UHASSELT

KNOWLEDGE IN ACTION

Faculteit Bedrijfseconomische Wetenschappen

master handelsingenieur in de beleidsinformatica

Masterthesis

Next-app predictie met Gated Recurrent Units en applicatie embeddings

Toon Heleven

Scriptie ingediend tot het behalen van de graad van master handelsingenieur in de beleidsinformatica

PROMOTOR :

Prof. dr. Benoit DEPAIRE

BEGELEIDER :

Mevrouw Leen JOOKEN

2020
2021

UHASSELT

KNOWLEDGE IN ACTION

www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

UHASSELT

KNOWLEDGE IN ACTION

Faculteit Bedrijfseconomische Wetenschappen

master handelsingenieur in de beleidsinformatica

Masterthesis

Next-app predictie met Gated Recurrent Units en applicatie embeddings

Toon Heleven

Scriptie ingediend tot het behalen van de graad van master handelsingenieur in de beleidsinformatica

PROMOTOR :

Prof. dr. Benoit DEPAIRE

BEGELEIDER :

Mevrouw Leen JOOKEN

Deze masterproef werd geschreven tijdens de COVID-19 crisis in 2020-2021. Deze wereldwijde gezondheidscrisis heeft mogelijk een impact gehad op het schrijf- en verwerkingsproces, de onderzoekshandelingen en de onderzoeksresultaten die aan de basis liggen van dit werkstuk.

Noname manuscript No.
(will be inserted by the editor)

Next-app predictie met Gated Recurrent Units en applicatie embeddings

Toon Heleven

Ontvangen: date / Geaccepteerd: date

Abstract Binnen next-app predictie wordt er getracht om het applicatiegebruik op een smartphone te voorspellen op basis van impliciete en expliciete data van het toestel. Deze voorspelling wordt gebruikt om de reactiesnelheid en de bedienbaarheid van het toestel te verhogen. Onder impliciete data valt ook de geschiedenis van geopende applicaties. Voorgaand werk bestudeerde reeds de toepasbaarheid van Long Short-Term Memory Recurrent Neural Networks om de geschiedenis van geopende applicaties te exploiteren voor deze predictie. Het Long Short-Term Memory model bleek hiervoor een geschikte kandidaat. In deze paper worden er drie vernieuwingen voorgesteld voor dit model in de vorm van Gated Recurrent Neural Networks, Bidirectional Recurrent Neural Networks en applicatie embeddings. Deze vernieuwingen worden geëvalueerd op drie event logs van geopende applicaties, waarvoor het Recurrent Neural Network gelijktijdig predicties zal maken en zal corrigeren voor mogelijke predictiefouten. Uit de resultaten blijkt dat Gated Recurrent Units beter geschikt zijn voor next-app predictie in deze online machine learning context, applicatie embeddings zorgen voor een verdere stijging in predictieaccuraatheid en Bidirectional Recurrent Neural Networks bieden geen meeropbrengst in predictieaccuraatheid.

Keywords Event Data · Next-app Predictie · Supervised Learning · Bidirectional Recurrent Neural Networks · Long Short-Term Memory · Gated Recurrent Unit · Embeddings · Online Machine Learning

Toon Heleven
Singelbeekstraat 30
Tel.: +32495933691
E-mail: toon.heleven@student.uhasselt.be

1 Introductie

Smartphones met hun touchscreen interfaces en talrijke sensoren zijn uitgegroeid tot toestellen die veelzijdig zijn in gebruik. De interactie tussen mens en smartphone is dan ook een interessant onderwerp geworden voor wetenschappelijk onderzoek. Het aantal applicaties die een gebruiker zal installeren, telt gemiddeld 56 tot 90. Voor sommige gebruikers geldt zelfs dat ze op een gegeven moment tot wel 150 applicaties geïnstalleerd hebben [16, 18]. Naarmate het aantal geïnstalleerde applicaties toeneemt, wordt het moeilijker om de gewenste applicatie te vinden. Eén van de onderwerpen voor wetenschappelijk onderzoek is dan ook next-app predictie geworden.

De voorspelling van de applicaties die de gebruiker wenst te openen, zou voor verbeteringen in het ontwerp van het toestel kunnen zorgen. Ten eerste kan het systeemgeheugen toegewezen worden op basis van applicaties die met grote waarschijnlijkheid gebruikt zullen worden. Ten tweede kunnen applicaties die waarschijnlijk niet geopend worden, gesloten worden om het batterijverbruik te optimaliseren. Tot slot zit er ook nog iets in voor de gebruiker, voor hem zou het systeem een dynamische set van applicaties kunnen voorstellen die hij waarschijnlijk wil gebruiken. In de praktijk is de adoptie van zulke systemen ook in opmars. Zo beschikt versie 11 van de Android software voor Google Pixel telefoons reeds over zo een dergelijk systeem.

De gebruiksgeschiedenis van applicaties kan onder andere gebruikt worden voor next-app predictie. Zo werd er reeds onderzoek gevoerd naar het exploiteren van lange termijn afhankelijkheden in een geschiedenis van meer dan 100 geopende applicaties [16]. Hier werd gebruik gemaakt van een Recurrent Neural Network (RNN) model voor het exploiteren van de gebruiksgeschiedenis.

Dit RNN model maakt gebruik van Long Short-Term Memory (LSTM) units in combinatie met een lange gebruiksgeschiedenis van meer dan 100 geopende applicaties. Elk van deze applicaties wordt er one-hot voorgesteld. Alternatieven voor deze LSTM units en one-hot voorstellingen van applicaties, worden in de paper van Xu e.a. [16] echter niet geëvalueerd.

Het onderwerp van dit onderzoek bestaat erin om vernieuwingen toe te passen op dit RNN model met als doel de top-5 predictieaccuraatheid ervan te verhogen. In dit onderzoek wordt er in de eerste plaats een framework gepresenteerd voor next-app predictie op basis van een deep learning model met Gated Recurrent Units en applicatie embeddings. Naast dit framework zal er ook een antwoord geboden worden op de volgende onderzoeksvragen:

- Leidt het gebruik van een Gated Recurrent Unit (GRU) tot betere resultaten dan het gebruik van een Long Short-Term Memory in next-app predictie?
- Wat is het effect van een Bidirectional Gated Recurrent Unit (BiGRU) ten opzichte van een Unidirectional Gated Recurrent Unit op de performantie van het next-app predictiemodel?
- Wat is het effect van applicatie embeddings op de performantie van het BiGRU en GRU predictiemodel?

In de volgende sectie zal het gerelateerde werk besproken worden. Gerelateerd werk situeert zich voor deze paper binnen next-app predictie, next-app predictie met Recurrent Neural Networks, Bidirectional Recurrent Neural Networks en embeddings. Na het gerelateerde werk zal het GRU next-app predictieframework besproken worden. Deze sectie zal een overzicht geven van de constructie van de Event Connection Graph, de manier waarop de Event Connection Graph gebruikt wordt om embeddings te verkrijgen en de pre-training en online learning fases van het GRU predictiemodel. In de sectie methodologie, die daarop volgt, worden de datasets en methoden besproken die gebruikt worden om het GRU predictiemodel te testen. De methodologie wordt gevolgd door de bespreking van de resultaten, waarna de resultaten uitgeklaard en besproken worden in de discussie. Tot slot worden de bevindingen bondig vermeld in de conclusie.

2 Gerelateerd werk

2.1 Next-app predictie

De taak van next-app predictie wordt beschreven als een top-k predictieprobleem met als doel om k mobiele applicaties te voorspellen. De top-k predictie is slechts accuraat als de applicatie die de gebruiker wil openen ook onder de voorspelde set van k applicaties valt. Naarmate k toeneemt, zal de top-k accuraatheid verhogen, maar met dalende meeropbrengst[1]. De waardebepaling van k wordt vaak verantwoord door het aantal iconen dat past op een rij van de gebruikersinterface van een smartphone toestel. In de meeste onderzoeken zal k meestal vier of vijf applicaties bedragen [1, 2, 9, 14].

Deze top-k predictie wordt gebaseerd op data die vergaard worden op smartphone toestellen. Afhankelijk van de studie zal er gebruik gemaakt worden van verschillende soorten data. In het werk van Xu e.a. [16] en Huang e.a. [8] observeren ze dat applicatiegebruik een sterke relatie heeft met tijd van de dag en locatie. In voorgaande studies werd dan ook gebruik gemaakt van de tijd van de dag en locatie voor de next-app predictie. Daarnaast werd er reeds gebruik gemaakt van data van het accu, de versnellingsmeter, Wi-Fi verbindingen, microfoon, Bluetooth en de status van de laadpoort [1].

