Lassi Högander

Steam-kaupan arvostelujen prosessointi peli-markkinan toiveiden mittaamiseen

Tieto- ja viestintätekniikka,
Datasta tekoälyyn

Insinööri (AMK)

Kevät 2024

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä: Högander Lassi

Työn nimi: Steam-kaupan arvostelujen prosessointi pelimarkkinan toiveiden mittaamiseen

Tutkintonimike: Insinööri (AMK), Tieto- ja Viestintätekniikka, Datasta Tekoälyyn

Asiasanat: Steam, Extract Transfer Load (ETL), verkkokauppa, API, Docker, Python-ohjelmointikieli

Tässä opinnäytetyössä pyrittiin kehittämään järjestelmä, joka tiivistäisi, mitä asiakkaat sanovat videopelistä. Vuosien varrella on huomattu kaksi ongelmaa peliteollisuudessa: trendejä jahdataan liian vahvasti ja mitä suurempi yhtiö tekee videopeliä, sitä vähemmän se kuuntelee asiakkaiden huolia ja mielipiteitä. Ongelmien lähteeksi on usein syytetty pelien tekijöiden puutteellisen kyvyn kuunnella heidän asiakkaitansa. ”Mutta miksi he eivät vain kuuntele enemmän”? Tälle kysymykselle on yksinkertainen vastaus, ei ole aikaa. Jos sadantuhannen (100.000) arvostelun joukosta kymmenentuhatta (10.000) ihmistä valittaa samasta asiasta, kuinka monta näistä arvosteluista pitäisi lukea ennen kuin asialle pitäisi tehdä jotakin? Se ei vaan ole järkevää laittaa ketään lukemaan kaikkia arvosteluja. Ja vaikka joku lukisikin ne, miten arvostelijoiden lukijan kuuluisi raportoida ongelmat? Ja tämä työvuori olisi jokaiselle pelille. Yhtiön näkökulmasta tämä on resurssien tuhlausta, mutta asiakkaan näkökulmasta tämä näyttää loukkaukselle.

Tässä työssä suunniteltiin, rakennettiin ja esiteltiin verkkopalvelu, josta videopelimarkkinoita tutkiva henkilö voisi saada Valve-korporaation Steam-kaupan verkkokauppapaikan arvosteluihin perustuvaa dataa. Työssä käytettiin git-versionhallintaa ohjelmakoodin tallentamiseen, Docker-virtuaalikoneita järjestelmän palveluiden pystytykseen, Flask-kirjastoa palveluiden datan siirtoon, Valven Steamworks API-työkaluja datan keräämiseen, Pandas-kirjastoa datan prosessointiin ja sen muodon hallintaan, MongoDB-tietokantaa datan tallentamiseen ja lopuksi TensorFlow-kirjastoa neuroverkon rakentamiseen. Neuroverkko opetettiin Steam-kaupan datalla.

Työn päätavoite saavutettiin, mutta lopputulokset eivät olleet tarkkoja. Tämä johtuu siitä, että työssä oleva luonnollisen kielen datan esiprosessointiskriptin luominen vei enemmän aikaa kuin muut työn osat, ja siltikään se ei ole riittävän perusteellinen data-aineiston puhdistuksessa. Tästä syystä monet sanat, jotka tarkoittavat samaa asiaa, lasketaan eri sanoina, ja toisinpäin että sanat, jotka kirjoitetaan samalla tavalla, mutta tarkoittavat eri asiaa, lasketaan samoiksi sanoiksi. Tämä hämmentää neuroverkkoa, jolloin sen tarkkuus pienentyy. Tästä kehityksen jatkaminen on kuitenkin helppoa, mutta aikaa kuluttavaa. Tämä datan esiprosessi on avain, jolla saadaan englanninkielisten Steam-kaupan asiakkaiden mieltymykset mitatuksi.

Docker oli ennestään tuttu työkalu palveluiden sekä niiden verkkoasetusten hallinnan, ylläpitoon ja rakentamiseen. Se yksinkertaistaa versiohallintaa sekä sillä voi helposti luoda turvallisen eristyksen mikropalveluiden kesken, sekä emokoneeseen. Dockerin sisäiset toiminnat helpottivat myös palveluiden välisen viestinnän kehitystä. Dockerin käyttö sujuvoitti erityisesti järjestelmän pystyttämistä ja hallintaa. Myös mahdollinen palveluiden skaalaaminen onnistuu Docker-työkalujen avulla.

Tietenkin koodin voisi optimoida vaihtamalla esimerkiksi Pandas-kirjaston uudempaan ja tehokkaampaan Polars-kirjastoon tai vaihtamalla paljon muistia vaativan MongoDB-tietokannan relaatiotietokantaan, kuten MariaDB. Polars-kirjastoon perehdyin vasta työn loppupuolella ja relaatiotietokannan käyttö olisi pidentänyt työnteon aikaa paljon. Mikäli järjestelmää kehittäisi vielä eteenpäin, se voisi olla vahva työkalu antamaan peliyhtiöille tai markkina-analytiikoille silmät ja korvat markkinoilla sanottaviin asioihin sekä toimimaan esimerkiksi pankille todisteena mahdollisena lainan hankintaan tai todisteena asiakkaiden mielipiteistä tietystä osaa peliä. Tällä hetkellä tämä työ on vain esimerkki siitä, mikä olisi mahdollista.

KAMK • University of Applied Sciences

Abstract

Author(s): Högander Lassi

Title of the Publication: Processing Steam Store's Review Data for Measuring Gaming Market's Demands

Degree Title: Bachelor of Engineering, Information and Communication Technology, from Data to Artificial Intelligence

Keywords: Steam, Extract Transfer Load (ETL), online store, API, Docker, Python-language

This thesis attempts to make a system which compresses the information what customers are saying about a game. In recent years, the gaming industry has had two self-harming trends: trends are chased too strongly and, the bigger the developer company, the less they listen to customers' complaints and worries. The cause of these problems was identified to be lack of ability to listen to the market's customers. This raises the question of why they won't just listen more? The answer is, there just isn't enough time. If in a hundred thousand (100.000) reviews ten thousand (10.000) complain about the same thing, how many of these reviews must be read until some action is taken? It just isn't efficient to have some people reading these manually. Even if they were, how would these problems be reported? On top of that, their work pile could exist for multiple games. From a company's perspective, this endeavour would just be a waste of resources. But from a customer's perspective, not doing that would be insulting.

In this project, a service was planned, built and presented which could be used by a person researching about gaming market's demands using Valve Corporation's Steam Store's review data. In the development, GIT was used for version control and progress upkeep. Docker was used for server deployment and network handling. Flask library for Python was used for server building and for data transfer. Valve's Steamworks API was used for collecting data. Pandas library for Python was used for data processing and moulding. MongoDB database architecture was used for data storing and transfer. Lastly, TensorFlow for Python was used for building a neural network which was taught using Steam Store's review data.

The project's main goal was achieved but the results were not very good. This is because this project's natural language preprocessing script took much more time than other tasks. And even then, it was not thorough enough. Therefore, many different words that mean the same thing are counted as different words and words that have many meanings count as the same word. This confuses the neural network which lowers its accuracy. This data's preprocessing is the key to measuring the customer's needs using English Steam reviews.

Docker was a familiar tool for building and configuring services. It simplifies version control and creates a safe quarantine from the host computer. Features in Docker made it easier to make safe communication between different services. The ease of deployment and control of servers streamlined the testing and development. The possible future service scaling is easy to configure using Docker's tools.

The code could be optimised in a variety of ways. Examples include changing the data processing and moulding library Pandas to newer and more optimised Polars or changing the database to a relational SQL based database like MariaDB. During the project, Polars only came into acknowledgement at the tail end, and using MongoDB instead of MariaDB was much simpler which allowed more time to be focused on other tasks. If the project would be fully realised and polished, it could be a powerful tool for giving game makers and market analysts eyes and ears for what the gaming market is discussing. For now, this just stands as a proof of concept.

Alkusanat

Kun olin suorittamassa asepalvelustani, yksi lause on jäänyt vahvasti mieleen tähän päivään asti: ”älä valita vaan, tee asialle jotain”. Tämä lause tulee aina mieleen, kun kuulen tai näen jotakin, mistä valittaisin. Koronavuosien eristyksen aikana tämä lause tuli järkyttävän usein päähäni, kun katsoin mitään videopeleihin liittyviä uutisia. Jostakin syystä nämä suuret peliyhtiöt eivät vain osanneet kuunnella heidän asiakkaitaan. Ehkä ei heti tule valmista ratkaisua tällä, mutta tämä on pohjatyö, jolla voisi tehdä asialle jotain.

Sisällys

Liitteet

1 Johdanto

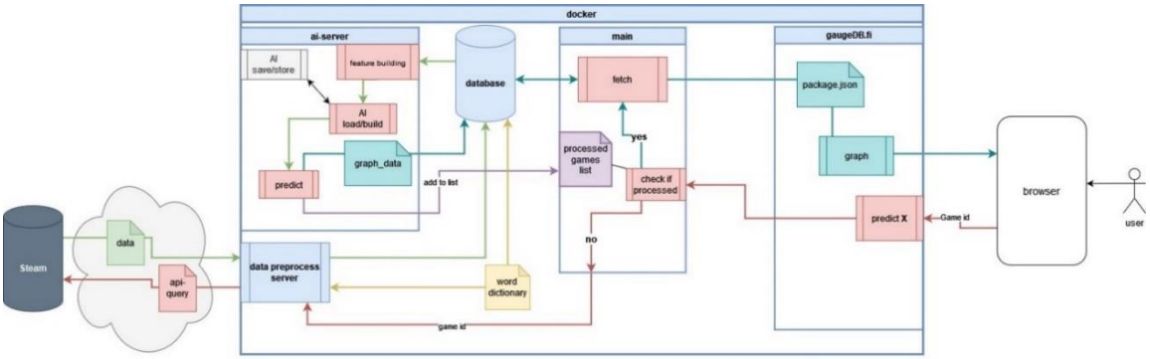
Videopelimarkkinoiden analysointiin löytyy alkeellisia työkaluja, jotka kertovat vain olemassa olevien pelien myyntiluvuista ja arvosteluista. Esimerkiksi arvosteluista saa tietoa käyttämällä Steam Review Explorer [1] tai Google Trends -työkalua [2], jotka auttavat selvittämään, kuinka paljon tietystä pelistä etsitään tietoa. Mutta mitä sitten pitäisi tehdä, jos esimerkiksi peliyhtiö haluaisi tietää mitä markkinoilla halutaan? Miten pitäisi saada järkevässä ajassa selville, mistä on pelimarkkinoilla kysyntää?

Kysynnän selvittämiseen yhtiön pitäisi tehdä omaa markkina-analyysia, jossa menisi resursseja, mitä heillä ei välttämättä ole. Jos yhtiöllä ei ole resursseja tehdä analyysiä, he joutuisivat tulemaan julkiseksi pelistään ja jälkeenpäin arvioimaan, oliko pelille kysyntää. Se ei ole kannattavaa, jos markkina on kylläinen heidän pelinsä genrestä juuri sillä hetkellä. Ei heidän pelinsä saisi mitään omaa julkisuutta tai tilaa markkinoilla. Tällöin myynnit kärsisivät. Tämä työ ei taianomaisesti ratkaise tätä pulmaa suoraan, vaan tekee pohjatyön tämän ongelman ratkaisemiseksi.

Ratkaisu olisi työkalu, johon peligenrejä tai avainsanoja syöttämällä saisi tietoa, kuinka paljon näille genreille tai niitä vastaaville avainsanoille on kysyntää pelimarkkinoilla. Tämä työkalu vaatisi pohjalle tavan hakea tieto markkinoilta, tallentaa ne, analysoida ne, muodostaa ajallisesti jaetut tietopaketit, muodostaa markkinatietoa tietopaketteja käyttäen ja muodostaa arvio esiteltävään sekä ymmärrettävään muotoon ja tuoda tämä käyttäjän eteen. Tämä työ keskittyy toimivuuteen, modulaarisuuteen ja automatisointiin, joten lopputuloksen tarkkuus sekä lopputuloksien esittely muoto eivät ole tärkeitä. Tavoitteena on luoda ohjelmistokokonaisuus, jossa Steam-kaupassa (engl. Steam Store) olevan pelin englanninkielisten arvostelujen data kerätään, puhdistetaan, tallennetaan, muokataan ja esitetään saman vuorokauden aikana. Tämä ilman, että kelvollista dataa katoaa matkan varrella. Tämän työn on tarkoitus olla laaja mutta ei syvä. Jos jotkin työn vaiheet eivät tunnu tulevan valmiiksi, niin ne jätetään keskeneräisiksi ja kirjataan keskeneräisyyden syy.

2 Järjestelmän arkkitehtuuri

Työn turvallisuuden ja ylläpidon helppoutta nostaa palveluiden suorittaminen virtuaaliympäristössä. Virtuaalikoneympäristöjen hallintaan käytettiin Docker-sovellusta [3]. Docker-termein suoritusympäristöä on tapana kutsua kontiksi. Tässä työssä kaikissa konteissa pyörii Pythonilla toteutettu Flask-mikropalvelin [4]. Poikkeuksena on kontti, mihin tallennetaan kaikki valmis data, eli tietokantakontti, jossa pyörii MongoDB [5]. Flask ja MongoDB valittiin työkaluiksi, koska ne olivat molemmat entuudestaan tuttuja ja MongoDB:n kanssa kehittäminen on nopeaa ja helppoa. Se on kuitenkin tietokoneen resursseille kalliimpaa. Näitä Docker-kontteja on viisi kappaletta, joista jokaisella on erikoistunut rooli systeemissä. Yksi kerää arvosteludatan, sekä esikäsittelee sen. Seuraavaksi data tallennetaan tietokantakontin sisälle. Lopuksi data voidaan siirtää eteenpäin jatkokäsittelyyn ja tekoälyyn tarkoitetulla palvelimella, josta sen tulokset tallennetaan takaisin tietokantaan. Näihin tuloksiin pääsee käsiksi valvontapalvelin, joka toimii asiakaspalvelimen ja kehityspuolen välikätenä. Konttien välinen yhteys ja tiedonsiirto on kuvattu kuvassa 1. Kuvassa 1 on halutun pelin nimi tai tuotenumeron kulku merkattu punaisella, datan kulku merkattu vihreällä ja valmiiden tuloksien kulku merkattu tumman turkoosilla.



