TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Infotehnoloogia eriala

Joel Edenberg

Edelaraudtee tehisintellekt

Bakalaureusetöö (6 EAP)

Juhendaja: prof. M. Koit

Autor: ……. “…….” juuni 2010
Juhendaja……“…….“ juuni 2010

Lubada kaitsmisele
Professor …….. ……“…….“ juuni 2010

TARTU 2010

Sisukord

Sissejuhatus

Töö eesmärgiks on luua tehisintellekti realiseeriv süsteem, mis suudab kasutajaga kirjalikult eesti keeles suhelda ning anda informatsiooni Edelaraudtee sõiduplaanide kohta. Rakendus peaks olema kasutatav nii lokaalselt käivitades kui ka internetis Java veebirakendusena. Kuna tehisintellekti pole õnnestunud luua tänapäevani, leidsin, et pole mõtet hakata programmeerima hetkel veel utoopilise eesmärgi nimel. Seega võtsin enda töö aluseks juba olemasoleva tehisintellekti põhimõtteid realiseeriva süsteemi ning keskendusin selle täiendamisele, saavutamaks seatud eesmärki. Seega on raskuspunktiks sõidugraafikute andmebaasi arusaadavaks tegemine konkreetsele tehisintellekti süsteemile.

Sarnaseid süsteeme on ka varem loodud, näiteks on olemas Reisiagent mille autor on Margus Treumuth. See rakendus suudab samuti suhelda inimesega kirjalikult eesti keeles ning oskab anda infot Tallinnast väljuvate lendude kohta. Treumuthi magistritöö [3] raames on antud süsteemi ka edasi arendatud ning muudetud ta veebipõhiseks. Treumuthi loodud on ka veebis kasutatav süsteem Alfred [4], mis oskab anda informatsiooni hetkel Tartu kinodes mängitavatest filmidest.

Võrreldes Edelaraudtee kodulehel asuva reiside veebiotsinguga (http://www.edel.ee) on võimalik uuest, bakalaureusetöös loodavast süsteemist infot saada kiiremini. Seda muidugi juhul, kui tead täpselt, kust kuhu soovid reisida. Kui edaspidi lisada kõnesünteesi ja kõnetuvastuse moodulid, siis saavad antud rakendust kasutada ka näiteks nägemispuudega inimesed. Samuti on võimalik antud rakendust kasutada lokaalselt ilma Internetti juurdepääsu omamata.

Bakalaureusetöö ülesehituse võib jagada kaheks osaks – esialgse tehisintellekti-süsteemi täiendamine ning andmete süsteemile arusaadavaks muutmine. Esimeses peatükis tutvustan lühidalt aluseks valitud tehisintellektisüsteemi ja selle tööpõhimõtteid. Teises peatükis annan ülevaate sellest, kuidas tuli seda süsteemi täiendada ning muuta, et täita seatud eesmärgid. Kolmandas peatükis kirjeldan rakendust, mille abil viisin andmed tehisintellektisüsteemile arusaadavale kujule, ning seejärel räägin tekkinud probleemidest ja võimalustest antud süsteemi edasi arendada.

Töö lisades on toodud:

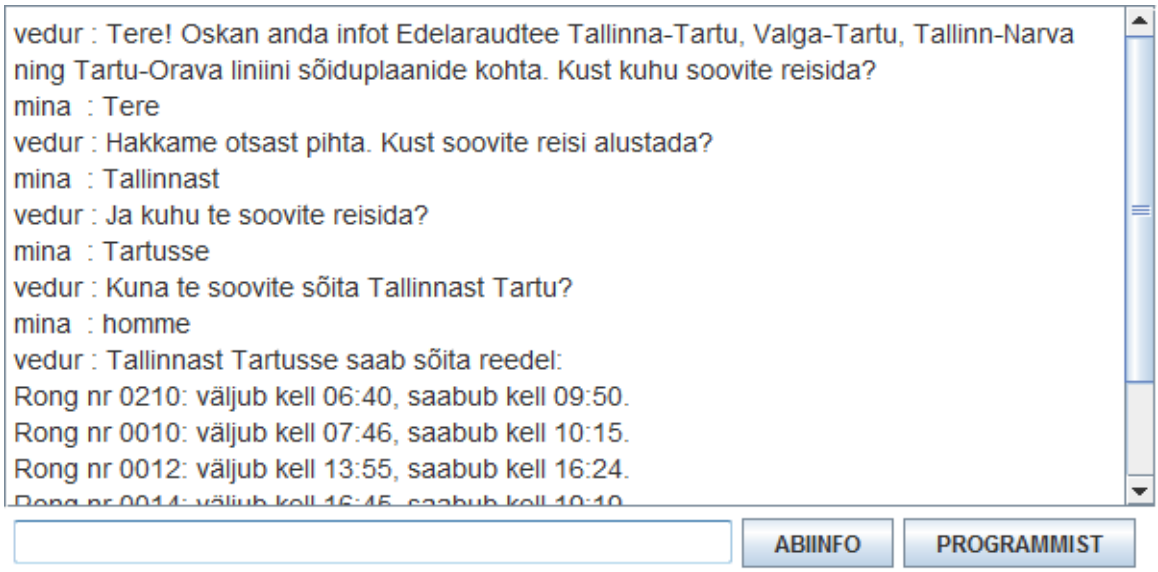
- väljavõte reeglite failist,
- lähteandmete fail init.dat,
- näide kohanimede failist,
- näide sõiduplaanide failist.

