παρακάτω αποφάσεις:

- να αποσυρθεί από το παιχνίδι,
- να χρησιμοποιήσει μία ή περισσότερες διαθέσιμες σανίδες σωτηρίας,
- να συνεχίσει να παίζει δίνοντας μια τυχαία απάντηση.

Η τυχαία στρατηγική μπορεί να είναι επίσης χρήσιμη σε ορισμένες καταστάσεις, ιδίως σε συγκεκριμένα επίπεδα του παιχνιδιού (6η ή 11η ερώτηση) δηλαδή όταν μια λανθασμένη απάντηση δεν προκαλεί καμία απώλεια στο κερδισμένο ποσό.

Ο αλγόριθμος λήψης αποφάσεων υλοποιεί μια απλή στρατηγική για τη διαχείριση των σανίδων σωτηρίας, πράγμα που σημαίνει ότι η σειρά της χρήσης τους είναι ανεξάρτητη από το επίπεδο της τρέχουσας ερώτησης. Το σύστημα αξιοποιεί πρώτα τη σανίδα σωτηρίας της δημοσκόπησης στο ακροατήριο και, αν δεν επιστρέψει μια υποψήφια απάντηση με μια καλή εμπιστοσύνη, τότε πάμε στο τηλέφωνο σε ένα φίλο, και, τέλος, στο 50:50. Άλλοι συγγραφείς αντιμετωπίζουν το πρόβλημα του προσδιορισμού μιας πιο δυναμικής στρατηγικής λήψης αποφάσεων για το παιχνίδι WWBM. Στο [26], η λήψη αποφάσεων κατασκευάζει ένα δέντρο απόφασης που κωδικοποιεί τις πιθανότητες σε κάθε πιθανή μελλοντική κατάσταση του παιχνιδιού. Το δέντρο αποτελείται από σκέλη (forks) απόφασης για την επιλογή αν πρέπει να απαντήσει στο ερώτημα, για να χρησιμοποιήσει μια σανίδα σωτηρίας, ή για να αποχωρήσει, και από σκέλη πιθανοτήτων για να κωδικοποιήσει την αβεβαιότητα του να απαντήσει σωστά στις ερωτήσεις. Η καλύτερη επιλογή επιτυγχάνεται με τη δράση που μεγιστοποιεί την αναμενόμενη χρησιμότητα. Οι πιθανότητες εκχωρούνται στους κόμβους του δένδρου με βάση τις επιδόσεις κατά το παρελθόν σε δείγμα των ερωτήσεων από το αντίστοιχο επίπεδο δυσκολίας. Μια διαφορετική στρατηγική που βασίζεται στον δυναμικό προγραμματισμό εκδόθηκε [42] για την ανάλυση δύο διαφορετικών στόχων:

- για τη μεγιστοποίηση της αναμενόμενης ανταμοιβής,
- για τη μεγιστοποίηση της πιθανότητας να φτάσουμε σε μια δεδομένη ερώτηση.

Η ανάλυση των αποτελεσμάτων που παρουσιάζονται στην εν λόγω εργασία [1] επέτρεψε να καθοριστεί η σειρά με την οποία η μονάδα λήψης αποφάσεων θα πρέπει να ελέγξει και να χρησιμοποιήσει τις διαθέσιμες βοήθειες. Πιο συγκεκριμένα, ανακαλύφθηκε ότι η Δημοσκόπηση του κοινού πρέπει να ελέγχεται πριν από τις άλλες βοήθειες, ενώ δεν υπάρχουν ειδικές ενδείξεις που να ορίζουν εάν το 50:50 η το τηλέφωνο σε ένα φίλο είναι καλύτερο.

Algorithm 1 Decision making algorithm.

```
1: procedure DECISION MAKING(< q, (c_A, c_B, c_C, c_D) >, lifelines)  ▷ Decision strategy based on the scores of the four candidate answers for question q, and the available lifelines
2:     BestAnswer ← BEST(< q, (c_A, c_B, c_C, c_D) >)
3:     SecondBestAnswer ← SECONDBEST(< q, (c_A, c_B, c_C, c_D) >)  ▷ Selection of the best and second best candidate answers
4:     if BestAnswer.score = 0
       or (BestAnswer.score − SecondBestAnswer.score)
       < (BestAnswer.score ∗ threshold) then
                ▷ Situation of uncertainty: system not confident in the best answer, or undecided between the best and second best candidate answer
5:         if CANUSE(Poll the Audience) then
6:             audienceAnswers ← USE(Poll the Audience)
7:             lifelines ← lifelines − {Poll the Audience}
8:             if audienceAnswers.score > 0 then
9:                 RETURN BEST(audienceAnswers)
10:            end if
11:        end if
12:        if CANUSE(Phone a Friend) then
13:            friendAnswer ← USE(Phone a Friend)
14:            lifelines ← lifelines − {Phone a Friend}
15:            if friendAnswer ≠ null then
16:                RETURN friendAnswer
17:            end if
18:        end if
19:        if (CANUSE(50:50) and CANRISK()) then
20:            50 : 50answers ← USE(50:50)
21:            lifelines ← lifelines − {50:50}
22:            if 50 : 50answers.score > 0 then
23:                RETURN BEST(50:50answers)
24:            else
25:                RETURN RANDOM(50:50answers)
26:            end if
27:        end if
28:        if CANRISK() then
29:            RETURN RANDOM(answers)  ▷ No more lifelines but the player can risk
30:        end if
31:        RETIRE()
32:    else
33:        RETURN BestAnswer
34:    end if
35: end procedure
```

Σχήμα 5: Ο αλγόριθμος λήψης απόφασης

Οι συναρτήσεις που χρησιμοποιούνται από τον αλγόριθμο λήψης απόφασης (Σχήμα 5) είναι οι εξής:

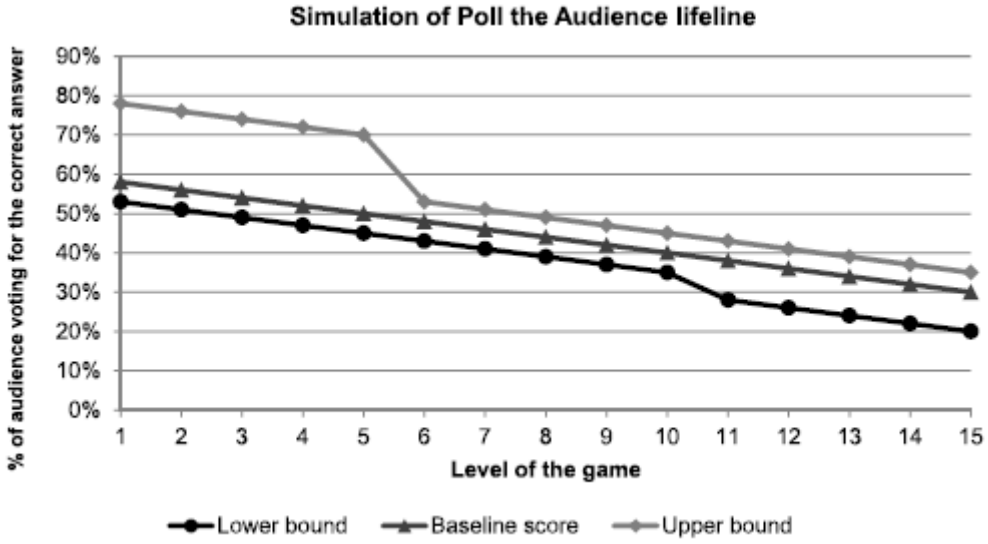
- Best(Answers) και SecondBest(Answers): επιστρέφουν τη καλύτερη και τη δεύτερη καλύτερη υποψήφια απάντηση σε ένα ερώτημα q, αντίστοιχα.
- CanUse(Lifeline): επιστρέφει true αν η συγκεκριμένη σανίδα σωτηρίας δεν έχει ακόμη χρησιμοποιηθεί στο σημερινό παιχνίδι, αλλιώς επιστρέφει false.
- Use(Lifeline): επιστρέφει ένα νέο σύνολο απαντήσεων μαζί με τις βαθμολογίες τους όπως λαμβάνονται μετά την χρήση της σανίδας σωτηρίας.
- CanRisk(): επιστρέφει true αν η τρέχουσα ερώτηση επιτρέπει στον παίκτη να παρέχει μια λανθασμένη απάντηση, χωρίς να χάσει τα κερδισμένα χρήματα (6η ή 11η ερώτηση), αλλιώς επιστρέφει false.

- Random(Answers): επιτρέπει στον παίκτη να παρέχει μια τυχαία απάντηση σε μια ερώτηση.
- Retire(): επιτρέπει στον εικονικό παίκτη να αποχωρήσει από το παιχνίδι και να πάρει τα κερδισμένα χρήματα.

Ο εικονικός παίκτης δεν χρησιμοποιεί καμία βοήθεια αν έχει αρκετή εμπιστοσύνη σε μία από τις τέσσερις απαντήσεις (if δήλωση στο βήμα 4). Στην περίπτωση αυτή η απάντηση που δίνεται είναι αυτή με την υψηλότερη βαθμολογία (βήμα 33). Εάν ο εικονικός παίκτης είναι σε κατάσταση αβεβαιότητας, διερευνά τις διαθέσιμες βοήθειες ξεκινώντας από βοήθεια του Κοινού (βήματα 5-11). Εάν ο παίκτης έχει αρκετή εμπιστοσύνη στην απάντηση που προβλέπεται από την εν λόγω σανίδα σωτηρίας (βήμα 8), επιστρέφει αυτή την απάντηση (βήμα 9), διαφορετικά συνεχίζει να εξερευνά τις άλλες διαθέσιμες βοήθειες. Τα βήματα 12-18 διαχειρίζονται την βοήθεια του τηλεφώνου σε φίλο, το οποίο επιτρέπει να επιστρέψει την απάντηση που ενδεχομένως να έδινε ένας φίλος (βήμα 16). Η χρήση της βοήθειας 50:50 είναι σχετική με το επίπεδο του παιχνιδιού που επιτεύχθηκε από τον παίκτη. Αν ο χρήστης μπορεί να διακινδυνεύσει, πράγμα που σημαίνει ότι τα χρήματα δεν θα χαθούν ακόμη και παρέχοντας μια λάθος απάντηση στο ερώτημα (βήμα 19), ο παίκτης χρησιμοποιεί το 50:50 (βήμα 20). Ο παίκτης επιλέγει την απάντηση με την υψηλότερη βαθμολογία (βήμα 23), αν έχει αρκετή εμπιστοσύνη στην απάντηση, διαφορετικά επιστρέφει τυχαία μία από τις δύο υπόλοιπες απαντήσεις (βήμα 25). Εάν ο παίκτης δεν μπορεί να επικαλεστεί τις βοήθειες ή δεν έχει αρκετή εμπιστοσύνη στις απαντήσεις που δόθηκαν από τις βοήθειες, τότε επιστρέφει μια τυχαία επιλεγμένη απάντηση σε περίπτωση που ο χρήστης μπορεί να διακινδυνεύσει (βήματα 28-30), αλλιώς αποσύρεται από το παιχνίδι (βήμα 31).

Δεδομένου ότι το παιχνίδι παίζεται με μια παραλλαγή από το επιτραπέζιο παιχνίδι, οι συγγραφείς του άρθρου [1] εφάρμοσαν μια συγκεκριμένη στρατηγική για την προσομοίωση της πραγματικής χρήσης των βοηθειών, καθώς η χρήση τους στο επιτραπέζιο παιχνίδι συνεπάγεται την αλληλεπίδραση με τους άλλους παίκτες. Το 50:50 προσομοιώνεται με τον ίδιο τρόπο όπως και στο πραγματικό παιχνίδι, δηλαδή με απομάκρυνση δύο λανθασμένων απαντήσεων μεταξύ των τεσσάρων υποψήφιων απαντήσεων. Όπως και στο πραγματικό παιχνίδι, το Τηλέφωνο σε ένα φίλο και η βοήθεια του κοινού δεν είναι πάντα σε θέση να επιστρέψουν τη σωστή απάντηση. Όσον αφορά το τηλέφωνο αξίζει να σημειωθεί ότι συνήθως όσο υψηλότερο είναι το επίπεδο του παιχνιδιού, τόσο πιο δύσκολο είναι να απαντήσει στην ερώτηση. Γι' αυτό, αυτή η σανίδα σωτηρίας λειτουργεί ως εξής: επιστρέφει πάντα τη σωστή απάντηση, όταν χρησιμοποιείται για τα επίπεδα από 1 έως 5, επιλέγει τυχαία ανάμεσα σε δύο εναλλακτικές λύσεις, δηλαδή παρέχοντας τη σωστή απάντηση ή καθόλου απάντηση, όταν χρησιμοποιείται για τα επίπεδα από 6 έως 10, επιλέγει τυχαία ανάμεσα σε τρεις εναλλακτικές λύσεις, δηλαδή παρέχοντας τη σωστή απάντηση, να μην επιστρέψει καμία απάντηση ή επιστρέφοντας μια λάθος απάντηση, όταν χρησιμοποιείται για επίπεδα από 11 έως 15. Από την άλλη πλευρά, η βοήθεια του κοινού προσομοιώνεται με σκοπό να κατανέμει τις ψήφους που προέρχονται από το κοινό (σε ποσοστό) μεταξύ των υποψηφίων απαντήσεων. Χωρίς ένα πραγματικό ακροατήριο, προσομοίωσαν την κατανομή των ψήφων χρησιμοποιώντας μια στρατηγική που να λαμβάνει υπόψη τόσο το τρέχον επίπεδο της ερώτησης και ένα βαθμό τυχαιότητας. Πρώτα εκχωρούν ένα ποσοστό του συνόλου των ψήφων στη σωστή απάντηση, που αντιπροσωπεύουν το ποσοστό του κοινού που θα γνωρίζουν τη σωστή απάντηση, μετά διανέμουνε τυχαία τις υπόλοιπες ψήφους μεταξύ των άλλων υποψήφιων απαντήσεων. Το ποσοστό των ψήφων που θα αναθέσουν στην σωστή απάντηση - αρχική τιμή - είναι αντιστρόφως ανάλογο προς το επίπεδο του παιχνιδιού, δηλαδή όσο πιο δύσκολο το ερώτημα, τόσο χαμηλότερη είναι η εμπιστοσύνη. Το βασικό ποσοστό των ψήφων κατανέμεται τυχαία ανάλογα με το επίπεδο του παιχνιδιού, μεταξύ του κάτω και άνω ορίου όπως

απεικονίζεται στο Σχήμα 6. Αξίζει να σημειωθεί ότι η διατάραξη (perturbation) του ποσοστού της γραμμής βάσης είναι διαφορετική για ερωτήσεις των οποίων το επίπεδο είναι μεταξύ 1 και 5, 6 και 10, 11 και 15. Για παράδειγμα, εάν ο παίκτης χρησιμοποιεί τη βοήθεια του κοινού στο επίπεδο 3, η βασική γραμμή έχει ποσοστό ίσο με 54%, και η τυχαία ενόχληση κυμαίνεται μεταξύ 49% και 74%. Οι υπόλοιπες ψήφοι (μεταξύ 51% και 26%) διανέμονται τυχαία μεταξύ των υπόλοιπων απαντήσεων. Εάν η σανίδα σωτηρίας χρησιμοποιείται σε επίπεδο 14, το βασικό ποσοστό είναι 32% και έχει τυχαία ενόχλησή μεταξύ 22% και 37%, και οι υπόλοιπες ψήφοι (μεταξύ 78% και 63%) διανέμονται τυχαία μεταξύ των υπόλοιπων απαντήσεων.



Σχήμα 6: Το ποσοστό του κοινού που απαντάει σωστά στην ερώτηση όταν χρησιμοποιείται η βοήθεια του κοινού.

Στην παρούσα υλοποίηση ακολουθήθηκε άλλη αναπαράσταση των σανίδων σωτηρίας η οποία θα αναλυθεί παρακάτω.

8. ΥΛΟΠΟΙΗΣΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

Η υλοποίηση του συστήματος μπορεί να χωριστεί σε 3 διακριτά κομμάτια:

- Κατασκευή συνόλου ερωτήσεων
- Υλοποίηση γραφικού περιβάλλοντος
- Υλοποίηση λειτουργικότητας για τον εικονικό παίκτη

Παρακάτω θα παρουσιαστούν αναλυτικά κάποιες τεχνικές λεπτομέρειες της υλοποίησης όπως ποια γλώσσα προγραμματισμού, εργαλεία, βιβλιοθήκες κτλ. χρησιμοποιήθηκαν. Επίσης θα γίνει μια ανάλυση των παραπάνω διακριτών κομματιών με παράλληλη παράθεση και σχολιασμό του κώδικά τους.

8.1 Τεχνικές λεπτομέρειες

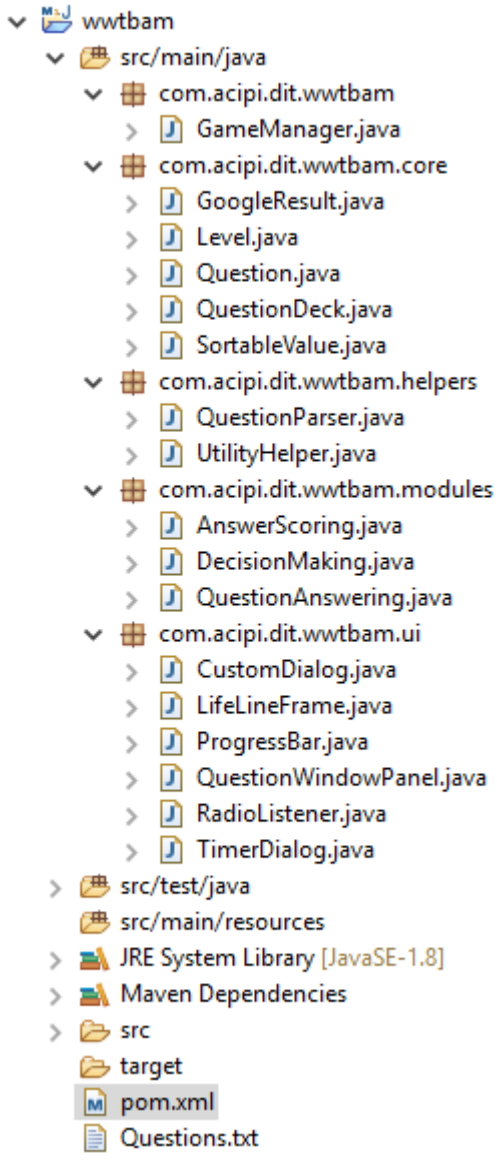
Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την υλοποίηση το συστήματος είναι η Java (έκδοση 8). Πιο συγκεκριμένα για την υλοποίηση του γραφικού περιβάλλοντος του παιχνιδιού χρησιμοποίησα το Java Swing που είναι μια βιβλιοθήκη στην Java με την οποία μπορείς να φτιάχνεις παράθυρα, panel, κουμπιά κτλ. Για την υλοποίηση του συστήματος χρησιμοποιήθηκε το Eclipse IDE Oxygen. Τα απαραίτητα plugins που πρέπει να είναι εγκατεστημένα στον Eclipse είναι τα εξής: Eclipse Maven Plugin (m2e) (http://www.eclipse.org/m2e-wtp). Αφού εγκαταστάθηκε το Eclipse και μπήκαν τα plugins το επόμενο βήμα ήταν να φτιαχτεί ένας κενός χώρος εργασίας (workspace) και ένα καινούργιο Java project το οποίο ονομάστηκε “wwtbam” (από τα αρχικά του Who Wants To Be A Millionaire). Το project αυτό είναι ένα Maven project. Το Maven είναι ένα σύστημα της Apache Software Foundation το οποίο βοηθάει στο χτίσιμο ενός Java project (διαχείριση βιβλιοθηκών, εκδόσεων κτλ.). Για περισσότερες πληροφορίες μπορείτε να μπείτε εδώ https://maven.apache.org/ . Το configuration του Maven γίνεται στο pom.xml. Για την ονομασία των packages χρησιμοποιείται στην αρχή το “com.acipi.dit.wwtbam”. Για την υλοποίηση χρειάστηκαν δύο βιβλιοθήκες, το jsoup και το commons-lang. Η πρώτη χρησιμοποιείται στο κομμάτι του QA ώστε να επικοινωνήσει το πρόγραμμα με την μηχανή αναζήτησης της Google και να διαβάσει τις σελίδες html που αυτή επιστρέφει. Η δεύτερη χρησιμοποιείται για να γίνει καλύτερη διαχείριση κάποιων συμβολοσειρών αφού αυτή προσφέρει κάποιες πολύ χρήσιμες συναρτήσεις όπως θα φανεί και παρακάτω.

8.2 Ανάλυση κώδικα

Έγινε μια προσπάθεια να χωριστεί η υλοποίηση σε διάφορα κομμάτια γι’ αυτό και φτιάχτηκαν πολλά packages όπως φαίνεται και στην Εικόνα 1. Έτσι φτιάχτηκαν τα εξής packages:

- **ui:** Εδώ βρίσκονται οι κλάσεις που υλοποιούν το γραφικό περιβάλλον του παιχνιδιού. Αυτές οι κλάσεις είναι οι: CustomDialog, LifeLineFrame, ProgressBar, QuestionWindowPanel, RadioListener και TimerDialog.
- **modules:** Εδώ βρίσκονται οι κλάσεις που υλοποιούν τις τρείς λειτουργικές μονάδες του εικονικού παίκτη όπως αυτές παρουσιάστηκαν στο αντίστοιχο κεφάλαιο παραπάνω. Αυτές οι κλάσεις είναι οι: AnswerScoring, DecisionMaking και QuestionAnswering.

- **helpers:** Εδώ βρίσκονται κάποιες κλάσεις που παρέχουν κάποιες βοηθητικές λειτουργίες στο σύστημα. Αυτές οι κλάσεις είναι οι: QuestionParser και UtilityHelper.
- **core:** Εδώ βρίσκονται οι κλάσεις που αποτελούν τους δομικούς λίθους της εφαρμογής. Είναι κυρίως κλάσεις που έχουν μόνο πεδία και setters / getters και χρησιμοποιούνται για την αναπαράσταση της πληροφορίας. Αυτές οι κλάσεις είναι οι: GoogleResult, Level, Question, QuestionDeck και SortableValue.
- **<base package>:** Εδώ βρίσκεται η βασική κλάση της εφαρμογής και αυτή που περιέχει την main συνάρτηση και την οποία τρέχουμε για να ξεκινήσουμε την εφαρμογή μας. Αυτή η κλάση αντιπροσωπεύει και την μονάδα του Διαχειριστή του Παιχνιδιού όπως περιγράφεται σε παραπάνω κεφάλαιο και είναι η GameManager.



Εικόνα 1: Η δομή του Java project όπως φαίνεται στο Eclipse IDE.

8.2.1 Κατασκευή του συνόλου ερωτήσεων

Το πρώτο βήμα για να υλοποιηθεί το παιχνίδι αυτό ήταν να φτιαχτεί το σύνολο των ερωτήσεων τις οποίες καλείται να απαντήσει ο παίκτης. Μιας και σκοπός ήταν να υλοποιηθεί η ελληνική έκδοση του παιχνιδιού οι ερωτήσεις αυτές οφείλουν να είναι στα ελληνικά. Μια επιλογή θα ήταν να πάρει κανείς τις ερωτήσεις από το αντίστοιχο επιτραπέζιο παιχνίδι και να τις γράψει με το χέρι σε ένα αρχείο. Αυτό όμως θα απαιτούσε ώρες πολλές και θα ήταν πολύ κουραστικό. Η άλλη επιλογή ήταν να βρεθούν έτοιμες οι ερωτήσεις στο διαδίκτυο. Μετά από πολύ ψάξιμο βρέθηκε σε αυτόν τον σύνδεσμο (https://tzampare.wordpress.com/2010/03/08/download-%CF%80%CE%BF%CE%B9%CE%BF%CF%82-%CE%B8%CE%B5%CE%BB%CE%B5%CE%B9-%CE%BD%CE%B1-%CE%B3%CE%B9%CE%BD%CE%B5%CE%B9-%CE%B5%CE%BA%CE%B1%CF%84%CE%BF%CE%BC%CE%BC%CF%85%CF%81%CE%B9%CE%BF%CF%85%CF%87%CE%BF/) η υλοποίηση της ελληνικής έκδοσης του παιχνιδιού. Η υλοποίηση αυτή είναι σε μορφή Shockwave Flash. Ο προγραμματιστής που το έφτιαξε παρέχει σε έναν φάκελο «library» 1001 .txt αρχεία τα οποία περιέχουν το καθένα τις 15 ερωτήσεις αυξημένης δυσκολίας που χρησιμοποιούνται σε μια εκτέλεση του παιχνιδιού. Αυτά τα αρχεία είναι από το qfile634.txt μέχρι το qfile1634.txt. Επειδή όμως οι πληροφορίες για την κάθε ερώτηση που βρίσκονται σε αυτά τα αρχεία είναι με την μορφή που την χρησιμοποιεί το πρόγραμμα που βρέθηκε, φτιάχτηκε ένας parser ο οποίος διαβάζει ένα-ένα αυτά τα αρχεία, εξάγει τις χρήσιμες πληροφορίες και στην συνέχεια τις γράφει στο αρχείο Questions.txt (όπως φαίνεται στην Εικόνα 1) σε συγκεκριμένη μορφή που μπορεί στην συνέχεια να διαβάσει το πρόγραμμα. Συγκεκριμένα η μορφή που έχουν οι πληροφορίες για την κάθε ερώτηση του παιχνιδιού στο αρχείο Questions.txt είναι η εξής (όπως φαίνεται και στην Εικόνα 2):

<Αριθμός του επιπέδου της ερώτησης> (από το 1 μέχρι το 15)

<Κείμενο της ερώτησης>

<Απάντηση Α>

<Απάντηση Β>

<Απάντηση Γ>

<Απάντηση Δ>

<Αριθμός της σωστής απάντησης> (1 για Α, 2 για Β, 3 για Γ και 4 για Δ)

<Ποσοστό % που παίρνει η απάντηση Α στην βοήθεια του κοινού>

<Ποσοστό % που παίρνει η απάντηση Β στην βοήθεια του κοινού>

<Ποσοστό % που παίρνει η απάντηση Γ στην βοήθεια του κοινού>

<Ποσοστό % που παίρνει η απάντηση Δ στην βοήθεια του κοινού>

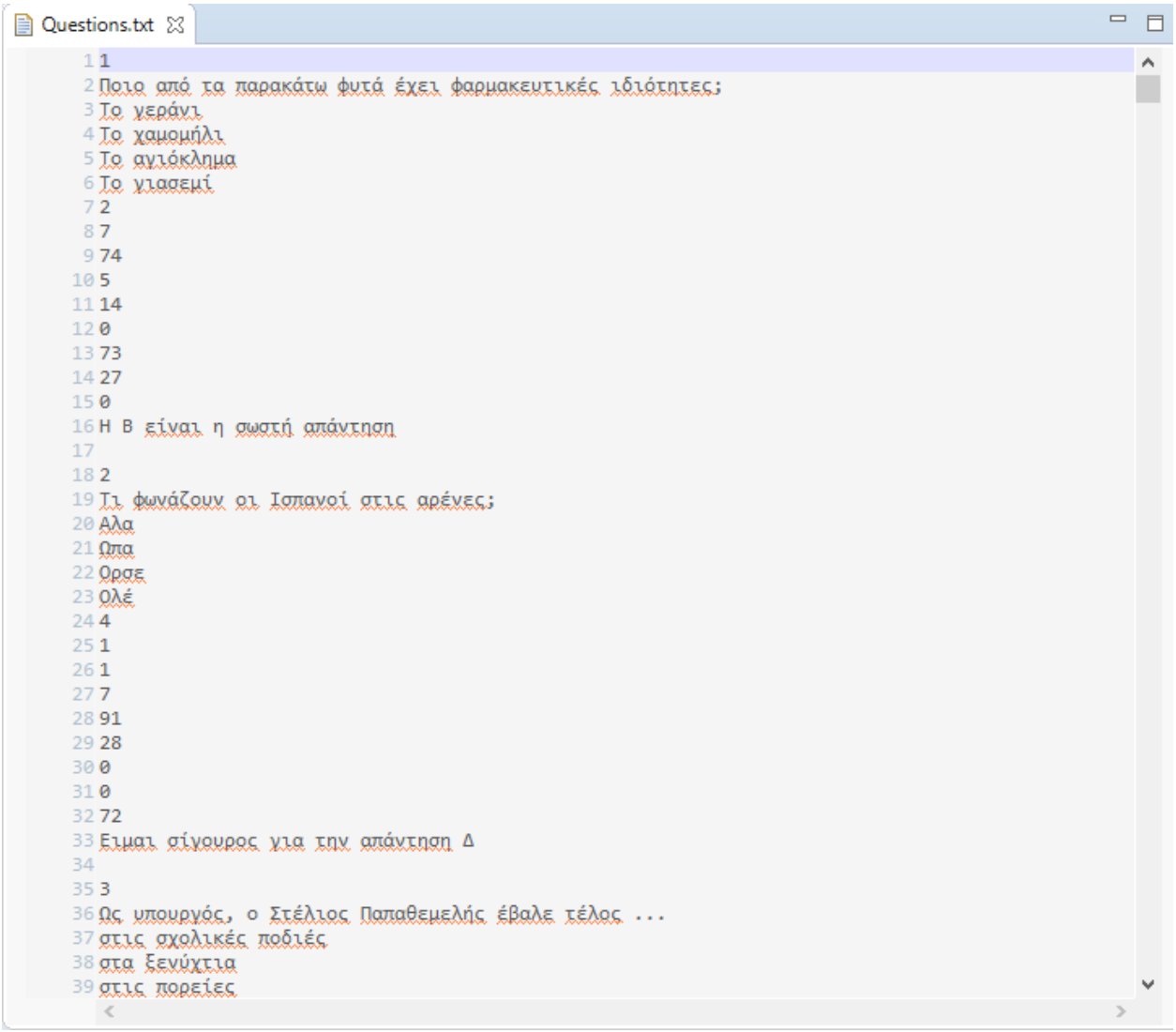
<Ποσοστό % που παίρνει η απάντηση Α στην βοήθεια 50/50>

<Ποσοστό % που παίρνει η απάντηση Β στην βοήθεια 50/50>

<Ποσοστό % που παίρνει η απάντηση Γ στην βοήθεια 50/50>

<Ποσοστό % που παίρνει η απάντηση Δ στην βοήθεια 50/50>

<Η απάντηση που δίνει ο φίλος του παίκτη στην βοήθεια του τηλεφώνου>



Εικόνα 2: Το αρχείο Questions.txt που περιέχει όλες τις ερωτήσεις του παιχνιδιού.

Εφόσον το κάθε αρχείο περιέχει 15 ερωτήσεις συνολικά το αρχείο Questions.txt θα περιέχει 15*1001 = 15015 ερωτήσεις. Ενδέχεται κάποιες από αυτές τις ερωτήσεις να μην διαβάζονται καθώς υπάρχει πρόβλημα με την κωδικοποίηση (encoding) στα ελληνικά. Για να μετατραπούν οι ερωτήσεις από την μορφή που βρίσκονταν στα αρχεία μέσα στον φάκελο “library” στην παραπάνω μορφή φτιάχτηκε ένας parser (QuestionParser.java) ο οποίος τρέχει αυτόνομα από το υπόλοιπο πρόγραμμα (έχει δικιά του main συνάρτηση) και αυτό που κάνει είναι να φτιάχνει ένα άδειο αρχείο Questions.txt και στην συνέχεια να διαβάζει ένα-ένα τα αρχεία qfileXXX.txt και να γεμίζει με ερωτήσεις το Questions.txt. Ο parser αυτός θα τρέξει μόνο μια φορά στην αρχή και το αρχείο Questions.txt ΔΕΝ πρέπει να υπάρχει. Εφόσον τρέξει ο parser και δημιουργηθεί το αρχείο Question.txt στην συνέχεια μπορεί να εκτελεστεί το πρόγραμμά όσες φορές επιθυμεί κάποιος. Πιο συγκεκριμένα:

QuestionParser.java

Δημιουργεί το αρχείο με τις ερωτήσεις του παιχνιδιού.

LIBRARY_LOCATION: Το μονοπάτι στο οποίο έχουμε βάλει τον φάκελο “library”. Στην δική μας περίπτωση είναι στο Desktop.

***read()*:** Παίρνει σαν όρισμα τον αύξοντα αριθμό (fileNum) του qfile που πρόκειται να επεξεργαστεί. Ανοίγει το αρχείο qfile(fileNum).txt με κωδικοποίηση Windows-1253 για να μπορέσει να διαβάσει τους ελληνικούς χαρακτήρες. Γράφει στο αρχείο Question.txt με κωδικοποίηση UTF-8, προσθέτοντας στο τέλος του αρχείου ενώ εάν το αρχείο δεν υπάρχει το δημιουργεί. Διαβάζει την μία και μοναδική γραμμή του αρχείου qfile και εξάγει όλες τις πληροφορίες που αναφέρονται παραπάνω βρίσκοντας τα αντίστοιχα πεδία μέσα στο κείμενο και αντικαθιστώντας διάφορους χαρακτήρες με τους σωστούς (όπως το '+' με το κενό, το '%22' με τα διπλά εισαγωγικά και το '%27' με την απόστροφο). Γράφει όλες τις πληροφορίες με την μορφή που αναφέραμε προηγουμένως και βάζει μια κενή γραμμή. Τέλος κλείνει τα αρχεία που άνοιξε για διάβασμα και γράψιμο.

***main()*:** Καλεί την συνάρτηση read() για καθένα από τα 1001 αρχεία qfileXXX.txt.

Ο κώδικας μπορεί να βρεθεί και στο Παράρτημα 1.

8.2.2 Υλοποίηση του γραφικού περιβάλλοντος

Όπως προαναφέρθηκε για να υλοποιηθεί ένα γραφικό περιβάλλον για το παιχνίδι, δηλαδή ένα περιβάλλον όπου ο ανθρώπινος παίκτης θα βλέπει την ερώτηση και τις 4 πιθανές απαντήσεις, θα διαλέγει την σωστή, θα μπορεί να χρησιμοποιεί τις βοήθειες και θα βλέπει ανά πάσα στιγμή σε ποιο επίπεδο είναι και πόσα χρήματα κερδίζει, χρησιμοποιήθηκε η βιβλιοθήκη της Java, Java Swing. Αυτή η βιβλιοθήκη δίνει την δυνατότητα στον προγραμματιστή να φτιάξει παράθυρα στα οποία μέσα μπορεί να βάλει κείμενο, κουμπιά κτλ. με τα οποία το πρόγραμμα μπορεί να αλληλοεπιδράσει με τον χρήστη. Στο κεφάλαιο 10 της ενδεικτικής εκτέλεσης του προγράμματος φαίνεται το γραφικό περιβάλλον του παιχνιδιού.