Kuva 1. Järjestelmän arkkitehtuuri

Tämän lisäksi kehityksen, sekä tulosten laadun tarkistamisen helpottamiseen käytettiin MongoDB-tietokannan sisällön tutkimiseen kehitettyä Mongo Express –sovellusta [6]. Tämän avulla on helppo nähdä, mitä tietokantaan on tallennettu. Järjestelmän osat on myös rakennettu skaalautuvaksi. Datan esiprosessipalvelin vaatii paljon muistia ja tehokkaan prosessorin. Tietokanta tarvitsee paljon tallennustilaa. Tekoälypalvelin hyötyisi nykyaikaisesta näytönohjaimesta tai vaihtoehtoisesti tensorisuorittimesta, jotka on kehitetty vain neuroverkkojen kehitystä ja suorittamista ajatellen. Valvonta- ja demopalvelimet tarvitsevat vain paljon muistia.

3 Steam-kaupan data

Tässä luvussa selitetään työssä käytetyn arvosteludatan lähde, kerääminen ja muoto. Ensin selitetään Valven Steam, ja mikä tekee siitä hyvän datan lähteen tähän työhön. Tämän jälkeen selitetään, millä työkalulla data kerätään ja mitä päätöksiä tähän on tehty. Sen jälkeen näytetään, minkä muotoista dataa tulee kerättyä ja miltä se näyttää Jupyter Notebook -ympäristössä.

3.1 Steam

Vuonna 2003 Valve korporaatio julkaisi Steam-sovelluksen, jota käyttäen Valve ja heidän asiakkaansa pystyivät päivittämään videopelejä internetin välityksellä [7]. Vuonna 2005 Valve julkaisi Steam Store -verkkokaupan, josta pystyy ostamaan videopelejä ja pelien kehittäjät pystyivät, Steamia käyttäen, korjaamaan tai kehittämään heidän julkaistuja pelejään. Steam-kaupassa on yli 50 000 peliä ja yli puolet kaikista tietokonepeleistä ladataan Steamin kautta [8]. Vuonna 2013 Valve lisäsi Steam-kauppaan peleihin kohdistuvan käyttäjäpohjaisen arvostelun, josta pelin julkaisija ei voinut jättäytyä pois. Arvostelun pystyy kirjoittamaan vain käyttäjä, joka on saanut pelin Steam-kirjastoonsa Steam-kaupasta, tai tuoteavaimella, jonka ovat ostaneet Steamin ulkopuolelta [9].

Näissä arvosteluissa on kaksi tärkeää tietoa. Ensimmäinen tärkeä tieto on, ehdottaako arvostelun kirjoittaja peliä muille: kyllä (sininen peukku ylös) tai ei (punertava peukku alas). Toinen tieto on itse tekstipohjainen arvostelu. Näitä arvosteluja voi Steamin käyttäjät arvioida avuliaaksi, avuttomaksi tai hauskaksi. Valven rakentama Steamworks on setti työkaluja, jotka on tarkoitettu pelien kehittäjien ja julkaisijoiden käyttöön [10], ja joiden kautta pystyy keräämään dataa Steamistä. Steamworks-settiin kuuluu rajapinta [11], jota käyttäen voi helposti kerätä Steam-kaupan arvosteluja.

3.2 Steamworks API

Ohjelmointirajapinta tai englanniksi Application Programming Interface (API) on termi, jota käytetään kahden erilaisen ohjelman väliseen kommunikaatioon. Koska palvelimet ovat ohjelmia, tämä termi pätee myös kahden palvelimen väliseen kommunikaatioon, tällöin nämä API-työkalut ovat vain URL-osoitteita. Steamworks sisältää API-osoitteita, joita kautta saadaan Steam-kaupan arvosteludataa. Palvelimien välisessä kommunikaatiossa pitää tehdä Hypertext Transfer Protocol -kutsuja (HTTP). Tämän työn ymmärtämiseen pitää tietää vain, että kun tehdään HTTP GET -kutsu, niin silloin lähetettään palvelimelle vähän dataa ja otetaan vastaan paljon dataa. Näitä vähäisiä datoja, joita lähetetään GET-pyynnöstä, kutsutaan parametreiksi.

Steamworks-arvosteludatan keräämiseen tarkoitettu HTTP GET-pyyntö näyttää tältä:

```
GET store.steampowered.com/appreviews/<appid>?json=1
```

`<appid>` tilalle kuuluu halutun pelin tai sovelluksen tuotenumero, joka löytyy Steam-kaupan URL-osoitteesta tai voi etsiä toisella API-työkalulla [12], joka antaa jokaisen Steam-kaupan tuotteen nimen ja tuotenumeron. Tätä käytetään työssä tarkistamaan, että haluttu tuote on edes olemassa. `Json`−parametrin arvo pitää olla yksi, mutta sen lisäksi on 8 mahdollista parametriä, joilla voi muuttaa saatavan datan muotoa tai sisältöä [13].

Taulukko 1. Datakeräyksen parametrit

Parametri	Selitys
`filter`	Missä järjestyksessä arvostelut tulevat?
`language`	Minkä kielistä dataa saadaan?
`day_range`	Kuinka monen päivän päästä vetää arvosteluja? Toimii vain, jos `filter=all`
`cursor`	Auttaa palvelinta ymmärtämään monesko pyyntö on kyseessä. Jokaisen pyynnön tuloksena on mukana seuraava Cursor-arvo. Ensimmäisessä pyynnössä pitää olla `cursor=*`
`review_type`	Arvo päättää, tuleeko vain positiivisia, negatiivisia tai merkattuja arvosteluja.
`purchase_type`	Arvolla voi erottaa, onko arvostelija hankkinut pelin Steamistä tai muualta.
`num_per_page`	Arvo päättää, montako arvostelua saadaan.
`filter_offtopic_activity`	Arvostelut, jotka ovat merkitty osaksi arvostelupommitusta, voidaan ottaa mukaan tai jättää pois tällä arvolla.

3.3 Arvosteludata

Kun aikaisemman kappaleen GET-pyyntö lähetetään Steamworksiin, palvelin lähettää takaisin json-muotoista dataa. Json on datastandardi, jota usein käytetään tiedon tallentamiseen ja palveluiden väliseen siirtoon tieto- ja viestintätekniikan (engl. Information and Communication Technology) alalla. Se sisältää monenlaista tietoa eri tasoilla. Taulukossa 2 on GET-pyynnön ensimmäisen tason avainsanat.

Taulukko 2. GET-pyynnön avainsanat

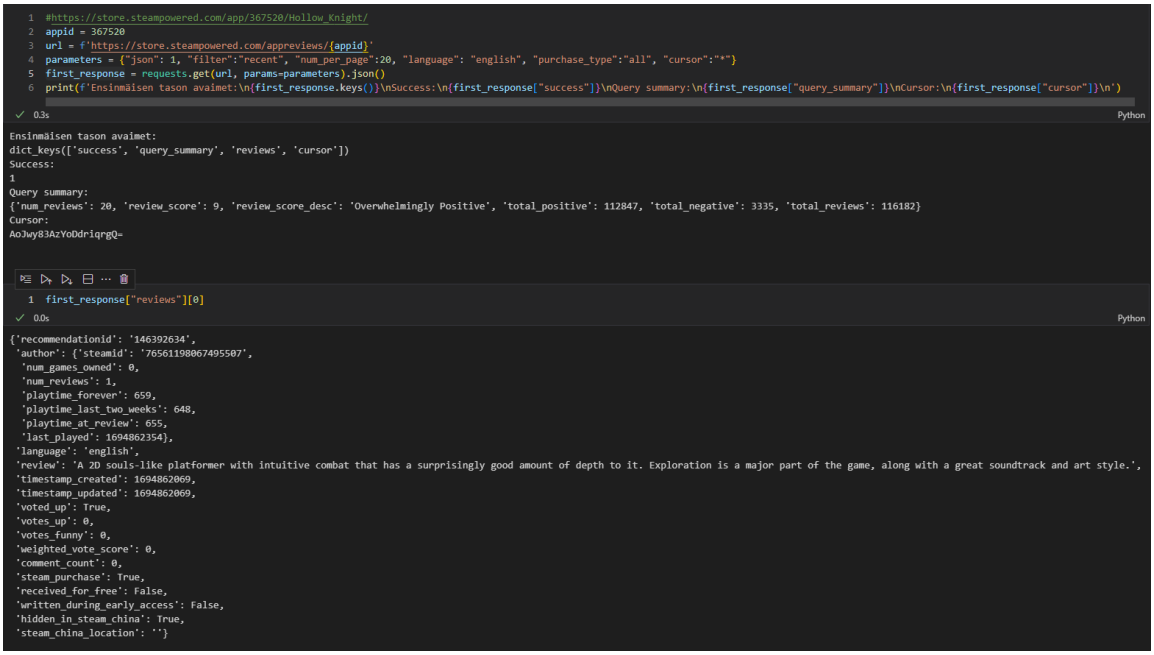
Avainsana	Sisältö
`success`	Jos ei ollut mitään ongelmia pyynnössä, tämän arvo on 1.
`query_summary`	Kirjasto, joka sisältää tuotteen yhteenlasketuja tietoja, esimerkiksi montako arvostelua annetuilla parametreillä saadaan yhteensä.
`cursor`	Teksti, jolla saa seuraavan setin dataa laittamalla tämän seuraavaan GET-pyynnön cursor-parametrin arvoksi.
`reviews`	Lista arvosteluolioita, jotka sisältävät yksittäisen arvostelun dataa.

`query_summary` sisältää arvon `total_reviews`, joka kertoo, kuinka monta arvostelua valituilla parametreillä voidaan saada yhteensä. Tätä lukua käyttäen voidaan laskea kuinka monta kutsua pitää tehdä. `Reviews`-arvosteluolioiden oleelliset avainsanat on listattu taulukkoon 3.

Taulukko 3. Työlle oleelliset avainsanat

Avainsana	Sisältö
`recommendationid`	Arvostelun ainutlaatuinen tunnus.
`author`	Objekti, joka sisältää kirjoittajasta olennaista tietoa esimerkiksi, kuinka kauan hän oli pelannut peliä, kun oli kirjoittanut arvostelun.
`review`	Itse käyttäjän kirjoittama arvostelu.
`timestamp_created`	Aika kun arvostelu luotiin.
`timestamp_updated`	Aika kun arvostelu oli viimeksi muutettu.
`voted_up`	Ehdottaako arvostelija peliä.
`votes_up`	Kuinka moni on arvioinut tämän arvostelun avuliaaksi.
`votes_funny`	Kuinka moni on arvioinut arvostelun hauskaksi.

Käyttämällä Pythonin standardikirjastoa ”requests”, voidaan tehdä HTTP GET -pyyntö Valven palvelimille, jotta saadaan arvosteludataa. Tästä esimerkki kuvassa 2, missä on käytetty ”Hollow Knight”-peliä esimerkkinä. Sen Steam-kaupan tuotenumero on 367520 [14].



Kuva 2. Steamworks API:n käyttöesimerkki

Kuvassa värillinen teksti on esimerkin koodi ja valkoinen teksti on, mitä se koodi tuottaa.

Kuvan ensimmäinen koodinpätkässä on pelin tuotenumero laitettu kappaleen 3.2 HTTP GET-pyyntöön. Parametreiksi on laitettu pakollinen `json` yhdeksi, `filter`-arvoksi ”recent”, joka järjestää arvostelut sen luomisajan mukaan eli tulee uusimmat arvostelun ensimmäisenä, `num_per_page` suurin sallittu arvo on sata (100), mutta tässä esimerkissä sen arvo on 20,

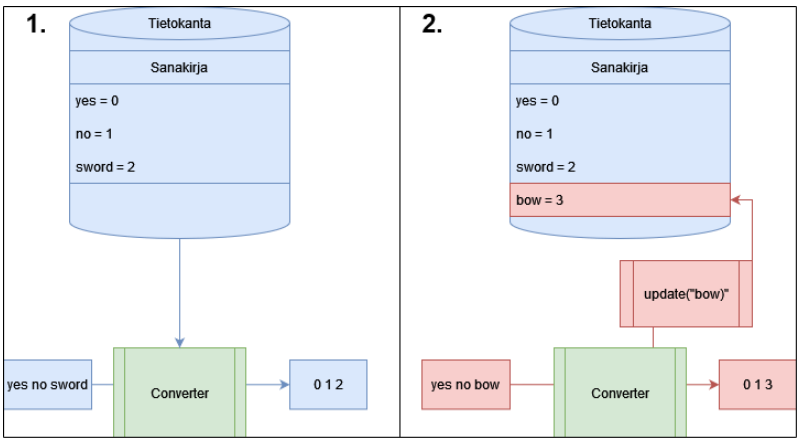
`language`-arvo on ”english”, joten saadaan kaikki arvostelut, jotka on merkattu englanninkielisiksi, `purchase_type`n, arvoksi on laitettu ”all”, joten tulee dataa muiltakin kuin vain Steam-kaupasta ostaneilta, `Cursor`-arvo on ”*”, koska tämä on ensimmäinen API-kutsu.

Toisessa koodinpätkässä kaivetaan ensimmäinen `reviews`-olio ja näytetään, mitä se sisältää. Melko keskellä tekstirivejä lukee, että `voted_up` on arvoltaan ”True”, joten tämä arvostelu ehdottaa peliä.

