ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

Bachelor-Thesis

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)
an der Hochschule für Technik und Wirtschaft des Saarlandes
im Studiengang Praktische Informatik
der Fakultät für Ingenieurwissenschaften

Zeitliche Analyse von Sprach- und Bedeutungsveränderungen mit Hilfe von Word Embeddings

vorgelegt von
Alexander Lukacs

betreut und begutachtet von
Prof. Dr.-Ing. Klaus Berberich

Saarbrücken, 25. 09 2018

Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit (bei einer Gruppenarbeit: den entsprechend gekennzeichneten Anteil der Arbeit) selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich erkläre hiermit weiterhin, dass die vorgelegte Arbeit zuvor weder von mir noch von einer anderen Person an dieser oder einer anderen Hochschule eingereicht wurde.

Darüber hinaus ist mir bekannt, dass die Unrichtigkeit dieser Erklärung eine Benotung der Arbeit mit der Note „nicht ausreichend“zur Folge hat und einen Ausschluss von der Erbringung weiterer Prüfungsleistungen zur Folge haben kann.

Saarbrücken, 25. 09 2018

Alexander Lukacs

Zusammenfassung

In dieser Arbeit wird gezeigt wie man mit Hilfe von Word Embeddings Sprach- und Bedeutungsveränderungen, über Zeit, analysieren kann. Es wird kurz erklärt, was neuronale Netze und was Word Embeddings sind. Nach der kurzen Einführung der zuvor genannten Theorien, werden dann noch auf verwandte Arbeiten hingewiesen, die ähnliche Themen bearbeiten, wie diese Arbeit hier. Danach wird die Vorgehensweise näher erläutert, es wird gezeigt was die Aufgabenstellung war und wie der Lösungsansatz war. Als Nächstes wird die Implementierung präsentiert, hier wird das Datenbank-Schema gezeigt und es wird auf verschiedene Abschnitte hingewiesen die wichtig sind. Daraufhin werde die Ergebnisse näher erläutert und der Sinn bzw. die Bedeutung der Ergebnisse besprochen. Und zu guter Letzt kommt das Fazit, wo noch einmal eine kurze und genauere Zusammenfassung der Arbeit ist und eine kleine Zukunftsaussicht, wie man weiter mit den Word Embeddings arbeiten könnte bzw. was die Auswertungen für die Zukunft bedeuten.

Danksagung

Ich möchte Herrn Prof. Klaus Berberich sehr herzlich danken für die vorzügliche und zeitintensive Betreuung dieser Bachelorarbeit.

Inhaltsverzeichnis

1 Einleitung

Seit Jahren verändern sich Wörter, manche Wörter bekommen eine andere Bedeutung, wie z.B. Dirne, früher stand das Wort für junges Mädchen oder Dienerin, heutzutage steht das Wort für Prostituierte, eine bestimmte Bedeutung bekommt ein neues Wort, wie z.B. Ipod statt Walkman, ein altes Wort verschwindet komplett, wie z.B. Heiermann, der Nickname für das 5-Mark-Stück, oder ein Wort bleibt über die Jahre gleich, wie z.B. Auto oder Haus. Aber woher will man wissen welche Wörter sich verändert haben, dies hat niemand genau vermerkt, aber alle Wörter sind schon mal vorgekommen, nämlich in den Zeitungen. Dann ist es ja einfach, man muss nur alle Zeitungen zusammen suchen und die Wörter miteinander vergleichen, das Problem ist nur, dass es sehr viele Zeitungen gibt. Und um alle Zeitungen zu vergleichen braucht man mehrere Jahre, deswegen gibt es eine einfache Lösung, man lässt den Computer alles vergleichen. Die Frage ist nur, wie kann der Computer die Wörter vergleichen? Da der Computer nicht wie ein Mensch denken kann, wird es mit dem Vergleichen schwer. Also müssen wir erst die Frage klären, wie kann der Computer wie ein Mensch denken? Dafür müsste man neuronale Netze kennen und es dann in ein künstliches neuronales Netz umwandlen können. Darauf aufbauend kann man mit Hilfe des Word Embeddings Wortrepräsentationen erkennen und die passenden Vektoren bilden.
Diese beiden Methoden werden im Kapitel Technische Grundlagen genauer erklärt.
Für diese Arbeit benutze ich beide Methoden, um die Fragen zu lösen, wann hat sich ein Wort verändert, welches Wort hat sich wann verändert und welche Wörter sind zueinander gleich bzw. ähnlich.