Η γενική ιδέα είναι ότι στην αρχή φορτώνονται στην μνήμη του προγράμματος όλες οι ερωτήσεις του παιχνιδιού από το αρχείο Questions.txt. Στην συνέχεια, ανοίγει ένα παράθυρο στο οποίο μπορείς να διαλέξεις κάποιες ρυθμίσεις για το παιχνίδι οι οποίες αφορούν τον εικονικό παίκτη εάν χρησιμοποιηθεί (το παιχνίδι έχει δυο εναλλακτικούς τρόπους με τους οποίους μπορεί να παιχθεί, είτε να το παίξει ένας ανθρώπινος παίκτης είτε να το παίξει ο εικονικός παίκτης). Σε αυτό το παράθυρο έχει το κουμπί «Ξεκινήστε» που όταν το πατήσεις ξεκινάει το παιχνίδι και το κουμπί «Οδηγίες» που όταν το πατήσεις ανοίγει ένα άλλο παράθυρο με τις οδηγίες χρήσης του παιχνιδιού. Όταν ο παίκτης πατήσει το κουμπί «Ξεκινήστε» τότε εμφανίζεται στο πάνω αριστερό μέρος της οθόνης ένα μικρό παράθυρο που περιέχει 3 κουμπιά, ένα για κάθε μια από τις 3 βοήθειες του παιχνιδιού («50/50», «Τηλέφωνο», «Κοινό»). Επίσης το κεντρικό παράθυρο αλλάζει και πλέον εμφανίζεται στο πάνω μέρος η ερώτηση που πρέπει να απαντήσει ο χρήστης και δίπλα της οι 4 πιθανές απαντήσεις Α), Β), Γ) και Δ). Κάτω δεξιά στο παράθυρο εμφανίζονται σε αύξουσα σειρά από κάτω προς τα πάνω τα 15 επίπεδα του παιχνιδιού και το χρηματικό ποσό που αντιστοιχεί στο κάθε επίπεδο. Τα επίπεδα στα οποία υπάρχει το λεγόμενο «μαξιλαράκι», δηλαδή το συγκεκριμένο ποσό διασφαλίζεται ότι θα δοθεί στον παίκτη ακόμα και αν απαντήσει λάθος σε κάποια ερώτηση, χρωματίζονται με κόκκινο χρώμα, και είναι τα επίπεδα 5, 10 και 15. Δίπλα στα επίπεδα υπάρχει μια μπάρα προόδου (progress bar) η οποία ανεβαίνει κατά ένα επίπεδο κάθε φορά που ο παίκτης απαντάει σωστά μια ερώτηση. Έτσι μπορεί να δει ανά πάσα στιγμή σε ποιο επίπεδο του παιχνιδιού βρίσκεται. Κάτω από την ερώτηση υπάρχουν δυο κουμπιά, το «Κατοχυρώστε» και το «Παραιτηθείτε». Με το πρώτο ο παίκτης κατοχυρώνει την απάντηση που διάλεξε παραπάνω ενώ με το δεύτερο αποχωρεί από το παιχνίδι παίρνοντας τα χρήματα που αντιστοιχούν στο επίπεδο το οποίο έχει φτάσει. Όταν ο χρήστης πατάει το κουμπί «Κατοχυρώστε» του εμφανίζεται

ένα μήνυμα επιβεβαίωσης που τον ρωτάει εάν είναι σίγουρος για την επιλογή του. Το μήνυμα αυτό όπως και όλα τα άλλα μηνύματα επιβεβαίωσης έχουν τις επιλογές «ΝΑΙ» και «ΟΧΙ» για το εάν ο χρήστης είναι σίγουρος για αυτό που πάτησε ή εάν άλλαξε τελικά γνώμη. Εάν ο χρήστης πατήσει «ΝΑΙ» και η απάντηση που διάλεξε είναι η σωστή τότε προχωράει στην επόμενη ερώτηση και η μπάρα προόδου ανεβαίνει στο επόμενο επίπεδο, ενώ εάν η απάντηση που διάλεξε είναι λανθασμένη τότε του εμφανίζεται ένα μήνυμα ότι αποχωρεί από το παιχνίδι χαμένος παίρνοντας τα χρήματα που έχει το εκάστοτε μαξιλάρι που έχει κατοχυρώσει (δηλαδή εάν ο χρήστης απαντήσει λάθος σε μια από τις 5 πρώτες ερωτήσεις τότε θα πάρει 0 ευρώ) και η εφαρμογή εκεί τερματίζει. Εάν ο χρήστης πατήσει «ΟΧΙ» τότε δεν γίνεται τίποτα και το σύστημα περιμένει ο χρήστης να διαλέξει κάποια άλλη ερώτηση ή να σιγουρευτεί για αυτήν που έχει ήδη διαλέξει. Εάν ο χρήστης ξεχάσει να διαλέξει κάποια απάντηση τότε αυτόματα διαλέγεται η απάντηση Α). Επίσης όταν εμφανίζεται η επόμενη ερώτηση μένει διαλεγμένη η απάντηση από το προηγούμενο επίπεδο και πρέπει ο χρήστης να σιγουρευτεί ότι θα διαλέξει μια απάντηση ακόμα και αν πρόκειται για την ίδια. Όταν ο χρήστης πατάει το κουμπί «Παραιτηθείτε» του εμφανίζεται μήνυμα επιβεβαίωσης για το εάν είναι σίγουρος ότι θέλει να αποχωρήσει από το παιχνίδι με το συγκεκριμένο ποσό. Εάν πατήσει «ΝΑΙ» τότε του εμφανίζεται μήνυμα ότι αποχωρεί από το παιχνίδι κερδισμένος παίρνοντας τα χρήματα που αντιστοιχούν στο επίπεδο που έχει φτάσει, αλλιώς εάν πατήσει «ΟΧΙ» τότε συνεχίζει να παίζει.

Όταν ο χρήστης πατήσει το κουμπί της βοήθειας «50/50» τότε εμφανίζεται μήνυμα που ενημερώνει τον χρήστη για τις δύο ερωτήσεις που είναι λάθος. Στην συνέχεια ο χρήστης πρέπει να διαλέξει ανάμεσα στις άλλες δύο ερωτήσεις που απομένουν. Όταν ο χρήστης πατήσει το κουμπί της βοήθειας «Τηλέφωνο» τότε εμφανίζεται ένα μήνυμα με την δήθεν συνομιλία που θα μπορούσε να έχει ο παίκτης με τον φίλο του. Σε αυτό το κείμενο της συνομιλίας στο τέλος εμπεριέχεται και η απάντηση που του έδωσε ο φίλος του για το ποια νομίζει αυτός πως είναι η σωστή απάντηση. Στην συνέχεια ο χρήστης πρέπει να διαλέξει χειροκίνητα αυτή την απάντηση που το πρότεινε ο φίλος του εάν και εφόσον συμφωνεί. Όταν ο χρήστης πατήσει το κουμπί της βοήθειας «Κοινό» τότε εμφανίζεται μήνυμα που του δείχνει τα αποτελέσματα της δήθεν ψηφοφορίας του κοινού. Συγκεκριμένα του έχει σε φθίνουσα σειρά το ποσοστό που πήρε η κάθε απάντηση. Στην συνέχεια ο χρήστης πρέπει χειροκίνητα να διαλέξει την απάντηση που έχει το μεγαλύτερο ποσοστό εάν και εφόσον συμφωνεί με το κοινό.

Για να φτιαχτεί όλο αυτό το γραφικό περιβάλλον εκτός από τις κλάσεις που υλοποιούν τα παράθυρα, τα κουμπιά τα μηνύματα κτλ. χρειάστηκε να φτιαχτούν και κάποιες κλάσεις που υλοποιούν τις δομές δεδομένων που χρειάζονται για να αναπαρασταθεί και να γίνει η διαχείριση για όλη την πληροφορία του παιχνιδιού. Αυτές οι κλάσεις θα παρουσιαστούν και θα αναλυθούν παρακάτω. Δεν θα γίνει ανάλυση σε αυτό το υποκεφάλαιο τυχόν κώδικα που περιέχουν αυτές οι κλάσεις ο οποίος χρησιμοποιείται στην λειτουργικότητα του εικονικού παίκτη. Δηλαδή ο κώδικας που θα παρουσιαστεί εδώ είναι καθαρά κώδικας που χρειάζεται το παιχνίδι για να το παίξει ένας ανθρώπινος παίκτης. Στο επόμενο υποκεφάλαιο που θα αναλυθεί ο εικονικός παίκτης θα ξανά αναφερθούν οι συγκεκριμένες κλάσεις αλλά αυτή την φορά για να σχολιαστεί ο κώδικας του εικονικού παίκτη. Πιο συγκεκριμένα οι κλάσεις είναι οι εξής:

UtilityHelper.java

Περιέχει διάφορες βοηθητικές συναρτήσεις γενικού περιεχομένου.

***numberToLetter()*:** Παίρνει σαν όρισμα έναν αριθμό από το 0 μέχρι το 3 και επιστρέφει το γράμμα της απάντησης που αντιστοιχεί σε αυτόν τον αριθμό. Το 0 αντιστοιχεί στο Α, το 1 στο Β, το 2 στο Γ και το 3 στο Δ. Εάν πάρει κάποιον άλλον αριθμό επιστρέφει Α. Αυτή η συνάρτηση βοηθάει συνήθως στα μηνύματα που εμφανίζονται στον χρήστη γιατί

μέσα στο πρόγραμμα για την κάθε απάντηση κρατάμε την θέση της που βρίσκεται στον πίνακα με τις απαντήσεις (οι πίνακες και οι λίστες ξεκινάνε από το 0), ενώ στον χρήστη δεν πρέπει να δώσουμε τον αριθμό αλλά το γράμμα της απάντησης.

Level.java

Συμβολίζει το επίπεδο του παιχνιδιού.

***MINIMUM*:** Το ελάχιστο επίπεδο του παιχνιδιού που είναι το 1.

***MAXIMUM*:** Το μέγιστο επίπεδο του παιχνιδιού που είναι το 15.

***levels*:** Ένας πίνακας με όλα τα ποσά για κάθε επίπεδο του παιχνιδιού.

***moneyAtLevel()*:** Παίρνει σαν όρισμα ένα επίπεδο του παιχνιδιού και αφού ελέγξει εάν είναι μέσα στα επιτρεπτά όρια επιστρέφει το ποσό που αντιστοιχεί σε αυτό το επίπεδο. Εάν δεν είναι μέσα στα όρια επιστρέφει 0.

moneyAtSafetyLevel(): Παίρνει σαν όρισμα το τρέχον επίπεδο του παιχνιδιού και αφού ελέγξει εάν είναι μέσα στα επιτρεπτά όρια βρίσκει ποιο είναι το επίπεδο ασφαλείας («μαξιλαράκι») που του αντιστοιχεί. Για τα επίπεδα από 1 μέχρι 4 το επίπεδο ασφάλειας είναι το 0, για τα επίπεδα 5 μέχρι 9 το επίπεδο ασφάλειας είναι το 5, για τα επίπεδα 10 μέχρι 14 το επίπεδο ασφάλειας είναι το 10 ενώ για το επίπεδο 15 το επίπεδο ασφάλειας είναι το 15. Στην συνέχεια επιστρέφει το ποσό που αντιστοιχεί στο κάθε επίπεδο ασφάλειας (στο επίπεδο ασφάλειας 0 το ποσό που αντιστοιχεί είναι 0 ευρώ). Η συνάρτηση αυτή καλείται όταν ο παίκτης δώσει λάθος απάντηση και πρέπει να του εμφανιστεί μήνυμα με το ποσό που έχει κερδίσει.

SortableValue.java

Δομή δεδομένων που βοηθάει στην ταξινόμηση των τιμών σε μια λίστα αλλά που θέλουμε να κρατήσουμε και την αρχική θέση της τιμής στην λίστα αυτή πριν γίνει η ταξινόμηση. Χρησιμοποιείται στην ταξινόμηση των ποσοστών των ψήφων του κοινού, των ποσοστών του 50/50 και των βαθμολογιών που επιστρέφονται από την μονάδα AS Κάνει implement το interface Comparable.

***value*:** Η τιμή που θέλουμε να βάλουμε μέσα στην λίστα.

***index*:** Η αρχική θέση της τιμής μέσα στην λίστα.

***getValue()*:** Επιστρέφει το value.

***setValue()*:** Δίνει τιμή στο value.

***getIndex()*:** Επιστρέφει το index.

***setIndex()*:** Δίνει τιμή στο index.

***compareTo()*:** Κάνει την ταξινόμηση σε φθίνουσα σειρά. Συγκρίνει δύο αντικείμενα τύπου SortableValue μεταξύ τους και επιστρέφει θετικό, 0 ή αρνητικό αριθμό.

Question.java

Αντιπροσωπεύει μια ερώτηση του παιχνιδιού. Εδώ περιέχονται όλες οι πληροφορίες που διαβάζονται από το αρχείο Questions.txt. Επίσης φυλάσσεται και η απάντηση που έχει διαλέξει ο χρήστης.

question: Το κείμενο της ερώτησης.

answers: Ένας πίνακας με τις τέσσερις απαντήσεις.

rightAnswer: Η θέση της σωστής απάντησης στον πίνακα. Από το 0 μέχρι το 3.

votes: Λίστα με τις τιμές των ψηφοφοριών του κοινού για κάθε ερώτηση. Η αναπαράσταση των τιμών είναι με την μορφή SortableValue.

fiftyFifty: Λίστα με τις τιμές των βαθμολογιών στην βοήθεια 50/50 για κάθε ερώτηση. Η αναπαράσταση των τιμών είναι με την μορφή SortableValue.

friendCall: Το κείμενο με τα λόγια που λέει ο φίλος του παίκτη στο τηλέφωνο.

level: Το επίπεδο στο οποίο ανήκει η ερώτηση.

selected: Η θέση της απάντησης που έχει διαλέξει ο χρήστης στον πίνακα με τις απαντήσεις.

Question(): Ο constructor της κλάσης με τον οποίο αρχικοποιούνται όλα τα πεδία της κλάσης για να δημιουργηθεί μια ερώτηση.

correct(): Επιστρέφει true εάν η σωστή απάντηση είναι η ίδια με αυτήν που έχει διαλέξει ο παίκτης, αλλιώς επιστρέφει false.

getQuestion(): Επιστρέφει το question.

getAnswers(): Επιστρέφει το answers.

getRightAnswer(): Επιστρέφει το rightAnswer.

getVotes(): Επιστρέφει το votes.

getFiftyFifty(): Επιστρέφει το fiftyFifty.

getFriendCall(): Επιστρέφει το friendCall.

getLevel(): Επιστρέφει το level.

getSelected(): Επιστρέφει το selected.

setSelected(): Δίνει μια τιμή στο selected.

QuestionDeck.java

Αντιπροσωπεύει την «τράπουλα» με τις ερωτήσεις του παιχνιδιού. Εδώ αποθηκεύονται όλες οι ερωτήσεις το παιχνιδιού και από εδώ διαλέγεται τυχαία μια ερώτηση από τον διαχειριστή του παιχνιδιού.

deck: Μια λίστα δύο επιπέδων στην οποία αποθηκεύουμε όλες τις ερωτήσεις του παιχνιδιού που διαβάσαμε από το αρχείο Questions.txt. Στο πρώτο επίπεδο έχουμε μια λίστα 15 θέσεων (μια θέση για κάθε επίπεδο του παιχνιδιού). Κάθε θέση αυτής της λίστας δείχνει σε μία δεύτερη λίστα που περιέχει όλες τις ερωτήσεις του παιχνιδιού που ανήκουν στο συγκεκριμένο επίπεδο (1001 ερωτήσεις).

in: Ένας BufferedReader για να διαβάσουμε από το αρχείο Questions.txt.

QuestionDeck(): Ο constructor της κλάσης. Εδώ αρχικοποιείται το in και το deck. Στην συνέχεια καλείται η συνάρτηση fill() για να γεμίσει η τράπουλα με τις ερωτήσεις.

getRandomQuestionAtLevel(): Παίρνει σαν όρισμα το επίπεδο από το οποίο θέλουμε μια ερώτηση και επιστρέφει μια τυχαία ερώτηση για αυτό το επίπεδο διαλέγοντας την από το deck.

fill(): Γεμίζει το deck με ερωτήσεις. Όσο υπάρχουν ακόμα ερωτήσεις να διαβαστούν, δηλαδή το doneLoading παραμένει false, καλεί την readQuestion() και βάζει την

ερώτηση στο deck στο αντίστοιχο επίπεδο που έχει η ερώτηση μαζί με τις άλλες ερωτήσεις του ίδιου επιπέδου. Το doneLoading θα γίνει true όταν δεν υπάρχουν άλλες ερωτήσεις να διαβαστούν από το αρχείο Questions.txt, δηλαδή φτάσουμε στο τέλος του, ή όταν συμβεί κάποιο σφάλμα κατά το διάβασμα του αρχείο και σκάσει κάποιο Exception. Στο τέλος κλείνει και το in.

readQuestion(): Διαβάζει από το αρχείο Questions.txt τις πληροφορίες για μια ερώτηση, φτιάχνει ένα αντικείμενο Question και το επιστρέφει. Διαβάζει μια-μια τις γραμμές και γεμίζει τα πεδία που αντιστοιχούν σε κάθε γραμμή μέχρι να διαβάσει μια κενή γραμμή, που σημαίνει ότι οι πληροφορίες για αυτήν την ερώτηση τελείωσαν. Ταξινομεί το votes και το fiftyFifty ώστε να είναι κατά φθίνουσα σειρά. Για να φτιάξει το αντικείμενο Question θα χρησιμοποιήσει τον constructor που παίρνει όλα τα πεδία. Τέλος επιστρέφει το αντικείμενο Question.

CustomDialog.java

Είναι το μήνυμα επιβεβαίωσης που εμφανίζεται στον χρήστη και τον ρωτάει εάν είναι σίγουρος για κάποια ενέργεια που θέλει να εκτελέσει. Έχει δύο κουμπιά, το «ΝΑΙ» και το «ΟΧΙ». Κάνει extend το JDialog.

yesButton: Το κουμπί «ΝΑΙ».

noButton: Το κουμπί «ΟΧΙ».

answer: Η επιλογή που πάτησε ο χρήστης. Είναι true εάν ο χρήστης πάτησε το κουμπί «ΝΑΙ» και false εάν ο χρήστης πάτησε το κουμπί «ΟΧΙ».

CustomDialog(): Ο constructor της κλάσης. Παίρνει σαν όρισμα το JFrame στο οποίο θα εμφανίσουμε το μήνυμα, εάν είναι modal, το κείμενο του μηνύματος που θα εμφανιστεί στον χρήστη και η μεταβλητή auto που όπως θα περιγράψουμε παρακάτω δηλώνει εάν το παιχνίδι παίζεται από τον εικονικό παίκτη ή όχι. Φτιάχνει ένα JPanel και βάζει μέσα το κείμενο του μηνύματος. Αρχικοποιεί τα δύο κουμπιά και τα προσθέτει και αυτά στο Panel. Ένα το παιχνίδι είναι σε auto mode τότε περιμένει δυο δευτερόλεπτα και πατάει αυτόματα το κουμπί «ΝΑΙ» (αυτό γίνεται ξεκινώντας έναν Timer και λέγοντάς του ότι θα τρέξει μετά από 2000 miliseconds για μια και μόνο φορά). Τέλος εμφανίζει τον εαυτό του στον χρήστη.

actionPerformed(): Αυτή η συνάρτηση τρέχει όταν πατηθεί κάποιο από τα δύο κουμπιά (επειδή η κλάση κάνει implement το ActionListener τον οποίο έχουμε βάλει στο δύο κουμπιά όταν γίνει κάποιο ActionEvent τότε θα κληθεί αυτή η συνάρτηση). Εάν το κουμπί που πατήθηκε είναι το «ΝΑΙ» τότε κάνει το answer true και κλείνει το μήνυμα διαλόγου. Εάν το κουμπί που πατήθηκε είναι το «ΟΧΙ» τότε κάνει το answer false και κλείνει το μήνυμα διαλόγου.

getAnswer(): Επιστρέφει το answer.

LifeLineFrame.java

Είναι το παράθυρο που έχει τα 3 κουμπιά με τις βοήθειες του παιχνιδιού.

fiftyFiftyButton: Το κουμπί «50/50».

phoneAFriendButton: Το κουμπί «Τηλέφωνο».

pollTheAudienceButton: Το κουμπί «Κοινό».

manager: Ο διαχειριστής του παιχνιδιού όπως αναφέραμε και πιο πάνω. Χρησιμοποιείται εδώ γιατί χρειαζόμαστε διάφορες λειτουργικότητες που μας προσφέρει,

όπως να πάρουμε την τωρινή ερώτηση και να δώσουμε τιμές σε διάφορες μεταβλητές του.

LifeLineFrame(): Ο constructor της κλάσης. Παίρνει σαν όρισμα τον διαχειριστή του παιχνιδιού. Φτιάχνει ένα Jpanel που βάζει μέσα στην κορυφή ένα άλλο Jpanel που περιέχει το κείμενο «Οι βοήθειές σας» και ακριβώς από κάτω ένα δεύτερο Jpanel που περιέχει τα 3 κουμπιά με τα βοήθειες. Αρχικοποιεί τα 3 κουμπιά και τα βάζει στο Jpanel τους. Τέλος αρχικοποιεί την μεταβλητή manager και κάνει εμφανές το panel στον χρήστη.

actionPerformed(): Ισχύει ότι και πριν. Εάν το κουμπί που πατήθηκε είναι το «50/50» τότε καλεί την getFiftyFiftyMessage() για να πάρει το μήνυμα που θα εμφανίσει στον χρήστη. Εάν το παιχνίδι είναι σε auto mode τότε εμφανίζει το μήνυμα στον παίκτη με το TimerDialog όπως θα δούμε παρακάτω και καλεί την συνάρτηση useFistyFifty() του DecisionMaking για να γίνουν όλες οι απαραίτητες ενέργειες του εικονικού παίκτη όπως θα δούμε παρακάτω. Εάν το παιχνίδι είναι σε manual mode τότε απλά εμφανίζει το μήνυμα στον χρήστη σε μορφή διαλόγου το οποίο για να κλείσει ο χρήστης πρέπει να πατήσει OK. Στην συνέχεια απενεργοποιεί το κουμπί «50/50» ώστε να μην μπορεί ο χρήστης να ξαναχρησιμοποιήσει αυτή την βοήθεια. Τέλος κάνει την μεταβλητή fiftyfiftyUsed του διαχειριστή του παιχνιδιού ίση με true ώστε να γνωρίζει ο εικονικός παίκτης ότι αυτή η βοήθεια έχει χρησιμοποιηθεί ήδη. Εάν το κουμπί που πατήθηκε είναι το «Τηλέφωνο» τότε καλεί την getPhoneAFriendMessage() για να πάρει το μήνυμα που θα εμφανίσει στον χρήστη. Εάν το παιχνίδι είναι σε auto mode τότε εμφανίζει το μήνυμα στον παίκτη με το TimerDialog και καλεί την συνάρτηση usePhoneAFriend() του DecisionMaking για να γίνουν όλες οι απαραίτητες ενέργειες του εικονικού παίκτη. Εάν το παιχνίδι είναι σε manual mode τότε απλά εμφανίζει το μήνυμα στον χρήστη σε μορφή διαλόγου το οποίο για να κλείσει ο χρήστης πρέπει να πατήσει OK. Στην συνέχεια απενεργοποιεί το κουμπί «Τηλέφωνο» ώστε να μην μπορεί ο χρήστης να ξαναχρησιμοποιήσει αυτή την βοήθεια. Τέλος κάνει την μεταβλητή phoneAFriendUsed του διαχειριστή του παιχνιδιού ίση με true ώστε να γνωρίζει ο εικονικός παίκτης ότι αυτή η βοήθεια έχει χρησιμοποιηθεί ήδη. Εάν το κουμπί που πατήθηκε είναι το «Κοινό» τότε καλεί την getPollTheAudienceMessage() για να πάρει το μήνυμα που θα εμφανίσει στον χρήστη. Εάν το παιχνίδι είναι σε auto mode τότε εμφανίζει το μήνυμα στον παίκτη με το TimerDialog και καλεί την συνάρτηση usePollTheAudience() του DecisionMaking για να γίνουν όλες οι απαραίτητες ενέργειες του εικονικού παίκτη. Εάν το παιχνίδι είναι σε manual mode τότε απλά εμφανίζει το μήνυμα στον χρήστη σε μορφή διαλόγου το οποίο για να κλείσει ο χρήστης πρέπει να πατήσει OK. Στην συνέχεια απενεργοποιεί το κουμπί «Κοινό» ώστε να μην μπορεί ο χρήστης να ξαναχρησιμοποιήσει αυτή την βοήθεια. Τέλος κάνει την μεταβλητή pollTheAudienceUsed του διαχειριστή του παιχνιδιού ίση με true ώστε να γνωρίζει ο εικονικός παίκτης ότι αυτή η βοήθεια έχει χρησιμοποιηθεί ήδη.

getFiftyFiftyMessage(): Επιστρέφει το κείμενο του μηνύματος που εμφανίζεται στον χρήστη όταν χρησιμοποιεί την βοήθεια του 50/50. Το κείμενο αυτό ενημερώνει τον χρήστη για το ποιες είναι οι δύο λανθασμένες απαντήσεις που πρέπει να αποκλείσει. Για να τις βρει παίρνει την λίστα fiftyFifty από SortableValues που βρίσκεται μέσα στο Question και η οποία είναι ταξινομημένη σε φθίνουσα σειρά και παίρνει το τρίτο και το τέταρτο στοιχείο της λίστας, που εξ' ορισμού θα έχουν το μικρότερο ποσοστό.

getPollTheAudienceMessage(): Επιστρέφει το κείμενο του μηνύματος που εμφανίζεται στον χρήστη όταν χρησιμοποιεί την βοήθεια του κοινού. Το κείμενο αυτό ενημερώνει τον χρήστη για τα ποσοστά που έδωσε το κοινό σε κάθε υποψήφια απάντηση δίνοντας το γράμμα της απάντησης και δίπλα το ποσοστό, σε φθίνουσα σειρά. Για να το κάνει αυτό παίρνει την λίστα votes από SortableValues που βρίσκεται

μέσα στο Question και η οποία είναι ήδη ταξινομημένη σε φθίνουσα σειρά και παίρνει έναν ένα τα στοιχεία της λίστας και εκτυπώνει την πληροφορία τους.

getPhoneAFriendMessage(): Επιστρέφει το κείμενο του μηνύματος που εμφανίζεται στον χρήστη όταν χρησιμοποιεί την βοήθεια του τηλεφώνου σε φίλο. Το κείμενο αυτό ενημερώνει τον χρήστη για την απάντηση που έδωσε ο φίλος του όταν τον πήρε τηλέφωνο. Για να το κάνει αυτό απλά εκτυπώνει το πεδίο friendCall του Question.

getFiftyFiftyButton(): Επιστρέφει το κουμπί fiftyFiftyButton.

getPhoneAFriendButton(): Επιστρέφει το κουμπί phoneAFriendButton.

getPollTheAudienceButton(): Επιστρέφει το κουμπί pollTheAudienceButton.

ProgressBar.java

Είναι η μπάρα προόδου που δείχνει σε ποιο επίπεδο έχει φτάσει ο παίκτης καθώς και το χρηματικό ποσό σε κάθε επίπεδο.

verticalbar: Το JProgressBar που αυξάνεται κάθε φορά που ο παίκτης απαντάει σωστά σε μια ερώτηση και πάει στο επόμενο επίπεδο. Χωρίζεται σε 16 επίπεδα (το πρώτο επίπεδο είναι το μηδενικό επίπεδο και δεν χρωματίζεται) και είναι κατακόρυφη.

ProgressBar(): Ο constructor της κλάσης. Αρχικοποιεί το verticalbar. Στην συνέχεια φτιάχνει ένα panel σε μορφή πίνακα ο οποίος έχει μία στήλη και 15 γραμμές, μία γραμμή για το ποσό κάθε επιπέδου του παιχνιδιού. Αυτό το panel μπαίνει δίπλα από την μπάρα προόδου. Στην συνέχεια γεμίζεται αυτός ο πίνακας με τα ποσά που παίρνουμε από τον πίνακα levels στην κλάση Level. Τα ποσά τοποθετούνται σε φθίνουσα σειρά από πάνω προς τα κάτω. Τα ποσά που αντιστοιχούνε στα μαξιλαράκια χρωματίζονται με κόκκινο χρώμα (αυτά είναι τα 1.000.000, 20.000 και 3.000).

incrementBar(): Αυξάνει την τιμή της μπάρας προόδου verticalbar κατά μια θέση ώστε να φανεί η πρόοδος στα επίπεδα. Καλείται όταν ο χρήστης απαντήσει σωστά σε μια ερώτηση.

RadioListener.java

Είναι η περιοχή που δείχνει το κείμενο της τωρινής ερώτησης και τις τέσσερις πιθανές απαντήσεις από τις οποίες μπορείς να διαλέξεις μόνο μια (radioButton).

questionLabel: Το κείμενο της ερώτησης.

buttons: Ένας πίνακας από 4 JRadioButtons, ένα για κάθε πιθανή απάντηση της ερώτησης.

question: Η τωρινή ερώτηση του παιχνιδιού.

RadioListener(): Ο constructor της κλάσης. Παίρνει σαν όρισμα την τρέχουσα ερώτηση του παιχνιδιού που φυλάσσεται στον διαχειριστή του παιχνιδιού και την βάζει στο πεδίο question. Φτιάχνει ένα Jpanel. Στην συνέχεια αρχικοποιεί το questionLabel όπου εκεί βάζει το κείμενο της ερώτησης του παιχνιδιού που το παίρνει από το question και το προσθέτει σε αυτό το panel. Αρχικοποιεί το buttons και σε κάθε μια από τις 4 θέσεις του πίνακα βάζει το κείμενο της αντίστοιχης απάντησης αφού στην αρχή βάλει και το γράμμα που αντιστοιχεί σε αυτή την απάντηση καλώντας την numberToLetter του UtilityHelper. Φτιάχνει ένα ButtonGroup και βάζει μέσα αυτά τα 4 JRadioButtons ώστε ο χρήστης να μην μπορεί να διαλέξει πάνω από ένα κάθε φορά. Τέλος προσθέτει τα κουμπιά στο panel και το εμφανίζει.

actionPerformed(): Ισχύει ό,τι και πριν. Ελέγχει ένα-ένα τα κουμπιά του πίνακα buttons. Εάν το κουμπί είναι πατημένο τότε καλεί την setSelected του Question με όρισμα την θέση του κουμπιού στον πίνακα buttons έτσι ώστε να καταγραφεί ποια απάντηση διάλεξε ο χρήστης.

changeQuestion(): Αλλάζει την ερώτηση που εμφανίζεται. Βάζει στην μεταβλητή question την καινούργια τωρινή ερώτηση που φορτώθηκε στον διαχειριστή του παιχνιδιού. Στο questionLabel βάζει το καινούργιο κείμενο της ερώτησης. Για καθένα από τα 4 κουμπιά βάζει το καινούργιο κείμενο της απάντησης και ξεδιαλέγει τυχόν διαλεγμένα από πριν κουμπιά. Τέλος ξανασχεδιάζει το panel για να εμφανιστούν οι αλλαγές που έγιναν.

getButtons(): Επιστρέφει το buttons.

QuestionWindowPanel.java

Είναι η περιοχή του κυρίως παραθύρου που δείχνει την περιοχή του RadioListener που περιγράψαμε παραπάνω μαζί με τα κουμπιά «Κατοχυρώστε» και «Παραιτηθείτε».

radioListener: Ο RadioListener που περιγράψαμε προηγουμένως.

lock: Το κουμπί «Κατοχυρώστε».

stop: Το κουμπί «Παραιτηθείτε»

manager: Ο διαχειριστής του παιχνιδιού. Ισχύουν τα ίδια με το LifelLineFrame.

QuestionWindowPanel(): Ο constructor της κλάσης. Παίρνει σαν όρισμα τον διαχειριστή του παιχνιδιού και τον βάζει στην μεταβλητή manager. Αρχικοποιεί το radioListener περνώντας του σαν όρισμα την τωρινή ερώτηση που βρίσκεται στο manager και το προσθέτει στον εαυτό του. Αρχικοποιεί τα δύο κουμπιά και τα προσθέτει και αυτά στον εαυτό του.

refreshWindow(): Καλεί την changeQuestion() του radioListener περνώντας σαν όρισμα την καινούργια τωρινή ερώτηση του παιχνιδιού. Ξανασχεδιάζει όλα τα στοιχεία του panel.

actionPerformed(): Ισχύει ό,τι και πριν. Εάν το κουμπί που θα πατηθεί είναι το «Κατοχυρώστε» φτιάχνει ένα CustomDialog ρωτώντας τον χρήστη εάν η απάντηση που διάλεξε είναι η τελική του απάντηση. Εάν ο χρήστης πατήσει «ΝΑΙ», δηλαδή εάν το answer είναι true, τότε καλεί την correct() του Question στην τωρινή ερώτηση για να ελέγξει εάν ο χρήστης διάλεξε την σωστή απάντηση. Εάν η απάντηση του χρήστη είναι η σωστή τότε καλεί την nextLevelQuestion() του GameManager που θα αναλύσουμε παρακάτω για να πάει στην επόμενη ερώτηση. Εάν η απάντηση του χρήστη είναι λάθος τότε εμφανίζει ένα μήνυμα στον χρήστη που του λέει ότι έχασε και ότι κερδίζει το ποσό που αντιστοιχεί στο μαξιλαράκι που έχει φτάσει, και τερματίζει την εφαρμογή. Εάν ο χρήστης πατήσει «ΟΧΙ» τότε δεν κάνει τίποτα. Εάν το κουμπί που θα πατηθεί είναι το «Παραιτηθείτε» τότε φτιάχνει ένα CustomDialog ρωτώντας τον χρήστη εάν είναι σίγουρος ότι θέλει να αποχωρήσει από το παιχνίδι παίρνοντας τα χρήματα που έχει κερδίσει μέχρι τώρα. Εάν ο χρήστης πατήσει «ΝΑΙ», δηλαδή εάν το answer είναι true, τότε εμφανίζει μήνυμα στον χρήστη ότι κέρδισε και ότι αποχωρεί από το παιχνίδι κερδίζοντας το συγκεκριμένο ποσό που αντιστοιχεί στο επίπεδο μέχρι το οποίο έφτασε. Εάν ο χρήστης πατήσει «ΟΧΙ» τότε δεν κάνει τίποτα.

windowStateChanged(): Δεν κάνει τίποτα απολύτως απλά χρειάζεται να γίνει implement.

getLock(): Επιστρέφει το lock.

getStop(): Επιστρέφει το stop.

getRadioListener(): Επιστρέφει το radioListener.

