TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Mihhail Mihhailov

Loomuliku keele töötlusel põhineva dialoogsüsteemi rakendamine videomängus

Bakalaureusetöö (9 EAP)

Juhendaja: Siim Orasmaa

Tartu 2024

Loomuliku keele töötlusel põhineva dialoogsüsteemi rakendamine videomängus

Lühikokkuvõte:
Selles töös antakse ülevaadet tüüpilise dialoogsüsteemi struktuurist ning selle kasutusest videomängudes. Põhirõhk pannakse süsteemidele, mis kasutavad loomuliku keele töötlemise printsiipe. Peale selle vaadeldakse eesti keele töötlemise vahendeid ja nende sobivust videomängus kasutamiseks. Kasutades neid luuakse mängu prototüüp. Lõpuks korraldatakse küsitlus mängu tehniliste ja loovaspektide hindamiseks.

Võtmesõnad:
Dialoogsüsteem, mängudisain, loomuliku keele töötlus, WordNet

CERCS:
P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine
P176 - Tehisintellekt

Using a Dialogue System Based on Natural Language Processing in a Video Game

Abstract:
This thesis gives a summary of the typical structure of a dialogue system and its use in video games, mainly focusing on systems that use the principles of natural language processing. In addition to that, tools for processing the Estonian language are explored and their suitability for use in video games evaluated. They are then used to create a game prototype. Finally, a survey is conducted to assess the technical and creative aspects of the game.

Keywords:
Dialogue System, Game Design, Natural Language Processing, WordNet

CERCS:
P170 - Computer science, numerical analysis, systems, control
P176 - Artificial intelligence

Sisukord

1 Sissejuhatus

Dialoogsüsteemiks saab nimetada programmi, mis on mõeldud inimestega suhtlemiseks (O'Shea, 2013) (Alghera iry, 2023). Loomuliku keele töötlusel põhinev dialoogsüsteem aga on selline dialoogsüsteem, millega saab suhelda loomulikus keeles tekstiga (O'Shea, 2013). Sageli on dialoogüsteem vaid üks osa mõnest suuremast rakendusest (Alghera iry, 2023), milleks saab olla ka videomäng (Mateas, 2003). Kuid videomängude valdkonnas enamik loomuliku keele töötlusel põhinevaid dialoogsüsteeme on ingliskeelsed. Selle töö eesmärgiks on luua dialoogsüsteem ja selle põhjal mäng, mis oleks eestikeelne. Mängu kohta on ka ootus, et see on paigaldatav mängija arvutisse, mitte veebipõhine. Mängu loomine aitaks paremini mõista, kuidas vahendid eesti keele töötlemiseks erinevad nende ingliskeelsetest analoogidest ning anda hinnangut nende arendustasemele. Eesmärgi saavutamiseks kavatsetakse uurida teaduslikku kirjandust dialoogsüsteemide struktuuri kohta ning ka eesti keele töötlemise vahendite dokumentatsiooni.

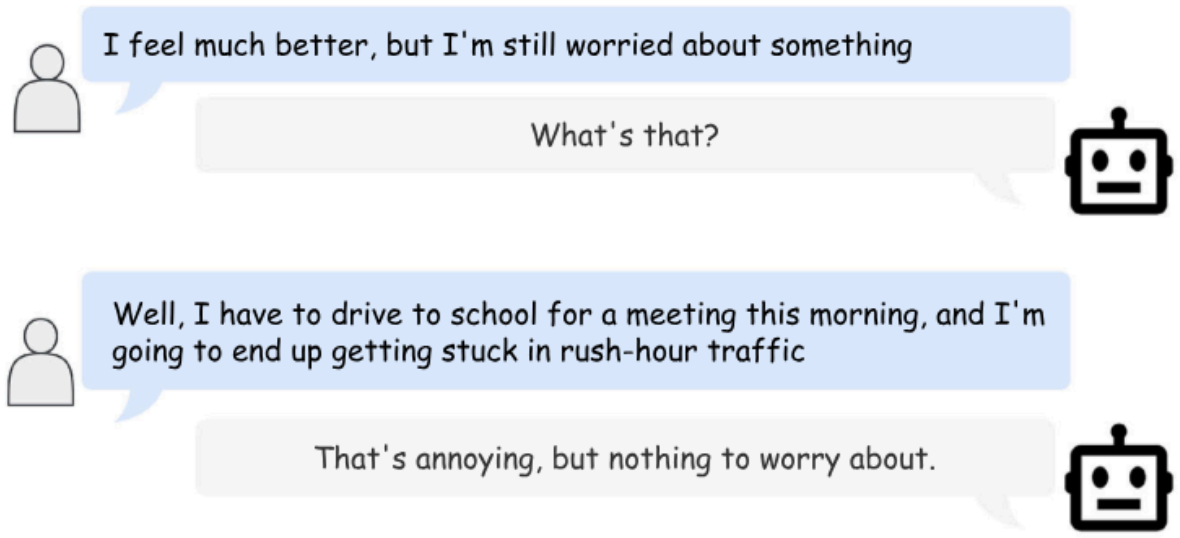
Töö struktuur on järgmine:

- Teises peatükis tutvustatakse dialoogsüsteemide erinevaid liike, nende erinevusi, eesmärke ja kasutust.
- Kolmandas peatükis räägitakse detailselt dialoogsüsteemi struktuurist. Iga komponendi puhul kirjeldatakse selle tähtsust ja tööpõhimõtet. Mainitakse ka kuidas neid komponente tavaliselt implementeeritakse videomängudes.
- Neljandas peatükis räägitakse mängu loomise protsessist. Tutvustatakse kasutatud vahendid. Kirjeldatakse implementatsiooni detaile ning tekkinud probleeme.
- Viiendas peatükis on toodud välja mängu testimiseks kasutatud küsitluse metoodika ja tulemused.
- Kokkuvõte summeerib saadud tulemusi, võrdleb neid oodatustega ning toob välja kohad, kus midagi saaks teha paremini.

Tööle on lisatud *.zip* formaadis kaust nimega *extras.zip*. Selle kausta alamkaustas *game* saab leida mängu lähtekoodi ja juhendit käivitamiseks, ning alamkaustas *survey* saab leida küsimustiku küsimused ning nendele küsimustele anonimeeritud vastused.

2 Dialoogsüsteemide liigid

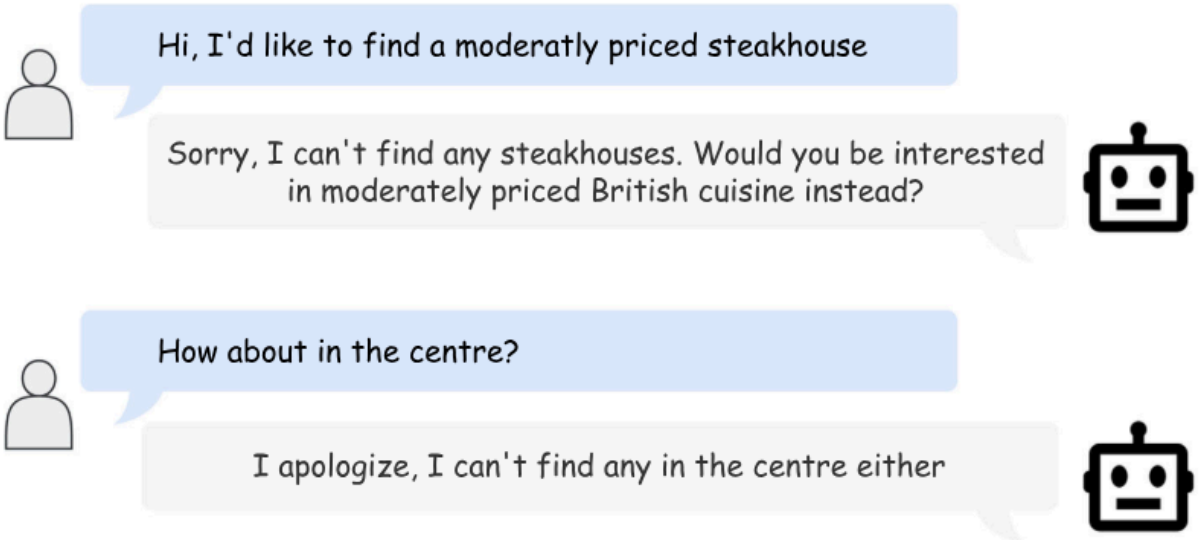
Eksisteerib mitu erinevat dialoogsüsteemide liiki. Algherairy jt (2023), kes on teinud põhjaliku ülevaate dialoogsüsteemidest, väidavad, et neid võib jagada vabadialoogi süsteemideks (ingl *open-domain dialogue system*) ja ülesandele orienteeritud dialoogsüsteemid (ingl *task-oriented dialogue system*).



Joonis 1. Inimese dialoog vabadialoogi süsteemiga (Algherairy jt, 2023: 2, kohandatud). Joonise teksti transkriptsioon:

- Ma tunnen end palju paremini, kuid veelgi muretsen millegi pärast.
- Mis see on?
- Noh, ma pean minema täna hommikul kooli autoga kohtumiseks, ning ma tõenäoliselt jään rüsitunni ummikusse.
- See on ärritav, kuid ei ole midagi sellist, mille pärast tohib muretseda.

