45s

Motivation und Problemstellung

Eine Aufgabe des NLP ist es, Beziehungen zwischen Wörtern darzustellen. Um solche Beziehungen darzustellen gibt es unterschiedliche Techniken bzw. Vorgehgensweisen. Neben z.B. LSI können semantische Beziehungen auch mit Word2Vec dargestellt werden. Und um Word2Vec soll es in dieser Arbeit auch gehen. Es soll untersucht werden, wie sich unterschiedliche Textgrundlagen (Korpora) als Trainingsdaten auf die Beziehungen zwischen den Wörtern im Modell auswirken.

Die Ergebnisse der Arbeit sollen es ermöglichen, dass für eine gewisse Fragestellung oder Anwendung des Word2Vec Modells, die richtigen Trainingsdaten ausgewählt werden können.

4,5m

Word2Vec

Wie der Name schon sagt, werden im Word2Vec Modell Worte als Vektoren dargestellt. Ein Ziel von Word2Vec ist es, semantisch oder syntaktisch ähnliche Worte zu finden. Die Ähnlichkeit zwischen zwei Worten kann mit der Kosinusähnlichkeit zwischen den Vektordarstellungen der Worte berechnet werden. Vor Word2Vec gab es schon andere statistische Methoden um Ähnlichkeiten zwischen Wörtern zu finden, z.B. LSI oder NNMF. Bei diesen Verfahren werden Wörter als semantisch ähnlich gefunden, wenn die beiden Wörter in vielen Dokumenten gemeinsam vorkommen. Im Word2Vec Modell werden aber Wörter dann als semantisch ähnlich erkannt, wenn sie häufig im gleichen Kontext vorkommen.

Bei der Modellbildung ist der erste Schritt, dass alle Wörter, die eine Mindestanzahl erreichen, in einem Vokabular abgebildet und dann mittels "distributed

representation" *(Vektor über Eigenschaften werden gebildet)* dargestellt werden. Diese Vektorrepräsentationen dienen dann als Input für ein neuronales Netz. Mikolov et al. *("Efficient estimation of word representations in vector space.")* haben Neuronale Netzte ohne verdeckte Schichten bevorzugt, da zwar die Genauigkeit bei mehreren verdeckten Schichten steigt, aber somit auch die Komplexität und Trainingsdauer. *(Mit einer optimierten Immplementierung haben Mikolov et al. auf einem einzigen Rechner mehr als 100 Milliarden Worte pro Tag trainieren können).*

Wie schon erwähnt, werden in Word2Vec Wörter als ähnlich erkannt, wenn diese in den Trainingsdaten oft im gleichen Kontext vorkommen. Dies wird durch die Arbeitsweise der neuronalen Netze erreicht. Mikolov et al. *("Efficient estimation of word representations in vector space.")* stellen zwei Architekurtypen von neruonalen Netzen vor:

CBOW und Skip-gram

!!!!Abbildung 3.1 einfügen!!!!

CBOW

Die CBOW-Architektur ist ein neuronales Netz ohne verdeckt Schichten und mit einer Projektionsschicht.

Beim Training von CBOW werden einige Wörter vor und einige Wörter nach dem aktuellen Wort benutzt um daraus möglichst genau das aktuelle Wort vorherzusagen. Trainingskriterium ist es hier, das momentane Wort korrekt vorherzusagen, bis das erreicht ist, werden die Vektordarstellungen dann immer wieder angepasst.

Skip-gram

Skiip-gram ist wie CBOW ein neuronales Netz ohne verdeckte Schichten, allerdings wird hier nicht aus dem Kontext ein Wort vorhergesagt, sondern anhand eines Wortes wird der Kontext/Wörter vor und nach dem aktuellen Wort vorhergesagt.

Zu dieser Architektur haben Mikolov et al. eine weitere Arbeit geschrieben *("Distributed representations of words and phrases and their com-positionality"*) in der sie die Trainingsfunktionen "hierarchical softmax" und "negative sampling" vorstellen.

