Semantik-Extraktion für Mengen-Phrasen in naturwissenschaftlichen Papieren durch Sequence-to-sequence Modelle

Alpcan Dalga

4. März 2021

Menschen sind schon recht gut im extrahieren. Problem

Zusammenfassung

Informationen aus Papieren zu extrahieren ist für Menschen nicht so einfach, da sie nicht nur einzelne semantische Aussagen sondern auch den Kontext verstehen und zusammenfügen müssen. Dabei können weitere Schwierigkeiten auftauchen wie Zeichen mit mehreren Bedeutungen. Um den Menschen zu helfen wurden schon verschiedene Ansätze um maschinell die Texte zu vereinfachen und eindeutig zu machen. In dieser Arbeit wird ein weiterer Ansatz getestet, der mithilfe von Sequence-to-sequence Modellen Mengen-Phrasen aus naturwissenschaftlichen Papieren extrahiert. Außerdem wurden alle nötigen Schritte für die Vorbereitung der Daten genauestens erklärt sowie ein Vergleich von zwei verschiedenen Frameworks für dieses Problem durchgeführt.

Inhaltsverzeichnis

1 Einleitung

Einheiten sind für Menschen sehr wichtig. Sie helfen uns genauer zu bestimmen was die Zahlen, die wir benutzen, ausdrücken sollen wie zum Beispiel „Sekunde“ oder aber auch „Knoten“. Von diesen beiden Beispielen ist das Zweite keine SI Einheit. Das Internationales Einheitensystem [SI06] oder aber auch kurz SI ist das verbreitetste Einheitensystem für physikalische Größen. Dabei werden Einheiten ausgewählte Größen festgelegt wie beispielsweise „Meter“ für die Länge. Wie lang ein Meter ist wird dabei durch Naturkonstanten definiert. Dieses System hat den Vorteil, dass alle Länder, die dieses System gesetzlich verpflichtend verwenden, ohne große Umrechnung Informationen austauschen können. Dadurch ist eine Fehlerquelle bei Projekten, die über Ländergrenzen hinweg durchgeführt werden, entfernt. Solche Fehler können zu großen Schäden führen wie bei der Mars Climate Orbiter Sonde [MCO]. Sie ging wegen eines Einheitenfehlers im Navigationsfehlers verloren. Jedoch hat nicht jedes Land SI gesetzlich verpflichtet wie beispielsweise die USA, wo auch das angloamerikanische Maßsystem verwendet wird. Dadurch entstehen viele verschiedene Abkürzungen die nicht einzigartig sein müssen wie man am Beispiel „mN“ was für „mini Newton“ aber auch „meter Newton“ stehen kann.

Derzeit existieren schon Programme, die in naturwissenschaftlichen Papieren Mengenphrasen extrahieren, jedoch arbeiten diese mit Formeln basierend auf den Symbolen der Mengenphrasen 2.2.[Rab17] Diese Variante der Semantik Extraktion hat Vor- und Nachteile. Ein Vorteil ist die Laufzeit, die im Vergleich zu einem Neuronalem Netzwerk verschwindend gering ist. Jedoch hat es den großen Nachteil, dass diese Art von Suche limitiert ist. Sie findet nur Symbole, die sie auch erwartet. Daher die Frage:

Forschungsfrage Kann man Sematik mit Machine Learning für Mengen-Phrasen in naturwissenschaftlichen Papieren extrahieren?

Forschungbeitrag Um diese Frage zu beantworten habe ich mit einem Sequence to Sequence Modell und den Trainingsdaten von Ulrich Rabenstein die Extraktion von Semantik probiert. Dabei habe ich das Modell auf einer NVIDIA GeForce GTX 1070 trainiert. Anschließend wurde mit ungesehenen Daten das Modell getestet. Dabei waren die Ergebnisse eher gut, da trotz dem Bearbeiten auf nur einer Grafikkarte und auf nur einem kleinen Teil der Daten schon brauchbare Ergebnisse geliefert wurden.

Overview Als aller erstes wird die Herkunft der Daten und die benutzten Frameworks angesprochen (Section 2). Darauf folgend wird die Implementierung der Vorbereitung und des Sequence-to-sequence Modelles erklärt (Section 3). In Section 4 wird die Durchführung des Trainings und die Ergebnisse genauer dargestellt bevor zum Schluss ein Fazit gezogen wird (Section 5).

2 Grundlagen

In diesem Kapitel wird gezeigt, woher die Dokumente fürs Trainieren und Auswerten der Sequence-to-sequence Modelle stammen. Zudem werden weitere Informationen über die Dokumente, die beim Trainieren geholfen haben genannt und erklärt. Des Weiteren werden themenbezogene Arbeiten erläutert. Darunter auch die Masterarbeit von Ulrich Rabenstein, die sehr wichtig für diese Arbeit ist. Zum Schluss werden benötigte Frameworks kurz beschrieben.

