technische universität
dortmund

Bachelorarbeit

Implicit Emotions in Tweets

Rahel Luise Wilking
Januar 2019

Gutachter:
Prof. Dr. Katharina Morik
M.Sc. Lukas Pfahler

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (LS-8)
http://www-ai.cs.uni-dortmund.de

Inhaltsverzeichnis

Kapitel 1

Einleitung

Emotionen sind ein großer Teil des menschlichen Lebens und ein zentraler Aspekt unserer sozialen Strukturen. Menschen drücken ihre Emotionen auf verschiedenste Arten und Weisen aus, unter anderem auch in Texten. In den Zeiten des Internets ist auch dieses reich an emotionalen Informationen, besonders in sozialen Netzwerken und auf micro-blogging Seiten wie Twitter[1]. Die Analyse dieser Texte kann Vorteile für verschiedenste Anwendungsbereiche bringen, beispielsweise für die Vermarktung von Produkten, subjektive Suchmaschinen, Schreibassistenzsysteme und intelligente Text-zu-Sprache Systeme [21].

Die große Menge an verfügbaren Daten macht eine Analyse per Hand allerdings nahezu unmöglich. Außerdem herrscht auch zwischen menschlichen Interpretationen von Texten Uneinigkeit. Daher werden maschinelle Lernverfahren eingesetzt, um Emotionen in Texten zu erkennen und einzuordnen.

Die Implicit Emotions Shared Task (IEST) des Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis (WASSA) 2018[2] stellt die Aufgabe, Twitter-Daten in sechs Emotions-Kategorien zu klassifizieren: Anger, Disgust, Fear, Joy, Sadness und Surprise. Diese Arbeit beschäftigt sich damit verschiedene maschinelle Lernverfahren und Feature-Sets auf ihre Eignung für diese Aufgabe zu untersuchen.

1.1 Motivation und Hintergrund

Im Bereich der Emotion Classification fanden neben der IEST schon weitere Shared Tasks statt, darunter die Semeval-2007 Task 14 zu Emotionen in Nachrichtenüberschriften, WASSA-2017 EmoInt zu Emotion Intensities und SemEval-2018 Task 1, welche sowohl Unteraufgaben zu Emotion Intensity, als auch Emotion Classification hat [33, 22, 20]. Die IEST legt den Fokus auf die Erkennung von Emotionen aus der impliziten Beschreibung, indem sie explizite Emotionsäußerungen entfernt und Daten sammelt, in denen mit hoher

[1] http://www.twitter.com
[2] http://www.implicitemotions.wassa2018.com

Wahrscheinlichkeit eine Beschreibung für den Auslöser der gesuchten Emotion enthalten ist [14].

Emotion Classification wurde bereits für vielen verschiedenen Medien durchgeführt [21], insbesondere jedoch auch auf Twitter-Daten [36, 17, 8, 25]. Dabei haben Twitter-Daten den Vorteil, in großer Menge öffentlich zugänglich zu sein[3] und eine Vielzahl an Themen zu behandeln. Andererseits haben Tweets einige Besonderheiten, wie informelle Sprache und besondere Zeichenkombinationen, welche sie von klassischen Texten unterscheiden. Dies sorgt möglicherweise dafür, dass Ansätze für klassische Textdaten nicht auf Twitter-Daten übertragbar sind [17].

Eine weitere Schwierigkeit besteht darin, dass für die Klassifikationsaufgaben Datensätze zur Verfügung stehen müssen, für welche die Labels bereits bekannt sind. Die Labels für große Mengen an Daten per Hand zu bestimmen ist ein sehr aufwendiger Prozess. Ansätze, dieses Problem zu lösen, indem Labels automatisch über Hashtags oder Emoticons festgelegt werden, nennen sich *distant supervision* [21].

Eine weitere Schwierigkeit bei der Emotion Classification ist, dass ausgedrückte Emotionen nicht einfach die Summe der emotionalen Assoziationen der vorkommenden Wörter sind. Zusätzlich können sie ausschließlich implizit ausgedrückt werden. Außerdem können Wörter in verschiedenen Kontexten verschiedene Emotionen vermitteln. Als Medium hat Text im Vergleich zu Sprache den Nachteil, dass zusätzliche Hinweise auf Emotionen wie Tonhöhe oder Betonungen in Textform nicht direkt vermittelt werden können. Dies macht es auch für Menschen schwierig Emotionen aus Texten eindeutig zu erkennen [21].

Für die Baseline der IEST wurde ein Modell mit logistischer Regression trainiert. Diese Methode wurde daher für diese Arbeit ausgewählt. Support Vector Machines (SVM) wurden bereits erfolgreich für die Aufgabe der Emotion Classification verwendet [17, 8, 25]. Character-level Convolutional Neural Networks (Character-CNN) wurden von Zhang et al. mit guten Ergebnissen für Text Classification verwendet und von Weihan Pang bereits auf Twitter-Daten für eine Sentiment Analysis Aufgabe angewendet [37, 29]. Diese beiden Methoden werden in den Experimenten mit der logistischen Regression verglichen. Weiterhin werden Experimente in Ensembles zusammengefasst.

Neben Wort-basierten Features wie tfidf, unigrams und bigrams werden in den Experimenten Features aus zwei verschiedenen Emotions-Lexika gewonnen, sowie aus zwei Language Models. Weiterhin wird ein auf Twitter-Daten trainiertes Word-Embedding für Features verwendet.

1.2 Aufbau der Arbeit

Die Arbeit besteht aus acht Kapiteln. Der Rest dieser Arbeit ist wie folgt aufgebaut. Das zweite Kapitel ordnet die Aufgabenstellung der Arbeit in ihren Forschungsbereich ein. Das

[3] https://www.developer.twitter.com/

darauffolgende Kapitel behandelt die Aufgabenstellung der Shared Task selbst. Im vierten Kapitel werden die hier verwendeten Methoden des maschinellen Lernens eingeführt. Das anschließende Kapitel behandelt die Feature-Sets und ihre Konstruktion. Danach wird in Kapitel sechs der Aufbau und die Durchführung der Experimente erklärt. Im nächsten Kapitel werden diese evaluiert. Abschließend werden die Ergebnisse der Arbeit zusammengefasst und ein Ausblick auf weitere Forschung gegeben.

Kapitel 2

Einordnung des Themas

Die für die Implicit Emotions Shared Task gestellte Aufgabe kann dem Forschungsbereich der Emotion Classification zugeordnet werden. Dieser selbst steht im Bezug zur Sentiment Analysis. Weiterhin ist es eine Klassifikationsaufgabe, welche es in den Bereich der Text Classification einordnet. Diese Bereiche sollen in diesem Kapitel genauer betrachtet werden.

2.1 Text Classification

Text Classification ist die Aufgabe, Dokumenten eine von m vorgegebenen Klassen zuzuordnen [37]. Eine Trainingsmenge ist eine Menge $S = \{(x_i, y_i)|i = 1, \ldots, p\}$ von Paaren (x_i, y_i), wobei x_i einen Text repräsentiert und y_i das bekannte, dem Text zugeornete Label ist. Eine Klassifikationsfunktion bildet die Textrepräsentationen auf die Menge der vorgegebenen Klassen ab. Sei g die tatsächliche Zuordnungsfunktion über die Textdaten. Das Ziel ist es, eine Klassifikationsfunktion f zu bestimmen, die g möglichst gut approximiert, also so, dass möglichst $f(x) = g(x)$ gilt [32].

Ist die Anzahl Klassen $m = 2$, so spricht man von einem *binären* Klassifikationsproblem, falls $m > 2$ ist, von einem *Multiklassen*-Problem.

Die Text Classification ist ein grundlegendes Problem im Natural Language Processing (NLP) [10] und hat viele Anwendungen, wie etwa die Klassifikation von Nachrichtenartikeln und die Extraktion interessanter Informationen aus dem Internet [9]. Dabei ist ein Großteil der Forschung daran interessiert Textdokumente entsprechend ihrer inhaltlichen Themen zu klassifizieren [28].

2.2 Sentiment Analysis

Sentiment Analysis ist ein breites Feld, welches sich mit der Erkennung von Meinungen, Sentimenten und Subjektivität in Text beschäftigt. Andere Begriffe für diesen Forschungsbereich sind *Opinion Mining* und *Subjectivity Analysis*. Dabei konzentriert sich Subjectivi-

ty Analysis vorherrschend auf die Erkennung von subjektiven Texten bzw. Textstellen im Kontrast zu objektiven Texten. Opinion Mining wird oft in einem Kontext verwendet, in dem Meinungen zu verschiedenen Aspekten eines Gegenstands oder einer Entität ermittelt werden sollen. Sentiment Analysis wird hingegen vorherrschend als Bezeichnung für die Bestimmung der Valenz oder Polarität von Texten, auch *sentiment*, verwendet [27]. Dieser Teil der Sentiment Analysis kann als Text Classification Problem angesehen werden. Dabei repräsentieren die Klassen positive, negative, und manchmal neutrale Polarität [21].

Sentiment Analysis hat viele Anwendungsgebiete, beispielsweise die Bestimmung der Polarität von Filmkritiken [28], oder auch als Subkomponente in Business Intelligence Systemen, um die Meinungen von Kunden zu analysieren [27].

2.3 Emotion Classification

Emotion Classification kann als eine Unterart der Sentiment Analysis angesehen werden. Sie betrachtet dabei die Klassifizierung von Text in Kategorien, die Emotionen entsprechen. Die vordefinierten Klassen sind dabei Basis-Emotionen. Allerdings gibt es verschiedene Theorien über Basis-Emotionen und in der Forschung wird daher nicht eine einheitliche Menge an Emotionen als Klassen verwendet [21].

Die automatische Erkennung von Emotionen in Text hat viele Anwendungsgebiete, beispielsweise emotional sensitive Dialogsysteme und Assistenzsysteme zum Schreiben gezielt emotionaler Texte, sowie verbesserte Text-zu-Sprache Systeme [24]. Emotion Classification wurde auf diversen Medien angewendet, wie Nachrichtenüberschriften [34] und Twitter-Daten [25].

Kapitel 3

Vorstellung der Aufgabe

Die Aufgabenstellung, auf der diese Arbeit aufbaut, entspricht der Implicit Emotions Shared Task (IEST) des Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis (WASSA) 2018[1]. Es handelt sich dabei um eine Klassifikationsaufgabe, in der Twitter[2]-Daten einer von sechs vorgegebenen Emotionen zugeordnet werden sollen. Diese Emotionen entsprechen den Basis-Emotionen nach Ekman [5]: Anger, Disgust, Fear, Joy, Sadness und Surprise. Dabei wurde jeweils ein mit einer dieser Emotionen korrespondierendes Emotionswort aus einem Tweet entfernt und durch einen Platzhalter ersetzt. Durch diesen Vorgang wurden auch die Labels gewonnen. Es wurden dabei natürlich nur Tweets ausgewählt, die eines dieser Emotionsworte enthalten, aber auch nur solche, in denen auf dieses unmittelbar ein Wort aus `(that|because|when)` folgt. Dies diente dem Zweck, automatisch Tweets zu wählen, die mit einer hohen Wahrscheinlichkeit den Grund für das Ausdrücken dieser Emotion enthalten. Durch diese Maßnahmen, also das Entfernen eines expliziten Emotionsausdrucks und der Inklusion eines Emotionsursprungs, sollte der Fokus der Shared Task, die Vorhersage aus den implizit im Text enthaltenen Informationen, notwendig gemacht werden [14]. Für eine Liste der Emotionsworte, sowie genaue Details zum Prozess der Datengewinnung siehe Klinger et al. [14].

Die Datensätze, sowie ein offizielles Evaluationsskript, wurden von den Veranstaltern der Shared Task bereitgestellt und sind auf der Veranstaltungs-Website[3] abrufbar.

