Institute for Natural Language Processing

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

The Impact of Intensifiers, Diminishers and Negations on Emotion Expressions

Florian Strohm

Course of Study:	Informatik
Examiner:	Dr. rer. nat. Roman Klinger, Prof. Dr. Sebastian Padó
Supervisor:	Dr. rer. nat. Roman Klinger
Commenced:	15. Februar 2017
Completed:	03. August 2017

Abstract

There are several areas of application for emotion detection systems, for example social media analysis, for which it is important to reliably recognize expressed emotions. This thesis takes negations, intensifiers and diminishers on emotion expressions in Tweets into account, in order to study whether this can improve an emotion detection system. It uses different emotion classifiers together with various modifier detection approaches to evaluate the impact of modifiers on emotion expressions. The results show that an emotion detection system can be slightly improved if negations are taken into account. The thesis also studies the correlation between modified emotion words and basic emotions to obtain a better understanding about modified emotions. The analysis of the results shows correlations between modified and basic emotions, which enables us to determine the expressed basic emotion of modified emotion words.

Kurzfassung

Es existieren viele verschiedene Anwendungsgebiete für Systeme die Emotionen erkennen können, beispielsweise die Analyse von sozialen Medien, für die eine verlässliche Erkennung von ausgedrückten Emotionen wichtig ist. In dieser Arbeit werden Wörter, welche ausgedrückte Emotionen negieren, verstärken oder abschwächen anhand von Tweets in Betracht gezogen, um zu untersuchen, ob dies ein System zur Emotionserkennung verbessern kann. Es werden verschiedene Methoden zur Emotionserkennung mit unterschiedlichen Verfahren zur Erkennung solcher Modifikatoren verbunden, um den Einfluss von Modifikationen auf ausgedrückte Emotionen zu evaluieren. Die Ergebnisse zeigen, dass ein System zur Emotionserkennung leicht verbessert werden kann, wenn Negationen mit in Betracht gezogen werden. In dieser Arbeit werden außerdem die Zusammenhänge von modifizierten Emotionen und Basisemotionen untersucht, um ein besseres Verständnis von modifizierten Emotionen zu erhalten. Die Analyse der Ergebnisse zeigt Verbindungen zwischen modifizierten Emotionen und Basisemotionen, was es ermöglicht, die Basisemotion von modifizierten Emotionen zu erkennen.

Contents

1 Introduction 15
  1.1 Motivation . . . 15
  1.2 Goals of this Thesis . . . 18
  1.3 Outline of this Thesis . . . 18

2 Background 21
  2.1 Fundamentals . . . 21
    2.1.1 Classification . . . 21
      2.1.1.1 Support Vector Machine Classifier . . . 22
      2.1.1.2 Cross Validation . . . 24
    2.1.2 Hill Climbing . . . 25
    2.1.3 Evaluation . . . 26
  2.2 Previous Work . . . 28
    2.2.1 Emotion Analysis . . . 28
    2.2.2 Modifier Detection . . . 29

3 Methodology 31
  3.1 Modifier Cue Detection . . . 31
    3.1.1 Modifier Lexicons . . . 31
  3.2 Modifier Scope Detection . . . 33
    3.2.1 Next-n Heuristic . . . 34
    3.2.2 Dependency Tree Relations . . . 34
    3.2.3 Binary SVM . . . 36
  3.3 Emotion Classification . . . 39
    3.3.1 Emotion Lexicon . . . 40
    3.3.2 Multiclass SVM . . . 42

4 Experiments 43
  4.1 Corpora . . . 43
    4.1.1 Automatically Annotated Corpora . . . 45
    4.1.2 Hand-Annotated Corpora . . . 46
  4.2 Experimental Setup . . . 48
  4.3 Results & Discussion . . . 49

5 Summary 63

A Additional Results 65

B Modifier Lexicons 67

Bibliography 69

List of Figures

2.1 Example of the SVM hyperplane creation. 22
2.2 Visualization of a mapping function. 23
2.3 Comparison of two hyperplanes created with and without cross validation. 24
2.4 Example function with a local and a global maximum. 25
2.5 Visualization of TP, TN, FP and FN. 26

3.1 Dependency tree of the sentence “I do not love or hate you.”. 35
3.2 Dependency tree of the Tweet “I put a lot of faith in Bernie’s judgement. ...”. 36

4.1 Annotations for the sentence “I do not love but hate you very much.”. . . 47
4.2 Different values of n for next-n modifier detection. 49

List of Tables

3.1 Lexicon snippets 32
3.2 Investigation of some questionable modifier. 32
3.3 Adversative conjunctions. 34
3.4 Features used for SVM modifier scope detection. 38
3.5 Example Tweet for each basic emotion. 39
3.6 NRC Emotion Lexicon example. 40

4.1 Filter hashtags for each basic emotion. 43
4.2 Emotion classification corpora. 45
4.3 Modifier detection corpora. 47
4.4 Comparison of modifier detection methods. 50
4.5 Support Vector Machine with unigram features using trainAll and testAll corpora. 52
4.6 Support Vector Machine with trigram features using trainEmoMod and testEmoMod corpora. 53
4.7 Word list classifier using trainEmoModEqual and testEmoMod corpora. . 54
4.8 Full versus no modifier detection comparison. 55
4.9 Word list neutral weighting matrix. 57
4.10 Word list negation weighting matrix. 58
4.11 Word list intensifier weighting matrix. 60
4.12 Word list diminisher weighting matrix. 61

A.1 Support Vector Machine with unigram features using trainEmoMod and testEmoMod corpora. 65

B.1 Negation lexicon. 67
B.2 Diminisher lexicon. 67
B.3 Intensifier lexicon. 68

List of Algorithms

3.1 Modifier cue detection using lexicons. 33
3.2 Modifier scope detection using next-n heuristic. 35
3.3 Modifier scope detection using the DepTree approach. 37

1 Introduction

1.1 Motivation

In recent years, the domain of social media has grown rapidly. One of the most popular social networks is the micro blogging platform Twitter. Twitter facilitates users to freely express their emotions, opinions and ideas to anyone by publishing short messages. Those messages, called Tweets, are limited to a maximum of 140 characters, which can require users to truncate their texts and express their thoughts in a compact way. As an example for the rapid growth of social media, we can consider some statistics about Twitter. It took three years after the release in 2006 to reach one billion sent Tweets, while in 2017 it does not even take two days for one billion tweets to be sent. The amount of Tweets sent per day went from 5,000 in 2007 to 500,000,000 in 2013, which is equal to nearly 1,000% gain in yearly volume of Tweets. Although this rapid grow cooled down massively, it is still estimated that the volume of Tweets will grow by 30% every year. [Pro17]

Regarding these statistics, we can conclude that the growth of social media also implies growth in volume of generated data, mainly in the form of text. To be able to analyze these large amounts of data, we need efficient and effective methods to extract information from text. One of the most used methods for data mining is classification, where a certain number of classes Y is defined and then each document gets assigned to a class $y \in Y$. A very common use case is spam filtering in an e-mail system, where incoming e-mails are classified either as $y =$ spam or $y =$ ham.

We apply one of the most common approaches for such classification tasks, called machine learning algorithms, in order to automatically classify a large data set. We need to provide sufficient training data for the machine learning algorithm to work reliably. The training data consists of documents for which we already know the correct classes and thus are able to train a model, which can then be used by our machine learning classifier to predict the classes of unknown data. Creating sufficient training data may be challenging, depending on the complexity of the task and on the cardinality of Y. We will also face the problem of creating training data in this thesis, which will be discussed later in Section 4.1.

Another example for classification is sentiment analysis, which is the task to assign a polarity value $y \in Y = \{\text{positive, negative, neutral}\}$ to a text fragment. Consider the Tweet "I like criticism. It makes you strong! #happiness #PositiveVibes #team"[1]. This Tweet has a positive polarity value because it contains more positive than negative weighted words such as "like" and "happiness". In contrast, the Tweet "That moment when you run into someone you dislike. #life #badnews #hate"[2] has a negative polarity value because it contains negative weighted words like "hate" and "dislike".

Sentiment analysis is a coarse classification and the informaton gained may not be sufficient for certain tasks. Emotion analysis provides more information because it is a more subdivided classification. It is the task to assign a basic emotion to a text fragment. It is not rigidly defined what the basic emotions are. This thesis will follow Ekman's definition of basic emotions [Ekm92] which are "disgust", "sadness", "anger", "joy", "fear" and "surprise". Instead of just assigning positive and negative polarity values, the emotions "joy" and "anger" would be assigned to the previous Tweets.

Emotion analysis can be useful in many different domains. The following list gives a few examples:

- Companies can analyze their Twitter feed to gain information about how their costumers react to announcements.
- Similarly, social media channels can be analyzed after a politician's speech to gain an impression about the medial reactions.
- There are customer service softwares like SmartSense™and NICE Perform™that recognize customer emotions and then perform prioritizing and routing of customers based on emotional content [DA08].
- Emotional information about users can be used for more customized advertising.
- In Human-Computer-Interaction, a computer can interact with a user differently, depending on the users emotional state.
- Text to speech systems can be improved by changing the voice depending on the underlying emotion of the text.
- The information gained using emotion analysis can be valuable in different research domains such as psychology.

[1]https://twitter.com/aislingameenan/status/772464390021013505
[2]https://twitter.com/AhmadSabri2/status/782973840179724288

A reliable classification of emotions is important for all these use cases and thus, researchers try to achieve improvements in this domain. It has been shown that a sentiment analysis system can be improved if intensifiers, diminishers and negations are taken into account. However, this has not yet been studied for emotion analysis. An example for each modifier type is shown in the following list of Tweets:

- Intensifier: "Wishing you a ***very*** happy day! #happiness #positivity"[3]
- Diminisher: "After a ***few*** months I finally get #xboxlive back. #happiness"[4]
- Negation: "I do***n't*** want to lose you but I am ***not*** happy"[5]

Considering the Tweet in the negation example, the expression "not happy" could be interpreted as "sadness", whereas in the Tweet "There is an alarm going off outside my window and I am ***not happy*** about it"[6], "not happy" rather indicates the emotion "anger". In both cases it seems important to consider modifications because otherwise a classifier would probably label both Tweets with the basic emotion "joy". This example also shows that classifying negated emotion words is not trivial and is dependent on its context. Although the phrase "not happy" can have different meanings, it was possible to determine the expressed basic emotion of both Tweets. Having said this, there are Tweets for which it is not easily possible to determine the corresponding basic emotion because of modified emotion words. Some example Tweets where this is the case are given in the following list:

- "I can smell The persons breath behind me on the bus and it smells like he ate a bag of **fucking dog shit and onions** #washyourmouthson..."[7]
  This Tweet could express the emotion "disgust", but due to the intensifier "fucking", it also expresses "anger".
- "I am **not sad** because you don't feel that way. I just feel aggrieved at how fast people change."[8]
  This Tweet could mainly express the emotion "sadness". With the use of the negation "not", it still expresses "sadness" but also "anger" and "surprise" because the second sentence comes more into focus.

[3]https://twitter.com/createtheripple/status/791500891740377089
[4]https://twitter.com/BryanKidder/status/798076624457596928
[5]https://twitter.com/judy_puckett/status/817976949096202240
[6]https://twitter.com/ahundreddoghugs/status/818251050746527748
[7]https://twitter.com/gabemchl/status/806400382721212416
[8]https://twitter.com/nurraihaniman/status/859722763845615616

- "Just a **bit happy** to be back in Ibiza..."[9]
  This Tweet could express the emotion "joy". Since it is diminished by the word "bit", it also expresses "sadness".

In this thesis, we implement and combine an emotion classification and modifier detection system in order to study if this enhances an emotion analysis system and how emotions change when they are modified. We attempt to provide a better understanding of modified emotions and examine whether corresponding non-negated emotions exist. For example, we want to be able to answer the question what the basic emotions of "not happy" and "not sad" are.

1.2 Goals of this Thesis

Considering the motivation of this thesis we can specify two different problems that need to be studied. We will conduct several experiments using different methods for modifier detection and emotion classification in order to achieve these goals.

1. It has been shown that by taking modifiers on sentiments into account, the performance of a sentiment analysis system can be improved. The hypothesis is, that an emotion analysis system also can be improved if intensifiers, diminishers and negations on emotion expressions are taken into account. The main goal of this thesis is to prove or decline this hypothesis.
2. We showed example Tweets containing modified emotion words, for which it is not straightforward to determine the basic emotions. The second goal of this thesis is to understand if specific modified emotions correspond to specific basic emotions.

1.3 Outline of this Thesis

Chapter 2 - Background explains all fundamentals that are used in this thesis. This includes classification, optimization and evaluation techniques. Furthermore, we present previous work on emotion analysis and modifier detection systems.

Chapter 3 - Methodology firstly introduces three different methods for modifier scope detection and then two approaches for emotion classification.

[9]https://twitter.com/bethholland0/status/755018165273391104

Chapter 4 - Results starts with the introduction to our used corpora. It discusses how the corpora are created and for what experiments they are used. Afterwards, the hypotheses are listed and it is explained how they relate to the goals of this thesis. Finally, the chapter shows and discusses the results of the experiments.

Chapter 5 - Summary summarizes the experiments and results of the thesis and shortly discusses possible future work.

2 Background

This chapter is divided into two sections. In Section 2.1, we explain the needed fundamentals for this thesis. Section 2.2 presents previous work in emotion analysis and modifier detection.

