Óbudai Egyetem

Doktori (PhD) értekezés

Eljárások, modellek és módszerek beszéd- és mozgáskorlátozott személyek támogatására intelligens térben

Simon-Nagy Gabriella

Témavezetők:

Dr. Várkonyiné Prof. Dr. Kóczy Annamária

Dr. Kutor László

Alkalmazott Informatikai és Alkalmazott Matematikai Doktori Iskola

Budapest, 2021. április 12.

Szigorlati bizottság:

Ladislav Főző, docens, doc. Ing. PhD
Takács Márta, egyetemi docens, PhD
Fullér Róbert, egyetemi tanár, DSc

Nyilvános védés teljes bizottsága:

Sima Dezső, professor emeritus, DSc
Horváth László, professor emeritus, CSc
Tóthné Laufer Edit, egyetemi docens, PhD
Lovassy Rita, egyetemi docens, PhD
Kovács Szilveszter, egyetemi docens, PhD
Dombi József, professor emeritus, DSc
Tóth János, professzor, PhD

Nyilvános védés időpontja:

Tartalom

Ábrajegyzék

Táblázatjegyzék

Köszönetnyilvánítás

Elsősorban szeretném megköszönni témavezetőim, Dr. Várkonyiné Prof. Dr. Kóczy Annamária és Dr. Kutor László útmutatását és támogatását. Köszönöm mindazon kollégáimnak a rám áldozott idejét, akik hasznos tanácsokkal láttak el a disszertáció tartalmi felépítését és szerkesztését illetően. Emellett külön köszönettel tartozom Dr. Jakab Gábornak, az Uzsoki utcai kórház osztályvezető főorvosának, aki a Sclerosis multiplex országosan elismert szakértőjeként segítséget nyújtott a betegség orvosi aspektusának megismerésében. Végül, de nem utolsó sorban köszönöm a családom támogatását és türelmét.

Bevezetés

A súlyosan mozgássérült személyek asszisztenciához való hozzáférése, a körülményekhez mérten önálló életvitel lehetőségei Magyarországon nem nevezhetők ideálisnak. Mind az utcák, mind a középületek akadálymentessége számos kívánnivalót hagy maga után. Bár a középületek akadálymentesítéséről törvény rendelkezik, ez azonban csak az újonnan épített vagy felújítandó épületekre vonatkozik. Számos régi épület csak részben akadálymentes, vagy a bejáratnál található lépcső miatt egyáltalán nem tekinthető kerekesszékkel járhatónak. Sajnos az ország szegényebb területein nem is várható javulás ezen a téren a közeljövőben, de a fővárosban is bőséggel találhatók még problémás középületek.

Az otthon falai között sem egyszerű a helyzet, hiszen aki nagyrészt képtelen a (koordinált) mozgásra, annak a legegyszerűbb tevékenységekhez is segítségre lenne szüksége, de az otthonápolás áldozatkész hozzátartozók és megfelelő anyagi lehetőségek mellett is nagyon nehezen biztosítható a nap 24 órájában.

Mindezt tetézi, hogy több olyan betegség is van, amely a mozgási képességek leépülése mellett a beszédminőséget is rontja (mind a beszédben résztvevő izmokra és idegekre, mind a légzésre való negatív hatás által). Ez nem csak a segítőkkel való kommunikációt nehezíti meg, de a beszédfelismeréssel vezérelhető segítő rendszerek használatát is ellehetetleníti hosszú távon.

Az önálló(bb) életvitel informatikai támogatását szolgáló rendszerek kisebb-nagyobb szolgáltatáskínálattal már régóta léteznek, de ezek elterjedése a könnyen hozzáférhető és megfizethető kategóriában csak akkor várható, ha kényelmi szolgáltatásként az ép felhasználók is tömegesen keresnék ezeket. Ebbe az irányba mutat például az Amazon Echo eszközök népszerűsége, a potenciális veszélyekre pedig remekül rávilágítanak az ugyanezen szolgáltatással kapcsolatban napvilágot látó mókás, bosszantó, vagy akár jelentős anyagi károkat okozó esetek.

A jelenleg elérhető, beszédfelismeréssel működő digitális asszisztens alkalmazások (Amazon Alexa, Apple Siri, Google Asszisztens) beszélőfüggetlen modellekkel működnek, amelyeknek a tanítására a felhasználók beszédét használják fel. A rendelkezésre álló rengeteg adat és nagy számítási kapacitás rendkívül jó minőségű beszédfelismerést eredményez, és angol nyelven már remekül használható a szolgáltatás. (Bár az agglutináló nyelveken való diktálás még hagy kívánni valót maga után.) De kezdetben nem volt ilyen jó a helyzet. A Google (magyar nyelvű) hangos keresés szolgáltatásának indulásakor például még látványos különbséget találtam a felismerési pontosságban a gyakran keresett, illetve a ritkán vagy egyáltalán nem keresett kifejezésekre. Míg a gyakran keresett kifejezéseket változatos akusztikus körülmények között is magabiztosan felismerte, addig a ritkán keresetteket még csendben is hajlamos volt tévesen azonosítani.

Belátható, hogy ha egy olyan szolgáltatás, mint az Amazon Alexa egy ép beszédű személyt is képes változatos módokon félreérteni, akkor egy súlyosan beszédfogyatékos felhasználó, akit a saját közvetlen hozzátartozói is alig értenek már meg, nem fogja tudni működtetni ezt a rendszert.

A kereskedelemben elérhető okosotthon rendszerek tudása korlátozott, éppen azért, mert a kellően széles ügyfélbázist jelentő ép felhasználókat célozzák meg. Magától értetődő, hogy egy ilyen rendszer képes egy egyszerű kimondott parancsszóra hangerőt állítani; talán még elképzelhető, hogy át tudja kapcsolni a tévét egy másik csatornára (bár a hagyományos televíziózást lassan, de biztosan kiszorítják az on-demand streaming szolgáltatások); de egy boltban vásárolt eszköz soha nem fogja tudni a beteg speciális ágyán a fejtámla magasságot állítani.

A célkitűzésem olyan eljárások, modellek és módszerek tervezése volt, amelyek mind a nyilvános térben, mind otthon segítenek áthidalni a fenti problémákat a meglévő megoldások és eszközök fejlesztésével, kiegészítésével.

A részben akadálymentes épületek problémájának áthidalására olyan beltéri akadálymentes útvonaltervező megoldást kerestem, amely speciális hardver telepítése nélkül lehetővé teszi az egyén igényeire szabott, általa (a lehető legkevesebb segítséggel) bejárható útvonal megtalálását.

A súlyosan mozgás- és beszédfogyatékos felhasználók okosotthon rendszereinek megfelelő felépítésére pedig olyan irányelveket terveztem, amelyek a speciális igények mellett is biztonságos használatot tesznek lehetővé, valamint a beszédfelismerés ismert módszereinek olyan jellegű továbbfejlesztését adtam meg, amely kifejezetten a torzult, és a betegség következtében lassan változó beszéd megbízható felismerését teszi lehetővé.

Az értekezés felépítése a következő: az 1. fejezet a beltéri útvonaltervezés ontológia alapú megoldásának akadálymentes útvonalakra való kiterjesztését, majd a címkézett tulajdonsággráf adatmodellre való áttérés motivációit és az általam kidolgozott módjait mutatom be. A 2. fejezetben a súlyosan mozgás- és beszédfogyatékkal élő személyek számára készült intelligens otthon rendszerek megbízhatósági és (kisebb részben) biztonsági megfontolásaival foglalkozom. A 3. fejezetben az előbbi témának kifejezetten a beszédfelismerőre vonatkozó vetületét vizsgálom, különös tekintettel a dizartria és a fokozatosan romló beszédminőség okozta nehézségekre.

A 4. fejezetben összefoglalom az eredményeimet, valamint kitérek azok alkalmazási lehetőségeire és a jövőbeli kutatási irányokra is. Az 5. fejezetben felsorolom a tézisekhez kapcsolódó és egyéb publikációimat.

1 Beltéri akadálymentes navigáció ontológia alapján

1.1 Motiváció

A kültéri navigáció napjainkban széles körben elérhető, mind autós és tömegközlekedési, mind gyalogosforgalomban, és többnyire elektronikus térképek és GPS koordináták alapján működik. Beltérben azonban nem érhető el megbízható GPS jel, ezért a jelenlegi, beltéri pozicionálásra használt megoldásokhoz kiegészítő rádiófrekvenciás berendezések telepítése szükséges, amely nagyobb épületek esetében meglehetősen költséges. Létezik azonban ehelyett más megközelítés, amely közelebb áll ahhoz a természetes nyelvi megoldáshoz, ahogyan az emberek útbaigazítást adnak egymásnak, azaz jól felismerhető tájékozódási pontok sorozatának és közöttük a haladási irányoknak a megadásával. Ez a navigációs módszer a lehető legolcsóbb és legegyszerűbb, és nagyobb épületekre, épületkomplexumokra (amilyenek a kórházak) is jól adaptálható.

Mindemellett az akadálymentes beltéri útvonalak tervezése is lehetséges, ha az épületek egyes részeinek akadálymentességi tulajdonságairól van információ a birtokunkban. Hiszen mindössze az útkereső algoritmust kell olyan módon módosítani, hogy ne vegye figyelembe a tervezés során az épület azon részeit, amelyek nem akadálymentesek. Az azonban, hogy egy folyosó vagy helyiség, vagy két emelet közötti átjárás akadálymentes-e, korántsem egyszerű kérdés.

A célom a részben akadálymentes épületek problémájának áthidalására olyan beltéri akadálymentes útvonaltervező megoldás megalkotása volt, amely speciális hardver telepítése nélkül lehetővé teszi az egyén igényeire szabott, általa (a lehető legkevesebb segítséggel) bejárható útvonal megtalálását. Ezt a feladatot egy a kutatócsoportunk által korábban megvalósított megoldásra építve kellett megoldanom. A két fő követelmény a testreszabhatóság érdekében a lehető legrészletesebb és legrugalmasabb adattárolás lehetővé tétele, valamint a gyakorlatban tolerálható idő alatt (stabilan 1 másodpercnél kevesebb) lefutó keresések biztosítása volt.

1.1.1 Akadálymentesség a törvényben és a gyakorlatban

Amikor az épített objektumok akadálymentességéről beszélünk, a megállapítások többnyire valamilyen törvényi előíráson alapulnak. Számos országban van hatályban esélyegyenlőségi jogszabály, amely biztosítja (biztosítaná) a közszolgáltatásokhoz, és így a középületekhez való egyenlő, akadálymentes hozzáférést. Ezek jellemzően vagy az emberi jogi törvények, vagy az építési előírások között találhatók. Az előbbire példa az ADA (American Disability Act – az Amerikai Egyesült Államok esélyegyenlőségi törvénye), amely megköveteli, hogy a középületek (beleértve az egészségügyi intézményeket is) a fogyatékkal élők számára használhatók legyenek [1]. Magyarországon a 253/1997. (XII. 20.) Kormányrendelet az országos településrendezési és építési követelményekről (OTÉK) által meghatározott definíciók és méretek az irányadók. Az OTÉK rendelkezik arról, hogy az újonnan épített és felújított középületek kerekesszékkel közlekedők számára akadálymentesek legyenek azáltal, hogy alternatív közlekedési módot biztosítanak a lépcsők mellett rámpa, lift vagy kerekesszék-lift formájában, az ajtók belső szélessége legalább 90 cm, és elérhetők benne akadálymentes mosdók [2].

Eszerint például egy épület akkor tekinthető teljesen akadálymentesnek, ha abban minden ajtó belső szélessége legalább 90 cm. Ez a kritérium garantálja azt, hogy bármilyen fajta kerekesszék akadálytalanul átjuthasson rajta. Ezért aztán a kézenfekvő megoldás az, ha a navigáció során a kerekesszékes útvonalak tervezésénél is ezt a szabályt tartjuk be, és egy ajtó akkor tekintendő „kerekesszék számára akadálymentesnek”, ha a mérete megfelel a törvényi előírásnak.

A törvényi szabályozás azonban csak az újonnan épített vagy felújítandó épületekre vonatkozik. Számos régi magyar középület van, amelyek csak részben akadálymentesek, és nem is várható változás ezen a téren a közeljövőben. Ahhoz, hogy a módszer praktikusan használható legyen a mindennapi szituációkban, az épület számítógépes ábrázolását ki kell terjeszteni a helyiségek, folyosók és POI-k (Point of Interest) részletes leírásával.

Néhány felhasználó számára a törvényben megszabott kritériumok szükségtelenül korlátozók lehetnek navigáció szempontjából. Vannak olyan kerekesszékesek, akik egy-két lépést meg tudnak tenni, ha nagyon muszáj. Létezik sport (aktív) kerekesszék, amelyet sportokhoz és tánchoz használnak, ezeknek nem jelent akadályt egy pár centiméteres küszöbön „átugratni", vagy akár néhány lépcsőfokon legurulni (ha van korlát, egy atletikusabb személy akár fel is húzza magát a székkel együtt). És talán a legfontosabb, hogy a kerekesszékek szélessége személyre szabott lehet: létezik olyan, amely akár egy 70 cm széles ajtón is könnyedén áthalad.

1.2 A kapcsolt nyílt adatok használata útvonaltervezésre

Munkámban kapcsolt adatokat és az adatok aktuális tárolási formátumának megfelelő lekérdezőnyelveket használok (SPARQL nyelvet RDF formátumra, Cypher nyelvet gráf adatmodellre) a pozíció- és útvonalinformációkat szolgáltató rugalmas API felépítésére.

Egy épület belső struktúrája leírható objektumok és a közöttük fennálló kapcsolatok formájában. Például az épület egy folyosója kapcsolatban van az ajtóval, amely erről a folyosóról nyílik, az ajtó kapcsolatban van a szobával, amelyhez tartozik, és így tovább. Ezért egy, az épületen belüli útvonal szintén megadható az épület objektumainak és a köztük lévő kapcsolatoknak a felhasználásával. Megfelelően definiált struktúrájú adathalmazon egy útvonalkeresés is implementálható, amely az épület két objektuma között köztes objektumok és kapcsolatok listájaként adja vissza a kiindulási és célobjektumot összekötő útvonalat.

A Linked Open Data (LOD - kapcsolt nyílt adatok) [3, 4] jól ismert módszer strukturált, egymással kapcsolatban álló adathalmazok közzétételére a weben, számítógépek számára értelmezhető formában. Lehetővé teszi különféle forrásokból származó adatok összekapcsolását és lekérdezését; egy adathalmazban elhelyezhetünk olyan kapcsolatokat (linkeket), amelyek egy másik adathalmaz elemeire hivatkoznak.

Az RDF [5, 6, 7, 8] W3C szabvány, eredetileg webes hivatkozások és metaadatok tárolására szolgált. Az RDF esetében az adatok alany – állítmány – tárgy hármasokból álló kijelentésekbe rendeződnek. Az állítmány (kapcsolatnak is nevezik) leírja az alany és a tárgy között fennálló kapcsolatot. Az alany és az állítmány URI-k, a tárgy pedig

URI vagy string literál. Ez a hármas tekinthető gráfnak olyan módon, hogy az alany és a tárgy gráfcsomópontoknak feleltethető meg, az állítmány pedig a köztük lévő irányított élnek. Az RDF lekérdező nyelve a SPARQL, amely távoli rokonságban áll az SQL-lel, de sajnálatos módon útkeresés jellegű lekérdezésekre korlátozottan alkalmas, és meglehetősen lassú is.

Mivel a LOD különösen alkalmas térbeli navigáció leírására, ezért a szakirodalomban természetesen találhatók erre példák, az alábbiakban a célkitűzésem szempontjából releváns források áttekintése következik.

1.2.1 Létező ontológiák navigáció leírására

A kapcsolt adathalmazok szótárakra, sémákra és ontológiákra támaszkodnak. Az ontológia egy adott tématerülethez kapcsolódó tudás reprezentációja számítógép által feldolgozható formában, beleértve a tudáselemek között kapcsolatokat is. [5] Több szerző is készített már beltéri navigációra szolgáló ontológiákat. Worboys [9] áttekintette ezeket, és definiált egy felső szintű taxonómiát, amely szemantikus és térbeli kategóriákba sorolja be a belteret leíró modelleket. A szemantikus modellek típusokat, azok tulajdonságait és kapcsolatait reprezentálják. A topologikus modellek a helyek közötti térbeli kapcsolatokat ábrázolják. A geometrikus modellek megadják a távolság mértékét is, és végül a hibrid vagy többrétegű modellek az előbbiek kombinációját adják.

Az OntoNav [10] szemantikus beltéri navigációs rendszer és egy ontológiai keretrendszer útkeresési feladatok kezelésére. Képes épületen belüli navigációra folyosókon keresztül, de nem képes helyiségen belüli navigációra, amely szükséges lenne például egy több bejárattal rendelkező, nagy méretű terem vagy aula esetén. A felhasznált INO ontológia nem érhető el, ráadásul a legrövidebb útvonalak választásához és az akadályok elkerüléséhez szükséges implicit tudás nem az ontológiába, hanem az útkereső algoritmusba van beépítve.

Az ONALIN [11] az útkeresés során képes figyelembe venni a különböző szükségletekkel és preferenciákkal rendelkező egyének igényeit; egyéb követelmények mellett tartalmazza az ADA által előírtakat is. Az épületeket folyosók hálózataként

modellezi, és képes megadni a felhasználó által megadott specifikus megszorításoknak megfelelő útvonalakat. Ezt úgy éri el, hogy minden lehetséges fogyatékossághoz előre számított hálózatot használ, és mindig a megfelelő hálózaton futtatja a keresést.

Scholz és Schabus [12] olyan beltéri navigációs ontológiát készített, ami támogatja a termelőeszközök autonóm navigációját termelési környezetben. Az eszközök egy workflow által meghatározott lépéseknek megfelelő sorrendben haladva, minden lépéshez megkeresik a megfelelő berendezést, és kiszámítják az adott berendezéshez a legmegfelelőbb útvonalat. A publikációban az útvonaltervezés részletei nincsenek kifejtve, az azonban tudható, hogy az algoritmust kifejezetten a konkrét eszköztulajdonságok (például méret vagy speciális kezelési instrukciók) figyelembevételével készítették.

A Geodint [13] az általános legrövidebb út (shortest path) algoritmussal navigál egy gráf modellen. Jelenleg a fenti ontológiák egyike sem érhető el, azonban az ismertetett szemantikus modell egyes részei adtak ötletet a az osztályhierarchia felépítéséhez. A fent említett eredményekhez képest a mi megközelítésünk főként abban tér el, hogy szabványos (vagy de facto szabványos) lekérdező nyelvekre igyekszik támaszkodni az útvonalak meghatározásánál. Így maga az útkeresés algoritmusa általános maradhat, mivel a specifikumokat mind a modell (az ontológia) tartalmazza. Így elkerülhető az, hogy a különféle területekre és esetekre különböző algoritmusokat kelljen tervezni és implementálni.

Benner és Karimi [14] megvizsgálta az elérhető, gyalogos navigációra szolgáló ontológiákat (beleértve az INO és ONALIN ontológiákat is), és nem találtak olyan megközelítést, amely minden fogyatékosságra általánosan illene, a létező megoldások csak a spektrum egy részét fedik le. Javaslatuk, hogy a jövőben jobban kell koncentrálni az épített környezet akadálymentességének szemantikájára.

1.2.2 Az akadálymentesség megállapítása a navigációban

Az akadálymentes navigáció megvalósításához szükséges az is, hogy az épület objektumainak akadálymentességi jellemzőit belefoglaljuk az ontológiába. Fontos kérdés tehát, hogyan definiáljuk egy épület, illetve egy épületrész akadálymentes voltát.

Az első közelítés a törvényi szabályozásra (tehát Magyarországon az OTÉK-ra) támaszkodik.

Az akadálymentességnek ezt a törvényi megközelítését láthatjuk a kültéri akadálymentes navigációt nyújtó szolgáltatások épületekre vonatkozó részeiben is.

Számos, egyetlen városra vonatkozó navigációs alkalmazás érhető el kerekesszékes útvonaltervezéshez. Léteznek olyan térképek is, amelyek információval szolgálnak a középületek akadálymentességéről. Kevés olyan alkalmazás létezik azonban, amely átfogó megoldással szolgál a fogyatékkal élő személyek számára a városi közlekedéshez. Talán a legismertebb nemzetközi projekt a Wheelmap.org, amely az OpenStreetMap-on alapuló megoldás. A Wheelmap (ahogyan azt a neve is mutatja) a kerekesszékkel közlekedők igényeire koncentrál, ezért a következő színkódolást alkalmazza: zöld szín jelenti a kerekesszék számára teljesen akadálymentes épületeket; sárga jelöli a részben akadálymentes helyszíneket; piros jelzés pedig azokat a helyeket, amelyek egyáltalán nem akadálymentesek. A térkép az akadálymentes mosdók elhelyezkedését is jelöli, amelyeket az alábbi kritériumok alapján határoz meg: az ajtó belső szélessége legalább 90 cm, belül legyen legalább 150 cm x 150 cm üres terület, a WC ülőke kerekesszék-magasságú, a mosdóban fel van szerelve kihajtható kapaszkodó, és a kézmosó az ülő pozícióhoz megfelelő magasságú [15].

1.2.3 Használati esetek

A közszolgáltatásokat tipikusan több különböző részlegre (osztályokra) osztják szét, az egyes osztályok az épület különböző részein, különböző folyosókon találhatók. A látogatók célja lehet az épületen belül egy konkrét szoba, egy megadott kategóriába eső tetszőleges helyiség (például egy közeli mosdó), egy adott osztály kifejezett szoba nélkül (például egy kórházban a beteg szeretne eljutni a kardiológiára), vagy valamilyen szolgáltatás helye (például ATM, kávéautomata, kijelölt dohányzóhely).

A következő használati esetek demonstrálják a közintézményekben való navigáció követelményeit egy kórházban való tájékozódás példáin keresztül:

A legegyszerűbb eset, amikor egy személy szeretne eljutni az épület egy konkrét pontjáról egy másik pontra, például a bejárattól szeretne egy útvonalat a 312. szobához.

Az útvonalat leíró instrukcióknak egyszerűnek és könnyen követhetőnek kell lenniük: a lifttel menjen fel a harmadik emeletre, a bal felé eső folyosón menjen el az italautomatáig, annak közelében találja meg a 312. szobát.

Egy kissé bonyolultabb szituáció áll elő akkor, amikor a látogatónak nincs egyértelmű, teljes tudása a célról. Jó példa lehet erre, amikor egy páciens egy adott orvoshoz érkezik vizsgálatra. Tudja az orvos nevét és az osztályt, amit keresnie kell (legyen például Dr. Kiss a felnőtt kardiológia járóbeteg részlegén), de nem tud konkrét szobaszámot. Egy jól használható navigációs alkalmazás ilyenkor kétféleképpen viselkedhet: vagy navigálja el a pácienst a megfelelő részleg nővérpultjához, ahol a beteg további információkat kérhet, vagy ajánlja fel az adott osztály orvosi szobáihoz tartozó részletes információkat, így a felhasználó a listából kiválaszthatja Dr. Kiss szobáját.

Egy másik példa, amikor egy látogató mosdót keres. Ebben az esetben teljesen közömbös a konkrét szoba száma, ehelyett az alkalmazás a legközelebbi látogatóknak fenntartott mosdóhoz kell navigálja a felhasználót (nem pedig a kórtermek mosdóihoz, vagy a személyzet számára fenntartott helyiségekhez).

Az alkalmazást használó személy kezdeményezhet keresést bizonyos szolgáltatásokra vagy részlegekre is (például merre van a sürgősségi ellátás, vagy a vérvétel).

Az útkeresés során figyelembe kell venni a folyosók, ajtók és egyéb épületrészek különféle tulajdonságait ahhoz, hogy a különféle mértékű fogyatékkal élők (legyen szó akár a kerekesszékkel közlekedőkről, akár a nehezen járó idősekről) igényeit ki tudjuk elégíteni. A keresés indításakor a felhasználó a kezdő és végpontokon kívül megadja az akadálymentességi preferenciákat is (akár előre beállított és eltárolt konfiguráció formájában). Ezeknek a preferencia beállításoknak pontosan igazodniuk kell a felhasználó speciális szükségleteihez és képességeihez, beleértve akár a maximális távolságot, amit pihenési lehetőség nélkül gyalogolni tudnak, a lépcsők számát, vagy az emelkedők/lejtők szögét.

Például egy kerekesszékkel közlekedő személynek kerülnie kell a lépcsőt, magas küszöböt vagy meredek emelkedőket tartalmazó útszakaszokat, és egyik emeletről a másikra csak lifttel tudnak eljutni. Ezen kívül szükségük van megfelelően széles ajtókra

is, amelyen át tudnak jutni a székkel. De a különböző kerekesszék típusok (például egy elektromos kerekesszék) különböző meredekségű utakat tudnak bejárni, és egy hagyományos kerekesszék esetében sem mindegy, hogy felfelé vagy lefelé közlekednénk a lejtőn. Az egyes kerekesszék modellek szélessége is eltérhet, a keskeny modellek akár a 90 cm-nél keskenyebb ajtókon is átférnek. A kerekesszékkel közlekedő személy kisebb akadályokon (például egy küszöbön) átjuthat egy kis segítséggel, ha van a közelben valaki, aki segíteni tud. Sport kerekesszéket használók akár több lépcsőfokon is „leugratnak", ha van korlát, amiben meg lehet kapaszkodni.

Egy idős, bottal járó beteg szeretné minimalizálni a gyaloglás mennyiségét és a lépcsők számát, de nem jelent számukra leküzdhetetlen akadályt, ha meg lehet pihenni közben.

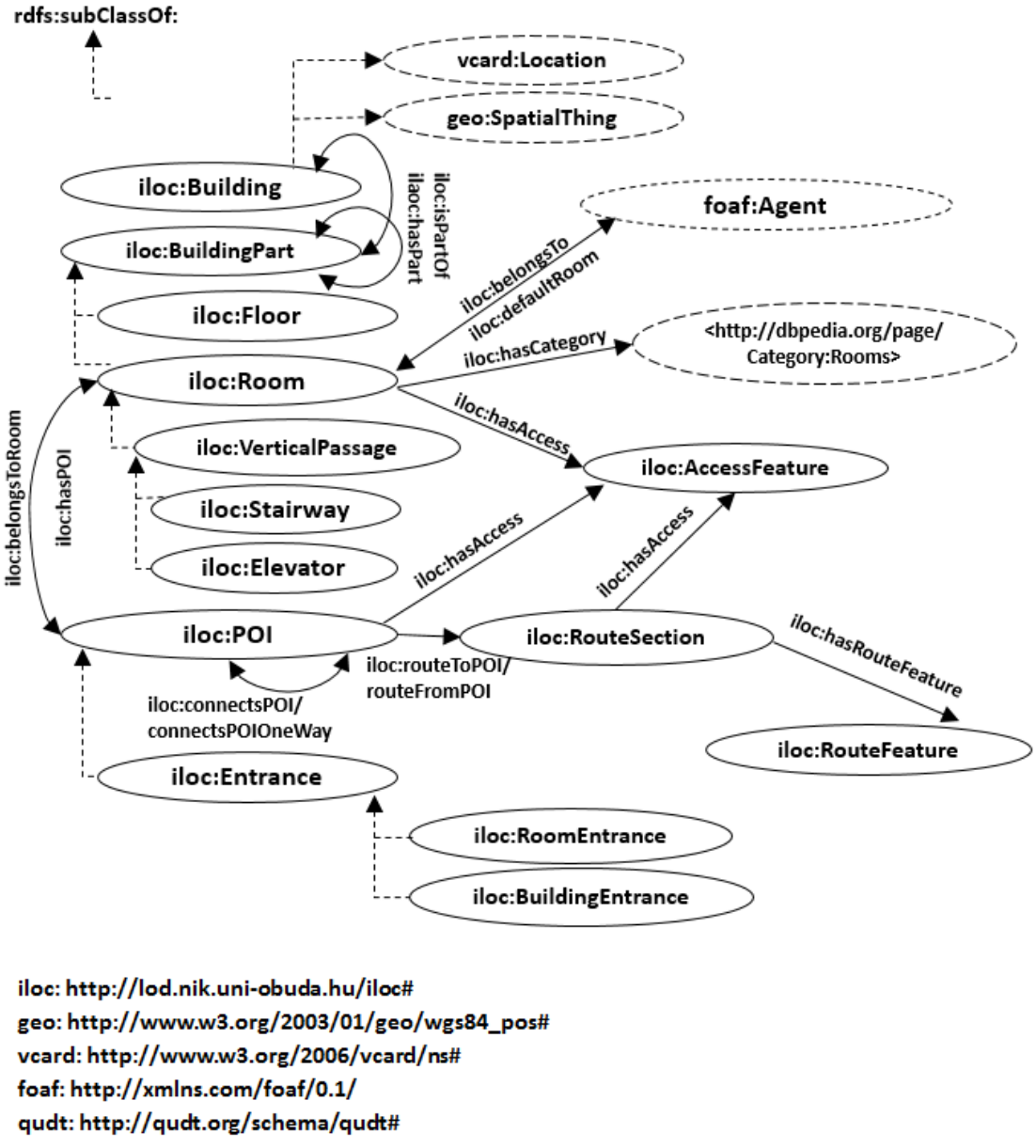
1.2.4 iLOC ontológia

Az ontológia segítségével megvalósított navigáció lehetőségeit Magyarországon is kutatják az MTA Számítástechnikai és Automatizálási Kutatóintézetében, a Debreceni Egyetemen, valamint az Óbudai Egyetem Neumann János Informatikai Karán dolgozó kollégák. A projektbe való bekapcsolódásom előtt kollégáim korábbi munkáikban [16, 17, 18] általános eljárásokat fejlesztettek ki a beltéri útvonalkeresésre. Ez azt jelenti, hogy a navigációs feladatot egy kiindulási és végponttal definiáljuk, amelyek egy épületben található általános objektumok, például POI-k, szobák vagy akár nagyobb egységek (részlegek).