2.1 Herkunft der Daten

Der Ursprung der Dokumente, die zum Trainieren verwendet worden sind, kommen aus dem e-print Archiv **arXiv** [ARX]. Alle diese LaTeX Dokumente kommen aus den Physik, Mathematik, Informatik, Statistik, Elektrotechnik und einigen weiteren Bereichen. Mit **LateXML** [LAX] wurden die Dokumente aus arXiv zu HTML5 Dateien umgewandelt, da das Bearbeiten von HTML5 Dateien einfacher ist als die von PDF Dateien. Der so enstandene Corpus wird auch **arXMLiv** [AXM] genannt. Außerdem sind die von Ulrich Rabensteins Programm gefundenen und identifizierten semantischen Ausdrücke als **Annotations** mit Hilfe von KAT gespeichert worden. Es sind jedoch nicht alle semantischen Ausdrücke in jeder Datei gefunden worden und gefundene Ausdrücke müssen auch nicht richtig identifiziert sein. Da jedoch die Menge an Daten, die benötigt wird zum Trainieren, derzeit nur schwer anders erreicht werden kann, werden sowohl die HTML5 Dateien als auch die Annotations zum Trainieren und Auswerten verwendet. KAT speichert die semantischen Ausdrücke in Form von Content MathML.

Content MathML ist ein Teil von **MathML**. MathML [MML] an sich ist eine Anwendung von XML um mathematische Ausdrücke sowohl strukturell als auch inhaltlich richtig darzustellen. Für die strukturelle Darstellung ist **Presentation MathML** zuständig. Dazu gehören das darstellen von Zahlen mit `mn-`, Operatoren mit `mo-`, Identifikatoren mit `mi-Tags` und deren Position mit verschiedenen anderen Tags wie beispielsweise „superscript“ modifiziert werden können. Eine Verwendung von den ersteren und einfacheren Tags ist im Beispiel 1 gezeigt, die genau so in einer Eingabedatei vorkommen kann.

Listing 1: Beispiel zu Presentation MathML "19s"

```
<mrow>
  <mn>19</mn>
  <mo>&InvisibleTimes;</mo>
  <mi>s</mi>
</mrow>
```

Das für diese Arbeit wichtigere **Content MathML** ist für die inhaltliche Darstellung verantwortlich. Im Gegensatz zum Presentation MathML wird keine infix sondern eine prefix Schreibweise verwendet. Funktionen werden dabei in einem `apply` Tag geschrieben, welche weitere Tags beinhalten kann. Ein solcher Block startet meist mit einem Operations-Tag wie `times` gefolgt von dem Argumenten der

Operation. Weitere Symbole mit ihrer Bedeutung können mit `csymbol` innerhalb des Blocks stehen. Im Beispiel 2 ist die Bedeutung vom Symbol s, also Sekunden, innerhalb des csymbol Tags. Was man auch noch im Beispiel sieht, ist, dass man Zahlen ähnlich zu Presentation MathML in `cn` Tags schachtelt.

Listing 2: Beispiel zu Content MathML aus den Trainingsdaten für "19s"

```
<apply>
  <times/>
  <cn>19</cn>
  <apply>
    <times/>
    <csymbol cd="second">s</csymbol>
  </apply>
</apply>
```

Ein gefundener semantischer Ausdruck sieht beispeilsweise in den Annotations wie folgt aus:

Listing 3: Auszug aus den Annotations für

```
<rdf:Description rdf:nodeID="KAT_26567838925">
  [...]
  <kat:annotates rdf:resource=Resource_Nodes/>
  <kat:contentmathml rdf:parseType="Literal" score="1">
    <apply>
      <times/>
      <cn>145</cn>
      <apply>
        <times/>
        <apply>
          <csymbol cd="Prefix">Prefix</csymbol>
          <csymbol cd="micro">μ</csymbol>
          <csymbol cd="meter">m</csymbol>
        </apply>
      </apply>
    </apply>
  </kat:contentmathml>
</rdf:Description>
```

Um es übersichtlicher zu machen wurden nur für diese Arbeit wichtigen Informationen aus den Annotations aufgelistet. Zudem wurde der Leserlichkeit halber das Intervall der Knoten 4, die hierdurch markiert wurden, ersetzt durch Resource_Nodes. Dieses Intervall muss noch mit Hilfe eines Url-Decoders dekodiert 5. Dekodiert sind 3 IDs zu erkennen:

- die erste ID, welche dem Abschnitt indem das Intervall ist gehört,
- die zweite ID, die dem ersten Knoten zugeordnet ist und
- die dritte ID, die das Ende des Intervalls kennzeichnet

Darauf folgend ist mindestens ein `contentmathml` Tag, das das Attribut score hat. Der Wert von score ist ein Wert zwischen 0 und 1 und gibt an, wie Wahrscheinlich

Warum könnte das passieren? (kurz im Nebensatz oder so erv

die Übersetzung ist wenn es mehrere `contentmathml` Tags für diesen markierten Bereich existieren. Alle score-Werte pro Bereich müssen zusammengerechnet nicht 1 ergeben. Innerhalb eines `contentmathml` Tags ist die Übersetzung in Content MathML geschrieben.