3.1 Daten

Für die Shared Task wurden insgesamt drei Datensätze konstruiert, jeweils ein Train-, Trial- und ein Test-Datensatz. Von diesen standen den Teilnehmern während der Shared Task nur die ersten beiden zur Verfügung und letzterer wurde für die abschließende Be-

[1] http://www.implicitemotions.wassa2018.com
[2] http://www.twitter.com
[3] http://www.implicitemotions.wassa2018.com/data
bzw. http://www.implicitemotions.wassa2018.com/evaluation

Emotion	Train	Trial	Test
Anger	25562	1600	4794
Disgust	25558	1597	4794
Fear	25575	1598	4791
Joy	27958	1736	5246
Sadness	23165	1460	4340
Surprise	25565	1600	4792
Sum	153383	9591	28757

Tabelle 3.1: Übersicht über die Klassenhäufigkeiten, übernommen aus Klinger et al. [14]

wertung der Einreichungen verwendet. Die Verteilung der Daten auf die Klassen ist über alle drei Datensätze hinweg relativ gleichmäßig. Die absoluten Häufigkeiten finden sich in Tabelle 3.1. Für die im Rahmen dieser Bachelorarbeit durchgeführten Experimente wurde der Train-Datensatz zum Trainieren der Modelle, der Trial-Datensatz zur Parameteroptimierung und der Test-Datensatz zur abschließenden Evaluation der vielversprechenden Modelle verwendet.

Ein typisches Datum aus dem Train-Datensatz kann in Beispiel 1 betrachtet werden. Eine Besonderheit in den Daten betrifft z.B. das Vorkommen von Daten mit `un[#TRIGGERWORD#]`, siehe Beispiel 2, bei denen direkt vor dem entfernten Emotionswort kein Leerzeichen stand, also das entfernte Wort nur ein Wortbestandteil war. Ähnliches passiert auch mit `#[#TRIGGERWORD#]`, siehe Beispiel 3, und wenigen weiteren Präfixen.

It makes me so [#TRIGGERWORD#] when everything works out in the end

Beispiel 1: Beispiel-Tweet mit Label ”joy”

I'll always be un[#TRIGGERWORD#] because happiness doesn't know me

Beispiel 2: Beispiel-Tweet mit Label ”joy”

You can't be #[#TRIGGERWORD#] when you are holding a #smoothie!

Beispiel 3: Beispiel-Tweet mit Label ”sad”

3.2 Emotionstheorien

Die IEST schreibt die Verwendung von sechs Basis-Emotionen vor. Diese basieren auf den Theorien von Ekman, welche ihren Ursprung in der Identifizierung verschiedener Gesichtsausdrücke hatten [5]. Diese Emotionskategorien wurden in der Forschung mehrfach verwendet [16, 25, 34]. Neben den Basis-Emotionen nach Ekman, gibt es allerdings auch noch andere Listen an Basis-Emotionen und weitere Emotionstheorien die, statt Emotionen in eine diskrete Anzahl Basis-Emotionen aufzuteilen, sie anhand von Ausprägungen verschiedener Dimensionen identifizieren. Dabei sind die am häufigsten auftretenden Dimensionen *arousal*, *valence* und *control*, welche auch unter anderen Namen vorkommen [30]. Es gibt also nicht eine allgemein akzeptierte Theorie über Emotionen. Allerdings hat die Theorie von Basis-Emotionen den Vorteil für Klassifikationsaufgaben, eine kleine diskrete Menge an Kategorien vorzuschreiben, welches die Erstellung annotierter Datensätze vereinfacht [21].

Kapitel 4

Methoden

Der Bereich des maschinellen Lernens bietet viele verschiedenen Methoden um die vorgegebene Aufgabenstellung zu lösen. Im Rahmen dieser Arbeit wurden Experimente mit drei verschiedenen maschinellen Lernverfahren durchgeführt, deren theoretische Grundlagen in diesem Kapitel erläutert werden. Dafür sei $S = \{(x_i, y_i) | i = 1, \ldots, p\}$ mit $x_i \in \mathbb{R}^n$ und $y_i \in \{-1, +1\}$ die Menge an gelabelten Trainingsbeispielen, wobei p die Anzahl an Trainingsbeispielen ist. Das x_i bezeichnet die Darstellung eines Tweets in einem $(n-1)$-dimensionalen Feature-Raum, mit einer zusätzlichen letzten Komponente. Diese wird für alle Daten auf 1 gesetzt, um Notationen zu vereinfachen, indem Konstanten-Parameter mit in ein Vektorprodukt inkorporiert werden können.

4.1 Logistische Regression

Die logistische Regression ist ein binäres Klassifizierungsverfahren, welches auf bereits gelabelten Daten trainiert wird. Dabei wird die bedingte Wahrscheinlichkeitsverteilung einer Klassenzugehörigkeit, abhängig von den Features eines Datums und einem Parametervektor $\alpha \in \mathbb{R}^n$, bestimmt als [4]:

$$P(y = +1 | x; \alpha) = \frac{1}{1 + \mathbf{e}^{-\alpha^\top x}}. \tag{4.1}$$

Dabei ist $\mathbf{e}$ die eulersche Zahl. Die notationelle Vereinfachung, die vorgenommen wurde, ermöglicht es, den konstanten Term im Exponenten mit in α zu inkorporieren. Die Wahrscheinlichkeit für die Klasse -1 ergibt sich über die Gegenwahrscheinlichkeit als

$$P(y = -1 | x; \alpha) = 1 - \frac{1}{1 + \mathbf{e}^{-\alpha^\top x}} = \frac{\mathbf{e}^{-\alpha^\top x}}{1 + \mathbf{e}^{-\alpha^\top x}} = \frac{1}{1 + \mathbf{e}^{\alpha^\top x}}. \tag{4.2}$$

Die beiden Wahrscheinlichkeiten lassen sich also in folgender Form schreiben:

$$P(y | x; \alpha) = \frac{1}{1 + \mathbf{e}^{-y\alpha^\top x}}. \tag{4.3}$$

Um die Parameter α der Wahrscheinlichkeitsverteilung zu bestimmen, wird eine Maximum-Likelihood-Schätzung über die Trainingsdaten vorgenommen. Dafür wird der folgende Term maximiert [4]:

$$\prod_{i=1}^{p} P(y = y_i | x = x_i; \alpha). \tag{4.4}$$

Dies ist äquivalent dazu, den Logarithmus dieses Ausdrucks zu maximieren, also

$$\max_{\alpha} \sum_{i=1}^{p} \log P(y = y_i | x = x_i; \alpha), \tag{4.5}$$

wobei log den natürlichen Logarithmus bezeichnet.

Für die Experimente in dieser Arbeit wird außerdem eine L2-Regularisierung vorgenommen. Dies äußert sich durch einen Regularisierungsterm in der Zielfunktion, wodurch sich das Problem wie folgt verändert [26]:

$$\max_{\alpha} \sum_{i=1}^{p} \log P(y = y_i | x = x_i; \alpha) - \lambda \alpha^\top \alpha. \tag{4.6}$$

Dabei kontrolliert $\lambda \geq 0$ die Stärke der Regularisierung. Dies lässt sich nun wie folgt umformen:

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{p} \log P(y = y_i | x = x_i; \alpha) - \lambda \alpha^\top \alpha \\ =&\max_{\alpha} \sum_{i=1}^{p} \log \frac{1}{1 + \mathbf{e}^{-y_i \alpha^\top x_i}} - \lambda \alpha^\top \alpha \\ =&\max_{\alpha} \sum_{i=1}^{p} -\log(1 + \mathbf{e}^{-y_i \alpha^\top x_i}) - \lambda \alpha^\top \alpha. \end{aligned} \tag{4.7}$$

Dies ist äquivalent zu

$$\min_{\alpha} \lambda \alpha^\top \alpha + \sum_{i=1}^{p} \log(1 + \mathbf{e}^{-y_i \alpha^\top x_i}). \tag{4.8}$$

Wird nun noch $\lambda = \frac{1}{2C}$ und $\alpha = w$ gesetzt, sowie die Funktion mit der konstanten C durchmultipliziert, ergibt sich die äquivalente Formulierung von LIBLINEAR, welche für die Experimente verwendet wird [6]:

$$\min_{w} \frac{1}{2} w^\top w + C \sum_{i=1}^{p} \log(1 + \mathbf{e}^{-y_i w^\top x_i}). \tag{4.9}$$

Ist die Wahrscheinlichkeitsverteilung bestimmt, so erfolgt die Klassifikation wie folgt:

$$f(x) = \begin{cases} +1 & \text{falls } P(y = +1 | x; \alpha) > 0.5 \\ -1 & \text{sonst} \end{cases}. \tag{4.10}$$

Da die logistische Regression eigentlich nur ein binärer Klassifizierer ist, wird für die Lösung eines Multiklassen-Problems mit LIBLINEAR die *one-vs-the-rest*-Strategie angewendet [6],

bei welcher für jede Klasse ein binäres Problem gelöst wird, in der nur die Beispiele dieser Klasse der Klasse $+1$ angehören und alle anderen der Klasse -1. Diese binären Probleme werden daraufhin zu einer einzigen Vorhersage kombiniert, indem das Maximum der Vorhersagen gewählt wird [1].

Die L2-regularisierte logistische Regression wurde als Methode gewählt um Vergleichbarkeit mit der Baseline der Shared Task zu schaffen, die ebenfalls auf dieser Methode basiert [14].

4.2 Lineare SVM

Die lineare Support Vector Machine (SVM) ist ein binäres Klassifizierungsverfahren, welches von Vapnik entwickelt wurde. Sie benötigt gelabelte Trainingsdaten und findet für die Trainingsdaten eine Hyperebene im Feature-Raum, welche die zwei Klassen mit maximalem Abstand separiert [9]. Der maximale Abstand ist von Interesse, um ein Modell zu trainieren, welches möglichst gut für ungesehene Daten generalisiert. Eine Hyperebene wird definiert durch eine Gleichung $w^\top x + b = 0$. Durch die oben vorgenommene notationelle Vereinfachung kann der Bias-Term b mit in das w inkorporiert werden und es ergibt sich eine Hyperebene als $w^\top x = 0$ mit $w \in \mathbb{R}^n$. Eine separierende Hyperebene muss nun erfüllen, dass für alle Trainingsbeispiele gilt:

$$y_i w^\top x_i \geq 1. \tag{4.11}$$

Die Beispiele, die diese Ungleichung mit Gleichheit erfüllen, werden Support Vectors genannt. Sie haben von der Hyperebene den Abstand $\frac{1}{\|w\|}$, wobei $\|\cdot\|$ die euklidische Norm bezeichnet [1]. Damit ist der Abstand, mit dem die beiden Klassen separiert werden $\frac{2}{\|w\|}$, also größer, wenn $\| w \|$ kleiner wird. Es ergibt sich das Optimierungsproblem [9]:

$$\begin{aligned} &\min_w \quad \| w \| \\ &\text{s.t. } y_i w^\top x_i \geq 1 \ \forall i = 1, \ldots, p. \end{aligned} \tag{4.12}$$

Für die Optimierung ist oft die bezüglich der optimalen Lösung äqivalente Formulierung dieses Problems, mit quadrierter und mit $\frac{1}{2}$ skalierter Zielfunktion einfacher. Mit $\| w \|^2 = w^\top w$ ergibt sich dann:

$$\begin{aligned} &\min_w \quad \frac{1}{2} w^\top w \\ &\text{s.t. } y_i w^\top x_i \geq 1 \ \forall i = 1, \ldots, p. \end{aligned} \tag{4.13}$$

Diese Variante der linearen SVM findet nur eine separierende Hyperebene, falls die Daten linear separierbar sind. Für Daten bei denen das nicht der Fall ist, kann diese Variante nicht angewendet werden. Die Lösung für dieses Problem ist es, Slack-Variablen in den

