Project Report:

Plutchik Vectors for Emotion Classification

Reflected Text Analysis beyond Linguistics
Summer term 2020
Lecturer: Dr. Nils Reiter

Wolfgang Tessadri
Matriculation number: 3405725
Msc. Computational Linguistics

Content

1 Introduction

As the numerous definitions given by various experts of the field and collected on the web page "whatisdigitalhumanities.com" show, the Digital Humanities cover a wide range of scientific topics and applicational domains. Being at the intersection of computing and the humanities one important domain is the automatized analysis and classification of texts of various forms and genres. This might involve tasks as diverse as genre classification (Lee & Myaeng 2002), author or topic identification (Stamatatos 2006), plot and structure analysis and the detection of relationship shifts of characters (Labatut & Bost 2019), e.g. in novels. Categories like genre, plot structure or authorship, however, are very complex variables and need to be operationalized in a certain way. In these regards the field of Computational Linguistics provides methods like syntactic parsing, POS-tagging and Named Entity Recognition to access potentially useful variables. Another, more recent approach contributing to the improvement of genre classification and plot analysis is the detection of emotional load in text passages. Kim et al. (2017: 17) examined the hypothesis that "emotion-related information correlates with particular genres". The authors show that stories in different genres follow different emotional arcs. Likewise, Reagan et al. (2016) determine six distinct emotional arcs, which are characteristic for the development of story plots.

While emotional load can, thus, provide useful information for various text classification tasks, the automatized classification of emotions by itself is not a trivial undertake. Schuff et al. (2017) test various ML systems for discrete emotion classification on their newly published SSEC corpus. Among Support Vector Machines, Maximum Entropy classifier, LSTM, Bi-LSTM and CNNs the Bi-LSTM reached the best performance with an F1-score of 62 over eight emotion categories. Bostan & Klinger (2018) report similar results for training and testing a Bag-of-words Maximum Entropy classifier on different combinations of emotion corpora as training and test sets. Even though the overall performance greatly varied among the different combinations the average F1-score in these experiments was 27.8 (for a full list of results see Bostan & Klinger 2018: 2114). These considerations underline the fact that the task of automatized emotion classification is by no means solved.

The project described in this report, thus, aims at developing an emotion classification approach improving current methods. In the context of this report improvement is measured along two dimensions:

1) Interpretability: Neural nets, the most prominent machine learning technique in NLP in these days, suffer from a lack of interpretability. In case of text-based tasks this is also attributable to the widespread use of word embeddings to represent textual input. As described in the ground-breaking paper by Mikolov et al. (2013) such embeddings are generated by training shallow neural nets with the task to predict a word from its context (CBOW) or the context from a word (Skip-gram). While these high-dimensional embeddings have led to performance improvements in various text classification tasks, their dimensions and respective weights are meaningless from a human perspective. The initial idea of the present project was, thus, to create "Plutchik emotion vectors" on word level, tailored to the application of emotion classification and with human-readable, i.e. interpretable, dimensions.

2) Performance: A new kind of textual representation like Plutchik vectors would not make sense if they would perform poorly when provided as an input to emotion classification systems. Therefore, the second goal of this project was to develop inputs which would not only be interpretable but would also lead to an increased or at least comparable performance of classification systems.

To describe which steps were performed to reach the listed goals the present paper is structured as follows: The following chapter, Chapter 2, first describes the theory underlying the created Plutchik vectors and the fundamental idea behind their dimensions. It then exemplifies how these ideas were transferred from theory into practice by elaborating on the Python modules, which were programmed to generate the final vectors. Chapter 3 provides an overview of the applied evaluation process and the achieved results. The project report is finalized by an outlook pointing to weaknesses and possible improvements of the generated vectors as well as open research questions.

2 Emotion Embeddings

2.1 Emotion Theories

When dealing with classification one fundamental question is, what is classified into, or, to phrase it differently: Which set of categories is applied for classification? In the case of emotion classification this is not a trivial question: What counts as an emotion is still a matter of debate. Theoretical considerations reach from defining emotions as mere side effects of physiological reactions (James-Lange theory: James 1884, Lange 1887) to the definition of emotions as the consequence of a cognitive evaluation process (Appraisal theories: e.g. Barrett 2017; Scherer 2005). The discussion is fuelled by the fact that even the set of basic emotions, i.e. the emotion categories which should be taken as a basis, are still controversial. Prominent models in the field are Paul Ekman's (Ekman 1972) and Robert Plutchik's (Plutchik 1980) emotion sets.

Ekman distinguishes between six basic emotions, which, he states, are genetically determined and therefore universal across cultures: joy, sadness, anger, fear, disgust, surprise.

This set of emotions he also proved to be valuable in the context of assigning emotions to certain facial expressions (Cohn et al. 2007). While Ekman's model is unidimensional, i.e. only distinguishing between types of emotion, Plutchik's model is characterized by a slightly increased complexity: Instead of six basic emotions, Plutchik distinguishes eight basic emotions, adding "trust" and "anticipation" to Ekman's emotion set. Moreover, he adds an intensity dimension to his model. These changes result in an emotion model, which nowadays is known as "Plutchik's wheel":

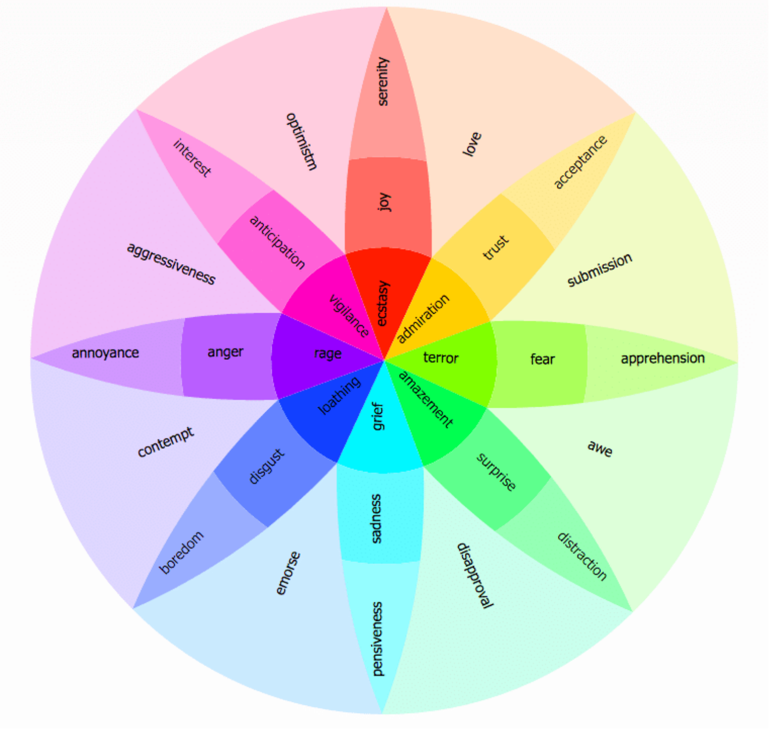


Figure 1: Plutchik's wheel of emotions[1]

[1] Taken from Nielek et al. (2017)

The inner-outer dimension in the above figure represents intensity. The closer to the centre of the wheel the more intense is the respective emotion. The spokes of the wheel represent the eight emotion categories mentioned above. In the middle of each spoke the basic emotion is displayed. In this way for each of the eight emotion categories a basic emotion (e.g. sadness) together with its less intense (e.g. pensiveness) and more intense (e.g. grief) variant are shown. Between the spokes are emotions which represent mixed forms of the adjacent emotions. Another interesting property of the described model is that the allocation of the emotions in the wheel is not random. Instead, the emotions are organized in pairs with each emotion having a counterpart: “joy” is opposed to “sadness”, “anger” to “fear”, “trust” to “disgust” and anticipation to surprise.

The property of four pairs of opposing emotions is the fundamental concept standing behind the Plutchik vectors described in this report. These will be addressed in the following chapter.

2.2 Plutchik Vectors

As described in the introduction the goal of the present project was to create vector representations on word level to enhance interpretability and performance of emotion classification systems. To achieve this the idea was to generate four-dimensional vectors on word level, which are conceptually based on Plutchik’s theory:

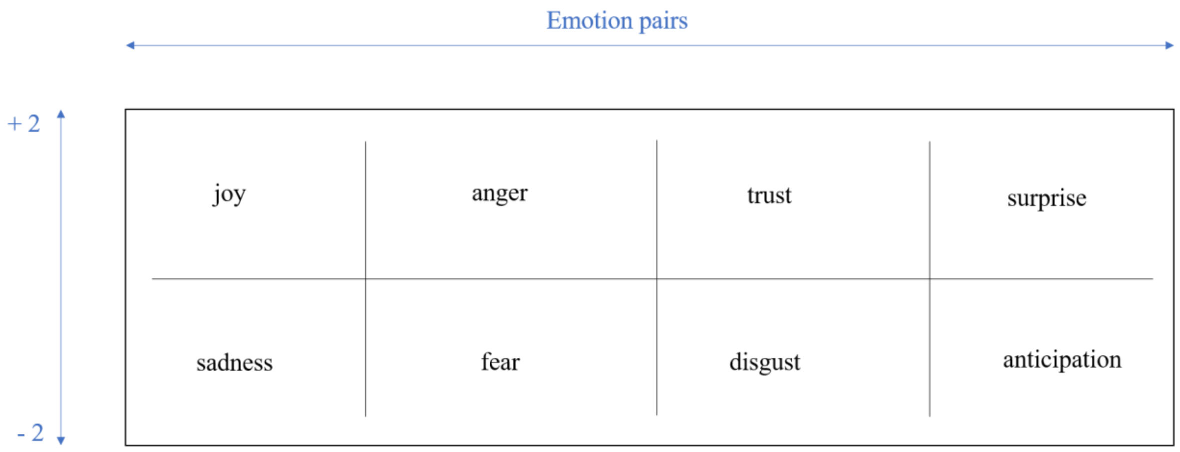


Figure 2: Dimensions emotion vectors

As shown in Figure 2 each dimension of these vectors stands for a pair of opposed emotions represented on the x-axis. The y-axis, again, indicates the intensity of one emotion prevailing over the other. A strong positive value in the first dimension, would, for example, indicate a high degree of joy associated with the word. A word like “happy” would, thus, achieve a high positive value in the first dimension, while “unhappy” would result in a (highly) negative value. The same applies to the other dimensions. A highly positive or negative value, thus, indicates the affinity of a certain word to the corresponding emotion category.