Listing 4: Resource_Nodes

```
<kat:annotates rdf:resource=
"http://localhost/reflastpaper.html#cse(
%2F%2F*%5B%40id%3D'id3'%5D%2C%2F%2F*%5B%40id%3D'id3.w112'
%5D%2C%2F%2F*%5B%40id%3D'id3.w114'%5D)"/>
```

Listing 5: Resource_Nodes (decodiert)

```
http://localhost/reflastpaper.html#cse(
//*[@id='id3'],//*[@id='id3.w112'],//*[@id='id3.w114'])
```

2.2 Themenbezogene Arbeiten

Die Idee Maßeinheiten zu Identifizieren wurde schon unter anderem von **Ulrich Rabenstein** [Rab17] erforscht. Mit Hilfe von regel-basierender Implementierung hat er in seiner Masterarbeit versucht bestimmten Zeichenketten Maßeinheiten zuzuordnen. Die gesuchten Zeichenketten werden katalogisiert in drei Gruppen, die komplexen semantischen Ausdrücke, die unter anderem einfache multiplikative Maßeinheiten beinhaltet, die semantische Ausdrücke mit hochgestelltem Index und semantische Ausdrücke, die ausgeschrieben sind wie beispielsweise „meter“ oder aber auch „bar“. Seine Ergebnisse hat er in einer neu erstellten Datei gespeichert.

Klarmachen, dass d

Dmitry Gorelenkov [Gor18] hat einen anderen Ansatz zur Semantik Extraktion versucht, nämlich mit einem gängigen Methoden der Mashine Learning Systemen. Er nahm Ausdrücke der Form „Zahl*Einheiten“ entgegen und versucht die dazugehörigen kontextspezifischen semantischen Bedeutungen auszugeben.

Mit der Umwandlung von Presentation MathML zu Content MathML hat sich **Akiko Aizawa** [Ngh+13] beschäftigt. Um dies zu schaffen hat sie ;) ine Support Vector Maschine (SVM) trainiert um die Identifizierer innerhalb der `mi` Tags von Presentation MathML zu klassifizieren. Dabei hat er auch den umliegenden Text mit der SVM übergeben. Der große Unterschied zu Rabenstein ist, dass dabei die Voraussetzung ist, dass die zu übersetzenden semantischen Ausdrücke in Presentation MathML geschrieben sein müssen.

2.3 Frameworks

Die Frameworks, die in dieser Arbeit verwendet wurden, sind KAT, LLamaPun, Pytorch und Tensorflow.

LLamaPun [LP] ist eine in Rust Implementierung, die in dieser Arbeit zur Vorbereitung der Daten verwendet wird. Damit wird mit LLamaPun über die Dokumente iteriert um eine Darstellung in Tokens zu bekommen.

Beschreibst du das irgendwo genaur? -> Vorverweis

Zur Vorbereitung wird außerdem **KAT** [KAT], KWARC Annotation Tool, benutzt, da die von Ulrich Rabenstein gefundenen sematischen Asudrücke damit annotiert wurden. Die Markierungen werden dafür alle in einer extra Datei aufgeschrieben. Eine Markierung besteht dabei hauptsächlich aus zwei IDs, die des Startknotens und die des Endknotens um den Bereich, der markiert wurde, klar zu definieren, und die in der Form von Content MathML geschriebenen Übersetzungen der semantischen Ausdrücke.

Pytorch [PT1] als auch **Tensorflow** [TF1] sind sehr gut dokumentierte und zwei der bekanntesten Bibliotheken für „Deep Learning“ sind. Ein sehr berühmtes Beispiel für den Einsatz von Tensorflow ist AlphaGoZero [AGZ]. Mit beiden Frameworks ist das Implementieren der Sequence-to-sequence Modelle vereinfacht worden. Genaueres zu Sequence-to-sequence Modelle findet man im Kapitel 3.2.

3 Architektur

Es wurden als Programmiersprache Rust und Python verwendet, da LLamaPun in Rust und Tensorflow und Pytorch in Python programmiert worden sind. Die Architektur kann man grob in 3 Teile aufteilen:

- die Vorbereitung der Daten in Rust, um aus den Dokumenten die einzelnen Wörter und möglichen semantischen Ausdrücke zu entnehmen,
- die Vorbereitung der Daten in Python, um die möglichen Ausdrücke zu spezifizieren und die Daten in der Form aufzuschreiben, in den das Sequence-to-Sequence Modell es benötigt und
- das Sequence-to-Sequence Modell an sich zum Trainieren und übersetzen der Daten

3.1 Vorbereitung der Daten

In der ersten Vorbereitungsstufe wird mithilfe von LLamaPun wird der Text in Tokens gespalten. Da es aufwändiger wäre, sofort zwischen Formeln und Wörtern zu unterscheiden, weil man dafür jedes Wort genau untersuchen müsste, wird erst zwischen **möglichen Formeln**, im folgendem immer **M-Formeln** genannt, und **Wörter** unterschieden. (wie z.B. ...)