Az elvárások komplexitása megkövetelte egy formális ontológia leírását, amely segít alkalmazni a kapcsolt adatokra vonatkozó elveket a publikált adatokra. Az ontológia fejlesztése során a célunk az irányelvekben [19] ismertetett 4 csillagos szótár megalkotása volt. A 4 csillagos szótár olyan szótár, amely adott ember által olvasható formában, számítógép által olvasható és feldolgozható formában (például RDFS, OWL), más szótárakra hivatkozik, és elérhetők metaadatok a szótárról (mint például a leíráshoz használt ontológia nyelv, az utolsó módosítás dátuma, licensz) számítógép által olvasható formában.

Ennek eredménye az 1.1 ábrán látható iLOC ontológia [20], amely egy általános épület belső szerkezetének leírását, az épületen belüli közlekedés lehetőségeit tartalmazta,

akadálymentességi információkat viszont kezdetben még nem. Azonban az iLOC ontológiát úgy tervezték, hogy egyszerűen bővíthető legyen további ontológiákkal, amelyek akár környezet-specifikus adatok (például kórházakra vonatkozóan [S15]), akár egyéb fontos tulajdonságok leírására szolgálnak. Az 1.1. ábra (és a további ontológia ábrák is) oválissal jelzi a főbb osztályokat, a folytonos nyilak pedig az osztályok közötti legfontosabb objektum tulajdonságokat.



1.1. ábra: Az iLOC ontológia

Az ebben az ontológiában definiált osztályok az iloc prefixet kapták, míg a más ontológiákból származó osztályok és tulajdonságok a saját prefixeikkel szerepelnek, és szaggatott vonallal határolt ellipszissel jelöljük. A szaggatott nyilak a leszármazási kapcsolatot (rdfs:subClassOf[2]) jelölik az osztályok között.

Az iLOC ontológiának három fő osztálya van: Building, BuildingPart és a POI. A Building osztály leszármazottja a vcard:Location és geo:SpatialThing osztályoknak, ezáltal a példányaihoz cím és földrajzi koordináták (hosszúság, szélesség) rendelhetők. A Building egyedek belső struktúráját hivatott leírni ez az ontológia. A BuildingPart absztrakt koncepció az épület különböző belső részeinek leírására. Két alosztálya a Floor és a Room. A Floor egyed az épület egy konkrét folyosóját reprezentálja. A Room megadhat egy konkrét szobát, illetve a VerticalPassage alosztályán keresztül speciálisan jelenthet olyan épületrészt, amely két emeletet (azaz két Floor példányt) köt össze. Ennek megfelelően a VerticalPassage rendelkezik két további alosztállyal, ezek az Elevator és a Stairway.

Az isPartOf objektum tulajdonság és a hozzá tartozó inverz tulajdonság, az isPart hierarchikus kapcsolatot fejeznek ki az épületen belül, például egy konkrét Room példány isPartOf kapcsolatban lehet egy Floor példánnyal, amennyiben az adott helyiségnek nyílik ajtaja az adott folyosóra. Fontos megjegyezni, hogy egy Room példány nem kötődik egyértelműen csak egyetlen folyosóhoz, hiszen előfordul olyan eset, amikor egy helyiségnek több különböző folyosóról is van bejárata. Egy másik példa erre az Elevator vagy Stairway egyedek, amelyeknek a kifejezett célja, hogy több folyosóhoz is tartozzanak. Ennek megfelelően egy Room példány több Floor példányhoz is kapcsolódhat az isPartOf tulajdonsággal.

A környezet-specifikus ontológiák definiálhatnak további Room alosztályokat, amelyek az adott használati esetre (például bevásárlóközpontok, egyetemek stb.) jellemzők; a hLOC egy ilyen környezet-specifikus ontológia kórházi épületekre, amelynek fejlesztésében már én is részt vettem [21]. A környezet-specifikus alosztályok nem játszanak különleges szerepet a navigációban, de szűrőként szolgálhatnak a célállomás

[2] rdfs: http://www.w3.org/TR/rdf-schema/

kiválasztásában. Ez különösen fontos akkor, amikor a felhasználó nem ismeri pontosan a célállomást, ilyenkor a támogatásként felajánlott lista kezelhető hosszúságúra szűrhető a szoba pontos típusa alapján.

A Room példányok tartozhatnak foaf:Agent egyedekhez, ezt a kapcsolatot a belongsTo tulajdonsággal adhatjuk meg. Ezen tulajdonság segítségével a Room példányokat hozzáköthetjük egy adott szervezethez vagy személyhez. Kórházi környezetben például így reprezentálhatjuk egy szobának valamely kórházi osztályhoz való tartozását, vagy hogy kinek a szobája az adott helyiség. A környezet-specifikus ontológiának tehát a megfelelő Room alosztályokon kívül definiálniuk kell a megfelelő Organization-alosztályokat is. Egy foaf:Organization egyedhez kapcsolhatunk egy „alapértelmezett helyiséget" a defaultRoomOf tulajdonsággal. Ez akkor lehet hasznos, ha a felhasználónak a keresett konkrét szobáról nem, azonban a keresett szervezeti egységről (kórházi osztály) van információja. Ilyen alapértelmezett helyiség lehet például az adott osztály nővérpultja. Természetesen a Room osztály példányait további, külső kategóriákba is besorolhatjuk a hasCategory tulajdonsággal, amely a Dbpedia[3]-ban meghatározott helyiség kategóriákra mutathat.

A POI (Point of Interest – hasznos, érdekes pont) osztály fontos szerepet játszik az iLOC által támogatott navigációs folyamatban. Egy útvonal egymással kapcsolatban álló POI-k szekvenciájából áll, amelyek a connectsPOI tulajdonsággal kapcsolódnak egymáshoz. Az iLOC-ban a POI osztálynak egy alosztálya van (Entrance), amely további két alosztállyal rendelkezik, ezek a RoomEntrance és a BuildingEntrance. Az Entrance példányokkal adjuk meg az épület belépési pontjait, illetve a Room egyedek közötti átjárási lehetőséget. Az ontológiában megszorításként szerepel, hogy minden épületnek és helyiségnek kell legyen legalább egy bejárata, és a RoomEntrance egyedek pontosan két Room egyedhez kell kapcsolódjanak. A fentiekhez hasonlóan a POI osztályhoz is megadhatók további alosztályok a környezet-specifikus ontológiákban. POI példány lehet például egy szobor, automata vagy információs tábla.

[3] http://wiki.dbpedia.org

A connectsPOI tulajdonság közvetlen kapcsolatot ír le két POI példány között, az útvonal két kapcsolódó POI között magától értetődően adottnak tekintendő az útvonaltervezés során. A connectsPOIOneWay tulajdonság a connectsPOI aszimmetrikus szülőtulajdonsága, amely egyirányú kapcsolatot ad meg két pont között (ámbár az ilyen eset viszonylag ritka, például az egy irányba működő forgókapuk a jogosultsághoz kötött beléptetésnél). A navigáció során előálló útvonal névvel ellátott, összekapcsolt POI-k sorozata. Ez a megközelítés olyan útvonal-leírásokat eredményez, mint például: „Lépjen be az épület bejáratán. Menjen el az információs tábla mellett. Menjen a lépcsőhöz. Menjen fel a 4. emeletre. Menjen el a bankautomata mellett. Keresse a 407. szobát.” A hasPOI tulajdonság (és inverze, a belongsToRoom) egy POI és egy Room egyed közötti kapcsolatot fejeznek ki, azaz azt, hogy egy konkrét helyiséghez tartozik egy adott POI.

A RouteSection osztály két szomszédos, összekapcsolt POI közötti bejárható útvonalat reprezentálja. A RouteSection egyed definiálásakor meg kell adni a végpontjait, azaz 2 db POI objektumot.

A kapcsolatok kardinalitását azok értelmezése diktálja, ezért a diagramon nem tartottam szükségesnek feltüntetni. Az ontológia leírásban a kardinalitás az owl:cardinality, owl:minCardinality és owl:maxCardinality megszorításokkal adható meg.

1.3 Az ontológia akadálymentességi kiegészítéseinek fejlesztése

1.3.1 A fejlesztés módszerei

Az ontológia fejlesztése során a diagram ábrázolás jelöléseit a Protégé[4] ontológia szerkesztő szoftver által alkalmazott jelölések alapján választottam meg, tehát ellipszisek jelölik az osztályokat, és nyilak a közöttük lévő kapcsolatokat. A kapcsolatok lehetnek a saját ontológia által meghatározott kapcsolatok, vagy publikus kapcsolat típusok, például az rdfs:subClassOf, amely az osztályhierarchia ábrázolására

[4] https://protege.stanford.edu/

hivatott. Az ontológia leírása során az RDF[5] (Resource Description Framework), RDFS[6] (RDF Schema) és OWL[7] (W3C Web Ontology Language) által meghatározott szintaktikát követtem.

Az iLOC számos publikus ontológiához kapcsolódik, és az akadálymentességi ontológia kiegészítésem is alkalmaz ilyet. Ezek közül a leglényegesebbek:

- a FOAF[8] (friend of a friend), amelyből az Agent és az Organization osztályokra hivatkozik az iLOC,
- a World Wide Web Consortium által létrehozott Geo szótár[9] (SpatialThing osztály) és vCard[10] ontológia (Location osztály)
- a QUDT[11] (Quantities, Units, Dimensions and Types) séma, amelyből a qudt:value-t és a qudt:unit-ot használtam fel az objektumok tulajdonságainak leírására.

Az ontológiákra való hivatkozás úgy történik, hogy az ontológia névtér azonosítójához (IRI – Internationalized Resource Identifier) egy prefixet rendelünk, és a továbbiakban ezzel a prefixszel hivatkozunk rá. Például egy ontológiát leíró dokumentum elején szereplő

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
```

kódsor azt határozza meg, hogy a dokumentum további részében az rdfs: rövidítés az RDF Schema 1.1 verzióját jelenti. Egy ontológiának megfelelően leírt adathalmazon SPARQL nyelvű lekérdezések futtathatók.

A tesztekhez Linked Data Platform (más néven triplestore, vagy subject-predicate-object database) szoftvereket (Apache Marmotta[12] és OpenLink Virtuoso[13]) használtam, amelyek képesek SPARQL lekérdezéseket futtatni.

---

[5] http://www.w3.org/1999/02/22-rdf-syntax-ns#
[6] http://www.w3.org/2000/01/rdf-schema#
[7] http://www.w3.org/2002/07/owl#
[8] http://xmlns.com/foaf/0.1/
[9] http://www.w3.org/2003/01/geo/wgs84_pos#
[10] http://www.w3.org/2006/vcard/ns
[11] http://qudt.org/schema/qudt#
[12] https://marmotta.apache.org/
[13] https://virtuoso.openlinksw.com

1.3.2 A folyosók mennyiségi, minőségi és funkcionális jellemzőinek leírása

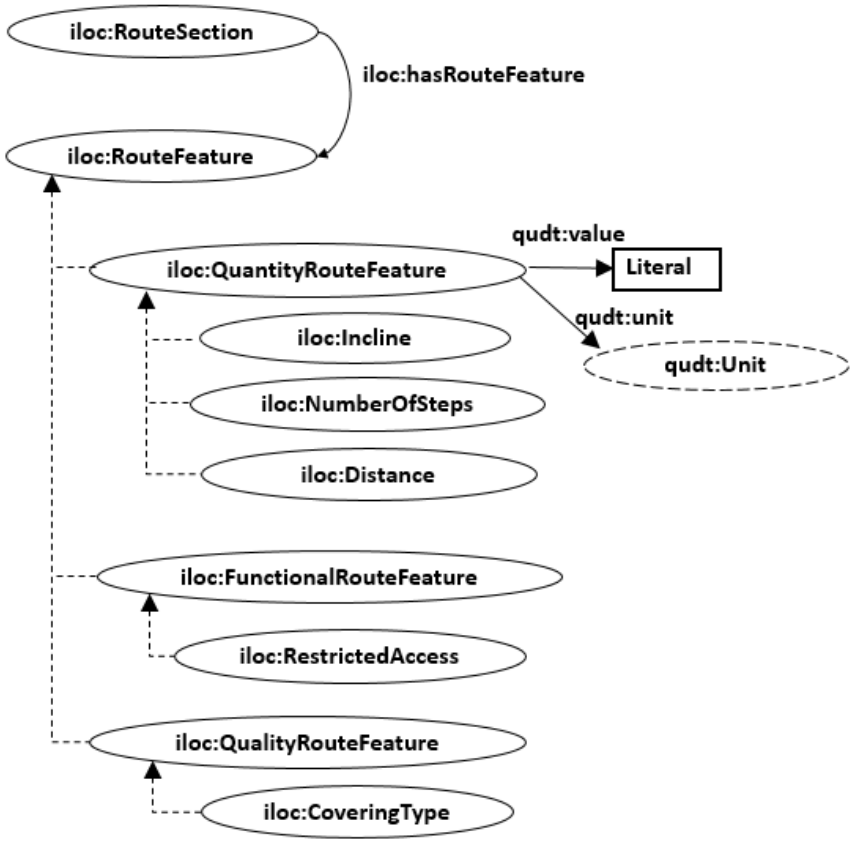
Az akadálymentes tervezéshez figyelembe kell venni az épület egyes részeinek akadálymentességi tulajdonságait, így lehetővé válik a mozgássérült látogatók és betegek számára való útvonaltervezés támogatása. Ezt úgy érjük el, hogy leírjuk a folyosók és szobák azon tulajdonságait, amelyek akadálymentességi szempontból relevánsak lehetnek, például a távolságot, lépcsők számát, akadályoknak a meglétét vagy hiányát, kisegítő felszereléseket (kapaszkodó, korlát, kerekesszék lift stb.).

A RouteSection példányokhoz ennek megfelelően tartozhatnak különféle attribútumok, ezek lehetnek minőségi (például a burkolat típusa), mennyiségi (például hossz, szélesség, meredekség, lépcsők száma), vagy funkcionális leírók és megszorítások (például csak belépőkártyával lehet áthaladni rajta). Ebből következően bizonyos akadálymentességi tulajdonságok (például kerekesszék számára akadálymentes) kikövetkeztethetők az útvonal tulajdonságaiból: mondhatjuk, hogy ha egy útszakasz szélessége meghaladja a 90 cm-t, a meredeksége 0 és a lépcsők száma is 0, akkor kerekesszékkel bejárhatónak tekinthető. Emellett természetesen manuálisan is megadhatók ezek az akadálymentességi tulajdonságok emberi döntés alapján.

Az iLOC jelenlegi állapotában akadálymentességi információkat a RouteSection egyedekhez a RouteFeature és AccessFeature osztályok segítségével adhatunk.

Az AccessFeature osztály már az iLOC korai verzióiban (a projektbe való bekapcsolódásom előtt) is szerepelt, ez reprezentálja a különböző fogyatékosságokat, amelyek speciális követelményeket jelentenek mind a RouteSection példányok bejárásában, mind a Room vagy POI példányok használatában. Mivel a különféle mozgási fogyatékosságok támogatása volt az elsődleges cél, ezért az AccessFeature osztályban a következő egyedeket definiálták: Wheelchair, EWheelchair (mint elektronikus kerekesszék), WheelchairWHelp (kerekesszék segítővel), Stretcher, Stroller. Ezeknek az egyedeknek a használata a törvényi definíció alapján ajánlott, azaz „kerekesszék számára akadálymentes” az a RouteSection, amelynek a paraméterei megfelelnek az építési törvényben leírtnak. A jövőben további egyedek adhatók a

listához az akadálymentességi kategóriák bővítése érdekében. A hasAccess tulajdonság használható az AccessFeature egyedeknek a különböző RouteSection, Room vagy POI példányokhoz kapcsolására.



1.2. ábra: Az iLOC:RouteFeature osztály és alosztályai

Az iLOC fejlesztésébe való bekapcsolódásom után rámutattam az AccessFeature osztály elégtelen leíró erejére, ami a gyakorlati használatban komoly problémákhoz vezethet. A folyosók járhatóságának pontos leírásához sokkal több adatot láttam szükségesnek, ezért született meg és került bele az iLOC későbbi verzióiba [22] a RouteFeature osztály és alosztályai. A RouteFeature szerepe, hogy az előzőekben tárgyalt, a navigáció testreszabásához használatos minőségi, mennyiségi és funkcionális attribútumokat leírja. Az 1.2. ábrán látható, hogy a RouteFeature osztálynak három közvetlen leszármazott osztálya van: QuantityRouteFeature, QualityRouteFeature és FunctionalRouteFeature, amelyeknek további leszármazottai vannak.

A QuantityRouteFeature alosztályok (pl. Distance, Incline, NumberOfSteps) célja a mennyiségi jellemzők megadása, ennek megfelelően egyedeinek egy számérték és egy mértékegység a tulajdonságai. A QUDT[14] ontológia felhasználható az utóbbi értékeinek megadására. Használhatók például az útszakaszok hosszának, lejtésének, a rajta előforduló lépcsőfokok számának megadása. A QualityRouteFeature alosztályok (pl. CoveringType) példányai leírják az adott útszakasz valamely specifikus minőségi tulajdonságát, például a csúszásgátló burkolat meglétét. A FunctionalRouteFeature alosztályok (pl. RestrictedAccess) az útszakasz funkcionalitásáról szolgáltatnak további információt, ilyen lehet például az a gyakran előforduló jelenség, hogy a folyosó egy bizonyos szakasza zárt, és csak kulcs vagy belépőkártya birtokában lehet bejárni. A másik használati lehetősége az épületben bekövetkezett változások (elromlott lift, lezárt folyosó) ábrázolása.

A hasRouteFeature tulajdonság kapcsolja össze a RouteSection egyedeket a hozzájuk tartozó RouteFeature egyedekkel. A fenti osztályhierarchia felépítésével a cél egy egyszerűen bővíthető keretrendszer megalkotása, nem feltétlenül az, hogy egy teljes és végleges megoldást adjon az épületek útvonalainak leírására, hiszen minden épületben lehetnek egyedi jellegzetességek.

A további munka során azonban a fenti megoldás még mindig nem bizonyult kellően rugalmasnak, ezért fejlesztettem ki az általam iACC (ACC: accessibility) névre keresztelt ontológia kiterjesztést.

1.3.3 Az iACC ontológia kiterjesztés

Belátható, hogy a törvényi kritériumok szerint részben akadálymentes vagy egyáltalán nem akadálymentes épületek mégis használhatók, bejárhatók lehetnek az egyéni képességek alapján, vagy abban az esetben, ha a mozgássérült személy biztosítani tud egy minimális segítséget akár kísérő, akár a helyi személyzet formájában.

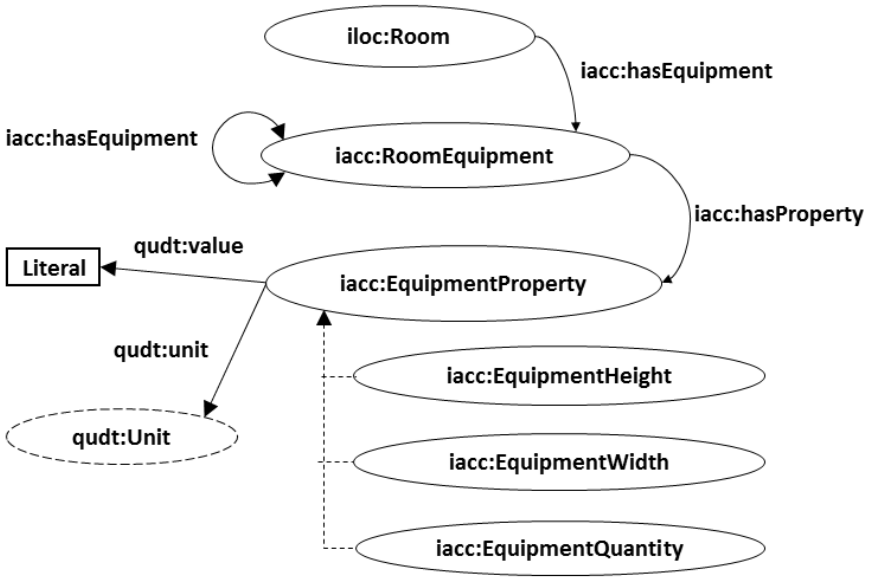
[14] *http://qudt.org/schema/qudt#*

Az eredeti iLOC-ban az objektumok akadálymentességi tulajdonságai az OTÉK által meghatározott definíciókon és méreteken alapultak, mivel Magyarországon ez a hivatalos követelmény. A később hozzáadott RouteFeature segítségével az útszakaszok, folyosók részletes leírása megoldódott, a POI-k és szobák akadálymentességének leírása azonban továbbra is csak egyszerű címkézésre korlátozódott. A részben akadálymentes középületek problémája miatt ez a gyakorlatban nem állta meg a helyét.

A felhasználó egyedi igényeinek való megfelelés érdekében megterveztem egy ontológia kiterjesztést úgy, hogy egyszerű igen/nem típusú címkék helyett az objektumoknak pontos méreteit tároljuk, méghozzá mindazon tulajdonságaikat, amelyek akadálymentességi szempontból relevánsak lehetnek. Így például nem csak a folyosók, hanem az ajtók szélességét is; továbbá a küszöbök és lépcsőfokok magasságát, villanykapcsolók elhelyezését stb.

Az előbbi példákon túl az akadálymentes útvonal tervezése során figyelembe vehető egyedi felhasználói igények listáját az határozza meg, hogy milyen adatokat rögzítünk az épület objektumairól. Az ontológia kiterjesztés tervezésekor éppen az volt a cél, hogy ábrázolható legyen egy helyiség összes tárgyának a szélessége, magassága, mennyisége, illetve egyáltalán a megléte vagy hiánya. Az útvonal keresése során mindezen adatokra lehetséges feltételt adni. A gyakorlatban persze praktikus azokra a tulajdonságokra szűrni, amelyek akadálymentességi problémát jelenthetnek. Például tárolható a mennyezet magassága, és ezáltal feltétel is adható rá a keresésnél, azonban ez nem releváns információ akadálymentességi szempontból. A lentebbi példákban ismertetek számos olyan tulajdonságot, amelyeket viszont érdemes szűrhetővé tenni, mivel gyakran jelenthetnek akadályt.

Az 1.3. ábrán az általam kidolgozott iACC kiterjesztés látható, valamint a kapcsolódása az iLOC ontológiához: az iloc:Room osztály kapcsolódik az iacc:RoomEquipment osztályhoz az iacc:hasEquipment tulajdonsággal. Az iacc:hasEquipment tulajdonság mutatja meg, hogy egy adott helyiséghez mely felszerelések tartoznak.



1.3. ábra: Az iACC ontológia kiterjesztés

Az iacc:RoomEquipment osztály írja le az akadálymentességi szempontból releváns felszereléseket, objektumokat. Ez lehet a helyiségnek valamely szerves része (például ablak, ajtó, küszöb), vagy utólag felszerelt kiegészítő (például lehajtható kapaszkodó a mozgássérült mosdóban). A RoomEquipment példányok kapcsolódhatnak más RoomEquipment példányokhoz is a hasEquipment tulajdonsággal. Ez tulajdonképpen tartalmazási kapcsolatot jelent a különböző objektumok között, például egy mosdóban (ami egy Room típusú objektum) lehet több WC fülke (amelyek RoomEquipment példányok), és a fülkék tartalmaznak WC-ket (amelyek szintén RoomEquipment példányok).

A RoomEquipment objektumokhoz több iacc:EquipmentProperty példány is kapcsolódhat az iacc:hasProperty tulajdonsággal. Az EquipmentProperty osztálynak három alosztályát definiáltam: ezek az EquipmentHeight, EquipmentWidth és EquipmentQuantity. Az EquipmentHeight az objektumok magasságát írja le, például ablak kilincsek, villanykapcsolók, tükrök és küszöbök esetében használatos. Az EquipmentWidth az ajtónyílások, WC fülkék, lépcsők stb. szélességét adja meg. Az EquipmentQuantity kettős szerepet játszik: tárolhat darabszámot, mint például a

lépcsőfokok számát két emelet között, de bináris adatot is (ilyenkor értéke 0 vagy 1 lehet), például, hogy van-e vészhívó gomb egy helyiségben. Az EquipmentProperty példányoknak értéke és mértékegysége is van, tehát egy ajtóhoz tartozó EquipmentWidth értéke lehet 80, a mértékegysége pedig centiméter.

A RoomEquipment-ek leírása azáltal is teljesebbé tehető, ha egy RoomEquipment objektumhoz több EquipmentProperty-t is megadunk, például a helyiségben található mosdókagylóknak megadhatjuk a magasságát (EquipmentHeight) és a darabszámát (EquipmentQuantity) is.

Ennek a megoldásnak egy gyengesége, hogy bizonyos speciális, bonyolult navigációs eseteket szöveges címkékkel tud kezelni, tipikusan a több különböző megközelíthetőségű ajtóval rendelkező termek problémáját. Legyen például egy nagyelőadónak 4 ajtaja, kettő a földszintről lépcsőn közelíthető meg, kettő pedig az 1. emeleti folyosóról akadálymentesen. Az iLOC a több ajtós felállást tökéletesen kezeli POI-k használatával (mindegyik ajtóhoz rendelünk egy POI-t, amelyeket elhelyezkedésüknek megfelelően kötünk a folyosókhoz), azonban a RoomEquipment, amely a releváns akadálymentességi információt tárolná, jelen formában csak szöveges hivatkozással tud utalni a hozzá tartozó POI-ra. Amennyiben egy épületben fennáll ilyen eset, akkor az iACC kiegészíthető úgy, hogy az iacc:hasEquipment kapcsolat POI és RoomEquipment közé is felvehető legyen.

1.3.4 Használati esetek

Ebben a részben a kifejlesztett ontológia kiterjesztés felhasználási lehetőségeit demonstrálom példaadatokkal és SPARQL lekérdezésekkel.

Vegyünk egy mosdót, amely a harmadik emeleten található. Három objektum leírást tárolunk hozzá: egy ajtó, egy fülke és egy tükör objektum adatait. Az ajtó legyen 80 centiméter széles, a tükör legyen 110 centiméter magasságban rögzítve a falra. A fülke legyen 300 centiméter széles, egy 48 centiméter magas WC-vel. Ennek az RDF leírása a következőképpen néz ki:

```
@prefix iloc: <http://lod.nik.uni-obuda.hu/iloc/iloc#> .
@prefix iacc: <http://lod.nik.uni-obuda.hu/iacc/iacc#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

:Restroom302 a iloc:Room;
rdfs:label "Restroom"@en;
iacc:hasEquipment :Restroom302Door,
   :Restroom302Stall,
   :Restroom302Mirror.