Tööga on kaasas CD, millel asuvad:

- kasutusjuhend,
- tarkvara lähtekood,
- kompileeritud kood,
- valmis genereeritud andmebaas ning html fail veebirakenduse kasutamiseks,
- Bakalaureusetöö kirjalik osa.

1. Programmi tutvustus

Töö aluspõhjaks on valitud Eliza-nimeline programm [1], mis on kirjutatud Java keeles (http://java.sun.com). Valisin just selle programmi, kuna Eliza oli suhteliselt hästi kommenteeritud ning tööpõhimõte lahti seletatud. Antud rakendus ei oska teostada sõnede morfoloogilist, süntaktilist ega semantilist analüüsi ja seega peab tuginema puhtalt võtmesõnede otsimisele sisendist. Kuid bakalaureusetöö ülesande lahendamiseks sobis selline lähenemine, sest tegu on kitsalt piiritletud ainevaldkonnaga (joonis 1).



Joonis 1. Pilt töötavast rakendusest.

Esialgset Eliza tehisintellekti kirjeldas Joseph Weizenbaum juba 1966 aastal. See oli esimene programm, mis proovis suhelda kasutajaga loomulikus (inglise) keeles. Eliza tööpõhimõte seisneb mustrite äratundmises ning neile sobilike vastuste andmises.
Aluspõhjana kasutatud Eliza programm on kirjutatud Charles Haydeni poolt ning see on otsene tõlgendus esimesest Eliza kirjeldusest. Programmi koodi on bakalaureusetöös selguse huvides ümber grupeeritud, kuid oma olemuselt on ta muutmata.

Kogu Eliza tegevust saab juhtida ühe sisendfaili abil, kuhu on salvestatud kogu tööks vajalik informatsioon. Programmi käivitamisel loetakse see fail sisse ning alustatakse kasutajaga suhtlemist. Iga lause, mille kasutaja sisestab, loetakse sisse eraldi, seejärel töödeldakse ning formuleeritakse vastus.

Sisendi töötlemine koosneb järgmistest sammudest:

1. Lause tükeldatakse eraldi sõnedeks tühikute abil. Igasugune edasine töötlemine toimub sõnede kui tervikute peal.
2. Toimub sõnede asendamine sisendis.
3. Luuakse loend kõigist lauses leiduvatest võtmesõnedest ning sorteeritakse see võtmesõnede kaalude kahanevas järjestuses.
4. Suurima kaaluga võtmesõnele otsitakse „lahtilammutamise“ reeglit. Esimene reegel, mis mustriga kattub, võetakse kasutusele. Kui ükski muster ei sobi, siis valitakse järgmine, madalama kaaluga võtmesõna.

5. Valitakse esimene vastuse kokkuliitmise muster vastavalt lahtilammutamise reeglile. Kui kokkuliitmise mustreid on antud mitu ning järgnev lause kasutab sama lahti-lammutamise reeglit, valitakse järgmine kokkuliitmise reegel.
6. Toimub sõnede asendamine vastuses.
7. Vastus kuvatakse kasutajale.