2.1 Fundamentals

2.1.1 Classification

Classification is the task to assign a document to one or more predefined classes. Each document is represented as a vector $\vec{x}$ defined as

$$\vec{x} = (v_1, v_2, ..., v_n) \in X, \tag{2.1}$$

where X is the set of all document vectors, v_i an explicit feature and n the number of features. Features are information extracted from a document, for example, the information whether specific words are present or absent in a given document. We further define Y as the set of all classes and $|Y| = k$. The goal of a classification task is to find the most accurate function

$$f : X \to Y : \vec{x} \mapsto f(\vec{x}). \tag{2.2}$$

We differentiate between two kinds of classification. Binary classification considers only two possible classes ($k = 2$) like in the spam filter example, whereas multiclass classification considers more than two classes. In this thesis, multiclass classification is used with $k = 6$, defining each class as one basic emotion. Our input data splits into training and test data. For the classifier to be able to find a good function f, it is necessary to provide sufficient training data. The training data is a set of documents whose correct classes are already known. This data can be used to train a model for the classifier, which is then used to classify the test data. The training data S is defined as

$$S = ((x_1, y_1), ..., (x_l, y_l)) \subseteq (X \times Y)^l, \tag{2.3}$$

with l being the amount of training data. The x_i are called instances (our training documents) and the y_i are the results (classes) of the corresponding x_i.

2.1.1.1 Support Vector Machine Classifier

The Support Vector Machine (SVM) is a very popular technique for data classification. The SVM creates a hyperplane that separates our document vectors $\vec{x_i}$ in the training data according to their class y_i. We define a hyperplane as a pair (w, b) with w being the orthogonal vector indicating the inclination of a line and b being the parallel shift distance from the origin. We further define the distance γ_i from one training document $s_i \in S$ to a hyperplane as

$$\gamma_i = y_i(\langle w \cdot x_i \rangle + b). \tag{2.4}$$

The SVM now optimizes the parameters in order to find the hyperplane that separates the training documents s_i according to their classes, while retaining the maximum margin to each s_i. This optimal hyperplane is called "maximum margin hyperplane". Figure 2.1 shows a hyperplane that separates the two classes "circle" and "square" perfectly while retaining the maximum margin. The dotted lines indicate the support vectors that are used to separate each class using the smallest margin to one class and the largest to the other. The hyperplane with the same distance to each support vector is finally chosen and can then be used to classify a test set.

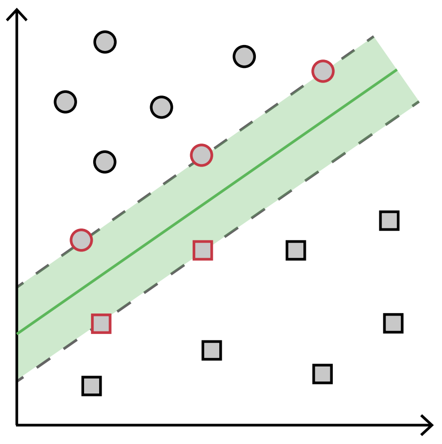

Figure 2.1: Example of the SVM hyperplane creation.

Not all classification problems are perfectly linearly separable so we introduce a penalty parameter C, which indicates to what extend we want to avoid misclassifying training examples. For large values of C, the SVM will create a hyperplane that separates the training vectors better, with the drawback of having smaller margins. On the other hand, for small values of C, it creates a hyperplane with a larger margin but will misclassify more training examples.

In order to increase the accuracy of the optimization, we map the training data x_i in higher dimensional spaces using the function

$$\phi : X \subseteq \mathbb{R}^n \mapsto F \subseteq \mathbb{R}^N, \tag{2.5}$$

because then it may be linearly separable more easily. Figure 2.2 shows an example where the training data becomes linearly separable after the mapping. The left side shows linear inseparable elements of class “circle” and “square” in the domain X. After the elements are mapped into the domain F by the function ϕ, a perfect separating hyperplane can be found.

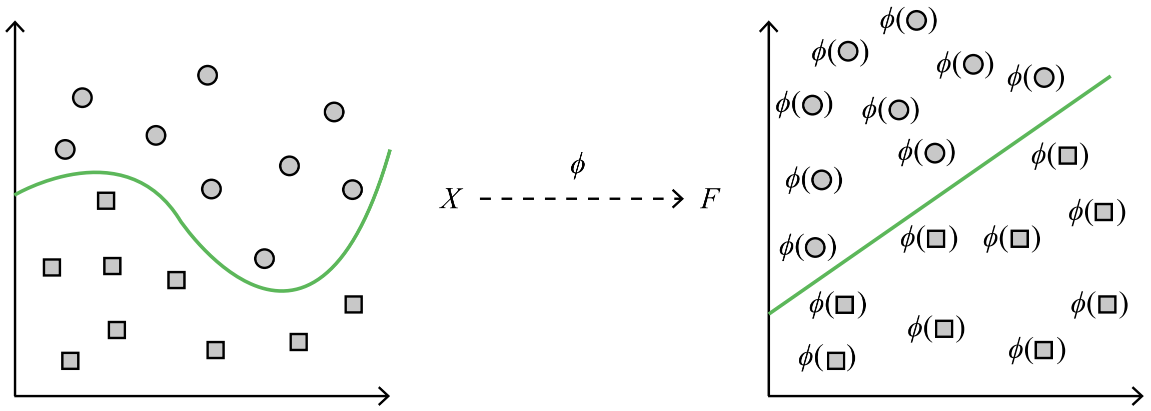

Figure 2.2: Visualization of a mapping function.

Furthermore, the so called kernel trick is used in order to avoid this explicit mapping of the training data. For this, we define the kernel function K as

$$K(x_i, x_j) = \langle \phi(x_i) \cdot \phi(x_j) \rangle. \tag{2.6}$$

Several different kernel functions exist and one has to find the optimal kernel function for a given classification problem. Three often used kernels are defined in the following list.

- Linear kernel: $K(x_i, x_j) = x_i^T x_j$.
- Polynomial kernel: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$.
- Radial basis function (RBF) kernel: $K(x_i, x_j) = exp(-\gamma \|x_i^T x_j\|^2), \gamma > 0)$

After a kernel is selected, we have to find the best values for C and, if necessary, kernel parameters (e.g. γ and r in the polynomial kernel). We describe the approach of finding the best values exemplary for the parameter C. A naive approach is to try different values of C and compare their performance. As suggested by Hsu et al. [CL08], we first run a coarse “grid-search” using different powers of two ($2^{-15} - 2^{15}$, step size one) to minimize the search space and then perform a finer search around the best performing area. To evaluate the performance for each value of C, we have to split our training data into training and validation data. We use the smaller training data to train a model with a specific value of C and then test and evaluate the performance on the validation data. If we use a static split of our training data, a problem called “overfitting” may occur. Overfitting is a phenomenon, meaning our trained model is too adjusted to the training data so that the model performs worse on other data.

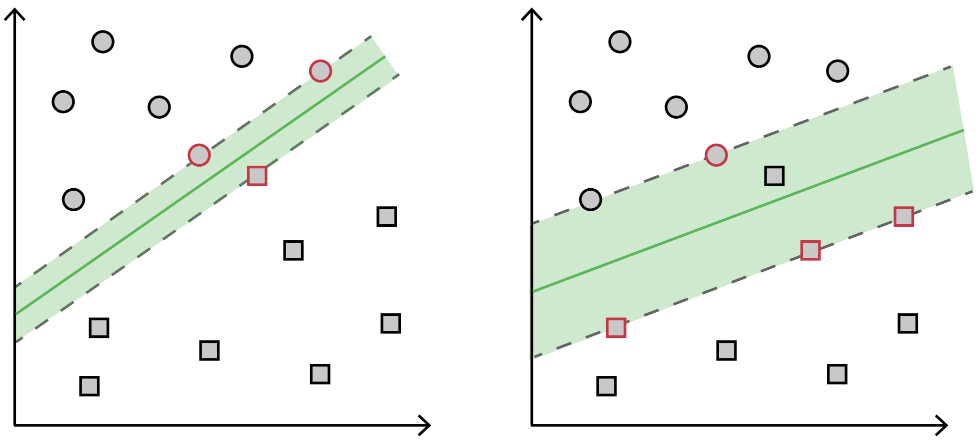

Figure 2.3: Comparison of two hyperplanes created with and without cross validation.

2.1.1.2 Cross Validation

To reduce overfitting, we use a technique named cross validation. The most common variant is called n-fold cross validation, where the training data is not split statically into a smaller training and validation data, but into n subsets $s_i \subset S$ of about equal size. Afterwards we train a model on $n-1$ subsets and use the remaining one as the validation set. This procedure is repeated n times, so that each subset $s_i \in \{s_1, s_2, ..., s_n\} = S$ is used as a validation set while $S \setminus s_i$ is used to train a model. After the procedure is finished we have n results, which we average and use this result as a measurement for a specific value of C. In Figure 2.3 we can see the comparison of an overfitted hyperplane (left) and a more general hyperplane (right). As the figure shows, the hyperplane on the right side missclassifies one s_i but has a much larger margin than the hyperplane on the left and will therefore probably perform better on test data. The use of cross validation helps to find values for the kernel parameters that lead to a more general hyperplane and therefore reduces overfitting.

2.1.2 Hill Climbing

Hill climbing is an optimization technique that maximizes/minimizes a target function $f(x)$, with x being a vector of discrete and/or continuous values. The algorithm iteratively searches the neighborhood of $f(x)$ by adjusting one element in the vector and tries to find a better $f(x')$ to start searching from there again. The algorithm stops if no better $f(x')$ can be found in the neighborhood of $f(x)$. There are three main variants of the hill climbing algorithm. The "simple hill climbing" chooses the first better successor whereas the "steepest ascent hill climbing" checks all successors and chooses the best one to proceed. The third variant is called "stochastic hill climbing", which chooses a random successor from all better successors. The probability of a selection varies with the steepness of the successors.
All hill climbing variants have the problem that they may only find a local maximum/minimum if the function f is not convex. Figure 2.4 shows an example function with a local and a global maximum. Depending on the start seed of the algorithm, it may reach the local maximum during the optimization process. It can not find any better $f(x')$ in the neighborhood of a maximum and therefore returns this as the optimal solution, but it is not since there is a better solution, the global maximum.

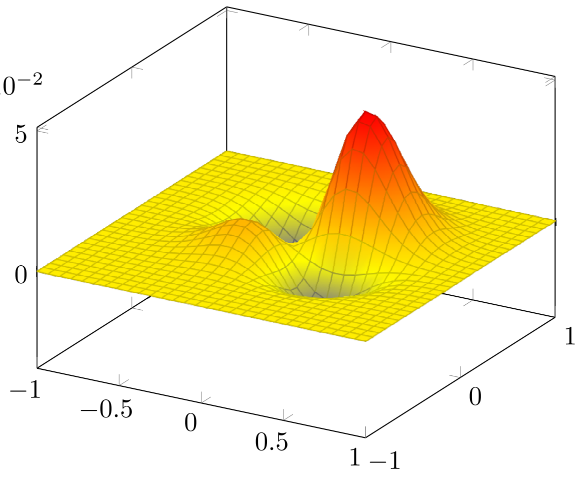

Figure 2.4: Example function with a local and a global maximum.

To increase the probability of finding the global maximum/minimum, we use a meta-algorithm on top of hill climbing, called random-restart hill climbing. The hill climbing algorithm is performed several times using different, random generated seeds for x. The current best performing x is kept until a better performing x' is found. Applying this meta-algorithm to the previous example function probably results in the optimal solution after a few iterations.

2.1.3 Evaluation

In order to compare different emotion and modifier detection methods we have to introduce evaluation techniques, which measure the performance of each approach. After a classifier predicted the class of a document there are four possible cases, which we count separably for each class y_i:

- True Positive (TP): the gold standard and the predicted class are both y_i.
- True Negative (TN): the gold standard and the predicted class are both not y_i.
- False Positive (FP): the gold standard is not y_i but the predicted one is.
- False Negative (FN): the gold standard is y_i but the predicted one is not.

See also Figure 2.5 for a visual representation of the four cases. The left half of the square contains documents of class y_i (squares) and the right side of class y_j (circles). Documents inside the large circle are classified as y_i. Green colored areas indicate that the classifier predicted these documents correctly, whereas the opposite applies for the red areas.

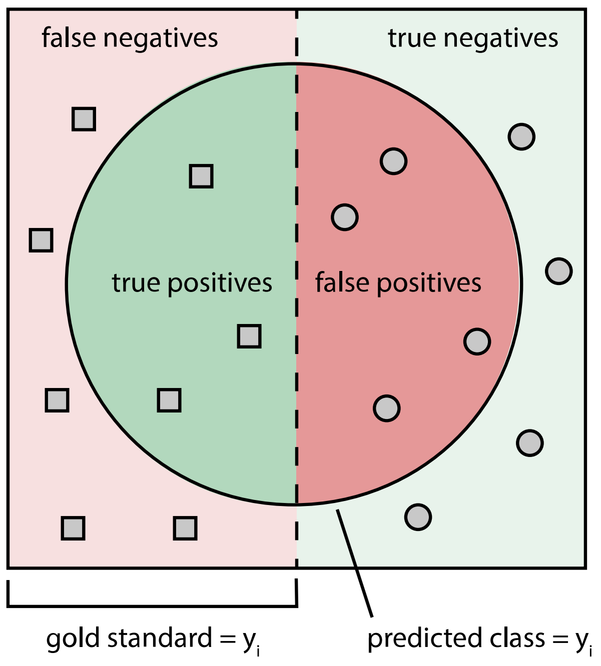

Figure 2.5: Visualization of TP, TN, FP and FN.

Using these four measured values, we can calculate the precision defined as

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i} = \frac{\text{documents correctly classified as } y_i}{\text{documents classified as } y_i} \tag{2.7}$$

and the recall defined as

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i} = \frac{\text{documents correctly classified as } y_i}{\text{documents with gold standard } y_i} \tag{2.8}$$

for all classes $y_i \in Y$. Precision or recall alone are no good indicators of how well our classifier performs, as we can easily achieve 100% precision for a class y_i if we classify all documents as y_j and 100% recall if we classify all documents as y_i. Thus, we combine these two measurements using the harmonic mean, called F-measure, which is defined as

$$F_{\beta i} = (1 + \beta^2) \cdot \frac{\text{precision}_i \cdot \text{recall}_i}{\beta^2 \cdot \text{precision}_i + \text{recall}_i} \tag{2.9}$$

with β indicating the weighting for recall$_i$. Usually we want a balanced F-measure, where recall and precision are considered equally important and therefore use $\beta = 1$.

