Statusreport zur Bachelorarbeit 'Semantische Beziehungen in Texten mit Word2Vec'

Ruben Müller

2015

Hinweis: Referenzangaben zum Projektplan sind kursiv und in runden Klammern im Format *(x,xx)*.

KW13

Am 25.03.2015 um 11 Uhr fand das Kick-Off Meeting*(1,01)* im Raum 321 der Hochschule der Medien mit Prof. Dr. Johannes Maucher, M.Sc. Andreas Stiegler (via Skype) und Ruben Müller statt. Hier wurde der Umfang und Inhalt der Bachelorarbeit besprochen.
Am 27.03.2015 habe ich den Projektplan(*2,01*), anhand einer Vorlage von `www.meinevorlagen.com`, angefangen und am 28.03.2015 fertig gestellt.

KW14

Am 30. und 31. März 2015 habe ich den Abstract*(2,02)* geschrieben. Um den Abstract zu schreiben, habe ich mich ein wenig in die Grundfunktionen von LaTeXeingearbeitet *(3,03)*.
Am 02. April 2015 habe ich mit der Einarbeitung in Word2Vec*(3,01)* und dem Wikipedia-Korpus*(3,02)* begonnen. Nachdem ich in PyCharm dann eine Out-of-Memory-Exception hatte, habe ich die neueste Version des Programms heruntergeladen (Version 4.0.5), die auch eine 64-Bit Version bereitstellt. Hier konnte ich dann die Größe des Speichers ausreichend vergrößern.
Mit dem Perl-Skript von `http://mattmahoney.net/dc/textdata.html`, unter Appendix A, kann der Wikipedia-Dump preprocessed werden. Allerdings ist dann der komplette Text in einer Zeile und enthält keine Satzzeichen mehr, die aber für Word2Vec gebraucht werden, da der Input hier Sätze sind. Hier ist das Perl-Skript anzupassen, was nicht viel Aufwand war. Da in der jetzt wegfallenden Regel auch sämtliche Wiki-Formatierungszeichen entfernt werden, müssen diese jetzt wieder in das Skript eingebaut werden.
Nach einigem ausprobieren und mehrfachem Skript ausführen, kam ich dann zum gewünschten Ergebnis.

KW15

Nach Ostern habe ich dann das Preprocessing der Wikipedia Korpora anhand des kleineren Korpus (die ersten 1 Milliarde Zeichen) weitergeführt. Um einen besseren Überblick zu erhalten, ob das Preprocessing das tut, was ich will, habe ich das große File in kleinere aufgesplittet um diese in Notepad++ überhaupt betrachten zu können. Als nächster Schritt kam die Aufsplittung in einzelne Sätze, da das Word2Vec Modell Sätze als Input verlangt. Die Tokenizierung habe ich mit dem NLTK Tokenizer ('tokenizers/punkt/english.pickle') gemacht.
Als ich die Sätze in ein Array einlesen wollte, um diese dann ins Modell zu geben, bekam ich immer wieder einen MemoryError. Ich habe dann versucht nur eine Teilmenge der Files einzulesen, was dann geklappt hat. Allerdings war mir unklar, was genau das Problem war, da die Fehlermeldung nicht aussagekräftig war.
Ich habe einiges ausprobiert, unter anderem die build_vocab()-Funktion des Word2Vec Modells, ich vermute aber, dass die Verwendung nicht ganz korrekt war.
Durch kleinere Denkfehler kam ich eine Weile nicht voran, bzw. durch die längere Laufzeit der Verarbeitung verging viel Zeit.