Nebenbedingungen einzuführen, die als verschlechternder Zusatz in die Zielfunktion eingehen. Bei der L2-loss Version der SVM, die für die Experimente verwendet wird, gehen die Slack-Variablen ξ_i quadratisch ein. Dies führt zur folgenden Formulierung:

$$\begin{aligned} \min_w \ & \frac{1}{2} w^\top w + C \sum_{i=1}^{p} \xi_i^2 \\ \text{s.t. } & y_i w^\top x_i \geq 1 - \xi_i \ \forall i = 1, \ldots, p \\ & \xi_i \geq 0 \ \forall i = 1, \ldots, p. \end{aligned} \tag{4.14}$$

Der zusätzliche Parameter C bestimmt, wie stark Fehler in den Nebenbedingungen in die Zielfunktion eingehen. Eine Umformulierung der Nebenbedingung führt zu

$$\xi_i \geq 1 - y_i w^\top x_i \ \forall i = 1, \ldots, p, \tag{4.15}$$

insgesamt gilt also:

$$\xi_i \geq \max(0, 1 - y_i w^\top x_i) \ \forall i = 1, \ldots, p. \tag{4.16}$$

Da wir minimieren, ersetzen wir also die ξ_i in der Zielfunktion durch $\max(0, 1 - y_i w^\top x_i)$ und erhalten die Formulierung des Optimierungsproblems für die L2-regularisierte L2-loss lineare SVM von LIBLINEAR, die für die Experimente verwendet wird [6]:

$$\min_w \ \frac{1}{2} w^\top w + C \sum_{i=1}^{p} (\max(0, 1 - y_i w^\top x_i))^2. \tag{4.17}$$

Die aus der Optimierung resultierende Hyperebene kann dann zur Klassifizierung neuer Beispiele verwendet werden über [9]:

$$f(x) = \begin{cases} +1 & \text{falls } w^\top x > 0 \\ -1 & \text{sonst} \end{cases}. \tag{4.18}$$

Da auch die lineare SVM eigentlich nur ein binärer Klassifizierer ist, wird für die Lösung eines Multiklassen-Problems mit LIBLINEAR die *one-vs-the-rest*-Strategie angewendet [6], wie im Falle der logistischen Regression.

4.3 Character-CNN

Convolutional Neural Networks (CNN) sind eine besondere Form eines Neural Networks. Ein Character-CNN arbeitet dabei auf Eingabedaten, die den einzelnen Zeichen aus dem Eingabetext entsprechen. Für die Repräsentation der Zeichen muss zunächst ein Alphabet gewählt werden. Für die Experimente in dieser Arbeit wurde dabei ein Alphabet konstruiert, indem die 128 in den Trainingsdaten am häufigsten vorkommenden Zeichen, die nicht das Leerzeichen oder besondere Verbindungszeichen und Modifizierungszeichen für Emoji sind, gewählt wurden. Dabei wurde bei Buchstaben kein Unterschied zwischen Groß- und

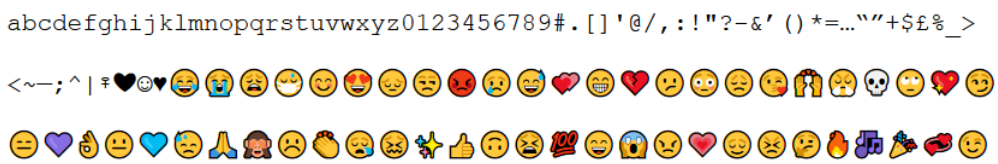

Abbildung 4.1: Das für die Character-CNN-Experimente verwendete Alphabet ohne den Newline-Character

Kleinbuchstaben gemacht, da dies nach Zhang et al. in der Regel zu besseren Ergebnissen führt [37]. Die Zeichen des resultierenden Alphabets, ohne den Newline-Character, finden sich in Abbildung 4.1.

Ist das Alphabet gewählt, kann ein Zeichen c nun in eine Vektordarstellung $x^{(c)} \in \{0,1\}^s$ überführt werden, wobei s die Größe des Alphabets ist, indem $x_i^{(c)} = 1$ gesetzt wird, falls c das i-te Zeichen im Alphabet ist und $x_i^{(c)} = 0$ für alle anderen i. Dies sorgt auch dafür, dass Zeichen, die nicht im Alphabet enthalten sind, auf den Nullvektor abgebildet werden. Weiterhin muss eine maximale Länge l_0 festgelegt werden. In den Versuchen wurde diese als 256 gewählt, da diese Länge den Großteil der Eingabedaten vollständig einschließt. Damit kann ein Eingabetweet $t = c_1 c_2 c_3 \dots c_l$, wobei die c_i die Zeichen des Tweets sind, in eine Matrix $M_t = (x^{(c_1)}\ x^{(c_2)}\ x^{(c_3)}\ \dots\ x^{(c_{l_0})}) \in \{0,1\}^{s \times l_0}$ umgewandelt werden. Dabei wird der Tweet t nach l_0 Zeichen abgeschnitten, falls $l > l_0$ gilt, oder, falls $l < l_0$, M_t mit Nullvektoren aufgefüllt.

Ein Netzwerk kann nun aus 1-D-Convolution-Layern und 1-D-Max-Pooling-Layern aufgebaut werden. Beschreibe $g : \{1, \dots, m\} \to \mathbb{R}$ die Funktion der Eingabedaten und $f : \{1, \dots, k\} \to \mathbb{R}$ die Kernel-Funktion, dann ist die 1-D-Convolution $h : \{1, \dots, \lfloor (m - k + 1)/d \rfloor\} \to \mathbb{R}$ mit Stride d wie folgt definiert:

$$h(y) = \sum_{x=1}^{k} f(x) \cdot g(y \cdot d - x + c), \tag{4.19}$$

wobei $c = k - d + 1$ eine Offset-Konstante ist [37]. Die 1-D-Max-Pooling Funktion $h : \{1, \dots, \lfloor (m - k + 1)/d \rfloor\} \to \mathbb{R}$ ist definiert als:

$$h(y) = \max_{x=1}^{k} g(y \cdot d - x + c), \tag{4.20}$$

wobei $c = k - d + 1$ wieder eine Offset-Konstante beschreibt [37].

Die Netzarchitektur, welche für die Experimente dieser Arbeit verwendet wurde, basiert auf der Architektur von Zhang et al., die bereits von Weihan Pang für Experimente auf Twitter-Daten für eine ähnliche Anwendung verwendet wurde [37, 29]. Diese wurde auf mehrere Weisen modifiziert, um einer deutlich kleineren Trainingsmenge als der aus den Experimenten von Zhang et al. gerecht zu werden. Sie besteht aus fünf 1-D-Convolution-Layern und drei Fully-Connected-Layern. Nach den ersten beiden Convolutions und nach der letzten Convolution folgt jeweils ein 1-D-Max-Pooling-Layer. Die Strides

Layer	Filter-Anzahl	Kernel-Größe	Pool-Größe
1	64|64|64	7|5|3	3
2	128	3	3
3	128	3	N/A
4	128	3	N/A
5	128	3	3

Tabelle 4.1: Übersicht über die Convolution-Layer

der Convolution-Layer sind meistens 1, im vierten Convolution-Layer beträgt sie 2 und das zweite Layer nutzt eine Dilation von 3. Das erste Layer ist dreigeteilt, mit jeweils 64 Filtern der Größen 3, 5 und 7. Außerdem nutzt dieses Layer Padding, um die Ausgabegröße dieser drei Teile gleich zu halten. Die restlichen Convolution-Layer haben jeweils 128 Filter mit Kernel-Größe 3. Eine Übersicht über die Konfiguration der Convolution-Layer findet sich in Tabelle 4.1. Die Pooling-Layer sind alle nicht-überlappend. Die Ausgabelänge nach dem letzten Convolution-Layer nach dem Pooling ist 3. Daraus ergibt sich die Eingabegröße des ersten Fully-Connected-Layer als $128 \cdot 3 = 384$. Eine Übersicht über die Fully-Connected-Layer findet sich in Tabelle 4.2. Die Aktivierungsfunktion für alle Layer, bis auf das Output-Layer, ist die Funktion $h(x) = max(0, x)$. Für das Output-Layer wird die Softmax-Aktivierung verwendet. Die Gewichte werden initialisiert über *glorot_uniform*, also einer Gleichverteilung in einem Rahmen, welcher von der Eingabe- und Ausgabegröße des Layers abhängt.

Das Modell wurde im Vergleich zu dem von Zhang et al. verkleinert durch das Weglassen eines Convolution-Layers und der Verkleinerung der Kernel-Größe im ersten und zweiten Layer, sowie der Filter-Anzahlen. Außerdem wurde die Größe der Fully-Connected-Layer deutlich reduziert. Dies verringert die Anzahl zu trainierender Parameter. Weitere Änderungen umfassen das Auslassen der Dropout-Module zwischen den Fully-Connected-Layern und die Änderung der Parameter-Initialisierung.

Das Training eines Netzwerks ist beispielsweise mit Stochastic Gradient Descent und Backpropagation möglich. Dabei werden die Werte der Convolution-Filter und die Gewichte der Fully-Connected-Layer optimiert.

Im Gegensatz zu den vorgestellten linearen Modellen werden die relevanten Features von einem Character-CNN also automatisch gelernt und nicht aufwändig konstruiert und ausgewählt.

4.4 Ensemble

Die Ergebnisse eines Klassifikators sind meist fehlerbehaftet, sie erreichen also nicht vollständig korrekt klassifizierte Ergebnisse auf Evaluationsdaten. Wenn verschiedene Klassifi-

Layer	Ausgabe-Größe
6	300
7	300
8	6

Tabelle 4.2: Übersicht über die Fully-Connected-Layer

katoren trainiert werden, kann es jedoch vorkommen, dass sie bei unterschiedlichen Daten Fehler machen. Dies ermöglicht, dass eine Kombination verschiedener Klassifikatoren nach bestimmten Regeln potentiell bessere Ergebnisse erzielt als die einzelnen Klassifikatoren. Eine solche Gruppe an Klassifikatoren nennt man auch ein Ensemble. Dabei wird versucht ein möglichst diverses Ensemble zu konstruieren. Eine Möglichkeit dies zu erreichen ist die Klassifikatoren auf verschiedenen Feature-Sets zu trainieren, eine andere verschiedene maschinelle Lernverfahren zu verwenden [1].

Für die Experimente in dieser Arbeit findet dabei die Majority-Voting-Methode Anwendung. Seien die Klassen $\{1, \ldots, r\}$ und $\{f_1, \ldots, f_k\}$ die Vorhersagefunktionen der Klassifikatoren des Ensembles. Sei $\Delta_{ij}(x) = 1$ falls $f_j(x) = i$ und $\Delta_{ij}(x) = 0$ sonst. Dann ist die Vorhersagefunktion des Ensembles:

$$f(x) = \underset{i=1}{\overset{r}{\operatorname{argmax}}} \left(\sum_{j=1}^{k} \Delta_{ij}(x) \right). \tag{4.21}$$

Es wird also diejenige Klasse gewählt, für die sich die meisten Klassifikatoren entschieden haben [13].

Die Verwendung von Ensembles für Sentiment Analysis konnte merkbare Verbesserungen erzielen [15], und die besten Teams bei der IEST nutzen alle Ensembles [14]. Dies lässt vermuten, dass die Verwendung von Ensembles einen positiven Effekt auf die Ergebnisse der Experimente haben kann.

Kapitel 5

Features

Die beiden Methoden logistische Regression und Support Vector Machine (SVM) benötigen Eingabedaten in einer Vektorrepräsentation. Dies wird realisiert mittels einer Bag-of-Features-Darstellung. Alle benutzten Features bilden Teile der Eingabetexte auf numerische Werte ab. Es werden diverse Features für die Experimente verwendet, deren Konstruktion in diesem Kapitel vorgestellt wird.