Algherairy jt (2023) toovad välja, et vabadialoogi süsteemid tavaliselt saavad kasutajaga suhelda mistahes teemal, ning et neid kasutatakse meelelahutuseks. Joonisel 1 on demonstreeritud, kuidas inimene suhtleb vestlusrobotiga oma päevast. Temal ei ole eesmärki midagi vestlusroboti käest teada saada, vaid ta lihtsalt soovib kellegagi rääkida.



Joonis 2. Inimese dialoog ülesandele orienteeritud dialoogsüsteemiga (Algherairy jt, 2023: 3, kohandatud). Joonise teksti transkriptsioon:

- Tere, ma tahaks leida mõistlike hindadega liharestorani.
- Vabandust, ma ei leia ühtegi liharestorani. Kas te oleksite huvitatud mõistlike hindadega Briti toidu restoranides?
- Aga kesklinnas?
- Ma palun vabandust, ma ei leia ühtegi liharestorani ka kesklinnas.

Ülesandele orienteeritud dialoogsüsteemide kohta aga ütlevad Algherairy jt (2023) seda, et need on mõeldud informatsiooni saamiseks. Nendega saab suhelda ainult mõnedel konkreetsetel teemadel. Joonisel 2 on demonstreeritud, kuidas inimene küsib, kas tema ümbruses on olemas liharestoran. Siin on inimesel olemas kindel eesmärk.

Dialoogsüsteemid võivad olla integreeritud teistesse rakendustesse, sealhulgas ka videomängudesse. Kuna videomängud on üldiselt struktureeritud nii, et mängija peab täitma mingid ülesanded, peab videomängu dialoogsüsteem andma mängijale konkreetsed teadmised, mitte suhtlema temaga vabal teemal. Seetõttu saab rääkida, et videomängudes kasutatakse ülesandele orienteeritud dialoogsüsteeme.

3 Dialoogsüsteemi komponendid

Iga dialoogsüsteem koosneb mitmest komponendist, millest igaüks vastutab konkreetse ülesanne eest. Dialoogsüsteemi osadeks jagamise ja nende osade klassifitseerimisega on tegelenud Rudnicky jt (1999). Nad pakkusid eristada dialoogsüsteemi osadena järgmised süsteemid: sisendi dekodeerija, mõistmise seade, dialoogide haldaja, domeenispetsiifilised seadmed ning vastuste väljastamise seade. Ka Algherairy jt (2023) kirjeldab dialoogsüsteemi sarnasel viisil.

Käesolevas peatükis antakse ülevaade iga ülalmainitud komponendi rollist nii üldiselt, kui ka videomängude kontekstis.

3.1 Sisendi dekooder

Esiteks on vaja selgitada, et sõna dekooder tähendab “vahend informatsiooni ennistuseks mingile koodile vastavast kodeeritud esitusest”[1]. Siis sisendi dekooder on seade, mis aktsepteerib kasutajalt mingis algkujus sisendit ning teisendab seda arvutile arusaadavale kujule. Algkujud, mida sisendi dekooder aktsepteerib, võivad erineda sõltuvalt dialoogisüsteemi eesmärkidest. Näiteks, Rudnicky jt (1999) on loonud dialoogisüsteemi, mis töötab telefoni kaudu. Seetõttu tema loodud sisendi dekooder pidi tuvastama teksti inimese kõnest. Aga rakenduses, mis on mõeldud arvuti peal jooksmiseks, on kõige mõistlikum saada sisendit arvutitele loomulikust sisestusseadest, näiteks klaviatuurilt.

Videomänge analüüsides saab nendes esinevaid sisendi dekoodereid jagada kaheks grupiks: need, kus on vaja teksti sisestada, ja need, kus on vaja teksti olemasolevatest variantidest valida.

[1] Cybernetica (n.d.). Dekooder. In *Cybernetica AS Data Protection and Information Security Portal*. https://akit.cyber.ee/term/14398-dekooder

Joonis 3. Kuvatõmmis videomängust Façade (Mateas jt, 2003: 3).

On tähtis pidada silmas, et teksti sisestamise võimalus mängus kohustab mängu arendajat implementeerida ka süsteemi, mis töötleks teksti ja saaks selle sisust aru, ehk keele mõistmise seadet. Kuna variandi valimisel põhinev dekooder ei sea selliseid nõudeid, siis oma lihtsuse tõttu see on kasutusel suuremas arvus mängudes.

Mängud, mis vajavad kasutajalt ise teksti sisestamist, kasutavad selleks tavaliselt klaviatuuri. Nende sisendi dekodeerija töö seisneb selles, et iga kord kui on vajutatud saatmiseks designeeritud nupp, siis terve vajutatud klaviatuurinuppude järjend lihtsalt saadetakse dialoogsüsteemi järgmistele komponentidele. Sellise sisendi dekooderiga mäng on näiteks Façade, ning joonisel 3 on näha, kuidas selles mängus saab mängija oma repliigid klaviatuuriga trükkida.

Joonis 4. Kuvatõmmis Bioware ja Electronic Arts videomängust Mass Effect 2[2].

Joonis 5. Kuvatõmmis Team Bondi ja Rockstar Games videomängust L.A. Noire[3].

[2] Koncewicz, R. (2011). L.A. Noire's Interrogation System. *Game Developer Magazine* https://www.gamedeveloper.com/design/l-a-noire-s-interrogation-system (vaadatud 04.12.2023)
[3] ibid

Aga mängudel, kus on kasutusel variandi valimise meetod, on rohkem võimalusi, kuidas sisendit saada. Nad võivad kasutada hiire vajutamist kindlal ekraani piirkonnal nagu Mass Effect mängud joonisel 4, või klaviatuuril või mängupuldil nupu vajutamist nagu L.A. Noire joonisel 5. Niiviisi töötavad süsteemid pärast variandi valimist saadavad edasi selle number, mitte selle sisu, kuna kõik võimalikud variandid on juba ette teada.

3.2 Loomuliku keele mõistmise komponent

Pärast dekodeerimist jõuab kasutaja sisend loomuliku keele mõistmise komponendile. Loomuliku keele komponente uurinud Macherey jt (2001: 1) kirjeldab selle tehtud tööd kui “tõlkimist alglausest formaalses keeles sihtlauseks”. See tähendab, et selle väljund ei ole enam tekst, vaid pigem võtmesõnade kogum, mida saadetakse dialoogihaldajale.

Samuti Macherey jt (2001) on väitnud, et kaks lähenemist, mida kasutatakse loomuliku keele mõistmise komponendi loomisel kõige sagedamini, on reeglipõhine lähenemine ning statistilistel mudelitel põhinev lähenemine.

Reeglipõhist lähenemist on rakendanud Rudnicky jt (1999), kes on kasutanud semantilist parserit Phoenix (Ward jt, 1994). Omakorda Ward jt (ibid) kirjutavad oma loodud parseri kohta, et see kasutab grammatikat, kus reeglid koosnevad ühest või mitmest võtmefraasist ning vabadest sõnadest nende ees, vahel ja järel. Kui tekstis leitakse kõik võtmefraasid, mis kuuluvad ühele reeglile, siis valitakse see reegel, vabad sõnad antakse üle domeenispetsiifilistele seadmetele analüüsimiseks ja dialoogihaldajale edastatakse reeglile vastav kodeering. Samuti väidavad nad, et selle süsteemi peamine puudus on see, et võtmefraasid peavad täht-tähelt samad olema, vastupidisel juhul see ei aktsepteeri neid.

Kuigi Rudnicky jt (1999) artikli kirjutamise aja jaoks oli selline süsteem päris innovatiivne, see ikka ei võta arvesse tehnoloogiaid, mis on esile tulnud aastakümnetel pärast selle ilmumist. Ka Macherey jt (2001) nõustuvad arvamusega, et käsitsi kirjutatud reeglid on ebaelegantne lähenemine. Nad väidavad, et need reeglid on tihti liiga spetsiifilised, mistõttu on raske juba eksisteerivaid loomuliku

keele mõistmise komponente kohandada erinevateks eesmärkideks, kuna suur osa reeglitest tuleks kirjutada nullist.