If we are using a multiclass classifier, we generate multiple F_1-measurements. In order to compare the overall performance of a classifier we need to calculate an average F_1-measure. There are two methods of calculating the average F_1-measure; one is called micro-average defined as

$$\text{Micro-average} = \frac{2 \cdot \text{precision}_{sum} \cdot \text{recall}_{sum}}{\text{precision}_{sum} + \text{recall}_{sum}} \tag{2.10}$$

with precision$_{sum}$ and recall$_{sum}$ being calculated using

$$\text{TP} = \textstyle\sum_{i=1}^{k} \text{TP}_i$$

$$\text{FP} = \textstyle\sum_{i=1}^{k} \text{FP}_i$$

$$\text{FN} = \textstyle\sum_{i=1}^{k} \text{FN}_i$$

and the other is called macro-average, defined as

$$\text{Macro-average} = \frac{1}{k} \sum_{i=1}^{k} F_i \tag{2.11}$$

The difference between these two approaches is that the micro-average method weights the different F_1-measurements according to the amount of documents the corresponding class contains, while the macro-average method treats each class equally. We use the macro-average method in this thesis, since we want to weight each basic emotion equally.

2.2 Previous Work

2.2.1 Emotion Analysis

For emotion analysis it is necessary to define basic emotions which can be used for classification. The most common definitions were proposed by Paul Ekman [Ekm92] and Robert Plutchik [Plu01]. Ekman defines “disgust”, “sadness”, “anger”, “happiness”, “fear” and “surprise” as the six basic emotions that all humans have in common. Plutchik instead, describes in his work eight different basic emotions, including the six defined by Ekman and additionally “anticipation” and “trust”. This eight basic emotions are grouped into four bipolar pairs: “joy” vs. “sadness”, “anger” vs. “fear”, “trust” vs “disgust” and “surprise” vs. “anticipation”. Many researchers use Ekmans defined basic emotions and therefore, we also use them in this thesis because this enables us to compare our results with previous work.

After the basic emotions are defined, the next step will be to develop a method to classify the text fragments. The classification by Danisman and Alpkocak [DA08] is based on a vector spaced model (VSM). They create a vector for each emotion set and for the to be classified data. This data vector is then assigned to the emotion with the highest cosine similarity. Their results show that VSM classification on short sentences achieve similar results to naive Bayes or SVM classifier. They also studied the influence of stemming on the emotion classification task and show that stemming results in a slightly better performance.

Balabantaray et al. [BMS12] used a Tweet based corpus for emotion classification, that was annotated by five different judges with an average annotator agreement of 71%. They used a SVM classifier and compared the performance of many different features. Some of their used features include: Unigrams, Bigrams, POS, POS-Bigrams, Emoticons and more. The results show that the highest emotion classification accuracy is achieved if all features are used.

Suttles and Ide [SI13] use the eight bipolar basic emotions defined by Plutchik for emotion classification. This enables them to use four binary classifiers instead of a multiclass classifier. They combine them to emulate a multiclass classifier and achieve higher accuracies than in previous reported multiclass classification studies.

Most research focuses on explicit emotion detection like in the sentence “The outcome of my exam makes me happy” whereas Udochukwu and He [UH15] created an emotion analysis system based on different rules to detect implicit emotions like in the sentence “I passed my exam” as well. Both of this sentences express the emotion “joy”, but the second sentence does not use any emotion bearing words and therefore only implying the emotion.

Many of the best performing approaches for emotion classification use machine learning classifier. The problem is that a very large amount of training data is needed for emotion detection, since the classifier has to learn to distinguish between many different classes. Manual labeling of such large data is very time consuming and therefore, researchers use methods to automatically label a large amount of data. Wang et al. [WCTS12] used the hashtags from Tweets to automatically label them with their expressed basic emotion. For example, in a Tweet like “I hate when my mom compares me to my friends. #annoying...”[1], they use “#annoying” to label the Tweet as “anger”. Using this method, they are able to create a training set containing millions of Tweets without any need of manual labeling. They studied some randomly sampled Tweets and developed a set of filtering heuristics to remove irrelevant Tweets and reduce the noisiness of the corpus. Their results show, that the performance of an emotion classification system using SVMs can be significantly improved for a very large amount of training data. By increasing the amount of training data from 1.000 to 2 million Tweets, the accuracy of the classification increased by about 22%.

Purver and Battersby [PB12] used a similar approach to create a large corpus of automatically annotated Tweets. They also used certain hashtags to automatically label Tweets, but studied a different approach using emoticons as well. For example, the emoticon “:)” was used to label Tweets as “joy”. They compared the emotion classification performance of these two approaches. The results show, that both approaches achieve similar results.

2.2.2 Modifier Detection

The first modifier detection methods were used to detect findings and diseases in biomedical texts more reliably. The most commonly used approach for modifier detection is firstly to detect modifier cues and then secondly to detect the scope of the cues. Chapman et al. [CBH+01] introduced the “NegEx” algorithm based on regular expressions. The algorithm searches biomedical texts for phrases indicating negations (cues) and then determines the scope of this phrases. The scope of a negating phrase is set to the next punctuation mark or to the next adversative conjunction.

In recent times, the area of application for modifier detection shifted towards sentiment analysis. At least for negations, it seems intuitive that they can affect the polarity of a sentiment and shift it from positive to negative and vise versa. The role of negations on sentiments is dependent on how fine-grained the classification task is. If the task is to classify whole documents, Pang et al. [PLV02] showed that by additionally taking

[1]https://twitter.com/_xCalifornian/status/135820316063703040

negations into account the improvement is negligible. If a document contains twenty phrases with sentiments and only two of them are negated, then those few negated sentiments hardly affect the results.

Councill et al. [CMV10] use a lexicon based approach for negation cue detection. They trained a conditional random field (CRF) using multiple features, for example, part of speech and dependency relations. They achieve a F_1 score of 80% for the negation scope detection on a product reviews corpus. Then, they implemented this negation detection approach into a sentiment classification system. The results show a significant improvement of the sentiment classification performance using negation detection.

Reitan et al. [RFGB15] use the same approach as Councill et al. [CMV10], but with a few additions. Instead of a product review corpus, they use a Tweet corpus for training and testing. Therefore, they added some Twitter specific modifier cues to the lexicon like “dnt” or “cudnt”. The results for the sentiment classification task show an improvement if the modifiers are taken into account.

Jia et al. [JYM09] use a parse tree, typed dependencies and special rules to determine the scope of a negation cue. Firstly, they create a candidate scope using the parse tree and a simple heuristic. They determine the lowest common ancestor (LCA) of the modifier cue and the word immediately after the cue. The candidate scope consists of all descendant leaf nodes of the LCA. Afterwards, they apply multiple heuristic rules on the candidate scope to determine the final modifier scope. Again, the results show an improvement for sentiment classification if modifiers are taken into account.

It is easy to implement a modifier detection system into a machine learning classifier, as the features can simply be changed if they are within a modifier scope. For others, as for example a word list classifier, modifier scopes can not easily be considered. In the work of Polanyi and Zaenen [PZ06], they present a method to resolve this problem in a sentiment classification task. To classify a document as either positive or negative, they count the number of positive and negative words occurring in a document. If there are more positive words in a document, it is classified as positive and vise versa. Now to consider modifiers, they do not count the words, but sum up weightings for each word. Positive words have a weighting of +2, while negative words have a weighting of -2. If a word is negated, its sign is simply shifted. If a word is intensified, its weighting changes to +3 or -3 respectively and if its diminished, it changes to +1 or -1 respectively. If the final weighting is > 0, the sentence is classified as postive, and if it is < 0 as negative sentiment. Kennedy and Inkpen [KI06] conducted experiments using this approach and showed, that if negations, intensifiers and diminishers are taken into account this way, a sentiment classification system can be improved.

3 Methodology

This chapter is divided into three sections. Section 3.1 describes the modifier cue detection task and how it is done in this thesis using modifier lexicons. Section 3.2 explains the modifier scope detection task and presents three different approaches, which we will compare in our experiments. Finally, Section 3.3 shows two different emotion classification methods.

3.1 Modifier Cue Detection

Modifier detection splits into modifier cue and scope detection (MSD). Before the scope of a modifier can be defined, the modifier token in a sentence has to be identified. We differentiate between an explicit modification as in "I do not like this movie.", where the explicit negation cue "not" can be identified, and an implicit modification as in the sentence "I like this movie as much as I like coffee, and I really hate coffee.". Both examples tell us that the person does not like the movie, but the implicit negation in the second sentence is much more difficult to detect than the explicit negation in the first one. We limit our work to explicit modifier cues, enabling us to use modifier lexicons for a simple modifier cue detection.

3.1.1 Modifier Lexicons

Firstly, we create a lexicon for each modifier type containing explicit modifier cues collected from different paper and websites[1][2] [TM86] [Rom12] [BCP+07] [CMV10]. Table 3.1 shows a snipped of each modifier lexicon (see complete lists in Appendix B).

However, some modifiers in these lists are questionable for our purpose as they can also be used in a non modifying context. For example the word "so" can be used as an adverb of degree and is therefore an intensifier cue like in the Tweet "**So** cute that Alex

[1]https://en.wikipedia.org/wiki/Intensifier
[2]https://www.englishclub.com/vocabulary/adverbs-degree.htm

Negation cues		Intensifier cues		Diminisher cues	
cannot	neither	absolutely	amazingly	barely	bit
never	no	completely	entirely	few	hardly
nobody	none	especially	extremely	less	nearly
nor	not	freaking	incredibly	negligibly	partly
nothing	nowhere	lot	really	practically	rarely
n‘t	without	super	very	some	sparsely

Table 3.1: Lexicon snippets

got me chocolates for national girlfriend day #lucky”[3], but also can be used for other purposes, for example as an interjection like in the Tweet “**So** this is what rock bottom looks like. #depress #sad #life”[4]. To analyze this problem we collect 100 random Tweets for each questionable modifier and calculate the probability of how often it is used in a non modifying context. Table 3.2 shows the results for some modifiers. The first column shows the modifier cue, the second column its type and the third column the percentage this cue is not used as modifier. If the percentage of a modifier is higher

Modifier Cue	Modifier Type	not-a-modifier percentage
awful	intensifier	**86%**
dead	intensifier	**95%**
holy	intensifier	**59%**
mad	intensifier	**96%**
much	intensifier	20%
real	intensifier	49%
really	intensifier	33%
so	intensifier	22%
too	intensifier	50%
lack	negation	12%
bit	diminisher	20%
little	diminisher	50%

Table 3.2: Investigation of some questionable modifier.

than a threshold of 50%, we delete it from the modifier cue lexicon, because this means

[3]https://twitter.com/skye_lilly_/status/760451487805890560
[4]https://twitter.com/ferliloo/status/783565601021104128

it is predominantly not used as a modifier. According to Table 3.2, we delete the words “awful”, “dead”, “holy” and “mad” from the modifier lexicons.

Once the lexicons are created, the modifier cue detection can begin as shown in Algorithm 3.1. We define a modifier cue of type k as C_k. For each word of a sentence we check if any lexicon l_k contains this particular word and if so, it is flagged as a modifier cue C_k. Considering the sentence “I am **not** happy”, the word “not” is flagged as a negation cue C_{neg}, since it is included in the negation lexicon.

Algorithm 3.1 Modifier cue detection using lexicons.

```
1:  procedure CUEDETECTION(sentence, modifierLexicons)
2:      for each word in sentence do
3:          if negationLexicon.contains(word) then
4:              word.isNegator(true)
5:          else if intensifierLexicon.contains(word) then
6:              word.isIntensifier(true)
7:          else if diminisherLexicon.contains(word) then
8:              word.isDiminisher(true)
9:          end if
10:     end for
11: end procedure
```

3.2 Modifier Scope Detection

After all modifier cues are annotated, the modifier scope of each cue has to be determined. The scope of a modifier cue tells which words are modified by this particular cue. Each word in the scope of a modifier cue C_k is annotated by adding a prefix. Depending on the modifier type we add the prefix “NOT_” for a negating scope, “INT_” for an intensifying scope or “DIM_” for a diminishing scope. Looking at the previous example “I am not happy”, “NOT_” is added to the word “happy”, because it is within the negation scope of “not”, creating the new word “NOT_happy”. We do not modify a token twice, for example, “NOT_INT_happy” because we would generate very specific features that may only occur once in a corpora. The modifier cues itself are still used as features since [PLV02] stated that removing negation cues after their scopes are determined has a slightly harmful effect on the performance. We assume that a similar behavior occurs for intensifier and diminisher cues.

As discussed in Section 2.2.2, several different approaches for modifier scope detection exist. We will introduce three approaches for modifier scope detection in the next sections.

3.2.1 Next-n Heuristic

The next-n approach is based on the technique introduced by Pang et al. [PLV02], where they set the modifier scope from the cue to the next punctuation mark. For example, in the Tweet “Happiness is **not** a goal; it is a by-product.”[5] the words “a” and “goal” would be negated, but not the words following the semicolon, creating the new sentence “Happiness is not NOT_a NOT_goal; it is a by-product.”.

We combine this technique with the NegEx algorithm from Chapman et al. [CBH+01]. In their algorithm, the scope does not only end after a punctuation mark but also after adversative conjunctions. Considering the sentence “I do **not** love **but** hate you.”, we do not negate every word after the negation cue “not”, which would be wrong since “hate” is not negated, but only the word “love”. Table 3.3 shows the full list of used adversative conjunctions according to Chapman et al. [CBH+01].

but	however	nevertheless
yet	though	although
still	except	

Table 3.3: Adversative conjunctions.