4 Datan esikäsittely

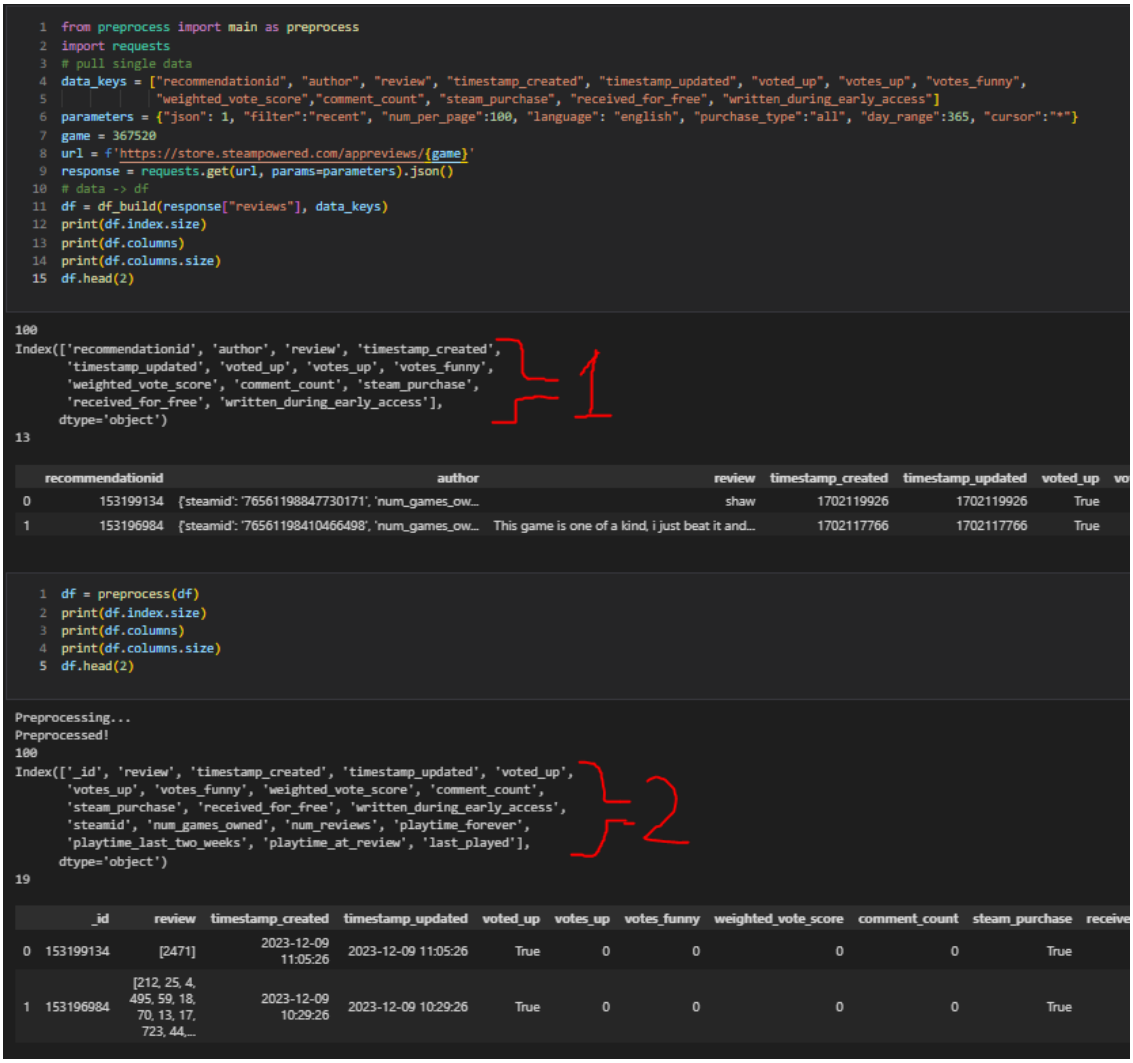
Ennen datan tallentamista tietokantaan se esikäsiteltiin poistamalla turha ja vääränlainen data. Huonoksi dataksi laskettiin sanat, joissa oli edes yksi kirjain, joka ei kuuluneet englannin kielen kirjaimiin. Esiprosessi muuttaa dataa neljällä eri tavalla: ajan muotoilu, kirjoittajaolion purkaminen, metatekstin puhdistaminen ja sanojen muuttaminen numeroiksi. Ajan muotoilussa datan `timestamp_updated` ja `timestamp_created` ajat muuttuvat Unix-aika (summa sekunteja) muodosta Pandas Series Datetime muotoon (esim. 2023-06-20 12:30:30), jotta MongoDB-tietokanta pystyy automaattisesti muuttamaan ne tietokannan käyttämään aikamuotoon. Tämä auttaa etsimään dataa tietyltä aikaväliltä tietokannasta.

Seuraavaksi, koska jokainen arvosteluolio sisältää `author`-olion, joka sisältää lisää dataa, josta osa on hyödyllistä, se pitää purkaa. Metatekstin poistamisessa poistetaan itse arvostelun sisällä olevat Steamin sisäiset tekstin muotoilutyökalut [15], poistaa linkit sekä tekstillä muodostetut kuvat. Tämän käsittelyn jälkeen ei kirjaimia enää poisteta, joten ne muutetaan kaikki pieniksi kirjaimiksi. Sanojen muuttaminen numeroiksi avustaa kahdella tavalla: tähän käytetty työkalu tallentaa jokaisen erikoisen sanan omaan keräykseen tietokannassa tällöin esikäsittelyn sekä laatua voidaan mitata ja virheille löytää esimerkkejä. Tämä sananmuuttajatyökalu on Python-luokka nimeltään ”Converter”. Converterin alustamisessa se hakee tietokannasta jo valmiiksi tallennetut sanat, kun tekstissä on tietokannalle uusi sana. Se tallennetaan tietokantaan omalla kokonaisluvullansa ja Converter päivittää tietonsa. Kuvassa 3 esitetään kaksi esimerkkitapausta. Ensimmäisessä kaikki sanat ovat tietokannalle tiedettyjä sanoja, joten Converter kääntää ne ilman, että tarvitsee ottaa tietokantapalveluun yhteyttä. Tapauksessa kaksi on uusi sana ”bow”, joten se lisää sen tietokantaan ja omaan sisäiseen kirjastoon, minkä jälkeen se kääntää tekstin ”bow” kokonaisluvuksi 3.



Kuva 3. Converter-työkalun idea

Kuvassa 4 on verrattavissa, mitä data sisältää ennen ja jälkeen esiprosessin. Punaisella on merkitty datasarakkeet: 1 on datakolumnit ennen esiprosessia ja 2 on datasarakkeet esiprosessin jälkeen.



Kuva 4. Esiprosessin esimerkkilopputulos

Kun on turhat datat poistettu ja ”author”-olioista on kerätty hyödylliset tiedot, datatyyppien määrä muuttuu 13 datatyypistä 19 datatyyppiin.

Esiprosessi on kielidatan analysoinnissa tärkein elementti, etenkin kun kyseessä on luonnollisesti muodostunut kielidata. Väärinkirjoitukset sekä erikoisesti kirjoitetut sanat esimerkiksi ”aktually” ja ”actually” lasketaan eri sanoiksi ja tämmöisien sanojen havaitseminen ja korjaaminen on jatkossa helpompaa sanakirjan avulla.

5 Datan tallennus

Työn tietokanta on Docker-kontissa pyörivä NoSQL-mallinen [16] MongoDB versio 6.0 [17]. MongoDB sisältää käyttäjäprofiilien luomisen. Näille profiileille pystyy antamaan erilaisia pääsylupia eri tietokantoihin. Näihin lupiin kuuluvat tietyn tietokannan lukulupa sekä tietyn tietokannan luku- ja muuttamislupa. Näitä profiileja on työssä luotu 4, joilla jokaiselle voi antaa omat käyttäjänimet sekä salasanat. Näiden profiilien ja lupien olemassaolo selventää jokaisen kontin tarkoitusta ja estää mahdollisessa jatkoskehityksessä: mikään kontti ei vahingossakaan koske dataan, johon sillä ei ole asiaa koskea. Nämä profiilit kuuluvat 3 Docker-kontille: esikäsittely, tekoäly- sekä järjestelmävalvoja-palvelimelle. Ja viimeinen profiili on tietokannan järjestelmävalvoja eli ”root”-käyttäjä. MongoDB-palvelimen sisälle muodostuu 3 oleellista tietokantaa: GameDatas, Dictionary ja Results. Palveliminen tietokantaluvat näkee taulukossa 4.

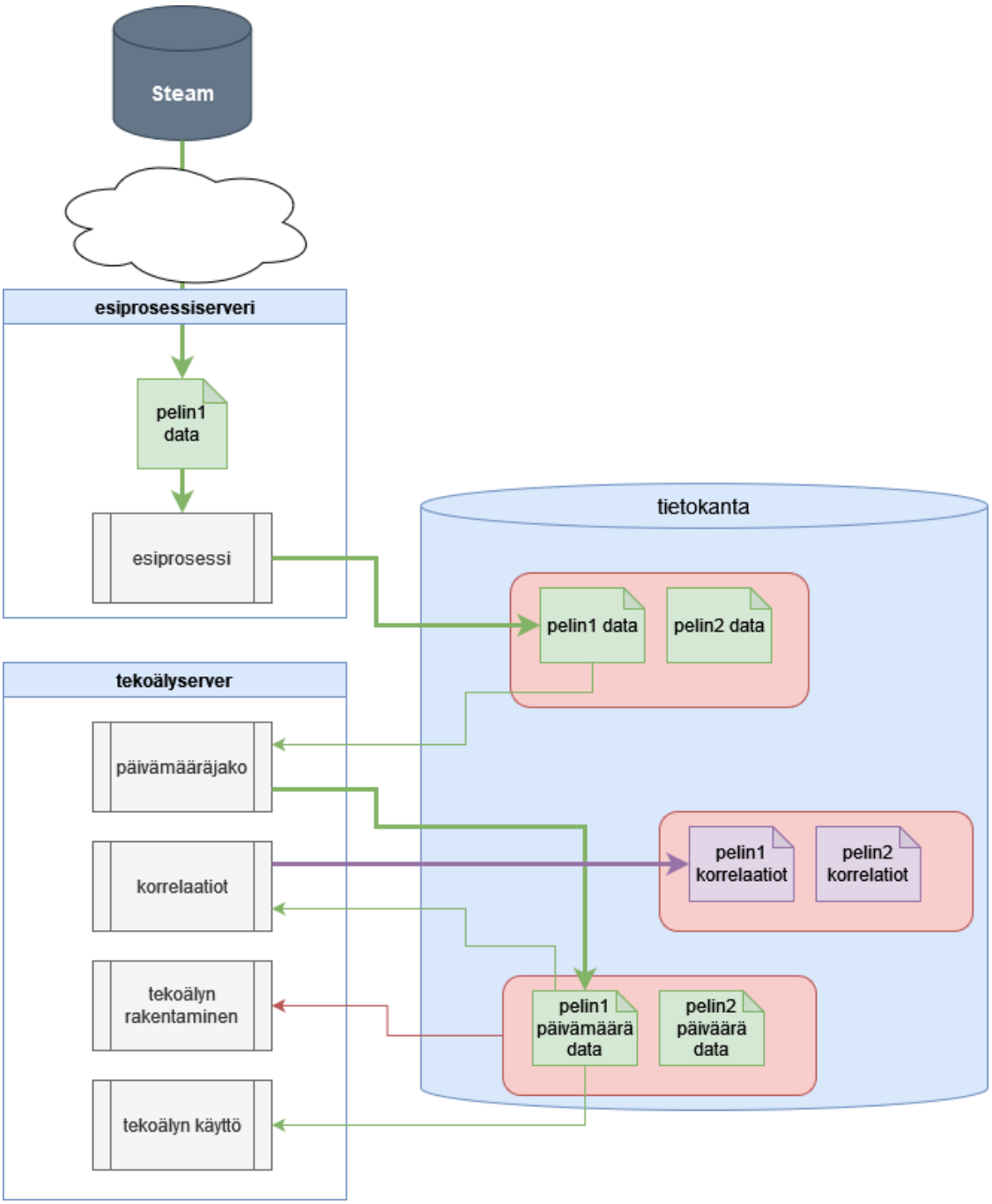
Taulukko 4. Tietokantaluvat palvelimille

	Esiprosessi	**Tekoäly**	**Järjestelmävalvoja**
`GameDatas`	Lukea ja kirjoittaa	Lukea	X
`Dictionary`	Lukea ja kirjoittaa	Lukea	X
`Results`	X	Lukea ja kirjoittaa	Lukea

GameDatas-tietokanta sisältää esiprosessipalvelimen esikäsittelyn läpi menneen datan. Ne on eroteltu pelin nimen mukaisiin kokoelmiin. Dictionary-tietokanta sisältää Converter-työkalussa käytetyn sanakirjan, jolla muutetaan sanat numeroiksi ja auttaa mittaamaan esiprosessin laatua. Results-tietokanta sisältää tekoälyserverin tuottamat tulokset. Näihin kuuluvat pelien päivämäärällisesti tiivistetyt datat sekä tästä muodostetut korrelaatiomatriisit. Nämä molemmat on laskettu pelikohtaisesti eli yksi matriisi sisältää vain yhden pelin arvosteludataa.

6 Datan jatkokäsittely

Tekoälypalvelimella on kolme erilaista työtä: arvosteludatan tiivistäminen, tekoälyn luominen ja tekoälyn käyttö. Kun arvosteludata on esiprosessin jälkeen tallennettu tietokantaan, sen voi tekoälyserveri ladata ja tiivistää sen ensimmäisenä päivämäärän mukaan. Tämän jälkeen joko muodostetaan sanakorrelaatio tai käytetään päivämääräistä dataa neuroverkon muodostamiseen sekä opettamiseen. Kuvassa 5 näkee datan elinkaaren.



Kuva 5. Arvosteludatan prosessointi

6.1 Päivämäärällinen jako

Ennen tekoälyn opettamista sekä korrelaatiotaulukoiden muodostamista, tarvitsee data jakaa päivämääriin. Tässä jakamisessa käytettään `timestamp_updated`-aikoja datan jakamiseen, sillä silloin saadaan uusimmat mielipiteet peleistä. Tällöin jos peli on muutettu paremmaksi tai huonommaksi, saadaan tuoreet mielipiteet. Data käsitellään Pandas-kehyksessä, jossa datan voi ryhmittää päivämäärän mukaan [18]. Ennen tätä ryhmittämistä muodostetaan kehykseen uusi datakolumni, jonka jokaiselle riville annetaan arvo 1. Tällöin saadaan arvostelujen määrä jokaisella päivällä, kun data ryhmitetään summaksi päivämäärän mukaan. Kuvassa 6 näytetään esimerkkidatan muodonmuutos vaiheittain.

```
time1 = datetime.datetime(year=2023,month=7,day=25,hour=10,minute=1,second=23)
time2 = datetime.datetime(year=2023,month=7,day=25,hour=14,minute=14,second=2)
time3 = datetime.datetime(year=2023,month=7,day=26,hour=10,minute=12,second=3)
df = pd.DataFrame({"timestamp_updated":[time1,time2,time3], "review":[[1,1],[2,2],[3,3,3]]})
print(df.dtypes)
df.head()
```

✓ 0.0s

```
timestamp_updated    datetime64[ns]
review                       object
dtype: object
```

	timestamp_updated	review
0	2023-07-25 10:01:23	[1, 1]
1	2023-07-25 14:14:02	[2, 2]
2	2023-07-26 10:12:03	[3, 3, 3]

```
df["timestamp_updated"] = df.timestamp_updated.dt.date
df["review_amount"] = 1
df.head()
```

✓ 0.0s

	timestamp_updated	review	review_amount
0	2023-07-25	[1, 1]	1
1	2023-07-25	[2, 2]	1
2	2023-07-26	[3, 3, 3]	1

```
df = unpack_review(df)
df
```

✓ 0.0s

	timestamp_updated	1	2	3	review_amount
1	2023-07-25	2	2	0	2
2	2023-07-26	0	0	3	1

Kuva 6. Päivämääräisen datan muodostamisesimerkki

Päivämäärällinen data tallennetaan Results-tietokantaan pelin nimen mukaan.