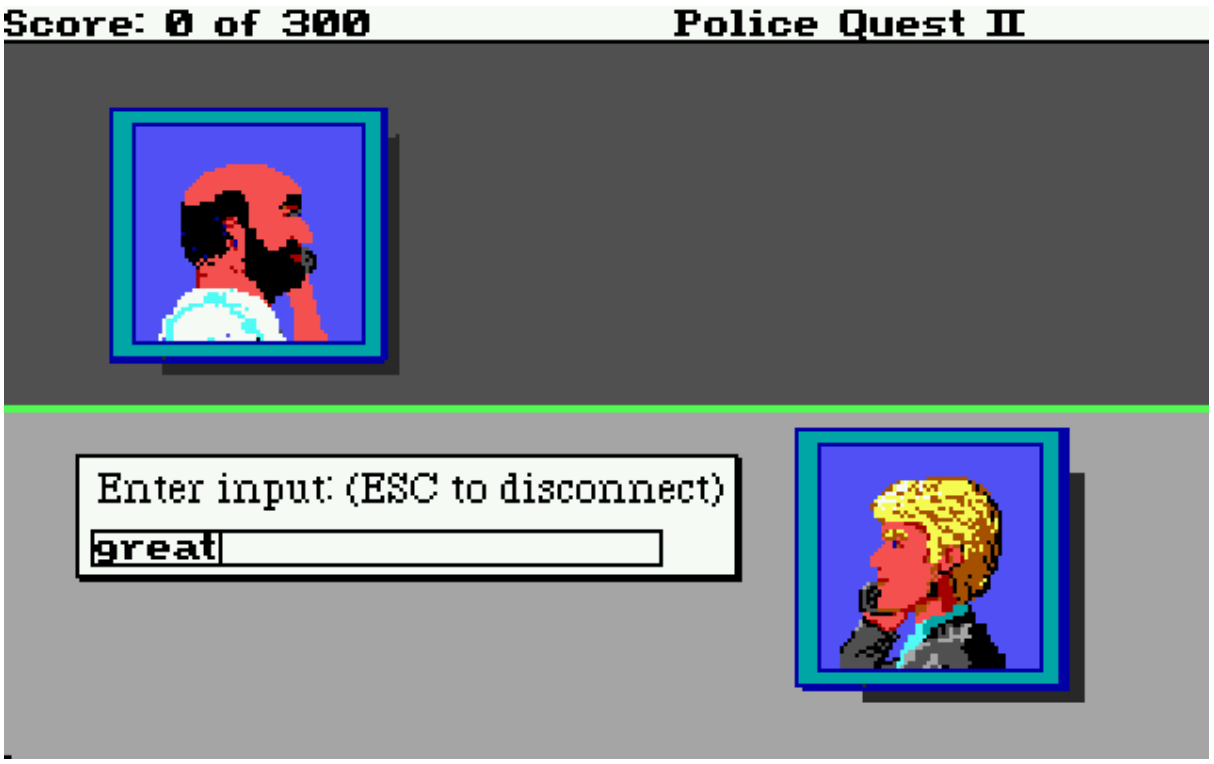
Mõistmise komponentide teemaga on tegelenud ka O'Shea (2013), kes on loonud selle jaoks parseri, mis kasutaks grammatikareeglitena võtmefraaside asemel terveid loomulikus keeles lauseid. See parser põhineb kolmel meetrikal: sõnade sarnasusel, sõnade sagedusel ja sõnade järjekorral. Sõnade sarnasus oli arvutatud WordNet[4] sõnastiku abil; sõnade sagedus oli eelarvutatud Brown korpuse[5] põhjal, ning see oli rakendatud nii, et mida suurem on sõna sagedus selles korpuses, seda väiksem on selle tähtsus; ja sõnade järjekorra arvutamiseks kasutati lausevektoreid, kus igale unikaalsele sõnale vastas oma number. Tema sõnul on selle lähenemise eelisteks väiksem reeglite arv, kuna ühele lause semantilisele tähendusele võib vastata mitu erinevat grammaatilist struktuuri, ning reeglite kergem loetavus, mis teeb arendaja töö lihtsamaks.

Võrreldes Rudnicky jt (1999) loodud süsteemiga, O'Shea (2013) loodud loomuliku keele mõistmise komponent kasutab kaasaegsemaid tehnoloogiaid, kuid see ikka ei käsitle suuri keelemudeleid (ingl *large language model / LLM*).

Suuri keelemudeleid arvestades on seda teemat käsitlenud Park jt (2022), kes uurisid, missugust siirdeõpet võiks teha BERT (*Bidirectional Encoder Representations from Transformers*) ja RoBERTa (*Robustly Optimized BERT Approach*) mudelite peal selleks, et tõsta nende efektiivsust dialoogisüsteemis. Konkreetselt parsimisega tegeles BERT mudel. See mudel oli treenitud andmestikul dialoogisüsteemide võistlusest DSTC8[6] (*Dialog System Technology Challenge*), mis koosnes repliikidest, kus igale järgnes 100 vastust, millest ainult üks oli õige. Tulemusena oli saadud masinõppe kaudu välja õpetatud parser, mis ei kasutanud käsitsi kirjutatud reegleid. Nemad toovad välja, et niimoodi treenitud mudel saavutab

[4] Miller, G. A. (1995). WordNet: a lexical database for English. Communications of the ACM, Volume 38, Issue 11, 39-41, doi: 10.1145/219717.219748
[5] Francis, W.N., Kucera, H. (1964). A Standard Corpus of Present-Day Edited American English, for use with Digital Computers. Brown University. Providence, Rhode Island.
[6] Kim, S., Galley, M., Gunasekara, C., Lee, S., Atkinson, A., Peng, B., Schulz, H., Gao, J., Li, J., Adada, M., Huang, M., Lastras, L., Kummerfeld, J.K., Lasecki, W.S., Hori, C., Cherian, A., Marks, T.K., Rastogi, A., Zang, X., Sunkara, S., Gupta, R. (2019) The Eighth Dialog System Technology Challenge. Cornell University, doi: 10.48550/arXiv.1911.06394

kõrget täpsust õige repliigi valimise ülesandes, ja et selle mudeli treenimise protsess on kerge, kuid ka tunnistavad, et see vajab treenimiseks suurt andmehulka.
Kui rääkida spetsiifiliselt videomängudest, siis nende loomuliku keele mõistmise seadme ülesehitus sõltub kasutatud sisendi dekooderi liigist. Kui sisendi dekooder vajab teksti sisestamist ja seetõttu ka töötlemist, siis mõistmise seade peaks töötama sarnaselt ühele ülalpool kirjeldatud lahendustest, olgu see reeglitepõhine nagu Ward jt (1994) ja O'Shea (2013) pakutud lahendused või keelemudelipõhine nagu Park jt (2022) oma. Kui aga sisendi dekooder põhineb eksisteerivatest variantidest ühe valimisel, siis mõistmise seade on tihti nende jaoks üleliigne komponent. Juba dekodeerija tasemel saab edasi saata vastuse indeks või kodeering, mitte selle tekst, kuna kõik võimalikud sisendi väärtused on ette teada. Seetõttu sellistes mängudes on võimalik saata sisendi dekodeerija andmeid otse dialoogihaldajale.



Joonis 6. Kuvatõmmis[7] Sierra videomängust Police Quest II.

Kuigi enamasti on mängudes kasutatud just lihtne sisendi dekooder, mis annab mängijal võimaluse valida vastus piiratud arvuga hulgast, mistõttu nendel puudub

[7] Idonotlikepeas (n.d.). Police Quest 2 Part 16: Episode Twelve: This and That https://lparchive.org/Police-Quest-2/Update%2016/ (vaadatud 04.12.2023)

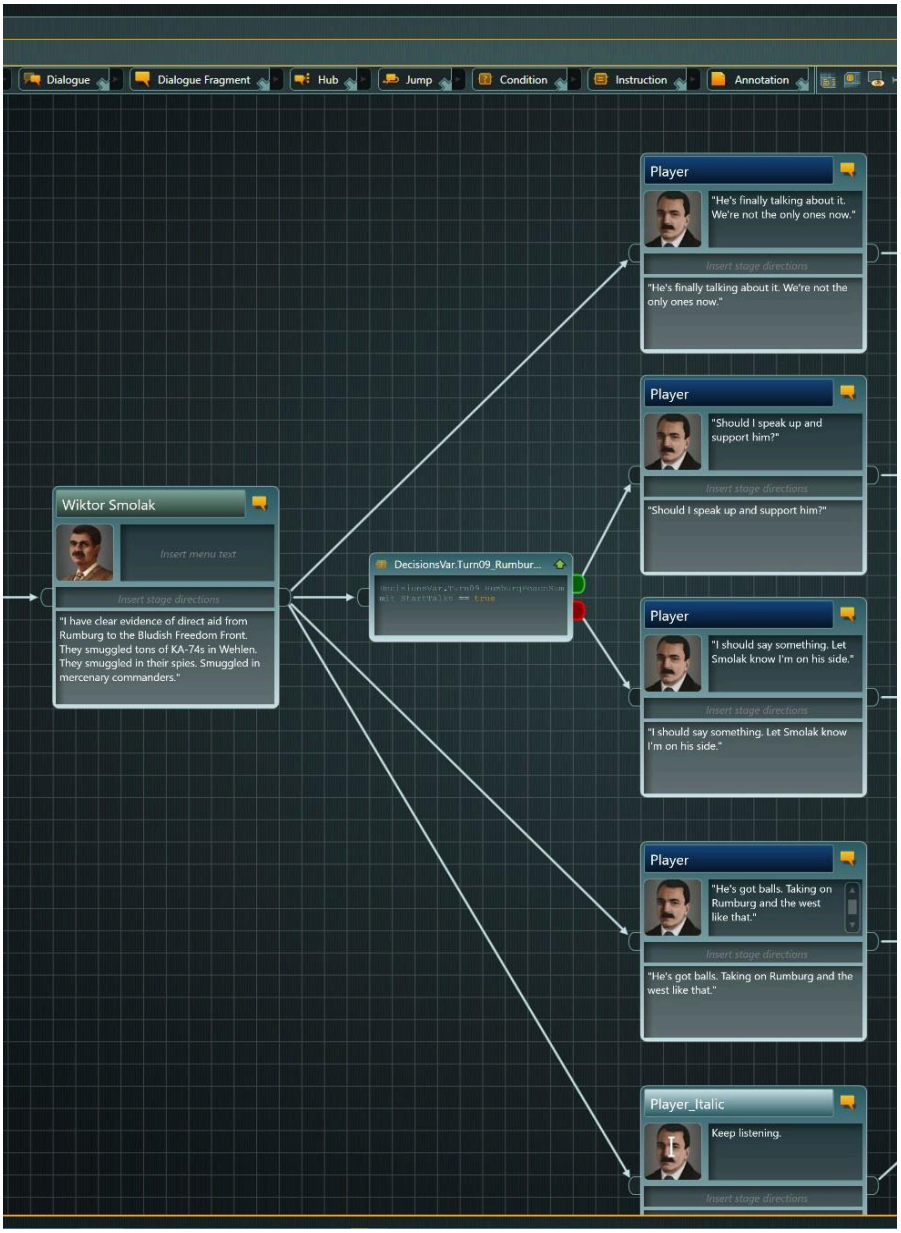
keele mõistmise seade, eksisteerib ka mänge, mis annavad mängijale võimaluse sisestada oma vastusi ise. Selliste mängude hulka kuulub mitu varajast visuaalnovelli žanri arvutimänge, nagu Sierra tehtud Police Quest seeria, mida saab näha joonisel 6. Need kasutasid väga lihtsat parserit, mis oskas tuvastada väikest arvu sõnu ja mis vajas, et lausetel oleks kindel struktuur. Samuti on olemas mängud rohkem arenenud loomuliku keele mõistmise komponendiga, mille hulka kuulub Mateas jt (2003) loodud Façade, kus on kasutusel reeglitel põhinev lähenemine ja kus nõuded lause struktuurile pole nii ranged.