Die Arbeit ist in sechs weitere Kapitel gegliedert. Im nächsten Kapitel stehen die Technischen Grundlagen. Im 3. Kapitel kommen die verwandten Arbeiten. Im 4. Kapitel kommt die Vorgehensweise der Arbeit. Das 5. Kapitel ist die Implementierung. Das 6. Kapitel ist die Evaluation der Ergebnisse. Und das letzte Kapitel ist das Fazit.

2 Technische Grundlagen

In diesem Kapitel werden die Grundlagen erläutert, die zum Verständins der Arbeit notwendig sind. Zuerst werden die neuronalen Netze bzw. die künstlichen neuronalen Netze erklärt. Und danach wird erläutert was Word Embeddings sind.

2.1 Künstliche neuronale Netze

Der Mensch kann viele verschiedene Aufgaben lösen, die der Computer jedoch besser und schneller lösen kann, bis auf Aufgaben die ein hohes Maß an Undeterminiertheit, Vagheit und Unschärfe aufzeichnen wie früher z.B. Gesicht- oder Spracherkennung. Heutzutage ist es jedoch möglich die Sprache oder das Gesicht zu erkennen, wie das Smartphone es uns zeigt. Aber dennoch ist es schwer die Denkweise des Menschen zu imiiteren. Und deswegen wird versucht ein künstliches neuronales Netz zu entwickeln, damit die Organisations- und Verarbeitungsprinzipien des menschlichen Gehirns von einem Computer übernommen werden können. Um auch Aufgaben die ein hohes Maß an Undeterminiertheit, Vagheit und Unschärfe aufzeichnen, besser und schneller bewältigt werden können. Aber was genau ist ein künstliches neuronales Netz? Ein künstlich neuronales Netz kann man sich wie ein Graph mit Knoten und Kanten, die gewichtet und gerichtet sind, vorstellen. Die Knoten sind in Schichten unterteilt, je mehr Knoten hintereinander kommen desto mehr Schichten wird es geben. Es gibt am Anfang des Graphen eine Eingabeschicht und am Ende eine Ausgabeschicht. Die Menge der Knoten in den Schichten sind, ist beliebig groß. Die Schichten die zwischen Ein- und Ausgabeschicht werden verdeckte, versteckte oder verborgene Schicht genannt. Die Knoten verarbeiten eine beliebige Anzahl numerischer Eingangswerte und erzeugen einen numerischen Ausgabewert. Eingangswerte sind entweder Ausgabewerte anderer Knoten oder Eingabemuster aus der Systemumgebung, also der netzwerkexternen Umwelt.

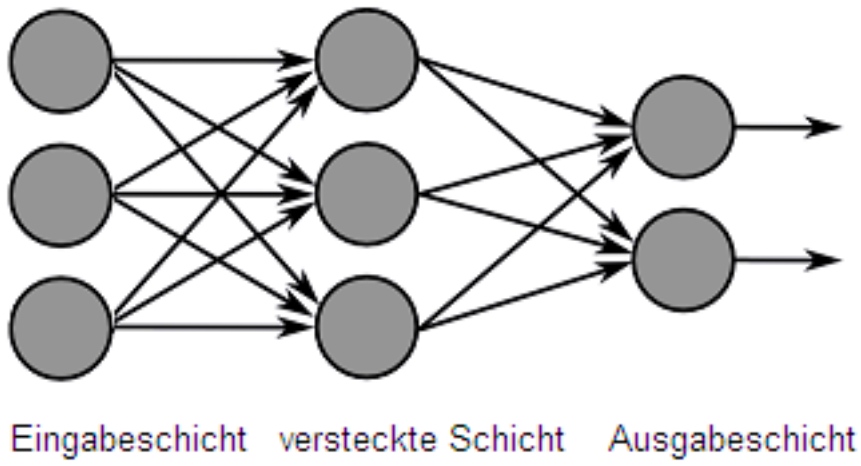

Abbildung 2.1: Aufbau eines Neuronales Netzwerk [12]