5.1 Bag-of-Words

In einem Bag-of-Words-Ansatz wird ein Text in seine Wörter aufgetrennt und diese, entsprechend einem Vokabular, gezählt. Die Anzahl wird an die dem im Vokabular enthaltenen Wort entsprechende Stelle des Feature-Vektors gesetzt. Wörter, die nicht im Vokabular enthalten sind, fallen weg. Das Vokabular kann dabei vorgegeben sein oder aus den Daten konstruiert werden. Bei diesem Verfahren gehen Informationen über die Wortreihenfolge verloren.

5.2 Tfidf

Die Abkürzung tfidf steht für term-frequency inverse-document-frequency. Grundlage für die tfidf-Features ist ein Bag-of-Words-Ansatz. Für diesen wurden zunächst die Tokens aus dem Text extrahiert, indem der TweetTokenizer vom Natural Language Toolkit (NLTK)[1] verwendet wurde. Dabei wurde sichergestellt, dass die Platzhalter `[#TRIGGERWORD#]` und `[NEWLINE]`, die in den Daten vorkommen, jeweils als ein einzelnes Token beibehalten werden. Da die Daten aus Tweets bestehen, wurde der TweetTokenizer gewählt, um Besonderheiten von Tweets, wie Emoticons aus Satzzeichen, zu berücksichtigen. Von den resultierenden Tokens des Trainingsdatensatzes wurden diejenigen beibehalten, die in den Daten mindestens zweimal auftreten. Dadurch wurden 103,276 Tokens zu 40,873 Tokens redu-

[1] https://www.nltk.org

ziert, also wurde die Feature-Anzahl um ca. 60% verringert. Durch diesen Prozess auf den Trainingsdaten wurde ein Vokabular erstellt. Die Tokens des Trial- und Test-Datensatzes wurden auf die gleiche Weise extrahiert, aber nur die im konstruierten Vokabular enthaltenen beibehalten. Für jedes verbleibende Token wurde die term-frequency (tf) für jedes Datum bestimmt, also die Häufigkeit des Auftretens in einem Tweet. Weiterhin wurde für jedes Token aus dem Vokabular seine inverse-document-frequency (idf) über die Trainingsdaten bestimmt. Sei p die Anzahl an Dokumenten im Trainingsdatensatz und $\{w_1, \ldots, w_n\}$ das Vokabular, dann ist die document-frequency $df(w_i)$ die Anzahl der Dokumente, die das Token w_i enthalten. Die idf wird wie folgt berechnet [9]:

$$idf(w_i) = \log\left(\frac{p}{df(w_i)}\right). \tag{5.1}$$

Die endgültigen tfidf-Features berechnen sich dann als

$$x_i^{(j)} = tf_j(w_i) \cdot idf(w_i), \tag{5.2}$$

wobei $tf_j(w_i)$ die term-frequency von Feature w_i in Datum j ist und $x_i^{(j)}$ die i-te Komponente des Feature-Vektors für Datum j bezeichnet. Das so kreierte Feature-Set wird im Weiteren mit *tfidf* bezeichnet.

Die tfidf-Features wurden ausgewählt, da sie ein typischer Ansatz für Text Classification Aufgaben sind [32].

5.3 n-grams

Wort-basierte n-grams sind eine Folge von n aufeinander folgenden Wörtern. Für $n > 1$ erfassen sie syntaktische Strukturen im Text durch ihre Beibehaltung der Wortreihenfolge. Die für die Konstruktion von n-grams verwendeten Tokens wurden wie für die tfidf-Features mit dem TweetTokenizer aus den Trainingsdaten extrahiert. Daraufhin wurden jeweils n in den Daten aufeinanderfolgend auftretende Tokens als ein n-gram zusammengefasst, auch alle überlappenden Vorkommnisse. Von den resultierenden n-grams wurden lediglich solche behalten, die in den Trainingsdaten mindestens zweimal vorkommen. Dies bildet das Vokabular. Für die Trial- und Test-Datensätze wurden die n-grams auf die gleiche Weise konstruiert, allerdings nur die im konstruierten Vokabular vorkommenden behalten. Daraufhin wurde für alle verbleibenden n-grams die term-frequency bestimmt und als Feature verwendet. Sei $\{g_1, \ldots, g_n\}$ das Vokabular, dann entspricht die i-te Komponente des j-ten Datums, $x_i^{(j)}$, der Anzahl des Auftretens von Feature g_i in Datum j.

Für die Experimente wurden unigrams und bigrams, also 1-grams und 2-grams, verwendet. Diese waren bereits Teil des Baseline-Modells [14]. Die Feature-Sets werden im Folgenden mit *unigrams* bzw. *bigrams* bezeichnet. Die unigram-Features entsprechen somit tfidf-Features ohne die idf-Gewichtung. Sie sind eine sehr simple Methode, einen Text

als einen Vektor von Zahlen darzustellen. Wörter sind relevante Features, da sie Emotionsinformationen enthalten können, wie die Existenz von Emotions-Lexika zeigt. Auch für die Shared Task haben alle Teilnehmer Wörter als Informationsquelle verwendet [14]. Bigrams haben den Vorteil Wortkonstruktionen zu erhalten und möglicherweise auch Negationen mit ihrem Ziel in Verbindung zu stellen.

Die unigram-Anzahl entspricht derjenigen der tfidf-Features. Bei den bigrams wurde die Anzahl durch das Filtern um ca. 75% von 773,345 auf 191,065 reduziert.

5.4 Emotions-Lexika

Ein Emotions-Lexikon assoziiert Wörter einer Sprache mit einer diskreten Menge an Emotionen, oder in manchen Fällen auch mit Werten für Emotions-Dimensionen [2]. Entsprechend der Aufgabenstellung wurden für die Experimente zwei Lexika der ersten Variante verwendet, das WordNet-Affect Lexikon und das NRC Emotion Lexicon. Für beide Lexika wurde im ersten Schritt für jede der für die Aufgabe relevanten sechs Emotionen eine Liste mit Wörtern erstellt, welche mit der jeweiligen Emotion assoziiert sind. Darauf wird in den Unterkapiteln genauer eingegangen. Im nächsten Schritt wurden unter Verwendung dieser Listen für jedes Lexikon zwei Arten von Features erstellt. In einer Variante wurden die Wörter für alle Emotionen zusammengenommen und als Vokabular für unigram-Features wie in Kapitel 5.3 verwendet. Diese Features werden nach dem Lexikon, mit welchem sie erzeugt wurden, als *wnawords* bzw. *nrcwords* bezeichnet. Die zweite Variante erstellt sechs Features per Datum, für jede Emotion eines. Dabei entspricht das Feature für Emotion e, x_e, der Anzahl Vorkommnisse von Wörtern auf der Liste, die mit der Emotion korrespondiert. Diese Anzahlen wurden über die wie in 5.3 beschriebene Tokenisierung gezählt. Die auf diese Weise kreierten Feature-Sets werden im Folgenden *wnacounts* bzw. *nrccounts* genannt.

Emotions-Lexika wurden von einigen Teilnehmern der IEST verwendet, unter anderem von dem bestplatzierten Team [14]. Für die Label-Erstellung wurde zwar ein Emotionswort aus den Daten entfernt, aber dies schließt nicht aus, dass in den noch vorhandenen Wörtern auch solche vorkommen, die mit einer Emotion assoziiert werden können.

5.4.1 WordNet-Affect

WordNet-Affect[2] ist eine Erweiterung von WordNet [19, 35]. Sie fügt eine neue Affekt-Hierarchie hinzu, die auf existierende Synsets von WordNet verweist. Diese Hierarchie enthält neben Emotionen auch Gefühle, Stimmungen und Weiteres. Ein Knoten der Hierarchie ist "emotion", welcher zunächst weiter in "positive-emotion", "negative-emotion", "neutral-emotion" und "ambiguous-emotion" eingeteilt wird. In der Ebene darunter finden sich dann

[2] erhältlich als Teil von WordNet Domains unter http://www.wndomains.fbk.eu/download.html

die verschiedenen Emotionen, die neben den für die Task verwendeten Emotionen auch solche wie "love", "shame" und viele andere umfassen. Unterhalb dieser Emotions-Knoten befinden sich noch weitere Hierarchien.

Um die Wortlisten zu konstruieren, wurde zunächst für jede Emotion eine anfängliche Liste an Wurzelknoten festgelegt. In den meisten Fällen entsprach diese genau den Labels der Shared Task, aber bei der Kategorie "sad" war der Wurzelknoten "sadness" und die Kategorie "fear" hatte drei Wurzelknoten für "positive-fear", "negative-fear" und "ambiguous-fear". Ausgehend von diesen wurden rekursiv alle Kategorien in der Hierarchie unter den Wurzelknoten der Liste von Knoten für die Emotion hinzugefügt. Ausgehend von diesen wurden die korrespondierenden Synset-Offsets für Synsets aus WordNet über die von WordNet-Affect bereitgestellten Verbindungen gesammelt. Schließlich wurden alle Wörter aus diesen Synsets gesammelt und Duplikate entfernt. Zu beachten ist, dass dieses Verfahren in einigen Wörtern resultierte, die eigentlich zusammengesetzte Phrasen aus bis zu vier Worten sind, mit "_" als Verbindung. Dementsprechend wurden für die Bestimmung der Häufigkeiten dieser, die bigrams, trigrams und quadgrams nach dem Verfahren in 5.3 erstellt. Für eine Übersicht über die Anzahl der mit den verschiedenen Emotionen assoziierten Begriffe, siehe Tabelle 5.1.

5.4.2 NRC Emotion Lexicon

Das NRC Emotion Lexicon[3] wurde von Forschern des National Research Council Canada mit Hilfe von crowdsourcing erstellt [23, 24]. Es umfasst die acht Basis-Emotionen nach Plutchik, enthält also neben den Emotionen nach Ekman auch noch "trust" und "anticipation" [23], sowie Annotationen für "positive" bzw."negative". Das Lexikon steht in zwei Formen zur Verfügung, der Annotation auf Sinnebene und der Annotation auf Wortebene. Dabei ist die Annotation auf Wortebene die Vereinigung aller Annotationen auf Sinnebene für ein Wort. Für die Erstellung der Wortlisten wurde die Annotation auf Wortebene verwendet. Für jedes Wort und jede Emotion, sowie "positive" und "negative", enthält diese Annotation einen binären Wert, ob das Wort mit der Emotion assoziiert wird oder nicht. Die Listen wurden konstruiert, indem alle Wörter gesammelt wurden, die einen Wert von 1 für die relevante Emotion haben. Für eine Übersicht der Worthäufigkeiten per Emotion siehe Tabelle 5.1.

5.5 Language Model

Ein Language Model lernt eine Wahrscheinlichkeitsverteilung über Sätze aus Symbolen einer Sprache. Dabei gibt es zähl-basierte Ansätze und Ansätze, die auf Neural Networks basieren [11]. Für die Experimente wurde ein bereits trainiertes Language Model von jeder

[3] erhältlich unter http://www.saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm

Emotion	WordNet-Affect	NRC Emotion Lexicon
Anger	142	1247
Disgust	58	1058
Fear	156	1476
Joy	192	689
Sadness	219	1191
Surprise	75	534
Insgesamt	818	3462

Tabelle 5.1: Übersicht über die Anzahl der assoziierten Begriffe in den Emotions-Lexika

dieser Arten verwendet. Ein trainiertes Language Model erlaubt es dann, anhand eines Satzpräfixes eine wahrscheinliche Fortsetzung zu konstruieren. Da die Language Models Satzpräfixe als Eingabe zur Vorhersage benötigen, wurden diese wie folgt erzeugt. Zunächst wurden die Daten am Platzhalter `[#TRIGGERWORD#]` getrennt und nur der links vom Platzhalter stehende Teil weiter betrachtet. In diesem wurden die Platzhalter `[NEWLINE]` durch den Newline-Character ersetzt. Als Präfix für die Vorhersage wurde dann nur der Teil dieses Strings verwendet, der vor dem letzten Leerzeichen im String steht. Diese Behandlung wurde gewählt, um gesondert mit Fällen wie in Beispiel 2 und Beispiel 3 aus Kapitel 3 umzugehen. Diese verbleibenden Wortstücke, die eigentlich Teil des Triggerworts sind, wurden separat gespeichert.