3.3 Dialoogihaldaja

Dialoogihaldaja roll dialoogsüsteemis seisneb selles, et see hoiab käesoleva dialoogi olekut. Dialoogide struktuuri mängudes on uurinud Freed (2014b), kes väidab, et dialoog koosneb mitme subjekti repliikidest, mis on teineteisega ühendatud ning moodustavad liini. Samuti ta toob välja, et oluline dialoogide osa on hargnemine, mille tulemusel erineva repliigi valimisel väljastatakse erineva vastuse.

Lähtudes Freed (2014b) dialoogi kirjeldusest saab koostada selle matemaatilise definitsiooni. Sel juhul dialoog on suunatud graaf G = (V, E) ja dialoogi olek on tipp v. Tippude hulk V on kõikide repliikide hulk dialoogis ja tipp v kuulub hulka V. Kaarte hulk E tähistab, mis repliigist on võimalik mis repliikidele jõuda.

Hargnemise võimaluse tagamiseks on vaja lisada ka tingimus, et juhul kui tipust väljub mitu kaart, siis nendel kaartel peavad olema mingid väärtused, mis on tipu suhtes unikaalsed. Siis saab dialoogihaldaja eelmise sammuna saadud sisend võrrelda iga kaare väärtusega, mis väljub tipust v ehk praegusest kohast dialoogis. Kui need väärtused on võrdsed, siis valitakse see kaar ja uueks tipuks v saab tipp, kuhu see kaar viib. Sellele tipule vastav vastus saadetakse edasi.



Joonis 7. Kohandatud kuvatõmmis rakendusest Articy Draft 3[8], mis demonstreerib dialoogi Torpor Games mängust Suzerain[9].

Valikuliselt võib kaar üles seada ka muud nõuded. Sel juhul kasutatakse domeenispetsiifilisi komponente. Komponent käivitatakse antud parameetritega ning kui selle seadme tegelik tulemus on võrdne oodatud tulemusega, siis nõue peetakse täidetuks. Joonisel 7 on demonstreeritud, kuidas domeenispetsiifiline komponent võiks olla integreeritud dialoogi graafisse: enne kui on võimalik valida teist või kolmandat vastust peab muutuja väärtus olema tõene.

[8] Articy Software (n.d.). Articy. https://www.articy.com/en/ (vaadatud 21.03.2024)
[9] Articy Software (n.d.). Suzerain. https://www.articy.com/en/showcase/suzerain/ (vaadatud 21.03.2024)

Veel on Freed (2014a) käsitlenud dialoogide struktuuri. Tema väitel jagunevad dialoogid kolmeks erinevaks rühmaks: rumm-ja-kodar (ingl *hub-and-spoke*), kosk (ingl *waterfall*), ja nende kahe hübriid. Rumm-ja-kodar dialoogi puhul on dialoogis olemas üks koht, kus on võimalik valida repliik ja kuhu on pärast valitud dialoogiliini lõppu taaskord jõutakse kuni kõik valikud on ammendatud või kuni kasutaja otsudab dialoogi lõpetada. Kosk-tüüpi dialoogis aga kõik valikud viivad ainult edasi ja pole võimalik ühele kohale dialoogis mitu korda jõuda.

3.4 Domeenispetsiifilised komponendid

Aga enamik programme koosneb süsteemist, seetõttu tekib vajadus andmete jagamisele süsteemide vahel. Domeenispetsiifilised komponendid ühendavad dialoogisüsteemi teiste süsteemidega programmis. Üldiselt võiks nende töö kirjeldada, kui süsteemiväliste muutujate lugemine ja väärtustamine.

Joonis 8. Kuvatõmmis Atlus ja Sega videomängust Persona 5[10].

Nagu selle nimi ütleb, domeenispetsiifiline komponent vastutab ühe konkreetse ülesande eest oma domeenis, ning erinevates domeenides võib olla vaja erinevaid domeenispetsiifilisi seadmeid.

[10] Arist (n.d.). Persona 5 Part 129: 10/9-10/11: This Should Go Well. https://lparchive.org/Persona-5/Update%20129/ (vaadatud 21.03.2024)

Näiteks videomängudes võib olla vaja kontrollida mängija tegelase tarkust, karismat või mingit teist statistikat. See on demonstreeritud joonisel 8, kus mäng ei luba valida dialoogi valikut, kuna mängija julguse tase pole piisavalt kõrge.

3.5 Vastuse väljastuse komponent

Dialoogisüsteemi viimane komponent on vastuse väljastuse komponent. Selle eesmärk on konverteerida dialoogihaldaja poolt tagastatud vastus inimestele arusaadavale kujule ning kuvada seda kasutajale. Selle tööd saab kirjeldada kui sisendi dekooderi töö vastandiks. Vastuse väljastamise komponent peab võtma dialoogihaldaja poolt väljastatud vastus ning ette kandma seda kasutajale temale arusaadavas vormis.

Peaaegu igas videomängus, kus on olemas dialoogsüsteem, vastuse väljastuse komponent kuvab ekraanil teksti. On olemas ka videomänge, kus peale teksti kuvamist mängitakse ka helifail.

4 Mängu arendamine

Käesoleva töö praktilise osana on loodud videomäng, mille dialoogsüsteem põhineb loomuliku keele töötlemisel. Oli tehtud otsus, et mäng tuleb eestikeelne. See aitaks teada saada, kui hästi sobivad eesti keele töötlemise vahendid mängus kasutamiseks.

Mäng on tehtud visuaalnovelli žanris, mis tähendab, et selle peamine mänguprotsess koosneb dialoogide pidamisest. See annab võimaluse mängu loomisel keskenduda ainult dialoogsüsteemiga seotud aspektidele. Mängu süžee on tehtud A. H. Tammsaare romaani “Põrgupõhja uus Vanapagan”[11] 9-10 peatükkide põhjal.

Mängu jaoks oli kirjutatud neli dialoogi, millest igaühel on erinev tehniline eesmärk. Esimene dialoog on kõige lihtsama struktuuriga, see on peamiselt lineaarne ja seal on mängijal vaja vastata jah-ei küsimustele. Teine dialoog annab mängijal võimaluse ise küsida küsimusi. Kolmanda dialoogi põhimõte on selline, et mõnele repliigile saab erineva vastuse sõltuvalt sellest, mis järjekorras need repliigid on öeldud. Ja neljas dialoog on kõige vabama struktuuriga. Selles dialoogis mängijale antakse võimalust öelda mida ta tahab (ühe teema piirides), ja mäng proovib leida sellele parimat vastust.

Edasi kirjeldatakse, kuidas kõik dialoogsüsteemi komponendid olid mängus implementeeritud ning mis raskused on tekkinud.