6.2 Korrelaatio

Tämän työn kontekstissa korrelaatio tarkoittaa, kuinka korkealla todennäköisyydellä jokin sana ilmenee toisen sanan kanssa. Esimerkkinä jos autopelillä yhteensä sata (100) arvostelua, niissä kaikissa mainitaan ”auto”, 25 arvostelussa mainitaan ”rata” ja 50 mainitaan ”nopeus”, mutta missään näissä arvosteluissa ei mainita sanaa ”rata”, tämän korrelaatiomatriisin näkee taulukossa 5.

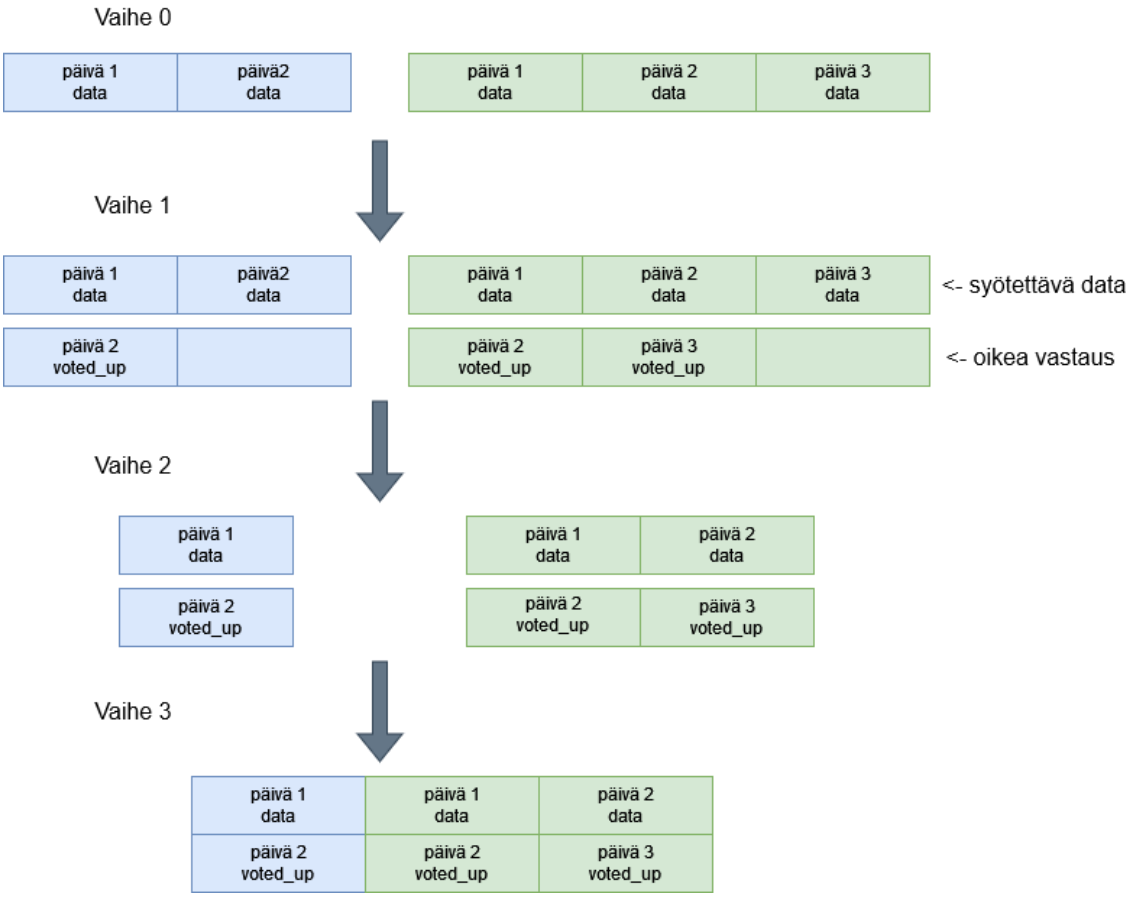
Taulukko 5. Korrelaatiomatriisi esimerkki

Sana	**auto**	**rata**	**nopeus**
auto	1,0	0,25	0,5
rata	0,25	1,0	-1,0
nopeus	0,5	-1,0	1,0

Arvostelujen sanojen korrelaatio voi antaa kuvaa, mitä asiakkaat ovat mieltä tietyistä pelin mekaanikoista tai hahmoista. Tämä toteutettiin käyttäen Pandas-kehysobjektin sisäistä ”corr”-funktiota [19] ”pearson”-metodilla. Tähän käytettiin kappaleen 6.1 päivämäärittäin jaettua dataa. Tämä korrelaatiodata tallentuu Results-tietokantaan. Pelikohtaisesti dataa voi tutkia selainpuolella antamalla pelin nimen, kuinka monta suurinta korrelaatiota otetaan sekä minkä sanojen korrelaatiot. Tästä enemmän kappaleessa 7 tuloksissa esimerkiksi kuva 13.

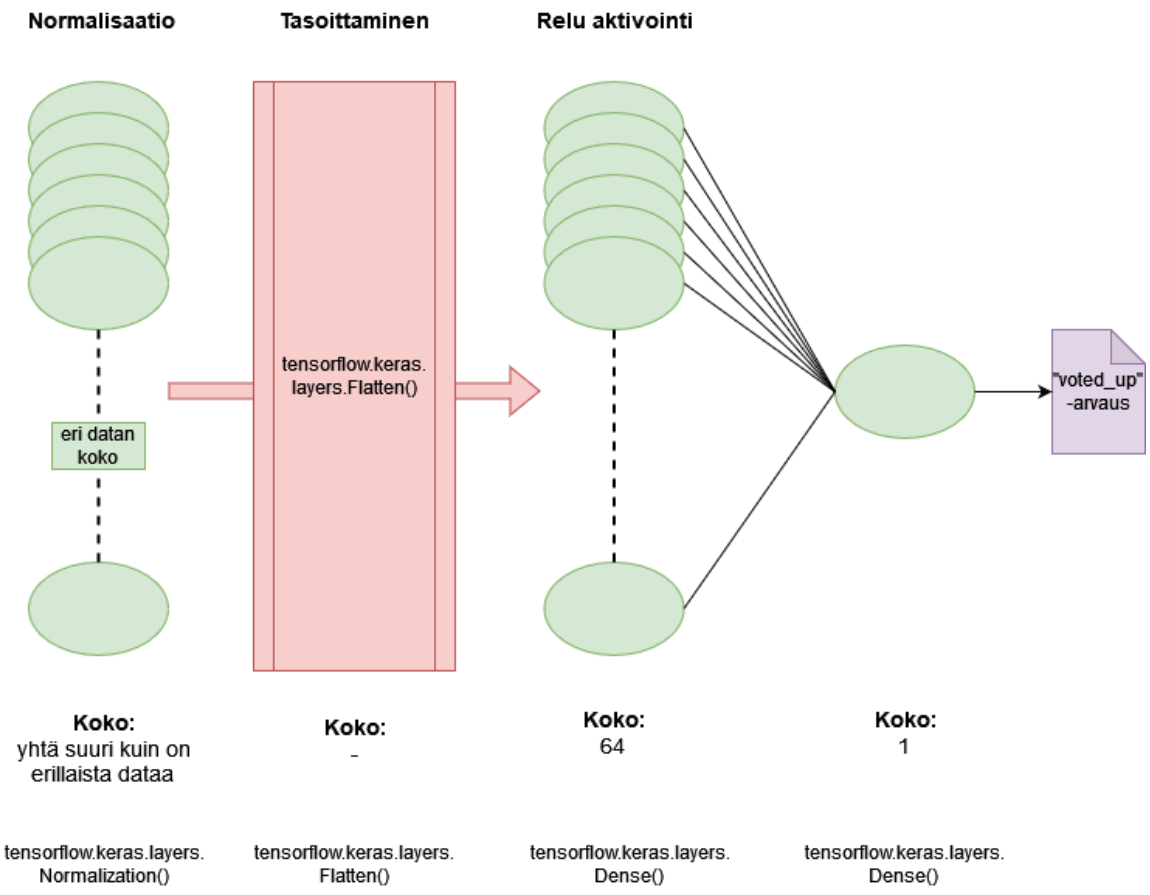
6.3 Neuroverkko

Neuroverkon rakentamiseen käytettiin kappaleen 6.1 päivämäärittäin jaettua data. Kun uusi neuroverkko rakennetaan tekoälyserverillä, se tulee yhdistää kaikki tietokannassa olevat päivämäärittäin jaetut datat ja muokata ne käytettävään muotoon. Kuvassa 7 on esimerkkinä kahdesta eri pelistä dataa. Vaiheessa 0 näkyy, että sinisestä pelistä on 2 päivän edestä dataa ja vihreästä on 3 päivän edestä dataa. Vaiheessa 1 otetaan seuraavan päivän tulokset talteen, että on oikeat vastaukset neuroverkolle. Vaiheessa 2 poistetaan viimeisen päivän data, koska niitä ei voi käyttää harjoittamiseen, sillä niille ei ole oikeita vastauksia. Vaiheessa 2 myös laskettiin, kuinka paljon: `voted_up`-, `votes_up`-, sekä `reviews_count`-arvot muuttuivat verrattuna edeltävään päivään. Vaiheessa 3 muodostetaan lopullinen harjoitusdata neuroverkkoon syötetystä datasta ja oikeista vastauksista.



Kuva 7. Opetusdatan muodostaminen

Tekoäly muodostamis- ja käyttämisesimerkeissä muodostetaan neuroverkko TensorFlow 2.13.0-version työkaluja käyttäen [20]. Kun uusi neuroverkko muodostetaan tekoälyserveri tallentaa muistiin eri sanojen määrä. Tämä toimii, koska sanakirjatietokannassa jokaisella sanalla on oma numeronsa ja kun uusi sana löytyy, se antaa uusimmalle sanalle suurimman numeron, kuten kuvassa 3. Tällöin kun käytetään neuroverkkoa ennustamaan uudella datalla, ohjelma pystyy poistamaan uusien sanojen datat. Työssä rakennettu neuroverkko muodostuu neljästä kerroksesta: normalisointikerros, jonka koko riippuu käytettävän datan sanarikkaudesta; tasoituskerros; ”dense”-kerros, jonka koko on 64 ja aktivointi funktiona toimii ”relu” ja viimeisenä kerroksena on toinen ”dense”-kerros, jonka koko on 1 ja syöttää ulos float-numeron eli desimaaleihin kykenevän datatyypin. Neuroverkon rakenne on visualisoitu kuvassa 8.



Kuva 8. Tekoälyn rakenne.

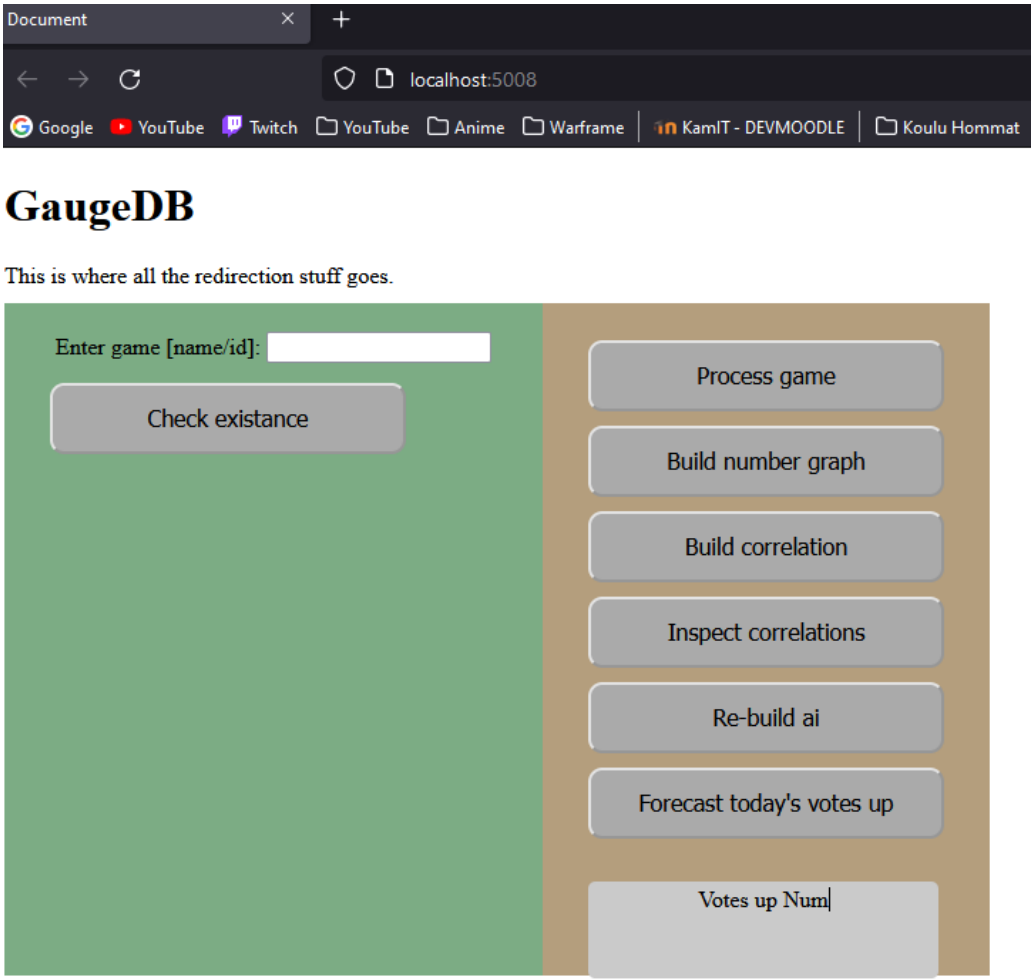
Kun neuroverkko on koulutettu, tallentuu tekoälyserverille kaksi asiaa: paikallisesti koulutuksessa ollut sanamäärä sekä itse neuroverkko. Tekoälyn rakentamisen testaamisessa käytettiin neljän pelin arvosteludataa: Hollow Knight [9], A Hat in Time [21], Slime Rancher [22], Warframe [23]. Nämä valittiin, että saataisiin mahdollisimman paljon erilaisia sanoja pienestä määrästä arvosteluista. Hollow Knight ja Warframe sisältävät paljon mekaanikkoja ja hahmoja, joilla on korkea mahdollisuus päätyä arvosteluihin. A Hat in Time ja Slime Rancher ovat paljon rennompia pelejä, joten niiden asiakaskunnat ovat monipuolisempia.