"Künstliche Neuronale Netze weisen Kernkomponenten oder Grundbausteine auf, die sich in allen Netzwerktypen wiederfinden. Die statischen Kernkomponenten geben KNN die räumliche Gestalt bzw. Struktur." [10]

2.2 Word Embeddings

Ein Algorithmus des Word Embeddings ist der sogenannte word2Vec-Algorithmus. Dieser Algorithmus ist ein unüberwachter Lernalgorithmus, mit ihm kann man die Beziehungen der Wörter erkennen und sie in Ähnlichkeits-Cluster unterteilen.
"Durch eine clevere Verwendung von Vektorräumen ist das Modell dann in der Lage, bestimmte Wörter durch einfache Vektorberechnungen nachzubilden, beispielsweise Knecht - Mann + Frau = Magd." [9]

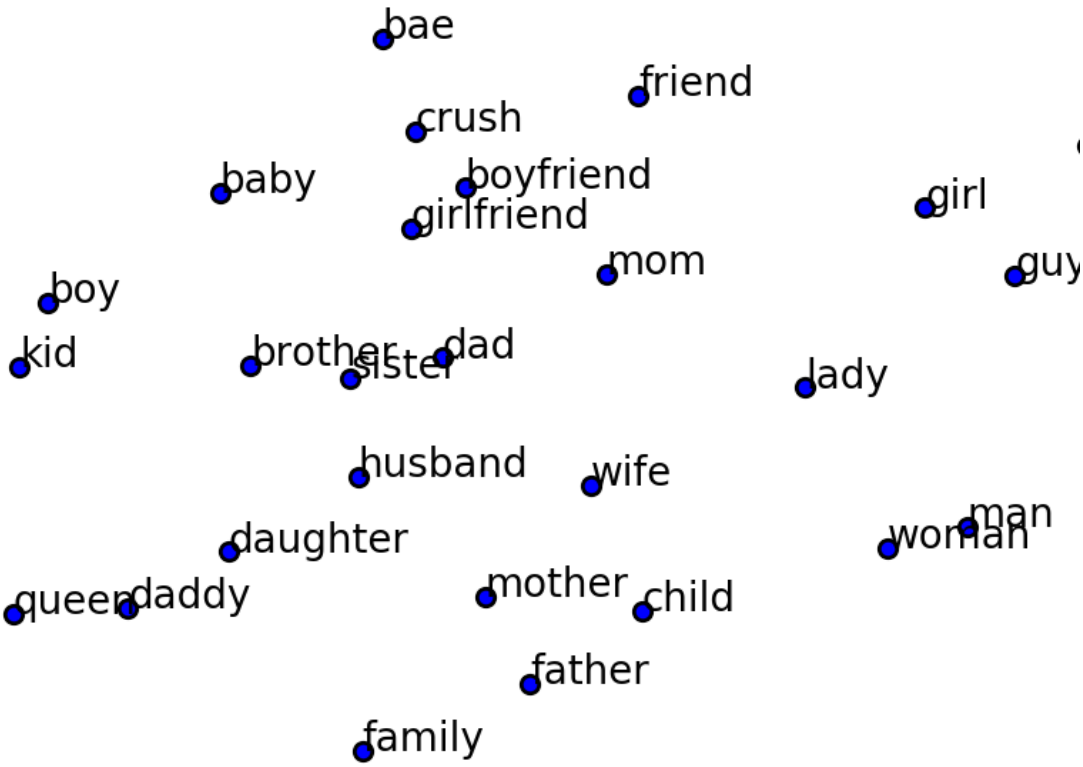

Abbildung 2.2: Darstellung eines Ähnlichkeits-Cluster [6]

Wie man im Bild sehen kann, sind die Wörter deren Bedeutung ähnlich sind nahe beieinander. Auch wenn es nicht immer mit allen Wörtern richtig ist.