Elizale etteantud sisendfaili kasutatakse selleks, et luua asendus- ja võtmesõnade loendid ning lahtilammutamise ja kokkuliitmise reeglid.
Sisendfaili read on rea alguses asuvate märgendsõnade abil jagatud kategooriasse, mis määravad ära, millisest loendist info pärit on:

initial – lause, mille programm väljastab esialgsel käivitamisel.
final – lause, mille programm väljastab töö lõpetamisel.
pre – osa asendusloendist, mis teostatakse enne sisendi töötlemist. Kui esimene sõna loendist esineb sisendlauses, siis asendatakse see ülejäänud sõnedega. Seda tehakse, enne kui hakatakse lauset töötlema.
post – osa asendusloendist, mis teostatakse enne vastuse tagastamist. Kui esimene sõna esineb vastuslauses, siis asendatakse see ülejäänud sõnedega. Seda tehakse vahetult enne vastuse tagastamist kasutajale.
key – võtmesõne. Suurema kaaluga võtmesõned on prioriteetsemad kui väiksema kaaluga. Kui kaalu pole ette antud, oletatakse, et selleks on 1.
decomp – sisendi lahtilammutamise reegel. Sümbol * tähendab suvalist tähtede jada.
reasmb – vastuse kokkuliitmise reegel. Tähtede hulka, mis lahtilammutamise reeglis märgiti tärniga, saab kasutada vastuse osana. Näiteks (2) sisestab tähed mis vastasid teisele tärnile lahtilammutamise reeglis.
synon – sünonüümide loetelu. Laialilammutamise reeglis saab märkida sõnesid märgi @ abil. Kui leidub sünonüüm märgitud sõnele, siis vastavad reeglile ka kõik selle sõna sünonüümid.

Näide laialilammutamise ning kokkuliitmise reeglist:

```
key: tere 5
  decomp: *
    reasmb: Tere ise ka!
```

Antud reegleid kasutatakse, kui sisendis leidus võtmesõna “tere”, mis oma kaaluga 5 oli lauses leiduv kõrgeim võtmesõna. Laialilammutamise reegel mingeid piiranguid ei sea, seega kasutatakse vastavat kokkuliitmise reeglit ning tagastatakse “Tere ise ka!”. Ühe mahukama reegli näide on toodud lisas 1.

Rakendusel on töötamiseks kaks võimalust - andmebaas võidakse sisse lugeda nii lokaalsest asukohast kui ka internetist. Tänu sellele saab antud programmi kasutada ka Java veebi-rakendusena.

2. Eliza täiendamine

Funktsionaalsuse poolelt kõige olulisema muudatusena tuli esialgsele koodile lisada konteksti salvestamise võime. See on realiseeritud nii, et programm jätab meelde rongi lähte- ja sihtkoha, kui talle need ette antud on. Alles siis, kui sisestada uus lähtepunkti nimi, puhastatakse mälu ning salvestatakse uus nimi. Selline lähenemine on vajalik selleks, et kasutaja saaks teostada päringuid erinevate päevade kohta samal liinil.
Lisaks, kuna lähteandmebaas oli suhteliselt suur, tuli ka sisendfaili genereerida võrdlemisi palju reegleid. Seoses sellega muutus fail, mille Eliza peab käivitamisel sisse laadima, umbes 4MB suuruseks. Kahjuks aga Java veebirakendused ei käivitu enne, kui kogu vajalik informatsioon on olemas. Seega tekkis olukord, kus aeglasema internetiühenduse puhul kuvati kasutajale tühja ekraani peaaegu 10 sekundit. Et vältida kasutajas tekkida võivat segadust, on lisatud avaekraan, kus palutakse kasutajal oodata (joonis 2). Tegemist on eraldi rakendusega, mis käivitub eraldi lõimel, kuid taustaks laaditakse põhirakendust. Kui kõik andmed on alla laaditud, suletakse lõim ning juhtimine antakse üle Eliza põhirakendusele.

Joonis 2. Pilt rakendusest andmebaasi allalaadimise ajal.

Kasutaja abistamiseks on lisatud ka kaks nuppu teksti sisestusvälja kõrvale, kust saab abi ning lisainformatsiooni programmi kohta.

Ka märksõnad „täna“ ning „homme“ püütakse Java poolt kinni ning asendatakse hetke nädalapäevaga. See võimaldab andmebaasil olla täielikult staatiline, kuid informatsiooni saab kasutaja küsida ka dünaamiliste parameetrite abil.

Java abil struktureeritakse ka rongide väljumisgraafikuid sisaldav väljund, et hõlbustada lugemist. Joonise 1 alumises servas on seda ka näha.

3. Andmebaasi genereerimine Elizale

Kuna Edelaraudtee sõidugraafikud on Internetis esitatud ainult tabelitena pdf failis [2], oli tarvis kirjutada programm, mis suudaks genereerida etteantud andmete põhjal Elizale sobiliku sisendfaili koos kõigi vajalike reeglitega. Valisin selle jaoks programmeerimiskeele Python. Kuna tegemist on väga kõrgetaseme keelega, on seal sõnetöötluseks väga head vahendid.