Wörter werden in die Grundform gebracht während des Erstellens der Token. Eine Umwandlung in die Grundform wird von LLamapun während des Erstellens der Token durchgeführt. Dies hilft uns die Wortmenge, unter den wir unterscheiden müssen, zu verringern. "Stemming

M-Formeln werden beim Token generieren erkannt und als diese markiert, damit sie in den nächsten Schritten schnell von den Wörtern unterschieden werden können. Im folgenden Beispiel ist 2 m und 250 cm eine M-Formel und deshalb markiert und alle anderen Wörter unmarkierte Wörter.

Ursprungstext : Ein Auto kann 2 m oder auch 250 cm lang sein.
Tokens : ein auto können 2 m oder auch 250 cm lang sein.

Wieso man nicht sofort zwischen Formeln und Wörtern unterscheidet kann man an einem Beispiel gut erkennen.

There is a graph with 20 leafs

Hierbei ist das „a“ als M-Formel markiert worden. Dank der Unterscheidung zwischen M-Formeln und Wörtern in dieser Phase kann in der nächsten Phase Arbeit gespart werden, da nun nur noch die M-Formel genauer untersucht werden müssen

statt allen Tokens.
In der nächsten Phase werden alle M-Formeln weiterverarbeitet. Es wird erst nun klargestellt welcher der M-Formeln tatsächlich Formeln mit Einheiten oder Formelzeichen sind und welche nur falsch markierte Tokens sind. Die falsch markierten Tokens werden wie Wörter behandelt. Um zu erkennen welche der M-Formeln falsch markiert wurden, muss nachgeschaut werden, ob die ID in den Annotations steht. Da nach dieser Phase die Daten bereit für die KI sein muss, muss die Ausgabe in der folgenden Form vorliegen:

[Ursprungstext, Bedeutung]

Dadurch ist es klar, dass Wörter ohne weiteres bearbeiten schon in dieser Form sind, da ihre Bedeutung gleich ihres Ursprungstextes ist. Bei Formeln ist dies leider nicht so einfach. Hierfür muss man erst mit der ID herausfinden, wie die Bedeutungen der einzelnen Einheiten und Formelzeichen, die in der Formel auftauchen, lauten. Die Bedeutungen sind in den Annotations unter den jeweiligen IDs gespeichert. Dabei muss man sicherstellen, dass die in den Annotations markierte Stelle nicht zu allgemein gefasst ist und man dadurch Wörter auch als Formeln falsch markiert. Um dies zu gewährleisten wird im unbearbeiteten Dokument nach dem dazugehörigen math-Knoten gesucht. Dies muss man nur einmal pro Formel machen, da alle Einheiten und Formelzeichen innerhalb einer Formel im selben math-Knoten stehen und man dank dem Knoten auch jede Einheit einzeln überprüfen kann. Daraufhin wird der Formelknoten weiter aufgeteilt, sodass nicht mehr die ganze Formel sondern zusammengehörende Einheiten und Formelzeichen in einem Token sich befinden. Welche Einheiten zusammengehören wird dabei durch die Annotations bestimmt indem diese unter der selben ID gespeichert sind. Beispielsweise könnte die folgende Formel in drei Token gespalten werden wenn in den Annotations dieser Bereich mit drei verschiedenen IDs gespeichert wurde.

$$\underline{2\ \mathrm{m/s}} * \underline{0.5\ \mathrm{h}} = \underline{3600\mathrm{m}}$$

Mit dem Knoten kann man nun klarstellen was alles zur Formel gehört.
In unserem Beispiel würde das Ergebnis der Vorbereitung folgendermaßen aussehen:

[Ein, ein]; [Auto, Auto]; [können, können]; [2 m, 2 Meter]; [oder, oder];
[auch, auch]; [250 cm, 250 Centimeter]; [lang, lang]; [sein., sein.]

Nun fügt man noch ein Endsymbol in regelmäßigen Abständen ein, beispielsweise am Ende jeden Satzes, um die Eingabe in passende Portionen einzuteilen. Damit ist man mit dem grundsätzlichen Vorbereiten der Daten für das Sequence to Sequence Modell fertig.

Um das Verhältnis zwischen Sätzen mit semantischen Ausdruck und zu verbessern, wurde zudem noch mit einer vorher festgelegten Wahrscheinlichkeit Sätze ohne semantische Ausdrücke verworfen. Dabei erhofft man sich eine bessere Erkennung von Ausdrücken.

Das ist gefuehlt r

mergen?

