Emotion Analysis – Assignment 2
Automated Emotion Classification

Felix Bühler, Maximilian Wegge, Carlotta Quensel

1 Feature-based ML

For our first task, we wanted to train a model that would be able to annotate the corpus we created for assignment 1. With this as the basis for our research hypothesis, we selected the data source and method.

Can our own corpus be automatically annotated by a ML algorithm that is trained on another dataset? How does the performance differ from our annotation?

Fig 1 Research question of the first task.

1.1 Task Specification and Corpus Choice

As our goal is to evaluate the classifier on our own corpus, the task needed to follow certain rules. Firstly, the method had to be ML-based, as a dictionary or OOC would not need training. Secondly, the data and method must align with the structure of our data. Therefore, we chose the SSEC corpus as our training data, as it uses Plutchik's eight base emotions for a multi-label annotation. The SSEC is tagged with only 1 or 0 to indicate if an emotion holds, which opposes our valence scale of 0 to 3. This is however not a problem after extracting the accumulated annotation between all annotators. As the κ-score for our agreement is too low to yield enough common valence annotation, the accumulated annotation is changed to a 1/0 annotation as in the SSEC. The task then is to find and train a ML-method that is suitable for multi-label classification.

1.2 Method

As stated above, the research question confines us to a feature-based or deep learning approach. As the tagged emotions in our corpus do not co-occur as simultaneously felt but rather as emitted by separate parts of the texts, we decided on Naïve Bayes as a feature-based ML algorithm. As part of python's NLTK library, Naïve Bayes is quick and easy to implement and to adjust to our needs as seen in the next step.

1.3 Implementation

Before training the classifier, several steps were taken to ensure that the learned features are as informative as possible.

1.3.1 Preprocessing

After extracting the raw text and labels of all documents, the text is first tokenized. These tokens are then given to NLTK's implementation of the Lancaster stemmer to normalize word endings and reduce the vocabulary size. To further reduce the number of words and to skip the most frequent and thus least informative words while training, stopwords were removed. The words were once more taken from NLTK and expanded to all frequent punctuation. This also includes 'twitter specific' symbols such as <#> and <@>. The following figure shows the process.

After the documents have the form of word lists, they are converted into word vectors, whose underlying vocabulary are all the terms of the training data. These vectors can then be given to the Naïve Bayes algorithm to use as feature sets for the classification task.

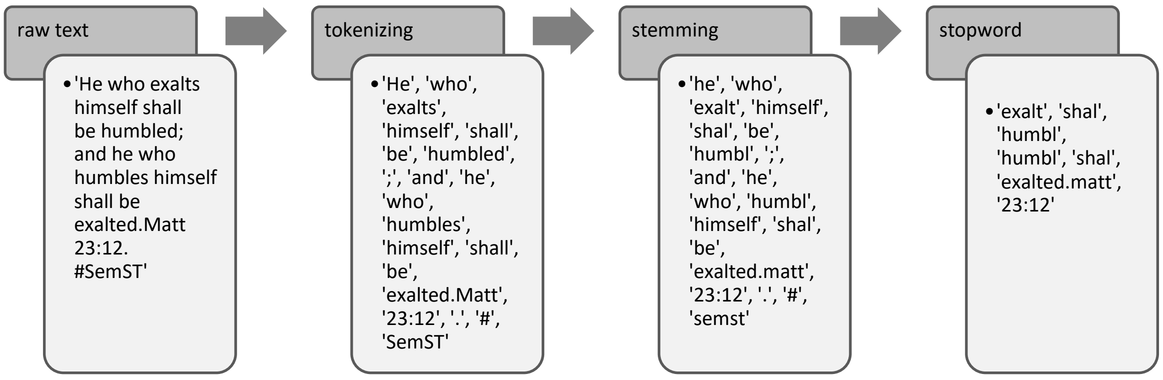

raw text
• 'He who exalts himself shall be humbled; and he who humbles himself shall be exalted.Matt 23:12. #SemST'
tokenizing
• 'He', 'who', 'exalts', 'himself', 'shall', 'be', 'humbled', ';', 'and', 'he', 'who', 'humbles', 'himself', 'shall', 'be', 'exalted.Matt', '23:12', '.', '#', 'SemST'
stemming
• 'he', 'who', 'exalt', 'himself', 'shal', 'be', 'humbl', ';', 'and', 'he', 'who', 'humbl', 'himself', 'shal', 'be', 'exalted.matt', '23:12', '.', '#', 'semst'
stopword
• 'exalt', 'shal', 'humbl', 'humbl', 'shal', 'exalted.matt', '23:12'