Der "hierarchical softmax" Lernalgorithmus ist eine effiziente Alternative zum allgemeinen "softmax" Algorithmus. Hier wird ein Huffman Binärbaum zur Repräsentation der Ausgabeschicht des neuronalen Netztes verwendet. Die Blätter sind die Ausgabeknoten des neuronalen Netztes und die inneren Knoten geben die relative Wahrscheinlichkeit ihrer Kindknoten an. So entstehen dann druch beliebige Pfade von der Wurzel zu den Blättern Wortwahrscheinlichkeiten. So müssen dann nicht alle W Ausgabeknoten ausgewertet werden, sondern nur ungefähr Log2(W).

Beim "negative sampling" wird nicht die Ähnlichkeit zu anderen Wörtern berechnet, sondern es werden zufällig Wörter ausgesucht und dann wird davon

ausgegangen, dass diese mit hoher Wahrscheinlichkeit unähnlich, dem zu testenden Wort sind. (Mittels logistischer Regression, soll dann ein ähnliches Wort aus der Rauschverteilung Pn(W) erkannt werden)

Unterschiedliche Arten von Ähnlichkeiten

Durch die Vektordarstellung der Wörter, erlaubt es das Modell, dass viele unterschiedliche Arten von Ähnlichkeiten zwischen Wörtern dargestellt/abgefragt werden können. Eine spezielle Art der Ähnlichkeit wird als "Offset-Vektor" zwischen zwei Vektordarstellungen von Wörtern dargestellt.

!!!!Abbildung 3.2 einfügen!!!!

So ist der Vektor zwischen Mann und Frau und zwischen König und Königin der gleiche. Das ist dann sozusagen der "Männlich-weiblich-Offset-Vektor".

Für die praktischen Teile habe ich die Gensim-Implementierung von Word2Vec benutzt.

3,5m

Daten und Vorverarbeitung

Für das Training von Word2Vec Modellen wird eine große Anzahl an Trainingsdaten benötigt. Eine große Anzahl bedeutet hier, im Optimalfall mehrere Milliarden von Wörtern. Auf der Googlecode-Seite von Word2Vec werden online verfügbare Datensätze aufgelistet. Unter anderem auch der komplette Wikipediakorpus und ein kleinerer Teilkorpus mit den ersten 1 Milliarde Zeichen. In dieser Arbeit sollen Modelle mit unterschiedlichen Trainingskorpora verglichen werden, also müssen diese unterschiedlichen Korpora erstellt werden. Die Wahl fiel dann zum einen auf den gesamten Wikipediakorpus und zum anderen auf einen domänenspezifischen Teilkorpus, der aus allen Artikeln des Gesamtkorpus besteht, die sich mit der Thematik Technologie/Pc/Internet befassen. Um diesen Teilkorpus zu erhalten, waren eingige Vorverarbeitungsschritte nötig. Der komplette Wikipediakorpus kann als große XML-Datei heruntergeladen werden. Um aus dieser Datei die einzelnen Artikel zu extrahieren, muss sie mit einem SAX-Parser geparst werden. Wenn dann die einzelnen Artikel als separate Dateien vorliegen, können diese Klassifiziert

werden. Für die Klassifizierung wurde der, im DataMining-Praktikum erstellte, Naive Bayes Klassifikator benutzt. Um ihn zu trainieren mussten zuerst von Hand Artikel getaggt werden. Für jede Klasse sollten 100 Artikel für das Training des NBCs gefunden werden. Und bei den fünf Klassen (tech, entertainment, sport, science, politic) waren das dann 500 Artikel die gefunden werden mussten. Das hat einige Zeit in Anspruch genommen, da oft die Wikipediaartikel nicht eindeutig zuordenbar waren und ich sie dann lieber nicht für das Training verwendet habe. Nach der Klassifizierung wurden dann alle Artikel mit einem Perlskript von Matt Mahoney, das leicht angepasst wurde, vom Wiki-Markup gereinigt. Dieses Skript wurde auch auf die große XML-Datei angewandt um den Gesamtkorpus zu reinigen. Nachdem die einzelnen Artikel gereinigt waren, wurden sie wieder zusammengefügt. Die Word2Vec-Implementierung in Gensim sieht es vor, dass die Trainingsdaten als Array aus einzelnen Sätzen eingegeben werden. Da die Datei mit dem kompletten Wikipediakorpus sehr groß ist, kann sie zwar direkt in Python geöffnet werden, allerdings wird das Array mit den einzelnen Sätzen zu groß *(bei über 40 GB habe ich dann abgebrochen).* Die Word2Vec Klasse kann alternativ zu dem Array aus Sätzen auch ein Objekt der Klasse "gensim.models.word2vec.LineSentence" entgegen