Text ist recht kon

da resourcen limitiert sind und die Trainingsdaten s

3.2 Sequence-to-sequence Modell

Für dieses Problem reichen Neuronale Netzwerke nicht aus, da die Eingabelänge und auch die Ausgabelänge nicht nur von Datei zu Datei sondern auch von Zeile zu Zeile sich unterscheiden können. Daher wurde ein sequence to sequence Modell[TF2] verwendet (siehe auch Abbildung 1). Das Modell ist realisiert durch zwei Neuronale Netzwerke, die miteinander kommunizieren. Dabei haben die beiden Netzwerke klare Rollen.

Das erste Netzwerk, auch **Encoder** genannt, ist dafür zuständig die Eingabe zu kodieren und erste Schritte der Übersetzung anzufangen. Der Encoder nimmt den Ursprungstext der Eingabe und berechnet für alle Wörter ein Vector, der für verwandte Wörter einen ähnlichen Wert aber bei ähnlich klingenden jedoch von der Bedeutung entfernten Wörtern einen größeren Werteabstand besitzt. Beispielsweise hätten die Wörter „Auto“ und „Autogramm“ einen großen Werteabstand und „lang“ und „Länge“ beinahe den selben Wert. Das Ergebnis und der letzte „hidden layer“ des Encoders sieht dabei der Benutzer des Modelles nicht, sondern wird an den **Decoder**, das zweite Netzwerk, übergeben.

-> Grundlagen?

Der Decoder bekommt außerdem noch das Endsymbol, welches als Startzeichen für den Decoder dient und kein Teil des Eingabe Alphabetes sein darf, gefolgt von seiner eigenen Ausgabe. Das Endsymbol ist ein Symbol, dass entweder bei der Vorbereitung entfernt wurde und somit nicht mehr in der Eingabe zu finden ist oder anderweitig sichergestellt ist, dass der Text dieses Symbol nicht beinhaltet. Des weiteren erhält der Decoder jeden Token, den er im Schritt bevor bearbeitet und ausgegeben hat wieder als eigene Eingabe bis wieder das Endsymbol eingelesen wird. In der Abbildung ist dies sehr gut zu erkennen durch die gestrichelten Pfeile von den oberen braunen zu den unteren roten Knoten in der nächsten Spalte.

Ein Durchlauf mit dem kurzen Satz „Das Auto wiegt 1t.“ könnte wie folgt aussehen:

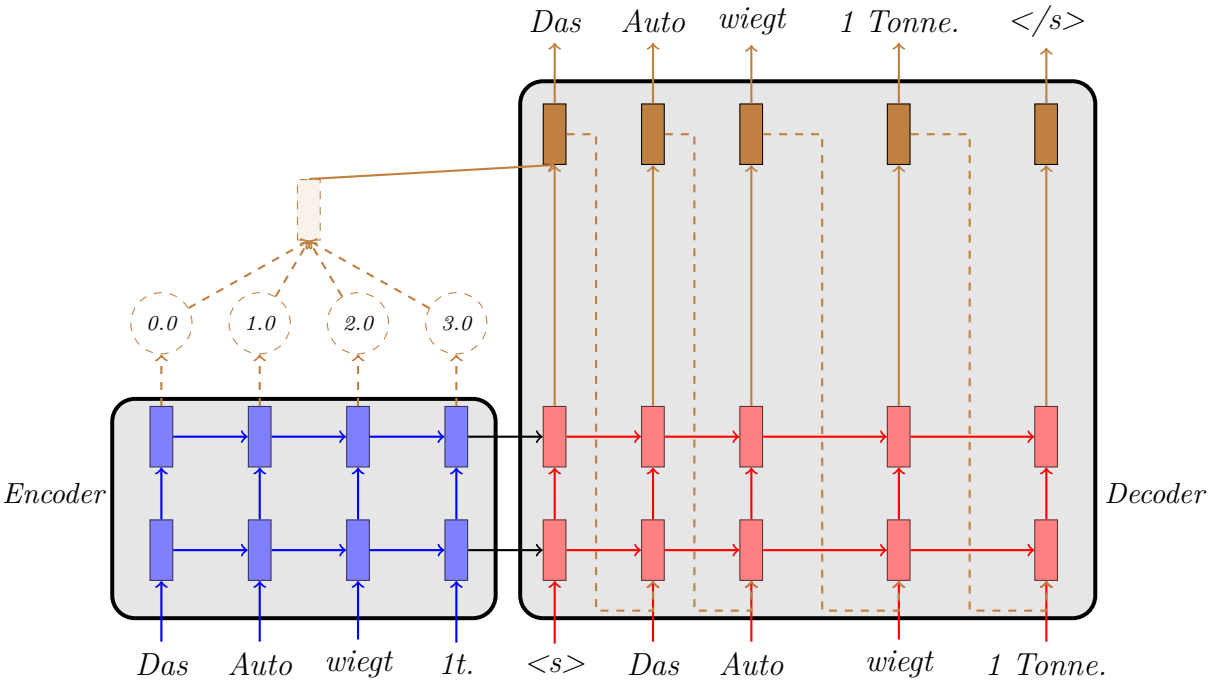

Abbildung 1: Das sequence to sequence Model [TF2]