7 Tulokset

Tämä kappale sisältää, miltä työn lopputulokset näyttävät selainpuolella. Tässä työssä on käytetty selainpuolen rakentamiseen vain HTML-, CSS- ja JavaScript-standarditoimintoja ja funktioita. Näin voidaan toimia, koska selainpuolella ei tapahdu mitään datan muodostelua, vain sen esittelyä.

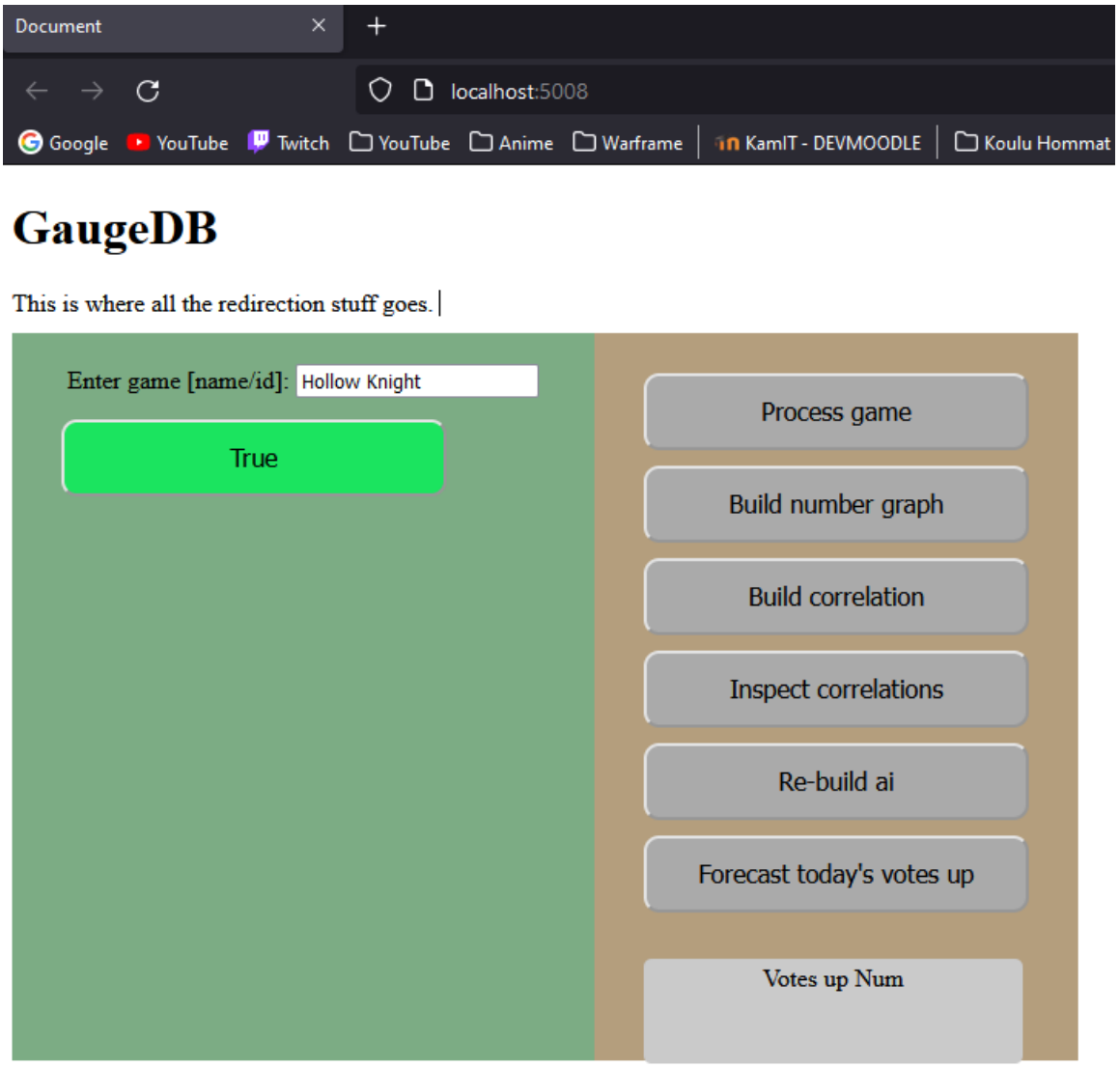
7.1 Sivun ulkonäkö

Demonstraation helpottamiseksi on luotu selainpuolinen sivu. Tämän sivuston kotisivulla on 6 paikallista toimintoa ja yksi uudelleenohjaava toimintopainiketta, jotka näkyvät kuvassa 9. Suurin osa näistä toiminnoista vaatii, että sivulla olevassa tekstikentässä on annettu Steam-kaupassa olevan pelin nimi.



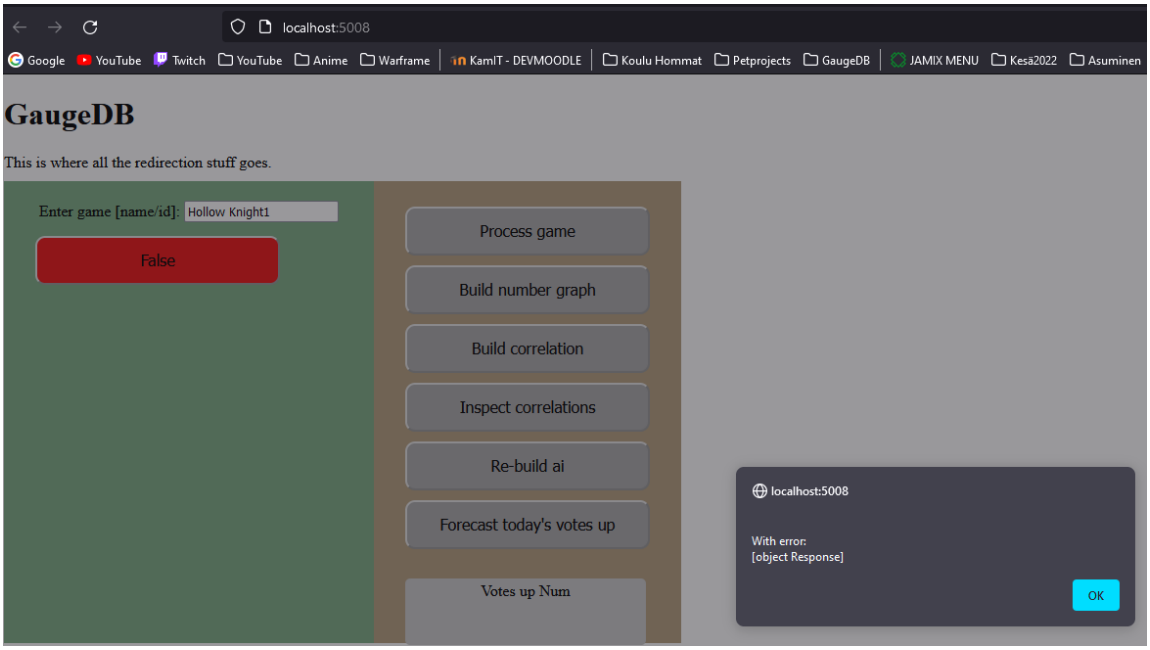
Kuva 9. Selainpuolen kotisivu

Kun tekstikenttään laittaa pelin nimen ja painaa painiketta, jossa lukee ”Check existance”, sivusto lähettää pyynnön valvontapalvelimelle, joka tarkistaa onko annettu nimi Steam-kaupan tuotelistalla [12]. Jos tuote on olemassa `Check existance` -painiketta muuttuu vihreäksi ja tekstiksi tulee ”True”, kuten kuvassa 10.



Kuva 10. Pelin nimi on oikeinkirjoitettu tarkistamisessa

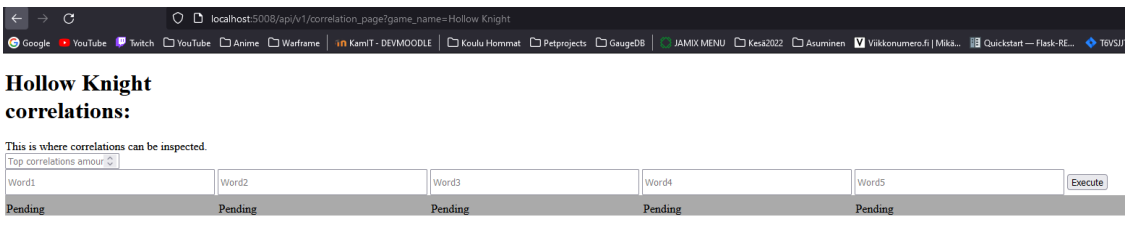
Jos peli ei ole olemassa tai se on vain väärinkirjoitettu, painike muuttuu punaiseksi, tekstiksi tulee ”False” ja selain heittää varoituksen. Tämän näkee kuvassa 11.



Kuva 11. Pelin nimi on kirjoitettu väärin tarkistuksessa

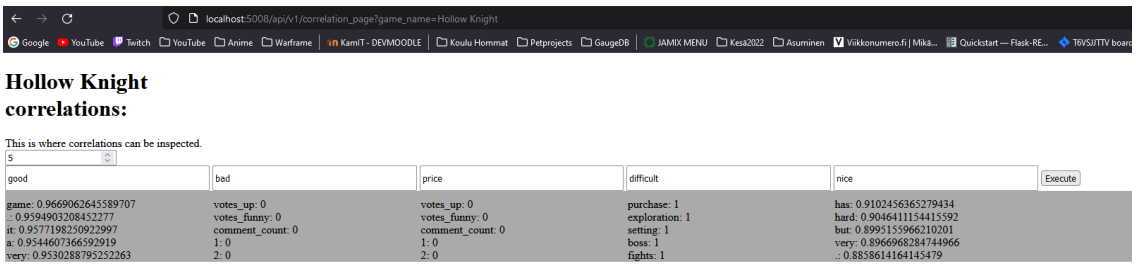
Jos painaa mitään muuta painiketta kuin `Re-build ai` ja tekstikentän peliä ei ole olemassa, sivusto heittää varoituksen. `Process game` -painike aloittaa pelin arvosteludatan keräämisen, esikäsittelyn ja tietokantaan tallentamisen. `Build number graph` -painamalla se jatkokäsittelee pelin arvosteludatan kappaleen 6.1 päivämäärälliseen muotoon.

`Build correlation` -painike käsittelee pelin päivämäärillisen datan korrelaatiomatriisiksi. Kun korrelaatiomatriisi on muodostettu, sitä voi tutkia, kun painaa sivuston ainoaa uudelleenohjaavaa painiketta, kun nimi on tekstikentässä. `Inspect correlation` -painike uudelleen ohjaa sivulle, missä on 6 tekstikenttää. Näistä alimpiin viiteen laitetaan jokaiseen sana, minkä korrelaatioista halutaan tietoa. Sivun ylimpään kenttään laitetaan numeraalinen arvo, joka on montako korrelaatiota ja joka esitetään kerralla jokaiselle sanalle. Tämän lisäksi sivulla on painike, jossa lukee `Execute`. Tätä painamalla sivusto lähettää valvontapalvelimelle pyynnön saada tietokannasta näiden sanojen tiedot. Kuvassa 12 on esimerkkinä, kun kotisivulla on tekstikentässä ”Hollow Knight” ja painaa `Inspect correlations` -painiketta.



Kuva 12. Korrelaatiosivuesimerkki

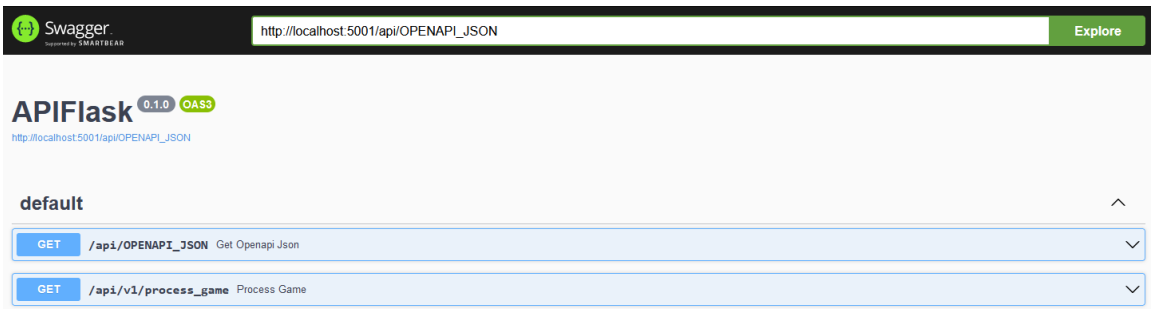
Kuvassa 13 on laitettu korrelaatiosanoiksi ”good”, ”bad”, ”price”, ”difficult” ja ”nice” ja sanamääräksi 5 sanaa.



Kuva 13. Korrelaatiosivun esimerkkikäytön tulos

`Re-build ai` -painike lähettää valvojaserverin kautta komennon tekoälyserverille uudelleen rakentamaan ja opettamaan tekoälyn. Tekoälyserveri yhdistää paikallisesti kaikki kappaleen 6.1 päivämäärillisen datan ja uudelleen rakentaa tekoälyn, koska saattaa olla uusia sanoja datassa. `Forecast today's votes_up` -painike lähettää, kun tekstikentässä on pelin nimi, valvontaserverin kautta komennon tekoälyserverille ottamaan tämänhetkisen tekoälyn ja syöttämään sille tekstikentässä olevan pelin päivämäärillisen datan eilisestä. Tällöin neuroverkko arvaa sen hetkisen päivän lopputuloksen, kuinka moni jättää sinä päivänä positiivisen arvostelun pelille. Tämä arvaus menee kotisivulla alimman napin alle, kuten kuvista 19–21 ilmeni.