Drei der vier am besten platzierten Teams der Shared Task haben Language Models verwendet [14]. Dies scheint Language Models zu einer relevanten Informationsquelle zu machen.

5.5.1 One Billion Word Language Model

Das One Billion Word Language Model[4] wurde auf der One Billion Word Benchmark trainiert [11, 3]. Es basiert auf einem Ansatz über Neural Networks, im speziellen Long Short-Term Memory (LSTM) Netzwerken und Convolutional Neural Networks (CNN). Das Vokabular des Modells umfasst ca. 800,000 Wörter [11]. Es bietet verschiedene Modi an, von denen der "sample" Modus relevant für die konstruierten Features ist. In seiner ursprünglichen Form wird diesem ein Satzpräfix übergeben und das Modell sagt für dieses konsekutiv das nächste Wort voraus, bis es ein Satzende vorhersagt oder eine maximale Wortanzahl erreicht. Dabei bestimmt das Modell in jedem Schritt eine Wahrscheinlichkeitsverteilung über das Vokabular, die bestimmt, wie wahrscheinlich ein Wort als Nächstes auftritt. Der Code für diesen Modus wurde adaptiert und angepasst, sodass lediglich die Wahrscheinlichkeitsverteilung für das erste nicht im Präfix vorhandene Wort bestimmt

[4] erhältlich unter https://www.github.com/tensorflow/models/tree/master/research/lm_1b

wird. Dabei wurden weiterhin für jedes Wort im Präfix die Berechnungsschritte durchgeführt, um aus dem gesamten zur Verfügung stehenden Kontext zu lernen. Die resultierende Wahrscheinlichkeitsverteilung wurde zur Konstruktion zweier Feature-Arten verwendet. Bezeichne $E = \{L_1, \ldots, L_6\}$ die Menge an Listen von Emotionssynonymen, die zur Datensatzkonstruktion verwendet wurden, siehe Tabelle 1 in Klinger et al. [14]. Sei (pr, r) ein Paar aus Präfix und Rest-String nach dem letzten Leerzeichen für ein Datum d und P_{pr} die vom Language Model berechnete Wahrscheinlichkeitsverteilung, dann ist

$$x_i^{(d)} = \max_{w \in L_i}(P_{pr}(r + w)), \tag{5.3}$$

wobei $x_i^{(d)}$ das i-te Feature für das Datum ist und $r + w$ hier die String-Konkatenation bezeichnet. Damit enthält der Feature-Vektor $x^{(d)}$ für Datum d in dieser Variante jeweils die maximalen Wahrscheinlichkeiten über alle der Emotion zugeordneten Wörter. Diese Features werden im Weiteren *lm_1b_probability* genannt. Die zweite Variante wandelt dies in binäre Features um, indem das Feature der Emotion mit der maximalen Wahrscheinlichkeit auf 1 und alle anderen auf 0 gesetzt werden. Dieses Feature-Set wird mit *lm_1b_binary* bezeichnet.

5.5.2 TwitterVerse

Das TwitterVerse[5] Language Model ist ein zähl-basiertes Language Model. Es wurde auf HC Corpora Daten trainiert, wobei davon nur die englischen Daten gewählt wurden. Davon sind ca. 13% Twitter-Daten, der Rest stammt aus Nachrichten-Artikeln und Blogs. Insgesamt umfasst der Datensatz 4,269,000 Zeilen und 101,393,000 Wörter, also nur etwa ein Achtel so viele wie der One Billion Word Datensatz. Die Daten wurden um Sonderzeichen, Zahlen und extra Whitespace bereinigt und in Wörter aufgetrennt. Aus diesen wurden unigrams, bigrams, trigrams und quadgrams konstruiert und ihr Vorkommen gezählt. Daraus wurden R-Dataframes mit zusätzlichen Statistiken für das Language Model konstruiert. Die Vorhersage des Language Models basiert auf dem Katz-Backoff-Modell [12]. Das ursprüngliche Modell berechnet für ein Präfix bis zu zehn wahrscheinliche Möglichkeiten für das nächste Wort. Diese Limitation wurde entfernt und es wurden alle Möglichkeiten vorhergesagt, außer in den Fällen, in denen aufgrund nicht existierender Präfix-Informationen alle Inhalte des unigram-Dataframe valide Vorhersagen waren. Dort wurde die Vorhersage auf die 2500 wahrscheinlichsten, also in den Trainingsdaten des Modells am häufigsten auftretenden, beschränkt. Aus den Vorhersagen des Language Models wurden wie folgt Features gewonnen. Bezeichne $E = \{L_1, \ldots, L_6\}$ die Menge an Listen von Emotionssynonymen, die zur Datensatzkonstruktion verwendet wurden, siehe Tabelle 1 in Klinger et al. [14]. Sei (pr, r) ein Paar aus Präfix und Rest-String nach dem letzten Leerzeichen für ein Datum d und M_{pr} die vom Language Model berechnete Menge an Vorhersagen. Dann wird

[5] erhältlich unter https://www.github.com/jayurbain/TwitterVerse

die i-te Komponente des Feature-Vektors $x^{(d)}$ auf 1 gesetzt, falls $\exists w \in L_i : r + w \in M_{pr}$ und sonst auf 0. Dabei bezeichnet $r + w$ hier die String-Konkatenation. Die so erzeugten Features werden mit *twitterverse* bezeichnet.

5.6 Word-Embeddings

Word-Embeddings bilden Wörter auf einen niedrig-dimensionalen Vektorraum ab. Niedrig-dimensional heißt dabei in der Größenordnung von einigen Hunderten. Dabei sollen Wörter mit ähnlicher semantischer Bedeutung auf ähnliche Vektoren abgebildet werden [21]. Die Vektoren sind reellwertig und enthalten üblicherweise keine Null-Features. Da Word-Embeddings allerdings nur für Wörter direkt trainiert sind, wird ein Satz, oder auch eine längere Wortfolge, in einen einzelnen Vektor überführt, indem der Mittelwert aller Wortvektoren des Satzes bestimmt wird [10]. Für die Experimente dieser Arbeit wurde das word2vec-twitter Word-Embedding genutzt. Embeddings wurden von nahezu allen Teilnehmern der Shared Task verwendet [14]. Sie wurden außerdem erfolgreich für die Verbesserung der Klassifikationsgenauigkeit im Bereich der Sentiment Analysis angewendet [21].

5.6.1 word2vec-twitter

Das word2vec-twitter[6] Word-Embedding ist auf Twitter-Daten mit dem word2vec-Algorithmus nach Mikolov trainiert worden [7, 18]. Die Trainingsmenge bestand dabei aus 400 Millionen englischsprachigen Tweets [7]. Die Word-Embeddings umschließen aufgrund der Art der Trainingsdaten auch Twitter-spezifische Besonderheiten wie Hashtags. Die Dimension des Word-Embeddings beträgt 400 und das Vokabular umfasst 3,039,345 Wörter. Für die Repräsentation der Daten wurden zunächst Tokens wie für die tfidf-Features gewonnen, jedoch nicht nach Häufigkeit gefiltert. Für jedes Datum wurde daraufhin von denjenigen seiner Tokens, die im Vokabular enthalten sind, der Mittelwert ihrer Embeddings gebildet, um das Datum zu repräsentieren. Die konstruierten Features werden im Weiteren mit *wordembedding* referenziert.

[6] erhältlich unter https://www.fredericgodin.com/software, für die Experimente verwendet wurde der Code von https://www.github.com/loretoparisi/word2vec-twitter

Kapitel 6

Experimente

Für diese Arbeit wurden Experimente mit den oben beschriebenen Methoden und Feature-Sets durchgeführt. Dabei wurden alle Modelle auf den gesamten Trainingsdaten trainiert. Für die Parameteroptimierung wurde der gesamte Trial-Datensatz verwendet. Für die Experimente mit logistischer Regression (LR) und der Support Vector Machine (SVM) wurden Experimente mit diversen Kombinationen von Feature-Sets durchgeführt.

6.1 Aufbau

Im Falle der logistischen Regression und der linearen SVM wurden zunächst für jedes Experiment die verwendeten Feature-Sets ausgewählt und aus diesen eine einzelne Trainingsdatei erstellt. Auf die gleiche Weise wurde jeweils eine Datei für die Trialdaten erstellt. Die Trainingsdateien wurden als Eingabe für LIBLINEAR [6] verwendet. Die Daten für das Character-level Convolutional Neural Network (Character-CNN) wurden, wie in Kapitel 4.3 beschrieben, in eine passende Eingaberepräsentation umgewandelt. Dabei wurde das Alphabet aus Abbildung 4.1 verwendet.

6.1.1 Logistische Regression und lineare SVM

Mit der logistischen Regression und der linearen SVM wurden jeweils die gleichen Feature-Kombinationen ausprobiert. Eine Übersicht über die verschiedenen Kombinationen findet sich in Tabelle 6.1 und Tabelle 6.2. Für die Experimente wurde jeweils die beste Kombination von Werten für den Regularisierungsparameter C und die Bias-Konstante B, beide aus der Menge $\{0.0001, 0.001, 0.01, 0.1, 1, 10\}$, gesucht. Die Güte dieser Kombinationen wurde dabei über den Trial-Datensatz bestimmt.

Da elf Feature-Sets zur Auswahl stehen, sind $2^{11} = 2048$ verschiedene Experimente pro Methode möglich, von denen hier jeweils 32 ausgetestet wurden.

		Experimente															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Features	tfidf	X												X			
	unigrams		X										X		X	X	X
	bigrams			X									X	X	X	X	X
	nrcwords				X										X		
	wnawords					X										X	
	nrccounts						X										X
	wnacounts							X									
	wordembedding								X								
	lm_1b_binary									X							
	lm_1b_probability										X						
	twitterverse											X					

Tabelle 6.1: Erster Teil der LR- bzw SVM-Experimente

		Experimente															
		17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Features	tfidf						X	X						X			
	unigrams	X	X	X	X	X	X	X						X			
	bigrams	X	X	X	X	X	X	X				X	X	X			
	nrcwords							X	X	X				X	X	X	X
	wnawords							X	X	X				X	X	X	X
	nrccounts							X	X	X				X	X	X	X
	wnacounts	X						X	X	X			X	X	X	X	X
	wordembedding		X					X				X		X	X		X
	lm_1b_binary			X				X	X		X			X	X	X	X
	lm_1b_probability				X			X	X		X			X	X	X	X
	twitterverse					X		X	X		X						X

Tabelle 6.2: Zweiter Teil der LR- bzw SVM-Experimente

1	{svm-1, svm-2, svm-3, lr-4, svm-5, svm-6, svm-7, lr-8, svm-9, svm-10, svm-11}
2	{svm-9, svm-10, svm-11}
3	{lr-4, svm-5, svm-6, svm-7}
4	{svm-2, svm-3, svm-7}
5	{svm-17, lr-18, svm-22}
6	{svm-12, svm-14, lr-17, svm-17, svm-20}
7	{svm-14, svm-15, svm-16, svm-17, lr-18, svm-19, svm-20, svm-21, svm-22}
8	{svm-17, lr-18, svm-22, lr-24}
9	{svm-17, lr-24, lr-27, svm-28}
10	{cnn, svm-17, lr-17}
11	{cnn, svm-12, svm-14, lr-17, svm-17, svm-20}
12	{cnn, svm-12, svm-17}
13	{cnn, svm-17, lr-32}
14	{cnn, svm-12, svm-14, svm-17, svm-20}
15	bestes Ensemble der Größe 3
16	bestes Ensemble der Größe 4

Tabelle 6.3: Übersicht über die Ensemble-Experimente

6.1.2 Character-CNN

Für die Versuche mit dem Character-CNN wurde die Architektur wie in Kapitel 4.3 genutzt. Als loss-Funktion wurde sparse-categorical-crossentropy verwendet. Für die Optimierung wurde Stochastic Gradient Descent mit einer initialer Lernrate von 0.01 eingesetzt. Die Lernrate wurde alle fünf Epochen geviertelt. Es wurden drei Initialisierungen des Experiments durchgeführt und das beste Ergebnis über diese gewählt. Für jede Initialisierung wurde das Modell 15 Epochen lang trainiert. Die beste Initialisierung wurde über die Ergebnisse auf dem Trial-Datensatz ermittelt.

