

Egy nyílt forráskódú magyar időpont-egyeztető chatbot

Nagy Soma Bálint, Herdinai Viktor, Farkas Richárd

Szegedi Tudományegyetem, Informatika Intézet
6720, Szeged, Árpád tér 2.
{nagysomabalint,herdinai.viktor98}@gmail.com,
rfarkas@inf.u-szeged.hu

Kivonat A dialógusrendszerek napjainkra a nyelvtechnológia egyik fontos területévé váltak. Jelen munkánkban egy általunk fejlesztett egyszerű magyar nyelvű szöveges chatbotot mutatunk be. A közzétett chatbot egy konzultációs alkalomra történő időpontfoglalást tesz lehetővé. Publikálunk továbbá egy dátum és időpont entitások strukturált kinyerésére fejlesztett könyvtárat, ami az időpontegyeztetésre szolgáló chatbotunk egy kulcsfontosságú része. Munkánk során feltérképeztük a magyar nyelvű chatbot fejlesztésre elérhető eszközöket. Jelen munka az első lépése annak, hogy megértsük, milyen nyelvtechnológiai kihívások állnak az üzleti chatbotok előtt, megismerjük a jelenkori dialógusrendszerek határait.

1 Bevezetés

A chatbotok elképesztő popularitásra tettek szert az elmúlt évtizedben. Míg korábban elsősorban kereskedelmi célú chatbot fejlesztés folyt, az elmúlt néhány évben a számítógépes nyelvészetet is elérte a chatbot láz. Míg korábban csak egyszerű, szabályalapú chatbotokat fejlesztettek, az elmúlt években több ezer nyelvtechnológiai publikáció született, amik a dialógusrendszerek problémáit és az azokra adott megoldásokat tárgyalják.

A magyarul “beszélő” chatbotok piaca nemzetközi szinten mozog. Ezeknek az üzleti chatbotoknak a működése azonban nem publikus, és a magyar nyelvű dialógus rendszereket folytató chatbotok tudományos szakirodalma pedig nagyon ritkás.

Úgy gondoljuk, hogy amennyiben a magyar üzleti chatbotok bonyolultabb, természetes emberi interakciókat akarnak kezelni, akkor szükségük van magyar nyelvi jelenségek kezelésére alkalmas algoritmusokra, amelyek ma még nem minden esetben állnak rendelkezésünkre. Jelen munkánkat az első lépésnek szánjuk azon az úton, hogy megértsük, milyen számítógépes nyelvészeti problémák lépnek fel üzleti dialógus rendszerekben, majd ezekre megoldásokat adjunk.

Annak érdekében, hogy a jelenlegi chatbotok határait és továbbfejlesztési igényét megértsük, egy egyszerű chatbot alkalmazást fejlesztettünk, melynek célja, hogy időpontot egyeztessen valamilyen hivatalos(abb) környezetben. Alkalmazási esetként

¹ A teljesség és sorrendiség igénye nélkül néhány magyar chatbot: talkabot.net, roborobo.hu, hanga.clementine.hu

beszédalapú (telefonos) ügyfélszolgálatok mögötti dialógusrendszert vagy szöveges chatbotot (például szakdolgozó konzultációs időpontot egyeztet a témavezetőjével) tartjuk a szemünk előtt. Mivel telefonos beszédalapú interfész mögött is helyesen kell működnie a rendszernek, ezért ebben a munkánkban csak folyószöveges kommunikációra koncentrálnunk (és például nem vizsgáljuk a chatbotok által gyakran használt kattintással listaelem választás lehetőségét).

Egy másik célunk volt, hogy a rendelkezésre álló nyílt forráskódú megoldásokat feltérképezzük. Úgy gondoljuk ugyanis, hogy a magyar számítógépes nyelvészeti közösségnek nem szabad egyik dialóguságens-fejlesztésére kialakított piaci keretrendszer mellett sem elköteleznie magát, és hiszünk abban, hogy a jövőben létrehozandó magyar nyelvtechnológiai megoldásainkat is nyílt forráskóddal kell elérhetővé tenni.

Cikkünkben bemutatjuk a nyílt forrású technológiákat használó időpont-egyeztető chatbotunkat, valamint az ennek megvalósításához kulcsfontosságú szöveges dátum felimerő, illetve időpont szöveg generálására fejlesztett szabályalapú rendszerünket, amelyek a <https://github.com/szegedai/hun-appointment-chatbot> és <https://github.com/szegedai/hun-date-parser> címen elérhetőek.

2 Kapcsolódó munkák

Legjobb tudomásunk szerint egyetlen tudományos publikáció született eddig, amely magyar nyelvű chatbotot mutat be: Kemény és Recski (2018) egy utazástervező Facebook chatbotot fejlesztett. Célul azt tűzték ki, hogy egy nagyon rövid kérdésre a megfelelő menetrendi választ kérdezzék le a BKK API használatával. Munkájuk számítógépes nyelvészeti vonatkozásai a nagyon tömör, nyelvileg helytelen kérdések megértésére irányultak. Ezzel szemben a mi időpont-egyeztető chatbotunk feltételezi a nyelvileg helyes bemenetet, és a több körben megvalósuló egyeztetésre fókuszál.

A Lara rendszer (Nagyfi, 2018) nyílt forráskódú, magyar chatbot fejlesztésére kidolgozott eszközök gyűjteménye, amely kiterjedt funkcionalitással rendelkezik. A Lara könyvtár képes felhasználói szándék meghatározására, bizonyos entitások kinyerésére, illetve számos olyan magyar nyelvre fejlesztett nyelvi megoldást tartalmaz, amely segítséget nyújt chatbotok fejlesztéséhez.

A chatbotunk elkészítésére végül nem a Lara-t, hanem az univerzálisabban használható Rasa könyvtárat alkalmaztuk, amely a Lara-val szemben elsősorban gépi tanulás alapú, és egy széleskörűen használható keretrendszert biztosít a fejlesztő számára. A dátumok és időpontok kinyerésére vizsgáltuk a Lara használatának lehetőségét, ami tapasztalatunk alapján pontosan ismeri fel ezeket az entitásokat, azonban mivel nem képes időintervallumok, valamint relatív módon megadott dátumok kinyerésére, végül erre a célra saját eszközt fejlesztettünk.

² Például dialogflow.cloud.google.com vagy aws.amazon.com/chatbot

3 Architektúra

A munkánk során egy automatizált, szövegalapú asszisztensek és chatbotok fejlesztésére készült keretrendszert, a Rasa-t (Bocklisch és mtsai., 2017) alkalmaztuk. A Rasa számos természetes nyelv feldolgozási feladat elvégzésére képes, Spacy-hez készült magyar nyelvű NLP modelleknek köszönhetően magyar nyelvvel is kompatibilis. Az időpont-egyeztetéshez dátum, illetve idő típusú entitások kinyerésére volt szükség. A Rasa által is támogatott névelem kinyerésén túl arra volt szükségünk, hogy ezeket az entitásokat strukturált formára hozzuk, az elérhető időpontokat tartalmazó adatbázissal össze tudjuk hasonlítani őket. Ennek a részfeladatnak a megoldására megvizsgáltuk az elérhető szoftvereket, végül azonban saját implementáció készítése mellett döntöttünk. Az általunk megvalósított könyvtár képes időpont és időintervallum entitások felismerésére és normalizálására, valamint a strukturált formában tárolt dátumokból és időpontokból hétköznapi beszédben használatos szöveges reprezentáció generálására, melynek segítségével a chatbottal folytatott párbeszéd hasonlóbbá válik a való életben folytatott beszélgetésekhez.

Az időpontegyeztető chatbotunk és a magyar nyelvű szöveges dátumokat kezelő kódjaink Apache 2.0, illetve MIT licensszel elérhetőek a <https://github.com/szegedai/hun-appointment-chatbot> és <https://github.com/szegedai/hun-date-parser> felületeken.

3.1 A Rasa keretrendszer

A Rasa egy nyílt forráskódú, gépi tanuló keretrendszer, amelynek célja dialógusalapú rendszerek fejlesztése. A Rasa képes gépi tanulás és előre definiált szabályrendszer alkalmazásával

- a felhasználó üzeneteiből a felhasználó szándékát meghatározni,
- kinyerni a szövegből a releváns entitásokat, valamint
- a beszélgetés irányát terelni.

A Rasa számos olyan NLP modellt tartalmaz, amelyre dialógusalapú rendszer fejlesztésekor szükség lehet. A keretrendszer tartalmazza a Rasa kutatói által publikált, általunk is alkalmazott mélytanuláson alapuló DIET (T. Bunk és mtsai., 2020) architektúra implementációját, ami felhasználói szándék és szövegben előforduló entitások egyidejű kinyerésére szolgál. A DIET egy multi-task tanuló, transzformer alapú architektúra. Fejlesztésekor fontos szempont volt a modularitás, ennek köszönhetően számos különböző, előre betanított szóbeágyazással képes együtt működni, illetve egyes részei (intent, illetve entitáskinyerés) külön is használhatók. A DIET a tanítómondatokon a fenti két feladat mellett egy nyelvi modellt is tanul, és ezen három komponens együttes hibájának a minimalizálására törekszik a tanulás során. Munkánk során az általuk bemutatott módon alkalmaztuk a DIET-et, azzal a kivétellel, hogy az entitáskinyerő lépést a DIET helyett mi valósítottuk meg.

Amellett, hogy a Rasa számos problémára kész megoldást kínál, lehetővé teszi bármilyen, fejlesztő által megvalósított feldolgozási lépés integrálását, a meglévő lépések széleskörű konfigurációját.

Azért választottuk a Rasa-t a chatbotunk alapjául, mert nyílt forráskódú, könnyen testreszabható és bő funkcionálitással rendelkezik. A Rasa mellett kipróbáltuk a

Deeppavlov keretrendszer is, azonban ezt jelentősen nehezebben használhatónak találtuk, valamint döntő kritérium volt, hogy a Rasa könnyebben kompatibilissé tehető a magyar nyelvvel, a Deeppavlov esetén erre a rendszer részét képező multilingual BERT (Devlin és mtsai. 2018) használata adhatott volna részben megoldást.

3.2 Természetes nyelv feldolgozási lépések magyar nyelven a Rasa-val

Számos hasonló célra készült szoftverrel ellentétben a Rasa nyelvagnosztikus, az általa használt modellek nyelvtől függetlenül alkalmazhatók. Ahhoz, hogy nyelvspecifikus ismeretekkel rendelkezzen a létrehozott asszisztens, a Spacy természetes nyelv feldolgozást lehetővé tevő könyvtárat, és a hozzá készült nyelvspecifikus NLP modelleket kell használnunk. Mivel a Spacy jelenleg nem rendelkezik hivatalos magyar nyelvű modellekkel, csupán a tokenizálást támogatja, ezért magyar korpuszon tanított, Spacy-vel kompatibilis, függetlenül közzétett modelleket (Orosz Gy., 2019) használtunk. A könyvtár telepítése után a Rasa automatikusan eléri a magyar nyelvű korpuszon tanított CBOW vektorokat, névelem felismerő, illetve lemmatizáló modelleket. Jelenlegi architektúránknak ezek közül a tokenizálás, illetve a folytonos vektortérbeli szóbeágyazás a részei.

Az üzenetből a felhasználó szándékát a transzformeralapú, a Rasa munkatársai által publikált és implementált DIET osztályozót alkalmaztuk, amely egy intentek és entitások egyidejű meghatározására képes architektúra. Mivel a rendszerünk által felismert entitások csupán a dátumok, illetve időpontok voltak, melyre saját entitáskinyerő lépést implementáltunk, ezért a DIET-et csak a felhasználói szándék kinyerésére használtuk. A DIET architektúra képes sűrű és ritka szóbeágyazások együttes használatával javítani a felismert intent pontosságán. A magyar nyelvű Spacy CBOW vektorok szolgálták a sűrű vektorokat, a száznál kevesebb tanítómondaton képzett szózsák modell kimenete pedig a ritka vektorokat. Ezen tanítómondatokat az intentek detektáláshoz határoztuk meg. A BOW elkészültekor használt szavak száma kevés, azonban felhasználását mégis előnyösnek ítéltük, hiszen a felhasználótól is várható időpontfoglaláskor gyakran használatos szavakat, kifejezéseket tartalmazza a tanítóadatbázisunk. Arra számítottunk, hogy a szótáron kívül eső szavak ritkák lesznek a használat során, így a BOW jó kiegészítés lehet a folytonos reprezentáció használata mellé

3.3 Felhasználói felület

Jelenleg a chatbottal CLI-n keresztül lehet beszélgetést folytatni, azonban a Rasa lehetőséget biztosít, hogy az elkészült chatbotot webhook-ként közzétegyük, így népszerű közösségi oldalak üzenetküldő szolgáltatásaival, valamint azonnali üzenetküldésre szolgáló applikációkkal integrálhassuk.

3.4 Szöveges dátumreprezentáció strukturált formára hozása

Ahhoz, hogy a felhasználó által igényelt időpontot összehasonlíthassuk az elérhető időpontok listájával, a természetes szövegben számos formában előforduló dátumra és időpontra utaló kifejezéseket strukturált formára kellett hoznunk. Ehhez két magyar nyelven elérhető, szabályalapú rendszert próbáltunk ki. Az első a Duckling (Facebook, 2016) volt, ami egy nyílt forráskódú, entitások elemzésére szolgáló szoftver. Tapasztalatunk alapján a Duckling nagyon jó pontossággal működik angol nyelven, azonban a magyar dokumentumokon nem képes dátumok robusztus felismerésére. Ennek oka az lehet, hogy angol nyelvre sokkal bővebb szabályrendszer áll rendelkezésre, valamint az, hogy a szoftver nem képes megfelelően kezelni a magyar nyelv agglutinációból adódó sajátosságait.

Ezt követően a Lara (Nagyfi R., 2018) nevű, speciálisan magyar nyelvre fejlesztett könyvtárat vizsgáltuk, amely tapasztalatunk szerint mind dátumok, mind időpontok kinyerésében jobban teljesített, mint a Duckling, azonban a Ducklinggal ellentétben nem képes időintervallumok, napszakok, napok nevével megnevezett dátumok értelmezésére. Vizsgáltuk a Lara által kezelt esetek lehetséges kibővítését, azonban felismertük, hogy a céljainkra egy olyan könyvtár felelne meg igazán, ami képes a dátumot és időt leíró szavakat időintervallumként értelmezni. Ezen tapasztalataink alapján úgy döntöttünk, hogy saját implementációt készítünk dátumok és időpontok strukturált formában való kinyerésére. Az elkészült szoftver Python nyelven készült, tervezésekor figyelmet fordítottunk arra, hogy amellet, hogy a saját céljainkra alkalmas legyen, egy mások által is könnyen felhasználható könyvtárat fejlesszünk.

3.5 Dátumok és időpontok természetes szöveges formára hozása

Ahhoz, hogy a chatbot válaszai a hétköznapi beszédben megszokotthoz hasonlóak lehessenek, a válaszokban megjelenő dátumokat és időpontokat strukturált reprezentációról természetes szöveggé alakítottuk. Mivel nem ismert számunkra erre létező megoldás, mi készítettünk implementációt a probléma megoldására. A megoldást jelentő programot a dátum és időpont kinyerő könyvtárunk részeként publikáljuk.

3.6 Az elérhető időpontok adatbázisa

Jelenleg a chatbot számára elérhető szabad időpontok listáját egy statikus fájlban tároljuk, a felhasználóval való időpont-egyeztetés során innen olvassuk ki azon időpontokat, amelyek a bot számára elfogadhatók, és amelyek közül a felhasználó kérésére ajánlatot tesz a bot. A jövőben természetesen ezt ki lehet majd cserélni valamilyen naptáralkalmazás API-n keresztüli hívására.

4 Eredmények

4.1 Magyar nyelvű természetes szövegben található dátumok kezelése

A korábban kifejtett okok miatt saját szoftverrel végezzük a dátumra, időintervallumra és időpontra utaló entitások kinyerését, valamint strukturált formára hozását. Munkánk során egy reguláris szabályokat alkalmazó, lehetőség szerint az írott természetes nyelvben előforduló jelenségeket kezelő dátumkinyerőt fejlesztettünk. A dátumkinyerő jelen verziója nem támaszkodik természetes szöveg feldolgozását segítő könyvtárakra, a néhány szükséges, szöveges előfeldolgozási lépést mi valósítottuk meg a program részeként.

Ahhoz, hogy széleskörűen használható legyen az elkészült szabályrendszer, valamint biztosítani tudja, hogy a felhasználó bármilyen formában hivatkozhatson a számára megfelelő időpontra, többféleképp előforduló dátum- és időpontalak kezelését kellett megvalósítanunk:

- Dátumok és időpontok konkrét megjelenései a szövegben (például: *2020 november 10., dec. 2, három óra, 15:00, háromnegyed kettő után öt perccel* stb.)
- Névvél megnevezett napszakok (például: *reggel, délben, este* stb.)
- Az aktuális időponthoz képest relatív időpontok (például: *jövő kedden, holnap* stb.)

Ezeket az eseteket szabályokká képeztük le, figyelmet fordítva arra, hogy a szabályrendszerünk szükség esetén bővíthető legyen. A program működése során a felhasználó által kért bemenetre illesztjük az egyes szabályokat. A könyvtár szabályai igyekezzenek lefedni a természetes nyelvben dátum, illetve időpont leírására használt kifejezésmódokat. Amennyiben több szabály is illeszkedik a felhasználói inputra, úgy ezek megfelelő kombinációja képi a kimenetet. Az egyes esetekhez tartozó kimenetet, abban az esetben, amikor sikeresen nyertünk ki a szövegből időintervallumot, egy kezdő- és egy végdátum alkotja.

Mivel a strukturált dátumformátum nem engedi meg, hogy bármilyen információ hiányozzon, ezért ha a felhasználó nem pontosított minden részletet a bemenetként szolgáló szövegben, egyszerű feltevések használatával pótoljuk a hiányzó mezőket. (Például: a *“kedden”* bemenet esetén feltételezzük, hogy az adott heti keddre utal a felhasználó, hiányzó év esetén az aktuális évet feltételezzük, csak időpont megadása esetén az aktuális napot, stb.)

Strukturált formából szöveget szintén szabályok alkalmazásával készítünk. Mivel egy dátumra többféleképpen is lehet hivatkozni (az adott dátumtól számított távolság, konkrét időpont, stb.), ezért ahol lehetséges, több érvényes jelöltet generálunk, melyek mind az adott időpontot írják le. Mivel a természetes nyelvben általában nem másodpercre pontosan fejezzük ki az időpontokat, ezért mi is biztosítjuk annak lehetőségét, hogy a felhasználó meghatározhassa, hogy milyen pontosságot fejezzen ki a generált szöveg.

Annak érdekében, hogy ne tartalmazzon redundáns információt a generált szöveg, és így a természetes beszédben használthoz hasonló kimeneteket képezhessünk, csak azokat a dátum által hordozott információkat képeztük le explicit módon szöveggé, amelyek feltétlenül szükségesek a dátum beazonosításához. A dátumkinyerőnél a

hiányos dátumrészekre alkalmazott feltevéseinket itt is érvényesnek tekintettük. (Például egy dátumhoz generált szöveg nem szükségszerűen tartalmazza az évet, ha aktuális évi dátumról van szó.)

4.2 A chatbot által kezelt szcenáriók

A chatbot képes több forgatókönyvet is kezelni, ezekhez elsősorban felhasználói szándékokat kellett definiálni. A jelenleg definiált intentek:

- üdvözlés,
- időpontfoglalási kérés,
- dátum megadása az igényelt időponthoz,
- igényelt idő megadása az időponthoz,
- elérhető időpont kérése a bottól, valamint
- a beszélgetés lezárását igénylő intent.

Ahhoz, hogy a felhasználói szándékokat felismerjük, tanító mondatokat kell megadni a Rasa-nak, arra, hogy milyen lehetséges felhasználói üzenetek tartoznak az egyes intentekhez. Tapasztalataink alapján ilyen alacsony számú intentnél 7-10 mondat esetén már megfelelő arányban azonosítja a helyes szándékot, de minél többet adunk meg, annál alacsonyabb a hibaarány. A chatbot tesztelése során azonosítottunk olyan lehetséges eseteket, amikor nem voltunk elégedettek a predikált szándékkal, ilyen esetben a tanítóhalmazt kibővítve a hibásan értékelt üzenetekhez hasonlókkal azt tapasztaltuk, hogy újbóli tanítás után a chatbot könnyen adaptálódott. Hasonló hatás érhető el a Rasa részét képező interaktív tanulást biztosító lehetőséggel, amelyet a chatbot jelenlegi verziójában nem alkalmaztunk. A chatbot a szándék felismerése után a beszélgetést ún. akciók végrehajtásával folytatja. Az akciók lehetnek konstans válaszok, de bonyolultabb esetben egyedi akciók implementálására is lehetőség van, ilyenkor egy meghatározott felépítésű osztály megvalósításával vihetünk komplexebb logikát a chatbot viselkedésébe. A chatbotunkhoz három egyedi akciót valósítottunk meg:

- Igényelt időpont összevetése a szabad időpontokkal, slot-ba mentése, visszajelzés arról, hogy az igényelt időpont megfelelő-e
- Ajánlatadás a szabad időpontok közül
- Slotok kiürítése

Azt, hogy mely intent után melyik akció hajtódjon végre, szabályok határozzák meg. Néhány egyszerű szabályt alkalmaztunk, további feltételek megadása nélkül, melyek leképzést adnak meg az összes intent és egy vagy több akció között.

A bot tanítása előtt lehetőség van „sztorik” megadására. Egy sztoriban egy lehetséges beszélgetést adunk meg absztrakt szinten, vagyis azt, hogy a beszélgetés során mely intentek kerülnének felismerésre, mely akciók futnának le, ezzel segítve a dialógusmodell tanulását. Fontos megjegyezni, hogy a sztorikban szereplő egymást követő intentek és akciók nem mondhatnak ellent a szabályokban leírtaknak, különben a tanítás sikertelen lesz. Több sztorit is megadtunk, amelyek forgatókönyvek egyes egyedi akciók használatára. Példa egy sztorira, ahol a dátum és az időpontot is megfelelő a chatbotnak:

```

- story: happy path
steps:
- intent: greet
- action: utter_greet
- intent: appointment_request
- action: validate_appointment
- intent: say_date
- slot_was_set:
  - date: "2021-02-17"
- action: validate_appointment
- intent: say_time
- slot_was_set:
  - time: "13:00"
- action: validate_appointment
- intent: request_end
- action: utter_bye

```

Látható, hogy egy sztori több valós beszélgetést lefed, hiszen a fenti sztori csak absztrakt szinten definiálja a beszélgetés folyását.

4.3 Időpont egyeztetés több körben

Olyan időpontegyeztetési folyamatokat modelleztünk, ahol egy tágabb időintervallumból kiindulva, a felhasználó és a bot a számukra alkalmas időtartamot néhány lépésben egyetlen konkrét időpontra szűkítjük, ezért ahol lehetséges, a fejlesztett rendszer időintervallumok kinyerésére törekszik a bemenetként szolgáló természetes szövegből.

Miután az opcionális kölcsönös üdvözlés megtörtént, a felhasználó jelezheti időpontfoglalási szándékát. Ha ez az üzenet nem tartalmazza a szükséges információkat a kívánt időpontra vonatkozóan, egy többfordulós folyamat kezdődik, amely során a felek kölcsönösen megfelelő időpontot keresnek.

Ha a felhasználó nem adta meg az időpontfoglalási szándékával együtt a kívánt dátumot, a chatbot rákérdez, hogy mikor felelne meg a felhasználónak. Ezután a beszélgetés szerteágazhat a válaszok alapján. Az említett dátumot a bot elfogadhatja, vagy elutasíthatja, dátum-ajánlatkérés esetén pedig két dátumot javasol. A felhasználó által kért dátum validálása a három akción egyikével történik, ezzel ellenőrizzük, hogy az adatbázisban szerepel-e az adott dátum, amennyiben igen, egy slotban eltároljuk, különben egy másik dátumot kérünk a felhasználótól. A dátumot tartalmazó slot beállítása után a bot “továblép” az időpont rögzítésére. A dátumhoz hasonlóan itt is az elfogadás, elutasítás, ajánlás esete következhet be. Az egyeztetés bármely része alatt a felhasználó kezdeményezheti, hogy a már megbeszélt dátum és időpont helyett találjanak egy másikat. Ennek hatására a slot-ban tárolt érték törlődik, megfelelő dátum esetén az új kerül beállításra. Sikeres időpontfoglalás után a chatbot megerősítésképp visszaolvassa a slotokból a rögzített időpontot mondatba foglalva.

4.4 Példa a chatbot működésére

Az alábbiakban a chatbot működését szemléltetjük a beszélgetés során háttérben zajló folyamatok magyarázatával kiegészítve. A beszélgetés tartalmaz példát a felhasználó

által ajánlott, a chatbot számára megfelelő, valamint nem megfelelő időpontokra, a chatbot által ajánlott időpontra, a felhasználó kérésére történő időpontfelülírássra. A beszélgetésben megfigyelhető, hogy a chatbot képes a beszélgetésből az időpont megállapításához szükséges dátum, illetve idő típusú entitások robusztus kinyerésére.

Felhasználó:	Szia	<i>(üdvözlés intent detektálva)</i>
Chatbot:	Szia! Miben segíthetek?	
Felhasználó:	Időpontot szeretnék foglalni	<i>(időpont foglalás intent detektálva, időpont nem nyerhető ki az üzenetből)</i>
Chatbot:	Ókés. Mikor lenne jó?	
Felhasználó:	Te mikor érnél rá?	<i>(időpont ajánlást kérő intent detektálva)</i>
Chatbot:	Legközelebb ma és holnap érek rá.	<i>(elérhető nap ajánlása az időpontok listája alapján)</i>
Felhasználó:	Akkor találkozunk <u>kedden</u>	<i>(entitás detektálása, normalizálása, összevetése az időpont listával, slot feltöltése az entitás értékével, vizsgált intervallum: aktuális hét kedd, a beszélgetést követő nap)</i>
Chatbot:	Ráérek holnap. Mikor lenne jó aznap?	
Felhasználó:	Igazából egész <u>délelőtt</u>	<i>(entitás detektálása, normalizálása, összevetése az időpont listával, vizsgált intervallum: kedd 10-12 óra)</i>
Chatbot:	Sajnos nem érek rá ekkor... Egy másik időpont esetleg?	
Felhasználó:	Akkor amikor te ráérsz	<i>(időpont ajánlást kérő intent detektálva)</i>
Chatbot:	Holnap ráérek reggel 9-kor és délután 1-kor.	<i>(elérhető időpont ajánlása az időpontok listája alapján)</i>
Felhasználó:	Találkozunk <u>reggel kilenckor</u>	<i>(entitás detektálása, normalizálása, összevetése az időpont listával, slot feltöltése az entitás értékével, vizsgált intervallum: kedd 9 óra)</i>
Chatbot:	Rendben, akkor holnap reggel 9-kor találkozunk.	
Felhasználó:	Lehetne inkább a <u>szerda</u>?	<i>(időpont felülírását kérő intent detektálva, entitás kinyerés, normalizálás, összevetés az elérhető időpontokkal, slot feltöltése, vizsgált intervallum: aktuális szerda)</i>
Chatbot:	Ráérek ezen a héten szerdán. Mikor lenne jó aznap?	
Felhasználó:	Bármikor	<i>(időpont ajánlást kérő intent detektálva)</i>
Chatbot:	Ezen a héten szerdán ráérek délután 2-kor.	
Felhasználó:	Rendben, legyen <u>14 órakor</u>	<i>(entitás kinyerés, összehasonlítás az elérhető időpontokkal, slot feltöltés, vizsgált intervallum: szerda 14 óra)</i>
Chatbot:	Rendben, akkor ezen a héten szerdán délután 2-kor találkozunk.	

Felhasználó: **Köszí** *(elköszönés intent detektálva)*
Chatbot: **Viszlát!**

Egy lehetséges alternatív út az üdvözlés után:

Felhasználó: **Időpontot szeretnék foglalni vasárnapra** *(foglalás intent, időpont kinyerés, vizsgált intervallum: aktuális vasárnap)*
Chatbot: **Sajnos nem érek rá akkor... Egy másik esetleg?**
Felhasználó: **Mikor vagy szabad?** *(időpont ajánlást kérő intent detektálva)*
Chatbot: **Legközelebb ma és holnap érek rá.**

Miként korábban is utaltunk rá, vannak olyan esetek, amelyekben a chatbot nem kezeli megfelelően a felhasználói inputot, előfordulhat, hogy hibásan vagy egyáltalán nem detektál intentet. Erre részben megoldásként szolgált az, hogy a chatbot olyan esetekben, amikor minden lehetséges intenthez tartozó konfidencia érték alacsony, egy alapértelmezett választ ad a felhasználónak, melyben arra kéri, hogy üzenetét ismételje meg más formában.

Példák alacsony konfidencia értékre:

Chatbot: **Okés. Mikor lenne jó?**
Felhasználó: **Most éppen ráérek.** *(alacsony konfidencia érték, fallback intent)*
Chatbot: **Elnézést, nem teljesen értettem. Át tudnád fogalmazni?**
Felhasználó: **Január 13-án ráérek.** *(foglalás intent, entitás kinyerés, vizsgált intervallum: aktuális nap)*
Chatbot: **Ráérek ma. Mikor lenne jó aznap?**
Felhasználó: **Szia** *(üdvözlés intent detektálva)*
Chatbot: **Szia! Miben segíthetek?**
Felhasználó: **Programozó szeretnék lenni.** *(nagyon alacsony konfidencia érték, fallback intent)*
Chatbot: **Elnézést, nem teljesen értettem. Át tudnád fogalmazni?**

5 Tapasztalatok és jövőbeli tervek

Jelenlegi tapasztalataink alapján a nyílt forrású Rasa keretrendszer alkalmas lehet komplexebb dialógusrendszerek megvalósítására is. Az egyedi akciók alkalmazásával könnyedén lehet a felhasználó kérésére az entitásoktól függő, dinamikus válaszokat adni. Használatukkal összeköthető a chatbot – API-kon keresztül – adatbázisokkal és egyéb webes szolgáltatásokkal.

A dátumfelismerés és a szöveges dátum generálással kapcsolatban azt tapasztaltuk, hogy a céljainkra már viszonylag egyszerű szabályrendszer is megfelelő, természetesen ahogyan a szabályok számát növeltük, úgy nőtt a rendszer robusztussága is.

A chatbotunk jelenleg nem képes bonyolultabb nyelvi szerkezetek feldolgozására. Eddigi tapasztalataink alapján, a legfontosabb szükséges magyar számítógépes nyelvészeti megoldások, amelyekre szükség lesz bonyolultabb szerkezetek megértéséhez:

- negáció detektálása (például: *“szerdán nem érek rá”*)
- módosító szerkezetek kezelése (például: *“szerda kivételével bármelyik nap jó”*)
- helytelen írásmódú, hiányos válaszok kezelése

A Rasa számos olyan képességgel rendelkezik, amelyeket a chatbotunk jelen verziójában nem használtunk, azonban a későbbiekben segítségünkre lehetnek, mint az interaktív tanulás a dialógusmodell fejlesztésére, tesztesetek megadása a dialógusrendszerre, vagy az integrációs lehetőségek közösségi platformokon.

A későbbiekben terveink közt szerepel a chatbot viselkedésének és teljesítményének kiértékelése emberi kérdőívek kitöltésével (Deriu és mtsai, 2020), valamint a chatbot meglévő működésének kibővítése, új funkciók hozzáadása. Ilyen funkció lehet, hogy a chatbot önmagától, külön kérés nélkül is képes legyen időpontot ajánlani. A jelenlegi statikus elérhető időpontok tárolását is leváltanánk egy valódi adatbázisra, ezzel a beszélgetés során egyeztetett időpontot foglaltként tudnánk megjelölni, elkerülve azt az esetet, hogy egy időpontot többen is lefoglaljanak. Továbbá a hitelesebb felhasználói élmény érdekében bővítenénk a válaszsablonok számát, így elkerülve az ismétlődő szerkezetű válaszokat a chatbot részéről.

Köszönetnyilvánítás

Farkas Richárd kutatási munkáját a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal Mesterséges Intelligencia Nemzeti Kiválósági Programja támogatta a 2018-1.2.1-NKP-2018-00008 azonosítójú projekt keretében.

Nagy Soma Bálint és Herdinai Viktor munkáját a "Integrált kutatói utánpótlás-képzési program az informatika és számítástudomány diszciplináris területein" című, EFOP-3.6.3-VEKOP-16-2017-0002 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

A publikációban szereplő kutatást az Innovációs és Technológiai Minisztérium és a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal támogatta a Mesterséges Intelligencia Nemzeti Laboratórium keretében.

Hivatkozások

- T. Bocklisch, J. Faulkner, N. Pawlowski, A. Nichol: Rasa: “Open Source Language Understanding and Dialogue Management”. *CoRR* abs/1712.05181 (2017)
- Burtsev M. és mtsai. “DeepPavlov: Open-Source Library for Dialogue Systems” Proceedings of ACL 2018, System Demonstrations. (2018)
- Jan Deriu, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, Mark Cieliebak: “Survey on evaluation methods for dialogue systems” *Artificial Intelligence Review* (2020)
- J. Devlin és mtsai. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. *CoRR* abs/1810.04805 (2018)
- Orosz Gy., “Hungarian models for spaCy”. <https://github.com/oroszgy/spacy-hungarian-models>. (2019).
- Facebook, “Duckling”. <https://github.com/facebook/duckling>. (2016)
- Kemény Boldizsár, Recski Gábor: “Természetes nyelvi interfész menetrend- és utazástervező szolgáltatásokhoz” Magyar Számítógépes Nyelvészeti konferencia (2018)
- Nagyfi R, “Lara” <https://github.com/sedthh/lara-hungarian-nlp>. (2018)
- T. Bunk, D. Varshneya, V. Vlasov, and A. Nichol: “DIET: Lightweight Language Understanding for Dialogue Systems”. (2020)