**Ανάπτυξη Διαλογικού Πράκτορα με Rasa Framework**

**Περιεχόμενα**

1. Ανάλυση προβλήματος
2. Ανάλυση χρηστών και σενάρια χρήσης
3. Rasa framework
4. Υλοποίηση και Αποτελέσματα
5. Συμπεράσματα

**1. Ανάλυση προβλήματος**

Ο σκοπός της παρούσας εργασίας είναι να δημιουργήσουμε ένα διαλογικό πράκτορα (chatbot) που θα μπορεί να εξυπηρετεί τις ανάγκες των φοιτητών Τμήματος Πληροφορικής και να παρέχει απαντήσεις στα ερωτήματά τους.

Στο πλαίσιο αυτής της εργασίας, θα δούμε πώς μπορεί να γίνει ανάπτυξη ενός διαλογικού πράκτορα (chatbot) με το Rasa framework. Συγκεκριμένα, θα αναπτύξουμε ένα chatbot που θα είναι σε θέση να απαντήσει σε ερωτήσεις σχετικά με το Τμήμα Πληροφορικής του ΑΠΘ, λειτουργώντας σαν γραμματεία. Ωστόσο, για τις ερωτήσεις που δεν σχετίζονται με το Τμήμα Πληροφορικής, το chatbot θα λαμβάνει απαντήσεις από το ChatGPT μέσω API.

Στο πλαίσιο αυτής της εργασίας θα μελετήσουμε την διαδικασία ανάπτυξης ενός chatbot και θα δούμε τι δυνατότητες μπορεί να προσφέρει ενσωμάτωση του ChatGPT στους χρήστες.

**2. Ανάλυση χρηστών και σενάρια χρήσης**

Οι κύριοι χρήστες που θα χρησιμοποιούν το chatbot θα είναι φοιτητές του Τμήματος Πληροφορικής, οι οποίοι θα θέτουν ερωτήσεις σχετικά με το Τμήμα και θα λαμβάνουν σχετικές απαντήσεις.

Παρακάτω αναλύονται κάποιες από αυτές τις ερωτήσεις και οι αναμενόμενες απαντήσεις που θα πρέπει να δίνει το chatbot:

**Γενικές Ερωτήσεις:**

**Ερώτηση**: Γεια σας!

**Απάντηση**: Γεια σας! Πώς μπορώ να σας βοηθήσω;

**Ερώτηση**: Ευχαριστώ για τη βοήθεια.

**Απάντηση**: Αν έχετε ερωτήσεις, είμαι πάντα έτοιμος να σας βοηθήσω.

**Ερώτηση**: Αντίο.

**Απάντηση**: Καλή συνέχεια. Είμαι πάντα έτοιμος να σας βοηθήσω.

**Ερώτηση**: Ποιος είσαι;

**Απάντηση**: Είμαι SecretaryChatbot που παρέχει πληροφορίες σχετικά με το Τμήμα Πληροφορικής του ΑΠΘ.

**Ειδικές Ερωτήσεις:**

**Ερώτηση**: Πώς μπορώ να πάρω βεβαίωση σπουδών;

**Απάντηση**: Μπορείς να πάρεις βεβαίωση σπουδών εδώ: [https://students.auth.gr/#/requests/list](https://students.auth.gr/" \l "/requests/list" \t "_new)

**Ερώτηση**: Ποια είναι η ιστοσελίδα του Τμήματος;

**Απάντηση**: Μπορείτε να βρείτε την ιστοσελίδα του Τμήματος στη διεύθυνση [https://www.csd.auth.gr/](https://www.csd.auth.gr/" \t "_new)

**Ερώτηση**: Ποιο είναι το email της γραμματείας;

**Απάντηση**: Το email της γραμματείας του Τμήματος Πληροφορικής είναι info@csd.auth.gr

**Ερώτηση**: Ποιο είναι το τηλέφωνο της γραμματείας;

**Απάντηση**: Το τηλέφωνο της γραμματείας είναι 2310-99-8420

**Ερώτηση**: Ποια είναι η διεύθυνση της γραμματείας;

**Απάντηση**: Το Τμήμα Πληροφορικής διαθέτει εγκαταστάσεις τόσο στο κέντρο της πόλης, εντός της Πανεπιστημιούπολης του ΑΠΘ, όσο και στην Καλαμαριά. Η διεύθυνση γραμματείας είναι:

Γραμματεία Τμήματος Πληροφορικής ΑΠΘ

Κτίριο Γραμματειών Σχολής Θετικών Επιστημών, Πανεπιστημιούπολη ΑΠΘ

54124 Θεσσαλονίκη

Οι εγκαταστάσεις του Τμήματος στην Καλαμαριά βρίσκονται στην Εθνικής Αντιστάσεως 16.

Περισσότερες πληροφορίες στην ιστοσελίδα του Τμήματος: [https://www.csd.auth.gr/school/access/](https://www.csd.auth.gr/school/access/" \t "_new)

**Ερώτηση**: Ποιες είναι οι ώρες λειτουργίας της γραμματείας;

**Απάντηση**: Η Γραμματεία εξυπηρετεί τις ώρες 08:30-15:30, κατά προτίμηση από απόσταση. Σε περίπτωση που απαιτείται η αυτοπρόσωπη προσέλευση στη Γραμματεία, αυτό γίνεται κατόπιν προγραμματισμένου ραντεβού, στις ώρες 11:00 – 12:30 (τηλ. 2310 998420, email: info@csd.auth.gr).