2.3 Zusammenhang von Word Embeddings und KNN

Word2vec benutzt Modelle, die verwendet werden um Word Embeddings zu erstellen. Diese Modelle sind flache und beliebig-schichtige neuronale Netze, die darauf trainiert sind, sprachliche Kontexte von Wörtern zu rekonstruieren. Als Eingabe wird eine großer Korpus von Texten benutzt, daraus wird ein Vektorraum erzeugt, der typischerweise aus mehreren hundert Dimensionen besteht, wobei jedem eindeutigen Wort im Korpus ein entsprechender Vektor im Raum zugeordnet wird. Wortvektoren werden in dem Vektorraum positioniert, so dass Wörter, die gemeinsame Kontexte in dem Korpus teilen, in dem Raum nahe beieinander angeordnet sind.

3 Verwandte Arbeiten

Im Bereich des Word Embeddings gibt es einige existierende Ergebnisse. Im Folgenden werden Projekte angesprochen, die eine Inspiration für diese Arbeit war.

3.1 Efficient Estimation of Word Representations in Vector Space

In diesem Projekt werden zwei neuartige Modellarchitekturen vorgeschlagen, um kontinuierliche Vektordarstellungen von Wörtern aus sehr großen Datensätzen zu berechnen. Die Qualität wird in einer Wortähnlichkeitsaufgabe gemessen, und danach mit den bisher besten Techniken verglichen, die auf verschiedenen Arten von neuronalen Netzen beruhen. [7]

3.2 Enriching Word Vectors with Subword Information

Hier wird erklärt wie Kontinuierliche Wortdarstellungen, die auf großen unbeschrifteten Korpora trainiert werden, für bestimmte Aufgaben nützlich sind. Populäre Modelle, die solche Darstellungen lernen, ignorieren die Morphologie von Wörtern, indem sie jedem Wort einen eindeutigen Vektor zuweisen. Dies ist eine Einschränkung, deshalb wird in diesem Artikel ein neuer Ansatz vorgeschlagen. [2]

3.3 GloVe: Global Vectors for Word Representation

In diesem Artikel werden neuere Methoden zum Lernen von Vektorraumdarstellungen von Wörtern bei denen es gelungen ist, feinkörnige semantische und syntaktische Gesetzmäßigkeiten unter Verwendung von Vektorarithmetik zu erfassen erklärt, aber der Ursprung dieser Gesetzmäßigkeiten ist undurchsichtig geblieben. Desweiteren werden die Modelleigenschaften explizit, die für solche Gesetzmäßigkeiten in Wortvektoren benötigt werden, analysiert. [8]

3.4 Distributional semantics for diachronic search

Dieser Artikel beschreibt ein System, dass über verschiedene Zeiträume hinweg, nach Wortähnlichkeiten sucht. Die Strategie basiert auf Verteilungsmodellen, die aus einer chronologisch strukturierten Sprachressource gewonnen werden. Die Modelle wurden mit Hilfe von abhängigkeitsbasierten Kontexten und die Reduzierung des Vektorraums erstellt, um damit die richtigen Wortkontexte auszuwählen. [3]

3.5 Word embeddings quantify 100 years of gender and ethnic stereotypes

Hier wird erklärt, dass Word embeddings ein leistungsfähiges Machine-Learning-Framework sind, die jedes englische Wort durch einen Vektor repräsentiert kann. Und das die geometrischen Beziehungen zwischen diesen Vektoren sinnvolle semantische Beziehungen zwischen den entsprechenden Wörtern erfassen kann. Zuzüglich wird gezeigt, wie die zeitliche Dynamik der Einbettung dazu beiträgt, Stereotype und Einstellungen gegenüber Frauen und ethnischen Minderheiten im 20. und 21. Jahrhundert in den USA zu quantifizieren. [4]

3.6 Temporal Word Analogies: Identifying Lexical Replacement with Diachronic Word Embeddings

In diesem Projekt wird das Konzept zeitlicher Wortanalogien vorgestellt. Eine wohlbekannte Eigenschaft von Word Embeddings ist, dass sie in der Lage sind, traditionelle Wortanalogien durch Vektoraddition effektiv zu modellieren. Hier wird gezeigt, dass zeitliche Wortanalogien effektiv mit diachronen Word Embeddings modelliert werden können. [11]