2.2 Sequentiële relatie tussen applicaties

Naast deze features wordt er ook wel eens gebruik gemaakt van de geschiedenis van geopende applicaties. Hier wordt verondersteld dat er een correlatie bestaat tussen twee of meer sequentieel geopende applicaties. Als een gebruiker bijvoorbeeld eerst reviews bekeken heeft in de Yelp app, dan is de kans groter dat deze gevolgd zal worden door Google Maps[7]. Om die reden werden Hidden Markov Modellen (HMM) ook al gebruikt voor next-app predictie, waarbij de vorige geopende applicatie gebruikt wordt in de predictie van de volgende geopende applicatie [7, 9].

Naast de Markov modellen, bestaan er echter ook andere manieren om deze sequentiële relatie tussen applicaties te exploiteren. Hieronder vallen ook de Recurrent Neural Networks (RNN). Recurrent Neural Networks zijn Artificial Neural Networks (ANN) die ook gebruikt worden voor het exploiteren van sequentiële data [15, 17].

2.3 Recurrent Neural Networks

RNN exploiteren, net zoals de HMM, de correlatie tussen sequentiële datapunten. RNN worden vaak gebruikt binnen Natural Language Processing (NLP) om modellen op te stellen die predicties maken op basis van tekstuele data. Zogenaamde Vanilla RNN's lijden echter aan het probleem van vanishing en exploding gradients, waarbij de grootte van de afgeleide tijdens back-

propagation te groot of te klein wordt om betekenisvolle langetermijnrelaties aan te leren [12, 5, 4]. Om aan de exploding en vanishing gradient van RNN tegemoet te komen, introduceerde Hochreiter en Schmidhuber [5] in 1997 het Long Short-Term Memory (LSTM) model.

In het werk van Xu e.a. [16] maken ze gebruik van deze LSTM units, die Hochreiter en Schmidhuber [5] introduceerden, voor next-app predictie. De geschiedenis van geopende applicaties combineren ze met informatie over tijd en locatie van de smartphone. Dit predictiemodel bereikt een top-5 accuraatheid van 80% tot 90% op de geteste datasets. Deze datasets bevatten echter zeer weinig unieke applicaties. Zo tellen ze gemiddeld maar 15 tot 40 unieke applicaties in hun datasets.

Een LSTM model is echter niet het enige RNN model dat het probleem van exploding en vanishing gradients weet aan te pakken. In 2013 introduceerde Cho e.a. [3] het concept van de Gated Recurrent Unit (GRU) die een lagere complexiteit heeft dan de LSTM unit. Deze verlaagde complexiteit zorgt er ook voor dat deze tot wel 30% sneller traint [17]. De predictie capaciteit van GRU modellen is vergelijkbaar aan of beter dan de performantie van LSTM modellen, maar is sterk afhankelijk van de toepassingscontext en het aantal beschikbare trainingssamples [4, 17].

2.4 Bidirectional Recurrent Neural Networks

Naast de introductie van LSTM modellen in 1997, publiceerde Schuster en Paliwal [13] in datzelfde jaar een paper waarin ze het concept van Bidirectional Recurrent Neural Networks (BRNN) introduceerden. Dit Bidirectional model zal trainen op een inputsequentie in zowel de chronologische volgorde als in de omgekeerde chronologische volgorde. Omdat dit zou zorgen voor een verdubbeling van het aantal RNN units, hebben ze het aantal modelparameters in het werk van Schuster en Paliwal [13] opzettelijk constant gehouden in hun experimenten. In de resultaten van hun experimenten toonden de onderzoekers aan dat dit voor betere resultaten zorgt voor zowel classificatie als regressie taken.

2.5 Embeddings

Een embedding is een vectorrepresentatie van een categorische variabele. Embeddings worden binnen het vakgebied van Natural Language Processing (NLP) gebruikt om aangeleerde representaties van woorden te verkrijgen [11, 10]. Deze representatie vervangt de one-hot representatie die bestaat uit een vector van nullen en een één. Aangezien deze one-hot representatie een positie in de vector vereist voor elk mogelijk woord, groeit de representatie naarmate het vocabularium groeit. Embeddings zorgen er daarentegen voor dat woorden een continue numerieke representatie krijgen in de vorm van een vector van slechts 200 tot 300 waarden.

De aangeleerde vector representaties van woorden, zoals ze in het werk van Mikolov e.a. [11] gepubliceerd werden, worden context-onafhankelijke woord embeddings genoemd. Zij krijgen de naam context-onafhankelijk omdat de representatie niet verandert voor homoniemen, die meer dan één betekenis kunnen dragen. Ondanks het feit dat het woord bank kan verwijzen naar een zitplaats of naar een financiële instelling, zal dit woord altijd voorgesteld worden door dezelfde numerieke waarden.

Het verkrijgen van deze context-onafhankelijke embeddings daarentegen hangt sterk af van de context van een woord. In het werk van Mikolov e.a. [11] stellen ze twee verschillende Feedforward Neural Networks voor die beide in staat zijn om embeddings aan te leren. Eén van die modellen is het Continuous Skip-Gram model. Het idee achter dit model is dat de betekenis van een woord afgeleid kan worden uit de context waarin het gebruikt wordt. Dit impliceert dat woorden die in een gelijkaardige context gebruikt worden, ook een gelijkaardige betekenis dragen.

Voor een corpus van geschreven documenten heeft het Continuous Skip-Gram model het doel om de contextwoorden rond een middelste woord te voorspellen. Deze context bestaat uit twee delen: de history en de future van een woord. De history bestaat uit de woorden die voorafgaan op het middelste woord en de future zijn de woorden die volgen op het middelste woord. Naarmate het model de distributie van contextwoorden aanleert voor elk woord in het beschikbare vocabularium, zullen de embedded representaties van woorden die in een gelijkaardige context voorkomen, convergeren.

Ondanks de relatief lage dimensionaliteit van woord embeddings, zijn zij in staat rijke syntactische en semantische informatie vast te leggen. In de eerste plaats krijgen woorden met een gelijkaardige betekenis ook een gelijkaardige numerieke vector representatie. De betekenis van deze woorden worden zo goed gecapteerd in de vectoren dat zelfs algebraïsche berekeningen met deze vectoren steek houden. De berekening van vector(”Koning”) - vector(”Man”) + vector(”Vrouw”) resulteert in een vector die het dichtst aansluit op de vector representatie van Koningin [10].

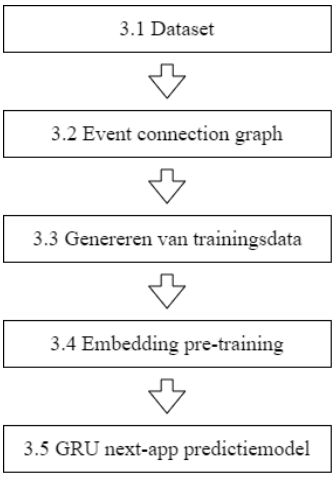
Aangeleerde embeddings werden ook succesvol toegepast buiten het vakgebied van NLP. In het werk van Hong e.a. [6] wordt de Event2Vec aanpak voorgesteld, waarbij tijdsgebonden events een vector representatie krijgen op basis van de sequentiële data waar ze in voorkomen. Door gebruik te maken van een Event Con-

nection Graph kan een Skip-Gram model er de relatie tussen events in de tijd aanleren. Daarnaast zorgt het gebruik van een Event Connection Graph ervoor dat er ongelimiteerd data gesampled kan worden voor het trainen van het Skip-Gram model.

Het gebruik van een Event Connection Graph met oog op next-app predictie is op zich geen nieuw idee. In het werk van Liao e.a. [9] werd de App Usage Graph (AUG) voorgesteld die de transitie probabiliteit tussen applicaties bevatten. In het werk van Zhou, Li en Liu [19] en Chen e.a. [2] worden er zelfs gelijkaardige app usage graphs gebruikt om embeddings aan te leren. Deze embeddings werden echter tot op heden nog niet toegepast op next-app predictie met RNN.

3 Framework

In deze paper wordt het GRU next-app predictiemodel voorgesteld. Deze sectie van de paper beschrijft de stappen binnen het GRU next-app predictieframework. Binnen dit framework wordt gebruik gemaakt van transfer learning, waarbij de kennis van een pre-trained algoritme gebruikt zal worden in het uiteindelijke GRU predictiemodel. Deze opgedane kennis zit vervat in applicatie embeddings die verkregen worden door het trainen van een Multi Layer Perceptron (MLP) model. Figuur 1 geeft een overzicht van de stappen binnen het GRU next-app predictieframework.



Figuur 1 Overzicht van de stappen in het GRU next-app predictieframework.

