ZHAW
Zürcher Hochschule für angewandte Wissenschaften

Projektarbeit

HS 2016

Domänenübergreifende Sentiment-Analyse mit Deep Convolutional Neural Networks

Autoren:
Dirk von Grünigen
Martin Weilenmann

Betreuer:
Dr. Mark Cieliebak
Dr. Stephan Neuhaus
Jan Deriu

8. Januar 2017

Zurich University
of Applied Sciences

School of Engineering

DECLARATION OF ORIGINALITY

Project Work at the School of Engineering

By submitting this project work, the undersigned student confirm that this work is his/her own work and was written without the help of a third party. (Group works: the performance of the other group members are not considered as third party).

The student declares that all sources in the text (including Internet pages) and appendices have been correctly disclosed. This means that there has been no plagiarism, i.e. no sections of the project work have been partially or wholly taken from other texts and represented as the student's own work or included without being correctly referenced.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

City, Date: Signature:

....

.....

.....

The original signed and dated document (no copies) must be included after the title sheet in the ZHAW version of all project works submitted.

Zusammenfassung

In der vorliegenden Arbeit werden die Möglichkeiten von Crossdomain Sentiment-Analyse mithilfe von Convolutional Neural Networks untersucht. Dabei ist das Ziel zu eruieren, inwiefern sich ein einzelner Sentiment-Klassifizierer auf verschiedenen Domänen verhält und ob es Sinn macht, Datensätze mehrerer Domänen in Kombination zu verwenden.

Als Erstes wird analysiert wie stark der Einfluss unterschiedlicher Word-Embeddings und Distant-Phasen auf einen Sentiment-Klassifizierer im Generellen ist und ob ein Zusammenhang zwischen den einzelnen Domänen und der besten Kombination an Word-Embeddings und Distant-Phasen existiert. Die Resultate zeigen, dass nicht anhand eines generellen Schemas entschieden werden kann, welches die optimale Wahl ist. Dies muss im Einzelfall analysiert werden.

Im zweiten Teil wird untersucht, wie sich verschiedene Domänen im gegebenen Kontext verhalten. Dabei werden verschiedene Sentiment-Klassfizierer auf unterschiedlichen Domänen trainiert und dann auf allen anderen Domänen evaluiert. Die Resultate zeigen, dass es keine einzelnen Domänen gibt, welche sich besser eignet, um einen generalistischen Sentiment-Klassifizierer zu trainieren. Die Varianz der Resultate ist erstaunlich hoch, was nicht nur mit den unterschiedlichen Domänen, sondern auch mit den eingeführten Eigenschaften der Eindeutigkeit und Konzentration der Sentiments in Texten zusammenhängt. Die Generalisierung von einer Domäne zu einer anderen ist meistens nicht ohne weiteres möglich.

Im nächsten Teil wird untersucht, ob es sich lohnt, Daten aus unterschiedlichen Domänen für das Training eines Sentiment-Klassifizierers zu verwenden. Dafür werden sogenannte “Augmentation” Experimente durchgeführt: Bei diesen werden Daten aus mehreren Domänen in verschiedenen Verhältnissen durchmischt und danach wird der resultierende Sentiment-Klassifizierer auf einzelnen Domänen evaluiert. Die Resultate der Experimente zeigen, dass sich dieses Vorgehen von Vorteil ist, sofern zu wenige annotierte Daten der Zieldomäne zur Verfügung stehen.

Zuletzt wird untersucht, wie die Performanz eines generalistisch ausgelegten Sentiment-Klassifiziers auf einzelnen Domänen ist. Dafür wurden mehrere dieser Klassifizierer auf sogenannten “Ablation” Datensätzen trainiert. Dabei kann festgestellt werden, dass die Performanz eines generalistischen Sentiment-Klassifiziers auf einzelnen Domänen nicht bedeutend schlechter ist und für gewisse Anwendungsfälle durchaus eine Option darstellt.

Abstract

In the following work we are going to investigate the potential of crossdomain sentiment-classification using convolutional neural networks. The main point of this research is to explore the topic of combining data from multiple sources to train such sentiment-classifiers.

In the first part, we analyse how well the sentiment-classifiers for different domains work together with multiple combinations of distant-phases and word-embeddings. It is shown, that there's no a priori answer on which combination is the best upfront; it has to be evaluated on each domain separately.

The next part focuses on how well specialized sentiment-classifiers work on different domains. For this purpose, multiple sentiment-classifiers are trained on a single domain and evaluated on all others. The results show, that there is no single best domain to train a generalized classifier and that the performance deteriorate if different domains are used for training and testing. Further, we determined that ambiguity in sentiments and the length of texts has an impact on the performance of the resulting classifier.

We conducted augmentation experiments to investigate the issue of combining data from multiple domains to train a sentiment-classifier. This means that different combinations of data from multiple domains is used to train the classifiers. The results of the experiments show, such an approach is only favorable if there is too little annotated data of the target domain.

As the last point, we evaluate the generalization performance of a sentiment-classifier. For this purpose, we used ablation datasets, which are combinations of all domains except for the target domain to train the classifiers. The results show that the performance of a generalized sentiment-classifier is only slightly worse than the one of specialized classifers and can such a procedure can even be useful in certain use-cases.

Inhaltsverzeichnis

1. Einführung

Seit den frühen 2000er Jahren befindet sich das Internet stark im Wandel: Aus einer grösstenteils statischen Sammlung von Informationen entwickelte sich im Lauf der Zeit eine dyamische, interaktive Plattform, auf welcher Menschen aus der ganzen Welt Inhalte erstellen und teilen können. Dieser Wandel wurde durch das Aufkommen von sozialen Netzwerken, wie zum Beispiel Facebook oder Twitter, nochmals beschleunigt. Durch diese massive Flut an benutzererstellten Inhalten sind neue Bedürfnisse entstanden, diese automatisch zu analysieren [19].

Bei der Sentiment-Analyse geht es darum, einen ganzen Text oder einen einzelnen Satz automatisch nach dessen Polarität, zum Beispiel positiv oder negativ, zu klassifizieren. Diese Klassifizierung kann dazu verwendet werden, die generelle Stimmung von Benutzer bezüglich eines Themas zu analysieren. Als Beispiel könnte man hier aufführen, dass ein Filmproduzent überwachen und analysieren möchte, wie gut sein neuester Film bei den Zuschauern, welche sich zum Beispiel auf Twitter oder Facebook dazu äussern, ankommt.

Die menschliche Sprache enthält allerdings viele Konstrukte, welche es erschweren diese Aufgabe zu automatisieren. Dazu zählen Sprachkonstrukte wie Synonyme oder Homonyme, also Wörter welche zwar gleich geschrieben werden aber unterschiedliche Bedeutungen haben (z.B. die Bank). Dann kommen noch sozialgesellschaftliche Phänomene wie Sarkasmus oder Zynismus dazu, welche nur mit einem genügend grossen Kontextwissen richtig verstanden und interpretiert werden können.

Der Schwerpunkt dieser Arbeit liegt darin zu analysieren, inwiefern sich Sentiment-Analyse mithilfe von Daten aus mehreren Domänen durchführen lässt. Dafür bauen wir auf Vorarbeiten von Deriu et al. [5] auf und verwenden das dort vorgestellte *Convolutional Neural Network*. Die Motivation bei der Verwendung von Daten aus mehreren Domänen ist, dass in der Praxis oft nicht genug grosse Datensätze vorhanden sind um mit solchen Systemen eine zufriedenstellende Performanz zu erreichen. Wir möchten überprüfen, ob wir dieses Problem mit Daten aus mehreren Domänen lösen können. Ausserdem ist es interessant zu analysieren, inwiefern die unterschiedlichen Textsorten und Domänen zusammenspielen und welche Konsequenzen das gleichzeitige Verwenden von Daten aus mehreren Domänen für den resultierenden Klassifizierer hat.

Im Laufe der Arbeit werden wir deshalb Experimente durchführen, um die folgenden Fragen zu beantworten:

- Inwiefern beeinflusst die Kombination verschiedener Word-Embeddings und Distant-

Phases die Performanz eines Sentiment-Klassifizierers? Gibt es eine Korrelation zwischen Textsorten und der besten Kombination?

- Wie gut ist die Performanz eines für eine Domäne spezialisierten Sentiment-Klassifizierers auf anderen Domänen?

- Ist es sinnvoll Daten aus verschiedenen Domänen zu kombinieren? Wenn ja, gibt es Kriterien, welche erfüllt werden müssen?

- Gibt es eine Möglichkeit einen generalistischen Sentiment-Klassifizierer zu erstellen, welcher auf mehreren Domänen gute Ergebnisse erzielt?

Die vorliegende Arbeit ist wie folgt strukturiert: Im Kapitel 2 werden die bisherigen Erkenntnisse im Bereich der Sentiment-Analyse zusammengefasst. Im Kapitel 3 werden die Grundlagen erläutert, mit deren Hilfe ein Sentiment-Klassifizierer arbeitet. In den Kapitel 4 und 5 werden die verwendeten Daten und das Vorgehen zum Durchführen der Experimente erklärt. Auf die Experimente selber sowie deren Resultate wird im Kapitel 6 eingegangen. Die Resultate der Experimente und die daraus gewonnenen Erkenntnisse werden im letzten Kapitel 7 zusammengefasst. Ausserdem wird auf mögliche Anknüpfungspunkte an diese Arbeit eingegangen.

2. Themenbezogene Arbeiten

Der Bereich der Sentiment-Analyse hat seit den ersten Arbeiten von Pang et al. [20] grosses öffentliches Interesse erlangt. Im Verlauf der letzten Jahre wurden in diesem Bereich viele Algorithmen und Techniken zur Sentiment-Klassifizierung entwickelt: Von *unsupervised* [29], über *semi-supervised* [8], [22] bis hin zu *supervised* [31] Verfahren wurden verwendet und es konnte gezeigt werden, dass die Sentiment-Klassifizierung mit maschinellen Lernverfahren durchaus erfolgversprechend ist. In den letzten Jahren wurden nun vermehrt Neuronale Netzwerke und *Deep Learning* Verfahren in der Sentiment-Klassifizierung eingesetzt [27], [34]. Dabei wurden vor allem *Convolutional Neural Networks* verwendet, nachdem diese im Bereich der Bildverarbeitung [14], [25] beachtliche Erfolge verzeichnen konnten. Diverse weitere Arbeiten haben das Potenzial von *Convolutional Neural Networks* im Bereich der Sentiment-Analyse bestätigt [5], [11], [12], [21]. Oft werden diese Lernverfahren mit *unsupervised* Verfahren [10], [15] kombiniert, um die Texte in eine für die Maschine verarbeitbare Repräsentation zu bringen.

Alle oben erläuterten Arbeiten beschäftigten sich allerdings mit der Klassifizierung von Texten aus der gleichen Domäne, mit der bereits das Training des Klassifizierers erfolgte. Der Bereich der Crossdomain Sentiment-Analyse ist deutlich weniger tief erforscht, wie die Sentiment-Analyse insgesamt. Es existieren aber einige Ansätze: Begonnen damit haben Blitzer et al. [2], welche versucht haben, die Problematik mit dem sogenannten *Structural Correspondence Learning* Algorithmus zu erfassen. Dabei werden die Daten der Ziel- und Fremddomäne verwendet, um eine Entsprechung (engl. correspondence) zwischen den unterschiedlichen Wörtern der einzelnen Domänen herzuleiten. Als weiteres Beispiel lässt sich das *Spectral Feature Alignment* Verfahren von Pan et al. [18] erwähnen; bei diesem wird versucht, domänenspezifische Wörter anhand deren Auftreten in Kombination mit domänenunabhängigen Wörtern, in einem bipartiten Graphen abzubilden. Anschliessend wird mit mithilfe der domänenunabhänigen Wörter die unterschiedliche Verwendung der domänenspezifischen Wörter in den verschiedenen Domänen abgebildet. Des Weiteren wurden von Bollegala et al. auch Anstrengungen unternommen, sentiment-sensitive Thesauri und Word-Embeddings zu entwickeln, um diese dann für die Crossdomain Sentiment-Klassifizierung einzusetzen [3], [4]. Die meisten der erwähnten Verfahren versuchen dabei mittels eines Algorithmus oder zusätzlichen Datenstrukturen, die Unterschiede zwischen der Ziel- und Fremddomäne auszugleichen, um so mit einem Klassifizierer Texte aus fremden Domänen richtig klassifizieren zu können.

Im Rahmen dieser Arbeit werden wir auf zusätzliche Verfahren und Algorithmen verzichten. Stattdessen werden wir versuchen mittels der Kombination von Datensätzen aus mehreren Domänen eine Verbesserung eines spezialisierten Klassifiziers zu erreichen. Dies wurde in ähnlicher Weise bereits von Li et al. analysiert, allerdings wurde dabei ein

Ensemble aus mehreren spezialisierten Support Vector Machines verwendet [13]. Wir hingegen werden die Verwendung eines einzelnen Klassifizierers, basierend auf einem Convolutional Neural Network, im Kontext von Crossdomain Sentiment-Analyse untersuchen. Ausserdem soll evaluiert werden, inwiefern sich eine Kombination von Daten aus mehreren Domänen verwenden lassen, um einen Klassfizierer zu erstellen, welcher generalistisch funktioniert. Also ein Klassifizierer, welcher nicht nur auf eine einzelne Domäne spezialisiert ist und trotzdem sinnvolle Resultate liefert, wenn er auf mehreren Domänen evaluiert wird.

3. Grundlagen

Im ersten Teil dieses Kapitels werden die theoretischen Grundlagen, welche in den folgenden Kapitel dieser Arbeit Verwendung finden, erläutert. Dabei werden zuerst grundlegende Begriffe wie Sentiment und Domäne definiert. Danach folgt eine Einführung in das maschinelle Lernen mit Neuronalen Netzwerken (NN), sowie der spezifischen Variante Convolutional Neural Network (CNN). Anschliessend wird das 3-stufige Lernverfahren, mit welchem die CNNs trainiert werden, erläutert.

Die folgenden Erläuterungen müssen im Kontext von Klassifizierungsproblemen betrachtet werden, zu welchen die Sentiment-Analyse gehört. Natürlich lassen sich NNs und CNNs auch für weitere Aufgaben (z.B. Dimensionalitätsreduktion, Vorraussagen von Werten) einsetzen, wobei diese Themen nicht Gegenstand dieser Arbeit sind und deswegen nicht mit einbezogen werden.

3.1. Definitionen

Sentiment Der *Sentiment* beschreibt das subjektive Empfinden, welches bei einer Person beim Lesen eines Textes ausgelöst wird. Ein Sentiment kann für einzelne Sätze, Abschnitte oder ganze Texte bestimmt werden. Im Rahmen dieser Arbeit werden die drei Sentiments *positiv*, *neutral* und *negativ* verwendet (vgl. Beispiele in Tabelle 3.1). Je nach Bedürfnis lässt sich diese Skala noch weiter verfeinern (z.B. zusätzlich mit "eher negativ" oder "eher positiv").

Sentiment	Beispiel
positiv	Ich liebe Donuts!
neutral	Dieser Stein ist grau.
negativ	Die Auflösung dieser Handy-Kamera ist sehr schlecht.

Tabelle 3.1.: Beispieltexte für die drei verschiedenen Sentiments

Der Sentiment eines Textes ist nicht eindeutig bestimmbar, da dieser vom subjektiven Empfinden der annotierenden Person abhängig ist. Der resultierende Sentiment ist demnach davon abhängig wer ihn beurteilt.

Im Laufe der Arbeit werden die Namen der einzelnen Sentiments oft in abgekürzter Form verwendet (z.B. bei der Verwendung für F1-Scores). Darum werden wir für die Namen

der Sentiments folgende Abkürzungen verwenden: *pos* für positiv, *neu* für neutral und *neg* für negativ.

Domäne Texte, welche die gleichen semantischen Konzepte beschreiben und aus gleichen oder ähnlichen Quellen stammen, werden zu einer Domäne zusammengefasst. Als Beispiele lassen sich hier Produktbewertungen, Tweets oder auch Nachrichtentexte erwähnen.

Crossdomain Mit *Crossdomain* (dt. domänenübergreifend) ist die Verwendung von Trainings- und Testdaten aus mehreren verschiedenen Domänen zum Training und/oder zur Validierung eines Sentiment-Klassifiziers gemeint.

Für die Durchführung dieser Arbeit sind keine formale Definitionen für *Sentiment* und *Domäne* nötig. Solche können allerdings in diversen anderen Arbeiten, unter anderem in [18] und [3], nachgeschlagen werden.

3.2. Neuronale Netzwerke

Neuronale Netzwerke, im Folgenden NN genannt, sind ein Modell des maschinellen Lernens, welches biologisch motiviert ist und sich lose an der Funktionsweise des menschlichen Gehirns orientiert. Im Folgenden wird auf die einzelnen Komponenten eines NN eingegangen.

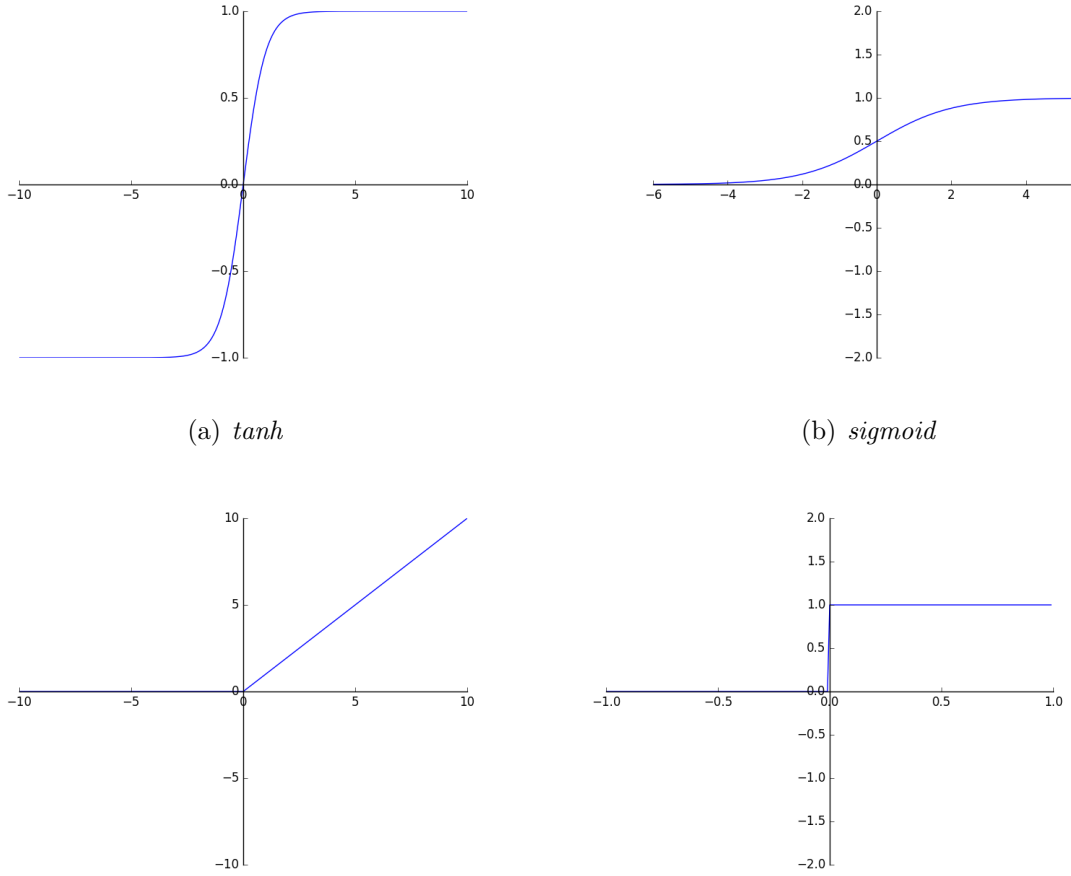
Neuron Ein NN besteht aus *Neuronen*, manchmal auch Perceptronen genannt. Diese bilden eine mathematische Funktion ab und sind die Grundbausteine eines jeden NN.