Seega on loodud programm, mis võtab lähteandmed kolmest failist – init.dat, plan.kohanimed.dat ja loc.kohanimed.dat.

Töökäik programmis on järgmine:

1. Loetakse sisse andmed init.dat failist ning kopeeritakse need tulemusfaili. Genereeritakse kohanimede sünonüümid (võimalike trükivigadega, alaleütlevas ning sisseütlevas käändes). Init.dat faili sisu on toodud lisas 2.
2. Genereeritakse loendid võimalikest kohanimedest. Lähtepunktid seestütlevas ja alaltütlevas käändes ning sihtpunktid nimetavas käändes.
3. Genereeritakse loend sõidugraafikust, kus elementideks on peatuste ning väljumise aegade paarid.
4. Igast peatusest lähtudes kontrollitakse, kas järgmistesse peatustesse saab sõita igal nädalapäeval, ning lisatakse kõik vastavad reeglid koos vastustega lõpptulemuse faili.
5. Iga erineva etteantud faili plan.kohanimed.dat ja loc.kohanimed.dat korral korratakse tsüklit algusest peale uuesti, kuni läbi on vaadatud kõik lähte-andmefailid.

Et lihtsustada programmiga suhtlust, on lisatud moodul, genereerimaks vigaseid kirjapilte kohanimedest. Need genereeritud nimed lisatakse „synon“ märgendi all sisendfaili, mida Eliza kasutab. Seega isegi siis, kui kasutaja teeb väikese kirjavea, vahetades ära mõned tähed nimes, saab programm aru, mida kasutaja mõtles. Iga nime kohta genereeritakse „tähtede arv sõnes – 2“ erinevat versiooni (näiteks: tartu, tatru, tratu).

Andmed loetakse sisse kahest failist:

- Failis **plan.kohanimed.dat** hoitakse sõiduplaanide infot. Iga liini kohta käiv info on kirjutatud eraldi reale. Rida on jagatud osadeks, kasutades „|“ märgendit. Rida algab liininumbriga. Peale esimest eraldusmärki loetletakse nädalapäevad, millal rong liigub antud liinil (E – esmaspäev, T – teisipäev jne). Seejärel tulevad asukohtade ning kellaaegade paarid, millal rong nendes peatuspunktides asub. Need paarid on omavahel eraldatud hüüumärgi abil. Sihtpunktidel, kus rong peatust ei tee, kuid mis asuvad marsruudil, märgitakse kellaajaks „-“.

  Näide andmetest:
  `0222|ETKNRLP|tallinn!16:05|Ülemiste!16:15|vesse!-|lagedi!16:30`

  Antud rida tähendab seda, et liin 0222 sõidab igal nädalapäeval, väljub Tallinnast kell 16:05 ning jõuab Lagedile kell 16:30. Rong peatub ka Ülemistes kell 16:15. Vesses peatust ei ole. Näidis sõiduliinide andmefailist on lisas 4.

- Failis **loc.kohanimed.dat** hoitakse sõiduliinil asuvaid kohanimesid. Igal real asub erinev kohanimi ning need on sellisel kujul, et lisades lõppu „sse“ või „lt“, saadakse nimest sisseütlev või alaleütlev kääne. Ebareeglipäraste käänamiste jaoks saab peale tühikut samale reale lisada ka juba moodustatud käändevormid kohanimedest, mis loetakse hiljem otse sisse ning lisatakse vastava kohanime sünonüümide alla.

  Näide kohanimedest:
  ```
  Kadrina
  Rakvere Rakverre
  ```

  Väljavõte kohanimede andmefailist on lisas 3.

Pythoni koodile antakse ette loend andmefailidest, mida tulemi genereerimisel kasutada. See loend võib olla kuitahes pikk. Failinimed väärtustatakse loendi elementideks. Hetkel kasutatavad lähtefailide loendid on järgmised:

```
planFiles = ["plan.valga.tartu.dat", "plan.tartu.orava.dat",
"plan.tallinn.tartu.dat", "plan.tallinn.narva.dat"]

locationFiles = ["loc.valga.tartu.dat", "loc.tartu.orava.dat",
"loc.tallinn.tartu.dat", "loc.tallinn.narva.dat"]
```

Seega vaadatakse andmebaasi loomisel läbi 4 sõiduliini. Näited nende failide sisust leiate lisadest 3 ja 4.
Tulemusfaili nimeks on „script“ ning see luuakse samasse kataloogi, kus asub Pythoni kood.