In sectie sectie 3.1 van het framework wordt de vereiste vorm beschreven van de dataset of event log met applicatiegebruik. Deze dataset zal dan gebruikt worden om een event connection graph te construeren die informatie bevat over de temporele relatie tussen applicaties. De constructie van deze event connection graph wordt beschreven in sectie 3.2. Deze event connection graph wordt dan gebruikt om trainingsdata te genereren voor een MLP model. De manier waarop data gegenereerd wordt uit deze event connection graph wordt beschreven in sectie 3.3. De manier waarop het MLP model applicatie embeddings aanleert op basis van deze gegenereerde data wordt beschreven in sectie 3.4. De architectuur van het uiteindelijke GRU predictiemodel en het gebruik van de applicatie embeddings in dit model wordt beschreven in sectie 3.5.

3.1 Vereiste vorm van de dataset met applicatiegebruik

Voor het opstellen van de event connection graph is in de eerste plaats een korte geschiedenis van 2000 geopende applicaties vereist. Deze geschiedenis wordt opgeslagen in een event log S, waar een event bestaat uit een geopende applicatie en een bijbehorend datum en tijdstip. De vereiste structuur van zo een event log wordt weergegeven in tabel 1.

Tabel 1 Structuur van de event log

Index	Applicatie	Tijdstip
1	Google Play Store	17:55:23
2	Evernote	17:57:10
3	Uber Eats	18:01:10

3.2 Constructie event connection graph

Op basis van deze event log zal de sequentiële relatie tussen applicaties vastgelegd worden in een event connection graph. De constructie van de event connection graph is analoog aan het werk van Hong e.a. [6]. In deze graph G = <V,E> zal elke applicatie voorgesteld worden door vertex V en de relatie tussen applicaties als de edges E tussen de vertices. Het gewicht van de directed edge E tussen vertices i en j geeft de sterkte van de sequentiële relatie tussen applicatie i en j aan. Dit gewicht wordt berekend volgens vergelijking 1.

$$G_{ij} = \sum_{1 \leq i < j < N} 1\{S(t_1) = e_i\} \wedge 1\{S(t_2) = e_j\} \delta(t_2 - t_1) \tag{1}$$

Bij elke applicatie e in event log S, hoort een tijdstip t. Deze vergelijking stelt dus dat δ toegevoegd wordt

aan G_{ij} telkens applicatie i gevolgd wordt door applicatie j. Als applicatie j dus op tijdstip t_2 plaatsvindt en applicatie i op tijdstip t_1, dan wordt δ berekend en toegevoegd aan G_{ij}. In vergelijking 2 wordt getoond hoe δ berekend wordt.

$$\delta(x) = \begin{cases} \exp(-x/\Delta), & \text{als } 0 \leq x < T \\ 0, & \text{anders} \end{cases} \quad (2)$$

δ is een dalende functie die voor een invoerwaarde gelijk aan nul, een functiewaarde van één produceert. Naarmate de invoerwaarde T benadert, daalt de functiewaarde van δ tot nul. De invoerwaarde van deze functie is het tijdsverschil in seconden tussen het openen van applicatie i en applicatie j. Naarmate het tijdsverschil tussen die twee geopende applicaties dan toeneemt, zal de waarde van δ dalen. T en Δ zijn twee hyperparameters van functie δ. T bepaalt het maximaal toegestane tijdsverschil in seconden tussen twee applicaties die nog als gerelateerd beschouwd worden. Als x de waarde van T overschrijdt, dan zal er niets toegevoegd worden aan G_{ij}. Δ bepaalt hoe snel de functiewaarde van δ afneemt naarmate het tijdsverschil tussen applicatie i en applicatie j groter wordt. Dat zal ervoor zorgen dat G_{ij} een grotere waarde krijgt als applicatie i en applicatie j elkaar vaker en met een kortere tussentijd opvolgen in de event log.

Uit de resultaten van dit onderzoek blijkt dat de top-5 predictieaccuraatheid van het GRU model niet erg gevoelig is aan de instelling van deze hyperparameters. Voor een kleine verbetering in predictieaccuraatheid wordt T best ingesteld op 3600 seconden en Δ op 900.

3.3 Genereren van trainingsdata

De trainingsdata voor het MLP model wordt gesampled uit de event connection graph. Eerst wordt er een willekeurige applicatie gekozen op basis van een uniforme kansverdeling. Deze applicatie zal de input vormen voor het MLP model. Voor deze applicatie wordt een opvolger gesampled op basis van de gewichten tussen de eerste applicatie en alle andere applicaties W_{ij}. Formule 3 geeft de kans weer dat applicatie i gevolgd zal worden door applicatie j. Deze applicatie zal de target of het label vormen van deze sample.

$$P(i \to j) = W_{ij} / \sum_i W_{ij} \quad (3)$$

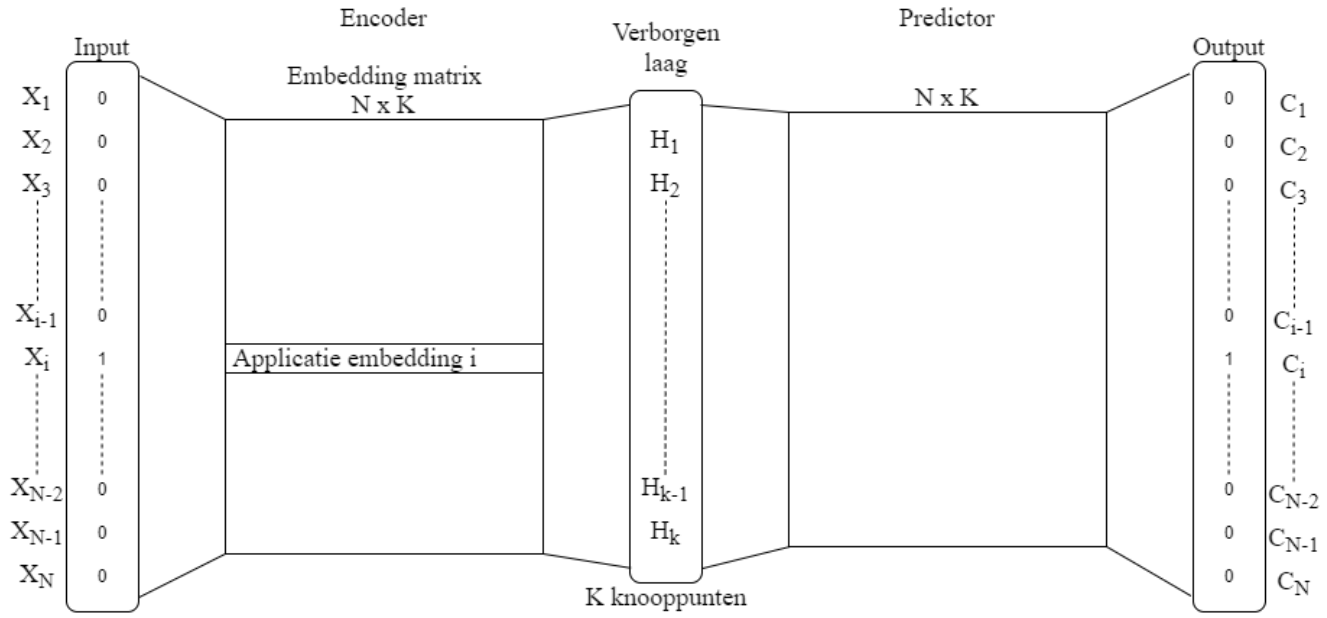
In de volgende stap zal er een MLP model getraind worden waaruit applicatieembeddings verkregen worden. Dit MLP zou direct getraind kunnen worden op de set van 2000 geopende applicaties. In het werk van Hong e.a. [6] beargumenteren ze echter dat gegenereerde data uit een event connection graph voor enkele voordelen kan zorgen. In de eerste plaats zal de gegenereerde data ook ongeziene, maar waarschijnlijke sequenties van applicaties bevatten. Dat komt omdat directed edges tussen twee applicaties ook een gewicht kunnen krijgen als deze applicaties nooit direct na elkaar geopend zijn. Daarnaast kunnen er meer dan 2000 samples verkregen worden uit de event connection graph. Hong e.a. [6] beargumenteert dat dat de onbalans in predictieklassen kan verhelpen.

3.4 Embedding pre-training

In sectie 2 werd reeds het concept van embeddings geïntroduceerd. In dit next-app predictieframework wordt eveneens gebruik gemaakt van embeddings voor het representeren van applicaties in de inputlaag van het GRU predictiemodel . De applicatie embeddings worden gebruikt als alternatief voor de zogenaamde one-hot representaties. Applicatie embeddings zullen in de eerste plaats het aantal waarden verlagen dat vereist is om een applicatie op een numerieke wijze voor te stellen aan het GRU next-app predictiemodel. Daarnaast zullen deze embeddings ook gelijkenissen en verschillen vertonen tussen applicaties. Deze gelijkenissen dienen niet langer aangeleerd te worden door het GRU next-app predictiemodel.