Fig 2 Stepwise depiction of the preprocessing

1.3.2 Naïve Bayes classifier

The Bayes method is used for single label classification, which is why we adapt it for the task. The classifier consists of eight separate classifiers, each of which is trained to annotate one of the eight emotion classes. For training, the feature sets are paired with the 1/0 label of each emotion class and every classifier is trained independently from the others. During the training, the classifier learns the probabilities of the label when observing certain features, the most informative of which are shown in figure 3.

joy:	**trust:**	**anger:**	**surprise:**
lovewin → 23.0 : 1	sham → 1 : 12.1	sin → 1 : 20.4	wow → 32.6 : 1
amaz → 18.7 : 1	bring → 9.9 : 1	fuck → 13.7 : 1	sud → 13.7 : 1
thank → 16.2 : 1	degr → 1 : 9.6	hat → 13.2 : 1	assum → 11.6 : 1
makeamericagreatagain → 12.2 : 1	hot → 1 : 9.6	teamjes → 1 : 11.8	c → 11.6 : 1
lov → 11.6 : 1	nic → 1 : 9.6	rap → 11.4 : 1	civil → 11.6 : 1
awesome → 10 : 1	delet → 1 : 7.1	tip → 1 : 11.1	hero → 11.6 : 1
dream → 10 : 1	delud → 1 : 7.1	awesom → 1 : 10.9	movy → 11.6 : 1
enjoy → 10 : 1	suck → 1 : 7.1	mary → 1 : 10.9	admir → 9.5 : 1
stat → 1 : 8.9	election2016 → 6.7 : 1	liberty → 1 : 9.9	pen → 9.5 : 1
effect → 1 : 8.4	freeallfo → 1 : 6.2	lie → 9.7 : 1	prowomancho → 9.5 : 1
sadness:	**disgust:**	**fear:**	**anticipation:**
victim → 9 : 1	rap → 29.8 : 1	risk → 12.8 : 1	- no features
sham → 8.5 : 1	hypocrit → 13.4 : 1	happy → 1 : 11.8	
liberty → 1 : 8	sham → 12.5 : 1	temp → 11.7 : 1	
road → 1 : 8	abus → 11.7 : 1	viol → 11.7 : 1	
abus → 7.9 : 1	bigot → 11.7 : 1	agend → 10.6 : 1	
makeamericagreatagain → 1 : 7.3	jes → 1 : 11.3	drought → 10.6 : 1	
oppress → 7.3 : 1	sin → 1 : 11.1	harass → 10.6 : 1	
inspir → 1 : 7.2	lying → 9.9 : 1	ter → 10.6 : 1	
war → 6.8 : 1	whyimnotvotingforhil → 9.9 : 1	hrc → 9.5 : 1	
degr → 6.7 : 1	idiot → 9.6 : 1	fear → 9.3 : 1	

Fig 3 The ten most informative features for all eight separate classifiers. The probability with which the emotion holds in case of observing the feature is denoted by 'holds' : 'holds not' after the arrow.

As demonstrated by the most informative features, the classifier reliably learns the emotion associated with each word. While the learned probabilities are generally understandable (i.e. the hashtag *#lovewins* denotes joy), the preprocessing leads to some transparency issues. The fourth most informative feature for surprise for example is stemmed to just one letter, which we can therefore make no assumptions about. A second problem arising from stemming is the most informative feature of disgust. While knowledge about the data base confirms that <rap> is the stem of <rape>, the classifier will not distinguish between the two concepts.