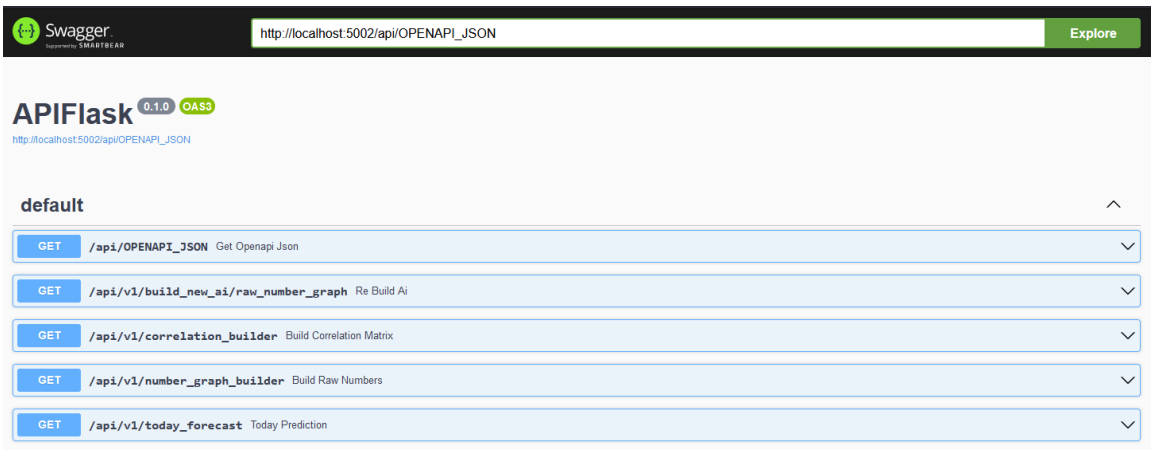
Docker konttien flask-palvelimien API-dokumentaatiota varten on luotu OpenApi JSON-tiedosto, jonka esittämiseen jokainen palvelin käyttää SwaggerUI-pohjaa. Jokaisessa on olemassa `/api/OPENAPI_JSON`, joka antaa OpeanApi JSON-tiedoston sekä `/api/docs`, jossa kuvat 14–18 oli otettu. Kuvassa 14 näkee esiprosessipalvelimen OpenApi-dokumentaation.



Kuva 14. Esiprosessipalvelimen OpenApi dokumentaatio SwaggerUI näkymällä

Esiprosessipalvelimen osoite `/api/v1/process_game` ottaa vastaan Steam-kaupan tuotenumeron tai pelin nimen ja lähtee prosessoimaan sen pelin Steam-arvosteluja. Kun kaikki arvostelut on prosessoitu eiliseen asti, palvelin lähettää signaalin, että peli on prosessoitu loppuun asti.

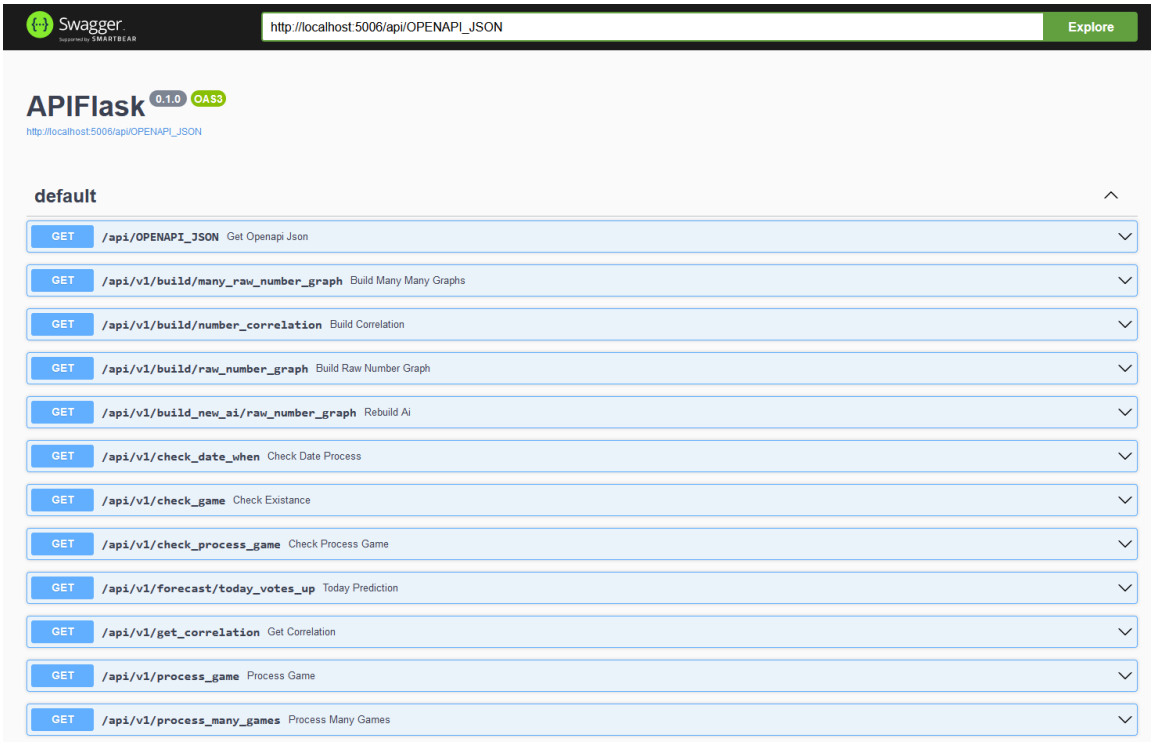
Esiprosessipalvelimen tuottaman datan jatkoprosessointi tapahtuu tekoälypalvelimella. Kuvassa 15 näkee tekoälypalvelimen eli jatkoprosessointipalvelimen OpenApi-dokumentaation.



Kuva 15. Tekoälypalvelimen OpenApi-dokumentaatio SwaggerUI-näkymällä

Tekoälypalvelin oli toinen, joka rakennettiin, joten sielläkin on pieni testijäännös. Tekoälypalvelimella on 4 toimintaa, joista kaikki paitsi neuroverkkorakentaja ottaa parametrina vain pelin nimen tai Steam-kaupan tuotenumeron. Kutsuessa `/api/v1/number_graph_builder` tekoälypalvelin tarkistaa ensin, onko annettu peli jo olemassa GameDatas-tietokannassa, jos ei niin se lähettää pyynnön esiprosessipalvelimelle pyynnön. Tämän jälkeen toteuttaa olemassa olevalle datalle kappaleen 6.1 päivämäärittäisen prosessoinnin ja tallentamisen. Kutsuessa `/api/v1/correlation_builder` tekoälypalvelin tarkistaa onko annetusta pelistä olemassa kappaleen 6.1 jälkeistä dataa Results-tietokannassa, jos ei se lähettää pyynnön itsellensä `/api/v1/number_graph_builder` -osoitteeseen. Tämän jälkeen suorittaa kappaleen 6.2 korrelaatiomatriisin luomisen prosessoinnin sekä tallentamisen. Kutsuessa `/api/v1/build_new_ai/raw_number_graph` tekoälypalvelin ottaa kaikki Results-tietokannassa olevat kappaleen 6.1 päivämäärittäisesti prosessoitujen pelien datat ja niitä käyttäen tekee kappaleen 6.3 mukaiset muutokset ja luo kuvan 8 mukaisen neuroverkon ja tallentaa sen tiedostoon. Kun tämä on onnistuneesti tehty neuroverkko, testataan ja sen testin tarkkuusprosentti palautetaan kutsun lähettäjälle. Kutsuessa `/api/v1/today_forecast` tekoälypalvelin tarkistaa, onko annetusta pelistä Results-tietokannassa eiliseen asti kappaleen 6.1 päivämäärittäisesti prosessoitua dataa, jos ei se lähettää pyynnön itsellensä

`/api/v1/number_graph_builder` -osoitteeseen. Tämän jälkeen se ottaa eilisen datan ja lataa viimeksi rakennetun neuroverkon ja syöttää siihen nämä eiliset datat. Tämän tulokset eli neuroverkon arvaus tämän päivän positiivisten arvostelujen kokonaismäärä annetulle pelille lähetetään kutsun lähettäjälle. Esiprosessi- ja tekoälypalvelimelle ei olisi tarkoitus käyttäjän itsestään tehdä kustuja, vaan vähintään tehdä ne valvontapalvelimen kautta, että ei mitään prosessoitaisi turhaan kahdesti. Kuvassa 16 on valvojapalvelimen OpenApi-dokumentaation.



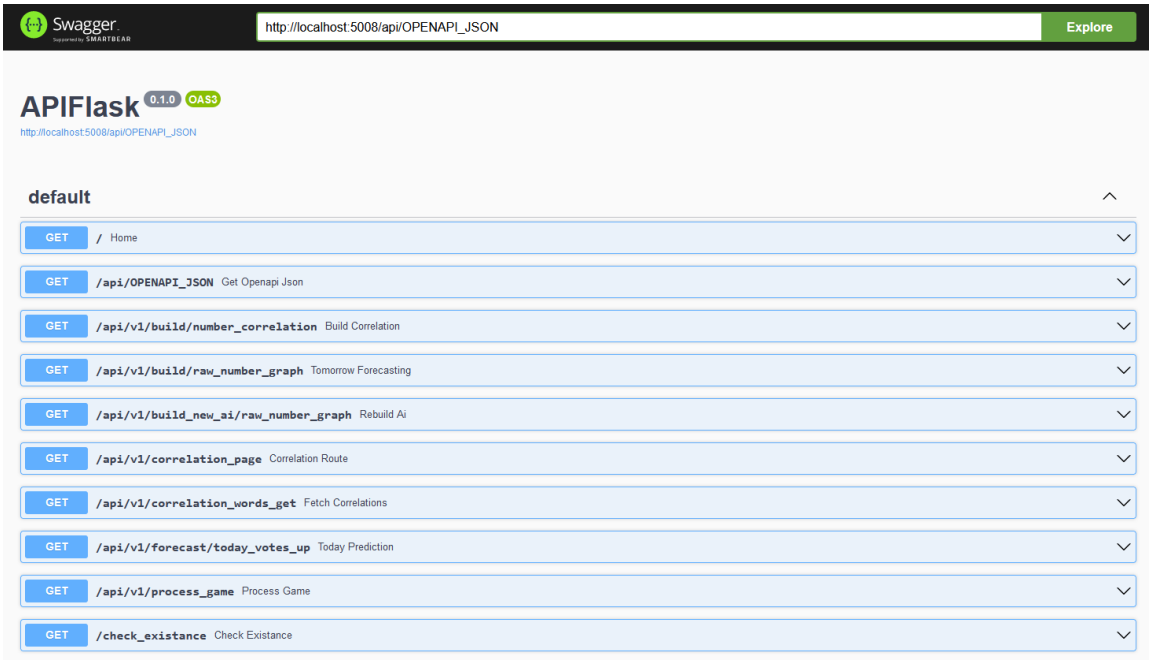
Kuva 16. Valvojapalvelimen OpenApi-dokumentaatio SwaggerUI-näkymällä

Valvojapalvelimen tärkein rooli on pitää kirjaa, mitä pelejä on jo prosessoitu ja milloin viimeksi. Jokainen prosessointikutsu, joka menee tämän palvelimen läpi, kirjataan ja tulevaisuudessa tarkistetaan, jos samaa peliä kutsutaan. Tämän kirjauksen avulla mitään peliä ei turhaan prosessoida kahdesti. Näihin tietoihin pääse käsiksi syöttämällä pelin nimen ja prosessointityylin kutsuun `/api/v1/check_date_when`, joka palauttaa suoraan päivämäärän, milloin kyseinen peli oli viimeksi annetulla tyylillä prosessoitu. `/api/v1/check_process_game` tekee saman, mutta palauttaa vain numeron, mikä kertoo, montako päivää viime prosessoinnista kulunut. `/api/v1/check_process_game`-kutsu tarkistaa, onko annettua peliä tai Steam-kaupan tuotenumeroa olemassa Steam-kaupassa. Kutsu `/api/v1/process_game` ottaa vastaan yhden Steam-kaupan pelin nimen tai tuotenumeron ja tarkistaa palvelimella, onko peliä jo prosessoitu ja jos on niin milloin viimeksi. Jos peliä ei ollut prosessoitu ikinä, niin se lähettää vain nimen esiprosessointipalvelimelle.

Jos peli oli prosessoitu, mutta ei samana päivänä, niin palvelin lähettää esiprosessipalvelimelle pelin nimen ja päivämäärän, milloin se oli viimeksi prosessoitu. Jos peli oli jo prosessoitu, niin palvelin vain palauttaa virheilmoituksen, että peli on jo prosessoitu. Useamman pelin prosessoinnin voi tehdä kutsulla `/api/v1/process_many_games`. Se ottaa vastaan listan Steam-kaupan pelien nimiä tai tuotenumeroita ja syöttää ne yksi kerrallaan valvojapalvelimen `/api/v1/process_game`-kutsuun. Kutsu `/api/v1/build/number_graph` toimii tekoälypalvelimen ja käyttäjäpuolen välikätenä ja ottaa vastaan pelin nimen tai tuotenumeron. Useamman pelin prosessoinnin voi tehdä kutsulla `/api/v1/build/many_raw_number_graph` syöttämällä listan pelien nimiä tai tuotenumeroita. Kutsuessa `/api/v1/build/number_correlation` järjestelmävalvojapalvelu tarkistaa, onko annettu pelistä sinä päivämääränä rakennettua korrelaatiomatriisia. Jos ei, niin se lähettää kutsun tekoälypalvelimelle, että se rakentaa sellaisen ja tallentaa sen Results tietokantaan.

`/api/v1/build/get_correlation` ottaa vastaan pelin ja 1–5 sanaa, joiden Results-tietokannasta löytyvien korrelaatiomatriisiarvot kerätään ja palautetaan kutsujalle JSON-paketissa. Kutsu `/api/v1/build_new_ai/raw_number_graph` toimii käyttäjän ja tekoälypalvelimen välikätenä. Eli kutsu vain laittaa järjestelmävalvojan lähettämään saman kutsun tekoälypalvelimelle, joka sitten rakentaa uuden neuroverkon kaikella kelvollisella datalla, mitä tietokannassa on. `/api/v1/forecast/todays_votes_up`-kutsu ottaa vastaan pelin ja lähettää pelin nimen ja saman kutsun tekoälypalvelimelle, jolloin se kerää pelin eilisen datan ja syöttää sen uusimpaan neuroverkkoon, joka pyrkii arvaamaan tämän päivän positiivisten arvostelujen summan ja palauttaa tämän summan kutsujalle.

Suurin osa asiakkaalle tarkoitetut työkalut ovat sivustopalvelimella, johon kuuluu kunnollinen nettisivu, jonka näkee esimerkiksi kuvissa 9–11. Tähän Sivustopalvelimella on myös OpenApi-dokumentointi, jonka näkee kuvassa 17.