Our approach is further inspired by Hue et al. [HL04], where they use a parameter n that indicates how many words after a negation cue are negated. Regarding the first example “Happiness is not a goal; it is a by-product.”, the negation scope is identical for values $n > 1$, but changes for $n = 1$ as then the word “goal” is no longer within the scope. The resulting method is shown in Algorithm 3.2.

3.2.2 Dependency Tree Relations

The following approach is inspired by the approach of Jia et al. [JYM09]. They use a dependency parser to retrieve the dependency tree of a sentence and then use this tree to extract a candidate negation scope. Afterwards, they apply several rules on the candidate scope to retrieve the final scope. We will also use the dependency tree of a sentence but apply a different technique for modifier scope detection. We do not need to determine the full scope of a cue and can therefore use a more simple approach, since we consider in particular emotional words and their modification, which will be motivated in Section 4.1.

[5]https://twitter.com/littleobsessian/status/864158607658582016

Algorithm 3.2 Modifier scope detection using next-n heuristic.

```
procedure NEXTN(sentence, $n$)
    modifierType $\leftarrow 0$
    modifyNextN $\leftarrow 0$
    for each word in sentence do
        if modifyNextN $> 0$ then
            if word.isAdversativeConjunction() or word.isPunctuation() then
                modifierType $\leftarrow 0$
                modifyNextN $\leftarrow 0$
            else
                word.setModified(modifierType)
                modifyNextN $\leftarrow$ modifyNextN $- 1$
            end if
        end if
        if word.isModifier then
            modifierType $\leftarrow$ word.getModifierType()
            modifyNextN $\leftarrow n$
        end if
    end for
end procedure
```

The basic idea of the scope detection algorithm is to check for each node in the dependency tree if any first order child is flagged as a modifier cue and if so, the current considered node is modified. Figure 3.1 shows a dependency tree for the sentence “I do not love or hate you.”, created by the Stanford CoreNLP library [MSB+14].

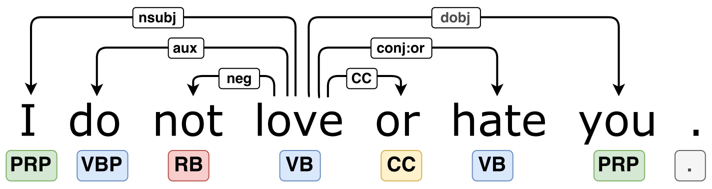

Figure 3.1: Dependency tree of the sentence “I do not love or hate you.”.

The node of the negation cue “not” is the first order child of the node “love” and thus, “love” will be negated. However, Figure 3.1 also shows the first problem where this approach fails. The node corresponding to “love” is not the only node within the negation scope, but also the node related to “hate”. We can see that the node “love” is connected to the node “hate” with a conjunction edge (“conj:or”). We use this information to improve our algorithm by additionally modifying nodes that are connected to already modified nodes via a conjunction edge. By doing so, the algorithm modifies the two emotional words in this example correctly but leads to misclassifications in other cases. If we change the previous example to “I do not love **but** hate you.”, the Stanford parser creates the same dependency tree as before, leading to a similar modifier scope. However, this time the scope is incorrect since “hate” is not negated because of the adversative conjunction “but”. To resolve this problem, we do not modify words that are connected with conjunctions from the Table 3.3.

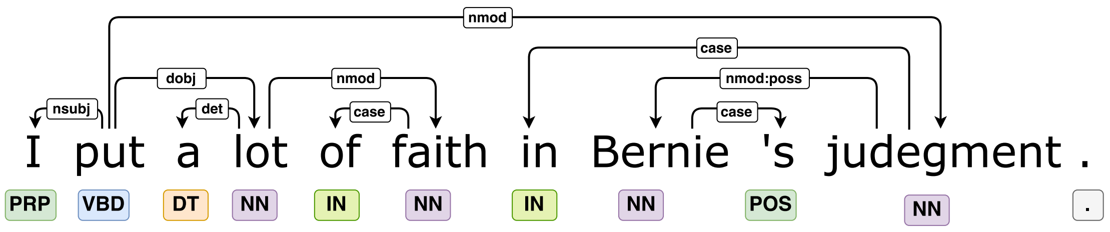

Figure 3.2: Dependency tree of the Tweet “I put a lot of faith in Bernie’s judgement. ...”.

Another case where this approach fails can be seen in Figure 3.2. It shows the dependency tree of the Tweet “I put a **lot** of **faith** in Bernie’s judgement. ...”[6]. We can identify that the word “faith” is intensified by the cue “lot”. The problem is that “lot” is not a child of “faith” but the opposite is the case. If we modify a word with a modifier as a parent as well, we would create more new false positives than we would eliminate and therefore do not consider this case. The resulting approach is shown in Algorithm 3.3

3.2.3 Binary SVM

The shared task at *SEM 2012 [MB12] focused on negation scope detection. Many of the best performing submissions use supervised machine learning approaches, which is the reason why we also introduce a machine learing approach for scope detection using SVMs. For our purposes we use the popular SVM library called LIBLINEAR [FCH+08] provided by Fan et al..

For each modifier type we train a binary SVM that decides if a token is modified or not. Hence, the SVMs do not detect an explicit modifier scope since we do not know by which cue a token is modified if there are multiple cues. As discussed previously in Section 3.2, we do not modify tokens twice and thus, we have to use a modifier policy. We expect negations to have the highest impact on the performance, so we first use the SVM for negation scope detection and mark negated tokens. Then the SVM for intensifier scope detection is used but ignores tokens that are already negated. Finally, the SVM for

[6]https://twitter.com/Highcentered/status/766094329416855552

Algorithm 3.3 Modifier scope detection using the DepTree approach.

```
procedure DEPTREE(sentence)
    dependencyTree ← createDepTree(sentence)
    for each node in dependencyTree do
        children ← getChildren(node)
        for each child in children do
            if child.isModifier then
                modifierType ← child.getModifierType()
                node.setModified(modifierType)
            end if
        end for
        if node.isModified then
            outEdges ← getOutgoingEdges(node)
            for each edge in outEdges do
                if edge.isNonAdversativeConjunction() then
                    modifierType ← node.getModifierType()
                    node ← getDestinationNode(edge)
                    node.setModified(modifierType)
                end if
            end for
        end if
    end for
end procedure
```

diminisher scope detection is applied ignoring negated and intensified tokens. Using this policy prevents multiple modifications of a token taken into consideration.

Councill et al. [CMV10] apply a supervised machine learning approach for negation scope detection using Conditional Random Fields (CRFs). We adopt the features they use and in addition, the feature called "Dep Dist.". See Table 3.4 for the full list of used features and their description. As Table 3.4 shows, we need the part of speech (POS) annotations of each token and the dependency tree of the corresponding sentence. For this, we again use the Stanford CoreNLP library [MSB+14]. To demonstrate the features we give a small example in the following list for the word "hate" using the sentence "I do not love or hate you." and its dependency tree seen in Figure 3.1 again.

- Word = hate
- POS = VB (verb)
- Right Dist. = 0 (no modifier cue to the right)

Feature	Description
Word	Normalized string of a token.
POS	Part of speech of a token.
Right Dist.	Token distance to the nearest explicit modifier cue in the sentence to the right of a token.
Left Dist.	Token distance to the nearest explicit modifier cue in the sentence to the left of a token.
Dep Dist.	Minimum number of edges that must be traversed in the dependency tree from a token to an explicit modifier cue.
Dep1 POS	Part of speech of the the first order parent of a token.
Dep1 Dist.	Minimum number of edges that must be traversed in the dependency tree from the first order parent of a token to an explicit modifier cue.
Dep2 POS	Part of speech of the second order parent of a token.
Dep2 Dist.	Minimum number of edges that must be traversed in the dependency tree from the second order parent of a token to an explicit modifier cue.

Table 3.4: Features used for SVM modifier scope detection.

- Left Dist. = 3
- Dep Dist. = 0 (is leaf node)
- Dep1 POS = VB (verb)
- Dep1 Dist. = 1
- Dep2 POS = null (first order parent is already root node)
- Dep2 Dist. = 0

We use the linear kernel function because it is the fastest kernel [CL08], which is favorable since we are using a very large amount of Tweets in our experiments. As the linear kernel contains no further kernel parameters that have to be optimized, we only need to find the best performing values C for each SVM. In order to find the best values for C, we perform a grid search with 10-fold cross-validation on the training data for each of the three SVMs.

3.3 Emotion Classification

The main task of our program is to determine the basic emotion of an input Tweet. We already specified that we are using the basic emotions defined by Ekman and therefore define the classes as $Y = \{\text{joy, anger, fear, sadness, surprise, disgust}\}$. Table 3.5 shows an example Tweet for each of our defined basic emotion. The program should be capable of distinguishing between these basic emotions and labeling each Tweet as one of them. In the following sections we present two different classifiers, which we use in the experiments.

Emotion	Example Tweet
Joy	“Wishing you a very happy day! #happiness #positivity”[7]
Anger	“That moment when you run into someone you dislike. #life #badnews #hate”
Fear	“I am going to present myself as a woman for the 1st time at a friend’s Halloween party. #scaredtodeath #trans #LGBT”[8]
Sadness	“Bored of living this live, all I just see around me is just #sadness :(”[9]
Surprise	“My sisters boyfriend walked in and I’m sitting in the kitchen eating a piece of cheese with no pants on #surprise”[10]
Disgust	“Walking in from last night smelling like a hookah bar #disgust”[11]

Table 3.5: Example Tweet for each basic emotion.

3.3.1 Emotion Lexicon

The first emotion classification approach uses an emotion lexicon that contains emotion bearing words together with an annotation that indicates which basic emotions they are corresponding to. In this thesis, we choose the NRC emotion lexicon by Mohammad and Turney [MT13]. Table 3.6 shows an example annotation for the emotion word “happy”. Each emotion bearing word is annotated with each of the eight basic emotions defined by Plutchik (including Ekmans six basic emotions) and positive/negative sentiment. The first column in the table shows the emotion word, the second column the basic emotion/sentiment and the third column indicates if the emotion word correlates to the basic emotion/sentiment.

Word	Emotion/Sentiment	Correlation
happy	anger	0
happy	anticipation	1
happy	disgust	0
happy	fear	0
happy	joy	1
happy	negative	0
happy	positive	1
happy	sadness	0
happy	surprise	0
happy	trust	1

Table 3.6: NRC Emotion Lexicon example.

The classifier iterates over an input Tweet and then checks for each word if the lexicon contains it. If the lexicon contains a word of the Tweet, the classifier checks if it correlates to a basic emotion. For each correlating basic emotion the algorithm adds up weightings $\{w_1, ..., w_6\} = \vec{w}$. Each of these weightings reflects one basic emotion. After all words in a Tweet are processed this way, the classifier classifies the Tweet as the basic emotion reflected by the highest weighting w_i. The different weightings for each basic emotion are stored in a 6×6 matrix where the nth row and column corresponds to one basic emotion. For example, if a Tweet contains the word “happy”, Table 3.6 shows that

[7]https://twitter.com/createtheripple/status/791500891740377089
[8]https://twitter.com/TransCartoonist/status/792503229326254080
[9]https://twitter.com/netman007/status/747860322464239616
[10]https://twitter.com/Kelsey_Storlien/status/783077613598347264
[11]https://twitter.com/kimberly_faaaye/status/762330001119117312

this correlates to the basic emotion “joy”. The classifier now looks up the row in the weighting matrix corresponding to “joy” and adds the six values in this row to the weightings $\vec{w}$.

More formal, the emotion of the to be classified Tweet is determined by the index i of the highest weighting in $\vec{w}$ defined as

$$\underset{i=1,\ldots,6}{\arg\max}(w_i), \tag{3.1}$$

with $\vec{w}$ being defined as

$$\vec{w} = \sum_{j=0}^{n} M^T \times \vec{b_j}. \tag{3.2}$$

M^T is the transposed weighting matrix and n the number of words in the Tweet. The vector $\vec{b_j}$ is a boolean vector that represents the correlations between the jth word and the basic emotions according to the emotion lexicon.

To consider modifiers we use a method inspired by Polanyi and Zaenen [PZ06]. In their work, they change the weighting of a word depending on whether it is modified or not. For this, we use three additional weighting matrices, one for each modifier type. After the classifier has checked if the emotion lexicon contains a word from an input Tweet, it also checks if this word is flagged as intensified, diminished or negated and then chooses the corresponding weighting matrix. The matrix which is used for non flagged words is called “neutral weighting matrix” and the other three are called “negation”, “intensifier” and “diminisher weighting matrix”. For example, if the word “happy” occurs and it is flagged as negated, the classifier adds the values to $\vec{w}$ that are stored in the negation weighting matrix in the row corresponding to “joy”.

Before we can classify Tweets using this approach, the classifier has to find the optimal values for each weighting matrix. For this, we apply the meta-algorithm “random-restart hill climbing” as discussed in Section 2.1.2 to each of the four weighting matrices. The function f takes the 36 cells of a matrix as an input and returns a F_1 score. The algorithm optimizes the value for each cell so that the resulting F_1 score is maximized. The function f is realized by the word list classifier, which uses the current to be optimized matrix to classify a training set and returns the F_1 score of this classification. The meta-algorithm stops after m iterations, giving us each m neutral, negation, intensifier and diminisher weighting matrices, which were all maximized in F_1 score and started at different seeds. Unlike normally, we do not proceed with the best performing matrices but calculate the average weighting matrices and proceed with them. This has the advantage that we can calculate the standard deviation for each cell of the four matrices, helping us to study the significance of each cell and therefore prevent wrong conclusions.