nehmen. In der Input-Datei dieser Klasse müssen die einzelnen Zeilen je einen Satz enthalten. So habe ich dann den Gesamtkorpus zuerst von Satzzeichen und Großschreibung befreit und dann die einzelnen Sätze in je eine Zeile gebracht. Damit konnte ich dann mittels der erwähnten "LineSentence" Klasse das Word2Vec Modell erstellen und trainieren. Beim Technologiekorpus bin ich dann gleich vorgegangen. Theoretisch wäre diese Datei nicht so groß gewesen und man hätte einfach ein Array mit den einzelnen Sätzen erstellen können, da ich die Skripte ja schon hatte habe ich einfach die gleiche Vorangehensweise benutzt.

4,5-5m

Wikipedia Korpus, verwendete Modellparameter und Testdaten

Die Wikipedia Daten wurden aus mehreren Gründen benutzt. Erstens ist die Menge der Daten sehr groß, was für eine gute Qualität der Wortvektoren im Word2Vec Modell wichtig ist. Im kompletten Korpus sind es über 2,9 Milliarden an Wörtern. Zudem ist die Qualität der Artikel gut, da es redaktionelle Texte sind und keine

Kommentare in Foren, wo die Rechtschreibung und Gramatik nicht immer so stimmt. Des Weiteren werden in Wikipedia eine sehr breite Menge an Themen beschrieben, was für das allgemeine Modell sehr hilfreich ist.

Da die Skip-gram Architektur im Bezug auf semantsiche Ähnlichkeit deutlich besser ist, als die CBOW Architektur, wurde diese Architektur ausgewählt, da in den Experimenten semantische Ähnlichkeiten überprüft werden sollen. Beim Lernalgorithmus des Neuronalen Netzs fiel die Wahl auf den hierarchical softmax Algorithmus, da dieser besser für selten vorkommende Wörter geeignet ist. Und so für die technologiespezifischen Daten besser passt. Zudem ist der negative sampling Lernalgorithmus eher geeignet für niedrigdimensionale Vektoren, und wie wir nacher noch sehen, werden eher hochdimensionale Wortvektoren benützt.

Um die weiteren Parameter für das Training des Word2Vec Modells herauszufinden, wurde der kleinere Wikipedia Auszug mit den ersten 1 Milliarde Zeichen als Trainingsdaten benutzt. Es wurden dann mehrere unterschiedliche Modelle mit unterschiedlichen Parametern erstellt und verglichen. Der Vergleich wurde mit der in der Gensim-Implementierung enthaltenen Accuracy()-Funktion

durchgeführt. Die Funktion erwartet einen Dateinamen einer Datei, in der jede Zeile aus zwei Wortpaaren besteht, die Worte der Paare haben die gleiche Beziehung untereinander. Zum Beispiel "Athens Greece Berlin Germany", hier stellen die Wortpaare die Beziehung zwischen der Hauptstadt und dem dazugehörigen Land dar. *(Diese Datei kann noch in Abschnitte unterteilt werden, was mit ": SECTION NAME" erreicht werden kann.)* Auf der Google Code Seite von Word2Vec ist eine solche Datei als Beispiel vorhanden, diese enthält 19544 solcher 4-Tupel *(und ist in 14 Kategorien eingeteilt capital-common-countries, capital-world, currency, city-in-state, family, gram1-adjective-to-adverb, gram2-opposite, gram3-comparative, gram4-superlative, gram5-present-participle, gram6-nationality-adjective, gram7-past-tense, gram8-plural, gram9-plural-verbs).* In der Gensim-Implementierung stellt das Modell eine Funktion bereit, die zwei Arrays und eine Zahl "N" entgegennimmt, das eine ist das "positive-Array", das andere das "negative-Array". In dieser Funktion werden die Wortvektoren der Worte im Array positive aufaddiert und die Vektoren aus dem negative-Array davon subtrahiert. Die Funktion liefert dann N Worte zurück, die eine maximale Ähnlichkeit mit dem Ergebnis der Vektoraddition und -subtraktion haben. Diese Funktion wird in der