4. Kasutusjuhend

Kuna nii andmebaasi genereerimine kui ka programmi uuesti kompileerimine nõuavad kirjutamise õiguseid, on soovitatav kogu CD sisu kopeerida kuhugi kohalikku kataloogi.
Uus andmebaas on vaja genereerida ainult juhul, kui soovite lisada või muuta andmeid. Andmebaas on genereeritud kujul kaasas ning asub CD juurkataloogis. Andmebaasi faili nimi on „script“.

Kataloogide sisu:
\Eliza - lokaalseks kasutamiseks kompileeritud rakendus
\andmebaasi_genereerimine - Pythonis kirjutatud rakendus genereerimaks andmebaasi
\eliza_lahtekood - rakenduse lähtekood

4.1. Andmebaasi uuendamine Pythoni programmi abil

Vastav rakendus asub kataloogis \andmebaasi_genereerimine.

1. Veenduge, et arvutisse on installeeritud Python. Kui ei ole, siis viimase versiooni saab alla laadida aadressilt: http://www.python.org/download/
2. Tehke vajalikud muudatused loc.*.dat ning plan.*.dat failidesse. Soovi korral võite ka luua uusi liine tähistavaid faile. Sellisel juhul tuleb Script.py failis muuta ridasid 613 ning 616. Peate lisama uued failinimed planFiles ja locationFiles järjenditesse.
3. Käivitage Script.py. Tulemusena peaks samasse kataloogi, kus asus Script.py, tekkima fail nimega skript (CD plaadilt antud rakendust käivitada ei saa, kuna puudub õigus sinna kirjutada).

4.2. Eliza lokaalse rakenduse loomine ja kasutamine

1. Veenduge, et arvutisse on installeeritud Sun Microsystems Java Runtime Environment(JRE). Kui ei ole, siis viimase versiooni saab alla laadida aadressilt: http://www.java.com/en/download/manual.jsp
2. Kui Te soovite rakendust ainult käivitada ega soovi seda uuesti kompileerida, minge neljanda sammu juurde.
3. Enne rakenduse kompileerimist veenduge, et Eliza.java failis on muutuja local tõeväärtus „true“. Kompileerige kogu Eliza pakett.
4. Liikuge konsooliga kataloogi, kus asub Eliza nimeline kataloog, milles on kompileeritud Java class failid.
5. Sisestage konsooli järgmine käsk:

```
> set CLASSPATH=.;[kataloog];
```

   [kataloog] asemele sisestage täistee kataloogi, kus hetkel konsooliga asute.
6. Rakenduse saate tööle panna käsuga:

```
> java Eliza.Qloader
```

Rakendus kasutab andmebaasina script faili, mis asub kataloogis, kus konsooliga asute. Kui soovite andmebaasi uuendada Pythoni programmi abil, siis tuleb saadud uus script fail tõsta siia kataloogi vana asemele.

4.3. Eliza Internetirakenduse loomine ja kasutamine

Rakenduse lähtekood asub kataloogis \eliza_lahtekood.

1. Veenduge, et arvutisse on installeeritud Sun Microsystems Java Runtime Environment (JRE). Kui ei ole, siis viimase versiooni saab alla laadida aadressilt: http://www.java.com/en/download/manual.jsp
2. Avage Eliza.java ning muutke parameetri scriptURL väärtus selliseks, et see viitaks Internetis asuvale script failile (script faili peate vastavale aadressile üles laadima).
3. Veenduge ka, et Eliza.java failis asuv parameeter local oleks tõeväärtusega "false".
4. Kompileerige kogu Eliza pakett.
5. Laadige kompileeritud failid Interneti kataloogi /eliza/Eliza. (/eliza on kataloog, kust pärast rakendusele juurdepääsete.)
6. Laadige Internetti CD plaadil leiduv index.html kataloogi /eliza/.
7. Minge veebilehitsejaga aadressile kogulehe-url/eliza

Töötav näide asub aadressil: http://www.joel.ee/eliza

5. Piirangud ning tekkinud probleemid

Antud süsteemi teadmuspagas on väga piiratud ja rääkida saab ainult sõidugraafikutest. Üldise testimise käigus panin loodud süsteemi ka enda koduleheküljele ja lasin inimestel seda katsetada (http://www.joel.ee/eliza). Tagasisidena öeldi, et vajalik informatsioon saadi üldjuhul probleemideta kätte, kuid rakendus satub kergesti segadusse, kui üritada rääkida suvalistel teemadel.

Kuna sõnede morfoloogilise analüüsi tegemine oleks läinud liiga mahukaks, tuleb reisi lähtepunkti asukoha nimi alati anda seestütlevas käändes.

Kuna iga reisiplaan vaadatakse läbi eraldi, siis tekkis olukord, kus näiteks Tallinn-Tartu ning Valga-Tartu liinidel on üks ühine peatus – Tartu. Kuna aga peale Tallinn-Tartu liini reeglite genereerimist loodi ka juba reegel võtmesõnaga „tartu“, siis hiljem vaadates Valga-Tartu liini, luuakse uuesti reegel võtmesõnaga „tartu“. Eliza programm otsib ainult esimest võtit, mis sõnele vastab, seetõttu teist reeglit ei vaadatud kunagi. Probleemi lahendamiseks tuleb vaadata jooksvalt ka järgmiste reisiplaanide peatuste nimesid ning kui leidub sama nimega peatus, tuleb kaks või enam reeglit ühendada.

Samuti osutus probleemiks küsimus, kunas puhastada mälu. Kui seda puhastada liiga tihedalt, võib tekkida probleem, et kasutaja ei jõua veel täpsustada soovitud reisi aegagi, enne kui rakendus unustab ära sõidu sihtpunktid. Samas võib kasutaja soovida muuta reisi algus- või lõpppunkti, seega peab selline võimalus olemas olema. Sai otsustatud sellise variandi kasuks, et kui sisendis antakse ette kohanimi seestütlevas käändes, siis puhastatakse mälu ning salvestatakse uus kohanimi lähtepunktina.

Väga keeruline on testida, kas kõik andmed, mis ette antakse, ka reaalselt reeglite kaudu lõpuks kasutajale kuvatakse. Kuna reeglite failis on peaaegu 60000 rida, on faili lugemine ning käsitsi tulemuste kontrollimine väga raske. Iga kahe peatuse vahel, kus teine peatus asub liinil peale esimest, luuakse reegel andmebaasi ning kontrollitakse, kas nende punktide vahel on võimalik reisida igal võimalikul nädalapäeval. Seega kasvab andmebaas võrdlemisi kiiresti. Kui liinil asub n peatust, siis luuakse $(n-1)^2$ reeglit. Lisaks tuleb kontrollida rongide liikumist nädalapäevade lõikes.

6. Edasiarendamise võimalused

Muutmaks loodud tehisintellekti-süsteemi suhtlust kasutajatega sujuvamaks, saaks lisada näiteks mooduli, mis salvestaks vestlused kasutajatega. See annaks võimaluse hiljem analüüsida logisid ning leida olukorrad, kus kasutaja sattus segadusse või rakendus ei osanud enam midagi mõistlikku vastata. Samuti saaks muuta Pythoni programmi tööd optimaalsemaks, vähendades lähte-andmefailide kasutamist. Seda saaks näiteks teha, lugedes andmefailid sisse ning salvestades nad muutujatesse. See kiirendaks oluliselt programmi tööd, kui sisendfailide arv kasvab väga suureks.

Pythoni programmis asuv GenerateRule meetod peaks siduma ka asukohad, mis on omavahel kaudselt seotud erinevate sõiduliinide abil. Tuleks kontrollida, kas kahte liini ühendavast punktist väljub samal päeval edasi rong soovitud sihtkohta. Samuti peaks GenerateRule meetodid kontrollima rekursiivselt kõiki kohanimesid kõigis järgnevates asukohafailides korduste suhtes. Kui kohanimed kattuvad, tuleks vastavad reeglid ühendada. Hetkel kontrollitakse ainult järgmist kahte asukohafaili korduste suhtes.

Lähteandmefaile võiks samuti genereerida Pythoni abil, et vähendada sisestusvigade tekkimise võimalusi. Python saaks jooksvalt kontrollida andmete vastavust teatavatele reeglitele ja seega muuta kogu protsessi vigadekindlamaks.

Lisades kõnetuvastuse ja kõnesünteesi moodulid, saaks antud rakendust kasutada ka nägemispuudega inimesed. See moodus oleks väga sobilik ka olukordades, kus klaviatuuri kasutamine on ebamugav – näiteks nutitelefonides.

Samuti võiks laiendada süsteemi andmebaasi piletihindadega.

7. Kokkuvõte

Oma töö eesmärgiks võtsin luua tehisintellekti tehnikaid realiseeriva süsteemi, mis suudaks eesti keeles kirjalikult kasutajaga suhelda ning väljastada infot Edelaraudtee sõiduplaanide kohta.
Selleks täiendasin tuntud programmi Eliza, lisades sellele kontekstitundliku mälu, võimaluse rakendust ka lokaalselt graafilises kasutajaliideses käivitada, märksõnade „täna“ ning „homme“ interpreteerimise konkreetseks nädalapäevaks ning parandasin kasutamis-mugavust. Kirjutasin ka Pythoni rakenduse, mis suudab etteantud sõiduplaanide info põhjal genereerida Eliza tööd juhtiva reeglite faili.
Kokkuvõtvalt võib öelda, et suutsin seatud eesmärgi täita. Suutsin luua rakenduse, mida saab kasutada nii veebipõhisena kui ka lokaalses arvutis. Lõpptulemus on nähtav ka Internetis aadressil http://www.joel.ee/eliza.

8. Artificial Intelligence of the Edelaraudtee Railway Company

Summary

Joel Edenberg

The objective of my bachelor thesis was to create a software which is able to give information about the train schedules of the Edelaraudtee Railway Company by communicating with user in written Estonian language.

To achieve this goal I enhanced a program called Eliza by adding context memory, ability to run the application locally with a graphical user interface, the interpretation of keywords “today” and “tomorrow” into given weekdays and improved the overall usability. I also wrote a program in Python for generating a control file based on the schedule data for Eliza.

In conclusion I achieved my goal. I managed to create a working application which could be used either online as an Java applet or offline locally. The final outcome can be seen at: http://www.joel.ee/eliza

9. Kasutatud kirjandus

1) Hayden, C. *Eliza*. http://chayden.net/eliza/Eliza.html. Vaadatud 27. mai 2010.
2) *Edelaraudtee sõiduplaanid*. http://www.edel.ee/soiduplaanid/. Vaadatud 27. mai 2010.
3) Treumuth, M. *Eesti dialoogikorpus ja selle töötlemise tarkvara*. Magistritöö. http://dspace.utlib.ee/dspace/bitstream/10062/1172/5/Treumuth.pdf. Vaadatud 27. mai 2010.
4) Treumuth, M. *Teatriagent (Alfred)*. http://www.dialoogid.ee/teatriagent. Vaadatud 27. mai 2010.

10. Lisad

Lisa 1: väljavõte reeglite failist

```
key: x2y37z4
  decomp: * @esmaspäev *
    reasmb: Tabiverelt Raasikusse  saab sõita (2): $Rong nr 0211: väljub
kell 07:52, saabub kell 10:11. $Rong nr 0213: väljub kell 18:24, saabub
kell 20:39.
  decomp: * @teisipäev *
    reasmb: Tabiverelt Raasikusse  saab sõita (2): $Rong nr 0211: väljub
kell 07:52, saabub kell 10:11. $Rong nr 0213: väljub kell 18:24, saabub
kell 20:39.
  decomp: * @kolmapäev *
    reasmb: Tabiverelt Raasikusse  saab sõita (2): $Rong nr 0211: väljub
kell 07:52, saabub kell 10:11. $Rong nr 0213: väljub kell 18:24, saabub
kell 20:39.
  decomp: * @neljapäev *
    reasmb: Tabiverelt Raasikusse  saab sõita (2): $Rong nr 0211: väljub
kell 07:52, saabub kell 10:11. $Rong nr 0213: väljub kell 18:24, saabub
kell 20:39.
  decomp: * @reede *
    reasmb: Tabiverelt Raasikusse  saab sõita (2): $Rong nr 0211: väljub
kell 07:52, saabub kell 10:11. $Rong nr 0213: väljub kell 18:24, saabub
kell 20:39.
  decomp: * @laupäev *
    reasmb: Tabiverelt Raasikusse  saab sõita (2): $Rong nr 0211: väljub
kell 07:52, saabub kell 10:11. $Rong nr 0213: väljub kell 18:24, saabub
kell 20:39.
  decomp: * @pühapäev *
    reasmb: Tabiverelt Raasikusse  saab sõita (2): $Rong nr 0211: väljub
kell 07:52, saabub kell 10:11. $Rong nr 0213: väljub kell 18:24, saabub
kell 20:39.
  decomp: *
    reasmb: Kuna te soovite sõita Tabiverelt Raasiku?
```

Lisa 2: lähteandmete fail init.dat

```
initial: Kust kuhu ning kuna te soovite reisida?
final: Nägemist!
quit: Nägemist
quit: Head aega
quit: lõpp
pre: ülemistest Ülemistest
pre: ülemistele Ülemistele
pre: ülemistesse Ülemistesse
-
key: xnone 0
  decomp: *
    reasmb: Hakkame otsast pihta. Kust soovite reisi alustada?
    reasmb: Ma ei saanud nüüd hästi aru. Aga kust soovite reisi alustada?
    reasmb: Ma oskan rääkida ainult sõidugraafikutest. Kust ja kuhu soovite
reisida?
    reasmb: Läheb keeruliseks. ?
    reasmb: Hakkame otsast pihta. Kust soovite reisi alustada?
key: ei 10
  decomp: *
    reasmb: Selge. Aga öelge kust ning kuhu te tahate siis reisida.
key: olgu 10
  decomp: *
    reasmb: Saan veel mõne sihtkohaga aidata?
key: selge 10
  decomp: *
    reasmb: Küsige julgelt kui veel küsimusi on!
key: aitäh 10
  decomp: *
    reasmb: Pole tänu väärt! Saan veel millegagi aidata?
key: aitab 10
  decomp: *
    reasmb: Hea, et sain olla kasulik!
key: ahah 10
  decomp: *
    reasmb: Saan veel millegagi aidata?
key: abi 100
  decomp: *
    reasmb: ==================================== $ABIINFO: $ Lähtejaama
nimi tuleb sisestada sisseütlevas käändes. $ Sihtkoha nimi võib olla nii
algvormis kui alaleütlevas käändes$ Reisimise ajaks võib öelda nii täna,
homme, kui ka konkreetse nädalapäeva.$ Näide: Sooviksin sõita täna
Tallinnast Tartusse.$====================================
key: informatsioon 100
  decomp: *
    reasmb: ==================================== $ Edelaraudtee
tehisintellekt versioon 1.0$ Tarkvara loodud bakalaureusetöö raames.$ Eliza
v0.1 porgrammeeritud Charles Hayden'i poolt.$$ Autor: Joel Edenberg$
Juhendaja: Mare Koit$ Tartu Üliool
2010$====================================
```

Lisa 3: kohanimede fail loc.tallinn.tartu.dat

```
Tallinna
Ülemiste
Lagedi
Aruküla Aruküllа
Raasiku
Kehra
Aegviidu
Nelijärve
Jäneda
Lehtse
Tapa
Tamsalu
Kiltsi
Rakke
Vägeva
Pedja
Jõgeva
Kaarepere
Tabivere Tabiverre
Tartu
```

Lisa 4: sõiduplaanide fail plan.tallinn.tartu.dat

```
0210|ETKNRLP|tallinn!06:40|Ülemiste!06:50|lagedi!07:00|aruküla!07:10|raasik
u!07:17|kehra!07:28|aegviidu!07:44|nelijärve!07:48|jäneda!07:52|lehtse!08:0
0|tapa!08:14|tamsalu!08:27|kiltse!08:34|rakke!08:43|vägeva!08:51|pedja!09:0
0|jõgeva!09:10|kaarepere!09:22|tabivere!09:33|tartu!09:50
0010|ETKNR|tallinn!07:46|Ülemiste!07:56|lagedi!-|aruküla!-|raasiku!-
|kehra!-|aegviidu!-|nelijärve!-|jäneda!-|lehtse!-
|tapa!09:00|tamsalu!9:13|kiltse!-|rakke!-|vägeva!-|pedja!-
|jõgeva!09:44|kaarepere!-|tabivere!-|tartu!10:15
0012|RP|tallinn!13:55|Ülemiste!14:05|lagedi!-|aruküla!-|raasiku!-|kehra!-
|aegviidu!-|nelijärve!-|jäneda!-|lehtse!-|tapa!15:09|tamsalu!15:22|kiltse!-
|rakke!-|vägeva!-|pedja!-|jõgeva!15:53|kaarepere!-|tabivere!-|tartu!16:24
0014|ETKNRLP|tallinn!16:45|Ülemiste!16:55|lagedi!-|aruküla!-|raasiku!-
|kehra!-|aegviidu!-|nelijärve!-|jäneda!-|lehtse!-
|tapa!17:59|tamsalu!18:12|kiltse!-|rakke!-|vägeva!-|pedja!-
|jõgeva!18:48|kaarepere!-|tabivere!-|tartu!19:19
0016|RP|tallinn!19:59|Ülemiste!20:09|lagedi!-|aruküla!-|raasiku!-|kehra!-
|aegviidu!-|nelijärve!-|jäneda!-|lehtse!-|tapa!21:13|tamsalu!21:26|kiltse!-
|rakke!-|vägeva!-|pedja!-|jõgeva!21:57|kaarepere!-|tabivere!-|tartu!22:28
```