**Ερώτηση**: Ποιο είναι το ωράριο λειτουργίας της βιβλιοθήκης;

**Απάντηση**: Το φοιτητικό αναγνωστήριο λειτουργεί 08.00-02.00 (μετά τα μεσάνυχτα) Δευ-Κυρ, τηλέφωνο επικοινωνίας 2310 995354. Μπορείς να βρεις περισσότερες πληροφορίες σχετικά με τη Βιβλιοθήκη εδώ: [https://www.lib.auth.gr/el/επικοινωνία](https://www.lib.auth.gr/el/%CE%B5%CF%80%CE%B9%CE%BA%CE%BF%CE%B9%CE%BD%CF%89%CE%BD%CE%AF%CE%B1" \t "_new)

**Ερώτηση**: Ποιες είναι οι ώρες λειτουργίας της λέσχης;

**Απάντηση**: Η διανομή του συσσιτίου πραγματοποιείται σύμφωνα με το παρακάτω ωράριο: Πρωινό: 8:30-10:00 / Μεσημεριανό: 12:00-16:00 / Δείπνο: 18:00-21:00. Μπορείς να βρεις περισσότερες πληροφορίες σχετικά με τη Λέσχη εδώ: [https://www.auth.gr/weekly-menu/](https://www.auth.gr/weekly-menu/" \t "_new)

**Ερώτηση**: Πού μπορώ να βρω τον οδηγό σπουδών;

**Απάντηση**: Μπορείς να βρεις πληροφορίες σχετικά με τον Οδηγό σπουδών εδώ: [https://www.csd.auth.gr/studies/undergraduate/study-guides/](https://www.csd.auth.gr/studies/undergraduate/study-guides/" \t "_new)

**Ερώτηση**: Πού μπορώ να βρω το ακαδημαϊκό ημερολόγιο;

**Απάντηση**: Μπορείς να βρεις πληροφορίες σχετικά με το Ακαδημαϊκό Ημερολόγιο εδώ: [https://www.csd.auth.gr/news/calendar/](https://www.csd.auth.gr/news/calendar/" \t "_new)

**Ερώτηση**: Πού μπορώ να βρω το πρόγραμμα μαθημάτων;

**Απάντηση**: Μπορείς να βρεις πληροφορίες σχετικά με το Πρόγραμμα μαθημάτων εδώ: [https://www.csd.auth.gr/news/schedules/](https://www.csd.auth.gr/news/schedules/" \t "_new)

**Ερώτηση**: Πού μπορώ να βρω το πρόγραμμα εξετάσεων;

**Απάντηση**: Μπορείς να βρεις πληροφορίες σχετικά με το Πρόγραμμα εξετάσεων εδώ: [https://www.csd.auth.gr/news/schedules/](https://www.csd.auth.gr/news/schedules/" \t "_new)

**Ερώτηση**: Πώς μπορώ να κάνω δήλωση μαθημάτων;

**Απάντηση**: Μπορείς να κάνεις δήλωση μαθημάτων εδώ: [https://students.auth.gr/#/registrations/courses/overview](https://students.auth.gr/" \l "/registrations/courses/overview" \t "_new). Από την ηλεκτρονική γραμματεία students.auth -> Δηλώσεις -> Δήλωση μαθημάτων. Δηλώσεις μαθημάτων γίνονται κάθε εξάμηνο (χειμερινό και εαρινό) τις ημερομηνίες σύμφωνα με τις ανακοινώσεις του Τμήματος. Δεν χρειάζεται να γίνει ξεχωριστή δήλωση για επαναληπτικές εξετάσεις Σεπτεμβρίου.

**Ερώτηση**: Πού μπορώ να βρω τις ανακοινώσεις του Τμήματος;

**Απάντηση**: Μπορείς να βρεις ανακοινώσεις του Τμήματος εδώ: [https://www.csd.auth.gr/category/announcements/](https://www.csd.auth.gr/category/announcements/" \t "_new)

**Ερώτηση**: Ποιες είναι οι προσφερόμενες ροές μαθημάτων;

**Απάντηση**: Μπορείς να βρεις πληροφορίες σχετικά με τις Ροές μαθημάτων στον Οδηγό σπουδών εδώ: [https://www.csd.auth.gr/studies/undergraduate/study-guides/](https://www.csd.auth.gr/studies/undergraduate/study-guides/" \t "_new). Οι προσφερόμενες ροές μαθημάτων είναι:

Α. Τεχνητή Νοημοσύνη

Β. Διαχείριση Δεδομένων και Ιστού

Γ. Τεχνολογίες Λογισμικού

Δ. Τεχνολογίες Μάθησης

Ε. Επιστημονική Υπολογιστική

ΣΤ. Δίκτυα Επικοινωνιών και Ασφάλεια Συστημάτων

Ζ. Υπολογιστικά και Επικοινωνιακά Συστήματα

Η. Ψηφιακά Μέσα – Υπολογιστική Νοημοσύνη

Θ. Ρομποτική-Αυτόνομα Συστήματα

Για τη λήψη του πτυχίου είναι υποχρεωτική η κατοχύρωση τουλάχιστον μιας (1) ροής μαθημάτων από τις ακόλουθες που προσφέρονται στο Τμήμα.