KW16

Am Anfang der Woche habe ich dann entdeckt, dass ich zwar PyCharm in der 64-bit Version nutze, allerdings die Python-Distribution nur in der 32-bit Version. So habe ich dann die 64-bit Version installiert und die benötigten Komponenten. Ich war sehr erfreut zu sehen, dass ich den kompletten kleinen Korpus ins Word2Vec-Modell laden konnte.
Dann habe ich mich an den kompletten Wikipediadump-Korpus getraut. Selbst das Preprocessing mittels Perl-Skript und meinen Skripts dauerte schon mehrere Stunden. Nachdem ich hier dann wieder einen MemoryError erhalten habe, habe ich das Preprocessing noch optimiert, dass nicht so viel im Arbeitsspeicher gehalten wird, sondern direkt in das separate File geschrieben wird.
Als ich dann die Sätze wieder in einem Array zwischenspeichern wollte, reichten mir meine 32 GB Arbeitsspeicher nicht aus und es gab wieder MemoryError. An einem Punkt, bevor das Programm abstürzte, sah es so aus, als ob einige Daten in die Auslagerungsdatei auf meiner Festplatte geschrieben wurde. Danach kam dann der Error auf. Da ich meine Auslagerungsdatei nur 4 GB groß hatte, da ich mir nie Gedanken gemacht habe, ob meine 32 GB RAM nicht ausreichen, war mein Plan die Auslagerungsdatei zu vergrößern und dann zu schauen ob dieses Mal der Speicher reicht. Aber auch mit 32 GB RAM und 24 GB Auslagerungsdatei kam wieder ein MemoryError. Da ich mein Betriebssystem nur auf einer 128 GB großen SSD habe, konnte ich die Auslagerungsdatei nicht vergrößern. Eine Überlegung war dann noch, auf einer größeren Festplatte Windows nochmals zu installieren und zu versuchen die Auslagerungsdatei so groß wie möglich zu machen. Allerdings wollte ich zuvor erst noch einmal überprüfen, ob es eine andere Möglichkeit gibt, die Sätze

in dem 18 GB großen File ins Word2Vec Modell zu bekommen.

KW17

Den ersten Versuch, den kompletten Wikipediadump ins Modell zu bekommen, habe ich mit 'gensim.models.word2vec.LineSentence()' gemacht. Dieser Versuch war dann auch erfolgreich. So konnte ich dann zum ersten Mal das Modell auf dem gesamten Wikipediadump erstellen und trainieren. Der erste Durchlauf, mit folgenden Parametern, $size = 200, window = 5, min\_count = 20, workers = 12$, dauerte dann ca. 6 Stunden (32 GB RAM, i7-3770 Quadcore bei 3,4 GHz).
Darauffolgend habe ich mir Gedanken gemacht, wie ich die großen XML-Files in einzelne Wikipedia-Artikel splitten kann, da diese ja klassifiziert werden sollen. Mittels eines SAX-Parsers habe ich dann die einzelnen Artikel in einzelne Files zwischengespeichert. Diese Files habe ich dann mittels Perl-Skript und meinen Preprocessing Skripten wieder in die gewünschte Form gebracht.

Ausblick auf die nächste Berichtsperiode:
Der komplette Wikipediadump soll mit guten Parametern im Word2Vec-Modell gelernt werden *(4,01)*. Des weiteren soll ein Domänenkorpus aus dem kompletten Wikipediadump extrahiert werden*(4,02)*. Das Verhalten der beiden Modelle soll dann später verglichen werden.

KW18

Verschiedene Parameter beim Lernen der Modelle und die Auswertung der $model.accuracy()$ Funktion mit den Testdaten aus
`http://word2vec.googlecode.com/svn/trunk/questions-words.txt`:

$size = 400, window = 8, min\_count = 25$

Trainingsdauer: 10,5h

2015-06-01 16:15:58,387 : INFO : capital-common-countries: 81.0% (410/506)
2015-06-01 16:21:38,740 : INFO : capital-world: 81.3% (1782/2192)
2015-06-01 16:21:50,073 : INFO : currency: 0.0% (0/70)
2015-06-01 16:27:57,434 : INFO : city-in-state: 50.9% (1121/2203)
2015-06-01 16:28:59,885 : INFO : family: 85.8% (326/380)
2015-06-01 16:31:12,608 : INFO : gram1-adjective-to-adverb: 16.5% (134/812)
2015-06-01 16:32:20,959 : INFO : gram2-opposite: 38.8% (163/420)
2015-06-01 16:35:39,388 : INFO : gram3-comparative: 73.4% (873/1190)
2015-06-01 16:36:56,835 : INFO : gram4-superlative: 35.3% (163/462)
2015-06-01 16:39:33,775 : INFO : gram5-present-participle: 52.3% (486/930)
2015-06-01 16:43:27,430 : INFO : gram6-nationality-adjective: 85.4% (1171/1371)
2015-06-01 16:47:42,424 : INFO : gram7-past-tense: 51.4% (723/1406)