Het doel van deze embedding is om de sequentiële relatie tussen twee opeenvolgende applicaties te capteren. Het Multi-Layer Perceptron (MLP) model dat verantwoordelijk is voor de creatie van de embeddings, zal dat doen op basis van de event connection graph. Aangezien het voor next-app predictie niet interessant lijkt om applicaties te representeren op basis van de applicaties die er vooraf aan geopend worden, zal het skip-gram model enkel getraind worden op de applicaties die direct of indirect volgen op een applicatie.

Het MLP model wordt getoond in figuur 2. Deze bestaat uit drie lagen van knooppunten: een inputlaag, een verborgen laag en een outputlaag. De inputlaag en de outputlaag bestaan uit N knooppunten, waarbij N gelijk is aan het aantal applicaties of vertices in de event connection graph. Het aantal knooppunten K in de verborgen laag is een hyperparameter van het model. Naarmate K toeneemt, zal het model beter in staat zijn om de sequentiële relatie tussen applicaties vast te leggen, maar met dalende meeropbrengst [10]. De verborgen laag krijgt geen activatiefunctie en op de outputlaag wordt een softmax functie toegepast. De gewichtenmatrix tussen de inputlaag en de verborgen laag is dan van de vorm (N x K), terwijl de gewichtenmatrix tussen de



Figuur 2 Achitectuur Multilayer Perceptron

verborgen laag en de outputlaag van de vorm (K x N) is.

Het MLP model vereist een one-hot voorstelling van zowel inputs als labels. Daarom worden applicaties omgevormd tot one-hot vectoren van N waarden. Elk unieke applicatie correspondeert dan met een positie in de vector die de waarde 1 toegewezen krijgt. De N-1 andere posities in de vector krijgen de waarde 0. Vervolgens wordt het model getraind met categorical crossentropy loss, die een fout berekent tussen de softmax output van het model en het corresponderende one-hot label. De gewichten in het netwerk zullen zich dan aanpassen aan deze fout volgens de Adam optimizer.

Voor elke applicatie of vertex V van de event connection graph zal het MLP model dan de gewichtsdistributie van de edges aanleren. Applicaties die vaak gevolgd worden door een gelijkaardige set van applicaties, zullen ook een gelijkaardige embedded representatie krijgen.

De dimensie van de verborgen laag K heeft een impact op de top-k accuraatheid van het GRU predictiemodel. Naarmate K toeneemt, zal de accuraatheid van dit model ook toenemen. Voor K<16 zal deze accuraatheid sterk stijgen, waarna de meeropbrengst stagneert. Uit experimenten blijkt dat de optimale grootte van de verborgen laag K=32 bedraagt.

3.5 GRU next-app predictiemodel

Zodra het MLP model getraind is, kan de informatie uit de embedding laag gebruikt worden in het GRU predictiemodel. Deze embedding laag bevat nu de sequentiële relatie tussen applicaties uit een event connection graph. Het wordt dan de verantwoordelijkheid van het GRU next-app predictiemodel model om op basis van een sequentie van 20 applicatie embeddings een voorspelling te maken van de applicatie die de gebruiker zal openen.

Het GRU predictiemodel bestaat uit drie lagen: een embeddinglaag, een laag met 128 Gated Recurrent Units en een feedforward laag met 200 knooppunten. De gewichten in de embeddinglaag worden overgenomen van het getrainde MLP model. Elke applicatie komt overeen met een unieke index die duidt op de corresponderende rij in de embeddingmatrix. Het is de verantwoordelijkheid van de embeddinglaag om twintig applicaties, die voorgesteld worden door twintig indices, om te zetten naar twintig applicatie embeddings en deze door te geven aan de GRU laag. De GRU laag zal op basis van deze twintig embeddings een vector van 128 waarden produceren die doorgegeven wordt aan de feedforward laag. Tot slot zal de feedforward laag een vector van 200 zogenaamde logits produceren.

Op deze logits wordt een logsoftmax functie toegepast, waarna de categorical crossentropy loss berekend wordt. De Adam optimizer zal dan gebruikt worden om de gewichten in het GRU next-app predictiemodel aan te passen aan de predictiefout.

3.5.1 Pre-training

Omdat de Event Connection Graph toch al een geschiedenis van 2000 geopende applicaties vereist, wordt het GRU predictiemodel ook nog pre-trained op deze sub-

set. Voor deze pre-training fase traint het model in batches van 16 samples. Het model blijft ook een stateless GRU model voor de pre-training fase, waarbij de hidden state op nul geïnitialiseerd wordt tussen batches. De Adam optimizer zal gebruikt worden voor de pre-training fase met een learning rate van 0.001. Daarnaast worden de gewichten in de embeddinglaag niet verder aangepast tijdens de pre-training fase van het GRU model.

Om overfitting te voorkomen, wordt er een stopconditie gebruikt voor het beëindigen van de pre-training fase. Hiervoor wordt de geschiedenis van 2000 geopende applicaties opgedeeld in een trainingset van 1600 samples en een testset van 400 samples. De testset wordt niet gebruikt om de gewichten van het GRU predictiemodel bij te stellen, maar wordt wel gebruikt om de categorical crossentropy loss te meten. De minimalisatie van deze loss op de testset is dan de stopconditie van de pre-training fase van het GRU predictiemodel.

Omdat het GRU predictiemodel gebruik maakt van applicatie embeddings werd de pre-training fase reeds stopgezet na drie epochs, wanneer het model elk datapunt driemaal gezien heeft. Dit is langer voor modellen die gebruik maken van one-hot representaties, waar tien epochs vereist zijn om de loss op de testset te minimaliseren.

3.5.2 Online machine learning

Na 2000 geopende applicaties zal het model verantwoordelijk worden voor het voorspellen van geopende applicaties. Het next-app predictieprobleem wordt hier als een online machine learning probleem gezien, waarvoor het GRU predictiemodel per sample een predictie dient te maken waarvoor het moet corrigeren voor een fout. De batch size van deze dataset is dus met andere woorden gelijk aan één. Aangezien de samples elkaar chronologisch opvolgen, wordt er best voorkomen dat dit model zijn hidden state tussen elke batch initialiseert met nulwaarden. Om die reden wordt er gekozen om het GRU model stateful te maken voor de online learning dataset, waarbij de hidden state van de vorige batch of sample de hidden state van de volgende predictie initialiseert. De learning rate van de Adam optimizer bedraagt ook 0.001 voor de online learning dataset.

Tijdens de online learning fase van het GRU model, zal deze ook applicaties tegenkomen die hij nooit eerder heeft kunnen zien. Aan de inputzijde van het model kan er dan een extra embedding toegevoegd worden aan de embedding matrix. Aangezien deze embedding willekeurig geïnitialiseerd wordt, kunnen de gewichten in de embeddinglaag van het GRU model bijgesteld worden voor de online-learning dataset. Voor nieuwe applicaties moet er aan de outputzijde van het model voor een overprovisionering van 200 knooppunten gezorgd worden.

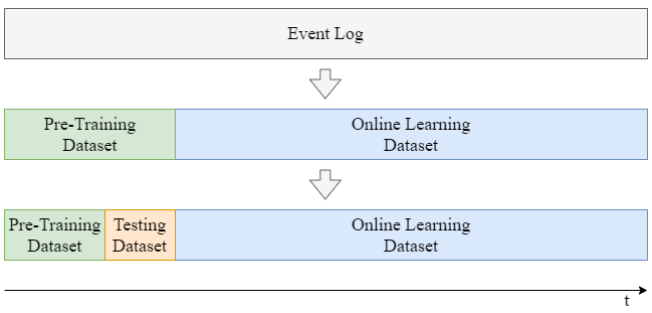
4 Methodologie

4.1 Beschrijving van de gebruikte datasets

De next-app predictiemodellen worden in deze paper toegepast en geëvalueerd op drie event logs. Elk van deze event logs is afkomstig van een Android toestel. Google houdt voor deze toestellen een geschiedenis van interacties bij zolang de gebruiker de toestemming hiervoor niet ontzegt. Als de gebruiker toegang heeft tot het Google account dat gekoppeld is aan het Android toestel, dan kan hij deze gebruiksgeschiedenis opvragen en downloaden in JSON formaat via Google Takeout.