3.3.2 Multiclass SVM

A popular approach for document classification is supervised machine learning. Three of the most common methods are maximum entropy, naive Bayes and SVM classification. Pang et al. [PLV02] compare these three methods in the domain of sentiment analysis and show that the SVM approach achieves the best results. Similarly, Danisman and Alpkocak [DA08] compare the three methods in the task of emotion classification and also show that the SVM approach leads to the best results. For this reason we also use a SVM based emotion classifier in this thesis. We implement a multiclass SVM by using the LIBLINEAR [FCH+08] library again. The SVM has to decide to what basic emotion $y \in Y$ an input Tweet belongs to. We generally use unigram features only for this SVM because if we would use n-grams with $n > 1$, the SVM may learn modifier detection implicitly. Consider the Tweet "Just had a kid dump water on me.... I am not happy #angry"[12] annotated with the basic emotion "anger". If we use unigrams, the SVM considers each word independent from each other, but if we use bigrams, it would learn that the phrase "not happy" corresponds to the basic emotion "anger". We want to prevent this implicit modifier detection because we can not measure its impact. However, we will conduct an emotion classification experiment using uni-, bi- and trigrams in order to study wether this makes the use of additional modifier detection systems unnecessary. Since the use of unigrams leads to a large amount of features, we use the linear kernel again because it also performs very well with a large feature space [CL08]. Therefore, we only have to find the best value for the parameter C by performing a grid search and 10-fold cross-validation.

[12]https://twitter.com/beefcakezombie/status/578348799518621698

4 Experiments

This chapter presents the different corpora in Section 4.1. We differentiate between automatically annotated corpora used for emotion classification and hand-annotated corpora used for the modifier scope detection. In Section 4.2 we present our hypotheses and the experiments we conduct in order to prove them. Finally, Section 4.3 shows and discusses the results of our experiments.

4.1 Corpora

In order to conduct experiments, we have to create corpora to train and test our different emotion classification and modifier detection methods. Creating large training and test corpora by hand is very time-consuming and therefore, we use an approach based on the paper by Wang et al. [WCTS12], that automatically annotates a large amount of Tweets in short time. We use Twitters streaming API which allows us to receive random Tweets from Twitter and then apply filters to store Tweets containing specific hashtags only. For this, we create a filter list for each basic emotion which contains multiple hashtags that indicate the corresponding basic emotion. If we apply this filters to the Tweet input stream, we can use the hashtags to label the Tweets with basic emotions automatically. Furthermore, we only use Tweets written in English. Table 4.1 shows all the filter hashtags for each basic emotions.

joy	anger	fear	sadness	surprise	disgust
#glad	#anger	#afraid	#bitter	#surprise	#disgust
#happiness	#hate	#angst	#grief	#surprised	
#happy	#hatred	#fear	#misery		
#joy	#rage	#panic	#sad		
#lucky		#scare	#sadness		
#luck		#worry	#sorrow		
#pleasure			#unhappy		

Table 4.1: Filter hashtags for each basic emotion.

Considering the Tweet "After a few months I finally get #xboxlive back. #happiness"[1], we use the word #happiness to label this Tweet as "joy" because the filter list for the basic emotion "joy" contains the word #happiness.

This method generates a large amount of annotated Tweets but also leads to the problem that the data is noisy. Using this approach, we assumed that each author labeled his Tweet correctly according to the expressed emotion, but, of course, this is not the case for all Tweets. The following list shows some examples of Tweets that are labeled wrongly.

- "Before her meltdown in 2007, Britney already dropped hints and warned us about her struggles through her song #Lucky, yet no one listened."[2].

  Labeled emotion: "joy". Actual emotion: "sadness".

- "Not to jump on the bandwagon, but in a shop window I saw four candles that grew out of actual fork handles. I giggled for ten minutes. #Sad"[3].

  Labeled emotion: "sadness". Actual emotion: "joy".

- "I got 2 horror movies from the library that i am going to enjoy right when i get home #horror"[4].

  Labeled emotion: "fear". Actual emotion: "joy".

Further disadvantages of corpora consisting of Tweets is that they often contain internet slang, for example as in the Tweet "A person #hates u for 1 of 3 reasons. They 1) want 2b U; 2) #hate themselves. 3) C U as a #threat. ..."[5], and also spelling mistakes, which would be very costly to auto-correct. Both problems result in very specific features that do not contribute much information for our classifiers.

Before we create our different corpora, we preprocess the gathered Tweets. We do not use hashtags as features for our emotion classifier, because we already used them to determine the gold standard emotions of the Tweets. Therefore, we change all words preceding with a "#" to the new word "XHASHTAGX" because this improves the performance of our classifier since we do not have to check each word whether it starts with a hashtag or not. This also improves the correctness of dependency trees if there are multiple succeeding hashtags at the end of a Tweet, like in "More love - less hate #PrayForOrlando #sadness #horrifying #pride #loveconquershate" and therefore

[1]https://twitter.com/BryanKidder/status/798076624457596928
[2]https://twitter.com/_reezyrae/status/773138426732228609
[3]https://twitter.com/IanMacGilp/status/715565041077133313
[4]https://twitter.com/officiallykari/status/458721953025503232
[5]https://twitter.com/JokeUntranslatd/status/752927796062593028

also slightly increases the performance for the modifier scope detection approach using dependency tree relations. Furthermore, we reduce all URLs to the word “XURLX” and all user references (“@USERNAME”) to “XUSERX” since they often occur only once in a corpus and contain no emotional information. This increases the performance of the SVM classifier because the dimension of the feature space is decreased.

4.1.1 Automatically Annotated Corpora

We use the automatically annotated Tweets to create several training and test corpora. Table 4.2 shows the different created corpora which we use in our experiments. For each corpus it shows how many Tweets of each basic emotions it contains and the total amount of Tweets.

emotion	trainAll	testAll	trainEmoMod	testEmoMod	trainEmoModEqual
joy	597.992	299.028	155.125	76.953	1.000
anger	59.591	29.501	29.491	14.024	1.000
fear	68.886	34.504	32.312	16.586	1.000
sadness	207.026	103.607	96.308	48.897	1.000
surprise	24.582	12.483	5.858	3.185	1.000
disgust	1.923	877	906	355	1.000
total	960.000	480.000	320.000	160.000	6.000

Table 4.2: Emotion classification corpora.

The corpora “trainAll” and “testAll” contain random selected Tweets. We use these two corpora to train our emotion classifier and to evaluate the “real world” performance and modifier impact. The corpora “trainEmoMod” and “testEmoMod” only include Tweets containing at least one word from the emotion lexicon and at least one modifier cue. Since our goal is to study the impact of modifiers on emotion expressions, we use these corpora to create a special environment. It is possible that the results using the “trainAll” and “testAll” corpora do not show any improvements if modifiers are taken into account, even if they improve the emotion classification. This is the case if the number of Tweets containing emotion and modifier words are a lot less in the real world environment than the Tweets containing no emotion and modifier words. To prevent wrong conclusions, we use the “trainEmoMod” and “testEmoMod” corpora to study the modifier impact on emotion expressions independently from the real word performance.

The previous corpora contain an unevenly balanced amount of Tweets for each basic emotion. We need an evenly balanced corpus for the word list classifier to train the weighting matrices with the same importance for each basic emotion. We want the basic emotions considered equally important because we will use the weighting matrices to study how modified emotion words correspond to basic emotions, which is our second goal of this thesis. For this purpose, we create the “trainEmoModEqual” corpus that contains an equal amount of Tweets per basic emotion. Similar to the “trainEmoMod” and “testEmoMod” corpora, this corpus also includes Tweets containing at least one emotion and modifier word only. The number of Tweets in this corpus is low compared to the others because the amount of Tweets relating to the basic emotion “disgust” is very low. As we want to create a corpus containing an evenly distributed amount of Tweets, we have to use a small amount of Tweets for each emotion. Another reason is that the algorithm has to classify and evaluate this corpus a few thousand times during the weighting matrices training and therefore have to compensate the large amount of time needed for the training.

4.1.2 Hand-Annotated Corpora

We have to be able to evaluate the performance of our modifier scope detection methods because we want to compare them in order to choose the best performing approach, which we will then use for the emotion classification experiments. Another reason why we need to know how well the modifier detection performs, is the fact that we can draw reliable conclusions only for a good performing system. For this purpose we need a corpus of Tweets containing modifier cues annotated with their scope. We can use this corpus to apply our modifier detection methods and calculate the F_1 score by comparing the results with the gold standard annotation. Furthermore, the modifier scope detection approach using SVMs also needs a corpus to train the hyperplanes. Since no such corpus exist for our purpose, we have to create and annotate a new corpus by hand.

The determination of full modifier scopes can be difficult and since the corpora is only annotated once, we would expect fuzzy annotations. To prevent this, we do not annotate the full scope of a cue but only emotion bearing words contained in the emotion lexicon because we assume that these words are the most important ones for emotion detection in Tweets. We create an annotation for each emotion-modifier word combination of a Tweet and flag them either as correlating (emotion word is within the scope of the cue) or not correlating. For example, Figure 4.1 visualizes the annotation scheme of the sentence “I do not love but hate you very much.”. The word “not” is a negation cue and “very” an intensifier cue while “love” and “hate” are emotion bearing words. As can be seen in the figure, this creates four emotion-modifier combinations but only two combinations correlate to each other.

To prevent typing errors and to check the annotations for consistency, we use a small annotation program. The program shows a Tweet containing at least one modifier cue

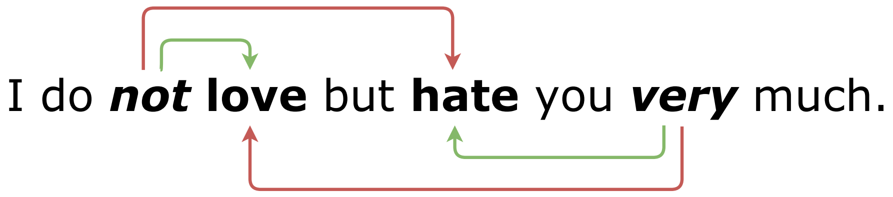

Figure 4.1: Annotations for the sentence “I do not love but hate you very much.”.

and one word from the emotion lexicon and adds an index to each word. The user then types in the indices for the modifier cue and for the emotion word and decides if they are correlating or not. Since we can not assume our lexicons to be complete, we use three different sampling methods. We not only show Tweets in the annotation program containing both, a modifier cue and a emotion word, but also Tweets containing only one of the two. The annotator then checks if the displayed Tweet also contains undetected modifier cues or emotion words. This allows us to find emotion words and modifier cues that may not be included in the lexicons. For example, by using this method we found several Tweets containing the words “soo” or “sooo” used as intensifiers and therefore added these to our intensifier cue lexicon. We annotated 1.000 Tweets resulting in a total of 1.913 modifier-emotion word pair annotations. We split these annotations to create four corpora. Table 4.3 shows the amount of annotations per modifier type in each corpus.

modifier type	modEval	trainNegSVM	trainIntSVM	trainDimSVM
negation	315	630	0	0
intensifier	249	0	497	0
diminisher	74	0	0	148
total	638	630	497	148

Table 4.3: Modifier detection corpora.

The corpus “modEval” contains 1/3 of the annotations from each modifier type. We use this corpora to evaluate the performance of the different modifier scope detection approaches. Furthermore, we create the three corpora “trainNegSVM”, “trainIntSVM” and “trainDimSVM” containing the remaining 2/3 annotations. The SVM scope detection approach uses this corpora to train the three hyperplanes. Table 4.3 also shows that of all detected modifiers, diminisher are by far the least common ones.

4.2 Experimental Setup

Now that we created all necessary corpora we can specify our hypotheses and the experiments we want to conduct in order to prove the hypotheses. We define the following null hypotheses relating to our goals outlined in Section 1.2:

H_1 : If **negations** and their scopes are considered, the performance of an emotion classification system can not be improved.

H_2 : If **intensifier** and their scopes are considered, the performance of an emotion classification system can not be improved.

H_3 : If **diminisher** and their scopes are considered, the performance of an emotion classification system can not be improved.

H_4 : If emotion bearing words are modified, their basic emotion can not be determined.

The hypotheses H_1 - H_3 relate to our first goal, namely to study whether an emotion classification system can be improved if modifiers are taken into account. We defined a hypothesis for each modifier type separately because they may not all have an impact on the classification task. We conduct several experiments using our emotion classifiers as defined in Section 3.3 and the previous described corpora. For the SVM emotion classifier we define the models "Model-Unigram" and "Model-Trigram". Model-Unigram contains unigram features only and is used to study the impact of modifiers on emotion expressions, while Model-Trigram contains uni-, bi- and trigram features and is used to examine if the implicit modifier detection makes the usage of explicit modifier detection systems unnecessary. We define the Model "Model-EmoDict" for the word list classifier, which uses the features contained in the NRC emotion lexicon by Mohammad and Turney [MT13]. We study the performance of the emotion classification systems without modifiers, for each modifier type individually and for all modifiers simultaneously taken into account.

Hypothesis H_4 relates to our second goal, that is to understand the correlation between modified emotion words and basic emotions. We study the different weighting matrices of the word list classifier and observe the basic emotion change of a Tweet when modifiers are taken into account. This provides information about the correlation between modified and non-modified emotions, which we use to prove or decline this hypothesis.

4.3 Results & Discussion

In this section we will conduct several experiments using the previous defined methods, corpora and models and discuss the results in order to prove or decline our hypotheses and to reach our thesis goals. Before we run the experiments on emotion classification, we have to evaluate and compare the different modifier detection methods so that we can use the best performing approach in our emotion classification experiments. At first, we will determine the best value of n for the next-n method as mentioned in Section 3.2.1. We apply the next-n heuristic on the corpus "modEval" several times for different values $n > 0$ and calculate the resulting macro-average F_1-score of the scope detection, demonstrated in Figure 4.2.