**Ερώτηση**: Πού μπορώ να βρω πληροφορίες σχετικά με την πρακτική άσκηση;

**Απάντηση**: Μπορείς να βρεις πληροφορίες σχετικά με την Πρακτική Άσκηση εδώ: [https://www.csd.auth.gr/studies/undergraduate/praktiki-askisi/](https://www.csd.auth.gr/studies/undergraduate/praktiki-askisi/" \t "_new)

**Ερώτηση**: Πού μπορώ να βρω πληροφορίες σχετικά με την εκπόνηση πτυχιακής εργασίας;

**Απάντηση**: Μπορείς να βρεις πληροφορίες σχετικά με την εκπόνηση Πτυχιακής εργασίας εδώ: [https://www.csd.auth.gr/studies/undergraduate/thesis-guide/](https://www.csd.auth.gr/studies/undergraduate/thesis-guide/" \t "_new)

**Ερωτήσεις εκτός πεδίου**

Για τις ερωτήσεις που δεν σχετίζονται με το Τμήμα Πληροφορικής, το Chatbot θα λαμβάνει απαντήσεις από το ChatGPT μέσω API.

**3. Rasa framework**

**4. Υλοποίηση και Αποτελέσματα**

**4.1 Αρχιτεκτονική της εφαρμογής**

Το Chatbot που θα αναπτύξουμε θα έχει παρακάτω αρχιτεκτονική:

Изображение выглядит как текст, диаграмма, зарисовка, снимок экрана

Автоматически созданное описание

Δηλαδή Chatbot παίρνει ως είσοδο ερώτηση του χρήστη , η οποία επεξεργάζεται από το User Message Analysis για να αναγνωριστεί η έννοια και κατηγορία της ερώτησης. Στη συνέχεια, ο Router αποφασίζει αν η ερώτηση πρέπει να απαντηθεί με μια προεπιλεγμένη απάντηση ή αν χρειάζεται η παρέμβαση του LLM. Αν χρειάζεται η παρέμβαση του LLM, τότε γίνεται επικοινωνία μέσω API για να δημιουργήσει την κατάλληλη απάντηση. Τέλος, ο Composer συνδυάζει τις απαντήσεις και τις επιστρέφει στον χρήστη.

Δηλαδή Chatbot δέχεται την ερώτηση του χρήστη αν η ερώτηση είναι σχετίζεται με το Τμήμα Πληροφορικής τότε απαντάει με μια προκαθορισμένη απάντηση, αν όμως η ερώτηση του χρήστη δεν σχετίζεται με το Τμήμα Πληροφορικής, το Chatbot χρησιμοποιεί το API του ChatGPT για να παράγει απάντηση.

Η ενσωμάτωση του ChatGPT API βελτιώνει εμπειρία του χρήστη και μπορεί να προσφέρει πολλές δυνατότητες. Το βασικό πλεονέκτημα αυτής της αρχιτεκτονικής είναι ότι chatbot μπορεί να καλύψει περισσότερο όγκο των ερωτήσεων από πολλές περιοχές και να δώσει ικανοποιητική απάντηση για διάφορα θέματα.

Πιο συγκεκριμένα, το ChatGPT είναι ιδανικό για τις γενικές ερωτήσεις, καθώς έχει εκπαιδευτεί σε ένα μεγάλο όγκο δεδομένων και μπορεί να απαντήσει σε ερωτήσεις από πολλές περιοχές. Ωστόσο, για τις ερωτήσεις που αφορούν ένα συγκεκριμένο τομέα, το chatbot μπορεί να δώσει προκαθορισμένες απαντήσεις, καθώς έχει εξειδικευμένη γνώση και πληροφορία για αυτόν τον τομέα.

Με αυτή την αρχιτεκτονική, μπορούμε να αναπτύξουμε ένα εξειδικευμένο chatbot-βοηθό που μπορεί να δώσει ικανοποιητικές απαντήσεις τόσο για γενικές ερωτήσεις όσο και για ερωτήσεις που αφορούν ένα συγκεκριμένο τομέα. Αυτό σημαίνει ότι μπορούμε να πετύχουμε μια μεγαλύτερη προσωμιότητα ενός ανθρώπου-υπάλληλου που έχει εξειδικευμένη γνώση για κάποιο τομέα αλλά μπορεί επίσης να απαντήσει και σε γενικές ερωτήσεις, προσφέροντας μια ολοκληρωμένη και αποτελεσματική εμπειρία χρήστη.

**4.2 Περιγραφή του backend κομμάτι**

Η ανάπτυξη του Chatbot έγινε στο λειτουργικό σύστημα Windows με την βοήθεια Visual Studio Code IDE.

**4.2.1 Εγκατάσταση του Rasa**

Πρώτα πρέπει να εγκαταστήσουμε Rasa τοπικά με χρήση της Python 3.10 και virtual environment.

Ελέγχουμε την έκδοση της Python με εντολή “python –version”:



Μετά εγκαταστήσουμε τη δυνατότητα δημιουργίας virtual environments σε Python, με την εντολή “python -m pip install virtualenv”:

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Για να δημιουργήσουμε ένα virtual environment, τρέχουμε την εντολή “python -m virtualenv venv”:



Βλέπουμε ότι δημιουργήθηκε ένα φάκελο venv:

Изображение выглядит как текст, снимок экрана, Шрифт, число

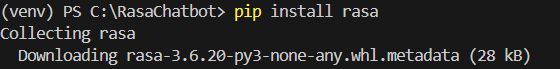
Автоматически созданное описание

Για να χρησιμοποιήσουμε το virtual environment που μόλις φτιάξαμε τρέχουμε την εντολή “venv\Scripts\activate”:

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Και μετά κάνουμε εγκατάσταση Rasa με εντολή “pip rasa install”:



Και επαληθεύουμετην εγκατάσταση με εντολή “rasa –version”:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

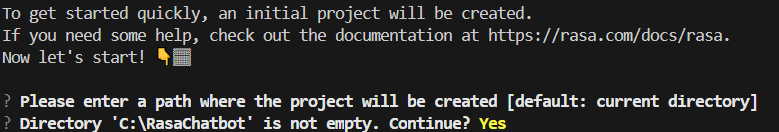
**4.2.2 Δημιουργία του initial project**

Για να φτιάξουμε ένα νέο project χρησιμοποιούμε την εντολή “rasa init”:



Ακολουθούμε τις οδηγίες για να δημιουργήσουμε ένα initial project με moodbot (<https://github.com/RasaHQ/rasa/tree/main/examples/moodbot>) , το οποίο είναι πάρα πολύ απλό chatbot που απλά ρωτάει για το πώς αισθάνεται ο χρήστης και αντιδρά ανάλογα με την απάντηση του χρήστη.

Επιλέγουμε φάκελο στο οποίο θα γίνει δημιουργία του project:



Και μετά βλέπουμε ότι δημιοργήθηκε το template project:

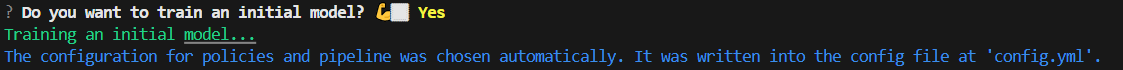


Και δημιουργήθηκαν απαραίτοιτοι φάκελοι:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Μετά μας προτείνουν να κάνουμε training του μοντέλου με εντολή “rasa train”:



Και βλέπουμε ότι δημιουργήθηκε το μοντέλο και αποθηκεύτηκε στο φάκελο models:

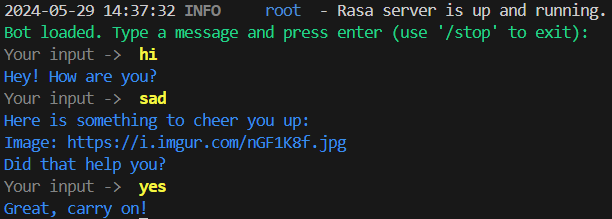
Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Και στη συνέχεια μας προτείνουν να να αλληλεπιδρούμε με chatbot μέσω γραμμής εντολών τρέχοντας εντολή “rasa shell”:



Βλέπουμε ότι ξεκίνησε o server και μπορούμε να συνομιλήσουμε με το chatbot:



**4.2.3 Ανάπτυξη του chatbot**

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Θα ξεκινήσουμε από το domain.yml, ορίζοντας όλα τα intents για κάθε ερώτηση αντίστοιχα

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Και επειτά προσθέτουμε και τις απαντήσεις στα responses:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Τώρα πρέπει να συμπληρώσουμε το nlu.yml που βρίσκεται στο φάκελο data με τα παραδείγματα για κάθε intent:

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Καλό είναι να δώσουμε αρκετό αριθμό παραδειγμάτων για κάθε intent για την καλύτερη αναγνώριση από το NLU.

Μετά από αυτό πρέπει να συμπληρώσουμε αρχεία stories.yml και rules.yml.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Τα rules συνήθως χρησιμοποιύμε για να ορίσουμε σύντομες, καλά καθορισμένες συνομιλίες, όπως χαιρετισμοί, FAQ, απλές φόρμες, που πάντα ακολουθούν την ίδια διαδρομή.

Ενώ τα stories αναπαριστούν πιθανά σενάρια συνομιλιών μεταξύ των χρηστών και του chatbot από τα οποία πρέπει να μάθει το chatbot και όχι ως αυστηρές οδηγίες. Έτσι μπορεί να γενικεύσει και να αντιμετωπίσει νέες και απρόβλεπτες καταστάσεις και να έχει μια πιο φυσική και λογική συνομιλία με τους χρήστες.

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Επίσης το Rasa μας δίνει δυνατότητα να δημιουργήσουμε stories με διαδραστική μάθηση με εντολή “rasa interactive”, με την οποία μπορούμε να δημιουργήσουμε τα stories απλά μιλώντας με το chatbot.

Τώρα πρέπει να συμπληρώσουμε το config.yml που περιέχει pipeline για το NLU και policies.

Για να διαχερίσουμε καλύτερα τα FAQs (συχνές ερωτήσεις) και Chitchat (συνομιλία που δεν σχετίζεται άμεσα με το στόχο, όπως χαιρετισμοί, ευχαριστίες, αποχαιρετισμοί) το Rasa μας προτείνει να χρησιμοποιούμε μια πολιτική διαχείρισης διαλόγου που βασίζεται σε κανόνες (το RulePolicy) και ένα pipeline για να επιστρέψουμε την κατάλληλη απάντηση για μια ερώτηση (το ResponseSelector) (<https://rasa.com/docs/rasa/chitchat-faqs#step-by-step-guide-on-using-response-selector-for-faqs-and-chitchat>):

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Οπότε πλέον μπορούμε να αντικαθιστούμε όλα τα FAQs για τα φοιτιτηκά θέματα με ένα rule:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Προσθέτουμε νέα intents “faq” και “chitchat” στο domain.yml και αλλάζουμε ονόματα για τα intents και responses στο nlu.yml και domain.yml:

Изображение выглядит как текст, Шрифт, снимок экрана, часы

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Με αυτό τον τρόπο μειώνουμε σημαντικά τον αριθμό των επαναλαμβανόμενων rules για FAQ.