Zu aller erst wird Wort für Wort angefangen beim „Das“ und aufgehört beim Endsymbol bearbeitet, welches in der Abbildung 1 mit „<s>“ bezeichnet wird. Das Bearbeiten passiert in den verschiedenen Schichten des Encoders, wobei hier der Wert 0.0 für das Wort „Das“, was symbolisieren soll, das es gar keinen Einfluss beim Bestimmen von „t“. Das liegt daran, dass Artikel in beinahe jedem Satz vorkommen und dadurch keine Informationen geben. Hingegen bekommt das Wort „wiegt“ den Wert 2.0, da es eher mit bestimmten Einheiten und Formelzeichen vorkommt wie beispielsweise Tonne oder anderen Gewichten. Und sehr wichtig für das neuronale Netzwerk ist die Zahl mit der Einheit „1t.“, welches übersetzt werden soll.

Die so berechneten Werte sind die Gewichte der letzten Encoder-layer, die mit in die Berechnung der Bedeutung der einzelnen Token einfließen.

Mit dem Endzeichen beginnt der Decoder die berechneten Werte und die letzte Schicht des Encoders einzulesen. Mit dem Endysmbol als erste Eingabe gibt der Decoder das Wort „Das“ aus. In den nächsten beiden Schritten passiert das gleiche, Der Decoder bekommt seine eigene Ausgabe vom letzten Schritt als Eingabe und gibt das nächste Wort aus, das wieder genauso aussieht wie die Eingabe, da sich das Wort auch nicht ändern sollte. Nun wird das Wort „wiegt“ eingelesen, was heißt, dass er nun das darauf folgende Token „1t.“ zu „1 Tonne.“ übersetzt. Mit der letzen Eingabe beendet er das übersetzen mit einem Endsymbol, welches sich vom Eingabe-Endsymbol unterscheiden darf.

Wären weitere Wörter gewesen, die nach „1t.“ im Satz stehen würden, würden sie auch in die Berechnung einfließen durch dieses Modell, da deren Werte schon im Encoder berechnet worden wären. Außerdem haben dadurch Formeln und Einheiten innerhalb eines Satzes einen Einfluss aufeinander.

Diese Herangehensweise wird genommen, da nicht nur das Wort oder die Formel an sich sondern der Kontext in dem das Wort oder die Formel steht wichtig für die Übersetzung und die Bedeutung ist. Dadurch wird versucht sich nicht nur auf Symbole in der Formel und ihre Wahrscheinlichkeit eine bestimmte Einheit zu bedeuten sondern weitere Informationen aus dem Kontext zu entnehmen um eine präzisere Übersetzung zu bekommen.
Mit dem folgendem Beispiel wird das nun besser veranschaulicht:

Man nimmt die beiden Sätze:

Um die Arbeit W zu berechnen benötigt man die Kraft F und die Strecke s.
Dadurch hat der Motor eine Leistung von 750 W.

Klingt gerade, als geh

Dabei wird allein nur durch das W nicht klar in welchem Fall die Einheit und in welchem das Formelzeichen gemeint ist. Doch wenn man allein die Wörter in unmittelbarer Nähe betrachtet wird aus einem blinden Raten eine Vorhersage, da nach der Zahl „750“ wahrscheinlich die Einheit und nach dem Wort „Arbeit“ das Formelzeichen gemeint ist. Das liegt unter anderem daran, dass nach einer Zahl meist kein Formelzeichen sondern eine Einheit oder Formel steht.

Die Vorhersage kann man dann noch weiter verbessern indem man nicht nur die nahen Wörter sondern den ganzen Satz untersucht. „Kraft“ und „Strecke“ haben eine bessere Verbindung mit dem Formelzeichen W als mit der Einheit. Im zweiten Satz hat man hingegen die Wörter „Leistung“ und „Motor“, die sehr stark hindeuten die Einheit Watt zu erwarten.

Dieses Spiel könnte man weiterführen und nicht nur einen Satz sondern ganze Absätze und Dateien als einen Input einlesen, jedoch wird dadurch irgendwann der Aufwand den man aufbringen muss um nur ein kleines Bisschen besser zu werden zu hoch und läuft Gefahr zur Überanpassung.

4 Training und Auswertung

Nun wird kurz die genauen Trainingsdaten, die verwendete Hardware und die Auslastung der Hardware für das jeweilige Framework. Daraufhin werden die Ergebnisse präsentiert. Zudem werden auch Fehlerquellen genannt und mögliche Lösungen zu den Fehlerquellen genannt.