2015-06-01 16:50:19,657 : INFO : gram8-plural: 64.4% (639/992)
2015-06-01 16:51:48,427 : INFO : gram9-plural-verbs: 50.0% (300/600)
2015-06-01 16:51:48,428 : INFO : total: 61.3% (8291/13534)

$size = 300, window = 8, min\_count = 10$

Trainingsdauer: 8,5h

2015-04-29 09:33:51,921 : INFO : capital-common-countries: 81.8% (414/506)
2015-04-29 09:43:43,187 : INFO : capital-world: 79.6% (1745/2192)
2015-04-29 09:44:02,069 : INFO : currency: 0.0% (0/70)
2015-04-29 09:53:58,558 : INFO : city-in-state: 47.3% (1043/2203)
2015-04-29 09:55:42,911 : INFO : family: 84.5% (321/380)
2015-04-29 09:59:26,937 : INFO : gram1-adjective-to-adverb: 16.6% (135/812)
2015-04-29 10:01:22,808 : INFO : gram2-opposite: 37.9% (159/420)
2015-04-29 10:06:50,884 : INFO : gram3-comparative: 68.4% (814/1190)
2015-04-29 10:08:57,792 : INFO : gram4-superlative: 33.1% (153/462)
2015-04-29 10:13:10,012 : INFO : gram5-present-participle: 48.6% (452/930)
2015-04-29 10:19:17,184 : INFO : gram6-nationality-adjective: 85.6% (1174/1371)
2015-04-29 10:25:34,414 : INFO : gram7-past-tense: 51.1% (718/1406)
2015-04-29 10:30:00,645 : INFO : gram8-plural: 59.3% (588/992)
2015-04-29 10:32:35,388 : INFO : gram9-plural-verbs: 46.3% (278/600)
2015-04-29 10:32:35,388 : INFO : total: 59.1% (7994/13534)

$size = 300, window = 10, min\_count = 5$

Trainingsdauer: 7,7h

2015-04-29 19:24:36,700 : INFO : capital-common-countries: 83.0% (420/506)
2015-04-29 19:45:12,723 : INFO : capital-world: 80.2% (1759/2192)
2015-04-29 19:45:50,961 : INFO : currency: 0.0% (0/70)
2015-04-29 20:05:06,782 : INFO : city-in-state: 50.4% (1111/2203)
2015-04-29 20:08:13,267 : INFO : family: 81.1% (308/380)
2015-04-29 20:14:57,813 : INFO : gram1-adjective-to-adverb: 16.5% (134/812)
2015-04-29 20:18:29,605 : INFO : gram2-opposite: 38.8% (163/420)
2015-04-29 20:28:27,904 : INFO : gram3-comparative: 68.9% (820/1190)
2015-04-29 20:32:19,788 : INFO : gram4-superlative: 28.4% (131/462)
2015-04-29 20:40:00,747 : INFO : gram5-present-participle: 49.9% (464/930)
2015-04-29 20:51:19,611 : INFO : gram6-nationality-adjective: 88.8% (1217/1371)
2015-04-29 21:02:59,292 : INFO : gram7-past-tense: 48.6% (683/1406)
2015-04-29 21:11:12,999 : INFO : gram8-plural: 60.3% (598/992)
2015-04-29 21:16:14,655 : INFO : gram9-plural-verbs: 44.7% (268/600)
2015-04-29 21:16:14,657 : INFO : total: 59.7% (8076/13534)

Der Wikipedia-Gesamtkorpus enthält 319.284.725 Sätze.

Um einen domänenspezifischen Teilkorpus aus dem Wikipedia-Gesamtkorpus zu generieren, habe ich einen Naiven-Bayes-Klassifikator (den selbst implementierten aus der dem Data-Mining Praktikum) ausgewählt. Dazu waren von Hand getaggte Artikel nötig, um

den Klassifikator zu trainieren. Ich habe folgende Kategorien ausgewählt:
tech, *entertainment*, *sport*, *science*, *politic*. Da viele Wikipedia-Artikel nicht eindeutig einzuordnen waren, mussten deutlich mehr Artikel angeschaut werden. Der Naiven-Bayes-Klassifikator wurde mit 100 Artikeln je Kategorie trainiert.