Diese Neuronen nehmen n Eingangswerte $\mathbf{x} = (x_0, x_1, \ldots, x_n)$ entgegen. Jedem dieser Eingangswerte x_n wird ein Gewicht w_n aus der Menge $\mathbf{w} = (w_0, w_1, \ldots, w_n)$ zugewiesen. Der Eingangswert x_0 wird fast immer mit 1 initialisiert und als Bias-Wert bezeichnet; dieser wird normalerweise nicht verändert. Dadurch wird die modellierte Funktion affin anstatt linear und die Anzahl der möglichen Funktionen, welche abgebildet werden können, steigt. Mithilfe der Eingangswerte $\mathbf{x}$, den zugehörigen Gewichten $\mathbf{w}$ und der Aktivierungsfunktion φ wird die Ausgabe o des Neurons, auch Aktivierung genannt, berechnet:

$$o = \varphi(\mathbf{w} \cdot \mathbf{x}) = \varphi\left(\sum_{i=0}^{n} w_i x_i\right) \tag{3.1}$$

Die Aktivierungsfunktion φ ist dafür zuständig den aus der Berechnung resultierenden Wert o in eine vordefinierte Spanne von Werten zu bringen. Beispiele für verwendete Ak-

tivierungsfunktionen sind $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, $\text{relu}(x) = \max\{0, x\}$, die *sigmoid* Funktion $s(x) = \frac{1}{1+e^{-x}}$ oder auch die *binary step* Funktion $b(x) = \left\{ \begin{array}{ll} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{array} \right\}$.



(a) *tanh* (b) *sigmoid*

(c) *relu* (d) *binary step*

Abbildung 3.1.: Plots der Aktivierungsfunktionen *tanh*, *sigmoid*, *relu* und *binary step*.

Schicht Mithilfe der zuvor erwähnten Neuronen werden die einzelnen Schichten eines NN aufgebaut. Eine Schicht setzt sich dabei aus mehreren Neuronen zusammen. Um ein vollständiges NN zu erhalten werden mehrere dieser Schichten hintereinander gereiht. Dabei gibt es drei essentielle Schichten, welche fast jedes NN hat: Eine *Eingabeschicht*, eine oder mehrere *Hidden-Schichten* oder *verborgene Schichten* und eine *Ausgabeschicht*; vgl. Abbildung 3.2.

Über die Eingabeschicht treten die Eingabedaten $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ in das Netzwerk ein. Der Bias-Wert x_0 wird dabei im Normalfall wieder auf 1 gesetzt. Pro Schicht gibt es in der Regel nur einen Bias-Wert bzw. nur ein Bias-Gewicht für alle Neuronen. Die Werte in $\mathbf{x}$ werden dann an die Neuronen der Hidden-Schicht weitergereicht und diese berechnen damit ihre Aktivierungen o_{ln}, wobei l für die Schicht und n für das entsprechende Neuron in dieser Schicht steht. Die Aktivierungen werden dann von der Hidden-Schicht

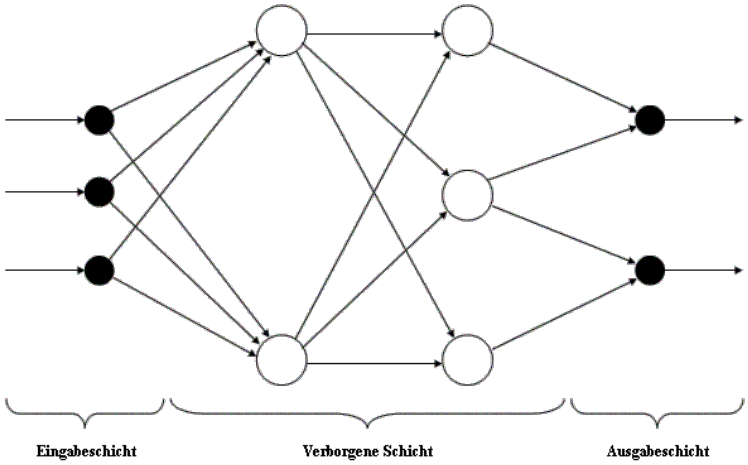


Abbildung 3.2.: Schematische Darstellung eines sehr einfachen NN. Links die Eingabeschicht, in der Mitte zwei Hidden-Schichten, rechts die Ausgabeschicht.

zur Ausgabeschicht weitergereicht, wo die sich darin enthaltenen Neuronen wiederum ihre Aktivierungen $o_{21}, o_{22}, \ldots, o_{2n}$ berechnen. Die Eingabewerte der einzelnen Neuronen der Ausgabeschicht sind dabei die Aktivierungen aller Neuronen der vorhergehenden Schicht. Eine solche Schicht wird auch als *fully-connected* Schicht bezeichnet, da die Eingabewerte jedes Neurons die Aktivierung aller Neuronen in der vorhergehenden Schicht sind. Die Ausgabe des NNs sind die Aktivierungen $o_{31}, o_{32}, \ldots, o_{3m}$ der Neuronen in der Ausgabeschicht, wobei m für die Anzahl der Neuronen in der Ausgabeschicht steht.

Dieser Vorgang des "Vorwärtsrechnens"wird als Vorwärtspropagierung bezeichnet. Bei einem NN mit mehr als einer Hidden-Schicht spricht man auch von einem *Deep Neural Network*. Die Anzahl der Schichten und Neuronen in einem NN hängt stark von der konkreten Problemstellung ab.

Backpropagation mit Gradient-Descent Wie bei fast allen Modellen des maschinellen Lernens lernt das NN mittels dem Optimieren einer Fehlerfunktion E. Als Fehlerfunktion kann jede Funktion dienen, mit welcher der Aussagefehler des Netzwerks quanitifiziert werden kann. Als Beispiele können hier die mittlere quadratische Abweichung $\text{MSE}(y_{true}, y_{pred}) = \frac{1}{n}\sum_{i=0}^{n}(y_{pred} - y_{true})^2$ oder auch die in dieser Arbeit verwendete Categorical Cross-Entropy $H(p,q) = -\sum xp(x)\log(q(x))$ aufgeführt werden. Das Optimieren dieser Fehlerfunktion wird meistens mittels der *Backpropagation*-Method in Verbindung mit dem *Gradient-Descent* Algorithmus durchgeführt. Der Algorithmus durchläuft dabei die folgenden drei Schritte:

1. Die Eingabewerte in das NN einführen und die Vorwärtspropagierung durchführen.

2. Die Ausgabewerte des NN werden mit dem erwarteten Resultat verglichen. Die Differenz der Ausgabewerte von den erwarteten Werten wird als Aussagefehler des NN bezeichnet.

3. Der Aussagefehler wird nun mittels Gradient-Descent durch das NN zurückpropagiert. Dabei werden die Gewichte der Eingänge aller Neuronen abhängig von ihrem Einfluss auf die berechneten Aussagefehler angepasst.

Die Berechnung des Einflusses eines einzelnen Gewichtes auf den Aussagefehler wird mittels dem Auswerten der partiellen Ableitung der Fehlerfunktion E bezüglich dem entsprechenden Gewicht festgestellt (vgl. Gleichung 3.2). Danach wird das Gewicht neu berechnet, indem der Wert der partiellen Ableitung mit der Lernrate η multipliziert vom aktuellen Gewicht subtrahiert wird. Die Lernrate gibt dabei an, wie stark sich ein einzelner Durchlauf von Backpropagation auf ein Gewicht auswirken soll.

Der neue Wert für das Gewicht i in der Schicht l wird bei gegebener Lernrate η und Fehlerfunktion E also wie folgt berechnet:

$$w_{li} \coloneqq w_{li} - \eta \frac{\delta E(\mathbf{w})}{\delta w_{li}} \tag{3.2}$$

Um alle Gewichte in vektorisierter Form zu verändern, wird der Gradient der Fehlerfunktion E bezüglich aller Gewichte $\mathbf{w}$ verwendet:

$$\mathbf{w} \coloneqq \mathbf{w} - \eta(\nabla_{\mathbf{w}} E(\mathbf{w})) \tag{3.3}$$

Gradient-Descent hat den Nachteil, dass der Erfolg des Algorithmus stark von der gewählten Lernrate η abhängt. Bei einer zu grossen Lernrate wird das Optimum möglicherweise übersprungen oder der Wert der Fehlerfunktion divergiert sogar; bei einer zu kleinen Lernrate dauert es sehr lange, bis das Optimum der Fehlerfunktion gefunden wird (vgl. Abbildung 3.3). Darum wird im Rahmen dieser Arbeit die weiterentwickelte Variante *AdaDelta* [32] verwendet. AdaDelta verwendet für jedes Gewicht eine eigene Lernrate, welche während des Ablaufs des Algorithmus automatisch in Abhängigkeit der vergangenen Durchläufe angepasst wird. Dies hat den grossen Vorteil, dass die Lernrate nicht mehr manuell definiert werden muss, sondern sich über die Zeit so anpasst, dass die Lernraten jederzeit sinnvolle Werte besitzen.

3.3. Convolutional Neural Network

Convolutional Neural Networks (CNN) sind eine spezielle Form der zuvor beschriebenen NN. Der grundlegende Unterschied besteht darin, dass die einzelnen Schichten nicht fully-connected sind, sondern mittels Filter über lokale Konnektivität versucht wird, Muster zu erkennen. Im Folgenden werden die einzelnen Komponenten des in dieser Arbeit verwendeten CNNs erläutert.

[1] http://cs231n.github.io/assets/nn3/learningrates.jpeg

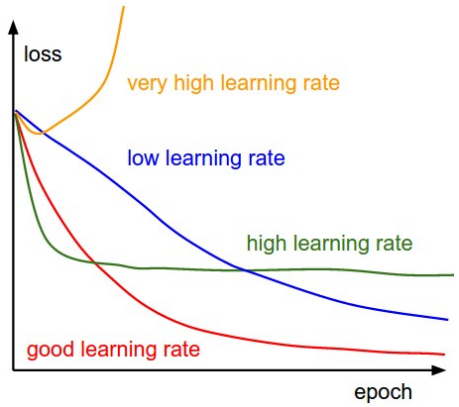


Abbildung 3.3.: Beispielhafte Gegenüberstellung verschiedener Lernraten.[1]

Filter Anstatt Neuronen, wie im klassischen Neuronalen Netzwerk, lernt eine CNN über die sogenannten Filter. Dabei handelt es sich um $n \times m$-Matrizen, welche Gewichte enthalten, analog zu den Gewichten der eingehenden Verbindungen bei Neuronen. Dabei werden die Filter-Matrizen über die Eingabedaten bewegt und es wird jeweils das innere Produkt der aktuellen Filter-Matrix, und dem aktuellen Ausschnitt der Eingabedaten berechnet, auf welchem der Filter positioniert ist. Das Bewegungsmuster des Filters wird *Stride* (dt. durchschreiten) genannt. Im Rahmen dieser Arbeit wird ein eindimensionaler Stride entlang der Satz-Matrix verwendet. Es ist aber durchaus möglich, mehrdimensionale Convolution-Operationen mit entsprechenden Strides zu definieren.

Die durch diese Convolution-Operation resultierenden Werte werden in der Form einer Resultat-Matrix, der sogenannten *Feature Map*, zwischengespeichert. Diese stellen also das Äquivalent zur Aktivierung einer Schicht in einem NN dar. Dabei werden alle resultierenden Feature-Maps einfach hintereinander gereiht und dienen als Eingabewerte für die nächste Schicht.

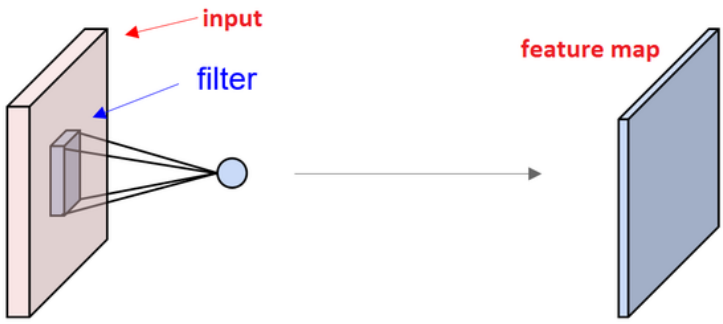


Abbildung 3.4.: Schematische Darstellung der Funktionsweise von Filter und Feature-Map innerhalb eines CNNs.[2]

Eine Schicht, welche diese Art von Berechnung verwendet, wird Convolutional (dt. sich falltend) Schicht genannt.

[2]https://www.quora.com/What-is-meant-by-feature-maps-in-convolutional-neural-networks

Max-Pooling Schicht In der *Max-Pooling* Schicht, manchmal auch *Subsampling* Schicht genannt, wird ein Grossteil der in den Feature-Maps vorhandenen Informationen verworfen. Dabei wird ein Fenster über die resultierenden Feature-Maps der vorhergehenden Schicht bewegt und der jeweils maximale Wert des entsprechenden Ausschnitts der aktuellen Feature-Map als Wert in die gepoolte Repräsentation übernommen. Das Fenster, welches über die Feature-Map bewegt wird, ist definiert durch die Dimensionalität des Fensters und dem Stride. Der Stride steht für das Bewegunsmuster des Fensters analog zum Filter. Die gepoolten Repräsentationen der Feature-Maps dienen dann als Eingabewerte für die nächste Schicht.

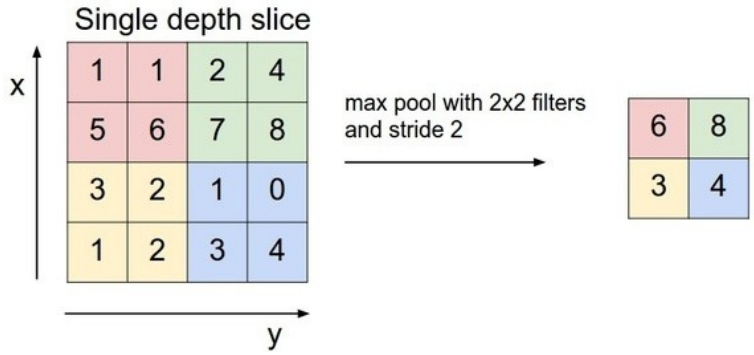


Abbildung 3.5.: Schematische Funktionsweise der Max-Pooling Schicht

Die Max-Pooling Schicht erfüllt zwei Aufgaben: Einerseits reduziert sie die Dimensionalität der zu verarbeitenden Daten, was Rechenzeit spart. Andererseits verwirft sie "unwichtige" Informationen indem sie nur die maximalen Werte der zuletzt berechneten Aktivierungen behält.

Convolutional+Max-Pooling Schicht Die Convolutional und Max-Pooling Schichten bilden die Grundbausteine der meisten CNN. Diese werden dabei mehrfach hintereinander gereiht. Das im Rahmen dieser Arbeit vewendete CNN hat beispielsweise zwei solcher aufeinanderfolgenden Convolutional+Max-Pooling Schichten.

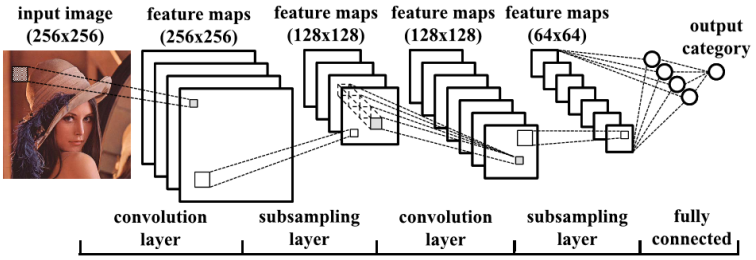


Abbildung 3.6.: Beispielhafte Darstellung eines kompletten CNN mit allen Schichten.[3]

Ausgabeschicht Die Ausgabeschicht des CNN ist gleich aufgebaut wie die eines "traditionellen" NN. Dabei wird eine fully-connected Schicht verwendet, bei welcher jedes

[3]https://www.semanticscholar.org/paper/f5f1beada9e269b2a7faed8dfe936919ac0c2397

Neuronen als Eingabewerte mit allen Aktivierungen der letzten Max-Pooling Schicht verknüpft sind. Die Anzahl Neuronen in der Ausgabeschicht entspricht dabei der Anzahl zu unterscheidenden Klassen. In dieser Arbeit entsprechen die drei verschiedenen Sentiments *neutral*, *negativ* und *positiv* den zu unterscheidenden Klassen.

3.4. 3-Phasen Lernen

Das Training der CNNs wird mithilfe des 3-stufigen Lernverfahren von Severyn et. al. [21] durchgeführt. Im Folgenden werden die einzelnen Schritte im Detail erläutert.

Word-Embeddings und Satz-Matrix Zuerst werden mithilfe von *word2vec* [15] und einem grossen Text-Corpus Word-Embeddings generiert. Dabei werden die gegebenen Wörter eines Vokabulars v so in einen reellen Vektorraum $\mathbb{R}^d$ eingebettet, dass die semantischen Beziehungen innerhalb der Wörter erhalten bleiben. Dies wird am Beispiel in Abbildung 3.7 ersichtlich: Die Wortvektoren für "Man" and "Woman" stehen in gleicher Weise zueinander wie die Wortvektoren "Uncle" zu "Aunt" bzw. "King" zu "Queen".

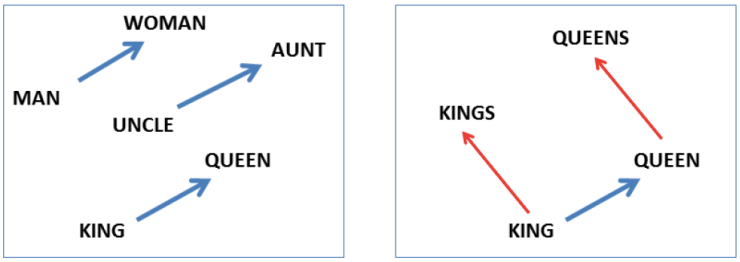


Abbildung 3.7.: Beispiel für semantische Beziehung von Wort-Vektoren. Die Wort-Vektoren für "Uncle" zu "Aunt" stehen in der gleichen Weise zueinander wie die Wort-Vektoren "King" zu "Queen"

Mithilfe der Word-Embeddings kann ein gegebener Satz nun als Konkatenation der Wort-Vektoren der einzelnen Wörter aufgefasst werden. Das bedeutet, dass ein Satz als $d \times n$ Matrix dargestellt werden kann, wenn n die Anzahl Wörter des Satzes darstellt und d die Anzahl Dimensionen der Word-Embeddings ist. Die i-te Spalte in der resultierenden Satz-Matrix entspricht dann dem Wort-Vektor für das i-te Wort im abzubildenden Satz.

Distant-Supervised Phase In einem zweiten Schritt wird die sogenannte *Distant-Supervised Phase* durchgeführt, im Folgenden *Distant-Phase* genannt. Wir verwenden einen ähnlichen Ansatz wie in [7]. Dabei wird das CNN mit einer grossen Menge an *weakly-labeled* (dt. schwach annotiert) Texten über eine Epoche hinweg vortrainiert. Weakly labeled bedeutet, dass der Sentiment eines Textes aus der Eigenschaft des Textes abgeleitet

wird und nicht von einem Menschen annotiert wurde. Beispiele für solche Eigenschaften, aus welchen sich ein Sentiment ableiten lässt, sind Emoticons in Tweets oder die Anzahl der vergebenen Sterne bei einem Review. Bei Emoticons lässt sich zum Beispiel aus dem lachenden Emoticon ":-)" ein positiver bzw. aus dem traurigen Emoticon ":-(" ein negativer Sentiment ableiten.

Eine genaue Erläuterung zur Generierung der Trainingsdaten für die Distant-Phase befindet sich im Kapitel 4. Das Training wird mit dem weiter oben beschriebenen Backpropagation Algorithmus durchgeführt.

Supervised Phase Im letzten Schritt wird das CNN mit den von Menschen annotierten Texten trainiert. Dieses Training wird mithilfe von Backpropagation mit AdaDelta durchgeführt. Dabei wird das sogenannte *Early Stopping* verwendet: Das Netzwerk wird solange trainiert, bis sich eine definierte Metrik über eine bestimmte Anzahl Epochen nicht mehr verbessert hat. In unserem Fall ist diese Metrik der F1-Score (siehe 3.5) über die positiven und negativen Datensätze. Auch hier wird der oben beschrieben Backpropagation-Algorithmus durchgeführt.

3.5. Evaluierungsmetrik

Im Folgenden wird die verwendete Evaluierungsmetrik, der *F1-Score*, erläutert.

Präzision & Ausbeute *Präzision* (engl. precision) und *Ausbeute* (engl. recall) sind Metriken, mit welchen die Performanz eines Klassifizierungs-Systems bezüglich einer der zu klassifizierenden Klassen evaluiert werden kann. Die Abkürzungen *tp*, *fp*, *fn* stehen in diesem Zusammenhang für *true positives*, *false positives* und *false negatives*. Dabei ist die Präzision das Verhältnis von richtig klassifizierten (tp) zu allen klassifizierten Datensätze der entsprechenden Klasse ($tp + fp$):

$$precision = \frac{tp}{tp + fp} \tag{3.4}$$

Die Ausbeute ist das Verhältnis von richtig klassifizierten Datensätzen zur Anzahl aller vorhandenen Datensätze der entsprechenden Klasse ($tp + fn$):

$$recall = \frac{tp}{tp + fn} \tag{3.5}$$

Mit diesen beiden Metriken kann die Performanz eines Klassifizierungs-Systems bezüglich einer der vorhandenen Klassen bewertet werden, allerdings haben diese zwei Nachteile: Einerseits sind es zwei Werte anstatt eines einzelnen Wertes. Dies macht die Beurteilung über die Performanz des Systems komplizierter. Ausserdem kann mittels "schummeln" eine sehr hohe Ausbeute erreicht werden, indem immer nur eine bestimmte Klasse zurückgegeben wird. Eine hohe Präzision kann erreicht werden indem das System sehr

"konservativ" arbeitet und nur in Fällen, bei denen die Klassifizierung eindeutig ist, die entsprechende Klasse zurückgibt.

F1-Score Um das oben beschriebene Problem zu lösen, wird der F1-Score verwendet. Dieser ist das harmonische Mittel von Präzision und Ausbeute:

$$\mathrm{F1} = \frac{2 \times \mathit{precision} \times \mathit{recall}}{\mathit{precision} + \mathit{recall}} \tag{3.6}$$

Durch diese Metrik kann die Performanz eines Systems bezüglich einer bestimmten Klasse mittels eines Wertes quantifiziert werden. Ausserdem löst dies das Problem, dass eine hohe Präzision bzw. Ausbeute erzielt werden kann, wenn das System "schummelt".

F1-Score über mehrere Klassen Der F1-Score selbst kann nur jeweils für eine einzelne Klasse bestimmt werden. Um nun aber eine einzige Metrik für die Messung der Performanz des Systems über mehrere Klassen hinweg zu erhalten, werden die F1-Scores der einzelnen Klassen summiert und durch die Anzahl der Klassen dividiert. Durch dieses Vorgehen erhält man folgende Gleichung, wobei k_i für eine einzelnen Klasse und n für die Anzahl der beachteten Klassen:

$$\mathrm{F1}_{k_0,k_1,\ldots,k_n} = \frac{\sum_{i=0}^{n} \mathrm{F1}_{k_i}}{n} \tag{3.7}$$

Diese Art des F1-Score über mehrere Klassen hinweg wird auch *macro-averaged* F1-Score genannt.

3.6. Technischer Aufbau

Im folgenden Abschnitt wird der technische Aufbau erläutert, welcher verwendet wird, um die in Kapitel 6 beschriebenen Experimente durchzuführen. Eine Beschreibung zur Verwendung des Systems befindet sich in Anhang B.

Vorarbeiten Der Grundaufbau der verwendeten Software wurde vom InIT mithilfe von `keras`[4] implementiert und zur Durchführung dieser Arbeit zur Verfügung gestellt. Im Rahmen dieses Grundaufbaus wurden die folgenden Funktionalitäten bereits implementiert:

- Implementation des CNN in `keras` und verwendung von `theano` [28] als Backend für die Graphical Processing Unit (GPU)s.
- Implementation von Evaluations-Metriken.
- Skripte mit den folgenden Funktionalitäten: Trainieren des CNN, Laden von TSV Dateien, Vorverarbeiten von Word-Embeddings.

[4] https://keras.io/

Anforderungen Ein zu implementierendes System, mit welchem die Experimente durchgeführt werden können, soll die folgenden Eigenschaften aufweisen:

- **Parametrisierbarkeit**: Dadurch, dass eine grosse Anzahl kleiner Experimente durchgeführt werden muss, soll das System die Möglichkeit bieten, Experimente parametrisiert durchzuführen.

- **Wiederholbarkeit**: Experimente sollen mit einem minimalen Mehraufwand mehrfach durchgeführt werden können.

- **Übersichtlichkeit**: Resultate der Experimente sollen übersichtlich und einfach zugänglich sein.

- **Auswertbarkeit**: Resultate sollen automatisiert ausgewertet werden können.

Funktionalität Um ein System, welches die oben beschriebenen Anforderungen erfüllt zu erhalten, werden die folgenden Komponenten implementiert:

- **Executor**: Der *Executor* ist zuständig für das Training der CNNs mithilfe von `keras`. Beim Start akzeptiert er die Konfiguration als Parameter. Das Experiment wird mit dem Laden der benötigten Daten und dem anschliessenden Training des CNN gestartet. Am Ende jeder Epoche wird das aktuelle CNN auf den Validierungsdaten getestet und die konfigurierten Metriken ausgewertet. Diese werden am Ende zusammen mit dem trainierten CNN (Gewichte im HDF5-Format[5], das CNN Model als JSON) in einen für das Experiment vorgesehenen Ordner gespeichert. Die Metriken werden ebenfalls in dem dafür vorgesehenen Ordner abgespeichert.

- **Config Management**: Experimente werden über Konfigurationen im JSON-Format[6] parametrisiert. Über diese Konfiguration können viele wichtige Parameter für die Ausführung festgelegt werden, so zum Beispiel: Anzahl Epochen, Trainings- und Validierungsdaten, Parameter für die k-fold Cross-Validation oder auch bereits trainierte Modelle können geladen werden. Detailierte Erläuterungen zu den einzelnen Parametern können im Anhang B gefunden werden.

- **DataLoader**: Mithilfe des *DataLoader* können Trainings- und Validierungsdaten im TSV[7] Dateiformat geladen werden. Die zu ladenden Daten können dabei aus einer oder mehreren TSV-Dateien stammen. Im Falle, dass mehrere TSV Dateien angegeben werden, kann über die Konfiguration das Verhältnis angegeben werden, in welchem die Daten aus den einzelnen Dateien verschmischt werden sollen.

- **Skripte**: Die Auswertung der einzelnen Experimente geschieht über dafür erstellte Skripte.

[5] https://support.hdfgroup.org/HDF5/
[6] http://www.json.org/
[7] https://reference.wolfram.com/language/ref/format/TSV.html

- **Weboberfläche**: Auf die Resultate der Experimente kann über eine eigens dafür entwickelte Weboberfläche zugegriffen werden. Ausserdem besteht die Möglichkeit Plots über die Metriken, welche während des Trainings- und Validierungsprozess gesammelt werden, zu erstellen.

Die oben beschriebenen Komponenten erlauben es, Experimente mittels JSON Konfigurationen zu starten und den gesamten Trainings- und Validierungsprozess mittels Metriken zu überwachen und zu dokumentieren.

Skripte Für die Durchführung der Experimente wurden diverse Skripte erstellt, um die Handhabung zu vereinfachen und Auswertungen zu ermöglichen. Die Liste der implementierten Scripts umfasst unter anderem die folgenden:

- Erstellen von Plots der Lernkurven und Metriken
- Erstellen von Word-Embeddings über einen Text-Corpus
- Erstellen von Statistiken zu Trainings- und Validierungsdaten
- Vorverarbeitung von Trainingsdaten für die Distant-Phase
- Erstellen von Visualisierungen von Word-Embeddings mittels PCA
- Diverse Wartungsskripte zur Generierung und Verwaltung von Experimenten

Weboberfläche Um die dritte Anforderung nach Übersichtlichkeit und Auswertbarkeit zu erfüllen, wird eine Weboberfläche umgesetzt, mit welchem die Parameter und Resultate aller durchgeführten Experimente übersichtlich und an einem Ort zur Verfügung gestellt werden. Für die Implementation wird die `python`[8] Bibliothek `flask`[9] verwendet.

Zur Auswertung der Experimente stehen drei Funktionen zur Verfügung:

- Die Oberfläche gewährt Zugriff auf alle JSON Konfigurationen, welche zu einem Experiment gehören. Dazu zählen die Konfiguration selbst, die gespeicherten Trainings- und Validierungsmetriken und das `keras` Model des CNN.

- Mittels der Plotting Funktion können Plots von Trainings- und Validierungsmetriken erstellt werden.

- Die gespeicherten Validierungs- und Trainingsmetriken können mithilfe von `math.js`[10] direkt im Browser ausgewertet werden.

[8] https://www.python.org/
[9] http://flask.pocoo.org/
[10] http://mathjs.org/

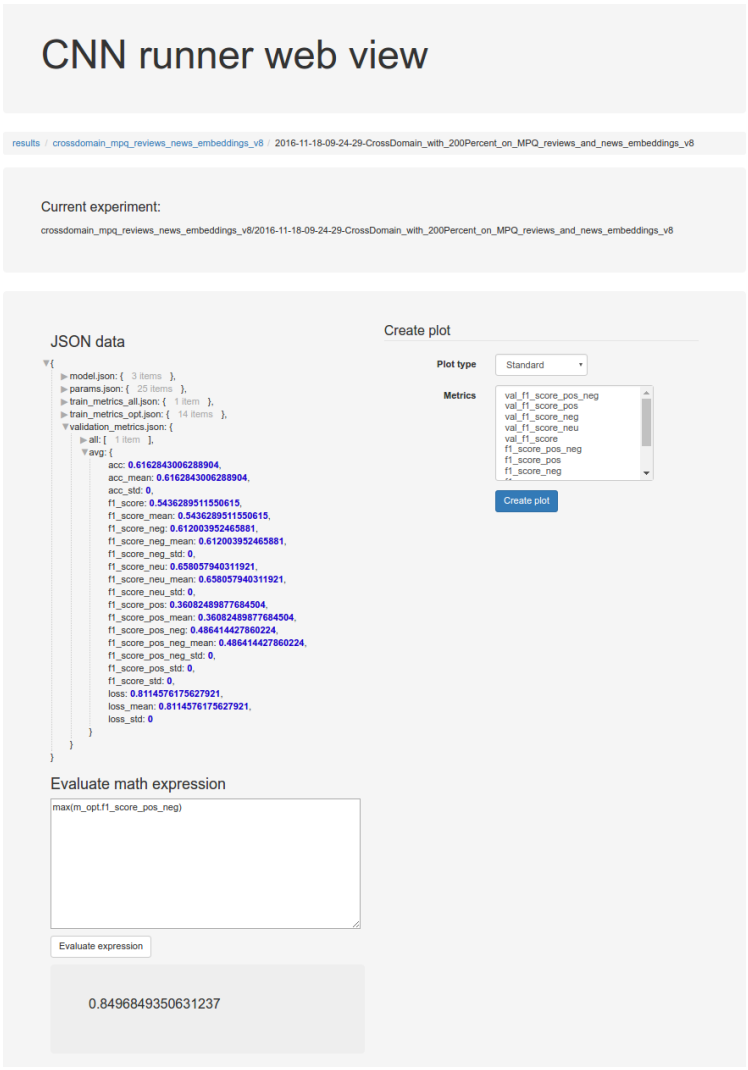


Abbildung 3.8.: Ansicht Experiment über Weboberfläche

Betriebssystem & Softwarepakete Alle Experimente werden mit dem oben beschriebenen Software-System durchgeführt. Auf den beiden verwendeten Computer-Systemen wird als Betriebssystem Ubuntu 16.04 installiert. Dazu werden `python` in der Version 3.5.2, Nvidia GPU Treiber und `cuda`[11] in der Version 8.0 als Abhängigkeiten von `theano` und `keras` installiert.

Hardware Zur Durchführung der Experimente werden zwei unterschiedliche Computer verwendet. Im ersten System (S1) ist eine Nvidia GTX970 GPU, einen Intel i7 4950K CPU und 16GB Arbeitsspeicher installiert. Das zweite System besitzt eine Nvidia GTX1070 GPU, einen Intel i7 6700K CPU und ebenfalls 16GB Arbeitsspeicher. Die Unterschiede in der Hardware haben keinen Einfluss auf die Resultate der Experimente, da auf beiden Systemen dasselbe Betriebssystem mit den gleichen Softwarepaketen verwendet wird.

[11] https://developer.nvidia.com/cuda-toolkit

4. Daten

Im folgenden Kapitel werden die im Laufe dieser Arbeit verwendeten Datensätze und Text-Corpora genauer beschrieben. Es wird erläutert woher die Daten stammen, welchen Zweck sie erfüllen und wie die einzelnen Datensätze aufgebaut sind. Des Weiteren wird erläutert, wie die Word-Embeddings und Datensätze für die Distant-Phase generiert werden.

Die Datensätze, welche in dieser Arbeit verwendet werden, können in zwei Klassen unterteilt werden:

- *Supervised*: Datensätze, welche Texte und Sentiment-Annotationen mitliefern, gehören zur Klasse der *supervised* Datensätze. Dazu zählen alle Datensätze, welche in den folgenden Experimenten zum Training und zur Evaluierung der CNNs verwendet werden.

- *Unsupervised*: Datensätze, welche Texte aber keine zugehörigen Sentiments mitliefern gehören zur Klasse der *unsupervised* Datensätze. Dazu zählen einerseits die Daten, mit welchen die Distant-Phase durchgeführt wird. Andererseits gehören hierzu auch die Text-Corpora mit welchen die Word-Embeddings generiert werden.

4.1. Supervised

Alle Datensätze, welche während des Trainings und der Evaluierung eines CNN verwendet werden, liefern Texte und die entsprechenden Sentiment-Annotationen mit. Die Datensätze wurden im Verhältnis 4:1 in Trainings- und Test-Datensätze aufgesplittet.

Trainingsdaten Die in der folgenden Tabelle aufgelisteten Datensätze werden für das Training der Convolutional Neural Network (CNN)s verwendet.

					Verteilung Sentiments			
Name	Textsorte	Anzahl Texte	Durchschnittliche Anzahl Zeichen	Durchschnittliche Anzahl Wörter	positiv	neutral	negativ	Quelle
DAI_tweets	Tweets	3274	63.4	16.1	19.4%	66.9%	13.6%	[17]
DIL_reviews	Reviews	3420	74.0	19.0	31.1%	50.8%	17.9%	[6]
HUL_reviews	Reviews	3156	70.7	18.6	28.4%	57.7%	13.9%	[9]
JCR_quotations	Quotations	1032	148.4	33.6	15.0%	71.3%	13.7%	[1]
MPQ_news	News	8888	123.3	26.9	14.8%	55.5%	29.7%	[30]
SEM_headlines	Headlines	1000	34.3	7.1	14.4%	61.0%	24.6%	[24]
SemEval_tweets	Tweets	8226	89.1	22.4	37.2%	48.1%	14.7%	[16]
TAC_reviews	Reviews	2152	88.6	22.4	36.3%	17.7%	46.0%	[26]

Tabelle 4.1.: Statistiken zu Texten und Sentiments der Trainingsdaten.

Testdaten Die in der folgenden Tabelle aufgelisteten Datensätze werden verwendet, um die Performanz eines trainierten CNN zu evaluieren.

					Verteilung Sentiments			
Name	Textsorte	Anzahl Texte	Durchschnittliche Anzahl Zeichen	Durchschnittliche Anzahl Wörter	positiv	neutral	negativ	Quelle
DAI_tweets	Tweets	819	66.5	16.8	19.8%	67.9%	12.3%	[17]
DIL_reviews	Reviews	855	74.9	19.2	31.6%	51.6%	16.8%	[6]
HUL_reviews	Reviews	789	63.3	17.1	21.7%	53.3%	25.0%	[9]
JCR_quotations	Quotations	258	143.6	32.6	14.7%	49.2%	36.1%	[1]
MPQ_news	News	2223	121.6	26.5	13.1%	55.1%	31.8%	[30]
SEM_headlines	Headlines	255	33.4	7.1	12.0%	61.6%	26.4%	[24]
SemEval_tweets	Tweets	3813	89.6	21.8	41.2%	43.0%	15.8%	[16]
TAC_reviews	Reviews	537	110.4	26.7	26.6%	12.1%	61.2%	[26]

Tabelle 4.2.: Statistiken zu Texten und Sentiments der Testdaten.

4.2. Unsupervised

Im Folgenden werden die unsupervised Datensätze erläutert. Einerseits beinhalten diese die Text-Corpora, mit welchen die Word-Embeddings generiert werden, andererseits gehören hierzu auch die Datensätze, welche für das Training der CNNs während der Distant-Phasen eingesetzt werden.

Text-Corpora Im Rahmen dieser Arbeit werden vier verschiedene Arten von Word-Embeddings verwendet. Diese werden mit verschiedenen Text-Corpora generiert, welche in der folgenden Tabelle aufgelistet sind:

Name	Beschreibung	Anzahl Texte	Quelle
News	News-Texte von diversen Webseiten	90 Mio.	STATMT Webseite[1]
Tweets	Sammlung von öffentlich verfügbaren Tweets	590 Mio.	Twitter-API[2]
Wiki	Texte aller Artikel auf Wikipedia mit mehr als 50 Wörtern	4.5 Mio.	Wikimedia[3]

Tabelle 4.3.: Statistiken zu Text-Corpora, welche für die Generierung der Word-Embeddings verwendet werden.

Zusätzlich werden während der Experimente zufällig initialisierte Word-Embeddings verwendet; diese werden im Folgenden mit *Random* gekennzeichnet. Die Generierung der Word-Embeddings wurde mittels *word2vec* [15] durchgeführt. Dabei wurden die folgenden Hyperparameter verwendet:

Name	Wert
algorithm	skip-gram
dimensions	52
window size	5
minimum count	15
sample	10^{-5}
hierarchical softmax[4]	0

Tabelle 4.4.: Hyperparameter, welche für die Generierung der Word-Embeddings mit word2vec verwendet werden.

[1] http://www.statmt.org/wmt14/training-monolingual-news-crawl/
[2] https://dev.twitter.com/rest/public
[3] https://dumps.wikimedia.org/enwiki/latest/
[4] 0 bedeutet das *negative sampling* verwendet wird, 1 entsprechend *hierarchical soft-max*

Datensätze für Distant-Phase Im Rahmen dieser Arbeit werden zwei verschiedene Datensätze für die Distant-Phase verwendet: Reviews von Amazon[5] und Tweets von Twitter[6]. Dabei werden die Sentiments bei den Reviews anhand der Anzahl vergebener Sterne (1-5) und bei den Tweets anhand der im Text vorhandenen Emoticons abgeleitet. Bei den Reviews werden alle Texte, welche mit einer Wertung von $x >= 4$ als positiv und $x < 3$ als negativ klassifiziert; die Restlichen werden als neutral angenommen. Bei den Tweets wird ein Lexikon von positiven (z.B. ":-)" oder ":-P") und negativen (z.B. ":-(" "-.-") Emoticons verwendet. Mithilfe dieses Lexikons und den im Tweet vorhandenen Emoticons wird dann der Sentiment des Tweets abgeleitet. Die Emoticons werden nach der Klassifizierung aus den Tweets entfernt.

Dieses Schema führt zu folgenden Datensätzen, welche während der Distant-Phase verwendet werden:

					Verteilung Sentiments			
Name	Textsorte	Anzahl Texte	Durchschnittliche Anzahl Zeichen	Durchschnittliche Anzahl Wörter	positiv	neutral	negativ	Quelle
Amazon Reviews	Reviews	82.4 Mio.	382.2	96.2	78.2%	8.5%	13.2%	[33]
Tweets	Tweets	100.0 Mio.	46.9	12.4	79.1%	0.0%	20.9%	Twitter-API[7]

Tabelle 4.5.: Statistiken zu Texten und Sentiments der weakly-supervised Datensätze.

Des Weiteren werden auch Experimente ohne Distant-Phase durchgeführt; diese werden mit *None* gekennzeichnet.

[5]https://www.amazon.com/
[6]https://twitter.com/
[7]https://dev.twitter.com/rest/public

5. Methodik

In diesem Kapitel wird erläutert, wie die im Folgenden beschriebenen Experimete durchgeführt werden. Dabei wird auf die verwendete Architektur des CNN, die Evaluierung sowie auf die Durchführung der Distant-Phasen eingegangen.

5.1. Architektur des verwendeten Convolutional Neural Network

Als Basis unseres CNN haben wir die Architektur von Deriu et. al. [5] verwendet. Dieses basiert wiederum auf der Architektur von Severyn et. al. [21].

Das verwendete CNN setzt sich aus zwei Convolutional+Max-Pooling Schichten (vgl. Kapitel 3) zusammen. Anschliessend wird eine fully-connected Hidden-Schicht hinzugefügt, welcher gemeinsam mit der Softmax-Regressions Schicht am Ende dafür zuständig ist, die Texte anhand des Sentiment zu klassifizieren.

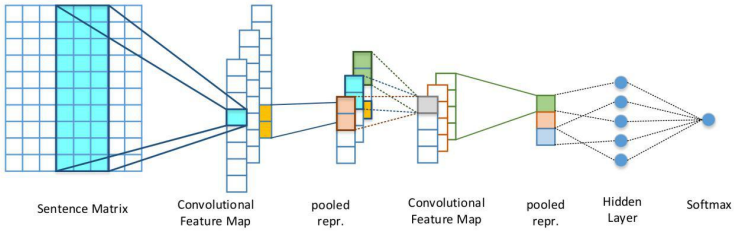


Abbildung 5.1.: Schematische Darstellung der Architektur des in dieser Arbeit verwendeten CNN [5].

Dabei werden die Hyperparameter gemäss Tabelle 5.1 für die Convolutional Schichten verwendet:

Name	Wert
Anzahl Convolutional Filter	200
Grösse der Filter-Fenster	6
Stride der Filter-Fenster	1
Grösse des Fensters der ersten Max-Pooling Schicht	4
Stride der ersten Max-Pooling Schicht	2
Aktivierungsfunktion	*relu*

Tabelle 5.1.: Hyperparameter, welche für die Convolutional und Max-Pooling Schichten des CNN verwendet werden.

Für die Hidden-Schicht am Ende des CNN wird eine Dimensionalität von 200 gewählt. Um Overfitting zu vermindern, wird nach der Hidden-Schicht eine Dropout-Schicht [23] zur Regularisierung hinzugefügt, p ist dabei auf 20% gesetzt. Das Training wird mittels Backpropagation und AdaDelta durchgeführt. Dabei wurden die Hyperparameter gemäss Tabelle 5.2 für AdaDelta verwendet:

Name	Wert
ϵ	10^{-6}
ρ	0.95

Tabelle 5.2.: Hyperparameter, welche für das Training des CNN mit AdaDelta verwendet werden.

Die oben erwähnten Hyperparameter für AdaDelta werden als Empfehlung aus [32] übernommen.

5.2. Durchführung der Distant-Phase

Bei den Experimenten in Kapitel 6 werden die CNNs jeweils pro Distant-Phase vortrainiert. Dabei kommen die in Kapitel 4 beschriebenen Datensätze zum Einsatz. Das CNN wird bei jedem der Experimente mit dem entsprechenden Datensatz über eine Epoche hinweg vortrainiert. Das daraus resultierende CNN wird dann als Basis für die weiteren Experimente verwendet. Bei den Experimenten, bei denen die Distant-Phase mit *None* gekennzeichnet wird, werden die CNNs nicht vortrainiert, sondern die Gewichte werden zufällig initialisiert.

Da die Datensätze für die Distant-Phase unterschiedlich gross sind, wurden jeweils nur 80 Mio. Datensätze verwendet, um gleiche Bedingungen für die beiden Datensätze zu schaffen.

5.3. Klassengewichte

Wie im Kapitel 4 ersichtlich ist, sind die Klassen in den einzelnen Datensätzen teilweise unterschiedlich stark repräsentiert. Das kann dazu führen, dass sich ein CNN vor allem an den stark vorhandenen Klassen orientiert und dadurch auf den Datensätzen der unterrepräsentierten Klassen schlechte Ergebnisse liefert. Um dieses Problem auszugleichen, werden sogenannte *Klassengewichte* verwendet. Diese werden als Faktoren verwendet, um die Fehlerfunktion entsprechend zu skalieren und so die Unterschiede in den Häufigkeiten der Klassen auszugleichen.

Wenn n die Anzahl aller Datensätze und m_x die Anzahl der Datensätze der entsprechenden Klassen $x \in S$ mit $S = \{\text{pos}, \text{neu}, \text{neg}\}$ darstellt, kann das Klassengewicht für den Sentiment x mit folgender Gleichung bestimmt werden:

$$c_x = \frac{n}{|S| \cdot m_x} \tag{5.1}$$

Die dadurch berechneten Klassengewichte entsprechen $c_x = 1.0$ im Falle einer perfekten Gleichverteilung der Klassen. Überrepräsentierte Klassen erhalten dadurch ein Klassengewicht $c_x < 1.0$ und Unterrepräsentierte entsprechend ein Klassengewicht $c_x > 1.0$.

Wenn die Gleichung 5.1 zur Illustration auf den Trainings-Datensatz *MPQ_news* angewendet wird, resultieren daraus die folgenden Klassengewichte:

$$c_{pos} = \frac{8888}{3 \cdot 1317} = 2.2495$$

$$c_{neg} = \frac{8888}{3 \cdot 2637} = 1.1234$$

$$c_{neu} = \frac{8888}{3 \cdot 4934} = 0.6004$$

5.4. Evaluierung der Resultate

Bei der Evaluierung aller nachfolgenden Experimente kommt ausschliesslich der $\text{F1}_{pos,neg}$ zum Einsatz, wie er während des SemEval-Wettbewerbs [16] verwendet wurde.

6. Experimente und Resultate

In diesem Kapitel werden der Aufbau der Experimente und die zugehörigen Resultate erläutert. Wir werden alle Experimente auch mit der Domäne *Union* durchführen, welche die Vereinigung der Daten aller Domänen darstellt. Bei der Erläuterung der Resultate ist mit alle Domänen jeweils exklusiv *Union* gemeint. Wir behandeln *Union* als einen Spezialfall. Wenn wir von einem CNN sprechen, welches nur mit Trainingsdaten von einer Domäne trainiert wurde, so bezeichnen wir dieses mit "*Domänenname*-CNN" (z.B. *DAI_tweets*-CNN). Weiterhin verwenden wir die Abkürzung TD für Target Domain (Zieldomäne) und FD für Foreign Domain (Fremddomäne).

6.1. Evaluierung beste Distant-Phase & Word-Embeddings pro Domäne

Ziel dieser Experimente ist es, herauszufinden welche Kombination von Word-Embeddings und Distant-Phase (siehe Kapitel 4.2) die besten Ergebnisse für die jeweilige Domäne ergibt. Aufgrund dieser Ergebnisse werden alle nachfolgenden Experimente mit der bestmöglichen Kombination durchgeführt.

6.1.1. Aufbau

Pro Domäne werden 12 Experimente durchgeführt, dies resultiert aus der Kombination mit 4 Word-Embeddings und den 3 Distant-Phasen. Bei jedem Experiment wird die Gesamtheit aller Trainingsdaten der TD verwendet, um das CNN zu trainieren. Während dem Training wird mit den Testdaten der TD validiert und nach dem Training wird das CNN mit den Testdaten der TD getestet. So erhalten wir einen Sentiment-Klassifizierer, welcher für eine TD spezialisiert ist.

6.1.2. Resultate

In der Tabelle 6.1 befinden sich die erzielten $F1_{pos,neg}$ nach dem testen mit der TD. Die besten Werte pro Domäne sind jeweils fett markiert.

	Dist. Phase	*DAI (T/ 3.2k)*	*DIL (R/ 3.4k)*	*HUL (R/ 3.1k)*	*JCR (Q/ 1k)*	*MPQ (N/ 8.8k)*	*Seval (T/ 8.2k)*	*SEM (H/ 3.1k)*	*TAC (R/ 2.1k)*	*Union*	Avg.
Random	None	0.599	0.509	0.513	0.263	0.469	0.598	0.436	0.577	0.550	0.502
	Reviews	0.698	0.595	0.659	0.401	0.539	0.659	0.477	**0.714**	0.603	0.594
	Twitter	0.684	0.523	0.517	0.399	0.490	0.659	0.468	0.612	0.595	0.550
News	None	0.631	0.485	0.480	0.405	**0.581**	0.673	0.527	0.644	0.615	0.560
	Reviews	0.692	0.590	0.649	0.433	0.563	0.685	0.519	0.682	**0.624**	0.604
	Twitter	0.678	0.546	0.554	0.412	0.567	**0.691**	0.539	0.676	0.444	0.568
Twitter	None	0.629	0.507	0.511	0.360	0.504	0.629	0.541	0.585	0.584	0.539
	Reviews	0.701	**0.603**	0.657	0.383	0.540	0.683	0.472	0.694	0.611	0.594
	Twitter	**0.734**	0.543	0.533	0.412	0.518	0.685	**0.554**	0.652	0.610	0.585
Wikipedia	None	0.553	0.455	0.451	0.375	0.529	0.613	0.506	0.570	0.567	0.513
	Reviews	0.661	0.569	**0.666**	**0.457**	0.542	0.622	0.500	0.684	0.577	0.586
	Twitter	0.663	0.520	0.531	0.411	0.544	0.642	0.505	0.619	0.580	0.557
Full Average	-	0.660	0.537	0.562	0.393	0.532	0.653	0.504	0.643	0.580	-
Rand. Avg.	-	0.660	0.542	0.563	0.354	0.499	0.639	0.460	0.635	0.583	0.548
News Avg.	-	0.667	0.540	0.561	**0.417**	**0.570**	**0.683**	**0.528**	**0.668**	0.561	0.577
Twit. Avg.	-	**0.688**	**0.551**	**0.573**	0.385	0.521	0.666	0.522	0.643	**0.602**	0.572
Wiki. Avg	-	0.626	0.515	0.549	0.414	0.538	0.626	0.504	0.625	0.575	0.552
-	None Avg.	0.603	0.489	0.489	0.351	0.520	0.628	0.502	0.594	0.579	0.528
-	Rev. Avg.	0.688	**0.589**	**0.658**	**0.418**	**0.546**	0.662	0.492	**0.694**	**0.604**	0.595
-	Twi. Avg.	**0.690**	0.533	0.539	0.409	0.530	**0.669**	**0.517**	0.640	0.558	0.565

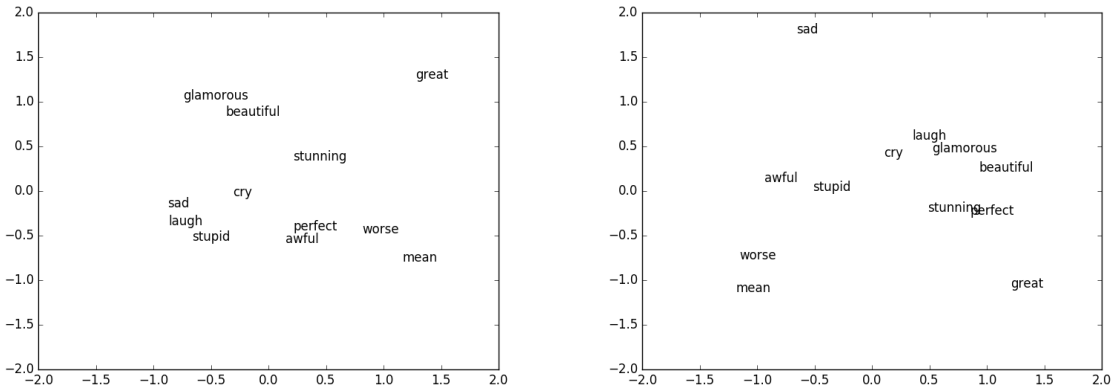
Tabelle 6.1.: Resultate aller Kombinationen aus Word-Embeddings und Distant-Phase pro Domäne. Die letzte Spalte ist der Durchschnittswert der jeweiligen Zeile und somit Domänen übergreifend Die unteren 8 Zeilen sind jeweils die Durchschnittswerte, entsprechend der Beschriftung, pro Domäne. Die Textsorten der Domäne sind folgendermassen codiert: T: Tweets, N: News, R: Reviews, H: Headlines and Q: Quotations.

$F1_{pos,neg}$ als Validierungsmetrik ist sinnvoll. Zur Validierung der Resultate verwenden wir ausschliesslich den $F1_{pos,neg}$ (siehe Abschnitt 3.6). Die ersten Ergebnisse bestätigen diese Wahl der Metrik. Der $F1_{neu}$ ist tendenziell höher als der $F1_{pos}$, oder $F1_{neg}$. Zusätzlich sind in den Trainings- und Testdaten die positiven und negativen Beispiele meist seltener vertreten (vgl. Tabelle 4.1). Anhand konkreter Zahlen können wir diese Vermutung des stets besseren $F1_{neu}$ bestätigen. Die Mittelwerte bei dem Experiment Crossdomain sind wie folgt: $F1_{pos}$ 0.479, $F1_{neg}$ 0.455 und $F1_{neu}$ 0.581. Dies, obwohl die Validierung bei diesen Experimenten nur auf dem $F1_{pos,neg}$ stattfindet und wir die Fehlerfunktion bei positiven und negativen Beispielen mit Klassengewichten (siehe Abschnitt 5.1) bereits verstärken.

Die Wahl der Distant-Phase und Word-Embeddings tragen massgeblich zum Resultat bei. Wie wir in der Tabelle 6.1 erkennen können, variieren die Resultate innerhalb einer Domäne stark. Der Unterschied vom besten $F1_{pos,neg}$ zum Schlechtesten beträgt in fast allen Domänen ungefähr 15 Punkte. Auch wenn wir die Experimente mit gleichen Word-Embeddings, oder diejenigen mit gleicher Distant-Phasen untereinander vergleichen, sind die Unterschiede teilweise sehr gross. Somit ist für uns klar ersichtlich, dass die gewählte Kombination aus Word-Embeddings und Distant-Phase einen erheblichen Einfluss auf das Resultat hat.

Random Word-Embeddings oder None Distant-Phase liefern nicht zwingendermassen schlechte Ergebnisse. Wir sehen in der Tabelle 6.1, dass die zufällige Initialisierung

der Word-Embeddings oder das Auslassen einer Distant-Phase nicht in allen Fällen einen negativen Effekt auf den resultierenden Sentiment-Klassifizierer hat. Wenn man jedoch die letzte Spalte Avg. betrachtet sehen wir, dass die Ergebnisse mit None Distant-Phase pro Word-Embeddings im Schnitt immer schlechter sind. Wichtig hierbei ist auch, dass die Word-Embeddings und die Distant-Phase sich gegenseitig beeinflussen, da die Word-Embeddings zu den Parametern des CNN gehören und somit auch durch Backpropagation angepasst werden. Aus der Tabelle entnehmen wir, dass bei jeder Domäne die Kombination aus Random Word-Emebddings und None Distant-Phase immer schlechte Ergebnisse liefert. In der Abbildung 6.1 sehen wir den Einfluss der Distant-Phase anhand einiger ausgesuchter Wörter. Die Word-Embeddings stammen aus dem Wikipedia Datensatz und die Distant-Phase wurde mit Reviews durchgeführt. Wir sehen, dass die Distanzen zwischen gegensätzlichen Wörtern nach der Distant-Phase jeweils ähnlich gross ist. Vor der Distant-Phase waren Wörter, die in einem ähnlich Kontext vorkommen können, nahe beieinander. Danach sind die Wörter stärker nach ihrem Sentiment gruppiert wie zuvor.



(a) Ausgesuchte Beispiele vor der Distant-Phase (b) Ausgesuchte Beispiele nach der Distant-Phase

Abbildung 6.1.: Veränderung Wikipedia Word-Embeddings durch die Review Distant-Phase. Die projezierte Darstellung wurde mittels PCA erstellt.

Union scheint eine Option zu sein. Wenn wir in der Tabelle 6.1 die Zeilen betrachten sehen wir, dass die einzelnen Werte bis zu 20 Punkte variieren. Vergleichen wir den Durchschnitt einer Zeile mit dem Ergebnis des *Union*-CNN, liegen die Werte immer sehr nahe beieinander und sind meistens sogar besser. Sofern nun ein Sentiment-Klassifizierer für eine noch unbekannte Domäne benötigt wird, für welche wir die beste Kombination aus Word-Embeddings und Distant-Phase noch nicht kennen, scheint es eine Option zu sein, mit einem *Union*-CNN zu starten.
Wir werden später diese Vermutung noch genauer prüfen, indem wir mit einer unbekannten Domäne ein *Union*-CNN testen. Zusätzlich möchten wir eruieren, ob sich gewisse Domänen besser eignen als Andere, um sie zu kombinieren. Zum jetzigen Zeitpunkt können wir aber die folgende Aussage treffen: Anstelle ein CNN einer zufälligen Domäne zu verwenden, um eine neue, unbekannte Domäne zu testen, scheint es ratsamer zu sein, ein *Uni-*

on-CNN zu verwenden. Wir verlieren dadurch vielleicht ein paar Punkte beim $F1_{pos,neg}$, entfernen jedoch die zum Teil recht grosse Varianz des $F1_{pos,neg}$.

6.2. Crossdomain

Mit dieser Experimentreihe möchten wir herausfinden, wie gut die besten CNN im Kapitel 6.1 auf einer fremden Domäne performen. Wir möchten dabei auch untersuchen, ob es Domänen gibt welche sich besser zur Kombination eignen als Andere.

6.2.1. Aufbau

Wir trainieren ein CNN mit den Trainingsdaten einer FD und validieren während dem Training auch auf derselbigen. Anschliessend testen wir einzeln den Sentiment-Klassifizierer auf allen anderen Domänen, welche in diesem Experiment die TDs sind. Wir verwenden in diesem Experiment jeweils die kompletten Datensätze an Trainings- und Testdaten. Man könnte sich auch überlegen, anstelle der besten Kombination aus Word-Embeddings und Distant-Phase für die FD, die beste Kombination für die TD zu wählen. Wir entschieden uns dafür, weil während dem Training auch die Embeddings angepasst werden und in einem Anwendungsfall die beste Kombination für die TD noch unbekannt ist.

6.2.2. Resultate

In der Tabelle 6.9 befinden sich die erzielten $F1_{pos,neg}$ nach dem Testen mit der TD. Die besten Werte pro Domäne sind jeweils fett markiert.

Die besten Resultate erhalten wir auf der eigenen Domäne. Wie erwartet erhalten wir das beste Ergebnis wenn wir auf der gleichen Domäne validieren und testen. Die Varianz in den Ergebnissen ist auffallend hoch. Es scheint, dass es strukturell grosse Unterschiede zwischen den verschiedenen Domänen gibt. Wir möchten herausfinden, ob und welche Strukturen relevant sind, um Domänen zusammenfassen zu können.

Training \ Test	*DAI T*	*DIL R*	*HUL R*	*JCR Q*	*MPQ N*	*Seval T*	*SEM H*	*TAC R*	*Union*
DAI T	**0.734**	0.401	0.397	0.269	0.161	0.554	0.283	0.369	0.447
DIL R	0.381	**0.603**	0.602	0.227	0.210	0.365	0.138	0.478	0.350
HUL R	0.404	0.567	**0.666**	0.312	0.252	0.392	0.176	0.535	0.373
JCR Q	0.450	0.402	0.452	**0.457**	0.319	0.402	0.254	0.461	0.384
MPQ N	0.495	0.307	0.318	0.411	**0.581**	0.471	0.313	0.402	0.489
Seval T	0.525	0.489	0.479	0.421	0.441	**0.691**	0.445	0.577	0.578
SEM H	0.360	0.188	0.181	0.054	0.148	0.250	**0.554**	0.247	0.227
TAC R	0.395	0.501	0.517	0.409	0.376	0.480	0.360	**0.714**	0.442
Union	**0.680**	0.529	0.566	0.418	0.559	0.671	0.357	0.576	0.624
FD Avg.	0.468	0.432	0.451	0.320	0.311	0.451	0.315	0.473	0.411
Diff.	0.212	0.096	0.114	0.098	0.248	0.221	0.042	0.103	0.213

Tabelle 6.2.: Resultate wenn auf einer Domäne trainiert und auf allen anderen Domänen getestet wird. Die Zeile *FD Avg.* beinhaltet den durchschnittlich erzielten Wert pro getestete Domäne, exklusiv die Domäne *Union*. In der Zeile *Diff.* steht die Differenz zwischen den Ergebnissen mit *Union* und dem Durchschnitt.

Ein Union-CNN ist nie besser als für die TD trainiertes CNN. Die Idee, dass mit einem *Union*-CNN möglicherweise bessere Ergebnisse erzielt werden können, als mit einem für die TD spezialisierten CNN, ist anhand unserer Ergebnisse widerlegt. Durch das Mischen verschiedener Domänen, respektive Textsorten, gewinnt das CNN offensichtlich keine tieferen strukturellen Informationen, welche zu einem besseren Verständnis einzelner Domänen führen würde. Wir halten jedoch fest, dass das einzelne Testen mit den TD Domänen *DAI_tweets*, *MPQ_news* und *SemEval_tweets* auf einem *Union*-CNN fast gleich gute Ergebnisse liefert, wie wenn wir mit den für die TD trainierten CNN testen. Beim Testen mit der Domäne *MPQ_news* erhalten wir sogar ein leicht besseres Ergebnis mit dem *Union*-CNN. Wenn wir die Ergebnisse des *Union*-CNN mit den durchschnittlichen Werten pro TD (Spalte) vergleichen, sind die *Union*-CNN Werte stets besser. Dies scheint unsere Vermutung in 6.2.2 zu bestätigen, dass wir für eine neue, unbekannte TD die zuverlässigeren Ergebnisse erhalten, wenn wir ein *Union*-CNN benutzen. Die Verwendung eines CNN, welches für eine einzelne FD trainiert wurde, ist nicht ratsam, da die Varianz der Werte sehr hoch ist.

Gewisse Domänen erzielen in Kombination durchwegs bessere Ergebnisse. Es gibt zwei Aspekte die wir uns in diesem Zusammenhang überlegen. Der erste Aspekt ist die Ausprägung des Sentiment (siehe Tabelle 6.3), also wie eindeutig positiv oder negativ ein Text verfasst wird. Der zweite Aspekt, wie konzentriert der Sentiment vorliegt, also wie lange die Texte der Domäne im Durchschnitt sind. Bei Domänen mit der Textsorte Tweets wie auch bei den Review Domänen, scheint die Ausprägung des Sentiment stark zu sein. Eine Privatperson schreibt sehr deutlich, wenn ihr etwas gefällt oder eben nicht. Dies im Gegensatz zu News Meldungen, Headlines und Quotations, bei welchen eine gewisse Professionalität verlangt wird und somit die Ausprägung des Sentiments weniger stark ist.

Die Konzentration des Sentiments unterscheidet sich ebenfalls (siehe Tabelle 6.3). Damit meinen wir, aus wie vielen Wörtern der Sentiment abgeleitet werden muss. Tweets haben den Vorteil, dass die Konzentration des Sentiments durch die Zeichenbeschränkung sehr hoch ist. Es wird oft mit wenigen Wörtern ein klar positiver oder negativer Sentiment beschrieben. Wie in der Tabelle 4.1 ersichtlich, haben die Domänen mit den Textsorten Reviews und Tweets durchschnittlich weniger Zeichen als die anderen Domänen. Einzige Ausnahme bildet die Domäne *SEM_headlines*, welche durchschnittlich aus noch weniger Zeichen besteht, jedoch fehlt bei dieser Domäne die starke Ausprägung des Sentiments. Die Performanz dieser Domäne ist wie in der Tabelle 6.2 ersichtlich eher schlecht. Vermutlich verträgt sich die Kombination aus schwacher Ausprägung des Sentiments und hoher Konzentration nicht gut.

Sentiment	Domäne	Beispielsatz	Auspräung Sentiment	Konzentration Sentiment
positiv	*DAI_tweets*	Rihana look hot!!	++	++
positiv	*MPQ_news*	The good news is Secretary of Defense Donald Rumsfeld will be visiting the detainees' Camp X-Ray the next morning.	−−	−
positiv	*SEM_headlines*	New gadgets galore at tech show.	−	++
negativ	*DAI_tweets*	Migraines suck. #migrainessuck	++	++
negativ	*MPQ_news*	Advocate Faris Abu-Hasan pointed to the violation of international laws represented in settlement activity and settlement building, and to Israel's continuing refusal to respond to international appeals and violation of relevant international laws.	−−	−−
negativ	*SEM_headlines*	Serbie rejects United Nation's Kosovo plan	−	+

Tabelle 6.3.: Beispieltexte für die unterschiedliche Ausprägung und Konzentration des Sentiments, abhängig der Domäne. Zur Verdeutlichung verteilen wir für die Eigenschaften Ausprägung und Konzentration subjektive Label von −− (schwache Ausprägung) bis ++ (starke Ausprägung).

Die Wahl der Domänen ist statistisch signifikant bezüglich dem $F1_{pos,neg}$**.** Wir sehen in der Tabelle 6.2, dass die Ergebnisse doch sehr stark variieren und möchten mit einer zweifaktoriellen Varianzanalyse untersuchen, ob die abhängige Variable $F1_{pos,neg}$ von den unabhängigen Variablen TD und FD abhängig ist. Als Signifikanzniveau wählen wir den Standartwert 0.05. Das Ergebnis dieser Analyse sehen wir in der Tabelle 6.4. Der P-Wert bei beiden unabhängigen Variablen liegt unter dem gewählten Signifikanzniveau von 0.05. Das bedeutet, beide unabhängigen Variablen sind statistisch signifikant und somit ist es relevant, mit welcher Domäne wir ein CNN trainieren oder testen. Der P-Wert, die Prüfgrösse F und der Kritische F-Wert zeigen uns, dass das Ergebnis stärker durch die FD beeinflusst wird als durch die TD. Dass der Resultierende $F1_{pos,neg}$ stärker von den Trainingsdaten abhängt als von der TD mit der wir den Sentiment-Klassifizierer testen, erstaunt uns nicht. Trotzdem wollten wir diesen Sachverhalt statistisch untersuchen, um herauszufinden, ob es Unterschiede zwischen den einzelnen Domänen gibt.

Streuungsursache	Quadratsummen	Freiheitsgrade	Mittlere Quadratsumme	Prüfgrösse (F)	P-Wert	Kritischer F-Wert
TD	0.357	8	0.045	3.516	0.00199	2.087
FD	0.566	8	0.071	5.577	0.00002	2.087
Zufallsfehler	0.811	64	-	-	-	-
Summe	0.012	80	-	-	-	-

Tabelle 6.4.: Ergebnis der zweifaktoriellen Varianzanalyse mit Signifikanzniveau 0.05. In der Spalte TD wird der Einfluss der Zieldomäne auf die Varianz angegeben und in der Spalte FD der Einfluss der Fremddomäne.

Die Datenmenge korreliert nur bedingt mit der Varianz. Wir wissen nun, dass die Wahl der TD und FD einen signifikanten Einfluss auf den resultierenden $F1_{pos,neg}$ hat. In der Tabelle 6.5 sehen wir die Varianz pro FD, die Anzahl Trainingsdaten und den durchschnittlichen $F1_{pos,neg}$. In der Tabelle 6.6 wird die Varianz, Anzahl der Testdaten und der durchschnittliche $F1_{pos,neg}$ pro TD aufgelistet. Als erstes möchten wir prüfen, ob die hohe Varianz mit der Menge an Trainingsdaten korreliert. Die beiden Streudiagramme 6.2 und 6.3 zeigen die zuvor berechneten Varianzen in Relation zu den Datenmengen. Leider haben wir diesbezüglich wenig Messergebnisse, trotzdem versuchen wir einen Trend zu erkennen. Deshalb haben wir eine lineare Trendlinie eingefügt. Bei 6.2 sehen wir, dass die Varianz mit steigender Anzahl Trainingsdaten nicht kleiner wird, es scheint sogar, dass die Varianz grösser wird. Wir vermuten jedoch, dass es andere Gründe geben muss, weshalb die Varianz steigt. Beim Streudiagramm 6.3 hingegen sehen wir, dass die Varianz mit steigender Anzahl an Datensätzen der Fremddomäne sinkt. Eine Begründung könnte sein, dass die Testsets teilweise sehr klein (z.B. 258, 255, 537) sind. Da wir jedoch die Menge an Testdaten nicht steuern können, ist diese Erkenntnis lediglich informativ. Wir halten folgende Erkenntnis fest: Die Ursache der Varianz in den Testdomänen hängt vermutlich teilweise mit kleinen Datensätzen zusammen. Die Ursache für die Varianz pro Trainingsdomäne scheint nicht mit der Menge an Daten zusammen zu hängen. Aufgrund dieser Erkenntnis muss es eine andere Ursache geben. Wir vermuten, dass es Domänen gibt, welche sich besser eignen zu kombinieren als andere. Diese These werden wir mit den Experimenten im Kapitel 6.3 untersuchen.

	Varianz	Anzahl Trainingsdaten	Avg $F1_{pos,neg}$
DAI T	0.018	3274	0.402
DIL R	0.018	3420	0.372
HUL R	0.022	3156	0.408
JCR Q	0.017	1032	0.398
MPQ N	0.026	8888	0.421
Seval T	0.021	8226	0.516
SEM H	0.017	1000	0.246
TAC R	0.019	2152	0.466

Tabelle 6.5.: Übersicht über die Varianz, Anzahl Trainingsdatensätze und der durchschnittliche $F1_{pos,neg}$ pro Fremddomäne. Diese Tabelle dient als Grundlage für die nachfolgenden Streudiagramme.

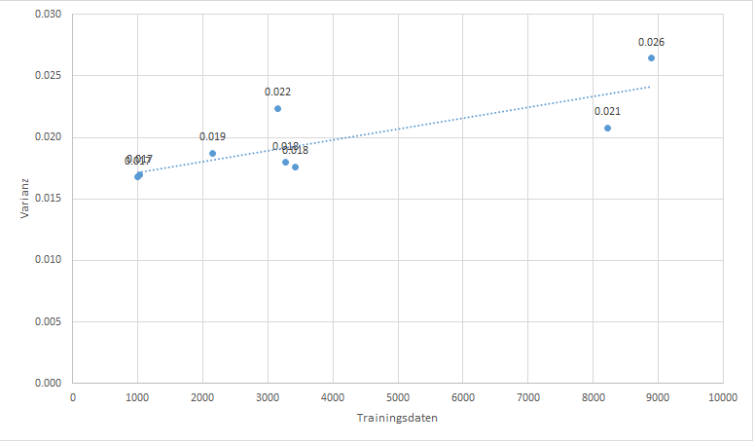


Abbildung 6.2.: Streudiagramm der Varianz in Relation zur Anzahl Trainingsdatensätze pro Fremddomäne mit einer linearen Trendgeraden.

	Varianz	Anzahl Trainingsdaten	Avg $F1_{pos,neg}$
DAI T	0.028	819	0.402
SEM R	0.028	855	0.372
DIL R	0.025	789	0.408
HUL Q	0.005	258	0.398
JCR N	0.009	2223	0.421
Seval T	0.007	3813	0.516
MPQ H	0.020	255	0.246
TAC R	0.012	537	0.466

Tabelle 6.6.: Übersicht über die Varianz, Anzahl Testdatensätze und der durchschnittliche $F1_{pos,neg}$ pro Zieldomäne. Diese Tabelle dient als Grundlage für die nachfolgenden Streudiagramme.

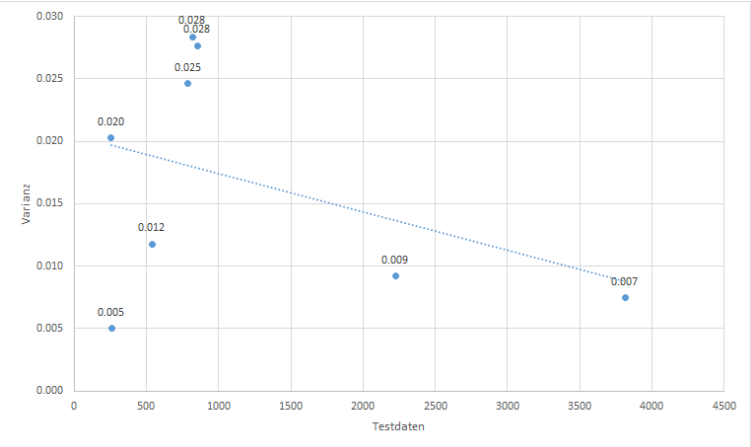


Abbildung 6.3.: Streudiagramm der Varianz in Relation zur Anzahl Testdaten pro Zieldomäne mit einer linearen Trendgeraden.

6.3. Augmentation

Mit den Augmentation Experimenten möchten wir herausfinden, ob es Domänen gibt, welche sich besser kombinieren lassen. Bei diesen Experimenten werden wir die Domäne *Union* nicht gesondert behandeln, sondern als eine normale Fremddomäne in die Experimente mit einbeziehen. Wir achten beim Durchmischen von Daten darauf, dass jede Domäne gleich stark vertreten ist. Man könnte auch argumentieren, dass anstelle jeder Domäne jede Textsorte gleich oft vertreten sein soll. Da wir aber nicht wissen, ob gleiche Textsorten sich gleich verhalten, wählen wir die Domäne als Unterscheidungskriterium.

6.3.1. Aufbau

Die Experimente sind grundsätzlich so aufgebaut, dass wir mit n Datensätzen einer Domäne starten und anschliessend schrittweise 0, $n/2$, n und $2n$ Trainingsdatensätze einer anderen Domäne für das Training des CNN hinzufügen. Nach jedem Schritt testen wir das CNN mit den Testdaten der TD. Bei Augmentation FD starten wir mit n TD Daten und fügen schrittweise Daten der FD hinzu. Bei Augmentation TD starten wir mit n FD Daten und fügen schrittweise Daten der TD dazu. Alle Experimente werden mit der besten Kombination aus Word-Embeddings und Distant-Phase für die TD durchgeführt (vgl. Kapitel 6.1). Das n ist in Abhängigkeit zur Datenmenge zu wählen. In der Tabellen 6.7 und 6.8 ist eine Übersicht, wie gross das n pro Experiment ist.

FD \ TD	*DAI* T	*DIL* R	*HUL* R	*JCR* Q	*MPQ* N	*Seval* T	*SEM* H	*TAC* R
DAI T	-	1637	1637	1032	1637	1637	1000	1637
DIL R	1710	-	1710	1032	1710	1710	1000	1710
HUL R	1578	1578	-	1032	1578	1578	1000	1578
JCR Q	516	516	516	-	516	516	516	516
MPQ N	3274	3420	3156	1032	-	4000	1000	2152
Seval T	3274	3420	3156	1032	4000	-	1000	2152
SEM H	500	500	500	500	500	500	-	500
TAC R	1076	1076	1076	1032	1076	1076	1000	-
Union	1637	1710	1578	516	4000	4000	500	1076

Tabelle 6.7.: Für die Experimentreihe Augmentation TD berechneten n pro Experiment.

FD \ TD	*DAI* T	*DIL* R	*HUL* R	*JCR* Q	*MPQ* N	SemEval T	*SEM* H	*TAC* R
DAI T	-	1710	1578	516	3274	3274	500	1076
DIL R	1637	-	1578	516	3420	3420	500	1076
HUL R	1637	1710	-	516	3156	3156	500	1076
JCR Q	1637	1032	1032	-	1032	1032	500	1032
MPQ N	1637	1710	1578	516	-	4000	500	1076
Seval T	1637	1710	1578	516	4000	-	500	1076
SEM H	1000	1000	1000	516	1000	1000	-	1000
TAC R	1637	1710	1578	516	2152	2152	500	-
Union	3274	3420	3156	1032	4000	4000	1000	2152

Tabelle 6.8.: Für die Experimentreihe Augmentation FD berechneten n pro Experiment.

6.3.2. Resultate Augmentation Allgemein

Das mischen mit einer FD erhöht die Performanz eines CNN nicht. In der Abbildung 6.4 sehen wir den Durchschnitt über alle $F1_{pos,neg}$ der Augmentation FD und Augmentation TD Experimente. Es ist ersichtlich, dass Augmentation FD tendenziell schlechter wird, umso mehr FD Daten hinzugefügt werden. Bei Augmentation TD ist es genau umgekehrt, der $F1_{pos,neg}$ verbessert sich mit steigender Anzahl TD Trainingsdaten wie in den Abbildungen A ersichtlich. Dieser Durchschnitt zeigt uns lediglich eine generelle Tendenz an, jedoch möchten wir herausfinden, ob sich gewisse Domänen mit Anderen kombinieren lassen. Deshalb untersuchen wir nun die beiden Experimente Augmentation FD und Augmentation TD einzeln pro TD und FD.

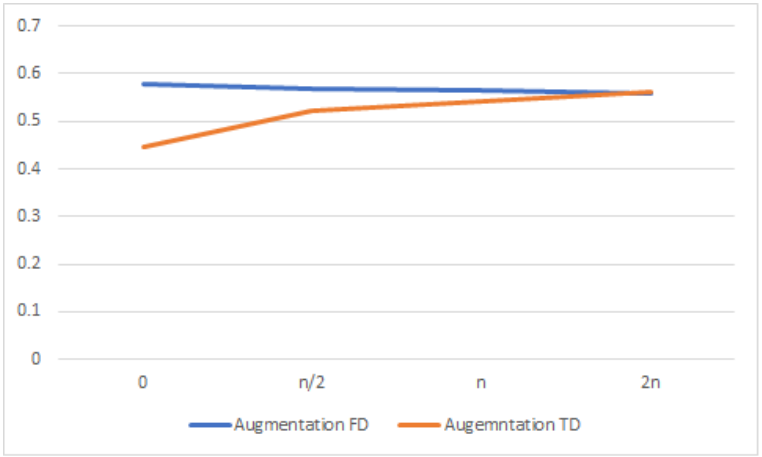


Abbildung 6.4.: Durchschnitt aller Augmentation FD und Augmentation TD Experimente

6.3.3. Resultate Augmentation TD

Domänen mit der gleichen Textsorte eigenen sich besser zum kombinieren. In den Abbildungen 6.5 und 6.6 für die Textsorte Tweets und den Abbildungen 6.7, 6.8 und 6.9 für die Textsorte Reviews ist ersichtlich, dass die Kombination von Domänen mit der gleichen Textsorte, zusätzlich auch die *Union* Domäne, immer eines der besten Ergebnisse liefert. Bei der Domäne TAC_*reviews* tritt dieses Verhalten schwächer auf und teilweise schneiden Domänen mit fremden Textsorten ebenfalls gut ab. Dass Domänen mit gleicher Textsorte sich besser eignen um sie zu kombinieren, erscheint aus menschlicher Sicht schlüssig, da gleiche Textsorten Ähnlichkeiten in der Struktur aufweisen. Da wir aber nicht genau wissen, auf welche Strukturen das CNN reagiert, ist dies für uns ein Erkenntnisgewinn. Wir werden im Kapitel 6.3.4 genauer untersuchen, wie sich die Resultate bei steigendem Anteil Datensätze der FD entwickelt.

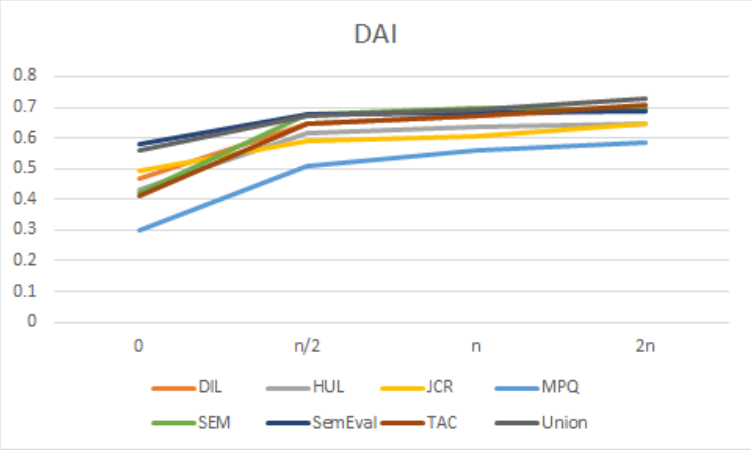


Abbildung 6.5.: Experiment Augmentation TD, Resultate pro FD, bei schrittweiser Erhöhung der TD Daten mit der Domäne *DAI_tweets*.

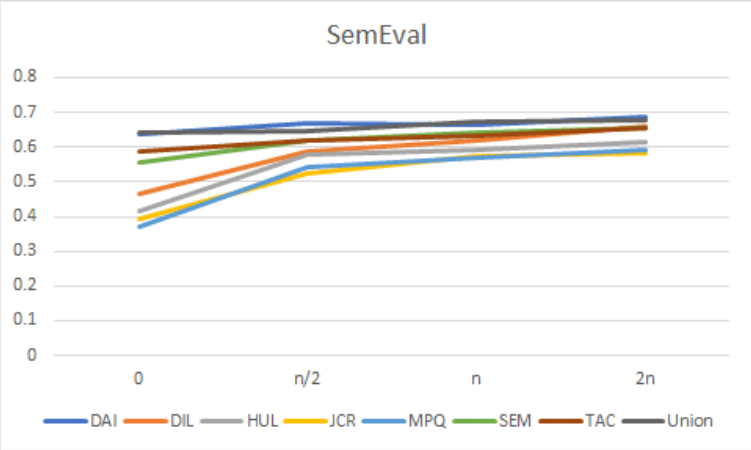


Abbildung 6.6.: Experiment Augmentation TD, Resultate pro FD, bei schrittweiser Erhöhung der TD Daten mit der Domäne *SemEval_tweets*.

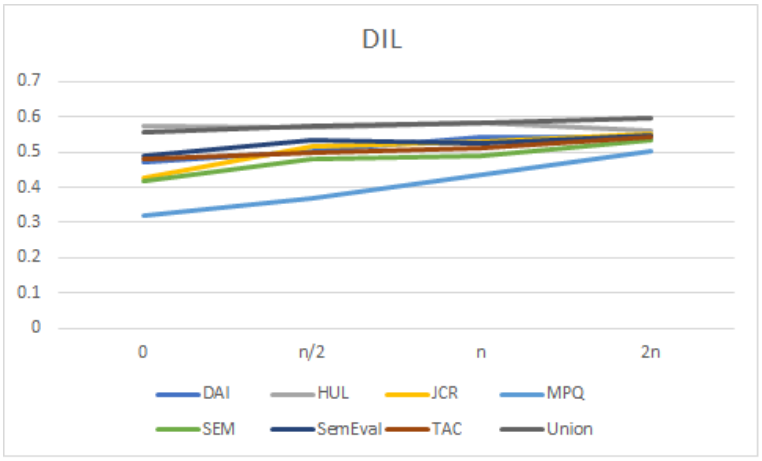


Abbildung 6.7.: Experiment Augmentation TD, Resultate pro FD, bei schrittweiser Erhöhung der TD Daten mit der Domäne *DIL_reviews*.

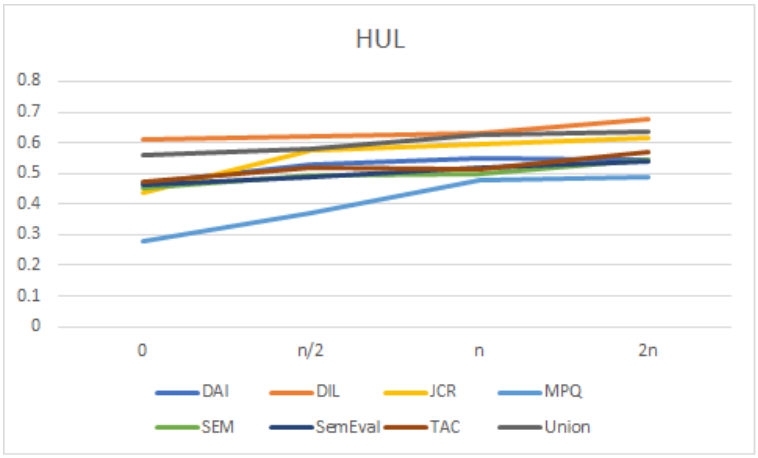


Abbildung 6.8.: Experiment Augmentation TD, Resultate pro FD, bei schrittweiser Erhöhung der TD Daten mit der Domäne *HUL_reviews*.

Es gibt deutliche Unterschiede in der Kombinationsfähigkeit der Domänen. Auffallend ist die schlechte Performanz, wenn wir als FD *MPQ_news* wählen. In der Abbildung 6.10 haben wir die Durchschnittswerte pro FD über alle Augmentation TD Experimente berechnet. Wir sehen hier, dass *MPQ_news* den tiefsten durchschnittlichen $F1_{pos,neg}$ erzielt. Dies kann nicht damit begründet werden, dass dies die einzige Domäne mit der Textsorte News ist, da auch andere Textsorten einmalig vorkommen und trotzdem im Schnitt deutlich besser sind als *MPQ_news*. Beim Durchsichten der Trainingsdaten ist uns bei dieser Domäne aufgefallen, dass einige Sätze doppelt vorkommen, jedoch mit unterschiedlichem Sentiment annotiert wurden. Dies könnte eine mögliche Ursache sein für die schlechte Performanz in der Kombination mit anderen Domänen. Die mögliche Ursache der suboptimalen Datenqualität verfolgen wir nicht weiter. Im Kontext mit Sentiment-Klassifizierer ist das Problem der Subjektivität eines Sentiments allgegenwärtig. Im Kapitel 6.3.4 werden wir das Verhalten der FD *MPQ_news* nochmals gesondert analysieren. Wir lesen aus der Abbildung 6.10 auch, dass die FD *Union* im Schnitt am besten abschneidet. Dies bestätigt noch einmal unsere bisherigen Beobachtungen, dass *Union* eine echte Option ist und im Bereich Crossdomain (vgl. Kapitel 6.2) fast immer bessere Ergebnisse

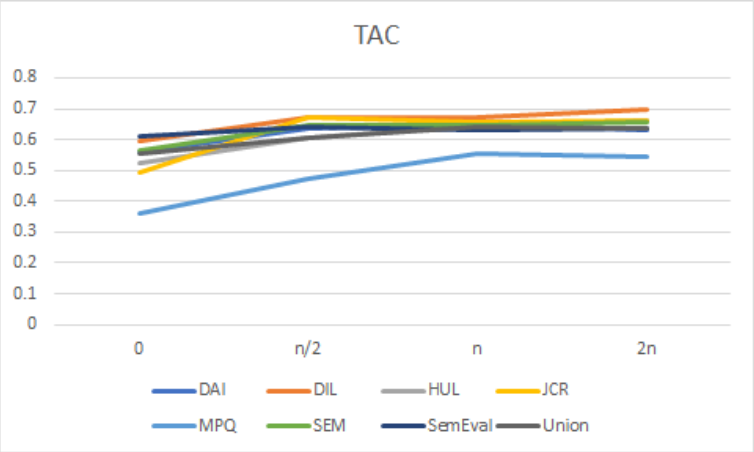


Abbildung 6.9.: Experiment Augmentation TD, Resultate pro FD, bei schrittweiser Erhöhung der TD Daten mit der Domäne *TAC_reviews*.

erzielt im Vergleich zu den anderen Domänen.

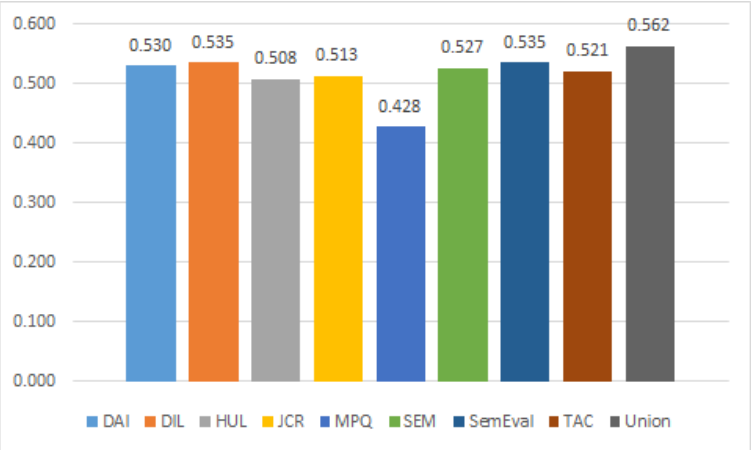


Abbildung 6.10.: Experiment Augmentation TD, Durchschnitt aller Ergebnisse pro FD

6.3.4. Resultate Augmentation FD

Die Domäne MPQ_news scheint Strukturen verschiedener Domänen aufzuweisen. Die Abbildung 6.11 zeigt die durchschnittlich erzielten $F1_{pos,neg}$ pro FD. Erstaunlicherweise erzielt nun die Domäne *MPQ_news* den höchsten Wert. Wir vermuten, dass die FD *MPQ_news* Strukturen verschiedener anderer Domänen aufweist und sie deshalb im Durchschnitt in diesem Experiment das beste Ergebnis erzielt. Wenn es jedoch darum geht, ein spezifisches CNN für eine bestimmte TD zu erstellen, von welcher wir keine annotierte Daten besitzen, haben wir in Abbildung 6.10 gesehen, dass sich *MPQ_news* nicht generell eignet.

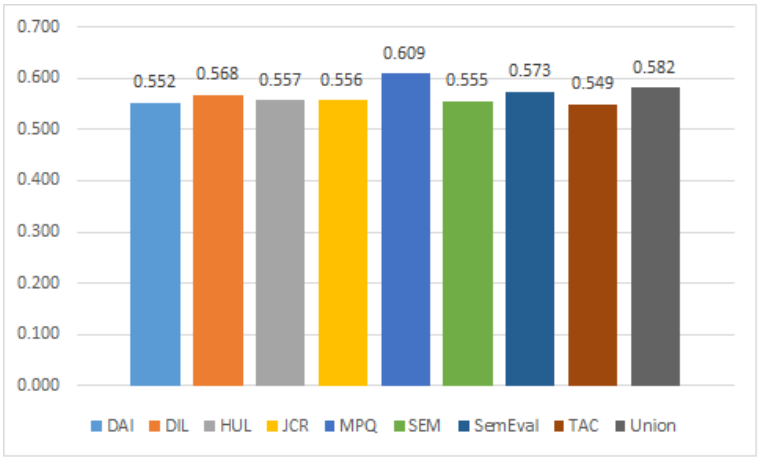


Abbildung 6.11.: Experiment Augmentation FD, Durchschnitt aller Ergebnisse pro FD

Die Domänen DAI_tweets und SemEval_tweets erzielen in Kombination gute Ergebnisse. In den Abbildungen 6.12 und 6.13 sehen wir, dass das hinzufügen einer Domäne mit gleicher Textsorte die Resultate nicht negativ beeinflusst. Im Vergleich zum Startwert verbessert sich die Qualität des Sentiment-Klassifizierers teilweise sogar. Dies bestätigt unsere Theorie, dass das Kombinieren von Domänen mit gleicher Textsorte keine wirkliche Verschlechterung für den Sentiment-Klassifizier zur Folge hat. Zusätzlich reagieren die beiden CNN aber auch gut auf die Kombination der FDs *MPQ_news* und *Union*.

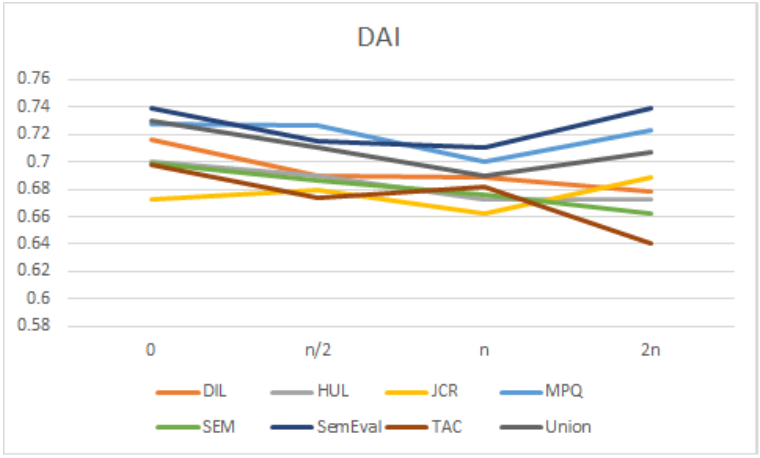


Abbildung 6.12.: Experiment Augmentation FD, Resultate pro FD, bei schrittweisem Hinzufügen von Datensätzen der FD mit der Domäne *DAI_tweets*

Die Textsorte Reviews bestätigen unsere Beobachtung bei Tweets nicht. Die Idee, dass gleiche Textsorten in der Kombination bessere Ergebnisse liefern als mit fremden Textsorten, ist in den Abbildungen 6.14, 6.15 und 6.16 für die Textsorte Reviews nicht ersichtlich. Die Domäne *TAC_reviews* scheint auf keine FD besser oder schlechter anzusprechen, die Ergebnisse verschlechtern sich bei allen FD gleich stark. Die Domäne *HUL_reviews* zeigt ein eigenartiges Verhalten, sie reagiert auf alle FD ziemlich identisch.

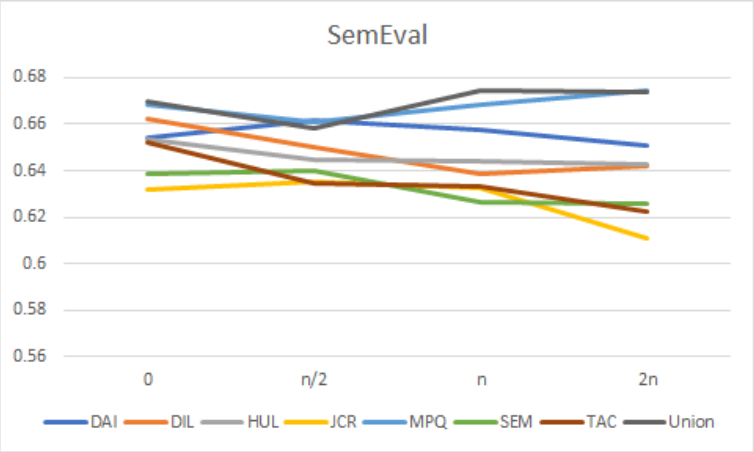


Abbildung 6.13.: Experiment Augmentation FD, Resultate pro FD, bei schrittweisem Hinzufügen von Datensätzen der FD mit der Domäne *SemEval_tweets*

Einzig die Domäne *DIL_reviews* scheint zumindest auf die FD *Union* positiv zu reagieren, indem sich der $F1_{pos,neg}$ leicht verbessert. Aufgrund dieses Verhaltens können wir bei den Reviews keine generelle Aussage bezüglich der Kombinationsfähigkeit machen.

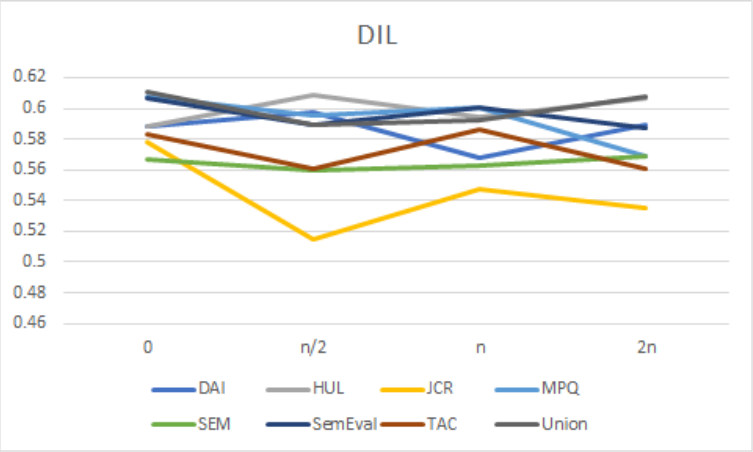


Abbildung 6.14.: Experiment Augmentation FD, Resultate pro FD, bei schrittweisem Hinzufügen von Datensätzen der FD mit der Domäne *DIL_reviews*

Die Domäne JCR_quotations ist als FD ungeeignet. In den Abbildungen 6.12, 6.13, 6.14, 6.17 und 6.18 erzielt die FD *JCR_quotations* die deutlich schlechtesten Ergebnisse. Die Ursache ist uns nicht ganz klar, wenn wir jedoch die Domäne *SEM_headlines* betrachten, erzielt auch diese FD schlechte Ergebnisse. Unsere Vermutung ist, dass die kleinen Datensätze der beiden Domänen die Ursache für die schlechte Kombinationsfähigkeit ist. Da die Anzahl Datensätze der TD in Relation mit der Anzahl Datensätze der FD steht, könnte sich die geringe Anzahl Trainingsdaten und Testdaten negativ auswirken. Man müsste dies mit weiteren Experimenten genauer untersuchen. Man könnte zum Beispiel

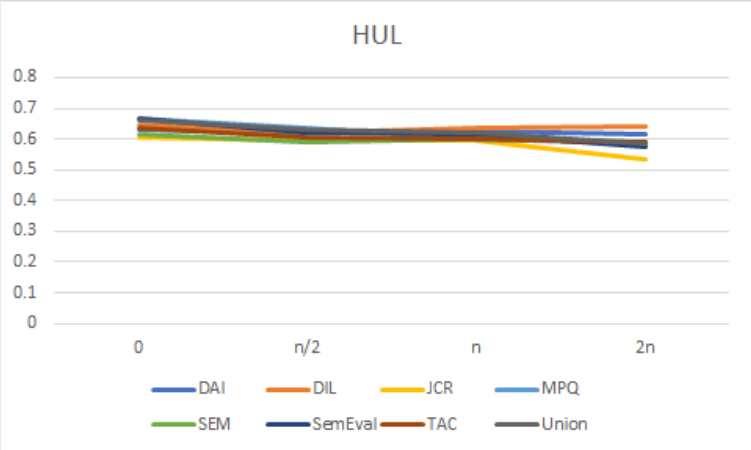


Abbildung 6.15.: Experiment Augmentation FD, Resultate pro FD, bei schrittweisem Hinzufügen von Datensätzen der FD mit der Domäne *HUL_reviews*

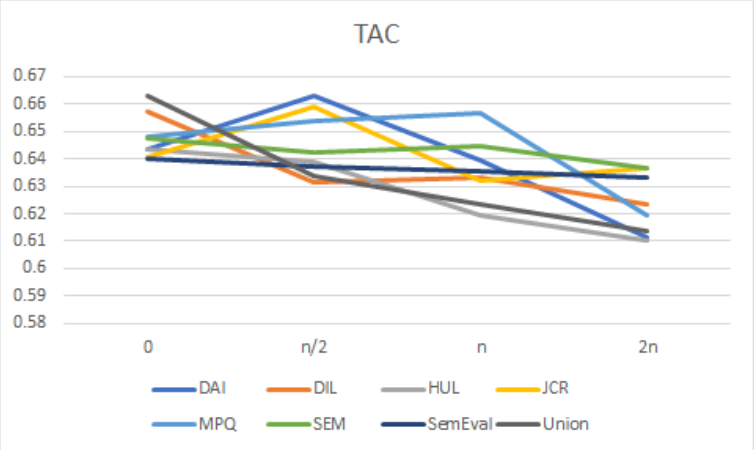


Abbildung 6.16.: Experiment Augmentation FD, Resultate pro FD, bei schrittweisem Hinzufügen von Datensätzen der FD mit der Domäne *TAC_reviews*

die Datensätze der TD für alle Experimente gleich setzen und anschliessend noch einmal die Ergebnisse vergleichen.

6.4. Ablation

Wir haben mehrfach beobachtet, dass ein *Union*-CNN im Vergleich mit allen anderen CNNs gute Ergebnisse liefert. In den *Union* Trainingsdaten befinden sich jedoch auch Daten der TD. Wir möchten nun herausfinden, ob diese Beobachtung auch zutrifft, wenn die TD gänzlich unbekannt ist. Als Beispiel für die TD *DAI_tweets* gehen wir folgendermassen vor: Wir kombinieren alle Trainigsdaten von allen Domänen, ausser diejenigen der TD *DAI_tweets*. Wir nennen diese neue Domäne *Ablation_DAI*.

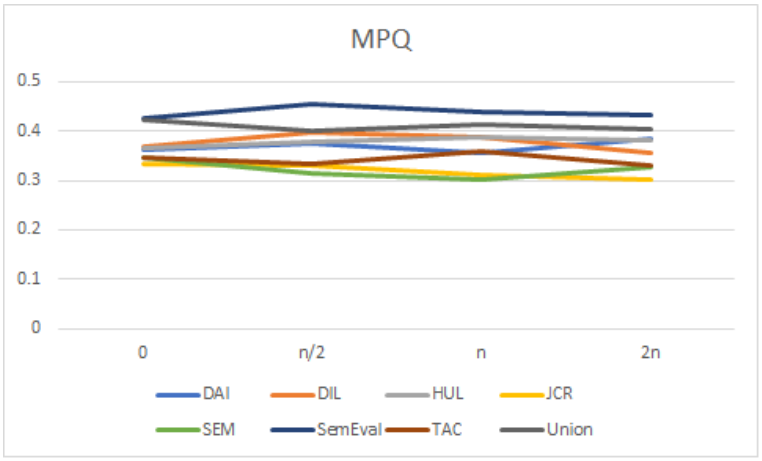


Abbildung 6.17.: Experiment Augmentation FD, Resultate pro FD, bei schrittweisem Hinzufügen von Datensätzen der FD mit der Domäne *MPQ_news*

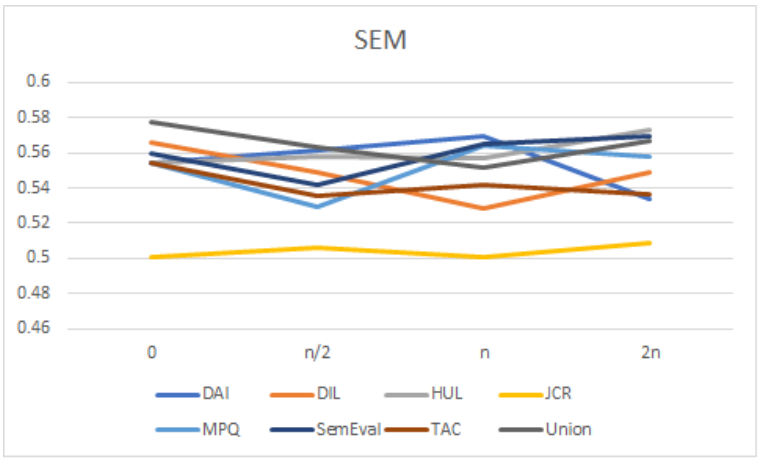


Abbildung 6.18.: Experiment Augmentation FD, Resultate pro FD, bei schrittweisem Hinzufügen von Datensätzen der FD mit der Domäne *SEM_headlines*

6.4.1. Aufbau

Wir trainieren ein CNN pro Ablation Domäne, wobei wir die beste Kombination aus Word-Embeddings und Distant-Phase pro TD als Basis verwenden. Für die Validierung während des Trainings benutzen wir die Testdaten der TD. Damit wir einen fairen Vergleichswert haben, werden die *Union* Experimente aus Kapitel 6.2 wiederholt; dabei benutzen wir auch hier jeweils die beste Kombination aus Word-Embeddings und Distant-Phase für jede TD.

6.4.2. Resultate

Die Ablation Experimente erzielen brauchbare Ergebnisse. Dass die $F1_{pos,neg}$ der Ablation Experimente teilweise besser sind als diejenigen der *Union* Experimente, überrascht

uns. Jedoch liegen die Ergebnisse mit +/- 5 Punkten relativ nahe beieinander, ausgenommen bei der Domäne *MPQ_news*, mit einer Differenz von 14 Punkten. Vermutlich müsste man das gesamte Experiment mit 3 Datensätzen wiederholen, wobei ein Datensatz für das Training, ein Weiterer für die Validierung und der Dritte für das anschliessende Testen verwendet wird. Trotzdem unterstützen diese Ergebnisse unsere These, dass sich ein *Union*-CNN für eine neue, unbekannte Domäne eignet. Wir haben eine viel geringere Varianz, als wenn wir zufällig ein für eine TD spezialisiertes CNN als Sentiment-Klassifizierer wählen.

	Ablation ohne TD	*Union*	Diff.
DAI T	0.658	0.725	0.067
DIL R	0.573	0.554	-0.019
HUL R	0.566	0.528	-0.038
JCR Q	0.485	0.465	-0.020
MPQ N	0.404	0.550	0.147
Seval T	0.658	0.690	0.032
SEM H	0.426	0.422	-0.005
TAC R	0.558	0.614	0.056

Tabelle 6.9.: Vergleich *Union* mit Ablation Experimenten und die Differenz der $F1_{pos,neg}$

7. Schlussfolgerungen

Mit unserem ersten Experiment im Kapitel 6.1 untersuchten wir den Einfluss von Word-Embeddings und Distant-Phases pro Domäne. Der $F1_{pos,neg}$ variiert pro Domäne um bis zu 15 Punkte. Es gab keine Word-Embeddings oder Distant-Phase welche für alle Domänen das beste Ergebnis erzielten. Wir konnten aufzeigen, dass eine Distant-Phase bei allen Domänen einen positiven Effekt hat. Einzige Ausnahme war die Domäne *MPQ_news*, welche das beste Ergebnis ohne Distant-Phase erzielte. Jedoch ist der Wert nur um 2 Punkte besser als der Zweitbeste. Wenn wir die Durschnittswerte pro Word-Embedding betrachten wird klar, dass die Domäne *MPQ_news* eine Ausnahme war. Pro Word-Embedding erzielten diejenigen Experimente ohne Distant-Phase immer die schlechtesten Durchschnittswerte. Auch die Verwendung von zufälligen Word-Embeddings war keine generelle Option. Lediglich bei der Domäne *TAC_reviews* erhielten wir damit das beste Resultat. Jedoch war auch hier das beste Ergebnis nur 2 Punkte besser als das Zweitbeste, welches mit Tweets Word-Embeddings durchgeführt wurde. Die Wahl der Word-Embeddings und Distant-Phase hat einen grossen Einfluss auf die Ergebnisse. Zusätzlich scheint die Wahl nicht mit der Textsorte zu korrelieren sondern hängt von der Domäne ab und muss für jede einzeln eruiert werden.

Im Kapitel 6.2 testeten wir ein für eine Domäne spezialisiertes CNN auf allen anderen Domänen. Ziel war es einen ersten Eindruck zu erlangen, wie sich die verschiedenen Domänen in Crossdomain Experimenten verhalten. Die Varianz in den Ergebnissen, mit Ausnahme des *Union*-CNN welches wir separat behandeln, war auffallend gross. Es eignet sich offensichtlich keine Domäne besser, um einen generalistischen Sentiment-Klassifizierer zu trainieren. Wir untersuchten ansatzweise mehrere potenzielle Ursachen für die hohe Varianz. Ein Teil der Varianz könnte mit kleinen Testdatensätzen erklärt werden, jedoch tritt das Verhalten auch bei grösseren Testdatensätzen auf. Aufgrund dieser Beobachtung muss es noch mindestens eine weitere Ursache für diese Varianz geben. Deshalb untersuchten wir die Datensätze der verschiedenen Domänen. Wir definierten und ordneten den Datensätzen zwei Eigenschaften zu: Die erste Eigenschaft ist die Ausprägung des Sentiments und die zweite Eigenschaft die Konzentration in welcher der Sentiment vorliegt. Beim Durchsichten der Datensätze bestätigte sich diese Vermutung, ein paar Beispiele sind in der Tabelle 6.3 ersichtlich. Dies erklärt, weshalb gewisse Domänen öfters gute Ergebnisse erzielen als andere. Die Performanz eines spezialisierten Sentiment-Klassifizierers variiert so stark, dass keine generelle Aussage getroffen werden kann bezüglich der Kriterien, bei welchen Domänen eine Kombination sinnvoll ist. Die Qualität der Ergebnisse hängt also signifikant von der gewählten Kombination von Domänen ab.

Aufgrund der vorhin definierten Eigenschaften führten wir die "Augmentation"-Experiment im Kapitel 6.3 durch. Wir wollten herausfinden, wie sich die Performanz eines Sentiment-

Klassifizierer verändert, wenn wir Trainingsdaten einer anderen Domäne für das Training hinzufügten. Generell verschlechtert sich dabei die Performanz. Bei einzelnen Experimenten jedoch wurden die Resultate nicht schlechter, teilweise verzeichneten wir eine leichte Steigerung. Eine Steigerung der Performanz trat dann auf, wenn wir Domänen mit der gleichen Textsorte kombinierten, wobei bei der Textsorte Tweets dieses Verhalten eindeutig war. Bei der Textsorte Reviews folgte die Domäne *TAC_reviews* unserer Vermutung nicht. Dies könnte mit den vorherig beschriebenen Eigenschaften, wie Ausprägung und Konzentration des Sentiments, zusammenhängen. Bei Tweets treten beide Eigenschaften in starker Form auf. Bei den Reviews variiert diese Ausprägung. Wir vermuten, dass Domänen welche diese beiden Eigenschaft stark aufweisen, sich besser kombinieren lassen. Hat man jedoch genug annotierte Daten einer Domäne, lohnt sich das Erweitern mit einer fremden Domäne nicht.

Wir beobachteten bei allen Experimenten, dass die Varianz im Zusammenhang mit *Union* deutlich geringer war im Vergleich zu allen anderen Domänen. Aufgrund dieser positiven Ergebnisse führten wir die Experimente Ablation im Kapitel 6.4 durch. Dabei entfernten wir im Datensatz *Union* die Daten der Zieldomänen um eine Verfälschung der Resultate zu vermeiden. Die Ergebnisse wichen nur gering von den *Union* Ergebnissen ab. Es bestätigt sich die Vermutung, dass man bei fehlenden Daten der Zieldomäne, welche unsere oben aufgeführten Eigenschaften nicht besitzen oder für eine unbekannte Zieldomäne, mit einem *Union*-CNN die besten Ergebnisse erzielt. Somit gibt es einen generalistischen Sentiment-Klassifizierer, welcher auf mehrere Domänen gute Ergebnisse erzielt.

Sofern wir einen Sentiment-Klassifizierer für eine Domäne benötigen, für welche wir keine annotierten Daten besitzen, scheint es abhängig von der Textsorte der neuen Domäne zwei Möglichkeiten zu geben. Sofern die Textsorte die Eigenschaften starke Ausprägung und hohe Konzentration des Sentiments aufweisen, können Trainingsdaten von einer Domäne mit gleicher Textsorte verwendet werden. Besitzt die neue Domäne nur eine dieser Eigenschaften, können wir keine Empfehlung zur Wahl der Domäne abgeben. In diesem Fall scheint es die beste Option zu sein, einen *Union* Sentiment-Klassifizierer einzusetzen.

Wenn die Zieldomäne unbekannt ist oder Daten mehrerer Domänen mit einem Sentiment-Klassifizierer klassifiziert werden müssen, empfehlen wir ebenfalls die Wahl eines *Union* Sentiment-Klassifizierers.

Weiterführende Arbeiten Wir möchte zum Schluss ein paar Anknüpfungspunkte an diese Arbeit mit dem Leser teilen. Dabei handelt es sich um offene Fragen, welche im Rahmen dieser Arbeit nicht geklärt werden konnten, wir aber im Zusammenhang Crossdomain als sinnvoll betrachten:

- Es benötigt noch weitere Untersuchungen bezüglich der Domäne *MPQ_news* und *Union*. Bei den Experimenten Crossdomain 6.2 war der $F1_{pos,neg}$ beim Testen mit *MPQ_news* auf dem *Union* Sentiment-Klassifizierer als einzige Domäne besser, wie auf dem eigenen Sentiment-Klassifizierer.

- Unsere These, dass ähnliche Textsorten sich besser zum kombinieren eignen, müsste mit mehr Experimenten bestätigt werden.

- Die Vermutung dass die Eigenschaften Ausprägung und Konzentration des Sentiments, einen direkten Einfluss auf die Qualität des Sentiment-Klassifizierers haben, muss weiter geprüft werden. Vermutlich könnte es sich hier lohnen diese Thematik mit Linguisten anzugehen.

- Wir starteten gegen Ende der Arbeit noch mit einer Hauptkomponentenanalyse (PCA), konnten diese aber nicht mehr rechtzeitig abschliessen. Es könnte sich lohnen, mit allen Faktoren und über alle Experimente hinweg, eine PCA durchzuführen. Wir erhoffen davon weitere Hinweise, inwiefern die verschiedenen Faktoren einen Einfluss auf den $F1_{pos,neg}$ haben.

- Ebenfalls versuchten wir am Ende, mehrere der besten CNNs pro Domäne (vgl. Kapitel 6.1) zu einem "Experten-CNN" zu kombinieren. Die Idee dabei ist, die bestehenden CNNs mittels einer Merge-Schicht[1] zu kombinieren um einen Sentiment-Klassifizierer zu erhalten, welchem das gesamte "Wissen" der Klassifizierer der einzelnen Domänen zu Verfügung steht. Unsere Tests zeigten, dass dieses Vorgehen durchaus Potential haben könnte: Unsere Resultate zeigten auf den einzelnen Domänen einen Anstieg des $F1_{pos,neg}$ von bis zu 7.5 Punkten. Allerdings bedarf es auch hier weiterer Untersuchungen um festzustellen, ob dieses Vorgehen generell sinnvoll ist.

[1] https://keras.io/layers/core/#merge

Appendix

A. Ergebnisse Experiment Augmentation TD

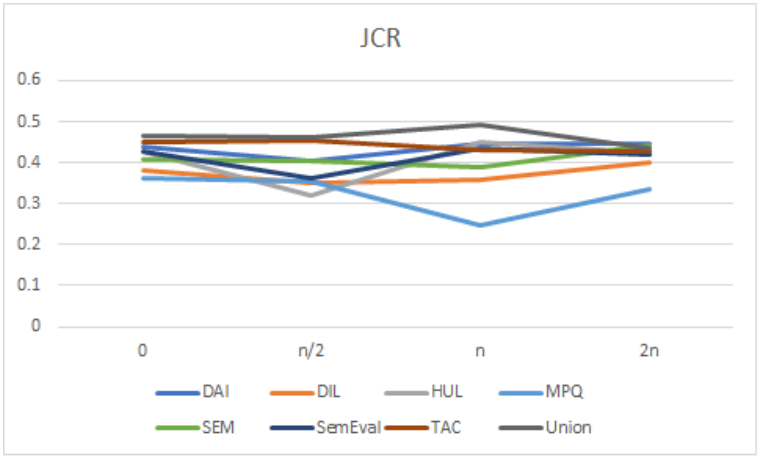


Abbildung A.1.: Experiment Augmentation TD, Resultate pro FD, bei schrittweiser Erhöhung der TD Daten mit der Domäne *JCR_quotations*

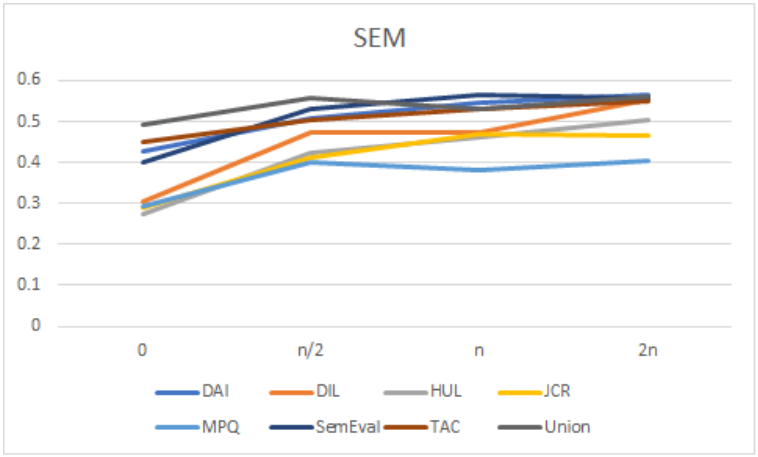


Abbildung A.2.: Experiment Augmentation TD, Resultate pro FD, bei schrittweiser Erhöhung der TD Daten mit der Domäne *SEM_headlines*

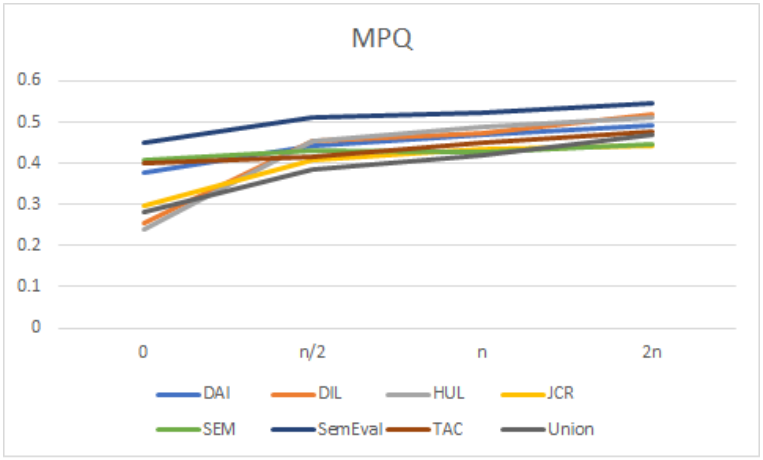


Abbildung A.3.: Experiment Augmentation TD, Resultate pro FD, bei schrittweiser Erhöhung der TD Daten mit der Domäne *MPQ_news*

B. Verwendung des Software-Systems

Im folgenden Kapitel wird erläutert, wie das im Rahmen dieser Arbeit implementierte Software-System (vgl. Kapitel 3) verwendet werden kann.

B.1. Download

Der Code des Software-Systems kann mittels `git`[1] heruntergeladen werden. Das Repository ist über den GitHub-Server der ZHAW verfügbar[2].

B.2. Voraussetzungen

Um die Software zu verwenden, müssen die folgenden Software-Pakete installiert sein:

- `python` in der Version 3.5.2[3]
- `anaconda` Toolkit in der Version 4.2.0[4]
- Sofern eine Nvidia GPU verwendet werden möchte:
  - Nvidia GPU Treiber[5] für installierte GPU
  - Nvidia `cuda` 8 Toolkit[6]

Die Experimente und Scripts können auch ohne GPU durchgeführt werden. Die Berechnungen finden dann auf der CPU des jeweiligen Systems statt. Allerdings führt das bei vielen Teilen des Systems zu deutlich höheren Laufzeiten (z.B. Training von CNN, Generieren von Word-Embeddings, ...).

Zusätzlich zu den oben aufgeführten Software-Paketen müssen die folgenden `python` Bibliotheken in der richtigen Version installiert sein:

[1] https://git-scm.com/
[2] https://github.engineering.zhaw.ch/vongrdir/PA-ML-2016
[3] https://www.python.org/
[4] https://www.continuum.io/downloads
[5] http://www.nvidia.de/Download/index.aspx
[6] https://developer.nvidia.com/cuda-toolkit

- `numpy` Version 1.11.1
- `theano` Version 0.8.2
- `keras` Version 1.1.0
- `nltk` Version 3.2.1
- `scikit_learn` Version 0.18
- `matplotlib` Version 1.5.3
- `gensim` Version 0.12.4
- `h5py` Version 2.6.0
- `flask` Version 0.11.1

Diese Bibliotheken können über das `conda`[7] Tool von `anaconda` oder den `python` Biblitheksmanager `pip`[8] installiert werden.

Um Kompatibilitätsprobleme zu vermeiden, wird empfohlen exakt die aufgeführte Version der jeweiligen Bibliothek zu installieren. Es kann allerdings gut sein, dass das Software-System auch mit anderen Versionen fehlerfrei funktioniert.

[7] http://conda.pydata.org/docs/using/pkgs.html
[8] https://packaging.python.org/installing/

B.3. Aufbau des Repository

Im Folgenden wird erläutert, wie die Struktur des Repositories aufgebaut ist. Dabei wird auf die Bedeutungen der einzelnen Ordner und deren Inhalt eingegangen.

Name des Ordner	Inhalt
`configs/`	Im Ordner `configs/` werden die JSON Konfigurationen aller Experimente abgelegt. Diese sind nach `group_id` in einzelnen Unterordner gruppiert.
`docs/`	Ist der Ordner, in welchem alle Dokumente, welche für das Erstellen des Berichts benötigt werden, liegen.
`embeddings/`	Ist standardmässig leer, sollte verwendet werden um die Word-Embeddings, welche für die Experimente benötigt werden, darin abzuspeichern.
`preprocessed/`	In diesem Ordner werden die vorverarbeiteten Trainingsdaten für die Distant-Phase abgelegt. Diese werden mittels des `preprocess_tsv_data.py`-Skripts erstellt.
`results/`	Hier werden die Resultate aller durchgeführten Experimente abgespeichert. Wie bereits bei den Konfigurationen werden die Resultate hier nach `group_id` gruppiert. Jedes Experiment erhält einen Ordner in welchem das finale CNN, die gesammelten Evaluierungs-Metriken sowie die Konfiguration des Experiments abgespeichert werden.
`scripts/`	Im Ordner `scripts/` liegen alle Skripts, welche nachfolgend erläutert werden.
`source/`	Hier liegt der eigentliche Source-Code des Systems. Darin befinden sich alle Teile, welche im Kapitel 3.6 erläutert wurden. Ausserdem befinden sich dort diverse `python` Module (z.B. `data_utils`), welche ebenfalls im Rahmen der Skripts benötigt werden.
`testdata/`	Alle Trainings-/Test-Datensätze, welche für die Experimente benötigt werden, liegen in diesem Ordner.
`vocabularies/`	Die zu den Word-Embeddings im Ordner `embeddings/` gehörenden Vokabulare.
`web/`	Im Ordner `web/` liegt der Source-Code der Weboberfläche.

Tabelle B.1.: Erläuterungen zum Aufbau des Repository.

B.4. Verwendung der Skripte

Im Folgenden werden die wichtigsten Skripte des Systems, welche zur Durchführung der Experimente notwendig sind, erläutert. Die Skripte selber liegen im Ordner `scripts/` des Repositories.

Name	Beschreibung
`aggregate_results_metrics.py`	Mithilfe dieses Skripts können die Resultate eines oder mehrerer Experimente aggregiert werden. Diese werden dann in eine CSV Datei im aktuellen Arbeitsverzeichnis abgelegt.
`extract_embeddings.py`	Word-Embeddings, welche mit `word2vec` erstellt wurden, müssen ausgepackt werden bevor diese in den Experimenten verwendet werden können. Dieses Skript ist dafür zuständig.
`extract_embeddings_glove.py`	Word-Embeddings, welche mit `glove` erstellt wurden, müssen ausgepackt werden bevor diese in den Experimenten verwendet werden können. Dieses Skript ist dafür zuständig.
`extract_embeddings_vocabulary.py`	Das Vokabular von Word-Embeddings können mithilfe dieses Skripts extrahiert werden.
`generate_data_statistics.py`	Mithilfe dieses Skripts können Statistiken ein oder mehrerer TSV Dateien erstellt werden. Dabei werden die darin enthaltenen Texte und Sentiments analysiert.
`preprocess_tsv_data.py`	Mithilfe dieses Skripts können Daten aus einer TSV-Datei vorverarbeitet und im HDF5 Format abgespeichert werden. Dies ist nötig um z.B. die Daten für die Distant-Phasen vorzubereiten.

Tabelle B.2.: Beschreibungen der wichtigsten Skripte zur Verwendung Software-Systems.

Es sind noch einige zusätzliche Skripte im Ordner `scripts/` vorhanden (z.B. für Erstellung von Plots, Verwaltung von Experimenten, etc.). Allerdings werden nicht alle benötigt um die Experimente durchzuführen und deswegen werden hier nur die wichtigsten erläutert.

In der folgenden Tabelle sind Beispielaufrufe für jedes Skript aufgelistet:

Name	Bespielaufruf
`aggregate_results_metrics.py`	python scripts/aggregate_results_metrics.py results/my-experiment-group/my-experiment
`extract_embeddings.py`	python scripts/extract_embeddings.py -e embeddings/my-embeddings -v vocabularies/vocabulary.pickle -o embeddings/my-extracted-embeddings
`extract_embeddings_glove.py`	python scripts/extract_embeddings_glove.py -e embeddings/my-glove-embeddings -v vocabularies/vocabulary.pickle -o embeddings/my-extracted-embeddings
`extract_embeddings_vocabulary.py`	python scripts/extract_embeddings_vocabulary.py.py -e embeddings/my-glove-embeddings -v vocabularies/vocabulary.pickle -o embeddings/my-extracted-embeddings
`generate_data_statistics.py`	python scripts/generate_data_statistics.py testdata/my-data.tsv
`preprocess_tsv_data.py`	python scripts/preprocess_tsv_data.py -t my-data.tsv -o preprocessed.h5 -s 140 -v vocabularies/my-vocabulary.pickle -m 1000000

Tabelle B.3.: Beispielaufrufe der wichtigsten Skripte zur Verwendung Software-Systems.

B.5. Durchführung von Experimenten

Um ein Experiment durchzuführen wird eine Konfiguration im JSON Format benötigt. Diese liegen alle im Ordner `configs/`. Im Folgenden werden die einzelnen Konfigurationsparameter erläutert:

Parametername	Standartwert	Beschreibung
`batch_size`	`500`	Mithilfe dieses Parameters kann konfiguriert werden mit welcher Batch-Size das Training des CNN durchgeführt wird.
`early_stopping_monitor_metric`	`val_f1_score_pos_neg`	Damit wird konfiguriert, welche Metrik durch das Early-Stopping überwacht werden soll.
`early_stopping_patience`	`75`	Über diesen Parameter kann konfiguriert werden wieviel Epochen ohne Fortschritt bezüglich der `early_stopping_monitor_metric` Metrik trainiert wird, bevor das Training abgebrochen wird.
`max_sent_length`	`140`	Mithilfe dieses Parameters kann angegeben werden, wieviele Wörter pro Satz maximal erlaubt sind. Längere Sätze werden demnach gekürzt.
`nb_epoch`	`1000`	Dieser Parameter ist dafür zuständig, über wieviel Epochen hinweg das Training durchgeführt wird.
`nb_kfold_cv`	`4`	Wenn dieser Parameter vorhanden ist, wird während des Trainings k-fold Cross-Validation verwendet. Bei einem Parameterwerte kleiner als 2 wird keine k-fold Cross-Validation durchgeführt.
`early_stopping_patience`	`75`	Über diesen Parameter kann konfiguriert werden wieviel Epochen ohne Fortschritt bezüglich der `early_stopping_monitor_stopping` trainiert wird, bevor das Training abgebrochen wird.
`name`	`null`	Muss gesetz werden, da das Experiment dem Namen entsprechend im `results/` Verzeichnis abgelegt wird.
`group_id`	`null`	Muss gesetzt werden, da Experimente nach diesem Namen im `results/` Verzeichnis gruppiert werden.
`model_json_path`	`null`	Gibt den Pfad zur JSON Datei mit dem zu ladenden `keras` Modell an. Darf nur angegeben werden sofern auch ein Wert für `model_weights_path` gesetzt ist, sonst tritt beim Start ein Fehler auf.
`model_id`	`1`	Damit können verschiedene Versionen des verwendeten CNN geladen werden (z.B. unterschiedliche Anzahl Convolutional Schichten). Welche ID für welches Modell steht ist in der Datei `source/model.py` ersichtlich.
`model_weights_path`	`null`	Gibt den Pfad zur HDF5 Datei mit den zu ladenden Gewichten an. Darf nur angegeben werden sofern auch ein Wert für `model_json_path` gesetzt ist.

Tabelle B.4.: Erklärungen der einzelnen Konfigurationsparameter des Software-Systems. (Teil 1)

Parametername	Standartwert	Beschreibung
monitor_metric	val_f1_score_pos_neg	Mithilfe dieses Parameters kann konfiguriert werden, welche Metrik ausschlaggebend ist um am Ende des Trainings zu entscheiden, welcher der beste Fold war. Wird nur beachtet wenn k-fold Cross-Validation aktiviert ist.
monitor_metric_mode	max	Mithilfe dieses Parameter wird konfiguriert ob die monitor_metric nach dem maximalen oder minimalen Wert überwacht werden soll.
preprocessed_data	null	Hier kann die HDF5 Datei, aus welcher die Trainingsdaten stammen, referenziert werden. Dann werden die Parameter validation_data_path sowie test_data ignoriert. Ausserdem muss der Parameter samples_per_epoch gesetzt werden.
randomize_test_data	true	Wenn diser Parameter auf true gesetzt wird, werden die geladenen Trainingsdaten zufällig durchmischt.
use_random_embeddings	false	Falls dieser Parameter auf true gesetzt wird werden die Word-Embeddings zufällig initialisiert.
set_class_weights	false	Entscheidet ob während des Trainings Klassengewichte verwendet werden sollen.
samples_per_epoch	0	Falls die Parameter tuse_preprocessed_data und preprocessed_data gesetzt sind, muss hier angegeben werden wieviel Datensätze aus den vorverarbeiteten Daten verwendet werden sollen.
test_data	null	Über diesen Parameter kann konfiguriert werden, welche Trainingsdaten verwendet werden sollen. Dabei muss entweder der Pfad zu einer TSV Datei angegeben werden, oder ein Objekt welches als Schlüssel die Pfade zu den zu ladenden TSV Dateien und als Werte die zu ladenden Anzahl Datensätze enthält. Wird ignoriert wenn vorverarbeitete Daten verwendet werden.
use_preprocessed_data	false	Muss auf true gesetzt werden falls der Parameter preprocessed_data gesetzt wurde.
validation_data_path	null	Über diesen Parameter wird konfiguriert, welche Daten für die Validierung während und am Ende des Trainings verwendet werden sollen. Der Aufbau ist dabei derselbe wie beim Parameter test_data. Wird ignoriert wenn vorverarbeitete Daten verwendet werden.
validation_split	0.0	Dieser Parameter gibt an, welcher Bruchteil der Trainingsdaten als Validierungsdaten verwendet werden sollen. Falls der Parameter validation_data_path gesetzt ist wird dieser Parameter ignoriert.

Tabelle B.5.: Erklärungen der einzelnen Konfigurationsparameter des Software-Systems. (Teil 2)

Die obigen Erläuterungen sind nicht abschliessend oder vollständig. Für die genaue Funktionsweise der weiteren Parameter wird empfohlen den Source-Code hinzu zu konsultieren.

Eine Beispiel-Konfiguration für ein einfaches Experiment sieht wie folgt aus:

```
{
  "test_data": {
    "testdata/MPQ_reviews_full.tsv": 3000,
    "testdata/DIL_Reviews.tsv": 3000
  },
  "name": "my-fancy-experiment",
  "group_id": "really-cool-group",
  "nb_epoch": 1000,
  "nb_kfold_cv": 4,
  "vocabulary_embeddings":
     "embeddings/emb_smiley_tweets_embedding_english_590M.npy",
  "validation_data_path": "testdata/MPQ_News.tsv",
  "vocabulary_path": "vocabularies/vocab_en300M_reduced.pickle"
}
```

Eine Beispiel-Konfiguration um eine Distant-Phase mit vorverarbeiteten Daten durchzuführen kann wie folgt aussehen:

```
{
  "group_id": "amazon_distant_model",
  "use_preprocessed_data": true,
  "preprocessed_data":
     "preprocessed/amazon_distant_train_preprocessed.hdf5",
  "samples_per_epoch": 82400000,
  "test_data": "testdata/amazon_distant_train.tsv",
  "name": "amazon_distant_model_82M_dedup_v2",
  "nb_epoch": 1,
  "max_sent_length": 140,
  "batch_size": 1000,
  "vocabulary_embeddings":
     "embeddings/word2vec_embeddings_en_news_emb.npy",
  "vocabulary_path": "vocabularies/vocab_news_emb.pickle"
}
```

Um ein Experiment zu starten sollte das `run.sh` Skript im Repository verwendet werden. Dabei können als Parameter alle JSON Konfigurationen jener Experimente mitgegeben werden, welche durchgeführt werden sollen. Wenn im `configs/` Verzeichnis die beiden Konfigurationen `config-1.json` und `config-2.json` liegen würden, könnten diese mit folgendem Befehl gestartet werden:

```
$ ./run.sh configs/config-1.json configs/config-2.json
```

Die Resultate werden dann in einem der `group_id` entsprechenden Verzeichnis innerhalb des `results/` Verzeichnisses abgelegt.

B.6. Weboberfläche

Mithilfe der bereitgestellten Weboberfläche können durchgeführte Experimente analysiert werden. Um dies zu ermöglichen sind drei Kernfunktionalitäten implementiert: Es wird eine Übersicht über alle JSON Dateien im entsprechenden Resultat-Ordner angezeigt, Metriken können mittels `math.js`[9] ausgewertet und es ist möglich Plots über Metriken mittels `matplotlib`[10] zu erstellen.

Die Weboberfläche selbst kann über den Befehl `python web/app.py` gestartet werden. Diese ist dann über den Browser mittels der URL `http://localhost:5000` erreichbar.

[9] http://mathjs.org/
[10] http://matplotlib.org/

Glossar

Ausbeute

Wert, welcher angibt wieviel Prozent der Datensätze einer Klasse als dieser zugehörig klassifiziert wurden.

Crossdomain

Vorgehen, bei welchem Daten aus mehreren Domänen verwendet werden.

Domäne

Texte, welche die gleichen semantischen Konzepte beschreiben und aus gleichen oder ähnlichen Quellen stammen werden zu einer Domäne zusammengefasst.

Epoche

Ein Durchlauf durch alle Trainingsdaten während des Trainings eines CNN wird Epoche genannt.

Filter

Grundbausteine eines CNN. Werden verwendet um die Aktivierungen einer Convolutional Schicht zu berechnen.

Neuron

Grundbausteine eines NN. Bilden eine mathematische Funktion ab.

Präzision

Wert, welcher angibt wieviel Prozent der beurteilten Datensätze korrekt klassifiziert wurden.

Schicht

Sammlung von mehreren Neuronen. Ein NN setzt sich im Normalfall aus mehreren solcher Schichten zusammen.

Abkürzungsverzeichnis

CNN

Convolutional Neural Network.

FD

Foreing Domain (dt. Fremddomäne).

GPU

Graphical Processing Unit.

NN

Neuronales Netzwerk.

TD

Target Domain (dt. Zieldomäne).

Tabellenverzeichnis

Abbildungsverzeichnis

Referenzen

[1] A. Balahur, R. Steinberger, M. Kabadjov, V. Zavarella, E. Van Der Goot, M. Halkia, B. Pouliquen und J. Belyaeva, „Sentiment analysis in the news“, in *Proceedings of the 7th International Conference on Language Resources and Evaluation*, European Language Resources Association, 2013, S. 2216–2220.

[2] J. Blitzer, M. Dredze, F. Pereira u. a., „Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification“, in *Proceedings of the 45th annual meeting on association for computational linguistics*, Association for Computational Linguistics, Bd. 7, 2007, S. 440–447.

[3] D. Bollegala, T. Mu und J. Y. Goulermas, „Cross-domain sentiment classification using sentiment sensitive embeddings“, *IEEE Transactions on Knowledge and Data Engineering*, Bd. 28, Nr. 2, S. 398–410, 2016.

[4] D. Bollegala, D. Weir und J. Carroll, „Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification“, in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, Association for Computational Linguistics, 2011, S. 132–141.

[5] J. Deriu, M. Gonzenbach, F. Uzdilli, A. Lucchi, V. De Luca und M. Jaggi, „Swisscheese at semeval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision“, in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, Association for Computational Linguistics, 2016, S. 1124–1128.

[6] X. Ding, B. Liu und P. S. Yu, „A holistic lexicon-based approach to opinion mining“, in *Proceedings of the 2008 international conference on web search and data mining*, Association for Computing Machinery, 2008, S. 231–240.

[7] A. Go, R. Bhayani und L. Huang, „Twitter sentiment classification using distant supervision“, *CS224N Project Report, Stanford*, 2009.

[8] A. B. Goldberg und X. Zhu, „Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization“, in *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, Association for Computational Linguistics, 2006, S. 45–52.

[9] M. Hu und B. Liu, „Mining and summarizing customer reviews“, in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, Association for Computing Machinery, 2004, S. 168–177.

[10] R. S. Jeffrey Pennington und C. D. Manning, „Glove: Global vectors for word representation“, in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, S. 1532–1543.

[11] R. Johnson und T. Zhang, „Semi-supervised convolutional neural networks for text categorization via region embedding“, in *Advances in neural information processing systems*, 2015, S. 919–927.

[12] Y. Kim, „Convolutional neural networks for sentence classification“, *ArXiv preprint arXiv:1408.5882*, 2014.

[13] S. Li und C. Zong, „Multi-domain sentiment classification“, in *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, Ser. HLT-Short '08, Association for Computational Linguistics, 2008, S. 257–260.

[14] J. Long, E. Shelhamer und T. Darrell, „Fully convolutional networks for semantic segmentation“, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, S. 3431–3440.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado und J. Dean, „Distributed representations of words and phrases and their compositionality“, in *Advances in neural information processing systems*, 2013, S. 3111–3119.

[16] P. Nakov, A. Ritter, S. Rosenthal, V. Stoyanov und F. Sebastiani, „SemEval-2016 task 4: Sentiment analysis in Twitter“, in *Proceedings of the 10th International Workshop on Semantic Evaluation*, Ser. SemEval '16, San Diego, California: Association for Computational Linguistics, Juni 2016.

[17] S. Narr, M. Hulfenhaus und S. Albayrak, „Language-independent twitter sentiment analysis“, *Knowledge Discovery and Machine Learning (KDML), LWA*, S. 12–14, 2012.

[18] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang und Z. Chen, „Cross-domain sentiment classification via spectral feature alignment“, in *Proceedings of the 19th International Conference on World Wide Web*, Association for Computing Machinery, 2010, S. 751–760.

[19] B. Pang und L. Lee, „Opinion mining and sentiment analysis“, *Foundations and trends in information retrieval*, Bd. 2, Nr. 1-2, S. 1–135, 2008.

[20] B. Pang, L. Lee und S. Vaithyanathan, „Thumbs up? sentiment classification using machine learning techniques“, in *The ACL-02 conference*, Morristown, NJ, USA: Association for Computational Linguistics, 2002, S. 79–86.

[21] A. Severyn und A. Moschitti, „Twitter sentiment analysis with deep convolutional neural networks“, in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Association for Computing Machinery, 2015, S. 959–962.

[22] V. Sindhwani und P. Melville, „Document-word co-regularization for semi-supervised sentiment analysis“, in *2008 Eighth IEEE International Conference on Data Mining*, IEEE, 2008, S. 1025–1030.

[23] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever und R. Salakhutdinov, „Dropout: A simple way to prevent neural networks from overfitting.“, *Journal of Machine Learning Research*, Bd. 15, Nr. 1, S. 1929–1958, 2014.

[24] C. Strapparava und R. Mihalcea, „Semeval-2007 task 14: Affective text“, in *Proceedings of the 4th International Workshop on Semantic Evaluations*, Association for Computational Linguistics, 2007, S. 70–74.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke und A. Rabinovich, „Going deeper with convolutions“, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, S. 1–9.

[26] O. Täckström und R. McDonald, „Discovering fine-grained sentiment with latent variable structured prediction models“, in *In Proceedings of the 33rd European Conference on Advances in Information Retrieval*, Springer, 2011, S. 368–374.

[27] D. Tang, F. Wei, B. Qin, T. Liu und M. Zhou, „Coooolll: A deep learning system for twitter sentiment classification“, in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 2014, S. 208–212.

[28] T. D. Team, „Theano: a Python framework for fast computation of mathematical expressions“, *ArXiv e-prints*, Bd. abs/1605.02688, Mai 2016. Adresse: `http://arxiv.org/abs/1605.02688`.

[29] P. D. Turney, „Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews“, in *Proceedings of the 40th annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2002, S. 417–424.

[30] J. Wiebe, T. Wilson und C. Cardie, „Annotating expressions of opinions and emotions in language“, *Language resources and evaluation*, Bd. 39, Nr. 2-3, S. 165–210, 2005.

[31] Q. Ye, Z. Zhang und R. Law, „Sentiment classification of online reviews to travel destinations by supervised machine learning approaches“, *Expert Systems with Applications*, Bd. 36, Nr. 3, S. 6527–6535, 2009.

[32] M. D. Zeiler, „Adadelta: An adaptive learning rate method“, *ArXiv preprint arXiv:1212.5701*, 2012.

[33] X. Zhang, J. Zhao und Y. LeCun, „Character-level convolutional networks for text classification“, in *Advances in Neural Information Processing Systems*, 2015, S. 649–657.

[34] S. Zhou, Q. Chen und X. Wang, „Active deep networks for semi-supervised sentiment classification“, in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, Association for Computational Linguistics, 2010, S. 1515–1523.