Και τέλος συμπληρώνουμε pipeline και policies στο αρχείο config.yml (<https://learning.rasa.com/conversational-ai-with-rasa/pipeline/>) ΄

**4.2.4 Custom actions**

Τώρα θα υλοποιήσουμε ένα custom action που θα μας δίνει την δυνατότητα να επιστρέψει τα στοιχεία του καθηγητή ανάλογα με το επώνυμο που θα δώσει χρήστη.

Στο domain.yml δημιουργούμε entity και slot “teacher”, και επειτά δημιουργούμε μια φόρμα “teacher\_info\_form”:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Στο αρχείο rules.yml δημιουργούμε έναν κανόνα για την ενεργοποιήση της φόρμας και ορίζουμε πότε θα εκτελεστεί την φόρμα. Το βήμα active\_loop: teacher\_info\_form δείχνει ότι η φόρμα θα ενεργοποιηθεί μετά την εκτέλεση του teacher\_info\_form.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Μετά ορίζουμε έναν κανόνα για το τι θα γίνει όταν ο χρήστης θα συμπληρώσει όλα τα slots:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Εδώ βλέπουμε ότι θα εκτελεστούν 2 custom actions: action\_teacher\_info και action\_reset\_slot.

Επίσης στο αρχείο domain.yml με ένα response με όνομα της μορφής utter\_ask\_<slot> μπορούμε να ορίσουμε το μήνυμα με το οποίο chatbot θα ζητάει από τον χρήστη να συμπληρώσει τα απαιτούμενα slots:

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

Ο κώδικας του custom action βρίσκεται στο φάκελο actions στο αχρείο actions.py:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Δημιουργούμε μια κλάση ActionTeacherInfo την οποία θα περιέχει 2 συναρτήσεις:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Συνάρτηση name πρέπει να επιστρέφει αντίστοιχο όνομα, όπως οαθυτό δηλώνεται στο αρχείο domain.yml

Και η συνάρτηση run περιέχει κώδικα που αντιστοιχίζει το επώνυμο του καθηγητή με τη λίστα contact\_info, αν δεν υπάρχει αντίστοιχο επώνυμο στη λίστα το chatbot θα επιστρέφει σχετικό μήνυμα, αλλιώς θα επιστρέφει τα στοιχεία επικοινωνίας του καθηγητή:

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

**4.2.5 Σύνδεση του Chatbot με ChatGPT API**

Οι ερωτήσεις που δεν σχετίζονται με το πεδίο γνώσης του chatbot θα απαντηθούν με την βοήθεια του ChatGPT.

Αρχικά το chatbot θα παίρνει ερώτηση από τον χρήστη και θα την αναλήσει, αν πρόκειται για την out-of-scope ερώτηση, τότε θα καλείται custom action μέσω το οποίο chatbot θα επικοινωνεί με την API και στη συνέχεια θα λαμβάνει απάντηση και θα την επιστρέφει στον χρήστη.

Διαχείριση και αναγνώριση out-of-scope ερωτήσεων θα γίνει με την βοήθεια του config.yml, όπου υπάρχει RulePolice. Εδώ βλέπουμε το όριο 0,4, το οποίο σημαίνει ότι αν η βεβαιότητα του NLU για το intent είναι κάτω από το 0,4, τότε θα εκτελεστεί action\_default\_fallback:

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Εμείς θα κάνουμε την αλλαγή αυτού του action και θα προσθέσουμε την σύνδεση με API του ChatGPT.

Ο κώδικας για το action θα βρίσκεται στο αρχείο actions.py και θα αποτελεί από μια κλάση ActionDefaultFallback, στην οποία δημιουργούμε μια συνάρτηση load\_api\_key για να πάρουμε το API-key που βρίσκεται στο αρχείο secret.json και στη συνέχεια κάνουμε σύνδεση στο ChatGPT και στέλνουμε ερώτηση που έδωσε ο χρήστης:

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Αφού θα πάρουμε απάντηση από το ChatGPT την επιστρέφουμε στον χρήστη στην συνάρτηση run με ένα μήνυμα σημειώνοντας, ότι chatbot έλαβε αυτή την απάντηση από το ChatGPT:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Επίσης προσθέτουμε κάποια παραδείγματα για το τι θεωρούμε out-of-scope ερώτηση στο αρχείο nlu.yml:

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Στο αρχειο rules πρέπει να δημιουργήσουμε 2 κανόνες, όπου θα ορίζουμε ότι στην περίπτωση που χρήστης δίνει out-of-scope ερώτηση πρέπει να τρέξουμε action\_default\_fallback μέσα στο οποίο θα γίνει κλίση του ChatGPT API:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**4.2.6 Επαλήθευση της λειτουργίας chatbot μέσω γραμμής εντολών**

Αφόυ κάναμε όλα τα παραπάνω βήματα μπορούμε να ελέγξουμε την λειτουργία του chatbot.