KW19

Weiteres tagging von Wikipedia-Artikeln.
Anpassen des Perl-Skript, da nicht bekannte Zeichen (Sonderzeichen etc) bisher mit " " ersetzt wurden, was dann Lücken in Wörter produziert und seltsame Wörter erzeugt hat. Nach der Anpassung wurden nicht bekannte Zeichen einfach aus dem Wort entfernt.

KW20

Die einzelnen Artikel mit dem Perl-Skript gereinigt.
Klassifizierung der Artikel mit dem trainierten Naiven-Bayes-Klassifikator und erstellen des Teilkorpus'.
Zwei Modelle mit den gleichen Parametern ($size = 300, window = 10, min\_count = 5$) aus dem Teilkorpus erstellt. Zum einen habe ich den Teilkorpus als Vokabular und Trainingsmenge benutzt, beim anderen nur als Vokabular und auf dem gesamten Wikipedia-Korpus trainiert.

KW21

Testdatenmenge aus Technologie-Begriffen erstellt um die nächsten Nachbarn dieser Worte innerhalb der einzelnen Korpora zu vergleichen *(5,01)*.
Bei dem implementierten Test werden jeweils die fünf nächsten Nachbarworte der drei Korpora (Gesamtkorpus, Teilkorpus trainiert auf dem Gesamtkorpus, Teilkorpus) aufgelistet und lesbar formatiert *(5,02)*.

KW22

Urlaub.

Ausblick auf die nächste Berichtsperiode:
Es soll ein weiterer Domänenkorpus erstellt werden (z.B. aus einem Game-Forum). Dazu sollen auch Testdaten erstellt und der Korpus damit getestet werden. Im Anschluss sollen dann die Ergebnisse aus dem Wikipedia-Korpus und dem anderen Domänenkorpus verglichen werden.

KW23

Krank.

KW24

Am 08.06.2015 um 13:15 Uhr fand das erste Statusmeeting*(5,05)* im Raum 321 der Hochschule der Medien mit Prof. Dr. Johannes Maucher, M.Sc. Andreas Stiegler (via Skype) und Ruben Müller statt.

Wie im Statusmeeting besprochen sollte getestet werden ob die Klassifizierung der Wikipedia-Artikel mit einer Support-Vector-Maschine besser gelingt als mit dem Naiven-Bayes-Classifier. Diese SVM habe ich dann versucht einzubinden und zu trainieren. Ich habe den SGDClassifier aus dem Package sklearn verwendet. Die Implementierung kostete mich viel Zeit, da ich kaum passende Beispiele gefunden habe und dann öfters in Fehler gelaufen bin. Nachdem das Skript dann lief musste ich feststellen, dass alle Artikel in die gleiche Klasse eingeteilt wurden. Habe ich diese Klasse dann beim Training herausgenommen wurden alle Artikel in eine andere, aber wiederum die gleiche, Klasse eingeteilt. Auch das ändern einiger Parameter im SGDClassifier brachte keine Besserung. Da ich bis dahin schon einige Zeit in die Implementierung der SVM gesteckt habe und es ja auch noch die Fragestellungen zu bearbeiten galt, habe ich mich entschlossen meine bisherige Klassifizierung mittels Naivem Bayes Klassifizierer zu behalten und die SVM nicht anzuwenden.

KW25

Im Statusmeeting wurde vereinbart, dass die trainierten Modelle mit ein paar Fragestellungen getestet werden sollen, bzw Hypothesen bestätigt oder widerlegt werden. Ich habe dann die Fragenstellungen/Hypothesen erarbeitet und mir Gedanken gemacht wie ich die Auswertung am besten darstelle. Ich habe mich entschieden alle Fragestellungen gleichzeitig zu bearbeiten und die Ergebnisse in Excel festzuhalten so musste ich nur einmal über alle Daten (235 Testworte) iterieren. Die Auswertung habe ich dann noch mit Kreisdiagrammen grafisch dargestellt.

Ausblick auf die nächste Berichtsperiode:
Verschriftlichung der Ergebnisse und Erkenntnisse.

KW26

Schreiben der Thesis.

KW27

Schreiben der Thesis.