4.1 Trainingsprozess

Beide Neuronalen Netzwerke wurden mit 110 reduzierten Dokumenten trainiert, da die verwendete Hardware nicht mehr erlaubt hat. Reduziert wurde mit einer Wahrscheinlichkeit von 80% (siehe 3.1), um so viele Dokumente wie möglich bearbeiten zu können ohne dabei in jedem Satz einen semantischen Ausdruck zu erwarten.

d.h. 80% der Sae

Die Vorbereitung der Daten auf einem Kern eines AMD Radeon 5 3600 Prozessors war im Vergleich zum eigentlichen Trainieren der Sequence-to-sequence Modelle schnell mit einer Laufzeit von ungefähr 3 Minuten. Trainiert wurde auf einer NVIDIA GeForce GTX 1070 Grafikkarte mit einer Trainingszeit von ungefähr 10 Stunden pro Sequence-to-sequence Modell. Beim Trainingsprozess lief es bei beiden Modellen beinahe identisch ab. Nur die Auslastung der Grafikkarte war verschieden. Pytorch verwendete nur 60% und Tensorflow hingegen 95% der Grafikkarte.

4.2 Ergebnisse

Da in dieser Arbeit die Daten von Ulrich Rabenstein als Trainingsdaten verwendet wurden und nicht von Hand ausgewertete Dokumente, wird angenommen, dass von Ulrich Rabenstein alle semantischen Ausdrücke gefunden wurde.

und korrekt identifiziert

Zum Testen der Modelle wurden 3 ungesehene und zufällig ausgewählte Dokumente mit insgesamt 410 semantischen Ausdrücken sowohl durch die beiden Modelle als auch per Hand evaluiert. Dabei ist das Problem aufgetreten, dass wegen den wenigen Trainingsdaten einige Wörter nicht erkannt werden konnten und somit die jeweiligen Sätze nicht übersetzt wurden. Um das Problem entgegen zu wirken wurden die Sätze aufgeteilt und nicht mehr wie im Trainingsprozess als ganze Sätze sondern Wörter und semantische Ausdrücke einzeln den Modellen übergeben. Wenn trotzdem semantische Ausdrücke nicht übersetzt wurden wegen einem nicht erkanntem Wort, wurde das als ein nicht erkanntes semantischen Ausdruck gezählt.

Das **Tensorflow** Modell hat dabei 404 von den 410 semantischen Ausdrücken erkannt. Nur 6 Ausdrücke wurden nicht erkannt. Ein semantischer Ausdruck wird dabei als erkannt gezählt wenn versucht wird den Ausdruck zu übersetzen ohne ihn dabei gleich der Eingabe zu lassen, da das nicht verändern des Ausdruckes nur bei nicht semantischen Ausdrücken passieren soll. Es wurden jedoch 521 Wörter fälschlicherweise als semantische Ausdrücke erkannt. Das führt zu einer Genauigkeit von 43.7% und eine Trefferquote von 98.5% was zu einem F-score von 60.5% führt.

Das solltest du ver

Im Vergleich dazu war das **Pytorch** Modell relativ schlecht. Es hat sowohl eine geringere Genauigkeit, nämlich 27.1%, da 293 Ausdrücke richtig und 787 fälschli-

Ueberall: besseres Wort fuer "semantischer Ausdruck"?

	Semantischer Ausdruck	Anderer Ausdruck	Gesamt
Erkannt	404	521	925
Nicht erkannt	6	-	6
Gesamt	410	521	

groessere Luecke

	Genauer Wert	Gerundet
Precision P	$P = 404/(404 + 521)$	43.7%
Recall R	$R = 404/(404 + 6)$	98.5%
F-score	$F = 2 * P * R/(P + R)$	60.5%

Tabelle 1: Ergebnisse der Auswertung des **Tensorflow** Sequence-to-sequence Modelles

	Semantischer Ausdruck	Anderer Ausdruck	Gesamt
Erkannt	293	787	1081
Nicht erkannt	117	-	117
Gesamt	410	787	

hier auch

	Genauer Wert	Gerundet
Precision P	$P = 293/(293 + 787)$	27.1%
Recall R	$R = 293/(293 + 117)$	71.5%
F-score	$F = 2 * P * R/(P + R)$	39.3%

Tabelle 2: Ergebnisse der Auswertung des **Pytorch** Sequence-to-sequence Modelles

cherweise als Ausdrücke erkannt wurden, als auch eine geringere Trefferquote von 71.5% hat. Dadurch hat das Pytorch Modell einen F-score 39.3%. Zusammengefasst stehen die Ergebnisse in der Tensorflow Tabelle 1 und in der Pytorch Tabelle 2.

