Lernfähiger Algorithmus zur Erkennung von Artikelrelevanz

Studiengang: MAS Data Science

Die Firma Adwired AG betreibt Medienmonitoring, und macht für ihre Firmenkunden Zusammenfassungen von allem, was an einem Tag über sie in den Medien gesagt wurde. Die Redaktoren lesen pro Tag hunderte von Artikeln, und müssen bestimmen, ob diese relevant oder nicht relevant sind. Diese Unterscheidung in relevant und nicht relevant wurde in dieser Arbeit automatisiert. Die Technik, die dabei angewendet wurde, war Evolution Strategy, mit einem neuronalen Netzwerk.

Ausgangslage

Die Firma Adwired AG bietet Dienstleistungen im Bereich der Markenbeobachtung und des Medienmonitorings an. Die von verschiedenen Providern gelieferten Social-, Online- und Printmedien werden gefiltert, und zeigen den Firmenkunden, wieviel und in welcher Art über ihre Marken berichtet wurde.
Für einige Kunden wird zusätzlich noch ein täglicher Pressespiegel erstellt, welcher eine Zusammenfassung aller hochrelevanten Artikel darstellt. Die gefundenen, vorgefilterten Artikel werden dafür von einer Redaktion gesichtet, und sie erstellt täglich eine Selektion von hochrelevanten Artikeln, die in den Pressespiegel einfliessen sollen.

Es wurden in den letzten drei Semestern drei verschiedene Methoden untersucht, um die Klassifizierung in „relevant“ und „nicht relevant“ zu automatisieren:
- Apache OpenNLP
- Apache Spark
- Evolution Strategies

Diese Arbeit stellt den dritten Teil dar, und das erste Mal dass eine Reinforcement Learning Methode dafür benutzt wird.

Theorie

Wie bei anderen Deep Learning Methoden besteht das Training hier auch daraus, dass die Gewichte eines virtuellen neuronalen Netzwerks so optimiert werden, dass bestimmte Inputdaten zu bestimmten Outputdaten (hier: „relevant“ oder „nicht relevant“) führen.
Nachdem die anfänglichen Gewichte zufällig gewählt wurden, fährt der Evolution Strategy Algorithmus so fort, wie die Evolution in der Natur: es werden über mehrere hundert Iterationen Modelle erstellt, deren Parameter leicht vom Ursprungsmodell abweichen. Dann wird geprüft, wie gut diese Modelle mit den Trainingsdaten abschneiden, und die besten von ihnen werden für die nächste Generation verwendet und erzeugen erneut „Nachkommen“, sodass die Gewichte immer besser werden.

Umsetzung

Da die Firma mit Java arbeitet, wurde das Programm auch in Java geschrieben, und nicht in Python.
Die Bibliothek DL4J bietet hierzu die perfekte Schnittstelle.
Inputdaten waren ein paar zehntausende Artikel, die in den Jahren zuvor bereits von menschlichen Redaktoren klassifiziert worden waren. An diesen wurden zuerst die für Textverarbeitung üblichen Preprocessing-Schritte wurden durchgeführt: Entfernung von Stop Words, Lemmatisierung, etc. Dann wurden sie mit einer Klasse von DL4J zu Vektoren gemacht.
Der Evolution Strategy Algorithmus selbst wurde von Hand implementiert. Er verwendet ein einfaches Neuronales Netz mit einem Hidden Layer, dessen Grösse wählbar ist. Andere Hyperparameter, die gewählt werden können, sind zum Beispiel die Grösse des Lernschrittes, die Population die jede Generation erstellt wird, usw.
Per Grid Search wurden viele Kombinationen von Hyperparametern ausprobiert und über Nacht laufen lassen, bis die Geeignetsten gefunden waren.

Ergebnis

Bei einer Trainingsdauer von 109 Minuten wird eine Accuracy von 89.9% erreicht. Lässt man das Programm 12 Stunden lang trainieren, beträgt die Accuracy 96%.
Die ist zwar nicht schlecht, aber nichts im Vergleich zu OpenNLP, welches mit einem Training von 40 Minuten eine Accuracy von 94.6% erreicht, oder Spark welches in nur 4 (!) Minuten auf 92.6% kommt. Somit steht fest, dass unter den drei untersuchten Methoden Spark die beste Option ist.

Manuela Lütolf
manuela_lutolf@hotmail.com