But how can this affinity be evaluated? The following chapter will describe how the theoretical notion of a Plutchik word vector was put into practice.

2.3 Implementation

The present chapter will describe the Plutchik vectors' implementation process in form of the programmed Python modules. The description is limited to a general explanation of module functionality. A more detailed specification is given in the form of code comments in the modules.

The first and central question which had to be answered in the implementation process was how the affinity of a word to an emotion could be measured. Only with this information at hand the relationship between pairs of emotions could be evaluated. A naïve approach would be to take pre-trained embeddings and to simply measure the cosine similarity of a certain word embedding to the embeddings for the terms "joy", "sadness", "anger" etc. This approach, however, has two major drawbacks:

1) The embeddings for different emotion terms do not necessarily represent the actual location of the emotion concept in the vector-space spanned by the embeddings. That is, the embedding for the term "joy" is not identical with the embedding of the concept "joy", even though they might show an increased similarity to each other.

2) As already mentioned, embeddings are trained by predicting in which context a certain word appears. This, however, leads to the side effect that antonyms, i.e. words which are opposed semantically, often show very similar vector representations. If we want, though, to evaluate the affinity of a word embedding to one or the other emotion of an opposed pair, this is an undesirable behaviour. The embedding for the word "happy" would, for example, have a high cosine similarity with the embedding for "joy" as well as for "sadness" as these terms appear in very similar contexts and, thus, are characterized by similar representations. Following the logic of Plutchik's theory, however, opposing emotions should have dissimilar representations.

The following two chapters, 2.3.1 and 2.3.2, will describe how these issues were addressed.

2.3.1 Modules 1-4: Dataset Creation and Word Choice

The solution to the first issue described in the previous chapter was to view each emotion as a core concept with a semantic field surrounding it: In this sense the fundamental idea was that, instead of simply taking the embedding of the emotion term, the actual representation of an emotion was approximated by evaluating which words frequently appear in the context of this emotion and averaging the respective word embeddings. In this way, the vector representation of an emotion is the average representation of words typically appearing in the context of a certain emotion. One big advantage of this approach is that nowadays various emotion corpora are available where textual material of variable length (sentences, tweets etc.) is labelled with emotions. These corpora were taken as a data basis. For this project, thus, to be "in the context of an emotion" does not refer to syntactic cooccurrence with an emotion term. Instead, the "context" of an emotion is defined as bearing a certain emotion label in an emotion corpus. Therefore, instead of "context of an emotion term", the notion "scope of an emotion label" is more accurate.

The first step to evaluate which words are highly associated with a certain emotion label was to prepare an appropriate dataset. This is done in module_1_dataset_creation. The module essentially loads and processes two emotion corpora files. The first and main corpus loaded is in fact a collection of corpora. This collection was created by Bostan & Klinger (2018), who provide various emotion corpora in a unified form in a single file. After loading this dataset, it contained 214283 rows of labelled texts of different length, deriving from fourteen different corpora. This dataset was further processed to exclude rows with empty values, rows labelled with “no emotion” and rows containing text in languages different than English (especially the TEC – Twitter Emotion Corpus – by Mohammad (2012) contained a lot of Dutch sentences). After these post-processing steps 102289 rows were left.

This dataset was combined with the enISEAR corpus, compiled via crowd-sourcing by Troiano et al. (2019). Finally, all rows, which were exclusively labelled with emotions other than the eight Plutchik emotions (e. g. shame and guilt) were removed from this combined dataset. This led to a dataset with 97008 rows, i.e. 97008 text instances with corresponding Plutchik emotion labels. The emotion labels were stored in list form as some rows/texts had multiple labels assigned.

After creation of the dataset emotion-typical words had to be evaluated. For this purpose, two strategies were implemented:

In module_2_tf_idf_word_choice typical words were evaluated using a tf.idf approach. tf.idf is an information retrieval measure assigning to each word in a collection of texts/documents a weight of importance in reference to the text/document it appears in. This weight specifies how much information the word conveys about the document it is in. The weight is computed using the formula:

$$\mathrm{tf\_idf}_{i,d} = \mathrm{tf}_{i,d} \cdot \mathrm{idf}_i$$

“tf” refers to “term frequency” and expresses how often a word appears in a certain document (logarithmized), while “idf” refers to “inverse document frequency” and is specified as the number of all documents in the collection divided by the number of documents the word appears in (logarithmized). Using this measure a tf.idf matrix was generated assigning a weight to each word-text combination in the previously generated dataset. For generation of the matrix the sci-kit learn method TfidfVectorizer was used.

This matrix, however, does only specify which words are typical for each single text in the created dataset. To determine word relevancies on the level of emotion categories, i.e. sets of texts bearing the same emotion label, an additional operation had to be performed: For each emotion the subset of texts bearing the respective label was extracted from the tf.idf-matrix. After extraction the weights for all words in the documents bearing the same label were summed up. At the end for each emotional category the 1500 words with the highest resulting sum were chosen. This led to the following top ten typical words for the eight Plutchik emotions:

joy	**sadness**	**anger**	**fear**
thank good thanks great day like ll happy time just	semst sad sorry just don day like miss time work	semst don just people angry hate like said anger time	semst just don today afraid like fear going day know
trust	**disgust**	**surprise**	**anticipation**
semst obama president barack vote god realdonaldtrump hillaryclinton women hillary	semst romney obama people like don just mitt felt think	really oh semst just did know don like didn day	obama vote romney election2012 debate barack election president excited 4moreyears

Table 1: Top ten emotion words with tf.idf

As can be seen, the approach is only partially successful. Although emotion typical words like “happy” for joy, “afraid” for fear and “angry” for anger were found, the selection is also flawed: In six of eight emotions the term “semst”, deriving from the hashtag “#semST”, was found to be meaningful. Moreover, the emotion words for trust, disgust and anticipation are heavily jagged towards political terms. The reason for that is that these labels were uncommonly frequently used in the Electoral-Tweets corpus published by Mohammad et al. (2015) and containing emotion labelled US-election tweets.

What’s more, the results are not equivalently expressive, because the emotion labels are not equally distributed, as can be seen from Figure 3:

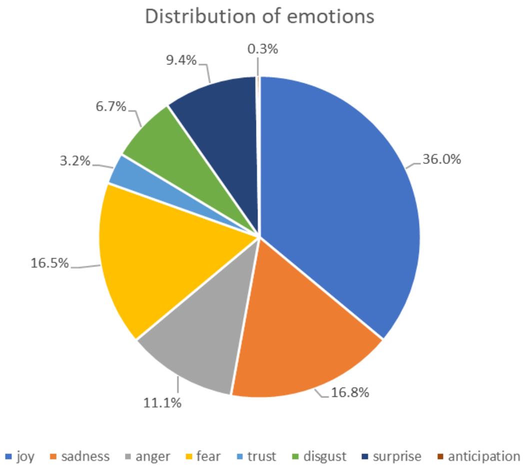


Figure 3: Relative frequency of emotion labels in created dataset

From 111174 emotion labels in the dataset[2] only 381 (0.3%) are labelled with “anticipation” and 3508 with “trust”, while more than a quarter is labelled with “joy”. Additionally, the 381 “anticipation” labels in the dataset all derive from the electoral-tweets corpus. This leads to the issue that the emotional concept of anticipation is exclusively defined in political terms.

As results were not fully convincing a second approach to evaluate emotion typical words was implemented: In module_3_pmi_word_choice instead of a tf.idf score a pointwise mutual information (pmi) score is assigned to each word. This score is defined as follows:

$$pmi_{w,c} = log\left(\frac{p(w,c)}{p(w) \cdot p(c)}\right)$$

The pmi is computed as the (logarithmized) joint probability of a word w and class c occurring together, divided by their prior probabilities in the dataset. Consequently, to evaluate this score for all words in the dataset the following steps were performed in the module:

Firstly, class probabilities were computed for all eight emotion classes. This involved counting how often each label appeared in the dataset. Then, class probabilities resulted from dividing the absolute frequency of one single label by the number of all labels.

Computing prior and joint word probability was slightly more complex. For this computation the texts in the dataset first had to be tokenized. For this purpose, a custom tokenizer was used. The prior word probability for a word, then, resulted from dividing the count of a specific word by the overall number of words in the dataset. For calculating the joint probability of a word and a specific emotion class, it was evaluated which and how often certain words appear in texts with a certain emotion label. This count was, again, divided by the overall number of words in the dataset. After this step all components to compute the pmi of a word for a class were available.

This approach, however, has one major drawback: pmi assigns high values to words which are good indicators for a class. This, in theory, is a desired behaviour, but leads to the fact, that low frequency words receive higher values. This is especially evident in the case of Hapaxlegomena: If a word appears only once in the whole dataset, then pmi will see it as a perfect indicator for the class it appears in. The problem with this is that the word is only a good indicator because of its low frequency and is not necessarily a typical proponent of the given class. To fix this, a threshold of minimum occurrences was introduced for each class. The value for the threshold was evaluated by testing and is the higher the more instances/texts a certain class had. In this way a list of typical words was generated for each emotion. Analogously to the tf.idf word lists, only the best 1500 words were chosen for each emotion category.

The following ten words were found most typical for the respective classes:

[2] As already mentioned, some of the texts in the dataset were multi-labelled.

joy	**sadness**	**anger**	**fear**
elated	fanzinator	tantrums	officialpwg
shivers	fanzoid321	indignant	intimidate
thebodyshopuk	anguished	rapeculture	lessâ\x80\x9d
mirth	disconsolate	kikme	iraqi
live.ly	despondent	enraged	co2
realmuckmaker	downcast	nohillary2016	face.~
heyday	dejected	stophillary2016	unga
exhilarating	inconsolable	resentful	alarming
chirp	mourn	~richard	jehoshaphat
gladness	heartbroken	livid	lessâ
trust	**disgust**	**surprise**	**anticipation**
pro-choice	pro-choice	puzzlement	romney.analysts
fed-up	fed-up	bewilderment	allthatsleft
detested	detested	nonplussed	gonnakillit
disgruntled	disgruntled	bewildered	blaiseff
revulsion	revulsion	unitednude	presidential
rapeculture	rapeculture	astonishment	acibookoutmtv
stophillary2016	stophillary2016	collector	nbcsnl
vomited	vomited	greysonchance	thingsthatneedtohappennow
liberalism	liberalism	astonished	douglasbass
disgusted	disgusted	iraqi	justinryanday

Table 2: Top ten emotion words with pmi

It is evident that also these results are not perfect as a lot of noise is contained in the words found most typical. Performance seems even worse than in case of tf.idf. This, however, is only because of the limited number of words shown here. When the first hundred words of the pmi and tf.idf word lists were compared, the words found by pmi were more convincing. However, even though the pointwise mutual information approach performed better when complemented with the described thresholds, these thresholds are not unproblematic. The threshold values were set by experimenting with different values and intuitively checking the first hundred output words for each emotion class. This, however, does not guarantee that there were not excluded good prototypical emotion words. For this reason, it was decided to combine both measures, i.e. tf.idf and pmi, to generate the final list of typical words for each emotion class.

This was done in module_4_tf_idf_pmi_combined. In this module the word lists per emotion determined by tf.idf and pmi are loaded and combined. The first obstacle in this combination process was the question how to combine them. A simple addition of the tf.idf and pmi score of a word in both approaches was not possible as scores were on a different scale (with tf.idf values much higher than pmi values). Therefore, to assign values on the same scale, scores were replaced with ranks: For each emotion class the words in the tf.idf word list and pmi word list were ordered and assigned a rank value. The next question was how these ranks should be weighted: Should a word with a high rank in the pmi word list have equal importance as a word with the same rank in the tf.idf word list? This question was, again, answered by testing different weightings. The best results were achieved with weighting pmi-ranks four times as much as tf.idf ranks. Words not appearing in one or the other of the word lists were assigned a constant of 1500 in the case of words missing in the pmi word list and 375 for words missing in the tf.idf word list.[3]

After word lists were combined in this way the next step was to make emotion word lists mutually exclusive: All words in the eight emotion word lists were compared and if a word appeared in more than one word list it was only kept in the word list, where it had the highest rank score.

As mentioned at the beginning of this chapter the idea of these lists of typical words per emotion was to retrieve word embeddings for all words of a list and average them to approximate the embedding of an emotional concept. In the context of this project 100-dimensional pre-trained GloVe (Global Vectors for Word Representation) embeddings were taken as a reference for embedding retrieval (Pennington et al. 2014). Thus, in a final step only those words were kept which also had an embedding representation in this resource[4].

These post-processing steps led to the final word lists which contained: 829 words for “joy”, 566 words for “sadness”, 420 words for “anger”, 619 words for “fear”, 462 words for “trust”, 432 words for “disgust”, 471 words for “surprise” and 408 words for “anticipation”.

The top ten words for each emotion were:

joy	**sadness**	**anger**	**fear**
thank	sad	angry	fear
thanks	died	anger	afraid
ll	sorry	rage	scared
happy	miss	fuck	nervous
sounds	lost	hate	frightened
great	https	furious	horror
nice	sadness	fucking	yesterday
yes	missing	offended	seen
welcome	death	stupid	anxiety
glad	rt	fuming	terror
trust	**disgust**	**surprise**	**anticipation**
hillary	disgusted	surprise	debate
abortion	romney	didn	debates
clinton	mitt	kidding	vote
equality	disgust	wow	obama
climate	republicans	believe	election
america	women	really	register
2016	gop	birthday	presidential
donald	feminists	oh	barack
scotus	feminist	actually	registered
trump	democrats	surprised	november

Table 3: Top ten emotion words with tf.idf and pmi combined

[3] The lower the rank, the better the position of the word in the ranking. If a word was missing in the pmi word list this was punished with a four times higher constant as if it would have been missing in the tf.idf list. The constant results from the maximum length of the word lists, which is 1500.

[4] Downloaded from https://nlp.stanford.edu/projects/glove/

As can be seen, even though there is still noise present, especially the results for joy, sadness, anger, fear and surprise look convincing. At this point, lists of typical words per emotion were available and, thus, the emotion embeddings could be generated by retrieving the words' GloVe embeddings and averaging them. This strategy, however, still suffered from the second issue described in the introductory part of this chapter: The averaged emotion vectors still showed a very high inter-cosine-similarity:

joy | sadness	anger | fear	trust | disgust	surprise | anticipation
0.92	0.96	0.92	0.96

Table 4: Cosine similarities of averaged emotion vectors

So, even though the sets of words were mutually exclusive for all emotion combinations, the resulting vectors were still very similar. As already described, this is undesired, because it does not correspond to Plutchik's assumptions. Moreover, if we want to evaluate the affinity/similarity of a word embedding to one or the other of an opposed emotion pair, it is crucial that they are as dissimilar as possible. For this reason, the following chapter describes one of the central parts of the present project: the emotion vector optimizer.

2.3.2 Module 5: Emotion Vector Optimizer

The idea behind the emotion vector optimizer was to build emotion vectors which were as dissimilar as possible to their respective counterpart while being as similar as possible to the average emotion vector representation mentioned in the previous chapter. For such optimization procedures the Python library scipy offers the "optimize"-method. This method requires as input one vector to be optimized and a corresponding cost function: The vector input were the aforementioned averaged vector representations of two opposed emotions[5]. The cost function had to be conceptualized manually and reflected the requirement of intra-class-similarity and inter-class-dissimilarity of the pair of opposed emotions:

$$
\begin{aligned}
cost = \alpha \cdot [\sum_{i}^{N}[cos\_sim(c_1, w_i^1) - 1]^2 + \sum_{j}^{M}[cos\_sim(c_2, w_j^2) - 1]^2] + \\
\beta \cdot [cos\_sim(c_1, c_2) + 1]^2 + \\
\gamma \cdot [(||c_1||)^2 \cdot (||c_2||)^2 - 1]
\end{aligned}
$$

Part 1 of the formula measures intra-class similarity: It iterates over all embedding representations of words evaluated as typical for the given emotions and computes the cosine similarity to their respective average representations. In case of "joy" vs. "sadness", for example, it would iteratively take the embeddings for "glad", "happy" etc. as well as "mourn", "heartbroken" etc. and compute all cosine similarities to the "joy" (for joy words) and "sadness" (for sadness words) average vectors respectively. It does this for all N or M words in the emotion word lists of the opposed emotions. To convert this into a cost function, -1 is subtracted from each cosine similarity measure, squaring the result. The reason why -1 was chosen is that

[5] The two vectors had to be concatenated beforehand as the optimize-method allowed only one vector input.

we want intra-class similarity, i.e. the similarity of all words to their average representation, to be as high as possible: If the cosine similarity of a word embedding and the average representation is highest, i.e. 1, a zero cost results.

The second line of the formula measures inter-class similarity: As the input to the optimizer and cost function are two conceptually opposed vectors, we want the resulting vectors to be as dissimilar as possible. Thus, cosine similarity between the average emotion representations must be as small as possible, which results in adding 1 to create a corresponding cost. The result is again squared.

The third part of the formula is a length normalization term, taking a 2-norm as basis for normalization. α, β and γ are hyperparameters to weight the parts of the cost function for manipulating the outcome of the optimization procedure.

Fed with this cost function and applying Powell-optimization (Powell 1970) the optimizer was able to successfully converge resulting in two weighted average emotion vectors for each pair of opposed emotions.

Moreover, later versions of the cost function were complemented with a reference and a vector set cost. The reference cost measured the distance of the weighted average emotion vector to a thematically related reference embedding. This reference embedding was the embedding of the emotion term. During the “joy” vs. “sadness” optimization process, for example, the distance to the embeddings for the terms “joy” and “sadness” is measured. In this way it should be guaranteed that the optimization process proceeds into a semantically reasonable direction. The vector set cost, again, reflected the mutual distance between all emotion vectors. Before this part of the cost function was introduced the optimization process only optimized for pairwise dissimilarity between opposed emotion. This, however, led to the fact that some non-opposed emotion vectors, like “disgust” and “fear” were undesirably close to each other. To increase distance between all vectors a term was integrated into the cost function measuring the mutual distance between all weighted emotion vectors.

This led to the following similarities between the resulting optimized emotion vectors:

	joy	**sadness**	**anger**	**fear**	**trust**	**disgust**	**surprise**	**anticip.**
joy	1.00	-0.62	-0.02	0.01	-0.06	-0.01	0.14	0.19
sadness		1.00	0.07	0.16	-0.02	-0.03	0.00	-0.05
anger			1.00	-0.54	-0.07	0.05	0.03	-0.01
fear				1.00	-0.04	0.04	0.08	0.20
trust					1.00	-0.58	0.02	-0.02
disgust						1.00	-0.02	-0.12
surprise							1.00	-0.39
anticip.								1.00

Table 5: Cosine similarities optimized emotion vectors

This matrix shows that the optimization process achieved the desired results: All emotion vectors are most dissimilar to their respective opposites (highlighted in red), while showing relatively balanced (dis)similarity to the other emotion vectors. The combinations “fear” and “sadness”, “surprise” and “joy” as well as “anticipation” and “joy” *and*“anticipation” and “fear” still show a slightly increased similarity (highlighted in green). These increased similarities seem reasonable from an intuitive perspective as, for example, fear could be seen as a form of negative anticipation and a positive surprise leads to joy. At this point, having generated the eight final emotion vectors, we have all components to build the word-level Plutchik vectors introduced in Chapter 2.2.

2.3.3 Module 6: Plutchik Vector Generator

The last step in the implementation process was the generation of the actual Plutchik vectors. This is done in module 6: The module first loads the necessary resources, i.e. the 100-dimensional emotion vectors generated in module 5 and the pre-trained GloVe embeddings. The generation process, then, is relatively simply. The module function iterates over all words being present in the GloVe embeddings file. After a word and its respective embedding is retrieved from the resource, the respective Plutchik vector is generated by the function “plutchik_vector_generator()”.

This function takes as an input the embedding vector of a word and a pair of opposed emotion vectors. It then computes the cosine similarity between the word embedding and each of the two emotion vectors. In this way the similarity of the word to one emotion or the other is evaluated. Similarity values are on the scale -1 (low similarity) and 1 (high similarity). To compute the value for one dimension of the Plutchik vector, the similarity to one emotion is subtracted from the similarity to the opposed emotion. Therefore, the final values for the dimensions range from 2 (complete affinity to the first emotion in a pair) to -2 (complete affinity to the second emotion in a pair).

To make an example: To create the first dimension of the Plutchik vector for the word “joyful”, the cosine similarity of the word to the joy emotion vector and the sadness emotion vector is evaluated. Then, the sadness cosine similarity is subtracted from the joy cosine similarity to create the value for the first dimension. The same procedure is applied to the three emotion pairs left. In this way a four-dimensional Plutchik vector is generated, which tells us, how much one emotion prevails over its opposite in a specific word.

To sum up, after the successful implementation process eight pairwise dissimilar emotion vectors were available. Based on these vectors, Plutchik vectors for 400000 words with corresponding Plutchik vectors were generated. The following chapter will take a closer look on the properties of the emotion vectors per se as well as the Plutchik vectors when tested within a system.

3 Evaluation

3.1 Emotion Vector Evaluation

In the previous chapter it was shown that the emotion vector optimization process was successful leading to dissimilar vector representations for opposed emotions. It is, however, not guaranteed that the optimization led to meaningful emotion vector representations from a semantic perspective. Therefore, before having a closer look at the performance of Plutchik vectors in a test system, this chapter will analyse some properties of the emotion vectors.

One way to test if the eight emotion vectors are also semantically meaningful is a nearest neighbours approach. As GloVe embeddings formed the building blocks of the emotion vectors, also the eight resulting emotion vectors are part of the vector space spanned by the embeddings. It is, thus, possible to calculate vector similarities between the eight generated emotion vectors and all word embeddings in the same vector space to evaluate which word embeddings are similar to the generated emotion vectors.

The following table shows the twenty nearest neighbours for each emotion vector:

	joy	**sadness**	**anger**	**fear**
Top 20 NNs	dessert: 0.56	grief: 0.65	rage: 0.73	anxiety: 0.75
	elegant: 0.54	grieving: 0.6	fury: 0.57	stress: 0.7
	dance: 0.53	anguish: 0.57	mob: 0.49	risk: 0.67
	wonderful: 0.53	victims: 0.57	revenge: 0.48	depression: 0.65
	tasting: 0.53	traumatized: 0.56	vengeful: 0.48	symptoms: 0.65
	splendid: 0.53	suffering: 0.55	hatred: 0.48	risks: 0.64
	elegance: 0.53	trauma: 0.54	murderous: 0.47	fatigue: 0.64
	breezy: 0.53	sorrow: 0.53	vengeance: 0.46	acute: 0.61
	perfect: 0.52	bereaved: 0.52	vigilante: 0.46	symptom: 0.61
	dazzling: 0.52	mourn: 0.52	rampage: 0.46	chronic: 0.61
	delicious: 0.51	survivors: 0.51	insults: 0.46	insomnia: 0.61
	musical: 0.5	distraught: 0.51	jealousy: 0.44	difficulty: 0.59
	fun: 0.5	relatives: 0.51	feud: 0.44	danger: 0.59
	pleasure: 0.5	traumatic: 0.5	defiance: 0.43	severe: 0.59
	delight: 0.5	wounds: 0.49	vendetta: 0.43	pain: 0.58
	repertory: 0.5	horrific: 0.49	enraged: 0.42	disorders: 0.58
	refreshing: 0.5	heal: 0.48	furious: 0.42	illness: 0.58
	dinner: 0.49	mourned: 0.48	savage: 0.42	headaches: 0.58
	presentation: 0.48	perished: 0.48	hate: 0.42	problems: 0.58
	superb: 0.48	painful: 0.48	violently: 0.41	effects: 0.57

	trust	**disgust**	**surprise**	**anticipation**
Top 20 NNs	trust: 0.74	vomit: 0.86	surprised: 0.87	anticipation: 0.89
	foundation: 0.6	excrement: 0.7	disappointed: 0.76	amid: 0.6
	institution: 0.59	feces: 0.68	pleased: 0.74	surge: 0.57
	partnership: 0.57	faeces: 0.61	shocked: 0.73	awaited: 0.56
	nonprofit: 0.56	saliva: 0.58	impressed: 0.68	easing: 0.55
	preservation: 0.56	urine: 0.57	delighted: 0.67	anticipated: 0.54
	establish: 0.55	flatulence: 0.55	stunned: 0.67	consolidation: 0.54
	established: 0.55	maggots: 0.55	amazed: 0.66	issuance: 0.53
	institutions: 0.54	urinate: 0.53	thrilled: 0.64	optimism: 0.53
	equality: 0.53	inhaling: 0.53	looked: 0.63	renewed: 0.53
	heritage: 0.53	entrails: 0.52	embarrassed: 0.63	anticipating: 0.53
	authority: 0.52	oily: 0.52	knew: 0.63	purchases: 0.52
	principles: 0.52	smeared: 0.52	liked: 0.62	cash: 0.52
	trusts: 0.52	smears: 0.5	noticed: 0.62	mounting: 0.52
	supports: 0.52	mucus: 0.5	obviously: 0.62	uncertainty: 0.52
	fund: 0.52	secretions: 0.5	convinced: 0.62	spurred: 0.52
	guarantee: 0.52	stains: 0.49	'm: 0.62	boosted: 0.51
	governance: 0.51	mislabeled: 0.49	astonished: 0.61	expectation: 0.51
	partners: 0.51	tasteless: 0.49	satisfied: 0.61	demand: 0.51
	commitment: 0.51	fingernails: 0.49	thought: 0.61	boost: 0.51

Table 6: Twenty nearest neighbours of optimized emotion vectors

As can be seen from table 6 the results look promising. Not only are the emotion vectors pairwise dissimilar, but they are also semantically highly meaningful. It seems that the optimizer was able to place the emotion vector representations into a vector subspace which captures the semantic field of each emotion, already mentioned in Chapter 2.3.1. This can be seen by the fact that the word lists do not only contain synonyms of the respective emotions but also entities which are conceptually related to an emotion: While the disgust vector is close to many body fluids commonly associated with disgust (“vomit”, “excrement”, “saliva”), the fear vector is most similar to fear-evoking concepts like “illness”, “pain” and “insomnia”. The joy vector, again, lies closest to concepts denoting a pleasant evening like “elegant”, “delicious”, “dance” or “fun”. Therefore, as is evident, the lists do not only comprise concepts related to the emotions by (near-)synonymy but also events and objects triggering a certain emotion.

From these results can be inferred that the chosen approach for vector generation was indeed able to find emotion vectors which denote a full emotional concept and, thus, fulfil all requirements following from Plutchik’s theory. This, however, still leaves open the question if they perform well as a basis for creation of the four-dimensional Plutchik vectors. As in case of the Plutchik vectors we want to access the prevalence of one emotion over the other on word level, a good measure of quality is if emotion typical words are more distinct, i.e. have a higher similarity discrepancy in terms of opposed emotions. “Higher” here is defined in reference to the simple average emotion vectors, which were taken as a basis for the optimization process but showed a high mutual similarity (see Chapter 2.3.1). To make an example: The word embedding of a word like “funeral” should have a high similarity to the optimized sadness vector and a low similarity to the optimized joy vector, such that the affinity to one or the other emotion is more distinct than in case of the non-optimized average emotion vectors. This was tested for several sets of words. For reasons of conciseness, in the following only one word per emotion pair is presented:

Emotion Pair	**joy vs. sadness**			**anger vs. fear**		
Tested word	“funeral”			“rage”		
	Similarity joy	Similarity sadness	Absolute Distance	Similarity anger	Similarity fear	Absolute Distance
Non-optimized	0.44	0.44	**0**	0.5	0.49	**0.01**
Optimized	0.07	0.39	**0.31**	0.73	0.09	**0.62**
Emotion Pair	**trust vs. disgust**			**surprise vs. anticipation**		
Tested word	“vomit”			“prophecy”		
	Similarity trust	Similarity disgust	Absolute Distance	Similarity surprise	Similarity anticipation	Absolute Distance
Non-optimized	0.02	0.18	**0.16**	0.16	0.15	**0.01**
Optimized	-0.46	0.86	**1.32**	-0.03	0.09	**0.12**

Table 7: Cosine similarity distances for words to optimized emotion vectors

As is evident from Table 7 the optimized emotion vectors show the desired behaviour: The absolute similarity distance in case of the optimized emotion vectors is always higher. While in case of “funeral” the similarity to the non-optimized and sadness vector is equal, sadness clearly prevails in the optimized case. The same applies to “rage” and “prophecy”, where the prevalence of the intuitively more related emotion (anger in case of “rage”; anticipation in case of “prophecy”) is also more distinct in the optimized case. In case of “vomit”, checked for affinity to trust and disgust, the dominance of disgust over trust is already present in the non-optimized case. This dominance is additionally amplified in case of the optimized emotion vectors.

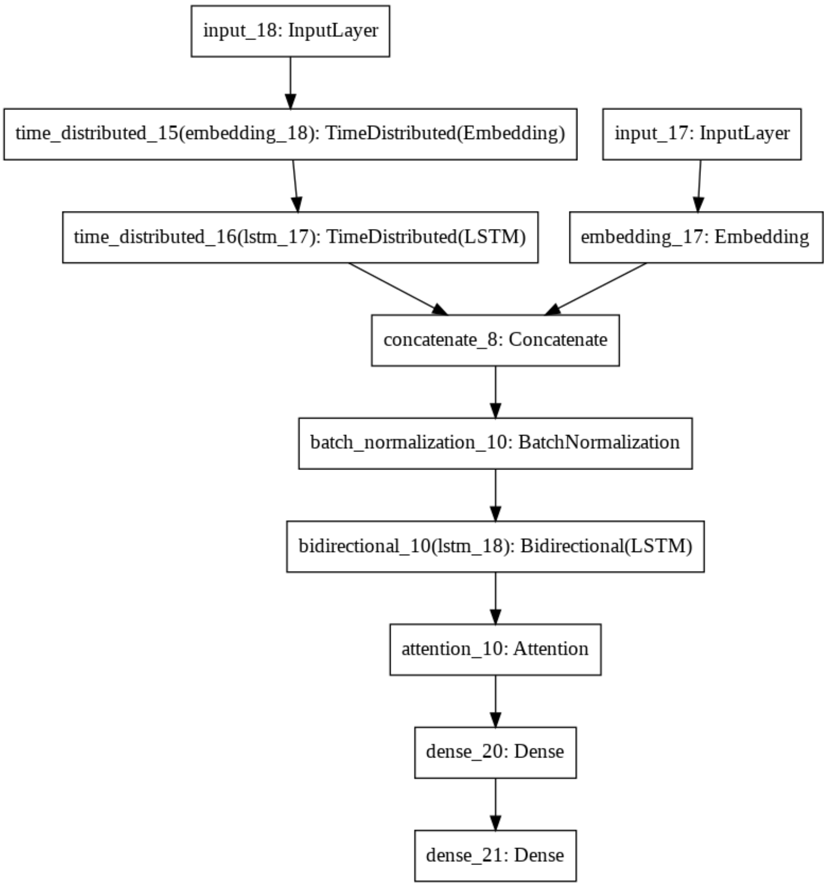
The present chapter showed that the implementation of emotion vectors following Plutchik’s theory was successful and that these emotion vectors formed a solid basis for the generation of the four-dimensional word level Plutchik vectors.

The next chapter will investigate the performance of the generated Plutchik vectors within a machine learning system.

3.2 Plutchik Vector Evaluation

3.2.1 Testing System

The performance of the generated four-dimensional Plutchik vectors was tested by using them as an input to a neural net for emotion classification. The neural net had the following architecture:


input_18: InputLayer
time_distributed_15(embedding_18): TimeDistributed(Embedding)
input_17: InputLayer
time_distributed_16(lstm_17): TimeDistributed(LSTM)
embedding_17: Embedding
concatenate_8: Concatenate
batch_normalization_10: BatchNormalization
bidirectional_10(lstm_18): Bidirectional(LSTM)
attention_10: Attention
dense_20: Dense
dense_21: Dense

The net has two input layers: input_18 represents an input layer for character IDs, input_17 an input layer for word IDs. The character IDs are sent to a character embedding layer, which assigns each character its corresponding embedding vector[6]. These character embeddings are then combined by a Bi-LSTM to a word embedding representation on character basis. This character loop was added to account for potentially unknown words in the test set, which would otherwise have a mere zero vector representation.

The word input, on the other hand, is directly sent to an embedding layer. Concatenate_8 concatenates the character embedding and the word embedding to one long vector, which is then used as an input for a Bi-LSTM (bidirectional_10). The Bi-LSTM processes all words of the given input sentence, outputting all hidden states. This leads to a matrix representation of a sentence with the dimensionality *words x per vector dimension*. This sentence matrix is then given to a self-attention layer which builds a sentence vector representation. The sentence

[6] This was possible, because the GloVe embeddings contained also embedding representations for single characters.

vector representation is sent through a dense layer before class probabilities are computed in the output layer dense_21 with softmax activation.

3.2.1 Corpora and Classes

For testing two different corpora were used, both drawn from the unified dataset mentioned in Chapter 2.3.1:

The "AffectiveText" corpus, published by Strapparava & Mihalcea (2007), contains 1250 news headlines which are single-labelled with the six basic emotions established by Ekman. The label distribution over instances is as follows[7].

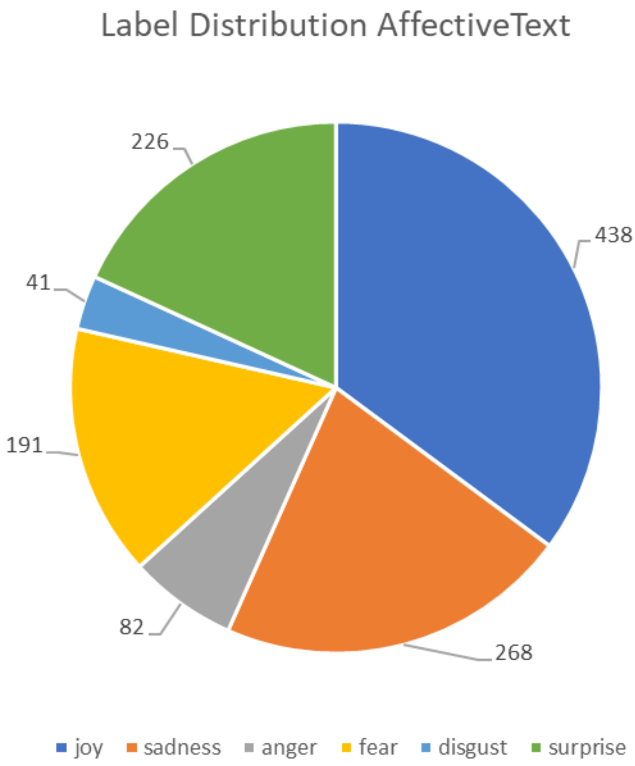


Figure 4: Label Distribution in AffectiveText corpus

As can be seen, the distribution of labels shows a prevalence for instances labelled with "joy", while "disgust" and "anger" labels are only sparsely represented.

The second corpus which was used for testing was the "Electoral-Tweets" corpus already described in Chapter 2.31. In contrast to the AffectiveText corpus the label set of Electoral-Tweets was a non-standard set, also containing, for example, a "confusion" label. The corpus instances labelled with such non-standard labels were excluded, such that only 4055 instances single-labelled with the eight Plutchik emotions remained.

[7] Four instances were excluded due to missing labels.

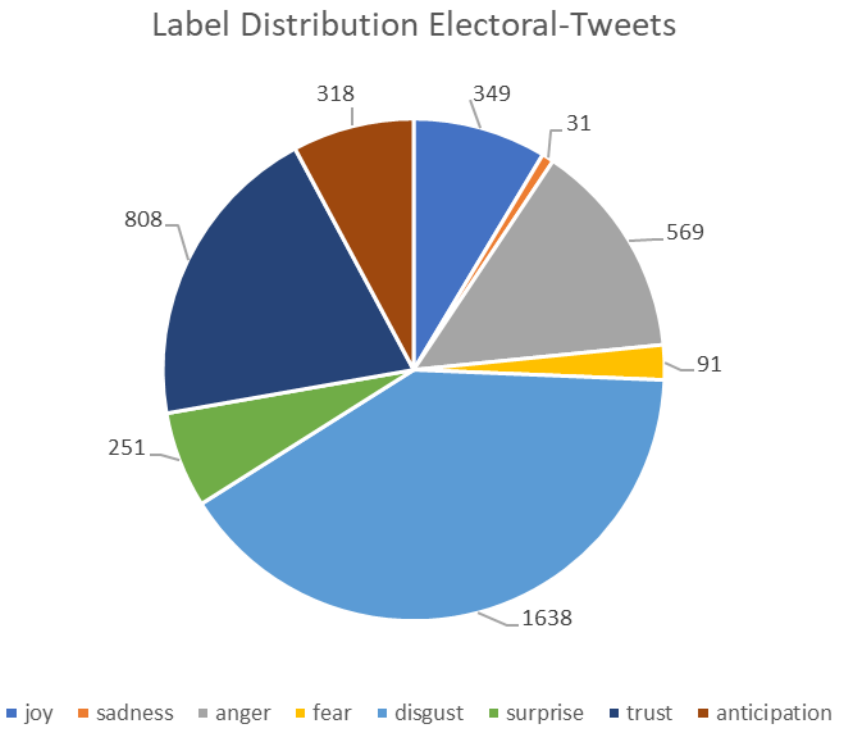


Figure 5: Label Distribution in Electoral-Tweets corpus

As can be seen from the pie chart, “disgust” was the dominant emotion in the resulting dataset, followed by “trust” and “anger”. Overall, the label distribution was relatively jagged.

The two described corpora were chosen, because of their differing label set: AffectiveText is based on the six Ekman’s emotion, while Electoral-Tweets is labelled with the two additional classes “trust” and “anticipation” corresponding, thus, to Plutchik’s label set. This provides the opportunity to test system performance not only on a different corpus and label distribution, but also on a different label set. As creation of Plutchik vectors was inspired by Plutchik’s assumptions, especially for the Electoral-Tweets corpus a performance improvement was expected.

3.2.2 Input Vectors and Training Process

If we speak of improvement of performance, it is important to specify the system, which is taken as a reference. In case of this project, as should have become clear, it is not the model type or architecture which differs, but the model input. Three types of neural net inputs were tested and their classification performance compared:

- 100-dimensional GloVe embeddings
- 4-dimensional Plutchik vectors
- 4-dimensional PCA reduced GloVe embeddings

The decision to perform a Principle Component Analysis (PCA) on GloVe embeddings reducing their dimensionality to the same dimensionality as the Plutchik vectors was based on the consideration, that a potentially better performance of the GloVe embeddings could be merely due to their higher dimensionality.

The resulting nets had the following type and number of parameters:

	GloVe Net	**Plutchik Net**	**GloVe PCA Net**
Trainable parameters	348024	3948	3948
Non-trainable parameters	347300	13892	13892
Total	695324	17840	17840

Table 8: Number of different parameters in Neural Nets

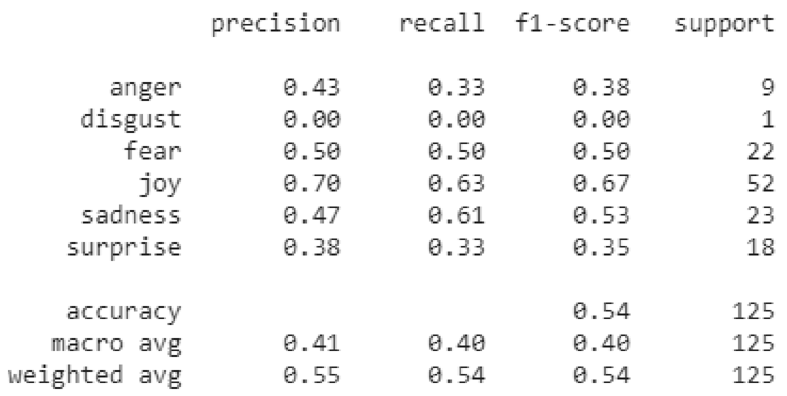
Because of the identical dimensionality of their inputs the Plutchik and GloVe PCA nets had the same number of trainable and non-trainable parameters. The non-trainable parameters mostly derived from the embedding layers, which were set non-trainable. The reason for that was that it was assumed that leaving them trainable would enhance GloVe system performance more than the performance of the Plutchik net, simply because of the higher dimensionality of the GloVe embeddings.

For training the AffectiveText and Electoral-Tweets corpora were split in training and test sets in the ratio 9:1. To test performance an ensemble model approach was chosen: Each net was trained with five different initializations for AffectiveText and three different initialization points in case of Electoral-Tweets. Each of these differently initialized nets was trained for 200 epochs in case of AffectiveText and 50 epochs in case of Electoral-Tweets. The lower number of initializations and epochs in case of Electoral-Tweets was due to the higher number of instances that had to be processed. The predicted class probabilities on the test set were averaged over the differently initialized models to generate the final prediction. The following chapter will report on the results given the test set predictions.

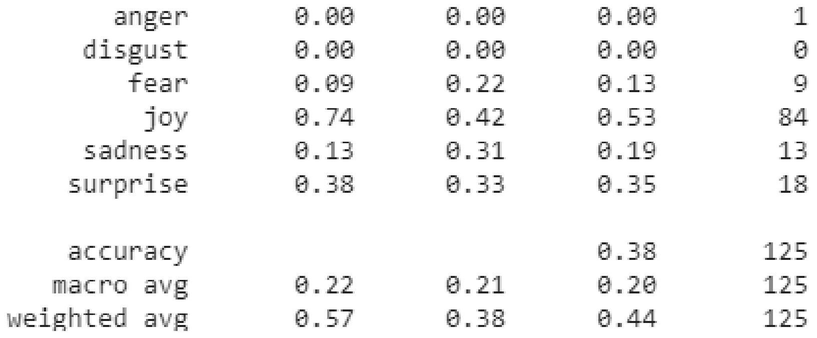
3.2.3 Results

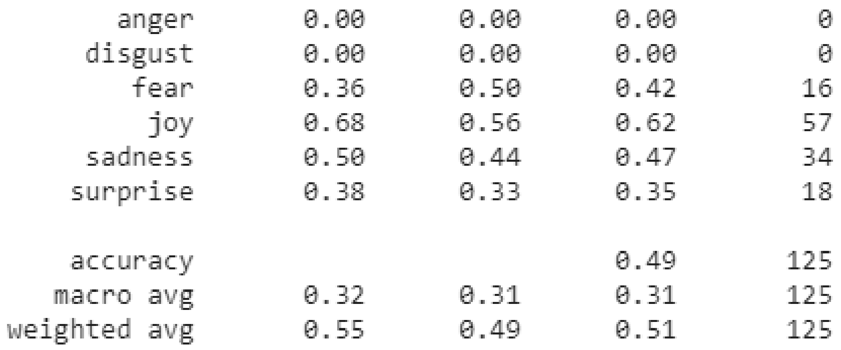
A common score to evaluate performance on multi-class classification tasks is the F1-score, the harmonic mean of precision and recall. In terms of F1 it is possible to compute a macro average and a micro average over classes. The micro average weights more frequent classes higher than more infrequent classes, while the macro average assigns the same weight to each class. Both of these metrics have their drawbacks: As all classes are weighted equally in case of macro average, the F1-score can still be high, even though a lot of instances might be wrongly classified. This happens if classification works on low frequent classes, but fails with classes of high frequency. Micro average does not suffer from this problem. On the other hand, the issue with micro average is that a classifier, which simply overfits to the most frequent classes, might still achieve a high micro average if class distribution is very jagged. As class distribution is not balanced in our corpora, the performance of systems is compared based on both values.

The following table shows the classification performance on the AffectiveText test set:

<table>
<tr><th></th><th>Performance on AffectiveText test set</th></tr>
<tr><td><em>GloVe Net</em></td><td>
<pre>
              precision   recall f1-score   support

       anger   0.43   0.33   0.38   9
     disgust   0.00   0.00   0.00   1
        fear   0.50   0.50   0.50   22
         joy   0.70   0.63   0.67   52
     sadness   0.47   0.61   0.53   23
    surprise   0.38   0.33   0.35   18

    accuracy   0.54   125
   macro avg   0.41   0.40   0.40   125
weighted avg   0.55   0.54   0.54   125
</pre></td></tr>
<tr><td><em>GloVe PCA Net</em></td><td>
<pre>
       anger   0.00   0.00   0.00   1
     disgust   0.00   0.00   0.00   0
        fear   0.09   0.22   0.13   9
         joy   0.74   0.42   0.53   84
     sadness   0.13   0.31   0.19   13
    surprise   0.38   0.33   0.35   18

    accuracy   0.38   125
   macro avg   0.22   0.21   0.20   125
weighted avg   0.57   0.38   0.44   125
</pre></td></tr>
<tr><td><em>Plutchik Net</em></td><td>
<pre>
       anger   0.00   0.00   0.00   0
     disgust   0.00   0.00   0.00   0
        fear   0.36   0.50   0.42   16
         joy   0.68   0.56   0.62   57
     sadness   0.50   0.44   0.47   34
    surprise   0.38   0.33   0.35   18

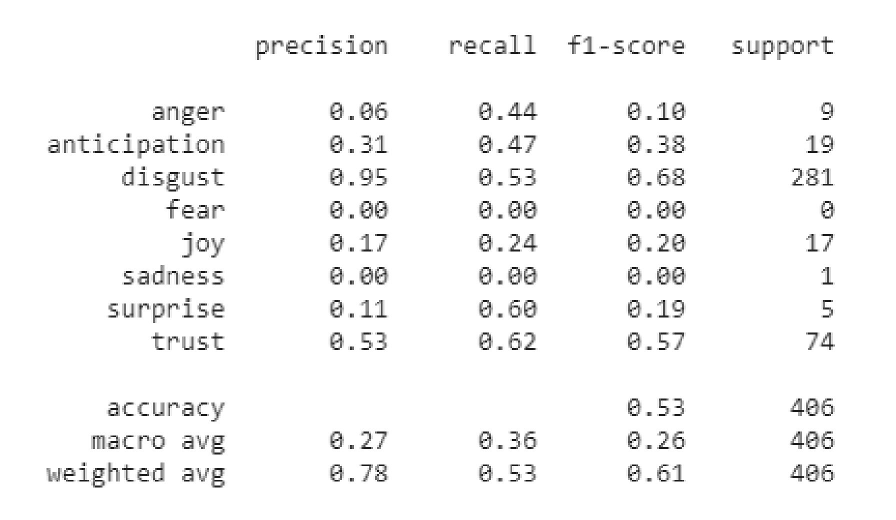
    accuracy   0.49   125
   macro avg   0.32   0.31   0.31   125
weighted avg   0.55   0.49   0.51   125
</pre></td></tr>
</table>

Table 9: Classification performance AffectiveText

As the table shows, the performance of the Plutchik Net with four-dimensional Plutchik vectors as input is worse compared to the GloVe Net both in terms of micro-average (here equivalent to accuracy; see sci-kit documentation for specifications) and macro-average. Especially the macro-average is much higher as the Plutchik Net fails to detect any instances of “surprise”. The results of the Plutchik Net are, however, not fully disappointing. Even though the GloVe Net had 88 times more trainable parameters, the classification performance of the Plutchik Net measured by F1 micro-average was still substantial and lies only five points below the GloVe Net performance. These results are especially impressive if we compare them with the performance of the GloVe PCA Net. Broken down to four dimensions the GloVe embeddings perform significantly worse than the Plutchik vectors.

The results on the Electoral-Tweets corpus, however, were less convincing:

<table>
<tr><th></th><th>Performance on Electoral-Tweets test set</th></tr>
<tr><td><em>GloVe Net</em></td><td>

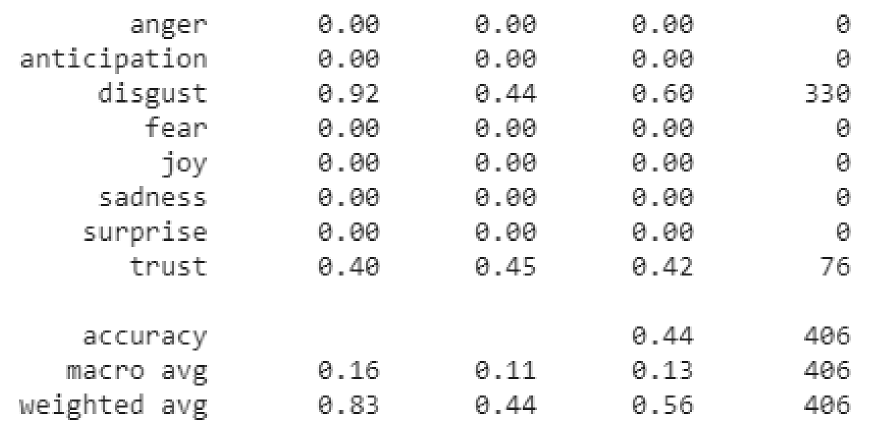


```
              precision    recall  f1-score   support

       anger       0.06      0.44      0.10         9
anticipation       0.31      0.47      0.38        19
     disgust       0.95      0.53      0.68       281
        fear       0.00      0.00      0.00         0
         joy       0.17      0.24      0.20        17
     sadness       0.00      0.00      0.00         1
    surprise       0.11      0.60      0.19         5
       trust       0.53      0.62      0.57        74

    accuracy                           0.53       406
   macro avg       0.27      0.36      0.26       406
weighted avg       0.78      0.53      0.61       406
```

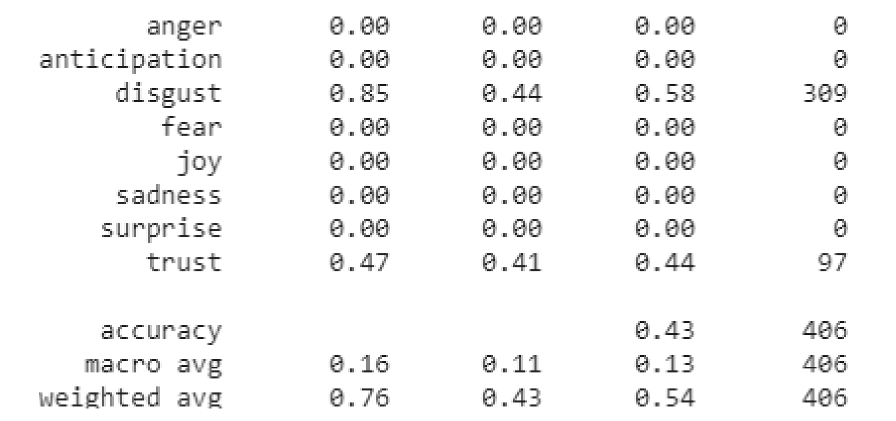
</td></tr>
<tr><td><em>GloVe PCA Net</em></td><td>



```
       anger       0.00      0.00      0.00         0
anticipation       0.00      0.00      0.00         0
     disgust       0.92      0.44      0.60       330
        fear       0.00      0.00      0.00         0
         joy       0.00      0.00      0.00         0
     sadness       0.00      0.00      0.00         0
    surprise       0.00      0.00      0.00         0
       trust       0.40      0.45      0.42        76

    accuracy                           0.44       406
   macro avg       0.16      0.11      0.13       406
weighted avg       0.83      0.44      0.56       406
```

</td></tr>
<tr><td><em>Plutchik Net</em></td><td>



```
       anger       0.00      0.00      0.00         0
anticipation       0.00      0.00      0.00         0
     disgust       0.85      0.44      0.58       309
        fear       0.00      0.00      0.00         0
         joy       0.00      0.00      0.00         0
     sadness       0.00      0.00      0.00         0
    surprise       0.00      0.00      0.00         0
       trust       0.47      0.41      0.44        97

    accuracy                           0.43       406
   macro avg       0.16      0.11      0.13       406
weighted avg       0.76      0.43      0.54       406
```

</td></tr>
</table>

Table 10: Classification performance Electoral-Tweets

In case of the Electoral-Tweets corpus classification performance of the Plutchik Net was significantly worse, with micro as well as macro average being approximately ten points below the GloVe Net. A closer inspection of Table 10 reveals that the Plutchik Net overfits to the most frequent classes “disgust” and “trust”. As these labels are very frequent in the dataset (see Chapter 3.2.1) and, thus, also in the test set, they are the only classes for which classification is learnt. Interestingly, also the GloVe PCA Net showed the same behaviour reaching the same classification performance.

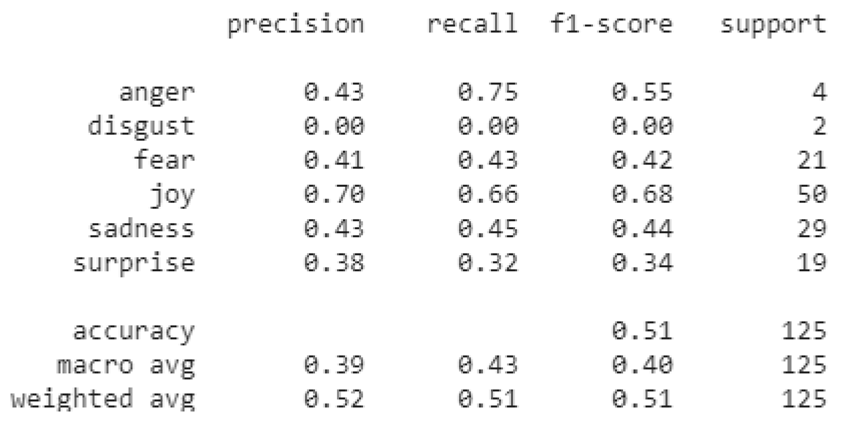
As the Plutchik Net showed a better performance than the GloVe PCA Net in the case of the first corpus with six classes, but not in case of the second corpus with eight classes, it was assumed that four dimensions might not be sufficient to predict eight classes, especially if class distribution is so imbalanced. For this reason, the Plutchik vectors were slightly adapted, i.e. extended to twelve dimensions: This was done by adding to the vectors also the similarities of the given words to the eight emotion vectors. A Plutchik vector, thus, now comprised the following twelve similarity dimensions (each emotion term in the table should be read as “similarity to emotion term”):

joy	sadness	joy – sadness	anger	fear	anger – fear	trust	disgust	trust – disgust	surprise	anticipation	surprise – anticipation

Table 11: Twelve-dimensional Plutchik vector design

Using these new, twelve-dimensional Plutchik vectors the number of trainable parameters in the Plutchik Net increased from 3948 to 26119. What’s more important, however, is that also the performance of the Plutchik Net increased for both corpora.

Performance Plutchik Net: AffectiveText

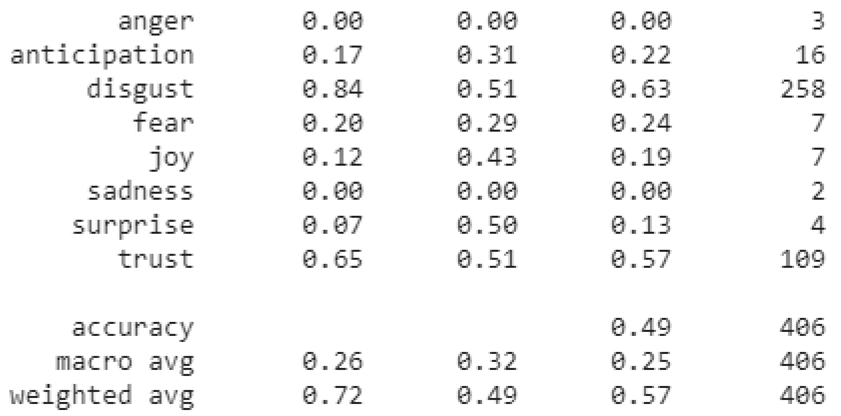


```
              precision    recall  f1-score   support

       anger       0.43      0.75      0.55         4
     disgust       0.00      0.00      0.00         2
        fear       0.41      0.43      0.42        21
         joy       0.70      0.66      0.68        50
     sadness       0.43      0.45      0.44        29
    surprise       0.38      0.32      0.34        19

    accuracy                           0.51       125
   macro avg       0.39      0.43      0.40       125
weighted avg       0.52      0.51      0.51       125
```

Performance Plutchik Net: Electoral-Tweets



```
       anger       0.00      0.00      0.00         3
anticipation       0.17      0.31      0.22        16
     disgust       0.84      0.51      0.63       258
        fear       0.20      0.29      0.24         7
         joy       0.12      0.43      0.19         7
     sadness       0.00      0.00      0.00         2
    surprise       0.07      0.50      0.13         4
       trust       0.65      0.51      0.57       109

    accuracy                           0.49       406
   macro avg       0.26      0.32      0.25       406
weighted avg       0.72      0.49      0.57       406
```

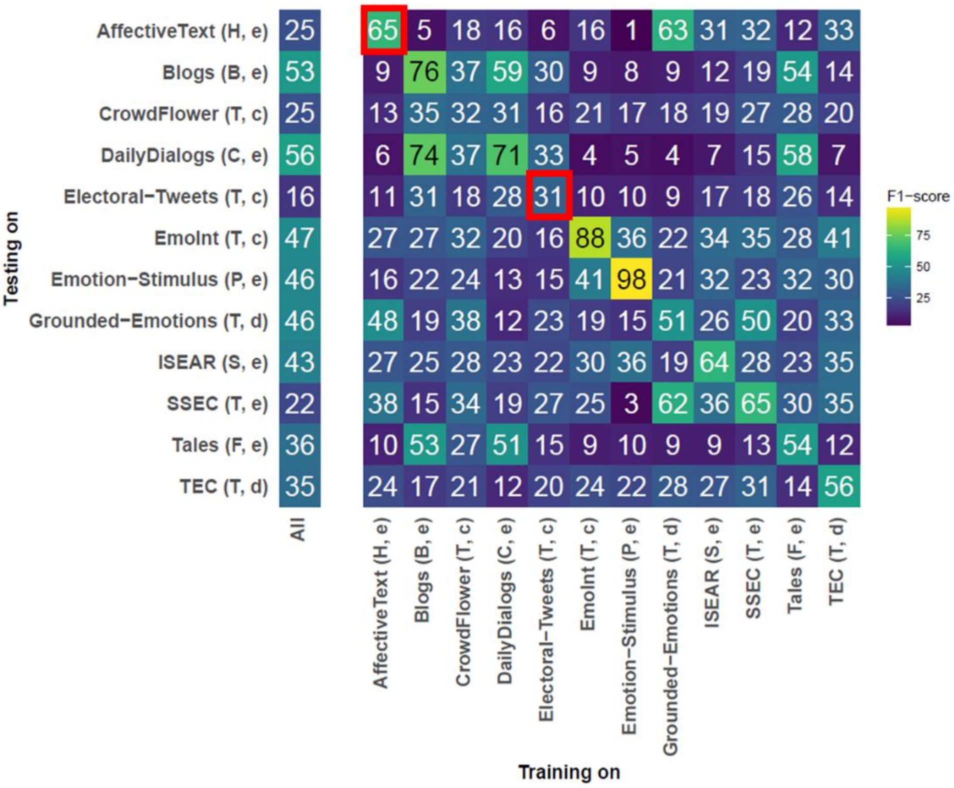
Table 12: Performance of 12-dimensional Plutchik vectors on AffectiveText and Electoral-Tweets

In case of AffectiveText the Plutchik Net with twelve-dimensional input reached a macro average score of 0.4, which is the same score as the GloVe Net achieved, even though the latter had a lot more parameters. The micro-average of 0.51 is only three points below the GloVe Net micro-average of 0.54. On "joy" the Plutchik Net even reached a higher F1 score than the GloVe net. In case of the Electoral-Tweets corpus the performance of the Plutchik Net was still worse than performance of the GloVe Net. Nevertheless, macro average nearly doubled in reference to the four-dimensional Plutchik vectors and was only one point lower than in case of the GloVe Net. In case of "fear" the Plutchik Net even reached a slightly higher F1 score.

3.3 Discussion of Results

The presented results have shown that the present project partially reached the goals specified in the introduction. The eight generated emotion vectors were most similar to terms closely related to the respective emotion. Moreover, these emotion vectors were opposed to their counterparts as defined in Plutchik's theory. The present project work, thus, could solve the problem of concepts being very similar to their opposites in a vector embedding space. From this expressiveness of the emotion vectors also follows that the dimensions of the Plutchik vectors are to some degree interpretable: By specifying the similarity between the word embedding and an emotion vector it is quantified how much of each emotion a word contains. I write "to some degree", because even though the emotion vectors seem to work well, they are still only a rough approximation of the actual position of the emotional concept in the vector embeddings space. This is shown by the fact that also function words, which are emotionally neutral, using the described technique got relatively high emotion scores. The definite article "the", for example, showed a similarity of 0.45 to the anticipation emotion vector, while the plural noun "anticipations", had a cosine similarity of 0. Another weakness is that for each word its similarity to each single emotion is determined, even though barely any word is specified in terms of all emotions. A word like "cake" might, for example, have a "joy", "disgust" and "surprise" component, but it is hard to imagine its connection with "trust". Ideally, in these cases and in terms of function words a neutral similarity of 0 should have been assigned. This, however, was not the case, which led to the fact that the Plutchik word vectors are emotionally "over-specified". This limits their interpretability and meaningfulness.

The second goal listed in the introduction was to achieve a performance improvement or at least comparable performance. Compared with the 100-dimensional GloVe vectors no increase in performance could be reached. Especially when fed with the four-dimensional Plutchik vectors the implemented neural net even performed significantly worse. The twelve-dimensional Plutchik vectors were, however, able to come close to the subsidiary goal of comparable performance. And most notably, they reached (nearly) the same macro F1-score and a comparable micro F1-score with a significantly lower dimensionality. But how do these scores relate to the general performance of systems in emotion classification tasks? To answer this question it is worth to take a look at the following table taken from Bostan & Klinger (2018: 2114):



	All	AffectiveText (H, e)	Blogs (B, e)	CrowdFlower (T, c)	DailyDialogs (C, e)	Electoral−Tweets (T, c)	EmoInt (T, c)	Emotion−Stimulus (P, e)	Grounded−Emotions (T, d)	ISEAR (S, e)	SSEC (T, e)	Tales (F, e)	TEC (T, d)
AffectiveText (H, e)	25	65	5	18	16	6	16	1	63	31	32	12	33
Blogs (B, e)	53	9	76	37	59	30	9	8	9	12	19	54	14
CrowdFlower (T, c)	25	13	35	32	31	16	21	17	18	19	27	28	20
DailyDialogs (C, e)	56	6	74	37	71	33	4	5	4	7	15	58	7
Electoral−Tweets (T, c)	16	11	31	18	28	31	10	10	9	17	18	26	14
EmoInt (T, c)	47	27	27	32	20	16	88	36	22	34	35	28	41
Emotion−Stimulus (P, e)	46	16	22	24	13	15	41	98	21	32	23	32	30
Grounded−Emotions (T, d)	46	48	19	38	12	23	19	15	51	26	50	20	33
ISEAR (S, e)	43	27	25	28	23	22	30	36	19	64	28	23	35
SSEC (T, e)	22	38	15	34	19	27	25	3	62	36	65	30	35
Tales (F, e)	36	10	53	27	51	15	9	10	9	9	13	54	12
TEC (T, d)	35	24	17	21	12	20	24	22	28	27	31	14	56

Figure 6: F1-score for emotion classification based on different combinations of train and test corpora

This confusion matrix shows micro-averaged F1 results for training and testing on different combinations of corpora. The scores which are relevant for the present project are framed red. In the context of the present project only combinations of train and test set deriving from the same corpus were tested. Bostan & Klinger (2018) used a Maximum Entropy classifier with BOW-features as input. As can be seen from the table they achieved a micro F1-score of 0.65 for AffectiveText and 0.31 for the Electoral-Tweets corpus. These results deviate from the results, which were achieved with the Plutchik Net. While the performance of the Plutchik Net (twelve-dimensions) was only at 0.51 on the AffectiveText corpus, i.e. 14 points below the performance of the ME classifier, it showed an 18 points higher performance in case of the Electoral Tweets corpus. It must remain unanswered how this performance variance across models might be explained.

Nevertheless, the results show that the performance of the Plutchik Net is not only comparable to the baseline used as a reference in this project but might also able to keep up with other systems. Further testing on other corpora and comparisons with other systems are necessary to corroborate this statement. The following chapter will briefly address possible improvements of the system and further promising directions.

4 Final Remarks and Outlook

The discussion of results in the previous chapter has shown that the goals in terms of interpretability and performance, which were formulated in the introduction could be partially reached. It became, however, also clear that the presented implementation of Plutchik vectors has still weaknesses. One weakness, the emotional “overspecification” of the Plutchik vectors, has already been pointed out: The generated emotion vectors taken as a reference point to evaluate emotional content on word level are not perfect. Even though they work for many content words, they fail to assign neutrality to function words. In the previous chapter it was assumed that such a neutrality might be expressed by a cosine similarity of 0 to any emotion vector. Neutrality, thus, would be expressed as being neither especially similar nor dissimilar to any emotion. It is, however, questionable, if this is a productive way of defining neutrality and, if not, how we could assess emotional neutrality otherwise.

Considering these limitations, it might be asked if Plutchik vector representations on word-level are the most sensible way to guarantee interpretability: In these terms, another approach which could lead to interesting results is to directly use the generated emotion vectors in a neural net. By doing so, a vector sentence representation could be compared to each emotion vector predicting the emotion with the most similar vector representation. In this way the classification decision could be more easily interpretable from a human perspective.

In general, however, interpretability is not a trivial matter. Even if we would have Plutchik vectors, which are perfectly interpretable and humanly intuitive it is still not guaranteed that a neural net interprets and uses dimensions in a human-like manner. An interesting question in these regards would be if a twelve- or four- dimensional sentence representation generated from Plutchik word vectors would also correspond to the distribution and intensity of emotions expected by humans. And if so, is this sentence correctly classified in these cases? This was not checked in the present project and must be addressed in future tests.

When it comes to testing, also other interesting questions are still open: How is performance of Plutchik vectors in cross-corpus scenarios, i.e. training and testing the system on different corpora as done by Bostan & Klinger (2018)? Would the system profit from setting the Plutchik embedding layers to trainable and would this lead to Plutchik vectors, which are still interpretable?

Other interesting questions concern different systems and domains: Would Plutchik vectors lead to a better performance in more transparent systems like Random Forest Classifiers? Would the features contribute to class decisions as expected, such that, for example, a sentence with a high value for “joy” would be labelled with “joy”? And are Plutchik vectors, which were explicitly designed for emotion classification, also applicable to other classification tasks?

All these questions have to remain unanswered and point to interesting further research directions in future work.

5 References

Barrett, Lisa F. 2017. The theory of constructed emotion: an active inference account of interoception and categorization. *Social cognitive and affective neuroscience* 12(1). 1–23.

Bostan, Laura-Ana-Maria & Roman Klinger. 2018. An analysis of annotated corpora for emotion classification in text. 2104–2119.

Cohn, Jeffrey, Zara Ambadar & Paul Ekman. 2007. Observer-Based Measurement of Facial Expression with the Facial Action Coding System. *The Handbook of Emotion Elicitation and Assessment*.

Ekman, Paul. 1972. Universals and Cultural Differences in Facial Expressions of Emotion. In *Nebraska Symposium on Motivation*: *Cole, J.* Lincoln, NE: University of Nebraska Press.

Heppler, Jason. What is Digital Humanities? http://whatisdigitalhumanities.com.

James, William. 1884. What is an Emotion? *Mind* (34). 188–205.

Kim, Evgeny, Sebastian Padó & Roman Klinger. 2017. Investigating the relationship between literary genres and emotional plot development. 17–26.

Labatut, Vincent & Xavier Bost. 2019. Extraction and Analysis of Fictional Character Networks: A Survey. *ACM Computing Surveys (CSUR)* 52(5). 1–40.

Lange, Carl. 1887. *Über Gemütsbewegungen*: *Ihr Wesen und ihr Einfluß auf körperliche, besonders auf krankhafte Lebenserscheinungen. Ein medizinisch-psychologische Studie.* Leipzig: Thomas.

Lee, Yong-Bae & Sung H. Myaeng. 2002. Text genre classification with genre-revealing and subject-revealing features. *ACM Special Interest Group on Information Retrieval*. 145.

Mikolov, Tomas, Kai Chen, Greg Corrado & Jeffrey Dean. 2013. *Efficient Estimation of Word Representations in Vector Space*.

Mohammad, Saif M. 2012. # Emotional tweets. 246–255.

Mohammad, Saif M., Xiaodan Zhu, Svetlana Kiritchenko & Joel Martin. 2015. Sentiment, emotion, purpose, and style in electoral tweets. *Information Processing & Management* 51(4). 480–499.

Nielek, Radoslaw, Miroslaw Ciastek & Wiesław Kopeć. 2017. Emotions make cities live: towards mapping emotions of older adults on urban space. 1076–1079.

Pennington, Jeffrey, Richard Socher & Christopher D. Manning. 2014. Glove: Global vectors for word representation. 1532–1543.

Plutchik, Robert. 1980. *Emotion*: *Theory, research, and experience* (Theories of emotion 1). New York: Academic.

Powell, M.J.D. 1970. A New Algorithm for Unconstrained Optimization. In J. B. Rosen, O. L. Mangasarian & K. Ritter (eds.), *Nonlinear Programming*: *Proceedings of a Symposium Conducted by the Mathematics Research Center, the University of Wisconsin, Madison, May*

4-6, 1970 (Publication no. 25 of the Mathematics Research Center, The University of Wisconsin), 31–65. Burlington: Elsevier Science.

Reagan, Andrew J., Lewis Mitchell, Dilan Kiley, Christopher M. Danforth & Peter S. Dodds. 2016. The emotional arcs of stories are dominated by six basic shapes. *EPJ Data Science* 5(1). 31.

Scherer, Klaus R. 2005. What are emotions? And how can they be measured? *Social Science Information* 44(4). 695–729.

Schuff, Hendrik, Jeremy Barnes, Julian Mohme, Sebastian Padó & Roman Klinger. 2017. Annotation, modelling and analysis of fine-grained emotions on a stance and sentiment detection corpus. 13–23.

Stamatatos, Efstathios. 2006. Ensemble-based author identification using character n-grams. 41–46.

Strapparava, Carlo & Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. 70–74.

Troiano, Enrica, Sebastian Padó & Roman Klinger. 2019. Crowdsourcing and Validating Event-focused Emotion Corpora for German and English. *arXiv preprint arXiv:1905.13618*.

6 List of Tables

7 List of Figures