6.1.3 Ensemble

Die Ensembles wurden aus den Experimenten der verschiedenen Methoden zusammengesetzt. Dabei wurden Ensembles verschiedener Größen ausprobiert. Da über alle Methoden insgesamt 65 Experimente zur Kombination zur Verfügung standen, gibt es $\binom{65}{k}$ mögliche Ensembles der Größe k. Für $k = 3$ sind dies bereits $\binom{65}{3} = 43{,}680$ Möglichkeiten. Es wurden daher ein paar wenige Ensembles ausgewählt, basierend auf Ergebnissen der einzelnen Experimente. Außerdem wurde für die Größen $k = 3$ und $k = 4$ auch das beste Ensemble dieser Größe bestimmt. Die ausprobierten Ensembles finden sich in Tabelle 6.3.

6.2 Implementation

Die Experimente wurden in Python durchgeführt, insbesondere wurden Jupyter[1] Notebooks verwendet. Für die logistische Regression und Support Vector Machine wurde dabei das Python-Interface von LIBLINEAR [6] genutzt. Das Character-CNN wurde mit Keras[2], mit TensorFlow[3] im Backend, implementiert. Für die Feature-Konstruktion wurde, neben dem NLTK[4] TweetTokenizer, die NLTK-Schnittstelle für WordNet [19] verwendet. Außerdem wurde der Code für das TwitterVerse Language Model in R angepasst und in Python über das rpy2-package integriert. Der für die Arbeit verwendete Code ist online verfügbar[5].

[1] https://www.jupyter.org
[2] https://www.keras.io
[3] https://www.tensorflow.org
[4] https://www.nltk.org
[5] https://www.github.com/RahelWilking/bachelorarbeit/tree/master/code

Kapitel 7

Evaluation

Die Evaluation der Experimente findet in zwei Stufen statt. Zunächst werden die Ergebnisse aller Experimente auf dem Trial-Datensatz betrachtet. Anhand dieser werden die vielversprechenden Varianten ausgewählt und auf den Test-Datensatz angewandt. Die Ergebnisse auf diesem werden anschließend genauer betrachtet und mit den Ergebnissen der Implicit Emotions Shared Task (IEST) verglichen.

7.1 Evaluationsmetriken

Zur Evaluation wird der macro-averaged F_1-score benutzt. Sei p die Anzahl der Klassen und f ein Klassifizierer. Für eine Klasse i sei TP_i die Anzahl der *true positives*, also der Beispiele x, die der Klasse i angehören und für die $f(x) = i$ gilt. Die Anzahl Beispiele der Klasse i für die $f(x) \neq i$ gilt, die *false negatives*, sei FN_i. Für Beispiele die i nicht angehören ist FP_i die Anzahl für die $f(x) = i$ gilt, dies sind die *false positives*, und TN_i die, für die $f(x) \neq i$ gilt, die *true negatives*. Der F_1-score kombiniert die Precision einer Klasse i

$$pre_i = \frac{TP_i}{TP_i + FP_i} \tag{7.1}$$

und den Recall dieser

$$rec_i = \frac{TP_i}{TP_i + FN_i} \tag{7.2}$$

wie folgt [32]:

$$F_{1i} = \frac{2 \cdot pre_i rec_i}{pre_i rec_i}. \tag{7.3}$$

Dieser Wert wird für alle Klassen einzeln berechnet und dann zu einem Wert zusammengefasst über macro-averaging [32]:

$$F_1 = \frac{\sum_{i=1}^{p} F_{1i}}{p}. \tag{7.4}$$

Der macro-averaged F_1-score ist das offizielle Vergleichsmaß der IEST und wird auch im bereitgestellten Evaluationsskript berechnet.

7.2 Einzelne Methoden

Die macro-F_1 Ergebnisse auf den Trialdaten für die Experimente mit logistischer Regression (LR) und der linearen Support Vector Machine (SVM) sind in Tabelle 7.1 bzw. Tabelle 7.2 zu finden, sie sind auf vier Nachkommastellen gerundet. Das jeweils beste erzielte Ergebnis ist in den Tabellen hervorgehoben. Die Nummern der Experimente korrespondieren mit denen aus Tabellen 6.1 und 6.2. Die Baseline erreicht auf den Trialdaten einen auf vier Nachkommastellen gerundeten macro-F_1 Wert von 0.6010. In Abbildung 7.1 sind die Abweichungen der Ergebnisse der LR- und SVM-Experimente von der Baseline visualisiert.

Beide Methoden erreichen den höchsten macro-F_1 Wert für Experiment 17, welches die Feature-Sets *unigrams*, *bigrams* und *wnacounts* umfasst. Dabei ist die SVM etwas besser als die logistische Regression. Die Hinzunahme der *wnacounts* als Features erreicht, gerade im Fall der SVM, allerdings nur eine minimale Verbesserung gegenüber der Verwendung von *unigrams* und *bigrams* alleine, was dem Experiment 12 entspricht. Weiterhin ist anzumerken, dass die Verwendung von *tfidf* Features alleine schlechtere Resultate bringt als die Verwendung roher Häufigkeiten mit *unigrams* (vergleiche Experimente 1 und 2). Dies gilt insbesondere für die Verbindung von *bigrams* mit jeweils einem dieser beiden Feature-Sets, wie durch den Vergleich von Experiment 12 und Experiment 13 deutlich wird.

In den Experimenten 1 bis 11, in denen jeweils ein Feature-Set alleine verwendet wurde, erreichen *bigrams* den höchsten macro-F_1 Wert, gefolgt von *unigrams*, *tfidf* und *wordembedding*. Den geringsten Wert erzielt Experiment 7, für welches *wnacounts* verwendet wurden. Dies ist besonders verwunderlich in Anbetracht dessen, dass bei Betrachtung der Experimente 14 bis 22, in welchen *unigrams* und *bigrams* jeweils mit einem der anderen Feature-Sets kombiniert wurden, lediglich Experiment 17 eine Verbesserung im Vergleich zu Experiment 12 erzielt hat. Alle anderen Ergänzungen von Feature-Sets zu *unigrams* und *bigrams* haben mindestens eine kleine Verschlechterung mit einher getragen, die drastischste dabei bei der Ergänzung von *tfidf* in Experiment 22.

In Experiment 23 wurden alle Features kombiniert, welches allerdings zu einem noch schlechteren Ergebnis als Experiment 22 geführt hat. Bereits das Weglassen eines einzelnen Feature-Sets wie *twitterverse* in Experiment 29 führt zu einer kleinen Verbesserung im macro-F_1 Wert. Im Gegensatz dazu führt das Weglassen ebendieses Feature-Sets, der Unterschied zwischen Experiment 32 und Experiment 30, zu einer Verschlechterung, ebenso bei Experiment 24 und 31.

Die Verwendung von lediglich den Emotions-Lexika-Features in Experiment 25, führt zu einem deutlich schlechteren Ergebnis als die ausschließliche Verwendung von Language Model Features in Experiment 26.

Allgemein scheint die SVM die erfolgreichere Methode zu sein. Sie erzielt in 18 der 32 Experimente ein besseres Ergebnis, die logistische Regression lediglich in 12. In Experiment 9 und 11 sind beide Methoden gleich gut, auch bezüglich der nicht-gerundeten

	1	2	3	4	5	6	7	8
macro-F_1	0.5485	0.5505	0.5818	0.2436	0.1405	0.1878	0.1190	0.4876
	9	10	11	12	13	14	15	16
macro-F_1	0.2918	0.2921	0.2208	0.6174	0.5862	0.6173	0.6173	0.6146
	17	18	19	20	21	22	23	24
macro-F_1	**0.6183**	0.6131	0.6142	0.6159	0.6118	0.5856	0.5779	0.3847
	25	26	27	28	29	30	31	32
macro-F_1	0.2680	0.3519	0.5700	0.5803	0.5800	0.5044	0.3747	0.5119

Tabelle 7.1: Ergebnisse der LR-Experimente

	1	2	3	4	5	6	7	8
macro-F_1	0.5498	0.5518	0.5851	0.2433	0.1464	0.1887	0.1215	0.4823
	9	10	11	12	13	14	15	16
macro-F_1	0.2918	0.2932	0.2208	0.6198	0.5873	0.6182	0.6179	0.6175
	17	18	19	20	21	22	23	24
macro-F_1	**0.6200**	0.6119	0.6162	0.6180	0.6135	0.5859	0.5752	0.3756
	25	26	27	28	29	30	31	32
macro-F_1	0.2664	0.3331	0.5690	0.5861	0.5781	0.5017	0.3559	0.5098

Tabelle 7.2: Ergebnisse der SVM-Experimente

F_1 Werte. Die logistische Regression scheint bei Experimenten mit *wordembedding* erfolgreicher zu sein, sowie bei Experimenten mit vielen Feature-Sets, gerade wenn diese nicht *tfidf*, *unigrams* oder *bigrams* sind. Der größte Unterschied zugunsten der SVM besteht bei Experiment 5 und der größte Unterschied zugunsten der LR bei Experiment 26.

Das Character-CNN hat auf den Trialdaten einen macro-F_1 Wert von 0.5520, auf vier Nachkommastellen gerundet, erreicht. Dies ist deutlich schlechter als die besten Resultate für die SVM und LR, allerdings etwa gleichwertig dazu, diese nur auf *unigrams* zu verwenden.