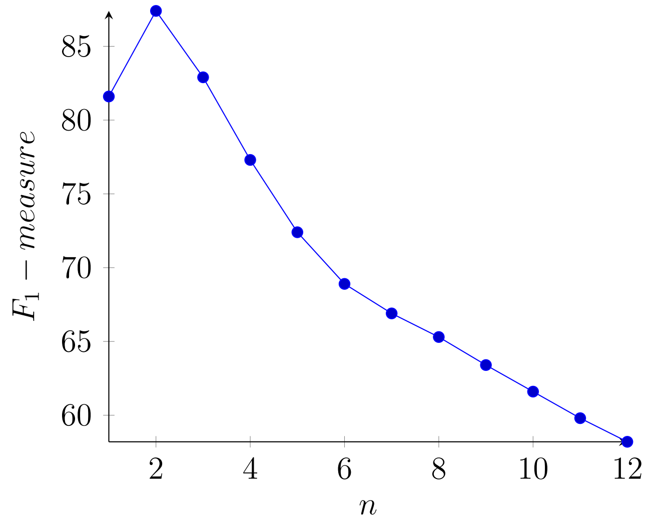

Figure 4.2: Different values of n for next-n modifier detection.

The figure shows that the highest F_1-score is achieved for $n = 2$ and the graph is strictly monotonically decreasing for values $n > 2$. The recall increases for values $n > 2$ but the precision decreases at a higher rate, leading to lower F_1-scores. Thus, we use $n = 2$ for all further experiments using the next-n method. This value for n is against our expectations because Reitan et al. [RFGB15] evaluated an average of $n = 3.8$ tokens in a negation scope using a Twitter corpus as well. A reason for this could be that intensifier and diminisher have a smaller scope. To analyze this we search the best value of n for each modifier type individually. However, $n = 2$ still results in the highest F_1-score for each modifier type. Another reason for the lower value of n might be the fact that we do not consider the full scope of a cue but only whether emotion words are inside or not. This would mean that emotion words are often closer to the modifier cue than other words in a scope.

Since we determined the best value for the next-n approach, we can compare our three modifier scope detection methods. We use the corpora “trainNegSVM”, “trainIntSVM” and “trainDimSVM” to train three SVMs for the modifier scope detection. We apply all three modifier scope detection approaches to the corpus “modEval” and evaluate the results, as can be seen in Table 4.4.

Modifier detection evaluation using modEval corpus									
	Next-2 heuristic			DepTree			SVM		
Modifier	P	R	F_1	P	R	F_1	P	R	F_1
Negator	93.6	87.9	**90.7**	93.0	80.4	86.2	78.7	89.4	83.7
Intensifier	91.7	93.7	**92.7**	90.7	83.0	86.7	91.4	89.4	90.4
Diminisher	72.8	88.9	**80.0**	75.0	50.0	60.0	66.7	55.6	60.7
Macro-avg.	86.0	90.2	**87.8**	86.3	71.1	77.7	78.9	78.2	78.3

Table 4.4: Comparison of modifier detection methods.

The table shows the precision (P), recall (R) and F_1-score (F_1) for each of the three methods and for each modifier type. We can see that the next-2 heuristic achieves by far the best results with an about 10% higher F_1 score compared to the other approaches. Compared to the similar method by [RFGB15], our next-2 approach achieves a 14.5% higher F_1 score. They achieve a higher recall because they modify all words after a cue to the next punctuation mark, but achieve a much lower precision. The reason for the good performing next-2 approach is probably because Tweets are mostly short and simple sentences. Obviously, this approach yields false negatives if an actual modified word is > 2 words to the right from a modifier cue away or if it is to the left of a cue like in the Tweet “**Enjoying** Amsterdam just a **bit**...”[6]. The actual modified emotion word “Enjoying” is to the left of the modifier cue “bit” and therefore the algorithm does not flag “Enjoying” as diminished. Furthermore, false positives occur if an actual not modified word is $<= 2$ words to the right from a modifier cue away. The SVM method achieves comparable results to the approach by Councill et al. [CMV10]. They achieve a F_1 score of 80% for negation scope detection on a product review corpus. Our SVM approach may outperform the next-2 heuristic if we use much larger training corpora. The major reason why the DepTree approach fails to detect modifier scopes is because of wrongly created dependency trees. For example in the Tweet “More love - less hate #PrayForOrlando...”[7], the generated dependency tree is incorrect due to missing punctuation. The diminisher “less” is not a child of “hate” and therefore, the modifier scope is wrong. If the Tweet

[6]https://twitter.com/RachelChudley/status/774685932339486720
[7]https://twitter.com/Mionic/status/742057483586408448

had a period after the word “hate”, the dependency tree would be generated correctly with “less” being a child of “hate”.

Furthermore, the table shows that all three approaches perform comparably bad on detecting diminisher scopes. For the next-2 approach, this is probably the case because diminisher cues appear more often after their scope than negation and intensifier cues, for example as in the Tweet “Live, love and laugh a **little**! ...”[8] and therefore, this method does not detect these scopes. For the DepTree approach, we observe that diminisher cues are more often not a direct child of correlating emotion words compared to negation and intensifier cues. Regarding the SVM approach, there are two main reasons for the bad performance. The first reason is our modifier policy described in Section 3.2.2, where we apply the SVM for diminisher scope detection after the other two SVMs. The second reason is due to the fact that we have the smallest training data available for the diminisher scope SVM.

Now that we determined the next-2 heuristic to be the best performing modifier scope detection method, we use this approach for the emotion classification experiments. We do not use stemming in our further experiments because test show that stemming leads to a negligible decrease in F_1 score. A reason for this is that we can lose information trough stemming. For example the word “loved” appears in more Tweets labeled as “sad” than in Tweets labeled as “joy” and the opposite is the case for the word “love”. If we apply stemming, we stem the word “loved” to “love” and thus changing its emotional information. Nevertheless, this finding is against our expectations, because the results of Danisman and Alpkocak [DA08] show a slight performance increase if stemming is applied. Since the impact of stemming is negligible in both works, we assume that the shifting influence of stemming occurs because of very different used corpora.

Our first emotion classification experiment uses the SVM classifier together with the model “Model-Unigram”. We train the classifier with the corpus “trainAll” and test the performance on the corpus “testAll”. We repeat this experiment four times; firstly without modifier detection and then considering each modifier type individually using the next-2 heuristic. Table 4.5 shows the precision, recall and the F_1 score of this four tests and highlights the best F_1 scores for each basic emotion.

As we can see in this table, the classifier performs very good for the emotion “joy” but fails to detect “disgust”. A reason for this may be due to the fact that the used corpora contain much more Tweets labeled as “joy” than as “disgust”. Another reason is probably that “joy” is the only pure positive basic emotion and is therefore easier for the classifier to detect, while there are multiple negative emotions. Since there are only a few disgust Tweets in our corpora, the classifier can not learn to distinguish between the different

[8]https://twitter.com/21DaysOf/status/730528657664434176

SVM + Model-Unigram using trainAll and testAll						
	no modifier detection			next-2 (negations only)		
emotion	P	R	F_1	P	R	F_1
joy	82.0	94.6	87.9	82.5	94.6	**88.1**
anger	68.3	32.2	43.7	67.2	33.4	**44.6**
fear	77.4	50.7	61.3	76.6	52.8	**62.5**
sadness	74.1	66.6	70.1	74.5	66.9	**70.5**
surprise	75.3	32.3	**45.2**	74.9	32.4	45.2
disgust	18.8	03.5	**05.8**	18.6	03.4	05.7
Macro- Avg.	66.0	46.6	52.3	65.7	47.2	**52.8**
	next-2 (intensifier only)			next-2 (diminisher only)		
emotion	P	R	F_1	P	R	F_1
joy	82.1	94.6	87.9	82.2	94.6	87.9
anger	67.9	32.5	**44.0**	68.0	33.0	**44.4**
fear	77.5	51.2	**61.7**	77.4	50.9	**61.4**
sadness	74.0	66.5	70.0	74.0	66.6	70.1
surprise	74.1	32.3	45.0	74.6	32.3	45.1
disgust	16.1	02.4	04.2	18.4	03.2	05.5
Macro- Avg.	65.3	46.6	52.1	65.7	46.8	**52.4**

Table 4.5: Support Vector Machine with unigram features using trainAll and testAll corpora.

negative emotions. It is also possible that “disgust” strongly correlates with other basic emotions, which we study in the later experiments.

Furthermore, the table shows that negation detection increases the F_1 score for almost every basic emotion except “disgust” but since the classifier generally fails to detect disgust, this result is questionable. Considering intensifier and diminisher shows a slight improvement for the emotions “anger” and “fear” but the overall classification performance does not change significantly.

We conduct the same experiment again, but this time using the corpora “trainEmoMod” for training and “testEmoMod” for testing. The results are similar to the previous experiments with a slightly higher impact of negation detection, which seems reasonable as the corpora contain more negations. We expected a higher impact of the modifier detection in this experiment since, unlike in the previous experiment, the used corpora only contain Tweets with at least one modifier. See Table A.1 in Appendix A for detailed results.

SVM + Model-Trigram using trainEmoMod and testEmoMod						
	no modifier detection			next-2 (negations only)		
emotion	P	R	F_1	P	R	F_1
joy	82.7	89.0	85.7	82.9	89.3	**86.0**
anger	63.9	45.4	53.1	64.6	45.2	**53.2**
fear	80.6	65.4	72.2	81.3	65.9	**72.8**
sadness	74.0	79.3	76.6	74.2	79.6	**76.8**
surprise	52.8	20.5	29.5	53.0	20.8	**29.9**
disgust	09.1	02.0	03.3	12.5	02.6	**04.3**
Macro- Avg.	60.5	50.3	53.4	61.4	50.6	**53.8**
	next-2 (intensifier only)			next-2 (diminisher only)		
emotion	P	R	F_1	P	R	F_1
joy	82.7	89.1	**85.8**	82.7	89.0	**85.8**
anger	64.0	45.0	52.9	64.3	45.5	**53.3**
fear	80.7	65.4	72.2	80.6	65.4	72.2
sadness	74.0	79.5	76.6	74.0	79.4	76.6
surprise	54.2	20.9	**30.2**	53.8	20.9	**30.1**
disgust	11.8	02.3	**03.8**	09.6	02.0	03.3
Macro- Avg.	61.3	50.4	**53.6**	60.8	50.4	**53.5**

Table 4.6: Support Vector Machine with trigram features using trainEmoMod and testEmoMod corpora.

The last experiment with the SVM classier is similar to the previous one but uses the model “Model-Trigram” this time and again the corpora “trainEmoMod” and “testEmoMod”. We conduct this experiment in order to study whether the usage of higher $n-$gram features makes the use of explicit modifier detection unnecessary. Table 4.6 shows the results of this experiment.

Since we determined that the next-2 heuristic achieves the highest F_1 score, we expected that the usage of trigram features implicitly models that modifier scope detection but as Table 4.6 shows this is not the case. We can notice that the classifier overall performs better using the model “Model-Trigram” but the average F_1 score still increases if we apply modifier detection. The table shows an increase of 0.4% in F_1 score which is slightly less than the increase achieved with the model “Model-Unigram”. This indicates some implicit modifier detection but the effect is negligible.

WL + Model-EmoDict using trainEmoModEqual and testEmoMod						
	no modifier detection			next-2 (negations only)		
emotion	P	R	F_1	P	R	F_1
joy	73.7	45.6	**56.3**	74.3	44.9	56.0
anger	24.7	33.4	**28.4**	23.6	35.0	28.2
fear	29.4	52.4	37.6	29.7	50.4	37.4
sadness	47.9	18.9	27.1	45.9	19.8	**27.7**
surprise	05.2	40.7	**09.2**	05.1	41.2	09.1
disgust	00.5	24.6	**01.0**	00.5	21.5	00.9
Macro- Avg.	30.2	35.9	**26.6**	29.9	35.5	26.5
	next-2 (intensifier only)			next-2 (diminisher only)		
emotion	P	R	F_1	P	R	F_1
joy	71.9	37.1	49.0	74.3	44.9	56.0
anger	25.1	30.7	27.6	23.6	35.0	28.2
fear	29.5	52.3	**37.8**	29.7	50.4	37.4
sadness	49.2	20.4	**28.8**	45.9	19.8	**27.7**
surprise	04.4	45.6	07.9	05.1	41.2	09.1
disgust	00.5	25.4	01.0	00.5	21.5	00.9
Macro- Avg.	30.1	35.3	26.3	29.9	35.5	26.5

Table 4.7: Word list classifier using trainEmoModEqual and testEmoMod corpora.

The next experiment we conduct uses the word list classifier together with the model “Model-EmoDict”. We train the weighting matrices using the corpus “trainEmoModEqual” and test the performance using the corpora “testEmoMod”. We do not apply the word list classifier to the corpus “testAll” since it includes Tweets that do not necessarily contain emotion words and therefore can not be classified by the word list classifier. Again, we test the performance without modifier detection and for each modifier taken into account individually. The results for this experiment can be seen in Table 4.7. As the table shows, applying modifier detection does not increase the performance of the classification. The only basic emotion for which a noticeable higher F_1 score can be achieved is “sadness”. The overall performance of the word list classifier is rather poor with F_1 scores about 26-27% and therefore we do not consider these results to be really meaningful. We primarily used the word list classifier to create the weighting matrices, which we will study later to achieve the second goal of this thesis.

Classifier comparison using testEmoMod corpus									
	No modifier detection								
	SVM + Model-Unigram			SVM + Model-Trigram			WL + Model-EmoDict		
emotion	P	R	F_1	P	R	F_1	P	R	F_1
joy	79.9	88.9	84.2	82.7	89.0	85.7	73.7	45.6	**56.3**
anger	63.0	40.1	49.0	63.9	45.4	53.1	24.7	33.4	**28.4**
fear	78.1	60.4	68.2	80.6	65.4	72.2	29.4	52.4	37.6
sadness	71.7	75.7	73.7	74.0	79.3	76.6	47.9	18.9	27.1
surprise	53.9	17.0	25.9	52.8	20.5	29.5	05.2	40.7	**09.2**
disgust	06.6	01.5	**02.4**	09.1	02.0	03.3	00.5	24.6	01.0
Macro- Avg.	58.9	47.3	50.5	60.5	50.3	53.4	30.2	35.9	**26.6**
	Full modifier detection with next-2								
	SVM + Model-Unigram			SVM + Model-Trigram			WL Model-EmoDict		
emotion	P	R	F_1	P	R	F_1	P	R	F_1
joy	81.1	89.3	**85.0**	83.0	89.4	**86.1**	72.5	36.7	48.7
anger	61.8	41.2	**49.4**	65.1	44.8	53.1	24.1	32.1	27.5
fear	76.0	63.8	**69.4**	81.3	65.5	**72.5**	30.1	50.0	37.6
sadness	73.1	75.7	**74.4**	73.9	79.8	**76.7**	47.8	21.3	**29.4**
surprise	50.2	18.7	**27.2**	54.8	21.2	**30.5**	04.3	47.2	07.9
disgust	02.7	00.9	01.3	14.1	02.6	**04.3**	00.5	22.6	01.0
Macro- Avg.	57.5	48.3	**51.1**	62.0	50.5	**53.9**	29.9	35.0	25.3

Table 4.8: Full versus no modifier detection comparison.