Αρχικά με εντολή “rasa train” θα εκπαιδεύουμε ένα μοντέλο χρησιμοποιώντας τα δεδομένα και τις ιστορίες του NLU:



Το εκπαιδευμένο μοντέλο θα αποθηκευτεί στο φάκελο “models”:



Στο αρχείο endpoints.yml πρέπει να ορίσουμε τη διεύθυνση στην οποία θα τρέχει action server:

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

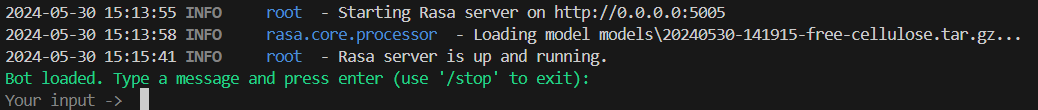
Μετά πρέπει πρώτα να ξεκινήσουμε action server με εντολή “rasa run actions”:



Βλέπουμε ότι server ξεκίνησε κανονικά και είναι διαθέσιμος στη θύρα 5055:



Τώρα μπορούμε σε ένα άλλο τερματικό να τρέξουμε εντολή “rasa shell”, που θα φορτώνει εκπαιδευμένο μοντέλο και θα μας επιτρέπει να μιλήσουμε με chatbot στη γραμμή εντολών:



Παρακάτω θα δούμε ότι τα παραδείγματα που αποδεικναύουν ότι το chatbot δουλεύει σωστά και είναι σε θέση να διαχειρίζεται τις ερωτήσεις πιθανών χρηστών.

Στο πρώτο παράδειγμα ακολουθούμε το σενάριο όπου ο χρήστης χαιρετάει το chatbot, ρωτάει αν είναι όντως chatbot και στη συνέχεια θέτει μια ερώτηση για τον οδηγό σπουδών:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

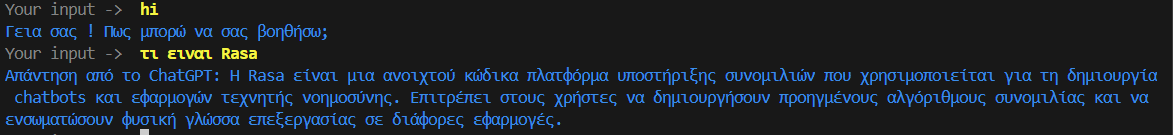
Στο άλλο παράδειγμα θα εξετάζουμε τη λειτουργία custom action για αναζήτης στοιχείων επικοινωνίας του καθηγητή:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Βλέπουμε ότι η φόρμα λειτουργεί σωστά, καθώς αν δεν δίνουμε αρχικά το επώνυμο του καθηγητή το chatbot θα το ζητήσει από τον χρήστη και μόνο τότε θα αναζητήσει στοιχεία επικοινωνίας του αντίστοιχου καθηγητή, ενώ αν το επώνυμο που έδωσε ο χρήστης δεν αντοιστοιχεί σε κάποιο επώνυμο του καθηγητή τότε θα εμφανίζεται σχετικό μήνυμα.

Τέλος θα δούμε την λειτουργία του chatbot στην περίπτωση που χρήστης δίνει ερώτηση που δεν σχετίζεται με το Τμήμα Πληροφορικής, και ο chatbot θα πρέπει να πάρει απάντηση από το ChatGPT:



Βλέπουμε ότι στο action server έγινε επιτυχής σύνδεση μέσω API, και ο διακομιστής του Open AI επιστρέφει “200 ΟΚ” και απάντηση για τον χρήστη εμφανίζεται κανονικά:



**4.3 Περιγραφή του frontend κομμάτι**

Έχουμε τρέξει το chatbot στην γραμμή εντολών, αλλά το Rasa παρέχει πολλές ενσωματωμένες connectors για σύνδεση σε κοινά κανάλια ανταλλαγής μηνυμάτων και φωνής. Μπορούμε επίσης να συνδεθούμε το chatbot με ιστότοπο ή με την εφαρμογή μας με προρυθμισμένα κανάλια REST ή να δημιουργήσουμε το δικό μας προσαρμοσμένο connector (<https://rasa.com/docs/rasa/messaging-and-voice-channels/>).

Εμείς θα ενσωματόσουμε το Chatbot μας στην ιστοσελίδα με την SocketIO κανάλι και πρώτα θα προσθέσουμε credentials στο αρχείο credentials.yml:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Εδώ οι δύο πρώτες τιμές διαμόρφωσης ορίζουν τα ονόματα συμβάντων που χρησιμοποιούνται από το Rasa κατά την αποστολή ή τη λήψη μηνυμάτων μέσω του socket.io.

Αφού ρυθμίσαμε το κανάλι SocketIO, μπορούμε να χρησιμοποιήσουμε το επίσημο Rasa Chat Widget σε οποιαδήποτε ιστοσελίδα.

Απλώς βάζουμε αυτό το κώδικα στο HTML της ιστοσελίδας, ορίζοντας τη διεύθυνση για το Rasa server:



Και έτσι το widget εμφανίζεται στην ιστοσελίδα μας:

Изображение выглядит как текст, снимок экрана, человек, дизайн

Автоматически созданное описание

Μετά ξεκινάμε το action server (με εντολή “rasa run actions”) και rasa server (με εντολή “rasa run -m models --enable-api --cors “\*” ”) για να ελέγξουμε ότι το chatbot μας λειτουργεί κανονικά και στην ιστοσελίδα:

Изображение выглядит как текст, снимок экрана, Веб-сайт, программное обеспечение

Автоматически созданное описание

Θα δούμε πιο προσεκτικά την εντολή “rasa run -m models --enable-api --cors “\*”“:

με εντολή «rasa run» ξεκινάμε το server με εκπαιδευμένο μοντέλο

«-m models» σημαίνει ότι το μοντέλο βρίσκεται στον κατάλογο "models"

«--enable-api» σημαίνει ενεργοποίηση της REST API για το μοντέλο

«--cors "\*”» σημαίνει ότι επιτρέπονται αιτήσεις από οποιαδήποτε προέλευση (domain, protocol, ή port), χωρίς περιορισμούς CORS.

