Machine Learning Training Data Generator

Studienarbeit

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Leunar Kalludra, Frederik Wagner

03.06.2019

Matrikelnummer, Kurs	7217648, 9417344, STG-TINF16D
Betreuer	Prof. Dr. Dirk Reichardt
Gutachter	Prof. Dr. Dirk Reichardt

Sperrvermerk

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anderslautende Genehmigung der Ausbildungsstätte vorliegt.

, 03.06.2019

_____

Leunar Kalludra, Frederik Wagner

Erklärung

Ich versichere hiermit, dass ich meine Studienarbeit mit dem Thema: *Machine Learning Training Data Generator* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

, 03.06.2019

_____

Leunar Kalludra, Frederik Wagner

Abstract

Abstract normalerweise auf Englisch. Siehe: http://www.dhbw.de/fileadmin/user/public/Dokumente/Portal/Richtlinien_Praxismodule_Studien_und_Bachelorarbeiten_JG2011ff.pdf (8.3.1 Inhaltsverzeichnis)

Ein „Abstract“ ist eine prägnante Inhaltsangabe, ein Abriss ohne Interpretation und Wertung einer wissenschaftlichen Arbeit. In DIN 1426 wird das (oder auch der) Abstract als Kurzreferat zur Inhaltsangabe beschrieben.

Objektivität soll sich jeder persönlichen Wertung enthalten

Kürze soll so kurz wie möglich sein

Genauigkeit soll genau die Inhalte und die Meinung der Originalarbeit wiedergeben

Üblicherweise müssen wissenschaftliche Artikel einen Abstract enthalten, typischerweise von 100-150 Wörtern, ohne Bilder und Literaturzitate und in einem Absatz.

Quelle: http://de.wikipedia.org/wiki/Abstract Abgerufen 07.07.2011

Diese etwa einseitige Zusammenfassung soll es dem Leser ermöglichen, Inhalt der Arbeit und Vorgehensweise des Autors rasch zu überblicken. Gegenstand des Abstract sind insbesondere

- Problemstellung der Arbeit,
- im Rahmen der Arbeit geprüfte Hypothesen bzw. beantwortete Fragen,
- der Analyse zugrunde liegende Methode,
- wesentliche, im Rahmen der Arbeit gewonnene Erkenntnisse,
- Einschränkungen des Gültigkeitsbereichs (der Erkenntnisse) sowie nicht beantwortete Fragen.

Quelle: http://www.ib.dhbw-mannheim.de/fileadmin/ms/bwl-ib/Downloads_alt/Leitfaden_31.05.pdf, S. 49

Inhaltsverzeichnis

Abkürzungsverzeichnis

AGPL Affero GNU General Public License
PRNG Pseudorandom Number Generator

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung

Insg. 6 Seiten

In letzter Zeit werden lernende Algorithmen in immer mehr Bereichen eingesetzt [1]. Diese Algorithmen müssen mit Daten trainiert werden. Allerdings ist es schwierig, geeignete Datensätze zu finden. Dies gilt besonders für akademische Zwecke, da hierbei oft mehrere Datensätze ähnlicher Art benötigt werden, die aber leichte Unterschiede aufweisen. So kann in der Vorlesung mit einem Datensatz geübt werden und in der Klausuraufgabe ein ähnlicher, den Studenten aber bisher unbekannter Datensatz, verwendet werden.

Im Moment werden die Daten durch Formeln in einer Tabellenkalkulation erstellt. Mit dieser Vorgehensweise ist es allerdings kompliziert, Zusammenhänge darzustellen. Auch sind Randbedingungen, wie z. B. "mind. 10 positive Beispiele generieren"nur mit hohem Aufwand umsetzbar.

1.1 Aufgabenstellung

1.2 Anforderungsanalyse

2 Stand der Technik

Themenumfeld und Stand der Technik erörtern (insg. 3 Seiten)

2.1 Datengeneratoren

Es gibt einige Datengeneratoren; diese sind jedoch extrem auf einen bestimmten Anwendungsfall zugeschnitten.

- **Scikit-Learn**: Die Machine Learning Bibliothek scikit-learn bietet Funktionalität, um Beispieldaten zu generieren. Allerdings bietet sie nur vorgefertigte Funktionen, die bestimmte Muster zufällig generieren. Die Funktionen lassen sich kaum anpassen und es ist nicht möglich, damit ein komplexes Modell abzubilden. [2, 3]
- **SynthOSNdataGenerator**: Dieses Programm ist in der Lage, die Struktur eines sozialen Netzwerks zu erzeugen. Auch hier ist die Funktionalität sehr auf der Anwendungsfall angepasst und nicht universell genug, um als Basis für die in dieser Arbeit zu entwickelnde Applikation zu dienen. [4]
- **log-synth**: Das Tool log-synth ist in der Lage, künstliche Log-Dateien zu erzeugen. Es erlaubt eine sehr flexible Konfiguration und hat bereits viele eingebaute Funktionen, jedoch ist es sehr schwierig, Zusammenhänge zwischen mehreren Eigenschaften zu modellieren. Dadurch kann auch dieses Tool nicht als Grundlage für die Arbeit genommen werden. [5]

2.2 Node Editor

Für die Entwicklung des Node Editors wird eine Möglichkeit benötigt spezifische grafische Elemente anzeigen zu können. Herkömmliche HTML-Elemente sind unvorteilhaft dafür zu benutzen, weil es ihnen an Flexibilität und Interagierbarkeit fehlt. Da der Benutzer in der Lage sein soll einzelne Nodes flexibel im Raum verschieben, konfigurieren und verbinden zu können, wird eine Zeichenfläche benötigt z.B. durch ein Canvas- oder SVG-Element.

Canvas ist ein HTML-Element, das verwendet wird, um Grafiken auf einer Webseite zu zeichnen. Es handelt sich um eine Bitmap auf der mittels einer grafischen Anwendungsprogrammierschnittstelle (API) pixelweise gezeichnet werden kann.

SVG-Elemente werden auch dazu verwendet Grafiken auf einer Zeichenfläche anzuzeigen, jedoch werden Objekte mit Vektoren dargestellt. Das HTML <svg>-Element ist ein Container, der die Nutzung von SVG-Grafiken auf Webseiten unterstützt.

Vorteile Nachteile HTML Canvas / SVG Warum Entscheidung für SVG?

2.2.1 Node-Editoren für JavaScript

Es gibt bereits einige offene Bibliotheken für JavaScript, die die Funktionalität eines Node-Editors bieten. Im Folgenden werden einige dieser Bibliotheken inklusive deren Vor- und Nachteile vorgestellt.

litegraph.js

litegraph.js ist eine Canvas-basierte Bibliothek. Damit bietet sie eine hohe Leistung. Es ist allerdings aufwändig, eigene Steuerelemente einzufügen, da diese selber gezeichnet werden müssen. Auch ist die Dokumentation spärlich.

ThreeNodes.js

Die Bibliothek *ThreeNodes.js* ist auf die Bearbeitung von WebGL-Szenen ausgelegt.

vue-blocks

vue-blocks ist eine rudimentäre Umsetzung eines Node-Editors mit VueJS. Allerdings sind die Möglichkeiten der Bibliothek sehr beschränkt; es gibt beispielsweise keine Möglichkeit, eigene Steuerelemente in den Nodes anzuzeigen.

Nodes

Nodes ist eine mit VueJS entwickelte Bibliothek, die auf dem HTML5-Canvas aufbaut. Der Autor schreibt allerdings selber, dass die Bibliothek mehr als Experiment gedacht war und man lieber eine eigene Bibliothek entwickeln solle anstatt diese Bibliothek zu verwenden.

linker

Linker

3 Grundlagen des Machine Learning

- Warum braucht man Trainingsdaten?
- Wie sehen Trainingsdaten aus?
- Was für Möglichkeiten gibt es, Trainingsdaten zu bekommen?

> „A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.“ **TODO: Quelle**

Das Ziel von maschinellem Lernen ist es, einen Algorithmus zu entwickeln, der anhand von Eingabedaten Ausgabedaten erzeugt. TODO

Die Vorgehensweise beim Trainieren kann hauptsächlich in zwei Gruppen eingeteilt werden: *Supervised Learning* und *Unsupervised Learning*. Beim Supervised Learning benötigt man in der Trainingsphase Daten, die sowohl aus den Eingabedaten, als auch aus den erwarteten Ausgabedaten bestehen. Der Algorithmus wird während des Trainings so angepasst, dass die Eingabedaten die erwarteten Ausgabedaten erzeugen. Häufig wird Supervised Learning für *Klassifikation* oder *Regression* eingesetzt. Bei der Klassifikation wird den Eingabedaten eine von mehreren vordefinierten Klassen zugewiesen. Bei der Regression dagegen wird

Sowohl für Supervised, als auch für Unsupervised Learning werden Trainingsdaten benötigt. Diese Trainingsdaten bestehen aus sogenannten Features. Die Features kann man sich wie Spalten einer Tabelle vorstellen.

Räume	Fläche (m^2)	Grundstücksfläche (m^2)	Baujahr	Preis (Euro)
8	240	633	1976	679000
5	130	504	1978	320000
6	175	247	2020	595000

Tabelle 3.1: Beispiel für Trainingsdaten einer Regression

4 Technologie Evaluation

In diesem Kapitel werden die Entscheidungen im Hinblick auf die verwendeten Technologien vorgestellt.

4.1 Plattform

Da die verwendete Plattform Basis für alle weiteren zu treffenden Entscheidungen ist, muss diese zuerst gewählt werden. Für die zu entwickelnde Applikation gab es dabei folgende technische Anforderungen an die Plattform:

- Grafische Benutzeroberfläche
- Effizientes Berechnen des Datensatzes aus dem Modell
- Möglichkeit zur Visualisierung von Daten

Die Applikation soll auf jeden Fall auf Windows laufen können, andere Betriebssysteme sollen nach Möglichkeit aber auch unterstützt werden. Aus diesen Anforderungen, sowie der Erfahrung der Teammitglieder, ergaben sich folgende Möglichkeiten:

- C#
- Java
- Web (HTML, CSS, JavaScript/TypeScript, Web Assembly)

TODO: Nur optimiert für Chrome

4.2 JavaScript / TypeScript

4.3 Chrome vs Electron

Electron ist eine Bibliothek für die Entwicklung von Cross-Platform-Applikationen unter der Nutzung von Webtechnologien (`https://electronjs.org/docs/tutorial/about`). Mit Electron können Web-Applikationen entwickelt werden, die als Desktop-Applikation

auf Windows, Linux oder MacOS laufen und Zugriff auf Betriebssystemfunktionalitäten wie zum Beispiel das Dateisystem haben.

TODO: Warum haben wir nicht Electron genommen?

5 Graph-Editor

Eine der Anforderungen war es, das Modell visuell bearbeiten zu können. Dafür war es notwendig, einen Graph-Editor zu entwickeln. Ein solcher Editor erlaubt es, Knoten hinzuzufügen bzw. zu entfernen und sie miteinander zu verbinden. Der Editor wurde als separate Bibliothek unter dem Namen *BaklavaJS* entwickelt und ist somit auch für andere Projekte verwendbar.

Der Graph besteht aus Knoten und Kanten. Ein Knoten ist dabei wie eine mathematische Funktion: Er führt einen Algorithmus auf die Eingangsdaten aus und gibt die erzeugten Ausgangsdaten aus. Im Gegensatz zu einem „herkömmlichen“ Graphen kann ein Knoten mehrere Kanten zu einem anderen Knoten haben.

Visuell wird ein Knoten in BaklavaJS folgendermaßen dargestellt:

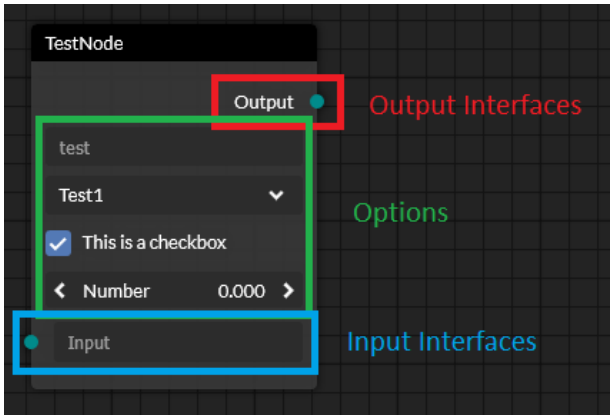

Abbildung 5.1: Aufbau eines Knotens in BaklavaJS

Jeder Knoten besteht aus drei Teilen:

- **Input Interfaces**: Die Eingangsschnittstellen eines Knotens werden benutzt, um Daten von anderen Knoten an diesen Knoten zu transferieren. Ist kein anderer Knoten verbunden, kann der Wert mittels eines Steuerelements auch direkt am Knoten eingestellt werden.
- **Options**: Hier können Werte eingestellt werden, die der Knoten für die Berechnung braucht, die aber beispielsweise zu komplex sind, um als Daten von anderen Knoten über Eingangsschnittstellen zu kommen.

- **Output Interfaces**: Die Ausgangsschnittstellen stellen das Ergebnis bereit, damit es von anderen Knoten benutzt werden kann.

5.0.1 Ausführungsreihenfolge des Graphen

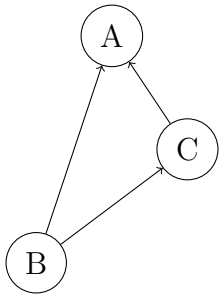

Abbildung 5.2: Beispielgraph

In Abbildung 5.2 ist ein Beispielgraph mit Knoten und Kanten zu sehen. Der Graph kann als Abhängigkeitsgraph interpretiert werden: Um den Knoten A auszuführen, müssen vorher die Knoten B und C ausgeführt werden, damit die Ergebnisse dieser Knoten verfügbar sind. Auch muss der Knoten B vor dem Knoten C ausgeführt werden. B hat keine Abhängigkeiten. Somit ergibt sich als Ausführungsreihenfolge $B \rightarrow C \rightarrow A$.

Die Ausführungsreihenfolge der Knoten muss folgende Bedingungen erfüllen:

- Jeder Knoten wird genau einmal ausgeführt
- Ein Knoten kann erst ausgeführt werden, wenn alle Knoten, die Kanten zu ihm haben, ausgeführt wurden

TODO: Topological Sorting

Topologische Sortierung kann nur auf azyklischen, gerichteten Graphen durchgeführt werden. Anhand des folgenden, zyklischen Graphen ist das leicht erkennbar:

Abbildung 5.3: Zyklischer Graph

Knoten B hat Knoten A als Abhängigkeit. Somit muss Knoten B vor Knoten A ausgeführt werden. Allerdings hat B den Knoten A als Abhängigkeit. Es müsste somit der jeweils andere Knoten zuerst berechnet werden, was nicht möglich ist. Aus diesem Grund darf der Graph keine Zyklen enthalten.

TODO: JSPerf mit Kahn's algorithm vs custom algorithm

Mit folgendem Algorithmus kann die Ausführungsreihenfolge bestimmt werden:

1. Adjazenzliste erstellen
2. Baum aufbauen mit Zykluserkennung
3. Breitensuche um die Ausführungsreihenfolge zu bestimmen

Algorithm 1 Baum aufbauen mit Zykluserkennung

```
1:  function findDescendants(treeNode, ancestors, adjacency)
2:      for all c in treeNode.children do
3:          if c in ancestors then
4:              Cycle detected
5:          end if
6:          ancestors.push(c)
7:          c.children = findChildren(c)
8:          findDescendants(c, ancestors, adjacency)
9:          ancestors.pop()
10:     end for
11: end function
```

Algorithm 2 Breitensuche um die Ausführungsreihenfolge zu bestimmen

```
1:  queue ← new Queue()
2:  stack ← new Stack()
3:  queue.push(root)
4:  while not queue.isEmpty() do
5:      current ← queue.dequeue()
6:      for all c in current.children do
7:          stack.push(c)
8:          queue.enqueue(c)
9:      end for
10: end while
11: calculationOrder ← new List()
12: while not stack.isEmpty() do
13:     current ← stack.pop()
14:     if not calculationOrder.contains(current) then
15:         calculationOrder.append(current)
16:     end if
17: end while
```

Am Beispiel des Graphen in Abbildung 5.2 sieht der Algorithmus folgendermaßen aus:

TODO:

5.0.2 Grafische Umsetzung

VueJS

VueJS ist ein JavaScript Framework zum Erstellen von Browser-Frontends. Ein wichtiges Konzept in Vue sind *Komponenten*. Komponenten sind kleine, eigenständige und wiederverwendbare UI-Elemente [6].

In Anlehnung an das Model-View-Viewmodel-Pattern ist Vue *reaktiv* gestaltet und bietet Datenbindung [7]. Reaktiv bedeutet, dass Daten im sogenannten *ViewModel* geändert werden können und diese Änderungen direkt in der *View* angezeigt werden. **TODO: Grafik zur Verdeutlichung**

Diese Art der Wiederverwendbarkeit ist essentiell, um die Knoten und Kanten zu zeichnen. Besonders hilfreich ist hierbei die `v-for`-Direktive. Sie erlaubt es, eine Liste von Komponenten zu rendern. Dabei werden Änderungen in der Liste über Datenbindung direkt in der UI übernommen. **TODO: Mehr Erklärung**

- Erklärung VueJS
- Wie werden die Kanten gezeichnet?
- Wie werden NodeOptions gezeichnet?
- Wie wird Zoom / Panning umgesetzt?

5.0.3 Plugin-System und Event-System

5.0.4 BaklavaJS Pakete

- **Core**: Dieses Paket enthält alle notwendigen Klassen und Funktionen, um Knoten hinzuzufügen **TODO: More**
- **Engine**: Erlaubt es, die Knoten in dem Graphen auszuführen und sorgt dafür, dass Daten über die Kanten übertragen werden

- **Interface-Types**: Fügt Typen zu Knoten-Schnittstellen hinzu. Standardmäßig können nur zwei Schnittstellen mit gleichem Typ miteinander verbunden werden. Es ist aber möglich, Ausnahmen anzugeben, sodass auch Schnittstellen unterschiedlichen Typs miteinander verbunden werden können. Dabei kann auch eine Konvertierungsfunktion mit angegeben werden. So kann zum Beispiel eine Number-Schnittstelle mit einer String-Schnittstelle verbunden werden. Die Konvertierungsfunktion wandelt dabei die Number in einen String um.
- **Vue-Renderer**: Der Renderer bietet eine Oberfläche für den Editor, welcher im Core implementiert ist.
- **Vue-Options**: Das Vue-Options-Plugin fügt vorgefertigte Knoten-Optionen zum Renderer hinzu.

6 Implementierung der Knoten

6.1 Knoten-Arten

Der Kern der Applikation bilden die verschiedenen Knoten: Mit ihnen kann das Modell aufgebaut werden. Dabei ist es wichtig, dass die Knoten die verschiedenen Anwendungsfälle abdecken.

Um herauszufinden, welche Knoten-Arten benötigt werden, wurden mehrere Beispiel-Modelle durchgespielt und evaluiert, mit welchen Knoten diese sich am besten umsetzen lassen.

Wohnungsbeispiel:

Algorithm 3 Beispielmodell Wohnungspreise

1: Fläche = ZufallGauss($\mu = 80$, $\sigma = 20$)
2: a = Mathematik(Division, Fläche, 40)
3: b = ZufallDiskret($-1 : 10\%, 0 : 80\%, 1 : 10\%$)
4: c = Mathematik(Addition, a, b)
5: Räume = Mathematik(Maximum, c, 1)
6: d = Fläche $\cdot$ 2500 + 250 $\cdot$ Räume
7: Preis = ZufallProzentual(d, 5%)

Die Knoten lassen sich in folgende Kategorien unterteilen:

- **Werteknoten** können benutzt werden, um den gleichen Wert an verschiedene andere Knoten weiterzugeben. Es gibt sie für die Datentypen *Boolean*, *Number* und *String*. Zusätzlich gibt es den Index-Knoten. Dieser gibt den aktuellen Index innerhalb des Berechnungsprozesses aus.
- **Zufallsknoten**
  - **Gleichverteilung** mit einstellbarem Minimum und Maximum
  - **Normalverteilung** mit einstellbarem Mittelwert μ und Standardabweichung σ
  - **Exponentialverteilung** mit einstellbarem λ

  - **Anpassbare Verteilung**: Bei diesem Knoten kann die Wahrscheinlichkeitsdichtefunktion über einen grafischen Editor eingestellt werden. Die Wahrscheinlichkeitsdichtefunktion kann sowohl diskret als auch kontinuierlich sein.
  - **Prozentuale Abweichung**: Dieser Knoten nimmt einen Wert und addiert einen zufälligen Wert zwischen $\pm\, inputValue \cdot \frac{percentage}{100}$

- **Berechnungsknoten**
  - **Mathematik**
  - **Funktion**
  - **Boolean**
- **Bedingungsknoten**
- **Ausgabeknoten**

7 Datengenerierung

7.1 Random Sampling

Mit Random Sampling ist eine Zufallsstichprobe gemeint bei der eine Stichprobe aus einer Grundgesamtheit X mit n Elementen gezogen wird. Seien die Elemente der Grundgesamheit $x_1, x_2, \ldots, x_n - 1, x_n$, dann unterliegt die Auswahl eines Elements $x \in M$ einem Auswahlverfahren, das angibt mit welcher Auftrittswahrscheinlichkeit $P(x)$ ein Element in die Stichprobe N gelangen kann. Es gelten die Bedingungen:

- $P(x) > 0$ (Positivität)
- $\sum_{i=1}^{n} P(x_i) = 1$ (Vollständigkeit)

Bei einer einfachen Zufallsstichprobe handelt es sich um die Auswahl einer Teilmenge einer statistischen Population, bei der jedes Element die gleiche Wahrscheinlichkeit $P(x) = \frac{1}{|M|} = \frac{1}{n}$ hat, ausgewählt zu werden. Zudem darf eine Ziehung aus einer Gesamtmenge nicht die nachfolgenden Ziehungen beeinflussen, weil sie unabhängig voneinander erfolgen müssen. Übertragen auf das typische Urnenexperiment der Stochastik, kommt es bei einer einfachen Zufallsstichprobe zu einer Ziehung mit Zurücklegen.

Für die Generation von Trainingsdaten ist das Random Sampling von großer Bedeutung, weil nur durch Einbeziehung einer Zufallskomponente neue Datensätze generiert werden können. Sollen beispielsweise zufällig ortspezfische Personendaten generiert werden, könnte die Herausforderung bestehen Daten von hunderten Personen aus nur vier Wohnorten zu generieren. Jeder Person soll dabei ein Wohnort zugewiesen werden, der zufällig bestimmt wird. Ist die Wahrscheinlichkeit für die Auswahl eines Wohnorts gleich groß, so ergibt sich eine einfache Zufallsstichprobe. Oft genügt eine Gleichverteilung jedoch nicht, um eine Zufallsvariable zu modellieren, weil in der Realität andere Faktoren die Wahrscheinlichkeitsverteilung beeinflussen. Will man bezogen auf die Generation von Personendaten beispielsweise Größen generieren, wird eine Normal-, auch Gauß- oder Glockenverteilung benötigt. Es kann auch sein, dass die Wahrscheinlichkeitsverteilung so spezifisch definiert ist, dass sie nicht mit einer einzigen Funktion dargestellt werden kann.

In den folgenden Kapiteln soll auf die genannten Fälle und deren Umsetzung eingegangen werden.

7.1.1 Generation von gleichverteilten Zufallszahlen

Für die einfache Zufallsstichprobe kann das Problem auf die Frage reduziert werden, wie eine gleichverteilte Zufallszahl in dem Intervall $I = [0; 1]$ generiert werden kann. Diese Zufallszahl kann dann auf ein beliebiges Intervall transformiert werden. Sei x_1 eine gleichverteilte Zufallszahl im Intervall $I_1 = [0; 1]$ und seien a, b Grenzen des Intervalls $I_2 = [a; b]$, so ist die Zufallszahl x_2 im intervall I_2 bestimmt durch $x_2 = x_1 \cdot b + a$. Durch das Runden auf die Einerstelle, können so auch diskrete Werte resultieren.

Um gleichverteilte Zufallszahlen generieren zu können wird ein Pseudorandom Number Generator (PRNG) verwendet. Ein PRNG oder auch Zufallszahlengenerator stellt einen Algorithmus dar, der eine Sequenz von Zufallszahlen generiert. Es handelt sich bei den generierten Zahlen um Pseudo-Zufallszahlen, weil das zugrundeliegende Verfahren deterministisch implementiert ist. Bei gleichen Eingangsparametern ist auch immer das gleiche Ergebnis zu erwarten. Der essentielle Parameter eines PRNG's ist der sogenannte Seed. Der Seed ist ein Startwert, der bei dem allerersten Aufruf des PRNG's initial gesetzt wird. Aufbauend auf diesem Seed können alle weiteren Zufallszahlen generiert werden. Jeder generierte Wert wird als neuer Startwert für das Verfahren verwendet. Da der mögliche Wertebereich durch eine festgelegte Speichergröße begrenzt ist, muss zwangsläufig irgendwann eine Zahl generiert werden, die bereits in der Sequenz aufgetaucht ist. Weil sich die generierte Sequenz wiederholt, ist der Zufallszahlengenerator periodisch. Simple lineare Kongruenzgeneratoren durchlaufen den Wertebereich im besten Fall einmal pro Periode, weil sie mit nur einem Seed als initialen Wert arbeiten. Andere PRNG's wie zum Beispiel der Mersenne-Twister arbeiten mit mehreren Seeds, wodurch eine erheblich größere Periodenlänge erreicht werden kann.

Ein Algorithmus kann Pseudo-Zufallszahlen generieren, die sich nur in einer bestimmten Anwendung nicht von echten Zufallszahlen unterscheiden lassen. Manche Generatoren sind deshalb für bestimmte Anwendungen gut geeignet und andere wiederum nicht. Müssen beispielsweise lange Sequenzen von Zufallszahlen generiert werden, ist ein linearer Kongruenzgenerator nicht zu empfehlen. Hingegen wäre er bei einer performanten Generation von kurzen Sequenzen eher geeignet. Echte Zufallszahlen verhalten sich immer gleich und bieten hohe Güte. Zur Generation von echten Zufallszahlen kommen pyhsikalische Zufallszahlengeneratoren zum Einsatz, die die naturgemäße Zufälligkeit von physikalischen Prozessen zu Nütze machen. Beispiele für solche physikalischen Prozesse sind Spannungsschwankungen an einer Z-Diode, thermisches Rauschen eines Widerstands und radioaktive Zerfallsvorgänge. Logischerweise ist die Benutzung eines Seeds und damit die Reproduzierbarkeit von physikalischen Zufallszahlengeneratoren nicht gegeben. Die Möglichkeit einen Seed verwenden zu können, hat allerdings den großen Vorteil der Reproduzierbarkeit von Ergebnissen, weshalb die Verwendung von echten Zufallszahlen unpassend ist. Gerade

bei der Erstellung von komplexen Zusammenhängen innerhalb der Applikation soll ein Neustart der Generation nach kleinen Veränderungen nicht komplett neue Daten generieren. Dem Benuter soll es möglich sein benutzerdefinierte Zufallsvariablen mit einem Seed zu versehen, um gleiche Ausgangswerte erhalten zu können.

Vorteile von Seeds:

- Portabilität: Durch Speicherung der Seed-Werte bei einem Projektexport, werden bei unverändertem Projekt immer gleiche Trainingsdaten generiert, sodass das Projekt geteilt und wiederverwendet werden kann.
- Debugging: Werden unerwartete Trainingsdaten generiert, können mithilfe von Seeds einzelne Änderungen vorgenommen werden, sodass die neuen Daten mit den letzten abgeglichen werden können

- Alte Library vs. neue Library - Benutzten PRNG vorstellen und Auswahlbegründung -> Güte

7.1.2 Generation von nicht-gleichverteilten Zufallszahlen

Warum brauchen wir auch nicht-verteilte Zufallswerte mit Beispielen? - Normalverteilung (Größe), Exponentialverteilung (Alter)

7.1.3 Generation von Zufallszahlen einer benutzerdefinierten Wahrscheinlichkeitsverteilung

Was ist damit gemeint und warum brauchen wir benutzerdefinierte Zufallszahlen? Wie ist das Problem bei benutzerdefinierten Wahrscheinlichkeitsverteilung und wie kann dies am einfachsten gelöst werden? Der CustomRandom-Node teilt sich in drei verschiedene Komponenten: 1. UI, 2. Interpolation, 3. Berechnung

Benutzerdefinierte Erstellung einer Wahrscheinlichkeitsverteilung

–> Anzeige und benutzerdefinierte Erstellung einer Wahrscheinlichkeitsverteilung im Graph-Editor - Chart.js, Canvas

Interpolation der erstellten Wahrscheinlichkeitsverteilung

- Chart.js bietet out-of-the-box bereits Interpolationsmöglichkeiten an, jedoch gibt es keine geeignete Schnittstelle, um auf die Interpolationsdaten zuzugreifen, daher muss eigener Algorithmus implementiert werden

Bestimmung der inversen kummulativen Wahrscheinlichkeitsverteilung

–> Bestimmung der inversen kummulativen Wahrscheinlichkeitsverteilung - Bestimmung der Integrierten Funktion - Bestimmung der Inversen - Sei x eine gleichverteilte reelle Zahl zwischen 0 und 1, so ist die custom-verteile Zahl bestimmt durch cdf-1(xmaxcdf)

7.2 Web Worker

JavaScript ist im Browser single-threaded. Das bedeutet, dass die Webseite „einfriert“, wenn eine Berechnung lange benötigt. Es kann also keine Interaktion mit der Seite mehr vorgenommen werden [8]. Solche Interaktionen sind beispielsweise das Klicken auf Buttons oder das Scrollen der Seite.

Für diese Arbeit muss jedoch eine große Anzahl von Datensätzen generiert werden. Dies soll möglichst schnell geschehen; gleichzeitig soll die Applikation trotzdem bedienbar bleiben und einen Fortschrittsbalken anzeigen.

Um diese Anforderungen umzusetzen, werden sogenannte *Web Worker* verwendet. Web Worker werden in separaten Hintegrundthreads ausgeführt und blockieren dadurch nicht den Hauptthread [9]. Da JavaScript nicht multithreading-fähig ist, besitzen der Hauptthread und die Web Worker keinen geteilten Adressraum. Es können also keine Objektreferenzen aus dem Hauptthread im Web Worker oder andersherum verwendet werden [9].

Die Kommunikation zwischen Hauptthread und Web Workern funktioniert über Nachrichten. Jeder Nachricht können Daten beigefügt werden. Weil keine Referenzen verwendet werden können, werden die Daten kopiert [9].

TODO: Structured Cloning / JSON Um komplexe Datentypen zwischen Workern und dem Hauptthread zu übertragen, gibt es mehrere Möglichkeiten:

- **Serialisierung über JSON**: Über die Standardfunktionen `JSON.stringify()` und `JSON.parse()` können komplexe Datentypen serialisiert beziehungsweise deserialisiert werden. So kann beispielsweise ein Objekt im Worker serialisiert werden, der entstandene String über `postMessage()` an den Hauptthread übertragen werden und

das Objekt dort deserialisiert werden. Mit dieser Methode können keine zyklischen Objekte (also Objekte, die eine Referenz auf sich selber enthalten) serialisiert werden [10]; dies stellt aber im Kontext dieser Arbeit kein Problem dar.

- **Structured Cloning**: Für die Übergabe von Objekten von und zu Workern wurde der Structured Cloning Algorithmus entwickelt. Dieser klont die Objekte und unterstützt dabei auch zyklische Referenzen [11].
- **Transferables**:

Auch wenn Transferables in der Theorie die schnellste Art sein sollten, Daten auszutauschen, zeigt sich in der Praxis, dass es stark auf die Art der Daten ankommt, welche Methode die schnellste ist. Im Kontext dieser Arbeit **TODO: Wir haben hier Objekte, werden nicht unterstützt, nur ArrayBuffer** https://www.joji.me/en-us/blog/performance-issue-of-using-massive-transferable-objects-in-web-worker https://nolanlawson.com/2016/02/29/high-performance-web-worker-messages/

TODO: Performanzmessung z. B. mit https://nolanlawson.github.io/webworker-postmessage

TODO: Umsetzung mit Codebeispielen

Idealerweise sollten die Daten parallel zur Generierung bereits in die Ausgabedatei geschrieben werden. Damit kann die Arbeitsspeicherauslastung gering gehalten werden, was besonders auf Geräten mit kleinem Arbeitsspeicher, wie zum Beispiel günstigen Laptops oder bei Mobilgeräten wichtig ist.

Allerdings ist mit JavaScript in Browsern aus Sicherheitsgründen kein direkter Zugriff auf das Dateisystem möglich.

- Worker / Multithreading
- Gefailter Ansatz mit Streamsaving

8 Ergebnisanalyse

8.1 Unit Tests

- Mocha & Chai
- Aufbau eines Tests
- Beispieltests

8.2 Usability-Tests

Beim Entwickeln eines Produkts besteht die Gefahr, dass die Entwickler die Komplexität und mögliche Schwachstellen im Bereich der User Experience übersehen [12]. Aus diesem Grund werden Usability-Tests durchgeführt. Diese sollen sicherstellen, dass das Produkt von den Anwendern möglichst einfach und effizient genutzt werden kann.

Laut Moser ist der Ablauf eines Usability-Tests in sechs Phasen gegliedert [13]:

1. Ziel und Zweck festlegen
2. Untersuchungsdesign entwerfen
3. Teilnehmer rekrutieren
4. Evaluation vorbereiten
5. Evaluation durchführen
6. Resultate auswerten

9 Fazit

Insg. 5 Seiten

10 Das erste Kapitel

Erste Erwähnung eines Akronyms wird als Fußnote angezeigt. Jede weitere wird nur verlinkt: Affero GNU General Public License (AGPL). [14]

Verweise auf das Glossar: Glossareintrag, Glossareinträge

Nur erwähnte Literaturverweise werden auch im Literaturverzeichnis gedruckt: [15], [16]

Abbildung 10.1: aus [17]

Looking for the one superhero comic you just have to read. Following the antics and adventures of May Mayday Parker, this Spider-book has everything you could want in a comic–action, laughs, mystery and someone in a Spidey suit. Collects Alias #1-28, What If. Jessica Jones had Joined the Avengers. In her inaugural arc, Jessicas life immediately becomes expendable when she uncovers the potentially explosive secret of one heros true identity. In her inaugural arc, Jessicas life immediately becomes expendable when she uncovers the potentially explosive secret of one heros true identity.

Once upon a time, Jessica Jones was a costumed super-hero, just not a very good one. First, a story where Wolverine and Hulk come together, and then Captain America and Cable meet up. In a city of Marvels, Jessica Jones never found her niche. The classic adventures of Spider-Man from the early days up until the 90s. Looking for the one superhero comic you just have to read. In her inaugural arc, Jessicas life immediately becomes expendable when she uncovers the potentially explosive secret of one heros true identity.

11 Beispiel Code-schnipsel einbinden

```
1 public class HelloWorld {
2   public static void main (String[] args) {
3     // Ausgabe Hello World!
4     System.out.println("Hello World!");
5   }
6 }
```

Listing 11.1: Code-Beispiel

```
1 def quicksort(liste):
2 if len(liste) <= 1:
3   return liste
4 pivotelement = liste.pop()
5 links = [element for element in liste if element < pivotelement]
6 rechts = [element for element in liste if element >= pivotelement]
7 return quicksort(links) + [pivotelement] + quicksort(rechts)
8 # Quelle: http://de.wikipedia.org/wiki/Python_(Programmiersprache)
```

Titel des Python-Codes

11.1 lorem ipsum

Looking for the one superhero comic you just have to read. Following the antics and adventures of May Mayday Parker, this Spider-book has everything you could want in a comic–action, laughs, mystery and someone in a Spidey suit. Collects Alias #1-28, What If. Jessica Jones had Joined the Avengers. In her inaugural arc, Jessicas life immediately becomes expendable when she uncovers the potentially explosive secret of one heros true identity.

Once upon a time, Jessica Jones was a costumed super-hero, just not a very good one. First, a story where Wolverine and Hulk come together, and then Captain America and Cable meet up. In a city of Marvels, Jessica Jones never found her niche. The classic adventures of Spider-Man from the early days up until the 90s. Looking for the one superhero comic you just have to read.

Meet all of Spideys deadly enemies, from the Green Goblin and Doctor Octopus to Venom and Carnage, plus see Peter Parker fall in love, face tragedy and triumph, and learn that

with great power comes great responsibility. In a city of Marvels, Jessica Jones never found her niche. Bitten by a radioactive spider, high school student Peter Parker gained the speed, strength and powers of a spider. Looking for the one superhero comic you just have to read. What do you get when you ask the question, What if Spider-Man had a daughter.

The classic adventures of Spider-Man from the early days up until the 90s. Amazing Spider-Man is the cornerstone of the Marvel Universe. But will each partner's combined strength be enough. Adopting the name Spider-Man, Peter hoped to start a career using his new abilities. Youve found it.

11.2 Verweis auf Code

Verweis auf den Code Listing 11.1.
und der Python-Code Listing 11.2.

Zweite Erwähnung einer Abkürzung AGPL (Erlärung wird nicht mehr angezeigt)

Literatur

[1] Rainald Menge-Sonnentag. „Kein Wunder“. In: *iX Developer 2018 - Machine Learning* (2018), S. 3.

[2] F. Pedregosa u. a. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830.

[3] scikit-learn. *Dataset loading utilities - scikit-learn 0.21.1 documentation*. Stand: 23.05.2019. URL: https://scikit-learn.org/stable/datasets/index.html#generated-datasets.

[4] David F. Nettleton. *SynthOSNdataGenerator*. 2018. URL: https://github.com/dnettlet/SynthOSNdataGenerator.

[5] Ted Dunning. *log-synth*. 2013. URL: https://github.com/tdunning/log-synth.

[6] Vue.js Community. *Composing with Components*. Stand: 23.05.2019. URL: https://vuejs.org/v2/guide/#Composing-with-Components.

[7] Vue.js Community. *The Vue Instance - Vue.js*. Stand: 23.05.2019. URL: https://vuejs.org/v2/guide/instance.html.

[8] Paul Lewis. *Optimize JavaScript Execution*. Stand: 23.05.2019. URL: https://developers.google.com/web/fundamentals/performance/rendering/optimize-javascript-execution#reduce_complexity_or_use_web_workers.

[9] Mozilla. *Web Workers API*. Stand: 23.05.2019. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API.

[10] Mozilla. *Issue with JSON.stringify() when serializing circular references*. Stand: 23.05.2019. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify#Issue_with_JSON.stringify()_when_serializing_circular_references.

[11] Mozilla. *The structured clone algorithm*. Stand: 23.05.2019. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Structured_clone_algorithm.

[12] Frank Witte. „Metriken für Usability-Tests“. In: *Metriken für das Testreporting: Analyse und Reporting für wirkungsvolles Testmanagement*. Wiesbaden: Springer Fachmedien Wiesbaden, 2018, S. 173–178. ISBN: 978-3-658-19845-9. DOI: 10.1007/978-3-658-19845-9_25. URL: https://doi.org/10.1007/978-3-658-19845-9_25.

[13] Christian Moser. „Usability Testing“. In: *User Experience Design: Mit erlebniszentrierter Softwareentwicklung zu Produkten, die begeistern*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 219–242. ISBN: 978-3-642-13363-3. DOI: 10.1007/978-3-642-13363-3_10. URL: https://doi.org/10.1007/978-3-642-13363-3_10.

[14] Free Software Foundation. *GNU AFFERO GENERAL PUBLIC LICENSE*. 2007. URL: http://www.gnu.org/licenses/agpl-3.0.de.html.

[15] Peter Baumgartner, Hartmut Häfele und Kornelia Maier-Häfele. *E-Learning Praxishandbuch : Auswahl von Lernplattformen; Marktübersicht, Funktionen, Fachbegriffe*. Innsbruck: StudienVerl., 2002. ISBN: 3-7065-1771-X.

[16] Stuard E. Dreyfus und Hubert L. Dreyfus. *A Five-Stage Model Of The Mental Activities Involved In Directed Skill Acquisition*. Techn. Ber. University of California, Berkley, 1980. URL: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA084551&Location=U2&doc=GetTRDoc.pdf.

[17] Max Mustermann. „tolles Musterthema“. Studienarbeit. Musterstadt, 2012.

Glossar

Glossareintrag

Ein Glossar beschreibt verschiedenste Dinge in kurzen Worten.

Anhang

(Beispielhafter Anhang)

A. Assignment

B. List of CD Contents

C. CD

B. List of CD Contents

- **Literature/**
  - **Citavi-Project(incl pdfs)/** ⇒ *Citavi (bibliography software) project with almost all found sources relating to this report. The PDFs linked to bibliography items therein are in the sub-directory 'CitaviFiles'*
    - bibliography.bib ⇒ *Exported Bibliography file with all sources*
    - Studienarbeit.ctv4 ⇒ *Citavi Project file*
    - **CitaviCovers/** ⇒ *Images of bibliography cover pages*
    - **CitaviFiles/** ⇒ *Cited and most other found PDF resources*
  - **eBooks/**
  - **JournalArticles/**
  - **Standards/**
  - **Websites/**
- **Presentation/**
  - presentation.pptx
  - presentation.pdf
- **Report/**
  - Aufgabenstellung.pdf
  - Studienarbeit2.pdf
  - **Latex-Files/** ⇒ *editable LaTeX files and other included files for this report*
    - **ads/** ⇒ *Front- and Backmatter*
    - **content/** ⇒ *Main part*
    - **images/** ⇒ *All used images*
    - **lang/** ⇒ *Language files for LaTeX template*