Basierend auf den Ergebnissen der Experimente auf den Trialdaten scheint das Experiment svm-17 am besten zu sein. Während der Shared Task wäre also dies die wahrscheinlich beste Einreichung gewesen. Parallel zum Vorgehen der Shared Task wurde deshalb nur dieses auf den Testdaten ausgewertet. Das Ergebnis findet sich in Tabelle 7.7. Das Experiment schneidet auf den Testdaten besser ab als die von den Veranstaltern bereitgestellte Baseline, ist aber weit entfernt von dem Ergebnis des besten für die Shared Task eingereichten Systems. Dieses wurde von Rozental et al. entwickelt und besteht aus einem

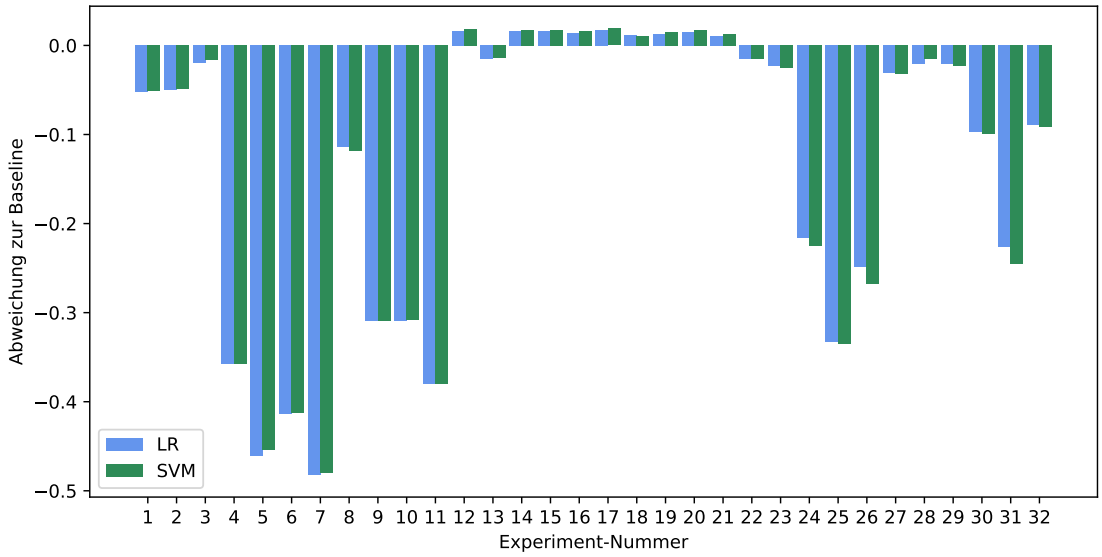

Abbildung 7.1: Abweichungen der LR- und SVM-Experimente von der Baseline

Ensemble-Ansatz. Für dieses wurden eigens Word-Embeddings und Language Models trainiert, auf einem extra konstruierten Twitter-Datensatz mit fünf Milliarden Tweets. Eines der verwendeten Language Models wurde über zwei Tage auf acht Nvidia Tesla V100 GPUs trainiert. Weiterhin wurden Systeme, welche für die SemEval-2018 Task 1 Sub-Aufgaben 1 und 5 entwickelt wurden, für Features verwendet. Weitere Features wurden über Universal Sentence Emebdding und DeepMoji gewonnen. Verbunden wurden diese Komponenten in einem mehrere Ebenen umfassenden Ensemble. In der ersten Ebene wurden Kombinationen aus bi-LSTMs und CNNs mit verschiedenen Features und Parametern trainiert, in der zweiten Ebene wurde darauf aufbauend ein weiteres Neural Network trainiert, welches acht Ensembles der ersten Ebene mit weiteren Features verbindet. Details zur Architektur finden sich in Rozental et al. [31].

Vergleicht man das Ergebnis des besten einzelnen Experiments mit der Tabelle 5 in Klinger et al. [14], in der die offiziellen Ergebnisse der Teilnehmer der Shared Task gelistet werden, so fällt das Experiment svm-17 zwischen die Plätze 18 und 19. Dabei korrespondiert der Platz 19 mit meiner eigenen Einreichung für die Shared Task, welche in etwa dem Experiment 12 entspricht.

Für das Experiment svm-17 wurde weiterhin über die sechs Klassen (A)nger, (D)isgust, (F)ear, (J)oy, (Sa)dness und (Su)rprise eine Konfusionsmatrix aufgestellt, siehe Tabelle 7.3. Zum Vergleich wurden die Konfusionsmatrizen für die Baseline und das beste System der IEST aus Klinger et al. [14] übernommen. Sie finden sich in Tabelle 7.4 und Tabelle 7.5.

Im Vergleich zur Baseline klassifiziert svm-17 fast alle Klassen häufiger richtig, der größte Unterschied besteht dabei bei der Klasse Sadness. Die einzige Klasse, für welche die Baseline eine höhere Anzahl true positives hat, ist Surprise mit einem einzigen richtig

		Predicted Labels					
		A	D	F	J	Sa	Su
Gold Labels	A	2494	493	509	354	311	633
	D	386	3019	250	203	374	562
	F	398	259	3041	274	236	583
	J	335	180	336	3794	266	335
	Sa	446	429	326	383	2450	306
	Su	385	489	464	286	239	2929

Tabelle 7.3: Konfusionsmatrix des Experiments svm-17 auf den Testdaten

		Predicted Labels					
		A	D	F	J	Sa	Su
Gold Labels	A	2431	476	496	390	410	426
	D	426	2991	245	213	397	522
	F	430	249	3016	327	251	518
	J	378	169	290	3698	366	345
	Sa	450	455	313	458	2335	329
	Su	411	508	454	310	279	2930

Tabelle 7.4: Konfusionsmatrix der Baseline auf den Testdaten, übernommen aus Klinger et al. [14]

		Predicted Labels					
		A	D	F	J	Sa	Su
Gold Labels	A	3182	313	293	224	329	453
	D	407	3344	134	102	336	471
	F	403	129	3490	196	190	383
	J	297	67	161	4284	220	217
	Sa	443	340	171	240	2947	199
	Su	411	367	293	209	176	3336

Tabelle 7.5: Konfusionsmatrix des besten Systems der IEST auf den Testdaten, übernommen aus Klinger et al. [14]

	1	2	3	4	5	6	7	8
macro-F_1	0.5336	0.2877	0.2119	0.5180	0.6181	0.6193	0.6192	0.6150
	9	10	11	12	13	14	15	16
macro-F_1	0.5995	0.6197	0.6202	0.6210	0.6033	0.6201	0.6223	**0.6249**

Tabelle 7.6: Ergebnisse der Ensemble-Experimente

klassifizierten Beispiel mehr. Das Experiment svm-17 sagt deutlich häufiger als die Baseline die Klasse Surprise vorher, wenn die eigentliche Klasse Anger ist. Dafür ist es jedoch deutlich besser darin Sadness und Joy auseinander zu halten.

Vergleicht man das Experiment svm-17 mit dem besten System der IEST, so klassifiziert letzteres, wie von den macro-F_1 Werten zu erwarten, alle Klassen besser. Die stärkste Verbesserung besteht in der Klassifizierung der Klasse Anger, die geringste bei Klasse Disgust. Nahezu jeder Wert des Experiments svm-17 ist schlechter, allerdings klassifiziert es etwas seltener Anger als Sadness und Disgust, Fear oder Surprise als Anger.

Die häufigste Fehlklassifikation von svm-17 besteht darin, Beispiele der Klasse Anger als Surprise zu klassifizieren, die geringste dabei Klasse Joy als Disgust zu klassifizieren. Letzteres deckt sich mit der Baseline sowie dem besten System der IEST.

7.3 Ensemble

Die Ensemble-Ergebnisse auf den Trialdaten sind in Tabelle 7.6 dargestellt. Der macro-F_1 Wert ist dabei jeweils auf vier Nachkommastellen gerundet und die Nummern der Experimente entsprechen denen aus Tabelle 6.3. Das beste Ergebnis ist in der Tabelle hervorgehoben und wird von ensemble-16 erreicht. Dieses entspricht dem über alle Einzel-Experimente bestmöglichen Ensemble der Größe 4, welches aus den Experimenten cnn, lr-3, svm-17 und svm-18 besteht. Das Experiment ensemble-15, welches dem besten Ensemble der Größe 3 entspricht, besteht aus den Experimenten cnn, svm-17 und lr-18. Interessant ist, dass das Experiment svm-18, im Gegensatz zu lr-18, Teil des besten Ensembles der Größe 4 ist, da beide auf den gleichen Features trainiert wurden und lr-18 ein besseres Ergebnis erzielt. Der Wert des besten Ensemble-Experiments ist leicht besser als der von svm-17, welches den besten Wert der einzelnen Experimente erreicht. Dabei schaffen es von den ausprobierten Kombinationen lediglich die Ensembles 11 bis 16 diesen Wert zu übertreffen und auch überhaupt den besten Wert eines einzelnen am Ensemble beteiligten Experiments zu übersteigen. Diese haben alle gemein, dass das cnn-Experiment Teil des Ensemble ist. Das einzige Ensemble mit cnn, welches den Wert nicht übersteigt, Experiment 10, erfährt auch nur eine minimale Verschlechterung.

	svm-17	ensemble-16	best-iest	baseline
macro-F_1	0.6141	0.6154	0.7145	0.5988

Tabelle 7.7: Ergebnisse der ausgewählten besten Experimente auf den Testdaten, verglichen mit dem besten Ergebnis der IEST und der Baseline

Für ensemble-1 wurden die Experimente gewählt, die jeweils nur auf einem Feature-Set durchgeführt wurden. Dabei wurde jeweils die bessere der beiden Methoden gewählt, bei Gleichheit wurde die SVM verwendet. Der erreichte Wert des Ensembles liegt unterhalb der besten drei einzelnen Ergebnisse, hat also eine deutliche Verschlechterung einhergebracht. Im Vergleich zu den Experimenten lr-23 und svm-23, in welchen die Feature-Sets direkt alle zusammen verwendet wurden, liefert die Ensemble-Kombination ebenfalls schlechtere Ergebnisse.

Die Ensembles 2 und 3 liefern im Vergleich zu ihren korrespondierenden direkten Kombinationen in lr-/svm-26 bzw. 25 auch deutlich verschlechterte Klassifikationen. Eine Kombination der einzelnen Experimente über die Feature-Sets die für svm-17 verwendet wurden, wie in ensemble-4, ist auch weit entfernt von der Güte der direkten Kombination als Features für einen linearen Klassifizierer. Ein wenig mehr Erfolg hat die Kombination von drei der besseren Einzel-Experimente in ensemble-5. Für das sechste Ensemble wurden allerdings die fünf besten Klassifizierer der Einzel-Experimente ausgewählt und auch diese erreichen keine Verbesserung über svm-17. Kombiniert mit dem cnn-Experiment in Ensemble Nummer 11 erreichen sie jedoch einen sehr leicht höheren Wert als dieses. Die Entfernung des schlechteren der beiden Experimente für Feature-Kombination 17 erleidet, wie ensemble-14 zeigt, eine minimale Verschlechterung.

In ensemble-7 wurden die jeweils besseren der Experimente 14 bis 22 kombiniert, welche alleine alle eine moderate bis gute Leistung bringen. Sie haben jeweils die *unigrams* und *bigrams* Features gemein und decken zusammen alle Feature-Sets ab. Ihre Kombination erreicht einen Wert nahe an svm-17, ist jedoch immer noch schlechter als dieses.

Wie für die einzelnen Experimente begründet, wurde lediglich für das beste Ensemble der Wert auf den Testdaten ermittelt und ist in Tabelle 7.7 aufgeführt. Wie auch auf den Trialdaten erzielt das Ensemble einen etwas höheren macro-F_1 Wert als svm-17 auf den Testdaten, allerdings fällt der Unterschied deutlich geringer aus. Das Ergebnis des Ensembles ist ebenfalls weit von dem besten System der IEST entfernt, schlägt allerdings die Baseline-Ergebnisse. Wie auch schon svm-17 würde ensemble-16 in Tabelle 5 von Klinger et al. [14] zwischen den Plätzen 18 und 19 eingeordnet werden.

Auch für das beste Ensemble-Ergebnis wurde die zugehörige Konfusionsmatrix aufgestellt, sie findet sich in Tabelle 7.8. Im Vergleich zu dem Experiment svm-17, dessen Konfusionsmatrix in Tabelle 7.3 dargestellt ist, klassifiziert das Ensemble Beispiele der Klassen Anger, Fear, Joy und Surprise häufiger richtig. Für die Klassen Disgust und Sad-

		Predicted Labels					
		A	D	F	J	Sa	Su
Gold Labels	A	2664	450	397	376	292	615
	D	468	3004	203	220	333	566
	F	540	259	2897	288	226	581
	J	395	161	268	3823	244	355
	Sa	511	452	260	404	2421	292
	Su	469	481	391	293	226	2932

Tabelle 7.8: Konfusionsmatrix des Experiments ensemble-16 auf den Testdaten

ness ist svm-17 der bessere Klassifikator. Das Ensemble erkennt falsche Emotionen deutlich seltener als Fear, verwechselt allerdings andere Klassen deutlich häufiger für Anger.

