Skript

Ich würde euch gerne heute das Thema meiner MA präsentieren.

In meiner MA geht es um die Klassifikation von deutschen Job Titlen in die Klassen der deutschen Taxonomy, der KLDB 2010, der Bundesagentur für Arbeit.

Konkreter möchte ich mir anschauen, welche Techniken sich eignen um die Texte numerisch zu so repräsentieren, so genannte vectorization oder feature extraction, dass sich verschiedene Klassifikationsalgorithmen verbessern.

Ich werde euch hiefür zunächst die bisherige Literatur vorstellen und daraus den Research Gap aufzeigen. Im nächsten Schritt erkläre ich euch die exakte Pipeline für den Klassifikationstask. Abschließend zeige ich euch die ersten Resultate mit anschließender Diskussion.

Die hauptsächlichen Herausforderungen des Tasks lassen sich in drei Punkten zusammenfassen:

* Domain spezifische Anforderungen
* Problem der Multiklassifizierung
* Die Länge der Texte für den Input

Ich möchte mich in meiner MA auf letzteres konzentrieren, weshalb ich auf die anderen beiden aus Zeitgründen jetzt nicht genau eingehen werden.

Das größte Problem bei kurzen Textsequenzen besteht darin, dass sie oft nur spärlich sind und ein geringes gemeinsames Auftreten von Wörtern gibt. Oft fehlt gemeinsamer Kontext zu erlernen von semantisch ähnlichen Wörtern. Oft liegt auch eine starke Ambiguität vor. Dazu werde ich euch nachher bei den Limitationen auch nochmal ein Beispiel zeigen.

Es gibt verschiedene Ansätze in der Literatur, um kurze Textsequenzen zu handeln. Meistens wird kritisiert, dass ein Bag of Words Ansatz problematisch ist. Stattdessen eignen sich Techniken bei denen die feature extraktion aus semantisch Beziehungen besteht.

Ein weiterer Diskussionspunkt ist die Frage nach dem Vektor per se. Es gibt dense und sparse Techniken. Für den Algorithmus gibt es kein wirklicher Konsens. Es werden vermehrt Deep Learning Ansätze angewandt, aber es gibt auch Ansätze mit traditionellen Methoden, wie SVM oder Logistic Regression.

So viel zur Literatur. Jetzt stellt sich natürlich die Frage, warum ist der Klassifikationstask relevant. Aus Literatur zeigt sich vor allem, dass es, soweit mir bekannt, keinen Versuch gibt für den deutschen Arbeitsmarkt. Das würde aber in verschiedenen Bereich downstream Tasks vereinfachen. Zusätzlich habe ich im Vergleich zu vielen anderen Arbeiten, den Vorteil einer soliden Datengrundlage, denn oft fehlen bei Job Titlen labels, was dazu führt, dass viele ML Techniken gar nicht erst angewandt werden können.

Wie möchte ich nun vorgehen? Hier sehr ihr eine Übersicht über die Pipeline. Ich werde nicht auf jeden Punkt genau eingehen, aber euch ein kurzer Überblick geben. Beginnen wir mit den Daten.

Meine Datengrundlage kommt von der Jobbörse. Die Daten enthalten zum einen den Title und eine so genannte Dkz, einen eindeutigen Schlüssel.

Zweitens die Taxonomy. Diese besteht aus 5 Leveln, hierarchisch gegliedert. Jedes Level besteht demnach aus einer unterschiedlichen Anzahl von Kategorien. Ihr seht hier im letzten Level muss sich der Klassifizierungsalgorithmus zwischen 1286 Klassen entscheiden. Die labels für die einzelnen Klassen setzen sich wie folgt zusammen. Auf Level 1 gibt es ja 10 Klassen. Diese sind von 0-9 nummeriert. Jeder diese Klassen hat Subkategorien, wie hier bspw. Die 43. Es können aber auch mehr sein. Diese wiederrum haben wieder Subkategorien und so weiter. Die Nummern ab Level 2 werden nicht nach Reihenfolge nummeriert, sondern haben oft eine Bedeutung. Die 5. Stelle z.B. sagt jede nach Nummer aus, welches Anforderungsniveau der Job hat. Eine 1 heißt beispielsweise Helferstelle.

In nächsten Schritt bilde ich meine Trainingsdaten:

Mithilfe der Dkz kann man auf einer der KLDB Klassen eindeutig schließen. Damit sieht dann der Input für Level 1 beispielsweise folgendermaßen aus:

Diese Daten habe ich dann preprocessed mit den typischen Techniken. Vielleicht hier interessant: Ich habe am meisten vorkommenden Wörter analysiert und daraus eine Liste mit Wörtern identifiziert, die ich dann entfernt habe. Erklärung anhand Berlins.

Ich kann nicht auf jede Technik genau aus zeitgründen eingehen, aber damit ihr eine Vorstellung habt möchte ich euch zwei Techniken ganz bildlich erklären. Als Baseline habe ich Count und TFIDF verwendet. Count erklären.

Im Gegensatz dazu ein dense Vektor …

PCA aus effizienzgründen im Sinne der Laufzeit der Algorithmen.