GameManager.java

Είναι ο διαχειριστής του παιχνιδιού. Αποτελεί το κεντρικό παράθυρο της εφαρμογής μέσα στο οποίο μπαίνουν και όλα τα παραπάνω γραφικά στοιχεία που περιγράψαμε. Εδώ είναι και η κλάση στην οποία βρίσκεται η main συνάρτηση που ξεκινάει την εφαρμογή. Εδώ εκτός από πληροφορίες για το γραφικό περιβάλλον του παιχνιδιού, γίνεται και ο συντονισμός του εικονικού παίκτη όπως θα περιγράψουμε παρακάτω. Τέλος, εδώ φυλάσσεται η τωρινή ερώτηση του παιχνιδιού, το τρέχον επίπεδο που βρίσκεται ο παίκτης, η τράπουλα με όλες τις ερωτήσεις του παιχνιδιού και οι μεταβλητές που δείχνουν εάν έχει χρησιμοποιηθεί κάποια από τις 3 βοήθειες.

startButton: Το κουμπί «Ξεκινήστε».

instructionButton: Το κουμπί «Οδηγίες».

autoButton: Η επιλογή «Auto».

wikipediaButton: Η επιλογή «Μόνο Wikipedia».

questionWindowPanel: Το QuestionWindowPanel που περιεγράφηκε παραπάνω.

lifeLineFrame: Το LifeLineFrame που περιγράφηκε παραπάνω.

progressBar: Το ProgressBar που περιγράφηκε παραπάνω.

startMenu: Το panel που θα μπουν όλα τα γραφικά στοιχεία της οθόνης που εμφανίζεται στον χρήστη με το που ξεκινήσει η εφαρμογή (τα κουμπιά «Ξεκινήστε» και «Οδηγίες», και οι επιλογές «Auto» και «Μόνο Wikipedia»).

east: Το panel που θα μπει η μπάρα προόδου ProgressBar που περιγράφηκε παραπάνω.

questionDeck: Το QuestionDeck που περιγράφηκε παραπάνω.

level: Το τρέχον επίπεδο του παιχνιδιού (παίρνει τιμές από 1 μέχρι 15).

currentQuestion: Η τρέχουσα ερώτηση του παιχνιδιού.

auto: Μεταβλητή που δείχνει εάν ο εικονικός παίκτης είναι ενεργός και άρα το παιχνίδι είναι σε auto mode ή όχι. Είναι true εάν ο χρήστης έχει διαλέξει την επιλογή «Auto» στην αρχική οθόνη, αλλιώς είναι false.

wikipedia: Μεταβλητή που δείχνει εάν η μηχανή αναζήτησης της Google πρέπει να ψάξει για τις απαντήσεις μόνο σε σελίδες της Wikipedia ή σε όλο το διαδίκτυο. Είναι true εάν ο χρήστης έχει διαλέξει την επιλογή «Μόνο Wikipedia» στην αρχική οθόνη, αλλιώς είναι false.

fiftyFiftyUsed: Μεταβλητή που δείχνει εάν έχει χρησιμοποιηθεί η βοήθεια 50/50 ή όχι. Είναι true εάν ο παίκτης χρησιμοποίησε την βοήθεια αλλιώς είναι false.

pollTheAudienceUsed: Μεταβλητή που δείχνει εάν έχει χρησιμοποιηθεί η βοήθεια του κοινού ή όχι. Είναι true εάν ο παίκτης χρησιμοποίησε την βοήθεια αλλιώς είναι false.

phoneAFriendUsed: Μεταβλητή που δείχνει εάν έχει χρησιμοποιηθεί η βοήθεια του τηλεφώνου σε φίλο ή όχι. Είναι true εάν ο παίκτης χρησιμοποίησε την βοήθεια αλλιώς είναι false.

GameManager(): Ο constructor της κλάσης. Φτιάχνει ένα καινούργιο QuestionDeck, αρχικοποιεί το level με την MINIMUM τιμή από την κλάση Level που είναι ίσο με 1 και διαλέγει τυχαία από το questionDeck μια ερώτηση για αυτό το επίπεδο (το 1) και την βάζει στο currentQuestion.

startGame(): Η συνάρτηση που ξεκινά το παιχνίδι. Φτιάχνει το κυρίως παράθυρο του παιχνιδιού στο οποίο δίνει τίτλο «Ποιός θέλει να γίνει εκατομμυριούχος;». Μετά φτιάχνει ένα καινούργιο QuestionWindowPanel το οποίο δεν φαίνεται στην αρχή στον χρήστη και το προσθέτει στο αρχικό JFrame στο πάνω κομμάτι του παραθύρου. Μετά φτιάχνει ένα καινούργιο ProgressBar και το προσθέτει στο panel east. Προσθέτει αυτό το east panel στο αρχικό JFrame στο δεξί κομμάτι του παραθύρου. Στη συνέχεια καλεί την συνάρτηση startMenuPanel() που θα δούμε παρακάτω και σχεδιάζει το αρχικό παράθυρο που θα δει ο χρήστης με το που ξεκινήσει το παιχνίδι και το προσθέτει στο αρχικό JFrame στο κέντρο. Τέλος φτιάχνει ένα καινούργιο LifeLineFrame και το κάνει αόρατο στον χρήστη, ενώ κάνει το αρχικό JFrame ορατό. Επομένως το αρχικό παράθυρο έχει ανοίξει και πλέον ο χρήστης μπορεί να αλληλοεπιδράσει με το γραφικό περιβάλλον και να αρχίσει να παίζει το παιχνίδι.

startMenuPanel(): Σχεδιάζει την αρχική οθόνη που βλέπει ο χρήστης με το που ξεκινήσει το παιχνίδι. Φτιάχνει τα κουμπιά «Ξεκινήστε» και «Οδηγίες» και τις επιλογές «Auto» και «Μόνο Wikipedia». Φτιάχνει ένα Jpanel και προσθέτει μέσα τα δύο κουμπιά και τις δύο επιλογές που έφτιαξε. Επιστρέφει αυτό το panel.

itemStateChanged(): Η κλάση GameManager κάνει implement ένα ItemListener ώστε να μπορεί να καταλάβεί πότε ο χρήστης αλληλοεπιδρά με τις δύο επιλογές JCheckBox. Για να πιάσουμε το ItemEvent που συμβαίνει πρέπει η κλάση να υλοποιήσει αυτήν την συνάρτηση. Εάν η επιλογή με την οποία αλληλοεπιδρά ο χρήστης είναι η «Auto» τότε ελέγχουμε εάν ο χρήστης διάλεξε την επιλογή και κάνουμε την μεταβλητή auto ίση με true αλλιώς εάν την ξεδιάλεξε τότε κάνουμε το auto ίσο με false. Εάν η επιλογή με την οποία αλληλοεπιδρά ο χρήστης είναι η «Μόνο Wikipedia» τότε ελέγχουμε εάν ο χρήστης διάλεξε την επιλογή και κάνουμε την μεταβλητή wikipedia ίση με true αλλιώς εάν την ξεδιάλεξε τότε κάνουμε το wikipedia ίσο με false.

nextLevelQuestion(): Κάνει όλες τις απαραίτητες ενέργειες για να προχωρήσει το παιχνίδι στην επόμενη ερώτηση. Αρχικά αυξάνει κατά ένα το επίπεδο του παιχνιδιού και καλεί την incrementBar() του ProgressBar. Ξανασχεδιάζει το παράθυρο. Εάν το επίπεδο του παιχνιδιού είναι ακόμη μικρότερο από το μέγιστο επίπεδο MAXIMUM του Level, τότε φορτώνει από το questionDeck μια τυχαία ερώτηση για το καινούργιο level και την βάζει στο currentQuestion. Καλεί την refreshWindow() του QuestionWindowPanel. Εάν το παιχνίδι είναι σε auto mode τότε περιμένει 3 δευτερόλεπτα και καλεί την συνάρτηση startVirtualPlayer() που θα δούμε παρακάτω για να ξεκινήσει τον εικονικό παίκτη. Αυτό γίνεται με ένα Timer όπως είδαμε και παραπάνω. Εάν όμως έχουμε φτάσει το μέγιστο επίπεδο του παιχνιδιού αυτό σημαίνει ότι δεν υπάρχει άλλο επίπεδο και ο παίχτης έχει νικήσει το παιχνίδι και είναι ένας εκατομμυριούχος. Σε αυτή την περίπτωση εμφανίζει ένα μήνυμα στον παίκτη που του λέει ότι κέρδισε και του δίνει συγχαρητήρια και τερματίζει την εφαρμογή.

actionPerformed(): Ισχύει ό,τι και πριν. Εάν το κουμπί που πατήθηκε είναι το κουμπί «Οδηγίες» τότε εμφανίζει ένα μήνυμα στον χρήστη που του εξηγεί τις οδηγίες του παιχνιδιού. Εάν το κουμπί που πατήθηκε είναι το κουμπί «Ξεκινήστε», τότε εμφανίζει το questionWindowPanel, το east panel και το LifeLineFrame ενώ κρύβει το startMenu. Εάν το παιχνίδι είναι σε auto mode, τότε περιμένει 3 δευτερόλεπτα και καλεί την συνάρτηση startVirtualPlayer() η οποία ξεκινά τον εικονικό παίκτη.

isAuto(): Επιστρέφει το auto.

getLevel(): Επιστρέφει το level.

getCurrentQuestion(): Επιστρέφει το currentQuestion.

isFiftyFiftyUsed(): Επιστρέφει το fiftyFiftyUsed.

isPollTheAudienceUsed(): Επιστρέφει το pollTheAudienceUsed.

isPhoneAFriendUsed(): Επιστρέφει το phoneAFriendUsed.

setFiftyFiftyUsed(): Δίνει μια τιμή στο fiftyFiftyUsed.

setPollTheAudienceUsed(): Δίνει μια τιμή στο pollTheAudienceUsed.

setPhoneAFriendUsed(): Δίνει μια τιμή στο phoneAFriendUsed.

main(): Φτιάχνει έναν GameManager και καλεί την συνάρτηση startGame().

8.2.3 Υλοποίηση του εικονικού παίκτη

Όταν υλοποιήθηκε ο εικονικός παίκτης ακολουθήθηκε η βασική γραμμή που είχαν ορίσει οι συγγραφείς του άρθρου [1], δηλαδή φτιάχτηκαν 3 μονάδες (QA, AS, DM). Ο εικονικός παίκτης παίζει το παιχνίδι μόνο όταν ο χρήστης που ξεκινάει την εφαρμογή πριν πατήσει το κουμπί «Ξεκινήστε» έχει διαλέξει πρώτα την επιλογή «Auto». Όταν λοιπόν το παιχνίδι ξεκινάει σε auto mode τότε η βασική ιδέα είναι ότι γίνεται μια προσπάθεια να προσομοιωθούν οι ενέργειες που θα έκανε ο ανθρώπινος παίκτης πατώντας τα διάφορα κουμπιά στο γραφικό περιβάλλον. Αυτό γίνεται με την βοήθεια Timers όπως αναφέρθηκε σε πολλές περιπτώσεις παραπάνω. Τα βασικά βήματα του εικονικού παίκτη είναι ότι πρώτα χρησιμοποιεί την μονάδα QA για να συλλέξει τα κείμενα με τις πληροφορίες από το διαδίκτυο για την κάθε ερώτηση, μετά χρησιμοποιεί την μονάδα AS για να βαθμολογήσει τις απαντήσεις. Εάν δεν μπορεί να πάρει μια απόφαση τότε ξανακαλεί την μονάδα AS αλλά αυτή την φορά χρησιμοποιώντας στις απαντήσεις και στο ψάξιμο κανονικές εκφράσεις (regular expressions). Εάν ακόμα δεν μπορεί να πάρει μια απόφαση τότε χρησιμοποιεί τις βοήθειες που του έχουν απομείνει και αποφασίζει τι θα κάνει χρησιμοποιώντας την μονάδα DM. Επομένως στην εφαρμογή αυτή ο εικονικός παίκτης θα φαίνεται όπως όταν παίζει και ο ανθρώπινος παίκτης και δεν θα παρουσιάζει απλά το αποτέλεσμα στο τέλος χωρίς να φαίνεται στον χρήστη τι έγινε ενδιάμεσα. Είναι δηλαδή σαν να παρακολουθεί κάποιος κάποιον άλλον να παίζει το παιχνίδι.

Για την υλοποίηση του εικονικού παίκτη φτιάχτηκαν οι παρακάτω κλάσεις (κάποιες είναι κοινές και με την υλοποίηση του γραφικού περιβάλλοντος αφού σε πολλές περιπτώσεις ο κώδικας δεν μπορούσε να διαχωριστεί πλήρως καθώς ο εικονικός παίκτης όπως αναφέρθηκε αλληλοεπιδρά με το γραφικό περιβάλλον):

UtilityHelper.java

***parseFriendAnswer()*:** Παίρνει σαν όρισμα το κείμενο της απάντησης που δήθεν έδωσε ο φίλος του παίκτη όταν χρησιμοποίησε την βοήθεια του τηλεφώνου. Αυτό το κείμενο είναι το ίδιο με αυτό που υπάρχει στα δεδομένα της ερώτησης στο αρχείο Questions.txt. Εάν το κείμενο περιέχει το γράμμα Α επιστρέφει 0, εάν το γράμμα Β επιστρέφει 1, εάν το γράμμα Γ επιστρέφει 2 και εάν το γράμμα Δ επιστρέφει 3. Το γράμμα κάθε φορά θα πρέπει να έχει πριν ένα κενό χαρακτήρα ώστε να μην μετράει εάν υπάρχει μέσα σε κάποια λέξη και θα πρέπει να ακολουθείται από κενό χαρακτήρα ή από τελεία για να μην μετράει όταν βρίσκεται στην αρχή κάποια λέξης. Εάν δεν βρει κανέναν από αυτά τα γράμματα τότε επιστρέφει 0 γιατί έτσι κι αλλιώς η βοήθεια του τηλεφώνου δεν χρησίμευσε επομένως επιλέγει στην τύχη την απάντηση Α.

TimerDialog.java

Είναι ένα μήνυμα που εμφανίζεται υποθετικά στον εικονικό παίκτη και κλείνει μόνο του μετά από κάποια δευτερόλεπτα σαν να το έκλεισε ο εικονικός παίχτης. Έχει φτιαχτεί καθαρά για να μας δώσει την αίσθηση ότι κάποιος όντως παίζει το παιχνίδι και για να έχουμε μια ιδέα τι συμβαίνει στο παιχνίδι όσο παίζει ο εικονικός παίκτης.

show(): Παίρνει σαν όρισμα τα δευτερόλεπτα timerLeftSecs που θα εμφανιστεί το μήνυμα και το κείμενο του μηνύματος displayText. Φτιάχνει ένα JLabel και ένα παράθυρο που σκρολάρει για να μπει μέσα το κείμενο του μηνύματος. Περιμένει ένα δευτερόλεπτο και ξεκινάει έναν Timer που τρέχει κάθε δευτερόλεπτο και ο οποίος όσο μένουν ακόμα δευτερόλεπτα συνεχίζει να εμφανίζει το μήνυμα και μειώνει κατά ένα τα δευτερόλεπτα που έχουν απομείνει. Ενώ εάν δεν έχουν απομείνει άλλα δευτερόλεπτα, δηλαδή πέρασαν timerLeftSecs από τότε που εμφανίστηκε το μήνυμα, τότε σταματάει τον Timer και εξαφανίζει το μήνυμα από την οθόνη.

GoogleResult.java

Δομή δεδομένων που βοηθά στο να κρατήσουμε και να αναπαραστήσουμε την πληροφορία που μας επιστρέφει η μηχανή αναζήτησης της Google. Για κάθε αποτέλεσμα που μας επέστρεψε η Google κρατάμε την επικεφαλίδα του αποτελέσματος, το link και μια λίστα από κείμενα που πήραμε όταν επισκεφτήκαμε το συγκεκριμένο link.

linkHref: Το link του αποτελέσματος.

linkText: Η επικεφαλίδα του αποτελέσματος.

passages: Λίστα με τα διάφορα κείμενα που πήραμε όταν επισκεφτήκαμε την σελίδα του αποτελέσματος.

GoogleResult(): Ο constructor της κλάσης. Αρχικοποιεί τα 3 πεδία της κλάσης.

getLinkHref(): Επιστρέφει το linkHref.

setLinkHref(): Δίνει μια τιμή στο linkHref.

getLinkText(): Επιστρέφει το linkText.

setLinkText(): Δίνει μια τιμή στο linkText.

getPassages(): Επιστρέφει το passages.

setPassages(): Δίνει μια τιμή στο passages.

QuestionAnswering.java

Υλοποιεί την μονάδα Απάντησης Ερωτήσεων (QA). Παίρνει το κείμενο μιας ερώτησης και ρωτάει αυτή την ερώτηση, αφού την καθαρίσει πρώτα από τυχόν περιττές λέξεις, στην μηχανή αναζήτησης της Google χρησιμοποιώντας το API που παρέχεται. Το API αυτό επιστρέφει μια σελίδα URL με τα αποτελέσματα της αναζήτησης που είναι αυτή που βλέπει κάποιος στον browser του όταν κάνει το ίδιο ερώτημα στην αρχική σελίδα της Google. Στην συνέχεια αυτή η HTML σελίδα σκανάρεται για να εξαχθεί ένας συγκεκριμένος αριθμός από αποτελέσματα. Για κάθε αποτέλεσμα GoogleResult κρατάμε τον τίτλο του και τον σύνδεσμό του τον οποίο και ακολουθούμε. Έτσι μας επιστρέφεται για κάθε αποτέλεσμα μια σελίδα HTML από την οποία εξάγουμε όλα τα πεδία κειμένου που υπάρχουν και τα σώζουμε σε μια λίστα. Τέλος η λίστα με όλα τα

GoogleResults επιστρέφεται. Σε αντίθεση με την μονάδα QA που έχει υλοποιηθεί στο άρθρο [1] η οποία είναι πολύ εξελιγμένη και κάνει διάφορες αναλύσεις στα κείμενα με τα αποτελέσματα, η μονάδα QA της παρούσας εργασίας υλοποιεί μόνο το Google Baseline (όπως θα δούμε παρακάτω) και κάνει μια dummy αναζήτηση. Παρέχει επίσης την δυνατότητα να ψάξουμε μόνο σελίδες της Wikipedia (Wikipedia Baseline).

Ο κώδικας μπορεί να βρεθεί και στο Παράρτημα 2.

GOOGLE_SEARCH_API_URL: Ο σύνδεσμος για το API της μηχανής αναζήτησης της Google. Είναι ίσο με «https://www.google.com/search?».

DEFAULT_RESULTS_NUMBER: Ο μέγιστος αριθμός των αποτελεσμάτων που θέλουμε να επιστρέψει η μονάδα QA. Είναι ίσο με 30. Υπάρχει πιθανότητα ορισμένες φορές η μηχανή αναζήτησης να μην μπορέσει να επιστρέψει τόσα αποτελέσματα αλλά λιγότερα.

DEFAULT_ELEMENT_TAG: Το HTML tag μέσα στο οποίο θα ψάξουμε για τα κομμάτια κειμένου. Είναι το «div».

DEFAULT_WIKIPEDIA_ELEMENT_TAG: Το HTML tag αλλά στην περίπτωση που ο χρήστης επιλέξει να αποτελέσματα να προέρχονται μόνο από σελίδες της Wikipedia. Είναι το «div#mw-content-text».

DEFAULT_WIKIPEDIA_SPECIFIC_DOMAIN: Το domain name από το οποίο θέλουμε να πούμε στο google search api να μας φέρει αποτελέσματα. Στην περίπτωσή μας είναι το domain name της Wikipedia που είναι ίσο με «wikipedia.org».

findResults(): Παίρνει σαν ορίσματα το κείμενο της ερώτησης του παιχνιδιού question, τον μέγιστο αριθμό αποτελεσμάτων που θέλουμε να μας επιστραφούν resultsNumber, το συγκεκριμένο domain name από το οποίο θέλουμε να μας έρθουν όλα τα αποτελέσματα specificDomain και το HTML tag από το οποίο θέλουμε να εξάγουμε τα κομμάτια κειμένου elementTag. Επιστρέφει μια λίστα από GoogleResult. Στην αρχή χτίζει το URL του google search api από το οποίο θα έρθουν τα αποτελέσματα. Αυτό το URL εκτός από το GOOGLE_SEARCH_API_URL παίρνει ένα όρισμα «q» που είναι η ερώτηση. Για να πάρουμε την ερώτηση πρέπει πρώτα να την καθαρίσουμε και να την φέρουμε στην μορφή που είναι κατανοητή από το api (αντί για κενά να έχει το σύμβολο '+'). Για το λόγο αυτό περνάμε το question μέσα από την συνάρτηση cleanQuestion() που θα δούμε παρακάτω. Επίσης παίρνει ένα όρισμα «num» που είναι ο αριθμός των αποτελεσμάτων που θέλουμε να μας επιστρέψει η μηχανή αναζήτησης. Κάποια από τα αποτελέσματα που θα μας επιστραφούν μπορεί να μην είναι απλές HTML σελίδες αλλά να είναι διάφορα αρχεία όπως .pdf ή .ppt κτλ. Αυτά τα αποτελέσματα δεν μας ενδιαφέρουν γιατί δεν μπορούμε να τα επεξεργαστούμε. Επομένως πρέπει να τα αποκλείσουμε. Για το λόγο αυτό στο num πρέπει να δώσουμε έναν αριθμό μεγαλύτερο από το όρισμα resultsNumber ώστε να έχουμε περισσότερα αποτελέσματα από αυτά που θα κρατήσουμε στο τέλος για να μπορούμε να πετάμε αυτά που δεν μας κάνουν. Εδώ έχουμε δώσει τον αριθμό 100 ώστε να είμαστε καλυμμένοι. Έτσι κι αλλιώς δεν πρόκειται να επεξεργαστούμε και τα 100 αποτελέσματα αλλά μέχρι να συμπληρώσουμε μια λίστα με resultsNumber αποτελέσματα. Εάν το όρισμα specificDomain δεν είναι κενό τότε βάζουμε στο api και την μεταβλητή «as_sitesearch» στο οποίο βάζουμε την τιμή του specificDomain. Εφόσον έχουμε κατασκευάσει πλέον ολόκληρο το URL που θα υποβάλουμε στο API είμαστε έτοιμοι να το καλέσουμε. Πριν το κάνουμε αυτό αρχικοποιούμε μια λίστα από GoogleResult για να βάλουμε μέσα τα αποτελέσματα και εάν το όρισμα resultsNumber δεν έχει οριστεί του δίνουμε την default τιμή DEFAULT_RESULTS_NUMBER. Εκτελούμε το URL του API με την βοήθεια της βιβλιοθήκης Jsoup και παίρνουμε την HTML σελίδα με τα αποτελέσματα. Βρίσκουμε τα αποτελέσματα μέσα στην σελίδα τα οποία αναγνωρίζονται από τον HTML tag «h3.r >

a». Για κάθε ένα τέτοιο αποτέλεσμα, το οποίο είναι τύπου Element, κρατάμε διάφορες πληροφορίες. Αυτές είναι ο σύνδεσμος του αποτελέσματος το οποίο το αποθηκεύουμε στην μεταβλητή linkHref και ο τίτλος του αποτελέσματος που το κρατάμε στην linkText. Στην συνέχεια ακολουθούμε πάλι με την βοήθεια της βιβλιοθήκης Jsoup τον σύνδεσμο του αποτελέσματος και παίρνουμε την HTML σελίδα που αντιστοιχεί σε αυτό. Εάν ο σύνδεσμος δεν δείχνει σε κάποια HTML σελίδα όπως αναφέραμε πριν απλά αγνοούμε αυτό το αποτέλεσμα και προχωράμε στο επόμενο (αυτό γίνεται με το να χειριστούμε το Exception που θα σκάσει όταν πάμε να ακολουθήσουμε τον σύνδεσμο). Δημιουργούμε ένα καινούργιο GoogleResult χρησιμοποιώντας τον constructor του που αναφέραμε πριν και περνώντας σαν ορίσματα το linkHref και το linkText. Μετά από την σελίδα HTML του αποτελέσματος παίρνουμε όλα τα Elements που αντιστοιχούν στο elementTag που δόθηκε σαν όρισμα ή στο DEFAULT_ELEMENT_TAG εάν το πρώτο είναι κενό. Από τα Elements αυτά εξάγουμε μόνο το κείμενο πετώντας τυχόν html ή css κώδικα. Βάζουμε όλα αυτά τα κείμενα στην λίστα passages του GoogleResult που φτιάξαμε προηγουμένως. Έτσι γεμίζει το GoogleResult και το βάζουμε στην λίστα με τα αποτελέσματα. Τελειώνουμε την επεξεργασία μας όταν η λίστα με τα αποτελέσματα φτάσει τον απαιτούμενο αριθμό ή όταν τελειώσουν όλα τα αποτελέσματα που επέστρεψε η μηχανή αναζήτησης. Τέλος επιστρέφει την λίστα με τα GoogleResult.

cleanQuestion(): Καθαρίζει τον κείμενο της ερώτησης το οποίο παίρνει σαν όρισμα (question) και επιστρέφει το καθαρισμένο κείμενο (cleanQuestion). Στην αρχή ελέγχει εάν το question περιέχει τον χαρακτήρα με τα διπλά αυτάκια ‘”’. Εάν το περιέχει, τότε κρατάμε μόνο αυτό που περιέχεται μέσα στα διπλά αυτάκια, καθώς εκεί είναι όλο το ζουμί της ερώτησης και όλα τα υπόλοιπα δεν είναι και τόσο χρήσιμα, οπότε μπορούμε να τα πετάξουμε. Αυτό είναι μια εμπειρική διαπίστωση και το χρησιμοποιούμε σαν ευρετικό. Στην συνέχεια εξαφανίζουμε από το κείμενο της ερώτησης τυχόν ‘…’ (περιέχονται σε πολλές ερωτήσεις) αφού οι τρείς τελίτσες δεν προσθέτουν τίποτα στην ερώτηση και αντιθέτως μπορούν να μπερδέψουν την μηχανή αναζήτησης. Επιπλέον εξαφανίζουμε τυχόν ερωτηματικά ‘;’ που περιέχονται στο τέλος της ερώτησης αφού δεν μας ενδιαφέρουν (ξέρουμε ότι πρόκειται για ερώτηση και η μηχανή αναζήτησης δεν δίνει σημασία στα ερωτηματικά). Τέλος επειδή το google search api δεν δέχεται κενά αλλά οι λέξεις πρέπει να συνδέονται με το σύμβολο ‘+’ αντικαθιστούμε όλα τα κενά με το ‘+’.

AnswerScoring.java

Υλοποιεί την μονάδα Βαθμολόγησης Απαντήσεων (AS). Παίρνει την λίστα με τα GoogleResult που επέστρεψε η μονάδα QA και αφού πρώτα καθαρίσει τις 4 απαντήσεις από τυχόν άχρηστες λέξεις και λέξεις που είναι κοινές στο τέλος και στην αρχή μετρά πόσες φορές βρίσκονται οι απαντήσεις μέσα στα κομμάτια κειμένου. Μετρά ξανά και πόσες φορές υπάρχουν οι απαντήσεις αφού πρώτα κάνει όλα τα γράμματα μικρά γιατί μπορεί στα κείμενα οι απαντήσεις να βρίσκονται με μικρά γράμματα. Επίσης εάν μια απάντηση βρίσκεται στον τίτλο ενός GoogleResult θα πάρει παραπάνω πόντους. Τέλος, εάν η απάντηση αποτελείται από παραπάνω από μια λέξεις, μας ενδιαφέρει να χρησιμοποιήσουμε κάποιο είδος κανονικής έκφρασης καθώς άλλες λέξεις μπορεί να περιέχονται ανάμεσα στις λέξεις της απάντησης και η απλή καταγραφή δεν πιάνει αυτές τις περιπτώσεις. Όπως όμως θα αναφέρουμε και παρακάτω η χρήση των regular expressions γίνεται όταν τα αποτελέσματα της απλής καταμέτρησης δεν είναι ικανά να μας δώσουν μια ξεκάθαρη απάντηση. Αξίζει να σημειωθεί πως εάν η ερώτηση περιέχει κάποιες αρνητικές λέξεις τότε η σωστή απάντηση είναι εκείνη που θα μαζέψει το λιγότερο σκορ, επομένως πολλαπλασιάζουμε όλα τα σκορ με το -1 ώστε αυτή με το μικρότερο σκορ να γίνει πλέον αυτή με το μεγαλύτερο και έτσι να δοθεί σαν τελική απάντηση. Σε αντίθεση με την μονάδα AS που έχουν υλοποιήσει οι συγγραφείς του

άρθρου [1] η οποία είναι πολύ πιο εξελιγμένη και χρησιμοποιεί διάφορα κριτήρια όπως είδαμε και παραπάνω όπως η διασπορά κτλ. για να βαθμολογήσει τις απαντήσεις εμείς απλώς μετράμε πόσες φορές βρίσκεται κάθε απάντηση μέσα στα χωρία κειμένου, χωρίς να ληφθεί υπόψιν κανένα κριτήριο.

Ο κώδικας μπορεί να βρεθεί και στο Παράρτημα 2.

FOUND_AT_TITLE_BONUS: Η επιπλέον βαθμολογία που δίνεται σε μια απάντηση εάν βρεθεί στον τίτλο του GoogleResult. Αυτό συμβαίνει γιατί ο τίτλος είναι πιο σημαντικός από τα χωρία κειμένου της σελίδας.

REGEX_TEXT_NUMBER_LIMIT: Όταν χρησιμοποιούμε στην βαθμολόγηση regular expressions το πρόγραμμα μπορεί να κρεμάσει λόγω του μεγάλου όγκου των χωρίων κειμένου. Για να μην γίνει αυτό ορίζουμε ένα όριο στο πόσα GoogleResults θα επεξεργαστούμε για να βρούμε τις εμφανίζεις των απαντήσεων. Η τιμή αυτή είναι ίση με 1. Όσο πιο μεγάλη είναι η τιμή τόσο πιο ακριβή αποτελέσματα θα δώσει η μονάδα AS αλλά τόσο περισσότερο χρόνο θα πάρει.

NEGATIVE_QUESTION_IDENTIFIER: Η λέξη που εάν εμφανιστεί στο κείμενο της ερώτησης καταλαβαίνουμε ότι η ερώτηση έχει αρνητική σημασία και άρα πρέπει να πολλαπλασιάσουμε τα αποτελέσματα με το -1. Η λέξη αυτή είναι το «δεν».

IRRELEVANT_WORDS_UPPER: Μια λίστα από λέξεις που εάν εμφανίζονται μέσα στην απάντηση μπορούμε να την βγάλουμε καθώς δεν θα κάνει καμία διαφορά στο νόημα της απάντησης και ίσα-ίσα που με το να βρίσκεται εκεί κάνει πολύ πιο δύσκολη την μέτρηση. Η λίστα αυτή περιέχει τις εξής λέξεις: "Τον", "Την", "Το", "Τα", "Τη", "Στη", "Στην", "Στο", "Ο", "Η", "Της", "Του", "Τους", "Οι", "Τις", "Στον".

IRRELEVANT_WORDS_LOWER: Η ίδια λίστα με την προηγούμενη αλλά με το πρώτο γράμμα της κάθε λέξης να είναι μικρό.

findAllOccurences(): Παίρνει σαν όρισμα το χωρίο κειμένου στο οποίο θα ψάξουμε για την απάντηση (text), τις λέξεις της απαντήσεις αφού έχουν φιλτραριστεί πρώτα (words) και μια μεταβλητή που δηλώνει εάν θέλουμε να χρησιμοποιήσουμε regular expressions ή όχι (useRegex). Καλεί την findOccurences() που περιγράφεται παρακάτω δύο φορές, μια με το words όπως είναι και μία με το words να είναι με όλα τα γράμματα μικρά. Προσθέτει τα δύο αποτελέσματα και επιστρέφει το άθροισμα.

findOccurences(): Παίρνει τα ίδια ορίσματα με την findAllOccurences(). Εάν το useRegex είναι ίσο με true τότε παίρνει τις λέξεις που περιέχονται μέσα στην απάντηση (words) και φτιάχνει ένα regular expression με αυτές παρεμβάλλοντας ενδιάμεσα οποιονδήποτε αριθμό από χαρακτήρες. Στην συνέχεια μετράει πόσες φορές εμφανίζεται αυτό το regular expression μέσα στο text. Εάν το useRegex είναι false τότε χρησιμοποιεί την countMatches() από το StringUtils που βρίσκεται στην βιβλιοθήκη commons-lang. Αυτή η συνάρτηση μετράει τον αριθμό των εμφανίσεων της συμβολοσειράς words μέσα στην συμβολοσειρά text και επιστρέφει τον αριθμό αυτό. Ο αιρθμός των εμφανίσεων είτε της μιας περίπτωσης είτε της άλλης επιστρέφεται στο τέλος.

removeSameFromBeggining(): Παίρνει σαν όρισμα έναν πίνακα από 4 συμβολοσειρές (στην περίπτωσή μας οι 4 απαντήσεις). Επιστρέφει έναν πίνακα από 4 συμβολοσειρές αφού πρώτα έχουν εξαφανιστεί από την αρχή των συμβολοσειρών τυχόν κοινές λέξεις. Για κάθε λέξη που περιέχεται στην πρώτη συμβολοσειρά ελέγχει μια-μια και τις υπόλοιπες 3 εάν ξεκινάνε με αυτή την λέξη. Εάν ναι τότε αφαιρεί την λέξη και από τις 4 συμβολοσειρές. Εάν όχι τότε τερματίζει και επιστρέφει τον πίνακα αφού δεν έχει νόημα να προχωρήσουμε στις επόμενες λέξεις.

removeSameFromEnd(): Παίρνει σαν όρισμα έναν πίνακα από 4 συμβολοσειρές (στην περίπτωσή μας οι 4 απαντήσεις). Επιστρέφει έναν πίνακα από 4 συμβολοσειρές αφού πρώτα έχουν εξαφανιστεί από το τέλος των συμβολοσειρών τυχόν κοινές λέξεις. Για κάθε λέξη που περιέχεται στην πρώτη συμβολοσειρά (ξεκινώντας από την τελευταία λέξη) ελέγχει μια-μια και τις υπόλοιπες 3 εάν τελειώνουν με αυτή την λέξη. Εάν ναι τότε αφαιρεί την λέξη και από τις 4 συμβολοσειρές. Εάν όχι τότε τερματίζει και επιστρέφει τον πίνακα αφού δεν έχει νόημα να προχωρήσουμε στις επόμενες λέξεις.

simplifyAnswers(): Παίρνει σαν όρισμα έναν πίνακα από 4 συμβολοσειρές (στην περίπτωσή μας οι 4 απαντήσεις). Επιστρέφει έναν πίνακα από 4 συμβολοσειρές αφού πρώτα έχουν εξαφανιστεί από τις συμβολοσειρές τυχόν λέξεις που είναι στην λίστα με τις άχρηστες λέξεις. Στην αρχή καλεί την removeSameFromBeggining() και μετά την removeSameFromEnd(). Μετά για κάθε μία από τις 4 συμβολοσειρές καλεί για κάθε λέξη που βρίσκεται στην λίστα με τις άχρηστες λέξεις IRRELEVANT_WORDS_UPPER και IRRELEVANT_WORDS_LOWER την συνάρτηση removeStart() του StringUtils η οποία αφαιρεί από ένα κείμενο μια λέξη εάν και μόνο εάν το κείμενο ξεκινάει με αυτήν την λέξη. Μας ενδιαφέρει να βγάλουμε από τις απαντήσεις τις άχρηστες λέξεις μόνο όταν βρίσκονται στην αρχή της απάντησης γιατί μετά αν τις βγάλουμε μπορεί να αλλοιώνουν το νόημα της απάντησης.

scoreAnswers(): Παίρνει σαν όρισμα το κείμενο της ερώτησης, την λίστα με τα GoogleResult που επέστρεψε η μονάδα QA, τον πίνακα με τις 4 πιθανές απαντήσεις και την μεταβλητή που δηλώνει εάν θέλουμε να χρησιμοποιήσουμε regular expressions ή όχι. Φτιάχνει έναν πίνακα 4 θέσεων όπου εκεί θα κρατηθούν οι βαθμολογίες της κάθε απάντησης (answerScores). Κάθε τιμή του πίνακα αυτού αρχικοποιείται με 0. Καλεί την simplifyAnswers() για να απλοποιήσει τις απαντήσεις. Για κάθε GoogleResult στην λίστα (εάν χρησιμοποιείται regular expressions ελέγχει και εάν έχουμε υπερβεί το όριο που ορίζει η REGEX_TEXT_NUMBER_LIMIT), ελέγχει πρώτα για κάθε απάντηση εάν βρίσκεται στον τίτλο. Εάν ναι τότε αυξάνει την βαθμολογία της απάντησης κατά FOUND_AT_TITLE_BONUS. Στην συνέχεια για κάθε απάντηση και για κάθε χωρίο κειμένου στην λίστα passages του GoogleResult καλεί την findAllOccurences() και προσθέτει τον αριθμό που επιστρέφει αυτή στην συνολική βαθμολογία της απάντησης. Στην συνέχεια εάν το κείμενο της ερώτησης περιέχει την αρνητική λέξη NEGATIVE_QUESTION_IDENTIFIER τότε πολλαπλασιάζει την βαθμολογία της κάθε απάντησης με -1. Τέλος επιστρέφει τον πίνακα με τις βαθμολογίες των απαντήσεων.

DecisionMaking.java

Υλοποιεί την μονάδα Λήψης Απόφασης (DM). Παίρνει τα αποτελέσματα που επέστρεψε η μονάδα QA και καλεί την μονάδα AS χωρίς την χρήση regular expressions. Εάν τα αποτελέσματα δεν είναι επαρκή για να αποφασίσει (εάν δηλαδή η μεγαλύτερη βαθμολογία διαφέρει από την δεύτερη καλύτερη βαθμολογία λιγότερο από ένα όριο που έχει οριστεί) τότε καλεί ξανά την AS αυτή τη φορά με την χρήση regular expressions. Εάν τα αποτελέσματα και πάλι δεν είναι επαρκή τότε ελέγχει ποιες βοήθειες έχουνε μείνει στο παιχνίδι και καλεί την αντίστοιχη βοήθεια (η σειρά που τις καλεί είναι «Κοινό», «Τηλέφωνο», «50/50»). Εάν δεν έχει μείνει καμία βοήθεια ελέγχει εάν βρίσκεται σε επίπεδο που μπορεί να ρισκάρει μια απάντηση χωρίς να χάσει λεφτά αφού έχει κλειδώσει μόλις ένα μαξιλαράκι (επίπεδα 6 κα 11) και δίνει μια απάντηση (αυτή με την μεγαλύτερη βαθμολογία). Αλλιώς αποχωρεί από το παιχνίδι εισπράττοντας τα χρήματα που έχει κερδίσει μέχρι τώρα. Εάν από την άλλη τα αποτελέσματα είναι επαρκή για να αποφασίσει δίνει την απάντηση που έχει την μεγαλύτερη βαθμολογία. Καθεμία από αυτές τις αποφάσεις έχει τον δικό της κωδικό ο οποίος επιστρέφεται πίσω στον διαχειριστή του παιχνιδιού που με την σειρά του τρέχει την αντίστοιχη συνάρτηση για να

εκτελέσει τις απαραίτητες ενέργειες. Ο αλγόριθμος που υλοποιήσαμε είναι αυτός που περιγράφουν οι συγγραφείς στο άρθρο [1] και που έχουν υλοποιήσει και οι ίδιοι στην δικιά τους DM μονάδα.

Ο κώδικας μπορεί να βρεθεί και στο Παράρτημα 2.

threshold: Το κατώτατο όριο που πρέπει να υπάρχει στην διαφορά ανάμεσα στην βαθμολογία της καλύτερης απάντησης και της δεύτερης καλύτερης απάντησης. Εάν η διαφορά αυτή είναι μικρότερη από αυτόν τον αριθμό τότε δεν μπορούμε να δώσουμε με ασφάλεια μια απάντηση. Έχει οριστεί ίσο με 1 γιατί αλλιώς ο εικονικός παίκτης δεν θα μπορέσει ποτέ να πάρει μια απόφαση.

GIVE_ANSWER: Ο κωδικός για να δοθεί μια απάντηση.

QUIT_GAME: Ο κωδικός για να παραιτηθεί ο παίκτης από το παιχνίδι.

USE_POLL_THE_AUDIENCE: Ο κωδικός για να χρησιμοποιηθεί η βοήθεια «Κοινό».

USE_PHONE_A_FRIEND: Ο κωδικός για να χρησιμοποιηθεί η βοήθεια «Τηλέφωνο».

USE_FIFTY_FIFTY: Ο κωδικός για να χρησιμοποιηθεί η βοήθεια «50/50».

makeDesicion(): Παίρνει σαν όρισμα την τρέχουσα ερώτηση του παιχνιδιού (q), την λίστα με τα αποτελέσματα GoogleResult που επιστρέφει η μονάδα QA (results), μια μεταβλητή που δείχνει εάν έχει χρησιμοποιηθεί η βοήθεια «50/50» (fiftyFiftyUsed), μια μεταβλητή που δείχνει εάν έχει χρησιμοποιηθεί η βοήθεια «Κοινό» (pollTheAudienceUsed) και μια μεταβλητή που δείχνει εάν έχει χρησιμοποιηθεί η βοήθεια «Τηλέφωνο» (phoneAFriendUsed). Επιστρέφει τον κωδικό της απόφασης που λαμβάνεται για να ενημερωθεί ο διαχειριστής του παιχνιδιού. Στην αρχή καλεί την scoreAnswers() του AnswerScoring χωρίς την χρήση regular expressions. Τον πίνακα με τα αποτελέσματα τον μετατρέπει σε μια λίστα από SortbleValues και τον σορτάρει σε φθίνουσα σειρά. Παίρνει την διαφορά του πρώτου στοιχείου της λίστας (του μεγαλύτερου δηλαδή) και του δεύτερου και ελέγχει εάν είναι μεγαλύτερη από το threshold. Εάν δεν είναι μεγαλύτερη τότε ξανακαλεί την scoreAnswers() του AnswerScoring με την χρήση regular expressions. Ταξινομεί ξανά τα αποτελέσματα. Βάζει ως selected απάντηση στο Question το πρώτο στοιχείο της λίστας από SortableValues (το index του στοιχείου αφού αυτό είναι που δείχνει ποια απάντηση είναι). Παίρνει ξανά την διαφορά του πρώτου στοιχείου της λίστας (του μεγαλύτερου δηλαδή) και του δεύτερου και ελέγχει εάν είναι μεγαλύτερη από το threshold. Εάν είναι μεγαλύτερη επιστρέφει GIVE_ANSWER. Εάν δεν είναι μεγαλύτερη τότε ελέγχει ποια βοήθεια δεν έχει χρησιμοποιηθεί με την σειρά που παρουσιάσαμε παραπάνω και επιστρέφει τον κωδικό της. Εάν έχουν χρησιμοποιηθεί όλες οι βοήθειες τότε ελέγχει εάν το επίπεδο της ερώτησης είναι ίσο με 6 ή 11 για να επιστρέψει GIVE_ANSWER. Εάν το επίπεδο της ερώτησης είναι κάποιο άλλο τότε επιστρέφει QUIT_GAME.

useFiftyFifty(): Παίρνει σαν όρισμα την τωρινή ερώτηση του παιχνιδιού Question. Βάζει ως selected απάντηση στο Question το πρώτο στοιχείο της λίστας fiftyFifty που βρίσκεται στο Question.

usePhoneAFriend(): Παίρνει σαν όρισμα την τωρινή ερώτηση του παιχνιδιού Question. Βάζει ως selected απάντηση στο Question αυτό που επιστρέφει η parseFriendsAnswer() του UtilityHelper που είδαμε παραπάνω όταν της δώσουμε σαν όρισμα το πεδίο friendCall του Question.

usePollTheAudience(): Παίρνει σαν όρισμα την τωρινή ερώτηση του παιχνιδιού Question. Βάζει ως selected απάντηση στο Question το πρώτο στοιχείο της λίστας votes που βρίσκεται στο Question.

GameManager.java

startVirtualPlayer(): Ξεκινά τον εικονικό παίκτη. Στην αρχή καλεί την συνάρτηση findResults() του QuestionAnswering όπως είδαμε και παραπάνω για να πάρει μια λίστα από GoogleResults. Εδώ αξίζει να σημειώσουμε ότι εάν το Wikipedia είναι ίσο με true τότε χρησιμοποιείται το DEFAULT_WIKIPEDIA_SPECIFIC_DOMAIN και το DEFAULT_WIKIPEDIA_ELEMENT_TAG που είδαμε παραπάνω αλλιώς null και στα δύο ορίσματα ώστε να πάρουν τις default τιμές. Στην συνέχεια περιμένει 2 δευτερόλεπτα πάλι με την χρήση ενός Timer και καλεί την συνάρτηση makeDecision() του DecisionMaking όπως είδαμε και παραπάνω για να πάρει την απόφαση του πήρε η μονάδα λήψης απόφασης. Τέλος με βάση τον κωδικό της απόφασης καλεί μια από τις παρακάτω συναρτήσεις που αντιστοιχεί σε αυτόν τον κωδικό. Πιο συγκεκριμένα το GIVE_ANSWER καλεί την συνάρτηση giveAnAnswer(), το QUIT_GAME την quitGame(), το USE_POLL_THE_AUDIENCE την usePollTheAudience(), το USE_PHONE_A_FRIEND την usePhoneAFriend() και το USE_FIFTY_FIFTY την useFiftyFifty(). Σε κάθε άλλη περίπτωση καλείται η quitGame().

useFiftyFifty(): Περιμένει 2 δευτερόλεπτα με την χρήση του Timer και πατάει αυτόματα το κουμπί «50/50» που βρίσκεται στο LifeLineFrame. Αφού γίνουν όλες οι απαραίτητες ενέργειες που γίνονται όταν πατιέται το κουμπί καλεί την giveAnAnswer() για να δώσει μια απάντηση.

usePhoneAFriend(): Περιμένει 2 δευτερόλεπτα με την χρήση του Timer και πατάει αυτόματα το κουμπί «Τηλέφωνο» που βρίσκεται στο LifeLineFrame. Αφού γίνουν όλες οι απαραίτητες ενέργειες που γίνονται όταν πατιέται το κουμπί καλεί την giveAnAnswer() για να δώσει μια απάντηση.

usePollTheAudience(): Περιμένει 2 δευτερόλεπτα με την χρήση του Timer και πατάει αυτόματα το κουμπί «Κοινό» που βρίσκεται στο LifeLineFrame. Αφού γίνουν όλες οι απαραίτητες ενέργειες που γίνονται όταν πατιέται το κουμπί καλεί την giveAnAnswer() για να δώσει μια απάντηση.

giveAnAnswer(): Παίρνει την απάντηση που έχει επιλέξει ο χρήστης (από 0 μέχρι 3) και μετά παίρνει το κουμπί από την λίστα buttons στο RadioListener που αντιστοιχεί σε αυτή την απάντηση και το διαλέγει και το κλικάρει. Στην συνέχεια περιμένει 2 δευτερόλεπτα και πατάει ο κουμπί «Κατοχυρώστε» που βρίσκεται στο QuestionWindowPanel.

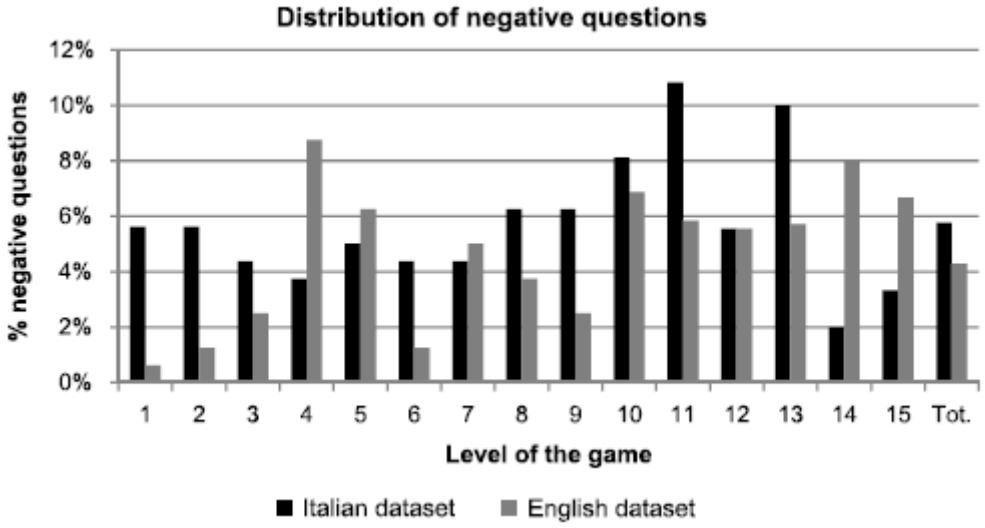
quitGame(): Περιμένει 2 δευτερόλεπτα με την χρήση του Timer και πατάει αυτόματα το κουμπί «Παραιτηθείτε» που βρίσκεται στο QuestionWindowPanel. Αφού γίνουν όλες οι απαραίτητες ενέργειες που γίνονται όταν πατιέται το κουμπί δεν κάνει τίποτα άλλο και η εφαρμογή τερματίζει.

9. ΠΕΙΡΑΜΑΤΙΚΗ ΑΞΙΟΛΟΓΗΣΗ

Ο στόχος της αξιολόγησης είναι διττός:

- να αξιολογήσει την αποτελεσματικότητα των μονάδων QA και AS στο να απαντήσουν στις ερωτήσεις του παιχνιδιού, και να συγκρίνει τα αποτελέσματα με εκείνα που επιτυγχάνονται με τους ανθρώπινους παίκτες. Το πείραμα συζητάται στην Ενότητα 9.1.
- να αξιολογήσει την ακρίβεια του εικονικού παίκτη στο να παίξει το παιχνίδι, λαμβάνοντας επίσης υπόψη τη στρατηγική που υλοποιείται από τη μονάδα λήψης αποφάσεων, και να συγκρίνει τα αποτελέσματα με εκείνα που επιτυγχάνονται με τους ανθρώπινους παίκτες. Το πείραμα συζητάται στην Ενότητα 9.2.

Οι συγγραφείς του άρθρου [1] χρησιμοποίησαν δύο σύνολα δεδομένων: ένα που περιέχει ερωτήσεις από την επίσημη ιταλική έκδοση του WWBM, και ένα που περιέχει ερωτήσεις από την αγγλική έκδοση. Και τα δύο σύνολα δεδομένων περιέχουν 1960 ερωτήσεις, υποδιαιρούνται σε 15 ομάδες, μία για κάθε επίπεδο του παιχνιδιού. Τα πρώτα 10 επίπεδα περιέχουν 160 ερωτήσεις το καθένα, ενώ τα επίπεδα 11, 12, 13, 14 και 15 περιέχουν 120, 90, 70, 50 και 30 ερωτήσεις, αντίστοιχα. 113 ερωτήσεις για την ιταλική έκδοση, και 84 για την αγγλική είναι σε αρνητική μορφή. Η κατανομή των αρνητικών ερωτήσεων ανά επίπεδο του παιχνιδιού αναφέρεται στο Σχήμα 7. Σε αυτή την υλοποίηση χρησιμοποιήθηκε ένα σύνολο δεδομένων για την ελληνική έκδοση του παιχνιδιού που βρέθηκε στο διαδίκτυο. Το σύνολο αυτό περιέχει 15.015 ερωτήσεις που υποδιαιρούνται σε 15 ομάδες, μία για κάθε επίπεδο του παιχνιδιού. Η κατανομή των ερωτήσεων σε κάθε επίπεδο είναι ομοιόμορφη, επομένως κάθε επίπεδο έχει 1001 ερωτήσεις. Κάποιες από τις ερωτήσεις χωρίς να είναι γνωστός ο ακριβής αριθμός (εξαιτίας του μεγάλου πλήθους των ερωτήσεων και του γεγονότος ότι το σύνολο βρέθηκε ήδη έτοιμο) είναι σε αρνητική μορφή. Διαχειρίζονται μόνο εκείνες οι ερωτήσεις που περιέχουν την λέξη κλειδί «δεν». Οι υπόλοιπες δεν λαμβάνονται υπόψιν.



Σχήμα 7: Κατανομή των αρνητικών ερωτήσεων σε κάθε επίπεδο του παιχνιδιού.

Ενώ υπάρχουν προηγούμενα αποτελέσματα για σύγκριση για την αγγλική έκδοση του παιχνιδιού (αν και όχι για το ίδιο σύνολο δεδομένων), οι συγγραφείς δεν έχουν επίγνωση προηγούμενων αποτελεσμάτων για την Ιταλική έκδοση, δηλαδή δεν έχουν καμία βασική γραμμή για να συγκρίνουν. Η μετρική που υιοθετήθηκε για την αξιολόγηση είναι η ακρίβεια (accuracy), δηλαδή η αναλογία των σωστών απαντήσεων, υπολογίζεται ως ο λόγος μεταξύ του αριθμού των σωστών απαντήσεων (n_c) και του συνολικού αριθμού των ερωτήσεων (n): ακρίβεια = n_c/n. Η σημασία των αποτελεσμάτων αξιολογείται χρησιμοποιώντας το τεστ McNemar με την εφαρμογή της διόρθωσης Bonferroni.

9.1 Πείραμα 1: αξιολόγηση της απόδοσης των μονάδων QA και AS.

Ο στόχος αυτού του πειράματος είναι να εκτιμηθεί η ακρίβεια των πέντε κριτηρίων για την βαθμολόγηση απάντησης, έχοντας ρυθμιστεί σωστά χρησιμοποιώντας τις παραμέτρους που περιγράφονται παραπάνω, και συγκεκριμένα:

- ο αριθμός των επεξεργασμένων κειμένων: οι συγγραφείς έλεγξαν 10 ρυθμίσεις, με τη χρήση των top-n, όπου n = 1, 2, 3, 4, 5, 10, 15, 20, 25, 30 κείμενα που επέστρεψε η μονάδα QA.
- η χρήση της βαθμολογίας w_i για τα κείμενα που επέστρεψε η μονάδα QA: έλεγξαν 2 ρυθμίσεις, με τη χρήση και χωρίς τη χρήση του βάρους για κάθε κείμενο που επέστρεψε η μονάδα QA.
- το επίπεδο γλωσσικής ανάλυσης που υιοθετήθηκε για την επεξεργασία των κειμένων που επέστρεψε η μονάδα QA: δοκίμασαν 3 ρυθμίσεις, που αντιπροσωπεύουν τα κείμενα χρησιμοποιώντας λέξεις-κλειδιά, μίσχους, ή λήμματα. Χρησιμοποίησαν επίσης και 2 επιπλέον ρυθμίσεις, με τη χρήση και χωρίς αφαίρεσης εξαιρούμενων λέξεων.
- η χρήση της επέκτασης ερωτήματος: έλεγξαν 2 ρυθμίσεις, με και χωρίς την επέκταση ερώτησης.

Συνολικά, έχουν ένα σύνολο από 1200 διαφορετικές ρυθμίσεις που χρησιμοποιούνται για να εκχωρήσουν μια βαθμολογία για κάθε μία από τις τέσσερις πιθανές απαντήσεις σε μια ερώτηση του παιχνιδιού: ξεκινώντας από αυτές τις βαθμολογίες, επιλέγεται η απάντηση με την υψηλότερη. Η μονάδα QA διαμορφώθηκε χρησιμοποιώντας όλα τα φίλτρα που αναφέρονται στο αντίστοιχο κεφάλαιο.

Ο Πίνακας 2 αναφέρει την ακρίβεια των 15 καλύτερων ρυθμίσεων, κατά μέσο όρο για το σύνολο των 1960 ερωτήσεων των ιταλικών και αγγλικών συνόλων δεδομένων που χρησιμοποίησαν οι συγγραφείς του άρθρου [1].

Αξίζει να σημειωθεί ότι η επικάλυψη και η πυκνότητα είναι τα καλύτερα κριτήρια απόδοσης τόσο για το ιταλικό όσο και για το αγγλικό σύνολο δεδομένων (Οι top-85 καλύτερες ρυθμίσεις λαμβάνονται με τη χρήση των εν λόγω κριτηρίων). Η ανάλυση των αποτελεσμάτων αποκαλύπτει τη χρησιμότητα του να λάβουμε υπόψη μας ένα σημαντικό αριθμό κειμένων, τη χρησιμότητα της χρησιμοποίησης της βαθμολογίας των κειμένων που επιστρέφονται από τη μονάδα QA, και την αποτελεσματικότητα της διαδικασίας απομάκρυνσης εξαιρούμενων λέξεων. Τέλος, η διαδικασία επέκτασης ερωτήματος φαίνεται να επηρεάζει αρνητικά την ακρίβεια του συστήματος. Το χειρότερο κριτήριο είναι ο Τίτλος Levenshtein και για τα ιταλικά και για τα αγγλικά: πράγματι, τα χειρότερα 100 αποτελέσματα προέκυψαν από το εν λόγω κριτήριο. Για να έχουμε μια σαφή εικόνα της ακρίβειας του συνόλου των ρυθμίσεων, ο Πίνακας 3, παρουσιάζει για κάθε κριτήριο, την καλύτερη και τη χειρότερη ρύθμιση.

Πίνακας 2: Απόδοση των 15 καλύτερων ρυθμίσεων (κατά μέσο όρο για το σύνολο των ερωτήσεων). Ακρωνύμια: P αριθμός των κειμένων, Lex χρήση λεξικοποίησης, KWD λέξεις-κλειδιά, LEM λήμματα, ST μίσχοι, SCORE χρήση της βαθμολογίας του κειμένου, SW χρήση της αφαίρεσης εξαιρουμένων λέξεων, QE χρήση της επέκτασης ερωτήματος.

Rank	Criterion	P	Lex	Score	SW	QE	Accuracy
Italian dataset							
1	Overlap	25	ST	Y	Y	N	64.29%
2	Overlap	25	LEM	Y	Y	N	64.29%
3	Density	3	KWD	Y	N	Y	64.03%
4	Density	30	ST	Y	Y	N	64.03%
5	Density	30	LEM	Y	Y	N	64.03%
6	Overlap	20	ST	Y	Y	N	63.78%
7	Overlap	20	LEM	Y	Y	N	63.78%
8	Overlap	30	ST	Y	Y	N	63.78%
9	Overlap	30	LEM	Y	Y	N	63.78%
10	Density	20	ST	Y	Y	N	63.27%
11	Density	20	LEM	Y	Y	N	63.27%
12	Density	25	KWD	Y	Y	N	63.01%
13	Overlap	15	ST	Y	Y	N	62.76%
14	Overlap	15	LEM	Y	Y	N	62.76%
15	Overlap	20	ST	N	Y	N	62.76%
English dataset							
1	Overlap	25	LEM	Y	Y	N	59.47%
2	Overlap	25	ST	Y	Y	N	59.38%
3	Density	3	KWD	Y	N	Y	59.26%
4	Overlap	20	ST	Y	Y	N	59.22%
5	Density	30	ST	Y	Y	N	59.08%
6	Density	30	LEM	Y	Y	N	59.08%
7	Overlap	30	ST	Y	Y	N	58.99%
8	Density	20	ST	Y	Y	N	58.84%
9	Overlap	15	LEM	Y	Y	N	58.72%
10	Density	25	KWD	Y	Y	N	58.37%
11	Overlap	30	LEM	Y	Y	N	58.35%
12	Density	20	LEM	Y	Y	N	58.21%
13	Overlap	20	LEM	Y	Y	N	58.14%
14	Overlap	20	ST	N	Y	N	57.99%
15	Overlap	15	ST	Y	Y	N	57.97%

Πίνακας 3: Η καλύτερη και η χειρότερη απόδοση για κάθε κριτήριο με την αντίστοιχη ρύθμισή του (κατά μέσο όρο για το σύνολο των ερωτήσεων). Ακρωνύμια: P αριθμός των κειμένων, Lex χρήση λεξικοποίησης, Score χρήση της βαθμολογίας του κειμένου, SW χρήση της αφαίρεσης εξαιρουμένων λέξεων, QE χρήση της επέκτασης ερωτήματος, ES ακριβές κομμάτι κειμένου, LCS (Longest Common Subsequence) Μέγιστη κοινή συμβολοσειρά, TL τίτλος Levenshtein, KWD λέξεις-κλειδιά, LEM λήμματα, ST μίσχοι.

Rank	Criterion	P	Lex	Score	SW	QE	Accuracy
Italian dataset							
BEST	Overlap	25	ST	Y	Y	N	64.29%
WORST	Overlap	1	KWD	Y	N	N	42.60%
BEST	Density	3	KWD	Y	N	Y	64.03%
WORST	Density	1	LEM	N	N	N	43.37%
BEST	ES	30	KWD	Y	Y	N	59.18%
WORST	ES	1	KWD	Y	N	N	42.09%
BEST	LCS	3	KWD	Y	N	Y	57.14%
WORST	LCS	1	LEM	Y	Y	N	41.07%
BEST	TL	1	KWD	Y	Y	Y	40.05%
WORST	TL	1	LEM	Y	N	N	20.15%
English dataset							
BEST	Overlap	25	LEM	Y	Y	N	59.47%
WORST	Overlap	1	KWD	Y	N	N	37.74%
BEST	Density	3	KWD	Y	N	Y	59.26%
WORST	Density	1	ST	Y	N	N	38.58%
BEST	ES	30	KWD	Y	Y	N	54.62%
WORST	ES	1	KWD	Y	N	N	37.46%
BEST	LCS	3	KWD	Y	N	Y	52.19%
WORST	LCS	1	LEM	Y	Y	N	36.04%
BEST	TL	1	KWD	Y	Y	Y	35.17%
WORST	TL	1	LEM	N	N	N	15.12%

Παρατηρούμε ότι οι καλύτερες και οι χειρότερες ρυθμίσεις του κάθε κριτηρίου για την ιταλική και την αγγλική γλώσσα είναι λίγο πολύ το ίδιο, ακόμη και αν η ακρίβεια που λαμβάνεται στο αγγλικό σύνολο δεδομένων είναι 5% χαμηλότερη από εκείνη που

λαμβάνεται για το ιταλικό, κατά μέσο όρο. Στατιστικές δοκιμές δείχνουν ότι δεν υπάρχει σημαντική διαφορά μεταξύ των καλύτερων ρυθμίσεων αλληλοεπικάλυψης και των καλύτερων ρυθμίσεων της πυκνότητας, αλλά η ακρίβειά τους είναι σημαντικά καλύτερη από τις καλύτερες ρυθμίσεις των άλλων κριτηρίων, ανεξάρτητα από τη γλώσσα ($P < 0,01$).

Ξεκινώντας από την ακρίβεια των μεμονωμένων ρυθμίσεων, προσπαθήσαμε να τα συνδυάσουμε με σκοπό να βελτιώσουμε τη συνολική ακρίβεια της βαθμολόγησης απαντήσεων. Σε αντίθεση με τον άπληστο συνδυασμό που χρησιμοποιήθηκε στο [38], στην εργασία αυτή [1] οι συγγραφείς προτείνουν ένα συνδυασμό που βασίζεται σε μια μάθηση κατά σημείο για να ταξινομήσουμε προσεγγίσεις με τη χρήση αλγορίθμων παλινδρόμησης [30], στην οποία τα ζεύγη (q, α) ερώτησης/απάντησης επισημαίνονται με τις αποφάσεις ενδιαφέροντος της απάντησης a σε σχέση με το ερώτημα q. Στο σκηνικό τους, η σωστή απάντηση σε ερώτηση του παιχνιδιού είναι χαρακτηρισμένη με την απόφαση ενδιαφέροντος 1, ενώ οι άλλες τρείς λανθασμένες απαντήσεις επισημαίνονται με 0. Κάθε παράδειγμα εκπαίδευσης αντιπροσωπεύεται χρησιμοποιώντας ένα διάνυσμα που αποτελείται από το επίπεδο της ερώτησης (από 1 έως 15), και όλες τις 1200 μονές βαθμολογίες που λαμβάνονται με τις προαναφερθείσες ρυθμίσεις. Έχουν επιλέξει τη χρήση τυχαίου δάσους (Random Forest) (RF) [8], μια μέθοδο συνολικής μάθησης, που συνδυάζει διαφορετικές ερμηνευτικές μεταβλητές και φτιάχνει δέντρα από διαφορετικά δείγματα των δεδομένων κατάρτισης και τυχαία υποσύνολα των χαρακτηριστικών των δεδομένων. Το τελικό αποτέλεσμα επιτυγχάνεται παίρνοντας το μέσο όρο των εξόδων των ενιαίων δέντρων. Η χρήση διαφορετικών δειγμάτων δεδομένων από την ίδια διανομή και διαφορετικών συνόλων χαρακτηριστικών για την εκμάθηση του κάθε ατομικού δέντρου εμποδίζει την υπερπροσαρμογή (overfitting). Υιοθέτησαν την υλοποίηση που παρέχεται από την βιβλιοθήκη RankLib.

Οι ερωτήσεις σε κάθε σύνολο δεδομένων χωρίζονται σε ένα σύνολο εκπαίδευσης Tr, και μια σειρά δοκιμών Ts. Η μεθοδολογία που υιοθετήθηκε για την απόκτηση των Tr και Ts ήταν η στρωματοποιημένη (stratified) 5-φορές διασταυρωμένη επικύρωση (cross validation), όπου η διαδικασία διαστρωμάτωσης διασφαλίζει ότι κάθε πτυχή περιέχει την ίδια κατανομή των ερωτήσεων για τα διαφορετικά επίπεδα του παιχνιδιού. Δεδομένου του μεγέθους του κάθε συνόλου δεδομένων (1960 ερωτήσεις), εφαρμόζοντας 5 φορές διασταυρωμένη επικύρωση σημαίνει ότι το σύνολο των δεδομένων χωρίζεται σε 5 ασυνεχή κομμάτια, που το καθένα περιέχει 392 ερωτήσεις. Το πείραμα εκτελέστηκε σε 5 βήματα. Σε κάθε βήμα, 4 κομμάτια χρησιμοποιήθηκαν ως σύνολο εκπαίδευσης Tr (1568 ερωτήσεις), ενώ το υπόλοιπο κομμάτι χρησιμοποιήθηκε ως σύνολο ελέγχου Ts. Τα στάδια επαναλήφθηκαν μέχρι καθένα από τα 5 ασυνεχή κομμάτια είχε χρησιμοποιηθεί ως Ts, και τα αποτελέσματα κατά μέσο όρο σε 5 σειρές. Για να συντονίσουν τις παραμέτρους της μάθησης σε κατάταξη (αριθμός δέντρων, ρυθμός μάθησης, ρυθμός υποδειγματοληψίας), χρησιμοποίησαν επίσης ένα σύνολο επικύρωσης (validation set), το οποίο λαμβάνεται με δειγματοληψία ερωτήσεων από το σύνολο εκπαίδευσης του κάθε γύρου. Πήραν δείγμα 12,5% του συνόλου εκπαίδευσης (196 ερωτήσεις), και ακόμη και σε αυτή την περίπτωση έλαβαν υπόψη την κατανομή των ερωτήσεων μεταξύ των διαφόρων επιπέδων (διαστρωμάτωση). Σε κάθε βήμα, το σύνολο εκπαίδευσης περιέχει 1372 ερωτήσεις, το σύνολο επικύρωσης περιείχε 196 ερωτήσεις και οι υπόλοιπες 392 ερωτήσεις που χρησιμοποιούνται ως σύνολο ελέγχου. Η τελική ακρίβεια που επιτυγχάνεται συνδυάζοντας όλες τις επιμέρους ρυθμίσεις μέσω της στρατηγικής της μάθησης σε κατάταξη είναι ίσο με 79,64% για τα Ιταλικά, και 76.41% για τα Αγγλικά (κατά μέσο όρο κατά τη διάρκεια των πέντε τρεξιμάτων), το οποίο είναι σημαντικά καλύτερο ($p << 0,0001$) από ό,τι η ακρίβεια της καλύτερης μεμονωμένης ρύθμισης (64,29% για τα Ιταλικά, 59.47% για τα Αγγλικά). Το αποτέλεσμα

που προκύπτει για τα αγγλικά είναι παρόμοιο με εκείνο που επιτυγχάνεται στο [26], στο οποίο το σύστημα απάντησε σωστά στο περίπου 75% των ερωτήσεων, ακόμη και αν χρησιμοποιήθηκε ένα διαφορετικό και μικρότερο σύνολο δεδομένων. Περισσότερες λεπτομέρειες σχετικά με την ακρίβεια για κάθε διαφορετικό επίπεδο δυσκολίας του παιχνιδιού παρουσιάζεται και συζητείται στην Ενότητα 9.1.3.

9.1.1 Αναπάντητα ερωτήματα και ανάλυση σφαλμάτων

Για να κατανοήσουμε τις ερωτήσεις για τις οποίες το σύστημα δεν είναι σε θέση να παρέχει μια σωστή απάντηση, τις κατατάξαμε σε συγκεκριμένες κατηγορίες, και πραγματοποιήσαμε μια σωστή ανάλυση των σφαλμάτων. Η ακόλουθη λίστα περιέχει κατηγορίες ερωτήσεων που παραμένουν αναπάντητες από το σύστημα του άρθρου [1] και κατά συνέπεια και από το δικό μας αφού προσπαθεί να το μιμηθεί:

- Ερωτήσεις σχετικά με έννοιες που δεν υπάρχουν στη Wikipedia / DBpedia: κάποια από αυτά τα ερωτήματα αφορούν την αστρολογία, παροιμίες και ρήσεις, και τη θρησκεία.
- Ερωτήσεις σχετικά με έννοιες που δεν είναι σαφείς στη Wikipedia / DBpedia: το σύστημα μπορεί να μην είναι σε θέση να δώσει μια απάντηση σε ορισμένες ερωτήσεις, ακόμα κι αν οι απαραίτητες πληροφορίες περιέχονται στις πηγές γνώσης. Για παράδειγμα, η ερώτηση «Ποιός από αυτούς τους ηθοποιούς δεν είναι γιος ενός πατέρα που είναι ηθοποιός?») απαιτεί να βρούμε την έννοια «ηθοποιός γιος ενός πατέρα ηθοποιού», η οποία δεν είναι σαφής στη Wikipedia / DBpedia.
- Ερωτήσεις των οποίων οι απαντήσεις αφορούν ειδικά τους αριθμούς, τις χρονικές περιόδους, και μαθηματικούς υπολογισμούς: το σύστημα αδυνατεί να παράσχει μια απάντηση όταν περιέχει ρωμαϊκούς αριθμούς, ή να υπονοεί τον υπολογισμό των χρονικών πλαισίων, ή κάποια μαθηματική πράξη. Για παράδειγμα, οι υποψήφιες απαντήσεις στο ερώτημα «Σε ποιόν αιώνα φτιάχτηκε το πρώτο στυλό?» είναι Α) XX Β) XVII Γ) XVIII D) XIX, αλλά το σύστημα δεν μπορεί να βρει ποια είναι η πιο σωστή από τις υποψήφιες απαντήσεις. Το ερώτημα «Πόσο καιρό ήταν ο William Harrison ένατος πρόεδρος των ΗΠΑ?» περιλαμβάνει τον υπολογισμό του χρονικού πλαισίου που το σύστημα δεν είναι σε θέση να πραγματοποιήσει. Το ερώτημα «Πόσες μοίρες είναι η καθεμία από τις άλλες δύο γωνίες ενός ισοσκελούς τριγώνου, αν κάποια γωνία είναι 120^{o} ?» απαιτεί ότι το σύστημα γνωρίζει ότι το σύνολο όλων των γωνιών σε κάθε Ευκλείδειο τρίγωνο είναι ίσο με 180^{o}.
- Ερωτήσεις που απαιτούν γνώση της γλώσσας: ερωτήσεις όπως «Στη Λατινική γλώσσα, η κλητική πληθυντικού είναι πάντα ίση με τι?», θα απαιτούσε ειδικές γνώσεις που δεν κωδικοποιούνται στη Wikipedia / DBpedia.
- Ερωτήσεις που χρειάζονται να κάνουν μια σύγκριση: ερωτήσεις όπως «Ποια από αυτές τις περιοχές της Ιταλίας έχει τη χαμηλότερη επιφάνεια?» θα απαιτούσε να κάνει μια σύγκριση και την επιλογή της ελάχιστης τιμής.
- Ερωτήσεις που απαιτούν γνώση των οπτικών ιδιοτήτων: ερωτήσεις όπως «Ποια από αυτές τις ομάδες ποδοσφαίρου δεν έχει λύκο ως σύμβολο?» απαιτούν τη γνώση οπτικών ιδιοτήτων.

Πίνακας 4: Ανάλυση σφάλματος για Ιταλικά και Αγγλικά.

Error type	%
Questions whose concepts do not occur in Wikipedia/DBpedia	13%
Questions whose concepts are not explicit in Wikipedia/DBpedia	15%
Questions which involve numbers, time and math	13%
Questions which require language proficiency	14%
Questions which require to make a comparison	7.50%
Questions which require knowledge about visual properties	7.50%
Other errors	30%

Η ανάλυση σφάλματος που διεξάγεται στα αποτελέσματα των πειραμάτων στο προηγούμενο τμήμα απεικονίζεται στον Πίνακα 4. Το 30% των άλλων σφαλμάτων οφείλονται σε λάθος βαθμολόγησης των απαντήσεων ή σε άλλους παράγοντες, όπως τα ευρετικά για να αναγνωρίζουμε και να διαχειριζόμαστε αρνητικές ερωτήσεις. Ως εκ τούτου, προκειμένου οι συγγραφείς να αξιολογήσουν πόσο αποτελεσματική είναι αυτή η στρατηγική, έχουν αξιολογήσει από την μια την ακρίβεια των κανονικών εκφράσεων στο να ανιχνεύουν σωστά αρνητικές ερωτήσεις, και από την άλλη πλευρά, την αποτελεσματικότητα της μεθόδου αντίστροφης βαθμολόγησης. Η ακρίβεια των ευρετικών που βασίζονται στη χρήση των κανονικών εκφράσεων είναι F_1 = 89,23% για το ιταλικό σύνολο δεδομένων, και F_1 = 98.20% για το αγγλικό, ενώ η μέθοδος αντίστροφης βαθμολόγησης αξιολογήθηκε λαμβάνοντας υπόψη το ποσοστό των αρνητικών ερωτήσεων για τις οποίες η στρατηγική αντίστροφης βαθμολόγησης ήταν χρήσιμη (δηλαδή η απάντηση με το χαμηλότερο σκορ ήταν σωστή), επιβλαβής (δηλαδή η απάντηση με το χαμηλότερο σκορ ήταν λάθος) ή αδιάφορο (δηλαδή η σωστή απάντηση δεν είναι ούτε αυτή με την υψηλότερη βαθμολογία, ούτε αυτή με το χαμηλότερο σκορ): τα ποσοστά του ιταλικού συνόλου δεδομένων είναι 80%, 9%, και 11%, αντίστοιχα, ενώ τα ποσοστά για το αγγλικό σύνολο δεδομένων είναι 96%, 1% και 3%, αντίστοιχα. Αυτό σημαίνει ότι η μέθοδος αντίστροφης βαθμολόγησης είναι αποτελεσματική στην πράξη.

Τέλος, προκειμένου να έχουν κάποιες λεπτομέρειες σχετικά με το όφελος της DBpedia, έχουν υπολογίσει τον αριθμό των ερωτήσεων για τις οποίες η μονάδα QA επέστρεψε κείμενα που προέρχονται από την DBpedia. Συγκεκριμένα, 290 ερωτήσεις για το ιταλικό σύνολο δεδομένων, και 293 για το αγγλικό βασίζονται σε κείμενα που προέρχονται από την DBpedia (εκτός από αυτά που προέρχονται από την Wikipedia), και αυτό σημαίνει ότι η απάντηση για εκείνες τις ερωτήσεις θα μπορούσε ενδεχομένως να εξαχθεί από την DBpedia. Για λόγους πληρότητας έχουν αξιολογήσει επίσης την ακρίβεια του ταξινομητή Rocchio που περιγράφεται στην ενότητα 5.1, το οποίο επιτρέπει να χαρτογραφήσουμε διάφορες λημματοποιήσεις των ερωτήσεων ζητώντας για μια συγκεκριμένη ιδιότητα την αντίστοιχη ιδιότητα της DBpedia. Η ακρίβεια του ταξινομητή, χρησιμοποιώντας το «αφήνουμε ένα έξω» (leave-one-out) είναι 85,59% για το αγγλικό σύνολο δεδομένων, και 83,57% για το ιταλικό.

9.1.2 Δοκιμές Κατάλυσης (ablation)

Για να αποκτήσουν οι συγγραφείς γνώσεις σχετικά με τη δύναμη των διαφόρων παραμέτρων που χρησιμοποιούνται για να ρυθμίσουν κάθε κριτήριο της βαθμολόγησης απάντησης, πραγματοποίησαν επιλογή χαρακτηριστικών με τη χρήση δοκιμών κατάλυσης, αφαιρώντας μονά χαρακτηριστικά ή ομάδες χαρακτηριστικών.

Όσον αφορά την κατάλυση ενός ενιαίου χαρακτηριστικού, εκπαίδευσαν ένα διαφορετικό μοντέλο με το να αφαιρέσουν κάθε χαρακτηριστικό που σχετίζεται με κάθε μεμονωμένη ρύθμιση και να μετρήσουν την αντίστοιχη προγνωστική ακρίβεια του μοντέλου της μάθησης σε κατάταξη (δηλαδή η διαδικασία επαναλαμβάνεται 1200 φορές). Η ανάλυση των αποτελεσμάτων αποκαλύπτει ότι μόνο στις 160 από τις 1200 φορές η ακρίβεια των μοντέλων μάθησης είναι χαμηλότερη από εκείνη που λαμβάνεται με την καλύτερη ρύθμιση, ανεξάρτητα από τη γλώσσα που χρησιμοποιήθηκε. Η μέγιστη μείωση της ακρίβειας είναι 4,84% για τα Ιταλικά, και 5,00% για τα αγγλικά. Είναι ενδιαφέρον να σημειωθεί ότι σε κάθε ένα από αυτά τα 160 μοντέλα το χαρακτηριστικό που αντιστοιχεί στην υιοθέτηση μιας ρύθμισης λέξεις-κλειδιών για την εκπροσώπηση κειμένων αφαιρείται. Ως εκ τούτου, το σήμα που έφεραν οι λέξεις-κλειδιά είναι πιο σχετικό από εκείνο που ήρθε από μίσχους και λήμματα.

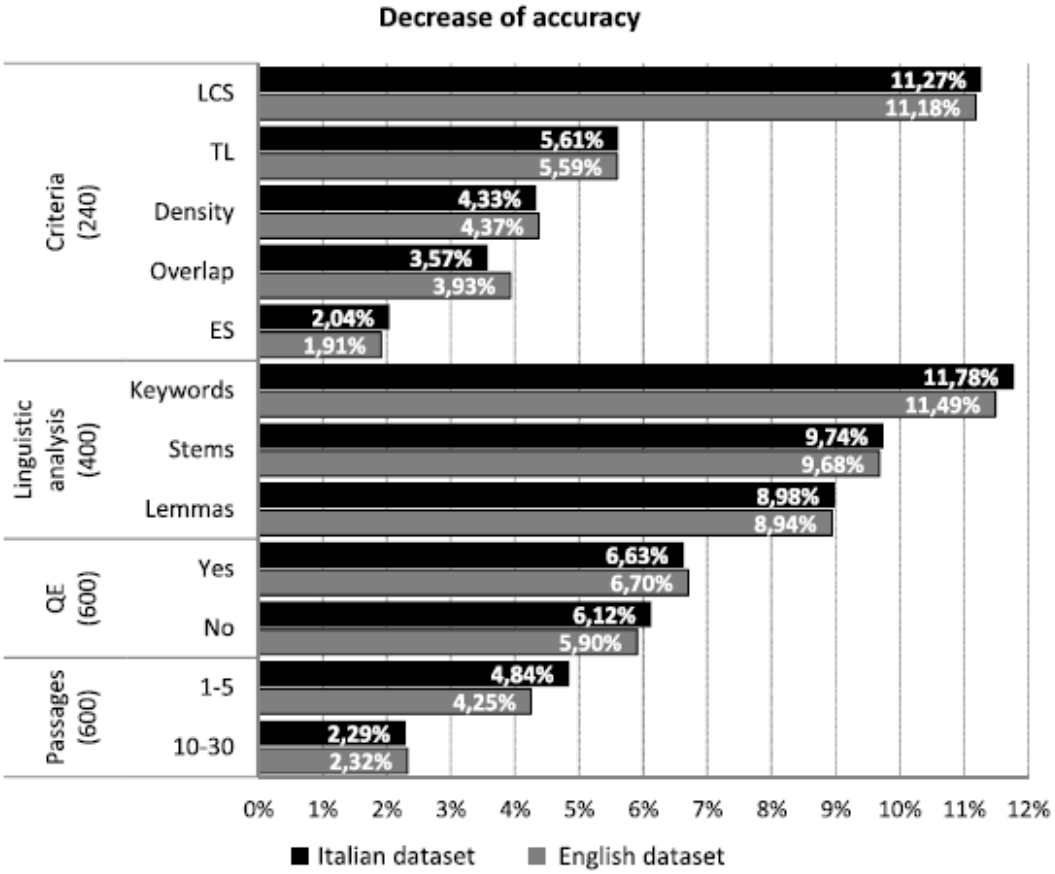
Πραγματοποίησαν επίσης δοκιμές κατάλυσης αφαιρώντας ομάδες χαρακτηριστικών. Όρισαν τις ακόλουθες δώδεκα διαφορετικές ομάδες:

- Πέντε ομάδες, εκάστη των οποίων αντιστοιχεί σε ένα διαφορετικό κριτήριο βαθμολόγησης απαντήσεων, δηλαδή κάθε ομάδα περιέχει όλα τα χαρακτηριστικά που αντιστοιχούν σε όλες τις ρυθμίσεις του κάθε επιμέρους κριτήριου.
- Τρεις ομάδες, εκάστη των οποίων αντιστοιχεί σε ένα διαφορετικό επίπεδο γλωσσικής ανάλυσης που θεσπίζεται για την επεξεργασία των χωρίων κειμένου που επέστρεψε η μονάδα QA.
- Δύο ομάδες, εκάστη των οποίων περιέχει όλες τις ρυθμίσεις με τη χρήση και χωρίς της στρατηγικής επέκτασης ερώτησης.
- Δύο ομάδες, εκάστη των οποίων περιέχει όλες τις ρυθμίσεις χρησιμοποιώντας έναν αριθμό κειμένων από 1 έως 5, και από 10 έως 30, αντίστοιχα.

Η Εικόνα 8 αναφέρει τη μείωση της ακρίβειας που επιτυγχάνεται με την εκπαίδευση διαφόρων μοντέλων από την αφαίρεση κάθε ενιαίας ομάδας χαρακτηριστικών. Ομάδες με υψηλότερες τιμές είναι καλύτερες, πράγμα που σημαίνει ότι αυτά τα χαρακτηριστικά έχουν μεγαλύτερο αντίκτυπο στη συνολική απόδοση του μοντέλου, δεδομένου ότι η κατάλυσή τους οδηγεί σε υψηλότερη μείωση της ακρίβειας. Όλα τα αποτελέσματα είναι στατιστικά σημαντικά όταν συγκρίνονται με την καλύτερη ρύθμιση που λαμβάνεται με την στρατηγική εκμάθησης σε βαθμό ($p << 0.0001$).

Όσον αφορά τις ομάδες που αντιστοιχούν στα κριτήρια βαθμολόγησης απάντησης, η καλύτερη απόδοση λαμβάνεται με LCS, που ακολουθείται από TL, Πυκνότητα, Επικάλυψη και ES. Μια σύγκριση με την απόδοση των μεμονωμένων ρυθμίσεων που αναφέρονται στον Πίνακα 2 και τον Πίνακα 3, δείχνει ότι τα κριτήρια Επικάλυψης και Πυκνότητας είναι οι καλύτερες επιδόσεις ξεχωριστά, αλλά όταν αφαιρούνται μοναδικά, η συνολική απόδοση δεν είναι σημαντικά διαφορετική. Αυτό υποστηρίζει τη διαπίστωση ότι το σήμα που φέρουν είναι πολύ αλληλεπικαλυπτόμενο. Από την άλλη πλευρά, το κριτήριο TL ήταν το χειρότερο όταν εκτελείται ατομικά, αλλά, όταν αφαιρεθεί ξεχωριστά, αυτό οδηγεί σε μια μείωση της συνολικής ακρίβεια μεγαλύτερη από 5%. Όσον αφορά τη γλωσσική ανάλυση που υιοθετήθηκε για την επεξεργασία των κειμένων που επέστρεψε η μονάδα QA, οι αναπαραστάσεις που βασίζονται σε λέξεις έχουν την καλύτερη απόδοση, ακολουθούμενη από αυτές με στελέχη και λήμματα που συμπεριφέρονται με παρόμοιο τρόπο. Αυτό είχε ήδη παρατηρηθεί από την αφαίρεση κάθε μιας ρύθμισης τη φορά, όπου η κατάλυση συνθέσεων που χρησιμοποιούν αναπαραστάσεις που βασίζονται σε μίσχους ή λήμματα δεν οδήγησε σε μείωση της ακρίβειας. Όσον αφορά τον μηχανισμό επέκτασης ερώτησης, οι μέθοδοι που το υιοθετούν δείχνουν μια ελαφρώς καλύτερη απόδοση (λιγότερο από 1%) από ό, τι εκείνοι που δεν το υιοθετούν, και αυτό είναι σύμφωνο με την απόδοση των μεμονωμένων ρυθμίσεων, στις οποίες

παρατηρήσαμε ότι η επέκταση ερωτήματος δεν φαίνεται να έχει επίπτωση στην ακρίβεια του συστήματος. Τέλος, οι ρυθμίσεις που χρησιμοποιούν έναν μικρότερο αριθμό κειμένων (από 1 έως 5) είναι καλύτερες από εκείνες που χρησιμοποιούν περισσότερα κείμενα (από 10 έως 30). Αυτό φαίνεται να αντικρούει τη διαπίστωση που προήλθε από την ανάλυση των top-15 ρυθμίσεων (Πίνακας 2), όπου η καλύτερη απόδοση προήλθε χρησιμοποιώντας 15 έως 30 κείμενα. Μια πιθανή ερμηνεία είναι ότι η υπάρχουσα επικάλυψη πληροφοριών χρησιμοποιώντας 15 έως 30 περάσματα είναι υψηλότερη από εκείνη χρησιμοποιώντας 1 έως 5 κείμενα, και αυτό οδήγησε τον πρώτο συνδυασμό να είναι λιγότερο αποτελεσματικός από τον δεύτερο.



Σχήμα 8: Μείωση της ακρίβειας από την κατάλυση ομάδας χαρακτηριστικών. Για κάθε ομάδα, ο αριθμός των χαρακτηριστικών που αφαιρούνται αναφέρεται επίσης.

9.1.3 Ανάλυση απόδοσης συστήματος και ανθρώπινης απόδοσης ανά επίπεδο

Οι συγγραφείς σύγκριναν την απόδοση της μονάδας QA και AS με εκείνη των ανθρώπινων παικτών για να δώσουν απαντήσεις σε ερωτήσεις από διαφορετικά επίπεδα δυσκολίας.

Το να παίξεις με επιτυχία το παιχνίδι WWBM σε μεγάλο βαθμό εξαρτάται από τη γνώση του παίκτη για το λαϊκό πολιτισμό, ως εκ τούτου, η σύγκριση με τους ανθρώπινους παράγοντες εκτελείται μόνο για το ιταλικό σύνολο δεδομένων. Αντίστοιχα από εμάς η σύγκριση γίνεται με το ελληνικό σύνολο δεδομένων καθώς όλοι οι παίχτες που χρησιμοποιήσαμε ήταν από την Ελλάδα. Για το σκοπό αυτό εμπλέκονται 98 ανθρώπινοι παίκτες, που επιλέχτηκαν με τη στρατηγική δειγματοληψίας διαθεσιμότητας [47] (Ιταλοί φοιτητές ή απόφοιτοι). Εμείς διαλέξαμε 20 ανθρώπινους παίχτες να παίξουν το παιχνίδι

προσπαθώντας να εφαρμόσουμε την στρατηγική δειγματοληψίας διαθεσιμότητας (οι παίχτες που επιλέχθηκαν ήταν είτε εργαζόμενοι στην εταιρεία «Τεχνολογίες ABAKA Α.Ε.» είτε μεταπτυχιακοί φοιτητές στο τμήμα Πληροφορικής και Τηλεπικοινωνιών ΕΚΠΑ). Το σύνολο δεδομένων των 1960 ιταλικών ερωτήσεων διαιρέθηκε τυχαία σε 98 ξένα σύνολα των 20 ερωτήσεων το καθένα. Κάθε σύνολο ανατέθηκε σε διαφορετικό χρήστη, ο οποίος έδινε απαντήσεις χωρίς να έχει τη δυνατότητα να συμβουλεύεται το Web ή άλλες πηγές γνώσης (το επίπεδο της κάθε ερώτησης δεν γνωστοποιείται προς τους χρήστες). Στην δικιά μας υλοποίηση οι ερωτήσεις που δίνονταν στον κάθε παίκτη διαλέγονταν με τυχαίο τρόπο από την εφαρμογή, δηλαδή δεν διαιρέσαμε τις ερωτήσεις σε σύνολα. Επίσης οι παίχτες δεν είχαν ούτε σε εμάς την δυνατότητα να συμβουλεύονται το Web ή άλλες πηγές γνώσης, ενώ το επίπεδο της κάθε ερώτησης τους γνωστοποιούνταν καθώς γνώριζαν σε ποιο επίπεδο βρίσκονταν κάθε χρονική στιγμή.

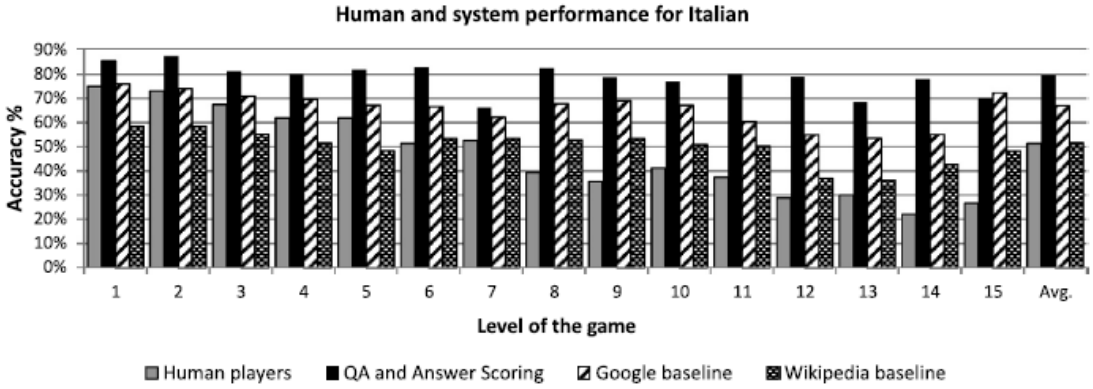
Οι συγγραφείς του άρθρου [1] ως βασική γραμμή υπέβαλαν στην Google κάθε ερώτηση και πήραν τα top-30 αποσπάσματα κειμένου που επέστρεψε η μηχανή αναζήτησης. Ως εκ τούτου, υπολόγισαν μια βαθμολογία για κάθε υποψήφια απάντηση πολλαπλασιάζοντας τον αριθμό των φορών που η απάντηση βρισκόταν στο κάθε απόσπασμα με το αντίστροφο της τάξης του αποσπάσματος. Αυτό περίπου κάναμε και εμείς στην δικιά μας υλοποίηση χωρίς όμως να πολλαπλασιάζουμε τον αριθμό των φορών που βρήκαμε την κάθε απάντηση με την τάξη του αποσπάσματος. Στην δική μας υλοποίηση χρησιμοποιούμε μόνο τον αριθμό των φορών για να βαθμολογήσουμε την κάθε απάντηση. Τέλος, έχουν επιλέξει την υποψήφια απάντηση με την υψηλότερη βαθμολογία (Μια τυχαία επιλογή υιοθετήθηκε για να σπάσει τις ισοπαλίες). Αναφέρονται σε αυτό ως βασική γραμμή Google (Google baseline). Αυτή την βασική γραμμή Google υλοποιήσαμε και εμείς σε αυτή την εργασία με κάποιες παραλλαγές.

Καθώς η Google ψάχνει ολόκληρο το Web, ενώ οι συγγραφείς χρησιμοποιούν πληροφορίες που είναι διαθέσιμες μόνο στην Wikipedia, αποφάσισαν να συγκρίνουν τα αποτελέσματα και σε μια πιο δίκαιη βάση. Ρώτησαν την Google, περιορίζοντας τα αποτελέσματα μόνο σε σελίδες που προέρχονται από τη Wikipedia, παίρνοντας τα top-30 αποσπάσματα και σκοράροντας με τον ίδιο τρόπο όπως και για τη βασική γραμμή Google. Αναφέρονται σε αυτό ως βασική γραμμή Wikipedia. Στην δική μας εργασία δώσαμε και εμείς την επιλογή το σύστημά μας να ψάχνει μόνο σε σελίδες που προέρχονται από την Wikipedia επομένως υλοποιήσαμε και αυτή την βασική γραμμή.

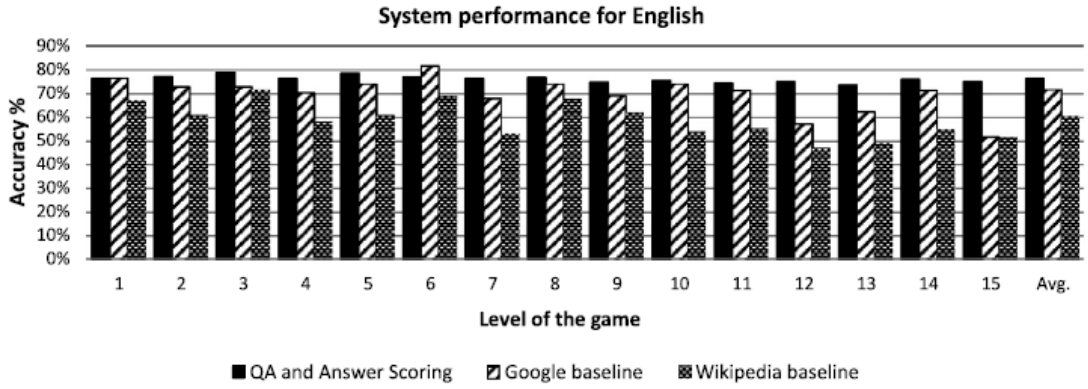
Στα Σχήματα 9 και 10 φαίνεται, για κάθε επίπεδο του παιχνιδιού:

- Η ακρίβεια των καλύτερων ρυθμίσεων που λαμβάνεται με την στρατηγική της μάθησης σε επίπεδο για την ιταλική και την αγγλική γλώσσα, των οποίων η μέση ακρίβεια είναι 79,64% (σ = 6,07%) για την ιταλική και 76,41% (σ = 1,45%) για τα αγγλικά.
- Η ακρίβεια που επιτυγχάνεται με τους ανθρώπινους παίκτες, η οποία είναι 51,33% (σ = 17.61%) για τα ιταλικά.
- Η ακρίβεια της βασικής γραμμής Google, η οποία είναι 67.13% για τα ιταλικά και 71,80% για τα αγγλικά.
- Η ακρίβεια της βασικής γραμμής Wikipedia, η οποία είναι 51,71% για τα ιταλικά και 60,65% για τα αγγλικά.

Η ακρίβεια του συστήματος είναι σημαντικά καλύτερη από τη βασική γραμμή Google, τη βασική γραμμή Wikipedia και την ακρίβεια των ανθρώπινων παικτών ($p << 0.0001$).



Σχήμα 9: Ακρίβεια σε κάθε επίπεδο της απόδοσης του συστήματος και της ανθρώπινης απόδοσης για τα Ιταλικά.



Σχήμα 10: Ακρίβεια της απόδοσης του συστήματος για τα Αγγλικά.

Το σύστημα έχει αρκετά παρόμοια επίδοση για όλα τα επίπεδα του παιχνιδιού. Όσον αφορά το ιταλικό σύνολο δεδομένων, η καλύτερη απόδοση λαμβάνεται για το επίπεδο 2, ενώ η χειρότερη λαμβάνεται για τα επίπεδα 7 και 13. Ωστόσο, η ανάλυση σφάλματος δεν ανέφερε κάποιο ειδικό πρόβλημα για τα εν λόγω επίπεδα: παρατήρησαν μόνο μια υψηλότερη συγκέντρωση ερωτήσεων που το σύστημα δεν είναι σε θέση να αντιμετωπίσει (βλέπε Ενότητα 9.1.1) σε μία από τις πέντε πτυχώσεις. Όσον αφορά τα αγγλικά, η απόδοση είναι σχεδόν σταθερή για όλα τα επίπεδα του παιχνιδιού (πράγματι η τυπική απόκλιση είναι πολύ χαμηλή).

Η ακρίβεια των ανθρώπινων παικτών (για τα Ιταλικά) μειώνεται σχεδόν μονότονα, με την καλύτερη απόδοση να λαμβάνεται στο πρώτο επίπεδο του παιχνιδιού, και τη χειρότερη στο επίπεδο 14. Η παρατήρηση αυτή είναι συνεπής με το γεγονός ότι τα χαμηλότερα επίπεδα του παιχνιδιού αντιστοιχούν σε ευκολότερες ερωτήσεις, ενώ υψηλότερα επίπεδα αντιστοιχούν σε περισσότερο πολύπλοκα ερωτήματα, για τα οποία μια βαθύτερη γνώση είναι απαραίτητη για να παρέχουν οι παίκτες τις σωστές απαντήσεις.

Το σύστημα (για την ιταλική γλώσσα) υπερτερεί σημαντικά έναντι των ανθρώπων για όλα τα 15 επίπεδα του παιχνιδιού (όχι μόνο κατά μέσο όρο). Αξίζει να αναλύσει κανείς την απόδοση των παικτών για τις ομάδες ερωτήσεων από 1 έως 5, 6 έως 10, και 11 έως 15, αντίστοιχα, όταν φτάσουν δηλαδή τα σημεία εγγύησης όπου τα χρήματα που κέρδισαν παραμένουν ασφαλή. Όσον αφορά την πρώτη ομάδα ερωτήσεων, οι

άνθρωποι έχουν την χειρότερη επίδοση στο (τέταρτο και) πέμπτο ερώτημα (61,88%), και αυτό δεν αποτελεί έκπληξη δεδομένου ότι αυτό είναι το πρώτο σημείο στο οποίο οι παίκτες έχουν την εγγύηση να κρατήσουν τα κερδισμένα χρήματα. Ακόμα κι αν το επίπεδο των ερωτήσεων δεν αποκαλύπτεται στους ανθρώπινους παίκτες, η υπόθεση μας είναι ότι η ερώτηση ορόσημο είναι πιθανόν πιο δύσκολη, λόγω του τρόπου που το παιχνίδι έχει σχεδιαστεί. Παραδόξως, αυτή η συμπεριφορά δεν παρατηρείται για τη δεύτερη ερώτηση ορόσημο, αν και η απόδοση για την ερώτηση αυτή είναι κάτω από το μέσο όρο. Τα αποτελέσματα που λαμβάνονται από τον άνθρωπο για την τελευταία ομάδα ερωτήσεων είναι όλα κάτω από το μέσο όρο, και δεν είναι συγκρίσιμα με εκείνα που λαμβάνονται από το σύστημα (απόλυτη διαφορά στην απόδοση που κυμαίνεται από + 38% έως + 56%).

Η βασική γραμμή Google είναι πάντα καλύτερη από τους ανθρώπινους παίκτες (για τα ιταλικά), αλλά είναι λιγότερο ακριβής από το σύστημα (η απόλυτη διαφορά στην απόδοση ανέρχεται σε -12,51% για τα Ιταλικά, και -4,61% για τα αγγλικά, κατά μέσο όρο). Η βασική γραμμή Wikipedia είναι λιγότερο ακριβής από τη βασική γραμμή Google για τα ιταλικά και για τα αγγλικά, και είναι ελαφρώς πιο ακριβής από την ανθρώπινη κατά μέσο όρο (για τα ιταλικά).

Για να συνοψίσουμε, ο εικονικός παίκτης κατασκευάστηκε με τη χρήση των μονάδων QA και AS και έχει τη δυνατότητα να νικήσει τους ανθρώπινους παίκτες όταν παίζουν ένα παιχνίδι με τους πραγματικούς κανόνες του, μιας και είναι σε θέση να απαντήσει σωστά στις ερωτήσεις σε διαφορετικά επίπεδα δυσκολίας. Οι συγγραφείς παρατηρούν πολύ παρόμοια απόδοση ανεξάρτητα από τη γλώσσα, δηλαδή οι μονάδες QA και AS λειτουργούν με τον ίδιο τρόπο για τα ιταλικά και τα αγγλικά. Οι μικρές διαφορές στην απόδοση είναι πιθανόν να οφείλονται στο διαφορετικό αριθμό των κειμένων που προέρχονται από την Wikipedia (σχεδόν ένα εκατομμύριο για τα ιταλικά, και τρία εκατομμύρια για τα αγγλικά), το οποίο θα μπορούσε να έχει επιπτώσεις στην απόδοση των μηχανών αναζήτησης στη μονάδα QA, και φυσικά στο γεγονός ότι τα δύο σύνολα δεδομένων δεν είναι συγκρίσιμα.

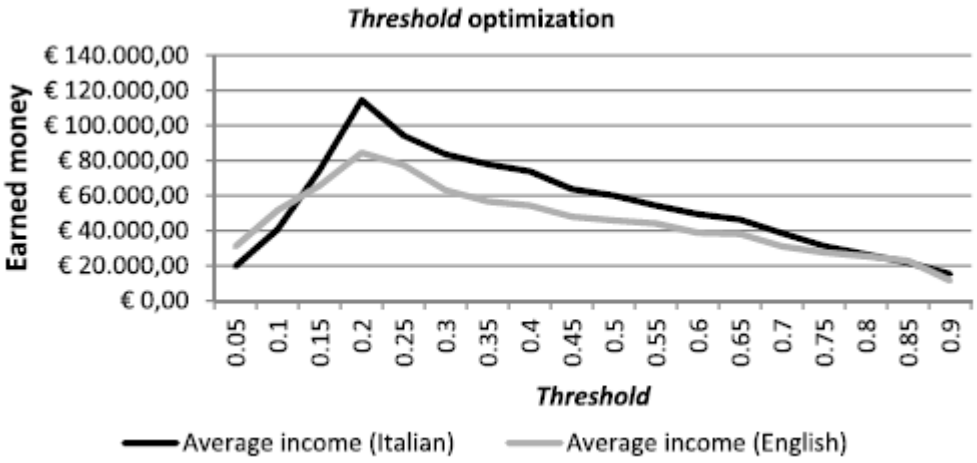
Ωστόσο, παίζοντας ένα πραγματικό παιχνίδι είναι ένα πολύ περίπλοκο έργο, δεδομένου ότι απαιτεί μια κατάλληλη στρατηγική για τη διαχείριση των σανίδων σωτηρίας, να αποφασίσεις αν θα απαντήσεις σε μια ερώτηση ή αν θα αποσυρθείς από το παιχνίδι παίρνοντας τα κερδισμένα χρήματα. Αυτός είναι ο σκοπός του πειράματος που περιγράφεται στο επόμενο τμήμα.

9.2 Πείραμα 2: αξιολόγηση του εικονικού παίκτη

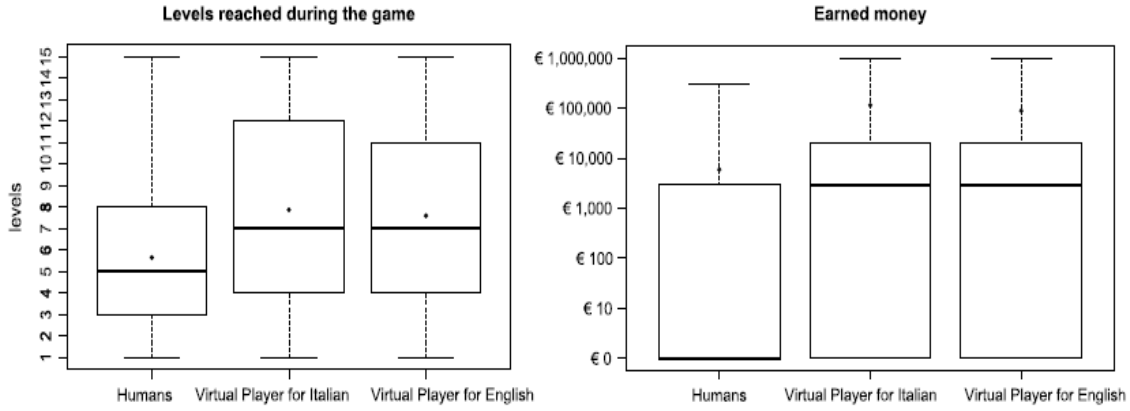
Ο στόχος αυτού του πειράματος είναι να αξιολογηθεί η ικανότητα του εικονικού παίκτη να παίξει το παιχνίδι, με την εφαρμογή μιας κατάλληλης στρατηγικής για να χρησιμοποιήσει τις σανίδες σωτηρίας, να αποφασίσει αν θα απαντήσει σε μια ερώτηση ακόμη και σε μια κατάσταση αβεβαιότητας ή να αποσυρθεί από το παιχνίδι παίρνοντας τα κερδισμένα χρήματα. Οι συγγραφείς, αλλά και σε αυτή την εργασία, έτρεξαν το πείραμα συγκρίνοντας την απόδοση των ανθρώπινων παικτών με την απόδοση που επιτυγχάνεται με την καλύτερη διαμόρφωση του συστήματος που λαμβάνεται με τη χρήση της προσέγγισης της μάθησης σε επίπεδο, και του οποίου συστήματος η στρατηγική για να παίξει το παιχνίδι ορίζεται από τον αλγόριθμο λήψης αποφάσεων που περιγράφεται παραπάνω. Η σύγκριση πραγματοποιείται αξιολογώντας το επίπεδο που επιτεύχθηκε κατά τη διάρκεια του παιχνιδιού και τα χρήματα που κέρδισε ο παίκτης (Η Ιταλική και η Αγγλική έκδοση του παιχνιδιού έχουν διαφορετικό νόμισμα και διαφορετικές χρηματικές αποδόσεις σε κάθε επίπεδο του παιχνιδιού. Για χάριν της

σύγκρισης αποφασίστηκε να χρησιμοποιήσουνε το ίδιο νόμισμα και τις ίδιες χρηματικές απολαβές ανά επίπεδο με τα Ιταλικά, δηλαδή το ευρώ). Όπως και στο προηγούμενο πείραμα, η σύγκριση μεταξύ της απόδοσης του εικονικού παίκτη και της ανθρώπινης πραγματοποιείται μόνο για τους Ιταλούς. Στην δικιά μας υλοποίηση η σύγκριση πραγματοποιείται για τους Έλληνες.

Χρησιμοποίησαν 35 άτομα, διαφορετικά από εκείνα που συμμετείχαν στο προηγούμενο πείραμα, που συνολικά έπαιξαν 325 παιχνίδια. Εμείς πήραμε τα αποτελέσματά μας από τα ίδια 20 άτομα που χρησιμοποιήσαμε και στο Πείραμα 1 και συλλέξαμε τις πληροφορίες μας από τα ίδια παιχνίδια που έπαιξαν εκείνοι. Συνολικά τους βάλαμε να παίξουν από 10 παιχνίδια ο καθένας, σύνολο 200 παιχνίδια. Στο άρθρο [1] κάθε παιχνίδι αποτελείται από ερωτήσεις που επιλέγονται από το Διαχειριστή του παιχνιδιού, το οποίο τυχαία διάλεγε ένα ερώτημα για κάθε επίπεδο, όσο το παιχνίδι προχωρούσε (παιχνίδια που παίζονταν από τον ίδιο ανθρώπινο παίκτη είχαν πάντα διαφορετικές ερωτήσεις για το ίδιο επίπεδο). Στην δικιά μας υλοποίηση ο Διαχειριστής του παιχνιδιού διάλεγε τυχαία μια ερώτηση για κάθε επίπεδο από εκείνες τις ερωτήσεις όμως που ανήκαν σε αυτό το επίπεδο. Βέβαια επειδή αυτή η επιλογή γινόταν τυχαία και κάθε παιχνίδι ξεκινούσε από την αρχή χωρίς να ξέρει ο διαχειριστής του παιχνιδιού ποιες ερωτήσεις είχε χρησιμοποιήσει σε προηγούμενα παιχνίδια ώστε να μην τις ξαναχρησιμοποιήσει υπήρχε πιθανότητα κάποια ερώτηση ένας παίκτης να την αντιμετωπίσει πάνω από μια φορά. Βέβαια λόγω του μεγάλου πλήθους των ερωτήσεων (1000 ερωτήσεις για κάθε επίπεδο) πειραματικά αυτό το πρόβλημα δεν προέκυψε σε κανέναν από τους 20 παίχτες. Από την άλλη πλευρά, ο εικονικός παίκτης έπαιξε 160 παιχνίδια, των οποίων οι ερωτήσεις επιλέγονται επίσης από το Διαχειριστή του παιχνιδιού τόσο από το ιταλικό όσο και από το αγγλικό σύνολο δεδομένων. Στην δικιά μας υλοποίηση ο εικονικός παίκτης έπαιξε και για το Πείραμα 1 και για το Πείραμα 2 συνολικά 100 παιχνίδια. Μια σημαντική ρύθμιση για τον εικονικό παίκτη είναι η τιμή του ορίου που χρησιμοποιείται από τον αλγόριθμο λήψης απόφασης για την αξιολόγηση μιας κατάστασης αβεβαιότητας (διαφορά μεταξύ της βαθμολογίας της καλύτερης και της δεύτερης καλύτερης υποψήφιας απάντησης). Η τιμή κατωφλίου βελτιστοποιήθηκε εμπειρικά μεταβάλλοντας την αξία της, και επιλέγοντας εκείνη την τιμή η οποία οδήγησε τον εικονικό παίκτη να επιτύχει το υψηλότερο μέσο εισόδημα στις ερωτήσεις στο σύνολο επικύρωσης της πρώτης πτυχής. Το κατώφλι τελικά ορίστηκε από τους συγγραφείς σε 0,2 και για τα ιταλικά και για τα αγγλικά (ένα γράφημα που δείχνει πόσο διαφορετικές τιμές επηρεάζουν το μέσο εισόδημα απεικονίζεται στο Σχήμα 11). Εμείς ορίσαμε το κατώφλι να είναι ίσο με 1, δηλαδή αρκεί η πρώτη καλύτερη απάντηση να έχει μεγαλύτερη βαθμολογία από την δεύτερη καλύτερη κατά 1. Εάν οι απαντήσεις ισοβαθμούν, τότε είτε χρησιμοποιούμε τις βοήθειες είτε αποσυρόμαστε από το παιχνίδι είτε απαντάμε στην τύχη μία από τις δύο απαντήσεις.



Σχήμα 11: Πλοκή του μέσου εισοδήματος για διαφορετικές τιμές του ορίου μέτρησης της διαφοράς μεταξύ της βαθμολογίας της καλύτερης και της δεύτερης καλύτερης υποψήφιας απάντησης.

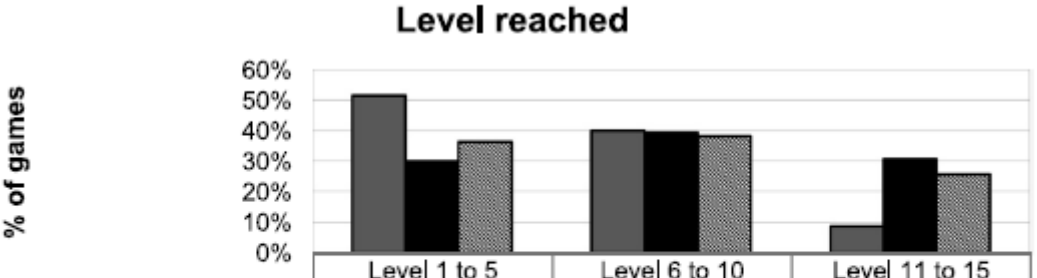


Σχήμα 12: Κατανομή των επιπέδων που επιτεύχθηκαν κατά τη διάρκεια του παιχνιδιού και τα χρήματα που κέρδισαν οι παίκτες (σε λογαριθμική κλίμακα). Τα άνω και κάτω άκρα των κιβωτίων αντιπροσωπεύουν το 3ο και 1ο τεταρτημόριο, αντίστοιχα. Η Διάμεσοι απεικονίζεται με συνεχείς γραμμές, οι μέσες τιμές με σταθερά σημεία.

Το Σχήμα 12 δείχνει την κατανομή των επιπέδων που επιτεύχθηκαν κατά τη διάρκεια του παιχνιδιού και τα χρήματα που κερδήθηκαν από τους παίκτες, ενώ τα Σχήματα 13 και 14 αναφέρουν την κατανομή των παιχνιδιών που έφτασαν σε ένα συγκεκριμένο επίπεδο και την κατανομή των παιχνιδιών που τελείωσαν με το εισόδημα σε ένα συγκεκριμένο διάστημα.

Όλοι οι παίκτες είναι σε θέση να φτάσουν στο τελευταίο επίπεδο του παιχνιδιού, αλλά το μέσο επίπεδο που επιτεύχθηκε από τον άνθρωπο είναι μεταξύ πέντε και έξι (5,65), σε αντίθεση με τον εικονικό παίκτη που φθάνει σε ένα επίπεδο μεταξύ επτά και οκτώ, 7.88 και 7.60 για τα ιταλικά και τα αγγλικά, αντίστοιχα. Αυτό τονίζεται επίσης από τη μέση τιμή, η οποία είναι το πέμπτο επίπεδο για τον άνθρωπο και το έβδομο για τον εικονικό παίκτη. Περισσότερες από τις μισές φορές (51,38%), οι ανθρώπινοι παίκτες τελείωσαν το παιχνίδι με την επίτευξη των επιπέδων από 1 έως 5, ενώ το 40% των φορών έφθασαν σε επίπεδα 6 έως 10. Λίγες φορές (8,62%) οι άνθρωποι ήταν σε θέση να φθάσουν τα τελευταία επίπεδα (το επίπεδο 15 έχει επιτευχθεί μόνο μία φορά). Η κατανομή των παιχνιδιών που τελείωσαν από τον εικονικό παίκτη στις τρεις ομάδες

ερωτήσεων είναι σχεδόν ίδια για τα ιταλικά και τα αγγλικά (με ελαφρώς καλύτερη απόδοση για τα ιταλικά), και αυτό είναι συνεπές με τα αποτελέσματα στα Σχήματα 9 και 10, στα οποία η ακρίβεια του συστήματος είναι πολύ παρόμοια για όλα τα επίπεδα του παιχνιδιού.



	Level 1 to 5	Level 6 to 10	Level 11 to 15
Human players	51.38%	40.00%	8.62%
Virtual player for Italian	30,00%	39,38%	30,62%
Virtual player for English	36,25%	38,13%	25,62%

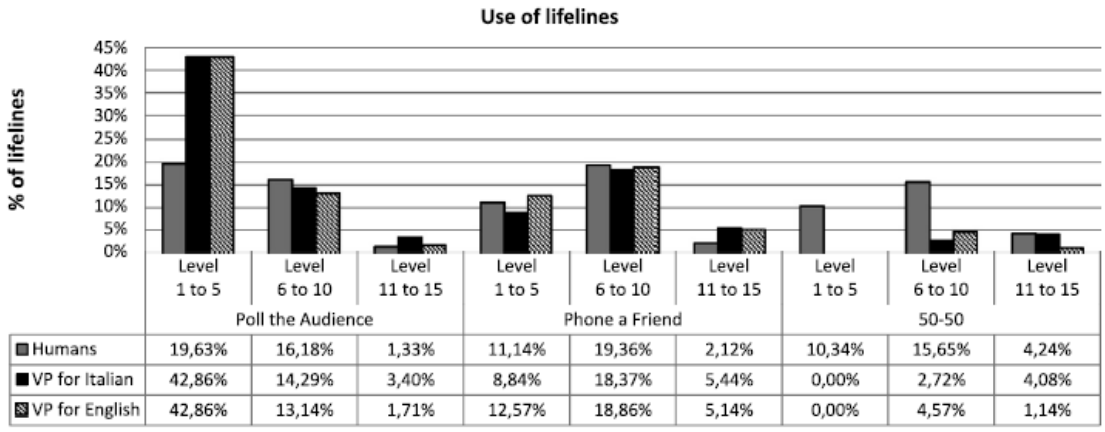
Σχήμα 13: Κατανομή των παιχνιδιών που έφτασαν σε κάποιο συγκεκριμένο επίπεδο.

	€ 0	]0, 3,000] €	]3,000, 20,000] €	]20,000, 1,000,000] €
Human players	51,38%	36,92%	8,31%	3,39%
Virtual player for Italian	28,75%	30,62%	25,63%	15,00%
Virtual player for English	35,62%	25,00%	23,75%	15,63%

Σχήμα 14: Κατανομή των παιχνιδιών που τελείωσαν με το εισόδημα σε ένα συγκεκριμένο διάστημα.

301 από 325 παιχνίδια (92,61%) που παίχτηκαν από ανθρώπους έληξαν εξαιτίας ενός σφάλματος στην απάντηση, ενώ 24 φορές (7,39%), ο παίκτης τελείωσε το παιχνίδι με τη συνταξιοδότηση και τη λήψη των κερδισμένων χρημάτων (σε τρεις περιπτώσεις οι παίκτες αποσύρθηκαν από το παιχνίδι στο επίπεδο 6 και 11, ακόμη και αν μια λανθασμένη απάντηση δεν θα είχε καμία επίδραση στα κερδισμένα χρήματα). Επιπλέον, οι ανθρώπινοι παίκτες ποτέ δεν τελείωσαν το παιχνίδι με το μέγιστο βραβείο. Διαφορετικά από τον άνθρωπο, ο εικονικός παίκτης ήταν σε θέση να ολοκληρώσει με επιτυχία το παιχνίδι. Πράγματι, κέρδισε 1.000.000 ευρώ 17 φορές (10,62%) για τα ιταλικά, και 12 φορές (7,50%) για την Αγγλικά. 116 παιχνίδια (72.50%) τελείωσαν εξαιτίας λάθος απάντησης για την ιταλική, ενώ 27 φορές (16.87%) αποσύρθηκε από το παιχνίδι, χωρίς να παρέχει την απάντηση. Ο εικονικός παίκτης για τα αγγλικά έληξε 114 παιχνίδια (71,25%) εξαιτίας λάθος απάντησης, και αποσύρθηκε από το παιχνίδι 34 φορές (21.25%). Το υψηλό ποσοστό των παιχνιδιών που έληξε ο εικονικός παίκτης χωρίς να παρέχει την απάντηση αναδεικνύει πόσο συντηρητική και στρατηγική χαμηλού κινδύνου ακολούθησε. Είναι ενδιαφέρον να σημειώσουμε ότι ο αλγόριθμος λήψης αποφάσεων δεν επιτρέπει στον εικονικό παίκτη να τελειώσει το παιχνίδι στο επίπεδο 6 ή 11.

Τα χρήματα που κέρδισαν οι άνθρωποι είναι 5926 ευρώ, κατά μέσο όρο, ενώ το μέσο εισόδημα του εικονικού παίκτη είναι σημαντικά υψηλότερο. Πράγματι, κέρδισε 114.531 ευρώ για τα Ιταλικά, και 88.878 ευρώ για τα αγγλικά. Τα λεπτομερή στοιχεία φαίνονται στο Σχήμα 14, όπου αναφέρονται τα παιχνίδια που έκλεισαν με μηδενικό εισόδημα, καθώς και τα παιχνίδια που κατέληξαν σε κάθε διάστημα που αντιστοιχεί σε κάποια ερώτηση ορόσημο. Τα περισσότερα από τα παιχνίδια που παίχτηκαν από τους ανθρώπους (88,30%) έκλεισαν με μηδενικό εισόδημα ή απαντώντας σε ερωτήσεις από την πρώτη ομάδα. Αυτό σημαίνει ότι έφτασαν στις δύο τελευταίες ομάδες ερωτήσεων το 11,70% των φορών, ενώ ο εικονικός παίκτης έφθασε στις τελευταίες δύο ομάδες ερωτήσεων περίπου το 40% των φορών και για τα ιταλικά και για τα αγγλικά. Το ότι το λιγότερο ποσοστό των παιχνιδιών έκλεισαν με μηδενικό εισόδημα από τον εικονικό παίκτη επιβεβαιώνει την συμπεριφορά του που ελαχιστοποιεί τον κίνδυνο (αν και κάποιες διαφορές μεταξύ των Ιταλικών και των αγγλικών υπάρχουν), σε συνδυασμό με την ικανότητά του να παρέχει σωστές απαντήσεις ανεξάρτητα από το επίπεδο του παιχνιδιού, και του επιτρέπει να σταματήσει τα παιχνίδια με υψηλότερο μέσο εισόδημα.



	Poll the Audience			Phone a Friend			50-50		
	Level 1 to 5	Level 6 to 10	Level 11 to 15	Level 1 to 5	Level 6 to 10	Level 11 to 15	Level 1 to 5	Level 6 to 10	Level 11 to 15
Humans	19,63%	16,18%	1,33%	11,14%	19,36%	2,12%	10,34%	15,65%	4,24%
VP for Italian	42,86%	14,29%	3,40%	8,84%	18,37%	5,44%	0,00%	2,72%	4,08%
VP for English	42,86%	13,14%	1,71%	12,57%	18,86%	5,14%	0,00%	4,57%	1,14%

Σχήμα 15: Κατανομή των σανίδων που χρησιμοποιούνται κατά την διάρκεια του παιχνιδιού.

Το Σχήμα 15 αναφέρει τη χρήση των σανίδων ασφαλείας κατά τα διάφορα στάδια του παιχνιδιού. Η σύγκριση των στρατηγικών που υιοθετούνται από τον άνθρωπο και τον εικονικό παίκτη δεν είναι εύλογη εφόσον ο τελευταίος χρησιμοποιεί ένα στατικό προκειμένου να χρησιμοποιήσει τις διαθέσιμες σανίδες ασφαλείας, όπως ορίζεται από τον αλγόριθμο λήψης αποφάσεων (δηλαδή βοήθεια του Κοινού, το τηλέφωνο, 50:50). Παρά τον περιορισμό αυτό, σε γενικές γραμμές, η στήριξη που παρέχεται από τις σανίδες βοήθειας στον εικονικό παίκτη είναι πολύτιμη. Πράγματι, ο εικονικός παίκτης είναι σε θέση να παρέχει τη σωστή απάντηση 114 φορές για την ιταλική και 141 φορές για την αγγλική χάρη στις βοήθειες. Οι ανθρώπινοι παίκτες και ο εικονικός παίκτης χρησιμοποιούν μια μόνο σανίδα σωτηρίας ανά παιχνίδι κατά μέσο όρο, ως εκ τούτου, συμπεριφέρονται με παρόμοιο τρόπο. Κατά τα πρώτα στάδια του παιχνιδιού, η συχνότητα χρήσης της βοήθειας του κοινού από τον εικονικό παίκτη (42,86% για τα ιταλικά και τα αγγλικά) είναι όση η συνολική συχνότητα χρήσης και των τριών διαθέσιμων σανίδων σωτηρίας από τους ανθρώπινους παίκτες (41.11%). Αυτό σημαίνει ότι ο εικονικός παίκτης εξαναγκάζεται να χρησιμοποιήσει αυτή τη σανίδα σωτηρίας πριν από τις άλλες, όπως ορίζεται στον αλγόριθμο λήψης αποφάσεων, ενώ οι ανθρώπινοι παίκτες χρησιμοποιούν όλες τις διαθέσιμες σανίδες σωτηρίας, παρόλο που προτιμούν την βοήθεια του κοινού κατά τη διάρκεια των πρώτων σταδίων του παιχνιδιού. Η συχνότητα της χρήσης του τηλεφώνου από τον άνθρωπο και τον εικονικό παίκτη είναι

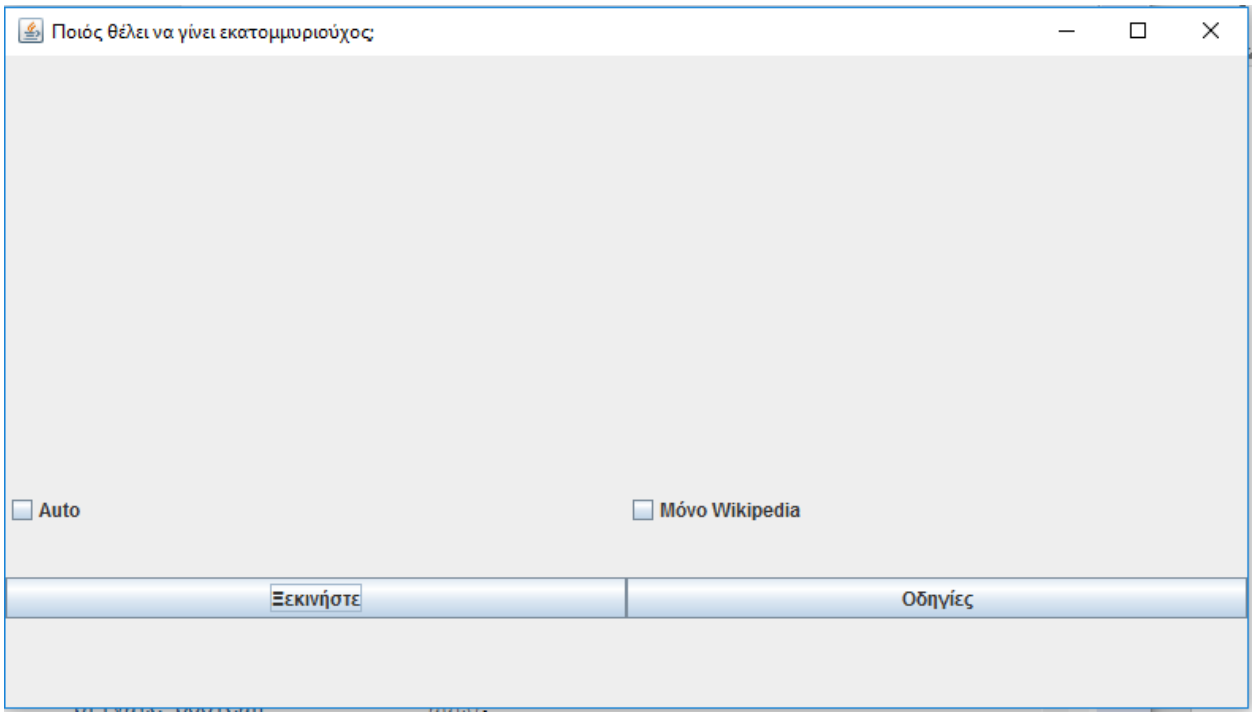
αρκετά παρόμοια. Όπως ήταν αναμενόμενο, το 50:50 δεν χρησιμοποιείται ποτέ από το εικονικό παίκτη κατά τη διάρκεια των πρώτων επιπέδων του παιχνιδιού, ενώ οι άνθρωποι το χρησιμοποιούν ομοιόμορφα σε όλα τα επίπεδα του παιχνιδιού.

Συνοψίζοντας, όπως αναμένεται από τα αποτελέσματα του πειράματος 1, ο εικονικός παίκτης είναι σε θέση να ξεπεράσει τον άνθρωπο, παρότι υιοθετεί μια πολύ απλή στρατηγική λήψης αποφάσεων. Η καλύτερη απόδοση είναι από την άποψη τόσο του μέσου επιπέδου που φτάνει κανείς όσο και των κερδισμένων χρημάτων στο τέλος του παιχνιδιού. Η απόδοση του εικονικού παίκτη για τα ιταλικά και τα αγγλικά είναι λίγο πολύ η ίδια, εκτός από το μέσο εισόδημα, το οποίο είναι πολύ καλύτερο για τα Ιταλικά από τα αγγλικά. Όπως παρατηρήθηκε για την εκτέλεση των QA και AS, αυτό είναι πιθανό να οφείλεται στο διαφορετικό μέγεθος των πηγών γνώσης για την ιταλική και την αγγλική γλώσσα.

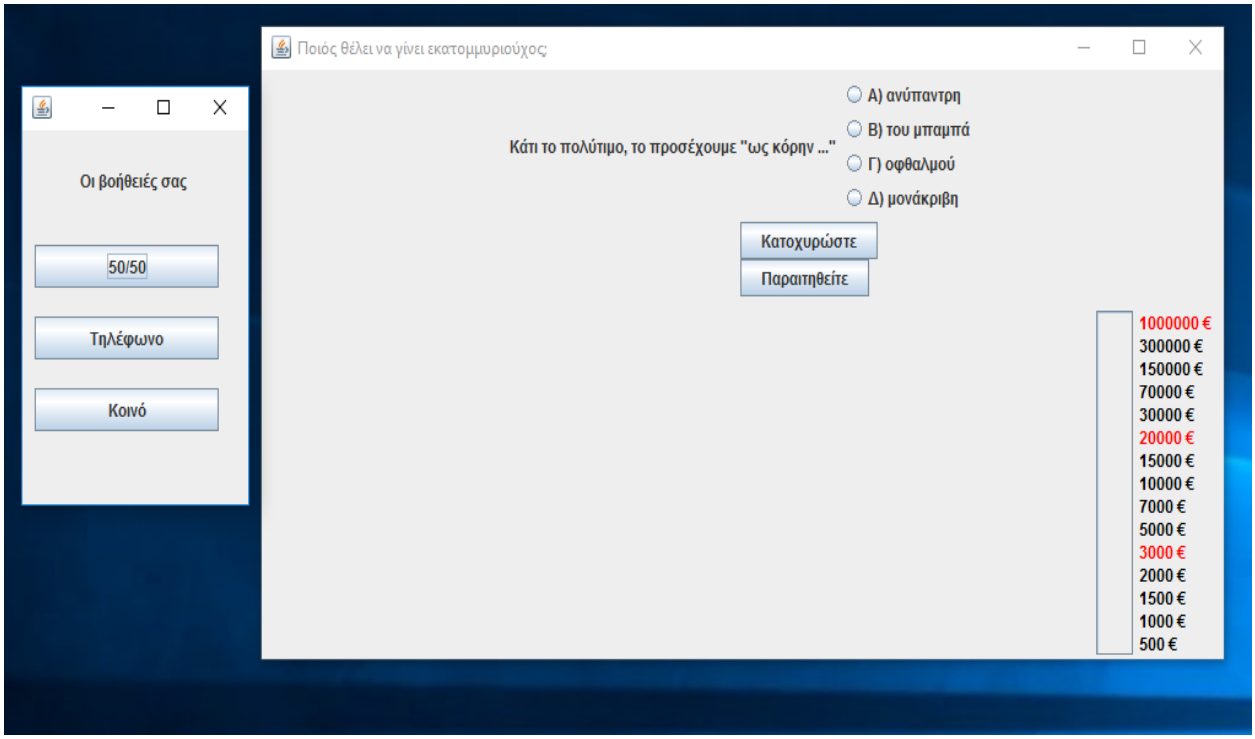
10. ΕΝΔΕΙΚΤΙΚΗ ΕΚΤΕΛΕΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Υπάρχουν δύο τρόποι για να εκτελεστεί το πρόγραμμα. Ο πρώτος τρόπος είναι να ανοιχτεί το eclipse project wwtbam που υπάρχει στα παραδοτέα αρχεία της εργασίας με το Eclipse IDE oxygen. Εκεί μέσα υπάρχει έτοιμο το αρχείο Questions.txt. Για να ξαναδημιουργηθεί πρέπει πρώτα να διαγραφεί και να τρέξει η κλάση QuestionParser.java (δεξί κλίκ -> Run as Java Application) αφού πρώτα έχει μπει ο φάκελος “library” (βρίσκεται στα παραδοτέα της εργασίας) στο Desktop του υπολογιστή και έχει δηλωθεί το path στην μεταβλητή LIBRARY_LOCATION. Κάνοντας ένα F5 στο project το καινούργιο Questions.txt θα εμφανιστεί. Στην συνέχεια για να τρέξει το κυρίως πρόγραμμα πρέπει να τρέξει η κλάση GameManager.java (δεξί κλίκ -> Run as Java Application). Ο δεύτερος τρόπος είναι να τρέξει το εκτελέσιμο wwtbam.jar που έχει φτιαχτεί ήδη και βρίσκεται στα παραδοτέα της εργασίας. Μπορεί κάποιος να βάλει το wwtbam.jar μαζί με το Questions.txt που επίσης υπάρχει στα παραδοτέα όπου θέλει στον υπολογιστή του και μετά να ανοίξει ένα command line και να τρέξει την εντολή «java -jar ./wwtbam.jar». Δυστυχώς με αυτόν τον τρόπο δεν μπορεί να τρέξει μόνος του τον QuestionParser για να φτιάξει από την αρχή το Questions.txt.

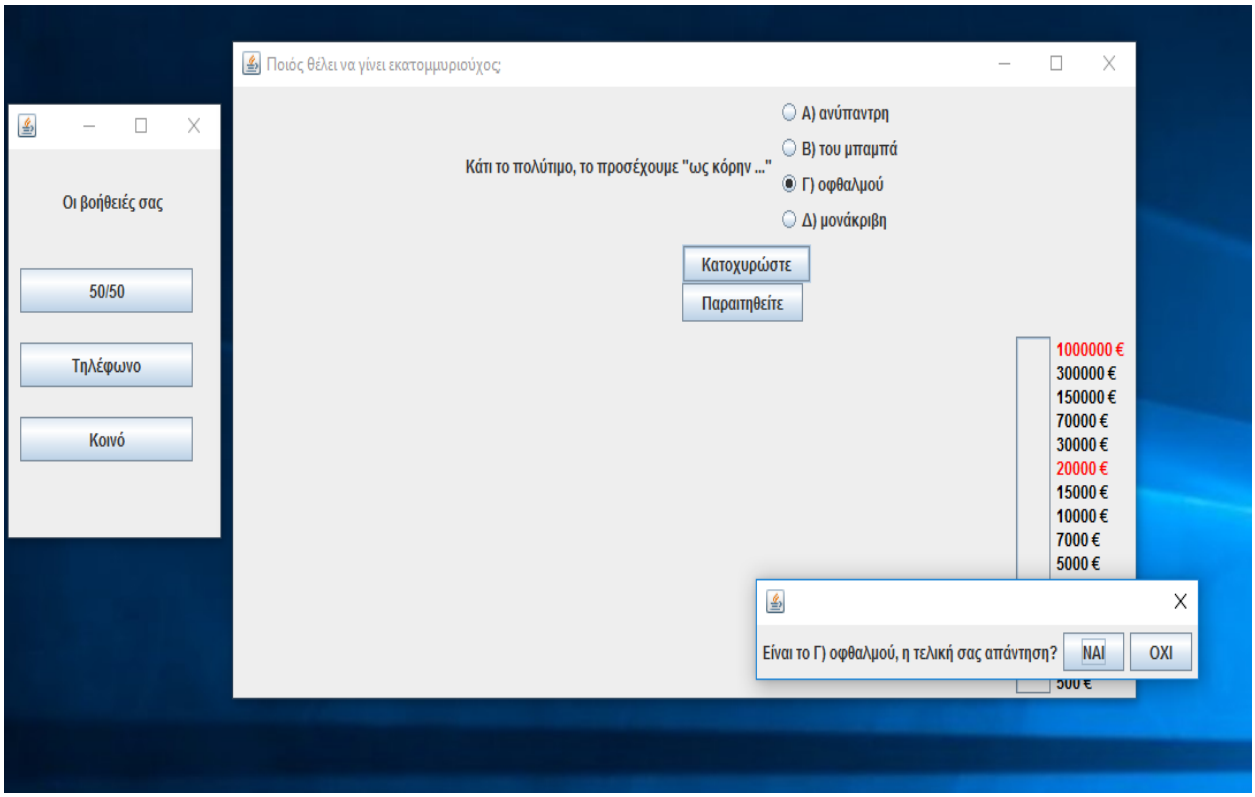
Όποιον τρόπο και από τους δύο να διαλέξει το αποτέλεσμα θα είναι το ίδιο αφού το παιχνίδι θα ξεκινήσει. Παρακάτω δίνονται μερικές εικόνες από μια ενδεικτική εκτέλεση του παιχνιδιού όταν αυτό παίζεται από έναν ανθρώπινο παίκτη έτσι ώστε να φανούν όλες οι πιθανές οθόνες που μπορεί να εμφανιστούν. Η εκτέλεση όταν το παιχνίδι παίζεται από τον εικονικό παίκτη (είναι δηλαδή σε auto mode) οπτικά δεν έχει καμία διαφορά αφού εμφανίζονται οι ίδιες οθόνες και τα ίδια μηνύματα. Μια ενδεικτική εκτέλεση είναι η εξής:



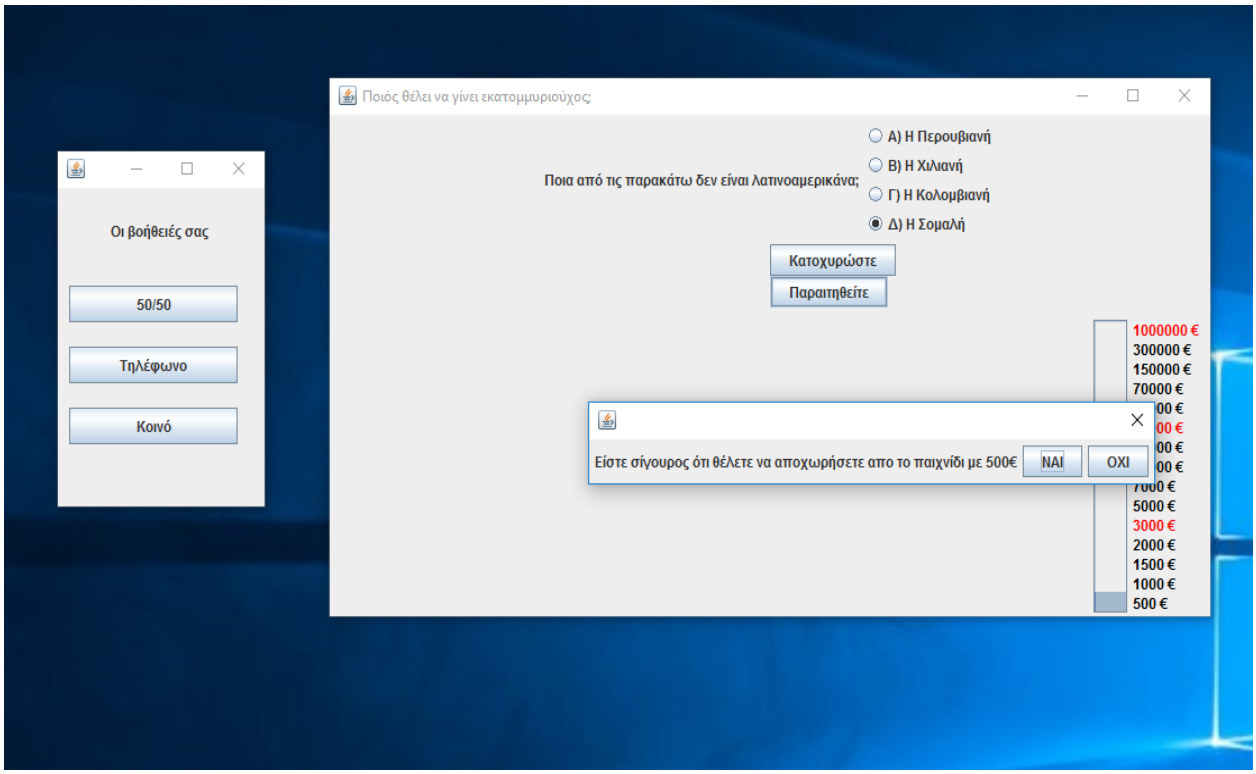
Εικόνα 3: Αρχική οθόνη του παιχνιδιού.



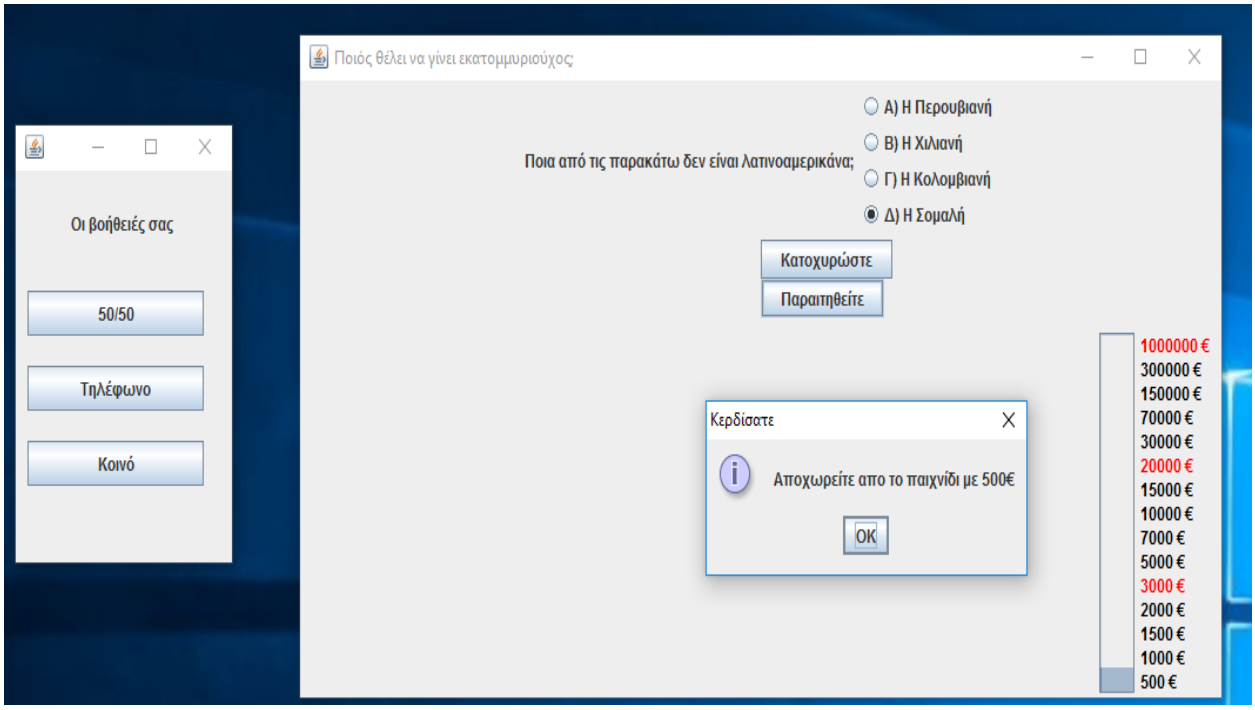
Εικόνα 4: Οθόνη με την πρώτη ερώτηση του παιχνιδιού.



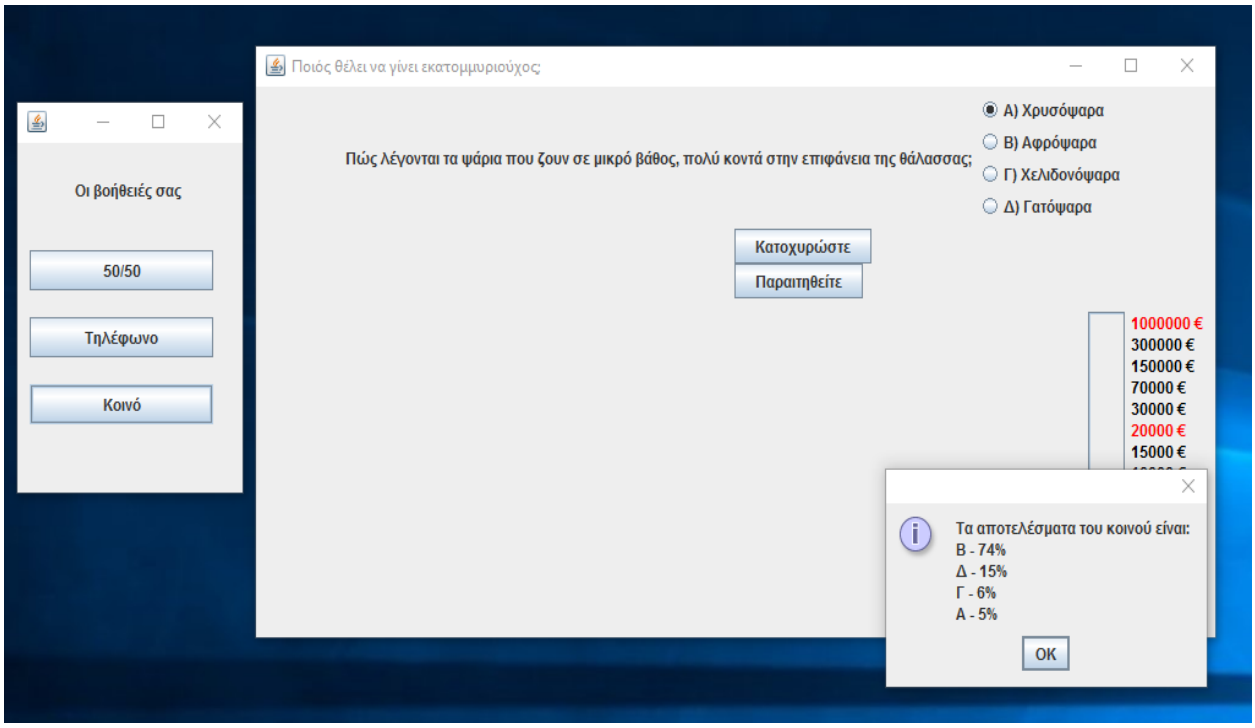
Εικόνα 5: Επιβεβαίωση της απάντησης που διάλεξε ο παίκτης.



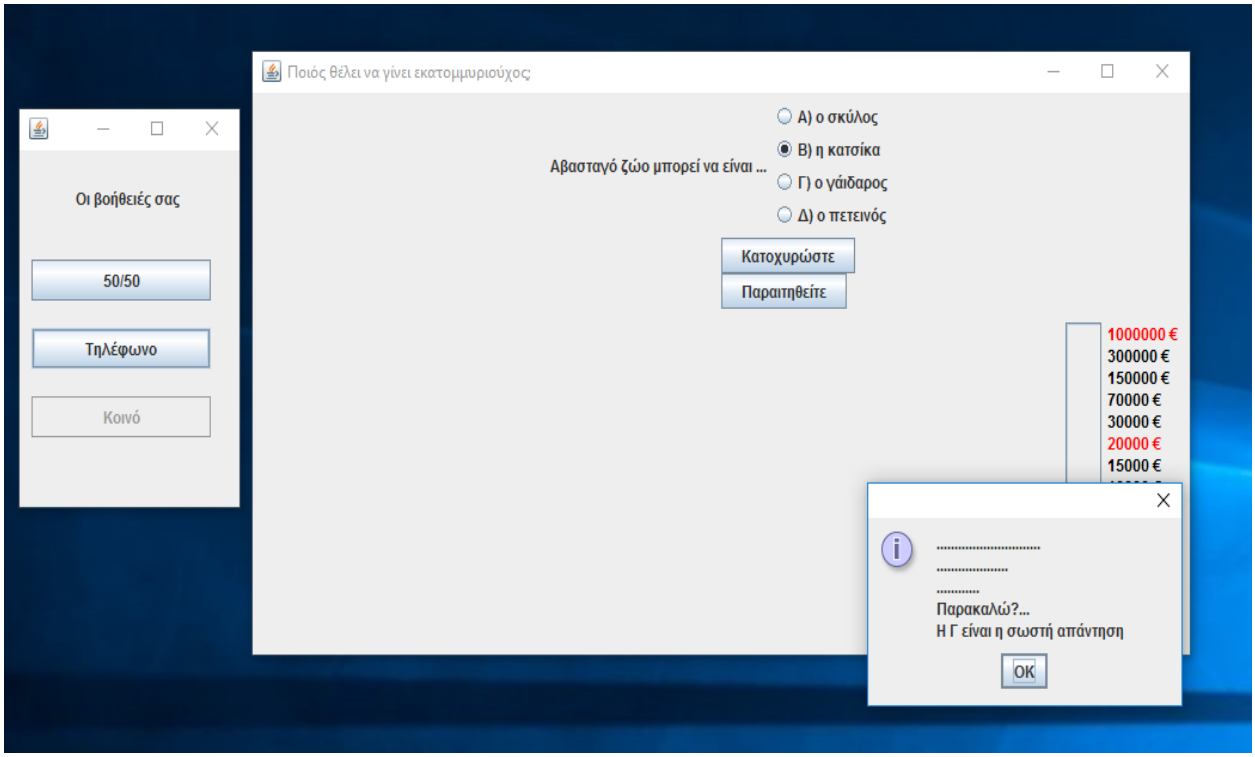
Εικόνα 6: Πάτημα του κουμπιού «Παραιτηθείτε» και επιβεβαίωση για παραίτηση από το παιχνίδι με την λήψη του κερδισμένου έως αυτό το επίπεδο ποσού.



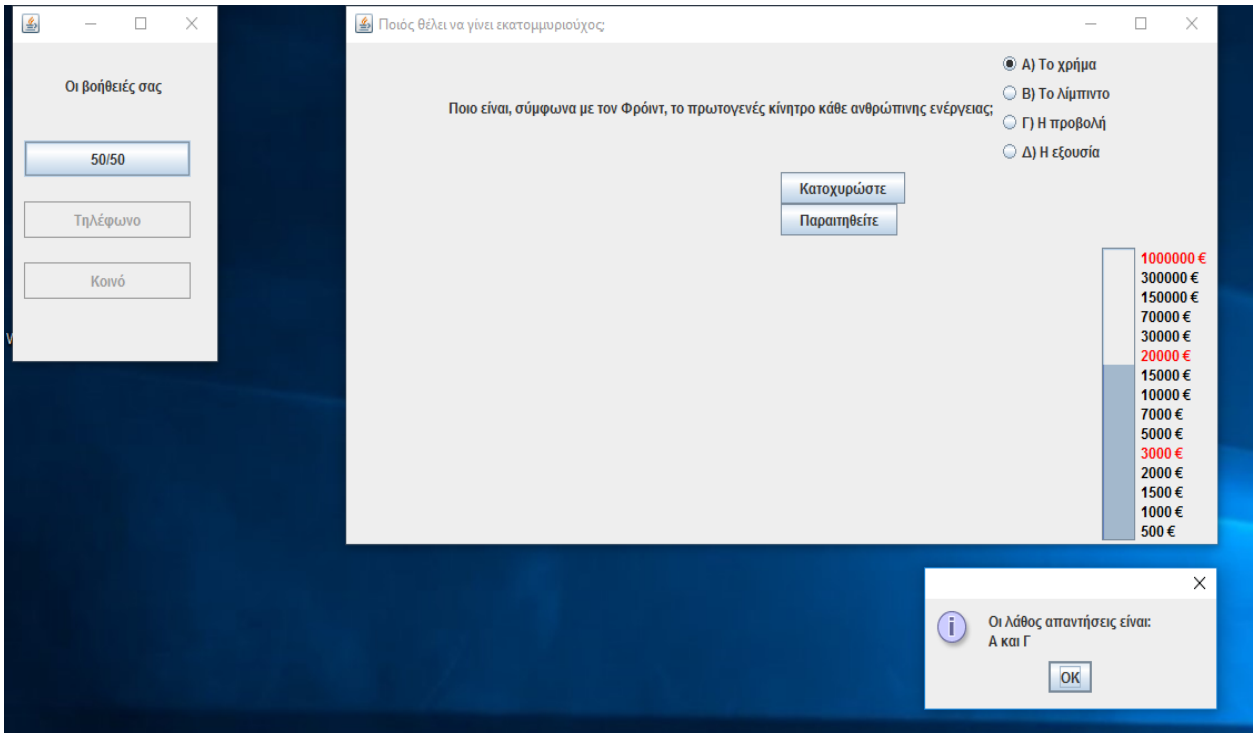
Εικόνα 7: Λήψη του κερδισμένου ποσού μετά την παραίτηση από το παιχνίδι.



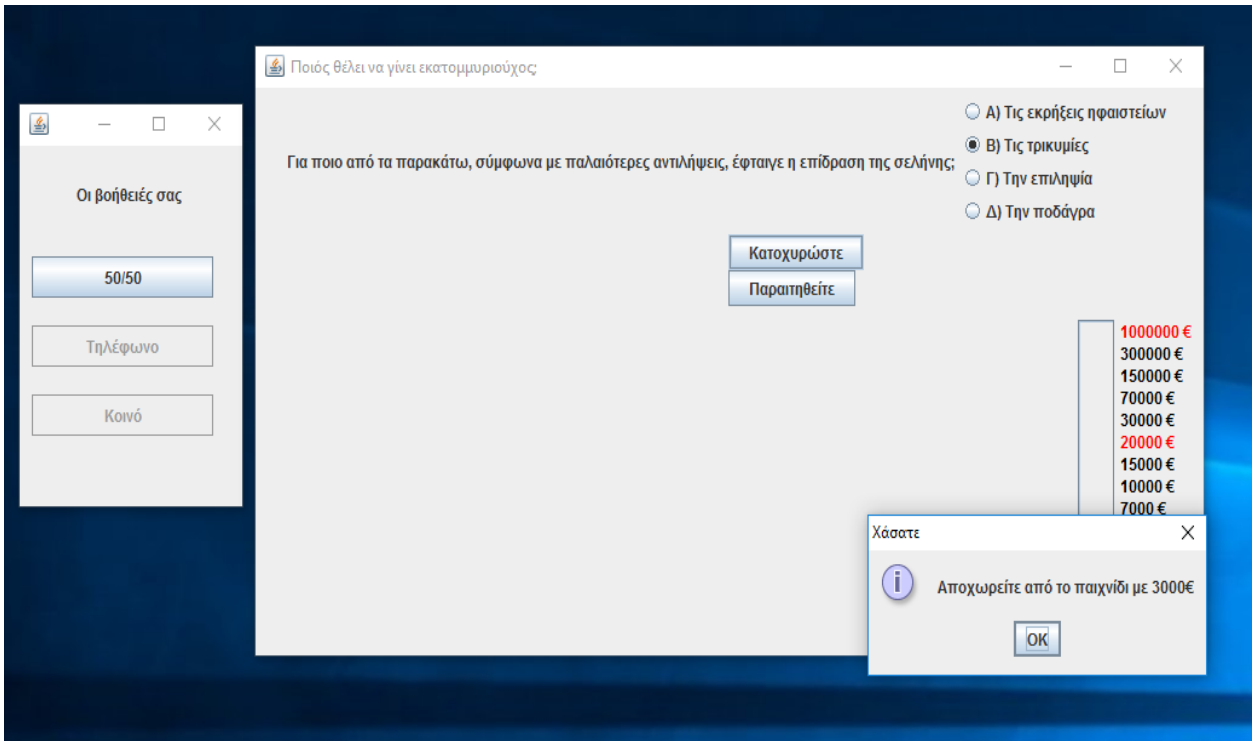
Εικόνα 8: Χρήση της βοήθειας «Κοινό» και εμφάνιση μηνύματος με τα αποτελέσματα του κοινού.



Εικόνα 9: Χρήση της βοήθειας «Τηλέφωνο» και εμφάνιση του μηνύματος της απάντησης που έδωσε στο τηλέφωνο ο φίλος του παίκτη.



Εικόνα 10: Χρήση της βοήθειας «50/50» και εμφάνιση του μηνύματος με τις δύο λανθασμένες απαντήσεις.



Εικόνα 11: Αποχώρηση από το παιχνίδι ύστερα από λάθος απάντηση σε ερώτηση και λήψη του ποσού στο μαξιλαράκι που έχει κλειδώσει ο παίκτης.

Το παιχνίδι έχει σχεδιαστεί με τέτοιο τρόπο όπου γίνονται ελάχιστες παραδοχές για την γλώσσα στην οποία είναι διατυπωμένες οι ερωτήσεις. Θεωρητικά λοιπόν έχοντας το αντίστοιχο αρχείο Questions.txt στην γλώσσα που επιθυμεί κάποιος θα μπορούσε να παίξει το παιχνίδι και σε manual και σε auto mode. Πρακτικά θα πρέπει να αλλάξουν

μερικές παράμετροι στον πηγαίο κώδικα του προγράμματος. Η βάση λοιπόν για να παιχτεί το παιχνίδι και σε άλλες γλώσσες είναι να υπάρχει το αρχείο Questions.txt. Εάν αυτό δεν υπάρχει αλλά υπάρχουν τα ίδια qfileXXX.txt στον φάκελο “library” (με τα δεδομένα να είναι στην νέα γλώσσα αλλά η σύνταξη του αρχείου να είναι η ίδια), χρησιμοποιώντας τον ίδιο QuestionParser.java μπορεί να φτιαχτεί το Questions.txt (εδώ να σημειωθεί ότι ανάλογα με την γλώσσα ίσως χρειαστεί να αλλάξει η κωδικοποίηση με την οποία διαβάζονται τα qfileXXX.txt αρχεία). Το επόμενο πράγμα που πρέπει να αλλάξει είναι το γραφικό περιβάλλον το οποίο είναι όλο στα Ελληνικά. Επομένως πρέπει να αλλαχτούν όλα τα κείμενα που εμφανίζονται σε κουμπιά και μηνύματα στην αντίστοιχη γλώσσα. Επίσης πρέπει και στο αρχείο UtilityHelper τα γράμματα «Α», «Β», «Γ» και «Δ» να αλλάξουν στα αντίστοιχα γράμματα της αλφαβήτου της εκάστοτε γλώσσας. Στο QuestionAnswering και στο DecisionMaking δεν χρειάζεται να αλλάξει τίποτα. Στο AnswerScoring όμως πρέπει να αλλάξουν οι μεταβλητές NEGATIVE_QUESTION_IDENTIFIER, IRRELEVANT_WORDS_UPPER, IRRELEVANT_WORDS_LOWER και να περιέχουν τις αντίστοιχες τιμές στην γλώσσα που θα χρησιμοποιηθεί.

Εάν γίνουν αυτές οι μικρές αλλαγές τότε το παιχνίδι μπορεί να παιχτεί σε οποιαδήποτε γλώσσα συμπεριλαμβανομένων και των δύο γλωσσών του άρθρου [1] (Αγγλικά και Ιταλικά).

11. ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην παρούσα εργασία μελετήσαμε δύο ζητήματα:

- Σε ποιο βαθμό μπορεί ένα σύστημα QA να σχεδιαστεί με έναν τρόπο ανεξάρτητο γλώσσας, διατηρώντας την αποτελεσματικότητά του;
- Μπορεί η Wikipedia και η DBpedia να χρησιμεύσουν ως αποτελεσματικές βάσεις γνώσεων για την απάντηση στις ερωτήσεις του παιχνιδιού WWBM;

Όσον αφορά το πρώτο ζήτημα, το έργο αυτό στην πραγματικότητα οδήγησε στον ορισμό ενός αποτελεσματικού συστήματος για QA ανεξάρτητο γλώσσας, στο οποίο τόσο η μονάδα Απάντησης Ερώτησης και Βαθμολόγησης Απάντησης ορίστηκαν χρησιμοποιώντας ένα σύνολο από φίλτρα που δεν είναι συναφή σε μια συγκεκριμένη γλώσσα. Ορίσαμε επίσης μια αποτελεσματική στρατηγική για να συνδυάσουμε διαφορετικά κριτήρια για τη βαθμολόγηση υποψήφιων απαντήσεων μέσω τεχνικών μηχανικής μάθησης. Τα πειράματα που πραγματοποιήθηκαν σε ιταλικά και αγγλικά έδειξαν την αποτελεσματικότητα της προσέγγισης.

Όσον αφορά το δεύτερο ζήτημα, χτίσαμε έναν εικονικό παίκτη για το παιχνίδι WWBM με βάση τις ακόλουθες ενότητες:

- Απάντηση Ερώτησης: είναι σε θέση να ανακτήσει χωρία κειμένου που σχετίζονται με ένα συγκεκριμένο ζήτημα που εκφράζεται σε φυσική γλώσσα, με τη χρήση των πηγών ανοικτής γνώσης Wikipedia και DBpedia.
- Βαθμολόγηση Απάντησης: υλοποιεί διάφορα ευρετικά με βάση την ανάλυση των αποτελεσμάτων που επιστρέφονται από την μονάδα QA, προκειμένου να εκχωρήσει μια βαθμολογία σε καθεμία από τις τέσσερις υποψήφιες απαντήσεις.
- Λήψη Αποφάσεων: επιλέγει τη στρατηγική για να παίξει το παιχνίδι, με την αξιοποίηση των βαθμολογιών των τεσσάρων υποψήφιων απαντήσεων, των διαθέσιμων σανίδων σωτηρίας, και το τρέχον επίπεδο του παιχνιδιού.

Ο εικονικός παίκτης ξεπέρασε τους ανθρώπινους παίκτες από την άποψη της μέσης ακρίβειας απαντώντας σωστά σε ερωτήσεις του παιχνιδιού, και από την άποψη της ικανότητας να παίζει αληθινά παιχνίδια με τους κανόνες τους.

Σχεδιάζουμε να ενισχύσουμε τον αλγόριθμο λήψης αποφάσεων, προκειμένου να καταστεί δυνατή μια πιο έξυπνη διαχείριση των σανίδων σωτηρίας, και μια λιγότερο συντηρητική στρατηγική η οποία θα μπορούσε να οδηγήσει σε μια πιο ουδέτερου κινδύνου παίκτη, με την ελπίδα ότι αυτές οι βελτιώσεις θα βελτιώσουν τη συνολική ακρίβεια του συστήματος. Επιπλέον, ειδικά ευρετικά θα μπορούσαν να επινοηθούν για να απαντήσουμε σε ερωτήσεις που είναι σήμερα αναπάντητες για το σύστημα μας (Ενότητα 9.1.1), προκειμένου να βελτιώσουμε περαιτέρω την απόδοσή του.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
module	μονάδα
Question Answering	Απάντηση Ερωτήσεων
Answer Scoring	Βαθμολόγηση Απαντήσεων
lifelines	σανίδες σωτηρίας
Decision Making	Λήψη Αποφάσεων
Artificial Intelligence	Τεχνητή Νοημοσύνη
closed world	κλειστού κόσμου
open world	ανοικτού κόσμου
word games	παιχνίδια με λέξεις
language games	γλωσσικά παιχνίδια
Who Wants to Be a Millionaire?	Ποιος θέλει να γίνει εκατομμυριούχος;
virtual player	εικονικός παίκτης
framework	πλαίσιο εργασίας
part-of-speech tagging	ταυτοποίηση μέρους ενός λόγου
guarantee points	σημεία εγγύησης
poll the audience	δημοσκόπηση του κοινού
phone a friend	τηλέφωνο σε ένα φίλο
Information Retrieval	Ανάκτηση Πληροφορίας
metadata	μεταδεδομένα
World Wide Web	Παγκόσμιος Ιστός
query	Ερώτημα
Natural Language Processing	Επεξεργασία Φυσικής Γλώσσας
cognitive	γνωστικός
Computer Science	Πληροφορική
language engineering	γλωσσική μηχανική
language technology	γλωσσική τεχνολογία
procedural	διαδικαστική
declarative	δηλωτική
inference rules	κανόνες συνεπαγωγής
Logic Programming	Λογικός Προγραμματισμός
Finite State Automata	Αυτόματα Πεπερασμένων Καταστάσεων
Finite State Transition Networks	Δίκτυα Μετάπτωσης Πεπερασμένων Καταστάσεων
Recursive Transition Networks	Αναδρομικά Δίκτυα Μετάπτωσης
Augmented Transition Networks	Διευρυμένα Δίκτυα Μετάπτωσης
action	ενέργεια
chart parsers	συντακτικοί αναλυτές χάρτη
inference	λογική συνεπαγωγή
Definite Clause Grammars	Γραμματικές Οριστικών Προτάσεων
Phrase Structure Grammars	Γραμματικές Φραστικής Δομής
syntactic analysis	συντακτική ανάλυση
parser	συντακτικός αναλυτής
parse tree	συντακτικό δένδρο
transition networks	δίκτυα μετάπτωσης
semantic analysis	σημασιολογική ανάλυση
ambiguity	πολυσημία
context	συμφραζόμενα

scripts	σενάρια
document retrieval	ανάκτηση εγγράφων
information extraction	εξαγωγή πληροφορίας
Data Mining	Εξόρυξη Δεδομένων
text categorization	κατηγοριοποίηση κειμένων
automated synopsis	αυτόματη περίληψη
machine translation	αυτόματη μηχανική μετάφραση
Word Processing Applications	Εφαρμογές Επεξεργασίας Κειμένου
clues	στοιχεία
clue answering	απάντηση στοιχείων
grid filling	γέμισμα πλέγματος
classifier	ταξινομητής
knowledge infusion	έγχυση γνώσης
linguistic competencies	γλωσσικές ικανότητες
reasoning	συλλογισμός
Machine Learning	Μηχανική Μάθηση
domain constraints	περιορισμοί χώρου
newswire reports	εκθέσεις ειδήσεων
cross-lingual	διαγλωσσικό
fact	γεγονός
closed-domain	κλειστού χώρου
open-domain	ανοικτού χώρου
restricted-domain	περιορισμένης περιοχής
core database	βάση δεδομένων πυρήνα
computational linguistics	υπολογιστική γλωσσολογία
Unix Consultant	Σύμβουλος Unix
text understanding	κατανόηση κειμένου
reasoning	αιτιολογία
index	αρχειοθέτηση
corpus	σώμα
shallow	ρηχός
knowledge base	βάση γνώσεων
filter	φίλτρο
string	συμβολοσειρά
answer extraction	εξαγωγή απάντησης
multiagent	πολύ-πρακτορική
agent	πράκτορας
meta-agent	μετα-πράκτορας
data redundancy	πλεονασμός δεδομένων
nuggets	ψήγματα
false positives	ψεύτικα θετικά
logical form	λογική της μορφής
keyword extraction	εξόρυξη λέξεων-κλειδιών
adverbial modifier	επιρρηματικός τροποποιητής
Named Entity Recognizer	Αναγνωριστής Επώνυμης Οντότητας
vector space model	μοντέλο διανυσματικού χώρου
inference technique	τεχνική συμπεράσματος
interactivity	δραστικότητα
clarification	αποσαφήνιση
caching	προσωρινή αποθήκευση

social media	μέσα κοινωνικής δικτύωσης
sentiment analysis	ανάλυση συναισθήματος
thematic roles	θεματικοί ρόλοι
pipeline	αγωγός
stopword removal	αφαίρεση εξαιρούμενων λέξεων
stemming	λέξεις που απορρέουν
lemmatization	λημματοποίηση
word sense disambiguation	αποσαφήνιση της αίσθησης της λέξης
semantic role labeling	σήμανση σημασιολογικού ρόλου
query reformulation	αναδιατύπωση του ερωτήματος
query-rewrite operator	χειριστής επανεγγραφής ερωτήματος
Distributional Semantic Model	Σημασιολογικά Μοντέλα Κατανομών
answer re-ranking	εκ νέου κατάταξη απαντήσεων
Recognizing Textual Entailment	Αναγνώριση Κειμενικής Συνεπαγωγής
Answer Validation Exercise	Άσκηση Επικύρωσης Απάντησης
unification method	μέθοδο ενοποίησης
Linked Data	Συνδεδεμένα Δεδομένα
linguistic analysis	γλωσσολογική ανάλυση
query annotation	σχολιασμός του ερωτήματος
semantic similarity	σημασιολογική ομοιότητα
Game Manager	Διαχειριστής Παιχνιδιού
heuristics	ευρετικά
indexing time	στιγμή ευρετηρίασης
query time	στιγμή του ερωτήματος
Question Analysis	Ανάλυση Ερώτησης
Named Entity Recognition	Αναγνώριση Επώνυμων Οντοτήτων
Search Engines	Μηχανές Αναζήτησης
exact sequence	ακριβής ακολουθία
normalization	κανονικοποίηση
Pivoted Normalized Document Length	Περιστρεφόμενο Κανονικοποιημένο Μήκος Εγγράφου
density	πυκνότητα
Minimal Span Weighting	Ελάχιστη Στάθμιση Διασποράς
distributional	κατανομή
Latent Semantic Analysis	Λανθάνουσα Σημασιολογική Ανάλυση
connectionist	ενωτικό
infobox	κουτί πληροφοριών
subject	θέμα
predicate	κατηγόρημα
object	αντικείμενο
Longest Common Subsequence	Μεγαλύτερη Κοινή Υποαλληλουχία
overlap	επικάλυψη
Exact Substring	Ακριβής Υποαλληλουχία
question expansion	επέκτασης ερωτήματος
regular expressions	κανονικές εκφράσεις
forks	σκέλη
perturbation	διατάραξη
accuracy	ακρίβεια
Random Forest	Τυχαίο Δάσος
overfitting	υπερπροσαρμογή

stratified	στρωματοποιημένη
cross validation	διασταυρωμένη επικύρωση
validation set	σύνολο επικύρωσης
leave-one-out	αφήνουμε ένα έξω
ablation	κατάλυση

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

QA	Question Answering
AS	Answer Scoring
DM	Decision Making
GM	Game Manager
AI	Artificial Intelligence
WWBM	Who Wants to Be a Millionaire
WWW	World Wide Web
NLP	Natural Language Processing
FSA	Finite State Automata
FSTN	Finite State Transition Networks
RTN	Recursive Transition Networks
ATN	Augmented Transition Networks
DCG	Definite Clause Grammars
PSG	Phrase Structure Grammars
ML	Machine Learning
UC	Unix Consultant
DSM	Distributional Semantic Model
RTE	Recognizing Textual Entailment
AVE	Answer Validation Exercise
NER	Named Entity Recognition
PNDL	Pivoted Normalized Document Length
LSA	Latent Semantic Analysis
LCS	Longest Common Subsequence
ES	Exact Substring
RF	Random Forest

ΠΑΡΑΡΤΗΜΑ Ι

QuestionParser.java

```
package com.acipi.dit.wwtbam.helpers;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

/**
 * @author Altin Cipi
 *
 */
public class QuestionParser
{
	public static final String LIBRARY_LOCATION =
"C:\\Users\\haris\\Desktop\\library\\";

	public static void read(int fileNum) throws Exception
	{
		BufferedReader in = new BufferedReader(new InputStreamReader(new
FileInputStream(new File(LIBRARY_LOCATION + "qfile" + fileNum + ".txt")), "Windows-
1253"));
		BufferedWriter out = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("Questions.txt", true), "utf-8"));

		String line = in.readLine();

		for (int i = 1; i <= 9; i++)
		{
			String question = line.substring(line.indexOf("q_" + i + "=") + 4,
line.indexOf("&a1_" + i + "=")).replace('+', ' ').replace("%22", "\"").replace("%27",
"'");
			String answerA = line.substring(line.indexOf("&a1_" + i + "=") + 6,
line.indexOf("&a2_" + i + "=")).replace('+', ' ').replace("%22", "\"").replace("%27",
"'");
			String answerB = line.substring(line.indexOf("&a2_" + i + "=") + 6,
line.indexOf("&a3_" + i + "=")).replace('+', ' ').replace("%22", "\"").replace("%27",
"'");
			String answerC = line.substring(line.indexOf("&a3_" + i + "=") + 6,
line.indexOf("&a4_" + i + "=")).replace('+', ' ').replace("%22", "\"").replace("%27",
"'");
			String answerD = line.substring(line.indexOf("&a4_" + i + "=") + 6,
line.indexOf("&rA_" + i + "=")).replace('+', ' ').replace("%22", "\"").replace("%27",
"'");
			String correctAnswer = line.substring(line.indexOf("&rA_" + i + "=")
+ 6, line.indexOf("&vote1_" + i + "="));
			String voteA = line.substring(line.indexOf("&vote1_" + i + "=") + 9,
line.indexOf("&vote2_" + i + "="));
			String voteB = line.substring(line.indexOf("&vote2_" + i + "=") + 9,
line.indexOf("&vote3_" + i + "="));
			String voteC = line.substring(line.indexOf("&vote3_" + i + "=") + 9,
line.indexOf("&vote4_" + i + "="));
			String voteD = line.substring(line.indexOf("&vote4_" + i + "=") + 9,
line.indexOf("&fiftyVote1_" + i + "="));
```

```
                String fiftyA = line.substring(line.indexOf("&fiftyVote1_" + i + "=")
+ 14, line.indexOf("&fiftyVote2_" + i + "="));
                String fiftyB = line.substring(line.indexOf("&fiftyVote2_" + i + "=")
+ 14, line.indexOf("&fiftyVote3_" + i + "="));
                String fiftyC = line.substring(line.indexOf("&fiftyVote3_" + i + "=")
+ 14, line.indexOf("&fiftyVote4_" + i + "="));
                String fiftyD = line.substring(line.indexOf("&fiftyVote4_" + i + "=")
+ 14, line.indexOf("&friendText_" + i + "="));
                String friendCall = line.substring(line.indexOf("&friendText_" + i +
"=") + 14, line.indexOf("&fiftyfifty1_" + i + "=")).replace('+', ' ').replace("%22",
"\"").replace("%27", "'");

                out.write(i + "\n");
                out.write(question + "\n");
                out.write(answerA + "\n");
                out.write(answerB + "\n");
                out.write(answerC + "\n");
                out.write(answerD + "\n");
                out.write(correctAnswer + "\n");
                out.write(voteA + "\n");
                out.write(voteB + "\n");
                out.write(voteC + "\n");
                out.write(voteD + "\n");
                out.write(fiftyA + "\n");
                out.write(fiftyB + "\n");
                out.write(fiftyC + "\n");
                out.write(fiftyD + "\n");
                out.write(friendCall + "\n");
                out.write("\n");
            }
            for (int i = 10; i <= 15; i++)
            {
                String question = line.substring(line.indexOf("q_" + i + "=") + 5,
line.indexOf("&a1_" + i + "=")).replace('+', ' ').replace("%22", "\"").replace("%27",
"'");
                String answerA = line.substring(line.indexOf("&a1_" + i + "=") + 7,
line.indexOf("&a2_" + i + "=")).replace('+', ' ').replace("%22", "\"").replace("%27",
"'");
                String answerB = line.substring(line.indexOf("&a2_" + i + "=") + 7,
line.indexOf("&a3_" + i + "=")).replace('+', ' ').replace("%22", "\"").replace("%27",
"'");
                String answerC = line.substring(line.indexOf("&a3_" + i + "=") + 7,
line.indexOf("&a4_" + i + "=")).replace('+', ' ').replace("%22", "\"").replace("%27",
"'");
                String answerD = line.substring(line.indexOf("&a4_" + i + "=") + 7,
line.indexOf("&rA_" + i + "=")).replace('+', ' ').replace("%22", "\"").replace("%27",
"'");
                String correctAnswer = line.substring(line.indexOf("&rA_" + i + "=")
+ 7, line.indexOf("&vote1_" + i + "="));
                String voteA = line.substring(line.indexOf("&vote1_" + i + "=") + 10,
line.indexOf("&vote2_" + i + "="));
                String voteB = line.substring(line.indexOf("&vote2_" + i + "=") + 10,
line.indexOf("&vote3_" + i + "="));
                String voteC = line.substring(line.indexOf("&vote3_" + i + "=") + 10,
line.indexOf("&vote4_" + i + "="));
                String voteD = line.substring(line.indexOf("&vote4_" + i + "=") + 10,
line.indexOf("&fiftyVote1_" + i + "="));
                String fiftyA = line.substring(line.indexOf("&fiftyVote1_" + i + "=")
+ 15, line.indexOf("&fiftyVote2_" + i + "="));
                String fiftyB = line.substring(line.indexOf("&fiftyVote2_" + i + "=")
+ 15, line.indexOf("&fiftyVote3_" + i + "="));
```

```
                String fiftyC = line.substring(line.indexOf("&fiftyVote3_" + i + "=")
+ 15, line.indexOf("&fiftyVote4_" + i + "="));
                String fiftyD = line.substring(line.indexOf("&fiftyVote4_" + i + "=")
+ 15, line.indexOf("&friendText_" + i + "="));
                String friendCall = line.substring(line.indexOf("&friendText_" + i +
"=") + 15, line.indexOf("&fiftyfifty1_" + i + "=")).replace('+', ' ').replace("%22",
"\"").replace("%27", "'");

                out.write(i + "\n");
                out.write(question + "\n");
                out.write(answerA + "\n");
                out.write(answerB + "\n");
                out.write(answerC + "\n");
                out.write(answerD + "\n");
                out.write(correctAnswer + "\n");
                out.write(voteA + "\n");
                out.write(voteB + "\n");
                out.write(voteC + "\n");
                out.write(voteD + "\n");
                out.write(fiftyA + "\n");
                out.write(fiftyB + "\n");
                out.write(fiftyC + "\n");
                out.write(fiftyD + "\n");
                out.write(friendCall + "\n");
                out.write("\n");
            }

            in.close();
            out.close();
        }

        public static void main(String[] args) throws Exception
        {
            for (int i = 634; i <= 1634; i++)
            {
                read(i);
            }
        }
}
```

ΠΑΡΑΡΤΗΜΑ ΙΙ

AnswerScoring.java

```
package com.acipi.dit.wwtbam.modules;

import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.commons.lang.StringUtils;

import com.acipi.dit.wwtbam.core.GoogleResult;

/**
 * @author Altin Cipi
 *
 */
public class AnswerScoring
{
	public static final int		FOUND_AT_TITLE_BONUS			= 10;
	public static final int		REGEX_TEXT_NUMBER_LIMIT	= 1;

	public static final String	NEGATIVE_QUESTION_IDENTIFIER = "δεν";
	public static final String[] IRRELEVANT_WORDS_UPPER		= new String[] {
"Τον", "Την", "Το", "Τα", "Τη", "Στη", "Στην", "Στο", "Ο", "Η", "Της", "Του", "Τους",
"Οι", "Τις", "Στον" };
	public static final String[] IRRELEVANT_WORDS_LOWER		= new String[] {
"τον", "την", "το", "τα", "τη", "στη", "στην", "στο", "ο", "η", "της", "του", "τους",
"οι", "τις", "στον" };

	private static int findAllOccurences(String text, String words, boolean
useRegex)
	{
		return findOccurences(text, words, useRegex) + findOccurences(text,
words.toLowerCase(), useRegex);
	}

	private static int findOccurences(String text, String words, boolean useRegex)
	{
		int counter = 0;

		if (useRegex)
		{
			String[] tokens = words.split(" ");
			String regexQuery = "";
			for (String token : tokens)
			{
				regexQuery += "(.*" + token + ".*)";
			}

			Pattern pattern = Pattern.compile(regexQuery);
			Matcher matcher = pattern.matcher(text);

			while (matcher.find())
			{
				counter++;
			}
		}
		else
		{
```

```
                counter = StringUtils.countMatches(text, words);
            }

            return counter;
        }

        private static String[] removeSameFromBeggining(String[] texts)
        {
            String a = texts[0];
            String b = texts[1];
            String c = texts[2];
            String d = texts[3];

            String[] tokens = a.split(" ");

            for (String token : tokens)
            {
                if (a.startsWith(token) && b.startsWith(token) && c.startsWith(token)
&& d.startsWith(token))
                {
                    a = a.replaceFirst(token + " ", "");
                    b = b.replaceFirst(token + " ", "");
                    c = c.replaceFirst(token + " ", "");
                    d = d.replaceFirst(token + " ", "");
                }
                else
                {
                    break;
                }
            }

            return new String[] { a, b, c, d };
        }

        private static String[] removeSameFromEnd(String[] texts)
        {
            String a = texts[0];
            String b = texts[1];
            String c = texts[2];
            String d = texts[3];

            String[] tokens = a.split(" ");

            for (int i = tokens.length - 1; i >= 0; i--)
            {
                if (a.endsWith(tokens[i]) && b.endsWith(tokens[i]) &&
c.endsWith(tokens[i]) && d.endsWith(tokens[i]))
                {
                    a = StringUtils.removeEnd(a, " " + tokens[i]);
                    b = StringUtils.removeEnd(b, " " + tokens[i]);
                    c = StringUtils.removeEnd(c, " " + tokens[i]);
                    d = StringUtils.removeEnd(d, " " + tokens[i]);
                }
                else
                {
                    break;
                }
            }

            return new String[] { a, b, c, d };
        }
```

```
	private static String[] simplifyAnswers(String[] answers)
	{
		String[] simplifiedAnswers = removeSameFromBeggining(answers);
		simplifiedAnswers = removeSameFromEnd(simplifiedAnswers);

		for (int i = 0; i < answers.length; i++)
		{
			for (String word : IRRELEVANT_WORDS_UPPER)
			{
				simplifiedAnswers[i] =
StringUtils.removeStart(simplifiedAnswers[i], word + " ");
			}
			for (String word : IRRELEVANT_WORDS_LOWER)
			{
				simplifiedAnswers[i] =
StringUtils.removeStart(simplifiedAnswers[i], word + " ");
			}
		}

		return simplifiedAnswers;
	}

	public static Integer[] scoreAnswers(String question, List<GoogleResult>
results, String[] answers, boolean useRegex)
	{
		Integer[] answerScores = new Integer[] { 0, 0, 0, 0 };

		String[] simplifiedAnswers = simplifyAnswers(answers);

		for (int k = 0; k < results.size(); k++)
		{
			if (useRegex && k >= REGEX_TEXT_NUMBER_LIMIT)
			{
				break;
			}

			for (int i = 0; i < simplifiedAnswers.length; i++)
			{
				if (results.get(k).getLinkText().contains(simplifiedAnswers[i]))
				{
					answerScores[i] += FOUND_AT_TITLE_BONUS;
				}
			}

			for (String passage : results.get(k).getPassages())
			{
				for (int i = 0; i < simplifiedAnswers.length; i++)
				{
					answerScores[i] += findAllOccurences(passage,
simplifiedAnswers[i], useRegex);
				}
			}
		}

		if (StringUtils.containsIgnoreCase(question,
NEGATIVE_QUESTION_IDENTIFIER))
		{
			for (int i = 0; i < answerScores.length; i++)
			{
				answerScores[i] = -1 * answerScores[i];
```

```
                }
            }

            System.out.println("[" + answers[0] + ", " + answers[1] + ", " +
answers[2] + ", " + answers[3] + "]");
            System.out.println("[" + simplifiedAnswers[0] + ", " +
simplifiedAnswers[1] + ", " + simplifiedAnswers[2] + ", " + simplifiedAnswers[3] +
"]");
            System.out.println("[" + answerScores[0] + ", " + answerScores[1] + ", " +
answerScores[2] + ", " + answerScores[3] + "]");

            return answerScores;
        }
}
```

DecisionMaking.java

```
package com.acipi.dit.wwtbam.modules;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import com.acipi.dit.wwtbam.core.GoogleResult;
import com.acipi.dit.wwtbam.core.Question;
import com.acipi.dit.wwtbam.core.SortableValue;
import com.acipi.dit.wwtbam.helpers.UtilityHelper;

/**
 * @author Altin Cipi
 *
 */
public class DecisionMaking
{
	public static final int	threshold			= 1;

	public static final int	GIVE_ANSWER				= 1;
	public static final int	QUIT_GAME			= 2;
	public static final int	USE_POLL_THE_AUDIENCE	= 3;
	public static final int	USE_PHONE_A_FRIEND		= 4;
	public static final int	USE_FIFTY_FIFTY		= 5;

	public static int makeDesicion(Question q, List<GoogleResult> results, boolean
fiftyFiftyUsed, boolean pollTheAudienceUsed, boolean phoneAFriendUsed)
	{
		Integer[] answerScores = AnswerScoring.scoreAnswers(q.getQuestion(),
results, q.getAnswers(), false);

		List<SortableValue> tempScores = new ArrayList<SortableValue>();
		tempScores.add(new SortableValue(answerScores[0], 0));
		tempScores.add(new SortableValue(answerScores[1], 1));
		tempScores.add(new SortableValue(answerScores[2], 2));
		tempScores.add(new SortableValue(answerScores[3], 3));

		Collections.sort(tempScores);

		if (tempScores.get(0).getValue() - tempScores.get(1).getValue() <
threshold)
		{
			answerScores = AnswerScoring.scoreAnswers(q.getQuestion(), results,
q.getAnswers(), true);

			tempScores = new ArrayList<SortableValue>();
			tempScores.add(new SortableValue(answerScores[0], 0));
			tempScores.add(new SortableValue(answerScores[1], 1));
			tempScores.add(new SortableValue(answerScores[2], 2));
			tempScores.add(new SortableValue(answerScores[3], 3));

			Collections.sort(tempScores);
		}

		q.setSelected(tempScores.get(0).getIndex());

		if (tempScores.get(0).getValue() - tempScores.get(1).getValue() <
threshold)
		{
```

```
                if (!pollTheAudienceUsed)
                {
                    return USE_POLL_THE_AUDIENCE;
                }
                if (!phoneAFriendUsed)
                {
                    return USE_PHONE_A_FRIEND;
                }
                if (!fiftyFiftyUsed)
                {
                    return USE_FIFTY_FIFTY;
                }

                if (q.getLevel() == 6 || q.getLevel() == 11)
                {
                    return GIVE_ANSWER;
                }
                else
                {
                    return QUIT_GAME;
                }
            }
            else
            {
                return GIVE_ANSWER;
            }
        }

        public static void useFiftyFifty(Question q)
        {
            q.setSelected(q.getFiftyFifty().get(0).getIndex());
        }

        public static void usePhoneAFriend(Question q)
        {
            q.setSelected(UtilityHelper.parseFriendsAnswer(q.getFriendCall()));
        }
}
```

QuestionAnswering.java

```
package com.acipi.dit.wwtbam.modules;

import java.util.ArrayList;
import java.util.List;

import org.apache.commons.lang.StringUtils;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

import com.acipi.dit.wwtbam.core.GoogleResult;

/**
 * @author Altin Cipi
 *
 */
public class QuestionAnswering
{
	public static final String GOOGLE_SEARCH_API_URL 		= 
"https://www.google.com/search?";
	public static final int 	DEFAULT_RESULTS_NUMBER 		= 30;
	public static final String DEFAULT_ELEMENT_TAG 		= "div";
	public static final String DEFAULT_WIKIPEDIA_ELEMENT_TAG  = "div#mw-content-
text";
	public static final String DEFAULT_WIKIPEDIA_SPECIFIC_DOMAIN = "wikipedia.org";

	public static List<GoogleResult> findResults(String question, Integer 
resultsNumber, String specificDomain, String elementTag)
	{
		StringBuffer request = new StringBuffer(GOOGLE_SEARCH_API_URL);
		request.append("q=" + cleanQuestion(question));
		request.append("&num=100");
		request.append(StringUtils.isNotEmpty(specificDomain) ? "&as_sitesearch=" 
+ specificDomain : "");

		List<GoogleResult> results = new ArrayList<GoogleResult>();

		if (resultsNumber == null)
		{
			resultsNumber = DEFAULT_RESULTS_NUMBER;
		}

		try
		{
			System.out.println("Sending request..." + request.toString());

			Document doc = 
Jsoup.connect(request.toString()).userAgent("Mozilla/5.0 (compatible; Googlebot/2.1; 
+http://www.google.com/bot.html)").timeout(5000).get();

			Elements links = doc.select("h3.r > a");

			for (Element link : links)
			{
				String linkHref = link.attr("href").substring(7, 
link.attr("href").indexOf("&")).replace("%25", "%");
				String linkText = link.text();
```

```
                    Document refDoc = null;

                    try
                    {
                         refDoc =
Jsoup.connect(linkHref).userAgent("Mozilla/5.0").timeout(5000).get();
                    }
                    catch (Exception e)
                    {
                         continue;
                    }

                    System.out.println(linkText + " (" + linkHref + ")");

                    GoogleResult result = new GoogleResult(linkHref, linkText);

                    Elements textPassages =
refDoc.select(StringUtils.isNotEmpty(elementTag) ? elementTag : DEFAULT_ELEMENT_TAG);
                    for (Element textPassage : textPassages)
                    {
                         result.getPassages().add(textPassage.text());
                    }

                    results.add(result);

                    if (results.size() == resultsNumber)
                    {
                         break;
                    }
               }
          }
          catch (Exception e)
          {
               e.printStackTrace();
          }

          System.out.println("Retrieved " + results.size() + " results");

          return results;
     }

     private static String cleanQuestion(String question)
     {
          String cleanQuestion = question;

          if (cleanQuestion.contains("\""))
          {
               cleanQuestion = cleanQuestion.substring(cleanQuestion.indexOf("\"") +
1, cleanQuestion.lastIndexOf("\""));
          }

          if (cleanQuestion.contains("..."))
          {
               cleanQuestion = cleanQuestion.replace("...", "");
          }

          if (cleanQuestion.contains(";"))
          {
               cleanQuestion = cleanQuestion.replace(";", "");
          }
```

```
            cleanQuestion = cleanQuestion.replace(" ", "+");

            return cleanQuestion;
        }
}
```

ΑΝΑΦΟΡΕΣ

[1] Piero Molino, Pasquale Lops, Giovanni Semeraro, Marco de Gemmis, Pierpaolo Basile, Playing with knowledge: A virtual player for “Who Wants to Be a Millionaire?” that leverages question answering techniques, in: Artificial Intelligence, 2015

[2] N. Aggarwal, P. Buitelaar, A system description of natural language query over DBpedia, in: C. Unger, P. Cimiano, V. Lopez, E. Motta, P. Buitelaar, R. Cyganiak (Eds.), Proceedings of Interacting with Linked Data (ILD 2012), Workshop Co-Located with ESWC 2012, in: CEUR Workshop Proceedings, vol. 913, 2012, pp. 96–99.

[3] K. Ahn, J. Bos, D. Kor, M. Nissim, B.L. Webber, J.R. Curran, Question answering with QED at TREC 2005, in: E.M. Voorhees, L.P. Buckland (Eds.), Proceedings of the Fourteenth Text REtrieval Conference, TREC 2005, in: Special Publications, vol. 500-266, National Institute of Standards and Technology (NIST), 2005.

[4] P. Basile, M. de Gemmis, P. Lops, G. Semeraro, Solving a complex language game by using knowledge-based word associations discovery, IEEE Trans. Comput. Intell. AI Games (2015), http://dx.doi.org/10.1109/TCIAIG.2014.2355859, in press.

[5] J. Berant, A. Chou, R. Frostig, P. Liang, Semantic parsing on freebase from question–answer pairs, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18–21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, ACL, 2013, pp. 1533–1544, a meeting of SIGDAT, a special interest group of the ACL.

[6] C. Bizer, The emerging Web of linked data, IEEE Intell. Syst. 24 (2009) 87–92.

[7] E. Breck, J.D. Burger, L. Ferro, L. Hirschman, D. House, M. Light, I. Mani, How to evaluate your question answering system every day and still get real work done, in: Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000, European Language Resources Association, 2000, pp. 1495–1500.

[8] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.

[9] E. Cabrio, A.P. Aprosio, J. Cojan, B. Magnini, F. Gandon, A. Lavelli, QAKiS@QALD-2, in: C. Unger, P. Cimiano, V. Lopez, E. Motta, P. Buitelaar, R. Cyganiak (Eds.), Proceedings of Interacting with Linked Data (ILD 2012), Workshop Co-Located with EWSC 2012, in: CEUR Workshop Proceedings, vol. 913, 2012, pp. 87–95.

[10] J.J. Castillo, The contribution of FaMAF at QA@CLEF 2008: answer validation exercise, in: Working Notes of the CLEF 2008 Workshop, 2008, pp. 17–19.

[11] J. Chen, A. Diekema, M.D. Taffet, N.J. McCracken, N.E. Ozgencil, O. Yilmazel, E.D. Liddy, Question answering: CNLP at the TREC-10 question answering track, in: Text Retrieval Conference, in: Special Publications, vol. 500-274, National Institute of Standards and Technology (NIST), 2001.

[12] P. Cimiano, M. Minock, Natural language interfaces: what is the problem? – a data-driven quantitative analysis, in: H. Horacek, E. Métais, R. Muñoz, M. Wolska (Eds.), Natural Language Processing and Information Systems, 14th International Conference on Applications of Natural Language to Information Systems, NLDB 2009, in: Lecture Notes in Computer Science, vol. 5723, Springer, 2010, pp. 192–206, revised papers.

[13] I. Dagan, O. Glickman, B. Magnini, The PASCAL recognising textual entailment challenge, in: J.Q. Candela, I. Dagan, B. Magnini, F. d’Alché Buc (Eds.), Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, in: Lecture Notes in Computer Science, vol. 3944, Springer, 2005, revised selected papers.

[14] D. Damljanovic, M. Agatonovic, H. Cunningham, FREyA: an interactive way of querying linked data using natural language, in: R. Garcia-Castro, D. Fensel, G. Antoniou (Eds.), The Semantic Web: ESWC 2011 Workshops, in: Lecture Notes in Computer Science, vol. 7117, Springer, 2012, pp. 125–138, revised selected papers.

[15] S. Dumais, M. Banko, E. Brill, J. Lin, A. Ng, Web question answering: is more always better?, in: Proceedings of the 25th ACM SIGIR Conference, SIGIR’02, ACM, New York, NY, USA, 2002, pp. 291–298.

[16] M. Ernandes, G. Angelini, M. Gori, WebCrow: a Web-based system for crossword solving, in: M.M. Veloso, S. Kambhampati (Eds.), AAAI, AAAI Press/The MIT Press, 2005, pp. 1412–1417.

[17] A. Fader, L. Zettlemoyer, O. Etzioni, Open question answering over curated and extracted knowledge bases, in: S.A. Macskassy, C. Perlich, J. Leskovec, W. Wang, R. Ghani (Eds.), The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, ACM, 2014, pp. 1156–1165.

[18] D. Ferrucci, A. Levas, S. Bagchi, D. Gondek, E.T. Mueller, Watson: beyond Jeopardy!, Artif. Intell. (2013) 93–105.

[19] D.A. Ferrucci, E.W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J.W. Murdock, E. Nyberg, J.M. Prager, N. Schlaefer, C.A. Welty, Building Watson: an overview of the DeepQA project, AI Mag. 31 (2010) 59–79.
[20] E. Gabrilovich, S. Markovitch, Computing semantic relatedness using Wikipedia-based explicit semantic analysis, in: M.M. Veloso (Ed.), Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007, Hyderabad, India, January 6–12, 2007, Morgan Kaufmann, 2007, pp. 1606–1611.
[21] I. Glöckner, University of Hagen at QA@CLEF: answer validation exercise, in: Working Notes of the CLEF 2008 Workshop, 2008, pp. 17–19.
[22] S.M. Harabagiu, S. Maiorano, Finding answers in large collections of texts: paragraph indexing + abductive inference, in: Proceedings of the AAAI Fall Symposium on Question Answering, AAAI, 1999, pp. 63–71.
[23] S.M. Harabagiu, D.I. Moldovan, M. Pas¸ ca, R. Mihalcea, M. Surdeanu, R.C. Bunescu, R. Girju, V. Rus, P. Morarescu, FALCON: boosting knowledge for answer engines, in: Text Retrieval Conference, in: Special Publications, vol. 500-249, National Institute of Standards and Technology (NIST), 2000, pp. 479–488.
[24] S.M. Harabagiu, M. Pas¸ ca, S.J. Maiorano, Experiments with open-domain textual question answering, in: Proceedings of the 18th Conference on Computational Linguistics – Volume 1, COLING'00, Association for Computational Linguistics, Stroudsburg, PA, USA, 2000, pp. 292–298.
[25] E.H. Hovy, L. Gerber, U. Hermjakob, M. Junk, C.Y. Lin, Question answering in Webclopedia, in: Text Retrieval Conference, in: Special Publications, vol. 500-249, National Institute of Standards and Technology (NIST), 2000, pp. 655–664.
[26] S.K. Lam, D.M. Pennock, D. Cosley, S. Lawrence, 1 billion pages = 1 million dollars? Mining the Web to play "Who Wants to Be a Millionaire?", in: C. Meek, U. Kjærulff (Eds.), Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence, Morgan Kaufmann, 2003, pp. 337–345.
[27] J.J. Lin, An exploration of the principles underlying redundancy-based factoid question answering, ACM Trans. Inf. Syst. 25 (2007).
[28] M.L. Littman, Review: computer language games, in: T.A. Marsland, I. Frank (Eds.), Computers and Games, 2nd Int. Conf., in: Lecture Notes in Computer Science, vol. 2063, Springer, 2000, pp. 396–404, revised papers.
[29] M.L. Littman, G.A. Keim, N. Shazeer, A probabilistic approach to solving crossword puzzles, Artif. Intell. 134 (2002) 23–55.
[30] T. Liu, Learning to Rank for Information Retrieval, Springer, 2011.
[31] V. Lopez, M. Fernández, E. Motta, N. Stieler, PowerAqua: supporting users in querying and exploring the Semantic Web, Semant. Web 3 (2012) 249–265.
[32] V. Lopez, C. Unger, P. Cimiano, E. Motta, Evaluating question answering over linked data, J. Web Semant. 21 (2013) 3–13.
[33] B. Magnini, M. Negri, R. Prevete, H. Tanev, Is it the right answer? Exploiting Web redundancy for answer validation, in: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, ACL, 2002, pp. 425–432.
[34] C.D. Manning, P. Raghavan, H. Schtze, Introduction to Information Retrieval, Cambridge University Press, New York, NY, USA, 2008.
[35] M. Maybury, O. Stock, W. Wahlster, Intelligent interactive entertainment grand challenges, IEEE Intell. Syst. 21 (2006) 14–18.
[36] P. Molino, P. Basile, QuestionCube: a framework for question answering, in: G. Amati, C. Carpineto, G. Semeraro (Eds.), Proceedings of the 3rd Italian Information Retrieval (IIR) Workshop, Bari, Italy, January 26–27, 2012, in: CEUR Workshop Proceedings, vol. 835, CEUR-WS.org, 2012, pp. 167–178.
[37] P. Molino, P. Basile, A. Caputo, P. Lops, G. Semeraro, Exploiting distributional semantic models in question answering, in: IEEE International Conference on Semantic Computing, IEEE, 2012, pp. 146–153.
[38] P. Molino, P. Basile, C. Santoro, P. Lops, M. de Gemmis, G. Semeraro, A virtual player for "Who Wants to Be a Millionaire?" based on question answering, in: M. Baldoni, C. Baroglio, G. Boella, R. Micalizio (Eds.), AI*IA 2013: Advances in Artificial Intelligence – XIIIth International Conference of the Italian Association for Artificial Intelligence, in: Lecture Notes in Computer Science, vol. 8249, Springer, 2013, pp. 205–216.
[39] C. Monz, Minimal span weighting retrieval for question answering, in: R. Gaizauskas, M. Greenwood, M. Hepple (Eds.), Proceedings of the SIGIR 2004 Workshop on Information Retrieval for Question Answering, pp. 23–30.
[40] M. Pas¸ ca, Open-Domain Question Answering from Large Text Collections, Studies in Computational Linguistics, CSLI Publications, 2003.

[41] A. Peñas, E.H. Hovy, P. Forner, Á. Rodrigo, R.F.E. Sutcliffe, R. Morante, QA4MRE 2011–2013: overview of question answering for machine reading evaluation, in: P. Forner, H. Müller, R. Paredes, P. Rosso, B. Stein (Eds.), Information Access Evaluation. Multilinguality, Multimodality, and Visualization – 4th International Conference of the CLEF Initiative, CLEF 2013, in: Lecture Notes in Computer Science, vol. 8138, Springer, 2013, pp. 303–320.
[42] F. Perea, J. Puerto, Dynamic programming analysis of the TV game "Who Wants to Be a Millionaire?", Eur. J. Oper. Res. 183 (2007) 805–811.
[43] S. Robertson, H. Zaragoza, The probabilistic relevance framework: BM25 and beyond, Found. Trends Inf. Retr. 3 (2009) 333–389.
[44] J. Rocchio, Relevance feedback information retrieval, in: G. Salton (Ed.), The SMART Retrieval System – Experiments in Automated Document Processing, Prentice-Hall, Englewood Cliffs, NJ, 1971, pp. 313–323.
[45] Á. Rodrigo, A. Peñas, F. Verdejo, UNED at answer validation exercise 2007, in: C. Peters, V. Jijkoun, T. Mandl, H. Müller, D.W. Oard, A. Peñas, V. Petras, D. Santos (Eds.), Advances in Multilingual and Multimodal Information Retrieval, 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007, in: Lecture Notes in Computer Science, vol. 5152, Springer, 2008, pp. 404–409, revised selected papers.
[46] Á. Rodrigo, A. Peñas, F. Verdejo, Overview of the answer validation exercise 2008, in: C. Peters, T. Deselaers, N. Ferro, J. Gonzalo, G.J.F. Jones, M. Kurimo, T. Mandl, A. Peñas, V. Petras (Eds.), Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, in: Lecture Notes in Computer Science, vol. 5706, Springer, 2009, pp. 296–313, revised selected papers.
[47] S. Royce, B. Straits, Approaches to Social Research, 3rd edition, Oxford University Press, New York, 1999.
[48] M. Sahlgren, An introduction to random indexing, in: Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, 2005.
[49] G. Semeraro, M. de Gemmis, P. Lops, P. Basile, An artificial player for a language game, IEEE Intell. Syst. 27 (2012) 36–43.
[50] J.A. Shaw, E.A. Fox, Combination of multiple searches, in: Proceedings of the Second Text REtrieval Conference, TREC-2, in: Special Publications, vol. 500-215, National Institute of Standards and Technology (NIST), 1994, pp. 243–252.
[51] A. Singhal, C. Buckley, M. Mitra, Pivoted document length normalization, in: Proceedings of the 19th ACM SIGIR Conference, SIGIR'96, ACM, New York, NY, USA, 1996, pp. 21–29.
[52] P. Smolensky, Tensor product variable binding and the representation of symbolic structures in connectionist systems, Artif. Intell. 46 (1990) 159–216.
[53] GREEN JR, Bert F; et al. (1961). "Baseball: an automatic question-answerer.". *western joint IRE-AIEE-ACM computer conference*: 219–224.
[54] Woods, William A; Kaplan, R. (1977). "Lunar rocks in natural English: Explorations in natural language question answering". *Linguistic structures processing 5*. **5**: 521–569.
[55] Hirschman, L. & Gaizauskas, R. (2001) Natural Language Question Answering. The View from Here. Natural Language Engineering (2001), 7:4:275-300 Cambridge University Press.
[56] Galitsky B, Pampapathi R. Can many agents answer questions better than one. *First Monday*. 2005.
[57] Lin, J. (2002). The Web as a Resource for Question Answering: Perspectives and Challenges. In Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002).
[58] Galitsky, Boris (2003). *Natural Language Question Answering System: Technique of Semantic Headers*. International Series on Advanced Intelligence. Volume 2. Australia: Advanced Knowledge International.
[59] Maybury, M. T. editor. 2004. New Directions in Question Answering. AAAI/MIT Press.
[60] BitCrawl by Hobson Lane at the Wayback Machine (archived October 27, 2012).
[61] Perera, R. and Perera, U. 2012. Towards a thematic role based target identification model for question answering.
[62] Bahadorreza Ofoghi; John Yearwood & Liping Ma (2008). *The impact of semantic class identification and semantic role labeling on natural language answer extraction*. The 30th European Conference on Information Retrieval (ECIR'08). Springer Berlin Heidelberg. pp. 430–437.
[63] Bahadorreza Ofoghi; John Yearwood & Liping Ma (2009). "The impact of frame semantic annotation levels, frame-alignment techniques, and fusion methods on factoid answer processing". *Journal of the American Society for Information Science and Technology*. **60** (2): 247–263.