Kuva 17. Sivustopalvelimen OpenApi-dokumentaatio SwaggerUI-näkymällä

Sivuston kotisivun eli kuvan 17 ylin GET tulokset näkee esimerkiksi kuvissa 9–11. Antaessa sivustossa pelin nimen tai Steam-kaupan tuotenumeron ja painamalla `Check Existance` -painiketta sivusto kutsuu `/check_existance`, joka toimii välikätenä valvontapalvelimelle, joten sivustopalvelin lähettää sinne nimellä tai tuotenumerolla varustetun kutsun `/api/v1/check_game`. Jolloin jos peli on olemassa sivuston `Check Existance` -painike, muuttuu vihreäksi tai punaiseksi, jos peliä ei ole olemassa.

Kun kotisivun tekstikentässä on peli ja painaa `Process game` -painiketta sivusto kutsuu pelin nimen kanssa sivustopalvelimen `/api/v1/process_game`-toiminnan, jolloin sama kutsu menee valvontapalvelimelle. Kun kotisivun tekstikentässä on peli ja painaa `Build number graph` -painiketta, sivusto kutsuu pelin nimen kanssa sivustopalvelimen `/api/v1/build/raw_number_grap`, jolloin sama kutsu menee valvontapalvelimelle. Kun kotisivun tekstikentässä on peli ja painaa `Build correlation` -painiketta, sivusto kutsuu pelin nimen kanssa sivustopalvelimen `/api/v1/build/number_correlation`, jolloin sama kutsu menee valvontapalvelimelle. Kun kotisivun tekstikentässä on peli ja painaa `Forecast today's votes up` -painiketta, sivusto kutsuu pelin nimen kanssa sivustopalvelimen `/api/v1/forecast/todays_votes_up`, jolloin sama kutsu menee valvontapalvelimelle. Kutsun tulos eli tekoälypalvelimella olevan neuroverkon veikkaus annetun pelin tämän päivän positiivisten arvostelujen summasta. Tämä summa sitten esitetään sivustolla kuten näkee kuvissa 19–22.

Kun `Re-build ai` -painiketta painetaan, sivusto kutsuu sivustopalvelimen `/api/v1/build_new_ai/raw_number_graph`, joka lähettää tekoälypalvelimeen kutsun rakentamaan uuden neuroverkon kaikella kappaleen 6.1 päivämäärällisesti jaettua dataa, mitä tietokannasta löytyy.

Kun kotisivun tekstikentässä on peli ja painaa `Inspect correlations` -painiketta, sivusto lähettää pelin nimellä varustetun kutsun sivustopalvelimen `/api/v1/correlation_page`-palveluun, joka antaa uuden sivuston, johon käyttäjä uudelleenohjataan. Uuden sivuston näkee kuvissa 12 ja 13. Siellä pystyy pelin arvosteluissa ilmestyneiden sanojen korrelaatioita tutkimaan 5 sanaa kerralla. Kun sanat on valittu ja annettu numero, kuinka monta riviä suurimpia arvoja halutaan nähdä jokaisesta sanasta. Voi painaa `Execute` -painiketta, jolloin sivusto lähettää sanat ja rivimäärän `/api/v1/correlation_words_get` kutsulla sivustopalvelimeen, jolloin se lähettää sanat kustulla `/api/v1/get_correlation` valvojapalvelimeen. Jolloin sivustopalvelin saa sanojen kaikki korrelaatiot, jolloin se ottaa halutun rivimäärän suurimpia arvoja ja palauttaa ne sivulle, joka esittää ne käyttäjälle kuten kuvassa 13.

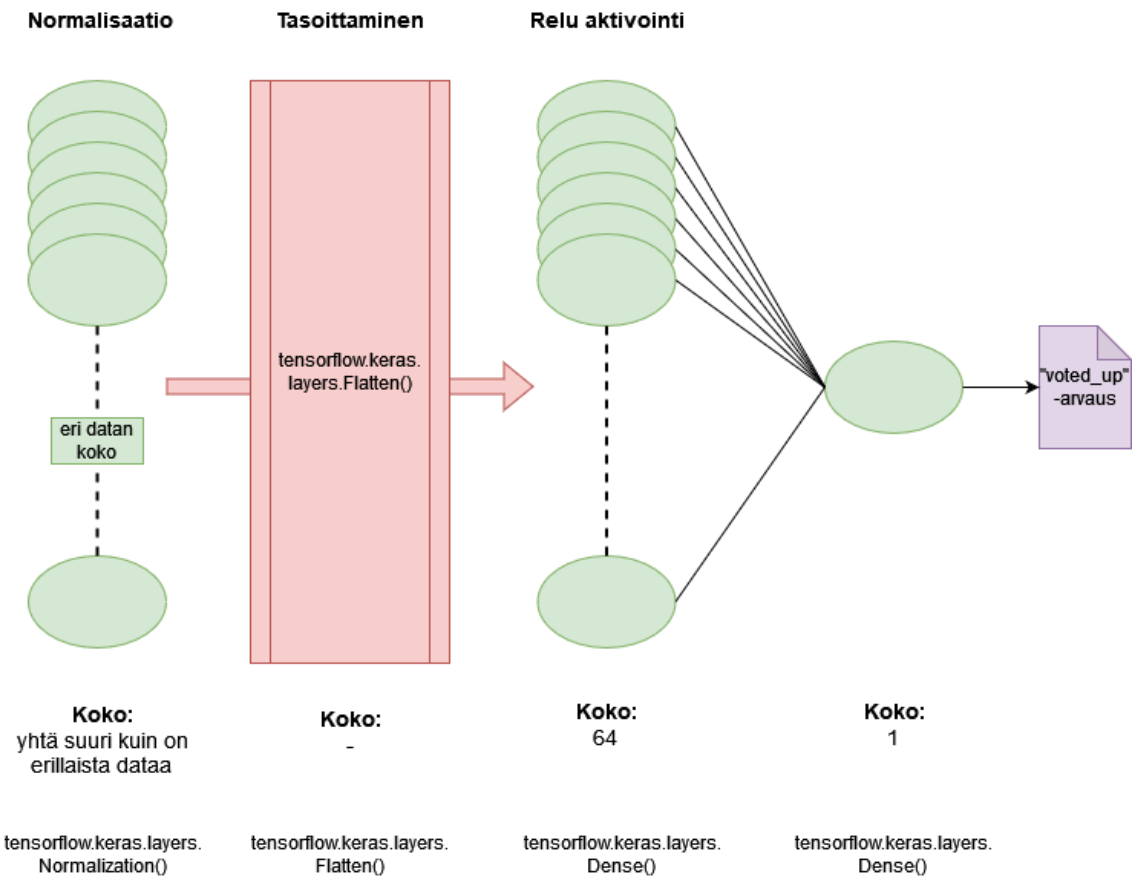
7.2 Datan kulku

Koko datan elinkaari testattiin neljällä pelillä: Hollow Knight, A Hat in Time, Slime Rancher ja Warframe. Esiprosessin jälkeen neljän pelien arvosteludatan levytilan koko oli 292 megatavua. Tämä jatkoprosessin jälkeen eli tekoälyn opettamiseen tarkoitetun datan (kappale 6.1) koko oli 97,1 megatavua. Testauksen aikana oli 2340 erilaista sanaa. Sanasarakkeen lisäksi otetaan mukaan 11 muuta arvoa, jotka ovat listattu taulukkoon 6.

Taulukko 6. Neuroverkon opetusdatojen selitykset

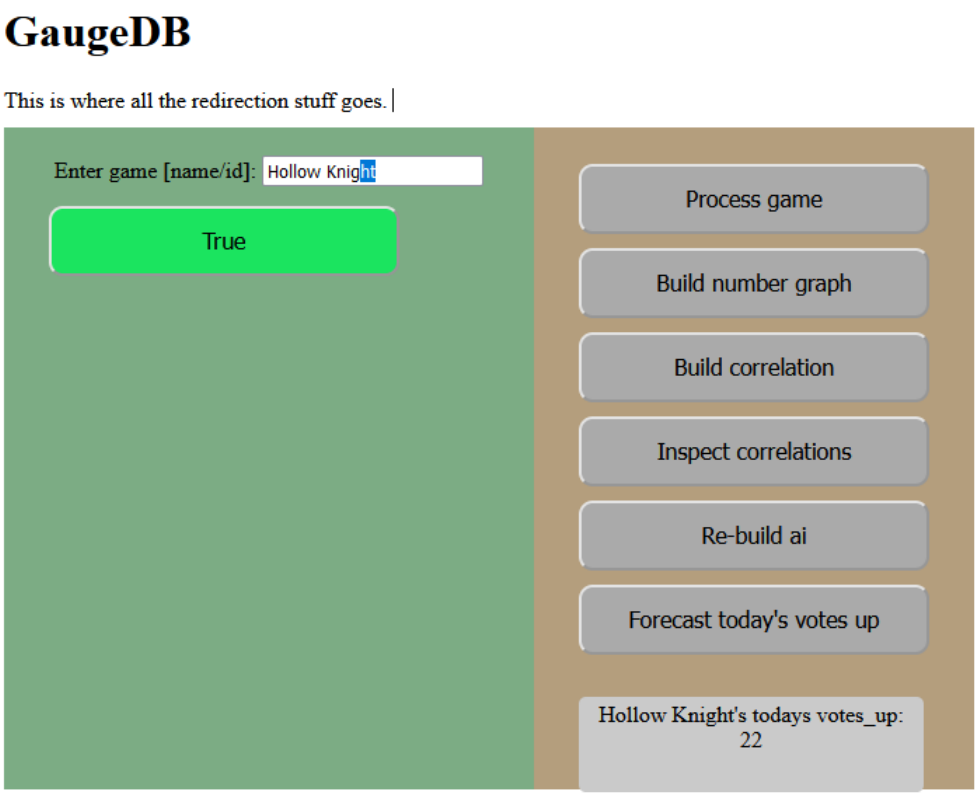
Sarakkeen nimi	Selitys
voted_up	Kuinka moni ehdotti peliä?
votes_up	Kuinka paljon päivän arvostelut ovat saaneet ääniä, että arvostelu oli hyödyllinen?
votes_funny	Päiväarvostelujen määrä siitä kuinka hauskoja arvostelut ovat.
comment_count	Kuinka paljon kommentteja päivän arvosteluissa on?
steam_purchase	Kuinka moni arvostelija hankki pelin Steam-storesta?
received_for_free	Kuinka moni arvostelija sai tuotteen ilmaiseksi?
written_during_early_access	Kuinka moni arvostelija kirjoitti arvostelun ennen tuotteen julkaisua?
reviews_count	Arvostelujen summa.
total_voted_up_percent	Kuinka paljon ”voted_up” muuttui viime päivään verrattuna?
total_votes_up_percent	Kuinka paljon ”votes_up” muuttui viime päivään verrattuna?
total_reviews_count_percent	Kuinka paljon ”reviews_count” muuttui viime päivään verrattuna?

Kun aikaisemmat sarakkeet ja sanamäärän laskee yhteen, on kuvan 14 ensimmäisen kerroksen koko 2351.

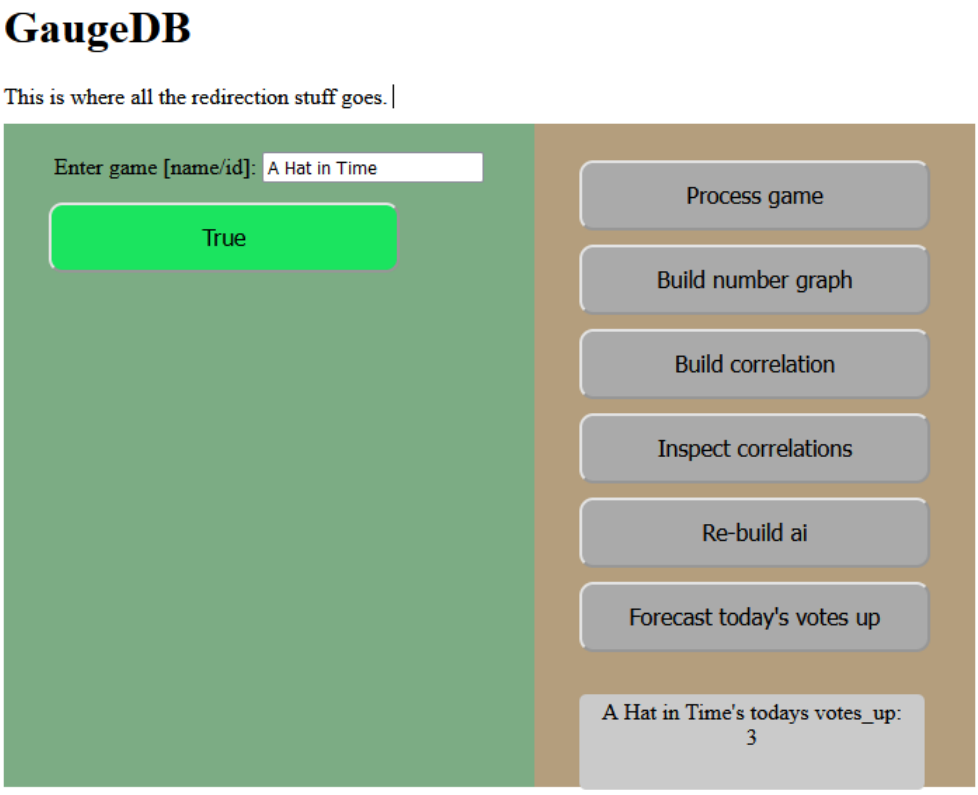


Kuva 18. Neuroverkon struktuuri

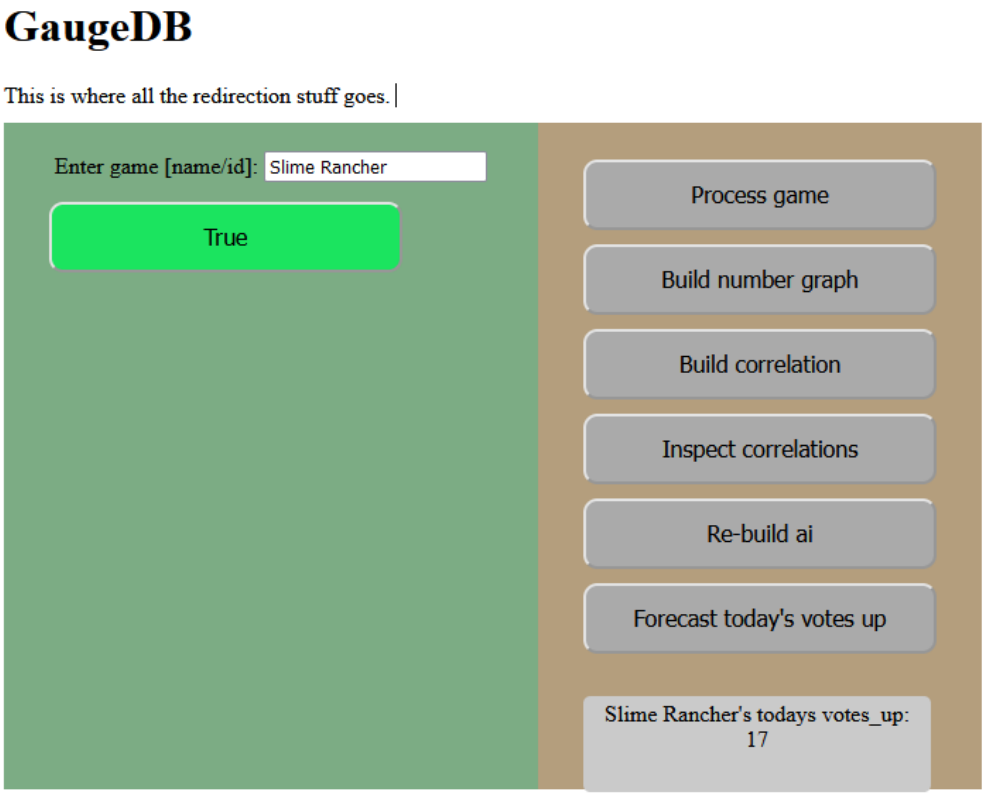
Kun neuroverkko koulutettiin 11. lokakuuta, opetusdata jaettiin kahtia, joista toista käytettiin itse neuroverkon opettamiseen ja toista osaa käytetään tekoälyn tarkkuuden laskentaan. Opettamiseen käytettiin 75 % datasta ja 25 % datasta tarkkuuden laskentaan. Tämän toteuttamiseen käytettiin ”sklearn”-Python kirjaston ”train_test_split”-työkalua [24]. Opetus nopeus oli 0.0001. Opetuskierroksi tai epookkeja oli 2. Neuroverkon tarkkuus oli noin 15 %. Kuvissa 19, 20, 21, 22 näkee, mitä neuroverkko arvasi olevan neljän testipelin positiivisten arvostelujen määrä 11. lokakuuta. Kuvien oton aikana sivulla oli kirjoitusvirhe. Neuroverkko arvaa ”voted_up”-arvoa, vaikka sivulla lukee ”votes_up”.