The last emotion classification experiment compares the performance without modifier detection and with all modifiers taken into account simultaneously. We use the SVM classifier with the model “Model-Unigram” and with “Model-Trigram” and also use the word list classifier with the model “Model-EmoDict”. The SVMs are trained on the corpus “trainEmoMod” and the word list classifier is trained using the corpus “trainEmoModEqual”. Then we evaluate the performance of all classifiers using the corpus “testEmoMod”. We conduct this experiment because the modifier types may interact with each other leading to a better performance when we consider all together. Table 4.8 shows the results for this experiment. The upper half of the table shows the results for each classifier without modifier detection and the bottom half with all modifiers considered simultaneously. The highest F_1 scores for each classifier are highlighted.

As can be seen in the table, the F_1 score increases for both SVM classifier experiments. The results for the word list classifier show a harmful effect through the modifier detection. This may be due to the fact that the weighting matrices were trained separably. Again, we do not consider the results as really meaningful. Furthermore, we can see that the detection of all modifiers simultaneously do not or negligibly improve the SVM classifier performance compared to the results achieved using only negation detection. The SVM classifier with the model "Model-Unigram" and negation detection achieves also a F_1 score of 51.1% and with the model "Model-Trigram" a score of 53.8% compared to 53.9%. We can conclude that the different modifier types do not really interact with each other and that intensifier and diminisher do not improve the classification performance.

All conducted experiments using the SVM classifier show a noticeable increase in F_1 score when negation detection is applied. We therefore reject the null hypothesis H_1 and accept its alternative hypothesis: if negations and their scopes are considered, the performance of an emotion classification system can be improved. Since the detection of intensifier and diminisher shows no consistent nor significant improvements, we do not reject the null hypotheses H_2 and H_3. This does of course not mean that they are approved.

Now we will analyze the weighting matrices used in the word list classifier experiments. The neutral weighting matrix was created by averaging a total of 28 separate optimized matrices using the random-restart hill climbing algorithm. The hill climbing algorithm performed on average 2720 optimization steps for each of this 28 matrices. The resulting averaged matrix can be seen in Table 4.9. The rows represent the basic emotions of the recognized words and the columns indicate the to be classified basic emotions. We define $c_{i,j}$ as the cell in row i and column j with $1 \leq i, j \leq 6$. For example, the cell $c_{1,5}$ shows the weighting of the class "surprise" for a Tweet containing a word relating to "joy". Besides the weighting, each cell also shows the standard deviation in brackets. We discuss the different weightings and show example Tweets to explain how the associated results come to be. The relevant emotion words in each Tweet are highlighted.

We can see that each basic emotion has the highest weighting to itself ($c_{i=j}$), which of course makes perfect sense. We can further see that "joy" has a slight correlation with "surprise" ($c_{1,5}$) and vice versa ($c_{5,1}$). The reason for this is a large amount of Tweets expressing a positive surprise like "Still can't believe my **cute baby** shower

Neutral weighting matrix						
emotion	joy	anger	fear	sadness	surprise	disgust
joy	**1.1** (0.9)	-0.5 (1.0)	0.1 (1.2)	0.2 (0.9)	**0.5** (0.7)	-0.6 (1.1)
anger	-1.3 (1.0)	**0.8** (0.7)	0.1 (0.5)	-0.1 (1.0)	-3.0 (1.7)	0.1 (0.8)
fear	-1.1 (1.5)	-0.1 (0.8)	**1.5** (1.2)	-0.3 (1.9)	0.1 (0.5)	-0.1 (0.9)
sadness	-1.1 (1.1)	**0.5** (1.0)	-0.2 (1.3)	**1.1** (1.2)	-0.5 (0.8)	0.0 (1.1)
surprise	**0.4** (1.1)	-2.6 (1.9)	-1.2 (1.9)	**0.3** (1.4)	**1.6** (1.3)	**0.5** (1.2)
disgust	0.0 (1.2)	**0.7** (0.8)	-0.7 (1.0)	-0.5 (1.3)	-1.9 (1.3)	**1.5** (1.1)

Table 4.9: Word list neutral weighting matrix.

#afternoontea #surprise #ourgirl"[9]. Further, the emotion "sadness" shows a slight correlation with "anger" ($c_{4,2}$). As an example for this we can consider the Tweet "I feel real **bad** when they use #religion as a reason to #hate ... Doesn't religion stand for the opposite?", which is labeled as "anger" but contains the word "bad" indicating the emotion "sadness". Also, "surprise" shows slight correlations with "joy" ($c_{5,1}$), "sadness" ($c_{5,4}$) and "disgust" ($c_{5,6}$). Some researches divide the emotion "surprise" into positive and negative surprise. Our results somewhat reflect this approach with "joy" indicating positive "surprise" and the other indicating negative "surprise" but since the weightings are low with a relatively high standard deviation, we hesitate to draw such conclusions. Lastly, "disgust" shows a correlation with "anger" ($c_{6,2}$). A reason for this might be that they share many words in the emotion lexicon, like the word "pathetic" contained in the Tweet "They are 'terrorists' not 'Islamists', you **pathetic** excuse for a journalist !!!! #hate..."[10].

Now we analyze the three weighting matrices used for modified emotions, starting with the negation matrix. A total of 49 matrices were created with an average of 1391 optimization steps. We average this matrices, creating the final negation weighting matrix shown in Table 4.10.

We can see that the negation of "joy" cancels out the weighing to itself completely ($c_{1,1}$). Instead it now strongly correlates with "sadness" ($c_{1,4}$), which seems intuitive. This

[9]https://twitter.com/jen_hankin/status/842373931403214848
[10]https://twitter.com/EzDimz/status/754433229923508224

Negation weighting matrix						
emotion	joy	anger	fear	sadness	surprise	disgust
joy	-0.1 (0.9)	**0.2** (0.9)	-1.4 (1.2)	**1.4** (1.0)	0.1 (0.8)	**0.3** (0.8)
anger	-0.5 (1.1)	**0.8** (1.1)	-0.1 (0.8)	-0.5 (1.4)	-0.6 (1.2)	0.2 (1.0)
fear	-0.3 (1.2)	0.1 (0.8)	**1.5** (1.2)	0.1 (1.1)	0.1 (0.8)	-0.4 (1.2)
sadness	**0.8** (1.0)	0.3 (0.8)	-0.3 (0.8)	0.1 (1.0)	-0.1 (0.8)	-0.3 (1.1)
surprise	-0.1 (1.1)	-0.6 (1.4)	0.1 (1.3)	-0.9 (1.5)	**1.8** (1.3)	**0.7** (1.1)
disgust	**0.7** (1.4)	**0.8** (1.0)	-0.5 (1.1)	-0.7 (1.3)	0.1 (1.3)	0.3 (0.9)

Table 4.10: Word list negation weighting matrix.

weighting can be explained with Tweets like "Not sure how this happened but in two days I've somehow gained 5 lbs...so **not happy** about this. #ugly #fatty #depressed #sad"[11]. Furthermore, "joy" indicates a correlation with "anger" ($c_{1,2}$) and "disgust" ($c_{1,6}$). The correlation to "anger" seems reasonable since we discussed an example Tweet in the introduction that contained negated "joy" and was labeled as "anger" but the standard deviation of these two values is much higher than the weightings itself so that these values are not reliable. Negated "anger" and "fear" still have a significant weighting to themselves only ($c_{2,2}$ and $c_{3,3}$). For negated "anger" there are two main reasons:

1. Tweets containing negated words correlating to "anger" and that are labeled as "anger" often contain sarcasm, like in the Tweet "As if things weren't bad enough.. Please.. Go ahead and raise my rent 320$ more.. I don't mind at all.. #hate #cantwaittomove"[12].

2. Another reason is that such Tweets often use comparisons like the phrase "no hate like" in the Tweet "There is **no hate like** #White #Christian #hate meriKKKha And this was not just 'students' this was parents and teachers as well"[13].

[11]https://twitter.com/HotDish85/status/810145108062588932
[12]https://twitter.com/catsmeowish/status/736322671684784128
[13]https://twitter.com/WldRvrFshr/status/810192492050186242O

The reason why negated “fear” still correlates to itself is because most of these Tweets want to encourage people not to have fear but still use fear related hashtags. This leads to noisy data as described in Section 4.1. The following list shows the problem with two example Tweets:

- “**Don’t worry**, let God take control. **#worry**”[14].
- “"**No fear** is stronger than you are." - Mark David Gerson **#fear** #quote #spirituality”[15].

It seems that negated “joy” not only correlates to “sadness” but also the other way round since negated “sadness” also shows a correlation with “joy” ($c_{4,1}$). Again, this weighting seems intuitive and can be explained with Tweets like “Yes! I’m about to eat this piece of cheesecake and I **don’t feel guilty** about it. #indulgingalittle #cheesecake #happy”[16]. Negated “surprise” also has the highest weighting to itself ($c_{5,5}$) due to linguistic devices such as questions and comparisons demonstrated in the following examples:

- Comparison: “**Ain’t no party like** a birthday party when @LJ_Rader shows up #surprise”[17].
- Question: “**Isn’t this lovely?** A #surprise birthday cake ordered by a #Lagos based client for her man...”[18].

Also, negated “surprise” shows slight correlations with “disgust” ($c_{5,6}$), which can be explained with Tweets like “It probably shouldn’t be shocking, but I’m always stunned by the level of pure hate that comes from some #leftists, just #disgusting ppl”[19]. This Tweet is labeled as “disgust” and contains the negated emotion word “shocking”, which corresponds to the basic emotion “surprise”. Negated “disgust” shows a slight correlation with “joy” ($c_{6,1}$) and “anger” ($c_{6,2}$). A closer look on the values shows that the correlation between negated “disgust” and “joy” ($c_{6,1}$) is almost as high as the one between negated “sadness” and “joy” ($c_{4,1}$). This parallel is not surprising as almost the half of the “disgust” emotion words overlaps with the ones corresponding to “sadness”. So in Tweets like “Be proud of who you are and **not ashamed** of how someone else sees you!!! #smile #happy“[20], the phrase “not ashamed” has an impact on the weightings corresponding to “joy” for both, negated “sadness” ($c_{4,1}$) and negated “disgust” ($c_{6,1}$). For a similar reason the correlation between negated “disgust” and “anger” ($c_{6,2}$) can be explained. As we

[14]https://twitter.com/goandshare/status/770074417787469824
[15]https://twitter.com/LS_IA_AD/status/829915608624013312
[16]https://twitter.com/_Angelic_Heart/status/818262321852256257
[17]https://twitter.com/caroline_p_s/status/713902920610095104
[18]https://twitter.com/Cynhamscakes1/status/755774733355016193
[19]https://twitter.com/ct_liberators/status/843163279828664321
[20]https://twitter.com/_NirayMars_/status/758686057227546624

can see, the weighting for negated “anger” to “anger” ($c_{2,2}$) is equal to the weighting for negated “disgust” to “anger” ($c_{6,2}$). This is because more than half of the emotion words correlating to “disgust” also correlate to “anger” and therefore affect both of these weightings.

Now that we analyzed the negation weighting matrix, we proceed with intensifier weighting matrix. The intensifier matrix was created by averaging 53 matrices, which were optimized in 1248 steps on average. The resulting matrix can be seen in Table 4.11.

Intensifier weighting matrix						
emotion	joy	anger	fear	sadness	surprise	disgust
joy	**1.0** (0.9)	-1.0 (1.3)	0.1 (1.2)	0.0 (1.1)	**1.0** (1.3)	-0.7 (1.3)
anger	-0.5 (1.1)	**1.2** (1.0)	0.2 (0.9)	**0.6** (1.0)	-0.6 (1.3)	0.2 (0.8)
fear	-0.6 (1.1)	0.1 (1.1)	**1.3** (1.1)	-0.4 (1.2)	0.3 (0.8)	0.2 (1.0)
sadness	-0.6 (1.1)	-0.1 (0.9)	-0.3 (0.9)	**1.1** (1.2)	0.1 (1.1)	**0.5** (1.1)
surprise	-0.4 (1.2)	0.6 (1.8)	-0.4 (1.4)	0.3 (1.6)	**2.1** (1.5)	-0.8 (1.4)
disgust	-0.3 (1.0)	0.1 (1.3)	-0.2 (1.1)	**0.7** (1.0)	-0.9 (1.5)	**1.5** (1.3)

Table 4.11: Word list intensifier weighting matrix.