Accuracy-Funktion benutzt, indem die ersten drei Worte des 4-Tupels in positive und negative eingeteilt werden und vierte Wort muss dann korrekt vom Model vorhergesagt werden. Hier: most_similar(positive=['Greece','Berlin'], negative=['Athens'], topn=1).

!!!!Tabelle 4.1 einfügen!!!!

Diese Werte beziehen sich auf den kleinen Wikipediakorpus. Mit den Parametern size=400, Window=10 und Min_count=5 wurde die beste Accuracy erreicht. Mit diesen Parametern beträgt die Trainingsdauer des Gesamtmodells über 10,5h. Durch einen Fehler im Perl-Reinigungsskript musste das Modell im nachhinein nochmals trainiert werden, hier fiel dann auch die Entscheidung, andere Parameter zu verwenden und so wurde size auf 300 verringert. Hier betrug die Trainingszeit dann nur noch etwas über 7,5h. Da sich das Perl-Skript noch wenige Male geringfügig änderte, war diese Entscheidung gut.

Für die Experimente wurden noch Testbegriffe gebraucht um die Modelle vergleichen zu können, so habe ich Wikipedia und Google nach Technologie/PC/Internet Stichwörtern durchsucht. 236 insgesamt.

9,5m

Experimente

Um die beiden Modelle zu vergleichen wurden fünf Experimente durchgeführt, in denen die semantischen Beziehungen zwischen Wörtern untersucht werden. In den Experimenten werden die Testworte mit den Wörtern verglichen, die von der Methode most_similar() zurück gegeben werden. Diese Wörter werde ich ab jetzt immer "ähnliche Wörter" nennen.

Synonymsuche durch Rekursion

In diesem Experiment soll herausgefunden werden, ob Synonyme im Word2Vec Modell gefunden werden können. Allerdings werden nicht der Testbegriff und die ähnlichen Wörter betrachtet, sondern die ähnlichen Wörter dienen erneut als Eingabe für die Suche mit der Funktion most_similar(). Diese Methode wird also 1 Mal rekursiv angewandt.

Vor dem eigentlichen Experiment mussten noch die Synonyme der Testbegriffe gefunden bzw festgelegt werden. Dies wurde mit einem Webservice *(unter thesaurus.altervista.org)* erledigt. *(Die Synonyme werden aus den Thesaurus-Lexikas von OpenOffice erhalten)* Allerdings liefert der Service nur für 108 der 236 Testwörter Synonyme, deshalb wurden in diesem Experiment auch nur diese 108 Wörter berücksichtigt. Im eigentlichen Experiment werden dann zuerst die fünf ähnlichsten Wörter des jeweiligen Testbegriffs mit most_similar() herausgefunden und die Rückgabewörter werden direkt wieder als Eingabe für ein erneutes Suchen mit most_similar() benutzt. Die somit erhaltenen 25 Wörter werden mit den Synonymen des jeweiligen Testbegriffs verglichen. Sollte es mindestens eine Übereinstimmung geben, wird dieses Wort als erfolgreich markiert.

!!!!Tabelle 5.1 einfügen!!!!

Wie die Tabelle zeigt, werden im Gesamtmodell nur bei zwölf, im Technologiemodell sogar nur bei zwei Testbegriffen Synonyme gefunden. Ein Grund dafür kann sein, dass durch die Rekrusion noch weiter entfernte Wörter gefunden

werden, die dann nicht mehr so viel mit dem Ausgangswort zu tun haben. Ein weiterer Grund kann auch sein, dass die Synonymliste nicht so ausführlich war.

Konkretisierungen

Ziel des Experiments soll sein, herauszufinden, ob die ähnlichen Wörter der Testdaten eine Konkretisierung des jeweiligen Suchbegriffs darstellen. Dies ist gerade bei mehrdeutigen Wörtern interessant, sollten dann nämlich Konkretisierungen einer Sinnrichtung häufiger in den ähnlichen Wörtern auftauchen, wird klar, welche Bedeutung in diesem Kontext/Domäne gemeint ist. Es wird vermutet, dass im Technologiemodell mehr Konkretisierungen auftauchen als im allgemeinen Gesamtmodell.

Im Experiment soll nur untersucht werden, OB eine Konkretisierung in den ähnlichen Wörtern vorhanden ist, und nicht, wie stark konkretisiert wird. *(Auto wird mit BMW, Mercedes, VW UND X5, A-Klasse, Tuareg konkretisiert)*. Um zu testen wie

stark konkretisiert wird, wäre ein zusätzliches Experiment nötig *(zeitaufwändig, da Vergleichsdaten erstellt werden müssten oder aber von Hand).*

Die Auswertung der ähnlichen Wörter erfolgte von Hand, da ich keine Auflistung von Wörtern und dazugehörigen Konkretisierungen gefunden habe. Diese Auswertung nahm einige Zeit in Anspruch, da ich viele Begriffe nachschlagen musste, denn um beurteilen zu können ob ein ähnliches Wort den Suchbegriff konkretisiert, muss die inhaltliche Beziehung zwischen den Wörtern bekannt sein.

(Wenn 3 von 5 eine Konkretisierung waren, dann als solche gezählt.)

!!!!Tabelle 5.3 einfügen!!!!

Bei 15,7% im Gesamtmodell und 27,5% im Technologiemodell haben die ähnlichen Wörter die jeweiligen Suchbegriffe konkretisiert. Die Annahme hat sich also bestätigt, dass im Technologiemodell mehr Konkretisierungen zu finden sind. *(Dies hat den Grund, dass sich die Daten im Gesamtmodell nicht auf eine Fachrichtung beschränken, wie dies im Technologiemodell der Fall ist. So werden beim Training mit Technologiedaten im Word2Vec Modell die Beziehungen beibehalten, die beim*

Training mit dem kompletten Wikipediakorpus durch andere, allgemeine Worte noch weiter verändert werden.)

Verallgemeinerungen

In diesem Experiment geht es darum, herauszufinden ob im allgemeinen Gesamtmodell die ähnlichen Wörter die Suchbegriffe mehr verallgemeinern als im Technologiemodell. Auch hier soll nicht die Stärke der Verallgemeinerung getestet werden, sondern nur, ob es überhaupt Verallgemeinerungen gibt.

Wie im vorherigen Experiment wurden hier die Testwörter und ihre ähnlichen Wörter wieder von Hand analysiert, da es auch hier keine Auflistungen gibt, die Verallgemeinerungen für Wörter darstellen.

!!!!Tabelle 5.5 einfügen!!!!

Bei 56,8% im Gesamtmodell und 28,4% im Technologiemodell verallgemeinern die ähnlichen Wörter die Suchbegriffe. Es bestätigt sich die Vermutung, dass im

allgemeinen Gesamtmodell mehr ähnliche Wörter die Suchbegriffe verallgemeinern, als im Technologiemodell. *(Dies hängt vermutlich mit der Tatsache zusammen, dass das Gesamtmodell mit Daten, die nicht auf ein spezielles Thema beschränkt sind, trainiert wurde und diese Begriffe in vielen unterschiedlichen Zusammenhängen im Trainingskorpus enthalten sind.)*

Unterschiedliche Beziehungen

Ganz allgemein gesehen können Beziehungen zwischen zwei Wörtern unterschiedliche Arten haben. Es kann unter anderem zwischen "gleiches", "gegenteiliges", "syntaktisches", "synonymes" und "in Beziehung" unterschieden werden. In diesem Experiment soll herausgefunden werden, ob die unterschiedlichen Modelle auch unterschiedliche Beziehungsarten zwischen den Testwörtern und ihren ähnlichen Wörtern haben. Eine weitere Kategorie wurde noch hinzugefügt, "verschiedenes" sie wird verwendet, wenn die ähnlichen Wörter nicht in mindestens drei Fällen eine gleiche spezielle Kategorie abdecken.