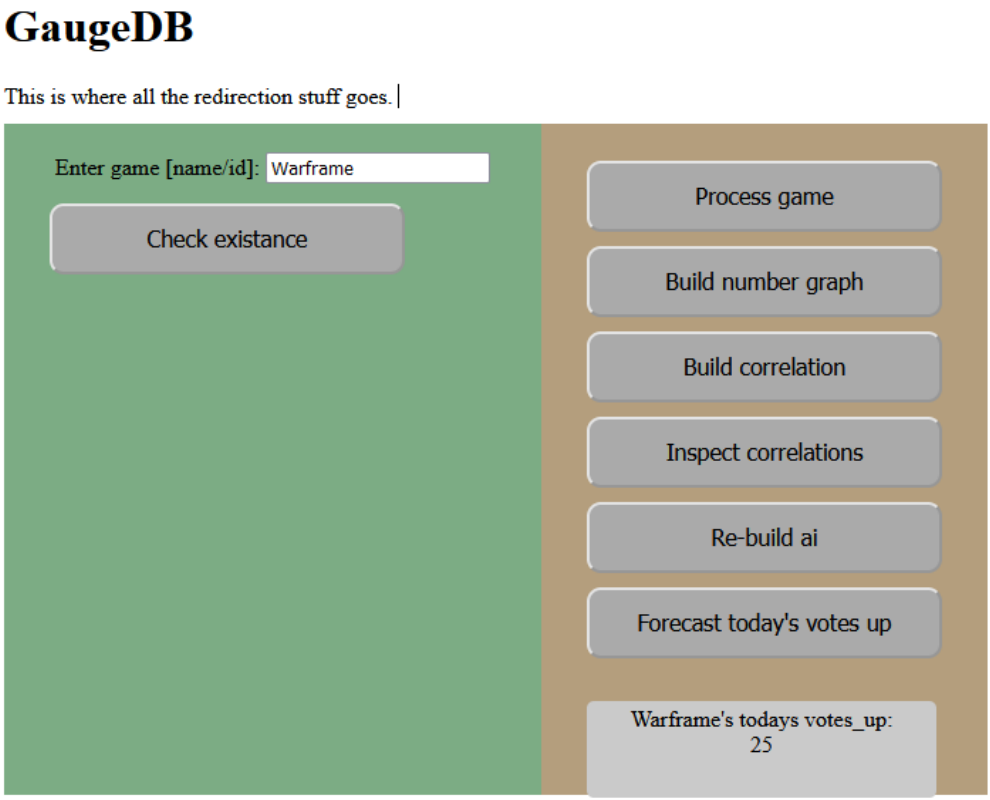
Kuva 19. Neuroverkon ”voted_up” -arvaus pelille Hollow Knight 11. lokakuuta



Kuva 20. Neuroverkon ”voted_up” -arvaus pelille A Hat in Time 11. lokakuuta



Kuva 21. Neuroverkon ”voted_up” -arvaus pelille Slime Rancher 11. lokakuuta



Kuva 22. Neuroverkon ”voted_up” -arvaus pelille Warframe 11. lokakuuta

7.3 Pilvipalvelin

Työ siirrettiin, pystytettiin ja testattiin CSC:n ePouta-pilvipalvelussa [25]. Ympäristön pohjaksi valittiin ”standard.3xlarge”, jossa mukana oli 8 virtuaaliprosessoria ja 62 gigatavua muistia. Avattiin myös ympäristöön portti, että pääsee valvontakonttiin käsiksi. Tällöin voitiin automatisoida useamman pelin prosessointi ilman, että prosessia tarvitsi keskeyttää. Luotiin ja yhdistettiin kaksi SSH-avainta: toinen, että pääse itse palveluympäristöön käsiksi toinen, että projektin pystyy lataamaan pilvipalvelimeen. Ympäristöön avattiin portti, että järjestelmävalvoja Docker-konttiin saa yhteyden. Tämän jälkeen pilvipalveluun asennettiin Docker.

Projektin lähdekoodiin kuuluu ”docker-compose.yml”-tiedosto, jolla voi Docker-konttien pystyttämistä helpottaa. Tämän tiedoston aktivointikomento `docker compose up` varoittaa puuttuvista arvoista, jos niitä ei ole luotu. Nämä arvot ovat tietokannan profiilien käyttäjät ja niiden salasanat. Nämä eivät tule projektin lähdekoodin mukana, joten ne annettiin komentoriville. Koska pilvipalveluun valittiin Ubuntu-ympäristö, nämä komennot yhdistettynä näyttivät komentorivillä tällaisille:

```
export USER1_NAME=esimerkki1; export USER1_PWD=esimerkki2;
```

Ensinmäinen sana ”export” kertoo Ubuntu-ympäristölle, että luotiin ympäristömuuttujaa. ”USER_NAME” on muuttujan nimi, jolle annetaan arvo ”esimerkki1”. Puolipiste (;) toimii Ubuntussa komentojakajana. Eli kun haluaa tehdä useamman komennon yhdellä tekstirivillä, se tehdään laittamalla komentojen väliin puolipisteen.

Kun arvot olivat annettu, pystyi työn pystyttämään komennolla `docker compose up`. Tämän jälkeen pystyi palvelua käyttämään normaalisti.

8 Mahdolliset jatkokehitystarpeet

Projektissa luotua sanakirjaston sisältöä selatessa huomattiin, että siellä oli paljon sanoja, jotka eivät olleet englanniksi, vaikka API-parametreissä oli kieleksi valittu englanti. Sekä monet sanat olivat suuria numeroita. Numerot nollasta kymmeneen (0–10) muutetaan esiprosessissa sanoiksi, kunhan ne eivät ole ”x/10”-formaatissa, mutta siitä eteenpäin se vain jättää ne. Esiprosessissa voisi myös olla oikeinkirjoitustarkistus, että pienet kirjoitusvirheet eivät loisi turhaan uuttaa sanaa sanakirjaan.

Pelin arvosteludatan ensimmäistä esiprosessia voisi optimoida. Monissa prosesseissa ei ole indikaatiota, milloin ne ovat valmiita. Neuroverkon suunnitteluun ja rakentamiseen ei ollut järkeä laittaa paljoa aikaa, koska esiprosessi ei tuottanut täysin puhdistettua dataa. Olisi turhaa yrittää tehdä tarkkaa neuroverkkoa, jos se data, jolla se oppii, on huonossa muodossa tai epätarkkaa. Kun esiprosessi on tarkempi ja nopeampi, voisi neuroverkkoon laittaa enemmän työtä. Kaiken tämän jälkeen voisi rakentaa johdannossa mainittua työkalua, joka lukisi pelien ostajien kiinnostuksia.

Selainpuolen ulkonäkö ja sen kehittämisessä olleet työkalut voisi vaihtaa modernimpaan esimerkiksi Streamlit, joka on minimaaliseen demonstraatioon tarkoitettu työkalu, joka toimii Python-pohjalla.

Työssä saatiin luotua systeemi, joka kerää arvosteludataa Steam-kaupasta, poistaa suuren osan huonosta datasta, jakaa datan päivämäärän mukaan tiivistettyihin riveihin, muodostaa sitä käyttäen neuroverkon ja korrelaatiomatriisin, tuo näiden tulokset selaimelle ja kaikki tämä pilvipalveluympäristöä käyttäen.

9 Lähteet

1 Hills J. Steam Review Explorer. [Datatutkintasivusto]. 2021. [viitattu 23.05.2024]. Saatavilla: https://project.joshhills.dev/steam-review-explorer/

2 Google. Google Trends. [Datatutkintasivusto]. 2006. [viitattu 23.05.2024]. Saatavilla: https://trends.google.com/trends/

3 Docker. Docker. [Tietokoneohjelmisto]. 2013. [viitattu 23.05.2024]. Saatavilla: https://www.docker.com/

4 Ronacher A. Flask. [Tietokoneohjelmisto]. 2010. [viitattu 23.05.2024]. Saatavilla: https://flask.palletsprojects.com

5 MongoDB. MongoDB. [Tietokoneohjelmisto]. 2009. [viitattu 23.05.2024]. Saatavilla: https://www.mongodb.com/

6 Mongo Express. [Tietokoneohjelmisto]. 2015. [viitattu 23.05.2024]. Saatavilla: https://hub.docker.com/_/mongo-express

7 PC GAMER. Wilde T ja Sayer M. The 19-year evolution of Steam. [Internet]. 2021. [viitattu 09.05.2024]. 13.11.2022. [viitattu 23.05.2024]. Saatavilla: https://www.pcgamer.com/steam-versions/

8 Elad B. 25+ Steam Statistics 2022 Users, Most Games, Market Share and Demographics. EnterpriseAppsToday. [Internet]. 05.03.2024. [viitattu 23.05.2024]. Saatavilla: https://www.enterpriseappstoday.com/stats/steam-statistics.html

9 Valve. Steam Reviews. [Tuote esittely]. 2013. [viitattu 23.05.2024]. Saatavilla: https://store.steampowered.com/reviews/

10 Valve. Steamworks. [Tuote esittely]. 2008. [viitattu 23.05.2024]. Saatavilla: https://partner.steamgames.com/

11 Valve. Steamworks API. [Ohjelmadokumentaatio]. 2008 [viitattu 23.05.2024]. Saatavilla: https://partner.steamgames.com/doc/webapi_overview

12 Valve. Steamworks ISteamApps Interface. [Tietokoneohjelmistodokumentaatio]. 2008. [viitattu 23.05.2024]. Saatavilla: https://partner.steamgames.com/doc/webapi/ISteamApps

13 Valve. Steamworks User Reviews. [Tietokoneohjelmistodokumentaatio]. 2013. [viitattu 23.05.2024]. Saatavilla: https://partner.steamgames.com/doc/store/getreviews

14 Team Cherry. Hollow Knight. [Videopeli]. 2017. [viitattu 23.05.2024] Saatavilla: https://store.steampowered.com/app/367520/Hollow_Knight/

15 Valve. Steam Text Formatting. [Ohjelmadokumentaatio]. [viitattu 23.05.2024]. Saatavilla: https://steamcommunity.com/comment/Recommendation/formattinghelp

16 TechTarget: Gillis A ja Botelho B. Defenition MongoDB. [Internet]. 2023. [viitattu 23.05.2024]. Saatavilla: https://www.techtarget.com/searchdatamanagement/definition/MongoDB

17 MongoDB. MongoDB (versio 6.0). [Tietokoneohjelmisto]. 2009 [viitattu 23.05.2024]. Saatavilla: https://www.mongodb.com/compatibility/docker

18 NumFOCUS. Pandas Groupby. [Tietokoneohjelmistodokumentaatio]. 2024 [viitattu 23.05.2024]. Saatavilla: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.groupby.html

19 NumFOCUS. Pandas Correlation. [Tietokoneohjelmistodokumentaatio]. 2024. [viitattu 23.05.2024]. Saatavilla: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html

20 Google Brain. TensorFlow. [Tietokoneohjelmisto] 2015 [viitattu 23.05.2024]. Saatavilla: https://www.tensorflow.org/

21 Gears for breakfast. A Hat in Time. [Videopeli] 2017. [viitattu 23.05.2024]. Saatavilla: https://store.steampowered.com/app/253230/A_Hat_in_Time/

22 Monomi Park. Slime rancher. [Videopeli]. 2017. [viitattu 23.05.2024]. Saatavilla: https://store.steampowered.com/app/433340/Slime_Rancher/

23 Digital Extremes. Warframe. [Videopeli]. 2013. [viitattu 23.05.2024]. Saatavilla: https://store.steampowered.com/app/230410/Warframe/

24 Scikit-learn. sklearn.model_selection.train_test_split. [Tietokoneohjelmistodokumentaatio] 2007. [viitattu 23.05.2024]. Saatavilla: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

25 CSC. Pouta pilvipalvelu. [viitattu 23.05.2024]. Saatavilla: https://research.csc.fi/-/cpouta

26 Vihreä käyrä. [Kuva]. 2016. [viitattu 23.05.2024]. Saatavilla: https://pixabay.com/vectors/arrow-business-financial-graph-1295953/

27 Suurennuslasi. [Kuva]. 2016. [viitattu 23.05.2024]. Saatavilla: https://pixabay.com/vectors/magnifying-glass-magnify-glass-1293096/

28 Ihmisjoukko. [Kuva]. 2016. [viitattu 23.05.2024]. Saatavilla: https://pixabay.com/vectors/crowd-demonstration-flag-man-march-1294991/