3.7 Dynamic Word Embeddings for Evolving Semantic Discovery

Dieser Artikel bezieht sich auf die Wortentwicklung, die auf wechselnde Bedeutungen und Assoziationen von Wörtern im Laufe der Zeit hindeuten, die als ein Nebenprodukt der Evolution der menschlichen Sprache ist. Durch die Wortentwicklung können die sozialen Trends und Sprachkonstrukte über verschiedene Perioden der Menschheitsgeschichte abgeleitet werden. Traditionelle Techniken wie das Wortdarstellungslernen erfassen jedoch die sich entwickelnde Sprachstruktur und das Vokabular nicht ausreichend, deshalb wird hier ein dynamisches statistisches Modell erklärt, das zeitbewusste Wortvektorrepräsentation lernt. [13]

3.8 Evaluating the Stability of Embedding-basedWord Similarities

Hier wird erklärt wie Word Embeddings zunehmend als ein Werkzeug verwendet werden kann, um Wortverbindungen in bestimmten Korpora zu untersuchen. Es wird noch erwähnt, wie empfindlich die Abstände zwischen den nächsten Nachbarn bei kleineren Änderungen reagieren. [1]

3.9 Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change

In diesem Artikel wird erläutert wie Wörter ihre Bedeutungen im Laufe der Zeit ändern können. Des Weiteren werden verschiedene Algorithmen des Word Embeddings verwendet, wie z.B. word2vec, diese werden verwendet um statistische Gesetze der semantischen Evolution aufzudecken. [5]

4 Vorgehensweise

In diesem Kapitel wird zuerst die Aufgabenstellung erklärt, um zu erfahren was eigentlich gemacht werden soll. Danach wird noch der Weg beschrieben wie man genau vorgegangen ist, um diese Aufgabe zu lösen.

4.1 Aufgabenstellung

Gelernte Wortrepräsentationen (word embeddings) stellen Wörter basierend auf ihren Kontexten als Vektoren dar. Diese Vektoren erfassen die Bedeutung der Wörter, so dass Wörter mit ähnlicher Bedeutung auf zueinander naheliegende Vektoren abgebildet werden. Zudem erlauben diese Vektoren einfache Arithmetik, subtrahiert man vom Vektor des Wortes *king* den Vektor des Wortes *man* und addiert jenen des Wortes *woman*, erhält man einen Vektor in der Nähe des Vektors des Wortes *queen*.
Die Aufgabe ist, inwiefern sich solche Wortrepräsentationen eignen, um auf langzeitlichen Dokumentensammlungen, die Veränderung unserer Sprache sowie die Veränderung von Bedeutungen zu analysieren.

4.2 Lösung

Die Daten die für die Analyse benötigt werden, kommen aus der New York Times, beginnend mit dem Jahr 1987 und endet mit dem Jahr 2007. Zur Vorbereitung der Analyse müssen erst alle benötigten Daten nach ihrer Jahreszahl aufgeteilt werden, zuzüglich müssen Groß- und Kleinschreibung sowie die Satzzeichen entfernt werden. Diese vorverarbeiteten Daten werden in entsprechende Textdokumente gespeichert. Nachdem eine gewisse Zeit vergangen ist, werden die Wortrepräsentationen für jedes der in der Dokumentensammlung enthaltenen Jahre berechnet und sind in einer Datenbank indexiert und gespeichert worden. Dies bildet die Grundlage der Analyse. Zur Speicherung der Daten wurde MySQL verwendet, auch wenn PostgreSQL als RDBMS möglich gewesen wäre, habe ich mich für MySQL entschieden, da ich früher schon mal mit MySQL gearbeitet habe. Nachdem alles in der Datenbank gespeichert ist, wird mit Hilfe von SQL-Kommandos die Daten, nach der Euklidischen Distanz und der Kosinus-Ähnlichkeit, ausgelesen, um zu ermitteln welche Wörter zu einem gesuchten Wort zu finden sind, welche sich in der Sprache sowie in der Bedeutung verändert haben.

5 Implementierung

In diesem Kapitel wird das Datenbank-Schema gezeigt und erklärt, sowie der Quellcode der SQL-Anweisungen.