Verglichen mit der Baseline in Tabelle 7.4 erkennt das Ensemble bis auf Fear alle Emotionen besser richtig. Die größte Verbesserung besteht dabei bei Klasse Anger. Diese wird allerdings allgemein vermehrt vorhergesagt, auch für Beispiele anderer Klassen. Die falschen Vorhersagen der Klasse Fear sind dagegen jedoch reduziert, allerdings auch die gesamte Zahl der Vorhersagen der Klasse Fear. Die Vorhersage der Klasse Joy ist insbesondere für die Klassen Fear und Sadness verringert, die der Klasse Sadness deutlich für die Klassen Anger und Joy, aber auch für andere Klassen. Deutlich höhere Fehlerraten treten bei der Vorhersage der Klasse Surprise für Beispiele der Klasse Anger auf. Dies ist auch der am häufigsten auftretende Fehler.

Die Ergebnisse des besten Systems der IEST aus Tabelle 7.5 zeigen, wie nach dem Vergleich des F_1-score zu erwarten war, dass das Ensemble in allen Klassen mehr Fehler macht. Etwas besser ist das Ensemble darin, Disgust nicht als Sadness zu erkennen. Alle anderen Fehler treten häufiger auf als beim System der IEST.

Der am häufigsten auftretende Fehler des Ensembles ist es, Anger-Beispiele als Surprise zu erkennen, wie schon beim Experiment svm-17. Die geringste Fehlerzahl tritt ebenfalls dabei auf Joy als Disgust zu erkennen, wie auch für die Baseline und das beste System der IEST. Joy und Disgust scheinen relativ einfach zu trennende Emotionen zu sein. Auch in die andere Richtung tritt eine relativ geringe Fehleranzahl auf, über alle betrachteten Konfusionsmatrizen jeweils die zweitgeringste.

Kapitel 8

Zusammenfassung und Ausblick

Diese Arbeit hat sich damit beschäftigt drei Methoden des maschinellen Lernens und elf Feature-Sets für eine Emotions-Klassifikationsaufgabe zu verwenden. Das erfolgreichste einzelne Experiment war dabei eine lineare Support Vector Machine mit *unigrams*, *bigrams* und *wnacounts* als Features. Die Support Vector Machine hat sich als etwas erfolgreicher als die logistische Regression erwiesen, wobei letztere allerdings ihre Stärke in Verwendung mit *wordembeddings* gezeigt hat, sowie bei Kombinationen von vielen verschiedenen Features. Das Character-CNN hat eine etwas schlechtere Leistung erbracht, war aber etwa auf dem Niveau einen der linearen Klassifizierer nur mit *unigrams* zu verwenden.

Für die Ensembles hat sich das CNN allerdings als wichtige Komponente erwiesen, die Teil aller Verbesserungen über die Einzel-Experimente war. Das beste Ensemble war dabei zusammengesetzt aus dem Character-CNN und den Experimenten lr-3, svm-17 und svm-18, beinhaltet also auch das beste Einzel-Experiment. Mit Ensembles konnten kleine Verbesserungen erreicht werden. Die beste Einreichung der IEST liegt deutlich über dem besten erzielten Ergebnis dieser Arbeit, nutzt allerdings auch extensive Ressourcen, um dies zu erreichen.

Eine Aufschlüsselung der Ergebnisse auf die einzelnen Klassen zeigte eine gute Trennbarkeit der Emotionen Joy und Disgust über die durchgeführten Experimente, sowie den Resultaten der IEST. Die von Mohammad et al. durchgeführte Vergleichsstudie mit menschlichen Annotationen zeigt, dass die automatischen Systeme bessere Ergebnisse als diese erzielen [14]. Dies motiviert eine weitere Verbesserung der automatischen Klassifikationssysteme im Bereich der Emotionserkennung. Mögliche Verbesserungen könnten im Bereich des Character-CNN erreicht werden, durch eine ausführlichere Beschäftigung mit der Konstruktion einer passenden Architektur. Die Ensemble-Kombination von Experimenten, die mit dem gleichen maschinellen Lernverfahren über verschiedene Feature-Sets trainiert wurden, hatte wenig Erfolg. Dafür hat sich aber die Kombination von sich deutlicher voneinander unterscheidenden Methoden, wie der SVM und dem Character-CNN als vielversprechend erwiesen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Alpaydin, Ethem: *Introduction to Machine Learning.* MIT Press, Dritte Auflage, 2014.

[2] Bradley, Margaret M. und Peter J. Lang: *Affective Norms for English Words (ANEW): Instruction manual and affective ratings.* Technischer Bericht C-1, The Center for Research in Psychophysiology, University of Florida, 1999.

[3] Chelba, Ciprian, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn und Tony Robinson: *One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling.* arXiv preprint arXiv:1312.3005, 2014.

[4] Dreiseitl, Stephan und Lucila Ohno-Machado: *Logistic regression and artificial neural network classification models: a methodology review.* Journal of Biomedical Informatics, 35:352–359, 2002.

[5] Ekman, Paul: *An Argument for Basic Emotions.* Cognition and Emotion, 6(3/4):169–200, 1992.

[6] Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang und Chih-Jen Lin: *LIBLINEAR: A library for large linear classification.* Journal of Machine Learning Research, 9:1871–1874, 2008.

[7] Godin, Fréderic, Baptist Vandersmissen, Wesley De Neve und Rik Van de Walle: *Multimedia Lab @ ACL W-NUT NER Shared Task: Named Entity Recognition for Twitter Microposts using Distributed Word Representations.* In: *Proceedings of the Workshop on Noisy User-generated Text*, Seiten 146–153. Association for Computational Linguistics, 2015.

[8] Hasan, Maryam, Elke A. Rundensteiner und Emmanuel Agu: *EMOTEX : Detecting Emotions in Twitter Messages.* In: *2014 ASE BIGDATA/SOCIALCOM/CYBERSECURITY Conference, Stanford University, May 27-31, 2014*, 2014.

[9] Joachims, Thorsten: *Text Categorization with Support Vector Machines: Learning with Many Relevant Features.* Technischer Bericht 23, Universität Dortmund, LS VIII-Report, 1997.

[10] Joulin, Armand, Edouard Grave, Piotr Bojanowski und Tomas Mikolov: *Bag of Tricks for Efficient Text Classification.* In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Seiten 427–431. Association for Computational Linguistics, 2017.

[11] Jozefowicz, Rafal, Oriol Vinyals, Mike Schuster, Noam Shazeer und Yonghui Wu: *Exploring the Limits of Language Modeling.* arXiv preprint arXiv:1602.02410, 2016.

[12] Katz, Slava M.: *Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer.* IEEE Transactions on Acoustics, Speech, and Signal Processing, 35:400–401, 1987.

[13] Kittler, Josef, Mohamad Hatef, Robert P. W. Duin und Jiri Matas: *On Combining Classifiers.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 20:226–239, 1998.

[14] Klinger, Roman, Orphee De Clercq, Saif Mohammad und Alexandra Balahur: *IEST: WASSA-2018 Implicit Emotions Shared Task.* In: *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Seiten 31–42. Association for Computational Linguistics, 2018.

[15] Li, Shoushan, Chengqing Zong und Xia Wang: *Sentiment Classification through Combining Classifiers with Multiple Feature Sets.* 2007 International Conference on Natural Language Processing and Knowledge Engineering, Seiten 135–140, 2007.

[16] Li, Weiyuan und Hua Xu: *Text-based emotion classification using emotion cause extraction.* Expert Systems with Applications, 41:1742–1749, 2014.

[17] Liew, Jasy Suet Yan und Howard R. Turtle: *Exploring Fine-Grained Emotion Detection in Tweets.* In: *Proceedings of the NAACL Student Research Workshop*, Seiten 73–80. Association for Computational Linguistics, 2016.

[18] Mikolov, Tomas, Kai Chen, Gregory S. Corrado und Jeffrey Dean: *Efficient Estimation of Word Representations in Vector Space.* CoRR, abs/1301.3781, 2013.

[19] Miller, George A.: *WordNet: A Lexical Database for English.* Communications of the ACM, 38(11):39–41, 1995.

[20] Mohammad, Saif, Felipe Bravo-Marquez, Mohammad Salameh und Svetlana Kiritchenko: *SemEval-2018 Task 1: Affect in Tweets.* In: *Proceedings of The 12th International Workshop on Semantic Evaluation*, Seiten 1–17. Association for Computational Linguistics, 2018.

[21] Mohammad, Saif M.: *Sentiment Analysis: Detecting Valence, Emotions, and Other Affectual States from Text.* In: Meiselman, Herb (Herausgeber): *Emotion Measurement.* Elsevier, 2016.

[22] Mohammad, Saif M. und Felipe Bravo-Marquez: *WASSA-2017 Shared Task on Emotion Intensity.* In: *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*, 2017.

[23] Mohammad, Saif M. und Peter D. Turney: *Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon.* In: *Proceedings of the NAACL-HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, 2010.

[24] Mohammad, Saif M. und Peter D. Turney: *Crowdsourcing a Word-Emotion Association Lexicon.* Computational Intelligence, 29(3):436–465, 2013.

[25] Nagarsekar, Uma, Aditi Mhapsekar, Priyanka Kulkarni und Dhananjay R. Kalbande: *Emotion detection from "the SMS of the internet".* In: *2013 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, Seiten 316–321, 2013.

[26] Ng, Andrew Y.: *Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance.* In: *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, 2004.

[27] Pang, Bo und Lillian Lee: *Opinion Mining and Sentiment Analysis.* Foundations and Trends in Information Retrieval, 2(1-2):1–135, 2008.

[28] Pang, Bo, Lillian Lee und Shivakumar Vaithyanathan: *Thumbs Up?: Sentiment Classification Using Machine Learning Techniques.* In: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, Seiten 79–86. Association for Computational Linguistics, 2002.

[29] Pang, Weihan: *A Machine Learning Approach for Aspect-based Sentiment Analysis on Social Media.* Masterarbeit, TU Dortmund, 2018.

[30] Picard, Rosalind W.: *Affective Computing.* Technischer Bericht 321, M.I.T Media Laboratory Perceptual Computing Section, 1995.

[31] Rozental, Alon, Daniel Fleischer und Zohar Kelrich: *Amobee at IEST 2018: Transfer Learning from Language Models.* In: *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Seiten 43–49. Association for Computational Linguistics, 2018.

[32] Sebastiani, Fabrizio: *Machine Learning in Automated Text Categorization.* ACM Computing Surveys (CSUR), 34(1):1–47, 2002.

[33] Strapparava, Carlo und Rada Mihalcea: *SemEval-2007 Task 14: Affective Text.* In: *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, Seiten 70–74. Association for Computational Linguistics, 2007.

[34] Strapparava, Carlo und Rada Mihalcea: *Learning to Identify Emotions in Text.* In: *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, Seiten 1556–1560, 2008.

[35] Strapparava, Carlo und Alessandro Valitutti: *WordNet-Affect: an affective extension of WordNet.* In: *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, Seiten 1083–1086, 2004.

[36] Wang, Wenbo, Lu Chen, Krishnaprasad Thirunarayan und Amit P. Sheth: *Harnessing Twitter "Big Data" for Automatic Emotion Identification.* In: *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust*, SOCIALCOM-PASSAT '12, Seiten 587–592. IEEE Computer Society, 2012.

[37] Zhang, Xiang, Junbo Zhao und Yann LeCun: *Character-level Convolutional Networks for Text Classification.* In: Cortes, C., N. D. Lawrence, D. D. Lee, M. Sugiyama und R. Garnett (Herausgeber): *Advances in Neural Information Processing Systems 28*, Seiten 649–657. Curran Associates, Inc., 2015.