Deze gebruiksgeschiedenis bevat, naast geopende applicaties, ook andere interacties met het toestel. Zo worden er ook events geregistreerd als de gebruiker zijn scherm aanzet of als hij zijn telefoon ontgrendelt en vergrendelt. Daarnaast bevatten deze event logs ook hulpen systeemfuncties die samen aangeroepen worden bij het openen van bepaalde applicaties. Alle events die niet rechtstreeks wijzen op het openen van een applicatie, worden handmatig verwijderd uit de event logs. Tot slot wordt er gekozen om niet te filteren op basis van de gebruiksfrequentie van applicaties. Op die manier wordt de performantie van de predictiemodellen zo realistisch mogelijk geschilderd.

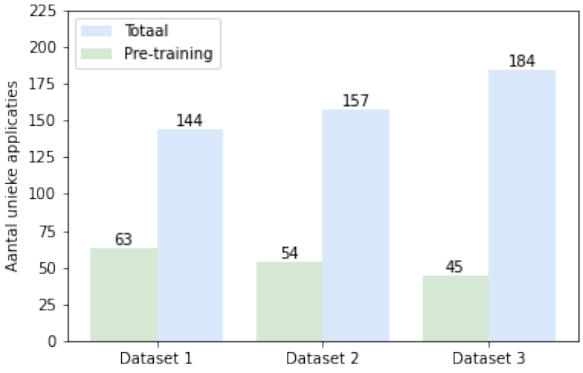
Na het filteren van de event logs, zullen deze nog ingedeeld worden in pre-training datasets en online learning datasets. Deze indeling wordt getoond in figuur 3. Er wordt gekozen om de eerste 2000 geopende applicaties in een pre-training dataset te stoppen. De pre-training dataset zal dan in eerste instantie gebruikt worden om een event connection graph te construeren. Deze event connection graph kan dan gebruikt worden om data te genereren voor het trainen van het MLP model, waar de applicatie embeddings uit voortkomen.



Figuur 3 Dataset pre-processing

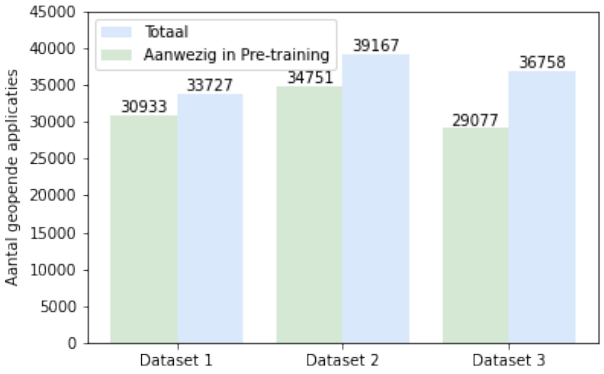
Daarnaast toont figuur 3 hoe de pre-training dataset opgesplitst wordt in een train- en testset van respectievelijk 1600 en 400 geopende applicaties. De trainingset wordt gebruikt om de RNN predictiemodellen reeds te pre-trainen. Op die manier krijgen modellen die niet gebruik maken van applicatie embeddings ook de kans om te trainen vooraleer ze predicties maken. De ongeziene testset van 400 applicaties wordt gebruikt om overfitting te voorkomen door de pre-trainingsfase vroegtijdig stop te zetten. Een lagere categorical cross-entropy loss op deze testset geeft aan dat een RNN model beter in staat is om voorspellingen te maken van ongeziene data. Daarom wordt de pre-trainingsfase stopgezet wanneer de loss op de testset zijn minimum heeft bereikt.

In figuur 4 wordt het totaal aantal unieke applicaties per dataset getoond. De eerste dataset bevat het kleinste aantal geïnstalleerde applicaties. Daarnaast bevat het pre-training gedeelte van deze dataset ook het grootste aantal unieke applicaties. Voor dataset één kunnen de RNN predictiemodellen dus starten aan de online learning dataset met kennis over een grotere fractie van het totaal aantal geïnstalleerde applicaties.



Figuur 4 Totaal aantal unieke applicaties in elke dataset en het aantal unieke applicaties beschikbaar tijdens de pre-training fase.

Figuur 5 geeft het totaal aantal geopende applicaties per dataset weer. Dit komt overeen met het aantal geregistreerde events in elke dataset. Daarnaast wordt hetzelfde ook weergegeven voor applicaties die reeds voorkomen in de pre-training datasets. Hoewel alle pre-training datasets minder dan de helft van de unieke applicaties bevatten, kan er uit deze figuur afgeleid worden dat dit wel de voornaamst gebruikte applicaties zijn.



Figuur 5 Totaal aantal geopende applicaties en het totaal aantal geopende applicaties die reeds beschikbaar zijn tijdens de pre-training fase.

4.2 Referentieheuristieken - en predictiemodellen

Het voorgestelde predictiemodel binnen het framework zal worden vergeleken met twee populaire heuristieken binnen next-app predictie en zes alternatieve RNN modellen met architecturale verschillen. De baseline heuristieken zijn belangrijk om de kenmerken van de datasets weer te geven. Deze twee heuristieken maken gebruik van gedragspatronen die machine learning algoritmes eveneens kunnen exploiteren om een hogere accuraatheid te verkrijgen. De volgende twee heuristieken zullen gebruikt worden ter referentie:

- Most Frequently Used (MFU) heuristiek: Uit een geschiedenis van geopende applicaties worden de vijf meest geopende applicaties bijgehouden. Deze vijf applicaties worden voorgesteld als next-app predictie.
- Most Recently Used (MRU) heuristiek: Uit een geschiedenis van geopende applicaties worden de vijf laatst geopende applicaties bijgehouden. Deze vijf applicaties worden voorgesteld als next-app predictie.

Alle RNN modellen die als referentie gebruikt worden, zullen de next-app predictie baseren op een gebruiksgeschiedenis van twintig geopende applicaties. Om een antwoord te bieden op de onderzoeksvragen van dit onderzoek, kunnen de RNN modellen op twee manieren verschillen van elkaar. Het eerste verschil is te vinden in de architectuur van de RNN laag. De RNN laag van het model kan een LSTM laag, een GRU laag of een BiGRU laag zijn. Het tweede verschil is te vinden in de manier waarop applicaties numeriek worden voorgesteld aan het model. Een applicatie kan zo een one-hot representatie ontvangen of hij kan voorgesteld worden door een aangeleerde embedding.

Deze referentie modellen zullen ceteris paribus vergeleken worden, waarbij opeenvolgende modellen slechts op één vlak verschillen. Tabel 2 beschrijft de verschillende combinaties van RNN lagen en representatiemethoden die voorkomen in de experimenten.

Tabel 2 Referentie RNN modellen in de experimenten.

RNN laag	Applicatie representatie
64 LSTM units	one-hot
128 LSTM units	one-hot
64 GRU units	one-hot
128 GRU units	one-hot
64x2 Bidirectional GRU units	one-hot
128 GRU units	embedding
64x2 Bidirectional GRU units	embedding

4.3 Evaluatie van de next-app predictiemodellen en heuristieken

Nadat de pre-train fase van de top-k next-app predictiemodellen is stopgezet, zullen de modellen en heuristieken uit sectie 4.2 toegepast worden op de online learning dataset. De predictiemodellen zullen de applicaties in sequentiële volgorde voorspellen. De top-5 accuraatheid op deze online-learning dataset zal doorslaggevend zijn voor het bepalen van het optimale model.

Voor de heuristieken wordt een top-5 voorspelling accuraat geacht wanneer de volgende applicatie kan worden gevonden in de voorgestelde set van vijf applicaties. Voor de RNN modellen wordt een softmax functie toegepast op de output. Deze softmax functie zorgt ervoor dat de output bestaat uit de voorspelde kansen voor elke next-app kandidaat. De vijf applicaties met de hoogste kans worden dan voorgesteld als top-5 next app predictie. De top-5 predictie is dan accuraat wanneer de volgende app ook onder deze voorgestelde set valt.

4.4 Hyperparameter configuratie van modellen met embeddings

De hyperparameters van de predictiemodellen die gebruik maken van embeddings zullen eveneens geoptimaliseerd worden op basis van een maximalisatie van de top-5 accuraatheid op de online learning datasets. Deze hyperparameters worden weergegeven in tabel 3. Voor de constructie van de Event Connection Graph zullen er op deze manier optimale waarden voor de hyperparameters T en Δ bepaald worden. Voor het MLP model zal de optimale waarde van de verborgen laag K bepaald worden, die de dimensionaliteit van de applicatieembeddings bepaalt.

Tabel 3 Geteste waarden van hyperparameters T, Δ en K.

T	Δ	K
1200	300	8
2400	300	16
3600	300	24
1200	900	32
2400	900	40
3600	900	48
1200	1200	56
2400	1200	64
3600	1200	-

Omwille van de onderlinge afhankelijkheid tussen T en Δ in de berekening van δ worden er combinaties van waarden voor hyperparameters T en Δ getest. De geteste waarden voor deze twee hyperparameters zijn analoog aan de waarden uit de paper van Xu e.a. [16]. Voor de waarden van K werd gekozen om veelvouden van acht te testen tot er een afnemende meeropbrengst in top-5 accuraatheid te bemerken was.

4.5 Implementatie van de next-app predictiemodellen en heuristieken

Alle predictiemodellen en heuristieken werden geschreven in Python 3.7.5, waar de PyTorch bibliotheek zorgt voor een efficiënte implementatie van RNN modellen door middel van just-in-time compilatie. Daarnaast werd er gebruik gemaakt van de CUDA 11.1 libraries van Nvidia. PyTorch, Pandas en Numpy werden gebruikt om de datasets uit te lezen en om te vormen naar trainingsdata. Alle experimenten werden uitgevoerd op een Intel i7 9700k met 32 gigabyte werkgeheugen en een Nvidia GTX 1070. Alle predictiemodellen en heuristieken voltooien hun tests in minder dan twee minuten.

5 Resultaten

5.1 Hyperparameter optimalisatie op het GRU predictiemodel

5.1.1 Effect embedding dimensie op online-learning accuraatheid

In tabel 4 wordt de top-5 accuraatheid weergegeven van het GRU predictiemodel voor verschillende embedding dimensies. Deze embedding dimensie bepaalt het aantal numerieke waarden dat het MLP model aanleert als

representatie van elke applicatie. Uit deze tabel kan afgeleid worden dat alle embedding dimensies vanaf 16 voor een gelijkaardige top-5 accuraatheid zorgen. Om hier toch de beste waarde uit te kiezen, wordt de gemiddelde top-5 accuraatheid over de drie datasets genomen. Hieruit blijkt dat een embedding dimensie van 32 leidt tot de hoogste gemiddelde top-5 accuraatheid van 66.35%. Dit is dan ook de grootte van de embedding dimensie voor verdere experimenten en vergelijkingen.

Tabel 4 Effect embedding dimensie op de top-5 accuraatheid van het GRU predictiemodel

Emb dim	Dataset 1	Dataset 2	Dataset 3	Gem.
8	74.71%	67.40%	53.58%	65.23%
16	76.21%	67.74%	53.86%	65.94%
24	76.97%	67.77%	53.80%	66.18%
32	77.16%	67.85%	54.03%	**66.35%**
40	77.03%	67.49%	53.96%	66.16%
48	77.09%	67.49%	53.61%	66.06%

5.1.2 Effect Event Connection Graph constructie op online-learning accuraatheid

In tabel 5 wordt de top-5 accuraatheid weergegeven voor negen combinaties van hyperparameters T en Δ. Deze beïnvloeden de constructie van de event connection graph. T bepaalt de maximale tussentijd in seconden tussen twee geopende applicaties die nog als gerelateerd beschouwd kunnen worden. Δ bepaalt hoe snel de relatie tussen twee geopende applicaties daalt naarmate hun tussentijd groter wordt.

Tabel 5 Effect hyperparameters van Event Connection Graph op de top-5 accuraatheid van het GRU predictiemodel

T (s)	Δ	Dataset 1	Dataset 2	Dataset 3
1200	300	76.97%	67.68%	54.07%
2400	300	77.02%	67.66%	53.72%
3600	300	76.92%	67.51%	53.77%
1200	900	77.09%	67.74%	53.75%
2400	900	77.01%	67.68%	53.91 %
3600	**900**	**77.11%**	**67.85%**	**53.53 %**
1200	1200	76.69%	67.67%	53.72%
2400	1200	77.01%	67.79%	53.71 %
3600	1200	76.99%	67.73%	54.12%

Hoewel de geobserveerde verschillen in accuraatheid miniem zijn, kunnen er toch kleine verschillen gemerkt worden in de gemiddelde top-5 accuraatheid voor verschillende waarden van hyperparameters. Voor de hyperparameter T geldt dat een waarde van T=3600 zorgt voor de hoogste gemiddelde top-5 accuraatheid over de datasets van 66.17%. Voor T=1200 en T=2400 bedraagt dat respectievelijk 66.15% en 66.17%. Voor de hyperparameter Δ geldt dat een waarde van Δ=900 zorgt voorde hoogste gemiddelde top-5 accuraatheid van 66.19%. Voor Δ=300 en Δ=1200 bedraagt dat 66.17% en 66.16%.

Voor T=3600 en Δ=900 behaald het GRU predictiemodel een gemiddelde accuraatheid van 66.16% over de drie datasets. De hyperparameters T en Δ zullen deze waarden bijgevolg aannemen voor verdere experimenten en vergelijkingen.

5.2 Vergelijking performantie met baseline- en lstm predictiemodellen

In tabel 6 wordt de accuraatheid van elk getest model weergegeven. Als baseline modellen worden hier ook de Most Frequently Used (MFU) en Most Recently Used (MRU) heuristieken weergegeven ter vergelijking. De performantie van de MFU en MRU heuristieken variëren van dataset tot dataset. Het applicatiegebruik in dataset 1 kan best beschreven worden door de MRU heuristiek, waar dat voor dataset 2 en 3 best door de MFU heuristiek beschreven kan worden. De RNN modellen kunnen deze heuristieken dan exploiteren om de accuraatheid van hun top-5 next-app predictie te verhogen. De performantie van de MFU en MRU heuristieken komen dan ook overeen met de accuraatheid van de RNN modellen op de verschillende datasets.

Tabel 6 Vergelijking in accuraatheid online-learning datasets

Predictiemodel		Top-5 Accuraatheid		
Predictor	Features	Dataset 1	Dataset 2	Dataset 3
MFU	-	62.63%	60.93%	41.47%
MRU	-	74.28%	53.30%	32.62%
lstm64	one-hot	72.24%	66.39%	51.05%
lstm128	one-hot	73.10%	66.90 %	51.65%
gru64	one-hot	74.91%	67.40%	52.27%
gru128	one-hot	76.10%	67.58%	52.63%
bigru64x2	one-hot	75.51%	67.21%	52.21%
gru128	embed	**77.11**%	**67.85**%	53.53%
bigru64x2	embed	76.50%	67.53 %	**53.58**%

Zowel het Long-Short Term Memory (LSTM) model met 64 verborgen LSTM units als met 128 LSTM units slagen er slechts in om voor twee van de drie datasets een hogere accuraatheid te behalen dan de MRU of MFU heuristiek. Het verhogen van het aantal verborgen LSTM units naar 128 zorgt wel voor een verbetering in accuraatheid.

De Gated Recurrent Unit (GRU) modellen slagen er daarentegen wel in om een hogere accuraatheid te behalen dan de heuristieken op alle online learning datasets. Het GRU model met 64 verbogen GRU units slaagt er eveneens in om een hogere accuraatheid te behalen op alle datasets dan het LSTM model met zelfs 128 verborgen units. Van alle modellen die gebruik maken van one-hot representaties van applicaties in de inputlaag behaalt het GRU model met 128 verborgen units de hoogste accuraatheid op de online learning datasets.

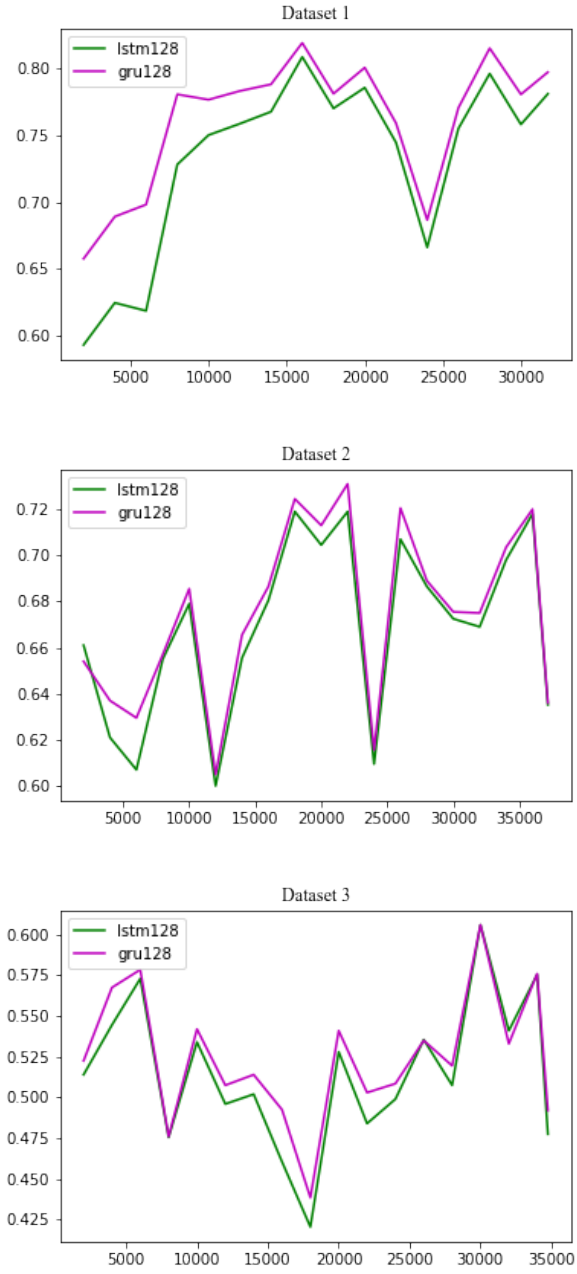
Het volgende model is een Bidirectional Gated Recurrent Unit (BiGRU) model met 64 verborgen GRU units per sequentievolgorde, voor een totaal van 128 GRU units. Deze slaagt er in om een gelijkaardige accuraatheid te behalen als het Unidirectional GRU model met een totaal van 64 verborgen GRU units. Er kan echter niet gesteld worden dat een Bidirectional model met 128 verborgen units erin slaagt om een hogere accuraatheid te behalen dan het Unidirectional GRU model dat eveneens 128 verborgen units telt.