5.1 Datenbank-Schema

Es gibt in der Datenbank zwei Tabellen und zwei Sichten. Diese sind in ihren entsprechenden Dimensionen unterteilt. Die Tabellen embeddings_100 und embeddings_50 werden für die SQL-Anweisung benötigt, die die Euklidische Distanz berechnet. Die Sichten Liste_Länge_100 und Liste_Länge_50 werden benötigt um für die SQL-Anweisung der Kosinus-Ähnlichkeit die Formel zu vereinfachen, da man für die Kosinus-Ähnlichkeit die Länge braucht. Zur schnelleren Ermittlung und zur Eindeutigkeit, der Daten, da es keinen Primärschlüssel gibt, wurde noch ein Index auf die Spalten Wort, Jahr, Dimension und Vektor angelegt.

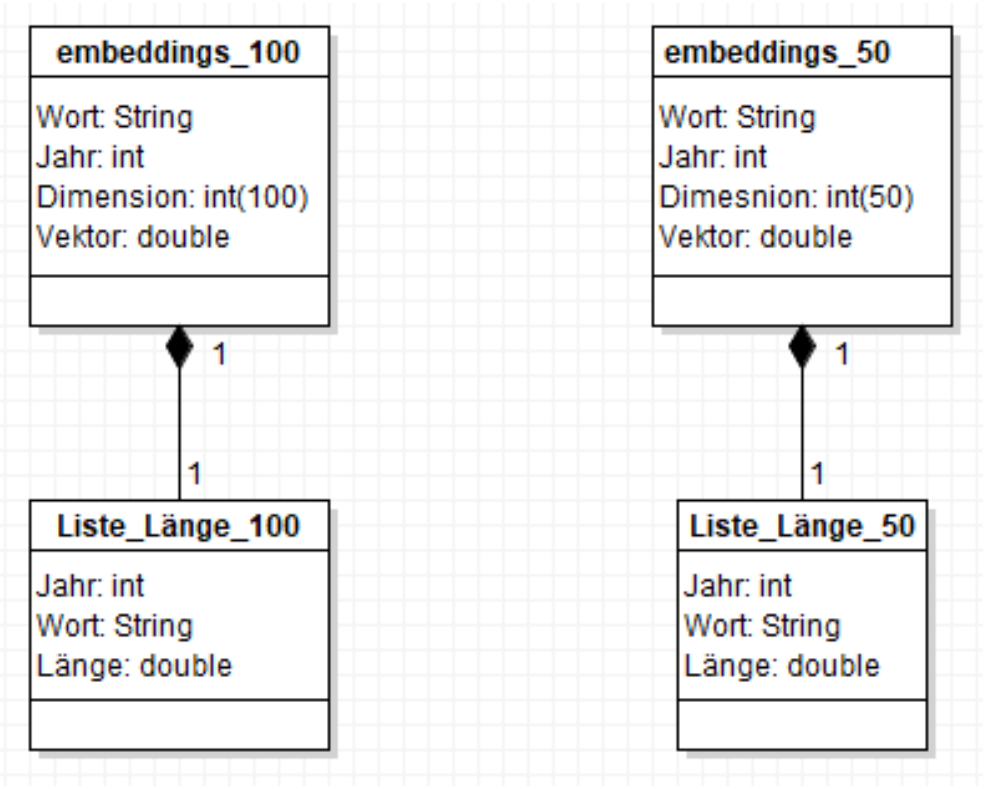

Abbildung 5.1: Struktur der Datenbank

5.2 Quellcode

5.2.1 Euklidische Disntanz

```
public String selectEuklidisch(String wort1, int jahr1, int jahr2)
    throws SQLException {
    Statement statement = connection.createStatement();
    String wort = null;
    String sim = null;
    int i = 0;
    StringBuilder stringBuilder = new StringBuilder();

    ResultSet rs = statement.executeQuery("SELECT e2.Wort, sqrt(sum(
        pow(e1.Vektor - e2.Vektor,2))) as sim" +
            " FROM " + TABLE + " e1, " + TABLE + " e2" +
            " WHERE e1.Wort='" + wort1 + "' AND e1.Jahr=" + jahr1 + "
                AND e2.Jahr=" + jahr2 +
            " AND e1.Dimension=e2.Dimension AND" +
            " (e2.Wort LIKE 'a%' OR e2.Wort LIKE 'b%' OR e2.Wort LIKE
                'c%' OR e2.Wort LIKE 'd%' OR e2.Wort LIKE 'e%' OR e2
                .Wort LIKE 'f%'" +
            " OR e2.Wort LIKE 'g%' OR e2.Wort LIKE 'h%' OR e2.Wort
                LIKE 'i%' OR e2.Wort LIKE 'j%' OR e2.Wort LIKE 'k%'
                OR e2.Wort LIKE 'l%'" +
            " OR e2.Wort LIKE 'm%' OR e2.Wort LIKE 'n%' OR e2.Wort
                LIKE 'o%' OR e2.Wort LIKE 'p%' OR e2.Wort LIKE 'q%'
                OR e2.Wort LIKE 'r%'" +
            " OR e2.Wort LIKE 's%' OR e2.Wort LIKE 't%' OR e2.Wort
                LIKE 'u%' OR e2.Wort LIKE 'v%' OR e2.Wort LIKE 'w%'
                OR e2.Wort LIKE 'x%'" +
            " OR e2.Wort LIKE 'y%' OR e2.Wort LIKE 'z%')" +
            " GROUP BY e2.Wort ORDER BY sim ASC LIMIT 1000");
    while (!rs.isLast()) {
        if (rs.next()) {
            i++;
            wort = rs.getString(1);
            sim = rs.getString(2);
            stringBuilder.append(i + ". " + wort + ", ").append(sim).
                append("\n");
        }
    }

    return stringBuilder.toString();
}
```