Τρέχουμε αυτή την εντολή για να αποφεύγουμε πιθανά errors που θα εμφανίζονταν αν τρέχαμε μόνο την εντολή “rasa run”:



**4.4 Deployment στο Docker Compose**

Τώρα θα δούμε πως μπορούμε να κάνουμε το Demployment του chatbot μας με την βοήθεια Docker Compose (<https://rasa.com/docs/rasa/docker/deploying-in-docker-compose/>).

Αρχικά στο φάκελο actions δημιουργούμε το αρχείο Dockerfile:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Και επίσης στο φάκελο actions δημιουργούμε αρχείο requirements-actions.txt με όλα τα imports που ορίζονται στο αρχείο actions.py:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Τώρα δημιουργούμε αντίστοιχο Dockerfile για το rasa server:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Επίσης πρέπει να προσθέσουμε αρχείο entrypoint.sh, η οδηγία ENTRYPOINT στο Dockerfile καθορίζει το σενάριο που πρέπει να εκτελεστεί όταν το container ξεκινά. Αυτό επιτρέπει στο container να ρυθμίσει και να εκτελέσει την εφαρμογή Rasa όταν ξεκινά:

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Τέλος δημιουργούμε docker-compose.yml αρχείο, που χρησιμοποιείται για να ορίσουμε και να διαχειριστούμε πολλαπλά Docker containers που εργάζονται μαζί για να λειτουργούν ως μια εφαρμογή:



Επίσης πρέπει να αλλάξουμε το localhost στο όνομα του action\_server στο αρχείο endpoints.yml:

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

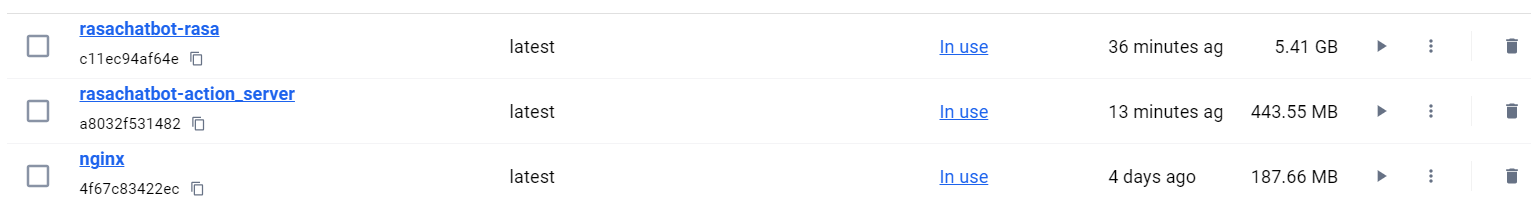
Εκτελούμε εντολή “docker-compose up -d –build” με την οποία το Docker θα δημιουργήσει τις εικόνες για κάθε υπηρεσία που ορίζονται στο αρχείο docker-compose.yml, χρησιμοποιώντας τα αρχεία Dockerfile και θα ξεκινήσει τα containers:

Ανοίγουμε το Docker Desktop, τρέχουμε εντολή “docker-compose up --build"

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

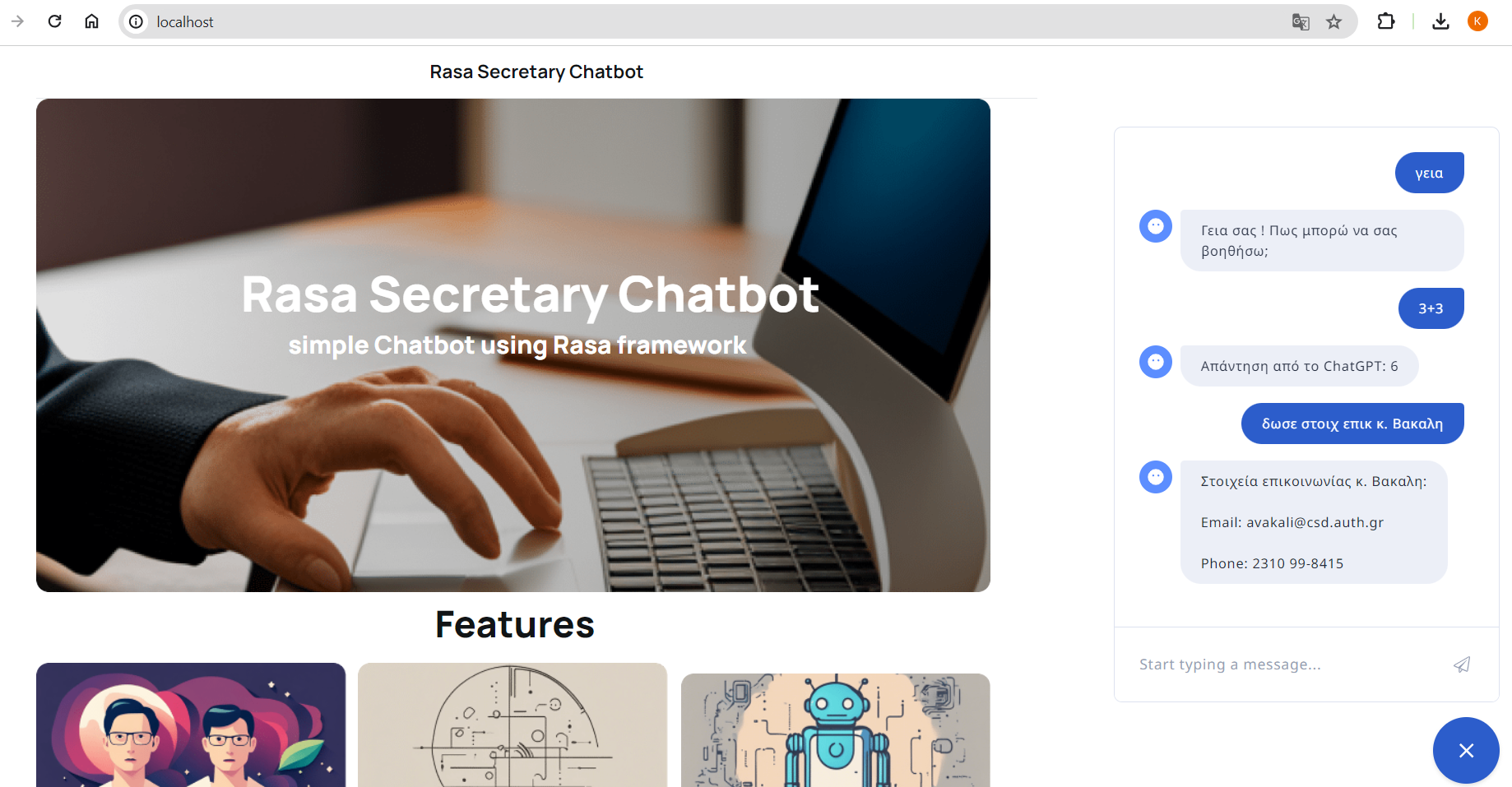
Βλέπουμε ότι δημιουργήθηκαν 3 εικόνες και τα αντοίστοιχα containers:



Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Ανοίγουμε localhost και επιβεβαιώνουμε ότι rasa server και action server λειτουργούν σωστά και το chatbot δίνει απαντήσεις σε όλα τα ερωτήματα του χρήστη:

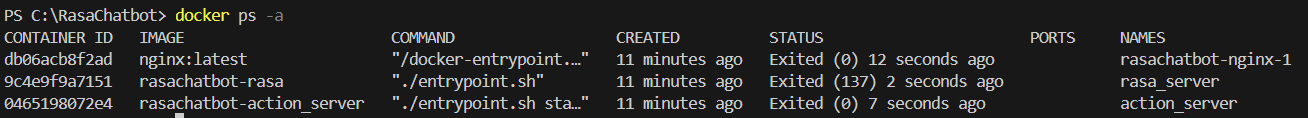


Το επόμενο βήμα είναι να δημοσιεύθουμε εικόνες μας στο Docker Hub, για αυτό δημιουργούμε το repository:

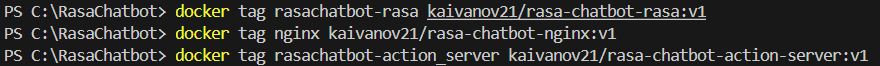
Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

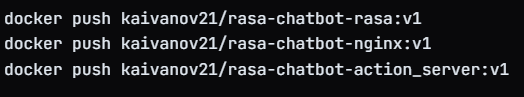
Εμφανίζουμε ολα τα images με εντολή “docker ps -a”:



Βάζουμε τα tags:



Και τώρα μπορούμε να τα ανεβάζουμε στο Docker Hub με εντολή push:



Μπορούμε να ελέγξουμε το repository και να δούμε διαθέσιμες εικόνες και να επιβεβαιώνουμε ότι εικόνες έχουν προστεθεί με επιτυχία:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Βλέπουμε ότι εικόνες έχουν δημοσιευθεί και πλέον είναι διαθέσιμα και μπορούν να εγκατασταθούν τοπικά από οποιονδήποτε με την εντολή pull.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

**5. Συμπεράσματα**

Στο πλαίσιο της παρούσας εργασίας, ολοκληρώθηκε επιτυχώς η ανάπτυξη ενός chatbot με την τεχνολογία Rasa. Το chatbot αναπτύχθηκε σύμφωνα με την αρχιτεκτονική που επιθυμούσαμε. Το chatbot λειτουργεί ικανοποιητικά και καλύπτει ανάγκες των χρηστών, είναι σε θέση να διακρίνει τον τύπο των ερωτήσεων και σε περίπτωση που ερώτηση δεν ανήκει στο πεδίο γνώσης του παίρνει απάντηση από το ChatGPT API.

Επίσης, υλοποιήσαμε την πιλοτική ιστοσελίδα στην οποία ενσωματόσαμε το chatbot μέσω widget, επιτρέποντας στους χρήστες να αλληλεπιδρούν με το chatbot σε ένα φιλικό και εύχρηστο περιβάλλον.

Τελικά, μπορούμε να επιβεβαιώσουμε ότι η Rasa μας δίνει πολλά εργαλεία για την ανάπτυξη του chatbot, όσο και για το deployment. Επίσης, μας δίνει την δυνατότητα σύνδεσης μέσω custom actions με πολλά άλλα εργαλεία μέσω API, επιτρέποντας την επέκταση και την προσαρμογή του chatbot στις ανάγκες των χρηστών.

Το chatbot μπορεί να βελτιωθεί με την προσθήκη νέων λειτουργίων, όπως υποστήριξη δεύτερης γλώσσας ή ενσωμάτωση άλλων συστημάτων για να παρέχει και να παρέχει μια πιο ολοκληρωμένη και εξατομικευμένη εμπειρία χρήστη.