Um die Genauigkeit zu verbessern sollte es ausreichen die Trainingsdaten nicht zu reduzieren oder deutlich mehr Dokumente zum Trainieren zu verwenden. Dadurch wird zwar die Trefferquote schlechter, jedoch sollte sich der F-score dadurch steigern. Leider konnte ich diese Vermutung nicht testen, da wegen meiner Hardware ich gezwungen war die Daten zu reduzieren.

kurz erwaehnen: wie viele trainingsdaten stehen zur verf

Erkannte semantische Ausdrücke wurden jedoch nicht immer richtig übersetzt. Vor allem Intervalle wie beispielsweise $5 - 10min$ und Tabellen machen große Probleme. Bei Intervallen liegt es wahrscheinlich am Bindestrich, da dieser falsch interpretiert wird. Bei Tabellen ist meiner Vermutung nach das Problem, dass beim Trainieren meist Zahlen vor Einheiten stehen, jedoch bei Tabellen das nicht der Fall sein muss wie man in der Tabelle 3 sehen kann. Dadurch werden sie als Variablen oder Eigennamen und nicht als Einheiten behandelt. Hier hilft wahrscheinlich auch der Ansatz, dass man mit mehr Dokumenten das Problem lösen kann.

Person	Länge[m]	Gewicht[kg]
Tomas	1,80	69
Oliver	1,93	75

Tabelle 3: Beispieltabelle, die zu einer falschen Übersetzung führen kann

Kannst du vielleicht sogar eine "echte" Tabelle aus einem der Dokumente nehmen?

5 Fazit

In dieser Arbeit wurde die Extraktion von semantischen Ausdrücken aus naturwissenschaftlichen Papieren untersucht, wobei der Fokus auf die Extraktion durch Sequence-to-sequence Modelle gesetzt wurde. Es wurde der grundsätzliche Aufbau von Sequence-to-sequence erklärt Modellen und wieso diese ein möglicher Weg sind mehr Informationen aus dem Kontext für die semantischen Ausdrücke zu ziehen. Zudem wurden zwei verschiedene Frameworks verwendet und verglichen (tensorflow und Pytorch). Trotz der für dieses Problem zu schwachen Hardware war das Ergebnis brauchbar.

Man müsste dieses Modell mit mehr Rechenleistung wiederholen um zu testen ob dadurch wirklich die Genauigkeit und somit der F-score steigt. Außerdem wäre es interessant herauszufinden wie gut das Modell im Vergleich zu den regelbasierten Parsern wie von Ulrich Rabenstein ist wenn man mit von Hand fehlerfrei notierten Dokumenten trainieren würde.

Literaturverzeichnis

[AGZ] Mastering the Game of Go without Human Knowledge. URL: https://discovery.ucl.ac.uk/id/eprint/10045895/1/agz_unformatted_nature.pdf (besucht am 18. 02. 2021).

[ARX] arXiv. URL: https://arxiv.org/ (besucht am 04. 03. 2021).

[AXM] SIGMathLing - Datasets and Resources. URL: https://sigmathling.kwarc.info/resources/ (besucht am 19. 02. 2021).

[Gor18] Dmitry Gorelenkov. Meaning Extraction in STEM-Documents by Machine-Learning Me B.Sc. Thesis. Sep. 2018. URL: https://gl.kwarc.info/supervision/BSc-archive/blob/master/2018/gorolenkov.pdf.

[KAT] KAT: an Annotation Tool for STEM Documents. URL: https://cermat.org/events/MathUI/15/proceedings/Lal-Kohlhase-Ginev_KAT_annotations_MathUI_15.pdf (besucht am 12. 02. 2021).

[LAX] LaTeXML A LaTeX to XML/HTML/MathML Converter. URL: https://dlmf.nist.gov/LaTeXML/ (besucht am 04. 03. 2021).

[LP] KWARC/LLamaPun. URL: https://github.com/KWARC/llamapun (besucht am 12. 02. 2021).

[MCO] Mars Climate Orbiter. URL: https://www5.in.tum.de/lehre/seminare/semsoft/unterlagen_02/erdfern/website/mco.html (besucht am 04. 03. 2021).

[MML] Mathematical Markup Language (MathML) Version 3.0 2nd Edition. URL: https://www.w3.org/TR/MathML3/Overview.html (besucht am 12. 02. 2021).

[Ngh+13] Minh-Quoc Nghiem, Giovanni Yoko Kristianto, Goran Topic und Akiko Aizawa. A hybrid approach for semantic enrichment of MathML mathematical expressio 2013. arXiv: 1305.7316 [cs.DL].

[PT1] PyTorch. URL: https://pytorch.org/ (besucht am 12. 02. 2021).

[Rab17] Ulrich Rabenstein. Meaning Extraction and Semantic Services in STEM-Documents. 2017.

[SI06] Organisation Intergouvernementale de la Convention du Mètre. The International System 2006. URL: https://www.physik.uni-wuerzburg.de/fileadmin/physik-fpraktikum/_imported/fileadmin/11999999/si_brochure_8_en.pdf (besucht am 04. 03. 2021).

[TF1] Tensorflow. URL: https://www.tensorflow.org/ (besucht am 12. 02. 2021).

[TF2] Neural machine translation with attention. URL: https://www.tensorflow.org/tutorials/text/nmt_with_attention (besucht am 22. 01. 2021).