Dieser Code sucht, mit Hilfe des SQL-Kommandos und der Formel zur Berechnung der Euklidischen Distanz, Wörter die nah bei dem gesuchten Wort, hier *'wort1'*, existieren.

5.2.2 Kosinus-Ähnlichkeit

```
public String selectCosinus(String wort1, int jahr1, int jahr2)
    throws SQLException {
    Statement statement = connection.createStatement();
    StringBuilder stringBuilder = new StringBuilder();
    String wort = null;
    String cos = null;
    int i = 0;

    ResultSet rs = statement.executeQuery("SELECT e2.Wort, sum(e1.
        Vektor * e2.Vektor) / (l1.Laenge * l2.Laenge) as cos" +
```

```
                " FROM " + TABLE + " e1 , " + TABLE + " e2 , " + VIEW + "
                    l1 , " + VIEW + " l2" +
                " WHERE e1.Wort='" + wort1 + "' AND e1.Jahr=" + jahr1 + "
                    AND e2.Jahr=" + jahr2 +
                " AND e1.Dimension=e2.Dimension AND e1.jahr=l1.jahr AND
                    e2.jahr=l2.jahr AND l1.Wort=e1.Wort AND l2.Wort=e2.
                    Wort" +
                " AND (e2.Wort LIKE 'a%' OR e2.Wort LIKE 'b%' OR e2.Wort
                    LIKE 'c%' OR e2.Wort LIKE 'd%' OR e2.Wort LIKE 'e%'
                    OR e2.Wort LIKE 'f%'" +
                " OR e2.Wort LIKE 'g%' OR e2.Wort LIKE 'h%' OR e2.Wort
                    LIKE 'i%' OR e2.Wort LIKE 'j%' OR e2.Wort LIKE 'k%'
                    OR e2.Wort LIKE 'l%'" +
                " OR e2.Wort LIKE 'm%' OR e2.Wort LIKE 'n%' OR e2.Wort
                    LIKE 'o%' OR e2.Wort LIKE 'p%' OR e2.Wort LIKE 'q%'
                    OR e2.Wort LIKE 'r%'" +
                " OR e2.Wort LIKE 's%' OR e2.Wort LIKE 't%' OR e2.Wort
                    LIKE 'u%' OR e2.Wort LIKE 'v%' OR e2.Wort LIKE 'w%'
                    OR e2.Wort LIKE 'x%'" +
                " OR e2.Wort LIKE 'y%' OR e2.Wort LIKE 'z%')" +
                " GROUP BY e2.Wort" +
                " ORDER BY cos DESC" +
                " LIMIT 1000 ");

        while (!rs.isLast()) {
            if (rs.next()) {
                i++;
                wort = rs.getString(1);
                cos = rs.getString(2);
                stringBuilder.append(i + ". " + wort + ", ").append(cos).
                    append("\n");
            }
        }
        return stringBuilder.toString();
    }
```

Dieser Code sucht, mit Hilfe des SQL-Kommandos und der Formel zur Berechnung der Kosinus-Ähnlichkeit, Wörter die nah bei dem gesuchten Wort, hier *'wort1'*, existieren. Dazu wird zur Vereinfachung der Formel die Sichten verwendet.

5.2.3 Sichten erstellen

```
public void createView() throws SQLException {
        Statement statement = connection.createStatement();
        statement.executeUpdate("CREATE VIEW Liste_Laenge_50 ( Jahr, Wort
            , Laenge ) AS SELECT Jahr, Wort, sqrt(sum(Vektor * Vektor))
            as Laenge" +
                " FROM embeddings_50"+
                " GROUP BY Jahr, Wort");
        statement.executeUpdate("CREATE VIEW Liste_Laenge_100 ( Jahr,
            Wort, Laenge ) AS SELECT Jahr, Wort, sqrt(sum(Vektor * Vektor
            )) as Laenge" +
                " FROM embeddings_100"+
                " GROUP BY Jahr, Wort");
    }
```

Dieser Code erstellt, mit Hilfe des SQL-Kommandos, die Sichten um die Länge jedes Wortes zu bekommen.

6 Evaluation

Hier werden die Ergebnisse gezeigt und beurteilt

7 Fazit

Hier wird ein Fazit gemacht

Literatur

[1] Maria Antoniak und David Mimno. *Evaluating the Stability of Embedding-based Word Similarities*. Techn. Ber. Cornell University, 2018.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin und Tomas Mikolov. *Enriching Word Vectors with Subword Information*. Techn. Ber. Facebook AI Research, Juni 2017.

[3] Pablo Gamallo, Iván Rodríguez-Torres und Marcos Garcia. „Distributional semantics for diachronic search“. In: *Elsevier* (2017).

[4] Nikhil Garg, Londa Schiebinger, Dan Jurafsky und James Zoue. „Word embeddings quantify 100 years of gender and ethnic stereotypes“. In: *PNAS* (2018).

[5] William L. Hamilton, Jure Leskovec und Dan Jurafsky. *Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change*. Techn. Ber. Department of Computer Science, Stanford University, Sep. 2016.

[6] Vukosi Marivate. *A dive into Word2Vec + TSNE for visualisation*. 2017. URL: http://www.vima.co.za/2017/03/21/a-dive-into-word2vec-tsne-for-visualisation/.

[7] Tomas Mikolov, Kai Chen, Greg Corrado und Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. Techn. Ber. Google Inc., 2013.

[8] Jeffrey Pennington, Richard Socher und Christopher D. Manning. *GloVe: Global Vectors for Word Representation*. Techn. Ber. Computer Science Department, Stanford University, 2014.

[9] Sebastian Raschka. *Machine Learning mit Python Das Praxis-Handbuch für Data Sience, Predictive Analytics und Deep Learning*. mitp Verlags GmbH & Co. KG, 2017. Kap. 8, S. 254.

[10] Stefan Strecker. *Künstliche Neuronale Netze : Aufbau und Funktionsweise*. ger. 1997. URL: http://geb.uni-giessen.de/geb/volltexte/2004/1697.

[11] Terrence Szymanski. *Temporal Word Analogies: Identifying Lexical Replacement with Diachronic Word Embeddings*. Techn. Ber. Insight Centre for Data Analytics University College Dublin, 2017.

[12] Rolf Todesco. *Hyper-Lexikon: Neuronales Netzwerk*. 2018. URL: https://www.hyperkommunikation.ch/lexikon/neuronales_netz.htm.

[13] Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao und Hui Xiong. *Dynamic Word Embeddings for Evolving Semantic Discovery*. Techn. Ber. Rutgers University, Technicolor Research und Amazon, 2018.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

KNN Künstliches neuronales Netzwerk

RDBMS Relational Database Management System

Kolophon

Dieses Dokument wurde mit der LaTeX-Vorlage für Abschlussarbeiten an der htw saar im Bereich Informatik/Mechatronik-Sensortechnik erstellt (Version 2.1). Die Vorlage wurde von Yves Hary und André Miede entwickelt (mit freundlicher Unterstützung von Thomas Kretschmer, Helmut G. Folz und Martina Lehser). Daten: (F)10.95 – (B)426.79135pt – (H)688.5567pt