Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή
στα Πληροφοριακά και Επικοινωνιακά Συστήματα

Μελέτη και Αξιολόγηση Πλατφορμών Διαλογικών Πρακτόρων για την Ενσωμάτωση Διεπαφών Φυσικής Γλώσσας σε Εφαρμογές

Παναγιώτης Χαλατσάκος

Επιβλέπων Καθηγητής
Βασίλης Ζαφείρης

Μάιος 2017

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μελέτη και Αξιολόγηση Πλατφορμών Διαλογικών Πρακτόρων για την Ενσωμάτωση Διεπαφών Φυσικής Γλώσσας σε Εφαρμογές

Παναγιώτης Χαλατσάκος

Επιβλέπων Καθηγητής
Βασίλης Ζαφείρης

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στα Πληροφοριακά Συστήματα

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Μάιος 2017

Περίληψη

Στόχος της μεταπτυχιακής διατριβής είναι η μελέτη και αξιολόγηση των σημαντικότερων πλατφορμών των διαλογικών πρακτόρων που κυκλοφορούν αυτή την χρονική περίοδο (Μάιος 2016 – Μάιος 2017) στην αγορά και ειδικότερα των πακέτων wit.ai, api.ai και Microsoft bot framework. Τα πακέτα αυτά χρησιμοποιούν την επεξεργασία φυσικής γλώσσας και ειδικότερα υπηρεσίες κατανόησης και επεξεργασίας φυσικής γλώσσας για την διεπαφή χρήστη και προγράμματος χρησιμοποιώντας φυσική γλώσσα και διάλογο ως τρόπο επικοινωνίας.

Τέθηκαν τα εξής ερευνητικά ερωτήματα:

1. Ποια είναι τα κυριότερα bot frameworks και πως συγκρίνονται βάσει των χαρακτηριστικών και δυνατοτήτων τους (π.χ. αρχιτεκτονικής, προγραμματιστικού μοντέλο, NLP υπηρεσιών που ενσωματώνουν);
2. Πως μπορεί να αξιοποιηθεί ένα bot framework για την βελτίωση της διεπαφής χρήστη μιας εφαρμογής;
3. Πόσο πρακτική και αποτελεσματική είναι η ενσωμάτωση σε μια εφαρμογή ενός προσωπικού βοηθού βασισμένου σε κάποιο από τα διαθέσιμα bot frameworks;

Η μέθοδος σύγκρισης των frameworks έγινε μέσω των χαρακτηριστικών που προσφέρει το κάθε πακέτο καθώς και στην εμπειρία χρήστη τόσο του κατασκευαστή ενός διαλογικού συστήματος όσο και του τελικού χρήστη. Τέλος δημιουργήθηκε ένας προσωπικός βοηθός στοιχημάτων ο οποίος μέσω του φυσικού διαλόγου προσπαθεί να διεκπεραιώσει ένα στοίχημα ή να φέρει παρεμφερείς πληροφορίες, ελαττώνοντας το χρόνο που ο χρήστης θα έκανε με μια παραδοσιακή εφαρμογή. Η ενσωμάτωση του διαλογικού βοηθού έγινε στο slack ένα κανάλι διαλογικών βοηθών.

Summary

The goals of this M.Sc. dissertation were to evaluate the frameworks of conversational agents (chatbots) that are freely available to the market between May 2016 and May 2017, especially wit.ai, api.ai and Microsoft bot framework. These frameworks use natural language processing at their core, which is essential for understanding and producing natural language in order to interface and communicate between the user and the software that they are designed for.

The following goals for this dissertation were set:

1. What are the most crucial bot frameworks and how are they compared by their characteristics and their potential (e.g. architecture, programming models, NLP services that they integrate)
2. How can a chatbot framework be utilized to improve the interface of a user and a software application?
3. How practical and efficient is the integration of a digital assistant to a software application based on some of the available bot frameworks?

For the dissertation, a comparison between the frameworks was conducted mostly based on the characteristics they provided along with the user experience for the creator of such mentioned chatbot and the end user who will try out the service. A digital bet assistant was created in order to display the practical use of the natural language interface via the dialog system; its function is to help a customer to place a bet, learn about the various bet types and calculate the winnings of the resulted bets. As a mean of communication and interface the digital assistant was integrated to slack a chatbot channel.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω όλους όσους πίστεψαν παρά τις αντίξοές συνθήκες στην ολοκλήρωση του στόχου μου.

Περιεχόμενα

Κεφάλαιο 1
Εισαγωγή

1.1 Επισκόπηση

Η συνομιλία μεταξύ μηχανής και ανθρώπου αποτελεί ένα πεδίο το οποίο συνδέει διάφορες επιστήμες ώστε να επιτευχθεί η επικοινωνία μεταξύ των χρηστών και των υπολογιστών χρησιμοποιώντας φυσική γλώσσα. Ένας σχετικός όρος στην διεπαφή χρήστη-υπολογιστή είναι ο διαλογικός πράκτορας (chatbot), ενας διαλογικός ευφυής πράκτορας ο οποίος επικοινωνεί με τους χρήστες με σειρά χρησιμοποιώντας φυσική γλώσσα. Οι διαλογικοί πράκτορες χρησιμοποιούνται για πολλούς σκοπούς στη σημερινή εποχή, όμως όλα βασίζονται σε συγκεκριμένους αλγόριθμους μηχανικής μάθησης και στατιστικής επεξεργασίας φυσικής γλώσσας.

1.2 Διαλογικοί πράκτορες, ΤΝ και Επεξεργασία Φυσικής Γλώσσας

Οι Barr & Feigenbaum (1981) ορίζουν την τεχνητή νοημοσύνη (Artificial Intelligence, AI / ΤΝ), ως τον τομέα της επιστήμης των υπολογιστών που ασχολείται με την σχεδίαση ευφυών υπολογιστικών συστημάτων, δηλαδή συστημάτων που επιδεικνύουν χαρακτηριστικά που σχετίζονται με την νοημοσύνη στην ανθρώπινη συμπεριφορά [1]. Για να γίνει αυτό, οι υπολογιστές πρέπει να είναι ικανοί να αντιλαμβάνονται την ανθρώπινη διάλεκτο. Η σχεδίαση και η υλοποίηση υπολογιστικών συστημάτων, οι οποίοι επεξεργάζονται και δημιουργούν φυσικές γλώσσες είναι ενας κλάδος της επιστήμης της ΤΝ γνωστός ως *επεξεργασία φυσικής γλώσσας (Natual Language Processing, NLP)*. Οι Jurafsky & Martin [2] ορίζουν την NLP ως "*υπολογιστικές τεχνικές που επεξεργάζουν ανθρώπινη ομιλία και ανθρώπινη γλώσσα, ως γλώσσα*". Οι επόμενες ενότητες του κεφαλαίου θα παρουσίασουν την σχέση των διαλογικών πρακτόρων και άλλων τμημάτων της ΤΝ και της NLP [2].

1.3 Υπολογιστική γλωσσολογία και Σώματα

Για την ανάλυση δεδομένων στη φυσική γλώσσα χρησιμοποιούμε μεγάλα σύνολα πληροφοριών (datasets) τα οποία ονομάζονται σώματα (*πλήθ: corpora., εν: corpus).* Η επιστήμη που εστιάζεται πάνω στη μελέτη αυτών των δεδομένων με επεξεργασία φυσικής γλώσσας λέγεται γλωσσολογία σωμάτων (corpus linguistics). Οι Rayson et al [3] την οριοθετούν ως *μια μεθοδολογία που μπορεί να εφαρμοστεί σε όλες τις μελέτες της γλωσσολογίας*, ενώ οι Aijmer et al [4] την ορίζουν ως την *μελέτη της γλώσσας βασισμένη σε σώματα κειμένων*.

Μια σχετική επιστήμη που ασχολείται με την επεξεργασία φυσικής γλώσσας είναι η υπολογιστική γλωσσολογία (computational linguistics) η οποία εστιάζεται στην μελέτη των φυσικών γλωσσών όπως η παραδοσιακή γλωσσολογία όμως με την χρήση υπολογιστών ως εργαλείο μοντελοποίησης (και πολλές φορές για επαλήθευση ή και μη) τμημάτων της θεωρίας της γλωσσολογίας που έχουν ιδιαίτερο ενδιαφέρον [5]. Οι πρώτες εφαρμογές υπολογιστικής γλωσσολογίας ήταν τα συστήματα μηχανικής μετάφρασης (machine translation systems) τα οποία αποτελούν και από τους πρώιμους κλάδους της επεξεργασίας φυσικής γλώσσας και εμφανίστηκαν το 1950.

1.4 Αναζήτηση της πληροφορίας και απόκτηση γνώσης

Η αναζήτηση της πληροφορίας απαιτεί ανάκτηση πληροφοριών (data retrieval) γενικότερα ανάκτηση της πληροφορίας (information retrieval) και ειδικότερα εξαγωγή πληροφορίων (information extraction). Μια διαδικασία εξαγωγής πληροφοριών σκοπεύει στην ανάκτηση όλων των αντικειμένων από μια καλά ορισμένη δομή (για παράδειγμα από μια σχεσιακή βάση δεδομένων) η οποία ικανοποιεί πλήρως (η σε μεγάλο βαθμό) τις παραμέτρους που έχει θέσει ο χρήστης. Η ανάκτηση της πληροφορίας (IR) και η εξαγωγή της πληροφορίας (IE) σχετίζονται με την Επεξεργασία Φυσικής Γλώσσας, όπου η διαδικασία ανάκτησης κειμένων (document retrieval) μπορεί να είναι χρήσιμη η σχετική με την ερώτηση που θέτει ο χρήστης. [6].

Οι Jurafsky & Manning ορίζουν την εξαγωγή πληροφορίας ως *την παραγωγή δομημένης αναπαράστασης σχετικών πληροφοριών είτε για την δημιουργία συσχετίσεων ή για την δημιουργία γνωσιακής βάσης* (knowledge base). Απλοποιημένα, τα συστήματα εξαγωγής πληροφοριών πρέπει να εξάγουν πραγματικά περιστατικά ώστε να ικανοποιούν το εξής κριτήριο: *"ΠΟΙΟΣ ΕΚΑΝΕ ΤΙ ΣΕ ΠΟΙΟΝ ΠΟΤΕ;" (WHO DID WHAT TO WHOM WHEN?)* [7]. Ο Wilks [8] θεωρεί οτι *κάθε εφαρμογή της εξαγωγής πληροφορίας απαιτεί πριν να έχει εκτελεστεί μια φάση ανάκτησης πληροφορίας.* Με αυτό το σκεπτικό, πρώτα ένα σύστημα IR θα συλλέξει τις πληροφορίες που χρειάζονται ώστε έπειτα ενα σύστημα IE να την χρησιμοποιήσει για να εξάγει την επιθυμητή πληροφορία σε ενα σαφές πλαίσιο εφαρμογής (coherent framework).

Οι τεχνικές που χρησιμοποιούνται για την ανάκτηση και την εξαγωγή πληροφοριών έχουν εξελιχθεί ανάμεσα στις δεκαετίες, αλλά υπάρχουν τρείς αναγνωρισμένες προσεγγίσεις για τα προβλήματα αυτά: Η χρήση regular expressions / χειροκίνητων κανόνων, η χρήση ταξινομητών (classifiers) και η χρήση ακολουθιακών μοντέλων (sequence models). Οι προσεγγίσεις αυτές χρησιμοποιούνται ευρέως όχι μόνο στο συγκεκριμένο τομέα αλλά και σε μια ευρεία άλλων προβλημάτων ή παρεμφερών προβλημάτων. Στο κεφάλαιο της επεξεργασίας φυσικής γλώσσας θα επεκταθούμε περισσότερο.

1.5 Αξιολόγηση των Διαλογικών Πρακτόρων

Το τεστ που πρότεινε ο Alan Turing, γνωστό και ως τεστ Turing [9], έχει δημιουργηθεί ως μια *μέθοδο εκτίμησης του κατά πόσο μια μηχανή μπορεί να παρουσιάσει δείγματα ευφυΐας*. Σε ένα πληροφοριακό σύστημα, αυτό αποδεικνύεται με το *πόσο καλά μπορεί να εξομοιωθεί μια ανθρώπινη συνομιλία*. Έχουν προταθεί αρκετές άλλες μέθοδοι μετά το turing test για την εκτίμηση των συστημάτων, αλλά αυτό είναι εκτός των ορίων της μεταπτυχιακής διατριβής.

1.6 Σκοπός της μεταπτυχιακής διατριβής

Η αυξανόμενη αποτελεσματικότητα αλγορίθμων μηχανικής μάθησης στην ανάλυση και κατανόηση φυσικής γλώσσας προσφέρει δυνατότητες αξιοποίησής τους σε πολλές εφαρμογές. Δεδομένου ότι η παραμετροποίηση και εκπαίδευση αυτών των αλγορίθμων απαιτεί ιδιαίτερη τεχνογνωσία και έχει μεγάλες απαιτήσεις σε υπολογιστική ισχύ και όγκο δεδομένων εκπαίδευσης, διάφορες εταιρείες (π.χ. Microsoft, Facebook) προσφέρουν τις δυνατότητές τους με τη μορφή ηλεκτρονικών υπηρεσιών, π.χ. υπηρεσίες ανάλυσης φωνής, ανάλυσης συναισθημάτων σε γραπτό λόγο. Μια βασική χρήση των υπηρεσιών ανάλυσης φυσικής γλώσσας είναι η ενσωμάτωση σε εφαρμογές μιας διεπαφής χρήστη που έχει τη μορφή προσωπικού βοηθού (bot). Ο προσωπικός βοηθός αλληλοεπιδρά με το χρήστη μέσω ανταλλαγής προφορικών ή γραπτών μηνυμάτων σε φυσική γλώσσα.

Σκοπός της μεταπτυχιακής διατριβής είναι η διερεύνηση και αξιολόγηση των δυνατοτήτων που προσφέρονται από τα διαθέσιμα bot frameworks, για ανάπτυξη και ενσωμάτωση σε εφαρμογές διεπαφών χρήστη βασισμένων σε φυσική γλώσσα. Η μεταπτυχιακή διατριβή, αρχικά, θα μελετήσει και θα συγκρίνει τις υπηρεσίες επεξεργασίας φυσικής γλώσσας (Natural Language Processing Services - NLP Services), που προσφέρονται από διάφορους παρόδους, όσον αφορά τα χαρακτηριστικά και τις δυνατότητές τους (π.χ. γλώσσες που δέχονται ως είσοδο, δυνατότητα επεξεργασίας φωνής ή γραπτού λόγου, μοντέλο μηχανικής μάθησης που αξιοποιούν, τιμολογιακή πολιτική και πολιτική χρήσης).

Στη συνέχεια, θα αναλύσει και θα συγκρίνει βασικά bot frameworks τα οποία παρέχουν ένα ολοκληρωμένο πλαίσιο ενσωμάτωσης NLP υπηρεσιών σε εφαρμογές. Η σύγκριση θα αφορά, κυρίως, την αρχιτεκτονική, το προγραμματιστικό τους μοντέλο και τις υπηρεσίες κατανόησης φυσικής γλώσσας (Natural Language Processing Understanding - NLPU) που ενσωματώνουν. Βάσει της αξιολόγησης θα γίνει επιλογή ενός δημόσια διαθέσιμου bot framework για την υλοποίηση ενός πρωτοτύπου, με στόχο την διερεύνηση σε βάθος δυσκολιών και περιορισμών που υπεισέρχεται η χρήση και ενσωμάτωσή τους σε εφαρμογές.

Η υλοποίηση θα αφορά την ενσωμάτωση ενός προσωπικού βοηθού σε ένα κανάλι διαλογικών πρακτόρων (slack) το οποίο θα ασχολείται με την διεκπεραίωση ενός στοιχήματος μέσω διαλόγου.

1.7 Βασικά ερευνητικά ερωτήματα

Το βασικό ερώτημα της μελέτης αυτής είναι κατά πόσο ένα από τα τρέχοντα λογισμικά πακέτα διαλογικών πρακτόρων που χρησιμοποιούν αρχές επεξεργασίας φυσικής γλώσσας μπορούν να αξιοποιηθούν έτσι ώστε να βοηθήσουν την βελτίωση της διεπαφής χρήστη και υπολογιστή. Ειδικότερα:

1. Ποια είναι τα κυριότερα bot frameworks και πως συγκρίνονται βάσει των χαρακτηριστικών και δυνατοτήτων τους (π.χ. αρχιτεκτονικής, προγραμματιστικού μοντέλο, NLP υπηρεσιών που ενσωματώνουν);
2. Πως μπορεί να αξιοποιηθεί ένα bot framework για την βελτίωση της διεπαφής χρήστη μιας εφαρμογής;
3. Πόσο πρακτική και αποτελεσματική είναι η ενσωμάτωση σε μια εφαρμογή ενός προσωπικού βοηθού βασισμένου σε κάποιο από τα διαθέσιμα bot frameworks;

Η εκπόνηση της μεταπτυχιακής διατριβή περιλαμβάνει δυο φάσεις: (α) βιβλιογραφική αναζήτηση σε δημοσιευμένα άρθρα και online πηγές για την ανάλυση και συγκριτική αξιολόγηση bot frameworks και (β) σχεδιασμός και υλοποίηση ενός πρωτοτύπου, με πεδίο εφαρμογής την ενσωμάτωση ενός διαλογικού βοηθού σε ένα κανάλι διαλογικών πρακτόρων (slack, skype, κτλ), με στόχο την αξιολόγηση περιορισμών και δυσκολιών στην υιοθέτηση ενός bot framework. Η επιλογή του bot framework που θα αξιοποιηθεί

στην υλοποίηση του πρωτοτύπου συστήματος θα γίνει βάσει της συγκριτικής αξιολόγησης της 1ης φάσης βάσει διαφόρων παραγόντων όπως εύρος παρεχόμενών NLP υπηρεσιών, ευχρηστία, σταθερότητα API, τεκμηρίωση κτλ.

Κεφάλαιο 2
Επεξεργασία Φυσικής Γλώσσας

2.1 Εισαγωγή

Η επεξεργασία φυσικής γλώσσας (natural language processing / NLP) είναι ένας τομέας στα πεδία επιστημών της επιστήμης υπολογιστών, τεχνητής νοημοσύνης και υπολογιστικής γλωσσολογίας που ασχολείται με την διασύνδεση των υπολογιστών και των φυσικών ανθρωπίνων γλωσσών. Ασχολείται δηλαδή, με την κατανόηση της ανθρώπινης γλώσσας από τους υπολογιστές καθώς και την παραγωγή ανθρώπινης γλώσσας από αυτούς [2].

Σήμερα, υπάρχουν ενδιαφέρουσες εφαρμογές της επεξεργασίας φυσικής γλώσσας σε διάφορους τομείς. Με την άνθιση του παγκόσμιου ιστού έχουμε τις μηχανές αναζήτησης οι οποίες χρησιμοποιούν NLP για την αναζήτηση κειμένου μέσα σε ιστοσελίδες από μεγάλους εμπορικούς κολοσσούς όπως η Alphabet (Google), Microsoft (Bing), το Yahoo! και λιγότερο δημοφιλής όπως το Baidu και το Yandex. Σε πρόσφατο παρελθόν, το υπολογιστικό σύστημα της IBM, Watson [10], μπόρεσε να «νικήσει» ανθρώπους σε τηλεπαιχνίδι γνώσεων χρησιμοποιώντας ένα σύστημα ερωτοαπαντήσεων [11]. Σε κάθε

συσκευή apple υπάρχει το Siri, ενας ψηφιακός βοηθός που παρέχει πληροφορίες από το διαδίκτυο με μια μορφή ερωτοαπαντήσεων. Η μηχανική μετάφραση, μία από τις πρώτες NLP διαδικασίες που οριοθετήθηκαν, ζει μέσω του Google Translate και άλλων εφαρμογών για υπολογιστές και υπολογιστικές συσκευές.

Παρόλα αυτά, σύμφωνα με τους Manning et al [12], ο ερευνητής που θα ασχοληθεί με την επεξεργασία φυσικής γλώσσας πρέπει να γνωρίζει ότι:

- Η επεξεργασία φυσικής γλώσσας είναι δύσκολη
- Ποια είναι τα κυριότερα προβλήματα που θα συναντήσει
- Με ποιες μεθόδους αυτά τα προβλήματα μπορούν να αντιμετωπιστούν
- Ποια είναι τα όρια των μεθόδων αυτών.

2.2 Στάδια επεξεργασίας φυσικής γλώσσας

Από τον ορισμό της επεξεργασίας φυσικής γλώσσας, μπορούμε να αναπαραστήσουμε την διαδικασία της κατανόησης και της παραγωγής γλώσσας ως μια διασωλήνωση (pipeline) λογισμικού [13] [14]:

Εικόνα 2.1 Βασικά στάδια επεξεργασίας φυσικής γλώσσας

Το παραπάνω σχήμα είναι ο οδηγός για την επεξεργασία φυσικής γλώσσας σε υψηλό επίπεδο. Ανάλογα με πεδίο έρευνας (task) που θα επιλεχθεί για την παραγωγή γλώσσας η διασωλήνωση μπορεί να περιέχει ένα η περισσότερα κομμάτια επεξεργασίας από την «μηχανή».

2.3 Βασικά πεδία έρευνας της επεξεργασίας φυσικής γλώσσας

Η λίστα που ακολουθεί καταγράφει τα βασικά πεδία έρευνας που εμφανίζονται συχνά στην Επεξεργασία Φυσικής Γλώσσας. Πολλά από αυτά τα πεδία έρευνας έχουν πλέον

εμπορική αξία και οι εφαρμογές τους χρησιμοποιούνται στην καθημερινή ζωή ενώ μερικά χρησιμοποιούνται ως υπό-πεδία έρευνας για την σύνθεση πολύπλοκων ζητημάτων [15]:

- **Αυτόματη Περίληψη (Automatic Summarization)**
  Η παραγωγή μιας αναγνώσιμης περίληψης έχοντας ως είσοδο ένα τμήμα κειμένου.
- **Μηχανική Μετάφραση (Machine Translation)**
  Η αυτόματη κατανόηση και παραγωγή κειμένου από μία ανθρώπινη γλώσσα σε μια άλλη. Λόγω της πολυμορφίας της κάθε γλώσσας, το πεδίο έρευνας αυτό, ανήκει σε μία ομάδα προβλημάτων με την ονομασία «AI-complete».
- **Ανάκτηση πληροφοριών (Information Retrieval, IR)**
  Το πεδίο έρευνας που ασχολείται με την αποθήκευση, αναζήτηση και ανάκτηση πληροφοριών. Είναι ένα διαφορετικό πεδίο, συγγενές με τις βάσεις δεδομένων, αλλά η ανάκτηση πληροφοριών βασίζεται σε τεχνικές επεξεργασίας φυσικής γλώσσας.
- **Εξαγωγή πληροφορίας (Information Extraction)**
  Αφορά την εξαγωγή σημασιολογικής πληροφορίας (semantic information) από ένα κείμενο. Τμήματα αυτής της εξαγωγής είναι η Αναγνώριση Ονομαστικών Οντοτήτων η εξαγωγή σχέσεων κ.α.
- **Αναγνώριση ονομαστικών οντοτήτων (Named Entity Recognition, NER)**
  Ο καθορισμός μέσα από ένα κείμενο, αντικείμενα (οντότητες) τα οποία ανήκουν σε κάποια κατηγορία, π.χ. πρόσωπα, τοποθεσίες, οργανισμούς κ.α.,.
- **Εξαγωγή σχέσεων (Relation Extraction)**
  Η εξαγωγή σχέσεων αφορά την συσχέτιση περισσοτέρων από μία οντοτήτων σε μια πρόταση.
- **Παραγωγή φυσικής γλώσσας (Natural Language Generation)**
  Η μετατροπή πληροφοριών αποθηκευμένων σε υπολογιστικές βάσεις δεδομένων (databases) σε φυσική ανθρώπινη γλώσσα.
- **Κατανόηση φυσικής γλώσσας (Natural Language Understanding)**
  Η μετατροπή κομματιών κειμένου ανθρώπινης γλώσσας σε πιο διακριτές τυποποιημένες αναπαραστάσεις όπως δομές λογικής πρώτης τάξεως που είναι πιο κατανοητές από προγράμματα υπολογιστών προς επεξεργασία.

- **Διεπαφές χρήστη με φυσική γλώσσα (Natural Language UI)**
  Οι διεπαφές με φυσική γλώσσα αποτελούν διεπαφές μεταξύ υπολογιστή και ανθρώπου, όπου μέρη του λόγου, όπως ρήματα, φράσεις και προτάσεις αποτελούν εντολές σε λογισμικό. Τμήμα της μεταπτυχιακής διατριβής θα ασχοληθεί με αυτό το κομμάτι σε επόμενα κεφάλαια.
- **Οπτική αναγνώριση χαρακτήρων (Optical Character Recognition)**
  Η διαδικασία στην οποία εικόνες ή γλύφοι (glyphs) αναγνωρίζονται ως κείμενα.
- **Κατηγοριοποίηση μέρους του λόγου (Part-Of-Speech Tagging)**
  Στη γλωσσολογία σωμάτων (corpus linguistics) η κατηγοριοποίηση του λόγου, ή αλλιώς γραμματική κατηγοριοποίηση ή αποσαφήνιση λέξης-κατηγορίας, είναι η διαδικασία επισημείωσης μιας λέξης σε ένα σώμα με το αντίστοιχο μέρος του λόγου, βασισμένη στον ορισμό της και στη πλαισίωση της, δηλαδή στην σχέση της λέξης με συνοριακές και συγγενείς λέξεις σε μια φράση, πρόταση ή παράγραφο. Απλουστευμένα, είναι η διαδικασία στην οποία η λέξη κατηγοριοποιείται σε ρήμα, ουσιαστικό, επίθετο, άρθρο κτλ.
- **Συντακτική Ανάλυση (Syntactic Parsing)**
  Είναι η διαδικασία στην οποία αναλύεται ένα τμήμα σώματος ώστε να βρεθεί η δομική περιγραφή που ορίζεται από μια γραμματική, Χρησιμοποιείται ως πεδίο έρευνας ή ενδιάμεσο στάδιο για περαιτέρω επεξεργασία. [16]
- **Ερωτοαπαντήσεις (Question Answering)**
  Είναι ένα βασικό πεδίο έρευνας κατά το οποίο ένα πρόγραμμα υπολογιστή μπορεί να συγκροτήσει τις απαντήσεις που του θέτει ως ερωτήματα ένας χρήστης, κάνοντας αναζήτηση σε βάσεις δεδομένων και γνωσιακές βάσεις ή άλλες δομές πληροφοριών. Στη μεταπτυχιακή διατριβή, θα ασχοληθούμε περισσότερο με αυτό το πεδίο έρευνας σε επόμενο κεφάλαιο.
- **Εξαγωγή σχέσεων (Relationship Extraction)**
  Μια εξαγωγή σχέσεων είναι ένα υποπεδίο έρευνας στο οποίο απαιτείται η ανίχνευση και η κατηγοριοποίηση αναφερόμενων σημασιολογικών σχέσεων (semantic relationships) μέσα σε ένα σετ ευρημάτων προερχόμενο κυρίως από κείμενο ή xml αρχείων. Είναι πολύ συγγενές πεδίο με την εξαγωγή πληροφοριών αλλά δεν απαιτεί την αφαίρεση επαναλαμβανόμενων σχέσεων και γενικά αποτελεί την εξαγωγή πολλαπλών διαφορετικών σχέσεων.
- **Αναγνώριση Φωνής (Speech Recognition)**
  Η αναγνώριση φωνής είναι ένα πεδίο έρευνας της υπολογιστικής γλωσσολογίας,

στο οποίο αναπτύσσονται μεθοδολογίες και τεχνολογίες που ενεργοποιούν την αναγνώριση και την μετάφραση φωνητικής γλώσσας σε κείμενο κατανοητό από υπολογιστές. Ενσωματώνει γνώση από γλωσσολογία, επιστήμη υπολογιστών και ηλεκτρονική μηχανική.

- **Τμηματοποίηση Φωνής (Speech Segmentation)**
  Είναι το υποπεδίο έρευνας στο οποίο γίνεται αναγνώριση των ορίων μεταξύ λέξεων, συλλαβών ή φωνήμιων σε φωνητική φυσική γλώσσα. Κυρίως αφορά στην ικανότητα του ανθρώπου να ξεχωρίζει τα παραπάνω αλλά μεταφράζεται και στις τεχνητές διαδικασίες της επεξεργασίας φυσικής γλώσσας.
- **Τμηματοποίηση Λέξεων (Word Segmentation)**
  Η τμηματοποίηση λέξεων είναι ένα πρόβλημα κατά το οποίο διαχωρίζεται μια συμβολοσειρά γραπτής γλώσσας στις λέξεις που συνθέτει.
- **Αποσαφήνιση εννοιών λέξεων (Word-Sense Disambiguation)**
  Στην υπολογιστική γλωσσολογία, η αποσαφήνιση εννοιών λέξεων είναι ένα ανοικτό πρόβλημα της επεξεργασίας φυσικής γλώσσας και της οντολογίας. Αφορά την αναγνώριση όπου η σημασία χρησιμοποιείται σε μια πρόταση, όταν η λέξη έχει πολλαπλές σημασίες. Η επίλυση του προβλήματος αυτού βοηθά στην κατανόηση σχετικού με υπολογιστή γραπτών όπως ομιλίας (discourse) και βελτιώνει την σχετικότητα σε μηχανές αναζήτησης, επίλυση αναφορών, συνεχικότητας κτλ.
- **Επεξεργασία φωνής (Speech Processing)**
  Η επεξεργασία φωνής αφορά την μελέτη ηχητικών σημάτων ομιλίας και τις μεθόδους που επεξεργάζονται αυτά τα σήματα. Τμήματα της επεξεργασίας φωνής αφορούν την απόκτηση, μετατροπή, αποθήκευση, μεταφορά και έξοδο (ακοή ή μεταφορά σε άλλο σύστημα) των σημάτων ομιλίας.
- **Άλλα πεδία έρευνας, υποπεδίο και προβλήματα**
  Υπάρχουν αρκετά ακόμα προς έρευνα και καθώς οι ανάγκες του ανθρώπου αυξάνονται τόσο διευρύνονται. Ο πλήρης κατάλογος των προβλημάτων αυτών είναι εκτός των στόχων αυτής της μεταπτυχιακής διατριβή.

Το κριτήριο που διαχωρίζει τα πεδία αυτά ανάλογα με την εφαρμογή τους, είναι [17]:

- ο ορισμός ενός χώρου εργασιών καλά διατυπωμένου
- ένα καθιερωμένο μετρικό σύστημα για την αξιολόγηση του πεδίου

- τυποποιημένα σώματα κειμένου ως δεδομένα
- διαγωνισμούς για το πεδίο έρευνας

Στα επόμενα μέρη του κεφαλαίου αυτού, θα εξετάσουμε εκτενώς μερικά από τα πεδία έρευνας που αναφέρθηκαν παραπάνω, καθώς και βασικές τεχνικές επεξεργασίας φυσικής γλώσσας. Λόγω της φύσεως των πεδίων ερευνών, θα γίνει αναφορά σε ορισμένες έννοιες από το πεδίο της στατιστικής επεξεργασίας φυσικής γλώσσας.

2.3.1 Προβλήματα που συναντάμε κατά την επεξεργασία φυσικής γλώσσας

Οι Jurafsky & Manning [7], έχουν κατηγοριοποιήσει τα προβλήματα που προκύπτουν από τα πεδία έρευνας ως εύκολα, μεσαία και δύσκολα να ερευνηθούν, χωρίς αυτό να είναι τελικό, μιας που οι εξελίξεις στους κλάδους που αφορούν είναι ραγδαίες.

Ασάφεια

Ένα από τα μεγαλύτερα προβλήματα κατά την κατανόηση της φυσικής γλώσσας μεταξύ χειριστή και υπολογιστή είναι η ασάφεια. Σύμφωνα με τους Anjali et al [18] και πληροφορίες από τα wit.ai [19] η ασάφεια ορίζεται ως η ύπαρξη περισσότερων από μια έννοια για την ίδια την λέξη ή την πρόταση ώστε να προκύψει διαφορετική κατανόηση από αυτή που θα έπρεπε.

Η επεξεργασία φυσικής γλώσσας βασισμένη σε κείμενο, έχει αρκετά επίπεδα ανάλυσης πριν το τελικό αποτέλεσμα. Αναφορικά είναι:

- Λεκτική ανάλυση
- Συντακτική ανάλυση
- Σημασιολογική ανάλυση
- Λεκτική (discourse) ανάλυση
- Πραγματιστική (pragmatic) ανάλυση

Σε καθένα από αυτά τα επίπεδα μπορεί να συμβεί η ασάφεια και είναι μια πρόκληση για κάθε ερευνητή. Οι Anjali et Al [18], διαχωρίζουν την ασάφεια σε αυτά τα επίπεδα όπως παρακάτω:

- Η **λεκτική ασάφεια** αφορά την ασάφεια στη λέξη, π.χ. «*red tape*» μπορεί να εννοείται ότι αφορά την γραφειοκρατία ή την πραγματική κόκκινη ταινία.
- Η **συντακτική ασάφεια** αφορά την ασάφεια που εμφανίζεται μέσα σε μια πρόταση. Η πρόταση «***Buy books for children***» μπορεί να αφορά είτε την αγορά βιβλίων για παιδιά ή την αγορά βιβλίων για **τα** παιδιά.
- Η **σημασιολογική ασάφεια** αφορά την ασάφεια που εμφανίζεται όταν έχουν ξεκαθαριστεί οι λεκτικές και οι συντακτικές ασάφειες. Σημαίνει ότι η πρόταση μπορεί να διαβαστεί με δύο (η περισσότερες) ερμηνείες.

Ας υποθέσουμε ότι έχουμε την εξής φράση: ***«Teachers strikes idle kids»***
Η μία ερμηνεία της πρότασης είναι ότι οι απεργίες των καθηγητών αφήνουν «τα παιδιά χωρίς εργασίες». Η δεύτερη ερμηνεία της πρότασης είναι ότι «οι καθηγητές χτυπούν τα οκνηρά παιδιά». Λόγω της διαφοράς γλώσσας, το παράδειγμα είναι πιο εύκολο να κατανοηθεί από έναν άνθρωπο, όμως στην μηχανή θα υπάρξει πρόβλημα, αν π.χ. εξετάζουμε το συναίσθημα της πρότασης. Στην πρώτη πρόταση αφορά (αρνητικό) συναίσθημα στις απεργίες των καθηγητών ενώ στη δεύτερη αφορά (αρνητικό) συναίσθημα στο πρόσωπο των καθηγητών [7].

- Η **λεκτική ασάφεια** απαιτεί την ύπαρξη ενός κοινού κόσμου (shared world) ή κοινής γνώσης και η ερμηνεία οφείλεται μέσα στα συμφραζόμενα (context) της πρότασης. Π.χ. «Susan called Mary. She was sick». Το «She» αναφέρεται είτε στην Susan είτε στην Mary.
- Η **πραγματιστική,** τέλος, **ασάφεια** αφορά μια περίπτωση οπού τα συμφραζόμενα της πρότασης δίνει πολλαπλή ερμηνεία.

2.3.2 Λοιπά προβλήματα

Άλλα προβλήματα που μπορεί να συναντήσουμε κατά την επεξεργασία φυσικής γλώσσας είναι η έλλειψη τυποποιημένης γλώσσας, όπως γραφή όλου του κειμένου σε κεφαλαία, χρήση greeklish αντί Ελληνικής γλώσσας, χρήση ιδιωματισμών και άλλα. Επίσης μπορούν να υπάρξουν προβλήματα με τα σημεία στίξης. Πολλές φορές οι εκφωνήσεις σε διάφορα επαγγέλματα χρησιμοποιούν τελείες, όπως π.χ. «Δρ.» αντί για «δόκτορας», PhD. αντί για διδακτορικό κα. Εάν χρησιμοποιήσουμε μια συντακτική ανάλυση η οποία συλλέγει τις προτάσεις αναλόγως των σημείων στίξης τότε σε περιπτώσεις σαν τις προαναφερθείσες θα υπάρξει πρόβλημα αφού θα πρέπει να βρεθεί που τελειώνει η μία πρόταση και αρχίζει η άλλη. Οι νεολογισμοί, οι νέες λέξεις δηλαδή που δημιουργούνται από ανάγκη ή γρηγοράδα επίσης μπορούν να δημιουργήσουν πρόβλημα εάν δεν υπάρχει κάποιο λεξικό το οποίο να μπορεί να έχει αυτές τις λέξεις. Το ίδιο ισχύει για ιδιωματισμούς οι οποίοι μπορούν να εμφανιστούν υπό μορφή παροιμιών αλλάζοντας τα συμφραζόμενα του κειμένου.

2.4 Κατανόηση φυσικής γλώσσας

Η κατανόηση φυσικής γλώσσας είναι ένα υποπεδίο έρευνας της επεξεργασίας φυσικής γλώσσας που ανήκει στην τεχνητή νοημοσύνη και αφορά την μηχανική κατανόηση ανάγνωσης. Θεωρείται ως ένα δύσκολο πρόβλημα της τεχνητής νοημοσύνης [20]. Ο Winograd, περιγράφει την διαδικασία της κατανόησης της φυσικής γλώσσας σαν μια μετατροπή από μια σειρά ήχων ή κειμένου σε μια εσωτερική παράσταση «νοήματος» [21].

Η διαδικασία της αποδόμησης και της ανάγνωσης της εισόδου (π.χ. ενός κειμένου) είναι πιο πολύπλοκη από την αντίστροφη διαδικασία της συγκρότησης κειμένου, όπως θα δούμε παρακάτω στην δημιουργία φυσικής γλώσσας, επειδή υπάρχει το ενδεχόμενο των άγνωστων και απρόσμενων χαρακτηριστικών της εισόδου και υπάρχουν διάφοροι παράγοντες που προκαθορίζονται κατά την εξαγωγή της γλώσσας.

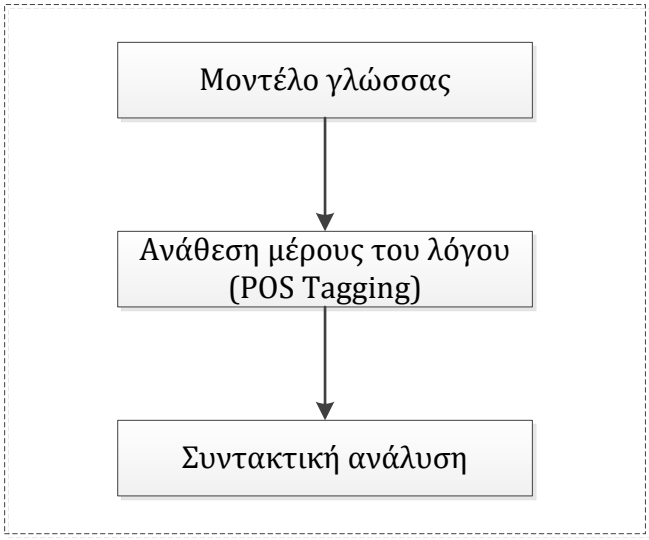
Υπάρχει μεγάλο ενδιαφέρον στην κατανόηση φυσικής γλώσσας, λόγω της σημασίας που έχει στην συγκομιδή ειδήσεων, κατηγοριοποίηση κειμένου, ενεργοποίηση φωνής και γενικώς στην ανάλυση μεγάλων όγκων κειμένου.

2.4.1 Συντακτικά δέντρα, στάδια επεξεργασίας NLP

Ο πρωταρχικός ρόλος κάθε εφαρμογής επεξεργασίας φυσικής γλώσσας είναι η δημιουργία ενός συντακτικού δέντρου (parse tree) με τιμές που ανήκουν σε μια συλλογή σε μια ανθρώπινη γλώσσα. Η συλλογή είναι η πρόταση ενώ οι τιμές αποτελούν τις λέξεις. Για την δημιουργία του δέντρου ανάλυσης όμως χρειάζεται η κατάταξη (classification) και η αναγνώριση (identification) των λέξεων της πρότασης. Αυτό μπορεί να επιτευχθεί με την γραμματική ανάλυση που χρησιμοποιείται στην ανθρώπινη γλώσσα, δηλαδή με την ανάθεση στις λέξεις, ρημάτων, ουσιαστικών, αντωνυμιών κτλ. Η γραμματική ανάλυση ορίζει ένα μοντέλο γλώσσας (language model) [22].

Χρησιμοποιώντας το μοντέλο της διασωλήνωσης, οι φάσεις στις οποίες γίνεται η αρχική κατανόηση των προτάσεων (ή των εγγράφων) αποτελούνται από:

1. Τη συγκρότηση του μοντέλου γλώσσας
2. Την ανάθεση μέρους του λόγου στις λέξεις των εγγράφων
3. Τη συντακτική ανάλυση



Εικόνα 2.2 Μοντέλο κατανόησης NLP

2.4.2 Μοντέλο γλώσσας

Το μοντέλο γλώσσας αποτελείται από ένα πιθανολογικό μοντέλο της γλώσσας που γίνεται η επεξεργασία και χρησιμοποιείται για τα επόμενα στάδια της NLP. Είναι ένα άκρως στατιστικό μοντέλο το οποίο αγνοεί την σημασιολογία των φράσεων του εγγράφου ενώ παράλληλα χρησιμοποιείται για την στατιστική κατανομή της γλώσσας.

Σύμφωνα με τον Jurafsky, ο στόχος του μοντέλου γλώσσας είναι ο υπολογισμός της πιθανότητας μιας σειράς λέξεων ή πρότασης σε ένα έγγραφο. Σε πολλούς τομείς της NLP χρειάζεται να υπάρχει η πιθανότητα σαν οδηγός για την διάκριση του αποτελέσματος μεταξύ «σωστού» και «λάθους». Στη στατιστική NLP, υποθέτοντας ότι υπάρχει μια σειρά με W_n λέξεις, τότε ο υπολογισμός P(W) δηλαδή η πιθανότητα εμφάνισης των λέξεων με αυτή τη σειρά υπολογίζεται ως:

$$P(W) = P(W_1, W_2, W_3, \dots, W_n)$$

Μια παρεμφερή διαδικασία, είναι ο υπολογισμός της πιθανότητας μιας επερχόμενης λέξης στην σειρά. Για τον υπολογισμό της πιθανότητας της W_5 λέξης, υπολογίζουμε ως:

$P(W_5|W_1, W_2, W_3, W_4)$ ή γενικότερα με $P(W_n|W_1, W_2, W_3, \dots, W_{n-1})$. Στατιστικά, αυτό καθορίζεται ως *μοντέλο γλώσσας* [23].

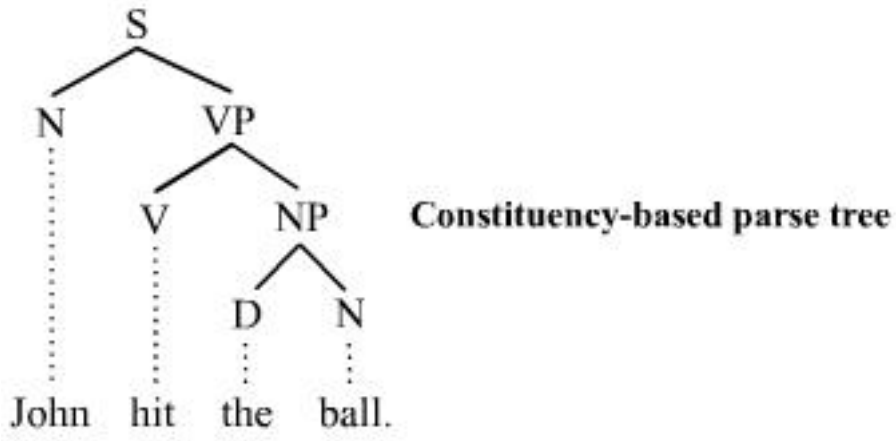
Υπάρχουν διάφοροι τρόποι μοντελοποίησης της γλώσσας. Οι πιο διαδεδομένοι είναι τα unigram, bigram και η γενική μορφή τους τα N-gram ενώ επίσης υπάρχουν μοντέλα βασισμένα σε δέντρα αποφάσεων, εκθετικά μοντέλα γλώσσας, μοντέλα συνεχούς διαστήματος και προσαρμοστικά μοντέλα μεταξύ άλλων. Ένα πεδίο της NLP που βασίζεται πολύ στα μοντέλα γλώσσας είναι η αναγνώριση φωνής. Αυτό οφείλεται λόγω στην μεγάλη σύγχυση των λέξεων που δημιουργείται κατά την δειγματοληψία και την ακρίβεια μετατροπής της φωνής σε μορφή αναγνώσιμη από μηχανή. Τα N-gram είναι ο πιο διαδεδομένος τρόπος μοντελοποίησης παρόμοιων προβλημάτων σε θέματα επεξεργασίας φυσικής γλώσσας [22].

2.4.3 Μέρη του λόγου

Πριν την συντακτική ανάλυση, οι λέξεις αναλύονται βάσει των μερών του λόγου. Δηλαδή γίνεται μια ανάθεση στις λέξεις με μια ετικέτα βασισμένη στα συμφραζόμενα της πρότασης που αναλύεται, σε ένα είδος ρήματος, ονόματος, αντωνυμίας, πρόθεσης και άλλων συντακτικών ετικετών. Η ανάθεση αυτή γίνεται βασισμένη είτε σε χειροποίητους κανόνες, οι οποίοι χρησιμοποιούν γλωσσικούς κανόνες ή κανόνες βασισμένους σε μηχανική μάθηση, όπως Κρυφά Μοντέλα Μάρκοβ (HMM), μέγιστης εντροπίας και υποθετικά τυχαία πεδία (conditional random fields). Η χρήση της μηχανικής μάθησης γίνεται για την εξάλειψη χειρωνακτικών κανόνων οι οποίοι είναι πολύπλοκοι και απαιτούν αρκετή επεξεργαστική ώρα.

2.4.4 Συντακτική ανάλυση

Το τελικό αποτέλεσμα είναι η δημιουργία ενός δέντρου ανάλυσης με ετικέτες τα μέρη του λόγου της πρότασης.



Εικόνα 2.3 Συντακτική ανάλυση (συντακτικό δέντρο) μιας πρότασης

2.5 Παραγωγή φυσικής γλώσσας

Η παραγωγή φυσικής γλώσσας είναι το πεδίο έρευνας της επεξεργασίας φυσικής γλώσσας που αναλαμβάνει την δημιουργία φυσικής γλώσσας από μια μηχανικά δομημένη αναπαράσταση όπως μια γνωσιακή βάση ή μια λογική φόρμα. Ο McDonald, καθορίζει την παραγωγή φυσικής γλώσσας ως μια διαδικασία στην οποία η σκέψη μεταφράζεται σε γραπτή γλώσσα [24].

Η παραγωγή φυσικής γλώσσας θεωρείται η αντίστροφη διαδικασία της κατανόησης φυσικής γλώσσας. Στην κατανόηση φυσικής γλώσσας, ο υπολογιστής αναλαμβάνει να ξεκαθαρίσει τις ασάφειες ώστε να κατηγοριοποιήσει τα δεδομένα σε μια μορφή αναγνώσιμη από το πρόγραμμα (ή μια μηχανή), ενώ στην παραγωγή φυσικής γλώσσας τα δεδομένα τα οποία μπορεί να είναι μερικές λέξεις μετατρέπονται σε γραπτές προτάσεις με μέρη του λόγου (ρήμα, ουσιαστικό, κτλ).

Το πιο χαρακτηριστικό παράδειγμα παραγωγής φυσικής γλώσσας είναι τα συστήματα παραγωγής περιεχομένου [25].

2.6 Ανάκτηση πληροφορίας

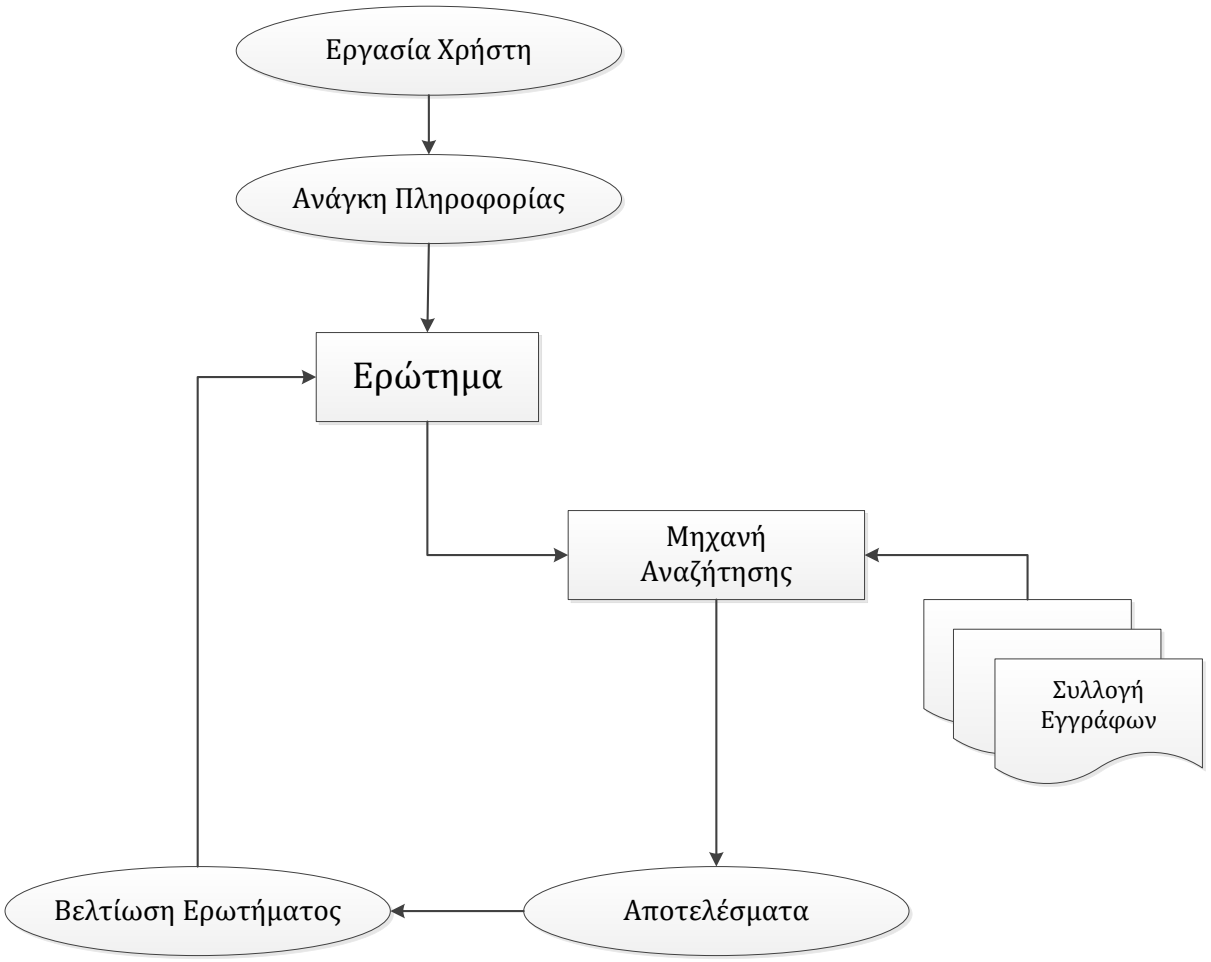
Ο Manning χαρακτηρίζει την ανάκτηση πληροφορίας (Information Retrieval) σαν την αναζήτηση υλικού (κυρίως έγγραφα) μιας μη δομημένης φύσης (κείμενο) που ικανοποιεί μια ανάγκη πληροφορίας μέσα από μεγάλες συλλογές (που βρίσκονται στον Η/Υ). [26]

Η ανάκτηση πληροφορίας στη σύγχρονη εποχή οδηγεί κυρίως στην αναζήτηση πληροφοριών μέσα από το διαδίκτυο, από μηχανές αναζήτησης όπως google, bing και yandex, αλλά οι τεχνικές και οι αλγόριθμοι που χρησιμοποιούνται είναι χρήσιμοι σε ένα μεγαλύτερο εύρος εφαρμογών όπως στην αναζήτηση e-mail, αναζήτηση σε συστήματα αρχείων, αναζήτηση σε γνωσιακές βάσεις και σε εξειδικευμένες βάσεις που χρησιμοποιούν ιατρικές και νομικές υπηρεσίες.

2.6.1 Βασικό μοντέλο αναζήτησης

Στην ανάκτηση πληροφορίας θεωρούμε ότι έχουμε μια συλλογή, ένα σετ εγγράφων είτε στατικό (μιας μορφής) ή δυναμικό διαφόρων μορφών. Ο στόχος είναι να γίνει η ανάκτηση των εγγράφων που περιέχουν την πληροφορία που είναι σχετική με την ανάγκη της πληροφορίας του χρήστη και βοηθά στην ολοκλήρωση μιας εργασίας του. Για τον σκοπό αυτό, χρησιμοποιούμε μοντέλα αναζήτησης.

Υπάρχουν διάφορα μοντέλα αναζήτησης, όμως το βασικότερο μοντέλο αναζήτησης που επικρατεί κυρίως στην αναζήτηση μέσω διαδικτύου και χρησιμοποιείται σε βασικές υπηρεσίες αρχιτεκτονικής διαδικτύου είναι η παρακάτω:



Εικόνα 2.4 Βασικό μοντέλο αναζήτησης

Ένα βασικό μοντέλο αναζήτησης κατά τον Manning, αρχίζει από την εργασία που θέλει να εκτελέσει ο χρήστης. Η εργασία αυτή μπορεί να είναι γνωστή ή να χρειάζεται επιπλέον πληροφορίες όποτε μεταβαίνει στο στάδιο της ανάγκης για αναζήτηση

πληροφοριών. Έχοντας μια βασική εικόνα στο μυαλό του η ανάγκη της πληροφορίας μετατρέπεται σε ένα ερώτημα όσο το δυνατόν πιο κοντινό σε κάποιο σύστημα ανάκτησης της πληροφορίας. Με την χρήση επεξεργασίας φυσικής γλώσσας που διαθέτουν σχεδόν όλες οι μηχανές αναζήτησης στο διαδίκτυο το ερώτημα μεταφράζεται σε μηχανική πληροφορία, η οποία εκτελεί την εργασία της ανάκτησης της πληροφορίας. Για να επιτύχει το στόχο αυτό πρέπει να αναζητήσει την εκάστοτε συλλογή εγγράφων (η οποία δεν περιορίζεται μόνο σε μορφή κειμένου, εξαρτάται από το είδος της αναζήτησης που θέλει να εκτελέσει ο χρήστης). Η μηχανή αναζήτησης θα επιστρέψει πίσω αποτελέσματα τα οποία μπορούν να εκτιμηθούν με μετρικά μεγέθη. Επίσης δίνεται η δυνατότητα να μπορέσει να βελτιώσει το αρχικό ερώτημα ώστε να φέρει πιο σχετικά με την ανάγκη της πληροφορίας κείμενα.

2.6.2 Αξιοποίηση των αποτελεσμάτων αναζήτησης

Για την εκτίμηση των αποτελεσμάτων αναζήτησης, χρησιμοποιούνται δύο μεγέθη της στατιστικής επεξεργασίας φυσικής γλώσσας, η ακρίβεια και η ανάκληση [27]:

- Η ακρίβεια (precision), δηλαδή πόσα από τα ανακτημένα έγγραφα είναι σχετικά με την ανάγκη της πληροφορίας του χρήστη. Εκφράζεται ως: $precision = \frac{|\{σχετικά\ έγγραφα\} \cap \{ανακτημένα\ έγγραφα\}|}{|\{ανακτημένα\ έγγραφα\}|}$.
- Η ανάκληση (recall), δηλαδή πόσα σχετικά έγγραφα από την συλλογή έχουν ανακληθεί. Εκφράζεται ως: $recall = \frac{|\{σχετικά\ έγγραφα\} \cap \{ανακτημένα\ έγγραφα\}|}{|\{ανακτημένα\ έγγραφα\}|}$.

2.7 Εξαγωγή πληροφορίας

Η εξαγωγή πληροφορίας έχει ως πεδίο έρευνας την αυτόματη εξαγωγή δομημένης πληροφορίας από μη δομημένα ή ήμιδομημένα έγγραφα τα οποία είναι μηχανικά αναγνώσιμα. Σύμφωνα με τους Cowie et al, η εξαγωγή πληροφορίας αποβλέπει στην διαδικασία μετασχηματισμού ενός πρωτεύοντος εγγράφου σε χρήσιμη πληροφορία υποπολλαπλάσια του αρχικού [28].

2.7.1 Εξαγωγή πληροφορίας και επεξεργασία φυσικής γλώσσας

Στην επεξεργασία φυσικής γλώσσας, η εξαγωγή πληροφορίας είναι ένα σημαντικό πεδίο έρευνας διότι:

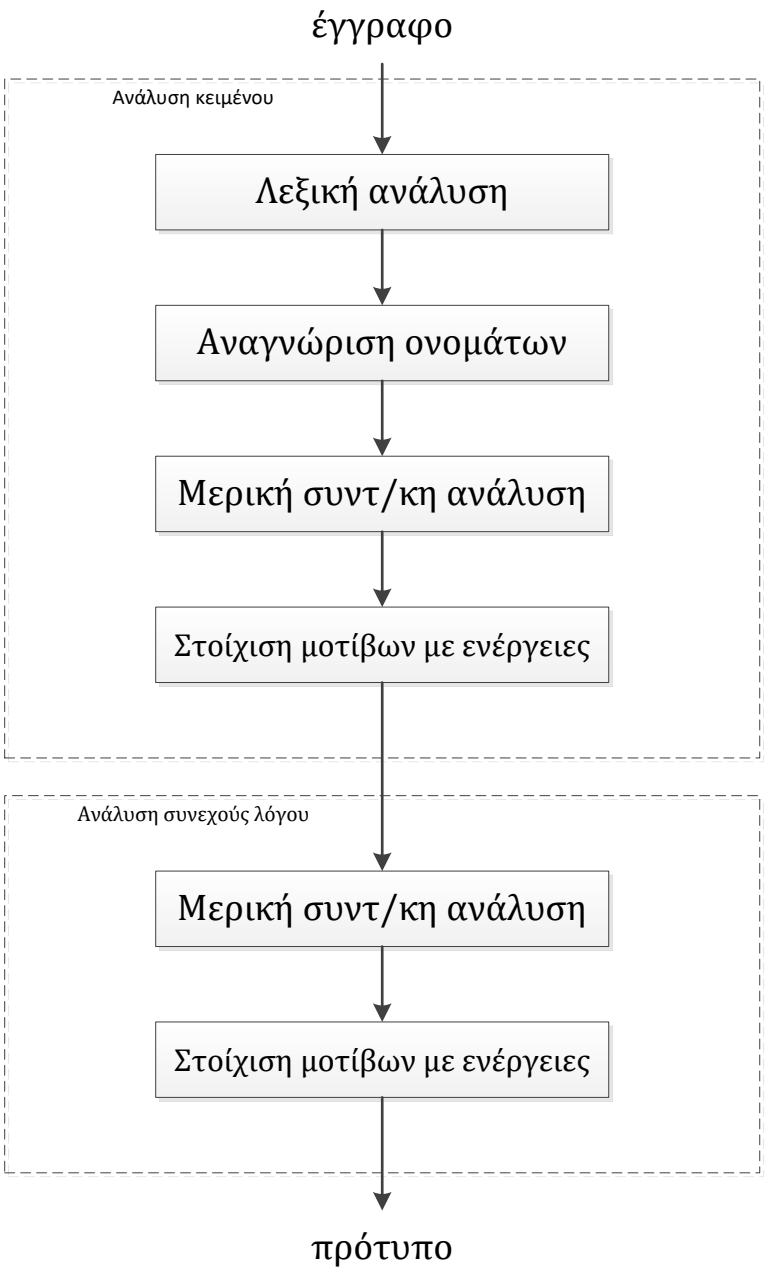
- Οι διαδικασίες εξαγωγής είναι καλά ορισμένες
- Η εξαγωγή πληροφορίας χρησιμοποιεί κείμενα από το πραγματικό κόσμο
- Η εξαγωγή πληροφορίας παρουσιάζει δύσκολα και ενδιαφέροντα προβλήματα επεξεργασίας φυσικής γλώσσας
- Η αξιολόγησή της εξαγωγής πληροφορίας είναι ένα συγκρινόμενο μέγεθος στο οποίο μπορεί να μετρηθεί και η ανθρώπινη απόδοση στην ίδια διαδικασία.

2.7.2 Στόχοι της εξαγωγής πληροφορίας

Σύμφωνα με τον Jurafsky, [29], οι στόχοι της εξαγωγής πληροφορίας είναι η οργάνωση της πληροφορίας με τέτοιο τρόπο ώστε να είναι χρήσιμη. Επίσης ο μετασχηματισμός της πληροφορίας σε μια σημαντικά ακριβή φόρμα ώστε να μπορεί να επεξεργαστεί περαιτέρω από αλγορίθμους υπολογιστών.

Για την παραπάνω επίτευξη, οι διαδικασίες που συνδέονται με την εξαγωγή πληροφορίας, αφορούν κυρίως διαδικασίες απλοποίησης κειμένου ώστε να παραχθεί μια δομημένη εικόνα της πληροφορίας που αποδίδεται σε ελεύθερο κείμενο. Συνήθως η αναπαράσταση αυτή γίνεται μέσω μιας μορφής πινάκων ή μιας βάσης δεδομένων. [30].

2.7.3 Βασική διαδικασία εξαγωγής πληροφορίας



Εικόνα 2.5 Βασική διαδικασία εξαγωγής πληροφορίας

Η βασική διαδικασία κατά τον Grishman είναι:

- Η αναζήτηση μοτίβων και δημιουργία δομών
  Για την αναζήτηση μοτίβων χρησιμοποιούνται ειδικές δομές αναζήτησης, οι *κανονικές εκφράσεις (regular expressions).* Χρησιμοποιώντας ειδικούς κανόνες

γίνεται το ταίριασμα λέξεων ως οντότητες. Σε αυτές τι ς οντότητες αποδίδεται μια ετικέτα (named entities).

- Η λεκτική ανάλυση του κειμένου
  Για την λεκτική ανάλυση του κειμένου, πρώτα γίνεται ο διαχωρισμός σε προτάσεις και έπειτα η κάθε πρόταση διαχωρίζεται σε λέξεις (tokenization). Κάθε λέξη ελέγχεται μέσω λεξικών για να καθοριστούν πιθανά μέρη του λόγου (part of speech) και χαρακτηριστικά (features).
- Η αναγνώριση ονομάτων
  Στην αναγνώριση ονομάτων, γίνεται η αναγνώριση τύπων κανονικών ονομάτων και άλλων δομών, όπως ημερομηνίες, αριθμητικά ποσά, συντομεύσεις κτλ. Η χρήση κανονικών εκφράσεων χρησιμοποιείται εδώ για την δημιουργία κατατάξεων και την αναγνώριση ψευδωνύμων, ακρωνύμιων της ίδιας λέξης.
- Η συντακτική δομή
  Στην υποδιαδικασία αυτή, ορίζονται μέρη του λόγου (part of speech) για κάθε λέξη της πρότασης. Έπειτα γίνεται αναγνώριση της ρίζας (root) της λέξης ανάλογα με το μέρος του λόγου και η εύρεση των καταλήξεων (stemming), κυρίως στα ρήματα για την εύρεση του χρόνου.
- Η στοίχιση μοτίβων με ενέργειες
  Έχοντας αναγνωρίσει τις λέξεις, γίνεται η εξαγωγή των ενεργειών (ή γεγονότων) τα οποία είναι σχετικά με το κείμενο που ελέγχεται. Αυτές οι ενέργειες συνδέουν οντότητες μεταξύ τους.
- Η ανάλυση συναναφοράς
  Η ανάλυση συναναφοράς, κυρίως γίνεται για την εξάλειψη αναφορών μέσα σε αναφορές. Εφόσον γίνει η ανάλυση του κειμένου, γίνεται η απόδοση αντωνυμιών που χρησιμοποιούνται αναφορικά μέσα στο κείμενο, για την εύρεση ονομάτων ή ενεργειών.
- Η εξαγωγή συμπεράσματος και συγχώνευση ενεργειών
  Οι ενέργειες μπορούν να επεκταθούν πέρα της μιας πρότασης ανάλογα με τον τρόπο έκφρασης ή γραφής στο κείμενο. Σε περίπτωση που γίνει ανίχνευση παρόμοιων ενεργειών χρειάζονται ειδικοί κανόνες για την αντιμετώπιση τέτοιων προβλημάτων. Κυρίως, το πρόβλημα εντοπίζεται, στην χρονική σημασία των λέξεων που μπορεί να υπάρξουν στο κείμενο, ώστε να υπάρξει συνοχή στις ενέργειες (πρώτα εκτελέστηκε μια ενέργεια/γεγονός Α μετά ένα γεγονός Β ενώ διαρκεί ένα γεγονός Γ, π.χ.)

2.8 Αναγνώριση Ονομαστικών Οντοτήτων

Ένα από τα υποπεδία της εξαγωγής πληροφορίας είναι η αναγνώριση ονομαστικών οντοτήτων (Named Entity Recognition, NER). Το πρόβλημα που καλύπτει η αναγνώριση ονομαστικών οντοτήτων αφορά την *αναζήτηση* και την *ταξινόμηση* ονομάτων σε ένα κείμενο.

2.8.1 Ονομαστικές Οντότητες, Ονόματα

Στην εξαγωγή πληροφοριών, μια ονομαστική οντότητα (named entity) είναι ένα αντικείμενο πραγματικού κόσμου, όπως μια τοποθεσία, ένα προϊόν ή ένα πρόσωπο κ.α., το οποίο αντιπροσωπεύεται από ένα όνομα. Συγκρίνοντας το με τις αρχές του αντικειμενοστραφούς προγραμματισμού, μια οντότητα είναι μια κλάση (class) και ένα όνομα είναι το στιγμιότυπο, το αντικείμενο της κλάσης. Για παράδειγμα, η **Αθήνα** είναι ένα στιγμιότυπο μιας **πόλης.** Ο Saul Kripke, ορίζει αυτή την ονομασία της οντότητας ως ένα αναγκαίο προσδιοριστικό περιορισμό, ώστε η διαδικασία επεξεργασίας να επικεντρωθεί σε συγκεκριμένο τμήμα. [31]

2.8.2 Αναγνώριση Ονομαστικών Οντοτήτων

Έστω ότι έχουμε την εξής πρόταση: *«A representative from WWF claimed that Bai Yun will be transferred to San Diego Zoo»*. Η παραπάνω πρόταση περιέχει τοποθεσίες, οργανισμούς και ονόματα. Το «WWF» αντιστοιχεί στον **οργανισμό**, «Bai Yun» είναι το **όνομα** και «San Diego Zoo» είναι η **τοποθεσία**.

Μια τυπική διαδικασία ενός συστήματος αναγνώρισης ονομαστικών οντοτήτων προτείνεται από τους Ratinov και Roth [32]. Τα στάδια που πρέπει να ακολουθηθούν είναι:

1. Μετατροπή σε κομμάτια (chunking) και αντιπροσώπευση κειμένου
2. Εφαρμογή αλγορίθμων εξαγωγής συμπεράσματος και επίλυσης ασάφειας
3. Μοντελοποίηση των μη-τοπικών εξαρτήσεων
4. Υλοποίηση εξωτερικών γνωσιακών πόρων και χαρτογραφήσεων

Η εξαγωγή συμπεράσματος χρησιμοποιείται για την απόφαση εάν ένα κομμάτι κειμένου είναι όντως μια οντότητα ή πιο ορθά, για να προσδιορίσει την ταξινόμηση αυτής της οντότητας. Ειδικά σε λέξεις οι οποίες μπορούν να παρουσίασουν ασάφεια όπως ονομασίες πόλεων, ή λέξεις οι οποίες έχουν ειδική σημασία αλλά και γενική, π.χ. Galaxy, για να δηλώσει το γαλαξία αλλά και ένα ραδιοφωνικό σταθμό. Η ανάλυση που απαιτείται, υλοποιείται κυρίως με μοτίβα ή με μηχανική μάθηση.

Οι μη τοπικές εξαρτήσεις αφορούν την ικανότητα του να μπορεί το λογισμικό που θα χρησιμοποιηθεί να αναγνωρίσει και να ταξινομήσει ίδιες οντότητες με διαφορετική γραφή. Π.χ. TOkyo ή tokyo ή Tokyo ανήκουν σε μια τοποθεσία. Άλλες αναγνωρίσεις αφορούν συντομογραφίες, π.χ. Dr. είναι συνώνυμο/συντόμευση του Doctor, άρα και πρέπει να οριστούν στην ίδια οντότητα (πρόσωπο ή ιδιότητα), η εκφωνήσεις, «Ο Πρόεδρος της Χώρας» με κάποιο όνομα «Πρόεδρος Τάδε».

Για την αναγνώριση ονομάτων, χρειάζεται η ύπαρξη λεξικών, τα οποία πρέπει να ανανεώνονται συνεχώς. Αυτό αυξάνει τις πιθανότητες να αναγνωριστούν ονόματα, οργανισμοί και τοποθεσίες οι οποίες δεν αναγνωρίστηκαν σωστά στην αρχή. Σε πολλές εργασίες της φυσικής γλώσσας η χρήση λεξικών και χαρτογραφήσεων είναι αναγκαία όποτε αυτή η διαδικασία, συνήθως είναι απαραίτητο βήμα [33].

Κεφάλαιο 3
Διαλογικοί πράκτορες

3.1 Εισαγωγή

Ένας διαλογικός πράκτορας (chatbot) είναι ένα πρόγραμμα υπολογιστή [34] που διεξάγει μια συνομιλία με την χρήση κειμενικών ή ακουστικών μεθόδων. Τα προγράμματα αυτά σχεδιάζονται έτσι ώστε να πλησιάζουν όσο πιο κοντά γίνεται την ανθρώπινη συμπεριφορά, προσομοιώνοντας ένα συνομιλητή. Οι διαλογικοί πράκτορες συνήθως χρησιμοποιούνται σε διαλογικά συστήματα (dialog systems) πρακτικά για διάφορους λόγους, όπως αυτοματοποιημένη υποστήριξη πελατών ή εξαγωγή πληροφοριών. Κυρίως οι διαλογικοί πράκτορες χρησιμοποιούν την επεξεργασία φυσικής γλώσσας για να αντλήσουν πολύπλοκες πληροφορίες, αλλά βασικά χρησιμοποιούν λέξεις κλειδιά κατά την είσοδο των πληροφοριών από το χρήστη και «απαντούν» αναλόγως. Υπάρχουν δύο ειδών διαλογικοί πράκτορες, αυτοί που χρησιμοποιούν ένα ορισμένο σετ κανόνων για την κατανόηση των πληροφοριών που εισάγει ο χρήστης και την παραγωγή αντίστοιχων απαντήσεων και αυτά που χρησιμοποιούν ένα ευρύτερο φάσμα τεχνητής νοημοσύνης ώστε να μπορούν να «μάθουν» από τις απαντήσεις τους και τις ερωτήσεις που κάνει ο χρήστης και να βελτιώνουν την γνωσιακή τους βάση, άρα και το ίδιο τους το πρόγραμμα.

3.2 Κατηγορίες και βασική αρχιτεκτονική Διαλογικών Πρακτόρων

Γενικώς, οι διαλογικοί πράκτορες συγκαταλέγονται σε δύο βασικές ομάδες ανάλογα με την χρήση τους:

- Ψυχαγωγικοί διαλογικοί πράκτορες
- Διαλογικοί πράκτορες για εμπορικούς σκοπούς

Ιστορικά, ο πρώτος διαλογικός πράκτορας το οποίο χρησιμοποιήθηκε για ψυχαγωγία ήταν το ELIZA [35], από το 1966 που είχε σχέση με ψυχοθεραπεία. Έκτοτε οι δημιουργοί των διαλογικών πρακτόρων προσπαθούν να δημιουργήσουν ένα πράκτορα ο οποίος θα προσπαθήσει να περάσει ως άνθρωπος, χρησιμοποιώντας το turing test ή άλλους διαγωνισμούς όπως το διαγωνισμό Loebner. Πρόσφατα η Microsoft δημιούργησε το Xiaoce και το Tay δύο διαλογικούς πράκτορες με παρόμοια συμπεριφορά. Οι απαντήσεις που πρέπει να παρέχει ένας διαλογικός πράκτορας στο χρήστη πρέπει να είναι ικανές ώστε να μπορεί ο χρήστης να συνεχίσει την συνομιλία [36]. Ωστόσο δεν είναι απαραίτητο στο διαλογικό πράκτορα να μπορεί να θυμάται την συνομιλία η τις λεπτομέρειες της. Τα ψυχαγωγικά bots έχουν ως σκοπό και ως μέτρο εκτίμησης την προσομοίωση τους με τον άνθρωπο, ενώ υπάρχουν και άλλα ποσοτικά τεστ εκτίμησης ανάλογα με τη διάρκεια της συνομιλίας. Εάν η διάρκεια της συνομιλίας είναι πολύ μικρή τότε ο διαλογικός πράκτορας δεν είναι αρκετά ψυχαγωγικός.

Οι εμπορικοί διαλογικοί πράκτορες συνήθως είναι γραμμικοί και βασισμένοι σε κάποια ανταλλαγή πληροφοριών για συγκεκριμένο σκοπό. Ένας διαλογικός πράκτορας ταξιδιωτικού οδηγού, έχει ως στόχο το κλείσιμο μιας σουίτας ξενοδοχείου ή την αγορά από το χρήστη ενός πακέτου διακοπών μέσα από απλές διαλογικές εκφράσεις. Ο ψηφιακός βοηθός της google αναλαμβάνει να παρέχει έτοιμες πληροφορίες στο χρήστη όταν τις ζητήσει. Ένας διαλογικός πράκτορας διαχείρισης βλαβών αναλαμβάνει να προσδιορίσει την βλάβη από μια γνωσιακή βάση και τελικώς να προσφέρει μια λύση είτε στο πρόβλημα είτε βρίσκοντας ένα τεχνικό που θα μπορέσει να εξετάσει περαιτέρω το πρόβλημα. Η εκτίμηση των εμπορικών διαλογικών πρακτόρων αφορά την έκβαση της χρήσης του ίδιου του προγράμματος. «Έγινε η κράτηση στο ξενοδοχείο;», «βρέθηκε η πληροφορία που ρώτησε ο χρήστης;», «επιλύθηκε το πρόβλημα;». Όλες οι απαντήσεις

αφορούν την ποιότητα του διαλογικού πράκτορα καθώς και την απόδοση του και συγκαταλέγονται στα σημεία σύγκρισης ενός διαλογικού πράκτορα.

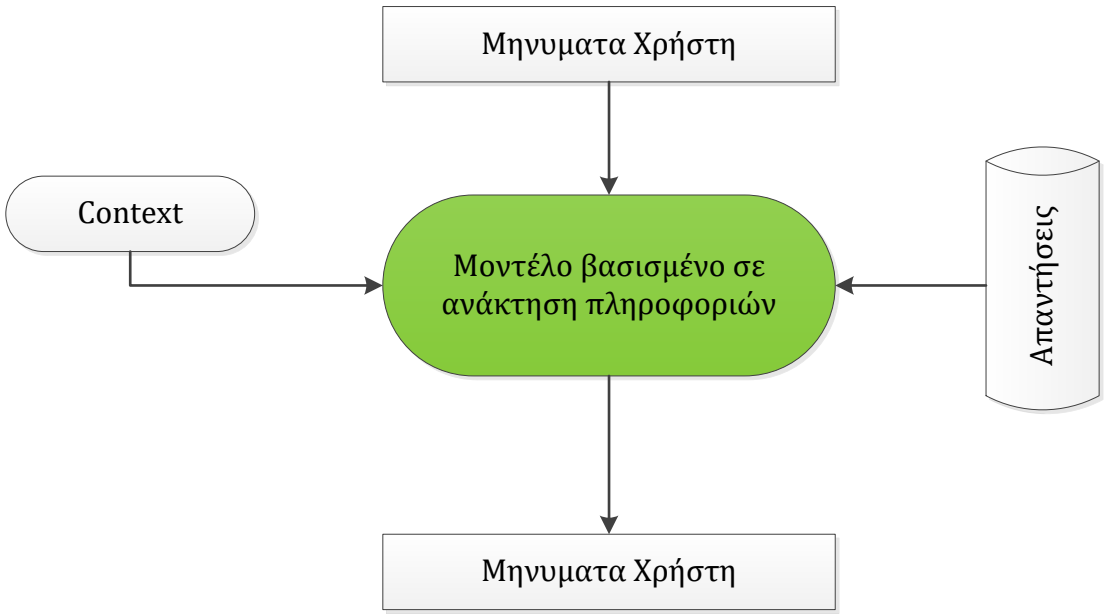
3.2.1 Μοντέλα παραγωγής απαντήσεων

Ο Denny Britz [37], συγκαταλέγει τα μοντέλα των διαλογικών πρακτόρων ως:

- Μοντέλα βασισμένα στην ανάκτηση πληροφοριών
- Παραγωγικά μοντέλα (Generative models)

Μοντέλα βασισμένα στην ανάκτηση πληροφοριών

Τα μοντέλα βασισμένα στην ανάκτηση πληροφοριών (retrieval-based models) είναι εύκολα στην υλοποίηση τους. Παρέχουν μια γραμμική μορφή σεναρίου (βλ. παρακάτω), οπότε και είναι πιο εύκολα στην πρόβλεψη των αποτελεσμάτων. Λόγω της πρακτικότητας των μοντέλων αυτών, υπάρχουν ήδη αρκετά διαθέσιμα API και αλγόριθμοι που τα υποστηρίζουν.



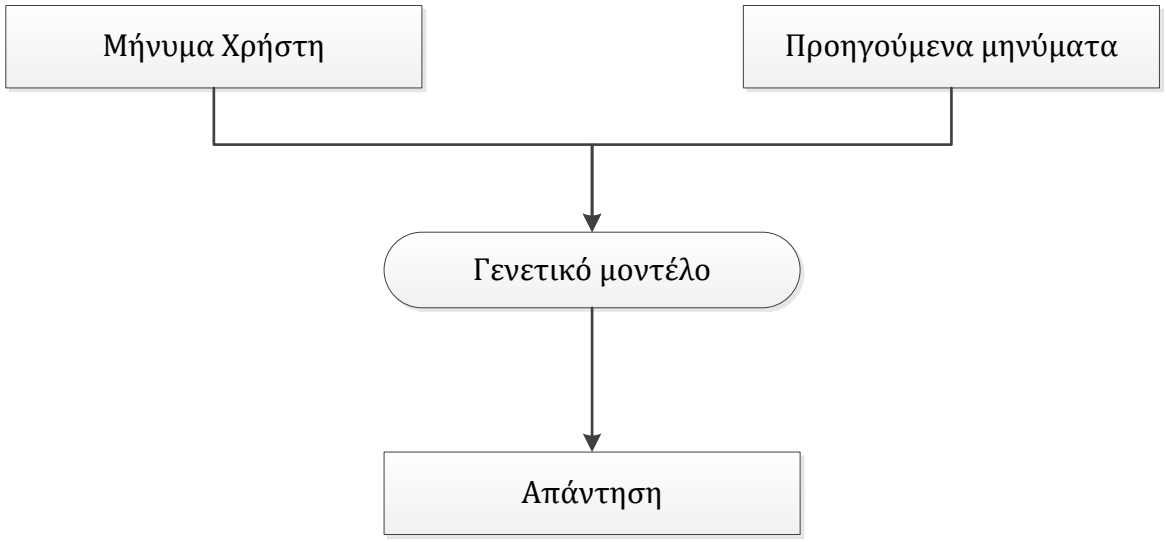
Εικόνα 3.1 Μοντέλο βασισμένο στην ανάκτηση πληροφοριών

Τέτοιου είδους μοντέλα απαιτούν την ύπαρξη κάποιου ευρετικού αλγόριθμου (heuristic algorithm). Μπορούν να χρησιμοποιηθούν αλγόριθμοι μέσω αναπαραστάσεων if-else

λογικής έως ταξινομητές μηχανικής μάθησης. Ο πιο απλός τρόπος είναι η χρήση κανόνων με μοτίβα (pattern-based rules). Όταν ο διαλογικός πράκτορας λάβει μια ερώτηση από το χρήστη (μήνυμα), αναζητεί όλα τα μοτίβα που αντιστοιχεί σε αυτό το μήνυμα. Εάν το μήνυμα βρεθεί, τότε συνεχίζει την διαδικασία ειδάλλως εμφανίζει ένα αντίστοιχο μήνυμα λάθους και προχωρά σε μια διαδικασία επανάληψης, ανάλογα με το προγραμματισμό του.

Παραγωγικά μοντέλα (Generative)

Τα παραγωγικά μοντέλα αποτελούν μια νέα προσέγγιση. Δεν βασίζονται σε έτοιμες απαντήσεις ή κανόνες με μοτίβα, χρησιμοποιούν κυρίως κανόνες από την μηχανική μετάφραση, αλλά αντί να μετατρέπουν μια γλώσσα σε μια άλλη, μετασχηματίζουν την είσοδο (το μήνυμα του χρήστη) σε μια αποδεκτή έξοδο (την απάντηση από το διαλογικό πράκτορα).



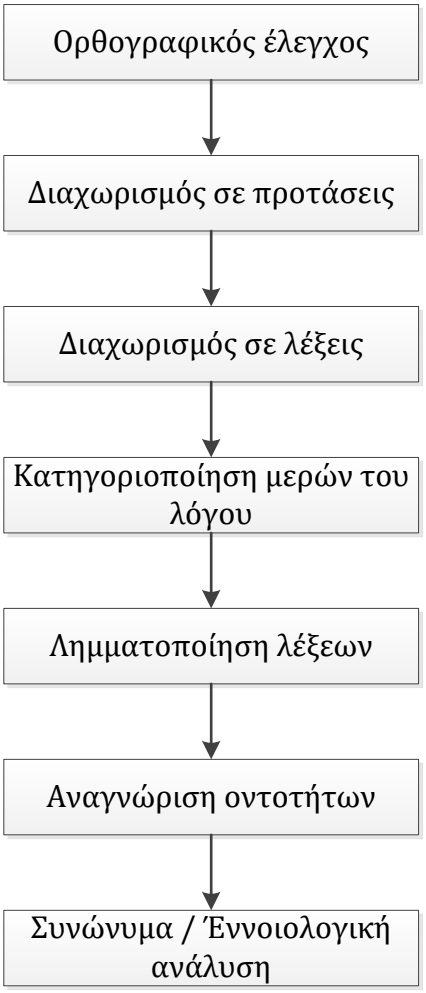
Εικόνα 3.2 Παραγωγικό μοντέλο

Τα παραγωγικά μοντέλα, λόγω της εξελιγμένης δυνατότητας να εξετάζουν τις οντότητες και το context των μηνυμάτων, μπορούν να φερθούν πιο έξυπνα, όμως είναι και πιο δύσκολη η εκμάθηση τους από το πρόγραμμα. Είναι πιο εύκολα στην αναγνώριση ειδικών περιπτώσεων (edge cases) σε σχέση με το μοντέλο ανάκτησης πληροφοριών, το οποίο είναι σχεδιασμένο κυρίως στο να μην έχει γραμματικά λάθη. Τα

γενετικά μοντέλα χρησιμοποιούν τη βαθιά γνώση (deep learning) και αποτελούν την βάση για ένα ευρύτερο πεδίο ερευνών.

3.2.2 Διασωλήνωση διαλογικών πρακτόρων

Οι διαλογικοί πράκτορες σαν προγράμματα, χρησιμοποιούν το μοντέλο διασωλήνωσης της επεξεργασίας φυσικής γλώσσας. Λόγω της διασωλήνωσης αρκετά τμήματα μπορούν να χρησιμοποιηθούν είτε τοπικά στον υπολογιστή (με την χρήση βιβλιοθηκών / αντικειμένων) ή μέσω υπηρεσιών διαδικτύου [38]. Ο όγκος πληροφοριών που συνήθως απαιτείται από πιο ανεπτυγμένους διαλογικούς πράκτορες τόσο για την εκμάθηση όσο για την εξαγωγή πληροφορίων όταν ζητηθεί από κάποιο χρήστη, επέτρεψε την ανάπτυξη τους σαν εφαρμογές με αρχιτεκτονική υπηρεσιών διαδικτύου. Έτσι τμήματα που έχουν σχέση με την επεξεργασία φυσικής γλώσσας ή ολοκληρωμένες εφαρμογές διαλογικών πρακτόρων χρησιμοποιούν εξυπηρετητές διαδικτύου καθώς και κανάλια διοχέτευσης διαλογικών πρακτόρων όπως το Facebook, το slack, το skype και αλλά για την παρουσία τους στο διαδίκτυο. Με τον ίδιο τρόπο, οι λειτουργίες του διαλογικού πράκτορα μπορούν να κάνουν κλήσεις απομακρυσμένων υπηρεσιών για να ψάξουν πληροφορίες σε γνωσιακές βάσεις ή βάσεις δεδομένων ή άλλες διεπαφές (APIs) που παρέχουν εμπορικές εταιρείες και πανεπιστήμια για την πρόσβαση στις πληροφορίες τους [39].



Εικόνα 3.3 Προτεινόμενα βήματα διασωλήνωσης NLP για διαλογικούς πράκτορες

Επειδή οι λειτουργίες των διαλογικών πρακτόρων είναι σχετικά ίδιες ανεξάρτητα της υλοποίησης που παρέχει η κάθε εταιρεία η ο κάθε ακαδημαϊκός πάροχος, έχει αρχίσει μια τυποποίηση των χαρακτηριστικών ώστε ο προγραμματιστής να μπορεί να έχει ένα σημείο αναφοράς μεταξύ των διαφόρων NLUs. Αυτά τα χαρακτηριστικά έχουν ως πηγή πεδία και υποπεδία της επεξεργασίας φυσικής γλώσσας, ομαδοποιημένα και απλοποιημένα, εφόσον παρέχονται ως έτοιμα εργαλεία/υπηρεσίες προς τον προγραμματιστή. Παράλληλα παρέχονται διεπαφές με επεξήγηση για τα επόμενα βήματα, ώστε η διαδικασία της εκμάθησης εκφράσεων να είναι σχετικά εύκολη και προσιτή χωρίς μεγάλη σημασία στο κώδικα παρά στην ίδια την διαδικασία.

3.3 Πλατφόρμες ανάπτυξης διαλογικών πρακτόρων

Οι διαλογικοί πράκτορες αποτελούν πλέον ένα αναπτυσσόμενο οικονομικό παράγοντα στην παγκόσμια αγορά. Έτσι εταιρείες πληροφορικής, όπως η Microsoft, η Google (Alphabet), το Facebook, η Amazon και άλλες αυτόνομες start ups προσπαθούν τόσο να βοηθήσουν στην ανάπτυξη και βελτίωση του λογισμικού, όσο και την εμπορική τους εκμετάλλευση.

Για αυτό το σκοπό, η πλειοψηφία των μεγάλων εταιρειών διαθέτει τις βιβλιοθήκες και το interface για την δημιουργία διαλογικών πρακτόρων στο διαδίκτυο, υπό την μορφή υπηρεσιών web, για την μηχανή φυσικής επεξεργασίας γλώσσας (NLUE), καθώς και γρήγορες γεννήτριες δημιουργίας διαλογικών πρακτόρων χρησιμοποιώντας web interfaces. Ο συνδυασμός των υπηρεσιών web μαζί με την συγκρότηση υπηρεσιών υπολογιστικού νέφους οδήγησε σε νέους εμπορικούς όρους που σχετίζονται με τους διαλογικούς πράκτορες όπως AI-as-a-service (orbit, amazon), cognitive web services (Microsoft) και conversation-as-a-service (Microsoft).

3.3.1 Ορολογίες διαλογικών πρακτόρων

Τα βασικά χαρακτηριστικά των διαλογικών πρακτόρων που συναντιούνται σε όλες τις υπηρεσίες είναι:

Οντότητες (entities)

Οι οντότητες αντιπροσωπεύουν έννοιες (ή ορισμούς) και χρησιμοποιούνται για την εξαγωγή παραμέτρων μέσα από ένα κείμενο φυσικής γλώσσας. Κατά την «εκπαίδευση» του διαλογικού πράκτορα, ίδιες παράμετροι που δίνουν ίδιο αποτέλεσμα ομαδοποιούνται. Για παράδειγμα, η οντότητα «θερμοκρασία» σε προτάσεις μπορεί να έχει ως εξής

- Σήμερα είχε ζέστη
- Ο καιρός κρύωσε
- Βάλε τον φούρνο στους 250 βαθμούς.

Οι παραπάνω εκφράσεις έχουν σχέση με την θερμοκρασία και αντίστοιχα οι παράμετροι που θα ανιχνευτούν θα είναι «ζέστη», «κρύο», «250 βαθμοί» αλλά όλοι θα χαρτογραφηθούν στην οντότητα «θερμοκρασία».

Context

Ένα από τα χαρακτηριστικά των σύγχρονων διαλογικών πρακτόρων είναι να μπορούν να διακρίνουν το context (συμφραζόμενα) των εκφράσεων του χρήστη. Χρησιμοποιείται για να ξεπεράσει την ασάφεια που πολλές φορές εμφανίζεται ανάλογα με τις λέξεις που χρησιμοποιούνται στην έκφραση. Αυτές οι ασάφειες μπορούν να αφορούν τοπογραφικούς ιδιωματισμούς ή προτιμήσεις που μπορεί να έχει ο χρήστης.

Για παράδειγμα, εάν ενας χρήστης ψάχνει σε μια βιβλιοθήκη τραγουδιών και του αρέσει ένα είδος μουσικής, χρησιμοποιώντας ένα διαλογικό σύστημα μπορεί να τονίσει ότι του αρέσει αυτό το είδος μουσικής και θα ήθελε να ακούσει περισσότερα παρόμοια τραγούδια σαν αυτό. Η μηχανή κατανόησης φυσικής γλώσσας θα προσπαθήσει να αναζητήσει ότι πιο κοντινότερο σε αυτό το είδος μουσικής υπάρχει.

Σκοπός χρήστη (intents)

Ένας σκοπός χρήστη αντιπροσωπεύει ένα μήνυμα που χαρτογράφει τι θέλει ένας χρήστης να κάνει και πώς αυτό θα υλοποιηθεί έπειτα από το λογισμικό. Το μήνυμα διατυπώνεται σε φυσική γλώσσα και αναλύεται στην μηχανή κατανόησης φυσικής γλώσσας.

Δράση (action)

Το επόμενο στάδιο από το σκοπό του χρήστη, μόλις ανιχνευθεί από την μηχανή κατανόησης είναι να επεξεργαστεί το μήνυμα που στάλθηκε και να ανατρέξει στις κατάλληλες προγραμματιστικές μεθόδους ώστε να παράγει αυτό που θέλει ο χρήστης.

Για παράδειγμα, σε ένα διάλογο ο οποίος αφορά την ερώτηση «τι καιρό κάνει στην Βουδαπέστη;», η δράση που θα πρέπει ο διαλογικός πράκτορας να κάνει είναι να συνδεθεί σε μια βάση δεδομένων (είτε τοπικά ή απομακρυσμένα μέσω web υπηρεσιών)

και να βρει τον καιρό που έχει τώρα η Βουδαπέστη. Έπειτα, θα πρέπει να επιστρέψει την απάντηση σε μια ανθρωπίνως αποδεκτή απάντηση, σε αντίθεση με τα δεδομένα που μπορεί να έχει ανακτήσει από τη βάση δεδομένων.

Εξαγωγή πληροφοριών μέσω διερεύνησης (slot-filling)

Ο τύπος διαλόγου που θα χρησιμοποιηθεί για τα παραδείγματα αλλά και για την υλοποίηση ονομάζεται Slot-Filling. Η μεθοδολογία του διαλόγου αυτού ακολουθεί μια γραμμική μορφή με διακλαδώσεις και αφορά την απόκτηση πληροφοριών από το χρήστη με μορφή φυσικής γλώσσας και ανάθεση τους ως παραμέτρους τόσο σαν οντότητες, ενέργειες, σκοπούς χρήστη όσο και σαν παραμέτρους για το υλοποιημένο πρόγραμμα. [40]

3.4 Περιγραφή frameworks

Στο υπόλοιπο του κεφαλαίου θα περιγράφουν τα frameworks wit.ai, api.ai και Microsoft bot framework, αρχίζοντας με γενικές πληροφορίες, συνεχίζοντας με τα χαρακτηριστικά που έχουν, παραδείγματα χρήσης και υποστηριζόμενες τεχνολογίες που μπορούν να παρέχουν. Σκοπός της περιγραφής, είναι η εκτίμηση των frameworks ως προς τις ανάγκες ανάπτυξης ενός προγράμματος ψηφιακού βοηθού, βρίσκοντας τα κατάλληλα χαρακτηριστικά και δυνατότητες τους καθώς και το βαθμό ευκολίας της ανάπτυξης ενός τέτοιου προγράμματος.

3.4.1 Case Study

Για να γίνει η περιγραφή και σύγκριση των frameworks, χρησιμοποιούμε ένα μικρό παράδειγμα το οποίο θα συγκροτεί ένα ψηφιακό βοηθό ανάκτησης καιρού. Αυτό μπορεί να είναι ενα μικρό κομμάτι σε ένα ευρύτερο ψηφιακό βοηθό ή ενα αρχικό συστατικό για ενα μεγαλύτερο διαλογικό σύστημα. Ο διάλογος θα γίνεται μεταξύ του προγράμματος και του χρήστη. Η πιο απλή μορφή του διαλόγου που επιδιώκουμε είναι η εξής:

Χρήστης: I would like to know the weather in Athens for tomorrow.
Chatbot: The weather in Athens would be 24 Celsius Degrees.

Στη παραπάνω πρόταση του χρήστη θεωρούμε ως σκοπό χρήστη (intent) την "αναζήτηση του καιρού" και ως ονομαστική οντότητα την λέξη Athens (Περιοχή). Επίσης χρησιμοποιούμε την μονάδα του χρόνου ως ονομαστική οντότητα για να ξεκαθαρίσουμε ποια χρονική στιγμή θέλουμε να γνωρίζουμε την θερμοκρασία στην τοποθεσία. Η μονάδα του χρόνου σε αυτή την περίπτωση λαμβάνεται υποχρεωτικά, αλλά χρησιμοποιείται διαφορετικά σύμφωνα με την λογική του προγράμματος.

Χρησιμοποιώντας τις οντότητες και τον σκοπό του χρήστη, το πρόγραμμα μπορεί να καταχωρίσει την οντότητα ως μεταβλητή απαραίτητη για την χρήση μιας υπηρεσίας καιρού η οποία δέχεται ως μεταβλητές την τοποθεσία. Έπειτα αυτό μπορεί να χρησιμοποιηθεί για να ανακτήσει τα δεδομένα (της υπηρεσίας) και να τα επιστρέψει στον χρήστη, παράγοντας το τελικό κείμενο "The weather in Athens for tomorrow is 24 Celsius Degrees". Εάν ο χρήστης μένει στην τοποθεσία της Αθήνας, τότε η παραπάνω πρόταση μπορεί να εκφραστεί ως:

Χρήστης: I would like to know the weather.

Λεκτικά και με την ανθρώπινη κατανόηση αυτό είναι μια ορθή πρόταση όμως στο κόσμο που δημιουργεί ο υπολογιστής για την κατανόηση της φυσικής γλώσσας, δεν είναι αρκετό. Ταυτόχρονα, λείπει το κομμάτι της μεταβλητής που θα χρειαστεί ως παράμετρο για την υπηρεσία ανάκτησης. Αν και υπάρχουν τεχνικοί τρόποι για να αναγνωριστεί η τοποθεσία του χρήστη με την χρήση GPS υπηρεσιών (http://ieeexplore.ieee.org/abstract/document/993780/) η τη χρήση IP Geolocation (http://dl.acm.org/citation.cfm?id=1864380), ο πιο άμεσος τρόπος μπορεί να γίνει μέσω ερωτήματος στο χρήστη:

Χρήστης: I would like to know the weather.
Chatbot: In which location?
Χρήστης: In Athens, Greece.
Chatbot: When?
Χρήστης: Tomorrow
Chatbot: The weather in Athens would be 24 Celsius Degrees.

Εδώ παρατηρούμε οτι η πολυπλοκότητα αυξάνεται καθώς αυξάνεται και η διάρκεια συνομιλίας του χρήστη. Η εφαρμογή μη γνωρίζοντας συγκεκριμένες πληροφορίες, μπορεί να προγραμματιστεί στην ανάκτηση της πληροφορίας από το χρήστη. Τελικός στόχος είναι η εκπλήρωση του σκοπού του χρήστη, της αναζήτησης του καιρού.

3.4.2 Σύνδεση κώδικα και σκοπού

Σύμφωνα με τη προηγούμενη ενότητα, κάθε σκοπός αντιστοιχεί σε μια ενέργεια η οποία πρέπει να εκφραστεί με την υλοποίηση, είτε τοπικά είτε με την χρήση μιας απομακρυσμένης υπηρεσίας, που θα υπολογίσει το καιρό σε κάποια χρονική στιγμή. Στην υλοποίηση, μεγάλο ρόλο παίζουν οι οντότητες οι οποίες αντιστοιχούν με τις μεταβλητές που θα πρέπει το υπόλοιπο πρόγραμμα να επεξεργαστεί, να στείλει απομακρυσμένα σε κάποιο server και να επεξεργαστεί το εισερχόμενο αποτέλεσμα ώστε να αποτελέσει μια ικανή έξοδος πρός το χρήστη, εκπληρώνοντας έτσι την παραγωγή κειμένου.