Voor het GRU model en BiGRU model met een totaal van 128 units werden er ook embeddings geïntroduceerd ter vervanging van de one-hot voorstellingen van applicaties. Het idee achter deze embeddings bestond erin om applicaties voor te stellen met minder numerieke waarden. Een bijkomend voordeel van deze embeddings is dat gelijkaardige applicaties ook gelijkaardige representaties krijgen. De embeddings, zoals ze hier verkregen en toegepast worden, leiden zowel voor het GRU als het BiGRU model voor een hogere top-5 accuraatheid op alle datasets.

5.3 Vergelijking van accuraatheid doorheen de tijd

In de vorige sectie wordt er geconstateerd dat het Gated Recurrent Unit model met 128 verborgen units en embeddings voor de hoogste gemiddelde top-5 accuraatheid zorgt. Vervolgens kan dan bekeken worden hoe dit GRU model erin slaagt om deze hogere accuraatheid te bereiken. Om dit te achterhalen zal de veranderlijke accuraatheid van elk model op de online learning dataset vergeleken worden. Omwille van de relatief slechte performantie van de MRU en MFU modellen, zullen deze voor de volgende vergelijkingen buiten beschouwing gelaten worden.

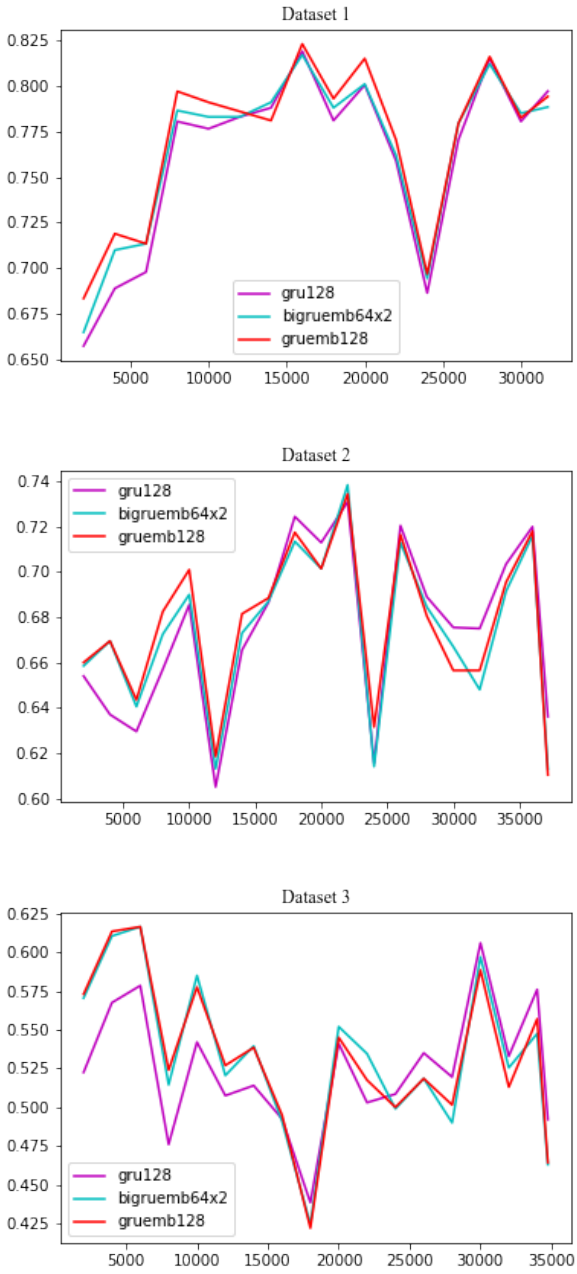
In figuur 6 wordt de veranderlijke top-5 accuraatheid vergeleken voor de LSTM en GRU modellen met 128 verborgen units. De gemiddelde accuraatheid wordt er getoond voor elke set van 2000 sequentieel geopende applicaties. Voor dataset 1 geldt dat deze reeds substantieel hogere accuraatheid heeft voor de eerste 2000 geopende applicaties. Deze voorsprong behoudt het doorheen de online learning dataset om een hogere gemiddelde accuraatheid te bekomen. Voor dataset 2 en 3 is er geen sprake van deze vroege voorsprong, maar voor deze data sets lijkt het model een hogere accuraatheid te behouden wanneer de gebruiker onvoorspelbaarder wordt.



Figuur 6 Vergelijking van de accuraatheid op de online learning datasets tussen het LSTM en het GRU model met 128 verborgen units en one-hot representaties van applicaties.

Het Bidirectional Gated Recurrent Unit model (BiGRU) codeert de sequentie van geopende applicaties

in twee richtingen. Dit zal ervoor zorgen dat de grootte van laag met gated recurrent units zal verdubbelen. Net als bij het GRU model met 128 verborgen GRU units zal een BiGRU laag met 64x2 verborgen units dus 128 numerieke waarden produceren voor de feed forward laag die erop volgt. Daarnaast werden er ook applicatie embeddings geïntroduceerd voor het GRU en BiGRU model. Deze vervangen de one-hot representatie die applicaties kregen voor de vorige modellen.



Figuur 7 Vergelijking van de accuraatheid op de online learning dataset tussen het GRU model met one-hot encoderingen en het GRU model met embeddings en het BiGRU model met embeddings.

In figuur 7 wordt de accuraatheid getoond van zowel de Unidirectional als het Bidirectional Gated Recurrent modellen met en zonder applicatie embeddings. Voor alle datasets kan gesteld worden dat de modellen die gebruik maken van applicatie embeddings met een hogere top-5 accuraatheid starten aan de online-learning dataset. Deze vroege voorsprong in accuraatheid is dan ook verantwoordelijk voor de hogere gemiddelde accuraatheid van deze modellen. Dat aangezien de modellen convergeren in top-5 accuraatheid later in de datasets. Voor dataset twee en drie kan zelfs gesteld worden dat de modellen die gebruik maken van applicatie embeddings een lagere top-5 accuraatheid behalen.

6 Discussie

De eerste onderzoeksvraag binnen dit onderzoek bestond erin de performantie van een LSTM architectuur te vergelijken met een alternatief GRU model. Uit de resultaten van dit onderzoek bleek dat een vergelijkbaar GRU next-app predictiemodel wel degelijk een betere top-5 predictie kan maken van de volgende geopende

applicatie. Deze hogere gemiddelde top-5 accuraatheid komt enerzijds voort uit een hogere initiële accuraatheid na het pre-trainen van het model. Het GRU model kan dus op eenzelfde pre-training dataset van 1600 geopende applicaties een hogere accuraatheid behalen dat generaliseerbaar is naar de online-learning dataset. Anderzijds slaagt het GRU model erin om deze voorsprong in top-5 accuraatheid verder te zetten doorheen de online-learning dataset. De veronderstelling is dat de Gated Recurrent Units beter presteren dan de Long-Short Term Memory units omdat ze minder trainbare parameters bevatten. Deze verlaagde complexiteit zorgt ervoor dat deze in een online-learning context, waar er bij voorkeur een kleine pre-training dataset gebruikt dient te worden, een hogere accuraatheid kan behalen.