Auch in diesem Experiment wurden die Testbegriffe und ihre ähnlichen Wörter von Hand analysiert und in die Kategorien eingeteilt.

!!!!Tabelle 5.7 einfügen!!!!

In beiden Modellen sind die Arten der Beziehungen die gleichen: "gleiches", "in Beziehung" und "verschiedenes". Es gibt zwar vereinzelt Beziehungen die "syntaktisches", "gegenteiliges" oder "synonymes" darstellen, allerdings kommen sie nur 1-2 mal in den ähnlichen Wörtern vor, und werden dann entweder mit "verschiedenes" markiert, oder aber, wenn drei mal "gleiches" oder "in Beziehung" vorhanden ist, dann als dieses. In beiden Modellen hat die Kategorie "in Beziehung" die meisten Eintragungen. Dies könnte damit erklärt werden, dass im Word2Vec Modell Wörter, die im gleichen Kontext stehen, ähnliche Vektordarstellungen haben. Die Beziehungsart "verschiedenes" sagt nichts über die tatsächlichen Beziehungen aus, außer, dass nicht eine Art dominant war.

Im Gesamtmodell kommt etwas häufiger die Art "gleiches" vor als im Technologiemodell.

Erkennen von Mehrdeutigkeit

Das letzte Experiment befasst sich mit der Mehrdeutigkeit von Wörtern. Es wird untersucht ob die ähnlichen Wörter die unterschiedlichen Bedeutungen häufiger im Gesamtmodell oder im Technologiemodell darstellen. Es werden nur die 66 mehrdeutigen Testbegriffe verwendet.

In diesem Experiment wurde auch wieder von Hand analysiert, ob zwei unterschiedliche Bedeutungen eines mehrdeutigen Wortes in den ähnlichen Wörtern vorhanden sind. War das der Fall, so wurde dieser Testbegriff entsprechend markiert.

!!!!Tabell 5.10 einfügen!!!!

In beiden Modellen wurden nur jeweils 7 Begriffe als mehrdeutig erkannt. Es gab nur einen Begriff, den beide Modelle gleichermaßen erkannten ("trojan"), ansonsten waren es 13 unterschiedliche Begriffe. Ein Grund der geringen Erkennungsrate könnte sein, dass die unterschiedlichen Bedeutungen oft sehr unterschiedliche Dinge beschreiben, und so kommen diese Begriffe selten im

gleichen Kontext vor, sodass sie dann vom Word2Vec Training erkannt würden. Mit diesem Ansatz kann also eine Mehrdeutigkeit nicht effektiv herausgefunden werden.

Bei 35 Testbegriffen wurde im einen Modell eine und im anderen Modell eine andere Bedeutung in den ähnlichen Wörtern dargestellt *(architecture - revival, neoclassical, gothic; architecture - armv7a, multicore, xmos; fat - calories, milk, fatfree; fat - ntfs, exfat, fat32).*

!!!!Tabelle 5.12 einfügen!!!!

Fazit und Ausblick

Zusammenfassend kann gesagt werden, dass sich das allgemeine Gesamtmodell dann besser eignet, wenn ein genereller Überblick über das Suchwort erhalten werden soll, denn in diesem Modell werden die Suchbegriffe eher verallgemeinert und die ähnlichen Worte repräsentieren mehr "gleiches".

Soll ein Suchbegriff genauer untersucht bzw. fokussierter betratchtet werden, ist das domänenspezifische Modell besser. Hier werden die Suchwörter mehr konkretisiert und liefern auch natürlich fachspezifischere ähnliche Wörter.

Weitere Arbeiten die sich in diesem Bereich mit Word2Vec beschäftigen, könnten weitere Experimente sein, die sich mit anderen Arten von Beziehungen beschäftigen. Z.B. könnte die Mehrdeutigkeit noch anders untersucht werden, indem man mehrere unterschiedliche Modelle hat, die mit unterschiedlichen Koropra trainiert wurden, und dann vergleicht man die ähnlichen Wörter eines Testbegriffes, sollten diese sehr unterschiedlich sein, könnte dieser Testbegriff Mehrdeutig sein.