3.5 Wit.ai

Το λογισμικό wit.ai είναι μια διεπαφή φυσικής γλώσσας για άλλα λογισμικά στα οποία είναι ικανό να μετατρέψει προτάσεις σε δομημένα δεδομένα. Το API του αξιοποιεί υπηρεσίες REST για να επιστρέψει το νόημα από μια πρόταση βασισμένη σε παραδείγματα [41]. Συντηρείται από την εταιρεία Facebook η οποία εξαγόρασε την αρχική εταιρεία παραγωγής του για την ενσωμάτωση του κώδικα του στις δικές της εφαρμογές αλλά και την διάθεση του σαν βιβλιοθήκη (API) .

3.5.1 Σχεδιασμός διαλογικού πράκτορα με το wit.ai

Η σχεδίαση και υλοποίηση ενός διαλογικού πράκτορα με το wit.ai, περιλαμβάνει διάφορα στάδια. Κυρίαρχο ρόλο, έχει το διαχειριστικό σύστημα του wit.ai, στο οποίο καθορίζονται οι απαραίτητες παράμετροι για την λειτουργία του διαλογικού πράκτορα. Το wit.ai δεν είναι μόνο μια μονάδα κατανόησης φυσικής γλώσσας. Αποτελεί ένα οικοσύστημα το οποίο ενσωματώνεται ως μία αυτόνομη υπηρεσία σε μια εφαρμογή η οποία έχει σχεδιαστεί να αντλεί πληροφορίες από το χρήστη μέσω διαλόγου, να ψάχνει σε γνωσιακές βάσεις και βάσεις δεδομένων για πληροφορίες έχοντας αναγνωρίσει ονομαστικές οντότητες ως παραμέτρους και να χρησιμοποιείται για να εκτελεί λειτουργίες οι οποίες καθορίζονται μέσω διαλόγου.

Εφαρμογές (Apps) στο wit.ai

Για να σχεδιαστεί ένας διαλογικός πράκτορας με το wit.ai, θα χρειαστεί μια κεντρική ιδέα υλοποίησης. Στο wit.ai αυτό ισοδυναμεί με μια εφαρμογή (app). Στις ρυθμίσεις της εφαρμογής ο δημιουργός του διαλογικού πράκτορα επιλέγει αν η εφαρμογή θα είναι δημόσια, η ιδιωτική, την γλώσσα στην οποία θα εκτελεί αλγόριθμους μηχανικής μάθησης για την επεξεργασία φυσικής γλώσσας και μια σύντομη περιγραφή του όλου έργου. Επίσης κρατά πληροφορίες οι οποίες χρειάζονται κατά την δημοσίευση της εφαρμογής ή την ενσωμάτωση της σε κάποιο δίαυλο διαλογικών πρακτόρων ή σε κάποια εφαρμογή/υπηρεσία κινητών συσκευών ή εφαρμογών web.

Το κυριότερο όμως χαρακτηριστικό που παρέχει η εφαρμογή είναι η αυτονομία των δεδομένων που περιέχει. Κάθε εφαρμογή στο wit.ai λειτουργεί απομονωμένα από άλλες εφαρμογές (sandbox). Αυτό σημαίνει ότι, οτιδήποτε πληροφορία σχετικά με την εφαρμογή, δεδομένα, url για υπηρεσίες, εκπαίδευση του διαλογικού πράκτορα, διάλογοι και ιστορίες που περιέχει, λειτουργούν στα πλαίσια της συγκεκριμένης εφαρμογής. Μια νέα εφαρμογή δεν μπορεί να προσπελάσει ποτέ τα δεδομένα μιας άλλης. Η εφαρμογή, είναι η κορυφή της ιεραρχίας δομημένων πληροφοριών για το διαλογικό πράκτορα στο wit.ai. Για την παραμετροποίηση του διαλογικού πράκτορα χρειάζεται η συγκρότηση ιστοριών, οντοτήτων και σκοπών χρήστη, οι οποίες θα αλληλοεπιδράσουν με τον προγραμματιστικό κώδικα που παράγει το wit.ai μέσω βιβλιοθηκών ή υπηρεσιών REST.

Ιστορίες (stories)

Οι ιστορίες αποτελούν το τμήμα του wit.ai στο οποίο απλώνεται ο σχεδιασμός της κύριας διαλογικής εφαρμογής. Αυτό σημαίνει ότι πριν την έναρξη μιας εφαρμογής διαλογικού πράκτορα, πρέπει να υπάρχει ένα πλάνο των ερωτήσεων και απαντήσεων που θα δεχτεί ο χρήστης και θα στείλει αντίστοιχα προς την εφαρμογή (διάλογος) η οποία θα χρησιμοποιεί το wit.ai. Ο διάλογος είναι η διεπαφή του χρήστη με το διαλογικό πράκτορα και εκτελείται γραμμικά, ακολουθώντας ορισμένες προτάσεις που εισάγει ο χρήστης στο κομμάτι των ιστοριών. Για παράδειγμα, ο χρήστης θέλει να μάθει την θερμοκρασία που θα έχει μια περιοχή σε μια συγκεκριμένη χρονική περίοδο ή στιγμή. Ο διαλογικός πράκτορας για να μπορέσει να αναγνωρίσει την μονάδα του χρόνου, την

τοποθεσία, ακόμα και την ερώτηση που θέλει να κάνει ο χρήστης, πρέπει να έχει οριστεί μέσα σε μια ιστορία.

Κάθε δομημένη πληροφορία που μπορεί να αναλυθεί μέσω της επεξεργασίας φυσικής γλώσσας, κατανοείται από το wit.ai και προχωρά στην αναγνώριση των παραμέτρων οι οποίες είναι απαραίτητες για να συνεχίσει ο διάλογος. Κάθε ιστορία αποτελείται από τμήματα (σενάρια) διαλόγων. Οι διάλογοι αυτοί έχουν κάποιο σκοπό ο οποίος τελικά θα μεταφραστεί προγραμματιστικά σε μια μέθοδο ή απομακρυσμένη κλήση συστήματος ή ακόμα μια REST υπηρεσία. Εφόσον η κατανόηση είναι επαρκής, η αξιολόγηση του διαλογικού πράκτορα θα γίνει με το πόσο ικανοποιητικά η ιστορία εκτελέστηκε επιτυχώς.

Στις ιστορίες, το wit.ai παρέχει μια εργαλειοθήκη στην οποία ο χρήστης μπορεί να σχεδιάσει 4 βασικά στοιχεία σχετικά με την δομή του διαλογικού πράκτορα. Οι δομές αυτές είναι:

1. **User Says**
   Η δομή «user says» χρησιμοποιείται για να εισάγει ο χρήστης τις πληροφορίες που ζητάει ο διαλογικός πράκτορας ή για να αρχίσει τον διάλογο. Στη λειτουργία αυτή, γίνεται η ανάλυση μιας πρότασης η οποία μπορεί να εισάγει ο χρήστης. Στην ανάλυση αυτή αναγνωρίζεται ο σκοπός (intent) και οι οντότητες/παράμετροι που έχουν βρεθεί. Η πρόταση αυτή αποτελεί το πρότυπο στο οποίο θα βασιστεί η μονάδα κατανόησης της φυσικής γλώσσας για να αντλήσει τις πληροφορίες (οντότητες) που χρειάζεται. Περαιτέρω ανάλυση γίνεται μέσω της εκμάθησης που παρέχει το wit.ai (βλ. παρακάτω)
2. **Bot Executes**
   Η λειτουργία αυτή χρησιμοποιείται για τον ορισμό ενεργειών (actions) οι οποίες θα μεταφραστούν σε προγραμματιστικές μεθόδους. Μαζί με την προγραμματιστική μέθοδο η λειτουργία αυτή αναλαμβάνει να στείλει στο πρόγραμμα ότι οντότητα έχει αναγνωρίσει και μπορεί να επεξεργαστεί. Τέλος κρατάει μια κατάσταση του διαλόγου (αντικείμενο context) στο οποίο διατηρούνται οι οντότητες και άλλες πληροφορίες σχετικά με τη ροή του διαλόγου. Στο τμήμα αυτό δημιουργούνται και τα context keys, τα οποία είναι placeholder μεταβλητές οι οποίες χρησιμοποιεί το context για να επιστρέψει την

απόκριση από την μέθοδο func ή να κάνει έλεγχο για τιμές οντοτήτων που δεν έχουν οριστεί (missing entities).

3. **Jump**

   Χρησιμοποιείται για την μετάβαση σε ένα άλλο σημείο του διαλόγου, εφόσον οι προϋποθέσεις πληρούνται. Τα σημεία του διαλόγου που είναι επιτρεπτά ορίζονται με ένα σύμβολο σελιδοδείκτη (bookmark) και μια λεζάντα την οποία χρησιμοποιεί για να μεταβεί στο σημείο αυτό, το οποίο μπορεί να είναι αρχή διαλόγου από το «user says», ή επανάληψη ερώτησης μέσω του «bot sends».

4. **Bot Sends**

   Εδώ ο διαλογικός πράκτορας πρέπει να επιστρέψει μια απάντηση στο χρήστη. Δημιουργεί δηλαδή φυσική γλώσσα (υπό την μορφή κειμένου string) στην οποία επικοινωνεί στο χρήστη είτε κάποιο αποτέλεσμα από κάποια κλήση υπηρεσίας ή κάποιο αποτέλεσμα από κάποιο τοπικό υπολογισμό. Οι μεταβλητές που ορίστηκαν στο τμήμα «Bot Executes», μπορούν να εμφανιστούν εδώ σαν πρότυπη απάντηση διαλόγου.

Ονομαστικές οντότητες / Παράμετροι μεθόδων στο wit.ai

Για να μπορέσει το wit.ai να χρησιμοποιήσει τις απαραίτητες πληροφορίες στην κλήση μεθόδων, τις παραμέτρους, πρέπει πρώτα να αναγνωριστούν από την ροή της φυσικής γλώσσας χρησιμοποιώντας ονομαστικές οντότητες. Οι ονομαστικές οντότητες στο wit.ai χρησιμοποιούν είτε την επεξεργασία φυσικής γλώσσας (nlp) ή την μηχανική μάθηση (machine learning) για την αυτόματη αναγνώριση τους μέσα σε μια πρόταση, ή την χρήση χειρωνακτικών τρόπων σε περίπτωση που οι αλγόριθμοι που παρέχουν δεν αναγνωρίσουν κάποια στο περιεχόμενο του κειμένου. Αυτό ορίζεται μέσα στο διαχειριστικό κομμάτι του wit.ai χρησιμοποιώντας την επιλογή των λέξεων οι οποίες μεταφράζονται ως ονομαστικές οντότητες. Καθώς γίνεται η εκπαίδευση του διαλογικού πράκτορα, η εφαρμογή αποκτά περισσότερες αναφορές ώστε να μπορεί να αναγνωρίσει καλύτερα τις ονομαστικές οντότητες.

Στο wit.ai, οι ονομαστικές οντότητες εκτός από τη σημασία που έχουν μέσα στην επεξεργασία φυσικής γλώσσας, αποτελούν το σημείο εισόδου των παραμέτρων για τις μεθόδους που θα χρησιμοποιηθούν μέσα στο τελικό πρόγραμμα του διαλογικού πράκτορα. Έτσι, βασικοί τύποι μεταβλητών π.χ. integer, string, boolean, float υπάρχουν

για τον ορισμό μιας απλής οντότητας, αν και προγραμματιστικά όλες συμπεριφέρονται σαν strings, ενώ οι κατασκευαστές του wit.ai έχουν ομαδοποιήσει και προκαθορίσει μεταβλητές οι οποίες έχουν άμεση σχέση με τα συμφραζόμενα της πρότασης. Π.χ. όταν σε μια πρόταση αναφέρεται χρόνος, αυτό θα μεταφραστεί αυτόματα σε μια μεταβλητή ημερομηνίας, ή ώρας ή ακόμα και μια χρονική περίοδος.

Σκοπός χρήστη στο wit.ai

Κάθε διάλογος της ιστορίας που υπάρχει στο wit.ai, αποτελείται από σκοπούς του χρήστη. Για να οριστεί ένας σκοπός χρήστη, χρησιμοποιείται μια ειδική προκαθορισμένη οντότητα που έχει το wit.ai, με την ονομασία intent. Σημειώνοντας ότι η πρόταση του διαλόγου έχει ως intent το σκοπό του χρήστη ή μια λέξη κλειδί σχετική με το σκοπό του χρήστη. Σε ένα πολύπλοκο διάλογο που εκτελείται από ένα διαλογικό πράκτορα, αυτή η ταξινόμηση των σκοπών χρήστη μπορεί να βοηθήσει στο προγραμματιστικό τμήμα της υλοποίησης της εφαρμογής.

Ενέργειες στο wit.ai

Μια ενέργεια στο wit.ai αντιστοιχεί σε μια προγραμματιστική μέθοδο που υλοποιείται στο πρόγραμμα του διαλογικού πράκτορα. Οι ενέργειες ορίζονται στο τμήμα που ενας διαλογικός πράκτορας έχει ήδη αναγνωρίσει τον σκοπό (Bot executes) και διακόπτει την λειτουργία του διαλόγου ώστε να ζητήσει πληροφορίες από μια εξωτερική πηγή η μια βάση δεδομένων ή να υπολογίσει κάτι χρησιμοποιώντας το ίδιο το πρόγραμμα του διαλογικού πράκτορα.

Διακλαδώσεις, σημεία ελέγχου του διαλόγου

Στο φυσικό διάλογο, έτσι και στους διαλογικούς πράκτορες, η έκβαση ενός διαλόγου εξαρτάται από τις απαντήσεις που δίνει ο χρήστης. Σε κάποια χρονική στιγμή, ο χρήστης που επικοινωνεί με τον διαλογικό πράκτορα θα αναγκαστεί να δώσει μια θετική ή μια αρνητική ή μια διφορούμενη απάντηση. Ανάλογα με το σχεδιασμό που έχει γίνει κατά την θεωρητική υλοποίηση του διαλογικού πράκτορα, είτε για ολόκληρο το διάλογο ή για το συγκεκριμένο κομμάτι του διαλόγου πρέπει να έχουν προβλεφθεί οι αντίστοιχες απαντήσεις. Το wit.ai μπορεί να ελέγξει αυτού του είδους τις απαντήσεις,

χρησιμοποιώντας διακλαδώσεις (forks). Έτσι μπορεί να αναλάβει να ελέγξει διάφορες απαντήσεις με ναι ή όχι ή εντολές ανοίγματος/κλεισίματος. Αντίστοιχα μπορεί να χρησιμοποιηθεί ένα εύρος απαντήσεων για διφορούμενες απαντήσεις, οι οποίες είτε θα οδηγήσουν το χρήστη να κάνει μια επιλογή των «σωστών απαντήσεων» ή να τερματίσουν τη ροή του διαλόγου.

Όπως προαναφέρθηκε, μπορούν να υπάρχουν πολλαπλοί σκοποί χρήστη σε ενα διάλογο. Αντίστοιχα όμως και ένας σκοπός μπορεί να έχει πολλαπλές ενέργειες στον ίδιο διάλογο. Έτσι ορίσματα κυρίως των λειτουργιών «bot sends» μπορούν να οδηγούν σε διαφορετικές κλήσεις μεθόδων για την εκτέλεση ενός σκοπού. Αυτό επιτυγχάνεται με την διακλάδωση των «bot sends». Εξαρτάται από την στρατηγική που θα επιλέξει ο σχεδιαστής του διαλογικού πράκτορα κατά την σύλληψη της ιδέας και την υλοποίηση της.

Τύποι διαλόγων στο wit.ai

Οι διάλογοι που το wit.ai μπορεί να χειριστεί είναι κυρίως γραμμικοί (Απρίλιος 2017). Η γραμμικότητα του διαλόγου δεν σημαίνει οτι δεν μπορεί να χρησιμοποιήσει διακλαδώσεις αλλά για το συγκεκριμένο είδος διαλογικού πράκτορα όλες οι λειτουργίες εκτελούνται ακολουθιακά ή τερματίζουν τον διάλογο όταν δεν εκπληρώνεται ένα τμήμα του διαλόγου. Παρόλα αυτά, προσφέρει ένα επιπλέον είδος συλλογής πληροφοριών μέσω διαλόγου, το slot-filling, κατά το οποίο ο δημιουργός του διαλογικού πράκτορα καθορίζει τις μεταβλητές/οντότητες που πρέπει να υπάρχουν ώστε να γίνει προχωρήσει η επιχειρησιακή λογική της απομακρυσμένης μεθόδου.

Σε ενα σύστημα που ο χρήστης θέλει να συλλέξει πληροφορίες για την ημερομηνία και την τοποθεσία, π.χ. για να κάνει κλήση μιας απομακρυσμένης μεθόδου η οποία θα έχει σχέση με τον καιρό, ο διάλογος και κατά συνέπεια ο διαλογικός πράκτορας, θα πρέπει να κρατάει ένα ιστορικό των μεταβλητών που καθορίστηκαν και των εναπομεινάντων αγνώστων παραμέτρων. Σημειώνεται, οτι όλες οι παράμετροι που λαμβάνουν μέρος σε αυτό το είδος διαλόγου είναι υποχρεωτικοί. Οι πληροφορίες που συλλέγονται, συλλέγονται μέσω επερωτήσεων στο χρήστη:

Χρήστης: I want to know the weather.

Εδώ ο χρήστης έχει δώσει αόριστα οτι θέλει να μάθει τον καιρό. Εάν υπήρχε ανάλυση των συμφραζομένων του διαλόγου, τότε θα θεωρούσε ότι εννοεί για την τοποθεσία που βρίσκεται αυτή τη στιγμή, κάτι που δεν συμβαίνει συνήθως. Εάν έχουν οριστεί οι παράμετροι ως υποχρεωτικοί, όπως τοποθεσία (wit/location) και χρόνος (wit/datetime), ο διάλογος θα συνεχίσει με την πρώτη διερευνητική ερώτηση που έχει ορίσει ο χρήστης:

Chatbot: In which location?
Χρήστης: In Athens, Greece.
Chatbot: When?
Χρήστης: In 10 minutes.
Chatbot: The weather in Athens, Greece would be 32^{o}C.

Αυτό που προσπαθεί το slot-filling είναι να ζητήσει τις υπόλοιπες πληροφορίες που δεν είχε κατά την αρχική ερώτηση. Σε περίπτωση που οι παράμετροι αναγνωριστούν ως οντότητες, τότε υπόκειται μέσα στην λογική του προγράμματος που υλοποιεί τον διαλογικό πράκτορα να αναγνωρίσει αν η παράμετρος είναι σωστή (type & value check) ή αν θα χρειαστεί ξανά ερώτηση, π.χ.:

Chatbot: When?
Χρήστης: In Red Triangle (Λάθος)
Chatbot: When?
Χρήστης: Tomorrow. (Σωστή)

Εδώ θα αναγνωριστεί μια οντότητα αλλά δεν είναι η οντότητα του χρόνου. Προγραμματιστικά, ο έλεγχος που γίνεται για το αν αναγνωρίστηκε σωστά η οντότητα ή όχι θα πρέπει να επιστρέψει λάθος, όποτε η ερώτηση του χρόνου θα πρέπει να ξαναρωτηθεί μέχρι να δοθεί μια σωστή απάντηση. Οι αποκρίσεις από το διαλογικό πράκτορα, ρυθμίζονται στο διαχειριστικό σύστημα στην επιλογή "bot executes", μέσω της προσθήκης context keys με πρόθεμα missing<Όνομα μεταβλητής> και διακλάδωσής τους. Η παρακολούθηση της κατάστασης της οντότητας, γίνεται μέσω του προγραμματιστικού αντικειμένου context το οποίο αναλαμβάνει το ρόλο της "συνεδρίας" του διαλόγου αλλά επίσης και του "κόσμου" που δημιουργείται καθώς ο διάλογος επεκτείνεται.

3.5.2 Προγραμματιστικός οδηγός για το wit.ai

Στο wit.ai, τμήμα του διαλογικού πράκτορα χρειάζεται να υλοποιηθεί με κώδικα εκτός του διαχειριστικού συστήματος. Αυτό επιτυγχάνεται μέσω έτοιμων βιβλιοθηκών που παρέχει το wit.ai [41] ή μέσω κλήσεων REST που υπάρχουν στην τεκμηρίωση του [42]. Για να συνδεθεί ο κώδικας με τη τρέχουσα εφαρμογή του wit.ai, θα χρειαστούμε ένα access token. Το access token το προμηθευόμαστε από το διαχειριστικό σύστημα του wit.ai, μαζί με το αναγνωριστικό της εφαρμογής (app ID).

Για την javascript η κεντρική βιβλιοθήκη επεξεργασίας του wit.ai είναι η βιβλιοθήκη **node-wit** [43]**.** Η βιβλιοθήκη node-wit παρέχει τις απαραίτητες μεθόδους για την επικοινωνία και υλοποίηση των επιμέρους τμημάτων κώδικα που χρειάζεται ώστε να γίνεται η αναγνώριση των σκοπών του χρήστη, των οντοτήτων και τελικώς η επίτευξη των ενεργειών (κλήσεις άλλων υπηρεσιών, τοπικές μέθοδοι) που έχουν οριοθετηθεί από την ιστορία.

Αρχικοποίηση wit.ai προγραμματιστικά

Το wit.ai ενσωματώνεται σε ένα node.js έργο, αρχικά εισάγοντας τις απαραίτητες βιβλιοθήκες:

```
const {wit, log, interactive} = require('node-wit');
```

Κώδικας 3.1 Εισαγωγή βιβλιοθηκών σε ενα αρχείο node.js

Έπειτα, ο προγραμματιστής πρέπει να αρχικοποιήσει το πελάτη για την πλατφόρμα wit, χρησιμοποιώντας το αντικείμενο Wit:

```
const client = new Wit({
accessToken: '<access token from wit.ai>',
actions: {…}
});
```

Κώδικας 3.2 αρχικοποίηση wit client

Το σκεπτικό πίσω από το wit.ai είναι η δημιουργία ενός διαλογικού πράκτορα ο οποίος μπορεί να ενσωματώνεται είτε σαν τμήμα λογισμικού (software component) σε μια τοπική/web εφαρμογή ή ως αυτόνομη υπηρεσία στην οποία άλλες υπηρεσίες θα συνδέονται για να χρησιμοποιήσουν το διάλογο.

Το wit.ai παρέχει δύο ειδών διαλογικούς πράκτορες. Τον διαδραστικό (interactive console) και τον πράκτορα μέσω υπηρεσιών. Ο πράκτορας μέσω υπηρεσιών χρησιμοποιείται μέσα σε προγραμματιστικές βιβλιοθήκες άλλων προγραμμάτων, όπως για παράδειγμα με την προγραμματιστική βιβλιοθήκη του καναλιού διαλογικού πράκτορα slack ή του skype και άλλων. Οι οδηγίες για την ανάπτυξη τέτοιων πρακτόρων βρίσκονται στα παραδείγματα του wit.ai στο github [44].

Ο διαδραστικός διαλογικός πράκτορας αρχικοποιείται χρησιμοποιώντας το αντικείμενο interactive στον κώδικα της εφαρμογής, αμέσως μετά την αρχικοποίηση του αντικειμένου Wit:

```
interactive(client);
```

Κώδικας 3.3 αρχικοποίηση διαδραστικού περιβάλλοντος wit.ai

Ορισμός ενεργειών στο κώδικα του wit.ai

Κατά την δημιουργία του διαλογικού πράκτορα, έγινε ορισμός κάποιων ενεργειών οι οποίες αντιστοιχούν σε σκοπούς του χρήστη. Για να υλοποιηθούν οι ενέργειες αυτές, χρησιμοποιούμε την παράμετρο actions από το αντικείμενο client του wit.ai. Η παράμετρος actions αποτελείται από ένα σύνολο προγραμματιστικών μεθόδων οι οποίες αντιστοιχούν στην ονομασία που δίνουμε στο πεδίο func της δομής «Bot Executes» στο διαχειριστικό του wit.ai. Αν η ιστορία που αναπτύσσουμε έχει π.χ. 2 func ορισμένες σε 2 «bot executes» δομές και αυτές είναι αντίστοιχα οι getTemperature() και getEvents(), τότε η παράμετρος action θα πρέπει να έχει αντίστοιχα ορισμένες τις μεθόδους αυτές.

Στο actions πάντα ορίζεται και μια μέθοδος `send` με παραμέτρους `request` και `response`. Η μέθοδος αυτή αναλαμβάνει να στείλει πίσω στο διαχειριστικό τμήματα κειμένου (παραγωγή φυσικής γλώσσας) τα οποία επεξεργάζεται ξανά για να βελτιώσει την κατανόηση του διαλογικού πράκτορα (περιοχή inbox, περιοχή understanding). Επίσης

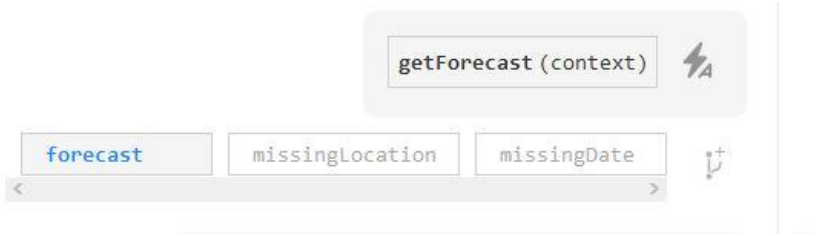
στέλνει προς το χρήστη το μήνυμα που έχουμε ορίσει σαν «bot sends». Ο κώδικας της μεθόδου αυτής είναι:

```
send(request, response) {
  return new Promise(function(resolve, reject) {
    console.log(response.text);
    return resolve();
  });
},
```

Κώδικας 3.4 ανάλυση της μεθόδου send.

Το αντικείμενο `Promise`, χρησιμοποιείται εδώ για να βοηθήσει την μέθοδο `send()` να εκτελεστεί ασύγχρονα, επιταχύνοντας έτσι την απόκριση του προγράμματος. Χρησιμοποιούμε την έξοδο της console για να προβάλλουμε το κείμενο απάντησης από το wit.ai, το οποίο αντιστοιχεί στην επόμενη ερώτηση που εκτελεί η ροή του διαλόγου.

Στο παράδειγμα του καιρού, έχουμε ορίσει στο διαχειριστικό την getForecast():



Εικόνα 3.4 Ορισμός της func στο "bot executes" στο διαχειριστικό wit.ai

Αντιστοίχως, στο κώδικα θα πρέπει να ορίσουμε την getForecast(context):

```
actions: {
send(request, response) {…},
getForecast({sessionId, context, text, entities}) {…}
},
```

Κώδικας 3.5 ορισμός της func στο κώδικα του διαλογικού πράκτορα

Παρατηρούμε ότι ενώ στο διαχειριστικό η μέθοδος getForecast έχει μόνο το context ως όρισμα, στο κώδικα η μέθοδος αυτή έχει περισσότερες παραμέτρους. Αυτό οφείλεται

στην αλλαγή των προδιαγραφών στο wit.ai, με τη χρήση νεότερης έκδοσης της βιβλιοθήκης node-wit.

Τα ορίσματα της μεθόδου αυτής είναι μια λίστα η οποία περιλαμβάνει το `sessionId`, το αντικείμενο `context`, το κείμενο (`text`) που εισάγει ο χρήστης, όπως το αντιλαμβάνεται το wit.ai και τα `entities`. Το `sessionId` χρησιμοποιείται ως ένα token για να μπορεί να στέλνει μηνύματα REST πίσω στο wit.ai. Οι βιβλιοθήκες που χρησιμοποιεί το wit.ai, στην ουσία ενθυλακώνουν αιτήσεις REST (POST/GET) ανάλογα με το προγραμματιστικό μοντέλο της κάθε γλώσσας προγραμματισμού. Η παράμετρος `context` είναι το αντικείμενο που κρατάει τις πληροφορίες του διαλόγου, όσο κρατάει η συνομιλία μέχρι την εκτέλεση του σκοπού του χρήστη. Η παράμετρος text παρουσιάζει το κείμενο που έδωσε ο χρήστης. Τα `entities`, αποτελούν ένα αντικείμενο το οποίο περιέχει όλες τις αναγνωρισμένες ονομαστικές οντότητες και τις τιμές τους καθώς επιπλέον μεταδεδομένα (metadata) που είναι χρήσιμα για την ανάλυση. Παράλληλα περιέχουν για στατιστικούς λόγους την πιθανότητα που η κάθε λέξη συσχετίστηκε με το συγκεκριμένο entity. Η παρακάτω λίστα entities ανακτήθηκε κατά την εκτέλεση του παραδείγματος του καιρού με αρχικό κείμενο:

Χρήστης: I want to know the weather in cuba now.

Το αποτέλεσμα είναι μια λίστα entities η οποία περιέχει την τοποθεσία, την ημερομηνία και το σκοπό (intent) του χρήστη:


```
{
 "location": [
   {
     "confidence": 0.994990102154969,
     "type": "value",
     "value": "cuba",
     "suggested": true
   }
 ],
 "datetime": [
   {
     "confidence": 0.9243621462064444,
     "values": [
       {
```

```
          "value": "2017-05-18T23:00:38.000+03:00",
          "grain": "second",
          "type": "value"
        }
      ],
      "value": "2017-05-18T23:00:38.000+03:00",
      "grain": "second",
      "type": "value"
    }
  ],
  "intent": [
    {
      "confidence": 1,
      "value": "forecast"
    }
  ]
}
```


Κώδικας 3.6 Ανάλυση του αντικειμένου entity του wit.ai.

Στην μέθοδο getForecast(), είναι απαραίτητες οι πληροφορίες τοποθεσίας και μη προαιρετικές οι πληροφορίες της ημερομηνίας. Παρόλα αυτά, χρησιμοποιούμε στην συλλογή μέσω slot-filling την ημερομηνία σαν υποχρεωτική παράμετρο.

Στον κώδικα της getForecast() μεθόδου, πρέπει να εξετάσουμε αν έχουν οριστεί σωστά οι παραπάνω πληροφορίες από τις οντότητες. Σε περίπτωση που δεν έχει οριστεί κάποια από αυτές τις πληροφορίες, το wit.ai μπαίνει σε διερευνητική διαδικασία μέσω slot-filling ψάχνοντας τις πληροφορίες αυτές, εκτελώντας τα σενάρια `missingDate` και `missingLocation`.

Για να ανακτήσουμε τις τιμές από το αντικείμενο των οντοτήτων, καταρχάς χρησιμοποιούμε μια βοηθητική μέθοδο, την firstEntityValue. Η μέθοδος αυτή έχει δύο ορίσματα: το αντικείμενο entities και την οντότητα entity που ψάχνουμε. Η firstEntityValue ψάχνει το αντικείμενο των entities για μια μεταβλητή με όνομα entity. Εάν βρεθεί η μεταβλητή τότε ελέγχει αν η μεταβλητή είναι μιας τιμής, αντικείμενο ή λίστα και επιστρέφει αναλόγως την τιμή (value) της. Εάν δεν βρεθεί, τότε επιστρέφει κενή τιμή (null value).

Σε περίπτωση που βρεθεί η τιμή που ζητάμε και είναι μέρος της διαδικασίας slot-filling, τότε θα πρέπει μέσα στο πρόγραμμα να απενεργοποιηθεί η συγκεκριμένη αναζήτηση της τιμής και η μεταβλητή αυτή να εισαχθεί στο αντικείμενο context ώστε να μην αναζητηθεί σε επόμενη κλήση της getForecast(). Για την μεταβλητή location, υπάρχει και η τιμή missingLocation (άρα είναι μέρος του slot-filling), οπότε η διαδικασία που θα πρέπει να ακολουθηθεί είναι:

```
var location = firstEntityValue(entities, “location”);
if (location) { context.location = location } else {
    context.missingLocation = true;
    delete context.forecast;
}
if (context.location) {
    delete context.missingLocation;
}
```

Κώδικας 3.7 Απόδοση τιμής στη μεταβλητή location

Στο παραπάνω κώδικα, γίνεται η ανάκτηση από το αντικείμενο entities της τιμής location. Σε περίπτωση που υπάρχει η τιμή, ανατίθεται και στο context ως παράμετρος του αντικειμένου. Ειδάλλως πρέπει να θεωρηθεί ότι δεν υπάρχει τιμή για την location και να αρχίσει η διαδικασία slot-filling, που γίνεται δηλώνοντας το `context.missingLocation` με τιμή true και διαγράφοντας την μεταβλητή forecast από το αντικείμενο context. Η μεταβλητή forecast κρατάει τις τελικές πληροφορίες που θα παρουσιαστούν στο χρήστη σαν κείμενο. Αντιθέτως, αν υπάρχει και η context.location ορισμένη, τότε διαγράφουμε το τμήμα του slot-filling σχετικά με το missingLocation μιας που θεωρείται ότι έχει ήδη ανακτηθεί η πληροφορία. Η διαδικασία αυτή χρησιμοποιείται για όλες τις μεταβλητές / οντότητες που χρησιμοποιούνται σε κάθε διάλογο και θεωρείται ως απαραίτητο κομμάτι πριν την κλήση της μεθόδου που θα επιστρέψει επεξεργασμένα δεδομένα για την παραγωγή κειμένου. Ο παραπάνω τρόπος ταυτοποίησης είναι ένας από τους προτεινόμενους τρόπους, χωρίς όμως να είναι ο ιδανικός ή ορθά διατυπωμένος για την υλοποίηση του διαλογικού πράκτορα.

Έχοντας ταυτοποιήσει και ελέγξει τις μεταβλητές που χρειαζόμαστε για την κλήση της υπηρεσίας μπορούμε να προχωρήσουμε στην εκτέλεση των απομακρυσμένων μεθόδων. Στο παράδειγμα, θέλουμε την κλήση της υπηρεσίας καιρού, η οποία όμως απαιτεί γεωγραφικό μήκος και πλάτος για να λειτουργήσει. Έτσι πρέπει να εκτελέσουμε πρώτα

την κλήση μιας άλλης υπηρεσίας η οποία θα μετατρέψει την μεταβλητή της τοποθεσίας σε ζεύγος τιμών longitude και latitude. Για τις κλήσεις αυτών των υπηρεσιών θα χρησιμοποιήσουμε μια επιπλέον βιβλιοθήκη ασύγχρονης κλήσης εξωτερικών υπηρεσιών την **axios** η οποία έχει ενσωματωμένη την λειτουργία των promises [45].

```
if (context.location && context.date) {
                    // url encode the location
                    var encodedAddress = encodeURIComponent(context.location);
                    var geocodeURL =
`https://maps.googleapis.com/maps/api/geocode/json?address=${encodedAddress}`;

                    // first do a reverse geolocation from the location given
                    return axios.get(geocodeURL).then((response) => {
                        if (response.data.status === 'ZERO_RESULTS') {
                            throw new Error('Unable to find that address');
                        }
                        var lat = response.data.results[0].geometry.location.lat;
                        var lng = response.data.results[0].geometry.location.lng;

                        // then do a weather forecast search to forecast.io
                        var weatherURL =
`https://api.darksky.net/forecast/39b78cd34c557b7ac589472a5c6a9c6b/${lat},${lng}?u
nits=si`;
                        console.log(response.data.results[0].formatted_address);
                        return axios.get(weatherURL);
                    }).then((response) => {
                        // if the chained Promise succeeds then set the
temperature to the context object
                        var temperature = response.data.currently.temperature;
                        var apparentTemperature =
response.data.currently.apparentTemperature;
                        // Create the context
                        context.forecast = `${temperature} degrees`;
                        clear_context = true;
                        // return the context object
                        return resolve(context);
                    }).catch((e) => {
                        // Error check for both google reverse code and
    // forecast.io
                        if (e.code === 'ENOTFOUND') {
                            console.log('Unable to connect to API server');
```

```
                } else {
                    console.log(e.message);
                }
                delete context.forecast;
            });
        } else {
            delete context.forecast;
        }
```

Κώδικας 3.8 εκτέλεση των επιμέρους υπηρεσιών για την ανάκτηση του καιρού

Η ροή του κώδικα χρησιμοποιείται ως εξής. Αρχικά γίνεται κλήση του google reverse geocode [46] με όρισμα την τιμή της μεταβλητής της τοποθεσίας. Εφόσον εκπληρωθεί το promise, θα επιστραφεί ένα ζεύγος τιμών lat, lng στο αντικείμενο response. Με αυτά τα ζεύγη τιμών τροφοδοτούμε την κλήση της δεύτερης υπηρεσίας, του forecast.io το οποίο θα μας επιστρέψει πίσω τις τιμές του καιρού. Οι τιμές αυτές είναι η κανονική θερμοκρασία και η αισθητή θερμοκρασία (μεταβλητές temperature, apparentTemperature). Για να επιστραφεί στο wit.ai η τιμή της θερμοκρασίας πρέπει να ορίσουμε τη παράμετρο του context, forecast:

```
var temperature = response.data.currently.temperature;
var apparentTemperature = response.data.currently.apparentTemperature;
// Create the context
context.forecast = `${temperature} degrees`;
clear_context = true;
// return the context object
return resolve(context);
```

Κώδικας 3.9 επιστροφή της μεταβλητής του καιρού πίσω στο wit.ai

Αυτό γίνεται μέσω της εντολής `context.forecast = `${temperature} degrees`;` Επιστρέφοντας το αντικείμενο context πίσω στο wit.ai, θα περιέχεται η πληροφορία που το πρόγραμμα αναγνώρισε ως σκοπό χρήστη (θερμοκρασία μιας περιοχής) και θα δημιουργηθεί μια νέα γραμμή κειμένου μέσα από το πρότυπο που έχει οριστεί στο αντίστοιχο τμήμα του «bot sends» στο διαχειριστικό.

Μηνύματα στο wit.ai

Κάθε ενέργεια που κάνει ο διαλογικός πράκτορας στο wit.ai χρησιμοποιεί το HTTP API, το οποίο έχει ενθυλακωθεί σε μεθόδους ανάλογα με την γλώσσα προγραμματισμού που χρησιμοποιείται. Έτσι ένας διάλογος είναι μια ανταλλαγή μηνυμάτων με την μέθοδο POST μεταξύ του εξυπηρετητή του wit.ai και της εφαρμογής (πελάτης) του διαλογικού πράκτορα. Σε περίπτωση που χρησιμοποιείται η επιλογή του διαδραστικού διαλογικού πράκτορα, μαζί με το αντικείμενο του πελάτη Wit, δημιουργείται και μια συνεδρία, η οποία αναλαμβάνει την διευθέτηση των μηνυμάτων αυτών, χωρίς να γνωρίζει ο χρήστης.

Στο παράδειγμα της εύρεσης του καιρού, ο χρήστης στέλνει την εξής φράση:

Χρήστης: I want to know the weather.

Το μήνυμα που στέλνει η βιβλιοθήκη node-wit στο wit.ai είναι το εξής:

```
POST https://api.wit.ai/converse?session_id=2916a300-3c02-11e7-96b6-
eb5848977155&q=I%20want%20to%20know%20the%20weather.
Response:
{"confidence":0.10687792566038574,"type":"action","action":"getForecast","entities
":{"intent":[{"confidence":1,"value":"forecast"}]}}
Context: {}
Response type: action
```

Κώδικας 3.10 Μήνυμα POST και απάντηση προς το wit.ai

Όταν γίνεται κλήση στο endpoint /converse της υπηρεσίας του wit.ai, αρχίζει η διαδικασία του διαλόγου. Παρατηρούμε ότι οι παράμετροι που λαμβάνει το wit.ai είναι το sessionId και το κείμενο «I want to know the weather», ενώ η απάντηση που στέλνει πίσω στην εφαρμογή του διαλογικού πράκτορα είναι ένα JSON αντικείμενο που έχει τα entities, την ενέργεια που πρέπει να εκτελεστεί (getForecast) και το σκοπό του χρήστη (forecast). Οι τιμές confidence που παρουσιάζονται είναι το ποσοστό πιθανότητας να ανήκει αυτή η φράση σε κάποια από τις αντίστοιχες δομές.

Αμέσως μετά την λήψη του αντικειμένου response, η ροή του προγράμματος προχωρά στην εκτέλεση της μεθόδου που ορίζει η παράμετρος action, δηλαδή της getForecast()

```
POST https://api.wit.ai/converse?session_id=2916a300-3c02-11e7-96b6-eb5848977155
Response: {"confidence":0.07028593231759495,"type":"msg","msg":"In which location would you like to know the weather?"}
Context: {"missingLocation":true,"missingDate":true}
Response type: msg
```

Κώδικας 3.11 Εκτέλεση της getForecast() χωρίς να υπάρχουν ορισμένες οι μεταβλητές

Πρώτο πράγμα που παρατηρούμε είναι ότι στο context δεν υπάρχουν οι μεταβλητές location και date, άρα θα ενεργοποιηθούν τα missingLocation και missingDate (με τιμές true). Αυτό σημαίνει ότι ο διάλογος θα προχωρήσει σε διαδικασία slot-filling και θα ανατρέξει στα αντίστοιχα τμήματα για να στείλει τις διερευνητικές ερωτήσεις. Ήδη το πρώτο μήνυμα έχει οριστεί:

Chatbot: In which location would you like to know the weather?

Για να μπορέσει να δώσει πίσω τον έλεγχο στο χρήστη (και να αναμένει την απάντηση του), πρέπει να σταλθεί ένα μήνυμα τύπου stop:

```
POST https://api.wit.ai/converse?session_id=2916a300-3c02-11e7-96b6-eb5848977155
Response: {"confidence":0.044326724677641434,"type":"stop"}
Context: {"missingLocation":true,"missingDate":true}
Response type: stop
```

Κώδικας 3.12 Μήνυμα επιστροφής ελέγχου του χρήστη

Εδώ ο χρήστης είναι έτοιμος να εισάγει το επόμενο μήνυμα του διαλόγου που έχει σχέση με τη τοποθεσία. Εάν είναι σωστή τοποθεσία τότε στο wit.ai θα αναγνωριστεί η αντίστοιχη οντότητα και θα ξαναγίνει κλήση της getForecast():

Χρήστης: In Nicosia, Cyprus

```
POST https://api.wit.ai/converse?session_id=2916a300-3c02-11e7-96b6-eb5848977155&q=In%20Nicosia%2C%20Cyprus
Response: {"confidence":0.02414588364323616,"type":"action","action":"getForecast","entities":{"location":[{"confidence":0.9955500792726141,"type":"value","value":"Nicosia, Cyprus","suggested":true}],"intent":[{"confidence":1,"value":"forecast"}]}}
Context: {"missingLocation":true,"missingDate":true}
Response type: action
POST https://api.wit.ai/converse?session_id=2916a300-3c02-11e7-96b6-eb5848977155
Response: {"confidence":0.02631931300232624,"type":"msg","msg":"What would be the date?"}
```

```
Context: {"missingDate":true,"location":"Nicosia, Cyprus"}
Response type: msg
```

Κώδικας 3.13 επικύρωση της οντότητας/μεταβλητής location

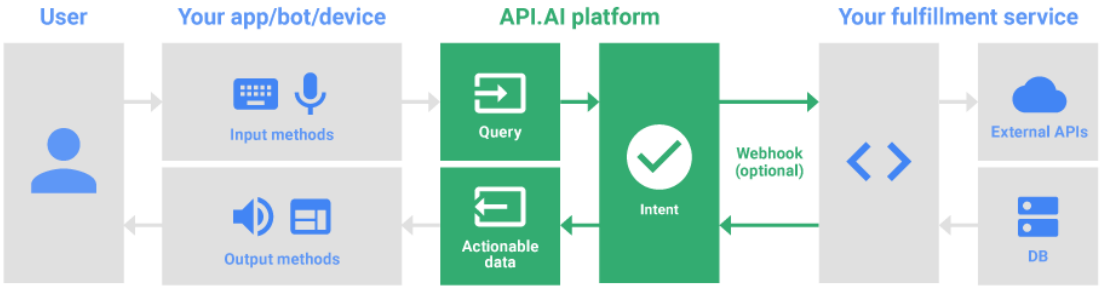
Παρατηρούμε ότι πρώτα γίνεται αναγνώριση της οντότητας, μετά η κλήση της getForecast(), όπου μέσα στην getForecast() γίνεται η ανάθεση της μεταβλητής location στο context και τέλος η συνέχεια για την ανάκτηση της δεύτερης πληροφορίας της ημερομηνίας. Έχοντας και τη τελευταία απαραίτητη οντότητα γίνεται η τελική κλήση της getForecast() η οποία περιέχει την πληροφορία πλέον:

```
POST https://api.wit.ai/converse?session_id=2916a300-3c02-11e7-96b6-eb5848977155
Response: {"confidence":0.025612548516793864,"type":"msg","msg":"The weather in
Nicosia, Cyprus will be 19.31 degrees in 2017-05-18T22:43:30.000+03:00"}
Context: {"location":"Nicosia, Cyprus","date":"2017-05-
18T22:43:30.000+03:00","forecast":"19.31 degrees"}
Response type: msg
The weather in Nicosia, Cyprus will be 19.31 degrees in 2017-05-
18T22:43:30.000+03:00
POST https://api.wit.ai/converse?session_id=2916a300-3c02-11e7-96b6-eb5848977155
Response: {"confidence":0.02410489702837743,"type":"stop"}
Context: {"location":"Nicosia, Cyprus","date":"2017-05-
18T22:43:30.000+03:00","forecast":"19.31 degrees"}
Response type: stop
```

Κώδικας 3.14 επιστροφή της πληροφορίας forecast

3.6 Api.ai

Το λογισμικό api.ai [47] είναι μια πλατφόρμα για την δημιουργία διαλογικών πρακτόρων και ειδικότερα διαλογικών διεπαφών για bot, εφαρμογές και κινητές/IoT συσκευές.



Εικόνα 3.5 γενική αρχιτεκτονική api.ai [45]

3.6.1 Σχεδίαση διαλογικού πράκτορα με το api.ai

Κώδικας3.15, η πλατφόρμα παρέχει τα τμήματα του API.AI Platform ως υπηρεσίες μέσα σε ένα πληροφοριακό σύστημα διαλογικού πράκτορα. Η υλοποίηση των επιμέρους τμημάτων, είσοδος/έξοδος πληροφοριών από/προς το χρήστη και η «πραγματοποίηση» (fulfillment) της επικοινωνίας με την υπηρεσία γίνεται από το δημιουργό του διαλογικού πράκτορα. Ο έλεγχος και η δημιουργία του τμήματος της πλατφόρμας του api.ai γίνεται μέσω ενός διαχειριστικού συστήματος που παρέχει το ίδιο το api.ai.

Πράκτορες στο api.ai

Κάθε υλοποίηση στο api.ai είναι ένας πράκτορας (agent). Αυτό σημαίνει ότι οτιδήποτε συνομιλία (dialog) υλοποιηθεί για το συγκεκριμένο πράκτορα παραμένει στα συμφραζόμενα του πράκτορα αυτού. Ο πράκτορας, αποτελεί επίσης και ένα έργο (project). Μέσα σε αυτό ενσωματώνεται η εκπαίδευση του διαλογικού πράκτορα καθώς και ο ορισμός ονομαστικών οντοτήτων (entities), σκοπών χρήστη (intents), ενσωματώσεων (integrations) και εκπληρώσεων (fulfillment). Οι βασικές ρυθμίσεις του πράκτορα είναι:

1. ο ορισμός του ονόματος, π.χ. WeatherAgent
2. ο ορισμός της ζώνης ώρας του κατασκευαστή του πράκτορα, που χρησιμοποιείται για τον υπολογισμό του χρόνου
3. η γλώσσα που θα χρησιμοποιήσει κατά την επεξεργασία φυσικής γλώσσας.

Με την δημιουργία του πράκτορα όλες οι επιλογές εκτός της φυσικής γλώσσας είναι διαχειρίσιμες από την επιλογή agent στο διαχειριστικό σύστημα του api.ai.

Ο πράκτορας θεωρείται ως η μονάδα κατανόησης φυσικής γλώσσας (NLU) του έργου. Είναι μια υπηρεσία η οποία συνδέεται με τα επιμέρους τμήματα εισόδου, εξόδου και επεξεργασίας (απομακρυσμένων κλήσεων συστήματος μέσω API, εσωτερικές μέθοδοι υπολογισμού και ανεύρεσης δεδομένων από βάσεις δεδομένων κτλ). Ο ρόλος μιας τέτοιας μονάδας είναι η μετατροπή της φυσικής γλώσσας σε δεδομένα εκμεταλλεύσιμα (μηνύματα) από το πρόγραμμα του διαλογικού πράκτορα. Τα μηνύματα αυτά είναι ικανά να αναγνωρίσουν τον σκοπό του χρήστη από την φυσική γλώσσα και να εκτελέσουν μια λειτουργία, μια ενέργεια.

Σκοποί χρήστη στο api.ai

Οι σκοποί χρήστη στο api.ai ορίζονται ως μια χαρτογράφηση μεταξύ των ενεργειών που θέλει να επικοινωνήσει ο χρήστης και πώς μεταφράζονται στο λογισμικό. Στο api.ai ο σκοπός του χρήστη είναι το αρχικό σημείο του διαλόγου στο οποίο ορίζεται η πρόταση (ή το πρότυπο της) που θα πρέπει ο χρήστης να εκφράσει ώστε να συνεχίσει ο διάλογος.

Οι σκοποί χρήστη ορίζονται από το διαχειριστικό τμήμα του api.ai. Ένας σκοπός χρήστη χωρίζεται νοητά στις εξής ενότητες:

1. «Ο χρήστης λέει…» (User says)
2. Ενέργεια (action)
3. Απάντηση (response)
4. Context

Στην ενότητα «Ο χρήστης λέει…», ο δημιουργός του διαλογικού πράκτορα εισάγει την πρόταση που θα αναλυθεί από το πράκτορα. Η πρόταση αυτή μπορεί να περιέχει οντότητες, όπως π.χ.“Book a ticket to Los Angeles on Monday”. Οι οντότητες που ανιχνεύονται σε αυτή την περίπτωση είναι η φράση «Los Angeles» και η λέξη «Monday». Εάν ανιχνευτούν σωστά οι οντότητες, τότε θα παρουσιαστούν ως υποσημειώσεις στο

πίνακα παραμέτρων-οντοτήτων-τιμών ο οποίος εμφανίζεται κάτω από την πρόταση που εισάχθηκε παραπάνω (βλ. οντότητες για περισσότερες πληροφορίες).

Στην ενότητα «ενέργεια», ο χρήστης εισάγει μια χαρακτηριστική λέξη η οποία θα καθορίσει την ενέργεια που θα εκτελέσει ο συγκεκριμένος σκοπός. Για παράδειγμα, εάν θέλει να στείλει ένα e-mail ως τελικός σκοπός η λέξη που θα το καθορίσει μπορεί να είναι «send_mail». Μαζί με την λέξη κλειδί για την ενέργεια, ο χρήστης μπορεί να καθορίσει και ποιες πληροφορίες είναι σημαντικές για την χρήση της ενέργειας. Προγραμματιστικά η λέξη καθώς και οι πληροφορίες χρησιμοποιούνται για την κλήση μιας μεθόδου με παραμέτρους.

Στην ενότητα «απάντηση», ο χρήστης μπορεί να καθορίσει τις απαντήσεις που θα επιστρέφει ο διαλογικός πράκτορας από το api.ai. Οι τύποι απαντήσεων που μπορεί να επιστρέψει είναι είτε κείμενο (text response) ή εμπλουτισμένο κείμενο (rich text format) το οποίο μπορεί να περιέχει εικόνες και κλιπ ήχου ή υπερσυνδέσμους web. Στην ενότητα αυτή, χρησιμοποιείται μια ιδιότυπη γλώσσα για την παρουσίαση μεταβλητών που έρχονται από τα αποτελέσματα των υπηρεσιών που κλήθηκαν [48].

Η ενότητα «contexts» αναλαμβάνει την «μνήμη» της συνομιλίας. Συνήθως οι συνομιλίες στο api.ai, δεν διατηρούνται μετά το πέρας των προτάσεων που καθορίζονται στο σκοπό χρήστη, δηλαδή εφόσον η έκβαση της συνομιλίας είναι είτε η εκτέλεση μιας ενέργειας ή η διακοπή της συνομιλίας, όλες οι πληροφορίες και οι οντότητες που ανιχνεύτηκαν κατά την διάρκεια του διαλόγου διαγράφονται. Μέσω των contexts αυτά παραμένουν (η μπορούν να διαγραφούν πρώιμα), ώστε να μεταφερθούν σε άλλα τμήματα συνομιλιών, μέσα όμως στα πλαίσια του πράκτορα (agent) που έχουν δημιουργηθεί. Με αυτό τον τρόπο λειτουργεί το slot-filling στο api.ai.

Ο καθορισμός αυτός γίνεται μέσα στο διαχειριστικό. Για κάθε context, το api.ai, έχει δύο επιλογές την είσοδο των context, όπου ο νέος σκοπός χρήστη κληρονομεί τις οντότητες από προηγούμενο σκοπό χρήστη και την έξοδο των context που καθορίζει τι πληροφορίες θα μπορεί να στέλνει ο συγκεκριμένος σκοπός χρήστη προς τους υπόλοιπους σκοπούς χρήστη. Το context, καθορίζεται από ένα όνομα μεταβλητής και ένα χρονικό όριο παραμονής. Το όνομα χρησιμοποιείται για να μπορεί ο προγραμματιστής να αντλήσει τις πληροφορίες στο προγραμματιστικό κομμάτι του

api.ai, ενώ το χρονικό όριο παραμονής χρησιμοποιείται για να καθορίσει πόσες φορές μπορεί να χρησιμοποιηθεί ξανά στον ίδιο σκοπό χρήστη το context αυτό.

Πολλές φορές, θα χρειαστεί να συνδεθεί ο σκοπός του χρήστη με έναν άλλο. Το api.ai παρέχει αυτή τη δυνατότητα μέσω μιας επιλογής που λέγεται «follow-up intent». Για να χρησιμοποιηθεί αυτή η επιλογή θα πρέπει στο κείμενο που επιστρέφει το api.ai να ορίζεται μια ερώτηση, ώστε ο χρήστης να μπορεί να απαντήσει σε αυτή και να μεταφερθεί ένα αντικείμενο context στην επόμενη (follow-up) ερώτηση. Οι follow up ερωτήσεις είναι η διακλάδωση του διαλόγου μέσα στο api.ai.

Οντότητες στο api.ai

Οι οντότητες εκπροσωπεύουν έννοιες στο φυσικό λόγο και μεταφράζονται ως ένα εργαλείο εξαγωγής παραμέτρων μέσω επεξεργασίας φυσικής γλώσσας στο api.ai. Το api.ai ορίζει τριών ειδών οντότητες:

1. Οντότητες Συστήματος
2. Οντότητες Προγραμματιστή
3. Οντότητες Χρήστη

Οι οντότητες συστήματος είναι προκαθορισμένες οντότητες που παρέχει το api.ai, όπως μορφοποιημένες ημερομηνίες, μονάδες μέτρησης και τοποθεσίες ή μονάδες γεωγραφικού μήκους και πλάτους. Οι οντότητες προγραμματιστή είναι παράμετροι απλοί ή σύνθετοι (composite) οι οποίες ορίζονται μέσω της επιλογής «entities» στο διαχειριστικό τμήμα του api.ai. Σκοπό έχουν την επέκταση του συστήματος των οντοτήτων για συγκεκριμένες χρήσεις. Αν για παράδειγμα δημιουργηθεί ένας διαλογικός πράκτορας παραγγελιών φαγητού, οι οντότητες που θα δημιουργηθούν από το προγραμματιστή θα αφορούν είδη φαγητού ή συστατικά φαγητού. Οι οντότητες χρήστη είναι οντότητες που δημιουργούνται καθώς ο χρήστης συνομιλεί με τον διαλογικό πράκτορα. Για παράδειγμα, εάν ο χρήστης θέλει να φτιάξει μια προσωπική λίστα τραγουδιών μέσω διαλόγου, τότε η λίστα αυτή θα αποθηκευτεί σε μια οντότητα χρήστη. Οι οντότητες χρήστη σε σχέση με τις υπόλοιπες οντότητες είναι προσωποποιημένες (personalized) για κάθε χρήστη της εφαρμογής. Δημιουργούνται

από την εφαρμογή εφόσον ο διάλογος φτάσει στο σημείο που να χρειαστεί μια τέτοια οντότητα, όπως στο παράδειγμα της μουσικής λίστας.

Σε μία αποστολή πληροφοριών, στο βήμα που ο διαλογικός πράκτορας στέλνει την ενέργεια και τις παραμέτρους προς την εφαρμογή, οι οντότητες πλέον έχουν το ρόλο της παραμέτρου μιας μεθόδου. Το api.ai αναλαμβάνει να κάνει μια επαλήθευση τύπου και εφόσον οριστεί ως υποχρεωτική μια επαλήθευση ύπαρξης της τιμής της μεταβλητής/οντότητας πριν κάνει αποστολή των πληροφοριών προς την υπηρεσία του διαλογικού πράκτορα. Σε περίπτωση που κάποιος από τους ελέγχους αποτύχει μέσα στην συνομιλία, αυτομάτως το api.ai θα κάνει την ερώτηση ξανά ή μια από τις ορισμένες ερωτήσεις για εξόρυξη των πληροφοριών από το χρήστη έως ότου οι προϋποθέσεις πληρούνται.

Ενέργειες και παράμετροι στο api.ai

Στο api.ai, όταν ο διάλογος φτάσει σε ένα σημείο που θα χρειαστεί να υπολογίσει κάτι ή να ζητήσει πληροφορίες από μια απομακρυσμένη υπηρεσία, η οποία είναι δεμένη με ένα σκοπό χρήστη, π.χ. να φέρει μια λίστα αγώνων, εκτελείται μια ενέργεια. Οι ενέργειες μπορούν να περιέχουν παραμέτρους οι οποίες μετασχηματίζονται από ονομαστικές οντότητες.

Οι παράμετροι αναγνωρίζονται μέσα από την ενότητα «Ο χρήστης λέει…» των σκοπών χρήστη. Εφόσον αναγνωριστεί μια παράμετρος η οποία αντιστοιχεί είτε σε οντότητα συστήματος ή οντότητα προγραμματιστή, θα σημειωθεί αυτόματα και θα αφεθεί στην κρίση του χρήστη εάν είναι απαραίτητη η όχι. Οι παράμετροι μπορεί να είναι στατικοί δηλαδή να περιέχουν μια προκαθορισμένη τιμή π.χ. length 100, ή να είναι δυναμικοί και να αναγνωρίζονται από το διάλογο. Επίσης μπορούν να είναι σύνθεση από δύο η περισσότερες οντότητες συστήματος η προγραμματιστή.

Εκπληρώσεις (Fulfillments) στο api.ai

Όταν η έκβαση του διαλόγου φτάσει στο σημείο να κληθεί μια απομακρυσμένη μέθοδος, τότε ο διαλογικός πράκτορας εκτελεί εσωτερικά μια ενέργεια εκπλήρωσης που θα αναλάβει την σύνδεση του διαλογικού πράκτορα με μια υπηρεσία, μέσω

webhooks. Το webhook είναι ένα endpoint υπηρεσίας διαδικτύου το οποίο χρησιμοποιεί το api.ai για να στέλνει τα μηνύματα του προς μία εφαρμογή διαλογικού πράκτορα ή κατευθείαν σε μια υπηρεσία πληροφοριών. Επειδή οι περισσότερες υπηρεσίες δεν λειτουργούν με τα ίδια μηνύματα όπως το api.ai, οι κατασκευαστές της πλατφόρμας προτείνουν την δημιουργία μιας ενδιάμεσης υπηρεσίας η οποία αναλαμβάνει το ρόλο του κατανεμητή των μηνυμάτων από την υπηρεσία api.ai, σε μια ή περισσότερες υπηρεσίες που χρησιμοποιούν κάποιο API ή τοπικές μεθόδους. Παρόλα αυτά, μαζί με τις ενσωματώσεις παρέχει διέξοδο σε αρκετές υπηρεσίες και κανάλια διαλογικών πρακτόρων.

Για να μπορεί ένας διαλογικός πράκτορας να είναι ικανός να εκτελεί εκπληρώσεις χρειάζονται 3 προϋποθέσεις:

- Μέσα από το διαχειριστικό σύστημα να έχει ενεργοποιηθεί η ιδιότητα χρήσης webhooks και να έχει οριστεί το endpoint που θα στέλνει πληροφορίες (μηνύματα) το api.ai
- Μέσα από την επιλογή του σκοπού του χρήστη να έχουν ενεργοποιηθεί οι εκπληρώσεις
- Η ζητούμενη υπηρεσία να μπορεί να δέχεται και να στέλνει μηνύματα JSON με συγκεκριμένο payload (βλ. προγραμματιστικό οδηγό στο api.ai).

Πριν το api.ai κάνει κλήση της ενέργειας για την εκπλήρωση, θα πρέπει να έχουν ικανοποιηθεί όλοι οι ορισμοί των παραμέτρων/οντοτήτων. Σε αυτό το κομμάτι, τον έλεγχο τον έχει αποκλειστικά η υπηρεσία api.ai και μόλις ολοκληρωθεί επιτυχώς κάνει αποστολή του μηνύματος, το οποίο περιέχει σε μορφή JSON ένα μήνυμα με το όνομα του σκοπού του χρήστη, το όνομα της ενέργειας και τις παραμέτρους τις οποίες στέλνει ο διαλογικός πράκτορας προς την υπηρεσία-πύλη. Έπειτα αναμένει απάντηση μηνύματος με συγκεκριμένο JSON payload το οποίο περιέχει την επεξεργασμένη απάντηση μαζί με οποιαδήποτε πληροφορία χρειάζεται να σταλθεί πίσω στον χρήστη ως απόκριση. Με την αποστολή της απόκρισης, το τμήμα του διαλόγου που έχει σχέση με αυτό το σκοπό, τερματίζεται.

Ενσωματώσεις στο api.ai

Οι διαλογικοί πράκτορες μπορούν να χρησιμοποιηθούν για τον εμπλουτισμό άλλων εφαρμογών οι οποίες υποστηρίζουν μηνύματα ή διάλογους, π.χ. ένα chat client προτάσεων σε ένα e-shop ή το Facebook messenger. Σε περίπτωση που ο δημιουργός του διαλογικού πράκτορα θέλει να διαθέσει την εφαρμογή του σε αυτά τα κανάλια, οι δημιουργοί του api.ai, ετοίμασαν σαν πρότυπα μικρές εφαρμογές οι οποίες μπορούν να συνδέσουν το κώδικα της υπηρεσίας του πράκτορα με κάποιο κανάλι ή μέρος μιας εφαρμογής web.

3.6.2 Προγραμματιστικός οδηγός για το api.ai

Το api.ai έχει την δυνατότητα να λειτουργήσει χωρίς προγραμματιστικό κομμάτι. Όμως οι διαλογικοί πράκτορες οι οποίοι συγκροτούνται με αυτό τον τρόπο είναι απλοί πράκτορες και όλα τα δεδομένα τους βρίσκονται ορισμένα μέσα στις οντότητες του διαχειριστικού τμήματος.

Σε περίπτωση όμως που χρειαστεί να υλοποιηθεί μια εκπλήρωση για να συνδεθεί ο διαλογικός πράκτορας σε μια υπηρεσία που θα επιστρέφει δεδομένα π.χ. η υπηρεσία καιρού ή να συνδεθεί με ένα έτοιμο υπολογιστικό σύστημα για να αντλήσει δεδομένα (με κάποιο RESTful τρόπο ή μέσω API), τότε θα χρειαστεί η δημιουργία μιας μικροεφαρμογής η οποία θα πρέπει να λειτουργεί ως υπηρεσία διαδικτύου και να είναι δημοσιευμένη σε κάποιο προσβάσιμο τμήμα του web. Ο λόγος που απαιτείται, είναι διότι η σύνδεση με το api.ai γίνεται μέσω του webhook που αναφέρθηκε σε προηγούμενο τμήμα του κεφαλαίου. Μέσω του webhook γίνεται η επικοινωνία της εφαρμογής με το περιβάλλον του api.ai το οποίο περιέχει όλα τα απαραίτητα συστατικά για την υλοποίηση του διαλογικού πράκτορα.

Κώδικας3.16. Βασικά χρειαζόμαστε μια υπηρεσία η οποία θα αναλάβει την είσοδο των μηνυμάτων που πληκτρολογεί ο χρήστης και την έξοδο από το api.ai που παράγει το κείμενο φυσικής γλώσσας **προς** τον χρήστη. Προγραμματιστικά, αυτό μπορεί να είναι ένας εξυπηρετητής ο οποίος να δέχεται μηνύματα στο webhook από μια διεύθυνση web, π.χ. https://my_weather_service.com/webhook.

Μηνύματα api.ai

Η επικοινωνία του api.ai και της εφαρμογής γίνεται μέσω μηνυμάτων JSON τα οποία στέλνει η υπηρεσία προς τον εξυπηρετητή-πύλη. Ο εξυπηρετητής-πύλη αναλαμβάνει να επεξεργαστεί το μήνυμα και να εξάγει από αυτό τις παραμέτρους, το σκοπό του χρήστη και τα context. Στο παράδειγμα του καιρού, η ερώτηση από τον χρήστη:

Χρήστης: I want to know the weather

Θα στείλει με ένα αίτημα POST στο endpoint /webhook το εξής μήνυμα [49]:


```
{
  "id": "9f77aa1e-685f-4910-8c0c-fbab108224e7",
  "timestamp": "2017-05-20T16:23:22.92Z",
  "lang": "en",
  "result": {
    "source": "agent",
    "resolvedQuery": "I want to know the weather",
    "action": "simpleWeatherEnquiry",
    "actionIncomplete": true,
    "parameters": {
      "date-time": "",
      "location": ""
    },
    "contexts": [
      {
        "name": "13dd17a6-19c8-46b2-b7ad-f8e69bda9aac_id_dialog_context",
        "parameters": {
          "date-time.original": "",
          "location.original": "",
          "date-time": "",
          "location": ""
        },
        "lifespan": 2
      },
      {
        "name": "weatherquery_dialog_context",
        "parameters": {
          "date-time.original": "",
```

```
          "location.original": "",
          "date-time": "",
          "location": ""
        },
        "lifespan": 2
      },
      {
        "name": "weatherquery_dialog_params_location",
        "parameters": {
          "date-time.original": "",
          "location.original": "",
          "date-time": "",
          "location": ""
        },
        "lifespan": 1
      }
    ],
    "metadata": {
      "intentId": "13dd17a6-19c8-46b2-b7ad-f8e69bda9aac",
      "webhookUsed": "true",
      "webhookForSlotFillingUsed": "false",
      "intentName": "WeatherQuery"
    },
    "fulfillment": {
      "speech": "Where?",
      "messages": [
        {
          "type": 0,
          "speech": "Where?"
        }
      ]
    },
    "score": 1
  },
  "status": {
    "code": 200,
    "errorType": "success"
  },
  "sessionId": "76da81d7-846b-4c38-a00b-d1269d68ab62"
}
```

Το οποίο είναι ένα αντικείμενο JSON. Τα κυριότερα σημεία του μηνύματος αυτού είναι το id και η δομή result. Το sessionid δημιουργείται αυτόματα από το api.ai και αφορά την συνεδρία στην οποία στέλνονται τα δεδομένα. Το result είναι το αντικείμενο το οποίο στέλνει το api.ai στην εφαρμογή του χρήστη και περιέχει:

1. Την πηγή του μηνύματος (source)
   Από ποιο πράκτορα έγινε η αποστολή του μηνύματος
2. Το ερώτημα (resolvedQuery) το οποίο ενεργοποίησε την αποστολή
3. Η ετικέτα της ενέργειας (action) που χρησιμοποιείται.
4. Αν η ενέργεια έχει ολοκληρωθεί (actionIncomplete)
   Σε περίπτωση που η τιμή αυτή είναι true, σημαίνει ότι το μήνυμα αυτό δεν στάλθηκε στην εφαρμογή του χρήστη, αλλά στην υπηρεσία του api.ai και το σύστημα είναι σε κατάσταση slot-filling.
5. Οι παράμετροι του διαλόγου (parameters)
   Απλές τιμές ή αντικείμενα τα οποία περιέχουν τις τιμές από τις αναγνωρισμένες οντότητες. Στην κατάσταση slot-filling αυτές οι μεταβλητές είναι κενές αφού αρχίζει η διερευνητική διαδικασία για να αποκτήσουν τιμή.
6. Contexts
   Στο api.ai μπορεί να υπάρχουν περισσότερα από ένα contexts ανάλογα από πού συνδέθηκε ο συγκεκριμένος σκοπός χρήστη. Πολλές φορές το ίδιο το api.ai θα δημιουργήσει context αντικείμενα για την εσωτερική λειτουργία του. Κάθε context αντικείμενο περιέχει το όνομα (μια αναγνωριστική μοναδιαία τιμή), τις παραμέτρους (τις οντότητες που αναγνώρισε) και το πόσο χρόνο αυτό το μήνυμα θα «ζήσει» (lifespan) σε κάθε νέα αίτηση /POST του χρήστη.
7. Μεταδεδομένα (metadata)
   Τα μεταδεδομένα που εμφανίζονται αφορούν την λειτουργία του σκοπού του χρήστη και την χρήση του webhook στην διαδικασία εκπλήρωσης (fulfillment).
8. Εκπλήρωση (fulfillment)
   Εδώ η εκπλήρωση αφορά το επόμενο βήμα ερωτήματος από την υπηρεσία ή την εφαρμογή του χρήστη. Περιέχει το κείμενο που θα εμφανιστεί μετά την απάντηση του χρήστη και αφορά το «Where?» που ρωτά τον χρήστη ποια τοποθεσία να εισάγει για να μάθει τον καιρό.

9. Κατάσταση http (status)

   Η κατάσταση αφορά τα μηνύματα κατάστασης αιτημάτων http τα οποία χρησιμοποιούνται για επαλήθευση σε μια REST υπηρεσία [50].

Η τεκμηρίωση του api.ai περιέχει όλες τις πληροφορίες σχετικά με τις παραμέτρους οι οποίες στέλνονται με το μήνυμα JSON [49].

Αντίστοιχα, ένα μήνυμα λήψης από την εφαρμογή χρήστη/υπηρεσία webhook [51] πρέπει να περιέχει τουλάχιστον τα εξής στοιχεία:

```
{
    "speech": `The weather in ${location} would be ${temperature} degrees.`,
    "displayText": `The weather in ${location} would be ${temperature} degrees.`,
    "source": "SimpleWeatherService"
}
```

Κώδικας 3.17 μήνυμα αποστολής από την εφαρμογή προς το api.ai

Τα παραπάνω στοιχεία είναι από κώδικα javascript και παράγουν το κείμενο φυσικής γλώσσας το οποίο είναι η απάντηση στο χρήστη. Τα επιμέρους πεδία και οι λειτουργίες τους είναι:

Πεδίο	**Λειτουργία**
speech	Παράγει την πρόταση/απάντηση στο χρήστη, καθώς και το πρότυπο για την δημιουργία φωνητικής απάντησης εάν υποστηρίζεται από την εφαρμογή πελάτη.
displayText	Λεπτομερές κείμενο σε περίπτωση που χρειαστεί επεξήγηση σε κάποια κινητή συσκευή
source	Η πηγή του μηνύματος, δηλαδή από ποια εφαρμογή ήρθε το μήνυμα αυτό

Πίνακας 3.1 επεξήγηση payload που στέλνει η εφαρμογή χρήστη στο api.ai

Ένα πλήρες μήνυμα-απάντηση της εφαρμογής είναι όπως το παρακάτω:

```
{
  "id": "92f87a95-8808-434c-adfb-e7e553893659",
  "timestamp": "2017-05-20T16:47:46.688Z",
  "lang": "en",
  "result": {
    "source": "agent",
    "resolvedQuery": "now",
```

```
    "action": "simpleWeatherEnquiry",
    "actionIncomplete": false,
    "parameters": {
      "date-time": "19:47:46",
      "location": {
        "city": "Athens"
      }
    },
    "contexts": [],
    "metadata": {
      "intentId": "13dd17a6-19c8-46b2-b7ad-f8e69bda9aac",
      "webhookUsed": "true",
      "webhookForSlotFillingUsed": "false",
      "webhookResponseTime": 2836,
      "intentName": "WeatherQuery"
    },
    "fulfillment": {
      "speech": "The weather in Athens would be 20.04 degrees.",
      "source": "SimpleWeatherService",
      "displayText": "The weather in Athens would be 20.04 degrees.",
      "messages": [
        {
          "type": 0,
          "speech": "The weather in Athens would be 20.04 degrees."
        }
      ]
    },
    "score": 1
  },
  "status": {
    "code": 200,
    "errorType": "success"
  },
  "sessionId": "76da81d7-846b-4c38-a00b-d1269d68ab62"
}
```


Κώδικας 3.18 μήνυμα response επεξεργασμένο από το api.ai

Παρατηρούμε ότι τα ελάχιστα πεδία που χρησιμοποιεί το μήνυμα λήψης, ενσωματώνονται ως τμήμα του result/fulfillment, πλήρως επεξεργασμένα:

```
    "fulfillment": {
      "speech": "The weather in Athens would be 20.04 degrees.",
      "source": "SimpleWeatherService",
      "displayText": "The weather in Athens would be 20.04 degrees.",
      "messages": [
        {
          "type": 0,
          "speech": "The weather in Athens would be 20.04 degrees."
        }
      ]
    },
```

Κώδικας 3.19 Τμήμα του μηνύματος επιστροφής στο api.ai

Προγραμματισμός της υπηρεσίας / εφαρμογής χρήστη

Για να φτιαχτεί μια υπηρεσία η οποία θα ικανοποιεί την εκπλήρωση του σκοπού του χρήστη, χρειάζεται η γλώσσα προγραμματισμού να μπορεί να προγραμματίζει εξυπηρετητές οι οποίοι θα δημιουργούν ένα endpoint το οποίο στην συγκεκριμένη περίπτωση έχει ονομαστεί /webhook. Επιπλέον θα πρέπει να μπορεί να επεξεργάζεται μηνύματα JSON (βλ. προηγούμενη ενότητα) και να εξάγει πληροφορίες από αυτά.

Για την υλοποίηση ενός προγράμματος/υπηρεσίας που θα ικανοποιεί τις παραπάνω προϋποθεσεις χρησιμοποιήθηκε η javascript και η βιβλιοθήκη προγραμματισμού node.js καθώς και η επέκταση της expressjs [52].

```
//
//  Open university of Cyprus
//  Information and Communication Systems
//  Panagiotis Chalatsakos (c) 2017
//  Student Id: 11300837
//
var express = require('express');
var bodyParser = require('body-parser');
var axios = require('axios');

var app = express();
app.use(bodyParser.json());

app.post('/webhook', (req, res) => {
```

```
    console.log(JSON.stringify(req.body, undefined, 2));

    var location = req.body.result.parameters.location.city;
    console.log(location);

    // url encode the location
    var encodedAddress = encodeURIComponent(location);
    var geocodeURL =
`https://maps.googleapis.com/maps/api/geocode/json?address=${encodedAddress}`;

    // first do a reverse geolocation from the location given
    return axios.get(geocodeURL).then((response) => {
        if (response.data.status === 'ZERO_RESULTS') {
            throw new Error('Unable to find that address');
        }
        var lat = response.data.results[0].geometry.location.lat;
        var lng = response.data.results[0].geometry.location.lng;

        // then do a weather forecast search to forecast.io
        var weatherURL =
`https://api.darksky.net/forecast/39b78cd34c557b7ac589472a5c6a9c6b/${lat},${lng}?u
nits=si`;
        return axios.get(weatherURL);
    }).then((response) => {
        // if the chained Promise succeeds then set the temperature to the context
object
        var temperature = response.data.currently.temperature;
        var apparentTemperature = response.data.currently.apparentTemperature;

        // Generate context and content in natural language
        res.send({
            "speech": `The weather in ${location} would be ${temperature}
degrees.`,
            "displayText": `The weather in ${location} would be ${temperature}
degrees.`,
            "source": "SimpleWeatherService"
        });
    }).catch((e) => {
        // Error check for both google reverse code and forecast.io
        if (e.code === 'ENOTFOUND') {
            console.log('Unable to connect to API server');
        } else {
```

```
            console.log(e.message);
        }
        res.status(404).send(e);
    });
});
app.listen(9000, () => {
    console.log(`Started on Port 9000.`);
});
module.exports = { app };
```

Κώδικας 3.20 Υλοποίηση υπηρεσίας καιρού για το api.ai

Στο συγκεκριμένο παράδειγμα, ο διάλογος γίνεται μέσω της δοκιμαστικής κονσόλας του api.ai, η οποία βρίσκεται στο δεξιό τμήμα του διαχειριστικού τμήματος του api.ai.

Μετά την εισαγωγή των βιβλιοθηκών express, body-parser και axios γίνεται η αρχικοποίηση της εργαλειοθήκης expressjs στο αντικείμενο app και η ανάθεση του middleware body-parser σε αυτό. Το middleware body-parser διαβάζει ένα μήνυμα JSON και μπορεί να το μετατρέψει σε αντικείμενα και μεταβλητές javascript.

Για την ενεργοποίηση του endpoint /webhook, ορίζουμε στο αντικείμενο app μια μέθοδο η οποία δηλώνει ότι θα επεξεργάζεται τα μηνύματα POST (`app.post()`) με παραμέτρους το endpoint /webhook και μια callback μέθοδο η οποία έχει ως παραμέτρους τα αντικείμενα request και response. Στο αντικείμενο request η εφαρμογή «ακούει» το μήνυμα που στέλνει το api.ai και μέσω της βιβλιοθήκης body-parser το επεξεργάζεται σε αντικείμενο javascript. Με το αντικείμενο response, η εφαρμογή μπορεί να στείλει πίσω στο api.ai πληροφορίες οι οποίες ορίζονται μέσα στην μέθοδο callback.

Για να λάβουμε την τιμή της οντότητας location από το api.ai το μόνο που χρειάζεται είναι να ορίσουμε μια μεταβλητή location η οποία θα διαβάσει από το request αντικείμενο την περιοχή. Γνωρίζουμε ότι το request αντικείμενο περιέχει ένα πεδίο result και μέσα σε αυτό υπάρχει ένα πεδίο parameters το οποίο περιέχει τις μεταβλητές οντότητες. Έτσι για να ορίσουμε την πληροφορία αυτή στο κώδικα χρησιμοποιούμε την εξής εντολή:

```
var location = req.body.result.parameters.location.city;
```

Κώδικας 3.21 απόδοση τιμής location στην μεταβλητή location.

Η μεταβλητή location από τη τεκμηρίωση του api.ai είναι μια σύνθετη μεταβλητή η οποία περιέχει μέσα της το πεδίο city όπου λέει την πόλη που ανιχνεύτηκε ως ονομαστική οντότητα τοποθεσίας από την μονάδα κατανόησης φυσικής γλώσσας του api.ai κατά την ερώτηση «where?».

Έχοντας την πληροφορία αυτή μπορούμε να καλέσουμε ασύγχρονα τις υπηρεσίες geocoding και forecast.io όπως εργαστήκαμε στο κεφάλαιο με το wit.ai. Η διαφορά εδώ είναι στην επιστροφή της πληροφορίας στο api.ai:

```
// if the chained Promise succeeds then set the temperature to the context object
var temperature = response.data.currently.temperature;
var apparentTemperature = response.data.currently.apparentTemperature;

// Generate context and content in natural language
res.send({
"speech": `The weather in ${location} would be ${temperature} degrees.`,
"displayText": `The weather in ${location} would be ${temperature} degrees.`,
"source": "SimpleWeatherService"
});
```

Κώδικας 3.22 Επιστροφή πληροφορίας καιρού στο api.ai

Επιστρέφουμε το μήνυμα στην υπηρεσία api.ai μέσω της μεθόδου send() η οποία υπάρχει στο αντικείμενο response της μεθόδου callback. Σε περίπτωση λάθους, χρησιμοποιούμε πάλι την μέθοδο send(), ορίζοντας όμως ένα μηνυμα λάθους http:

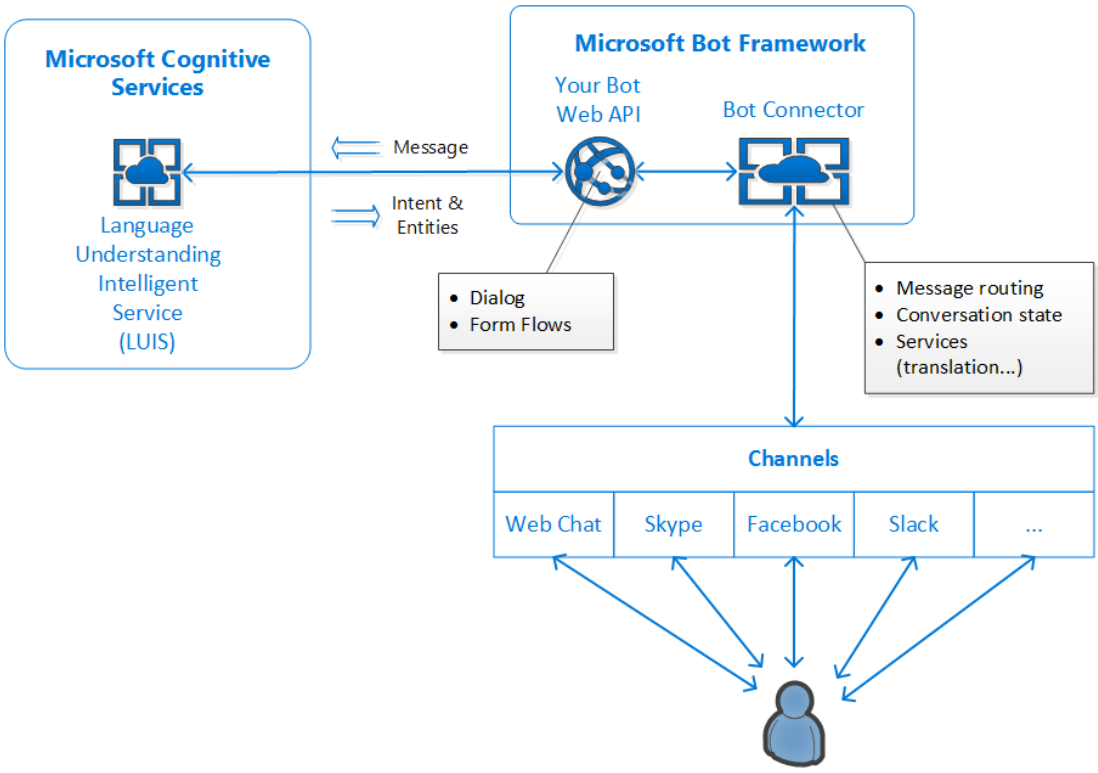
```
res.status(400).send(e)
```

Κώδικας 3.23 Αποστολή μηνύματος λάθους από την εφαρμογή χρήστη/υπηρεσία.

3.7 Bot Framework

Το Microsoft Bot Framework είναι μια σουίτα/πλατφόρμα εφαρμογών που χρησιμοποιείται για την δημιουργία και ανάπτυξη διαλογικών bots. Η σουίτα αποτελείται από το Bot Builder SDK, Bot Connector, το Developer Portal και το Bot Directory. Παρέχει επίσης και ένα προσομοιωτή για την τοπική ανάπτυξη του διαλογικού πράκτορα. (Microsoft Cognitive Services, 2017). Η γενική υλοποίηση των

υπηρεσιών που παρέχει ακολουθούν το πρότυπο της υπολογιστικής νέφους και των μικρουπηρεσιών. Παρατηρώντας το σχήμα, κάθε τμήμα της υλοποίησης αποτελεί μια υπηρεσία απομακρυσμένη η οποία στηρίζεται σε επιμέρους συστατικά, κυρίως στο νέφος Azure.



Εικόνα 3.6 Υλοποίηση υπηρεσιών μέσω Bot Framework

Το Microsoft bot framework είναι πιο "ανοιχτό" σε σχέση με τα υπόλοιπα λογισμικά που εξετάζουμε, ως προς την επεκτασιμότητα. Παράλληλα χρησιμοποιεί ένα μοντέλο πολλαπλών υπηρεσιών REST οι οποίες επικοινωνούν με το API, αλλά παράλληλα επιτρέπει την αντικατάσταση τους από υπηρεσίες τρίτων συμβατών, ειδικά στην περίπτωση χρήσης της μηχανής κατανόησης φυσικής γλώσσας.

Παρέχει διάφορα είδη υλοποίησης, όπως για παράδειγμα την χρήση εξωτερικής μηχανής κατανόησης φυσικής γλώσσας [53] ή χρήση χειρωνακτικών τρόπων εύρεσης των σκοπών του χρήστη μέσω κανονικών εκφράσεων. Για την υλοποίηση μπορεί κάποιος να χρησιμοποιήσει είτε την κονσόλα (τερματικό / command prompt) ή να δημιουργήσει το πρόγραμμα σαν εφαρμογή διαδικτύου. Αυτό βοηθά στην αντιμετώπιση προβλημάτων που τυχόν συναντήσει σε πραγματικές υλοποιήσεις όταν εγκατασταθεί σε ενα εξυπηρετητή η εφαρμογή.

3.7.1 Σχεδίαση διαλογικού πράκτορα με το Microsoft bot framework

Η σχεδίαση ενός διαλογικού πράκτορα με το Microsoft bot framework αρχίζει με την σχεδίαση των υπηρεσιών που θα χρησιμοποιήσει. Το Microsoft bot framework χρησιμοποιεί την λογική των αυτόνομων υπηρεσιών, οι οποίες συνδέονται μεταξύ τους με μηνύματα από και προς των υπηρεσιών αυτών. Για να μπορέσει μια εφαρμογή σχεδιασμένη με το bot framework να χαρακτηριστεί ως διαλογικός πράκτορας, πρέπει να έχει σύνδεση με τρείς βασικές υπηρεσίες:

- Μια υπηρεσία μηχανής κατανόησης φυσικής γλώσσας
- Την υπηρεσία bot connector
- Την κυρίως εφαρμογή διαλογικού πράκτορα
- (προαιρετικά) ένα κανάλι διαλογικού πράκτορα

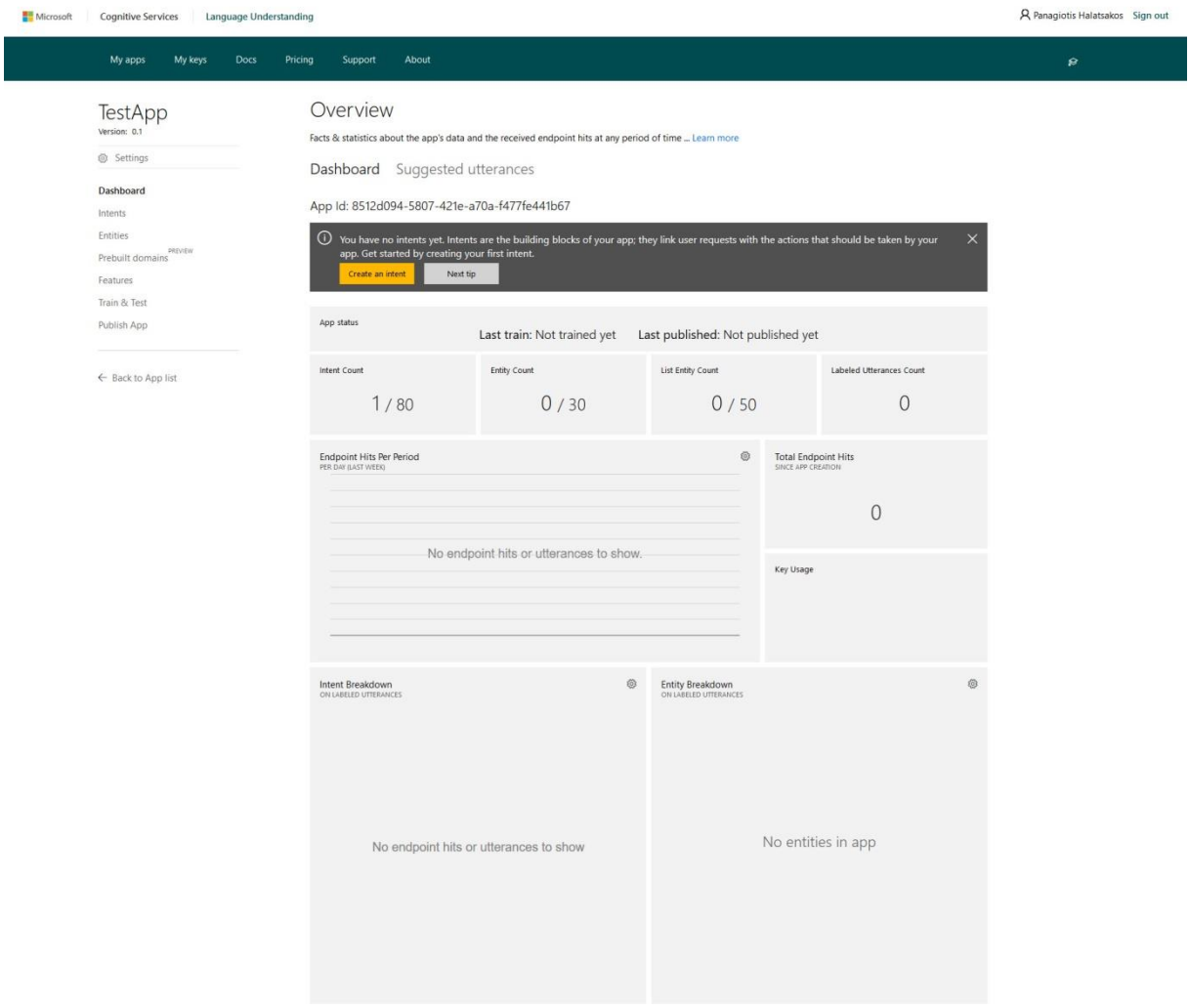
Μηχανή κατανόησης φυσικής γλώσσας – LUIS

Το κύριο συστατικό για την χρήση του framework του διαλογικού πράκτορα είναι η σύνδεση του με μια μηχανή κατανόησης φυσικής γλώσσας. Η προτεινόμενη μηχανή είναι η μηχανή LUIS που προέρχεται από τα αρχικά "Language Understanding Intelligent Service". Πρόκειται για μια υπηρεσία της Microsoft Cognitive Services η οποία αναλαμβάνει να δώσει στον χρήστη που θέλει να δημιουργήσει ένα διαλογικό πράκτορα, εργαλεία για την δημιουργία **σκοπών χρήστη, ενεργειών, οντοτήτων, μοντέλων κατανόησης γλώσσας** και εκπαίδευσης των μοντέλων αυτών.

Παράλληλα επιτρέπει την πρόσβαση των στοιχείων του έργου μέσω RESTFul υπηρεσιών ώστε να μπορεί ο προγραμματιστής να υλοποιήσει το δικό του σύστημα στην δική του γλώσσα. Υπάρχουν έτοιμα διαθέσιμα πακέτα για γλώσσες προγραμματισμού τόσο σε συμβατικούς υπολογιστές όσο και κινητές συσκευές. Τα πακέτα αυτά είναι για τις γλώσσες C#, Node.JS, Android Java και Python.

Διαθέτει επίσης μοντέλα κατανόησης γλώσσας [54]. Αυτά μπορούν να χρησιμοποιηθούν είτε σε έτοιμες εφαρμογές είτε για βάση για να χτιστούν πολύπλοκες εφαρμογές που χρειάζονται έτοιμα, υλοποιημένα και επιβεβαιωμένα εμπορικά κομμάτια. Ενα από αυτά τα μοντέλα κατανόησης γλώσσας χρησιμοποιείται στο

λογισμικό Cortana [55], το ψηφιακό βοηθό της Microsoft για κινητές και desktop εφαρμογές.



Εικόνα 3.7 Το διαχειριστικό της NLU LUIS

Στην υλοποίηση του παραδείγματος, θα χρησιμοποιήσουμε αυτό το μοντέλο λόγω της αξιοπιστίας που έχει. Θα εξετάσουμε όμως τα επιμέρους στοιχεία του LUIS μιας που είναι τμήματα της σύγκρισης των frameworks που θα ακολουθήσει.

Microsoft Bot Connector

Το Microsoft Bot Connector είναι ο συνδετικός κρίκος που αναλαμβάνει την διευθέτηση του διαλογικού συστήματος και την επικοινωνία τόσο της μηχανής κατανόησης φυσικής γλώσσας όσο και της διοχέτευσης των μηνυμάτων (πληροφοριών), πρός το χρήστη (τοπική υλοποίηση), ή σε κάποιο κανάλι (slack, skype, web site) διαλογικού πράκτορα.

Κυρίως εφαρμογή διαλογικού πράκτορα

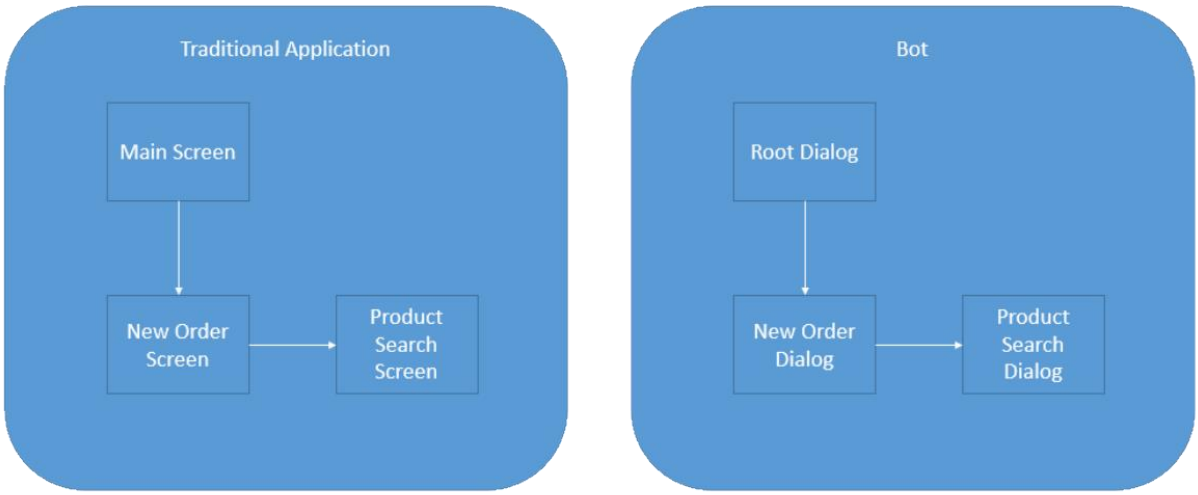
Η κυρίως εφαρμογή της υλοποίησης του διαλογικού πράκτορα με το bot framework, είναι μια εφαρμογή ή υπηρεσία η οποία χρησιμοποιείται για να καθορίζει την ροή του διαλόγου και να στέλνει ή να λαμβάνει αντίστοιχα μηνύματα από το χρήστη, τα οποία θα στέλνει στο bot connector. Τα μηνύματα αυτά μεταφέρονται στην μηχανή κατανόησης (κατανόηση NLP) για μετέπειτα επεξεργασία. Η επεξεργασία θα επιστρέψει πίσω στην εφαρμογή το σκοπό του χρήστη (intent) και τις οντότητες (entities) που αναγνωρίστηκαν ως παράμετροι.

Η εφαρμογή αναλαμβάνει, εφόσον έχει τις απαραίτητες πληροφορίες, να κάνει κλήση σε απομακρυσμένες υπηρεσίες και σε εσωτερικές μεθόδους ώστε να παραχθεί το κείμενο της φυσικής γλώσσας που θα επιστραφεί στο χρήστη ως απάντηση. Το μήνυμα που περιέχει το κείμενο στέλνεται στο bot connector και από εκεί στο κατάλληλο δίαυλο επικοινωνίας.

Η ροή μιας εφαρμογής διαλογικού βοηθού στο bot framework μπορεί να χαρακτηριστεί ισάξια όπως μια παραδοσιακή εφαρμογή. Σε μια παραδοσιακή εφαρμογή έχουμε μια κεντρική οθόνη στην οποία υπάρχουν διάφορες ενέργειες, όπως π.χ. η δημιουργία μιας νέας παραγγελίας. Αν για παράδειγμα έχουμε την δημιουργία μιας νέας παραγγελίας σε ένα ηλεκτρονικό κατάστημα, τότε:

Σε μία παραδοσιακή εφαρμογή, η οθόνη της νέας παραγγελίας θα παραμείνει ενεργή, μέχρι να ολοκληρωθεί μια ενέργεια, όπως μια αναζήτηση προϊόντων. Μέχρι να ολοκληρωθεί η αναζήτηση των προϊόντων, μπορεί να υπάρχουν ενδιάμεσα στάδια.

Αντίστοιχα, σε ένα διαλογικό πράκτορα με το bot framework, όλα έχουν σχέση με τον αρχικό διάλογο (root dialog). Ο αρχικός διάλογος, επικαλείται τον διάλογο της νέας παραγγελίας (new order dialog). Από αυτή τη στιγμή, ο σκοπός του χρήστη είναι η νέα παραγγελία, οπότε ο διάλογος αυτός αντικαθιστά τον αρχικό διάλογο και παραμένει μέχρι να τελειώσει η διαδικασία ή να αρχίσει ένας νέος διάλογος για κάποιο άλλο σκοπό χρήστη. Όταν τελειώσει η διαδικασία, ο διάλογος επιστρέφει πίσω στον αρχικό διάλογο.



Εικόνα 3.8 παραλληλισμός εφαρμογής και διαλόγου [56]

3.7.2 Επεξεργασία φυσικής γλώσσας με το bot framework

Το bot framework για την επεξεργασία και κατανόηση της φυσικής γλώσσας δεν χρησιμοποιεί κάποια βιβλιοθήκη ενσωματωμένη με το υπόλοιπο σύστημα σε σχέση με άλλα frameworks. Η αρχιτεκτονική του είναι σχεδιασμένη έτσι ώστε να μπορεί να χρησιμοποιήσει μια εξωτερική υπηρεσία για μονάδα κατανόησης της φυσικής γλώσσας, όπου με τα κατάλληλα μηνύματα από και προς αυτή την υπηρεσία μπορεί να επεξεργαστεί το διάλογο που εκτελείται μέσω του διαλογικού πράκτορα. Προτείνει την χρήση της υπηρεσίας LUIS μιας που είναι βέλτιστα ρυθμισμένη για αυτό το σκοπό, χωρίς όμως να αποκλείσει άλλες υπηρεσίες οι οποίες εφόσον έχουν το ίδιο σύστημα μηνυμάτων από και προς το bot connector να μπορούν να χρησιμοποιηθούν ως υποκατάστατο. Η υπηρεσία LUIS παρέχει ένα διαχειριστικό κομμάτι μέσω web, στο οποίο υπάρχουν μέσα όλες οι επιλογές ελέγχου και εκπαίδευσης του τμήματος φυσικής γλώσσας που χρειάζεται ο διαλογικός πράκτορας [57].

Εφαρμογή LUIS

Μια εφαρμογή LUIS είναι το έργο το οποίο περιλαμβάνει τα χαρακτηριστικά του διαλογικού πράκτορα, δηλαδή τις οντότητες, τους σκοπούς του χρήστη, την εκπαίδευση του καθώς και μοναδικές παροχές της Microsoft σχετικά με τους διαλογικούς πράκτορες όπως προκαθορισμένους τομείς (domains) και άλλες δυνατότητες (features).

Σκοπός χρήστη στο LUIS

Ένας διάλογος στο bot framework γίνεται με την αναγνώριση ενός σκοπού χρήστη (intent) από το LUIS. Αυτό συμβαίνει όταν αναγνωριστεί μια έκφραση (utterance) ή μια παρεμφερή έκφραση από το LUIS, θα ανατρέξει στην αντίστοιχη ετικέτα (label) του σκοπού του χρήστη. Πληκτρολογώντας μια φράση στο πεδίο utterance, το LUIS θα δημιουργήσει μια νέα εγγραφή σκοπού χρήστη. Για παράδειγμα, μια φράση μπορεί να είναι «I want to know the weather in Paris, now».

Οντότητες στο LUIS

Με την δημιουργία του σκοπού χρήστη, στην ίδια οθόνη του διαχειριστικού, ο προγραμματιστής έχει την δυνατότητα να ορίσει πάνω στην έκφραση που πληκτρολόγησε τις σχετικές οντότητες. Έτσι για παράδειγμα στη φράση «I want to know the weather in Paris, now» μπορεί να αναγνωρίσει χειρωνακτικά ότι το Paris αντιστοιχεί σε μια οντότητα τοποθεσίας και το now σε μια οντότητα χρόνου.

Αντίστοιχα, στην επιλογή «οντότητες» (Entities) από το διαχειριστικό, ο προγραμματιστής μπορεί να ελέγξει που χρησιμοποιείται η οντότητα αυτή (σε ποια έκφραση) και τον τύπο της οντότητας καθώς και να τη μετατρέψει σε μια πιο σύνθετη οντότητα.

Εκπαίδευση διαλογικού πράκτορα

Η σημασία των οντοτήτων φαίνεται κατά την εκπαίδευση του διαλογικού πράκτορα. Όταν μέσα από το διαχειριστικό στην επιλογή «εκπαίδευσης» (Train & Test), γίνει η εκπαίδευση των σκοπών του χρήστη, κάθε φορά που εισάγονται περισσότερες λεκτικές

πληροφορίες και οντότητες που μπορούν να καθορίσουν καλύτερα το σκοπό του χρήστη, οι πιθανότητες να ταιριάζει η έκφραση αυτή στην ετικέτα του σκοπού είναι μεγαλύτερες. Έτσι για παράδειγμα αν εισάγουμε την πρόταση «I want to know the weather», η πιθανότητα να ταιριάζει η έκφραση με την ετικέτα “weatherForecast” είναι 0.75 (75%) σε σχέση με μια πρόταση που περιέχει μέσα και οντότητες. Οι πιθανότητες αυτές μειώνονται όταν υπάρχουν περισσότεροι από έναν σκοπό χρήστη.

Προκαθορισμένα πεδία (domains)

Τα προκαθορισμένα πεδία είναι έτοιμες σχεδόν εφαρμογές διαλογικών βοηθών οι οποίες μπορεί να χρειάζονται στην καθημερινή ζωή των χρηστών. Περιέχουν δεδομένα, σκοπούς χρήστη και οντότητες οι οποίες είναι καθορισμένες για ένα συγκεκριμένο τομέα, π.χ. για Ψυχαγωγία, για καιρό, για μηχανική μετάφραση, για κράτηση σε ένα εστιατόριο και άλλες. Βασίζονται στην προγραμματιστική αρχή «Don’t Repeat Yourself» [58] και στην αρχή της Microsoft «η εμπειρία του χρήστη πρώτα» (User experience first) .

3.7.3 Προγραμματιστικός οδηγός για το bot framework

Το bot framework είναι κυρίως προσανατολισμένο σε προγραμματιστική υλοποίηση παρά σε χρήση ενός διαχειριστικού συστήματος. Αυτό συμβαίνει λόγω του ότι η Microsoft επιδιώκει την χρήση υπηρεσιών οι οποίες συνθέτουν μια λειτουργία, όπως τον διαλογικό πράκτορα. Τόσο για λόγους αποκλιμάκωσης όσο και για λόγους αδιάλειπτης λειτουργίας.

Bot Connector

Στο προγραμματιστικό επίπεδο, ο bot connector είναι μια υπηρεσία η οποία διαχειρίζεται τα μηνύματα μεταξύ των επιμέρους υπηρεσιών του διαλογικού πράκτορα. Επίσης οργανώνει την επικοινωνία των καναλιών του διαλογικού πράκτορα (π.χ. Facebook, skype, slack). Όλα αυτά γίνονται μέσω μηνυμάτων http POST.

Χρησιμοποιώντας την javascript, το SDK του bot connector περιέχει δύο κλάσεις, την UniversalBot και την ChatConnector. Η κλάση UniversalBot διαχειρίζεται τις συνομιλίες μεταξύ του χρήστη και του διαλογικού πράκτορα, ενώ η κλάση ChatConnector συνδέει το διαλογικό πράκτορα με την υπηρεσία του Bot Connector.

```
var builder = require('botbuilder');
// Init Bot Connector
var connector = new builder.ChatConnector({
    appId: process.env.MICROSOFT_APP_ID,
    appPassword: process.env.MICROSOFT_APP_PASSWORD
});

// Init Bot
var bot = new builder.UniversalBot(connector);
server.post('/api/messages', connector.listen());
```

Κώδικας 3.24 Αρχικοποίηση universalbot και connector

Μηνύματα χρήστη

Τα μηνύματα χρήστη αποτελούνται από κείμενο το οποίο ανάλογα με την χρήση του μπορεί να οριστεί διαφορετικά. Έτσι έχουμε κείμενο το οποίο χρησιμοποιείται για την γραπτή επικοινωνία, για την φωνητική (μετατροπή κειμένου σε φωνής ή φωνής σε κείμενο) και για την παρουσίαση μέσω εμπλουτισμού του (rich text format). Ένα αντικείμενο UniversalBot αναλαμβάνει να στείλει τα μηνύματα μέσω της συνεδρίας που δημιουργεί στο Bot Connector.

Διάλογοι στο bot framework

Οι διάλογοι στο bot framework είναι ένα είδος σωρού. Εφόσον οριστεί το αντικείμενο UniversalBot, κάθε διάλογος ακολουθεί μια διαδρομή αρχίζοντας από την αρχική διαδρομή η οποία δηλώνεται με το σύμβολο slash (/). Κάθε επιπλέον διαδρομή συνεχίζεται από αυτό το σύμβολο. Π.χ. για να μάθουμε το όνομα ενός χρήστη μπορούμε να χρησιμοποιήσουμε μια διαδρομή /profile για να δηλώσουμε το όνομα του διαλόγου αυτού. Η συνεδρία που περιέχει το UniversalBot καθορίζει τη ροή της συνομιλίας. Π.χ. αν το πρόγραμμα εξετάσει ότι δεν υπάρχει η παράμετρος ονόματος στην αρχική συνεδρία τότε μεταβαίνει στο διάλογο /profile για να «μάθει» το όνομα και να το εισάγει στην συνεδρία και έπειτα ξαναγυρίζει στον αρχικό διάλογο. Η λογική αυτή ονομάζεται λογική waterfall [59].

Ένα παράδειγμα κώδικα διαλόγου είναι:

```
bot.dialog('/', [
    function (session) {
        session.beginDialog('/askName');
    },
    function (session, results) {
        session.send('Hello %s!', results.response);
    }
]);
bot.dialog('/askName', [
    function (session) {
        builder.Prompts.text(session, 'Hi! What is your name?');
    },
    function (session, results) {
        session.endDialogWithResult(results);
    }
]);
```

Κώδικας 3.25 Παράδειγμα 2 διαλόγων στο bot framework [60].

Κώδικας3.26 παρατηρούμε ότι κάθε διάλογος αποτελείται από ένα πίνακα μεθόδων. Σε κάθε μέθοδο υπάρχει η συνεδρία η οποία αποτελεί το context του διαλόγου για το bot framework. Το αντικείμενο builder αποτελεί μέρος του botbuilder και χρησιμοποιείται εδώ για να στείλει ένα μήνυμα προς τον χρήστη το οποίο περιμένει την απάντηση του. Ειδάλλως χρησιμοποιείται το αντικείμενο session.send() το οποίο στέλνει μόνο ενημερωτικό μήνυμα προς το χρήστη.

Διακοπές στο bot framework

Μερικές φορές ο έλεγχος του διαλόγου θα χρειαστεί να τερματιστεί νωρίς ή να χρειαστούν διακοπές, όπως για την βοήθεια του χρήστη ή να επανεκκινηθεί ο διάλογος επειδή κάποιο στοιχείο του διαλόγου δεν διατυπώθηκε σωστά. Το bot framework χρησιμοποιεί διακοπές για αυτό το λόγο στο πρόγραμμα του διαλογικού πράκτορα.

Οι διακοπές είναι αλυσιδωτές μέθοδοι οι οποίες χρησιμοποιούνται μετά τον βασικό ορισμό μιας διαδρομής. Σαν ορίσματα παραμέτρων δέχονται ένα μοτίβο κανονικών εκφράσεων (matches), π.χ. για την ακύρωση ενός διαλόγου `/^cancel/i` και μια προτροπή του διαλογικού πράκτορα (confirmPrompt).

```
// Add dialog to handle 'Buy' button click
bot.dialog('buyButtonClick', [ ... waterfall steps ... ])
    .triggerAction({
        matches: /(buy|add)\s.*shirt/i
        confirmPrompt: "This will cancel adding the current item. Are you sure?"
    })
    .cancelAction('cancelBuy', "Ok... Item canceled", {
        matches: /^cancel/i,
        confirmPrompt: "are you sure?"
    })
    .reloadAction('reloadBuy', "Restarting order.", {
        matches: /^start over/i,
        confirmPrompt: "are you sure?"
    });
```

Κώδικας 3.27 είδη διακοπών (action) στο διάλογο.

Χρήση φυσικής γλώσσας στο bot framework

Μέχρι αυτό το σημείο οι εντολές που αναφέρθηκαν παραπάνω, δεν έχουν ανάμειξη με την μονάδα κατανόησης φυσικής γλώσσας (π.χ. το LUIS). Είναι τμήματα διαλόγων τα οποία μπορούν να εκτελεστούν μηχανικά, έχοντας σχεδιάσει και προβλέψει την έκβαση της συνομιλίας. Απλές κανονικές εκφράσεις ως μοτίβα είναι αρκετές για την διεκπεραίωση απλών διαλόγων. Η ενσωμάτωση όμως μιας μονάδας κατανόησης φυσικής γλώσσας βοηθάει στην αναγνώριση περιπτώσεων οι οποίες μπορεί να μην είναι φανερές στην αρχή της σχεδίασης του διαλογικού πράκτορα (edge cases). Επίσης

βοηθάει στην αναγνώριση του σκοπού του χρήστη, το οποίο είναι ένα από τα δυνατά σημεία ενός διαλογικού πράκτορα.

Το bot framework παρέχει τριών ειδών αναγνωρίσεις φυσικής γλώσσας (recognizers). Έχει έναν απλό αναγνωριστή ο οποίος χρησιμοποιεί κανονικές εκφράσεις:

```
// Add regular expression recognizer to bot
bot.recognizer(new builder.RegExpRecognizer( "CancelIntent", { en-us:
/^(cancel|nevermind)/i, ja-jp: /^(キャンセル)/ }));
```

Κώδικας 3.28 Αναγνώριση μέσω κανονικών εκφράσεων [61]

Τον αναγνωριστή μέσω της χρήσης της υπηρεσίας LUIS:

```
// Add global LUIS recognizer to bot
var model = process.env.model ||
'https://api.projectoxford.ai/luis/v2.0/apps/c413b2ef-382c-45bd-8ff0-
f76d60e2a821?subscription-key=6d0966209c6e4f6b835ce34492f3e6d9';
bot.recognizer(new builder.LuisRecognizer(model));
```

Κώδικας 3.29 Αναγνώριση μέσω LUIS [61]

Τέλος, μπορεί να χρησιμοποιηθεί ένας προσαρμοσμένος αναγνωριστής [61]. Χρησιμοποιώντας τους αναγνωριστές στο διάλογο, δεν χρησιμοποιείται πλέον η διαδικασία καταρράκτης όπως στο απλό διαλογικό σύστημα μέσω των διαδρομών, αλλά μπορούμε να προγραμματίσουμε το λογισμικό να παρατηρεί τις εκφράσεις και να βρίσκει τις αντίστοιχες ετικέτες σκοπών του χρήστη. Στο παράδειγμα του καιρού, ο σκοπός του χρήστη από τα προεπιλεγμένα δεδομένα ονομάζεται: `builtin.intent.weather.check_weather`. Έτσι, στο διάλογο για να μπορέσει να ανιχνεύσει αυτό το συγκεκριμένο σκοπό χρήστη το μόνο που χρειαζόμαστε είναι να δηλώσουμε τον αναγνωριστή του LUIS και έπειτα να ορίσουμε ότι το αντικείμενο του διαλόγου θα αντιστοιχήσει το συγκεκριμένο σκοπό χρήστη όταν τον αναγνωρίσει:

```
// Start Intent-to-Dialog service with LUIS
var dialog = new builder.IntentDialog({recognizers: [recognizer]});
bot.dialog('/', dialog);

// dialog matching
dialog.matches("builtin.intent.weather.check_weather", [
```

Κώδικας 3.30 προγραμματισμός του λογισμικού για αναγνώριση μέσω σκοπού χρήστη.

Διαδικασία slot-filling με το bot framework

Η διαδικασία slot-filling στο bot framework γίνεται στο προγραμματιστικό τμήμα. Στο κώδικα του διαλόγου, ελέγχουμε το αντικείμενο της συνεδρίας (session). Εάν υπάρχει ορισμένη η τιμή που θέλουμε από τις οντότητες που έχουν ανιχνευτεί μέσω του LUIS τότε μπορούμε να προχωρήσουμε στο επόμενο σημείο του διαλόγου. Ειδάλλως θα πρέπει να ρωτήσουμε τον χρήστη διερευνητικά για την τιμή της παραμέτρου, η οποία στην κλήση της επόμενης callback μεθόδου θα πρέπει να εισαχθεί μέσα στην συνεδρία προγραμματιστικά. Ειδάλλως ο διάλογος θα ξαναγυρίσει στην αρχή όπου ζητάει την πληροφορία για την τιμή της παραμέτρου. Η ιεραρχία της αναζήτησης των τιμών αυτών γίνεται ιεραρχικά, οπότε ο προγραμματιστής καθορίζει ποια παράμετρο θα πρέπει να ζητηθεί πρώτα.

Η διαδικασία αυτή συνεχίζεται, έως ότου όλες οι τιμές των παραμέτρων που χρειαζόμαστε έχουν ολοκληρωθεί, οπότε και μπορεί να γίνει η επεξεργασία πλέον των παραμέτρων για το σκοπό του χρήστη, δηλαδή είτε την κλήση μιας απομακρυσμένης υπηρεσίας ή την κλήση μιας τοπικής μεθόδου που θα υπολογίζει κάποιο αποτέλεσμα. Με την χρήση της μεθόδου session.send() γίνεται η τελική παραγωγή της φυσικής γλώσσας και έπειτα ο διάλογος επιστρέφει στο αρχικό σημείο του.

3.8 Σύγκριση των λογισμικών διαλογικών πρακτόρων

Όπως αναφέρθηκε παραπάνω, οι διαλογικοί πράκτορες χωρίζονται σε ψυχαγωγικούς και σε εμπορικούς. Οι ψυχαγωγικοί διαλογικοί πράκτορες έχουν την τάση να σχεδιάζονται με γνώμονα το Turing test ή άλλους διαγωνισμούς οι οποίοι θα ξεχωρίσουν την τεχνητή από την ανθρώπινη νοημοσύνη και είναι εκτός των ορίων της μεταπτυχιακής διατριβής. Τα κριτήρια των εμπορικών πρακτόρων όμως αποτελούν μέτρο ποιοτικής σύγκρισης και χρησιμοποιούνται για την εκτίμηση των λογισμικών δημιουργίας διαλογικών πρακτόρων.

3.8.1 Κριτήρια σύγκρισης

Ένα από τα πιο σημαντικά θέματα [62] που κάθε σχεδιαστής διαλογικού πράκτορα πρέπει να έχει υπόψη είναι η εμπειρία χρήστη που θα πρέπει να παρέχει μέσω του διαλόγου. Έτσι οι παράγοντες-κλειδιά για την επιτυχία ενός διαλογικού πράκτορα πρέπει να είναι:

1. Πόσος χρόνος χρειάζεται για την εκπλήρωση ενός σκοπού χρήστη σε ένα διάλογο;
2. Η χρήση διαλόγου λύνει το πρόβλημα σε σχέση με άλλους τρόπους (π.χ. χρήση μιας φόρμας);
3. Μπορεί ο διαλογικός πράκτορας να εκτελεστεί σε περισσότερες από μία πλατφόρμες ή κινητές συσκευές ή κανάλια διαλογικών πρακτόρων;

Στον αντίποδα, παράγοντες που δεν εγγυώνται την επιτυχία ενός διαλογικού συστήματος είναι:

- Πόσο «έξυπνος» ένας διαλογικός πράκτορας είναι;
- Πόση επεξεργασία φυσικής γλώσσας ένας διαλογικός πράκτορας χρησιμοποιεί;
- Χρησιμοποιούνται δευτερεύουσες υπηρεσίες, π.χ. φωνητικές υπηρεσίες;

3.8.2 Σύγκριση των bot frameworks

Κατά την δημιουργία των διαλογικών πρακτόρων με τα τρία πακέτα λογισμικού χρησιμοποιήθηκαν σχεδόν ίδιες έννοιες. Κάθε πακέτο διαλογικού πράκτορα είχε ένα έργο (project) το οποίο στο wit.ai ήταν η εφαρμογή (app), στο api.ai ήταν ο διαλογικός πράκτορας (agent) και στο bot framework ήταν είτε ένα έργο ή ένα δημοσιευμένο bot στην διαχειριστική κονσόλα του bot framework (Register a bot). Για το bot framework επίσης χρειάζεται η συγκρότηση ενός έργου στην διαχειριστική κονσόλα του LUIS μιας που η φυσική γλώσσα είναι ένα πρόσθετο συστατικό και δεν είναι ενσωματωμένο στο σύστημα του διαλογικού πράκτορα. Κάθε πακέτο παρείχε τα ίδια χαρακτηριστικά: Μονάδα κατανόησης φυσικής γλώσσας, αναγνώριση σκοπού του χρήστη, αναγνώριση οντοτήτων, δημιουργία συνεδρίας (context) για το διάλογο, δημιουργία ενεργειών και εκπαίδευση του διαλογικού πράκτορα.

Το wit.ai και το api.ai είχαν ενσωματωμένα τις μονάδες κατανόησης φυσικής γλώσσας. Οι μηχανές κατανόησης φυσικής γλώσσας χρησιμοποιούνται κυρίως για την αναγνώριση των σκοπών του χρήστη μέσα από εκφράσεις οι οποίες είναι έτοιμες μέσω μηχανικής μάθησης, χειρωνακτικών κανόνων (κανονικές εκφράσεις / regular expressions) και εκπαίδευσης από τον δημιουργό μέσω ανάλυσης των εκφράσεων που άλλοι χρήστες εισήγαγαν κατά την χρήση του διαλογικού πράκτορα. Ο τρόπος υλοποίησης τους δεν είναι γνωστός στο κοινό αυτή την χρονική στιγμή δηλαδή κατά πόσο χρησιμοποιείται η μηχανική μάθηση υπέρ των χειρωνακτικών κανόνων στις υπηρεσίες που συγκροτούν το wit.ai και το api.ai. Στο bot framework, χρησιμοποιούνται τριών ειδών μηχανές κατανόησης για τον διαλογικό πράκτορα. Η απλή μηχανή κατανόησης η οποία βασίζεται σε χειρωνακτικούς κανόνες, η μηχανή LUIS ή κάποια εξωτερική υπηρεσία κατανόησης φυσικής γλώσσας και τέλος η δυνατότητα να δημιουργήσει ο χρήστης μια δικιά του υλοποίηση μηχανής κατανόησης φυσικής γλώσσας και να την χρησιμοποιήσει στο διαλογικό πράκτορα ενός έργου bot framework.

Στην αναγνώριση σκοπού του χρήστη, το wit.ai απαιτεί την δημιουργία ιστοριών πρώτα και μετά τον ορισμό των εκφράσεων που ο τελικός χρήστης θα διατυπώσει. Στον ορισμό αυτό, ο σκοπός του χρήστη ορίζεται ως οντότητα η οποία έπειτα χρησιμοποιείται ως παράμετρο στο υλοποιημένο πρόγραμμα. Η μηχανή κατανόησης του wit.ai αναλαμβάνει να βρει με βαθμό αξιοπιστίας (ποσοστό ή confidence) πόσο κοντά στο σκοπό του χρήστη είναι η έκφραση άρα και να αποσαφηνιστεί. Η μηχανή κατανόησης LUIS του bot framework καθώς και η μηχανή κατανόησης του api.ai αναγνωρίζει τους σκοπούς του χρήστη βασισμένους στην ετικέτα (utterance) ή στο τίτλο (intent title) που τους έχει δοθεί σε ξεχωριστή ενότητα μέσα στο διαχειριστικό. Το αποτέλεσμα της εύρεσης όμως είναι πάντα το ίδιο, απαιτεί εκπαίδευση του μοντέλου για να γνωρίζει ποιες από τις εκφράσεις τελικά ανήκει στο συγκεκριμένο σκοπό του χρήστη.

Σε κάθε framework υπάρχουν προκαθορισμένες οντότητες (built-in entities) οι οποίες ορίζονται από τους δημιουργούς του εκάστοτε πακέτου. Στο wit.ai αυτές οι οντότητες έχουν το πρόθεμα **wit/** [63] ενώ στο api.ai χρησιμοποιούν το πρόθεμα **@system.** [64] Στο bot framework οι οντότητες αυτές έχουν το πρόθεμα **builtin.** [65]. Οι επιπλέον οντότητες που αφορούν το έργο ή τις επιλογές χρήστη ορίζονται μέσα από το

διαχειριστικό σύστημα, με ένα παρόμοιο τρόπο σε όλα τα frameworks. Στο wit.ai και στο api.ai σημειώνονται επισημαίνοντας την λέξη ή τις λέξεις που θέλουμε να οριστούν ως οντότητες και δημιουργείται μια νέα ομάδα οντοτήτων. Στο bot framework αυτό συμβαίνει μέσω του LUIS στο οποίο ορίζουμε την οντότητα μέσα στο σκοπό του χρήστη. Το api.ai παράλληλα παρέχει την δυνατότητα να δημιουργηθούν πιο πολύπλοκες μονάδες οντοτήτων (composite entities) καθώς και οντότητες οι οποίες δημιουργούνται κατά την εκτέλεση του διαλογικού πράκτορα και χρησιμοποιούνται μόνο για το διάστημα της συνεδρίας (user entities).

Η συνεδρία (context) υπάρχει σε όλα τα framework. Στο wit.ai και στο api.ai υπάρχει εμφανέστατα μέσω των αντικειμένων context, ενώ στο bot framework το αντικείμενο UniversalBot παρέχει ένα αντικείμενο session το οποίο χρησιμοποιείται τόσο για την συνεδρία της εφαρμογής όσο και για τη συνεδρία του διαλόγου. Χωρίς τη συνεδρία κάθε διάλογος δεν θα μπορούσε να «θυμάται» προηγούμενες καταστάσεις. Με την συνεδρία, οντότητες που έχουν μετατραπεί σε μεταβλητές μπορούν να παραμένουν σε επόμενα βήματα του διαλόγου, ενώ οι ίδιες οι συνεδρίες να καθορίζουν την ροή του διαλόγου. Το api.ai έχει εκτενές έλεγχο των συνεδριών μέσω του διαχειριστικού συστήματος. Κάθε αντικείμενο συνεδρίας που δημιουργείται (μπορεί να έχει περισσότερο από ένα, το οποίο αντικατοπτρίζει την κατάσταση του σκοπού του χρήστη), μπορεί να μεταφερθεί σε μια άλλη κατάσταση σκοπού του χρήστη μαζί με τις μεταβλητές και τις επιλογές του χρήστη. Αυτό είναι χρήσιμο όταν έχεις αρκετούς σκοπούς χρήστη οι οποίοι μπορούν να χρησιμοποιούν ίδια δεδομένα, «ακούγοντας» τα μέσω της συνεδρίας και να αναλαμβάνουν τον έλεγχο του διαλόγου [66]. Ένα αντίστοιχο χαρακτηριστικό που παρέχει το api.ai είναι οι επακόλουθοι σκοποί χρήστη (follow-up intents). Χρησιμοποιούν το αντικείμενο context για να καθορίσουν την άμεση ροή του διαλόγου, π.χ. αν ο χρήστης θελήσει να απαντήσει Ναι / Όχι ή κάτι άλλο, όμως δεν παρέχεται για την κανονική ροή του διαλόγου. Κυρίως μετατρέπει την διαδικασία του διαλόγου σε μια μηχανή πεπερασμένων καταστάσεων.

Η δημιουργία ενεργειών (actions) στα frameworks ποικίλλει. Κύριο ρόλο έχει το διαχειριστικό κομμάτι της εφαρμογής του διαλογικού πράκτορα. Έτσι στο wit.ai και στο api.ai υπάρχει τμήμα της εφαρμογής αφιερωμένο στις ενέργειες του χρήστη, ενώ στο bot framework η ενέργεια του χρήστη μπορεί να οριστεί σαν μια ενδιάμεση προγραμματιστική εντολή κατά την διαδικασία του διαλόγου waterfall ή μέσα στις

μεθόδους που καθορίζουν το διάλογο. Στο wit.ai κάθε func η οποία ορίζεται στο διαχειριστικό κομμάτι των ιστοριών με τη δομή «bot executes» πρέπει να οριστεί αυτούσια στο πρόγραμμα. Στο api.ai οι ενέργειες ορίζονται ως μια ετικέτα η οποία εξετάζεται μέσα από το μήνυμα που στέλνει η υπηρεσία στην εφαρμογή του διαλογικού πράκτορα. Σε αυτό το κομμάτι το bot framework έχει πιο ελεύθερη χρήση του προγραμματιστικού μοντέλου σε σχέση με τις παρεχόμενες βιβλιοθήκες του wit.ai και του api.ai, που σημαίνει ότι μπορεί να εκτελεί ενέργειες χωρίς να χρειάζεται να τερματίσει το διάλογο. Αυτό βοηθά στην ελαχιστοποίηση κάποιων βημάτων του διαλόγου οι οποίοι απαιτούνται στα άλλα frameworks.

Η ενσωμάτωση των διαλογικών πρακτόρων σε κανάλια διαλογικών πρακτόρων είναι εύκολη στο api.ai και στο bot framework. Μέσα από το διαχειριστικό του api.ai υπάρχει η επιλογή ενσωματώσεις (integrations) η οποία μετατρέπει την υπηρεσία που διαχειρίζεται το api.ai σαν μια ενδιάμεση υπηρεσία επικοινωνίας με κάποιο κανάλι, π.χ. slack. Έτσι η είσοδος των εκφράσεων από το χρήστη καθώς και η έξοδος των αποτελεσμάτων από την υπηρεσία μπορεί να διαβαστεί από το κανάλι χωρίς την ύπαρξη κάποιου επιπλέον λογισμικού ενσωμάτωσης. Αντίστοιχα το bot framework μέσω του Bot Connector και της δυνατότητας έκδοσης του διαλογικού πράκτορα μπορεί να κάνει παρόμοια εργασία για διάφορα κανάλια. Στο wit.ai η υλοποίηση αυτή γίνεται προγραμματιστικά, χρησιμοποιώντας τις βιβλιοθήκες που κάθε κανάλι διαλογικού πράκτορα παρέχει για την επέκταση του. Αυτό σημαίνει ότι θα χρειαστεί επιπλέον χρόνος για την αποσφαλμάτωση της διεπαφής μεταξύ του καναλιού και του διαλογικού πράκτορα.

Όλα τα frameworks υποστηρίζουν υπηρεσίες φωνής. Η υλοποίηση τους είναι διαφορετική σε κάθε framework, μιας που η λειτουργία αυτή βρίσκεται ακόμα σε δοκιμαστικό στάδιο. Στο wit.ai υπάρχει ειδικό endpoint το οποίο παρέχει μετατροπή ενός αρχείου wav σε κείμενο [67]. Στο api.ai η λειτουργία αυτή εμφανίζεται στη δοκιμαστική κονσόλα, εφόσον υπάρχει υποστήριξη από το browser (chrome αυτή τη στιγμή) και σε επιλεγμένες ενσωματώσεις (π.χ. google assistant, Cortana), οι οποίες απαιτούν προγραμματισμό από την πλευρά των εφαρμογών που ενσωματώνεται η υπηρεσία api.ai. Στο bot framework υπάρχει ως υπηρεσία μέσω των Microsoft Cognitive Services [68].

Η διαθεσιμότητα βιβλιοθηκών και τεκμηρίωσης για τα frameworks είναι ένας μεγάλος παράγοντας κατά την υλοποίηση των διαλογικών πρακτόρων. Στη διάρκεια της μεταπτυχιακής διατριβής το λογισμικό των διαλογικών πρακτόρων ανανεώνονταν συνεχώς, έχοντας κάποιες ανακρίβειες με την υπάρχουσα τεκμηρίωση. Αυτό είχε ως αποτέλεσμα την αποφυγή χρήσης κάποιων frameworks για κάποιο διάστημα μέχρι να υπάρξει ανανέωση ή μέχρι να βρεθεί λύση στα προβλήματα που δημιουργήθηκαν. Το προσωπικό υποστήριξης των υπηρεσιών αυτών υπήρξε εξυπηρετικό αρκετές φορές όμως υπήρξε ένα παράθυρο χρόνου μεταξύ των ερωτημάτων και των απαντήσεων για την συνέχιση της υλοποίησης των διαλογικών πρακτόρων. Πολλές φορές χρειάστηκε να γίνει ανάλυση των μηνυμάτων που στέλνει η εκάστοτε υπηρεσία από και προς την εφαρμογή του διαλογικού πράκτορα (reverse engineering) ώστε να κατανοηθεί ο σκοπός των δομών του μηνύματος μέχρι να υπάρξει ανανέωση της τεκμηρίωσης ή και των βιβλιοθηκών API. Ένας λόγος που προτιμήθηκε η javascript έναντι της java ή μιας γλώσσας του visual studio (c#, c++, visual basic) για την υλοποίηση των διαλογικών πρακτόρων ήταν λόγω αυτής της δυσκολίας.

Μέσα στο χρονικό διάστημα που εκπονήθηκε η μεταπτυχιακή διατριβή μερικές λειτουργίες άλλαξαν ενώ ενσωματώθηκαν κάποιες άλλες. Στο wit.ai άλλαξε ο τρόπος λειτουργίας των μηνυμάτων και των REST endpoints που είχε αρχικά η τεκμηρίωση. Επίσης αφαιρέθηκαν οι σύνθετες παράμετροι / οντότητες, οι οποίες θα ενσωματωθούν ξανά σε μελλοντική έκδοση σύμφωνα με τους κατασκευαστές του. Τέλος εμπλουτίστηκε η μηχανή κατανόησης με την χρήση της duckling, για την αναγνώριση οντοτήτων [69]. Στο api.ai καταργήθηκαν τα προκαθορισμένα πεδία (domains) ενώ μερικές υλοποιήσεις μεταφέρθηκαν στις ενσωματώσεις (integrations) από διάφορες υπηρεσίες της google. Κυρίως λόγω της εξαγοράς του api.ai από την google, υπάρχει μια γενική σύγκλιση των υπηρεσιών του api.ai με αυτές της google. Τέλος στο bot framework άλλαξε η τεκμηρίωση σε μια πιο ολοκληρωμένη κατάσταση, όμως παράλληλα οι υπηρεσίες του bot framework μεταφέρθηκαν στο νέφος azure.

Το κόστος της υλοποίησης των εφαρμογών με τα bot frameworks ποικίλλει. Κυρίως η διάθεση των api.ai και wit.ai είναι δωρεάν (χωρίς αυτό να αλλάξει στο μέλλον). Το bot framework με την «ενηλικίωση» του μεταφέρθηκε στο νέφος azure. Διατίθεται δωρεάν με περιορισμούς στους developers με ελεύθερη στέγαση των υπηρεσιών για δοκιμές, όμως η τελική δημοσιευμένη υπηρεσία πρέπει να είναι στεγασμένη στο νέφος azure, το

οποίο παρέχει ειδικό κοστολόγιο τόσο για την στέγαση, συντήρηση και δημιουργία διαλογικών μονάδων κατανόησης μέσω του LUIS.

Στο επίπεδο της υλοποιημένης εφαρμογής του καιρού, παρατηρήθηκε ότι η υλοποίηση του καιρού ήταν πιο εύκολη με το api.ai επειδή όλη η υλοποίηση έγινε μέσα στο διαχειριστικό σύστημα και η κλήση των εξωτερικών υπηρεσιών μόνο χρειάστηκαν την υλοποίηση ενός προγράμματος node.js, ενώ την μεγαλύτερη δυσκολία είχε η υλοποίηση με το wit.ai λόγω της αλλαγής της τεκμηρίωσης το διάστημα της δημιουργίας του διαλογικού πράκτορα. Στο bot framework χρησιμοποιήθηκαν τα δεδομένα της εφαρμογής Cortana στα οποία υπήρχαν προεγκατεστημένα και οι σκοποί του χρήστη και οι οντότητες που κάποιος θα μπορούσε να ρωτήσει τον καιρό και αυτό επιτάχυνε την υλοποίηση, αλλά υπήρχε πάλι κάποιος χρόνος ο οποίος χρειάστηκε για την κατανόηση των δομών των μηνυμάτων του διαλογικού πράκτορα.

Ο χρόνος απόκρισης των διαλογικών βοηθών ποικίλλει. Στις περιπτώσεις που υπήρχαν σωστά διατυπωμένες ερωτήσεις μέσα στο διάλογο, δηλαδή μαζί με την ερώτηση του καιρού υπήρχε η τοποθεσία και ο χρόνος η εύρεση του καιρού ήταν σύντομη, εφόσον η τοποθεσία είχε διατυπωθεί σωστά. Λόγω της υλοποίησης και της χρήσης της υπηρεσίας google geocoding, πολλές φορές, τοποθεσίες όπως «Athens» ή «Peru» έπρεπε να οριστούν με ένα δεύτερο όρισμα, π.χ. την χώρα που βρίσκονται δηλαδή «Athens, Greece» ή «Peru, South America», για να βρεθεί η σωστή τοποθεσία για geocoding. Φυσικά αυτό οφείλεται στην ποιότητα της υπηρεσίας που χρησιμοποιήθηκε και κατά πόσο η δική της υπηρεσία κατανόησης φυσικής γλώσσας λειτουργεί σωστά η με ποιες προϋποθέσεις. Ένα άλλο παράδειγμα προβληματικής αναγνώρισης τοποθεσιών ήταν όταν ο χρήστης αντί να εισάγει μια τοποθεσία, έβαζε ως προορισμό το «here». Για το διαλογικό πράκτορα, το here αντιστοιχούσε στην τοποθεσία που βρίσκεται η υπηρεσία του διαλογικού πράκτορα και όχι η υλοποίηση του, για το wit.ai έβρισκε ότι ήταν στην Αμερική π.χ., ενώ άλλες υλοποιήσεις θεωρούσαν το «here» ως την default κενή απάντηση, επειδή υπήρχε έλλειψη μιας υπηρεσίας gps. Όσο λιγότερες πληροφορίες (οντότητες) εισάγονται στην ερώτηση τόσο περισσότερος χρόνος απαιτείται λόγω της χρήσης του slot-filling το οποίο αρχίζει την διαδικασία διερευνητικών ερωτήσεων. Επίσης στο wit.ai παρατηρήθηκε ότι η υλοποίηση της ιστορίας (και η έλλειψη δευτερευόντων ιστοριών) οδηγεί το διάλογο κατευθείαν στην διαδικασία slot-filling, π.χ. αν ο χρήστης γράψει στο wit.ai:

Χρήστης: Hello

Chatbot: Where?

Το οποίο δηλώνει ότι δεν αναγνωρίζει καλά το σκοπό χρήστη όταν λείπουν περισσότερες πληροφορίες η περισσότερες ιστορίες. Το bot framework (LUIS) έχει ένα καλά εκπαιδευμένο σύστημα μέσω των λειτουργιών της εφαρμογής Cortana οπότε μπορεί να αναγνωρίσει καλύτερα τους σκοπούς χρήστη και να αγνοήσει οτιδήποτε δεν έχει σχέση με τον καιρό. Τέλος το api.ai έχει κάποιες έτοιμες απαντήσεις κυρίως για small talk (δηλαδή μικροκουβέντας), οι οποίες μπορούν να χρησιμοποιηθούν μέχρι να εισάγει ο χρήστης την σωστή ερώτηση για να εκκινήσει τον διάλογο. Γενικώς αυτές οι απαντήσεις αφήνονται στην κρίση του δημιουργού του διαλογικού πράκτορα καθώς και αν θα δώσει κάποια βοήθεια στο χρήστη για να είναι πιο «ανθρώπινη» η συζήτηση και να μην απαιτεί αφαιρετική ικανότητα από το χρήστη ώστε να καταλάβει τι λειτουργία έχει ο διαλογικός πράκτορας ώστε να οριοθετήσει τους σκοπούς του.

Στο παρακάτω πίνακα εμφανίζονται συνοπτικά όλα τα αποτελέσματα των συγκρίσεων των frameworks που χρησιμοποιήθηκαν σε αυτή τη μεταπτυχιακή διατριβή:

	Wit.ai	**Api.ai**	**Bot Framework**
Διαχειριστικό σύστημα	Ενσωματωμένο στην υπηρεσία wit.ai	Ενσωματωμένο στην υπηρεσία wit.ai	Διπλό διαχειριστικό σύστημα, στο bot connector και στην υπηρεσία LUIS
Υποστήριξη φυσικής γλώσσας	Ενσωματωμένη στην υπηρεσία	Ενσωματωμένη στην υπηρεσία	Είτε χρησιμοποιώντας κανονικές εκφράσεις ή μέσω της υπηρεσίας LUIS
Αναγνώριση σκοπών χρήστη	Ορισμός από το διαχειριστικό μέσω ειδικής οντότητας στην ενότητα «ιστορίες», δομή «user says». Εκπαίδευση μέσω του διαχειριστικού συστήματος σε νέες εκφράσεις	Ενσωματωμένη στην υπηρεσία. Περιέχει ειδική ενότητα «intents» η οποία ελέγχει και εκκινεί το διάλογο. Εκπαίδευση μέσω του διαχειριστικού συστήματος σε νέες εκφράσεις	Ενσωματωμένη στην υπηρεσία LUIS. Προαιρετική χρήση, δυνατότητα χρήσης κανονικών εκφράσεων ή προσαρμοσμένου προγραμματιστικού κώδικα στην υλοποίηση του προγράμματος (recognizers).
Αναγνώριση οντοτήτων	Αναγνώριση από το διαχειριστικό,	Αναγνώριση από το διαχειριστικό,	Ενσωματωμένη δυνατότητα

	προκαθορισμένες οντότητες Οντότητες που καθορίζει ο δημιουργός του διαλογικού πράκτορα	προκαθορισμένες οντότητες Οντότητες που καθορίζει ο δημιουργός του διαλογικού πράκτορα	αναγνώρισης στην υπηρεσία LUIS.
Αναγνώριση ενεργειών	Ορισμός στο διαχειριστικό σύστημα, εκτέλεση μέσα από την υλοποίηση της εφαρμογής	Ορισμός στο διαχειριστικό σύστημα, εκτέλεση μέσα από την υλοποίηση της εφαρμογής	Ορισμός μέσα στην εφαρμογή όποτε χρειάζεται, αποσύνδεση από τον σκοπό χρήστη.
Προκαθορισμένα πεδία (domains)	-	-	Περιέχει μέσω της υπηρεσίας LUIS
Ενσωματώσεις	-	Παρέχονται μέσα από το διαχειριστικό με βοήθεια υλοποίησης τους	Παρέχονται μέσα από το Bot Connector με την βοήθεια υλοποίησης τους, χρήση του νέφους Azure
Χρόνος υλοποίησης	Ποικίλλει ανάλογα με το διάλογο και παρελκόμενες εκφράσεις (small talk)	Ποικίλλει ανάλογα με το διάλογο και παρελκόμενες εκφράσεις (small talk)	Ποικίλλει ανάλογα με το διάλογο και παρελκόμενες εκφράσεις (small talk)
Ταχύτητα εκτέλεσης	Ποικίλλει ανάλογα με την χρήση των υπηρεσιών που απαιτούνται στην εκτέλεση των σκοπών του χρήστη	Μπορεί να είναι γρηγορότερη ανάλογα με τη μορφή του διαλόγου	Ποικίλλει ανάλογα με την χρήση των υπηρεσιών που απαιτούνται στην εκτέλεση των σκοπών του χρήστη
Λειτουργία γραμμικού διαλόγου	Υποστηρίζεται	Υποστηρίζεται	Υποστηρίζεται
Λειτουργία slot-filling	Υποστηρίζεται	Υποστηρίζεται μέσω του διαχειριστικού και μέσω υπηρεσιών με fulfillment	Υποστηρίζεται
Λειτουργία μη γραμμικού διαλόγου (αλλαγή σκοπού χρήστη)	Δεν υποστηρίζεται	Υποστηρίζεται υπό προϋποθέσεις	Υποστηρίζεται αλλά μόνο με προκαθορισμένα δεδομένα από μοντέλα πληροφοριών (π.χ. Cortana)
Μορφή κειμένων	Υποστηρίζει εμπλουτισμένο κείμενο (πολυμέσα,	Υποστηρίζει εμπλουτισμένο κείμενο (πολυμέσα,	Υποστηρίζει εμπλουτισμένο κείμενο (πολυμέσα,

	φωνή)	φωνή)	φωνή)
Τεκμηρίωση	Επαρκής αλλά ασταθείς λόγω ενημερώσεων	Πολύ καλή και σταθερή	Πολύ καλή
Βιβλιοθήκες API	Περιορισμένες, χρειάζεται τεκμηρίωση στο HTTP API	Περισσότερες βιβλιοθήκες από όλα τα framework που συγκρίθηκαν	Περιορισμένες στις γλώσσες που υποστηρίζει το visual studio, το HTTP API χρειάζεται καλύτερη τεκμηρίωση
Κόστος	Δωρεάν διάθεση	Δωρεάν διάθεση	Δωρεάν διάθεση για τους προγραμματιστές με περιορισμούς Αγορά μέσω του νέφους Azure.

Πίνακας 3.2 Συγκεντρωτικός οδηγός σύγκρισης των frameworks

Από τα παραπάνω frameworks, το api.ai έχει τα πιο σταθερά χαρακτηριστικά για την υλοποίηση ενός διαλογικού πράκτορα. Το επόμενο κεφάλαιο θα χρησιμοποιήσει το api.ai για την πλήρη υλοποίηση ενός διαλογικού πράκτορα ο οποίος θα χρησιμοποιήσει τα περισσότερα από τα χαρακτηριστικά που αναφέρθηκαν κατά την σύγκριση.

Κεφάλαιο 4
Υλοποίηση διαλογικού βοηθού

4.1 Εισαγωγή

Στη τελευταία δεκαετία η τεχνολογία έχει εισχωρήσει δυναμικά στην ψυχαγωγία σε διάφορους τομείς. Ένας από αυτούς τους τομείς είναι και το διαδικτυακό νόμιμο στοίχημα. Αρχικά, δημιουργήθηκαν εφαρμογές client/server οι οποίες ανέλαβαν τον εκσυγχρονισμό των εμπορικών διαδικασιών των operators ώστε να υπάρχουν δομημένα δεδομένα. Η διαδικασία συνεχίστηκε με την δημιουργία ιστοχώρων στους οποίους ο ενδιαφερόμενος παίκτης μπορεί να έχει πρόσβαση εύκολα σε όλους τους αγώνες τόσο σε μεμονωμένες χώρες όπως η Αγγλία και η Ιρλανδία, όσο και παγκόσμια. Αρχικά οι ιστοχώροι αυτοί λειτουργούσαν με την μορφή email ή φορμών που οδηγούσαν σε κάποια ενέργεια αποστολής email.

Όσο η τεχνολογία δημιουργούσε νέες δομές, αλγόριθμους και τεχνικές, οι ιστοχώροι αυτοί μετεξελίχθηκαν σε ειδικές διαδικτυακές εφαρμογές (web apps) οι οποίες ονομάζονται sportsbook. Σε ενα sportsbook, εφαρμόζονται αρκετές τεχνολογίες οι οποίες σχετίζονται με την λειτουργία του ιστοχώρου όπως τεχνολογίες μετάδοσης εικόνας και ήχου σε πραγματικό χρόνο (streaming technologies), responsive web design για την ενιαία εμπειρία χρήστη καθώς και τεχνολογίες οι οποίες βρίσκονται στο

backend της εφαρμογής, όπως big data, χρήση enterprise databases και αρχιτεκτονικής υπηρεσιών μέσω διαδικτύου (service oriented architecture).

Με την άνθιση των κινητών συσκευών (mobile devices) και των εφαρμογών τους, τα sportsbook αρχίζουν να μετεξελίσσονται έτσι ώστε να είναι clients σε μια πιο μεγάλη πλατφόρμα με RESTful υλοποιήσεις υπηρεσιών (RESTful web services). Αυτό σημαίνει οτι η υλοποίηση τους γίνεται με τέτοιο τρόπο που οι υπηρεσίες REST ή το API που δημιουργούν εταιρείες ανάπτυξης πλατφορμών sportsbook μπορούν να χρησιμοποιηθούν, στο επίπεδο που επιτρέπουν, ως πηγή για μια εφαρμογή chatbot.

Ένα πρόβλημα που απασχολεί τις εταιρείες στοιχημάτων είναι η κάλυψη των παραμέτρων που χρειάζονται για την διεκπεραίωση ενός στοιχήματος (bet placement) όσο πιο σωστά γίνεται. Για το λόγο αυτό, η διεπαφή που παρέχουν στο sportsbook, είναι όσο το δυνατόν πιο απλή αλλά όσο οι συνδυασμοί ενός στοιχήματος πληθαίνουν, αυξάνεται και η πολυπλοκότητα. Ένας ψηφιακός διαλογικός βοηθός στοιχήματος (digital bet assistant) θα μπορούσε να βοηθήσει, κάνοντας τις ερωτήσεις που χρειάζεται για την διεκπεραίωση, σε οποιαδήποτε μέσο, είτε είναι το sportsbook, είτε είναι ενας γενικότερος ψηφιακός βοηθός (π.χ. Cortana) ή μια άλλη ενέργεια που χρειάζεται στοιχεία στοιχήματος.

Στη συνέχεια του κεφαλαίου, θα γίνει η προσπάθεια δημιουργίας ενός τέτοιου συστήματος διαλογικού βοηθού, το οποίο θα μπορεί να χρησιμοποιηθεί ως ενδιάμεσος (Middleware) σε διάφορα κανάλια chatbot. Στόχος είναι να εξεταστεί πόσο πρακτική και αποτελεσματική είναι η ενσωμάτωση του βοηθού σε μια πλατφόρμα sportsbook.

4.2 Έννοιες στοιχήματος

Ενα στοίχημα είναι μια "συμφωνία", στην οποία άνθρωποι προσπαθούν να προβλέψουν/μαντέψουν τι θα συμβεί επι πληρωμή. Οι προβλέψεις αυτές ή πιθανότητες κέρδους, εκφράζονται είτε ως κλάσματα (π.χ. 4/1) ή ως δεκαδικοί αριθμοί (π.χ. 4.00). Υπάρχουν δύο περιπτώσεις έκβασης του αποτελέσματος:

1. Η πρόβλεψη να βγει αληθινή και να Κερδίσει το ποσό που στοιχημάτισε αυτός που πρόβλεψε σωστά.

2. Η πρόβλεψη να μη βγει αληθινή και να χαθεί το ποσό που στοιχημάτισαν αυτοί που δεν πρόβλεψαν σωστά.

Στη πρώτη περίπτωση, αν στοιχημάτισε 1 ευρώ και οι πιθανότητες είναι 4/1, τότε το ποσό που κερδίζει θα είναι (4+1)*1 = 5 ευρώ. Στη δεύτερη περίπτωση, χάνει τα λεφτά που στοιχημάτισε.

4.2.1 Αγορές

Σε έναν αγώνα όπου καλείται ο χρήστης να στοιχηματίσει, υπάρχουν διαφόρων ειδών αγορές. Οι αγορές αυτές καθορίζουν που μπορεί κάποιος να στοιχηματίσει. Έτσι, αν ο αγώνας είναι ο αγώνας "Ομάδα Α / Ομάδα Β", τότε ο χρήστης μπορεί π.χ. να ποντάρει στην έκβαση του αποτελέσματος, δηλαδή να νικήσει η Ομάδα Α ή να νικήσει η Ομάδα Β ή να υπάρξει ισοπαλία. Κάθε έκβαση έχει μια πιθανότητα η οποία υπολογίζεται από κάποιο αλγόριθμο και είναι οι πιθανότητες της νίκης, π.χ. 4/1. Άλλο παράδειγμα αγοράς είναι να στοιχηματίσει πάνω σε κάποιο παίκτη, οτι αυτός θα κάνει τα περισσότερα καλάθια σε ενα αγώνα μπάσκετ ή θα βάλει τα περισσότερα γκολ.

4.2.2 Τύποι στοιχήματος

Κάθε εταιρεία στοιχήματος, για να αυξήσει το πελατολόγιο τής, μπορεί να δημιουργήσει διάφορους τύπους στοιχημάτων [70]. Υπάρχουν όμως κοινοί τύποι στοιχημάτων οι οποίοι χρησιμοποιούνται ως αναφορά για την δημιουργία πολύπλοκων στοιχημάτων ή συνδυασμών που θα οδηγήσουν σε πολύπλοκα στοιχήματα. Οι πιο κοινοί τύποι στοιχημάτων είναι:

- Single Bet
  Για να κερδίσει ο χρήστης, πρέπει να κερδίσει η επιλογή (selection/outcome) του.
- Double Bet
  Για να κερδίσει ο χρήστης, πρέπει να επιλέξει 2 επιλογές και να κερδίσουν και οι δύο

- Treble Bet

  Για να κερδίσει ο χρήστης, πρέπει να επιλέξει 3 επιλογές και να κερδίσουν και οι τρείς
- Accumulator Bet

  Πέρα των τριών επιλογών, κάθε επιλογή που επιλέγει επιπλέον πρέπει και να κερδίζει. Υπάρχουν accumulator στοιχήματα που μπορεί να αφορούν ακόμα και 60+ επιλογές.

4.2.3 Μορφή στοιχήματος

Παράλληλα ο χρήστης μπορεί να επιλέξει μια μορφή που θα έχει το στοίχημα. Αν και είναι και αυτό ένας τύπος στοιχήματος αναφέρεται ξεχωριστά. Οι μορφές του στοιχήματος είναι τυποποιημένα, τρείς:

- Win

  Ενα στοίχημα που πρέπει να κερδηθεί κάθε επιλογή του.
- Place

  Συνήθως άφορά ιπποδρομίες, είναι ένα στοίχημα, όπου η επιλογή του αναβάτη πρέπει να είναι μέσα στην εξάδα που κερδίζει.
- Each-Way

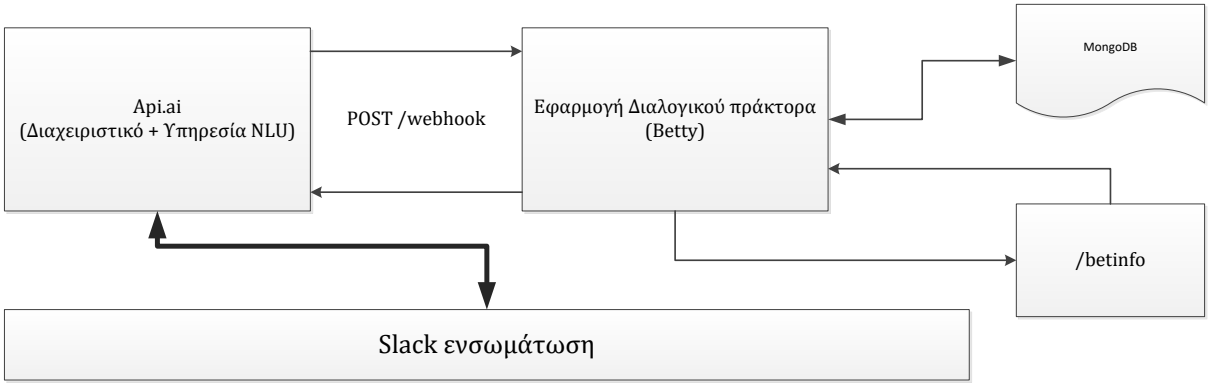
  Το each-way bet είναι ενας συνδυασμός των παραπάνω μορφών στοιχήματος για ενα single bet ή για ενα πολλαπλό στοίχημα (πάνω από 2 επιλογές).

4.3 Σχεδιασμός chatbot

Για την υλοποίηση του ψηφιακού βοηθού στοιχημάτων, θα χρησιμοποιηθεί η υπηρεσία api.ai και η γλώσσα javascript με node.js. Λόγω της πολυπλοκότητας που παρουσιάζει ένα sportsbook, θεωρούμε οτι υπάρχει ήδη έτοιμη μια υλοποίηση και ο ψηφιακός βοηθός θα είναι μια υπηρεσία (RESTful service) σε αυτό.

4.3.1 Αρχιτεκτονική διαλογικού βοηθού

Ένα διάγραμμα υπηρεσιών και endpoints της εφαρμογής είναι όπως το παρακάτω:



Εικόνα 4.1 Διάγραμμα υπηρεσιών της εφαρμογής (architecture)

Χρησιμοποιείται η υπηρεσία api.ai ως διαχειριστικό τμήμα του διαλογικού βοηθού (NLU) αλλά και ως ρυθμιστή για την ενσωμάτωση slack που αποτελεί τον τρόπο που συλλέγονται οι εκφράσεις του χρήστη αλλά και η οθόνη που παρατηρεί τις αποκρίσεις από το api.ai.

Μέσω του webhook /webhook το api.ai στέλνει τα μηνύματα ενεργειών (actions) καθώς και τις παραμέτρους/οντότητες και τα αντικείμενα context στην εφαρμογή. Η εφαρμογή έχει ρυθμίσει να «ακούει» τα μηνύματα που έρχονται από την αίτηση POST /webhook ενώ η εφαρμογή api.ai «ακούει» τις απαντήσεις των μηνυμάτων που στέλνει η εφαρμογή του διαλογικού βοηθού. Στην εφαρμογή έχει ρυθμιστεί επίσης σε άλλο endpoint η υπηρεσία /betInfo. Αυτή είναι προσβάσιμο μέσω GET /betinfo και επιστρέφει πίσω κείμενο με μορφή text/plain το οποίο περιέχει τις πληροφορίες που χρειάζεται ο σκοπός χρήστη «Bet Information». Η σύνδεση στη βάση δεδομένων γίνεται είτε σε τοπικό επίπεδο (κατά το development) ή σε μια απομακρυσμένη υπηρεσία

βάσης δεδομένων (mLab). Η στέγαση της εφαρμογής του διαλογικού βοηθού γίνεται στα πλαίσια της μεταπτυχιακής διατριβής στο νέφος openshift της Redhat. Με αυτό τον τρόπο μπορούμε να επικοινωνήσουμε με την υπηρεσία webhook του api.ai και επίσης να μεταφέρει τα μηνύματα του api.ai στην ενσωμάτωση slack ώστε να γίνει ταυτόχρονα και διεπαφή χρήστη – προγράμματος.

4.3.2 Υπηρεσία api.ai

Η υπηρεσία api.ai αναλαμβάνει την υποστήριξη του διαλογικού βοηθού. Παρέχει το module της μηχανής επεξεργασίας φυσικής γλώσσας και κατανόησης που χρειαζόμαστε. Για να επικοινωνήσει με άλλες υπηρεσίες πρέπει να συνδεθεί μέσω webhooks και να στέλνει μηνύματα με JSON payload σε συγκεκριμένα endpoints.

4.3.3 Node.js και javascript

Για την υλοποίηση της βασικής εφαρμογής και των επιμέρους υπηρεσιών της χρησιμοποιήθηκε η γλώσσα προγραμματισμού javascript και η εργαλειοθήκη node.js. Επίσης χρησιμοποιήθηκαν οι βιβλιοθήκες expressjs, body-parser, mongoose και axios.

Node.js

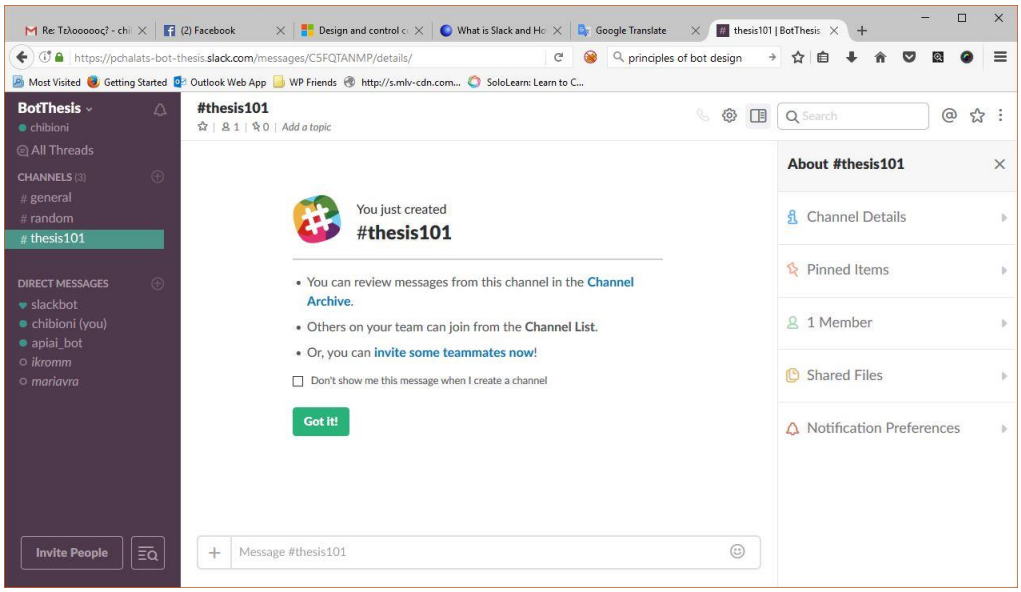
Προτιμήθηκε η node.js [71] γιατί είναι μια εργαλειοθήκη η οποία έχει επεκτάσεις και βιβλιοθήκες για όλα τα frameworks των διαλογικών πρακτόρων που εξετάστηκαν στην μεταπτυχιακή διατριβή. Παράλληλα έχει εύκολο σχεδιασμό εφαρμογών, ειδικά υπηρεσιών στο διαδίκτυο μέσω της επέκτασης expressjs. Όλες οι υπηρεσίες χρησιμοποιούν μηνύματα JSON για την επικοινωνία μεταξύ των υπηρεσιών τους και είναι εύκολο να αποκωδικοποιηθούν μέσω της javascript/node.js και της επέκτασης (Middleware) body-parser. Η βιβλιοθήκη mongoose, είναι ένα σύστημα ORM για την βάση δεδομένων nosql mongodb. Τέλος η βιβλιοθήκη axios είναι μια βιβλιοθήκη ασύγχρονης κλήσης υπηρεσιών και http endpoints για την επίσπευση των διαδικασιών ανάκτησης πληροφοριών από αυτές.

4.3.4 MongoDB

Για την βάση δεδομένων χρησιμοποιήθηκε η mongodb [72]. Η mongodb είναι μια nosql βάση δεδομένων η οποία αποθηκεύει έγγραφα (documents) αντί για πίνακες. Είναι ταχύτατη σε μεγάλες αιτήσεις δεδομένων και μπορεί εύκολα να κλιμακωθεί (scale up) εάν υπάρχει μεγάλη ζήτηση πόρων στο σύστημα και εφόσον υπάρχει αντίστοιχη υλοποίηση.

4.3.5 Δημοσίευση του διαλογικού βοηθού στο slack

Το slack αποτελεί μια μετεξέλιξη του IRC στοχευμένο στην συνεργασία ατόμων μέσα σε ομάδες είτε οργανισμούς είτε ατομικά, κάτω από μία εφαρμογή web. Οι προγραμματιστές του slack παρέχουν διαρκώς νέες καινοτομίες σχετικά με την επικοινωνία, όπως ενσωμάτωση υπηρεσιών φωνής, βίντεο και εμπλουτισμένου κειμένου. Τελευταία, έχουν ανοίξει το προγραμματιστικό τους μοντέλο με την χρήση REST προεκτάσεων και βιβλιοθήκες προγραμματισμού API συνδέοντας τη πλατφόρμα με άλλες υπηρεσίες και εφαρμογές διαδικτύου. Ένας τομέας αυτών των υπηρεσιών είναι και οι διαλογικοί πράκτορες. Έτσι, το καθιστά ιδανικό για την στέγαση διαφόρων ψηφιακών βοηθών γενικού και ειδικού περιεχομένου.



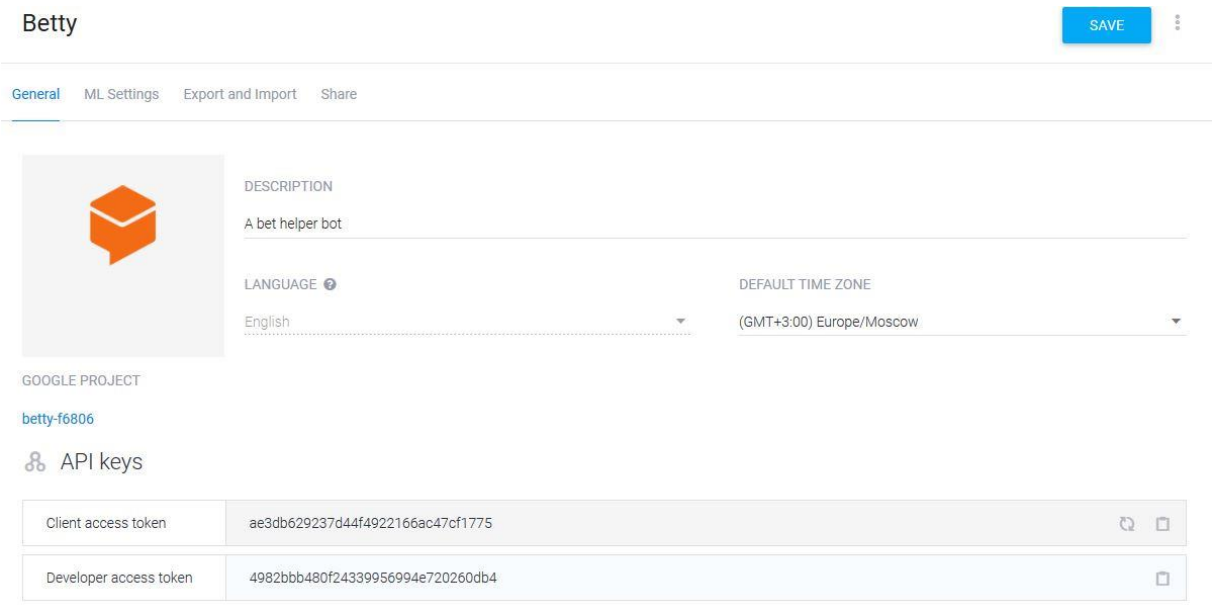
Εικόνα 4.2 οθόνη slack

4.4 Δημιουργία διαλογικού πράκτορα στο api.ai

Πριν αναλύσουμε τις ενέργειες χρήστη, τις οντότητες και το προγραμματιστικό κομμάτι του διαλογικού βοηθού, πρέπει να δημιουργηθεί ο αντίστοιχος διαλογικός πράκτορας στο διαχειριστικό σύστημα του api.ai.

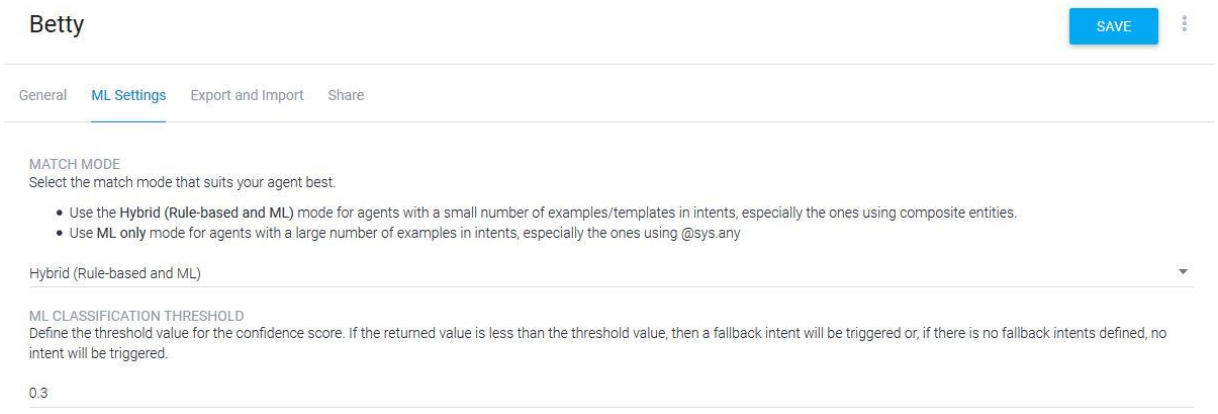
4.4.1 Δημιουργία πράκτορα

Η δημιουργία πράκτορα έγινε με τα εξής στοιχεία:



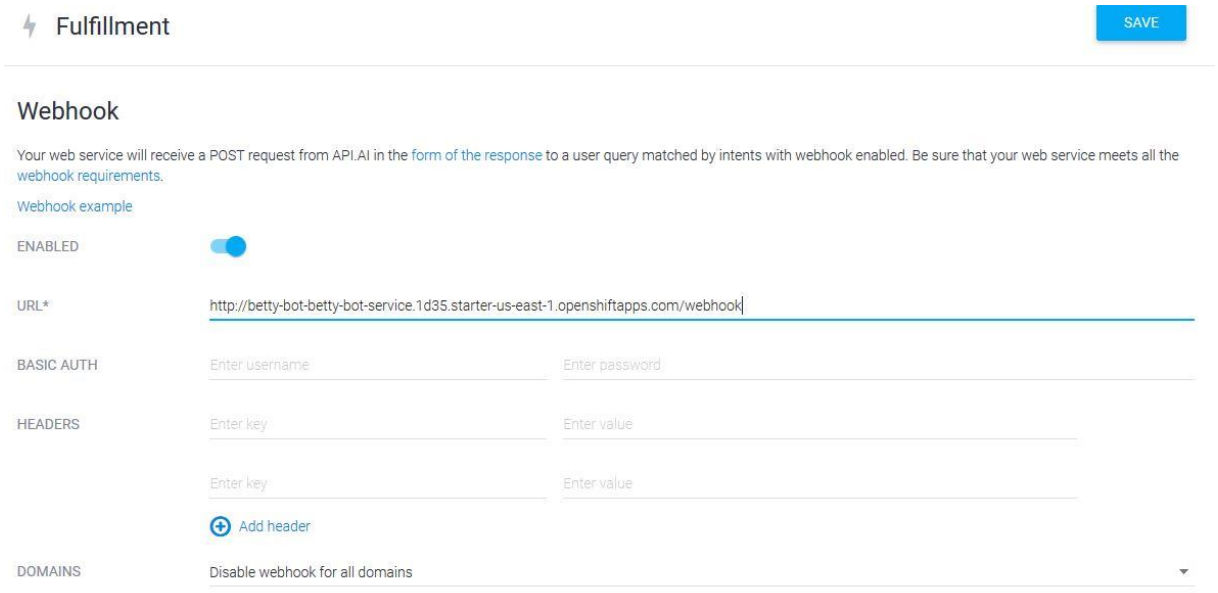
Εικόνα 4.3 Επιλογές διαλογικού πράκτορα

Σαν όνομα χρησιμοποιείται εσωτερικά το όνομα Betty. Χρησιμοποιώντας ένα όνομα δίνει μια ψυχαγωγική αίσθηση στο διαλογικό βοηθό ειδικά σε θέματα ρύθμισης μικρόκουβέντας (small talk). Άλλα εναλλακτικά ονόματα που θα μπορούσαν να χρησιμοποιηθούν είναι το BetHelperAgent ή κάτι παρεμφερές.



Εικόνα 4.4 Ρυθμίσεις μηχανικής μάθησης

Εδώ φαίνονται οι ρυθμίσεις μηχανικής μάθησης που χρησιμοποιήθηκαν. Η τιμή ML CLASSIFICATION THRESHOLD σημαίνει ότι στην αναγνώριση οντοτήτων και σκοπών χρήστη αν η τιμή που αποδώσει το σύστημα είναι μικρότερη από 0.3 τότε δεν θα συμβεί τίποτα σαν ενέργεια ή θα ενεργοποιηθούν άσχετες εκφράσεις στο διαλογικό βοηθό «Huh?» «This has no sense» κτλ.



Εικόνα 4.5 Χρήση server για δημοσίευση του διαλογικού βοηθού.

Στα πλαίσια της μεταπτυχιακής διατριβής και για τις ανάγκες των καναλιών διαλογικών πρακτόρων, ο διαλογικός βοηθός πρέπει να μεταφορτωθεί σε ένα κανονικό

εξυπηρετητή ώστε να φαίνεται ενεργός. Για αυτό το σκοπό χρησιμοποιήθηκε η υπηρεσία νέφους openshift της redhat στην οποία μπορεί να δημοσιευτεί μια μικρή εφαρμογή node.js με περιορισμό την χρήση 1 gb και μέχρι 2 επεξεργαστές.

4.5 Υλοποίηση σκοπών χρήστη (intents)

Ο πράκτορας σχεδιάστηκε με γνώμονα την εμπειρία του χρήστη πρώτα. Έτσι τμήματα του σχεδιασμού αφορούν μικροαλληλεπιδράσεις με τον χρήστη (small talk, χαιρετισμοί, βοήθεια) και τμήματα αφορούν την επιχειρησιακή λογική δηλαδή την προβολή αγώνων (events) για την πραγματοποίηση στοιχημάτων.

4.5.1 Μικροκουβέντες

Μέρος του διαλογικού βοηθού είναι η αλληλεπίδραση ώστε να μπορέσει να μην φαίνεται μηχανική η συμπεριφορά του. Έτσι μέσω της επιλογής small talk του api.ai εμπλουτίστηκε με μικρές ερωτήσεις οι οποίες αφορούν την κοινωνική αλληλεπίδραση με τον χρήστη. Αυτές οι αλληλεπιδράσεις δεν αποτελούν κάποιο σκοπό χρήστη αλλά συγκαταλέγονται μέσα σε αυτούς. Άλλες αλληλεπιδράσεις αφορούν την απάντηση στον χρήστη όταν πληκτρολογήσει «Hello» ή «Hi». Αυτές αφορούν χαιρετισμούς όπως στην φυσιολογική ζωή.

4.5.2 Βοήθεια χρήστη

Ένα τμήμα του διαλόγου, αφορά την καθοδήγηση του χρήστη ως εισαγωγή στις επιχειρησιακές επιλογές που έχει σχεδιαστεί ο διαλογικός βοηθός. Αυτό επιτυγχάνεται μέσω της αναζήτησης βοήθειας. Έτσι αν ο χρήστης ζητήσει βοήθεια:

Χρήστης: Help
Chatbot: Hi, I am Betty your bet assistant. Here you can:
ask about a bet type (or show you the whole list to choose)
see some events
ask about your winnings
What would you like to do?

Παράδειγμα 4.1 Αναζήτηση βοήθειας από το διαλογικό βοηθό

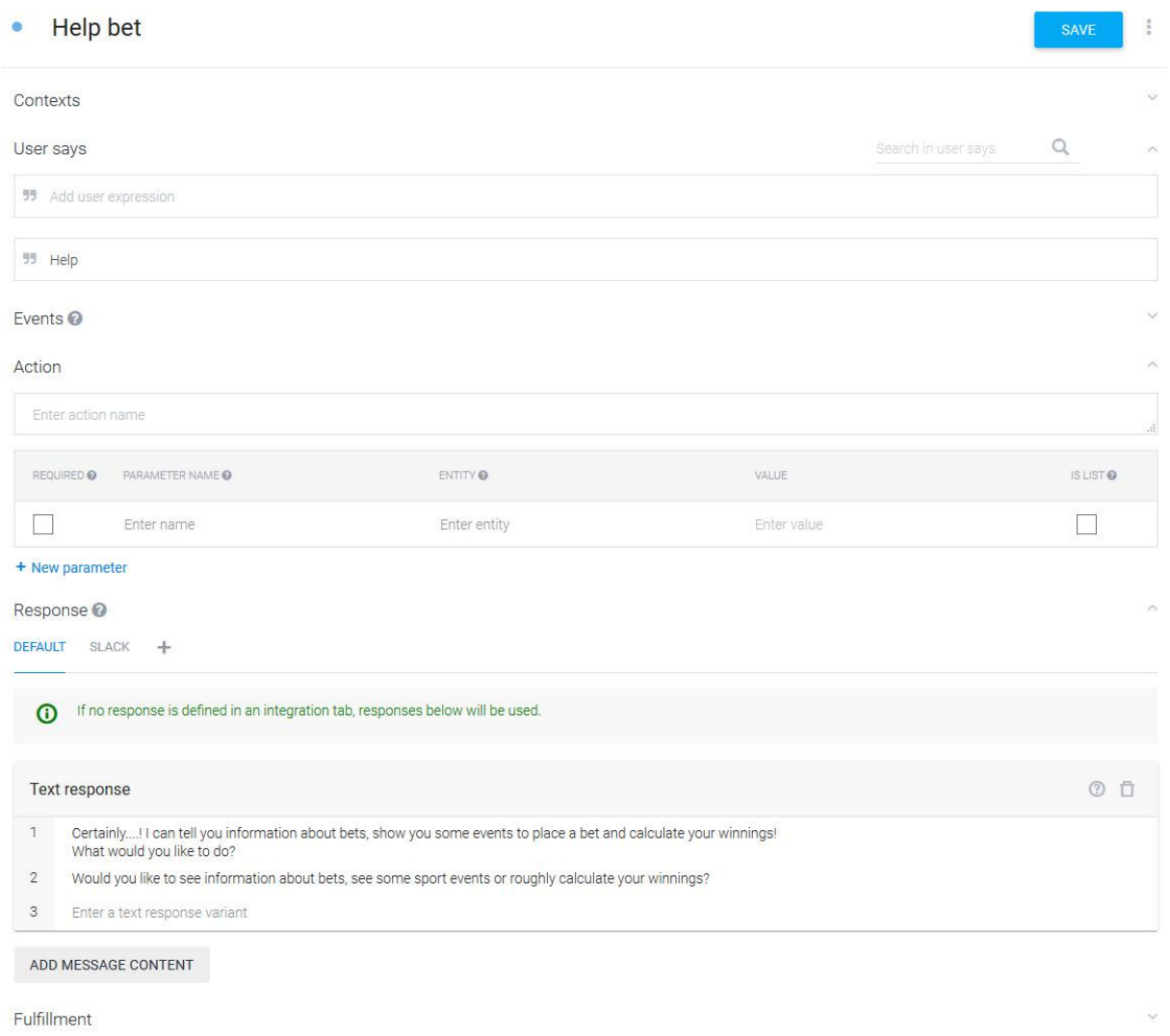
Ο διαλογικός βοηθός αναλαμβάνει να καθοδηγήσει τον χρήστη στις κατάλληλες επιλογές. Οι επιλογές αυτές είναι:

- Προβολή των τύπων στοιχήματος και επερώτηση για αυτά
- Προβολή των αθλητικών γεγονότων
- Προβολή των πιθανών κερδών

Στο τέλος της βοήθειας, ο διαλογικός βοηθός ρωτάει τον χρήστη τι επιλογή θέλει να κάνει. Σε αυτή τη φάση της υλοποίησης η επιλογή δεν είναι υποχρεωτική οπότε δεν χρειάζεται να ρωτήσει κάτι αυτή τη στιγμή.

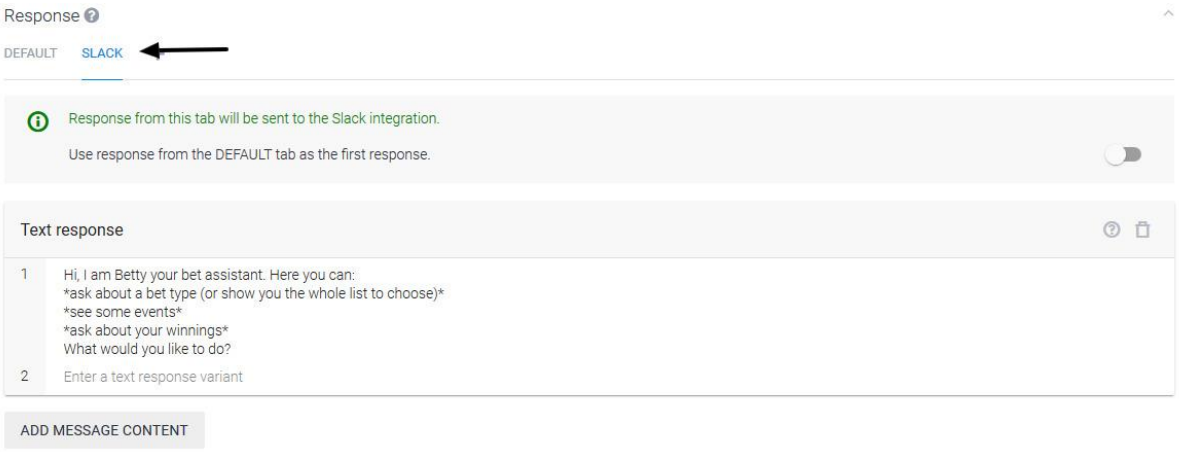
Υλοποίηση στο διαχειριστικό του api.ai

Για να υλοποιήσουμε την συγκεκριμένη ενότητα στο api.ai χρησιμοποιήθηκε η επιλογή «σκοπός χρήστη». Η μόνη έκφραση που διατυπώθηκε στο συγκεκριμένο σκοπό είναι η λέξη help. Μιας που ο σκοπός αυτός δεν χρησιμοποιεί κάποια εισαγωγή πληροφοριών από βάση δεδομένων ή οντότητες, ο ορισμός ήταν πιο εύκολος:



Εικόνα 4.6 Ορισμός της βοήθειας χρήστη

Αυτό που χρησιμοποιήθηκε σε αυτό το παράδειγμα είναι η χρήση της μορφοποίησης για κανάλια διαλογικών πρακτόρων. Έτσι οι αποκρίσεις που έχουμε από την υπηρεσία σαν default αποκρίσεις φαίνονται στην παραπάνω εικόνα, ενώ ορίστηκαν καλύτερες αποκρίσεις για την υπηρεσία slack:



Εικόνα 4.7 Διαφορετική απόκριση της υπηρεσίας σε slack

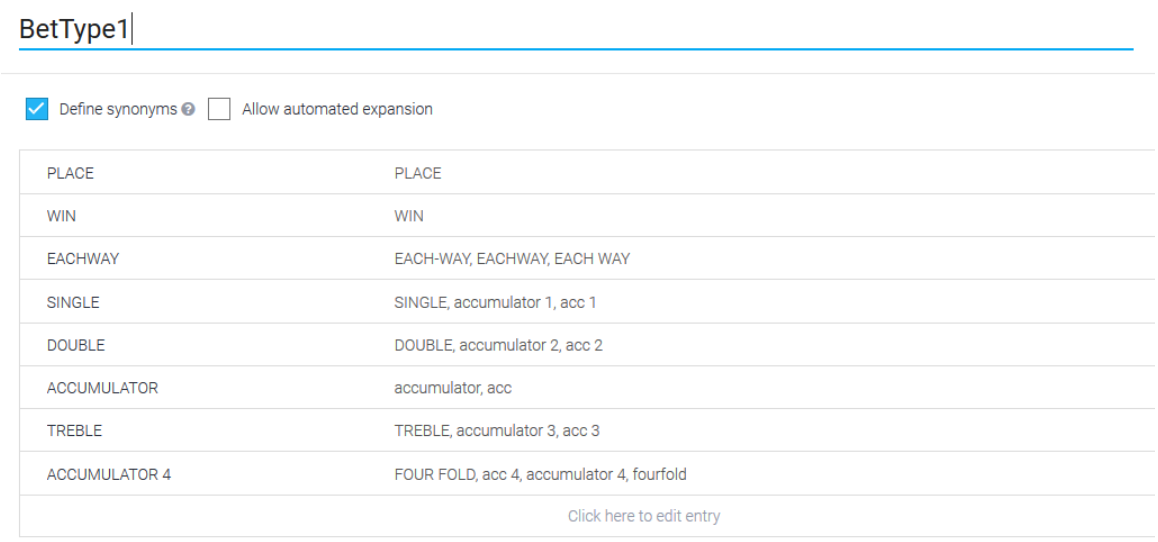
Η διαφορά που παρέχεται στο slack είναι ότι περιέχει μορφοποιήσεις εμπλουτισμένου κείμενου που αναγνωρίζονται από το slack, όπως το σύμβολο * που σημαίνει έντονα γράμματα.

4.5.3 Πληροφορίες στοιχήματος Ι

Στις πληροφορίες στοιχήματος ο σκοπός του χρήστη είναι η αναζήτηση ενός τύπου στοιχήματος με το όνομα του. Έτσι αν ο χρήστης ρωτήσει «I want to know about a place bet» τότε η μηχανή κατανόησης θα αναγνωρίσει ότι αυτό ανήκει στην πληροφορία στοιχήματος για το PLACE στοίχημα. Μια άλλη επερώτηση είναι «What is an accumulator bet?». Σε αυτή τη περίπτωση ο διαλογικός βοηθός πρέπει να επιστρέψει τις πληροφορίες σχετικά με το single ή accumulator bet.

Υλοποίηση στο διαχειριστικό του api.ai

Η υλοποίηση στο διαχειριστικό του api.ai του παραπάνω σκοπού χρήστη έγινε σε δύο στάδια. Πρώτα ορίστηκαν οι βασικοί τύποι οντοτήτων για τους τύπους στοιχήματος. Εφόσον οι οντότητες αυτές είναι εξειδικευμένες και όχι κοινές σε διάφορα έργα για να υπάρχουν σαν οντότητες συστήματος, έπρεπε να οριστούν από το χρήστη. Στην επιλογή «Οντότητες», δημιουργήθηκε ενας νέος τύπος με το όνομα BetType1. Το api.ai εισάγει τις οντότητες αυτές σαν **@BetType1**:

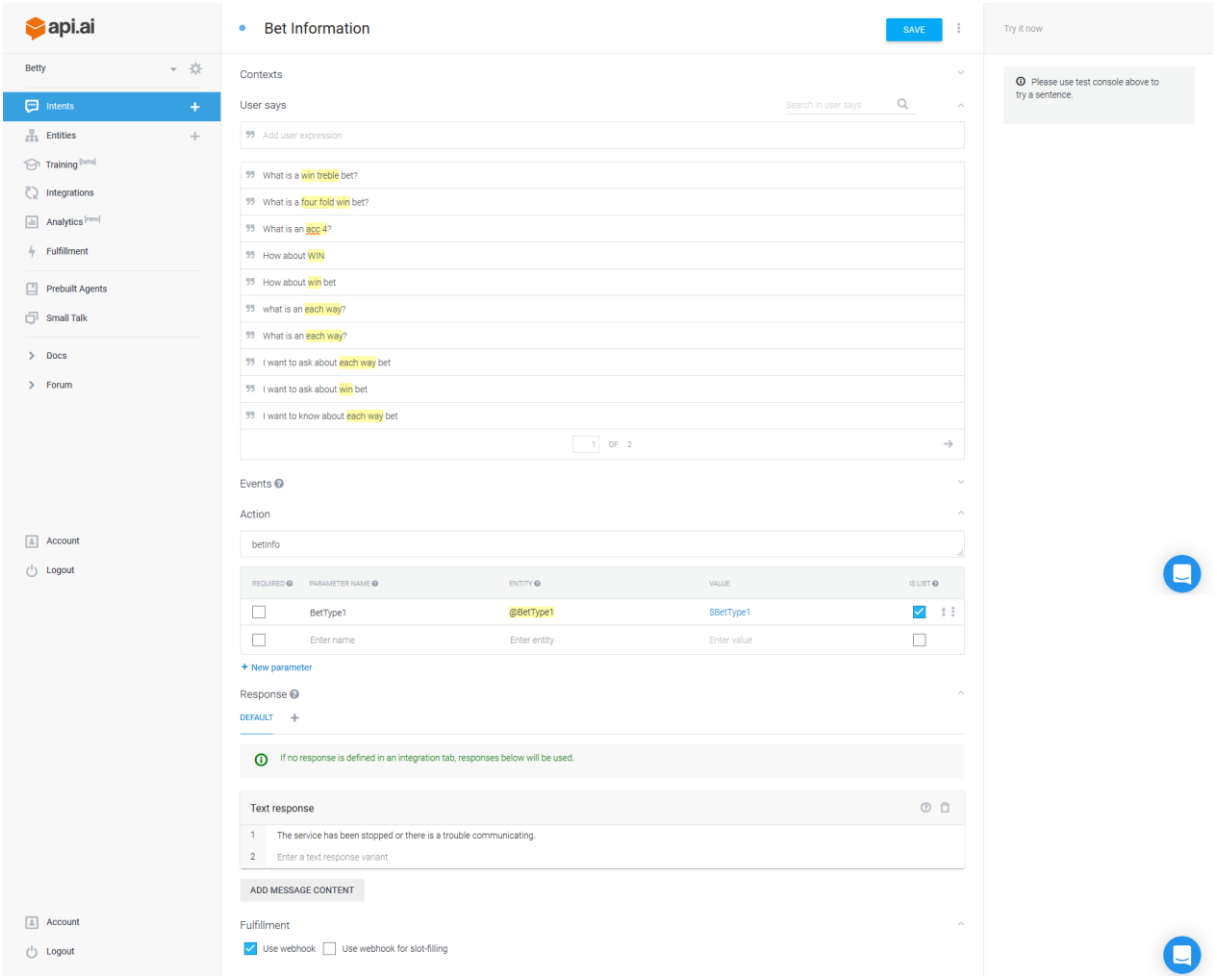


Εικόνα 4.8 Ορισμός τύπων στοιχήματος

Οι τιμές στις οντότητες περνάνε ως key-value. Αυτό σημαίνει ότι η αριστερή στήλη είναι το κλειδί, ενώ η δεξιά στήλη είναι η τιμή που μπορεί να δεχτεί η οντότητα. Με την ενεργοποίηση του «define synonyms», μπορούμε να εισάγουμε συνώνυμα στις τιμές της οντότητας, όπως συντομεύσεις για τα accumulator ή άλλες λέξεις. Κάθε φορά που θα εισάγουμε στην έκφραση μας ένα κείμενο π.χ. four fold τότε η μηχανή κατανόησης θα προσπαθήσει να βρει το αντίστοιχο κλειδί της οντότητας και θα το περάσει σαν τιμή στην παράμετρο @BetType1.

Το σκεπτικό είναι ότι όταν πληκτρολογήσει ο χρήστης «I want to know about four fold bet», Η τιμή που θα αποκτήσει η οντότητα @BetType1 θα είναι το «ACCUMULATOR 4». Αυτό θα είναι το κλειδί μετέπειτα για να βρει τη κατάλληλη εγγραφή στη βάση δεδομένων (βλ. παρακάτω).

Επόμενο στάδιο είναι η δημιουργία του σκοπού του χρήστη για την πληροφορία του στοιχήματος. Στην οθόνη του διαχειριστικού «σκοποί χρήστη» (intents) δημιουργούμε τον εξής σκοπό χρήστη:



Εικόνα 4.9 Σκοπός χρήστη: Bet Information (Πληροφορία χρήστη)

Τέλος, δηλώνουμε ότι ο σκοπός χρήστη θα χρησιμοποιηθεί μέσω webhook. Αυτό σημαίνει ότι δεν θα επεξεργαστεί μόνο από το διαχειριστικό σύστημα του api.ai αλλά θα περιμένει να εκπληρωθεί η ενέργεια μέσω της υπηρεσίας ανεύρεσης τύπων στοιχήματος και περιμένει πίσω ως έκφραση, ένα κείμενο γραμμένο σε φυσική γλώσσα.

Υλοποίηση στη βάση δεδομένων

Στην mongodb η συλλογή εγγράφων που κρατά τις πληροφορίες για τους τύπους στοιχημάτων ονομάζεται **BetType**. Κάθε έγγραφο αποτελείται από δυναμικά πεδία τα οποία ορίζονται για τις ανάγκες των τύπων στοιχήματος. Ένας από τους λόγους που ενσωματώθηκαν οι τύποι στοιχήματος και οι μορφές στοιχήματος είναι για να προβληθεί αυτή η ευελιξία που παρέχει η mongodb.

Το σχήμα του εγγράφου ορίζεται ως εξής:

```
{
    "_id" : ObjectId("5921bd30c3dd93cc9536d24d"),
    "title" : "Place Bet",
    "code" : "PLACE",
    "description" : "A bet that produces a return only if the selection finishes first or within a predetermined number of positions (places) of the winner of an event. The return is often based on a fixed proportion of the win odds of the selection."
}
```

Κώδικας 4.1 Σχήμα ενός εγγράφου bet type (μορφή στοιχήματος)

Ενώ ένα έγγραφο τύπου στοιχήματος ορίζεται ως:

```
{
    "_id" : ObjectId("5921d4c6c3dd93cc9536d789"),
    "title" : "Treble bet",
    "code" : "TREBLE",
    "description" : "A bet on three selections; all three of which must win to gain a return.",
    "selections" : 3
}
```

Κώδικας 4.2 Σχήμα εγγράφου bet type (τύπος στοιχήματος)

Τα πεδία του σχήματος και οι τύποι δεδομένων τους καθώς και η χρήση τους παρουσιάζονται στο παρακάτω πίνακα:

Πεδίο	**Τύπος**	**Λειτουργία**
_id	ObjectId	Αναγνωριστικό εγγράφου
title	String	Τίτλος τύπου στοιχήματος
code	String	Αναγνωριστικό τύπου στοιχήματος
description	String	Περιγραφή του τύπου στοιχήματος
selections	Int32	Αριθμός επιλογών μέσα στο στοίχημα (μόνο για τύπο στοιχημάτων)

Πίνακας 4.1 σχήμα BetType

Όταν προγραμματιστικά η υπηρεσία θα προσπαθήσει να αναζητήσει ένα έγγραφο μέσα από την συλλογή BetType θα αναζητήσει το έγγραφο μέσω του πεδίου code. Το πεδίο code έχει τις εγγραφές του στα κεφαλαία ώστε να ταιριάζουν μαζί με τις τιμές που παίρνουν οι οντότητες όταν μετατρέπονται σε παράμετροι κατά την αποστολή τους

στο λογισμικό του διαλογικού πράκτορα. Μια τυπική αναζήτηση σε mongoose είναι η εξής:

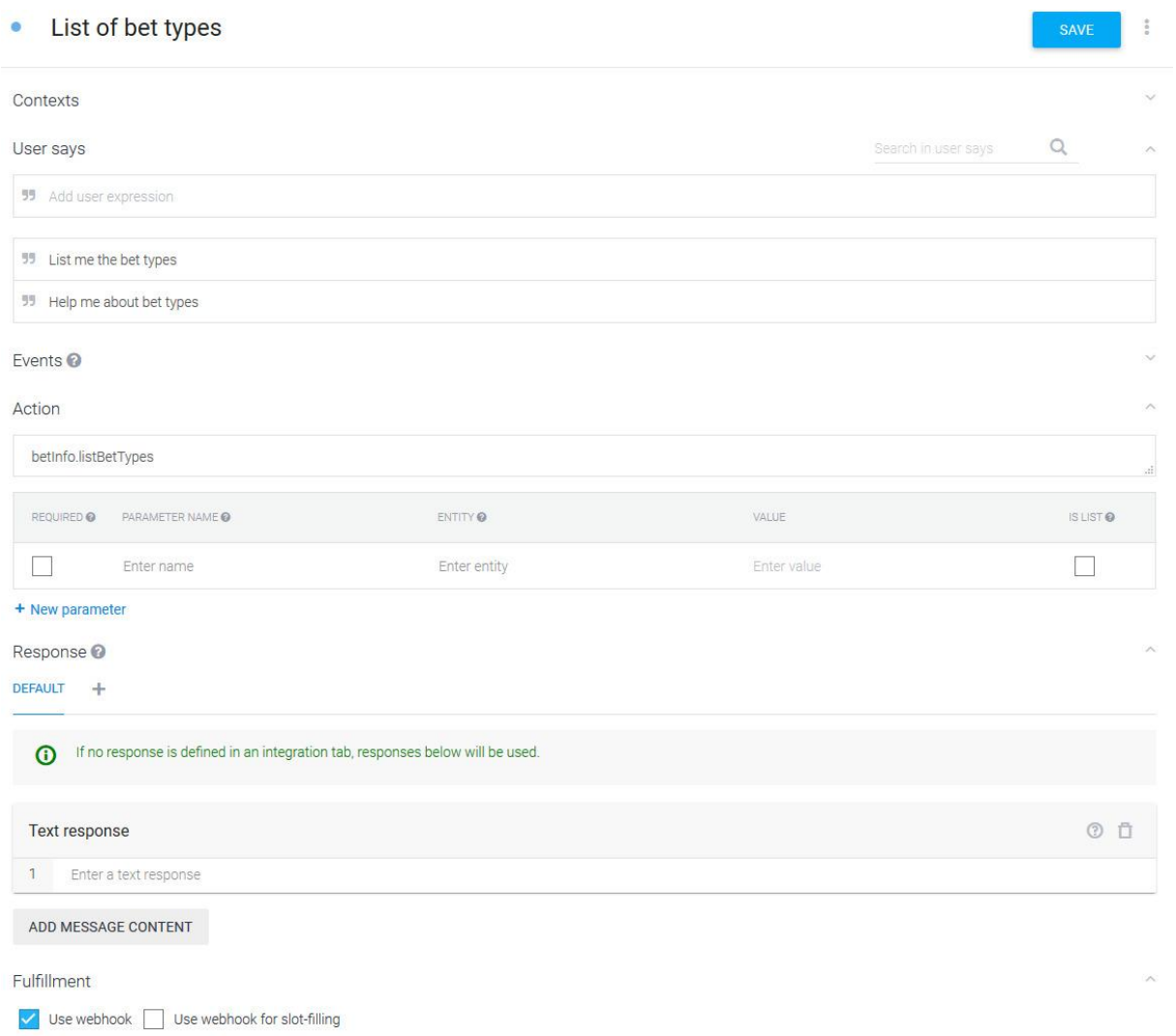
```
// inside /webhook
var bettype1 = req.body.result.parameters.BetType1;
BetType.find({ code: bettype1 }).then((docs) => {
	docs.each((item) => {
		item = `${item.title} (code: ${item.code}): ${item.description}`;
	});
	res.send(item);
});
```

Κώδικας 4.3 τυπική ερώτηση κώδικα mongodb

Το παραπάνω κομμάτι κώδικα θα ανακτήσει από τα δεδομένα POST που στέλνει η εκπλήρωση /webhook στο πρόγραμμα του διαλογικού βοηθού την τιμή της μεταβλητής BetType1 (δηλαδή την τιμή που έχει η @BetType1, π.χ. PLACE), έπειτα θα κάνει ερώτηση στο μοντέλο BetType της βιβλιοθήκης Mongoose να βρει όλες τις εγγραφές με τον κώδικα που έχει το bettype1 (δηλαδή το PLACE) και για κάθε ένα docs που επιστρέφει να δημιουργήσει μια νέα μορφοποίηση (παραγωγή φυσικής γλώσσας). Στο τέλος επιστρέφεται στο api.ai η νέα συμβολοσειρά που έχει τις πληροφορίες για το συγκεκριμένο Bet Type.

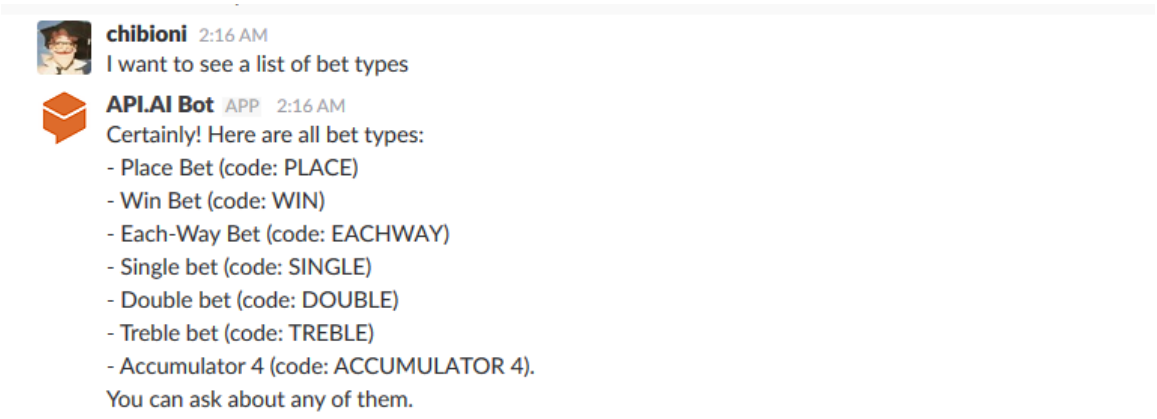
4.5.4 Πληροφορίες στοιχήματος ΙΙ

Ένα δεύτερο τμήμα βοήθειας που προσφέρεται στο χρήστη είναι η ονομαστική λίστα των στοιχημάτων και οι κωδικοί που είναι εγγεγραμμένοι μέσα στην βάση δεδομένων. Το βήμα αυτό προστέθηκε όταν η εμπειρία του χρήστη κατά την αναζήτηση πληροφοριών στοιχήματος δεν ήταν η αναμενόμενη, μιας που ένας νέος χρήστης που δεν γνωρίζει από στοιχήματα δεν μπορεί να ζητήσει πληροφορίες για αυτά. Έτσι δημιουργήθηκε ενας νέος σκοπός χρήστη ο οποίος παρουσιάζει μια λίστα με τα στοιχήματα του χρήστη.



Εικόνα 4.10 Ορισμός της λίστας τύπων στοιχήματος

Στη συγκεκριμένη περίπτωση δεν χρειαζόμαστε νέες οντότητες μιας που έχουν οριστεί από το σκοπό χρήστη «Bet information» (@BetType1). Για την δημιουργία του νέου σκοπού «List of Bet Types» χρησιμοποιήθηκαν εκφράσεις οι οποίες δεν απαιτούν οντότητες. Αυτό που χρειάζονται όμως είναι σύνδεση με την βάση δεδομένων, άρα θα πρέπει να κάνουν κλήση στο endpoint /webhook του διαλογικού βοηθού. Ως ενέργεια χρήστη ορίστηκε η ετικέτα **betInfo.listBetTypes**.



Εικόνα 4.11 συγκεντρωτική λίστα στοιχημάτων μέσα από το slack

Προγραμματιστικά, εργαζόμαστε όπως στο προηγούμενο παράδειγμα της ανάκτησης ενός από τους τύπους στοιχήματος, όμως τώρα αυτό που αλλάζει είναι ότι δεν χρησιμοποιείται η παράμετρος code:

```
// inside /webhook
BetType.find().then((docs) => {
	docs.each((item) => {
		item = `${item.title} (code: ${item.code}): ${item.description}`;
	});
	res.send(item);
});
```

Κώδικας 4.4 συγκεντρωτική λίστα στοιχημάτων.

4.5.5 Διεξαγωγή στοιχήματος

Οι προαπαιτούμενες παράμετροι για να βάλει κάποιος ένα στοίχημα είναι δύο: το ποσό που θα διαθέσει για το στοίχημα και η επιλογή (selection/outcome) ενός αθλητικού αγώνα. Η επιλογή ποικίλλει ανάλογα με την αγορά του στοιχήματος. Από τους τύπους στοιχήματος γνωρίζουμε ότι υπάρχουν διάφορα είδη στοιχήματος όπως double treble κτλ. Λόγω του τρόπου υλοποίησης τους δεν χρησιμοποιήθηκαν στην τρέχουσα μεταπτυχιακή διατριβή. Έτσι θεωρούμε ότι κάθε στοίχημα αντιστοιχεί σε ένα αγώνισμα, με ένα ποσό διάθεσης x, το οποίο μπορεί να είναι σε οποιαδήποτε αναγνωρισμένο νόμισμα (ποσό + τύπος νομίσματος). Άρα όλα τα στοιχήματα είναι τύπου WIN και Single. Επίσης οι αγώνες είναι αγώνες μπάσκετ και ποδοσφαίρου (basketball / football).

Αναζήτηση αγώνων

Το ποσό διάθεσης στους αγώνες είναι μια προσωπική επιλογή, όμως ο χρήστης εάν δεν γνωρίζει την τυποποίηση που παρέχει κάθε σύστημα στοιχήματος θα έχει πρόβλημα στην διατύπωση της έκφρασης που θα εκπληρώσει το σκοπό του, δηλαδή την διεκπεραίωση ενός στοιχήματος. Έτσι το πρώτο βήμα σαν εμπειρία χρήστη, είναι να δει μια λίστα από αγωνίσματα. Σε μια τυπική εφαρμογή στοιχημάτων μέσω web, συμβαίνει ακριβώς η ίδια κατάσταση. Ο χρήστης καθώς προηγείται σε ενα website στοιχημάτων το πρώτο πράγμα που παρατηρεί είναι η λίστα των αγώνων, οι αποδόσεις (odds) και οι αναθέσεις των αποδόσεων σε ομάδες ή εκβάσεις αγώνα ανάλογα με την αγορά στοιχήματος που έχει ρυθμιστεί.

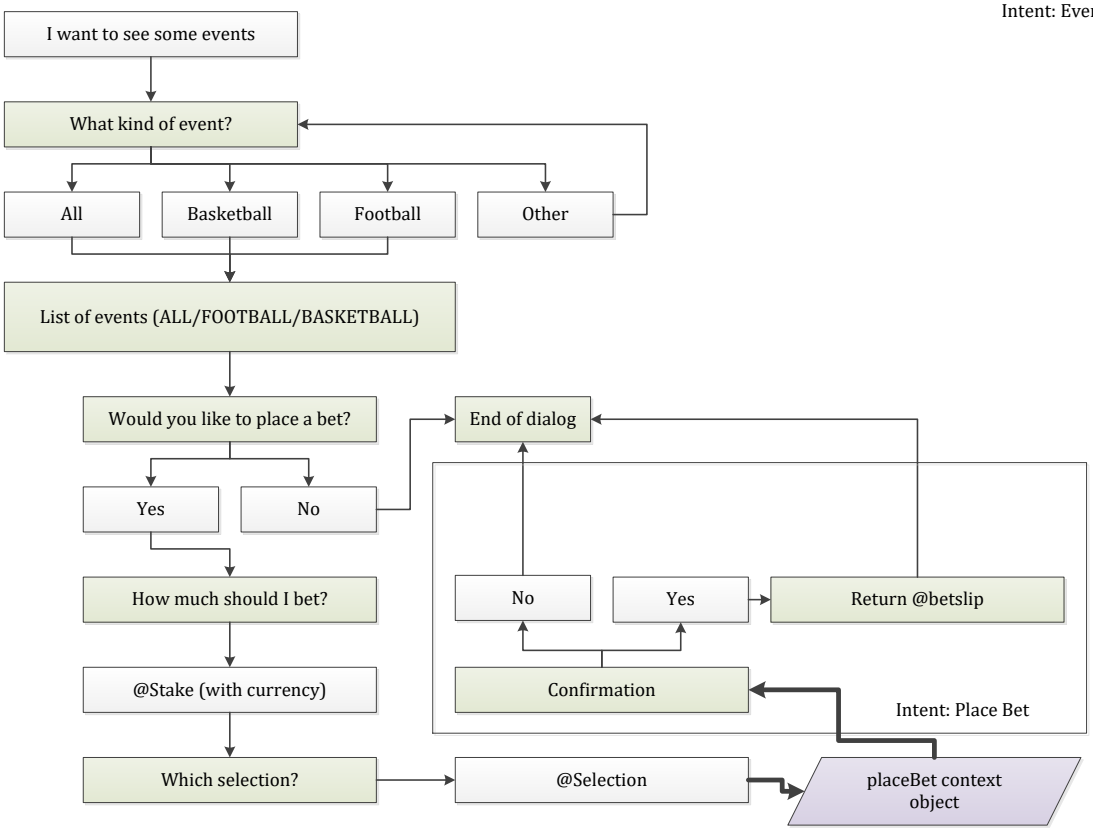
Έτσι σαν πρώτη εισαγωγή στη διεκπεραίωση του στοιχήματος είναι ο διαλογικός πράκτορας να του εμφανίσει μια λίστα από αθλήματα στα οποία μπορεί να ποντάρει μαζί με τις αποδόσεις τους. Η μορφή απάντησης που επιλέχθηκε για αυτό το σκοπό είναι η εξής:

```
Τίτλος αγώνα
    Ομάδα #1 (απόδοση)    Κωδικός επιλογής (selection)
    Ισοπαλία (απόδοση)    Κωδικός επιλογής (selection)
    Ομάδα #3 (απόδοση)    Κωδικός επιλογής (selection)
```

Κώδικας 4.5 προτεινόμενη παραγωγή κειμένου για αγωνίσματα

Ροή διαλόγου στοιχήματος

Η ροή της διεκπεραίωσης ενός στοιχήματος παρουσιάζεται στην παρακάτω εικόνα. Ο χρήστης αναζητεί πρώτα το αγώνισμα που τον ενδιαφέρει. Ο διαλογικός βοηθός έπειτα τον ρωτάει εάν θέλει να ποντάρει σε κάποιο από αυτά τα αγωνίσματα. Εδώ υπάρχει μια διακλάδωση η οποία μπορεί να τερματίσει τον διάλογο ή να συνεχίσει στο επόμενο στάδιο το οποίο είναι η διεξαγωγή ερωτήσεων slot-filling. Οι ερωτήσεις slot filling έχουν ως σκοπό την εξασφάλιση ότι τα στοιχεία που θα εισάγει ο χρήστης είναι σωστά. Σε ένα πραγματικό σύστημα, λόγω της συναλλαγής με χρήματα πρέπει να υπάρχουν όλες οι κατάλληλες δικλείδες ασφαλείας για αυτό το σκοπό. Τέλος ο διαλογικός βοηθός ρωτάει την επιβεβαίωση πριν αποστείλει τα δεδομένα στην υπηρεσία bet placement. Η απάντηση από την υπηρεσία είναι ένα betslipid το οποίο είναι το κλειδί για την αναζήτηση κερδών.

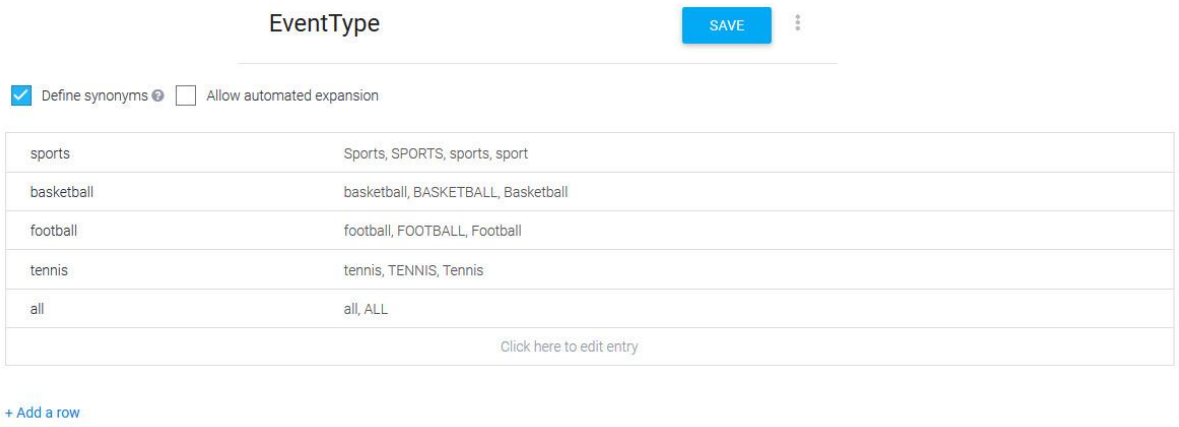


Εικόνα 4.12 Η ροή της διεκπεραίωσης στοιχήματος

Υλοποίηση στο διαχειριστικό του api.ai - οντότητες

Η υλοποίηση του παραπάνω σχήματος στο διαχειριστικό του api.ai θα ακολουθήσει την φυσιολογική ροή που εργαστήκαμε έως τώρα. Πρώτα θα οριστούν οι επιμέρους οντότητες, έπειτα οι σκοποί του χρήστη και τέλος η εκπαίδευση των μοντέλων με νέες εκφράσεις αν χρειαστεί.

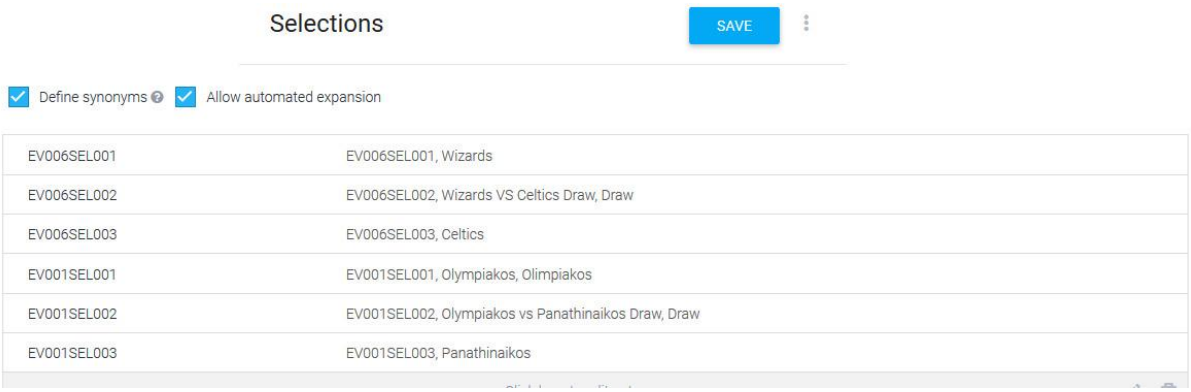
Οι οντότητες που λαμβάνουν μέρος σε αυτή τη διαδικασία είναι ο τύπος του αθλητικού γεγονότος @EventType και ο τύπος των επιλογών @Selection. Επίσης η μορφή του αποκόμματος του στοιχήματος (betslip) παρουσιάζει ένα ενδιαφέρον στην αναγνώριση του από το διαλογικό βοηθό οπότε θα ορίσουμε και για αυτό μια ξεχωριστή οντότητα.



Εικόνα 4.13 Ορισμός της @EventType οντότητας στο api.ai

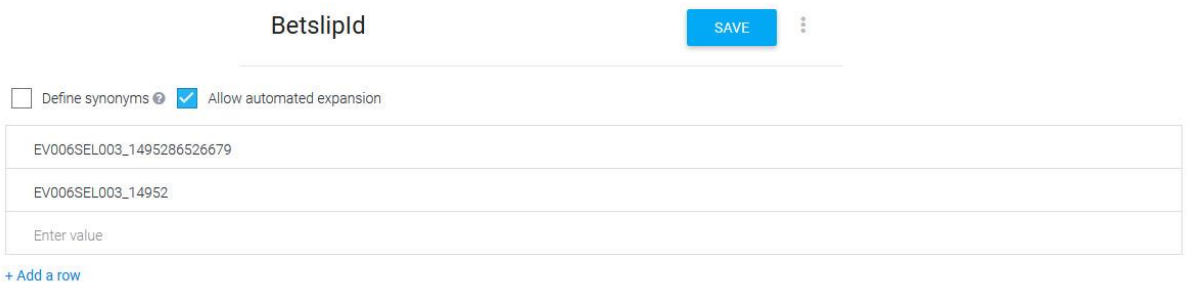
Η οντότητα @EventType περιέχει τις μεταβλητές κλειδιά με μικρά γράμματα. Στην αναζήτηση από την βάση δεδομένων αυτό το κλειδί θα μετατραπεί σε κεφαλαία μέσω κώδικα. Επίσης υπάρχουν δύο επιπλέον τιμές στην οντότητα, η τιμή all και η τιμή sports. Χρησιμοποιούνται ως λέξεις σχετικές με την προβολή όλων των events.

Για τον ορισμό της οντότητας Selections (επιλογών αγώνων), θα χρησιμοποιηθεί μια πειραματική λειτουργία του api.ai. Η αυτόματη επέκταση των τιμών των ονομαστικών οντοτήτων (Automated expansion).



Εικόνα 4.14 Ορισμός επιλογών αγωνισμάτων

Τέλος η επιλογή αυτή θα χρησιμοποιηθεί και για το @Betslipid. Αυτό που προσπαθούμε να επιτύχουμε με την επιλογή αυτόματης επέκτασης είναι μια αυτόματη αναγνώριση παρεμφερών τύπων κωδικών. Σε περίπτωση που δεν είναι επαρκές αυτό για να αναγνωριστεί ο κωδικός του γεγονότος π.χ., χρησιμοποιείται η επιλογή «εκπαίδευση» από το διαχειριστικό σύστημα για να αποσαφηνιστούν οι λέξεις και οι τύποι κωδικού.

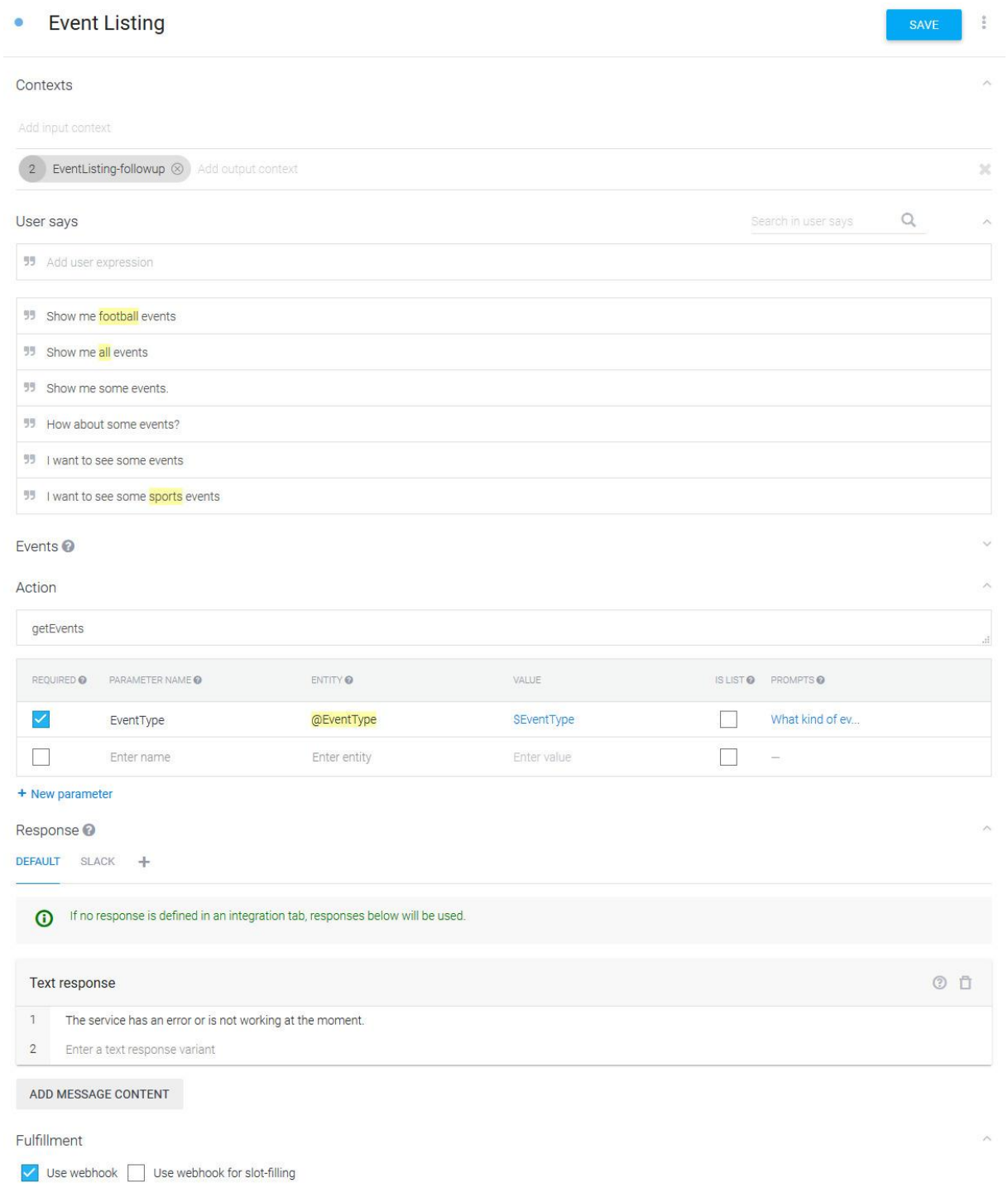


Εικόνα 4.15 Αναγνώριση τύπων betslip

Υλοποίηση στο διαχειριστικό api.ai – Σκοποί χρήστη

Για την διεκπεραίωση του στοιχήματος, έχουν οριστεί 4 σκοποί χρήστη. Πρώτα από όλα είναι η αναζήτηση των αθλητικών αγώνων. Ο χρήστης ύστερα από την προτροπή της βοήθειας ρωτά τον διαλογικό βοηθό ότι θέλει να δει τη λίστα των αγωνισμάτων για να αρχίσει ο διάλογος. Σε περίπτωση που δεν έχει ορίσει κάποιο αγώνισμα ή δεν έχει δηλώσει ότι θέλει να δει όλους τους αγώνες τότε ο διαλογικός βοηθός ρωτάει για ποιο αγώνα θέλει να δει τα αγωνίσματα. Οι τρέχουσες πληροφορίες αποθηκευμένες στη βάση δεδομένων έχουν αγωνίσματα σχετικά με ποδόσφαιρο και μπάσκετ.

Εφόσον έχει οριστεί η οντότητα @EventType τότε μπορεί να αρχίσει η διαδικασία ερωτήσεων για την διεκπεραίωση του στοιχήματος. Αρχικά ο διαλογικός βοηθός ρωτάει το χρήστη αν θέλει να βάλει ένα στοίχημα. Έπειτα τον ρωτάει πόσο θα ποντάρει και μετά σε ποια επιλογή. Λόγω της σχέσης επιλογής με αγωνίσματα, η ερώτηση του στοιχήματος γίνεται μέσα στον ίδιο σκοπό χρήστη – τη λίστα των αγωνισμάτων. Για την ερώτηση αυτή χρησιμοποιούνται δύο ειδικοί σκοποί χρήστη, οι επακόλουθοι σκοποί χρήστη (follow-up intents). Ο ένας θα διευθετήσει την αρνητική απάντηση (τερματισμός διαλόγου) και ο άλλος την θετική απάντηση στην οποία θα εισάγει τα δεδομένα σε ένα αντικείμενο context με την ονομασία placeBet. Ο ορισμός του πρώτου σκοπού χρήστη – Event Listing παρουσιάζεται στην επόμενη εικόνα:

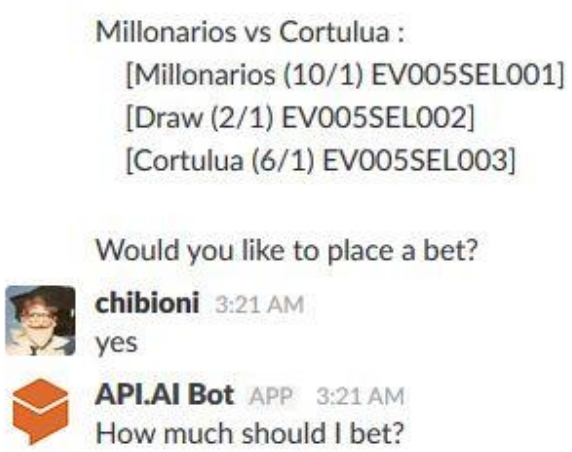


Εικόνα 4.16 Ορισμός του αρχικού event listing σκοπού χρήστη

Παρατηρούμε ότι, ο σκοπός χρήστη θα χρησιμοποιήσει μια εκπλήρωση (αναζήτηση λίστας αγώνων), η οντότητα @EventType είναι υποχρεωτική (άρα γίνεται μια μικρή διαδικασία slot-filling) και έχει οριστεί ένα αντικείμενο context με ονομασία EventListing-followup. Το αντικείμενο context αυτό έχει ως σκοπό την συνέχιση της

ερώτησης (follow-up), στους επόμενους σκοπούς χρήστη της ίδιας κατηγορίας. Η ενέργεια του χρήστη εδώ ορίζεται ως η **getEvents**.

Στην συνέχεια και εφόσον έχει λάβει ο χρήστης την απάντηση, έχει ληφθεί υπόψη ότι χρειάζεται ο διαλογικός βοηθός να ρωτήσει τον χρήστη αν θέλει να βάλει ένα στοίχημα. Αυτό επικοινωνείται πολύ απλά μέσα στην λογική της επιστροφής του κειμένου στο χρήστη, ώστε ο χρήστης να μπορέσει να απαντήσει θετικά η αρνητικά.

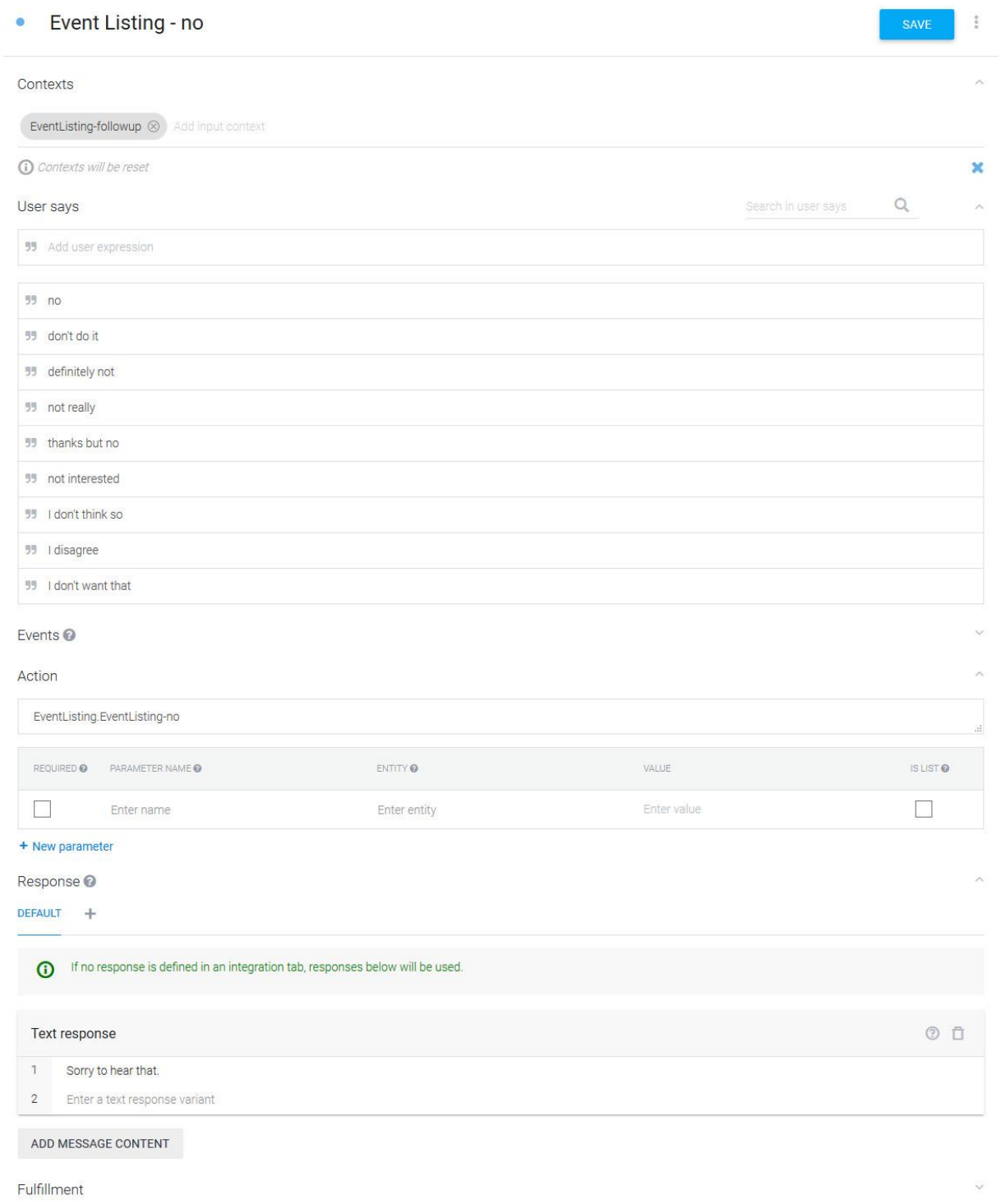


Εικόνα 4.17 Συνέχιση του σκοπού χρήστη με ερώτηση στοιχήματος

Για την διαχείριση του ΝΑΙ/ΟΧΙ του διαλογικού βοηθού στην ερώτηση «Would you like to place a bet?», χρησιμοποιούνται επερχόμενοι σκοποί χρήστη. Ορίζονται πάνω στο διαχειριστικό σύστημα έχοντας επιλεγμένο τον αρχικό σκοπό χρήστη (Event Listing).

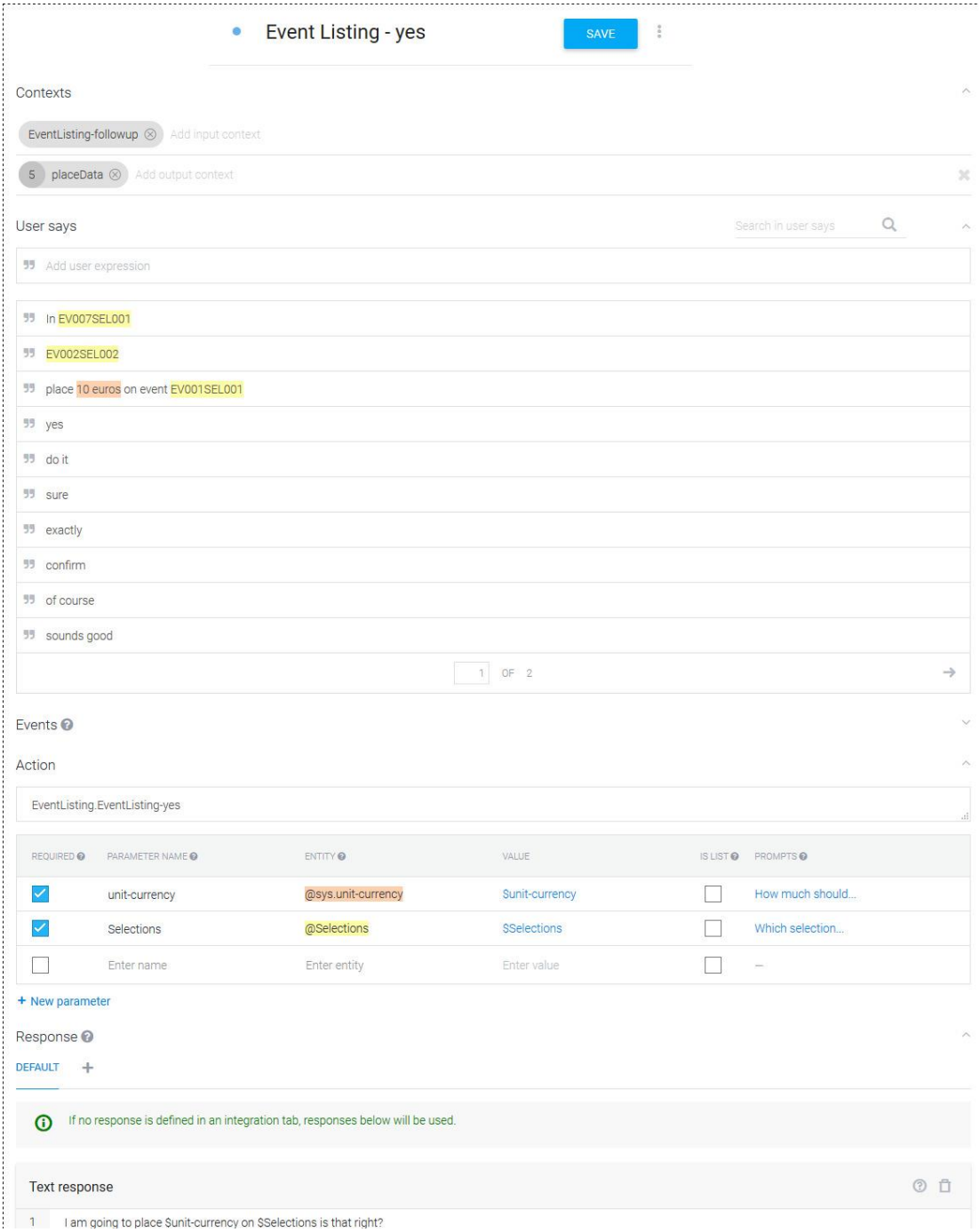
Εικόνα 4.18 Επερχόμενοι σκοποί χρήστη

Οι ορισμοί των σκοπών χρήστη χρησιμοποιούν το αντικείμενο context EventListing-followup. Για την αρνητική ερώτηση:



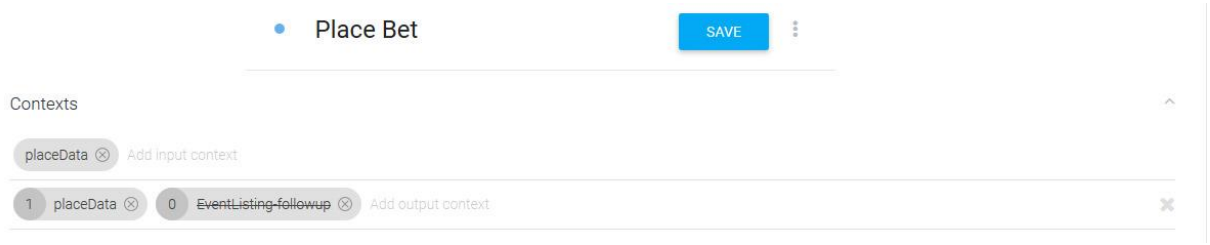
Εικόνα 4.19 Αρνητική απάντηση

Παρατηρείται ότι, ορίζουμε μια απάντηση στο text response, ενώ το αντικείμενο EventListing-followup βρίσκεται στην είσοδο των contexts (στην προηγούμενη οθόνη ήταν στην έξοδο των contexts). Επίσης η έξοδος των contexts έχει οριστεί να επαναφερθεί (reset) που σημαίνει ότι όλα τα context θα διαγραφούν σε αυτή την στιγμή του διαλόγου, άρα και θα τελειώσει ο διάλογος.



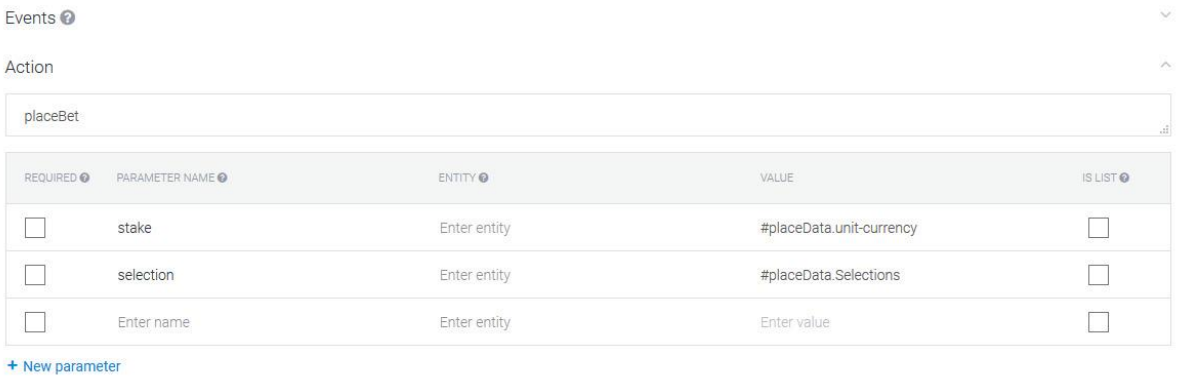
Εικόνα 4.20 Θετική απάντηση

Στη θετική απάντηση ορίζονται ως υποχρεωτικές παράμετροι οι τιμές από τις οντότητες @unit-currency (ποσό που διαθέτει ο χρήστης) και @Selections. Οι παράμετροι αυτοί εκτός από την διαδικασία slot-filling για την συμπλήρωση τους μεταφέρονται στο αντικείμενο context **placeData** που βρίσκεται στην έξοδο των contexts. Έτσι, όταν μεταφερθεί ο έλεγχος στο επόμενο σκοπό χρήστη «Place Bet», θα υπάρχουν οι πληροφορίες στο αντικείμενο και θα μπορούν να είναι προσβάσιμες μέσω του #placeData.unit-currency και #placeData.Selections.



Εικόνα 4.21 διαχείριση context objects στο σκοπό χρήστη Place Bet

Εδώ παρατηρείται ότι το context εισόδου placeData διαθέτει τις πληροφορίες του στην Place Bet. Παράλληλα ορίζεται ο χρόνος παραμονής του που είναι 1, άρα σημαίνει ότι μπορεί να διαβαστεί από ένα ακόμα σκοπό χρήστη και μετά θα αφαιρεθούν οι πληροφορίες από την συνεδρία του. Επίσης στο ίδιο τμήμα εξόδου των contexts δηλώνουμε ότι το EventListing.followup context αντικείμενο ακυρώνεται όποτε από την στιγμή αυτή δεν χρησιμοποιείται ξανά σαν πληροφορία.



Εικόνα 4.22 Λίστα παραμέτρων με την χρήση context πληροφοριών

Στην ίδια οθόνη ορίζουμε πλέον τις μεταβλητές που θα σταλθούν με εκπλήρωση στην ενέργεια placeBet **μέσα** από το αντικείμενο context #placeData. Έτσι στο πρόγραμμα οι πληροφορίες του context θα γίνουν αυτόματα ως μεταβλητές της μεθόδου placeBet. Τέλος ενεργοποιούμε την εκπλήρωση σε αυτό το σκοπό χρήστη.

Υλοποίηση στη βάση δεδομένων.

Το σχήμα που χρησιμοποιούμε για την βάση δεδομένων mongodb για τους αγώνες αποθηκεύεται στην συλλογή Events. Αντίστοιχα το σχήμα για την αποθήκευση των στοιχημάτων αποθηκεύεται στην συλλογή Bets.

```
{
    "_id" : ObjectId("59220514aa03100019f9d6ca"),
    "selection" : "EV007SEL001",
    "amount" : 10,
    "currency" : "EUR",
    "odds" : "3/1",
    "betslip" : "EV007SEL001_1495401748742",
    "__v" : 0
}
```

Κώδικας 4.6 Παράδειγμα εγγράφου Bets

Στο έγγραφο Bets τα πεδία που χρησιμοποιούνται είναι:

Πεδίο	**Τύπος πεδίου**	**Χρήση**
selection	String	Κωδικός πεδίου επιλογής αγώνα
amount	Numeric	Ποσό στοιχήματος
currency	String	Νόμισμα (μονάδα)
odds	String	Πιθανότητες νίκης της επιλογής του event
betslip	String	Κωδικός betslip

Πίνακας 4.2 Επεξήγηση πεδίων του σχήματος Bets

Αντίστοιχα για την εγγραφή events:

```
{
    "_id" : ObjectId("591385423476ebd2a3dbac0b"),
    "title" : "Vietnam vs Argentina",
    "event_id" : "EV002",
    "event_date" : "30/6/2017",
    "starts_at" : "17:00",
    "selection" : [
        {
            "title" : "Vietnam",
            "odds" : "3/1",
            "select_id" : "EV002SEL001"
        },
        {
            "title" : "Draw",
            "odds" : "1/1",
            "select_id" : "EV002SEL002"
        },
```

```
        {
            "title" : "Argentina",
            "odds" : "12/1",
            "select_id" : "EV002SEL003"
        }
    ],
    "sport" : "FOOTBALL"
}
```

Κώδικας 4.7 Παράδειγμα εγγράφου Event

Στο έγγραφο Event τα πεδία που χρησιμοποιούνται είναι:

Πεδίο	**Τύπος πεδίου**	**Χρήση**
title	String	Τίτλος αγωνίσματος
event_id	String	Κωδικός αγωνίσματος
event_date	Date	Ημερομηνία διεξαγωγής του αγωνίσματος
starts_at	String	Ώρα έναρξης
Selection	Array	Πίνακας επιλογών αγωνίσματος
Selection.title	String	Τίτλος επιλογής αγωνίσματος
Selection.odds	String	Πιθανότητες κέρδους της επιλογής
Selection.select_id	String	Κωδικός ης επιλογής του αγωνίσματος
sport	String	Κωδικός ταξινόμησης αθλήματος

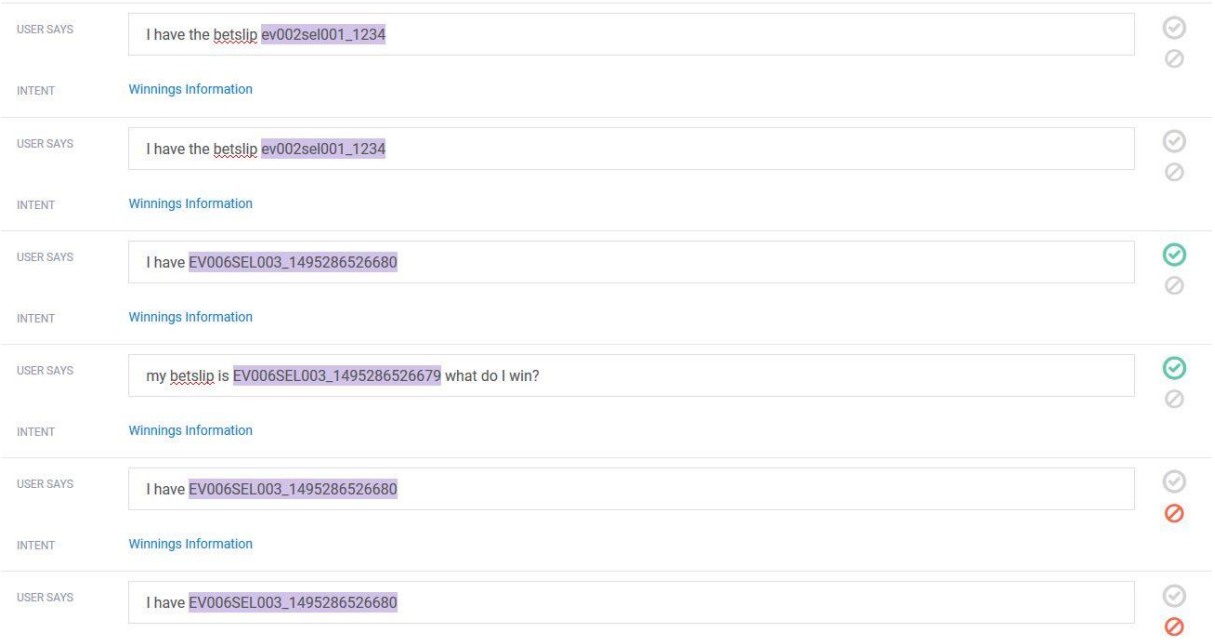
Πίνακας 4.3 Σχήμα εγγράφου Event

Στο έγγραφο Event παρατηρούμε ότι η συλλογή μπορεί να χρησιμοποιήσει και πίνακες σε σχέση με τις πατροπαράδοτες σχεσιακές βάσεις δεδομένων, δίνοντας μια υπεροχή στο χρόνο υλοποίησης (λιγότεροι πίνακες) της εφαρμογής.

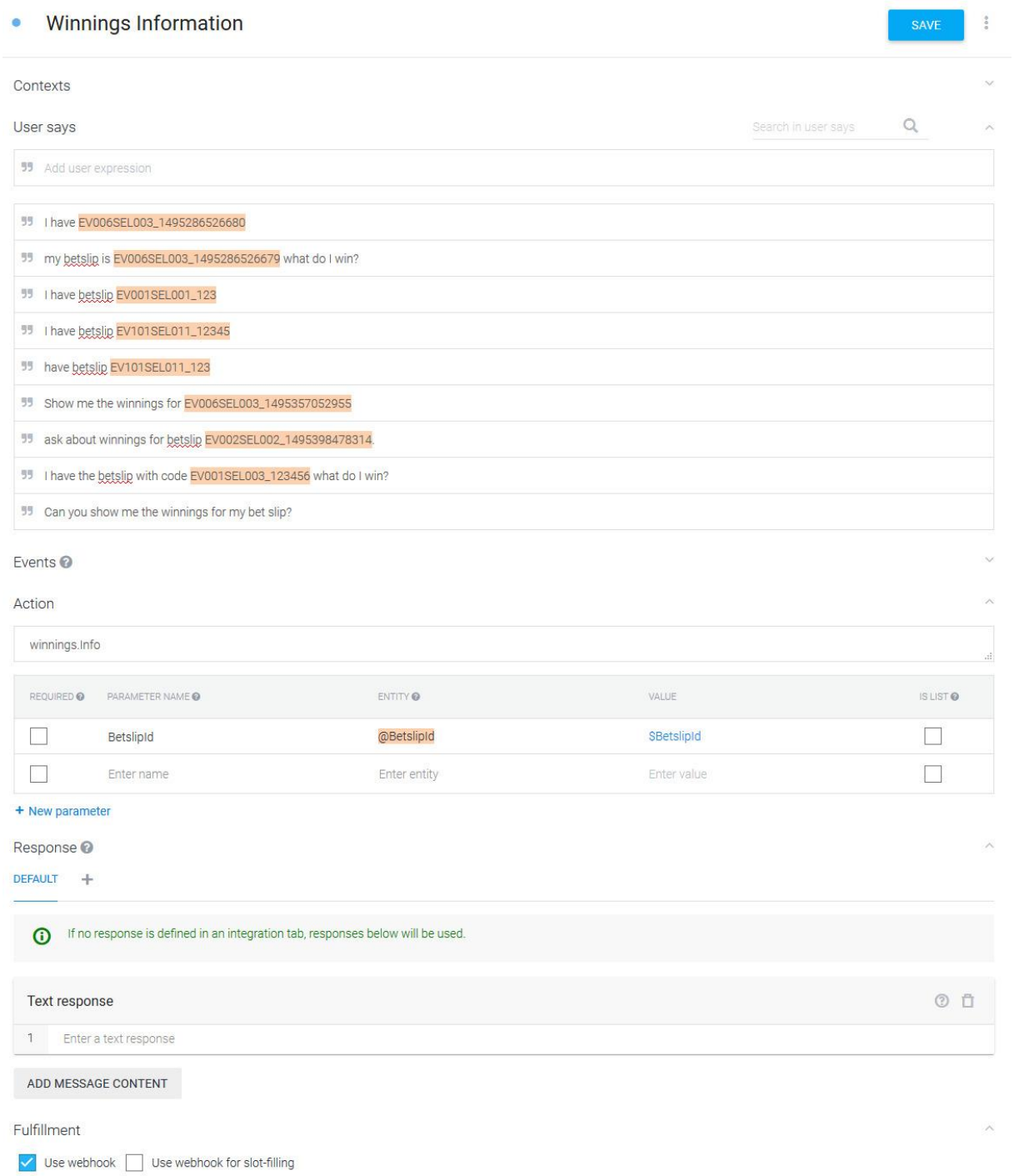
4.5.6 Αποτελέσματα κερδών

Ο τελικός σκοπός χρήστη είναι τα αποτελέσματα κερδών. Σε αυτό το σκοπό χρήστη, χρησιμοποιούμε την οντότητα @Betslipid. Η οντότητα @Betslip κρατά την πληροφορία του betslip η οποία δημιουργείται δυναμικά στο πίνακα Bets. Για να δημιουργηθεί η πληροφορία αυτή χρησιμοποιούμε ενδεικτικά το κωδικό της επιλογής του γεγονότος

μαζί με το timestamp του τρέχοντος χρόνου συνδεδεμένο με ένα underscore. Για την χρήση αυτού του σκοπού χρήστη χρησιμοποιήθηκε πολύ η επιλογή της εκπαίδευσης. Κάποια τμήματα εκφράσεων δεν αναγνωρίστηκαν σωστά, οπότε χρειάστηκε ο έλεγχος τόσο των σκοπών χρήστη, όσο και των οντοτήτων για την σωστή «εκμάθηση» των μεταβλητών.



Εικόνα 4.23 Εκμάθηση betslip.



Εικόνα 4.24 Ορισμός σκοπού χρήστη πληροφορίες κερδών

4.6 Υλοποίηση προγραμματιστικής εφαρμογής και υπηρεσιών

Το κύριο κομμάτι της εφαρμογής του διαλογικού βοηθού είναι υλοποιημένο σε javascript. Όλη η εφαρμογή λειτουργεί σε ένα αρχείο το οποίο είναι σχεδιασμένο έτσι ώστε να καλύπτει και την δευτερεύουσα υπηρεσία πληροφορίας στοιχημάτων.

```
//
//  Open university of Cyprus
//  Information and Communication Systems
//  Panagiotis Chalatsakos (c) 2017
//  Student Id: 11300837
//
const express = require('express');
const bodyParser = require('body-parser');
const _ = require('lodash');
const axios = require('axios');
const querystring = require("querystring");

const { mongoose } = require('./db/mongoose');
const { Event } = require('./models/event');
const { Bet } = require('./models/bet');
const { BetType } = require('./models/bettype1');

const eventTypes = ['football', 'tennis', 'basketball'];

const app = express();
app.use(bodyParser.json());

var botResponse = {};

app.get('/betinfo', (req, res) => {
    var qBetType1 = req.query.bettype1;
    var qBetType2 = req.query.bettype2;

    var info = "";

    if (qBetType1 !== 'undefined') {
        BetType.find({ code: qBetType1 }).then((docs) => {
            _.each(docs, (doc) => {
                info += `- ${doc.title} (code: ${doc.code}):
\n\t${doc.description}`;
                if (doc.selections !== undefined) {
```

```
                    info += `Number of selections to win: ${doc.selections}`;
                }
                info += `\n`;
            });
            res.status(200).send(info);
        }).catch((e) => {
            res.status(400).send("The service could not bring back any data.");
        });
    }
});

app.post('/webhook', (req, res) => {

    var action = req.body.result.action;
    var searchQuery = {};
    var selectionArray = [];
    switch (action) {
        case "winnings.Info":
            var betslipid = req.body.result.parameters.BetslipId;
            var botResponse = {};
            Bet.find({ betslip: betslipid }).then((docs) => {
                if (docs.length === 0) {
                    botResponse = {
                        "speech": `No betslip found. Please make sure you have
typed it correctly.`,
                        "displayText": `No betslip found. Please make sure you
have typed it correctly.`,
                        "source": "winningsInfoServiceApp"
                    };
                } else {
                    var msg = "";
                    _.each(docs, (item) => {
                        var value = eval(item.odds);
                        var price = item.amount * value;
                        var winnings = price + item.amount;
                        msg += `Bet Slip: ${item.betslip} with stake:
${item.amount} ${item.currency} and odds: ${item.odds} wins: ${winnings}
${item.currency}`;
                    });
                    botResponse = {
                        "speech": msg,
                        "displayText": msg,
```

```
                    "source": "winningInfoServiceApp"
                };
            }
            res.status(200).send(botResponse);
        });
        break;
    case "betInfo":
        var bettype1 = req.body.result.parameters.BetType1;
        let varToSend = "";
        if (_.isArray(bettype1)) {
            _.each(bettype1, (item) => {
                varToSend += `bettype1[]=${item}&`;
            });
            varToSend = varToSend.slice(0, -1);
        } else {
            varToSend = `bettype1=${bettype1}`;
        }


        const bet_info_url = `http://betty-bot-betty-bot-service.1d35.starter-
us-east-1.openshiftapps.com/betinfo/?${varToSend}`;
        //const bet_info_url = `http://127.0.0.1:8080/betinfo/?${varToSend}`;

        axios.get(bet_info_url).then((response) => {
            let botResponse = {
                "speech": `${response.data}`,
                "displayText": `${response.data}`,
                "source": "betInfoServiceApp"
            };
            res.status(200).send(botResponse);
        }).catch((err) => {
            res.status(400).send(err);
        });
        break;
    case "betInfo.listBetTypes":
        var returnText = "";
        BetType.find().then((docs) => {
            _.each(docs, (item) => {
                returnText += `\n- ${item.title} (code: ${item.code})`;
            });
            let botResponse = {
```

```
                    "speech": `Certainly! Here are all bet types:
${returnText}.\nYou can ask about any of them.`,
                    "displayText": `Certainly! Here are all bet types:
${returnText}.\n You can ask about any of them.`,
                    "source": "betInfoServiceApp"
                };
                res.status(200).send(botResponse);
            });
            break;
        case "placeBet":
            var selection = req.body.result.parameters.selection;
            var amount = req.body.result.parameters.stake.amount;
            var currency = req.body.result.parameters.stake.currency;
            var d = new Date();
            var n = d.getTime();
            var betslip = `${selection}_${n}`;

            Event.find({} , {selection: { $elemMatch: {select_id:
selection}}}).then((doc) => {
                var odds = "";
                _.each(doc, (item) => {
                    _.each(item.selection, (item2) => {
                        odds = item2.odds;
                        return false;
                    });
                });
                console.log(odds);
                var bet = new Bet({
                    selection: selection,
                    amount: amount,
                    currency: currency,
                    odds: odds,
                    betslip: betslip
                });

                bet.save().then((doc) => {
                    botResponse = {
                        "speech": `Bet placed. Your betslip id is: ${betslip}.`,
                        "displayText": `Bet has been placed.`,
                        "source": "betServiceApp"
                    };
```

```
                    res.status(200).send(botResponse);
                }, (e) => {
                    res.status(400).send(e);
                });
            });
            break;


        case "getEvents":
            let eventType = req.body.result.parameters.EventType;
            if (eventType === undefined || eventType.toUpperCase() == 'ALL' ||
eventType.toUpperCase() == 'SPORTS') {
                console.log("DEV: All events");
            } else {
                console.log("DEV: ", eventType);
                searchQuery = {
                    sport: eventType.toUpperCase()
                };
            }


            Event.find(searchQuery).then((events) => {
                if (events.length === 0) {
                    botResponse = {
                        "speech": `I haven't found any events of type
${eventType}...`,
                        "displayText": `I haven't found any events of type
${eventType}`,
                        "source": "eventServiceApp"
                    };
                } else {
                    let results = [];
                    _.each(events, (item) => {
                        var selections = "";
                        _.each(item.selection, (result) => {
                            selections += `\t[${result.title} (${result.odds})
${result.select_id}]\n`;
                            selectionArray.push(result);
                        });


                        results += `${item.title} : \n${selections}\n`;
                    });
                    botResponse = {
```

```
                        "speech": `Displaying ${events.length}
events:\n${results}\n\nWould you like to place a bet?`,
                        "displayText": `Displaying ${events.length} events...`,
                        "source": "eventServiceApp"
                    };
                }
                res.status(200).send(botResponse);
            }, (err) => {
                res.status(400).send(err);
            });
            break;

        default:
            botResponse = {
                "speech": "I did not get that quite...",
                "displayText": "I did not get that quite...",
                "source": "aiServiceApp"
            };
            res.status(200).send(botResponse);
    }
});

var server_port = process.env.OPENSHIFT_NODEJS_PORT || 8080;
var server_ip_address = process.env.OPENSHIFT_NODEJS_IP || '127.0.0.1';

app.listen(server_port, () => {
    console.log(`Service started at ${server_ip_address}:${server_port}`);
});
```

Κώδικας 4.8 Ο κώδικας υλοποίησης του διαλογικού βοηθού

4.6.1 Ανάλυση του κώδικα της υλοποίησης

Αρχικά γίνεται η εισαγωγή των βιβλιοθηκών και η αρχικοποίηση τους ως αντικείμενα:

```
const express = require('express');
const bodyParser = require('body-parser');
const _ = require('lodash');
const axios = require('axios');
const querystring = require("querystring");
```

Έπειτα γίνεται η εισαγωγή των τοπικών αρχείων βιβλιοθηκών και προτύπων (models) για την βάση δεδομένων:

```
const { mongoose } = require('./db/mongoose');
const { Event } = require('./models/event');
const { Bet } = require('./models/bet');
const { BetType } = require('./models/bettype1');
```

Κώδικας 4.9 εισαγωγή βιβλιοθηκών

Ορίζονται τα βασικά αθλήματα τα οποία συγκρίνονται με την τιμή που λαμβάνουμε από την παράμετρο @EventType. Οτιδήποτε δεν ανήκει σε αυτό το πίνακα τιμών απορρίπτεται, εκτός των τιμών all και sports οι οποίες χρησιμοποιούνται για όλα τα αθλήματα. Παράλληλα αρχικοποιούμε την κλάση express() ως entry point της εφαρμογής (app) και ορίζουμε το middleware bodyParser ώστε να επεξεργάζεται τα μηνύματα JSON. Τέλος αρχικοποιούμε την μεταβλητή botResponse η οποία θα επιστρέφει ένα αντικείμενο που θα περιέχει τις απαντήσεις πίσω στο api.ai:

```
const eventTypes = ['football', 'tennis', 'basketball'];
const app = express();

app.use(bodyParser.json());

var botResponse = {};
```

Κώδικας 4.10 Αρχικοποίηση expressjs και botresponse.

Το επόμενο κομμάτι κώδικα αφορά την υπηρεσία με endpoint GET /betInfo:

```
app.get('/betinfo', (req, res) => {
    var qBetType1 = req.query.bettype1;
    var info = "";

    if (qBetType1 !== 'undefined') {
        BetType.find({ code: qBetType1 }).then((docs) => {
            _.each(docs, (doc) => {
                info += `- ${doc.title} (code: ${doc.code}):
\n\t${doc.description}`;
                if (doc.selections !== undefined) {
                    info += `Number of selections to win: ${doc.selections}`;
                }
```

```
                info += `\n`;
            });
            res.status(200).send(info);
        }).catch((e) => {
            res.status(400).send("The service could not bring back any data.");
        });
    }
});
```

Κώδικας 4.11 ορισμός του endpoint /betinfo

Αρχικά ορίζεται η μέθοδος GET /betinfo. Κάθε μέθοδος στην expressjs έχει δύο αντικείμενα το request και response. Από το request αντικείμενο δέχεται το τύπο του στοιχήματος ο οποίος έρχεται urlencoded και αποθηκεύεται στην μεταβλητή qBetType1.

Έπειτα γίνεται ο έλεγχος αν αυτό που λήφθηκε υπάρχει (δηλαδή έχει μια τιμή). Διαφορετικά η javascript ορίζει ότι η μεταβλητή είναι undefined. Χρησιμοποιούμε το μοντέλο του Mongoose BetType για να βρούμε την συγκεκριμένη μεταβλητή μέσα στην συλλογή των εγγράφων με το κλειδί code. Μόλις βρεθεί, χρησιμοποιεί την μέθοδο then (λόγω promise), για να επιστρέψει πίσω όλα τα έγγραφα που βρήκε με αυτό το κλειδί. Είναι γνωστό ότι για κάθε code αντιστοιχεί μόνο ένα έγγραφο, όμως για προστασία χρησιμοποιείται η each μέθοδος του lodash για να εξεταστούν όλα τα έγγραφα που επιστράφηκαν. Ταυτόχρονα εισάγεται στην μεταβλητή info η τελική μορφοποίηση του κειμένου της πληροφορίας του στοιχήματος. Σε περίπτωση που δεν έχει αριθμό επιλογών (άρα ανήκει σε ένα από τα WIN, PLACE, EACH-WAY) τότε η πληροφορία αυτή δεν συμπεριλαμβάνεται στη μορφοποίηση.

Έπειτα επιστρέφεται με http status code 200 η πληροφορία στην οθόνη του χρήστη. Σε περίπτωση λάθους, επιστρέφεται το λάθος με http status code 400.

Επεξεργασία webhook (endpoint POST /webhook)

Το κεντρικό κομμάτι της εφαρμογής του διαλογικού βοηθού είναι η επεξεργασία των μηνυμάτων που έρχονται από την υπηρεσία api.ai και να πράξει αναλόγως της ετικέτας action. Καταρχάς ορίζεται το endpoint POST /webhook:

```
app.post('/webhook', (req, res) => { }
```

Έπειτα εσωτερικά χρειάζεται να γνωρίζουμε ποιο action θα διαχειριστεί η εφαρμογή. Έτσι λαμβάνεται η τιμή μέσω του μηνύματος JSON από το bodyParser αντικείμενο που αναλύει το μήνυμα σε αντικείμενο javascript:

```
    var action = req.body.result.action;
```

Αρχικοποιούνται οι παράμετροι searchQuery, selectionArray και έπειτα γίνεται η διακλάδωση βάσει της ετικέτας action:

Ετικέτα	**Λειτουργία**
winnings.Info	Προβολή κερδών βάσει του betslipid
betInfo	Πληροφορία στοιχημάτων
betInfo.listBetTypes	Λίστα όλων των τύπων στοιχημάτων
placeBet	Διεκπεραίωση στοιχήματος
getEvents	Λίστα των αθλητικών γεγονότων
default	Απόκριση σε περίπτωση λάθους

Πίνακας 4.4 Λίστα ετικετών action χρήστη

Προβολή κερδών βάσει του betslipid

```
            var betslipid = req.body.result.parameters.BetslipId;
            var botResponse = {};
            Bet.find({ betslip: betslipid }).then((docs) => {
                if (docs.length === 0) {
                    botResponse = {
                        "speech": `No betslip found. Please make sure you have
typed it correctly.`,
                        "displayText": `No betslip found. Please make sure you
have typed it correctly.`,
                        "source": "winningsInfoServiceApp"
                    };
                } else {
                    var msg = "";
                    _.each(docs, (item) => {
                        var value = eval(item.odds);
                        var price = item.amount * value;
                        var winnings = price + item.amount;
```

```
                        msg += `Bet Slip: ${item.betslip} with stake:
${item.amount} ${item.currency} and odds: ${item.odds} wins: ${winnings}
${item.currency}`;
                    });
                    botResponse = {
                        "speech": msg,
                        "displayText": msg,
                        "source": "winningInfoServiceApp"
                    };
                }
                res.status(200).send(botResponse);
            });
```

Σε αυτό το κομμάτι κώδικα, λαμβάνεται η πληροφορία betslipid από την τιμή της παραμέτρου που αντιστοιχεί στην οντότητα @Betslipid. Χρησιμοποιείται το μοντέλο Bet της συλλογής Bets για την ανάκτηση των εγγράφων. Σε περίπτωση που δεν βρεθούν έγγραφα (δεν υπάρχει το συγκεκριμένο betslipid) τότε πρέπει να επιστραφεί μήνυμα λάθους. Διαφορετικά υπολογίζεται η τιμή των κερδών και επιστρέφεται ένα μήνυμα με το προβλεπόμενο κέρδος για αυτό το στοίχημα.

Πληροφορία στοιχημάτων

```
            var bettype1 = req.body.result.parameters.BetType1;
            let varToSend = "";
            if (_.isArray(bettype1)) {
                _.each(bettype1, (item) => {
                    varToSend += `bettype1[]=${item}&`;
                });
                varToSend = varToSend.slice(0, -1);
            } else {
                varToSend = `bettype1=${bettype1}`;
            }



            const bet_info_url = `http://betty-bot-betty-bot-service.1d35.starter-
us-east-1.openshiftapps.com/betinfo/?${varToSend}`;
            //const bet_info_url = `http://127.0.0.1:8080/betinfo/?${varToSend}`;


            axios.get(bet_info_url).then((response) => {
                let botResponse = {
```

```
                "speech": `${response.data}`,
                "displayText": `${response.data}`,
                "source": "betInfoServiceApp"
            };
            res.status(200).send(botResponse);
        }).catch((err) => {
            res.status(400).send(err);
        });
```

Πρόκειται για την πληροφορία που έρχεται από το μήνυμα της υπηρεσίας api.ai και όχι της δευτερεύουσας υπηρεσίας GET /betInfo. Σε αυτή τη διαδικασία, το πρόγραμμα ανακτά την πληροφορία από την παράμετρο BetType1. Η παράμετρος αυτή μπορεί να είναι πίνακας με την μορφή query string (var[]=value&var[]=value) οπότε ανάλογα με τις τιμές που λαμβάνουμε δημιουργείται είτε η μοναδική τιμή η ο πίνακας. Έπειτα στέλνεται με την χρήση της βιβλιοθήκης axios στο endpoint /betInfo. Σε αυτή την υλοποίηση το endpoint βρίσκεται στον ίδιο κώδικα με την κεντρική εφαρμογή. Όμως αυτό μπορεί να είναι μια άλλη υπηρεσία σε ένα απομακρυσμένο εξυπηρετητή. Με την αλλαγή του url μπορεί να ανακτήσει τις πληροφορίες από εκείνο το σημείο.

Κατά την λήψη των απαντήσεων από το endpoint δημιουργείται το κείμενο που θα σταλθεί πίσω στο api.ai. Σε περίπτωση λάθους στέλνεται το μήνυμα λάθους.

Λίστα όλων των τύπων στοιχημάτων

Η λίστα όλων των στοιχημάτων βρίσκεται μέσα στην εφαρμογή του χρήστη και όχι στο endpoint GET /betInfo. Ο λόγος είναι ότι χρησιμοποιεί κοινή βάση δεδομένων οπότε μπορεί να ανακτήσει τις πληροφορίες χωρίς την χρήση άλλης υπηρεσίας.

```
            var returnText = "";
            BetType.find().then((docs) => {
                _.each(docs, (item) => {
                    returnText += `\n- ${item.title} (code: ${item.code})`;
                });
                let botResponse = {
                    "speech": `Certainly! Here are all bet types:
${returnText}.\nYou can ask about any of them.`,
```

```
                    "displayText": `Certainly! Here are all bet types:
${returnText}.\n You can ask about any of them.`,
                    "source": "betInfoServiceApp"
                };
                res.status(200).send(botResponse);
            });
            break;
```

Το μόνο που κάνει αυτή η διαδικασία είναι να ανακτήσει όλα τα έγγραφα από τη συλλογή BetTypes και να τα παρουσιάζει με το format και την απάντηση που ορίζεται στο botResponse αντικείμενο. Το αντικείμενο αυτό στέλνεται πίσω στο api.ai.

Διεκπεραίωση στοιχήματος

```
            var selection = req.body.result.parameters.selection;
            var amount = req.body.result.parameters.stake.amount;
            var currency = req.body.result.parameters.stake.currency;
            var d = new Date();
            var n = d.getTime();
            var betslip = `${selection}_${n}`;

            Event.find({} , {selection: { $elemMatch: {select_id:
selection}}}).then((doc) => {
                var odds = "";
                _.each(doc, (item) => {
                    _.each(item.selection, (item2) => {
                        odds = item2.odds;
                        return false;
                    });
                });
                console.log(odds);
                var bet = new Bet({
                    selection: selection,
                    amount: amount,
                    currency: currency,
                    odds: odds,
                    betslip: betslip
                });
```

```
            bet.save().then((doc) => {
                botResponse = {
                    "speech": `Bet placed. Your betslip id is: ${betslip}.`,
                    "displayText": `Bet has been placed.`,
                    "source": "betServiceApp"
                };
                res.status(200).send(botResponse);
            }, (e) => {
                res.status(400).send(e);
            });
        });
```

Στη διεκπεραίωση του στοιχήματος, γίνεται ανάκτηση από το μήνυμα του api.ai τις παραμέτρους της επιλογής αγώνα, του ποσού που θα διαθέσει ο χρήστης και του νομίσματος του χρήστη. Δημιουργείται επίσης το κλειδί που θα χρησιμεύσει στην δημιουργία του betslip. Έπειτα γίνεται η αναζήτηση στα γεγονότα χρήστη για τις πιθανότητες του επιλεγμένου γεγονότος. Ένα από τα προβλήματα του Mongoose είναι ότι δεν χειρίζεται τόσο καλά τους πίνακες όσο η ίδια η mongodb και χρειάζεται το σύστημα των πολλαπλών loops για να γίνει ανάκτηση των πεδίων. Έχοντας βρει τη τιμή της μεταβλητής odds, δημιουργείται μια νέα εγγραφή στο πίνακα Bets και έπειτα επιστρέφει το κλειδί του betslip στο χρήστη. Θεωρούμε για τα πλαίσια της μεταπτυχιακής διατριβής ότι η εισαγωγή ήταν επιτυχής.

Λίστα των αθλητικών γεγονότων

```
        let eventType = req.body.result.parameters.EventType;
        if (eventType === undefined || eventType.toUpperCase() == 'ALL' ||
eventType.toUpperCase() == 'SPORTS') {
            console.log("DEV: All events");
        } else {
            console.log("DEV: ", eventType);
            searchQuery = {
                sport: eventType.toUpperCase()
            };
        }

        Event.find(searchQuery).then((events) => {
```

```
                if (events.length === 0) {
                    botResponse = {
                        "speech": `I haven't found any events of type
${eventType}...`,
                        "displayText": `I haven't found any events of type
${eventType}`,
                        "source": "eventServiceApp"
                    };
                } else {
                    let results = [];
                    _.each(events, (item) => {
                        var selections = "";
                        _.each(item.selection, (result) => {
                            selections += `\t[${result.title} (${result.odds})
${result.select_id}]\n`;
                            selectionArray.push(result);
                        });

                        results += `${item.title} : \n${selections}\n`;
                    });
                    botResponse = {
                        "speech": `Displaying ${events.length}
events:\n${results}\n\nWould you like to place a bet?`,
                        "displayText": `Displaying ${events.length} events...`,
                        "source": "eventServiceApp"
                    };
                }
                res.status(200).send(botResponse);
            }, (err) => {
                res.status(400).send(err);
            });
```

Στη λίστα των αθλητικών γεγονότων, γίνεται ανάκτηση την τιμή της παραμέτρου της οντότητας @EventType. Σε περίπτωση που περιέχει τη τιμή «all» ή «sports», το πρόγραμμα θα επιστρέψει όλα τα σπορ. Ειδάλλως θα επιστρέψει ένα από τα sport που υπάρχουν ορισμένα στις οντότητες @EventType.

Η αναζήτηση γίνεται στην συλλογή Events. Σε περίπτωση που δεν βρεθούν έγγραφα επιστρέφει μήνυμα που εξηγεί ότι δεν έχει βρει κάποιο έγγραφο. Διαφορετικά δημιουργεί ένα μορφοποιημένο πίνακα με τις πληροφορίες των γεγονότων, όπου και τις

επιστρέφει στο χρήστη. Λόγω του ότι η ενέργεια αυτή θα συνεχιστεί με νέα έκφραση από το χρήστη, στα δεδομένα επιστροφής, εισάγεται και μια επιπλέον ερώτηση για την συνέχιση και διεκπεραίωση ενός στοιχήματος.

Απόκριση σε περίπτωση λάθους

Η απόκριση αυτή αφορά την περίπτωση που δεν έχει βρεθεί κάποια ενέργεια χρήστη (άρα και κάποιος σκοπός χρήστη). Έτσι επιστρέφει ένα μήνυμα λάθους και προτροπής.

Δημοσίευση της υπηρεσίας

```
var server_port = process.env.OPENSHIFT_NODEJS_PORT || 8080;
var server_ip_address = process.env.OPENSHIFT_NODEJS_IP || '127.0.0.1';

app.listen(server_port, () => {
    console.log(`Service started at ${server_ip_address}:${server_port}`);
});
```

Τέλος ο παραπάνω κώδικας δημοσιεύει την υπηρεσία του διαλογικού βοηθού στην διεύθυνση του νέφους του openshift ή τοπικά στην port 8080 και ενημερώνει το διαχειριστή του προγράμματος ότι έχει εκκινηθεί η υπηρεσία.

Μοντέλα mongoose

Παράλληλα με τον κώδικα παρατίθενται τα αρχεία μοντέλων mongoose της υπηρεσίας:

```
var mongoose = require('mongoose');

var BetSchema = mongoose.Schema({
    selection: {
        type: String
    },
    amount: {
        type: Number
    },
    currency: {
        type: String
```

```
    },
    odds: {
        type: String
    },
    betslip: {
        type: String
    }
});

var Bet = mongoose.model('Bet', BetSchema, 'Bets');

module.exports = { Bet };
```

Κώδικας 4.12 Αρχείο server/model/bet.js

```
var mongoose = require('mongoose');

var BetTypeSchema = mongoose.Schema({
    title: {
        type: String
    },
    code: {
        type: String
    },
    description: {
        type: String
    },
    selections: {
        type: Number
    }
});

var BetType = mongoose.model('BetType', BetTypeSchema, 'BetType');

module.exports = { BetType };
```

Κώδικας 4.13 Αρχείο server/models/bettype1.js

```
const mongoose = require('mongoose');

const Selection = mongoose.Schema({
    title: String,
```

```
    odds: String,
    select_id: String
});

const Event = mongoose.model('Event', {
    title: {
        type: String
    },
    event_id: {
        type: String
    },
    event_date: {
        type: Date
    },
    starts_at: {
        type: Number
    },
    sport: {
        type: String
    },
    selection: [Selection]
}, "Events");

module.exports = { Event };
```

Κώδικας 4.14 Αρχειο server/models/event.js

```
const mongoose = require('mongoose');
mongoose.Promise = global.Promise;
mongoose.connect('mongodb://pchalats:Panahs18s!@ds133291.mlab.com:33291/betservice
infodb');
mongoose.set('debug', true);
module.exports = { mongoose };
```

Κώδικας 4.15 Αρχείο server/db/mongoose.js

Κεφάλαιο 5
Επίλογος

5.1 Επανεξέταση των ερευνητικών ερωτημάτων

Κατά την συγκρότηση της μεταπτυχιακής διατριβής είχαν τεθεί τα ακόλουθα ερευνητικά ερωτήματα:

1. Ποια είναι τα κυριότερα bot frameworks και πως συγκρίνονται βάσει των χαρακτηριστικών και δυνατοτήτων τους (π.χ. αρχιτεκτονικής, προγραμματιστικού μοντέλο, NLP υπηρεσιών που ενσωματώνουν);
2. Πως μπορεί να αξιοποιηθεί ένα bot framework για την βελτίωση της διεπαφής χρήστη μιας εφαρμογής;
3. Πόσο πρακτική και αποτελεσματική είναι η ενσωμάτωση σε μια εφαρμογή ενός προσωπικού βοηθού βασισμένου σε κάποιο από τα διαθέσιμα bot frameworks;

Στα πλαίσια της έρευνας που διεξαχθεί για την εκπόνηση της μεταπτυχιακής διατριβής τα ερωτήματα αυτά απαντήθηκαν η παραμένουν ακόμα να απαντηθούν σε μελλοντικές έρευνες.

Στο ερώτημα 1 τα κυριότερα bot frameworks είναι τα frameworks που εξετάστηκαν, δηλαδή το wit.ai, api.ai και bot framework. Στο κεφάλαιο 3 αναφέρεται η διαδικασία σύγκρισης αυτών των frameworks. Με την χρήση των smartphones η χρήση των έξυπνων διαλογικών πρακτόρων είναι το επόμενο βήμα στην εξέλιξη της διεπαφής του ανθρώπου με την μηχανή. Λόγω του μεγάλου ενδιαφέροντος που υπάρχει, είναι δυνατή η δημιουργία μιας οικονομίας βασισμένης στην δημιουργία διαλογικών βοηθών. Έτσι νέα framework θα εμφανίζονται πολλές φορές για να καλύψουν τις ανάγκες που κάποιο υπάρχον bot framework δεν επαρκεί. Μια από τις σημαντικότερες NLP υπηρεσίες που κυκλοφορούν σαν υπηρεσία στο διαδίκτυο είναι η μηχανή κατανόησης φυσικής γλώσσας. Μέσω αυτής ο χρήστης μπορεί να προσθέσει στο διάλογο αναγνώριση ονομαστικών οντοτήτων, αναγνώριση σκοπών χρήστη και σε πειραματικό στάδιο αναγνώριση συναισθημάτων. Μια τέτοια υπηρεσία είναι το LUIS της Microsoft.

Στο ερώτημα 2, ένα bot framework επιδιώκει την αντικατάσταση του παραδοσιακού τρόπου οπτικής διεπαφής και χρήσης φορμών και οθονών λογισμικού με την χρήση του διαλόγου μέσω φυσικής γλώσσας. Όπως και στις παραδοσιακές εφαρμογές λογισμικού, η εμπειρία του χρήστη είναι ένα από τα κυριότερα κριτήρια, έτσι και η εμπειρία του χρήστη καθώς και η ποιότητα του διαλόγου, δηλαδή πόσο καλά στοχευμένος είναι πόσο περιγραφικός, με την χρήση ονομαστικών οντοτήτων ή εμφανών σκοπών χρήστη μπορεί να βελτιώσει την διεπαφή μεταξύ χρήστη και μηχανής.

Για το ερώτημα 3, η εμπειρία της δημιουργίας και της ενσωμάτωσης ενός διαλογικού βοηθού έδειξε ότι:

1. Απαιτείται μελέτη και ένα μέσο θεωρητικό υπόβαθρο στην επεξεργασία φυσικής γλώσσας και στην μηχανική μάθηση ώστε να βοηθηθεί ο δημιουργός σε θέματα ασάφειας που τυχόν θα δημιουργηθούν κατά την δημιουργία του διαλογικού πράκτορα.
2. Η επεξεργασία φυσικής γλώσσας δεν θα βοηθήσει ένα διαλογικό πράκτορα να πετύχει το σκοπό του. Ο διάλογος που διατυπώνεται πρέπει να είναι καλά σχεδιασμένος ώστε να προβλέπονται τυχόν προβλήματα κατά την διεξαγωγή του. Σε αυτό το κομμάτι, βοηθάει πολύ η υπηρεσία διαλογικού βοηθού που χρησιμοποιείται να έχει δυνατότητα μετεκπαίδευσης του μοντέλου που δημιουργείται για το διαλογικό βοηθό.

3. Η χρήση μηχανικής μάθησης δεν θα κάνει το διαλογικό πράκτορα πιο έξυπνο. Σε περιπτώσεις εμπορικής χρήσης καλύτερα είναι να μπορεί ο διαλογικός πράκτορας να ικανοποιεί τους σκοπούς του χρήστη πρώτα και μετά να εξεταστούν οι περιπτώσεις προσθήκης προσωπικότητας και «εξυπνάδας».
4. Η ενσωμάτωση είναι σχετικά αμελητέα διαδικασία, η τεχνολογία κυρίως βασίζεται σε τεχνολογία ανταλλαγής μηνυμάτων μεταξύ υπηρεσιών οπότε οι υλοποιήσεις είναι εύκολες εφόσον υπάρχει καλή τεκμηρίωση.

5.2 Μελλοντικές έρευνες και επεκτάσεις της μεταπτυχιακής διατριβής

Η μεταπτυχιακή διατριβή προσπάθησε να θέσει τις βάσεις για την επεξεργασία φυσικής γλώσσας και την κατανόηση της καθώς και την εισαγωγή για υλοποίηση διαλογικών πρακτόρων. Λόγω του ενδιαφέροντος της έρευνας πάνω στους διαλογικούς πράκτορες τόσο σε θέματα τεχνητής νοημοσύνης όσο και σε θέματα ανάκτησης πληροφοριών και εξαγωγής πληροφοριών ένας ενδιαφέρον τομέας ενασχόλησης είναι το data mining. Με την χρήση μεγάλων buzzwords όπως big data, ένα σύστημα διαλογικού πράκτορα το οποίο θα μπορούσε να επεξεργαστεί ένα τέτοιο όγκο δεδομένων θα ήταν ένας χρήσιμος βοηθός. Άλλος τομέας ενασχόλησης είναι το internet of things. Συνδέοντας τα πάντα με τους πάντες, οι διαλογικοί πράκτορες μπορούν να χρησιμοποιηθούν ως ρυθμιστές των πληροφοριών που δέχονται από τις συσκευές IoT [73].

Βιβλιογραφία

[1] I. Vlachavas, K. P., V. N., K. F και S. H., Artificial Intelligence - 3rd Edition, Thessaloniki, Greece: University of Macedonia Press / Greece, 2011.

[2] D. Jurafsky και J. H. Martin, Speech and Language Processing, New Jersey: Prentice Hall, Englewood Cliffs, New Jersey 07632, 2000.

[3] A. Wilson, P. Rayson και T. McEnery, A Rainbow of Corpora: Corpus Linguistics and the Languages of the World., München: Lincom-Europa, 2003.

[4] K. Aijmer και B. Altenberg, «English corpus linguistics,» *English for Specific Purposes,* pp. 109-112, 1994.

[5] B. B. και J. Pustejovsky, «Lexical Knowledge Representation and Natural Language Processing,» *Artificial Intelligence 63,* pp. 193-223, 1993.

[6] R. Baeza-Yates και B. Ribiero-Neto, Modern Information Retrieval, Harlow, Essex, UK: Addison Wesley Longman Limited, 1999.

[7] D. Jurafsky και C. Manning, «Course Introduction - Stanford NLP,» 31 March 2012. [Ηλεκτρονικό]. Available: https://www.youtube.com/watch?v=nfoudtpBV68&list=PL6397E4B26D00A269.

[8] Y. Wilks, «Information extraction as a core language technology,» *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology,* pp. 1-9, 2005.

[9] Wikipedia, «The Turing Test,» 19 April 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Turing_test.

[10] IBM, «Build a chatbot with Watson,» IBM, [Ηλεκτρονικό]. Available: https://www.ibm.com/watson/how-to-build-a-chatbot/. [Πρόσβαση 18 May 2017].

[11] J. Best, «IBM Watson: The inside story of how the Jeopardy-winning supercomputer was born, and what it wants to do next,» Techrepublic, [Ηλεκτρονικό]. Available: http://www.techrepublic.com/article/ibm-watson-the-inside-story-of-how-the-jeopardy-winning-supercomputer-was-born-and-what-it-wants-to-do-next/.

[12] C. Manning και H. Schutze, Foundations of statistical natural language processing, MIT Press, 1999.

[13] D. Radev, «Introduction - Natural Language Processing | University of Michigan,» 24 March 2016. [Ηλεκτρονικό]. Available: https://www.youtube.com/watch?v=n25JjoixM3I.

[14] D. Radev, «EECS 595 / SI 561 / LING 541 - Natural Language Processing Lecture Notes 02,» 2015. [Ηλεκτρονικό]. Available: http://clair.si.umich.edu/NLP-fall2015/notes/nlp02.pptx.

[15] Wikipedia, «Wikipedia - Natural language processing - Major evaluations and

task,» 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Natural_language_processing#Major_evaluations_and_tasks. [Πρόσβαση 09 April 2017].

[16] Handbook of Natural Language Processing, Second Edition, Chapman & Hall Crc, 2010.

[17] P. Paroubek, S. Chaudiron και H. Lynette, «Principles of Evaluation in Natural Language Processing,» *Traitement Automatique des Langues, ATALA,* τόμ. 1, αρ. 48, p. 7031, 2017.

[18] M. K. Anjali και A. P. Babu, «Ambiguities in Natural Language Processing,» *International Journal of Innovative Research in Computer and Communication Engineering,* pp. 392-394, 2014.

[19] Wit.Ai Blog, «Ambiguity in natural language,» 27 June 2014. [Ηλεκτρονικό]. Available: https://wit.ai/blog/2014/06/27/ambiguity-in-natural-language.

[20] «Natural Language Understanding,» 27 03 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Natural_language_understanding.

[21] T. Winograd, «Understanding natural language,» *Cognitive Psychology,* p. 1–191, 1972.

[22] N. Ranjan, K. Mundada, K. Phaltane και A. Said, «A survey on techniques in NLP,» *International Journal of Computer Applications,* pp. 6-9, 2016.

[23] D. Jurafsky και C. Manning, «4 - 1 - Introduction to N-grams- Stanford NLP - Professor Dan Jurafsky & Chris Manning,» 02 April 2012. [Ηλεκτρονικό]. Available: https://www.youtube.com/watch?v=s3kKlUBa3b0.

[24] D. D. McDonald, «Natural Language Generation,» *Handbook of Natural Language Processing,* 2000.

[25] E. Reiter και R. Dale, «Building Natural Language Generation Systems,» σε *Studies in Natural Language Processing*, Cambridge, United Kingdom, Cambridge University Press, 2000, p. xxi+248.

[26] C. Manning και D. Jurafsky, «18 - 1 - Introduction to Information Retrieval-Stanford NLP-Professor Dan Jurafsky & Chris Manning,» 21 April 2012. [Ηλεκτρονικό]. Available: https://www.youtube.com/watch?v=5L1qemKyUKA.

[27] R. Baeza-Yates και R.-N. Berthier, Modern Information Retrieval, Edinburgh Gate: ACM Press, 1999.

[28] J. Cowie και W. Lehnert, «Information Extraction,» *Communications of the ACM,* pp. 80-91, 1996.

[29] D. Jurafsky, «Information Extraction and Named Entity Recognition Lecture Notes,» [Ηλεκτρονικό]. Available: https://web.stanford.edu/class/cs124/lec/Information_Extraction_and_Named_Entity_Recognition.pdf.

[30] R. Grishman, «Information Extration A Multidisciplinary Approach to an Emerging Information Technology,» *Lecture Notes in Computer Science,* pp. 10-27, 2005.

[31] S. Kripke, Naming and Necessity, Boston: Harvard University Press, 1982.

[32] L. Ratinov και D. Roth, «Design Challenges and Misconceptions in Named Entity Recognition,» σε *CoNLL '09 Proceedings of the Thirteenth Conference on Computational Natural Language Learning* , Boulder, Colorado, 2009.

[33] B. Data Community DC, «An Introduction to Named Entity Recognition in Natural Language Processing - Part 1,» 17 April 2013. [Ηλεκτρονικό]. Available: http://www.datacommunitydc.org/blog/2013/04/a-survey-of-stochastic-and-gazetteer-based-approaches-for-named-entity-recognition.

[34] H. Nwana, «Software Agents: An overview,» *Knowledge Engineering Review,* pp. 205-244, October/November 1996.

[35] J. Weisenbaum, «ELIZA—a computer program for the study of natural language communication between man and machine,» *Communications of the ACM,* τόμ. 9, αρ. 1, pp. 36-45, 1966.

[36] J. Toscano, «Designing A Chatbot Conversation: How To Keep Users In The Loop,» 1 Feb 2017. [Ηλεκτρονικό]. Available: https://uxdesign.cc/designing-a-chatbot-conversation-how-to-keep-users-in-the-loop-4d3a29e44de4. [Πρόσβαση 22 May 2017].

[37] D. Britz, «Deep Learning for chatbots, part I: Introduction,» 06 04 2016. [Ηλεκτρονικό]. Available: http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/.

[38] W. R. και G. T., « A General Architecture for Connecting NLP Frameworks and Desktop Clients Using Web Services.,» *Kapetanios E., Sugumaran V., Spiliopoulou M. (eds) Natural Language and Information Systems.,* τόμ. 5039, αρ. NLDB 2008. Lecture Notes in Computer Science, pp. 317-322, 2008.

[39] P. Surmenok, «Natural Language Pipeline for Chatbots,» 03 November 2016. [Ηλεκτρονικό]. Available: http://pavel.surmenok.com/2016/11/05/natural-language-pipeline-for-chatbots/. [Πρόσβαση 09 May 2017].

[40] T. Api.Ai, «Slot-Filling,» 25 April 2017. [Ηλεκτρονικό]. Available: https://docs.api.ai/docs/guidelines-slot-filling.

[41] Wit.ai, «Wit.ai Documentation,» [Ηλεκτρονικό]. Available: https://wit.ai/docs. [Πρόσβαση 18 May 2017].

[42] Wit.ai, «HTTP API Reference 2017,» [Ηλεκτρονικό]. Available: https://wit.ai/docs/http/20170307. [Πρόσβαση 18 May 2017].

[43] Wit.ai, «node-wit library,» 30 Nov 2016. [Ηλεκτρονικό]. Available: https://github.com/wit-ai/node-wit. [Πρόσβαση 18 May 2017].

[44] Wit.ai, «github node-wit examples,» [Ηλεκτρονικό]. Available: https://github.com/wit-ai/node-wit/blob/master/examples/messenger.js. [Πρόσβαση 18 May 2017].

[45] Archibald, Jake, «JavaScript Promises: an Introduction,» 12 May 2017. [Ηλεκτρονικό]. Available: https://developers.google.com/web/fundamentals/getting-started/primers/promises. [Πρόσβαση 15 May 2017].

[46] G. Maps, «Reverse Geocoding,» Alphabet Inc., [Ηλεκτρονικό]. Available: https://developers.google.com/maps/documentation/javascript/examples/geocoding-reverse.

[47] api.ai, «api.ai documentation (key concepts),» 23 03 2017. [Ηλεκτρονικό]. Available: https://docs.api.ai/docs/key-concepts.

[48] API.AI, «APi.AI concept: intents,» 11 April 2017. [Ηλεκτρονικό]. Available: https://docs.api.ai/docs/concept-intents#response. [Πρόσβαση 15 May 2017].

[49] «Api.ai Query Webhook Response,» 2017. [Ηλεκτρονικό]. Available: https://docs.api.ai/docs/query#response. [Πρόσβαση 18 May 2017].

[50] «HTTP Status Codes,» 2017. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes. [Πρόσβαση 18 May 2017].

[51] «Webhook - response message,» [Ηλεκτρονικό]. Available: https://docs.api.ai/docs/webhook. [Πρόσβαση 18 May 2017].

[52] P. Chalatsakos, «Simple Weather Service (github),» [Ηλεκτρονικό]. Available:

https://github.com/ditikos/SimpleWeatherService. [Πρόσβαση 20 May 2017].

[53] M. C. Services, «Microsoft Azure: Language Understanding Intelligent Service,» Microsoft, [Ηλεκτρονικό]. Available: https://azure.microsoft.com/en-us/services/cognitive-services/language-understanding-intelligent-service/. [Πρόσβαση 18 May 2017].

[54] M. L. /. Azure, «Cortana Prebuilt App,» [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/luis/cortana-prebuilt-app. [Πρόσβαση 16 May 2017].

[55] M. C. Services, «What is Cortana?,» Microsoft, [Ηλεκτρονικό]. Available: https://www.microsoft.com/en/mobile/experiences/cortana/. [Πρόσβαση 18 May 2017].

[56] «Design and control conversation flow,» [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/bot-framework/bot-design-conversation-flow. [Πρόσβαση 22 May 2017].

[57] «Luis Application Console,» Microsoft Inc., [Ηλεκτρονικό]. Available: https://www.luis.ai. [Πρόσβαση 18 May 2017].

[58] «Don't Repeat Yourself principle,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Don%27t_repeat_yourself.

[59] «Define conversation steps with waterfalls,» Microsoft Inc., 3 May 2017. [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/bot-framework/nodejs/bot-builder-nodejs-dialog-waterfall. [Πρόσβαση 19 May 2017].

[60] «Manage conversation flow with dialogs,» Microsoft Inc., [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/bot-framework/nodejs/bot-builder-nodejs-dialog-manage-conversation. [Πρόσβαση 19 May 2017].

[61] «Recognize user intent,» Microsoft Inc., [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/bot-framework/nodejs/bot-builder-nodejs-recognize-intent. [Πρόσβαση 19 May 2017].

[62] «Principles of Bot Design,» 10 May 2017. [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/bot-framework/bot-design-principles. [Πρόσβαση 17 May 2017].

[63] «Built in entities,» [Ηλεκτρονικό]. Available: https://wit.ai/docs/complete-guide#built-in-entities-link. [Πρόσβαση 19 May 2017].

[64] «Api.ai Entities,» [Ηλεκτρονικό]. Available: https://docs.api.ai/docs/concept-entities. [Πρόσβαση 19 May 2017].

[65] «Builtin entities in LUIS,» [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/luis/pre-builtentities. [Πρόσβαση 19 May 2017].

[66] «API.AI Contexts Guidelines,» [Ηλεκτρονικό]. Available: https://docs.api.ai/docs/guidelines-contexts.

[67] «Wit.ai Speech Api,» 12 February 2014. [Ηλεκτρονικό]. Available: https://wit.ai/blog/2014/02/12/speech-api. [Πρόσβαση 19 May 2017].

[68] «Speech recognition and conversation,» [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/bot-framework/cognitive-services-bot-intelligence-overview#speech-recognition-and-conversion. [Πρόσβαση 19 May 2017].

[69] «Duckling entity recognizer,» [Ηλεκτρονικό]. Available: https://duckling.wit.ai/.

[70] «Glossary of bets offered by UK Bookmakers,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Glossary_of_bets_offered_by_UK_bookmakers.

[Πρόσβαση 20 May 2017].

[71] «About node.js,» [Ηλεκτρονικό]. Available: https://nodejs.org/en/about/. [Πρόσβαση 20 May 2017].

[72] «What is mongodb,» [Ηλεκτρονικό]. Available: https://www.mongodb.com/mongodb-architecture. [Πρόσβαση 20 May 2017].

[73] «Chatbots a bright future in IoT,» [Ηλεκτρονικό]. Available: https://chatbotsmagazine.com/chatbots-a-bright-future-in-iot-93fb615b2286.

[74] ProgrammableWeb, «Wit.AI API,» 20 03 2017. [Ηλεκτρονικό]. Available: https://www.programmableweb.com/api/witai.

[75] C. B. Wit.AI, «wit.ai,» 20 03 2017. [Ηλεκτρονικό]. Available: https://wit.ai/blog/2017/02/10/wit-community-update. [Πρόσβαση 20 03 2017].

[76] D. Sky, «About Us,» 21 03 2017. [Ηλεκτρονικό]. Available: https://darksky.net/about/.

[77] Microsoft Cognitive Services, «Microsoft Bot Framework Overview,» 23 03 2017. [Ηλεκτρονικό]. Available: https://docs.botframework.com/en-us/.

[78] «Luis: Help & Docs,» 27 03 2017. [Ηλεκτρονικό]. Available: https://www.luis.ai/home/help#luis-help.

[79] Microsoft Web, «Azure Cognitive Services,» 3 27 2017. [Ηλεκτρονικό]. Available: https://azure.microsoft.com/en-us/services/cognitive-services/.

[80] A. Barr και E. A. Feigenbaum, The handbook of Artificial Intelligence, volume 1, Stanford, California: HeurisTech Press, 1981.

[81] M. C. Services, «Bot Framework - Principles of bot design,» 10 May 2017. [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/bot-framework/bot-design-principles. [Πρόσβαση 15 May 2017].

[82] M. B. Framework, «Key concepts in the Bot Builder SDK for Node.js,» 10 May 2017. [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/bot-framework/nodejs/bot-builder-nodejs-concepts. [Πρόσβαση 15 May 2017].

[83] M. B. Framework, «Key concepts in the Bot Builder SDK for .NET,» 10 May 2017. [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/bot-framework/dotnet/bot-builder-dotnet-concepts. [Πρόσβαση 15 May 2017].

[84] Api.Ai, «Api.Ai Key Concepts,» [Ηλεκτρονικό]. Available: https://docs.api.ai/docs/key-concepts. [Πρόσβαση 10 May 2017].

[85] «Python NLTK,» [Ηλεκτρονικό]. Available: http://www.nltk.org/. [Πρόσβαση 20 May 2017].

[86] «Flask microservice framework for Python,» [Ηλεκτρονικό]. Available: http://flask.pocoo.org/. [Πρόσβαση 21 May 2017].