Tallinna Ülikool
Informaatika Instituut

Uurimus internetipõhistest interaktiivsetest tehisintellekti rakendustest

Seminaritöö

Autor: Elar Huik

Juhendaja: PhD Erika Matsak

Autor: ..... „ ....., 2015

Juhendaja: ..... „ ....., 2015

Instituudi direktor: ..... „ ....., 2015

Tallinn 2015

SISUKORD

SISSEJUHATUS

Taust

Viimase aastakümne tehnoloogia arengu käigus on arvutid muutunud kiiremaks, odavamaks ja selle tulemusena on tavaelu üha rohkem ühildunud masinatega. Hea näide sellest on nutitelefonide ja tahvelarvutite suurenenud populaarsus. Kuna üha rohkem inimesi on harjunud kasutama arvuteid, siis loob see uusi võimalusi tehisintellekti kasutamiseks ja ka lihtsustab integratsiooni.

Nutitelefonides võib leida rakendusi nagu Siri [1] või Google Now [2], mis kujutavad endast personaalset intelligentset agenti, mis saavad kasutaja loomulikust keelest aru ja abistavad erinevate ülesannetega. Hiljuti on Amazon tulnud välja koduse lahendusega Amazon Echo [3], mis vastab inimese küsimustele ilmateate, uudiste või muude küsimuste kohta vikipeedia abiga. Asutused on hakanud kasutama oma veebilehtedel automatiseeritud assistente, mis põhiliselt tegelevad klienditeenindusega või aitavad kasutajal liigelda nende leheküljel. Näitena võib tuua Rootsi mööblifirma IKEA assistendi [4], nimega Anna, mis vastab toodetega seotud teemade kohta nagu hind, mõõtmed jne. Annal on ka avatar, mis innustab kliente temaga suhtlema.

Ulmekirjanduses ja ka filmitööstuses on kaua käsitletud masinate ja inimeste vaheliste interaktsioonide võimalusi. Tavaliselt küll on masinintellekt siis kas halb või hea. Film "Her", mis sobib viimasesse kategooriasse, on inspireeritud agendist ALICE [5]. Film on intrigeeriv vaade võimalikku tulevikku, kus inimesed võivad ennast emotsionaalselt siduda masinatega. Autor nimetab just seda filmi, kuna ALICE ja tema programmeerimismeetod on üks teemadest käesolevas uurimuses.

Eelnevad uurimistööd on leidnud, et interaktiivsete agentide kasutamine võib suurendada inimeste usaldust informatsiooni kohta, mida neile antakse. Lisaks võivad agendid esindada sotsiaalseid rolle, õpetada sotsiaalseid kombeid ja muuta liideste kasutamist lihtsamaks [6].

Agente saab kasutada ka meelelahutusotstarbetel, informatsiooni hankimiseks, e-kaubanduses assistendina, suunamaks klienti, töövahendina keele õppimiseks [7] või kaasata agenti avatud õppimise mudelites, et aidata õpilastel paremini teemat omistada [8].

Praktilisi kasutusi agentidele on palju. Igal asutusel eksisteerib kasutajatugi formaadis, kus klient suhtleb asutuse töötajaga. Asendades töötaja adekvaatse agendiga, saab pakkuda ööpäevaringset kasutajatuge ja rakendada tööjõudu muudel otstarbetel. Meditsiinis saaks kasutada agente diagnooside määramiseks või õpetamaks ravimeetodeid erinevatele haigustele, haavadele vms. Võimalik oleks kasutada agente ka ohtlike inimeste tabamiseks jututubadest või sotsiaalmeedia lehekülgedelt.

Autor valis teema, kuna see on intrigeeriv ja huvi pakkuv ning plaanib teemat jätkata bakalaureusetöös.

Ajalugu

Intelligentne agent ehk ka dialoogsüsteem on programm, mis suhtleb kasutajaga loomulikus keeles. Tegemist ei ole uue tehnoloogiaga. Dialoogsüsteemide arendus on kestnud 50 aastat [9, lk 188]. Esimene tuntuks saanud dialoogsüsteem, mis suhtles inglise keeles, oli 1965. a Joseph Weizenbaum poolt loodud programm ELIZA [9, lk 183].

Järgnev dialoog toimus ELIZA ja inimese vahel [10]:

```
Inimene: Perhaps I could learn to get along with my mother.
ELIZA: Tell me more about your family.
Inimene: My mother takes care of me.
ELIZA: Who else in your family takes care of you.
Inimene: My father.
ELIZA: Your father.
Inimene: You are like my father in some ways.
ELIZA: What resemblance do you see.
Inimene: You are not very aggressive but I think you don't
want me to notice that.
ELIZA: What makes you think I am not very aggressive.
Inimene: You don't argue with me.
```

ELIZA tundub arukana, aga tegelikult ei saa aru mida inimene räägib või mida ta ise ütleb. Programm vastab kasutajale kindlate reeglite põhjal. ELIZA otsib kasutaja öeldust kindlaid sõnu, millele ta "oskab" vastata [9, lk 184]. Joseph Weintraub ostis programmi ELIZA ja arendas tema funktsiooniks olla arvuti terapeut ja on võitnud 4 Loebneri võistlust selle uuendatud programmiga [11, 12].

"Kas masin suudab mõelda?" Sellele küsimusele vastas Alan Turing jaatavalt, aga kesksem küsimus oli "Kui masin suudaks mõelda, kuidas me aru saaksime?". Turingi arvamus oli, et kui masina vastused on eristamatud inimese vastustest, siis võib öelda, et masin mõtleb. Seda nimetatakse Turingi testiks ja 1990 loodi Hugh Loebneri ja Cambridge Käitumisuuringute keskuse poolt esimene võistlus leidmaks arvutiprogramm, mis läbib Turingi testi. Testi läbimise auhinnaks on 100 000 dollarit ja kuldmedal. Esimene võistlus toimus 1991 ja toimub tänaseni annuaalselt. Igal aastal väljastatakse pronksmedal ja rahaline summa kõige inimlikumale programmile. Turingi testi pole ükski masin siiani läbinud [11].

Uurimuse eesmärk

Uurimuse eesmärgiks on klassifitseerida olemasolevaid meetodeid interaktiivsete agentide loomiseks ja analüüsida probleeme, mis agentidel võivad esineda.

Autor kavatseb käsitleda kolme erinevat tehisintellekti loomise süsteemi, millest kõik töötavad sama põhimõttega, kuid erinevad ehituse ja tööloogika puhul piisavalt, et eristada tugevusi ja nõrkusi ka süsteemide vahel. Töö sisu osas loob autor ülevaate iga agendi loomisprotsessist, peale mida tuleb osa, mis sisaldab analüüsi agentide tugevustest, nõrkustest ja probleemide kirjeldusest, mida tuleb arvestada vastava agendi loomisel. Seejärel klassifitseerib autor agendid tähtsaimate omaduste abil.

Uurimisplaan

Autor viib läbi uurimuse, käsitledes piisaval määral agentidega seotud ajalugu. Selgitab põhimõtteid ja loogikat mille abil agent suhtleb, lisades näiteid, mis illustreerivad teooriat.

Peale sisulist uuringut, määrab autor agentide tugevused, nõrkused ja võrdleb neid ning kirjeldab probleeme, mis agentidega võivad esineda. Seejärel klassifitseerib autor agendid nende omaduste põhjal.

MÕISTETE JA LÜHENDITE NIMEKIRI

AIML - Artificial Intelligence Markup Language.

AI - Artificial Intelligenge ehk eesti keeles tehisintellekt.

AI Engine - Interpretaator mida kasutavad Personality Forge keskkonnas loodud agendid.

A.L.I.C.E. - Artificial Linguistic Internet Computer Entity

API - Application Programming Interface ehk liides kasutamaks muud programmi.

CBR - Case Based Reasoning.

Flash - multimeedia ja tarkvara platvorm vektorgraafika ja animatsioonide loomiseks.
HTML - Hüperteksti märgistuskeel, mida kasutatakse veebilehtede loomiseks.

Intelligentne agent - arvuti programm, mis suudab suhelda inimesega loomulikus keeles. Sünonüümid: jutuagent, juturobot, diaaloogsüsteem, TIS.

Interpretaator - programm, mille abil AIML, Cleverscript ja PF koodid töötavad.

NLP - Natural Language Processing.

N^+ - Naturaalarvude hulk {1, 2, 3, 4, ... }.

PF - Lühend sõnadest: Personality Forge

Parsima - liigendama plokke lauseteks, lauseid avaldisteks, avaldisi tehtemärkideks ja operandideks [27].

Statement - Ehk lause on ilmutatult lõpetatud süntaktiline üksus, mis esitab deklaratsiooni või kirjutab ette tööüksuse, näidates sooritatavad toimingud, neis toiminguis kasutatavad operandid (kui neid on) ja võimalike tulemite paigutuse [27].

Teadmusbaas - TISi tuumaks on teadmusbaas, mis võimaldab süsteemil probleemide lahendamisel valida efektiivseid meetodeid ja teha arukaid otsuseid [9 ,lk 15].

TI - Tehisintellekt. Informaatika haru, mis tegeleb selliste andmetöötlussüsteemide arendusega, mis täidavad inimmõistusele omaseid funktsioone, nagu arutlemine, õppimine ja enesetäiendus [27].

TIS - tehisintellekti süsteem

UTF-8 - rahvusvaheline standard tähtede kodeerimiseks arvutites.

XML - Extensible Markup Language ehk laiendatav märgistuskeel.

1 AIML AGENT

AIML arendati Dr. Richard S. Wallace ja Alicebot tarkvara kogukonna poolt aastatel 1995-2000. See oli aluseks esimeseks Alicebot agendiks nimega A.L.I.C.E. [14]. Tänapäevaks on ALICE võitnud 3 Loebneri võistlust ja on üks tuntumaid juturoboteid [11, 13]. AIML arendus on jätkuv ja kuna AIML on vabavarana kättesaadav, on kõikidel võimalik aidata AIMLi luua paremaks.

AIML loomisel järgitud eesmärgid [14]:

- Olla lihtsalt õpitav ja kasutatav.
- Sisaldada vähimat elementide hulka, mis on vajalik, et võimaldada stiimuli-vastuse põhist teadmussüsteemi.
- Olla XML baasil loodud ja sellega ühilduv.
- AIMLi töötlevaid programme peab olema lihtne kirjutada.
- AIML elemendid peavad olema inimeste jaoks loetavad ja arusaadavad.
- AIML ei sõltu teistest keeltest.

AIML agente saab luua ja tööle panna veebilehe www.pandorabots.com Playground keskkonnas. Pandorabotsis võib oma agendi teadmusbaasi, kas üles laadida, või kasutajaliidest kasutades, ükshaaval sisendeid ja väljundeid kirjutades. Keskkonna kasutamiseks on vaja luua konto. Kasutamaks agenti, mis töötab Pandorabots serverites, enda veebilehekülgedel või rakendustes, on vaja kasutata nende pakutud APIt. Tasuta API leping lubab omada 2 agenti ja teha 25 interaktsiooni päevas. Rohkemate interaktsioonide jaoks on vaja maksta kuupõhist tasu.

Omades oskusi või teadmisi muudest programmeerimiskeeltest, on võimalik kasutada tasuta AIML interpretaatorprogramme leheküljelt www.alicebot.org/downloads/programs.html ja need seadistada töötama oma serveris või rakenduses.

1.1 AIML elementide ülevaade

Käesolev peatükk käsitleb erinevate AIML elementide omadusi, kirjutamise struktuuri ja sisaldab koodinäited, mis illustreerivad teoreetilist osa.

Vastavalt XML standardile on igal AIML objektil kindel loogiline ja füüsiline struktuur, mida tuleb järgida agendi kirjutamisel. Iga objekt koosneb elementidest millest igaühel on omad märgendid. Märgendite vahele kirjutatakse agendi teadmus. Järgnevates peatükkides kirjeldatakse erinevate elementide rolli, nendele vastavate märgendite süntaksi, struktuurilist paiknemist AIML dokumendis. Süntaksivigade esinemisel muutub .aiml fail interpretaatori jaoks loetamatuks ja väljastatakse veateade [14].

AIML faili sisu kirjutamise süntaks on järgnev:

```
<elemendinimi> ParameetriteLoetelu </elemendinimi>
```

Igal elemendil on algus märgend ja lõpu märgend. "ParameetriteLoetelu" on võimalik elemendi sisu, mis võib koosneda teistest elementidest või/ka tekstist. See mis elemendi sisse võib kirjutada sõltub elemendi tüübist, kuna igal elemendil on oma otstarbe ja struktuur, mida järgida [14, 15].

1.1.1 **<aiml>**

Iga AIML dokumendi sisu algab <aiml> märgisega ja lõppeb </aiml> märgisega. Tabel 1. AIML koodi näidis illustreerib <aiml> märgiste kasutust. <aiml> ja </aiml> märgiste vahel võivad alamelementidena esineda <topic> ja <category>.

Tabel 1. AIML koodi näidis

```
<aiml>
 <category>
    <pattern> TERE AGENT </pattern>
    <template>
       Tere sõber!
    </template>
 </category>
</aiml>
```

1.1.2 **<category> ehk teadmusosake**

AIML dialoogi põhiosaks on <category> elemendid, ehk kategooriad. Iga kategooria on osake agendi kogu teadmusbaasist. Iga kategooria koosneb kasutaja sisendist (pattern), mis on lause kujul ja vastusest (template) sellele sisendile, mille agent ette kannab [14, 15].

Teadmusbaas, mis on kirjutatud AIML koodis, moodustatakse kategooriate komplektist. Kategooriad organiseeritakse vastavalt teemadele ja salvestatakse .aiml laiendiga faili. Kategooriate sees olev teadmus kirjutatakse <category> ja </category> märgendite vahele [14, 15]. Tabel 1 sisaldab näidist kategooria märgendite kohta ja vastab järgnevale dialoogi väljavõttele kasutaja ja agendi vahel:

```
Kasutaja: Tere Agent!
Agent: Tere sõber!
```

Kategooria peab asuma <aiml> elemendi sees ja peab kindlasti sisaldama elemente <pattern> ja <template>.

1.1.3 **<pattern> ehk sisendmall**

<pattern> ja </pattern> märgiste vahel asub võimalik kasutajapoolne sisend. Igas kategooria elemendis on ainult üks <pattern> element ja see peab olema ka esimene element, mis kirjutatakse kategooriasse. Algoritm, mis töötleb sisendit, eemaldab kõik kirjavahemärgid ja seetõttu tuleb kirjutada kasutaja sisend ilma erimärkideta [14]. Sisend ei ole tõstutundlik. Tabelis 1 kirjutatud <pattern> element tuvastaks järgnevad kasutajapoolsed sisendid: "Tere, Agent!", "TERE. AGENT","tere AGENT", jne. Erimärkide ainukeseks erandiks on asterisk *.

1.1.4 **<template> ehk väljundmall**

<template> element sisaldab võimalikku vastust, mida agent esitab kasutajale. See element peab asetsema kategooria elemendi sees ja olema kirjutatud peale <pattern> elementi. Väljundmall võimaldab andmeid salvestada, anda juhuslikke vastuseid või kutsuda esile vastuseid teistest kategooriatest [15]. Tabelis 1 annab <template> element vastuse kasutajale, mis antud juhul oleks "Tere sõber!".

Eelnevas alampeatükis sai mainitud erimärki *. Selle nimetus on wildcard ja ta funktsioon on olla muutuja, mis hõivab sisend-tekstis mingi kindla osa, et siis kasutada seda osa tekstist väljundmallis, või luua rohkem võimalusi, et sisend tuvastatakse. Tabelis 2 on kasutatud * märki nii sisendis kui ka väljundis ja dialoog, mis võidakse genereerida selle koodi poolt oleks:

```
Kasutaja: Mulle meeldib ujuda talvel!
Agent: Mulle ei meeldi ujuda talvel :(
```

Tabel 2. * kasutamine sisendis ja väljundis.

```
<category>
   <pattern> MULLE MEELDIB * </pattern>
   <template>
      Mulle ei meeldi *  :(
   </template>
</category>
```

Wildcard võib väga palju sõnu enda alla võtta ja kuigi see võib hea olla, on see samuti ka halb. Kui kasutaja peaks ütlema "Mulle meeldib tavaliselt joosta hommikuti kui liiklust on vähe, aga kuidas sinuga on?" siis agendi vastus Tabeli 2 koodi põhjal oleks absurdne. Seetõttu peetakse AIML keelt limiteeritud oma sisendi tuvastusvõime poolest ja väljendusvaeseks oma lihtsuse tõttu [25]. Kuna agendi looja peab kõiki võimalikke lause kirjutusviise käsitlema, saavutatakse hea AIML agent alles kümnete tuhandete kategooriate loomise järel [26].

1.1.5 **<star index = "n"/>**

See märgis omistab tükikese tekstist, mis asub kasutaja poolt öeldud sisendlauses ja kasutatakse koos wildcardidega. Indeks n näitab sisendfraasi osa, mis omistatakse [14].
Näide illustreerimaks seda:

- `<star index="1"/>` on samaväärne esimese fragmendiga tekstist.
- `<star index="2"/>` on samaväärne teise fragmendiga tekstist.
- `<star index="3"/>` on samaväärne kolmanda fragmendiga tekstist.

Atribuut `index="n"` on valikuline ja selle kasutamine pole vajalik kui sisendis on üksik wildcard. Märgis `<star/>` samaväärne märgisele `<star index="1">`.

Tabel 3. <star> märgise kasutuse näidis.

```
<category>
   <pattern> MULLE MEELDIB * </pattern>
   <template>
      Mulle ka meeldib <star/>
   </template>
</category>

<category>
   <pattern> * ON * </pattern>
   <template>
      Millal <star index="1"/> ei ole <star index="2"/>?
   </template>
</category>
```

Tabeli 3 põhjal on võimalik järgmine interaktsioon kasutaja ja agendi vahel:

```
Kasutaja: Puud on raagus.
Agent: Millal puud ei ole raagus?
```

1.1.6 **<srai> käsklus**

Üks kasulik omadus AIML keeles on võimalus seostada teistest kategooriatest pärit <pattern> elemente ühe <template> elemendi sees, ehk AIML interpretaator saab efektiivselt vasteid otsida erinevatest sisendmallidest. See võimalus on saavutatav <srai> elemendi abil [14, 15].

Esimene <srai> kasutusviis on jagada keeruline kasutaja sisend lihtsamateks osadeks [15, 16]. Näiteks küsimus "Kes on X?", kus X esindab ükskõik millist isikut, on võimalik kirjutada erinevatel viisidel, nagu "Kas sa tead kes X on?" ja "Mida sa veel tead X kohta?". Et katta võimalikult palju grammatilise viise, kuidas mingi subjekti kohta vesteldakse, on abiks <srai> element ja Tabel 4 toob näite, kuidas esialgsest sisendist vähendatakse sõnu, et leida vastus.

Tabel 4. Sõnade vähendamise näide <srai> abil.

```
<category>
   <pattern> KES ON ALAN TURING </pattern>
   <template>
      Alan Turing oli Briti matemaatik, krüptograaf ja arvutiteadlane.
   </template>
</category>

<category>
   <pattern> KES ON ALBERT EINSTEIN </pattern>
   <template>
      Albert Einstein oli 20. sajandi kõige mõjukam füüsik.
   </template>
</category>

<category>
   <pattern> KAS SA TEAD KES ON * </pattern>
   <template>
      <srai> KES ON <star/> </srai>
   </template>
</category>
```

Tabelis 4 on kategooria, mis räägib Alan Turingist ja kategooria, mis räägib Albert Einsteinist. Esimene kategooria tuvastab kasutaja sisendi "KES ON ALAN TURING" ja teine kategooria küsimuse "KES ON ALBERT EINSTEIN". Arvestades, et kasutaja võib samadest isikutest erinevatel viisidel rääkida, siis on eraldi kategooria, mis modelleerib erinevat grammatilist sisendit. Selles viimases kategoorias tuvastatakse sisend, kui kasutaja küsib "KAS SA TEAD KES ON *", kus * on siis muutuja, mille eesmärk on identifitseerida isiku nimi, kellest räägitakse. Kasutaja sisendi vastavusel viimasele sisendmallile käivitub käsklus <srai> KES ON <star/> </srai> ja suunab otsima vastust teisest kategooriast. <star/> märgis omandab muutuja * poolt sisendis omistatud nime. Vastavalt, lühendatakse sisend "KAS TEAD KES ALAN TURING" kujule "KES ON ALAN TURING" ja vastus sellele saadakse esimesest kategooriast.

Tabelis 5 on näidatud, kuidas on võimalik kasutada <srai> elementi sünonüümide käsitlemiseks.

Tabel 5. <srai> kasutus sünonüümide jaoks.

```
<category>
   <pattern> ÜLIKOOL </pattern>
   <template>
      See on õppeasutus.
   </template>
</category>

<category>
   <pattern> KÕRGKOOL </pattern>
   <template>
      <srai> ÜLIKOOL </srai>
   </template>
</category>
```

1.1.7 **<random> ja <li> juhuslikkuse tekitamiseks**

<random> elementi kasutatakse, et ühele sisendile oleks mitu erinevat vastust. Sellise meetodi korral peab iga väljundsõnum olema <li> elementidega piiritletud, moodustades loetelu, millest valitakse juhuslikult üks, mida esitada väljundiks [14, 15]. Tabelis 6 on esitatud näide juhuslikest vastustes.

Tabel 6. <random> ja <li> märgiste kasutamise näide.

```
<category>
   <pattern> TERE </pattern>
   <template>
      <random>
        <li> Meeldiv tutvuda! </li>
        <li> Tere! </li>
        <li> Tere sullegi! </li>
      </random>
   </template>
</category>
```

Tabeli 6 koodi põhjal on võimalik järgnev dialoog kasutaja ja agendi vahel:

```
Kasutaja: Tere!
Agent: Tere!

Kasutaja: Tere.
Agent: Meeldiv tutvuda!
```

1.1.8 **<set> ja <get> õppimiseks ja mäletamiseks**

Elemendid <set> ja <get> võimaldavad agendil töötada muutujatega ja on ainuke meetod, kuidas agent õpib midagi kasutajalt. Osad muutujad, nagu agendiga seotud informatsioon (nt. nimi, vanus, sugu, elukoht), on juba keelde sisse ehitatud (kujul bot_name, bot_age, jne). Uusi muutujaid saavad luua programmeerijad, defineerides muutuja nime ja omistades sellele algse väärtuse [14, 15].

Muutujate loomiseks kasutatakse <set> elementi ja see peab asetsema <template> märgiste vahel. Süntaks on järgnev:

`<set name = "muutujaNimi"> muutujaVäärtus </set>`

Loodud muutuja nimeks saab vastavalt `muutujaNimi` ja väärtus mis selles muutujas omistatakse või uuendatakse on `muutujaVäärtus`. Tabel 7 on näide <set> märgiste kasutamisest muutuja loomiseks.

Tabel 7. <set> märgiste kasutamise näide.

```
<category>
   <pattern> MINU NIMI ON * </pattern>
   <template>
      Tere <set name="kasutajaNimi"> <star/> </set>
   </template>
</category>
```

Selleks, et kasutada õpitud informatsiooni, mis salvestati <set> elemendi abil on vaja kasutada <get> elementi. Samuti, peab see element asetsema <template> märgiste vahel. Süntaks on järgnev: `<get name = "muutujaNimi"/>`

Tabel 8 illustreerib <get> märgiste kasutamist. Arvestades koodi, mis oli Tabelis 7, siis Tabelis 8 kasutatakse väljundmallis eelnevalt salvestatud muutujat kasutajaNimi.

Tabel 8. <get> märgiste kasutamise näide.

```
<category>
   <pattern> HEAD UND </pattern>
   <template>
      Head und <get name="kasutajaNimi"/>
   </template>
</category>
```

Tabeli 7 ja 8 AIML koodide põhjal võib esineda järgnev interaktsioon kasutaja ja agendi vahel:

```
Kasutaja: Minu nimi on Elar.
Agent: Tere Elar
Kasutaja: Head und!
Agent: Head und Elar
```

1.1.9 **<topic> ehk teemade grupeerimine**

Antud elementi kasutatakse organiseerimaks teemasid, mille raames agent suudab rääkida. Selle realiseerimiseks asetatakse kõik ühise temaatikaga kategooriad vastava <topic> elemendi sisse. See element võimaldab simuleerida tähtsat osa inimestevahelises dialoogis. Nimelt juhtida jutt kindla teemani, rääkida sellest teemast ja ära tunda, millal teemat on vahetatud jutuajamise käigus [15]. Tabel 9 illustreerib <topic> märgiste kasutust. Näites on agent programmeeritud rääkima lilledest ja see teema asetseb märgise <topic name="lilled"> ja </topic> vahel. Jutuajamise käigus tuvastatakse teema vahetus siis, kui muutujale "topic" omistatakse väärtus "lilled" läbi käskluse <set name="topic">flowers</set>.

Tabel 9. Näide <topic> kasutusest.

```
<category>
   <pattern> RÄÄGIME LILLEDEST </pattern>
   <template>
      Olgu nii! <set name="topic">lilled</set>
   </template>
</category>

<topic name="lilled">
  <category>
     <pattern> * </pattern>
     <template>
        Lilled lõhnavad paremini kui inimesed.
     </template>
  </category>

  <category>
     <pattern> MULLE MEELDIVAD * </pattern>
     <template>
        Mulle meeldivad lilled ka!
     </template>
  </category>
</topic>
```

Võimalik dialoog kasutaja ja agendi vahel Tabeli 9 koodi põhjal:

```
Kasutaja: Räägime lilledest
Agent: Olgu nii!
Kasutaja: Mulle meeldivad liiliad!
Agent: Mulle meeldivad lilled ka!
Kasutaja: Miks sulle lilled meeldivad?
Agent: Lilled lõhnavad paremini kui inimesed.
```

2 CLEVERSCRIPT AGENT

Cleverscript on TI kirjutamise formaat ja tarkvara intelligentse dialoogsüsteemi loomiseks, mis on arendatud ettevõtte Existor poolt. Cleverscript põhineb Cleverbotil, mille loojaks on Rollo Carpenter, kes on aastatel 2005 ja 2006 võitnud Loebneri auhinna juturobotiga Jabberwacky [11].

Cleverscript agente saab luua www.cleverscript.com keskkonnas, kui on registreeritud kasutaja. Kirjutatud agentidega saab tasuta vestelda samas keskkonnas. Et kasutada agente enda veebisaitidel, rakendustes vms, saab seda teha kasutades APIt. API kasutamine muutub tasuliseks peale 1000 interaktsiooni kasutamist.

Agentide kirjutamiseks ei ole vaja omada programmeerimisoskusi. Agendile omane teadmusbaas kirjutatakse valmis tabeliprogrammi abiga ja põhiideeks on kirjutada sisendeid (mida agent tuvastab) ja väljundeid (mida agent öelda saab).

Cleverscript töötab UTF-8 tekstiga ja suudab reageerida igas inimkeeles. Cleverscripti on testitud ja arendatud ka vene keeles Kirillitsa alfabeediga [17].

Agendile on võimalik lisada ka täielikult animeeritud avatar koos kõnetuvastus- ja kõnesünteesimoodulitega. Töötavat avatari saab näha leheküljel www.cleverscript.com/demos/avatar-demo/.

2.1 Cleverscripti ülevaade

Cleverscript agendi teadmusbaas programmeeritakse kasutades tabeliprogrammi nagu OpenOffice, MicroSoft Excel või Google Docs.

Tabel 10 sisaldab näidet kõige lihtsamast töötavast agendist. Tabeli esimesel real olevad nimetused peavad olema igas Cleverscript tabelis, muidu ei ole interpretaatori jaoks tegu Cleverscript tabeliga [18].

Tabel 10. Lihtsa Cleverscript agendi näide.

Type	Label	Description	Text	If	Learn	Goto	Accuracy
output_start	tervitus	Esimene asi mida agent ütleb	Tere, ma olen sinu esimene agent!				

Tabelis 10 programmeeritud agent oskab teha ainult üht ja selleks on öelda vestluse alguses "Tere ma olen sinu esimene agent!". Seejärel, olenemata kasutaja järgnevatest sisenditest, vastab ta sama informatsiooniga. Tüüp "output_start" peab olema kirjas igal agendil ja selle sees olev sisu öeldakse igal korral kui agendiga alustatakse vestlust [18].

Type nimetusega veergude alla kirjutatakse tegevuse tüüp nt: "output_start", "input", "output", "phrase".

Label veeru all määratakse tegevuse nimetus. See on vajalik, et kasutada muid funktsionaalsusi, mida Cleverscript pakub.

Description veerg sisaldab üldjuhul kirjeldust tegevuse kohta või kasutaja sisendi kirjeldust sõltuvalt tegevuse tüübist.

Text veergu kirjutatakse kasutaja võimalik sisend või agendi poolt väljastatav sõnum sõltuvalt tegevuse tüübist.

2.1.1 **Sisendid ja väljundid**

Sisendi tüübiks on input ja väljundi tüübiks output. Selleks, et kasutaja sisendile määrata korrektne agendi reaktsioon kasutatakse veergu "Goto". Sisendi real kirjutatakse "Goto" veergu vastava output tüübi "Label" (nimetus). Tabel 11 illustreerib seda.

Tabel 11. Näide sisendi ja väljundi ühendamisest Goto abil.

Type	Label	Description	Text	If	Learn	Goto	Accuracy
output_start	tervitus	Esimene ...	Tere!				
input	tere	Kasutaja ütleb tere	tere			tere_vastu	75
input	ükskõik	Iga muu sisendi korral	ükskõik			uups	0
output	tere_vastu	Agent tervitab vastu	Tere sullegi.				
output	uups	Agent vabandab	Vabandust, mul pole vastust sellele.				

Peale "Tere!" ütlemist, ootab agent kasutaja sisendit. Sisendi saamisel võrreldakse seda kahe olemasoleva input reaga. Kui kasutaja ütleb midagi, mis on 75% sarnane tekstile "tere", siis loetakse Goto veerust, et on vaja käivitada output nimetusega "tere_vastu". Öeldes midagi, mis on 0% sarnane tekstile "ükskõik", käivitatakse output nimetusega "uups".

Veerg nimega "Accuracy" (täpsus) määrab, kui palju peab kasutaja sisend sarnanema tekstile. Kui täpsus oleks määratud 100% peale, siis peaks kasutaja täpselt kirjutama "tere". Kui muuta täpsus 20% peale, siis võib kasutaja sisestada "tteerrrrreee" ja see sobituks tekstiga "tere". Sisendi ja teksti võrdlemisel ignoreeritakse suur- ja väiketähti ning lausete lõpus olevaid kirjavahemärke. Jättes input real täpsuse veerg tühjaks, omistatakse, vaikimisi, selle väärtuseks 75%.

Tabel 12. Teine näide Goto kasutamisest ja blank_inputi tüüp.

Type	Label	Description	Text	If	Learn	Goto	Accuracy
output_start	tervitus	Esimene ...	Tere!				
input	tere	Kasutaja ütleb tere	tere			küs_õun	75
output	küs_õun	Agent küsib	Kas sulle meeldivad õunad?				
input	õun_j	Kasutajale meeldib	jah			õun_jah	20
input	õun_e	Ei meeldi	ei			õun_ei	20
output	õun_jah	Agent nõustub	Mulle ka!				
output	õun_ei	Agent ei nõustu	Kahju.				
output	blank_input	Vastust tühjale sisendile	Ma ei kuule sind.				

Tabelis 12 toodud näite põhjal, kui kasutaja ütleb "tere", siis agent liigub output reale nimetusega "küs_õun" ja küsib "Kas sulle meedlivad õunad?". Kui kasutaja vastab sisendiga, mis on vähemalt 20% ulatuses sarnane tekstiga "jah", siis agent vastab tekstiga, mis asetseb output real nimetusega "õun_jah". Eitava vastuse korral agent vastab output reaga "õun_ei".

Järjekord, kuidas input ja output kirjutatakse, on väga oluline. Input nimetusega "tere" on kirjutatud peale output_starti, seega see on alati aktiivne. Kasutaja võib igal hetkel vestluses öelda "tere" ja agent küsib talt uuesti õunte kohta.

Kuid input read nimetustega "õun_j" ning "õun_e" ei esine output_start rea all, vaid peale tavalist output rida nimetusega "küs_õun". Seega need kaks on ainult aktiivsed

peale seda, kui agent on käivitanud output real "küs_õun". Agent eelistab inputi, mis asub outputist all, juhul kui tal on valik, ehk mõlemad võimalikud inputid sobivad oma miinimum täpsusprotsendiga.

Kui 2 või enam inputi on vasteks ühele sisendile, siis valitakse see input, mille täpsusprotsent on suurem. Kui ka täpsusprotsent on sama, valitakse see, mis esineb esimesena tabelis.

Viimase rea nimetus tabelis 12 on blank_input. Kui kasutaja sisestab tühja sisendi või sisendi, millele ei ole input vastet, kuvatakse blank_inputi sees olev tekst.

Input read, millel puudub Goto käsklus, suunatakse output reale, mille nimetus on sama mis inputil, muudel juhtudel peaksid nimetused olema unikaalsed.

Tabel 13. Näide sisendist ja väljundist ilma Goto käskluseta.

Type	Label	Description	Text	If	Learn	Goto	Accuracy
output_start	algus	Esimene ...	Räägi minuga.				
input	tere	Kasutaja ütleb tere	tere				75
input	hüvasti	Kasutaja lahkub	kohtumiseni				
output	tere	"tere" vastus	Meeldiv tutvuda.				
output	hüvasti	"hüvasti" vastus	Nägemiseni.				20

Kuna Goto veerud on mõlema sisendi korral tühjad, siis otsitakse väljundiks output rida, mille nimetus ühtib inputi omaga.

Veel üks viis, kuidas sisendeid ja väljundeid kirjutada on "inout" tüüpi kasutades. Tabel 14 illustreerib, kuidas sisend ja väljund võivad asetseda ühel real. See meetod võimaldab väga lihtsalt ja kiirelt kirjutada teadmust Cleverscript agendile.

Tabel 14. Näide inout tüübi kasutusest.

Type	Label	Description	Text	If	Learn	Goto	Accuracy
output_start	algus	Esimene ...	Räägi minuga.				
inout		tere	Tere!				75
inout		kuidas läheb	Päris hästi.				75
inout		muu sisend	Ei saa aru.				0

Nagu näha, puuduvad inout tüüpi ridadel nimetused, sest sisendile vastav väljund on defineeritud samal real. Sisendiks Tabelis 14 on veerg "Description" ja väljundiks veerg "Text". Täpsuse protsent määratakse sisendile. Muidu töötavad samamoodi nagu output tüüpi read ehk agent väljastab vastuse peale sisendi saamist.

2.1.2 **Fraasid**

Lihtsa tegevuse nagu tervitamise kirjutamiseks, on mitmeid võimalus. Kasutaja võib kirjutada "Tere", "Jou", "Hei", "Tšau" jms. Selleks, et käsitleda kõiki võimalikke sünonüüme, ja mitte kirjutada igaühele neist sisendrida ainulaadse nimetusega, on võimalik kõik sünonüümid kirjutada vastava input rea teksti veeru alla. Seda illustreerib Tabel 15.

Tabel 15. Sünonüümide käsitlemine loeteluna.

Type	Label	Description	Text	If	Learn	Goto	Accuracy
output_start	algus	Esimene ...	Räägi minuga.				
input	tere	Kasutaja ütleb tere	tere			tervita	60
			hei				
			tšau				
			tervist				
output	tervita	Agent vastab	Tervist!				

Tekstitulp võib ulatada mitmele reale. Seda meetodit kasutades tuleks ülejäänud tulbad jätta tühjaks, kuna igale reale kehtivad samad sätted, mis on määratud esimesele input reale.

Alternatiivselt võib ridade asemel kirjutada teksti lahtrisse erinevad sünonüümid eraldades need / abil. Nt: tere / hei / tšau / tervist.

Tabel 16 tutvustab peamist põhjust, miks Cleverscripti on võimas vahend agendi loomiseks. See jätkab eelneva näite põhjal ja meetodi nimetus on phrases (fraasid).

Tabel 16. Fraaside näide sünonüümide käsitlemiseks.

Type	Label	Description	Text	If	Learn	Goto	Accuracy
output_start	algus	Esimene ...	Räägi minuga.				
input	inim_tere	Kasutaja ütleb tere	((tere))			tervita	60
			((tere)) agent				
output	tervita	Agent vastab	Tervist!				
phrase	tere	Esimene fraas	tere				
			hei				
			tšau				
			ciao				

Tabelis 16 on sisendil "inim_tere" kaks variatsiooni, "((tere))" ja "((tere)) agent". Kahekordsete sulgudega ümbritsetud sõnad on viited fraasile. "((tere))" viitab tabelis allpool asetsevale tüübile phrase, mille nimetus on tere. See fraas sisaldab 4 võimalikku sünonüümi, mida kasutaja võib kasutada, et teretada. Fraaside üks

omadusi on multiplikatiivsus. Antud näites "((tere))" ja "((tere)) agent" korrutatuna 4 variatsiooniga, annavad vaste kaheksale võimalikule sisendile, mida kasutaja võib öelda.

Fraasid võivad sisaldada teisi fraase ja olla rekursiivsed ehk viidata ka endasse tagasi. Selliseid võimalusi kasutades on potentsiaal luua miljoneid variatsioone, mida agent suudab töödelda kiirelt ja efektiivselt [18]. Lisaks saab fraase kasutada nii sisendis kui ka väljundis. Tabel 17 illustreerib pikemalt fraaside kasutamist.

Tabel 17. Inglise keelset vestlust modelleeriv näide fraaside laiemast kasutusest [18].

Type	Label	Description	Text	If	Learn	Goto	Accuracy
output_start	welcome	Esimene öeldud asi	Hi there!				
input	asks_name	Kasutaja küsib nime	((what is)) your name				40
			((what are)) ((you)) ((called))				
			Who are ((you))				
output	asks_name	Agent vastab	My name is ((bot_name)).				
			I am called ((bot_name)).				
			I am ((bot_name)).				
			((bot_name)).				
phrase	what is		((what)) is				
			((what))'s				
			((what))s				
phrase	what are		((what)) are				
			((what))'re				
			((what))re				
phrase	what		what / wot / wat / whaat / wut				
phrase	you		you / u / ya				
phrase	called		called / named				100
phrase	bot_name		William				

Tänu fraasidele aktsepteeritakse 87 erinevat sisendit ja see on kirja pandud 12 rea abil. Väljundi genereerimiseks on vastavalt 4 varianti, mis on kirja pandud 5 rea abil.

Näites on fraasi "called" täpsuseks määratud 100%. See meetod on eelkõige kasulik, kui mingi osa kasutaja sisendist peab väga täpne olema, näiteks paroolide korral. Vaikimisi on fraaside täpsuse väärtus 30% [18].

2.1.3 **Wildcard ehk tundmatu sisend**

Sarnaselt AIML keele eripärale, on ka Cleverscriptis kasutuses wildcardid ja nende funktsioon on sisuliselt sama.

Cleverscriptis on kahte tüüpi wildcarde. Esiteks, alakriips "_" on vasteks ükskõik millisele ühele tähele. Teiseks, tilde "~" on vasteks nullile või enamale märgile.[18] Nt: Kui kasutaja sisend oleks "tere", siis seda saaks kirjeldada kujul "t__e" ning sobiks pea 100% täpsusega, kuna alakriipse eeldatakse olevat tähtedeks e ja r. Sarnaselt sobiks ka kirjutada "t~", mis samuti sobituks sisendiga pea 100%, kuna tilde võtab enda rolliks "ere" osa. Kasutades wildcarde on aga täpsus alati alla 100%, kuna wildcard vasteid ei peeta kunagi nii tõesteks vasteteks, nagu oleks “tere”=”tere". Seetõttu ei saa ka sisendi täpsuseks, mis kasutab wildcarde, määrata 100%. Tabelis 18 on näide tilde kasutamisest sisendis.

Tabel 18. Wildcardi kasutuse näide.

Type	Label	Description	Text	If	Learn	Goto	Accuracy
output_start	algus	Esimene ...	Arva, mis on mu lemmikvärv?				
input	õige	Kasutaja arvab õigesti	~ ((bot_värv))				90
input	muu	Vale pakkumine	muu sisend				0
output	õige	Arvati õigesti	Jah! ((bot_värv)) on mu lemmikvärv.				
output	muu	Vale vastus	Proovi uuesti arvata.				
phrase	bot_värv	Lemmikvärv	sinine				

Esialgu küsib agent kasutaja käest oma lemmikvärvi. Ilma wildcardideta oleks vaja ennustada kõiki võimalikke viise, kuidas kasutaja võiks vastata sellele küsimusele. Nüüd aga, kuni kasutaja sisend lõpeb sõnaga “sinine”, loetakse see õigeks vastuseks. Selline meetod on just hea spetsiifiliste sõnade otsimiseks sisenditest.

2.1.4 **Õppimine ja mälu kasutamine**

Ainuüksi sisendite, väljundite ja fraasidega on võimalik luua üpris keerukas agent, mis vastab kasutaja usutavalt ja annab kasulikku informatsiooni. Cleverscript agent, suudab kasutaja kohta õppida ja õpitud informatsiooni hiljem kasutada.

Õppimine on realiseeritud muutujate kasutamisega ja seda illutreerib Tabel 19.

Tabel 19. Learn veerg, ehk õppimine.

Type	Label	Description	Text	If	Learn	Goto	Accuracy
output_start	algus	Esimene asi mida agent ütleb.	Räägime.				
input	tere	Kasutaja tervitab	((tervitus))			vastus	90
output	vastus	Agendi vastus	Tere!				
phrase	tervitus	Tervituse fraasid	tere		sõbralik=jah		
			hei		sõbralik=vist		
			olgu		sõbralik=ei		

Tervituse fraasi “tere” kõrval on kirjas “sõbralik=jah”. Kui kasutaja sisendiks juhtub olema “tere”, luuakse muutuja “sõbralik”, mille väärtuseks omistatakse “jah” ehk kasutaja on sõbralik. Öeldes “hei” või "olgu", määratakse muutuja väärtuseks “vist” või "ei". Olenemata tuvastatud fraasist, vastab agent alati "Tere!".

Öeldes midagi, mis ei leia vastet fraasis olevale loetelule, naaseb agent output_start reale ja tervitab kasutajat. Selle tegevuse tulemusena kustutatakse kõik olemasolevad muutujad koos väärtustega. Seetõttu oleks hea kasutada väljundit nimetusega blank_input, et vältida seniilsust.

Üks meetod, salvestatud andmete kasutamiseks vestluses, on If veergu kasutades ja seda illustreerib Tabel 20.

Tabel 20. Agendi poolt õpitud andmete kasutamine vestluses.

Type	Label	Description	Text	If	Learn	Goto	Accuracy
...	...	...	...				
input	küsib	Kasutaja küsib	~ ((arva))			vastus	30
output	vastus	Agendi vastus	Sa oled sõbralik.	sõbralik=jah			
			Sa oled vist sõbralik.	sõbralik=vist			
			Ma arvan et sul on halb tuju.	sõbralik=ei			
phrase	arva	Fraasid	arvamus / arvad				

Arvestades Tabeli 19 ja 20 koodi koos, võib kasutaja peale esialgset tervitust küsida agendi käest "Mida sina arvad?" ja agent avaldab oma arvamuse, vastavalt sellele, milline väärtus on muutujale "sõbralik" omistatud.
"If" veerg on loogikatehete jaoks. Peale sisendi tuvastamist, minnakse väljundisse nimetusega "vastus" ja siis hakkab tööle If veerus olev loogika avaldis. Kui Learn veerus loodi muutuja ja anti sellele väärtus, siis If veerus kontrollitakse muutujate väärtust vastavalt programmeeritud kriteeriumile. If tingimused võivad ka

keerulisemad olla, kontrollides mitmete muutujate väärtusi korraga ja lubades kasutada >, < ja muid loogika operaatoreid [18].

Kui üks väljundis olevatest If veergudest osutub tõeseks, väljastatakse tõese loogikatehte real asuv tekst.

Teine meetod kasutamaks salvestatud andmeid vestluses on kirjutada väljundi rea teksti sisse süntaksiga: `$muutujaNimi$`. Antud meetod on kuvatud Tabelis 21.

Tabel 21. Salvestatud andmete kasutamine väljundi sees.

Type	Label	Description	Text	If	Learn	Goto	Accuracy
...	...	...	...				
input	küsib	Kasutaja küsib	~ ma ~ sõbralik ~			vastus	30
output	vastus	Agendi vastus	Sõbralik? $sõbralik$.				

Arvestades Tabeli 19 ja 21 koodi kui ühte tervikut, võib kasutaja küsida peale tervitust "Kas sa arvad, et ma olen sõbralik inimene?" ja agendi vastus võib sisaldada ühte väärtust kolmest võimalikust nt "Sõbralik? vist."

2.1.5 **Emotsioonid ja reaktsioonid**

Cleverscripti on sisse ehitatud käsklused emotsioonide ja reaktsioonide mõjutamiseks, mis väljenduvad visuaalse avatari näol. Emotsioonid ja reaktsioonid käivitatakse spetsiaalsete Clever Data (Nutikad Andmed) käskluste abil väljundrea tekstis [18].

Näide süntaksist:

```
$clever_output$ Reaction: $reaction$ with tone $reaction_tone$. Emotion: $emotion$ with tone $emotion_tone$.
```

Ülalolev süntaks kirjeldab avatari näoilme muutust. Esialgu kuvab avatar reaktsiooni ilmet, peale mida jääb avatar kuvama mingit emotsionaalset seisundit.

3 PERSONALITY FORGE AGENT

Benji Adams on www.personalityforge.com saidi ja AI Engine loojaks. Tema esimene kokkupuude tehisintellektiga oli 90-datel programmiga Eliza. See programm inspireeris teda nii väga, et ta tahtis luua oma TISi. Aastaid hiljem avastas ta enda jaoks lihtsa programmeerimiskeele ColdFusion, mille abil ta hakkas arendama AI Engine. Tema esimeseks tupikuks oli ebapiisav teadmine inimsuhtlusest ja kuidas seda võiks modelleerida oma programmis. Peale intensiivset otsimist, leidis ta grupi inimesi Princetoni Kognitiivse Teaduse osakonnast, kes olid loonud semantilise sõnade võrgustiku nimega WordNet. Need inimesed olid loonud leksikograafilise viitesüsteemi, mis põhines tolle aja psühholingvistilistel teooriatel, kuidas me (inimesed) mõtleme ja mõistame sõnu. Tänu nende inimeste teadustööle, oli Benjil enam kui vajalik teadmine, kuidas oma TIS edasi arendada [19].

AI Engine, mis on iga PF agendi südamikuks, kasutab nii loomuliku keele automaattöötlust (NLP) kui ka juhtumi-põhist loogikat (CBR) sisendi töötlemisel. Esiteks programm kasutab NLP, et parsida lauseid ja tükeldada need osadeks, et määratae sõnade seos teiste sõnadega lauses. Seejärel CBR meetodiga sobitatakse lause tükikesi olemasolevate võtmefraasidega kuni leitakse sobivate võtmefraaside hulk. Siis arvutatakse sellest hulgast parim võtmefraas, millele vastatakse. AI Engine sisene skriptimise keel AI Script võimaldab agendi loojal omada suuremat kontrolli agendi käitumise üle [20].

Selleks, et agenti luua PF keskkonnas on vaja registreerida omale konto, peale mida on võimalik hakata looma enda äranägemise järgi jutuagenti. Teadmusbaasi loomiseks ei ole vaja omada programmeerimisoskusi ja keskkonnas on lihtne kasutusliides sisendite ja väljundite kirjutamiseks. Peatükk 3.1 käsitleb täpsemalt seda teemat.

Loodud agendile on võimalik lisada ka pluginaid, mis kujutavad endast andmebaase sõnadest. Iga plugin sisaldab sõnu, mis on seotud kindla teemaga. Olemasolevad

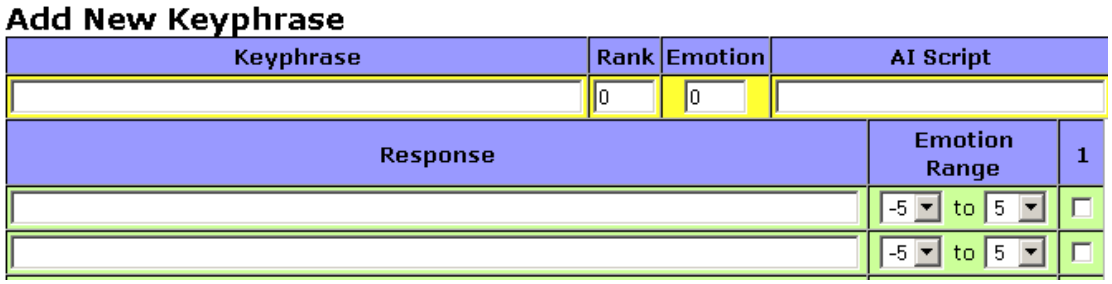
pluginad on loodud PF looja või teiste PF kasutajate poolt ja neid saab ka ise koostada ning jagada teistega.

Valmis agenti saab kasutada PF leheküljel või integreerida agent oma leheküljele kasutades APIt [21].

PF keskkond on üldiselt tasuta, aga API kasutamisel on sõnumite edastamisel kuulised piirangud, mida saab suurendada, makstes kuutasu. Tasuline on ka agendile visuaalse avatari lisamine.

3.1 Teadmusbaasi loomine

Agendi teadmusbaasi loomine PF keskkonnas käib läbi võtmefraaside ja nendele sobivate vastuste lisamise kasutades kasutajaliidest mis on kujutatud Joonisel 1.



Joonis 1. Liides mille abil lisatakse agendile teadmust.

Joonisel 1 olevate lahtrite ning märksõnade selgitused:

- **Keyphrase** - võtmefraas, on sõnade jada, mille agent "ära tunneb". Agent otsib sisendist neid fraase ja kui sisendis leidub mõni fraas, mis on kirjas, käivitatakse sobiv vastus [22].
- **Rank** - järk. Igal võtmefraasil on järk, mis aitab määrata milline võtmefraas valitakse kui agent leiab mitu võtmefraasi ühest sisendist. Enamikel võtmefraasidel võib olla järguks 0 kuna AI Engine teeb üpris head tööd määramaks, milline vastus on parim valik, aga kohtades kus on vaja kohendada vastuste tasakaalu on võimalik järku muuta [22].
- **Emotion range** - emotsioonide ulatus. Agentidel on aktiivne emotsionaalne elu, milles enamus on sisse ehitatud AI Engine-isse ja see läbi on võtmefraaside emotsiooni aste määratud 0. Määrates vastustele ulatuse, saab

panna vastuse sõltuma agendi tujust. Ulatus -5 kuni 5 vastab alati, kui ulatus määrata ainult positiivsele, siis agent vastab ainult kui tal on piisavalt positiivne tuju [22].

- **Response** - vastused sisendile (võtmefraasile). Peale kasutaja sisendist kõige tähtsama võtmefraasi leidmist, valitakse juhuslikult üks vastustest, mida agent ütleb. See juhuslikkus on sisse programmeeritud, et vältida kordumist, mistõttu on parem kui igal võtmefraasil on rohkelt erinevaid vastuseid [22].
- **Plug-in** - lihtsalt plugin. Annab võimaluse kasutada eelnevalt teiste kasutajate looduid spetsiifilisi andmebaase, et lihtsustada sünonüümidele ja teemadele vastavate sõnade kattuvust. Plugina kasutamist saab määrata agendi sätetest ja süntaks plugina kasutamiseks võtmefraasi või vastuse sees oleks järgnev: `(pluginaNimi)`.

  Nt plugin loomade teema kohta “(animal)”. Kui kasutada seda pluginat vastuse väljal, siis tuleks kirjutada "I like (animal)s" ja kasutajale esitatakse vastus kujul “I like eagles” või “I like horses”. Sõna, mis valitakse vastava plugina andmebaasist, on juhuslik [23].
- **AIScript** - PF tehisintellekti skriptimise keel. See võimaldab suurendada kontrolli mälu, emotsioonide ja väljundite üle. Selle abil on võimalik kasutaja kohta infot salvestada mällu ja hiljem kasutada neid “mälestusi”. Samuti saab lisada tingimusi väljunditele, häälestada emotsioonide esinemist ja muud [24].
- **1** - tulp mille märgendiks on 1, viitab valikule, kus antud vastus esitatakse ainult üks kord igale kasutajale. Ei ole loogiline küsida kasutajalt, kas tal on õde või vend, ja siis järgmine päev küsida sama asja. Kui inimene peaks küsima mitu korda kelleltki “Kas sul on õde või venda?”, arvaks, et tal on mäluga probleeme. Kõik vastused, aga ei tohi olla märgitud “1”, vähemalt üks vastus peab olema vaba sellest märgendist [22].

3.1.1 **Võtmefraasid ja vastused**

Iga kasutaja sisend, mis saadetakse PF agendile läbib esiteks protseduuri, mida nimetatakse inglise keeles Pre-Processing (eeltöötlemine). Sellel sammul muudetakse mitmeid asju originaalsisendis.

Esiteks, kõik lühendatud sõnad (can't, don't, they're jne) laiendatakse samaväärseteks terviklikeks sõnadeks (can not, do not, they are jne). Seega tuleb veenduda, et võtmefraasid ei sisalda kontraktsioone. Mõnikord võib fraas tunduda keeleliselt veidrana, nagu "why do not you" aga agendi jaoks on see sama, mis "why don't you". Seejärel, artikleid lihtsustatakse. Kõik "an" sõnad asendatakse "a".

Järgmisena asendatakse kõik slängid nende tegelike vastetega. Näiteks, "ya" muudetakse "you". Siis kontrollitakse õigekirja. Hüüatused nagu "hahahahaha", "oooooh" eemaldatakse, kuid nende tähendus märgitakse üles. Kuna kõik see on optimiseeritud inglise keele jaoks, siis on PF poolt pakutud agendid rangelt inglisekeelsed [23].

Iga võtmefraas kirjutatakse eraldi koos endale omaste vastustega ja muude sätetega. Kuna inimesed võivad ühemõttelist lauset kirjutada erinevatel viisidel, siis võib kirjutada mitu võtmefraasi järjest, kasutades koma eraldusmärgina. Joonis 2 näitab sarnaste võtmefraaside loetelu ja ükskikut võtmefraasi.

Keyphrase	Rank	Emote	Response	1
do you like, do you desire	0	0	Let me think... Yes, I do.	
			Not a bit. Do you?	
			Not as much as my sister does. Do you have a sister?	•
do you seek	0	0	Well, I like it, but I dont really seek it out.	
			goto do you like,do you desire	

Joonis 2. Võtmefraaside ja vastuste näide.

Joonises 2 on ühes vastustes kasutatud märgendeid
. Sellise vastuse kuvamisel näeb kasutaja
 märgiste asemel reavahetusi

Näide võimalikust vestlusest Joonise 2 fraaside ja väljundite põhjal:

```
Kasutaja: Do you like waffles?
Agent: Let me think...


       Yes, I do.
Kasutaja: Do you seek the best waffles?
Agent: Not as much as my sister does. Do you have a sister?
```

Joonisel 2 on viimasel real kirjas "goto do you like,do you desire". Jooniselt on ka näha, et esimesed võtmefraasid on "do you like, do you desire". Selleks, et ühes võtmefraasis kasutada mingi muu võtmefraasi vastuseid on käsklus "goto". Goto sarnaneb nii nimelt kui ka tegevuse poolest Cleverscripti samanimelise veeruga.

Suunamaks ühest võtmefraasist teise, tuleks esialgse võtmefraasi vastusesse kirjutada järgneva süntaksiga käsklus: `goto võtmefraasiNimi`

3.1.2 **Wildcard**

Eelnevalt AIML ja Cleverscript peatükkides on juba kirjas wildcard mõistest ja funktsioonist, PF kasutab wildcardi samal otstarbel. PFil on lisaks tüüpilistele wildcardidel võimalik kasutada sõnatüübi wildcarde ja plugin-wildcarde, mis sarnanevad funktsionaalsuse poolest Cleverscripti fraasidega.

Järgnevad on võimalikud wildcardid [24]:

- **Pehme wildcard *** - leiab vaste kas ühele või tühjale sõnale. Näiteks: Võtmefraas "do you * like" tuvastab sisendid "Do you like?", "Do you really like?".
- **Range wildcard (*)** - peab leidma vaste vähemalt ühele sõnale. Näiteks: Võtmefraas "my (*) likes" tuvastab sisendid "My dog likes.", "My big yellow fern likes.", aga ei tuvasta sisendit "My likes and dislikes are well known.".
- **Sõnatüübi wildcardid** - võtmefraasid ja vastused võivad sisaldada nii ühe kui mitme sõnatüübiga wildcarde, mis koosnevad erilistest sõnadest või sõnade gruppidedest. Nt võtmefraas "you are (adjective)" tuvastab sisendi "You are silly!" ja "You are brilliant!", aga mitte "You are horse!", kuna hobune on nimisõna. Tabelis 22 on välja toodud võimalikud sõnatüübi wildcardid ja sobivad sõnad, mis tuvastatakse.

- **Loetelu wildcard** - Sulgudesse võib kirjutada vajalikud sõnad, mida vaja tuvastada, kasutades eraldusmärgiks |. Nt võtmefraas "are you a (guy|man|dude|boy)" tuvastab "Are you a guy?", "Are you a man?" jne aga ei tuvasta "Are you a giraffe?", kuna sellist sõna ei ole wildcardi loetelus kirjas.
- **Plugin wildcardina** - Kirjutades pluginaid võtmefraasidesse, käituvad need nagu loetelu wildcardid. Need wildcardid leiavad sisendist ükskõik millise vaste, kui see on selle plugina listis. Nt võtmefraas "(badweather) * coming" tuvastab "There is a thunderstorm coming soon."

Iga wildcardi jaoks, mis on võtmefraasis, genereeritakse numbriga võti, vastavalt esinemise järjekorrale, kujul (key#), kus # on mingi arv hulgast N^+. Neid võtmeid saab kasutada vastustes ja nad sisaldavad kindlat tükikest kasutaja sisendist.

Võtmefraas "i (*) this (*) is" ja vastuse "I also (key1) that this (key2) is (postkey). Great minds think alike!" puhul võib agendi ja kasutaja vahel esineda järgnev dialoog:

```
Kasutaja: I think this book about quantum entanglement is
really tied up.
Agent: I also think this book about quantum entanglement is
really tied up. Great minds think alike!
```

Antud dialoogis (key1) on "think" ja (key2) on "book about quantum entanglement".

Dialoogis on kasutusel (postkey). Tegemist on genereeritud pluginiga lause osast, mis esineb peale võtmefraasi vaste leidmist, ehk antud näites sõna “really tied up". Sarnane plugin genereeritakse lause osale, mis esineb enne võtmefraasi tuvastamist ja selleks on (prekey).

Tabel 22. Kasutatavad sõnatüübi wildcardid.

Üksikud sõnad	Näited	
(noun) nimisõna	paper	muffin
(verb) tegusõna	slide	walking
(adj) või **(adjective)** omadussõna	silly	yellow
(adv) või **(adverb)** määrsõna	quietly	yesterday
(prep) eessõna	from	between
(artops) artikkel või asesõna	the	your
Kombinatsioon	Näited	
(adjnoun)	silly paper	yellow muffin
(adjartnoun)	the silly paper	your yellow muffin
(np) või **(adjartnounprep)**	the silly paper by the house	your yellow muffin on a table
(verbadv)	walking quietly	coughed loudly
(vp) või **(verbadvprep)**	ran very quickly over	fall down suddenly

3.1.3 **Algoritm vastuse valimiseks**

Nagu eelnevalt mainitud, on igal võtmefraasil oma järk. See järk määrab ära, kui tähtis on fraas. Agent vastab võtmefraasile, millel on kõige kõrgem järk. Lisaks on veel faktoreid, mis automaatselt mõjutavad iga võtmefraasi lõplikku järku, kui see leitakse sisendist.

Algoritmi kirjeldus, mis arvutab igale tuvastatud võtmefraasile järgu [21] on järgnev:

- **Sõna järjekorra boonus** - Võtmefraasid järjestatakse automaatselt vastavalt lauses esinemise koha järgi. Kui võtmefraas esineb lause alguses, on sellel fraasil kõrgem järk kui teistel. Näide: Võtmefraaside “what” ja “my” olemasolul, kus mõlema võtmefraasi järgud on looja poolt määratud samaks, ja sisendi “What is my favorite drink?” sisselugemisel ja töötlemisel, käivitatakse “what” võtmefraas. Fraaside järke suurendatakse järgnevalt: Lauses esimesel kohal olev fraas saab +12, teisel kohal +6, kolmandal +4, neljandal +3 ja alates viiendast kuni kaheksandani +2 ning kõik edasised esinevad fraasid saavad +1 oma järgule. Kui fraasil “my” oleks määratud järguks 7, siis sama sisendi “What is my favorite drink?” tulemusena

käivitatakse “my” võtmefraasis leiduv vastus, kuna selle fraasi järguks oleks 13, mis on suurem kui “what”, millel on järguks 12.

- **Lause järjekorra boonus** - Üldiselt on parim vastata kõige viimasele lausele. Viimases lauses olevad fraasid saavad +0 järgule, eelviimane -4, kolmas viimasest saab -8 järgule jne.
- **Võtmefraasi pikkuse boonus** - Pikem fraas on üldiselt paremini vastavuses kui lühike fraas, seetõttu võtmefraasi sõnade pikkus lisatakse lõplikku järku. Kui fraasis on kasutusel *, siis järk saab -2, kuna täpne vaste on parem, kui osaline vaste. Teistel võtmefraaside pluginatel on kohandatud pikkuse boonus.

Agent salvestab listi milliseid vastuseid ta kellele on öelnud ja juhuslikult valib vastuseid, kuni kõik vastused on ühe korra valitud [21]. Seejärel ta tühjendab listi ja alustab uuesti listi täitmist. See ennetab, et sama vastus esineks kahel järjestikusel korral ja loob igale vastusele madalama esinemissageduse.

Järgu uuendamisega ja öeldu mäletamisega, on PF agendi vestlused küllaltki ettearvamatud.

3.1.4 **AI Script**

AI Script on PF agentide sisemine programmeerimiskeel. AI Scripti abil saab programmeerida agendi mälu kasutust, emotsioone ja agendi näoilmet. AI Script kirjutatakse vastavasse AI Script lahtrisse nagu on näha Joonis 1.

Skripti süntaks on järgnev: <?PF statement; ?> või <?PF statement; statement; ?> mitme lause jaoks. Iga lause lõpeb semikooloniga.

Agendil on kahte sorti mälu. Esimene mälu "Standard Memory", mis koosneb muutujatest, nendele omistatud väärtustest ja selles olevat informatsiooni seostatakse teiste kasutajaga (nt räägitud teemad). Teine on "Self Memory", mis sisaldab agendi omadusi ja kogemusi, mis on kogutud erinevatelt kasutajatelt.[24]

AI Scriptide kirjutamine:

- **"Standard Memory" süntaks** -

  Lause süntaks: <?PF default "mälusisu" as "mälunimi"; ?>

  Näide: <?PF default "Nick" as "nickname"; ?>

- **"Self Memory"süntaks -** <?PF self: default "mälusisu" as "mälunimi"; ?>
- **Emotsiooniseisundi muutmine -**

  Lause süntaks: <?PF emotion: emotsioonimuutus; ?>

  Näide: <?PF emotion: 1; ?>
- **Näoilme muutmine -** Lause süntaks: <?PF express: väljendus; ?>

  Näide: <?PF express: amused; ?>

Millisesse mälutüüpi andmed salvestatakse, määrab "self:" kirje olemasolu enne "default" kirjet. "mälusisu" on väärtus mille saab loodav muutuja ehk "mälunimi".

Agendi emotsioon muudetakse vastavaks "emotsioonimuutus" väärtusega. Võimalikud väärtused on -5 kuni 5. Agendi emotsiooni muutmiseks ei pea aga ilmtingimata kasutama AI Scripti, vaid saab ka määrata Joonisel 1 näidatud liidese abil. Näoilmete võimalikud väljendused on: normal, happy, angry, averse, sad, evil, furning, hurt, surprised, insulted, confused, amused, asking. Kui agent aktiveerib vastuse võtmefraasile, millel on kirjas emotsiooniga seotud käsklus, muudetakse agendi visuaalset väljanägemist, kui jutt toimub Flash jutuaknas [24].

Vastuse sõltuvust agendi tujust saab määrata kasutades loogika kirjutusviisi: <?PF if emotion > 3; ?>

4 AGENTIDE VÕRDLUSANALÜÜS

Käesolevas peatükis käsitleb autor agentidega seonduvaid tugevusi, nõrkusi, toob välja erinevused ja probleemid ning klassifitseerib agentide omadusi.

4.1 AIML tugevused ja nõrkused

+ Programmeerimisoskusi ei ole vaja, aga omades teadmisi HTMList või XMList kiirendab AIML omandamist ja arusaamist.
+ Elementide paiknemise loogika ja struktuur on ühtne iga sisendi ja väljundi kirjutamise jaoks.
+ Wildcardid aitavad laiendada sisendi sobitumist suurema variatsiooni kasutaja sisenditega.
+ Kasutaja sisendi osasid saab kasutada väljundis või salvestada sisendites leiduvat informatsiooni ja seda hiljem kasutada muudes väljundites.
+ Sisendite kirjutamine on lihtne, pole tõstutundlik ja vaja pole lisada kirjavahemärke. Agendi sisendid ja väljundid võib kirjutada mistahes inimkeeles.
+ Programme, mis interpreteerivad AIML keelt on palju ja neid on loodud kasutades erinevaid programmeerimiskeeli. Lihtsuse tõttu poleks raske ka ise luua programmi, mis töötleks AIML koodi.
− Kuna AIMLil on piiratud sisendi-tuvastus võimed, siis on vaja luua suur hulk kategooriaid, et agendi vastused oleksid usutavad ja inimlikud. Selle jaoks oleks kategooriate arv vajalik kümnete tuhandete ringis.
− Kasutaja sisend peab vastama mingile kirjasolevale sisendmallile, et agent saaks vastata sellele. Pole olemas sünonüümide loetelusid nagu PF või Cleverscriptis. Iga võimaliku sünonüümi käsitlemiseks on vaja kas kirjutada

täiesti uus kategooria või kasutada <srai> elemente rekursiivseks otsimiseks muudest kategooriatest.

- Märgendid on kohmakad ja võtavad väga palju ruumi. Dokumendis liiklemine muutub väga tüütuks ja raskeks visuaalselt kui see sisaldab palju kategooriaid.
- Sisendites ei arvestata kirjavahemärke ega tõstutähti, mille tõttu kaob sisenditest emotsioonide tuvastamise võimalus.
- Inimestele ei meeldi, kui midagi korratakse ja seda pannakse tähele. Järestikuste ühesuguste sisendite järel vastab agent alati samade vastustega [25].
- Emotsioonide väljendamiseks pole muud viisi, kui kirjutada vastuseid kindla tooni ja stiiliga.

4.2 Cleverscript tugevused ja nõrkused

+ Teadmust kirjutada on väga lihtne, isegi et lihtsam kui AIMLis, ja tabeli sisu defineerivad väljad võtavad enda alla ainult 1 rea.
+ Sisendite ja väljundite paigutamisel kindla järjekorraga on võimalik luua vastuseid, mida agent vastab ainult siis, kui eelnevalt on kasutaja öelnud õige sisendi.
+ Sisenditele, fraasidele saab määrata täpsuse protsendi, mis aitab suurendada võimalust sisendi aktsepteerimiseks kui kasutaja peaks kirjutama kirjavigadega.
+ Agent suudab õppida. Learn veeru kasutamisel saab luua muutujaid vajalike väärtustega, mida hiljem kasutada muudel otstarbetel.
+ Agent kasutab loogikat. Vastuste valikut saab suunata kasutades loogikavalemeid.
+ Agendil on emotsioonidega ja reaktsioonidega seotud käsklusi, mida väljendab visuaalne avatar. Avatari kasutamine on aga valikuline.

+ Kõnetuvastus ja kõnesünteesi moodulid on kasutatavad avatariga. Eelkõige on see mugav, kuna siis ei pea kirjutama agendile, vaid saab rääkides mikrofoni, anda edasi oma sõnumi.
+ Nagu ka AIMLis on wildcardide kasutamine suvalise sisendi katmiseks.
+ Fraasid on väga võimas, kerge ja kiire meetod, et katta suurt hulka kasutaja sisendeid, kas siis sünonüümide näol või lausemallide kujul.
+ Suhtlus agendiga saab toimuda igas inimkeeles.
− Kasutaja sisendit töödeldakse ja tõstutundlikkus ei oma tähtsust nagu ka kirjavahemärgid. Kaovad kasutaja emotsioonide tuvastamine nagu nt "KARJUMINE".
− Fraaside rekursiivsel kasutamisel võib esineda tulemusi, milles aktsepteeritakse absurdne sisend ja antakse ebakohane vastus. Tuleb olla väga tähelepanelik fraaside kasutamisel.
− Cleverscriptis on ka keerukamaid kasutusi loogikatehete jaoks, millest arusaamiseks on vaja olla kogenud mõne programmeerimiskeelega.
− Ainukene võimalus kasutada Cleverscripti väljaspool nende lehekülge on API kaudu.
− Emotsioonid ja reaktsioonid on mõeldud ainult koos avatari kasutamisega.

4.3 PF tugevused ja nõrkused

+ Teadmuse modelleerimine võtmefraaside kaudu, millele agent reageerib. Erinevalt AIMList ja Cleverscriptist, ei pea kirjutama tervet lause sisu. Tänu sellisele ehitusele, on kasutajate sisendeid väga lihtne kirjutada.
+ Kasutaja sisendite eeltöötlemise algoritm lihtsustab veelgi sisendite kirjutamist kuna annab kindla reeglistiku sõnade kirjutamise stiilist.

+ Agent on ettearvamatu, vastates erinevalt, olenevalt sisendis esinevate võtmefraaside järkudest ja ei korda ennast, kuna mäletab, mida on eelnevalt öelnud.
+ PF looja on suurt rõhku pannud emotsioonide kasutamiseks agentides, et luua võimalikult inimlik agent. Võtmefraasidele saab määrata emotsioonilisi väärtusi ja vastused saab panna sõltuma agendi emotsioonilisest seisust.
+ Agent õpib oma vestluskaaslaste kohta ja seostab nendega seda informatsiooni. Õppimine ei piirdu ainult kasutaja-spetsiifiliste andmetega, aga ka kogemustega, mida ta võib kasutada suhtlemisel teiste kasutajatega.
+ Wildcardide käsitlemiseks on palju võimalusi ja lihtsustab pikemate võtmefraaside kirjutamist ning võtmefraaside arvu samamõtteliste lausete jaoks.
+ Pluginad suurendavad samuti sobivate sisendite ulatust kui ka väljundite ettearvamatust, sest väljundis valitakse juhuslik sõna plugina andmebaasist. Cleverscripti fraasidega.
− Suurim nõrkus on optimiseeritus ainult inglise keele jaoks.
− Paarisõnalised fraasid ja järkude määramine fraasidele, võivad tekitada imelikke vastuseid, mis ei taba sisendi mõtet.
− Avatari reaktsioonid on väga piiratud ja kuvatud tegelane on koomiksi-stiilis.

4.4 Agentide üldised sarnasused

Igal agendil on meetodid tundmatu teksti käsitlemiseks wildcardide abil. Olemas on meetodid andmete salvestamiseks ja kasutamiseks õiges kontekstis, millega imiteeritakse inimese mälu.

Vastused sisendile on reegli-põhised ehk igale sisendile on määratud kindlad vastused, agendid ei genereeri ise juhuslikke vastuseid.

Teadmuse kirjutamine on lihtne ja loogiline. Kirjutamist on lihtsustatud ka asjaoluga, et kasutaja sisendeid töödeldakse mingi algoritmiga enne, kui hakatakse vasteid otsima.

Cleverscript ja PF sisaldavad meetodeid nagu Phrases ja Plug-in, et lihtsustada tohutult sünonüümide ja sama-mõtteliste lausete erineva kirjapildi katmist.

AIML ja PF kasutavad sarnaseid võtteid, et kasutaja sisendit, ümberkirjuatmise teel, kasutada vastustest, vastavalt <star> ja (key) meetodite abil.

4.5 Agentidevahelised erinevused

PF on teadmuse kirjutamise poolest kõige parem. Kui Cleverscript ja AIML nõuavad kogu sisendi korrektset modelleerimist, siis PF käsitleb kindlaid võtmesõnu. Nt on võimalik vaid tähtsad määrsõnad, tegusõnad, omadussõnad vms kirjeldada fraasis ilma ebavajalike eessõnadeta ja muu "täitega". Kui PF otsib fraasi "I like", siis Cleverscriptis ja AIML tuleks sama kirjutada kasutades wildcarde: "* I like.", "I like *", "* I like *".

AIMLil on keeruline tegeleda sünonüümidega. <srai> kasutamine sellisel otstarbel tähendab, et väga palju kategooriaid on vaja juurde kirjutada ning iga kategooria koosneb paljudes elementidest. Samuti on raskusi <set> ja <get> kasutamisega suurtes teadmusbaasides, kuna koodi tuleb palju kirjutada. Millegi muutmiseks on vaja see koodist üles leida ja see on aeganõudev.

Kui idee teadmuse kirjutamiseks on sama (mingile sisendile leidub mingi väljund), siis struktuur ja põhimõtted, mille järgi need kirjutatakse, on erinevad.

AIMLi emotsioonilised omadused on väga nõrgad võrreldes teistega. Seevastu PFil on igale asjale võimalik anda emotsiooniline muutuja.

PFil on parim algoritm, mille eesmärgiks on muuta agent võimalikult ettearvamatuks oma vastuste poolest, nii nagu inimesed, ja takistada imelikku kordamist, mida inimesed üldjuhul ka ei tee.

Cleverscriptis saab agendi juttu muuta privaatseks, ehk osad vastused öeldakse ainult siis, kui eelnevalt on õigeid asju küsitud/räägitud. AIML aga räägiks kõik oma "saladused" välja.

PF jääb alla teistele agentidele oma keelelise piirangu tõttu, aga on parim viis modelleerida emotsionaalset agenti, kelle vastuste stiil sõltub eelnenud interaktsioonidest. Solvangute puhul tuju langeb, komplimentide puhul tuju tõuseb ja

vastused on avatumad. Vastavalt tujule võivad ka osad vastused muutuda kättesaamatuteks, kui inimene ei muuda oma sisendite tooni agendi suhtes.

4.6 Omaduste klassifitseerimine

Järgnevad tabelid klassifitseerivad agentide omadusi. Autor on omadused jaganud kolme suuremasse kategooriasse. Nimelt inimlikkusega seonduvad omadused, suhtlemisega seonduvad omadused ja masinatele kohased, tehnilised omadused.

Tabel 23. Inimlikkus agentides.

Omadus	AIML	Cleverscript	Personality Forge
Õppimine ja mäletamine	<set> ja <get> meetodid	Learn ja If veergude kasutamine	Self ja Default Memory kasutamine
Sisendi kordus	Juhuslikkus ilma kontrollita.	Ettearvamatus fraaside kasutamisel väljundis.	Algoritm mis takistab vastuste kordamist.
Privaatsus	Puudub	Mõned vastused ei ole aktiivsed kui puudub eelnev vajalik interaktsioon.	Vastuste kättesaadavus sõltuv emotsionaalsest seisundist.
Väljendusrikkus	Teostatav ainult kirjastiiliga.	Kirjastiil ja avatari kasutamine	Kirjastiil ja avatari kasutus.
Kogemused	Kogemused piirduvad ühe vestluse raamidesse.	Kogemused piirduvad ühe vestluse raamidesse.	Kogemused ühest vestlusest on kasutatavad edasistest vestlustes.

Tabeli 23 põhjal saab öelda, et kõige rohkem inimlikke omadusi on PF agendil, kuna agent simuleerib enim, mida inimene teeks. Eelise teiste agentide ees annab PFile, tema mitmed algoritmid, mis muudavad agendi rohkem ettearvamatuks, kui teised.

Tabel 24. Agentide suhtlemisoskus.

Omadus	AIML	Cleverscript	Personality Forge
Kõnesüntees	Puudub.	Avataril naiselik hääl.	Puudub.
Kõnetuvastus	Puudub.	Koos avatari kasutamisega.	Puudub
Suhtlus loomulikus keeles	Jah.	Jah.	Jah.
Keelte oskus	Igas keeles.	Igas keeles.	Inglise keel.
Sisendist aru saamine	Modelleerida terviklik sisend.	Modelleerida terviklik sisend.	Modelleerida fraas, lause tükike, mis leidub sisendis.
Sünonüümidest arusaamine	<srai> abil viitamine	Phrases ja loetelud.	Plug-in ja loetelud.
Sõnavara	Kesine sõnavara käsitlemine.	Suur sõnavara tuvastamiseks ja geneeerimiseks fraaside korrutamise abil.	Pluginate lisamine kui ka sisseehitatud pluginad aitavad sõnavara suurendada.

Tabeli 24 põhjal on parimaks multilinguaalseks suhtlejaks Cleverscript. Tugevaks eeliseks on ka oskus suhelda räägitavas keeles, peale tavalise tekstipõhise suhtluse.

Tabel 25. Agentide tehnilised omadused.

Omadus	AIML	Cleverscript	Personality Forge
Kasutusvõimalused	API võimalus ja programmid erinevates keeltes!	API - igas rakenduses kasutatav	API - igas rakenduses kasutatav
Vabavara	Jah.	Ei.	Ei.
Avatari realistlikus	Puudub	Realistlik 3D nimese mudel koos hästi animeeritud ilmetemuutustega.	Animeeritud pildid, koomiksi stiilis.
Avatari väljendusrikkus	Puudub	50 reaktsiooni, 70 emotsiooni.	Kümmekond emotsiooni seisundit.
Rekursioonimeetod	<srai>	Phrases	Puudub.
Meetodit tundmatu sisendi käsitlemiseks	*	~ ja _ ning fraasid.	*, (*) ja pluginad.

Tabeli 25 põhjal pakub Cleverscript ka parimaid tehnilisi võimalusi omades paremaid võimalusi realistliku avatari kasutamiseks.

Tabelite põhjal võib teha järeldusi, aga agendi efektiivsus ja oskus vastata võimalikult inimlikult, sõltub agendi looja pühendumusest ja loovusest. PF ja Cleverscript võivad olla mugavamad rohkemate funktsioonide poolest, aga PF agent on piiratud ainult ühte keelde ja Cleverscripti kasutamine on üldjuhul tasuline. AIML seevastu paistab esile, kuna tegemist on vabavaraga ja leidub mitmeid AIML implementatsioonprogramme, mida pakutakse tasuta.

KOKKUVÕTE

Autor uuris kolme erinevat meetodit tehisintellekti rakenduste loomiseks. Uurimuse läbiviimiseks töötas autor läbi agentidega seotud teoreetilise tausta, mis hõlmas manuaale, õpetusi ja erialakirjandust, mis käsitlesid ajalugu, uuringuid ning artikleid agentide kasutusvõimalustest ja teiste autorite kogemustest agentide kasutamisel.

Autori püstitatud eesmärgid on saavutatud. Uurimus tutvustas teemaga seonduvat ajalugu. Kirjeldas agentide toimimise tagamaid koos illustreerivate näidetega. Koostas võrdlusanalüüsi, milles autor tõi välja tugevused, nõrkused, sarnasused, erinevused, probleemid ja klassifitseeris agentide omadused.

Antud uurimus annab hea ülevaate tehisintellekti meetoditest, mida tänapäeval saab rakendada veebilehtedel, rakendustes, mängudes vms. Uurimus aitab neid, kes soovivad tutvuda olemasolevate võimalustega agentide implementeerimiseks oma süsteemides või kes ei ole kindlad, millise TIS kasuks otsustada.

Uurimus annab autorile hea aluse, mille põhjal jätkata bakalaureusetööd tehisintellekti teemal.

KASUTATUD KIRJANDUS

1. *Apple - iOS 8 - Siri* (2015). https://www.apple.com/ios/siri/ (01.02.2015)
2. *Google Now. Õige teave õigel ajal.* (2015). https://www.google.com/landing/now/ (01.02.2015)
3. *Introducing Amazon Echo* (2015). http://www.amazon.com/oc/echo/ref_=ods_dp_ae (01.02.2015)
4. *Ask Anna!* (2015) http://www.ikea.com/us/en/ (17.01.2015)
5. Morais, B. (The New Yorker). (19.11.2013). *Can Humans Fall in Love with Bots?* http://www.newyorker.com/tech/elements/can-humans-fall-in-love-with-bots#livefyre (25.01.2015)
6. Reeves, B. (2004). *The Benefits of Interactive Online Characters* https://www.sitepal.com/pdf/casestudy/Stanford_University_avatar_case_study.pdf (24.01.2015)
7. Shawar, B. A., Atwell, E. (2007). *Chatbots: are they really useful?* http://www.comp.leeds.ac.uk/eric/abushawar07ldvfj.pdf (24.01.2015)
8. Kerly, A., Hall, P., Bull, S. (2006). *Bringing chatbots into education: Towards natural language negotiation of open learner models* http://www.sciencedirect.com.ezproxy.tlu.ee/science/article/pii/S0950705106001912 (26.01.2015)
9. Koit, M. (2011). *Tehisintellekt* http://dspace.utlib.ee/dspace/bitstream/handle/10062/28296/tehisintellekt.pdf?sequence=2 (16.01.2015)
10. Weizenbaum J. (1966). *ELIZA - A Computer Program For the Stufy of Natural Language Communication between Man And Machine* http://www.cse.buffalo.edu/~rapaport/572/S02/weizenbaum.eliza.1966.pdf (30.01.2015)

11. *Home Page of The Loebner Prize in Artificial Intelligence* (2014). http://www.loebner.net/Prizef/loebner-prize.html (27.01.2015)

12. *History of the PC Therapist* (puudub info). http://loebner.net/Prizef/weintraub-bio.html (01.02.2015)

13. *Alicebot blog* (2013). http://alicebot.blogspot.com/ (26.01.2015)

14. Wallace, R., Bush, Noel. (2011). *Artificial Intelligence Markup Language (AIML) Version 1.0.1* http://www.alicebot.org/TR/2011/ (27.01.2015)

15. Marietto, M. Aguiar, R., Barbosa, G., Botelho, W., Pimentel, E., Franca, R., Silva, V. (2013). *Artificial Intelligence Markup Language: A Brief Tutorial* http://arxiv.org/ftp/arxiv/papers/1307/1307.3091.pdf (03.02.2015)

16. Wallace, R. (2003). *The Elements of AIML Style* http://www.alicebot.org/style.pdf (27.01.2015)

17. *About Cleverscript* (2014). http://www.cleverscript.com/about/ (07.02.2015)

18. *Cleverscript Manual and Tutorial* (2014). http://www.cleverscript.com/CSL/CleverScriptManual.pdf (07.02.2015)

19. *About The Personality Forge* (2014). http://www.personalityforge.com/info.php (11.02.2015)

20. *AI Engine Features* (2014). http://www.personalityforge.com/ai-engine.php (11.02.2015)

21. *The Chat Bot API* (2014). http://www.personalityforge.com/chatbotapi.php (11.02.2015)

22. *Bot Development - Beginner* (2014). http://www.personalityforge.com/book.php (11.02.2015)

23. *Bot Development - Advanced* (2014). http://www.personalityforge.com/book-advanced.php (11.02.2015)

24. Bot Development - Expert (2014) http://www.personalityforge.com/book-expert.php (11.02.2015)

25. Wilcox, B., Wilcox, S. (2011). *Suzette, the Most Human Computer* http://chatscript.sourceforge.net/Documentation/Suzette_The_Most_Human_Computer.pdf (17.02.2015)

26. McNeal, Michele L., Newyear, D. (2013). *Chatbot Creation Options* https://www.questia.com/library/journal/1G1-376070856/chatbot-creation-options (20.02.2015)

27. *IT terministandardi sõnastik* (2014). http://www.eki.ee/dict/its/ (25.02.2015)