The table shows that, again, each emotion has the highest weighting to itself, which also seems plausible. The weightings for intensified “joy” are similar to the weightings in the neutral weighting matrix but with a noticeable higher weighting for “surprise” ($c_{1,5}$). The reason for this is that the authors often use intensified emotions to express their positive surprise, like in the Tweet “Got this in the mail from work! **So cute**! #birthday #birthdaycard #birthdaysurprise #surprise”[21]. We can further see that a slight correlation to “sadness” exist for intensified “anger” ($c_{2,4}$). This correlation was not present in the neutral weighting matrix because if authors use emotion words that represent “anger” in a Tweet labeled as “sad”, they usually intensify them as in the Tweet “**So much anger & hate** from the #left. What they dislike in everyone else is exactly whats inside themselves. #colbert #TheResistance #sad”[22]. Intensified “sadness” shows no more a correlation with “anger” ($c_{4,2}$) but now with “disgust” ($c_{4,6}$) because of Tweets

[21]https://twitter.com/MousyGirlAuthor/status/766040801046650885
[22]https://twitter.com/rural_new/status/860829778487783424

like “Wow, I was **so wrong** about u. Your HYPOCRISY, EVIL MINDED ARROGANT SELF finally came out! #DISGUST“[23]. Intensified “disgust” shows no more a correlation with “anger” ($c_{6,2}$) but with “sadness” ($c_{6,4}$). This correlation shows similarities with intesified “sadness“ ($c_{4,6}$) due to the overlaps of emotions words.

Lastly, we will investigate the diminisher weighting matrix. This matrix was created by averaging 64 matrices, which again were separately optimized with an average of 990 steps. Table 4.12 shows the resulting diminisher matrix.

Diminisher weighting matrix						
emotion	joy	anger	fear	sadness	surprise	disgust
joy	**1.3** (1.3)	0.1 (1.0)	0.2 (0.8)	-0.2 (1.0)	**0.8** (1.3)	0.1 (0.8)
anger	-0.1 (1.0)	**0.6** (1.2)	0.2 (1.1)	0.3 (1.2)	0.3 (1.0)	-0.1 (0.8)
fear	-0.6 (1.4)	0.2 (1.1)	**0.7** (1.0)	-0.3 (1.2)	**1.3** (1.2)	0.2 (0.9)
sadness	**0.8** (1.1)	0.1 (1.3)	0.0 (1.0)	**1.0** (1.1)	-0.5 (1.5)	-0.7 (1.4)
surprise	0.1 (1.0)	-0.4 (1.3)	0.1 (1.0)	0.0 (1.0)	**1.7** (1.2)	-0.2 (0.9)
disgust	-0.2 (1.0)	-0.1 (1.0)	-0.1 (0.7)	0.1 (1.0)	-0.4 (1.3)	**1.1** (1.2)

Table 4.12: Word list diminisher weighting matrix.

As the table shows, each emotion has again a high weighting to itself. The correlation with “surprise” for the diminished emotion “joy” ($c_{1,5}$) is similar to the neutral weighting matrix. Diminished anger shows some correlations to “fear” ($c_{2,3}$), “sadness” ($c_{2,4}$) and “surprise” ($c_{2,5}$) but since these values are low and have a much higher standard deviation, we do not interpret too much in these weightings. Furthermore, we can see that diminished “fear” has a higher weighting to “surprise” ($c_{3,5}$) than to itself ($c_{3,3}$). The investigation of the corpus “trainEmoModEqual”, which was used to create this matrix shows, that it contains only very few Tweets containing diminished words that express “fear” and that are labeled as “surprise”. Most of these Tweets contain a phrase similar to “little surprise” and according to the emotion lexicon the word “surprise” also relates to the basic emotion “fear”. Therefore the value of the cell $c_{3,5}$ is not reliable. Finally,

[23]https://twitter.com/lezlie_9198/status/837153018013437952

diminished “sadness” shows slight correlations with “joy” ($c_{4,1}$) because of Tweets like “pray more and **worry less** #pray #faith #love #peace #happiness...”[24].

As we can determine the correlation of modified emotion words and basic emotions, we can reject the null hypothesis H_4 and accept the alternative hypothesis: if emotion bearing words are modified, their basic emotion can be determined.

[24]https://twitter.com/daintydame502/status/794310491480829952

5 Summary

This thesis studies the correlation between basic and modified emotions. Furthermore, it examines the impact of modifiers on emotion expressions by measuring the performance with and without modifier detection on a corpus containing real Tweets.

To measure the impact we implemented a modifier detection system. This system splits into modifier cue and modifier scope detection. For the cue detection, we used a simple lexicon approach. The modifier scope detection is realized in three different ways. One approach is called next-n heuristic, which modifies all words that are $< n$ words away to the right of the cue but stops after punctuation marks and adversative conjunctions. The DepTree approach checks for each node of a dependency tree if it has a modifier as a child and if so, it modifies this node. It also modifies nodes when they are connected to an already modified node via a non-adversative conjunction edge. Finally, the SVM approach trains a SVM for each modifier type, that decide if a word in a Tweet is modified or not.

We created a corpus by hand in order to train the SVMs and to evaluate the performance of each approach. We annotated the scope of modifiers in a total of 1000 Tweets. Using this corpus to evaluated the performance of each method, we found out that the next-2 heuristic performs best.

We implemented two emotion classification system, one using a SVM and the other using an emotion lexicon. We combined this with the next-2 modifier detection approach to study the impact modifiers. The results show a noticeable improvement of the performance if negations are taken into account. We could not observe an improvement when intensifier and diminisher are considered.

Further, we studied the weighting matrices of the word list classifier to gain a better understanding about modified emotions. Using this matrices, we show correlations between basic emotions and are able to determine the basic emotion of Tweets containing modified emotion words.

In the introduction we asked what the basic emotions of “not happy” and “not sad” are. We can now answer this questions: “not happy” almost solely correlates to “sadness” and with a few exceptions to “anger” and “disgust”, while “not sad” predominantly correlates to “joy” and a little bit to “anger”.

Future work could focus on intensifier and diminisher scopes to show whether these can also increase the performance of an emotion classification system. A modifier scope detection system that determines the full scope of a modifier cue could be implemented using a more sophisticated approach.

A Additional Results

SVM + Model-Unigram using trainEmoMod and testEmoMod corpora						
	no modifier detection			next-2 (negations only)		
emotion	P	R	F_1	P	R	F_1
joy	79.9	88.9	84.2	81.1	89.1	**84.9**
anger	63.0	40.1	49.0	63.2	40.2	**49.1**
fear	78.1	60.4	68.2	75.8	63.8	**69.3**
sadness	71.7	75.7	73.7	72.6	76.2	**74.3**
surprise	53.9	17.0	25.9	51.4	18.3	**27.0**
disgust	06.6	01.5	**02.4**	04.6	01.2	01.9
Macro- Avg.	58.9	47.3	50.5	58.1	48.1	**51.1**
	next-2 (intensifier only)			next-2 (diminisher only)		
emotion	P	R	F_1	P	R	F_1
joy	79.7	89.4	84.3	80.3	88.7	84.3
anger	63.8	40.0	**49.1**	66.0	38.3	48.5
fear	78.0	60.1	67.9	73.9	62.4	67.7
sadness	72.2	75.1	73.7	71.8	75.8	73.7
surprise	51.8	17.9	26.6	51.7	17.3	26.0
disgust	03.9	00.9	01.4	06.5	01.5	**02.4**
Macro- Avg.	58.2	47.2	50.5	58.4	47.3	50.4

Table A.1: Support Vector Machine with unigram features using trainEmoMod and testEmoMod corpora.

B Modifier Lexicons

List of all negation cues				
aint	cannot	cant	darent	denied
denies	didnt	doesnt	dont	hadnt
hasnt	havent	havnt	isnt	lack
lacking	lacks	mightnt	mustnt	neednt
neither	never	no	nobody	none
noone	nor	not	nothing	nowhere
n´t	n’t	n‘t	oughtnt	shant
shouldnt	wasnt	without	wouldnt	

Table B.1: Negation lexicon.

List of all diminisher cues				
almost	barely	bit	exiguous	faintly
fairly	few	fewer	hardly	insignificantly
kinda	less	little	marginally	moderately
modicum	mostly	nearly	negligibly	partly
partially	practically	quite	rather	rarely
relatively	reasonably	scanty	scarcely	slightly
some	somewhat	sparsely	tolerably	triflingly
virtually				

Table B.2: Diminisher lexicon.

List of all intensifier cues				
-ass	absolutely	altogether	amazingly	astoundingly
awfully	badly	decidedly	bitterly	bloody
colossally	completely	damn	deeply	drastically
dreadfully	entirely	enormously	especially	exceptionally
excessively	extraordinarily	extremely	fantastically	freaking
frightfully	fucking	fully	greatly	hella
highly	incredibly	insanely	intensely	immensely
largely	literally	lot	lots	massive
mightily	more	outrageously	particularly	perfectly
phenomenally	pretty	radically	real	really
remarkably	purely	so	soo	sooo
super	supremely	surpassingly	strikingly	strongly
terribly	terrifically	totally	thoroughly	truly
unusually	utterly	very	wicked	

Table B.3: Intensifier lexicon.

Bibliography

[BCP+07] F. Benamara, C. Cesarano, A. Picariello, D. R. Recupero, V. S. Subrahmanian. "Sentiment analysis: Adjectives and adverbs are better than adjectives alone." In: *ICWSM*. 2007 (cit. on p. 31).

[BMS12] R. C. Balabantaray, M. Mohammad, N. Sharma. "Multi-class twitter emotion classification: A new approach." In: *International Journal of Applied Information Systems* 4.1 (2012), pp. 48–53 (cit. on p. 28).

[CBH+01] W. W. Chapman, W. Bridewell, P. Hanbury, G. F. Cooper, B. G. Buchanan. "A Simple Algorithm for Identifying Negated Findings and Diseases in Discharge Summaries." In: *Journal of Biomedical Informatics* 34 (2001), pp. 301–310 (cit. on pp. 29, 34).

[CL08] Chih-Wei Hsu, Chih-Chung Chang, C.-J. Lin. "A Practical Guide to Support Vector Classification." In: *BJU international* 101 (2008), pp. 1396–400 (cit. on pp. 23, 39, 42).

[CMV10] I. G. Councill, R. McDonald, L. Velikovich. "What's great and what's not: learning to classify the scope of negation for improved sentiment analysis." In: *Proceedings of the ACL Workshop on Negation and Speculation in Natural Language Processing Uppsala Sweden* (2010), pp. 51–59 (cit. on pp. 30, 31, 37, 50).

[DA08] T. Danisman, A. Alpkocak. "Feeler: Emotion classification of text using vector space model." In: *AISB 2008 Convention Communication,* (2008) (cit. on pp. 16, 28, 42, 51).

[Ekm92] P. Ekman. *An argument for basic emotions*. 1992 (cit. on pp. 16, 28).

[FCH+08] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin. "LIBLINEAR: A Library for Large Linear Classification." In: *Journal of Machine Learning Research* 9 (2008), pp. 1871–1874 (cit. on pp. 36, 42).

[HL04] M. Hu, B. Liu. "Mining and summarizing customer reviews." In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2004, pp. 168–177 (cit. on p. 34).

[JYM09] L. Jia, C. Yu, W. Meng. “The effect of negation on sentiment analysis and retrieval effectiveness.” In: *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM ’09* (2009), pp. 1827–1830 (cit. on pp. 30, 34).

[KI06] A. Kennedy, D. Inkpen. “Sentiment classification of movie reviews using contextual valence shifters.” In: *Computational intelligence* 22.2 (2006), pp. 110–125 (cit. on p. 30).

[MB12] R. Morante, E. Blanco. “* SEM 2012 shared task: Resolving the scope and focus of negation.” In: *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics. 2012, pp. 265–274 (cit. on p. 36).

[MSB+14] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, D. McClosky. “The Stanford CoreNLP Natural Language Processing Toolkit.” In: *Association for Computational Linguistics (ACL) System Demonstrations*. 2014, pp. 55–60. URL: http://www.aclweb.org/anthology/P/P14/P14-5010 (cit. on pp. 35, 37).

[MT13] S. M. Mohammad, P. D. Turney. “NRC Emotion Lexicon.” In: (2013) (cit. on pp. 40, 48).

[PB12] M. Purver, S. Battersby. “Experimenting with Distant Supervision for Emotion Classification.” In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics* (2012), pp. 482–491 (cit. on p. 29).

[Plu01] R. Plutchik. *The nature of emotions: Human emotions have deep evolutionary roots*. 2001 (cit. on p. 28).

[PLV02] B. Pang, L. Lee, S. Vaithyanathan. “Thumbs up?: sentiment classification using machine learning techniques.” In: *... of the ACL-02 conference on ...* (2002), pp. 79–86 (cit. on pp. 29, 33, 34, 42).

[Pro17] R. T. S. Project. *Internet Live Stats*. July 2017. URL: http://www.internetlivestats.com (cit. on p. 15).

[PZ06] L. Polanyi, A. Zaenen. “Contextual valence shifters.” In: *Computing attitude and affect in text: Theory and Applications* 20 (2006), pp. 1–10 (cit. on pp. 30, 41).

[RFGB15] J. Reitan, J. Faret, B. Gambäck, L. Bungum. “Negation Scope Detection for Twitter Sentiment Analysis.” In: *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA’15)* (2015), pp. 99–108 (cit. on pp. 30, 49, 50).

[Rom12] S. Romero. “This is so cool! - A Comparative Corpus Study on Intensifiers in British and American English.” In: (2012) (cit. on p. 31).

[SI13] J. Suttles, N. Ide. “Distant supervision for emotion classification with discrete binary values.” In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7817 LNCS (2013), pp. 121–136 (cit. on p. 28).

[TM86] A. J. Thomson, A. V. Martinet. *A practical English grammar*. Oxford, 1986 (cit. on p. 31).

[UH15] O. Udochukwu, Y. He. “A rule-based approach to implicit emotion detection in text.” In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9103 (2015), pp. 197–203 (cit. on p. 28).

[WCTS12] W. Wang, L. Chen, K. Thirunarayan, A. P. Sheth. “Harnessing Twitter ‘Big Data’ for Automatic Emotion Identificatio.” In: *Proceedings - 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust and 2012 ASE/IEEE International Conference on Social Computing, SocialCom/PASSAT 2012*. 2012, pp. 587–592. ISBN: 9780769548487 (cit. on pp. 29, 43).

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature