Rapid Frontend Prototyping with Deep Learning

Simon Deussen

24. August 2018

Generierung von HTML/CSS Code anhand von pixelbasierten Screenshots und -designs. Aufbauend auf dem Pix2Code Paper [4], wird diese Arbeit eine eigene Implementation mit einem anderen Framework sowie einer ausführlichereren DSL erstellen. Durch automatisches Erstellen von Frontend-Code kann diese Anwendung rapide Entwicklungszyklen realisieren.

Inhaltsverzeichnis

1 Einleitung

Viele client-basierte Anwendungen brauchen ein schönes Frontend. Dieses soll gleichzeitig funktional und übersichtlich sein, sowie die Firma durch gutes Design repräsentieren. Seit Apple & Co ist es unglaublich wichtig gutes Design im Frontend zu haben, da sich die Kunden sonst schnell eine Alternative suchen. Um nun ein großartiges Frontend zu realisieren benötigt man ein enge Kooperation von Designern und Entwicklern, da hier zwei, sich kaum überschneidende, Skillsets gebraucht werden. Im der klassichen Entwicklung sieht diese Zusammenarbeit folgendermaßen aus:
Ein Designer macht einen grafischen Entwurf, dieser wird vom Kunde abgenommen, dann geht er zu dem Entwickler, der nun zu aller erst Markup für den Content und anschließend das Design und die richtige Darstellung nach bauen muss. Für jede grafische Veränderung muss dieser Prozess wieder ausgeführt werden. Für die meisten Entwickler, ist die Markup und CSS erstellung der widrigste Part der ganzen Arbeit, da er recht zeit-aufwendig, repetetiv und langweilig ist. Es gab bisher viele Ansätze diese Arbeit zu automatisieren, zB durch Tools in dem man gleichzeitig Designen und den Markup exportieren kann. Leider sind diese Tools entweder nicht besonders gut die Designs zu erstellen oder darin den Markup zu exportieren.

Eine Abhilfe soll diese Arbeit liefern: Sie ermöglicht, dass der Designer mit seinem bevorzugten Tools das Design baut und der Entwickler mit einem Mausklick das fertige Markup bekommt. So kann sich der Entwickler vollends auf die Realisierung des Verhaltens und der Logik der Anwendung konzentrieren.

2 Ähnliche Arbeiten

Diese Arbeit basiert auf dem Pix2Code Paper von Tony Beltramelli [4] . Er war der erste der Code anhand von visuellen Input generieren kann. Anderen Ansätze wie DeepCoder [3] benötigen komplizierte DSL als Input und schränken so die Benutzbarkeit stark ein.

Visuelle Versuche mit Android-GUIs von Nguyen [5] benötigt ebenfalls unpraktische von Experten erstellte Heuristiken. Pix2Code ist so das erste Paper das einen allgemeinen Input hat, und daraus momentan drei verschiedene Targetsprachen hat. Zum einen kann es HTMl/CSS Code erstellen, zum anderen aber auch Android- und iOS-Markup. Siehe Orginal Code auf Github [2]

3 Motivation

Computergenierte Programme werden die Zukunft der Software Entwicklung sein und diesen Bereich auch grundsätzlich verändern.

Ich denke das sich die Webtechnologien auch in der Desktop Umgebung durchsetzen. Da Plattform unabhängig und sehr stark optimiert. Sehr einfach zu lernen, weit verbreitet. Zum Beispiel Electron [1] ermöglich den einfachen Einsatz durch einen eingebetten Browser.

4 Benutzte Technologien

4.1 Neuronale Netzwerke

Neuronale Netzwerke sind einfach zu benutzende Modelle, welche nicht-lineare Abhängigkeiten mit vielen latenten Variablen stochastisch abbilden können [6]. Im einfachen Sinne, sind sie gerichtete Graphen, deren Knoten oder Nodes aus ihren Inputs Werte errechnen und die an die folgenden Nodes weitergeben. Hierbei werden zwischen 3 verschiedenen Arten von Nodes unterschieden:

Input Nodes Über diese Nodes bekommt das Netzwerke die Input Parameter.

Hidden Nodes Nodes, welche das netzwerke-interne Modell repräsentieren.

Output Nodes Diese Nodes bilden die Repräsentation des Ergebisses ab.