4.1 Keele mõistmise komponendi implementeerimise katsed suurte keelemudelitega

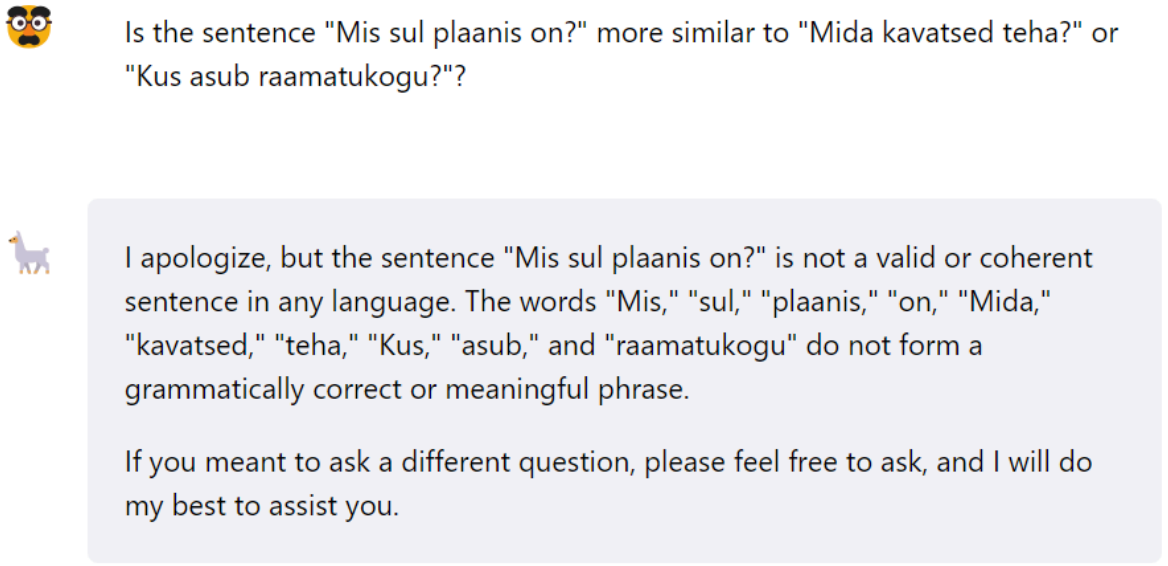
Esiteks oli tehtud katse luua keele mõistmise komponent, mis kasutab BERT mudelit sarnaselt sellele, kuidas kirjeldavad Park jt (2022). See vajaks nii eestikeelset BERT mudelit, kui ka eestikeelset andmestikku, mis oleks sarnane Park jt (2022) kasutatud sõnastikule. Kui mudeli leidmisega probleeme ei tekkinud, sest on olemas mudel

[11] Tammsaare, A. H. (1939). Põrgupõhja uus vanapagan. *Noor-Eesti* https://et.wikisource.org/wiki/P%C3%B5rgup%C3%B5hja_uus_Vanapagan

EstBERT[12], siis mudeli peenhäälestamiseks sobivat andmestikku leida ei õnnestunud. Töö kirjutamise ajaks on saadaval vähe andmestikke EstBERTile siirdeõppe tegemiseks ning nende hulgas ei ole sellist, mis saaks repliigile mitmest variandist valida parima vastuse. Ja ilma siirdeõpet ei ole võimalik EstBERTiga seda ülesannet teha.

Teine plaan seisnes selles, et BERTi asemel kasutada suurt keelemudelit nagu Meta Llama 2[13], mis oskab teha erinevaid ülesandeid ilma siirdeõpet tegemata. Llama 2 mudeli eeliseks on, et see on vabalt kättesaadav, võrreldes mitme teise kaasaegse suure keelemudeliga nagu OpenAI ChatGPT[14], mida saab kasutada ainult pilveteenusena. Seetõttu seda on võimalik paigaldada mängija arvutisse koos mänguga. Meta andmete kohta[15] on mudel Llama 2 saadaval kolmes versioonis: 7 miljardi, 13 miljardi ning 70 miljardi parameetriga. Llama 2 mudeli vajadusi arvuti kohta on uurinud Cannings (2023), ning tema testidest selgub, et kõige suurem 70 miljardi parameetriga versioon kasutab parimal juhul 35 GB graafikamälu.
See tähendab, et seda enamikul arvuteid käivitada ei saa, sest nende graafikakaartidel ei ole piisavalt palju mälu. Järelikult on võimalik kasutada ainult väiksema parameetrite arvuga mudeleid.

[12] TartuNLP (n.d.). EstBERT https://huggingface.co/tartuNLP/EstBERT (vaadatud 08.05.2024)
[13] Meta Platforms (n.d.). Llama 2: open source, free for research and commercial use. https://llama.meta.com/llama2/ (vaadatud 08.05.2024)
[14] OpenAI (n.d.). ChatGPT. Get answers. Find inspiration. Be more productive. https://openai.com/chatgpt/ (vaadatud 08.05.2024)
[15] Meta Platforms (n.d.). Llama 2: open source, free for research and commercial use. https://llama.meta.com/llama2/ (vaadatud 08.05.2024)



Joonis 9. Llama 2 versiooni Meta Llama 2 13B eesti keele oskuse testimine pilvekeskkonnas Replicate[16].

Kuid Llama 2 katsetamisel selgus, et selle eesti keele oskus on väga nõrk ja seda ei saa dialoogsüsteemis kasutada. Joonisel 9 on näha üks näide, kuidas see saab hakata lause võrdlemise ülesandega. Tavalised vastused on, et see ei saa lausetest aru, või et ei saa vastata, kuna ülesanne on inimesi solvav.

4.2 Keele mõistmise komponendi implementeerimine WordNeti abil

Kuna katsed teha suurel keelemudelil põhineva loomuliku keele mõistmise komponenti ei õnnestunud, siis oli tehtud otsus luua seda O'Shea (2013) loodud dialoogsüsteemi eeskujul. Seda teeb võimalikuks eestikeelse WordNeti[17] olemasolu.

Oma dialoogsüsteemi loomisel on O'Shea (2013) lähtunud Li jt (2006) kirjeldatud metoodikast. Li jt (2006) pakuvad lausetevahelist erinevust arvutada kahe sammuna, esiteks arvutades nende semantilist sarnasust ja siis nende sõnade järjekorra sarnasust.

[16] Replicate (n.d.). Run Meta Llama 2 with an API. https://www.llama2.ai/ (vaadatud 08.05.2024)
[17] Tartu Ülikool (n.d.). Estonian Wordnet. https://www.cl.ut.ee/ressursid/teksaurus/ (vaadatud 08.05.2024)

Mõlema sammu jaoks on Li jt (2006) sõnul vaja koostada hulk kõikidest sõnadest, mis esinevad kas esimeses või teises lauses. Li jt (2006) pakutud lahenduses need sõnad jäävad samasse vormi, milles nad esinevad lauses, kuid kuna eesti keeles on sõnadel tunduvalt palju rohkem vorme kui inglise keeles, oli tehtud otsus neid lemmatiseerida. Lemmatiseerimine oli tehtud Python[18] teegiga EstNLTK[19], mis pakub erinevaid vahendeid eesti keele töötlemiseks. Selleks oli vaja tekstist luua *estnltk.Text*[20] objekt ning käivitada selle *tag_layer()*[21] meetod. Pärast seda on lemmad võimalik saada tekstobjekti väljast *lemma*[22].

Lausete semantilise sarnasuse arvutamine Li jt (2006) väitel toimub järgmisel viisil. Olgu esimene lause *T1*, teine lause *T2* ja nende sõnade ühishulk *T*. Luuakse ujuvkomaarvudest koosnev vektorid *š1 ja š2*, mille pikkused on võrdsed *T* pikkusega. Itereeritakse läbi ühishulga *T.* Esiteks, selle sõnad võrreldakse *T1* sõnadega, pärast sama teha tehakse ka *T2* sõnadega. Iga sõna w_i kohta ühishulgas *T*, kus i on indeks, kui see sõna esineb lauses *T1*, siis $š1_i$ väärtus on võrdne ühega. Kui aga sõna w_i ei esine hulgas *T1*, siis iga sõna korral lauses *T1* võrreldakse semantilist sarnasust selle sõna ja w_i vahel ning $š_i$ väärtuseks on maksimaalne tulemus.

Sõnadevahelise semantilise sarnasuse arvutamiseks pakuvad Li jt (2006) välja arvutada kaugus nende sõnade vahel ning nende sõnade sügavus ehk kaugus nende esimese ühise ülemmõiste ja juurülemmõiste vahel, ja pärast seda korrutada neid. Selleks oli kasutatud WordNeti.

Sõnadevaheline kaugus Li jt (2006) mõistes saab arvutada valemiga

---

[18] Python Software Foundation (n.d.). About Python. https://www.python.org/about/ (vaadatud 08.05.2024)
[19] Laur, S., Orasmaa, S., Särg, D., Tammo, P. (2020). EstNLTK 1.6: Remastered Estonian NLP Pipeline. *Proceedings of The 12th Language Resources and Evaluation Conference pages 7154--7162*, https://aclanthology.org/2020.lrec-1.884
[20] Laur, S., Orasmaa, S., Särg, D., Tammo, P. (n.d.). Basic API of EstNLTK 1.7. https://github.com/estnltk/estnltk/blob/main/tutorials/basics/introduction_to_estnltk_api.ipynb (vaadatud 08.05.2024)
[21] ibid
[22] ibid

$$e^{-alpha \cdot path\_length(w_1, w_2)}$$

kus *e* on Euleri arv, *alpha* on koefitsient, mis sõltub kasutatavast sõnastikust kauguse arvutamiseks ja *path_length()* on kaugus sõnade vahel.
Selles töös oli otsustatud kasutada eestikeelse WordNetiga suhtlemiseks EstNLTK, sest see pakub põhjalikku dokumentatsiooni ning sest see oli juba kasutatud eelmises sammus. Kuna EstNLTK ei paku oma moodulis *estnltk.wordnet*[23] funktsiooni kauguse arvutamiseks, siis oli selle saamiseks esialgu kasutatud funktsioon *estnltk.wordnet.Wordnet.path_similarity()*[24], mis arvutab sõnade sarnasust, ning pärast seda oli tehtud tehe

$$\left(\frac{1}{path\_similarity(w_1, w_2)}\right) - 1$$

mis on vastupidine tehe sellele, mida kasutatakse sõnade sarnasuse arvutamiseks sõnade kaugusest.

Li jt (2006) väidavad, et parimaks *alpha* koefitsiendi väärtuseks ingliskeelse WordNeti jaoks on 0,2. Selles töös oli see väärtus kasutatud ka eestikeelse WordNeti jaoks.

Veel väidavad Li jt (2006), et mida sagedamini sõnad esinevad tekstikorpuses, seda vähem on nende tähtsus. Nad on kasutanud selle omaduse sõnade semantilise sarnasuse arvutamises. Iga w_i puhul on nad arvutanud sõna informatiivsust kasutades valemi

$$1 - \frac{log(n+1)}{log(N+1)}$$

kus *N* on kõikide sõnade arv tekstikorpuses ja *n* on antud sõna esinemiste arv korpuses. Pärast sõnade informatiivsuse arvutamist olid *š1* ja *š2* väärtused vastava sõna informatiivsusega.

[23] Laur, S., Orasmaa, S., Särg, D., Tammo, P. (n.d.). Wordnet. https://github.com/estnltk/estnltk/blob/main/tutorials/wordnet/wordnet.ipynb (vaadatud 08.05.2024)
[24] ibid

Li jt (2006) on kasutanud sõnade informatiivsuse leidmiseks Brown korpuse. Selles töös oli aga kasutatud Eesti kirjakeele sagedussõnastik[25], täpsemalt 10000 kõige sagedasemat lemmat. Sõnade sagedused on sõnastikus välja toodud, ja nende koguarv oli arvutatud kõikide sõnade sageduste summana.

Sõnade sügavuse arvutamiseks pakuvad Li jt (2006) valemi

$$\frac{(e^{beta \cdot h} - e^{-beta \cdot h})}{(e^{beta \cdot h} + e^{-beta \cdot h})}$$

kus *beta* on samuti koefitsient, mis sõltub kasutatud sõnastikust ja *h* on kaugus esimese ühise ülemmõiste ja juurülemmõiste vahel.

Li jt (2006) on kasutanud *beta* koefitsiendi väärtuseks 0,45, seetõttu ka selles töös oli see kasutatud.

Sõnade esimese ühise ülemmõiste leidmiseks oli selles töös kasutatud funktsioon *estnltk.wordnet.Wordnet.lowest_common_hypernyms()*[26], juurülemmõiste leidmiseks *estnltk.wordnet.Synset.root_hypernyms()*[27]. Siis nende vahel oli arvutatud kaugus nagu see oli tehtud üleval.

Kui nii sõnade kaugus, kui ka nende sügavus oli leitud, siis need olid korrutatud, nagu on kirjas Li jt (2006) juhises.

Kui vektorid *š1* ja *š2* olid täidetud, siis lausete semantiline sarnasuse arvutamiseks saab Li jt (2006) sõnul kasutada valemit

$$\frac{\text{š1} \cdot \text{š2}}{||\text{š1}|| \cdot ||\text{š2}||}$$

kus lugeja on kahe vektori skalaarkorrutis ja nimetaja on vektori pikkuste korrutis ehk selle liikmete ruutude summa ruutjuurde korrutis.

---

[25] Kaalep, H.J., Muischnek, K. (2002). Eesti kirjakeele sagedussõnastik. *TÜ kirjastus.* https://www.cl.ut.ee/ressursid/sagedused/index.php?lang=et
[26] ibid
[27] ibid

Peale lausete semantilist sarnasust on Li jt (2006) sõnul vaja arvutada nende lausete sõnade järjekorra sarnasust, sest nende arvates saavad ka sama sõnadega laused omada erinevat tähendust, kui need sõnad on erinevas järjekorras.

Sõnade järjekorra võrdlemiseks pakuvad Li jt (2006) luua täisarvulised vektorid *r1* ja *r2*, mis on ka sama pikkusega kui hulk *T*. Taaskord itereeritakse üle iga sõna w_i hulgas *T*. Iga sõna w_i puhul vaadatakse selle positsiooni sõnas *T1*. Kui seda selles lauses ei ole, siis vaadatakse sellele kõige sarnasema sõna positsiooni, kus kõige sarnasem sõna oli juba leitud semantilise sarnasuse arvutamisel. Siis kohale $r1_i$ pannakse see positsioon. Sama protsess tehakse ka vektoriga *r2*.

Teades *r1* ja *r2* väärtused, Li jt (2006) arvutavad lausete sõnade järjekorra sarnasust järgmiselt:

$$1 - \frac{||r1 - r2||}{||r1 + r2||}$$

Kui nii lausete semantiline sarnasus, kui ka lausete sõnade järjekorra sarnasus on teada, siis lausete sarnasuse leidmiseks saab Li jt (2006) sõnul kasutada valemit

$$gamma \cdot semantic\_similarity + (1 - gamma) \cdot word\_order\_similarity$$

kus *semantic_similarity* on lausete semantiline sarnasus, *word_order_similarity* on sõnade järjekorra sarnasus ja *gamma* on koefitsient, mis määrab mõlema tähtsust. Li jt (2006) ütleb *gamma* kohta vaid see, et see peab olema rohkem kui 0,5, sest semantiline sarnasus mängib suuremat rolli, kui sõnade järjekord. O'Shea (2013) aga pakub konkreetne väärtus 0,8, mis oli ka selles töös kasutatud.

4.3 Ülejäänud komponentide implementeerimine

Sisendi dekooder ja vastuse väljastuse komponent videomängu puhul suurel määral sõltuvad mängumootorist, kuna just mängumootor vastutab madalatasemeliste kompoentidega, milleks on sisend ja väljund, suhtlemise eest. Kuna loomuliku keele mõistmise komponendi implementatsioon eeldab keele Python kasutamist, siis

mängumootor peab toetama Python keeles skriptimist. Otsus oli tehtud Pygame[28] mootori kasuks, sest sellel mootoril on olemas põhjalik dokumentatsioon ja suur kogukond kasutajaid, kelle käest saab probleemide korral nõu küsida.

Sisendi dekooder oli implementeeritud kasutades Pygame sündmuste süsteemi. Kõik toimunud sündmused olid saadud funktsiooniga *pygame.event.get()*[29], pärast seda nendest otsiti *pygame.KEYDOWN*[30] tüüpi sündmusi. Klahvide puhul, millel on olemas *event.unicode*[31] väärtus, see väärtus lisatakse sõnumi muutuja lõppu. Klahvi *pygame.K_BACKSPACE*[32] ehk tagasilükkeklahvi puhul sõnumi muutujast kustutatakse viimane sümbol. Klahvi *pygame.K_RETURN*[33] ehk *enter* klahvi ehk sisestusklahvi puhul sõnumi muutuja sisu saadetakse edasi keele mõistmise komponendile.

Vastuse väljastuse komponent kasutab Pygame klassi *pygame.font.Font*[34], mis loeb sisse kirjatüüpi *.ttf* formaadis, ning selle meetodi *render()*[35], millele antakse argumendina sõnumi sisaldavat muutujat ning mis renderdab antud teksti antud kirjatüübiga. Seda saab siis näidata ekraanil meetodiga *pygame.Surface.blit()*[36].

Dialoogihaldaja valmistamisel oli tähtis teha seda nii, et hiljem oleks dialoogide testimine võimalikult kiire ja raskusevaba. Lähenedes sellest põhimõttest oli dialoogihaldaja loomisel kasutatud rumm-ja-kodar lähenemist nagu see oli Freed (2014a) poolt kirjeldatud, kuna seal on igas dialoogis ainult üks hargnemise koht. Oli loodud klass dialoogi repliigi jaoks, mis sisaldab repliigi teksti ja järgmist repliiki. Veel oli loodud klass valitava repliigi jaoks, mis peale ülalmainitud väljade sisaldab ka eeltingimuse ja efekti väljad, mis on mõlemad *Callable*[37] tüüpi ehk millele on võimalik omistada lambda-funktsioonid. Tänu sellele on võimalik lugeda ja teha

---

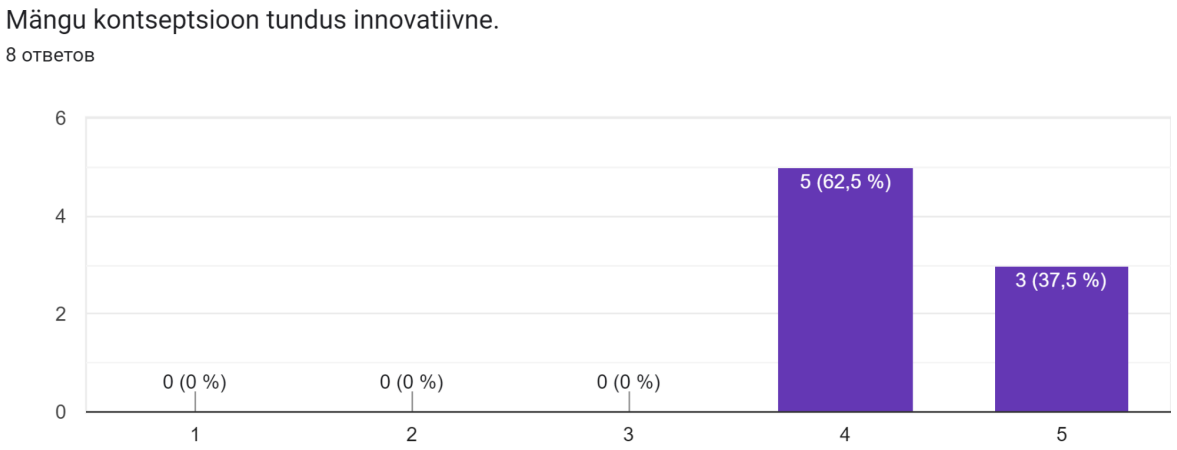
[28] Pygame (n.d.). About. https://www.pygame.org/wiki/about (vaadatud 08.05.2024)
[29] Pygame (n.d.). pygame.event https://www.pygame.org/docs/ref/event.html (vaadatud 08.05.2024)
[30] Pygame (n.d.). pygame.key https://www.pygame.org/docs/ref/key.html (vaadatud 08.05.2024)
[31] ibid
[32] ibid
[33] ibid
[34] Pygame (n.d.). pygame.font https://www.pygame.org/docs/ref/font.html (vaadatud 08.05.2024)
[35] ibid
[36] Pygame (n.d.). pygame.Surface https://www.pygame.org/docs/ref/surface.html (vaadatud 08.05.2024)
[37] Python Software Foundation (n.d.). Typing — Support for type hints. https://docs.python.org/3/library/typing.html (vaadatud 08.05.2024)

operatsioone globaalsete muutujatega ja seetõttu integreerida mängusse domeenispetsiifilised komponendid.

Lõpuks oli loodud dialoogihaldaja klass, mis sisaldab dialoogi esimest repliiki, dialoogi praegust repliiki ning listi valitavatest repliikidest. Sellel klassil oli loodud meetod *advance()*, mis leiab praegusele repliigile järgneva repliigi ja omistab seda praeguse repliigi väljale. Oli loodud ka meetod *choose()*, mis võtab argumendina sõnet ning iga repliigi puhul valitavate repliikide listist kontrollib selle eeltingimust ning leiab loomuliku keele mõistmise komponendi abil selle sarnasust argumendina antud repliigiga. Kõige kõrgema sarnasuse skooriga repliik valitakse, käivitatakse selle efekt ning praeguseks repliigiks omistatakse sellele järgnev repliik.

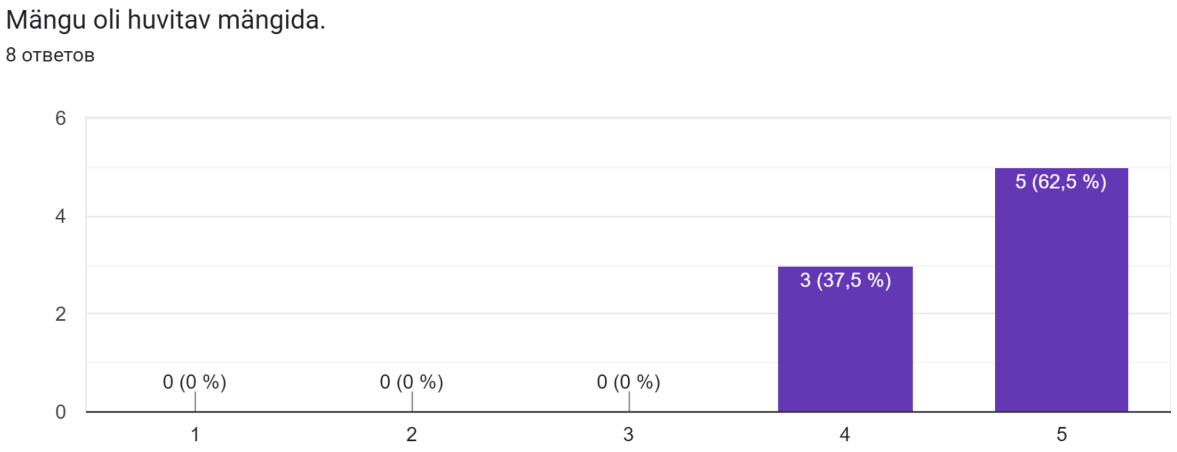
5 Mängu testimine

Pärast mängu valmimist oli korraldatud ka selle testimine. Mängu testimises osalesid teised Tartu Ülikooli tudengid. Kokku oli kaheksa vastajat. Nad pidid proovima mängu ning täitma küsimustiku. Küsimustiku eesmärk oli teada saada, kui korrektselt ja kui kiiresti töötab dialoogsüsteem, ning kas sellel põhinevat mängu on üldse huvitav mängida või mitte. Selles peatükis analüüsitakse saadud vastusi küsimustikule.



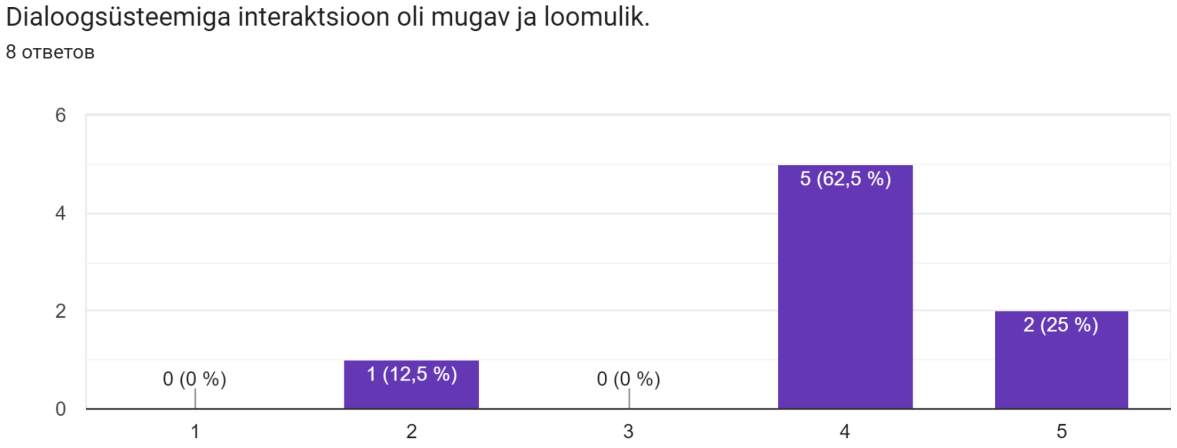
Joonis 10. Vastajate nõusolek väitega “mängu kontseptsioon tundus innovatiivne”, kus hinne 5 tähendab täielikku nõusolekut ja hinne 1 tähendab nõusoleku puudumist.

Nagu demonstreeritud joonisel 10, testijad kõrgelt hindasid mängu innovatiivsust. See tähendab, et nende arvates on selle mängu dialoogsüsteemi implementatsioon märkimisväärselt erinev sellest, mida tavaliselt saab näha mängus. Seda tõestavad ka testijate kommentaarid (saadaval koos küsimustikuga failis *survey/survey.csv*). Näiteks, üks testija väidab, et kasutatud dialoogsüsteem on “uus lähenemine”.



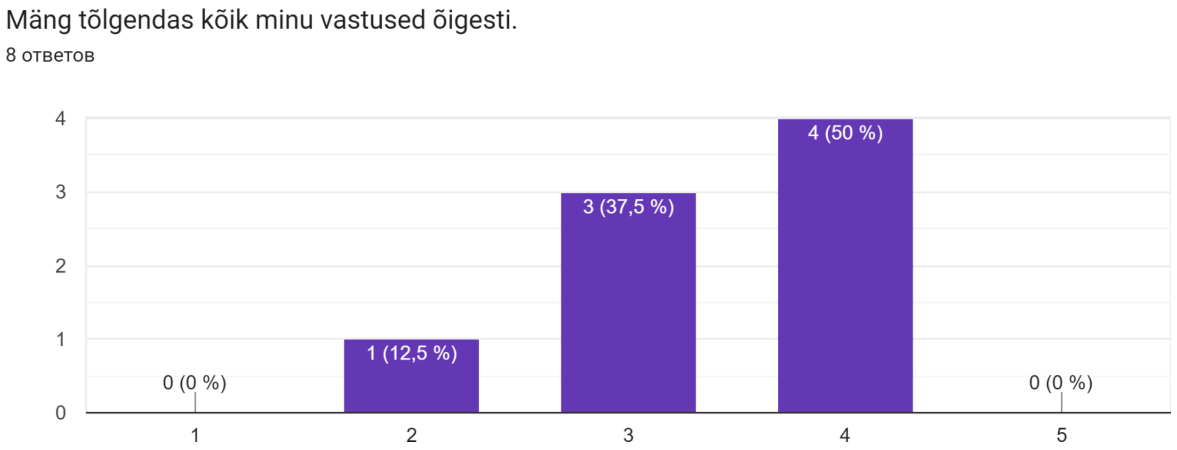
Joonis 11. Vastajate nõusolek väitega “mängu oli huvitav mängida”, kus hinne 5 tähendab täielikku nõusolekut ja hinne 1 tähendab nõusoleku puudumist.

Peale selle, joonis 11 demonstreerib, et testijad leidsid mängu väga huvitav olevat. Enamik saadud hinnetest olid viied. See tähendab, et mängu kontsepti implementeerimine oli edukas. Kuid on mõne testija poolt toodud esile ka üks puudus: sisestatud repliigi analüüs ja korrektse jätku leidmine võtab päris palju aega.



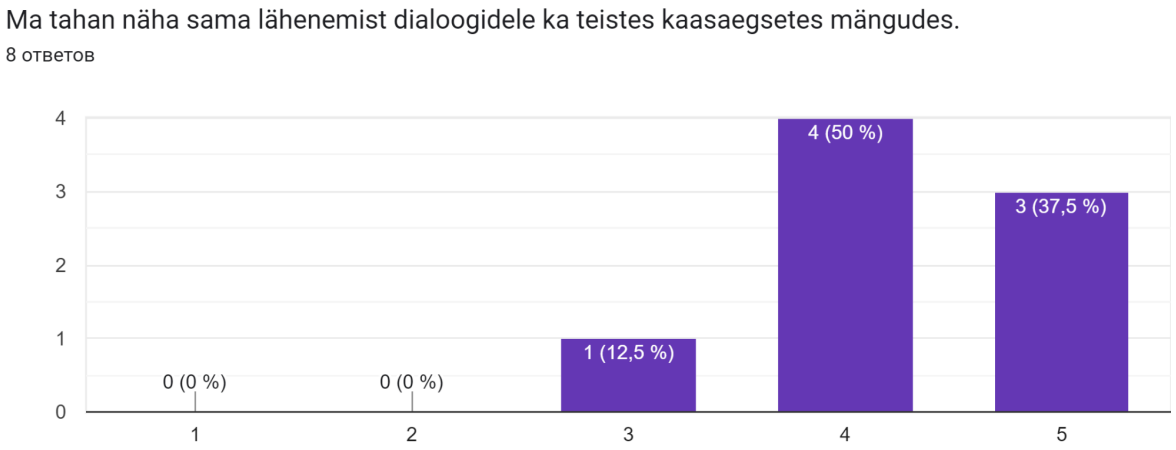
Joonis 12. Vastajate nõusolek väitega “dialoogsüsteemiga interaktsioon oli mugav ja loomulik”, kus hinne 5 tähendab täielikku nõusolekut ja hinne 1 tähendab nõusoleku puudumist.

Joonisel 12 on näidatud, et ka inimese interaktsioon dialoogsüsteemiga enamasti ei tekitanud probleeme. Kuid ühe vastaja arvates ei olnud see piisavalt mugav. Aga tema ei toonud näiteid, täpselt mis interaktsiooni aspekt saaks parem olla.



Joonis 13. Vastajate nõusolek väitega “mäng tõlgendas kõik minu vastused õigesti”, kus hinne 5 tähendab täielikku nõusolekut ja hinne 1 tähendab nõusoleku puudumist.

Kui rääkida sellest, mida testijad arvasid nende sisendi tõlgendamist mängu poolt, siis joonise 13 järgi oli see vaid keskmisel tasemel. Ühes vastuses oli põhjusena toodud välja see, et mäng ei suutnud valida õiged vastused juhul, kui mängija andis lühikesi vastusi.



Joonis 14. Vastajate nõusolek väitega “ma tahan näha sama lähenemist dialoogidele ka teistes kaasaegsetes mängudes”, kus hinne 5 tähendab täielikku nõusolekut ja hinne 1 tähendab nõusoleku puudumist.

Testijatele oli esitatud ka küsimus, kas nad tahaksid näha loomuliku keele töötlusel põhinevaid dialoogsüsteeme ka teistes kaasaegsetes mängudes. Nagu on näidatud joonisel 14, ka sellele küsimusele vastasid testijad enamasti positiivselt. Sellest võib järeldada, et sellel kontseptil on olemas potentsiaal.

6 Kokkuvõte

Käesoleva töö eesmärgiks oli luua mäng, mis kasutaks loomuliku keele töötlusel põhinevat dialoogsüsteemi ja kus see dialoogsüsteem töötleks eesti keelt. Selleks oli üle vaadatud kirjandus, mis käsitleb dialoogsüsteemi struktuuri. Veel oli analüüsitud see, kuidas on dialoogsüsteemid implementeeritud videomängudes. Mängu loomise protsessis oli katsetatud lähenemine, mis kasutab keelemudeleid, kuid oli leitud, et vahendeid eesti keeles lausete võrdlemiseks keelemudelite abil on puudu. Lõpuks oli loodud dialoogsüsteem, kus lausete võrdlemine töötab eestikeelse WordNeti ja Eesti kirjakeele sagedussõnastiku põhjal. Mäng sai testijatelt enamasti positiivset tagasisidet, seetõttu saab väita, et püstitatud eesmärk oli saavutatud.

Testimise jooksul oli välja selgitatud üks tähtis aspekt, mis võiks olla tehtud paremini. Loodud dialoogsüsteem võtab päris pikka aega lausete võrdlemiseks. Ja peamiseks mängu edasiarendusvõimaluseks on dialoogsüsteemi kiiruse optimeerimine. Võiks uurida, kui hästi mäng töötab kui oleks puudu mõni keeletöötluse samm, näiteks sõnade järjekorra sarnasuse arvutamine.

Veel üks võimalus, kuidas saaks mängu edasi arendada, on suurema dialoogide arvu loomine. Kuna loodud mäng on põhimõtteliselt prototüüp, siis selle jaoks oli loodud ainult nii palju dialooge, et demonstreerida kontsepti töövõimet. Kui arendada selle dialoogsüsteemiga suuremat mängu, siis saaks katsetada ka, kui hästi see saab hakkama keerulisemate dialoogide struktuuridega, kus näiteks on olemas mitu kümmet erinevat varianti, mida mängija saab öelda.

Mis puutub eesti keele töötluse teemasse, siis selle edasiarendamiseks on olemas palju võimalusi.

- Dialoogsüsteemi implementeerimisel oli vajalik mõnes kohas kasutada koefitsiente, mis sõltuvad kasutatud vahendist. Selles töös olid kasutatud samad koefitsientide väärtused, mis olid pakutud ingliskeelsetes allikates. Saaks uurida, mis on optimaalsed koefitsientide väärtused vahendite jaoks, mis töötlevad eesti keelt.

- Loodud dialoogsüsteem kasutab vananenud lähenemist, kuna uuem BERTil põhinev lähenemine eeldab siirdeõppeks vajaliku andmestiku olemasolu, millega sellele saab õpetada tervete lausete võrdlemist. Saaks luua sellise andmestiku eestikeelsete andmetega selleks, et EstBERT saaks ka seda ülesannet täita.
- Juhul, kui antud andmestik on loodud, saab võrrelda ka EstBERTi ja WordNeti põhjal tehtud dialoogsüsteemide kiirust ja täpsust.

Viidatud kirjandus

Algherairy, A., Ahmed, M. (2023). A review of dialogue systems: current trends and future directions. *Neural Computing and Applications*, doi: 10.1007/s00521-023-09322-1

Cannings, N. (2023). Unlocking Maximum Inference Capability: A Deep Dive into Llama2–70B on an 80GB A100 GPU. *Medium.com* https://nigelcannings.medium.com/unlocking-maximum-inference-capability-a-deep-dive-into-llama2-70b-on-an-80gb-a100-gpu-2ab1158d6b0b (vaadatud 08.05.2024)

Freed, A. (2014a). Branching Conversation Systems and the Working Writer, Part 2: Design Considerations. *Game Developer Magazine* https://www.gamedeveloper.com/design/branching-conversation-systems-and-the-working-writer-part-2-design-considerations (vaadatud 08.05.2024)

Freed, A. (2014b). Branching Conversation Systems and the Working Writer, Part 3: Building a Conversation Tree. *Game Developer Magazine* https://www.gamedeveloper.com/design/branching-conversation-systems-and-the-working-writer-part-3-building-a-conversation-tree (vaadatud 08.05.2024)

Li, Y., McLean, D., Bandar, Z., O'Shea., J.D., Crockett, K. (2006). Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 8, pp. 1138-1150*, doi: 10.1109/TKDE.2006.130

Macherey, K., Och, F.J., Ney, H. (2001). Natural language understanding using statistical machine translation. *Proc. 7th European Conference on Speech Communication and Technology (Eurospeech 2001)*, 2205-2208, doi: 10.21437/Eurospeech.2001-520

Mateas, M., Stern, A. (2003). Façade: An Experiment in Building a Fully-Realized Interactive Drama. *Game Developers Conference, San Jose, April 2003* https://www.cp.eng.chula.ac.th/~vishnu/gameResearch/story_november_2005/MateasSternGDC03.pdf (vaadatud 04.12.2023)

O’Shea, K. (2013). Natural language scripting within conversational agent design. *Springer Science+Business Media New York*, 40:189–197, doi: 10.1007/s10489-012-0408-2

Park, Y., Ko, Y., Seo, J. (2022). BERT-based response selection in dialogue systems using utterance. *Expert Systems With Applications*, doi: 10.1016/j.eswa.2022.118277

Rudnicky, A., Thayer, E., Constantinides, P., Tchou C., Shern, R., Lenzo, K., Xu, W., Oh, A. (1999). Creating natural dialogs in the Carnegie Mellon University Communicator System. *Carnegie Mellon University*, doi: 10.21437/Eurospeech.1999-344

Ward, W., Issar, S. (1994). Recent improvements in the CMU spoken language understanding system*. Proceedings of the ARPA Human Language Technology Workshop, March 1994*, 213-216, doi: 10.3115/1075812.1075857

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Mihhail Mihhailov

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   Loomuliku keele töötlusel põhineva dialoogsüsteemi rakendamine videomängus,

   mille juhendaja on Siim Orasmaa,

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Mihhail Mihhailov
15.05.2024