:Restroom302Door a iacc:RoomEquipment;
   rdfs:label "Door"@en;
   iacc:hasProperty [
        a iacc:EquipmentWidth;
        qudt:value 80;
        qudt:unit qudt:centimeter.]

:Restroom302Stall a iacc:RoomEquipment;
   rdfs:label "Stall"@en;
   iacc:hasEquipment :Restroom302StallToilet;
   iacc:hasProperty [
        a iacc:EquipmentWidth;
        qudt:value 300;
        qudt:unit qudt:centimeter.]

:Restroom302StallToilet a iacc:RoomEquipment;
   rdfs:label "Toilet"@en;
   iacc:hasProperty [
        a iacc:EquipmentHeight;
        qudt:value 48;
        qudt:unit qudt:centimeter.]

:Restroom302Mirror a iacc:RoomEquipment;
   rdfs:label "Mirror"@en;
   iacc:hasProperty [
        a iacc:EquipmentHeight;
        qudt:value 110;
        qudt:unit qudt:centimeter.]
```

A következő SPARQL lekérdezés arra szolgál, hogy találjunk egy mosdót legalább 70 centiméter széles ajtóval, és egy olyan tükörrel, amely legfeljebb 120 centiméter magasságban található a falon. A 70 centiméter szélességű ajtó nem számít akadálymentesnek sem a magyar, sem a külföldi szabályozás szerint, de néhány keskenyebb kerekesszék átfér rajta.

```
prefix iloc: <http://lod.nik.uni-obuda.hu/iloc/iloc#>
prefix iacc: <http://lod.nik.uni-obuda.hu/iacc/iacc#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?fl ?rnum
WHERE {
   ?rr a iloc:Room;
        rdfs:label "Restroom".
   OPTIONAL {?floor rdfs:label ?fl.}
   OPTIONAL {?rr rdfs:label ?rnum.}
   ?rr iloc:isPartOf ?floor.
   ?rr iacc:hasEquipment ?door.
   ?rr iacc:hasEquipment ?mirror.
   ?door rdfs:label "Door";
        iacc:hasProperty ?w.
   ?w a iacc:EquipmentWidth;
        qudt:value ?ewidth.
   ?mirror rdfs:label "Mirror";
        iacc:hasProperty ?h.
   ?h a iacc:EquipmentHeight;
        qudt:value ?eheight.
   FILTER ( ?ewidth > 69 && ?eheight < 121 )
}
```

A következő lekérdezés a főbejárattól a legközelebbi olyan mosdót keresi, amelyben van legalább 250 centiméter széles fülke. (A SPARQL 1.1 tulajdonságai miatt definiálni kell egy adott számú köztes POI-t – ez ebben a példában 3 – amely így az út maximális hossza lesz.)

```
prefix iloc: <http://lod.nik.uni-obuda.hu/iloc/iloc#>
prefix iacc: <http://lod.nik.uni-obuda.hu/iacc/iacc#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix ex: <http://example.org/>

SELECT ?poil1 ?poil2 ?poil3 ?endl
WHERE {
  BIND (ex:MainEntrance AS ?start ).
  ?end a iloc:Room;
        rdfs:label "Restroom".
  OPTIONAL {?p1 rdfs:label ?poil1.}
  OPTIONAL {?p2 rdfs:label ?poil2.}
  OPTIONAL {?p3 rdfs:label ?poil3.}
  OPTIONAL {?end rdfs:label ?endl.}
  ?start iloc:connectsPOI ?p1.
  ?p1 iloc:connectsPOI ?p2.
  ?p2 iloc:connectsPOI ?p3.
  ?plast iloc:belongsToRoom ?end.
   ?end iacc:hasEquipment ?stall.
   ?stall rdfs:label "Stall";
        iacc:hasProperty ?w.
   ?w a iacc:EquipmentWidth;
        qudt:value ?ewidth.
  FILTER ((?p3 = ?plast || ?p2 = ?plast || ?p1 = ?plast) &&
ewidth > 249 )
BIND ((if( ?p3 = ?plast , 3, if( ?p2 = ?plast , 2, if( ?p1 =
?plast , 1, -1)))) AS ?distance)
} ORDER BY ?distance LIMIT 1
```

1.3.5 Összegzés

Az itt ismertetett ontológia kiegészítés képessé teszi az iLoc ontológiát az épületen belüli objektumok akadálymentesség szempontjából releváns tulajdonságainak részletes leírására, ezáltal pedig megalapozza egy rugalmasan paraméterezhető, maximálisan a felhasználó egyéni igényeire és képességeire szabható beltéri útvonaltervező alkalmazás elkészítését. A tesztelt SPARQL lekérdezések, bár helyes eredményt adtak, nagy mennyiségű adatokon azonban meglehetősen lassan futnak le, érthető módon, hiszen a SPARQL-t nem ilyen jellegű feladatokra tervezték és optimalizálták. Az Apache

Marmotta és az OpenLink Virtuoso szerverek rendkívül különbözően teljesítettek ugyanazon az adathalmazon ugyanazokat a lekérdezéseket futtatva. A Virtuoso azért volt jelentősen gyorsabb a kettő közül, mert nem szabványos megoldásokat is használ.

Maga az adattárolás módja sem célszerű gráfbejáró algoritmust tartalmazó lekérdezések futtatására, mivel nem tárol olyan adatot az alany – állítmány – tárgy hármasokból álló kijelentésekben, amelyek a kapcsolatok gyors bejárását szolgálnák.

1.4 Áttérés gráf adatmodellre

Az előző példákban az adatok leírására RDF formátumot használtam. Van azonban más mód is a gráf szerkezetű adatok leírására. Az RDF-hez hasonlóan a címkézett tulajdonsággráf [23] (labeled property graph – LPG) is elterjedt módszer erre a feladatra. Ez az adatmodell olyan irányított gráf, amelyben mind a csomópontoknak, mind az éleknek lehetnek címkéik és kulcs-érték párok formájában megadott tulajdonságaik is. Az éleket kapcsolatoknak is nevezzük.

A legelterjedtebb, címkézett tulajdonsággráf adatmodellt használó NoSQL adatbázis-kezelő a Neo4j. Java alapú, de természetesen nem csak Java alkalmazásokból érhető el [24]. Natív gráfadatbázis, azaz adattárolási módja a gyors gráfbejárásra optimalizált olyan módon, hogy közvetlen, memóriacímre hivatkozó mutatókkal valósítja meg a kapcsolatok tárolását. Így sokkal hatékonyabb a nem-natív gráfadatbázisoknál, amelyek csak absztrakciós rétegek másfajta adatmodellel dolgozó adatbázisokra építve, és amelyekben további adatszerkezetek betöltése és vizsgálata szükséges a kapcsolódó csomópontok eléréséhez. Ezért tehát az RDF-fel szemben (amely inkább statikus vagy ritkán változó adatok tárolására alkalmas, dinamikus rendszerekben és gyors lekérdezés-futtatásra kevésbé) sokkal megfelelőbbnek tűnik az akadálymentes beltéri útvonaltervezés megvalósítására.

Deklaratív lekérdező nyelve a Cypher [25], amely jelentősen eltér ugyan az SQL nyelvtől, de a gráfokban előforduló mintázatokat rendkívül szemléletesen képes ábrázolni, éppen ezért a bonyolultabb Cypher lekérdezések is áttekinthetők és könnyen megérthetők.

1.4.1 A fejlesztés módszerei

A címkézett tulajdonsággráf adatszerkezetbe való áttérés során több lehetséges transzformációt is alkottam, azaz olyan megfeleltetéseket a két gráftípus között, amely az RDF-ben ábrázolt elemekhez vagy elemcsoportokhoz LPG elemeket vagy elemcsoportokat rendel úgy, hogy az ábrázolt információ-tartalom a lehető legteljesebb mértékben megjelenjen az eredményül kapott gráfban is. Az így keletkezett adatmodelleknek megfelelően tesztadatokat vittem be az adatbázisba, majd különböző lekérdezésekkel teszteltem azokat.

A gráfadatbázison végzett tesztekhez a Neo4j[15] szoftver 3.4 Community verzióját használtam. A Neo4j hasznos tulajdonsága, hogy számos fájlformátumból képes adatokat importálni (elsősorban a JSON és CSV támogatott, de XML vagy XLS beolvasására is van lehetőség), így az OWL-ban meglévő adatok a triplestore-ból exportálva betölthetők címkézett tulajdonsággráf adatszerkezetbe. A Neo4j képes futtatni a Cypher mellett Gremlin nyelvű lekérdezéseket is, míg az előbbi deklaratív, az utóbbi elsősorban procedurális szemléletű (bár deklaratív lekérdezések alkotására is van lehetőség). A tesztjeimhez Cyphert használtam, mivel egyrészt ez a Neo4j hivatalosan támogatott lekérdezőnyelve, másrészt pedig felfogásában jobban hasonlít a SPARQL-hez, ezért könnyebben összehasonlítható azzal. Emellett elérhető számos ismertebb gráfbejáró algoritmus optimalizált implementációja is, amelyeket egyszerűen kombinálhatunk a Cypher mintaillesztő képességével.

A tesztlekérdezések futtatása előtt gondoskodtam arról, hogy a Neo4j gyorsítótár funkciója ki legyen kapcsolva, hiszen ez a korábban már futtatott lekérdezések végrehajtását jelentősen felgyorsítja, és ilyen módon hamis időeredményeket okozna a tesztelés során.

[15] https://neo4j.com/

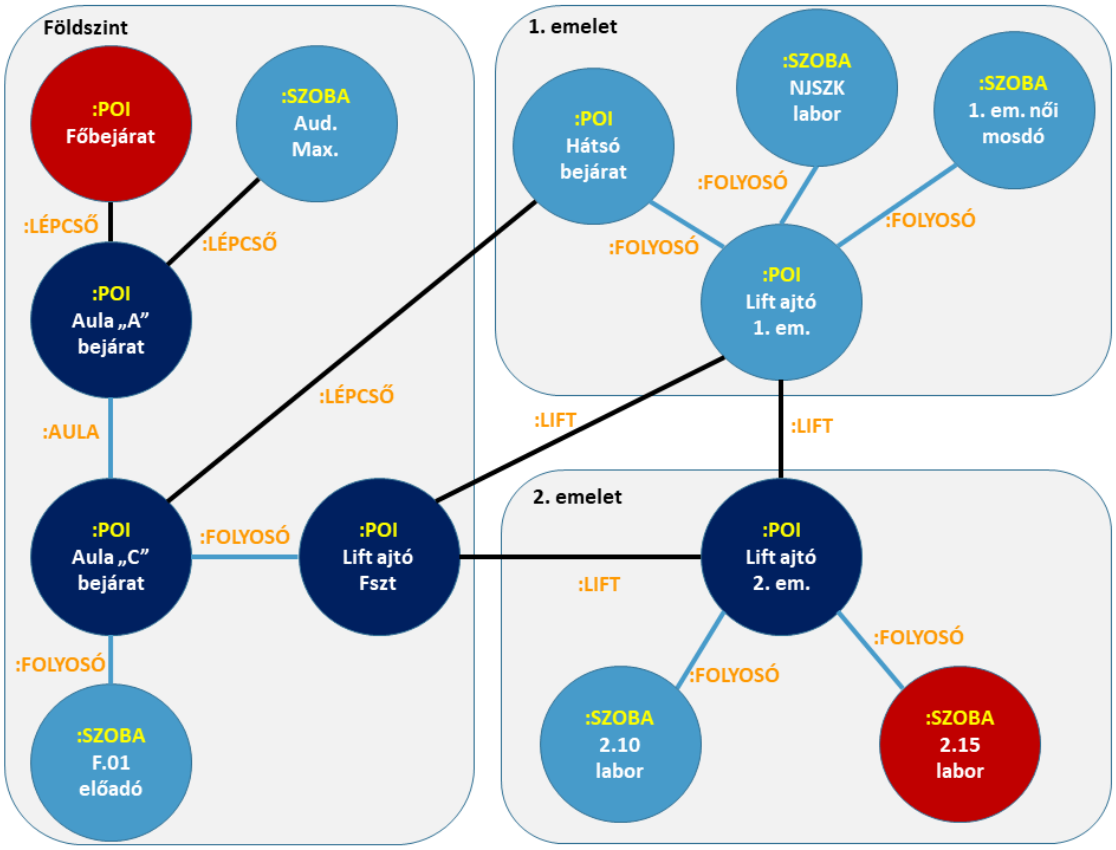
1.4.2 Útvonaltervezési feladat ábrázolása tulajdonsággráf adatmodellben

A címkézett tulajdonsággráf (LPG) olyan irányított gráf, amelyben a csúcsok és az élek is rendelkezhetnek címkékkel és tulajdonságokkal. A címkék egyszerű szövegek, a tulajdonságok kulcs-érték párok formájában adhatók meg, ahol a kulcs a tulajdonság neve. Gráfadatbázis terminológiában a csúcsokat csomópontoknak (node), az éleket kapcsolatoknak (relationship) is nevezzük.

Útvonaltervezési feladatok LPG szerkezetben való ábrázolásának alapelve az, hogy az ábrázolt terület jelentőséggel bíró pontjait (beltérben például a szobák, folyosók elágazásai, POI-k) csomópontokként jelenítjük meg, a típusukat címke formájában rendeljük hozzájuk, releváns tulajdonságaik kulcs-érték párok formájában tárolhatók. Két csomópont között akkor van kapcsolat, ha köztük egyéb csomópontok érintése nélkül bejárható út létezik (beltérben például folyosó, lépcsőház, lift). A kapcsolatok típusa és tulajdonságai (például folyosószakasz hossza, lépcsőfokok száma) szintén tárolhatók a címkézett tulajdonsággráfban.

Az 1.4 ábrán a Neumann János Informatikai Kar épületének egy része látható, a szemléletesség kedvéért kissé egyszerűsítve. Körrel jelöltem a csomópontokat, ezen belül a piros kitöltés a kiindulási és célpontokat jelöli, a sötétebb kék kitöltés pedig a kettő közötti legrövidebb út által érintett csomópontokat. A körön belül sárga színnel van feltüntetve a csomópont címkéje (Neo4j szintaktika szerint kettősponttal kezdve), fehérrel pedig a csomópont "Név" tulajdonságának értéke. A csomópontok közötti kapcsolatok mindkét irányba járhatók, címkéjük narancssárgával van feltüntetve mellettük.

Tegyük fel, hogy egy hallgató a főbejárattól a 2.15 laborba szeretne eljutni. Ekkor a gráfon a legrövidebb utat meghatározva az ábrán látható eredményt kapjuk, és a tárolt adatokból gyakorlatilag kiolvashatók a szóbeli navigációs instrukciók: a hallgató a főbejárattól a lépcsőn felmegy az aulába, azon keresztül az aula túlsó végében található lifthez megy. A lifttel felmegy a 2. emeletre, majd a folyosón át eljut a 2.15 laborig.



1.4. ábra: Példa útvonaltervezési feladat ábrázolására LPG-ben

Ha részletesebb ábrázolásra van igény, akkor az LPG rugalmassága lehetővé teszi akár azt is, hogy maguk a folyosószakaszok, lépcsőházak, stb. is csomópontokként jelenjenek meg a gráfban. Ekkor két csomópont között akkor van kapcsolat, ha azok közvetlenül érintkeznek, például egy szoba ajtaja az adott folyosóra nyílik.

1.4.3 Ontológiák transzformálása tulajdonsággráf adatmodellbe

A cél az iLOC és iACC ontológiák transzformálása címkézett tulajdonsággráf modellbe [26] olyan módon, hogy az összes adat és összefüggés átvihető legyen a gráfadatbázisba, és a lekérdezések futtatása minél gyorsabb legyen. A kiindulási alap az RDF-ben leírt adathalmaz volt. Számos különböző eljárás létezik az RDF-ből címkézett tulajdonsággráfba (azaz valamely triplestore-ból a Neo4j-be) való konverzióra [27]. Az általam használt eljárás a következő szabályokon [28] alapuló megközelítés:

1. Az alany – állítmány – tárgy hármasok alanyai gráfcsomópontokra (node-okra) képződnek le a Neo4j-ben. Minden ilyen node rendelkezik egy kulcs-érték pár formájában adott tulajdonsággal, amelynek a kulcsa „uri”, az értéke pedig az adott RDF erőforrás URI-ja.
2. Az állítmányok kétféleképpen transzformálódhatnak át címkézett tulajdonsággráfba. Amennyiben a hármas tárgya literál, akkor az állítmány az alany node tulajdonságaként jelenik meg, amelynek értéke maga a literál lesz.
3. Amennyiben a tárgy RDF erőforrás, akkor az állítmány kapcsolatként fog megjelenni a tulajdonsággráfban az alanyt és a tárgyat reprezentáló gráfcsomópontok között.
4. Az rdf:type kifejezések a Neo4j-ben címkékké (azaz kategóriákká) képződnek le. Ezzel a módszerrel tudjuk ábrázolni azt a fontos metaadatot, hogy egy adott objektum mely osztály példánya, tehát „milyen típusú”. (Az RDF-ben nem csak az adatok, hanem a metaadatok is hármasokként jelennek meg, azonban ezeket nem feltétlenül célszerű a fenti szabályok alapján transzformálni.)

A 4. pontban leírt szabály azonban nem teljesen megfelelő, ha az ontológiában szereplő alosztálynak egy példányát kell leírnunk, ebben az esetben ugyanis az osztályhierarchiára vonatkozó információ a transzformáció során elveszhet. Az alábbi példa szemlélteti a problémát: az iACC-ban az EquipmentProperty osztály három alosztálya az EquipmentHeight, EquipmentWidth és EquipmentQuantity (1.3 ábra). Ha egy objektum az EquipmentWidth-nek egy példánya, akkor a 4. pontban leírt szabály alkalmazásával egy olyan node-ot kapunk, amely vagy EquipmentProperty, vagy EquipmentWidth címkével rendelkezne. Az első esetben a specializáció általi előnyöket (és metainformációt) veszítjük el, az utóbbi esetben pedig a generalizáció általiakat, ha csak nem tároljuk ezeket az információkat valamilyen más módon a gráfban.

A [28] által leírt, fentebb ismertetett szabályokat a saját feladatomhoz a következőképpen módosítottam:

1. Az alany – állítmány – tárgy hármasok alanyai ugyanúgy gráfcsomópontokra (node-okra) képződnek le, két módosítással:
    - Az RDF 1.1 verzióban már a URI helyett annak általánosítása, az IRI (Internationalized Resource Identifier) szerepel, így én is ezt használtam.

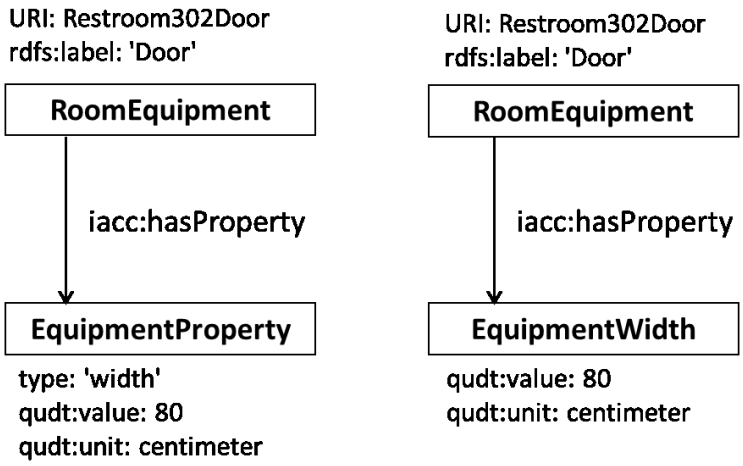
- Az IRI-t sem szükséges tulajdonság formában átvinni, amennyiben a csomópont egyéb tulajdonságai (például a neve) kielégítően azonosítják azt.

2. Az állítmány transzformálása, amennyiben a hármas tárgya literál, ugyanúgy node tulajdonságra történik, azzal a kiegészítéssel, hogy típuskonverzióra lehet szükség, hiszen a Neo4j csak néhány tulajdonság-típust támogat (Integer, Float, String, Boolean, Point, dátum-idő típusok), míg az RDF több tucatnyit. A szám jellegű RDF típusok Integerre vagy Floatra konvertálhatók (esetleges pontosság vesztés mellett), egyebek pedig Stringre.
3. Az állítmány transzformálása, amennyiben a tárgy RDF erőforrás, a [28] 3. pontjával egyező.
4. Az rdf:type kifejezések leképezése címkékre történik akkor, ha az adott osztálynak nincs szülő osztálya. Ha van, akkor ennek transzformálására kétféle módszert adok meg, amelyeket alább összehasonlítok majd:
    a. A címke a szülő osztály neve lesz, míg a példány közvetlen típusa (vagy annak rövidítése) egy "type" tulajdonságként jelenik meg.
    b. A címke a közvetlen típus lesz, ekkor azonban a szülő osztályra (az osztályhierarchiára) vonatkozó információ a transzformáció során elvész.

Példán keresztül szemléltetve a 4.a. és a 4.b. közötti különbség az alábbi módon néz ki. A következő RDF leírás az iACC osztályain alapul; egy olyan helyiséget ír le, amely rendelkezik egy 80 centiméter széles ajtóval.

```
@prefix iloc: <http://lod.nik.uni-obuda.hu/iloc/iloc#> .
@prefix iacc: <http://lod.nik.uni-obuda.hu/iacc/iacc#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
:Restroom302 a iloc:Room;
rdfs:label "Restroom"@en;
iacc:hasEquipment :Restroom302Door.
:Restroom302Door a iacc:RoomEquipment;
   rdfs:label "Door"@en;
   iacc:hasProperty [
        a iacc:EquipmentWidth;
        qudt:value 80;
        qudt:unit qudt:centimeter.]
```

Ebben az esetben a transzformáció egy névtelen gráfcsomópontot eredményez, amelynek a címkéje lehet EquipmentProperty, tehát a szülő osztály neve, vagy EquipmentWidth, az alosztály neve (1.5 ábra).



1.5. ábra: Különféle átírási lehetőségek szemléltetése

Amennyiben az alosztályt választjuk címkeként, akkor elvész a szülő osztályhoz tartozás, mint információ, amennyiben az ősosztályok neveit nem adjuk hozzá további címkékként a csomóponthoz. Ha a szülő osztályt választjuk, akkor az alosztály, mint tulajdonság megjelenhet a gráfcsomópont egyéb tulajdonságai mellett. (A fenti szabályok szerint a qudt:value és a qudt:unit is csomópont tulajdonságokká alakulnak.)

Három lehetőséget mérlegeltem az alosztály transzformáció megvalósításához, mindhármat a Cypher nyelv megfelelő CREATE utasításával fogom szemléltetni.

Az első lehetőség:

```
CREATE (r:Room { name:"Restroom 302", label:"Restroom" });
MATCH (r:Room {name:"Restroom 302"})
MERGE (r)-[:HAS_EQUIPMENT]->
(e1:RoomEquipment { name:"Restroom 302 Door", label: "Door"})
-[:HAS_PROPERTY]->
(p1:EquipmentProperty { type: "width",value: 80, unit:
centimeter})
```

A második lehetőség:

```
CREATE (r:Room { name:"Restroom 302", label:"Restroom" });
MATCH (r:Room { name:"Restroom 302"})
MERGE (r)-[:HAS_EQUIPMENT]->
(e1:RoomEquipment { name:"Restroom 302 Door", label: "Door"})
-[:HAS_PROPERTY]->
(p1:EquipmentWidth { value: 80, unit: centimeter})
```

A harmadik lehetőség nem teljes transzformációt valósít meg, mivel a teljesítmény növelése céljából kihagyja a mértékegységet, mint nem létfontosságú információt.

Ez természetesen csak akkor működőképes, ha az adatmodellt használó alkalmazás a hasonló mennyiségeket mindig azonos mértékegységekkel tekinti, és az adattárolás is ennek megfelelően történik.

```
CREATE (r:Room { name:"Restroom 302", label:"Restroom" });
MATCH (r:Room { name:"Restroom 302"})
MERGE (r)-[:HAS_EQUIPMENT]->
(e1:RoomEquipment { name:"Restroom 302 Door", label:"Door",
width: 80})
```

Ez utóbbi esetben az EquipmentProperty objektum, mint önálló gráfcsomópont teljesen kimarad, ezáltal jelentősen kevesebb csomópontból álló gráfot eredményez, amelyen ennek megfelelően a bejárás is gyorsabb, a lekérdezések pedig kevésbé összetettek lesznek. Nyilvánvalóan megvan az a hátránya, hogy az RDF-ben leírt adatok egy része elvész, tehát ezt csak tesztelési céllal vettem fel a mérések közé.

1.4.4 Példa lekérdezések és mérési eredmények

Az adatmodelleket egy egyszerűbb és egy bonyolultabb lekérdezés futtatásával hasonlítottam össze. A lekérdezések feltételeit természetesen hozzáigazítottam a három különböző reprezentációhoz:

1. RoomEquipment címkéjű node-ok, amelyek height, width illetve quantity tulajdonságokat tartalmazhatnak (ez az egyszerűsített modell teszt céllal)

2. Külön :EquipmentProperty címkéjű node-ok, amelyek „type” tulajdonságként tartalmazzák azt, hogy melyik alosztályhoz tartozik az adott node

3. Külön node-ok az alosztálynak megfelelő (azaz :EquipmentWidth, :EquipmentHeight vagy :EquipmentQuantity) címkékkel.

Az első lekérdezés egy diabetológiai orvosi szoba és egy közeli akadálymentes mosdó közötti útvonalat állapítja meg. Az akadálymentesség kritériuma ez esetben a megfelelő magasságú mosdókagyló.

Első lekérdezés 1. verzió:

```
match p = shortestPath((start)-[*]-(end))
where end.name='Second Floor Toilet'
and start.name='Diabetology Consulting Room'
and all (ns in nodes(p) where
ns.isAccessibleByWheelchair=true)
and (end)-[:HAS_EQUIPMENT]-
(:RoomEquipment{name:"Accessible Sink", quantity:1.0, height:
1.5})
return extract(n in nodes(p) | n.name) as names;
```

Első lekérdezés 2. verzió:

```
match p = shortestPath((start)-[*]-(end))
where end.name='Second Floor Toilet'
and start.name='Diabetology Consulting Room'
and all (ns in nodes(p) where
ns.isAccessibleByWheelchair=true)
and (end)-[:HAS_EQUIPMENT]-(:RoomEquipment {name: "Accessible
Sink"})
-[:HAS_PROPERTY]-(:EquipmentProperty{type:"quantity",
value:1.0})
and (end)-[:HAS_EQUIPMENT]-(:RoomEquipment {name: "Accessible
Sink"})
-[:HAS_PROPERTY]-(:EquipmentProperty {type:"height", value:
1.5})
return extract(n in nodes(p) | n.name) as names;
```

Első lekérdezés 3. verzió:

```
match p = shortestPath((start)-[*]-(end))
where end.name='Second Floor Toilet'
and start.name='Diabetology Consulting Room'
and all (ns in nodes(p) where
ns.isAccessibleByWheelchair=true)
and (end)-[:HAS_EQUIPMENT]-(:RoomEquipment {name: "Accessible
Sink"})
-[:HAS_PROPERTY]-(:EquipmentQuantity{value: 1.0})
and (end)-[:HAS_EQUIPMENT]-(:RoomEquipment {name: "Accessible
Sink"})
-[:HAS_PROPERTY]-(:EquipmentHeight {value: 1.5})
return extract(n in nodes(p) | n.name) as names;
```

A lekérdezéseket különböző start és cél helyszínekre (különböző hosszúságú utakra) is lefuttattam. Olyan útvonal is szerepelt, amely gyakorlatilag az egész épület bejárását jelenti, tehát nem csak rövid, néhány csomópontos utakra történt meg a mérés. Összesen 4-féle cél node-ot helyettesítettem be, ezekből 4 különböző hosszúságú utat kaptam.

A második lekérdezés sokkal összetettebb olyan értelemben, hogy több akadálymentességi feltételt is vizsgál a célobjektum esetében. Ebből is három verzió készült a három adatmodellhez, ám mivel a fenti lekérdezések alapján a különbségek egyértelműek, ezért itt csak a harmadik adatmodell-típushoz tartozót ismertetem:

```
match p = shortestPath((start)-[*]-(end))
where end.name='Second Floor Toilet'
and start.name='Diabetology Consulting Room'
and all (ns in nodes(p) where
ns.isAccessibleByWheelchair=true)
and (end)-[:HAS_EQUIPMENT]-(:RoomEquipment {name:"Accessible
Sink"})
-[:HAS_PROPERTY]-(:EquipmentQuantity {value: 1.0})
and (end)-[:HAS_EQUIPMENT]-(:RoomEquipment {name: "Accessible
Sink"})
-[:HAS_PROPERTY]-(:EquipmentHeight {value: 1.5})
and (end)-[:HAS_EQUIPMENT]-(:RoomEquipment {name: "Accessible
Mirror"})
```

```
-[:HAS_PROPERTY]-(:EquipmentQuantity {value: 1.0})
and (end)-[:HAS_EQUIPMENT]-(:RoomEquipment {name: "Accessible
Mirror"})
-[:HAS_PROPERTY]-(:EquipmentHeight {value: 1.0})
and (end)-[:HAS_EQUIPMENT]-(:RoomEquipment {name:"Ramp"})
-[:HAS_PROPERTY]-(:EquipmentQuantity {value: 1.0})
and (end)-[:HAS_EQUIPMENT]-(:RoomEquipment {name:"Ramp"})
-[:HAS_PROPERTY]-(:EquipmentWidth {value: 2.0})
return extract(n in nodes(p) | n.name) as names;
```

Az így előálló 24 lekérdezést többször lefuttattam az átlagos futási idő megállapításához. Az eredmények az 1.1. számú táblázatban találhatók, az értékek milliszekundumban értendők. Az eredmények nyilvánvalóan nem önmagukban, hanem egymáshoz képest, relatíve értékelhetők, hiszen a konkrét futási idő függ a konfigurációtól, amelyen fut az adatbázis-kezelő.

M1, M2 és M3 jelölik a fent leírt három adatmodellt. R1, R2, R3 és R4 jelölik a négy különböző cél objektumot (ezzel együtt a különböző hosszúságú utakat).

1.1. táblázat: A tesztlekérdezések átlagos futási idői (ms)

Első lekérdezés					
	R1	**R2**	**R3**	**R4**	**Avg**
M1	39	45	40	38	**40.5**
M2	64	60	53	52	**57.25**
M3	69	65	62	60	**64**
Második lekérdezés					
	R1	**R2**	**R3**	**R4**	**Avg**
M1	52	56	53	62	**55.75**
M2	136	138	142	151	**141.75**
M3	141	149	151	158	**149.75**

A korábbi adatmodellen való futási időkkel összehasonlítva jelentős gyorsulás látszik. Míg az Apache Marmottán több másodperces futási idők jellemzők (1-10 s közötti tartomány), és bár a Virtuoso nem szabványos megoldásai valamivel gyorsabbak voltak (1 s körüli idők), a gráfadatbázis 50-150 ms futási idői legalább egy nagyságrenddel kedvezőbbek.

1.4.5 Összegzés

Ahogyan az várható volt, az első adatmodellen végrehajtott bejárások jelentősen gyorsabbak voltak a kevesebb felderítendő csomópont miatt. Megállapítható, hogy amennyiben konkrét alkalmazásokban szükségessé válik a lekérdezések gyorsítása, akkor az adatmodell célorientált egyszerűsítése jó eredményt hozhat.

Bár a 2. modell ebben az esetben egy kicsivel gyorsabbnak bizonyult, a különbség nem jelentős a letöltés idejéhez képest (egy átlagos középületet leíró gráf mérete mellett legalábbis), ezért általános érvényű következtetést ebből nem lehet levonni. Általános felhasználásra vonatkozóan további tesztek szükségesek a döntéshez.

A tesztek igazolták viszont azt az előfeltevést, amely az adattárolás módjában meglévő különbség miatt amúgy is sejthető volt, hogy a Neo4j gráfadatbázisban ugyanazon útvonaltervező lekérdezések sokkal gyorsabban lefutnak, mint RDF store-ban tárolt adatokon. Ezáltal éles rendszerben, nagyobb mennyiségű felhasználó kiszolgálása is gazdaságosan megvalósíthatóvá válik.

1.5 Kapcsolódó tézisek

1. téziscsoport: akadálymentes beltéri navigáció

1.1 tézis

Egy meglévő, épületek belső szerkezetét leíró ontológiához kidolgoztam egy olyan kiterjesztést, amely lehetővé teszi az adott ontológia szerint leírt épületekben a beltéri akadálymentes navigáció implementálását különféle fokú mozgássérültség szintek speciális igényeihez igazodva [S1, S2].

1.2 tézis

Kidolgoztam egy megfeleltetést, amely lehetővé teszi a fenti ontológia szerint leírt adatok transzformálását gráfadatbázisban való tároláshoz, és a gráfadatbázison olyan lekérdezések futtatását, amely mozgássérült felhasználók részletesen megadott igényeinek megfelelő útvonalat talál az épület két pontja között, amennyiben ilyen létezik [S3, S4].

2 Beszédfelismerés robusztussága intelligens otthon rendszerekben

2.1 Motiváció

A világ minden országában emelkedik a populáció átlagéletkora. A [29] szerint 1950 óta a várható életkor két évtizeddel növekedett, ezáltal az idősek aránya a társadalomban egyre nagyobb lesz. Globális szinten szintén igaz, hogy az élettartam nő és a termékenység csökken. Ez azt jelenti, hogy az idősek száma a következő évtizedekben várhatóan hasonló trendet követ majd.

A világ népességének elöregedése miatt a gondozás és gyógykezelés költsége is emelkedik. Bár a modern orvostudomány lehetővé teszi az egészségünk megőrzését életünk nagyobb részében, az életkor jelentős rizikófaktort jelent olyan civilizációs betegségek tekintetében, mint a cukorbetegség, rák, valamint a szív- és érrendszeri panaszok [30]. A testi egészség és mozgásképesség az életkor előrehaladtával folyamatosan romlik.

Számos idős embernek kell öregek otthonába vagy egyéb intézménybe költöznie, amikor már nem képesek többé önálló életvitelt folytatni. Az életvitel-segítő rendszerek olyan otthoni környezetet biztosítanak, amely támogatja az idős vagy mozgássérült személyeket a mindennapi feladataik elvégzésében. Gondosan megtervezett funkciói által egy életvitel-segítő rendszer hozzájárul az önállóbb életvitel fenntartásához, támogatja a napi feladatok elvégzését, és csökkenti a személyes gondozás igényét.

2.1.1 Az okosotthon rendszerekkel szemben támasztott kritériumok

Az életvitel-segítő rendszerek hardver- és szoftverrendszerek, beágyazott rendszerek és harmadik féltől származó szolgáltatások összességét jelentik. Számos esetben a felhasználói felületet gépi beszéd és beszédfelismerés formájában valósítják meg, ezáltal a súlyosan mozgássérült személyek számára is használhatók. A beszédvezérelt életvitel-segítő rendszereknek azonban meg kell felelniük egy sor speciális

követelménynek, amely eltér a hagyományos szoftvertermékekre vonatkozó követelményektől.

Az intelligens otthon rendszerek vezérlőszoftvere biztonságkritikus rendszernek tekinthető, és elvárt a közel valós idejű működés megvalósítása, azonban néhány kiegészítő kritériumnak is teljesülnie kell a hasonló (real-time, biztonságkritikus) szoftverekhez képest. Ezért az ilyen rendszerek tervezésénél a szokásos gyakorlatokon túl további szempontokat is szükséges figyelembe venni.

A biztonságkritikus rendszerek esetében a legfontosabb a veszélyhelyzetek elkerülése. A biztonság-központú és a megbízhatóság-központú mérnöki tervezés között a fő különbség az, hogy amíg az előbbi a veszélyhelyzetek elkerülésére fókuszál, addig az utóbbi az ilyen helyzetekből eredő költségek csökkentésére.

Az életvitel-segítő rendszerek működési kritériumai a veszélyek elkerülésére kell koncentráljanak elsősorban, és a megbízhatóságot is ebből a szemszögből érdemes megközelíteni.

2.1.2 A mozgássérültséget okozó betegségek beszédre gyakorolt hatása

Egyes krónikus neuromuszkuláris betegségek, mint az amiotrófiás laterálszklerózis (ALS) [31] vagy a szklerózis multiplex (multiple sclerosis - MS) [32] a mozgásképesség károsítása mellett dizartriát is okoznak: ezen motoros rendellenesség tünetei közé tartozik a beszéd torzulása, zavart szenvedhet az artikuláció, prozódia, beszédhangképzés (fonáció). Mivel a fenti betegségek progresszív természetűek, ezért a mozgásfunkciók leépüléséhez hasonlóan a beszéd károsodása is egyre fokozódhat az idő előrehaladtával. A beszéd eltorzulása súlyos akadályt jelenthet a betegeknek a napi tevékenységeik során, szélsőséges esetben csaknem ellehetetleníti a szociális interakciókat.

A kutatások azt mutatják, hogy ezen betegségek esetén a dizartria a többi tünettel együtt fokozatosan rosszabbodik az idő előrehaladtával [33, 34], és mivel az orvostudomány mai állása szerint sajnos gyógyíthatatlan betegségekről van szó, ezért meglehetősen szélsőséges állapotok állhatnak elő. Az is tipikus jelenség, hogy a magánhangzók hosszabb ideig maradnak kivehetőek, mint a mássalhangzók [35].

A mozgássérült személyek életvitelét nagyban segítenék a beszédvezérelt berendezések, azonban a betegség ezen aspektusa komoly kihívás elé állítja az intelligens otthon rendszerek beszédfelismerő moduljának tervezőit: nem csak a rosszul kivehető kiejtés problémáját kell megoldani, hanem a lassú, de folyamatos változást is kezelni kell. Az előző részben tárgyalt szempontoknak megfelelően, a súlyosan mozgássérült egyének intelligens otthon rendszereiben minimalizálni (ha lehetséges, akkor kizárni) szükséges a parancsok hibás felismerését, mivel a felhasználónak nincs alternatív módja a hibás parancsvégrehajtás korrekciójára. Például a ház fűtésrendszerének vezérlése esetén, ha az intelligens otthon rendszer egy félreértett hangparancs következtében nem kívánt értékre állítja a termosztátot, egy ép felhasználó odasétálhat a legközelebbi vezérlőpanelhez és korrigálhatja a hibát. Hasonló helyzetben egy súlyosan mozgássérült felhasználó nem képes ugyanerre, meg kell várnia egy segítő személy érkezését. Ezért arra kell törekedni, hogy a hangparancsok felismerését a lehető legegyértelműbbé tegyük; ennek lehetséges megoldásait a dizartria tipikus tünetei alapján kell megalkotni.

2.1.2.1 A dizartria orvosi megközelítése

Az orvostudományi területen zajlott kutatások eredményei [33, 34, 35, 36, 37, 38] jelentős mennyiségű adatot szolgáltatnak a neuromuszkuláris betegségekben szenvedők beszédének változásával kapcsolatban. Az irodalom áttekintése alapján megállapítható, hogy míg bizonyos beszédjellemzők jelentősen károsodnak, addig más jellemzők viszonylag változatlanok maradnak. Habár minden betegség másképp hat a beszédre, és az egyes betegek is némileg különböző tünetegyütteseket mutathatnak, a későbbi változások a beszéd jellegzetességeinek terén megjósolhatók a korai tünetek megjelenése alapján. Például a szklerózis multiplex esetén megfigyelhető, hogy azon páciensek, akiknek már a betegség korai szakaszában megjelennek dizartriás tünetei, a későbbiekben sokkal súlyosabb dizartriára számíthatnak. A tünetek típusa pedig megjósolható annak alapján, hogy a betegség az agy mely területeit károsítja jobban.

A beszédfelismerésben jelentős nehézséget okoznak az alábbi tünetek:

- Artikulációs problémák:
  Az ajkak és a nyelv gyors és pontos mozgásának hiánya elkent beszédet és kevésbé érthető mássalhangzókat eredményez.
- A hangerő szabályozásának hiánya:
  Rövid hangosabb szakaszok a beszédben, vagy tartósan halk beszéd, vagy egy kifejezésen belül csökkenő hangerő (a légzési nehézségek miatt).
- Időzítési eltérések:
  Bizonyos beszédszegmensek meghosszabbodása, vagy rövid szakaszokból álló, abnormálisan gyors beszéd (a légzési nehézségek miatt).
- Ismétlés:
  Bizonyos beszédszegmensek vagy rövid szavak ismétlése spontán beszédben (ellentétben a betanult szövegekkel, amilyen például a versmondás vagy éneklés, amely jellemzően tiszta marad).
- Szaggatott beszéd:
  A szavak széttöredezése szótagokra, változó mennyiségű szünetekkel, a beszéd természetes ritmusa károsodik.
- Diszfónia:
  Nyers, nazális, fátyolos hang, a hangszín és hangerő kevéssé változik.

A dizartria tüneteiben kifejezetten biológiai alapú nemi különbségek nem mutatkoznak, azonban vannak olyan szociokulturális tényezők, amelyek befolyásolhatják a beszédhangképzést. Olyan társadalmakban, ahol tömegjelenségnek tekinthető a függőséget okozó, bódító hatású anyagok használata, ott a szerválasztásban megmutatkozó nemi különbségek okozhatnak eltérést a beszéd alakulásában (például Magyarországon az egészségügyi dolgozók megfigyelései szerint a férfiak az alkoholt, míg a nők a dohányzást preferálják az állapotuk pszichológiai hatásaival való „megküzdés” érdekében).

2.1.3 Egy életvitel-segítő rendszer potenciális veszélyei

Az életvitel-segítő rendszerek által teremtett potenciális veszélyhelyzeteket illusztrálja a következő példa:

Egy szklerózis multiplexben szenvedő hölgy otthonát felszerelték egy beszédvezérelt életvitel-segítő rendszerrel. A hölgy súlyosan mozgássérült, csak a fejét képes mozgatni, a végtagjait egyáltalán nem. Korábban minden nap járt hozzá egy nővér, akire szüksége volt minden tevékenységéhez a beszélgetést kivéve. Az életvitel-segítő rendszerre támaszkodva önállóan képes ki- és bekapcsolni a televíziót, hangerőt állítani, csatornát váltani, Skype-hívásokat kezdeményezni és fogadni, a speciális ágya fej- és láb pozícióit állítani, lámpát kapcsolni, csengetni az emeleten tartózkodó segítőknek, valamint segélyhívást kezdeményezni.

Háromféle veszélykategóriát különböztethetünk meg ennek a rendszernek a működésében, amely veszélyeztetheti a felhasználót: teljes működésképtelenség, részleges működésképtelenség, valamint hibás működés, ideértve a beszédparancsok hibás felismerését.

A) Teljes működésképtelenség

Ha a rendszer nem működik valamilyen oknál fogva, és erről nem tájékoztatja a felhasználót és a segítőket, az azért veszélyes, mert mindenki aszerint szervezi a teendőit, mintha a rendszer működne. Tehát akár magára is hagyják a beteget egy időre annak tudatában, hogy segítséget tud hívni, ha szükséges. Amikor egy sikertelen parancsvégrehajtás után kiderül, hogy ez nem így van, a felhasználó (szó szerint) már semmit sem tehet.

B) Részleges működésképtelenség

Ilyen szituáció többnyire hardverhiba miatt fordul elő, amikor a rendszer valamelyik hardverkomponense elromlik, vagy akár csak lemerül benne az elem. Ez jobb esetben az adott eszköz használhatatlanságát jelenti, miközben egyéb funkciók használhatók maradnak. Egy gondatlanul tervezett vezérlőszoftver azonban akár le is fagyhat, amennyiben valamelyik hardverkomponens nem válaszol. Ebben az esetben az A kategória szerinti helyzet áll elő, azaz a rendszer teljesen működésképtelenné válik az újraindításig. A moduláris felépítéssel megelőzhető ez a fajta probléma.

Ha egy modul nem válaszol, a vezérlőszoftver továbbra is üzemképes marad, és értesíti a felhasználót (és lehetőleg a rendszer operátorait és a segítő személyzetet is) a hibáról.

C) Hibás működés

Ha a rendszer működik, de hibásan hajtja végre az utasításokat, az többnyire a hibás beszédfelismerésnek köszönhető. Ilyenkor kétféle hiba fordulhat elő.

1) Hamis pozitív

A hamis pozitív felismerés azt jelenti, hogy a rendszer olyan parancsot detektál és hajt végre, amely nem hangzott el. Bizonyos fajta felismerő algoritmusok érzékenyek a háttérzajra, különösen, ha az emberi hangot tartalmaz (ilyen például egy TV műsor a háttérben), és megpróbálják ezt mindenképp valamelyik paranccsal azonosítani. Ha ez megtörténik, az olyan veszélyes szituációkhoz vezethet, mint például az ágy pozíció véletlenszerű állítgatása, a fűtés vagy a hangerő szélsőségesen magas értékre való beállítása.

2) Hamis negatív

Ez a hiba az előző ellenkezőjének tekinthető, azaz a rendszer nem ismer fel, és ezért nem hajt végre egy kiadott parancsot. Az esetek többségében ez legfeljebb kényelmetlenséget okoz, mivel többször, esetleg tisztábban ejtve meg kell ismételni a parancsszót. Van azonban egy kivétel ez alól: egy olyan parancs, amelynek a figyelmen kívül hagyása súlyos következményekkel jár, mégpedig a segélyhívás. Ezt a funkciót különleges körültekintéssel kell implementálni, és prioritással kezelni a többi parancshoz képest.

Ezen veszélyek és hibák elkerülését szigorúan szem előtt kell tartani az okosotthon rendszerek tervezése és megvalósítása során. Olyan kritériumokat és elveket kell találni (különösen a beszédfelismerő algoritmusok megválasztásánál, megtervezésénél), amelyek ezt elősegítik.

2.2 Kritériumok megbízható életvitel-segítő rendszerek tervezésére és fejlesztésére

Egy biztonság-kritikus rendszer (definíciója szerint) képes kárt okozni a felhasználóknak. Ezért célszerű olyan szoftverfejlesztési módszereket alkalmazni, amelyek garantálják a robusztus, hibatűrő működést.

Bár számos módszert és javaslatot találunk a helyesen működő, real-time és biztonság-kritikus rendszerek tervezéséhez, kevés információ található arról, hogy milyen konkrét kritériumoknak kell megfeleljenek ezek a rendszerek. Ha találunk, azok is többnyire erőművekre, repülésre és ehhez hasonló rendszerekre vonatkoznak. Ennek oka a széles körben elérhető, megvásárolható életvitel-segítő rendszerek viszonylagos újdonságában kereshető.

Ezzel szemben számos konkrét biztonsági kritériumot találunk az IT biztonság területén. Ezek átalakítása általános működésbeli kiválósági kritériumokká hozzájárulhat az életvitel-segítő rendszerek működésének javításához.

A COBIT keretrendszer egy világszerte jól ismert és elfogadott eszköztár az információbiztonság területén. A COBIT kritériumok általánosítása és alkalmazása beszédvezérelt intelligens otthon rendszerekre olyan irányelveket eredményez, amelyek segítségével megbízhatóbb életvitel-segítő rendszereket tervezhetünk.

2.2.1 A vállalatok működtetési kritériumainak értelmezése az életvitel-segítő rendszerekre

Az "Információra és a kapcsolatos technológiára vonatkozó kontroll célkitűzések" (Control Objectives for Information and related Technology – COBIT®) keretrendszer egy világszerte ismert és elismert audit eszköztár és bevált gyakorlatok gyűjteménye, amelyet az ISACA (Information Systems Audit and Control Association) nevű szervezet bocsát ki. A COBIT 7 kritériumot határoz meg a vállalatok és egyéb szervezetek biztonságos és hatékony információ-kezelésének elősegítéséhez: ezek az eredményesség, hatékonyság, bizalmasság, sértetlenség, rendelkezésre állás, megfelelőség és megbízhatóság [39].

[40] szerint ezen kritériumok általánosíthatók vállalati működési kritériumokká, tehát ebből kiindulva a COBIT nem kizárólag az információ-kezelésre vonatkoztatható, hanem komplex, emberek által működtetett rendszerekre is.

Egy vállalat és egy AAL rendszer működésében számos hasonlóság figyelhető meg. Mindkettő esetében igaz, hogy egy vezető személy (AAL-nél a felhasználó) valamely cél elérése érdekében utasítást ad, amelynek hatására a rendszer ágensei az utasításnak megfelelő tevékenységeket végzik el. Mind a vállalati tevékenységnél, mind az okosotthon működésénél fontos kritérium a megbízható és hatékony végrehajtás, a törvényi szabályozásoknak és ipari szabványoknak való megfelelés, valamint a hosszú távon fenntartható és fejleszthető rendszer kialakítása. Végül mindkét rendszerre igaz, hogy a kiadott utasítás végrehajtása során fennáll a tévedés lehetősége, azonban a megfelelő módszerek bevezetésével a működés robusztussága növelhető.

Az életvitel-segítő rendszerekre való értelmezhetőség vizsgálata során számos olyan szempontot találtam, amelyeket saját gyakorlati tapasztalataim is megerősítettek, és amelyek a későbbi munkám során fontos alapelvként jelennek meg. Az alábbiakban ezeket mutatom be.

A. Hatékonyság

A COBIT definíció:

„Az információ hatékony, ha előre eltervezett, dokumentált és költséghatékony módon szolgáltatják, figyelembe véve a humán és anyagi erőforrások optimális kihasználását és a problémamegoldás módját.”

Okosotthon vonatkozás:

Egy okosotthon rendszer akkor tekinthető hatékonynak, ha az erőforrások (hardver, hálózat, villamos energia) kihasználása tervezett, költséghatékony módon történik.

A mérsékelt erőforrásigény alapvető szempont az életvitel-segítő rendszerek tervezésénél és megvalósításánál. Mivel a vezérlőszoftvernek folyamatosan aktívan figyelnie kell a bejövő utasításokra, az energiafogyasztás különösen fontos faktor, ezért jó döntés lehet ezt alacsony fogyasztású SBC (single board computer – egykártyás

számítógép) hardveren futtatni, azonban ekkor számolni kell a korlátozott számítási kapacitással és memóriával.

Az SBC-k előnye az alacsony fogyasztás mellett az is, hogy zajtalan működésre képesek. A PC-n futó vezérlőszoftver egyik legzavaróbb hátránya a ventillátor zúgása.

B. Rendelkezésre állás

A COBIT definíció:

„Az információ rendelkezésre állása azt jelenti, hogy egy adott ügyet tekintve, az rendelkezésére áll minden kompetens résztvevőnek tervezett, előre jelezhető, dokumentált módon, a rendelkezésre állásra vonatkozó előzetes megegyezések szerint (hozzáférés módja, időintervallum stb.).”

Okosotthon vonatkozás:

A rendszer rendelkezésre kell álljon minden arra jogosult felhasználó számára (aki általában egyedül a sérült személy, más hangjára nem szabad reagálnia a rendszernek), amikor szükség van rá (ami azt jelenti, hogy folyamatosan, napi 24 órában). Emellett az is szükséges, hogy a rendszer közel valós időben működjön, azaz minimális késleltetéssel reagáljon a felhasználók utasításaira. A nagy mértékű (vagy éppen változó mértékű) késleltetés ugyanis megingatja a felhasználó bizalmát a rendszerben.

Egy AAL rendszernek mindig elérhetőnek kell lennie, különösen, ha egy súlyosan mozgássérült személy otthoni környezetét működteti, mivel az ilyen felhasználóknak nincs ezen kívül egyéb lehetőségük akár a legegyszerűbb napi tevékenységek (mint például a televízió átkapcsolása másik csatornára, vagy a világítás felkapcsolása) elvégzésére sem.

A redundancia és a modularitás két fontos összetevő a folyamatos rendelkezésre állás biztosítása szempontjából. Az AAL vezérlőszoftverének képesnek kell lennie a rendszermodulok állapotának monitorozására, és meghibásodás esetén képesnek kell lennie átkapcsolni a tartalék hardver modulra. Ha tartalék modulok nem érhetők el, akkor a szoftvernek értesítenie kell a felhasználót (és esetleg a kijelölt gondviselőket

vagy távoli segítőket) a hibáról, a megmaradt modulokat pedig továbbra is megfelelően működtetnie kell.

A lokális komponensek működtetése mellett nem szabad megfeledkezni a hálózati kapcsolatról sem, ha van olyan szolgáltatás, amely erre támaszkodik. Tipikusan szokott ilyen lenni, hiszen akár a netrádió szolgáltatások, akár az internetes audio- és videohívások (Skype, Messenger, Viber stb.) kézenfekvő módon integrálhatók egy AAL rendszerbe. Amennyiben a segélyhívás az előbb felsorolt szolgáltatók egyikén keresztül valósulna meg, akkor az internetkapcsolat kritikus része a rendszernek, amelynek a rendelkezésre állását akár redundancia által, de biztosítani kell.

Bizonyos rendszerekben maga a beszédfelismerés is megvalósulhat távoli szolgáltatásként, valamely tech óriáscég (Google, Microsoft) kapcsolódó szolgáltatásának igénybevételével. Ez azonban nem csak azért kerülendő, mert így a teljes működés kiszolgáltatott az internetkapcsolat meglétének, hanem azért is, mert számos olyan betegség létezik, amely mozgásfogyatékosság mellett beszédfogyatékosságot is előidéz az idő előrehaladtával (ilyen például az ALS vagy a szklerózis multiplex). Ezekben az esetekben az adott személyre tanított, egyedi modellek sokkal biztosabban felismerik a kimondott szavakat.

C. Megbízhatóság

A COBIT definíció:

„Egy vállalat információs rendszere megbízható, ha az információ feldolgozása olyan módon szervezett, hogy az előzetes megegyezés szerinti adatokat úgy biztosítja, hogy az alkalmazottak munkáját a legjobb szakmai gyakorlat szerint támogatja.”

Okosotthon vonatkozás:

A rendszer megbízható, ha a felhasználót a számára szükséges időben és módon támogatja a napi tevékenységei elvégzésében és az önálló életvitelben, és minden alkalommal következetesen működik.

Az AAL rendszernek az utasításokat következetesen, a felhasználó által elvárt módon kell végrehajtania. (Nem kreálhat több problémát, mint amennyit megold.) Minden

lényeges eseményről (főképpen az esetleges hibákról) informálnia kell a felhasználót, hiszen ezáltal alakulhat ki a megfelelő bizalom a felhasználó részéről.

El kell kerülni (vagy automatikusan, rövid idő alatt meg kell oldani) a szoftver szélsőséges lelassulását vagy lefagyását, és általában minden olyan helyzetet, amely a beszéd inputon kívül egyéb felhasználói beavatkozást igényelne.

Elfogadható az, ha nem ideális akusztikus körülmények között a beszédfelismerő modul nem képes azonosítani a kimondott parancsot (hamis negatív eredmény). Az azonban elfogadhatatlan, ha egy AAL rendszer olyan műveletet hajt végre, amelyre nem utasították (hamis pozitív válasz). Az előzőleg említett segélyhívás esete az egyetlen kivétel ez alól.

Ez egy kritikus megszorítás a beszédfelismerő algoritmus tervezése során. Bizonytalanság esetén a szoftver megerősítést kérhet a beszédinterfészen keresztül („Arra gondolt, hogy...”). Ajánlatos jelentősen különböző, egyszerűen azonosítható szavakat vagy kifejezéseket választani a megerősítés vagy elvetés kifejezésére, hiszen bizonytalan felismerés esetén eleve feltételezhető a nem ideális, zajos akusztikus környezet.

2.2.2 A követelmények összegzése

Összefoglalva tehát, egy súlyosan beszéd- és mozgásfogyatékos személy okosotthon rendszere olyan (közel) állandó rendelkezésre állású megoldás kell legyen, amely a felhasználója által beszéddel vezérelhető, a kimondott parancsokat nagy biztonsággal felismeri még dizartriás beszéd esetén is, illetve bizonytalan felismerés esetén inkább nem hajt végre utasítást, és megerősítést kér (kivétel ez alól a segélykérés). A beszédfelismerő külső szolgáltatásokra nem támaszkodik, hogy az internetkapcsolattól való függést kiküszöbölje. A parancsok végrehajtása konzisztens, és közel azonos válaszidővel működik. A központi vezérlő szoftvermodul lehetőleg kis fogyasztású, csendes SBC-n fusson.

2.3 Okosotthon rendszerek hangutasításainak távolsága dizartriában szenvedő felhasználók esetén

A korábban ismertetett dizartria-tünetek és a betegség progresszív természete nyilvánvalóan megnehezíti a beszéd biztonságos felismerését. Az osztályozás pontosságának szinten tartására (a felismerő algoritmusok módosításán, fejlesztésén túl is) adódik egy ötlet: válasszunk olyan parancsszavakat, amelyek kellően különbözőek ahhoz, hogy nagy fokú beszédromlás mellett is viszonylag jól megkülönböztethetők maradnak. Ehhez azonban szükséges annak a definiálása, hogy a dizartria viszonylatában mit jelent a „kellően különböző", hiszen a tünetek nem egyformán érintik a beszéd minden jellemzőjét.

Célom egy olyan távolság megadása, amely az orvosi irodalomból vett adatok alapján alkalmas két hangparancs-kifejezés közötti távolság dizartria-specifikus mérésére, és ezáltal elősegíti a beszédvezérelt rendszer utasításkészletének összeválogatását. A távolsággal szemben támasztott követelmények:

- A kifejezések különböző részeinek és tulajdonságainak hozzájárulása a távolsághoz attól függjön, hogy jellemzően mekkora a dizartria hatása az adott részre.
    - A stabilabb, kevésbé változó részekben és tulajdonságokban meglévő különbségek nagyobb távolságot eredményezzenek.
    - Ezzel szemben az erős torzulásnak kitett jellemzők kevesebbet számítsanak a távolság meghatározásában, mivel a felismerésük kétséges, és idővel megváltozhatnak vagy eltűnhetnek a beszédből.
- A távolságot meghatározó függvény paramétereit lehetséges legyen a betegségnek (szakorvos által előre jelzett) várható alakulásához igazítani.

2.3.1 Szövegeken értelmezett távolság metrikák a szakirodalomban

Számos metrikát ismerünk karaktersorozatok távolságának meghatározására. Közülük a legegyszerűbb a Hamming távolság, amely a két karaktersorozat megfelelő pozícióiban lévő karakterek közti eltérések száma. Definíciójának megfelelően csak egyenlő hosszúságú karaktersorozatokra értelmezett, ezért ritkán használják általános szövegek

közötti távolság megadására. Bár a Hamming távolság általánosítható különböző hosszúságú szövegekre olyan módon, hogy a rövidebb szöveget kiegészítjük üres karakterekkel, ez a megoldás rendkívül hasonló lenne a Levenshtein távolsághoz (szerkesztési távolság).

Két szöveg Levenshtein távolságát azon egykarakteres módosítások (csere, beszúrás, törlés) minimális száma adja, amely ahhoz szükséges, hogy az egyik szöveget a másikba átalakítsuk [41, 42]. Ezt a metrikát általában rövid szövegek illeszkedésének vizsgálatára használják olyan esetekben, amikor kevés eltérés várható (pl. keresés szótárban, helyesírásellenőrzés), de hosszabb szövegek közötti távolság számítása ezzel a módszerrel meglehetősen költséges művelet lenne.

A Damerau-Levenshtein távolság kiegészíti az eredeti Levenshtein távolságot a szomszédos karakterek cseréjének műveletével. A leghosszabb közös részsorozat (longest common subsequence - LCS) olyan távolságot ad meg, ahol csak a beszúrás/törlés művelet engedélyezett (a részsorozat definíciója itt: egyik karaktersorozat akkor részsorozata a másiknak, ha abból karakterek eltávolításával előállítható) [43]. A „szerkesztési távolság” megnevezést egyébként gyakran használják olyan általános string metrikák megnevezésére, ahol a megengedett módosító műveletek a hozzájuk kapcsolódó költség értékével a metrika paramétereként adottak.

Léteznek egyéb megoldások is a szövegek közti távolság számítására, amelyek nem feltétlenül metrikák matematikai értelemben. Ilyen például a Jaro-Winkler távolság, amelyet duplikáció-detektálásra fejlesztettek ki. Ez az eljárás az egyező karakterek száma és a szükséges cserék száma alapján adja meg a hasonlóság mértékét. Emellett (paraméterezéstől függően) hasonlóbbnak tekintheti azokat a szövegeket, amelyeknek az eleje egyezik [41, 42].

A fenti eljárások közös tulajdonsága, hogy nem képesek a karakterek „fontosságát” figyelembe venni, azaz bizonyos megadott karakterek egyezése nem produkál nagyobb hasonlósági értéket, mint más karaktereké. (A Jaro-Winkler távolság pozíció-alapú differenciálása azonban jó irányba mutat a probléma szempontjából.)

2.3.2 A dizartria-specifikus távolság leírása

Az általam megalkotott távolság akkor metrika, ha teljesíti az elvárt matematikai tulajdonságokat (nemnegatív érték, szimmetria, egyenlőségi tulajdonság, háromszög-egyenlőtlenség). A dizartria-specifikus távolság a paraméterezett szerkesztési távolságon alapul, ahol a megengedett szerkesztő műveletek a beszúrás, törlés és csere, nemnegatív költséggel. Ez a fajta távolság megfelel a metrika tulajdonságoknak abban az esetben, ha a paramétereket úgy tudjuk megválasztani (a beteg beszédének állapota szerint), hogy minden szerkesztő művelethez létezik egy egyenlő költségű inverz művelet, valamint a műveletek költsége nem függ a szövegen belüli pozíciótól.

A dizartria jellemző tünetei további követelményeket támasztanak:

- Ha a légzés károsodott, akkor a szótagok (magánhangzók) hossza nem vehető figyelembe (ahogy bizonyos nyelvekben, például a magyar vagy a finn nyelvek esetében normális esetben különbséget jelenthet), azaz a rövid magánhangzók és hosszú párjaik esetében a csere költsége 0 kell legyen.
- Ha a beszéd szaggatott, akkor a szavak határait nem lehet megbízhatóan detektálni, tehát az összetett szavak és külön írt verziójuk (például „sok szög” vagy „sokszög”) között a távolság 0.
- Ha a páciensnek artikulációs problémái vannak, akkor egy magánhangzó cseréje jelentősen nagyobb súllyal kell rendelkezzen, mint egy mássalhangzóé, azaz például a „meg” és a „megy” szavak távolsága kisebb kell legyen, mint a „meg” és „mag” szavak távolsága.
- Ha beszéd közben csökken a hangerő, akkor a Jaro-Winkler távolsághoz hasonlóan nagyobb súllyal különböztethetjük meg a kifejezések elejét (azonban ez megsértheti az inverz műveletek egyenlő költségének kritériumát).

A beszédszegmensek akaratlan ismétlésének problémáját ebben az esetben nem szükséges figyelembe venni, mivel ez a tünet spontán beszédben fordul elő, míg egy betanult, esetleg évek óta használt hangutasítás kiadása nem tekinthető spontán beszédnek. A fentiek alapján a következő távolságot alkottam meg.

Tegyük fel, hogy adott egy P és egy R parancskifejezés karaktersorozatok formájában, azaz a hasonlítani kívánt parancs hangfelvételén már detektáltuk a fonémákat és karaktersorozattá írtuk át (ennek módszere a jelen feladatnak nem része). A $P = p_1\ p_2\ ...\ p_m$ és $R = r_1\ r_2\ ...\ r_n$ parancskifejezésekre, a D(m,n) távolság az alábbi rekurzív szabályok szerint számítható:

$$D(0,j) = \sum_{k=1}^{j} b_k c_{insert}(r_k) \tag{1}$$

$$D(i,0) = \sum_{k=1}^{i} b_k c_{delete}(p_k) \tag{2}$$

$$D(i,j) = \begin{cases} D(i-1,j-1) & if\ p_i = r_j, \\ min\begin{cases} D(i,j-1) + b_j c_{insert}(r_j) \\ D(i-1,j) + b_i c_{delete}(p_i) \\ D(i-1,j-1) + b_j c_{substitute}(p_i, r_j) \end{cases} & if\ p_i \neq r_j \end{cases} \tag{3}$$

ahol:

- *m* a *P* parancskifejezés karaktereinek száma,
- *n* az *R* parancskifejezés karaktereinek száma,
- $i = 1 \ldots m$ a *P* parancskifejezés aktuális karaktere,
- $j = 1 \ldots n$ az *R* parancskifejezés aktuális karaktere,
- c_{insert}, c_{delete}, és $c_{substitute}$ a megfelelő szerkesztő műveletek költségei a beteg állapotának, tüneteinek orvosi értékelése szerint,
- b_i a karaktersorozat *i*-edik pozíciójában lévő karakternek a súlya aszerint, hogy mennyire van közel a kifejezés elejéhez,
- $D(i, j)$ pedig a távolságot jelenti a *P* első *i* betűje és az *R* első *j* betűje között (speciális eset a $D(0, j)$, amely az üres string és az *R* első *j* betűje közötti távolságot adja meg, hasonlóképpen a $D(i, 0)$).

Mássalhangzó-kiejtési zavarok esetén a csere műveletét korlátozhatjuk olyan módon, hogy csak az azonos fonémaosztályokba tartozó hangok közötti cserét engedélyezzük, vagy magánhangzót csak magánhangzóra, mássalhangzót csak mássalhangzóra lehet cserélni.

A b_i súlyok meghatározása szintén orvosi állapotfelmérés alapján történik, annak alapján, hogy a betegnek gyengült-e a légzése olyan mértékben, hogy az egy parancs kiejtésének időtartama alatt jelentősebb hangerőcsökkenést eredményezzen.

Ha a tünetek szükségessé teszik, akkor a hangutasítások előfeldolgozása során figyelmen kívül hagyhatjuk a beszéd közbeni szüneteket, illetve az összevetés során a parancsokból eltávolíthatjuk a szóközöket, és (amennyiben az adott nyelv szükségessé teszi) a hosszú-rövid magánhangzópárokat egységesre cserélhetjük.

2.3.3 Használati esetek

Ebben a fejezetben két példán keresztül mutatom be a dizartria-specifikus távolság és a Levenshtein metrika közötti különbséget.

Az intelligens otthon rendszerek parancskészletében gyakran előforduló szavak a „megnyitás” és a „vészhívás”. Egységnyi szerkesztési költségekkel számolva a kettő közötti Levenshtein távolság 6-nak adódik, mivel ennyi csere szükséges ahhoz, hogy az egyiket a másikba transzformáljuk. Ennek alapján a két szó nagyon különbözőnek tűnhet.

Tegyük fel, hogy az intelligens otthon rendszer felhasználójának artikulációs problémái vannak, és hajlamos hosszabban ejteni a szótagokat. Az előfeldolgozás során tehát szükséges egységesíteni a mássalhangzókat. Továbbá tegyük fel, hogy a páciens orvosi értékelése alapján megállapítottuk, hogy a *c(magánhangzó) = 1* és a *c(mássalhangzó) = 0,4* azaz a magánhangzókon és mássalhangzókon végzett műveletek költsége jelentősen eltérő. Ebben a konkrét esetben tehát a „megnyitas” és „veszhivas” között 4 db mássalhangzócsere szükséges, amelynek költsége összesen 1,6 lesz. Ez a szám jobban tükrözi a páciens állapotával járó beszédtorzulások jellegét és mértékét.

Egy másik tipikus probléma az intelligens otthon rendszerekben a hasonlóan kezdődő utasítások előfordulása, amennyiben a felhasználónak a légzési nehézségei miatt elhalkul a beszéde. Ilyen esetekben a kifejezések végének eltérését kisebb súllyal kell figyelembe venni. Például vegyünk egy olyan esetet, amikor $i < 9$ esetén a $b_i = 1$, aztán lineárisan csökken $i = 18$ pozícióig a páciens orvosi értékelése alapján. A „Kapcsold be a tévét” és a „Kapcsold be a rádiót” utasítások közötti Levenshtein távolság 5, míg a dizartria-specifikus távolság csak 2.

2.3.4 Összegzés

Ezen példák esetén azért helytállóbb a beszédfelismerő algoritmus szemszögéből a kisebb távolság, mert a megadott körülmények esetén a megadott kifejezéspárok a dizartriával érintett páciensek beszédében sokkal hasonlóbban hangzanak, mint az ép beszédben. Ha igazán robusztus felismerést szeretnénk megvalósítani, akkor a beszédfelismerő algoritmus továbbfejlesztése mellett szükséges minden egyéb lehetőség megragadása is, amely hosszú távon biztonságosabbá teszi a rendszer használatát.

Az ismertetett dizartria-specifikus távolság használata különösen fontos a súlyos dizartriával küzdő betegek intelligens otthon rendszereinek tervezésénél. A megfelelően választott parancsokat ugyanis sokkal nagyobb biztonsággal lehet felismerni a betegség későbbi szakaszaiban is. Akár egy minimális távolság meghatározása is szükséges lehet a beteg állapotának felmérése és prediktálása szerint.

A dizartria-specifikus távolság hasznossága nyilvánvalóan azon múlik, hogy a paramétereket az orvosi állapotértékelésnek és az állapotromlás előre jelzésének megfelelően állítsuk be. Az utasítások későbbi megváltoztatása nem szerencsés, különösen ha a felhasználó már évek óta használja napi szinten a rendszert, mert ekkor már nagyon nehéz lenne áttanulni más utasításokra. Nyilvánvalóan nem lehetetlen, főleg fiatalabb felhasználók esetén, hiszen ha valakinek ez az egyetlen módja arra, hogy bármit is önállóan elvégezzen, akkor várhatóan maximálisan együttműködő lesz a rendszer megtanulásában, újratanulásában is akár. De minden szempontból sokkal praktikusabb, ha a kezdeti utasításkészlettel működik a rendszer teljes élettartama alatt a vezérlés.

Különösen fontos szem előtt tartani, hogy a segélyhívás utasítás távolsága az összes többi parancstól kellően nagy legyen. Egyrészt azért, mert így biztosabb a felismerése ennek a létfontosságú utasításnak, másrészt pedig téves kategorizálás esetén kisebb eséllyel kerül tévesen aktiválásra a vészhívás funkció.

2.4 Beszédfelismerés anytime algoritmusokkal

A 2.2. fejezetben ismertetett biztonsági megfontolások jelentős része a vezérlés, azon belül a beszédfelismerés biztonságára vonatkozik. Bár a jól megválasztott utasításkészlet sokat képes javítani a helyzeten, mégis szükség lehet magának a felismerő algoritmusnak a módosítására, hogy a követelményeknek meg tudjon felelni. A dizartria amellett, hogy nehezítő körülményként jelenik meg, egyúttal irányt is mutat a probléma megoldása felé, amennyiben a tipikus tüneteket figyelembe vesszük a beszédfelismerés megtervezése és megvalósítása során. Egy ilyen lehetséges irány az anytime algoritmusok alkalmazása.

Az ide vonatkozó irodalmat áttekintve megállapítható, hogy a modern kötött szavas, kis méretű parancskészlettel működő beszédfelismerők pontossága 95% körül mozog [44], de az algoritmustól és egyéb tényezőktől függően lehet 80% körül is [45]. A fonémák osztályozásának pontosságára modern módszerekkel 85-90% körüli értékeket találunk [46], például a TIMIT (Texas Instruments and Massachusetts Institute of Technology) korpuszra található legjobb PER (Phoneme Error Rate) értékek 16,5-17%-osak [47, 48]. Nyelvi modell nélküli, egyszerűbb fonémaosztályozók 65-75% körüli pontosságra képesek [49].

A célom az, hogy olyan beszédfelismerő algoritmust hozzak létre, amely jól tolerálja a dizartria okozta téves fonémafelismeréseket, képes közel real-time választ produkálni, és megbízhatósága megközelíti az egészséges beszédre jelenleg létező megoldásokét. Tehát a fenti számokat tekintve, a saját megoldásomnak viszonylag magas PER mellett is meg kell közelítenie a 95%-os pontosságot.

2.4.1 Az anytime rendszerek jellemzői

Az anytime algoritmusok, modellek és rendszerek olyan speciális valós idejű rendszerek, amelyek képesek változó mennyiségű, vagy kevés erőforrással működni (ide értendő a futási idő vagy a rendelkezésre álló adatok is), a kimenet minőségének csökkenése árán. Ez azt jelenti, hogy tetszőleges idejű futtatást követően képesek kimenetet szolgáltatni, amelyhez hozzá tartozik a minőség, illetve hiba mérőszáma is. Minél hosszabb idő (vagy minél több erőforrás) áll rendelkezésre, annál jobb minőségű

megoldást adnak, tehát az adott helyzetben elérhető legjobb eredményt produkálják. Az anytime módszer nem pusztán a pontosságra optimalizál, hanem a számítás költsége és a kimenet minősége között teremt egyensúlyt, ahol a költségbe beleértendő a számítási idő és a számításhoz használt erőforrások. Ez a fajta rugalmasság teszi alkalmassá az ezeket az algoritmusokat arra, hogy változó körülmények között is működőképesek maradjanak a teljesítmény kritikus csökkenése nélkül [50, 51].

Alapvetően kétfajta megközelítést alkalmazhatunk az anytime rendszerekben. Az egyik az iteratív (megszakítható) algoritmusok használata, mivel ezek mindig képesek valamilyen (nem biztos, hogy elegendően pontos) megoldást szolgáltatni, és a válasz pontossága egyre nő, ahogy a számítás folytatódik. A másik előnyük, hogy nem szükséges előre ismerni a rendelkezésre álló, illetve szükséges számítási időt és kapacitást. Egyszerűen addig fut az algoritmus, amíg szükség nem lesz az eredményre, ekkor megszakítható, és az eredmény kinyerhető. Sajnálatos módon nem minden problémára adódik kézenfekvő iteratív algoritmus. Az úgynevezett „szerződés-típusú" algoritmusok jellegzetessége, hogy ugyan hasonló kompromisszumot kínálnak a futási idő és a kimenet minősége között, de egy előzetesen meghatározott erőforrás- és időszükséglettel rendelkeznek. Ha a meghatározott idő eltelte előtt szakítjuk meg a szerződés-típusú algoritmust, akkor nem garantált, hogy használható eredményt ad [51].

Szerződés-típusú algoritmusok alkalmazásánál az a megközelítés is hatékony lehet, hogy a rendelkezésre álló idő függvényében nem ugyanazt a megoldási módszert futtatjuk hosszabb ideig, hanem egy pontosabb, jobb, ám nagyobb időszükséglettel bíró algoritmust választunk helyette. Lehetséges a szerződés-típusú algoritmusokból iteratív algoritmusok konstruálása egy általános módszer segítségével, amelynek lényege, hogy a szerződés-típusú algoritmust újra és újra lefuttatjuk egyre növekvő futási időkkel. Amikor az iteráció megszakításra kerül, akkor a legutóbbi érvényes eredményt adjuk vissza [52, 53].

Mind a megszakítható, mind a szerződés-típusú algoritmusok kimenetének minősége jellemezhető egy időfüggvénnyel, amelyet teljesítmény-profilnak nevezünk, és lehet folytonos vagy diszkrét is. Lehetséges olyan eset is, amikor a kimenet minősége nem

csak az időtől, hanem a bemenettől is függ, ezt a feltételes teljesítmény-profil függvény írja le [51].

Az anytime algoritmusok másik fontos időfüggő jellemzője a kimenet hasznosságának a mértéke. A real-time rendszerek alapvető tulajdonsága, hogy a számított eredménynek csak egy bizonyos időkorláton belül van jelentősége [50]. Lehetséges olyan rendszer, amikor az időkorlát letelte után a hasznosság mértéke azonnal nullára esik (ezeket nevezzük „merev" valós idejű rendszereknek), és olyan is, ahol fokozatosan csökken le a hasznosság az időkorlátot közelítve (ezek a „lágy" valós idejű rendszerek). Az előbbire példa egy robot sebességének és irányának valós idejű vezérlése: egy akadály detektálásakor a sebesség- és iránykorrekció számításának és végrehajtásának meg kell történnie az akadály eléréséig, hiszen az ütközés bekövetkezése után már zéró a hasznossága a kiszámított korrekciós értékeknek. Az utóbbi kategóriába esik egy intelligens otthon beszédvezérlése: ha a kiadott utasítást csak 5 másodperc után hajtja végre a rendszer, az nem teljesen haszontalan, azonban a parancs kiadása után az idő előrehaladtával a felhasználó egyre türelmetlenebb lesz, esetleg megpróbál másik parancsot kiadni, vagy megismételni az előzőt. Ez pedig félreértéshez vagy hibás működéshez vezethet, amely szélsőséges esetben a felhasználó biztonságát is veszélyeztetheti. Belátható, hogy az anytime algoritmusok ütemezésénél a futási idő két korlátja a kimenet minőségének és hasznosságának elvárt küszöbértékei lesznek.

Az anytime algoritmusok futási idejének meghatározására két főbb megközelítés létezik. Az első, egyszerűbb módszer az, hogy mindenképp egy előre meghatározott ideig futtatjuk az algoritmust egyéb tényezőktől függetlenül. Ez akkor használható, ha legfeljebb kis mértékű bizonytalanság van a kimenet minőségének javulása és az eredmény hasznosságának csökkenési üteme tekintetében. A másik megközelítés a végrehajtás monitorozása. A monitorozó folyamat feladata meghatározni az anytime algoritmus aktuálisan elérhető kimenetének minőségét, ennek alapján prediktálni a további minőségjavulás esélyét, és döntést hozni arról, hogy folytatódjon-e az algoritmus futása, vagy azt megszakítva az aktuális kimenet kerüljön felhasználásra. Az optimális monitoring policy maximalizálja a kimenet várható hasznosságát azáltal, hogy az aktuális futási idő és kimeneti minőség ismeretében eldönti a szükséges további

futási idő mennyiségét, valamint, hogy ezen idő letelte után majd újra monitorozás szükséges, vagy monitorozás nélkül álljon le az algoritmus [54].

Ha egy problémát nem lehet egyetlen anytime algoritmussal leírni, akkor megközelíthető olyan módon is, hogy az adott problémát felosztjuk kisebb részekre, modulokra, amelyek lehetnek megszakítható vagy szerződés-típusú komponensek. (Az így kapott összetett algoritmusok szerződés-típusúnak tekinthetők.) Az ilyen moduláris felépítésű anytime rendszerek igényelnek egy monitorozó egységet, amely a modulok közötti erőforrás-szétosztást felügyeli, illetve a modulok végrehajtását ütemezi. A teljes rendszer teljesítmény-profilja megalkotható a modulok teljesítmény-profil függvényeiből [50, 52].

2.4.2 Anytime beszédfelismerő algoritmusok a szakirodalomban

Iteratív (megszakítható) algoritmusokat ritkán használnak a beszédfelismerés területén annak ellenére, hogy általános jelfeldolgozási feladatokban használatosak, és alkalmasak a real-time rendszerekben fennálló időkorlátnak való megfelelésre. Ennek oka, hogy a modern beszédfelismerő algoritmusok jellemzően fonémák (esetleg 2-3 fonémából álló csoportok) osztályozásán alapulnak, a céljuk pedig nagy méretű szókészletre való alkalmazhatóság. A beszélt nyelvekre általában igaz, hogy vannak egymáshoz nagyon hasonló, mindössze egy-két fonémában eltérő szavak. Diktálás jellegű feladatnál tehát szükséges a teljes szó minden fonémájának pontos osztályozása, ráadásul a kimenet minőségének elfogadható szintje is igen magas emiatt. (Egy diktáló szoftver esetében például, ha mondatonként csak 2-3 szót kézzel kell javítani, az már komolyan visszaveti a használhatóságot, pedig a fonémák százalékában kifejezve igen alacsony a téves osztályozás mértéke.)

Található néhány kivételes példa az anytime megközelítés alkalmazására beszédfelismerés jellegű projektben, például [55, 56] esetében természetes nyelvi mondatok szerkezetének valós idejű elemzésénél használják, a keresési tér méretének csökkentésére. Ez a megközelítés (a lehetséges találatok mennyiségének szisztematikus redukálása) adta az ötletet a saját megoldásomhoz, mivel a stratégia alkalmazható a fix

utasításkészlettel rendelkező okosotthon rendszereknél az elhangzott utasítások osztályozásában is.

2.5 Beszédfelismerés anytime elemző modulokkal

2.5.1 A fejlesztés módszerei

A fejlesztés során abból a gondolatból indultam ki, hogy a klasszikus beszédfelismerés kipróbált, jól működő lépéseit és módszereit alapvetően megtartva, de részleteiben a dizartria tünetei által támasztott követelményekhez igazítva érhetek el jó eredményt. A fejlesztés irányát nagy mértékben meghatározták a 2.2. fejezetben ismertetett elvek.

A beszédfelismerés alapelve, hogy kinyerjük a rögzített hangminta releváns jellemzőit, amelyekből fonéma szintű modellt alkotunk. A fonéma a nyelv legkisebb egysége, ami önálló jelentéssel vagy jelentés-megkülönböztető szereppel bír (nem mindig egyenértékű a beszédhanggal, mivel előfordul, hogy ugyanannak a fonémának pozíciótól függően más-más kiejtései lesznek), ezért szokás a beszédfelismerés feladatát olyan osztályozási feladatnak tekinteni, ahol a rögzített hangot egymást átfedő keretekre bontjuk, keretenként kiszámítjuk a spektrumát, az így összeálló spektrogramon jellemzően további feldolgozást végzünk (például az emberi hallás változó frekvencia-felbontását modellező mel-skálának megfelelően átszámítjuk), meghatározzuk a hangok határait, majd ezeket az egységeket kíséreljük meg fonéma osztályokba sorolni valamilyen módszer, modell segítségével. A kapott sorozatról pedig eldöntjük, hogy a szótárunk melyik elemére illeszkedik legjobban. Ezt az egyszerű lépéssorozatot természetesen kiegészítheti számos előfeldolgozó eljárás vagy speciális megoldás attól függően, hogy milyen környezetben, milyen célra készül a beszédfelismerő.

Esetemben a klasszikus kötött szavas, beszélőfüggő felismerési feladat kiegészült a dizartria tünetei által okozott problémákkal. A tervezés során abból kellett kiindulnom, hogy a légzés és az artikuláció sérülése miatt a beszélő hangerő-kontrollja bizonytalan, a beszéd szaggatott, bizonyos hangok (jellemzően a mássalhangzók) elmosódnak, a szótagok hossza eltérhet a normálistól. Ráadásul a betegség progresszív természete miatt a beszéd minősége is romlik idővel, tehát nem elég a pillanatnyi kiejtésre betanítani egy modellt úgy-ahogy. Hosszú távon leginkább a magánhangzók maradnak

felismerhetők, tehát elsősorban ezekre lehet alapozni a felismerést, ám a magánhangzók esetén is számítani kell némi torzulásra: például a rövid-hosszú magánhangzópárok megkülönböztetése nehézkessé válhat egy idő után, vagy a normális esetben különböző nyelvállásokkal (alsó-középső-felső) képezett megfelelő magánhangzók kiejtése hasonlóvá válik.

Az átlagos magnitudó-különbség függvény (Average Magnitude Difference Function - AMDF) használható a beszéd alapfrekvenciájának megállapítására [57], a beszéd határainak meghatározására a felvett hang és annak eltolt változata közötti távolság megadásával. Az AMDF hasonló az autokorreláció függvényhez, és a szakirodalomban gyakran használt módszerként említik a beszéd zöngés fonémáinak (azaz a magánhangzók és a zöngés mássalhangzók) keresésére [72], ezért felhasználtam a mintában előforduló magánhangzók célzott vizsgálatára. A függvény az n. beszédszegmensre az alábbi módon számítható:

$$AMDF_n(k) = \frac{1}{N} \sum_{i=n-N+1}^{n} |x(i) - x(i-k)| \qquad (4)$$

ahol k a beszéd becsült periódusideje, N az ablakméret (tipikusan 10-30 ms), x a beszédjel vektora, és x(n) a szegmens utolsó értéke.

A magánhangzók olyan, önmagukban kiejthető beszédhangok, amikor a levegő nagyrészt akadálytalanul távozik a tüdőből, megrezegtetve a hangszálakat. Ezen fonémák esetén megállapíthatunk rezonancia frekvenciákat, amelyeket a vokális traktus határoz meg, ezeket formánsoknak nevezzük. A formánsok csúcsokként jelennek meg a hang spektrumában az egyes magánhangzókra jellemző, jól azonosítható módon, így az első három formáns (F1, F2 és F3) kiszámításával lehetővé teszik a magánhangzók formáns alapú osztályozását [58]. Ezt a megközelítést használtam a beszédfelismeréssel kapcsolatos tesztjeimben.

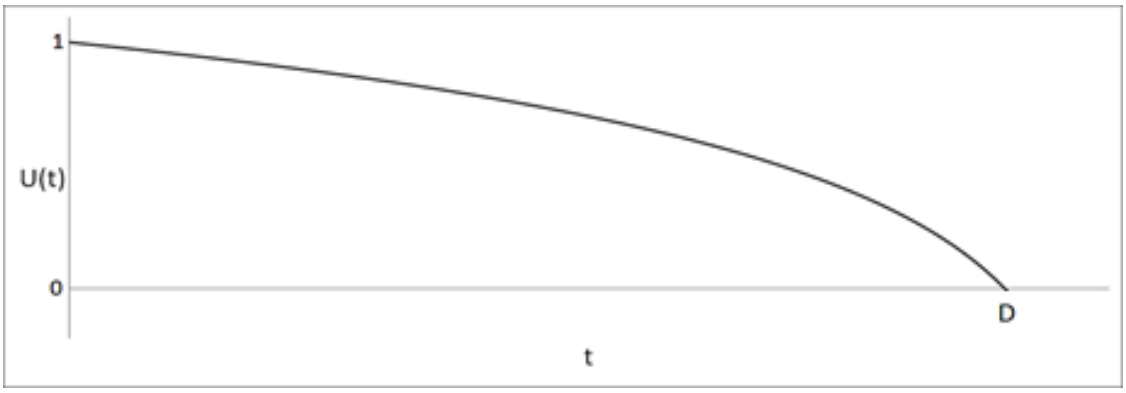
A teszteléshez használt hangminták rögzítését, szerkesztését, vizualizálását és elemzését is a Matlab R2011a verziójában futtattam, a Signal Processing Toolbox függvényeinek felhasználásával.

2.5.2 Saját megoldás ismertetése

Az általam javasolt anytime beszédfelismerés alapelve a következő: adott egy fix utasítás halmaz, amelyeket menüstruktúrába szervezünk. A menüben bárhol tartózkodva ennek az utasításkészletnek egy meghatározott részhalmaza érhető el. Az algoritmus célja annak a meghatározása egy beérkező beszédminta alapján, hogy melyik utasítás hangzott el a legnagyobb eséllyel (nagy valószínűséggel az aktív részhalmaz egy eleme, de nem teljesen biztos), illetve hogy ez az esély eléri-e az adott utasításra vonatkozó elfogadási küszöböt.

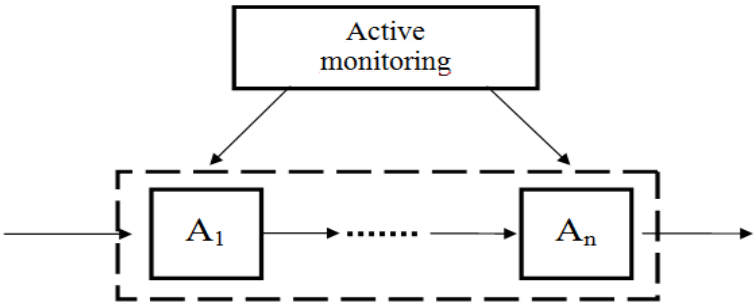
A korábbi fejezetekben ismertetett speciális követelmények és adottságok miatt a téves parancsaktiválások számát a lehetséges minimumra kell szorítani. Figyelembe kell venni a felhasználó esetleges dizartriáját is, tehát elsősorban olyan beszédjellemzőkre célszerű támaszkodni, amelyeket kevésbé érintenek a betegség tünetei. Ezzel együtt olyan módszer kidolgozása a cél, amely kellően robusztus ahhoz, hogy akár bizonyos mennyiségű téves fonémaosztályozást is elviseljen.

Nyilvánvaló követelmény, hogy egy parancs kimondása után a rendszer rövid időn belül (körülbelül 2-3 másodperc) reagáljon. Ha ugyanis ez nem teljesül, akkor fennáll a veszély, hogy a felhasználó türelmetlen lesz, és megismétli az utasítást. Ez nem kívánt parancsvégrehajtáshoz vezethet, így tehát igyekezni kell elkerülni. Eszerint egy időfüggő hasznossági függvény (U(t)) rendelhető a beszédfelismerő kimenetéhez, amelyet úgy választunk meg, hogy a t növekedésével az értéke csökkenjen, és a 2-3 másodperces időkorlát elteltével érje el a nullát, mivel ennél később már nem lesz hasznos az eredmény a rendszer vezérlésében (2.1. ábra). Emellett a helyes osztályozás esélye tekinthető az eredmény minőségének.



2.1. ábra: Időfüggő hasznosság függvényre példa

A fentiek alapján felvázolhatunk egy anytime beszédfelismerő algoritmust, amely anytime elemző modulok (A_1 .. A_n) lineáris kompozíciójából és egy aktív monitorozó egységből áll, utóbbi a modulok láncolatának a futtatását ütemezi, és szükség esetén megszakíthatja vagy átrendezheti azokat (2.2. ábra).



2.2. ábra: Moduláris anytime rendszer

Mindegyik modul kimenete egy becslés arra vonatkozóan, hogy a bemenetként kapott beszédminta mekkora eséllyel felel meg egy-egy rendelkezésre álló parancskifejezésnek (azaz tulajdonképpen 0 és 1 közötti számok vektoráról van szó). Az egyes modulok a beszéd különböző tulajdonságait számítják ki, és ennek alapján módosítják minden parancskifejezésre a találat valószínűségét. A gyakorlatban ez úgy játszódik le, hogy a legvalószínűbb eredmény közvetlen keresése helyett szisztematikusan kizárjuk a legkevésbé valószínűeket. A különböző beszédjellemzők szerinti elemzés addig

futtatható, amíg (még a hasznossági időkorláton belül) kellő biztonsággal ki nem emelkedik egyetlen utasítás a halmazból.

2.5.3 Lényeges anytime elemző modulok ismertetése

Számos környezeti és beszédparaméter van, amely még zajos környezetben is meglehetősen pontosan felismerhető. Az alábbiakban példaként szerepel néhány elemző modul működési elve, amelyek ezeket számítják ki a parancsok valószínűségeinek meghatározására. Számos ilyen modul implementálható iteratív algoritmussal, például a beszédfelismerésben alapvető fontosságú Fourier transzformációnak is léteznek megszakítható implementációi. Ilyenek az Anytime Fast Fourier Transformation (AnFFT) [59] vagy egy korábbi megoldás, a DFT-IR (Discrete Fourier Transformation – Incremental Refinement) [60].

2.5.3.1 Menü hierarchia modul

A rendszer parancskifejezéseit menüstruktúrába szervezzük, amely elősegíti az emlékezetből történő használatot. A főmenüből a felhasználó a különböző almenükbe navigálhat (ágy vezérlése, televízió, rádió, lámpa stb.), ahol az adott csoportba tartozó utasítások érhetők el. Ezen kívül vannak olyan utasítások is, amelyek a menüstruktúra bármely pontján elérhetők (vészhívás, nővérhívás, kilépés).

Tekintsük például a rádióvezérlő almenüt, amelybe akkor lép be a felhasználó, ha rádiót szeretne hallgatni. Belépés után elérhetővé válnak a csatornaváltó és hangerőszabályzó utasítások. Látható, hogy az aktuális pozíció a menüben erősen meghatározza az utasítások elhangzásának valószínűségét: nem túl nagy az esélye (ámbár nem is nulla), hogy a rádió menüben a felhasználó az ágya magasságát szeretné állítani. Ezért „nulladik feldolgozási lépésként” a menü alapján beállíthatjuk az utasítás valószínűségek kezdőértékeit a következő függvény szerint:

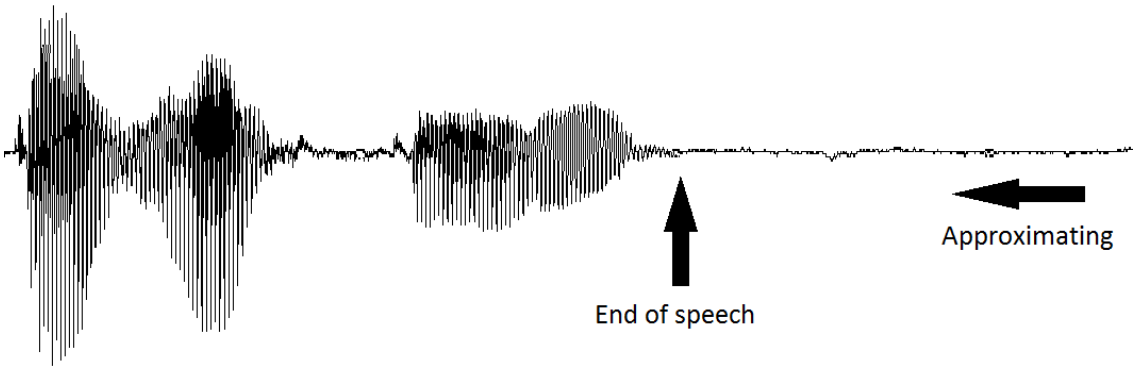
$$P1(c) = \begin{cases} 1, \text{ha } c \text{ elérhető az aktuális almenüben} \\ 0.1, \text{egyébként.} \end{cases} \quad (5)$$

ahol a c jelenti az utasítást, P1(c) pedig az adott c utasításnak a valószínűsége az aktuális menü pozícióban.

2.5.3.2 Utasításhossz modul

Egy kifejezés kimondásának ideje természetesen alkalomról alkalomra változik, de ez jól megállapítható határok között marad. Minden kifejezésre létezik egy minimális idő, amely mindenképp szükséges a felhasználó számára a kimondáshoz, fizikailag képtelenség rövidebb idő alatt kiejteni a szavakat. A kimondás maximális idejére is adható egy jó becslés, amelynél hosszabbra nem fog elnyúlni (a magánhangzók szándékos elnyújtásával nyilván bármilyen hosszúra lehet „énekelni” egy szót, de a felhasználónak nem célja megtéveszteni a rendszert). Ezen alsó és felső korlátoknak megfelelően szintén számos utasítás kizárható, vagy a valószínűsége alacsonyra állítható.

A rögzített beszéd hosszának megállapítása megoldható anytime módon. Ha a felvételt a beszéd kezdete aktiválja, és ezután fix hosszúságú felvétel történik, akkor a cél a beszéd végének a meghatározása. Ez iteratív módon közelíthető úgy, hogy a felvett hangnak valamely beszéddetektálásra alkalmas jellemzőjét kiszámítjuk (időtartományban például a nullátmenetek száma, autokorreláció vagy AMDF), és a vektor végétől elindulva addig haladunk visszafelé, amíg a hangmintában beszédet nem detektálunk. Az algoritmus pillanatnyi kimenete a felvétel elejétől az aktuálisan vizsgált pozícióig tartó időtartam, tehát minél tovább futtatjuk az algoritmust, annál pontosabb értéket fog adni a beszéd hosszára egészen addig, amíg a tényleges értéket meg nem találja (2.3. ábra).



2.3. ábra: A beszéd hosszának közelítése az időtartománybeli jelen szemléltetve

A beszéd hosszát úgy közelítjük, hogy az AMDF-ben visszafelé haladva beszédet keresünk. A természetes beszédben előforduló tempóbeli eltéréseket a valószínűség becslésénél figyelembe kell venni. Ennek megfelelően a c utasítás valószínűsége az alábbi függvény szerint adható meg:

$$P2(c) = \begin{cases} 1, & \text{if } T - v_1 \leq t \leq T + v_1 \\ 0, & \text{if } T - v_1 - v_2 > t \text{ or } T + v_1 + v_2 < t \\ 1 - \frac{|T-t|-v_1}{v_2}, & \text{egyébként} \end{cases} \quad (6)$$

ahol T jelenti a c átlagos hosszát, t a bemeneti beszédminta hosszát, v1 és v2 pedig előre definiált tolerancia zónát a T körül.

2.5.3.3 Magánhangzó elemző modul

A magánhangzók olyan, önmagukban kiejthető beszédhangok, amikor a levegő nagyrészt akadálytalanul távozik a tüdőből, megrezegtetve a hangszálakat. Ezen fonémák esetén megállapíthatunk rezonancia frekvenciákat, amelyeket a vokális traktus határoz meg, ezeket formánsoknak nevezzük. A formánsok csúcsokként jelennek meg a hang spektrumában az egyes magánhangzókra jellemző, jól azonosítható módon, így az első három formáns (F1, F2 és F3) kiszámításával lehetővé teszik a magánhangzók formáns alapú osztályozását (sőt formánsszintetizálási eljárással működik a mesterséges beszéd-előállítás is).

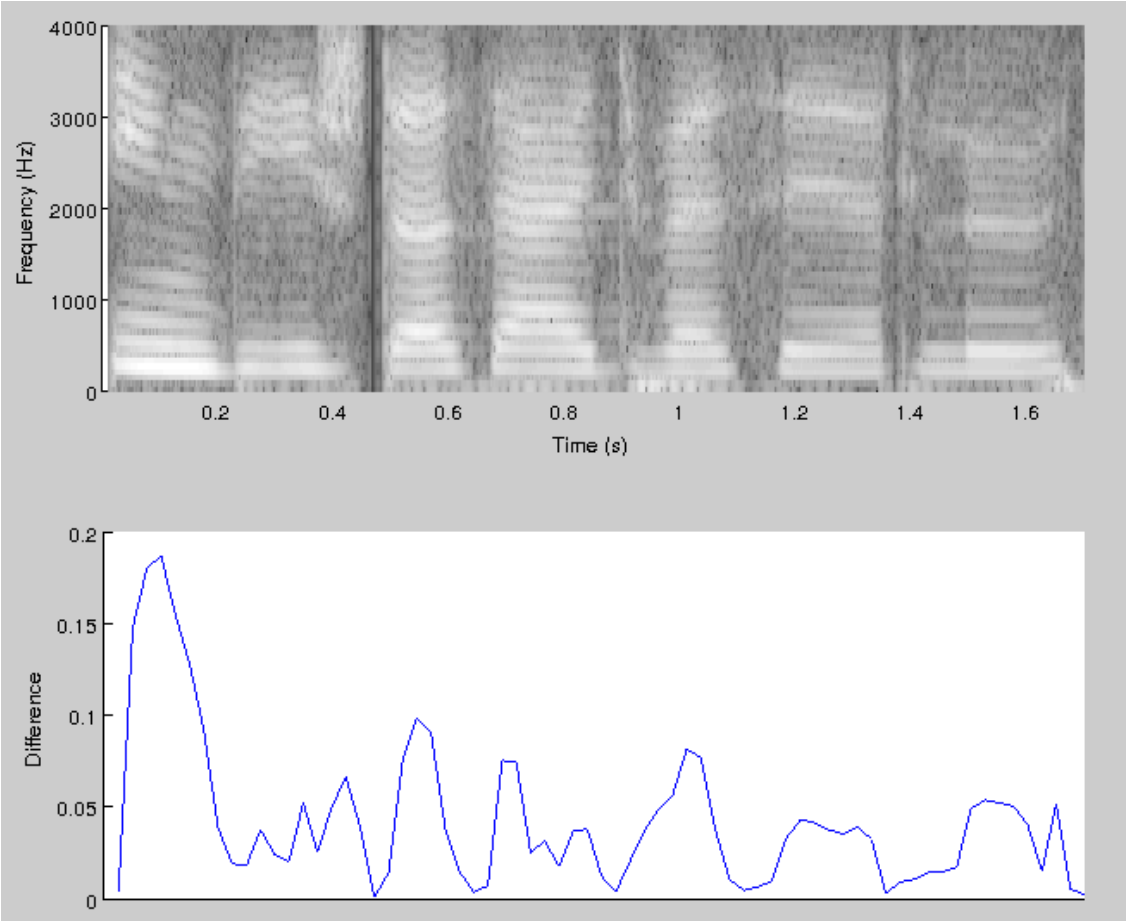
Korlátozott számú, megfelelően megválogatott utasításhalmazból nagy biztonsággal ki tudjuk választani a megfelelőt a beszédmintában található magánhangzók azonosításával [58].

A magyar mássalhangzókhoz megállapíthatók átlagos formáns frekvenciák (2.1. táblázat), azonban a gyakorlatban az egyes magánhangzókhoz inkább frekvencia tartományok rendelhetők, amelyek ráadásul egyénenként eltérőek, és bizonyos mássalhangzók esetében átfedőek is.

2.1. táblázat: a magyar mássalhangzók átlagos formáns frekvenciái [58]

	F1 (Hz)	F2 (Hz)	F3 (Hz)
a	631	1415	2985
á	799	1750	2877
e	594	1997	2941
é	397	2500	3045
i	279	2596	3211
í	287	2655	3123
o, ó	437	1129	3010
ö, ő	389	1867	2750

Az eddigiekhez képest a formánsok meghatározása meglehetősen számításintenzív, tehát a bemenetként kapott hangfelvétel minden egyes szegmensére kiszámítani nem gazdaságos, ha csak a magánhangzókra vagyunk kíváncsiak. Mivel a gyorsan számítható AMDF függvény megmutatja a beszéd zöngés fonémáit (azaz a magánhangzókat és a zöngés mássalhangzókat), ezért célszerű csak ezen pozíciók körül számítani a formánsokat. A 2.4. ábrán látható ugyanazon beszédminta ("Indítsd a rádiót") spektrogramja és AMDF függvénye, mindkettő x tengelye az idő. Megfigyelhető a spektrogram magasabb intenzitású részeinek és az AMDF csúcsainak egybeesése. A magánhangzók és a mássalhangzók megkülönböztethetők a formánsaik alapján. Ezzel a módszerrel a számítási idő jelentősen lecsökkenthető, azonban előfordulhat, hogy esetleg egy-egy magánhangzót nem sikerül detektálni, ha az túl halk, vagy összeolvad egy másik hanggal.



2.4. ábra: Beszéd spektrogramja és AMDF-e ("Indítsd a rádiót" utasítás)

A magánhangzók kiejtése is egyénenként és alkalmanként eltérő lehet (ez bármilyen beszédhangra is igaz), kifejezetten igaz ez a dizartriával élő felhasználókra. Számukra szükséges egy olyan személyre szabott modell megalkotása, amely amellett, hogy reprezentálja a felhasználó aktuális kiejtését, a további használat során majd adaptálható is a beszéd változásához. Lehetséges, hogy a felhasználó annyira hasonlóan ejt ki két különböző mássalhangzót, hogy azokat nem lehet megbízhatóan megkülönböztetni. (Ez ép beszédű személyeknél is előfordul néha, például az é í-nek, az ó ú-nak hallatszik bizonyos szituációkban, és csak az agyunk korrigálja a nyelv lehetséges szavainak ismeretében és a szövegkörnyezet alapján.)

A fentiek alapján egy adott v magánhangzó előfordulásának valószínűsége a c magánhangzó-listájának megfelelő pozíciójában:

$$P3(c) = \begin{cases} 1, & \text{ha } v \text{ a megfelelő pozícióban van } c-\text{ben} \\ 0.8, & \text{ha a } c \text{ megfelelő pozíciójában hasonló mgh. van} \\ 0.5, & \text{ha } v \text{ egy szomszédos pozícióban van } c-\text{ben} \\ 0.2, & \text{egyébként} \end{cases} \quad (7)$$

A pozíció értelmezése itt a következő: c magánhangzó-listája az a karaktersorozat, amelyet a c-ből a mássalhangzók eltávolításával nyerünk. Ez a lista a c magánhangzóit tartalmazza abban a sorrendben, ahogyan a c-ben előfordulnak. Egy magánhangzó pozíciója a c magánhangzó-listájában az elem sorszáma a listában (hanyadik magánhangzó a c-ben).

Minden megtalált magánhangzóval csökken a helytelen megoldások valószínűség értéke, míg ideálisan csak egyetlen megoldás marad az elfogadási küszöb fölött.

2.5.4 Tesztesetek leírása

A következőkben két konkrét példával demonstrálom a rendszer működését.

Tesztkörnyezet

A teszteket relatíve csendes környezetben futtattam (a háttérzaj 50 dB-nél kisebb). A teszteléshez használt hangminták rögzítését, szerkesztését, vizualizálását és elemzését is a Matlab szoftverrel, a Signal Processing Toolbox függvényeinek felhasználásával végeztem.

A tesztkörnyezethez a mintát egy éles üzemben lévő okosotthon rendszer adta, amelynek későbbi továbbfejlesztése, illetve újratervezése az itt ismertetett eredmények alapján egy jövőbeli célom. A rendszer egy súlyosan mozgássérült, alig észrevehető mértékben beszédfogyatékos, szklerózis multiplexben szenvedő hölgy lakásában üzemelt sok éven át, egészen a közelmúltig.

A szakdolgozat projektből kinőtt rendszer egy közönséges PC-n fut, a vezérlőszoftver tartalmazza a beszédfelismerő megoldást, a beérkező utasítások szerint hívja a külső szoftveres szolgáltatásokat (Skype, netrádió), és USB porton keresztül csatlakoztatott vezérlőegységgel működteti a különféle hardveres kiegészítőket (módosított távirányító a televízióhoz és az ágyhoz, nővérhívó csengő stb.). Ezen funkcióknak megfelelő az

utasításkészlete is, amelyet még a legelső üzembe állításkor a szakdolgozó hallgatók állítottak össze (emiatt nem feltétlenül ideális a célra, de az újratervezésig nem akartam módosítani). A tesztekhez az eredeti utasításkészletet használtam, de nyilvánvalóan nem az éles üzemben lévő gépen futtattam őket.

Az utasításkészlet 36 darab magyar nyelvű utasításból áll, amelyek a következők:

(1) vészhívás
(2) kilépés
(3) hívd a nővért
(4) lámpa
(5) mondd a pontos időt
(6) megerősítés
(7) indítsd a Skype-ot
(8) indítsd a rádiót
(9) indítsd el a tévét
(10) indítsd az ágyvezérlőt
(11) következő
(12) előző
(13) következő partner
(14) előző partner
(15) hívás indítása
(16) vedd fel a telefont
(17) telefon vége
(18) program újraindítása
(19) kilépés a Skype-ból
(20) továbblépés
(21) következő csatorna
(22) előző csatorna
(23) emeld a hangerőt
(24) csökkentsd a hangerőt
(25) rádió megállítása
(26) kilépés a rádióból
(27) megnyitás
(28) kapcsold a tévét
(29) állítsd az időzítőt
(30) kilépés a tévéből
(31) emeld a fejem
(32) süllyeszd a fejem
(33) emeld a lábam
(34) süllyeszd a lábam
(35) kilépés az ágyvezérlőből
(36) mondd a hőmérsékletet

Az utasítások fastruktúrába rendeződnek, a főmenüből 4 almenü nyílik. Néhány utasítás több almenüben is elérhető, 4 utasítás pedig mindig elérhető.

- Főmenü parancsai: 4, 5, 6, 7, 8, 9, 10, 36
- Skype almenü parancsai: 11, 12, 13, 14, 15, 16, 17, 19, 20
- Televízió almenü parancsai: 21, 22, 23, 24, 28, 29, 30
- Rádió almenü parancsai: 21, 22, 23, 24, 25, 26, 27
- Ágy almenü parancsai: 31, 32, 33, 34, 35
- Mindig elérhető: 1, 2, 3, 18

1. példa:

Az 1. példában azt mutatom be, hogyan befolyásolja az eredményt, ha egy magánhangzót az osztályozás során összetévesztünk egy másik, hasonló magánhangzóval. A magánhangzók osztályozásához szándékosan egy primitív, vektorok távolságán alapuló módszert használtam, hogy a hasonló magánhangzók téves besorolásának hatását teszteljem.

Az elhangzott utasítás a „vészhívás”, de a magánhangzó-felismerő modul az „é” hangot helytelenül „í”-nek azonosítja. (Lásd a 2.2. táblázatot) A teszteset kiindulási menüpozíciója a Televízió almenü. A hangmintában a beszéd hossza 1,225 másodpercnek adódik.

3-féle modult használunk, jelölje az „A” a menü szerint szűkítő modult, „B” a kimondott kifejezés hosszát megállapító modult, „C” pedig a magánhangzó kereső modult. A modulokat A, B, C, C, … sorrendben futtatjuk, a C-t a szükséges darabszámszor újra futtatva, egy újabb magánhangzó meghatározására. Minden modul a saját számításai alapján korrigálja a parancshalmaz elemeinek valószínűség értékeit (megszorozva a korábbi értéket a saját maga által számítottal), de csak azokkal foglalkozik, amelyek még a küszöb fölött vannak, tehát még nem kerültek kizárásra a korábbi modulok által.

Az elemzés addig fut, amíg az utasítások közül már csak legfeljebb egynek lesz 0,5 fölötti a valószínűsége. (A konkrét küszöb környezettől és beszélőtől függően állítható, de tipikusan a helyes utasítás valószínűsége 1-hez közeli marad, míg a többi gyorsan

lecsökken.) A 0,2 alá csökkent utasításokra már nem frissítjük tovább a valószínűséget, ezek már kizártnak tekinthetők.

Az A modul végrehajtása után 11 esélyes utasítás marad a 36-ból, de még egyiket sem zártuk ki. A B modul ezt 6-ra csökkenti a hossz vizsgálata alapján. Az első magánhangzó (helytelenül) „í”-nek adódik, ez kizár további 3 utasítást, de a helyes megoldás még esélyes. A következő két mássalhangzó kiértékelésével a rendszer helyesen azonosítja a kimondott utasítást, az összes végrehajtási idő 14,8 ms. (A számítás részeredményei a 2.2. számú táblázatban találhatók.)

2.2. táblázat: a „vészhívás” utasítás felismerésének részeredményei

Parancs sorszáma	**Frissített valószínűségek a modul futtatása után**				
	A modul: Menü	***B modul: Hossz***	***C modul: 1. mgh: ‘í’***	***C modul: 2. mgh: ‘í’***	***C modul: 3. mgh: ‘á’***
1	**1**	**1**	**0,8**	**0,8**	**0,8**
2	**1**	**1**	**0,8**	**0,64**	0,128
3	**1**	**1**	**1**	0,2	
4	0,1	0,1			
5	0,1	0			
6	0,1	0,08			
7	0,1	0,07			
8	0,1	0,06			
9	0,1	0,1			
10	0,1	0			
11	0,1	0,1			
12	0,1	0,1			
13	0,1	0,1			
14	0,1	0,1			
15	0,1	0,08			
16	0,1	0,1			
17	0,1	0			
18	**1**	0			
19	0,1	0			
20	0,1	0,1			
21	**1**	0			
22	**1**	**0,6**	0,12		
23	**1**	0			
24	**1**	**0,7**	0,14		
25	0,1	0			
26	0,1	0			
27	0,1	0,1			

Parancs sorszáma	Frissített valószínűségek a modul futtatása után				
	A modul: Menü	*B modul: Hossz*	*C modul: 1. mgh: 'í'*	*C modul: 2. mgh: 'í'*	*C modul: 3. mgh: 'á'*
28	**1**	**1**	0		
29	**1**	0			
30	**1**	0			
31	0,1	0,1			
32	0,1	0,1			
33	0,1	0,1			
34	0,1	0,07			
35	0,1	0			
36	0,1	0,1			

2. példa:

A 2. példa egy olyan esetet mutat be, amikor nem is szükséges az összes magánhangzó azonosítása ahhoz, hogy nagy biztonsággal kiválaszthassuk a korrekt utasítást. Ebben az esetben a Skype almenüből indulunk, 1,3 másodperc hosszúságú kifejezést detektálunk. A 2.3. számú táblázat mutatja a részeredményeket (ezúttal csak az almenü szerint relevánsak szerepelnek). A végrehajtás ideje 15,9 ms volt.

2.3. táblázat: A „továbblépés" utasítás felismerésének részeredményei

Parancs sorszáma	Frissített valószínűségek a modul futtatása után			
	A modul: Menü	*B modul: Hossz*	*C modul: 1. mgh: 'o'*	*C modul: 2. mgh: 'á'*
1	**1**	**1**	0,2	
2	**1**	**1**	0,2	
3	**1**	**1**	**0,5**	0,4
11	**1**	**1**	0,2	
12	**1**	**1**	0,2	
13	**1**	**1**	0,2	
14	**1**	**1**	0,2	
15	**1**	**1**	0,2	
16	**1**	**1**	0,2	
17	**1**	0		
18	**1**	0		
19	**1**	0		
20	**1**	**1**	**1**	**1**

2.5.5 Összegzés

A tesztek futtatása során elsősorban arra voltam kíváncsi, hogy magas hibaarányú fonémafelismerés mellett hogyan teljesít az algoritmusom. Ezért szándékosan két primitív fonémaosztályozási eljárást használtam: a sablon alapú mintaillesztés önmagában 36,7%-os PER (Phoneme Error Rate) értéket produkált, a K-közép klaszterezés pedig 26,3%-osat. Az első módszerrel a rendszer a parancsok 88,5%-át helyesen, 3,5%-át tévesen ismerte fel, 8% pedig fals negatív eredményt hozott, ám többnyire ekkor is a helyes parancshoz tartozott a legmagasabb valószínűség érték. A második módszerrel 91,9% helyes, 2,7% téves, és 5,4% fals negatív találat született.

Az irodalomban található értékekkel összevetve elmondható, hogy a célkitűzésemet sikerült teljesíteni, a kapott eredmények megközelítik (némely megoldásokkal összehasonlítva meg is haladják) az egészséges beszédre tervezett parancsfelismerők megbízhatóságát.

Összességében elmondható, hogy az anytime megoldás megbízható felismerést biztosít korlátozott erőforrások mellett is. Ugyanakkor bőséges erőforrásmennyiség rendelkezésre állása esetén a felvázolt algoritmus arra is alkalmas, hogy további szűrő és elemző módszerek alkalmazásával nem ideális akusztikus környezetben robusztusabb, pontosabb felismerést biztosítson a környezethez igazodva (tehát nem minden esetben növeli meg a feldolgozási időt, csak amennyiben azt szükségesnek ítéli).

2.6 Működés közbeni adaptáció lehetősége

A neuromuszkuláris betegségeknek a beszédre gyakorolt negatív hatása komoly kihívás elé állítja a mozgássérülteknek szánt beszédvezérelt rendszerek tervezőit. Egy intelligens otthon parancsszavakkal vezérelve óriási segítséget jelenthet egy súlyosan mozgássérült személy életében, azonban a fokozatosan eltorzuló beszédet idővel egyre kevésbé fogja felismerni a beszédfelismerő modul. Egy beszélőfüggetlen modell esetén nyilvánvaló a feladat kilátástalansága, de még egy beszélőfüggő, a beteg aktuális beszédére tanított modell hatékonysága is idővel leromlik. Egy súlyosan mozgássérült személy esetében viszont nem engedhető meg a hibás felismerésből adódó téves

végrehajtás, hiszen az intelligens otthon rendszertől független, manuális korrekció nem lehetséges, vagy esetleg csak segítő személyek igénybevételével. Az ideális megoldás a problémára egy olyan beszédfelismerő, amely képes követni a beszéd lassú, de folyamatos romlását, amíg csak lehetséges. Ehhez lehetőleg ne igényeljen rendszeres időközönkénti újratanítást, de a használat közben begyűjtött esetlegesen zajos, vagy mások beszédét is tartalmazó mintákkal való tanítás ne zavarja össze.

A célom megfogalmazni egy olyan beszédfelismerő algoritmus elveit, amely képes igazodni a felhasználó idővel változó beszédjellemzőihez, mindezt az éles használat közben teszi, de úgy, hogy az adaptáció során a beszédfelismerő hatékonysága ne csökkenjen.

2.6.1 Több felismerő modulból álló beszédfelismerő rendszerek

Számos szabadalom és néhány tudományos cikk is található olyan beszédfelismerő rendszerekről, amelyek több felismerő modult tartalmaznak, azonban ezeknek a célja a felismerés pontosságának a növelése. Az egyes felismerő modulok azonos típusúak, csak működési paramétereikben különböznek. Ugyanazon beszédmintát megkapva bemenetként, az egyedi kimeneteik egy kombinált kimenetet adnak valamilyen döntési folyamat eredményeképp [61, 62].

Olyan példákat is találhatunk a szakirodalomban, ahol az osztályozás felgyorsítása a cél, és ehhez különböző típusú neurális hálókat alkalmaznak (például egyet az előfeldolgozásra, egy másikat a pontos felismerésre) [63]. A kombinált felismerő működhet fonéma, szótag vagy akár teljes szavak szintjén is [64, 65, 66].

2.6.2 Adaptív módszerek beszédfelismerési feladatokra

A beszédfelismerésben széles körben használatosak különféle adaptív eljárások különféle célokra, ilyenek például a zaj kompenzáció [67, 68], egy adott akcentushoz való igazítás [69], vagy hasznos tanítóminták kiválasztása címkézésre egy nagyobb címkézetlen korpuszból [70]. Maga az adaptálódó rendszer jellemzően valamilyen jól ismert lágy számítási módszer, mint a neurális háló [71] vagy fuzzy filter [68].

A folyamatos, használat közbeni adaptáció azonban veszélyeztetheti a felismerés pontosságát, különösen mivel az átlagos élettérben megtalálható háttérzajok emberi beszédet is tartalmazhatnak, akár más személyek beszéde, akár a televízió vagy rádió műsorainak formájában.

2.6.3 Adaptív beszédfelismerés több felismerő kombinálásával

A fentebb ismertetett problémákból (dizartria tünetei, működési biztonság szintje a súlyos mozgásképtelenség miatt) adódnak bizonyos speciális követelmények a folyamatosan adaptálódó beszédfelismerő modellekre vonatkozóan. Ezeket a követelményeket, valamint a rendszer működésére vonatkozó szabályokat és döntéshozó módszereket a következőképpen fogalmaztam meg:

- **Lassan változó beszédjellemzők követése**
  A beszédfelismerő modell legyen képes az egyre romló minőségű beszédet követni, és ameddig csak lehet, pontosan osztályozni.
- **Online (használat közbeni) adaptáció**
  Az intelligens otthon rendszer folyamatos használatban van, ezért a lassan változó beszédhez való adaptáció lehetőleg a normál működés megszakítása nélkül kell végbe menjen.
- **A felismerés pontossága ne csökkenjen**
  Az adaptációs eljárás legyen robusztus zajos környezetben, tehát a zajos tanítómintáknak ne legyen negatív hatása a pontosságra. Azonban átmenetileg csökkenhet a pontosság a felhasználó beszédének a megváltozása miatt.
- **Hibás osztályozás detektálása**
  A rendszernek detektálnia kell, ha egy parancs felismerése téves volt. Ez nem triviális, mivel a felhasználónak erősen korlátozottak a lehetőségei a hibás működés korrigálásában. Célszerű, ha lehetőség van a hiba szóbeli jelzésére (valamilyen speciális, biztosan felismerhető parancskifejezés formájában) vagy bármilyen olyan módon, amelyre a felhasználó képes.

A fenti követelményeket teljesíteni lehet olyan módon, hogy egyetlen felismerő helyett többet használunk. Több felismerő kombinációja ugyanis felhasználható a hibás

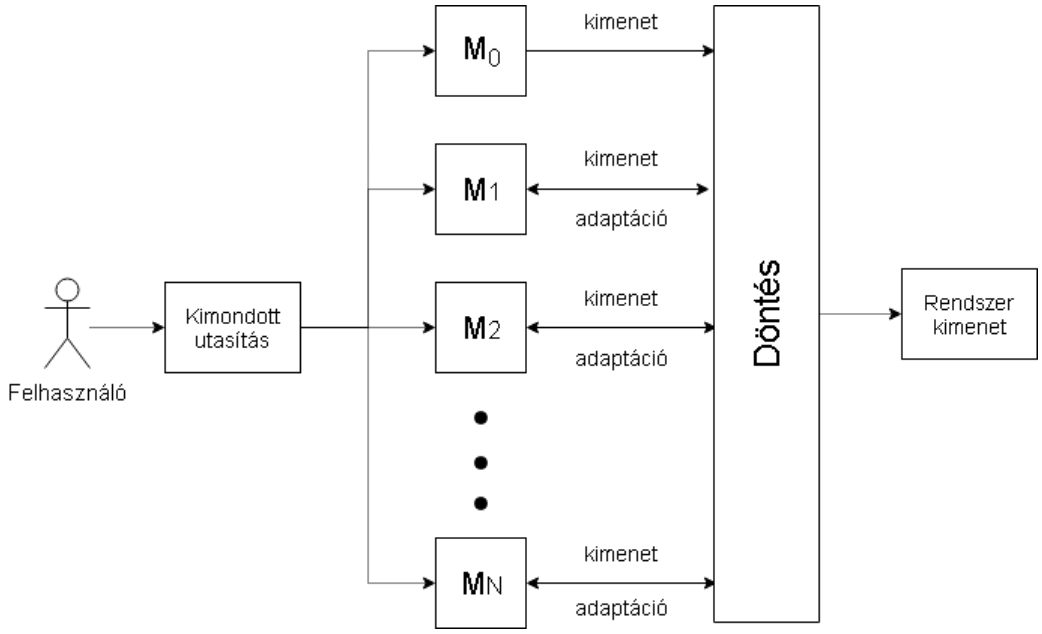
adaptációból fakadó téves osztályozás esélyének csökkentésére. Egy-egy felismerő modul lehet bármilyen adaptálható modellt tartalmazó megközelítés. Egy ilyen összetett felismerő rendszer felépítésénél a következőket kell figyelembe venni:

- A felismerő modulok legyenek egymástól függetlenek, tehát az egyik hibás működése ne legyen hatással a többire.
- A különféle adaptációs stratégiák (és/vagy tanító minták) használata hasznos lehet a modellek közötti diverzitás megtartására, és a zajtűrés elősegítésére.
- Az elsődleges modell változatlansága biztosítja, hogy a tanítás során a teljes rendszer pontossága nem csökken. A pontosság növelése megoldható az elsődleges modell lecserélésével, ha az adaptálódó modellek között előáll egy pontosabb példány.
- Az adaptálódó felismerők különböző súllyal vehetnek részt a rendszer kimenetének előállításában. (Szélsőséges esetben akár 0 súllyal is, ilyenkor mindig csak az elsődleges modell kimenetét vesszük figyelembe.)

A fenti megfontolások szerint egy konkrét megoldást építve, az alábbi szabályokkal rendelkező keretrendszer lehetővé teszi egy több felismerőből álló adaptív rendszer megvalósítását:

1. A rendelkezésre álló felismerők közül kiválasztjuk a legpontosabbat, ezt az elsődleges modellt kizárjuk az adaptációból, változatlan marad a működés során.
2. Egy adaptálódó modellből akkor lesz elsődleges modell, ha a hibaaránya egy előre meghatározott darabszámú felismerés alatt kisebb, mint az elsődlegesé. Ebben az esetben ez lesz az új elsődleges modell, valamint ennek egy másolata tovább adaptálódik.
3. Egy adaptálódó modell akkor törlődik, ha a hibaaránya egy előre meghatározott darabszámú felismerés alatt egy adott küszöbértéknél magasabb.
4. Az adaptálódó modellek darabszáma egy konstans értéken tartható olyan módon, hogy a legrosszabb modell(eke)t töröljük, illetve a legjobb modellből másolatot készítünk.

5. Ha a modelleket különböző stratégiák szerint tanítjuk (például az egyik minden bejövő inputot tanul, egy másik csak a hibásan osztályozottakat stb.), akkor az összes modell egyidejű pontosság-csökkenésének kicsi az esélye.



2.5. ábra: A rendszer felépítése

2.6.4 Döntéshozó módszerek

Számos módszer használható több különálló felismerő kimenetének kombinálására. A legegyszerűbb módszer a közönséges többségi szavazás, ami azt jelenti, hogy az az érték lesz a végső kimenet, amelyet a legtöbb modell választotta. Ez akkor használható, ha az összes résztvevő jósága ugyanakkorának feltételezhető, vagy ha nincs mód a résztvevők jóságának megállapítására. A jelen esetben azonban nem ez a helyzet, az egyes modellek jósága megállapítható a korábbi kimenetek helyessége alapján. Tehát célszerű az egyes modellek kimeneteinek figyelembevételét súlyozni az adott modell megbízhatósága szerint.

Tegyük fel, hogy *M* darab lehetséges kimenete van a rendszernek (o_1 ... o_M), definiáljuk az $X_i(o_j)$ függvényt a következőképpen:

$$X_i(o_j) = \begin{cases} 1, \text{ha az } \boldsymbol{i} - \text{edik felismerő az } \boldsymbol{o_j} \text{kimenetet adja} \\ 0, \text{egyébként.} \end{cases} \quad (8)$$

ahol $i = 0 \ldots N$ a felismerő sorszáma olyan módon, hogy $i = 0$ az elsődleges felismerő, $i = 1 \ldots N$ pedig az adaptálódó felismerők sorszáma; $j = 1 \ldots M$ a kimenetek sorszámait jelöli, M a kimenetek összes darabszáma.

Ekkor minden lehetséges o_j kimenetre kiszámítható egy S_j pontszám:

$$S_j = \sum_{i=0}^{N} X_i(o_j)\, r(i) \quad (9)$$

ahol $r(i)$ az i-edik felismerő megbízhatóságával arányos súlytényező, N pedig az összes adaptálódó felismerő száma.

Lehetőség van olyan üzemmódban működtetni a rendszert, hogy csak az elsődleges felismerő kimenetét vesszük figyelembe. Ebben az esetben a további felismerők csak a működés közbeni biztonságos adaptáció célját szolgálják, a tényleges felismerésben nem vesznek részt egészen addig, amíg elsődlegessé nem válnak. Ilyenkor a súlytényezők értékei a következőképpen alakulnak:

$$r(i) = \begin{cases} 1, ha\ i = 0 \\ 0, egyébként \end{cases} \quad (10)$$

Az egyszerű többségi szavazásos üzemmód esetén pedig az

$$r(i) = \frac{1}{N+1} \quad (11)$$

minden $i = 0 \ldots N$ felismerőhöz tartozó súlytényezőre.

A rendszer kombinált kimenete meghatározható a maximális S_j meghatározásával. Miután a kimenet kiszámításra került, az adaptálódó modelleket a tanulási stratégiáiknak megfelelően tanítani kell, valamint a hibaarányukat frissíteni. Ezután meg kell vizsgálni a fenti szabályok szerint, van-e törlendő modell, vagy le kell-e cserélni az elsődleges modellt.

Felvetődik a kérdés, hogy működés közben hogyan detektáljuk a téves osztályozást. Ha a keretrendszer által produkált kimenet helyes volt, akkor az ettől eltérő eredményt számító felismerők osztályoztak tévesen. Ha azonban a keretrendszer kimenete téves volt, akkor a felhasználó visszajelzése alapján meg kell állapítani a helyes parancsot, és eszerint leellenőrizni az egyes felismerők kimeneteit, valamint az adaptálódó felismerők esetében a tanítást is eszerint kell elvégezni.

2.6.5 Összegzés

Bár a keretrendszer hosszú távon rendkívüli mértékben növelheti a beszédfelismerés eredményességét újratanítás nélkül, az implementálása nagy körültekintést igényel, mivel hibás beállítások esetén a keretrendszer nem biztosítja a meglévő pontosság megtartását, és zajos (azaz valóságos) környezetben ez használhatatlanná teheti a beszédfelismerő modult.

2.7 Kapcsolódó tézisek

2. téziscsoport: beszédfelismerés robusztussága intelligens otthon rendszerekben

2.1 tézis

Dizartriában szenvedő felhasználók számára készült beszédvezérléshez megalkottam egy szövegekre értelmezett távolság mértéket, amely dizartria-specifikus távolságot állapít meg két utasítás között beszédvezérelt intelligens otthon rendszerekben való használat szempontjából [S5, S6].

2.2 tézis

Kidolgoztam egy anytime megközelítést a beszédvezérelt intelligens otthon rendszerek beszédfelismerő algoritmusának felépítésére a robusztus működés elősegítése céljából [S7, S8, S9, S10, S11].

2.3 tézis

Megfogalmaztam egy olyan, több beszédfelismerő modulból álló keretrendszernek az elveit, amelynek célja beszédvezérelt intelligens otthon rendszerek beszédmodelljének működés közbeni folyamatos adaptációja olyan módon, hogy a felismerési pontosság eközben ne csökkenjen [S12, S13, S14].

3 Összefoglalás

A beszéd- és mozgásfogyatékossággal élő személyek helyzete számos okból rendkívül nehéz, a betegségükkel való megküzdésen túl is. A (lehetőségekhez képest) önálló életvitel kialakítása nagy mennyiségű emberi, anyagi és technológiai erőforrást igényel, amelyek az esetek nagy részében csak szűkösen állnak rendelkezésre. A társadalom átlagos anyagi helyzetű tagjai számára elérhető technológiai megoldások többnyire nem is illeszkednek megfelelően a betegek igényeihez.

A kutatómunkám célja olyan eljárások és módszerek kidolgozása volt, amelyek mind nyilvános térben, mind otthon képesek támogatni a beszéd- és mozgásfogyatékossággal élő személyeket a napi tevékenységeik elvégzésében olyan módon, hogy minél kevesebb emberi segítséget kelljen igénybe venniük. Az eredményeim két nagyobb csoportra bonthatók: a középületekben használható, valamint az intelligens otthon rendszerek kapcsán alkalmazható megoldásokra.

A publikus térben talán a legnagyobb gondot az akadálymentes közlekedés okozza. Míg a kültéri navigációs alkalmazásoknak léteznek akadálymentes útvonaltervezést támogató változatai, addig beltérben, a középületekben az akadálymentes útvonalak megtalálása nem megoldott. A megoldási lehetőségek körét erősen megköti, hogy ezekre az épületekre ritkán tud a fenntartó nagyobb mennyiségű anyagi erőforrást rááldozni. Éppen ezért fordulnak elő szép számmal részben akadálymentes, esetleg egyáltalán nem akadálymentes középületek, és éppen ezért nem jöhetett szóba speciális hardvereszközök felszerelésén alapuló módszer. Az általam tervezett ontológia, és ennek átültetése címkézett tulajdonsággráf adatbázisba, lehetővé teszi a kiegészítő hardver nélküli beltéri akadálymentes navigációt, amelynek paraméterei részletesen a betegek igényeire és képességeire szabhatók.

Az intelligens otthon rendszerek, és az ezekben használható beszédvezérlés irodalma bőséges, jól használható megoldások állnak már rendelkezésre. Ezek azonban nem veszik figyelembe azokat a speciális kritériumokat, amelyeket a súlyos mozgás-, és beszédfogyatékosság támaszt. Egy tipikus beszédvezérelt okosotthon rendszer nem képes eléggé biztonságosan és robusztusan működni egy olyan személy otthonában, aki

nem képes a parancsszavakat tisztán kimondani, és főleg nem képes az esetlegesen tévesen végrehajtott utasításokat néhány mozdulattal korrigálni. Ezen a területen olyan eljárások és módszerek fejlesztésére koncentráltam, amelyek a meglévő, átlagos felhasználó esetében jól működő megoldásokat egészítik ki, illetve szervezik át olyan módon, hogy a szükséges biztonsági kritériumoknak megfelelő működést biztosítsanak. A beszédfelismerésre koncentrálva, a dizartria jellemző tüneteit figyelembe véve javasoltam módszereket a betegség során folyamatosan romló minőségű beszéd robusztusabb felismerésére.

3.1 Tézisek összefoglalása

1. téziscsoport: akadálymentes beltéri navigáció

1.1 tézis

Egy meglévő, épületek belső szerkezetét leíró ontológiához kidolgoztam egy olyan kiterjesztést, amely lehetővé teszi az adott ontológia szerint leírt épületekben a beltéri akadálymentes navigáció implementálását különféle fokú mozgássérültség szintek speciális igényeihez igazodva [S1, S2].

1.2 tézis

Kidolgoztam egy megfeleltetést, amely lehetővé teszi a fenti ontológia szerint leírt adatok transzformálását gráfadatbázisban való tároláshoz, és a gráfadatbázison olyan lekérdezések futtatását, amely mozgássérült felhasználók részletesen megadott igényeinek megfelelő útvonalat talál az épület két pontja között, amennyiben ilyen létezik [S3, S4].

2. téziscsoport: beszédfelismerés robusztussága intelligens otthon rendszerekben

2.1 tézis

Dizartriában szenvedő felhasználók számára készült beszédvezérléshez megalkottam egy szövegekre értelmezett távolság mértéket, amely dizartria-specifikus távolságot állapít meg két utasítás között beszédvezérelt intelligens otthon rendszerekben való használat szempontjából [S5, S6].

2.2 tézis

Kidolgoztam egy anytime megközelítést a beszédvezérelt intelligens otthon rendszerek beszédfelismerő algoritmusának felépítésére a robusztus működés elősegítése céljából [S7, S8, S9, S10, S11].

2.3 tézis

Megfogalmaztam egy olyan, több beszédfelismerő modulból álló keretrendszernek az elveit, amelynek célja beszédvezérelt intelligens otthon rendszerek beszédmodelljének működés közbeni folyamatos adaptációja olyan módon, hogy a felismerési pontosság eközben ne csökkenjen [S12, S13, S14].

3.2 Az eredmények alkalmazásáról

Ez a fejezet az értekezésben bemutatott eljárások, modellek és módszerek alkalmazhatóságát tárgyalja, figyelembe véve a technológia és a társadalom jelenlegi, széles körben tapasztalható helyzetét.

A munkám során igyekeztem mindig az aktuálisan fennálló körülményekből és lehetőségekből kiindulni, nem pedig egy ideális, illetve jövőbeli helyzethez tervezni. Ez számos korlátozó tényezőt jelentett, amelyek megnehezítették a megoldások kialakítását, de az azonnali gyakorlati alkalmazást ez a szemlélet jelentősen megkönnyíti.

3.2.1 A beltéri akadálymentes navigáció implementálása valós környezetben

A beltéri akadálymentes navigáció gráfadatbázisra épülő megvalósítása (ugyanúgy, mint a korábbi, RDF store-ban tárolt verzió) megköveteli az épület szerkezetének, helyiségeinek, valamint azok akadálymentességi tulajdonságainak leírását, és a későbbiekben természetesen ezen adathalmaznak a karbantartását, aktualizálását is. Ez a feladat emberi erőforrást igényel, a teljes automatizálás lehetősége ugyanis egyelőre nem látszik reálisnak.

Az adatok beviteléhez mindenképpen érdemes egy grafikus felülettel rendelkező programot adni, amely megkönnyíti a bevitelt végző személy dolgát. Hiszen egy adminisztrátori jellegű feladatokat ellátó dolgozó végezné tipikusan ezt a munkát, akitől nem várható el a Neo4j adatbeszúró szintaktikájának, vagy egyéb strukturált leírónyelvnek (amit a gráfadatbázis importálni képes, például a JSON vagy CSV) az ismerete. Ez a program, azon túl, hogy képes az egyszerű és áttekinthető felületen bevitt adatokból a megfelelő formátumú fájlokat legenerálni, szolgálhatna egyéb kényelmi funkciókkal is, amelyek az épületek elrendezésének a tipikusan ismétlődő jellegére alapozhatnak. Jellegzetes példa erre egy többemeletes épületen belül, hogy a mosdók praktikus okokból minden emeleten ugyanott, tehát egymás felett helyezkednek el, és a férfi – női – mozgássérült mosdók egymáshoz képesti elrendezése is ugyanaz. A szobák felszereléseinek a tulajdonságai is jellemzően megegyeznek, mint például az ajtó szélessége, a villanykapcsoló magassága stb. Ezeket elegendő lenne egyszer bevinni, és a program gondoskodna róla, hogy minden helyiséghez automatikusan társításra kerüljenek.

Az útvonaltervező hasznosságának jelentős komponensét adja a felhasználói felület használhatósága. Tipikusan mozgássérült felhasználók esetében ez nem csak a program képernyőjének és az adatbevitelnek a logikus kialakítását, és a minél gyorsabb és intuitívabb használatra való törekedést jelenti. Az általános ergonómiai megfontolások mellett legalább ilyen fontos a felhasználói felületnek a mozgássérült felhasználók számára akadálymentes kialakítása is. A Web Content Accessibility Guidelines[16] (WCAG) jelenleg a globálisan elfogadott szabvány az akadálymentes weboldalak és szoftverek tervezésére és implementálására. Léteznek emellett egyéb szempontrendszerek és segédanyagok is, de ezek értelemszerűen mind ugyanazokat az alapelveket fogalmazzák meg. A mozgássérült felhasználók szempontjából ezek az alapelvek a következők:

[16] https://www.w3.org/TR/WCAG21/

- A felhasználói felület működtethető legyen speciális beviteli módszerekkel, azaz
    - csak billentyűzettel, ha valaki nem képes a standard mutatóeszközt eléggé precízen kezelni; ebben az esetben gondoskodni kell róla, hogy a felület minden input funkciót ellátó eleme billentyűzettel fókuszálható legyen, azaz TAB billentyűvel rá lehessen állítani a fókuszt és szintén billentyűk segítségével aktiválható legyen;
    - speciális mutatóeszközzel, mint amilyen a PC-hez csatlakoztatható fejegér, lábegér vagy joystick, érintőképernyős használatnál pedig valamely mozgásképes testrészhez rögzített érintőpálca; ebben az esetben az akadálymentes használat egyik alapköve a felület kattintható elemeinek megfelelő mérete.
- A felhasználói felület biztosítson elegendő időt a bevitel elvégzésére, tehát automatikus váltások vagy (látszólagos) inaktivitás miatti kiléptetés ne, vagy csak hosszabbítható időkorláttal forduljon elő.
- A felhasználói felület minél kevesebb interakciót igényeljen a keresés adatainak megadásához, ennek több módja is van:
    - a kezdő és célállomás begépelését váltsuk ki valamilyen egyéb megoldással, például legördülőből való kiválasztással vagy a kezdőpont automatikus meghatározásával (célszerűen elhelyezett QR kódok vagy NFC címkék leolvasásával ez olcsón kivitelezhető [S16]);
    - a több oldalnyi elemet tartalmazó listák (így a legördülők is) legyenek egy-két billentyűleütéssel szűrhetők, hogy ne legyen szükség hosszas görgetésre a kiválasztásnál;
    - a felhasználó akadálymentességi preferenciáit ne kelljen újra és újra megadni, az legyen eltárolható egy felhasználói profilban, vagy akár tanulja meg automatikusan a szoftver.

Fontos megjegyzés a fentiekhez, hogy az akadálymentes szoftvertervezés egyik alapvető feltevése, hogy a felhasználónak van valamilyen módszere arra, hogy a számítógép vagy táblagép vagy mobiltelefon bemeneti perifériáit kezelni és kimeneti perifériáit érzékelni tudja. A mozgássérült felhasználók esetén ez nyilván a bemenet működtetését jelenti. A gépi beszédfelismeréssel működő diktálás abban az esetben

lehet hasznos segédeszköz a gépelés kiváltására, ha a felhasználónak nincs vagy enyhe mértékű a beszédfogyatékossága, ezért erre ne alapozzunk.

3.2.2 Okosotthon rendszerek kialakítása beszéd- és mozgássérült személyek számára

Az általam alkotott módszerek leginkább az okosotthon rendszerek tervezésében játszanak szerepet, már üzemben lévő megoldás megbízhatóbbá alakítására korlátozottan, vagy legalábbis jelentős átalakítások mellett alkalmasak. Jól kiegészítik a jelenleg bevett, megszokott megoldásokat, esetleg egybe is vágnak ezekkel. Ez utóbbira jó példa az okosotthon moduljainak a szabványok felületeken keresztül való csatlakoztatására vonatkozó biztonsági kritérium.

Amiről mindenképp szót kell ejtenem az üzemeltetői tapasztalataim kapcsán, az a terméktámogatás és karbantartás kérdése. Egy súlyosan mozgássérült felhasználó otthonában ez egy 24 órában működő rendszer, amely, ha megáll, akkor megáll az élet is. A felhasználó semmit sem fog tudni csinálni, amíg a hibát el nem hárítja valaki. Gondoljunk csak arra, hogy mi történik, ha a beszédfelismerés céljából felszerelt mikrofonban lemerül az elem, vagy a vezérlő számítógép operációs rendszerében történt váratlan frissítés miatt újraindítás válik szükségessé.

Ezért ezeknek a rendszereknek a tervezésénél két kritikus üzemeltetési szempont van, amit mindenféleképpen figyelembe kell venni. Az egyik, hogy a hardver jellegű problémákat (amennyire ez lehetséges) a segítők a helyszínen el tudják végezni. Tehát a rendszer összeszerelésénél figyelni kell arra, hogy a hozzátartozó vagy a nővér is könnyedén tudjon elemet cserélni, esetleg egy kihúzódott kábelt újra összedugni.

A másik pedig, hogy a szoftver jellegű problémák orvoslása viszont csak a legritkább esetben várható el a helyszínen lévő személyektől, ezért (nyilván megfelelő autentikációval működő, védett csatornán keresztüli) távoli felügyeleti lehetőséget kell biztosítani a vezérlő számítógéphez. Ezen keresztül az informatikus akár éjjel vagy utazás közben is, néhány perc alatt elvégezhet egy diagnosztikát vagy újraindítást.

Ha a felhasználó állapota lehetővé teszi, akkor a biztonság kedvéért adjunk alternatív segítségkérő csatornát az üzemeltető felé. Erre egy működő megoldás, hogy az okosotthon rendszeren keresztül elérhető internetes kapcsolattartáson kívül a felhasználó mobiltelefonon is tudjon üzenni, ha a rendszer valamiért működésképtelen állapotba került volna.

Különösen a rendszer éles üzembe állítása utáni időszakban rendkívül hasznos lehet egy tesztkörnyezet felépítése, amely a legfontosabb elemeiben (de a vezérlőszámítógép hardverében és operációs rendszerében, a vezérlőszoftverben és a beszédfelismerő modulban mindenképpen) megegyezik a helyszínen üzemelő rendszerrel. Váratlan, nehezen magyarázható hiba előfordulása esetén ez lehetővé teszi (legalább is nagyban elősegíti) a jelenség reprodukálását, javítását, és a javított verzió előzetes tesztelését. Az informatikusnak a helyszínre utazása és az ottani munkavégzése ugyanis sokszor körülményes és kényelmetlen, ezért érdemes ezt elkerülni, ha csak lehetséges.

Fejlesztői szemlélet tekintetében fel kell készülni arra, hogy a felhasználó egyéni képességei, állapota miatt személyre szabott megoldásokra lehet szükség a rendszerrel való interakció kialakításában. Ebbe az is beleértendő, hogy előfordulhat, hogy nem lehetséges minden szempontból tökéletes és kényelmes módot találni erre. Arra azonban többnyire lehet számítani, hogy a felhasználó és a segítői is maximálisan együttműködők lesznek a rendszer használatának elsajátításában, a kisebb kényelmetlenségek áthidalásában, a rendszeres, egyszerűbb karbantartási feladatok elvégzésében. Hiszen egy jól megtervezett rendszer olyan szintű életminőségbeli javulást tud garantálni a segítők jelentős tehermentesítése mellett, hogy a használói érdemesnek tartják majd arra, hogy belefektessék az időt és energiát mind a kezdeti tanulási folyamatba, mind a későbbi esetleges karbantartási feladatokba.

Ugyanakkor arra ügyelni kell, hogy a rendszer nem okozhat több nehézséget, mint amennyit megold, mert akkor már nem képes kielégíteni a felhasználó által támasztott elvárásokat. Erre egy hétköznapi példa, hogy a felhasználó készségesen megtanulja majd (amennyire tőle telik) a rendszer számára érthetően kimondani a parancskifejezéseket, esetleg meg is ismétli egyszer-egyszer, ha szükséges. Ám ha

mindent többször kell elismételni, mire egyszer felismeri a beszédfelismerő modul, az már hosszabb távon nem elfogadható kompromisszum.

3.3 Jövőbeli kutatási irányok

A következőkben azokat a további kutatási lehetőségeket vázolom fel, amelyek az eredményeim továbbfejlesztését és bővítését célozzák.

3.3.1 Beltéri akadálymentes navigáció

A kutatásom kiindulópontja egy beltéri navigációra szánt ontológia volt, ezért a munkám során nem tértem ki a kültéri használat lehetőségeire. Léteznek kültéri, illetve bel- és kültéri navigációra tervezett ontológiák (Geodint, Yang-Worboys), tervezem ezek részletes akadálymentességi fókuszú vizsgálatát.

A földrajzi adatok gráfadatbázisban való tárolására is léteznek megoldások (Neo4j Spatial project[17], Oracle Spatial and Graph[18]), amelyek segítségével a kültéri navigáció megvalósítható.

Az általam fejlesztett ontológia kiegészítés és a gráfadatbázis kisebb mértékű bővítéssel és átalakítással illeszthető egy városon belüli navigációs megoldáshoz is, mivel lehetőséget ad tetszőleges objektumok akadálymentességgel kapcsolatos tulajdonságainak részletes leírására.

3.3.2 Biztonságos használatot elősegítő megoldások okosotthon rendszerben beszéd- és mozgássérült személyek számára

Az általam megalkotott módszerek továbbfejlesztésének legfontosabb eleme egy olyan korpusz lenne, amely számos, elsősorban szklerózis multiplexszel élő beteg beszédét tartalmazná. Olyan mintákra lenne szükség, amelyek között megtalálható mind betanult, mind spontán beszéd, hiszen ezeket különböző módon érinti a dizartria. A munkám során figyelembe vett jelenségek többnyire az orvosi szakirodalomból, illetve hallás

[17] https://neo4j-contrib.github.io/spatial/
[18] https://www.oracle.com/database/technologies/spatialandgraph.html

utáni megfigyelésből származnak, ezért lenne hasznos a lehető legtöbb különböző tünet megfigyelése és mérése, ugyanis a módszereim finomhangolásának kulcsfontosságú eleme a betegek egyedileg jellemző tünetegyütteseihez való alkalmazkodás.

Hosszú távon rendkívül komoly tudományos értékkel bírna egy olyan nagy méretű beszéd-adatbázis, amelyben ugyanazon betegek beszédének változásait lehetne végig követni akár évtizedes távlatokban. Természetesen ezeknek a mintáknak a szakszerű címkézéséhez és kategorizálásához szakorvosi segítség szükséges, hiszen a betegség tüneteinek szakértője tud rámutatni az egyes megfigyelt tünetek tipikus vagy atipikus voltára, illetve a betegség várható lefolyásával összefüggésben a dizartria tüneteinek várható változására is.

Egy ilyen, évtizedeket felölelő korpusz lehetővé tenné annak a vizsgálatát is, hogy a korai beszédtünetek képesek-e előre jelezni a betegség típusát (relapszáló-remittáló, primer progresszív, szekunder progresszív vagy progresszív relapszáló) és hosszú távú prognózisát.

4 Saját publikációk

4.1 Tézisekhez kapcsolódó saját publikációk

[S1] Fleiner, R., Szász, B., Simon-Nagy, G., Micsik, M., "Indoor navigation for motion disabled persons in medical facilities", Acta Polytechnica Hungarica, Vol. 14, No. 1, 2017.

[S2] Simon-Nagy, G., Fleiner, R., "Ontology Extension for Personalized Accessible Indoor Navigation", Advances in Intelligent Systems and Computing 660, 2017, pp. 281-288.

[S3] Chalhoub, N., Simon-Nagy, G., "Indoor Navigation based on Linked Data at Honvéd Hospital, Budapest" In Proc. IEEE 12th International Symposium on Applied Computational Intelligence and Informatics, SACI 2018, Temesvár, Románia: IEEE Romania Section, IEEE Hungary Section, May 17-19, 2018, pp. 485-490.

[S4] Simon-Nagy, G., Chalhoub, N., Fleiner, R., "Graph-based Data for Accessible Indoor Navigation" In Proc. IEEE 23th International Conference on Intelligent Engineering Systems, INES 2019, Gödöllő, Hungary, Apr. 25-27, 2019.

[S5] Simon-Nagy, G., Várkonyi-Kóczy, A.R., "Distance metric for speech commands in smart home systems of dysarthric users", Advances in Intelligent Systems and Computing 519, Recent Global Research and Education: Technological Challenges. Berlin; Heidelberg: Springer Verlag, 2017, pp. 325-330.

[S6] Simon-Nagy, G., Várkonyi-Kóczy, A. R., "Distance metric for speech commands in smart home systems of dysarthric users", In Proc: 15th Int. Conf. on Global Research and Education in Intelligent Systems, Inter-academia’2016, Warsaw, Poland, Sept. 26-28, pp. PS31-1-PS31-6.

[S7] Nagy, G., "An interpretation of the COBIT information criteria to operational criteria of voice controlled Ambient Assisted Living systems", In Proc. 5th IEEE International Symposium on Logistics and Industrial Informatics, LINDI 2013, Wildau, Germany, Sept. 5–7, 2013, pp. 49-53.

[S8] Nagy, G., "Construction of anytime algorithms for robust speech recognition", Advanced Materials Research 1117, 2015, pp. 265-268.

[S9] Nagy, G., "Construction of Anytime Algorithms for Robust Speech Recognition", In Proc. 13th Int. Conf. on Global Research and Education in Intelligent Systems, Interacademia'2014, Riga, Lettország, 2014, pp. 170-171.

[S10] Nagy, G., Várkonyi-Kóczy, A.R., Tóth, J.T., "An Anytime Voice Controlled Ambient Assisted Living System for Motion Disabled Persons", In Proc. IEEE International Symposium on Medical Measurements and Applications, MeMeA'2015, Torino, Italy, May 7-9, 2015, pp. 163-168.

[S11] Várkonyi-Kóczy, A. R., Kiss, D., Nagy, G., Tóth, J., "Anytime Classification", In Proc. 7th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management, IEEE HNICEM - ISCIII 2014, Puerto Princesa, Philippines, Nov. 12-14, 2014. paper 162.1-6.

[S12] Nagy, G., Kutor, L., "Multiple ANN Recognizers for Adaptive Recognition of the Speech of Dysarthric Patients in AAL Systems", In Proc. Advancing Assistive Technology and eAccessibility for People with Disabilities and the Aging Population, AAATE 2015, Budapest, Hungary, Sept. 9-12, 2015, pp. 1013-1016.

[S13] Simon-Nagy, G., Várkonyi-Kóczy, A.R., "Adaptive Speech Recognition Framework for Dysarthric Patients", JJAP Conf. Proc. Vol. 4, 2016, Cikk azonosítója: 011614.

[S14] Nagy, G., Várkonyi-Kóczy, A. R., "Adaptive Speech Recognition Framework for Dysarthric Patients", In Proc. 14th Int. Conf. on Global Research and Education in Intelligent Systems, Interacademia'2015, Hamamatsu, Japán, 2015, pp. 130-131.

4.2 Tézisekhez nem kapcsolódó saját publikációk

[15] Fleiner, R., Simon-Nagy, G., Szász, B., "Accessible Indoor Navigation based on Linked Data in Hospitals", In Proc. IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016, Budapest, Hungary, Oct. 9-12, 2016, pp. 002425-002430.

[16] Kutor, L., Domozi, Zs., Nagy, G., Peller, Cs., Véső, T., "Object Identification and Local Information Services Using Near Field Communication", In Proc. Regional Conference on Embedded and Ambient Systems, RCEAS 2007, Budapest, Hungary, Nov. 22-24, 2007, 91 p.

5 Irodalomjegyzék

[1] The Americans with Disabilities Act of 1990 and Revised ADA Regulations Implementing Title II and Title III, http://www.ada.gov/2010_regs.htm

[2] 253/1997. (XII. 20.) Korm. rendelet az országos településrendezési és építési követelményekről - Akadálymentesítésre vonatkozó kivonat, Mozgáskorlátozottak Egyesületeinek Országos Szövetsége (MEOSZ), www.meosz.hu/doc/p26-2otek_segedlet.doc

[3] Berners-Lee, T., "Linked data - design issues", 2006, http://www.w3.org/DesignIssues/LinkedData.html

[4] Bizer, C., Heath, T., Berners-Lee, T., "Linked Data—The Story So Far", International Journal on Semantic Web and Information Systems 5 (3), 2009, pp. 1–22.

[5] Hitzler, P., Krotzsch, M., Rudolph, S., "Foundations of semantic web technologies", CRC Press, 2009

[6] Hayes, P., "RDF Semantics", 2004, http://www.w3.org/TR/2004/REC-rdf-mt-20040210/

[7] Lassila, O., Swick, R.R., "Resource Description Framework (RDF) Model and Syntax Specification", 1999, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[8] Cyganiak, R. et al., "RDF 1.1 concepts and abstract syntax" W3C recommendation, 25(02), 2014.

[9] Worboys, M., "Modeling Indoor Space", In Proc. 3rd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness, Chicago Illinois, USA, Nov. 2011, pp. 1-6.

[10] Anagnostopoulos, C., et al., "OntoNav: A Semantic Indoor Navigation System", In Proc. 1st Workshop on Semantics in Mobile Environments in conjunction with 6th ACM SIGMOBILE / SIGMOD International Conference on Mobile Data Management, Agua Napa, Cyprus, May 9-13, 2005.

[11] Dudas, P. M., Ghafourian, M., Karimi, H., "ONALIN: Ontology and Algorithm for Indoor Routing", In Proc. Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, MDM'09, Taipei, Taiwan, May 18-21, 2009, pp. 720-725.

[12] Scholz, J., Schabus, S. "An Indoor Navigation Ontology for Production Assets in a Production Environment", In Proc. International Conference on Geographic Information Science, Vienna, Austria, Sept. 24-26, 2014, pp. 204-220.

[13] Matuszka, T., Gombos, G., Kiss, A., "A New Approach for Indoor Navigation Using Semantic Webtechnologies and Augmented Reality", In Proc. Virtual Augmented and Mixed Reality. Designing and Developing Augmented and Virtual Environments, Las Vegas, NV, USA, July 21-26, 2013, pp. 202-210.

[14] Benner, J. G., Karimi, H. A., "Accessible Wayfinding Ontologies for People with Disabilities", Extended Abstract for the RDWG Online Symposium on Accessible Way-Finding Using Web Technologies, Dec. 3, 2014.

[15] Wheelmap.org. How does Wheelmap work, https://news.wheelmap.org/en/faq/

[16] Szász B, Fleiner R, Micsik A., "iLOC - Building In-Door Navigation Services using Linked Data". In Proc. SEMPDS-2016 Posters&Demos@SEMANTiCS 2016 and SuCCESS'16 Workshop. Leipzig, Germany, Sept. 12-15, 2016, pp. 1-4.

[17] Szász, B., Fleiner, R., Micsik, A., "Practical uses of location and event data as Linked Open University Data", In Proc. 1st International Conference and Exhibition on Future RFID Technologies, Eger, Hungary, Nov. 5–7, 2014. pp. 151–159.

[18] Fleiner, R., Szász, B., Piros, P., "Indoor Navigation Linked Data at Óbuda University", In Proc. 11th IEEE International Symposium on Applied Computational Intelligence and Informatics, SACI 2016, Timisoara, Romania, May 12-14, 2016, pp. 25-30.

[19] Janowicz, K., et al., "Five Stars of Linked Data Vocabulary Use". Semantic Web, 5(3), 2014, pp. 173-176.

[20] Szász, B., Fleiner, R., Micsik, A., "iLOC – An Indoor Ontology", 2015, http://lod.nik.uni-obuda.hu/iloc/iloc-20151201.html

[21] Fleiner, R., Simon-Nagy, G., Szász, B., "Accessible Indoor Navigation based on Linked Data in Hospitals", In Proc. IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016, Budapest, Hungary, Oct. 9-12, 2016, pp. 002425-002430.

[22] Szász, B., Fleiner, R., Micsik, A., Simon-Nagy, G., "iLOC – An Indoor Ontology", 2016, http://lod.nik.uni-obuda.hu/iloc/iloc-20161107.owl

[23] Robinson, I., Webber, J., Eifrem, E., "Graph databases: new opportunities for connected data", O'Reilly Media, Inc., 2015.

[24] Vukotic, A. et al, "Neo4j in Action", Manning Publications, 2014.

[25] Harrison, G., "Next Generation Databases: NoSQL and Big Data", Apress, 2015.

[26] Barrasa, J., "RDF Triple Stores vs. Labeled Property Graphs: What's the Difference?", 2017, https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/

[27] Hartig, O., "Reconciliation of RDF* and property graphs", arXiv preprint arXiv:1409.3288., 2014.

[28] Barrasa, J., "Importing RDF data into Neo4j", 2016, https://jbarrasa.com/2016/06/07/importing-rdf-data-into-neo4j/

[29] United Nations Department of Economic and Social Affairs, Population Division, "World population ageing 2009", ESA/P/WP/212, December 2009, pp. 4.

[30] Bloom, D. E., Boersch-Supan, A., McGee, P., Seike, A., "Population aging: facts, challenges, and responses", PGDA Working Paper No. 71, May 2011, pp. 1–6.

[31] Kiernan, M. C. et al., "Amyotrophic lateral sclerosis", The Lancet, Volume 377, Issue 9769, 12–18 March, 2011, pp. 942-955.

[32] Kister, I. et al., "Natural History of Multiple Sclerosis Symptoms", International Journal of MS Care: Fall 2013, Vol. 15, No. 3, pp. 146-156.

[33] Hartelius, L., Runmarker, B., Andersen, O., "Prevalence and characteristics of dysarthria in a multiple-sclerosis incidence cohort: relation to neurological data", Folia Phoniatrica et Logopaedica: Official Organ of the International Association of Logopedics and Phoniatrics (IALP), 2000, pp 160-177.

[34] Tomik, B., Guiloff, R. J., "Dysarthria in amyotrophic lateral sclerosis: A review", Amyotrophic Lateral Sclerosis Vol. 11, 2010, pp. 4-15.

[35] Hartelius, L. et al., "Temporal speech characteristics of individuals with multiple sclerosis and ataxic dysarthria: 'scanning speech' revisited", Folia Phoniatrica et Logopaedica: Official Organ of the International Association of Logopedics and Phoniatrics (IALP), 2000, pp. 28-38.

[36] Rosen, K.M., Goozée, J.V., Murdoch, B.E., "Examining the effects of Multiple Sclerosis on speech production: Does phonetic structure matter?", J. Commun. Disord. 41(1), 2008, pp. 49-69.

[37] Feijó, A.V. et al., "Acoustic analysis of voice in multiple sclerosis patients", J. Voice 18(3), 2004, pp. 341-347.

[38] Kuo, C., Tjaden, K., "Acoustic variation during passage reading for speakers with dysarthria and healthy controls", J. Commun. Disord. 62, 2016, pp. 30-44.

[39] COBIT 4.1 Framework, Management Guidelines, Maturity Models, Copyright © IT Governance Institute, 2007.

[40] Szenes, K., "Enterprise Governance Against Hacking", In Proc. 3rd IEEE International Symposium on Logistics and Industrial Informatics, LINDI 2011, Aug. 25-27, 2011, Budapest, Hungary, pp. 229-233.

[41] Bilenko M. et al., "Adaptive name matching in information integration", IEEE Intell. Syst. 18 (5), 2003, pp. 16-23.

[42] Cohen, W.W., Ravikumar, P., Fienberg, S.E., "A Comparison of String Distance Metrics for Name-Matching Tasks", In Proc. IJCAI-03 Workshop on Information Integration, Acapulco, Mexico, August 9-10,2003, pp. 73-78.

[43] Bard, G. V., "Spelling-error tolerant, order-independent pass-phrases via the Damerau-Levenshtein string-edit distance metric", In Proc. 5th Australasian symposium on ACSW frontiers - Volume 68 (ACSW '07), Darlinghurst, Australia, Jan. 30 - Febr. 2, 2007. pp 117-124.

[44] de Andrade, D. C., Leo, S., Viana, M. L. D. S. Bernkopf, C., "A neural attention model for speech command recognition". CoRR, abs/1808.08929., 2018.

[45] Revathi, A. et al., "Isolated Command Recognition Using MFCC and Clustering Algorithm", SN COMPUT. SCI. 1, 82 (2020). https://doi.org/10.1007/s42979-020-0093-x

[46] Kocsor, A., Tóth, L., "Application of Kernel-Based Feature Space Transformations and Learning Methods to Phoneme Classification", Applied Intelligence 21, 2004, pp 129–142.

[47] Tóth, L., "Phone recognition with hierarchical convolutional deep maxout networks", EURASIP Journal on Audio, Speech, and Music Processing. 2015. 10.1186/s13636-015-0068-3.

[48] Michálek J., Vaněk J., "A Survey of Recent DNN Architectures on the TIMIT Phone Recognition Task". In Proc. Text, Speech, and Dialogue. TSD 2018. Lecture Notes in Computer Science, vol 11107. Springer, Cham. https://doi.org/10.1007/978-3-030-00794-2_47

[49] Gosztolya, G., "Is AdaBoost competitive for phoneme classification?", IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 2014, pp. 61-66.

[50] Zilberstein, S., "Operational Rationality through Compilation of Anytime Algorithms (dissertation)", 1993.

[51] Zilberstein S., "Using Anytime Algorithms in Intelligent Systems", AI Magazine Volume 17 Number 3, 1996.

[52] Russell, S. J., Zilberstein, S., "Composing Real-Time Systems", In Proc. of the Twelfth International Joint Conference on Artificial Intelligence, Sydney, Australia, Aug. 24-30, 1991, pp. 212–217.

[53] Zilberstein, S. et al., "Optimal sequencing of contract algorithms", Annals of Mathematics and Artificial Intelligence 39, 2003, pp. 1-8.

[54] Hansen, E. A., Zilberstein, S., "Monitoring the Progress of Anytime Problem-Solving", In Proc. 13th National Conference on Artificial Intelligence, Portland, Oregon, Aug. 1996, pp. 1229-1234.

[55] George, S., Zukerman, I., Niemann, M., "An Anytime Algorithm for Interpreting Arguments," PRICAI 2004: Trends in Artificial Intelligence Lecture Notes in Computer Science, Vol. 3157, 2004, pp. 311-321.

[56] Menzel, W. "Parsing of Spoken Language under Time Constraints", In Proc. 11th European Conference on Artificial Intelligence, Amsterdam, The Netherlands, August 1994, pp. 560–564.

[57] Suma, S.A., Gurumurthy, K.S., "Novel pitch extraction methods using average magnitude difference function (AMDF) for LPC speech coders in noisy environments", In Proc. of the 2nd International Conference on Signal Processing Systems, ICSPS 2010, Vol. 1, Dalian, China, July 5-7, 2010, pp. 636-640.

[58] Abari, K., Rácz, Zs. Zs., Olaszy, G., "Formant maps in Hungarian vowels - online data inventory for research, and education", In Proc. of the Int. Conf. on Speech and Technology, Interspeech 2011, Florence, Italy, August 27-31, 2011, pp. 1609-1612.

[59] Várkonyi-Kóczy, A. R., "Fast Anytime Fuzzy Fourier Estimation of Multisine Signals", IEEE Trans. on Instrumentation and Measurement, Vol. 58, No 5, May 2009, pp. 1763-1770.

[60] Nawab, S. H. et al. "Approximate Signal Processing", Journal of VLSI Signal Processing 15, 1997, pp. 177–200.

[61] Baker, J. K., "Speech recognition using multiple recognizers (selectively) applied to the same input sample", U.S. Patent 6 122 613, Sept. 19, 2000.

[62] Endo, N. et al., "Speech recognition system having multiple speech recognizers", U.S. Patent 7 228 275, June 5, 2007.

[63] Murveit, H. et al., "Speech recognition system to selectively utilize different speech recognition techniques over multiple speech recognition passes", U.S. Patent 7 058 573, June 6, 2006.

[64] Schwenk, H., Gauvain, J., "Combining Multiple Speech Recognizers using Voting and Language Model Information", In Proc. IEEE International Conference on Speech and Language Processing, ICSLP 2000, Beijing, China, Oct. 16-20, 2000, pp. 915-918.

[65] Natori, S., Nishizaki, H., Sekiguchi, Y., "Japanese Spoken Term Detection Using Syllable Transition Network Derived from Multiple Speech Recognizers' Outputs", In Proc. 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010, Makuhari, Chiba, Japan, Sept. 26-30, 2010, pp. 681-684.

[66] Natori, S. et al., "Spoken Term Detection Using Phoneme Transition Network from Multiple Speech Recognizers' Outputs", Information and Media Technologies 8.2, 2013, pp. 457-466.

[67] Kalinli, O. et al., "Noise Adaptive Training for Robust Automatic Speech Recognition", IEEE Trans. Audio, Speech, and Language Processing, Vol. 18, Issue 8, 2010, pp. 1889-1901.

[68] Juang, C., Lin, C., "Noisy speech processing by recurrently adaptive fuzzy filters" IEEE Trans. Fuzzy Systems, Vol. 9, Issue 1, 2001, pp. 139-152.

[69] Wang, H., Wang, L., Liu, X., "Multi-level adaptive network for accented Mandarin speech recognition", In Proc. 4th IEEE Int. Conf. on Information Science and Technology (ICIST), Shenzhen, China, Apr. 26-28, 2014, pp. 602-605.

[70] Riccardi, G., Hakkani-Tur, D., "Active learning: theory and applications to automatic speech recognition", IEEE Trans. Speech and Audio Processing, Vol. 13, Issue 4, 2005, pp. 504-511.

[71] Xue, S. et al., "Fast Adaptation of Deep Neural Network Based on Discriminant Codes for Speech Recognition", IEEE/ACM Trans. Audio, Speech, and Language Processing, Vol. 22, Issue 12, 2014, pp. 1713-1725.

[72] Németh, G., Olaszy, G. (szerk.), "A magyar beszéd" Akadémiai Kiadó, Budapest, 2010.

6 Rövidítésjegyzék

AAL	Ambient Assisted Living
ADA	Americans with Disabilities Act
ALS	amiotrófiás laterálszklerózis (Lou Gehrig-betegség)
AMDF	Average Magnitude Difference Function
API	Application Programming Interface
ATM	Automated Teller Machine (bankautomata)
COBIT	Control Objectives for Information and Related Technologies
CSV	Comma-Separated Values
FFT	Fast Fourier Transformation
GPS	Global Positioning System
IEEE	Institute of Electrical and Electronics Engineers
IRI	Internationalized Resource Identifier
ISM	Industrial, Scientific and Medical (radio bands)
IT	Information Technology
JSON	JavaScript Object Notation
LOD	Linked Open Data
LPG	Labeled Property Graph
MS	Multiple Sclerosis (szklerózis multiplex)
NFC	Near Field Communication
OTÉK	Országos Településrendezési és Építési Követelmények
OWL	Web Ontology Language (nem elírás, valóban fel van cserélve)
POI	Point of Interest
RDF	Resource Description Framework

RFID	Radio Frequency IDentification (HF RFID: High Frequency RFID)
QR	Quick Response (code)
SBC	Single-Board Computer
SPARQL	SPARQL Protocol and RDF Query Language (rekurzív rövidítés)
SQL	Structured Query Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
XLS	Excel Spreadsheet
XML	Extensible Markup Language