Nachdem die Node aus den Inputs einen Wert errechnet hat, geht dieser durch eine Aktivierungsfunktion. Diese Funktion stellt den Zusammenhang zwischen dem Input und dem Aktivitätslevel der Node her. Man unterscheidet zwischen folgenden Aktivitätsfunktionen

Lineare Aktivitätsfunktion Der einfachste Fall, linearer Zusammenhang zwischen Inputs und Output.

Lineare Aktivitätsfunktion mit Schwelle Linearer Zusammenhang ab einem Schwellwert. Sehr nützlich um Rauschen herauszufiltern. Ein häufig genutzte Abhandlung davon:

ReLU Hier werden nur der positive Werte weitergeleitet: $f_x = x^+ = max(0, x)$

Binäre Schwellenfunktion Nur zwei Zustände möglich: 0 oder 1 (oder auch -1 oder 1)

Sigmoide Aktivitätsfunktion Benutzung entweder einer logistischen oder Tangens-Hyperbolicus Funktion. Diese Funktionen gehen bei sehr großen Werten gegen 1 und bei sehr negativen Werten gegen 0 (logistische Funktion) oder -1 (Tangens-Hyperbolicus Funktion). Diese Funktion bietet den Vorteil das sie das Aktivitätlevel begrenzt.

Jede der Nodes hat eine bestimmte Anzahl an Verbindungen, diese hängt von der Art der Nodes und deren Zweck ab. Wichtig ist jedoch, das jede Node mit mehreren anderen Nodes verbunden ist, dies soll heißen, den Output mehrerer Nodes als Input zu bekommen und den eigenen Output als Input für die folgenden Nodes weiterzuleiten. Die Stärke der Abhängigkeit zwischen zwei Nodes wird als Gewicht ausgedrückt. Jede Verbindung in einem Neuronalen Netzwerk hat ein Gewicht, welches mit dem Output der vorangegangen Node multipliziert wird bevor es als Input weiter verwendet wird. TODO BIAS

Das netzwerk-interne Modell wird in diesen Gewichten abgespeichert. Es repräsentiert also das Wissen, dass durch das Training entstanden ist.

Das Training eines Netzwerkes, ist das schrittweiße Anpassen der Gewichte bis es ein gutes Modell des Problems gelernt hat. Die Stärke von Neoronalen Netzen liegt darin, aus großen Mengen von Daten Gesetzmäßigkeiten oder Patterns zu erkennen. Ein einfaches Beispiel ist die Objekterkennung. Wenn ein Netzwerk Alltagsgegenstände erkennen soll, lernt es die Pixelgruppen welche ein Tisch von einem Bett unterscheiden. Damit dies funktioniert braucht man eine große Menge an Daten. Zunächst diwr das Training in zwei verschieden Arten unterteilt:

Supervised learning Innerhalb des Trainingsdatensets, hat jeder Datensatz einen vorgegeben Output Label. Zum Beispiel ein Bild von einem Auto ist auch so gekennzeichent. Nun werden so lange die Gewichte des Netzwerkes optimiert, bis ein jeweiliger Input auch den richtigen Output erzeugt.

Unupervised learning Hier hat das Trainingsdatenset keine Label. Die Gewichtsveränderungen erfolgen im Bezug zur der Ähnlichkeit von Inputs. Das soll heisen, wenn es viele verschiedene Bilder bekommt, werden Bilder mit ähnlichen Inhalten eine hohe Nähe aufweißen, ein Bild von einem PKW wird näher an dem Bild von einem LKW sein als an dem Bild von einem Apfel.

Literatur

[1] Electron. https://www.electronjs.org/.

[2] pix2code. https://www.github.com/tonybeltramelli/pix2code.

[3] Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. Deepcoder: Learning to write programs. *CoRR*, abs/1611.01989, 2016.

[4] Tony Beltramelli. pix2code: Generating code from a graphical user interface screenshot. *CoRR*, abs/1705.07962, 2017.

[5] Tuan Anh Nguyen and Christoph Csallner. Reverse engineering mobile application user interfaces with remaui (t). *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 248–259, 2015.

[6] Günter Daniel Rey and Fabian Beck. Neuronale netze - eine einführung. http://www.neuronalesnetz.de/downloads/neuronalesnetz_de.pdf.