De tweede onderzoeksvraag van dit onderzoek had betrekking op het gebruik van een Bidirectional Recurrent Neural Network model voor het encoderen van de geschiedenis van applicatiegebruik. Deze encodering zorgt ervoor dat elke applicatie gekarakteriseerd kan worden door de applicaties die er aan voorafgaan en erop volgen. Voor zowel regressie- als classificatieproblemen kan dit voor betere prestaties zorgen van een RNN model [13]. Uit de resultaten van deze studie blijkt echter dat een Bidirectional RNN model met Gated Recurrent Units geen hogere top-5 accuraatheid behaalt op de next-app predictie dan een vergelijkbaar Unidirectional RNN. Er zou gesteld kunnen worden dat er voor dit predictieprobleem niet genoeg meerinformatie ontstaat om te compenseren voor de bijkomende complexiteit van een Bidirectional RNN model.

Tot slot werd er binnen dit onderzoek ook gebruik gemaakt van embeddings als alternatief voor de one-hot voorstelling van applicaties. De applicatie embeddings, zoals ze hier tot stand gekomen zijn en toegepast worden, hebben geleid tot een hogere gemiddelde top-5 accuraatheid van het BiGRU en GRU next-app predictiemodel.

Om aan het veranderlijke gedrag van gebruikers toe te komen werd er ook gekozen om de gewichten van de embedding laag trainbaar te houden. Op die manier kan het model de veranderlijke sequentiële relatie tussen applicaties updaten. Anderzijds stelt dat het model in staat om een embedding aan te leren voor applicaties die geen pre-trained embedding hebben kunnen krijgen. Desalniettemin lijkt het dat deze modellen in accuraatheid afnemen naarmate er nieuwe applicaties geïnstalleerd worden.

Een interessant pad voor verder onderzoek zou zijn om aan de huidige tekortkomingen van applicatie embeddings toe te komen. Deze tekortkomingen zouden aangepakt kunnen worden door universele applicatie embeddings te verkrijgen en toe te passen in combinatie met een GRU predictiemodel. Deze universele applicatie embeddings zouden niet langer verkregen worden uit de event log van één gebruiker, maar event logs van verschillende gebruikers. Als er voldoende event logs gebruikt worden voor het pre-trainen van deze embeddings, zullen deze embeddings de meest populaire applicaties bevatten. Het downstream GRU predictiemodel zal de embeddings dan niet langer moeten bijstellen, maar kan gebruik maken van unknown tokens voor de weinige onbekende applicaties die overblijven aan de inputzijde van het model.

Voor het verkrijgen van deze embeddings kan er ook gebruikt gemaakt worden van Transformer netwerken in plaats van het Skip-Gram model. Transformer netwerken worden gebruik binnen Natural Language Processing in onder andere BERT en GPT-3 om context-afhankelijke embeddings van woorden te verkrijgen. Voor applicatie embeddings zou dat betekenen dat applicaties die veelzijdig zijn in gebruik, zoals Google Chrome, een numerieke representatie zullen krijgen afhankelijk van de applicaties die ervoor en erna geopend worden.

7 Conclusie

Uit de resultaten van dit onderzoek blijkt dat Gated Recurrent Units (GRU) geschikter zijn dan Long Short-Term Memory (LSTM) voor top-5 next-app predictie op basis van een korte geschiedenis van twintig geopende applicaties. Er wordt verondersteld dat de verlaagde complexiteit van een GRU voordelen biedt wanneer een kleine pre-training dataset wenselijk is en wanneer het predictie probleem zich in een online learning context plaatsvindt.

Een Bidirectional GRU model behaalt geen hogere accuraatheid dan een Unidirectional GRU model voor de top-5 next-app predictie. Vermoedelijk weegt de meerinformatie van de tweerichtings encodering niet op tegen de bijkomende complexiteit van het Bidirectional model.

Embeddings verkregen uit een korte gebruiksgeschiedenis van één gebruiker door middel van een event connection graph zorgt voor een verhoogde accuraatheid van de BiGRU en GRU next-app predictiemodellen in vergelijking met het one-hot alternatief. Deze verhoogde accuraatheid is het gevolg van een hogere accuraatheid waarmee de modellen met embeddings aan de online-learning dataset starten. Een gecondenseerde representatie van applicaties biedt dus voordelen, maar het model lijkt niet in staat dit voordeel te behouden later in de online-learning datasets. Dit lijkt te komen doordat het RNN model verantwoordelijk is heeft om bestaande applicatie embeddings up-to-date te houden en nieuwe applicatie embeddings aan te leren.

Referenties

[1] Hong Cao en Miao Lin. "Mining smartphone data for app usage prediction and recommendations: A survey". In: *Pervasive and Mobile Computing* 37 (20 jan 2017), p. 1–22. DOI: 10.1016/j.pmcj.2017.01.007.

[2] Xinlei Chen e.a. "CAP: Context-aware App Usage Prediction with Heterogeneous Graph Embedding". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3 (29 mrt 2019), p. 1–25. DOI: 10.1145/3314391.

[3] Kyunghyun Cho e.a. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: (3 jun 2014). DOI: 10.3115/v1/D14-1179.

[4] Junyoung Chung e.a. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *arXiv:1412.3555 [cs]* (11 dec 2014). arXiv: 1412.3555. (Bezocht op 05-08-2021).

[5] Sepp Hochreiter en Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (1 dec 1997), p. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[6] Shenda Hong e.a. "Event2vec: Learning Representations of Events on Temporal Sequences". In: 3 aug 2017, p. 33–47. ISBN: 978-3-319-63563-7. DOI: 10.1007/978-3-319-63564-4_3.

[7] Anna Huang. "Similarity Measures for Text Document Clustering". In: (2008).

[8] Ke Huang e.a. "Predicting mobile application usage using contextual information". In: 5 sep 2012, p. 1059–1065. DOI: 10.1145/2370216.2370442.

[9] Zhung-Xun Liao e.a. "On the Feature Discovery for App Usage Prediction in Smartphones". In: *2013 IEEE 13th International Conference on Data Mining*. 2013 IEEE 13th International Conference on Data Mining. ISSN: 2374-8486. Dec 2013, p. 1127–1132. DOI: 10.1109/ICDM.2013.130.

[10] Tomas Mikolov, Wen-tau Yih en Geoffrey Zweig. "Linguistic Regularities in Continuous Space Word Representations". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. NAACL-HLT 2013. Atlanta, Georgia: Association for Computational Linguistics, jun 2013, p. 746–751. (Bezocht op 02-05-2021).

[11] Tomas Mikolov e.a. "Efficient Estimation of Word Representations in Vector Space". In: *arXiv:1301.3781 [cs]* (6 sep 2013). arXiv: 1301.3781.

[12] Razvan Pascanu, Tomas Mikolov en Y. Bengio. "On the difficulty of training Recurrent Neural Networks". In: *30th International Conference on Machine Learning, ICML 2013* (21 nov 2012).

[13] M. Schuster en K.K. Paliwal. "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11 (nov 1997). Conference Name: IEEE Transactions on Signal Processing, p. 2673–2681. ISSN: 1941-0476. DOI: 10.1109/78.650093.

[14] Zhihao Shen e.a. "DeepAPP: a deep reinforcement learning framework for mobile application usage prediction". In: *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*. SenSys '19. New York, NY, USA: Association for Computing Machinery, 10 nov 2019, p. 153–165. ISBN: 978-1-4503-6950-3. DOI: 10.1145/3356250.3360038. (Bezocht op 24-06-2021).

[15] Alex Sherstinsky. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network". In: *Physica D: Nonlinear Phenomena* 404 (1 mrt 2020), p. 132306. ISSN: 0167-2789. DOI: 10.1016/j.physd.2019.132306. (Bezocht op 07-07-2021).

[16] Shijian Xu e.a. "Predicting Smartphone App Usage with Recurrent Neural Networks". In: *Wireless Algorithms, Systems, and Applications*. Red. door Sriram Chellappan, Wei Cheng en Wei Li. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, p. 532–544. ISBN: 978-3-319-94268-1. DOI: 10.1007/978-3-319-94268-1_44.

[17] Shudong Yang, Xueying Yu en Ying Zhou. "LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example". In: *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*. 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI). Jun 2020, p. 98–101. DOI: 10.1109/IWECAI50956.2020.00027.

[18] Sha Zhao e.a. "AppUsage2Vec: Modeling Smartphone App Usage for Prediction". In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 2019 IEEE 35th International Conference on Data Engineering (ICDE). ISSN: 2375-026X. Apr 2019, p. 1322–1333. DOI: 10.1109/ICDE.2019.00120.

[19] Yifei Zhou, Shaoyong Li en Yaping Liu. "Graph-based Method for App Usage Prediction with Attributed Heterogeneous Network Embedding". In: *Future Internet* 12.3 (mrt 2020). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 58. DOI: 10.3390/fi12030058. (Bezocht op 02-06-2021).