1.4 Evaluation

The first observation made without consulting test data, is that while stemming does reduce the vocabulary size, lemmatizing the tokens might be a better choice. That way, while morphologically related words are merged, problems such as described in the last section can be avoided. This is a better solution for emotion classification, since morphologically related words are more likely to also be semantically and thus emotionally similar than graphically similar words. Another observation is the absence of learned features for anticipation. This is due to the non-occurrence of the tag in the training data. Therefore, all features are learned as 100% denoting no anticipation.

1.4.1 Evaluation on similar data

To measure the success of the classifier on our own corpus, we first evaluate it on a held-out section of the SSEC corpus. There, the classifier reaches the scores seen in figure 4.

	joy	**sadness**	**trust**	**disgust**	**anger**	**fear**	**surprise**	**anticipation**
precision	0.654	0.687	0.612	0.653	0.747	0.533	0.209	0
recall	0.621	0.75	0.644	0.7	0.813	0.596	0.267	0
accuracy	0.728	0.655	0.579	0.664	0.707	0.637	0.692	1
f1	0.637	0.717	0.628	0.676	0.779	0.563	0.235	0

Fig 4 Evaluation of all eight classifiers trained and tested on different parts of the SSEC corpus.

The classifier performs well in assigning the labels of the separate categories. As seen in the difference between accuracy and the f1 score, most classifiers tend annotate more texts positive than wanted (i.e., acc(sadness) = 0.655, fs(sadness) = 0.717). The lower accuracy score can be ascribed to the number of true negatives which are not considered when calculating f1. The above explained problem with anticipation is also visible, as there are also no occurrences of the label in the test set. Thus, the SSEC is no viable option to train a classifier for its annotation. The accuracy of the accumulated classifier is 0.101, which is low compared to the separate classifiers but was expected. As only documents where all eight texts are correctly labelled are counted toward the accumulated value, the classifier must be eight times as good to reach a similar number as the individual classifiers. When relating this to the individual values, the accumulated classifier has a satisfactory accuracy.

1.4.2 Evaluation on our corpus

Our self-annotated corpus consists of 100 comments in the reddit thread "*Redditors who live with their SOs, what changed the most when you moved in together?*" The title of this thread suggests a possible difficulty for the classifier. Even though the SSEC's domain is general, most documents discuss opinions on political topics (i.e., the 2016 US election, the marriage equality law, and abortion laws). Thus, our corpus' domain of relationships and cohabitation is foreign to the classifier. As the question also addresses a change, anticipation and surprise will occur often. Already on similar data, surprise has the worst performance while anticipation is not learned at all.

Our corpus is processed the same as the SSEC corpus. While the classifier results in the scores in figure 5., its accumulated accuracy remains 0. This is explainable through the overall lower scores on the corpus and the added complexity of matching the average annotation between just three annotators. As the last assignment showed, while the κ-score between 1/0 annotations was considerably high, the agreement on the valence scale was significantly lower. The accumulated annotation is computed from the average of the valence labels, which means the annotation is much sparser than usual.

	joy	sadness	trust	disgust	anger	fear	surprise	anticipation
precision	0.23	0.055	0.141	0.078	0.167	0	0.176	0
recall	0.609	0.8	0.562	0.8	0.667	0	0.353	0
accuracy	0.434	0.293	0.374	0.515	0.333	0.657	0.606	0.98
f1	0.333	0.103	0.225	0.143	0.267	0	0.235	0

Fig 5 Evaluation of the classifier trained on SSEC and tested on our annotated reddit comments.

Our inter-annotator agreement also explains the high accuracy for anticipation. While every annotator separately annotated the class often, the domain of the corpus makes the annotation noisy, which leads to only two occurrences of anticipation in the final accumulated annotation. The classifier also annotates too many instances overall as seen by the difference between precision and recall. The recall scores are lower than but comparable to the performance on the initial test data, but the classifier produces many false positives, which results in bad precision scores.

To answer the research question, the classifier is reasonably suited to annotate our corpus like we did. While it does not match our own annotation, the latter is already an average of dissimilar annotations. Therefore, the classifier does not perform well with the average annotation as the gold labels, but each individual annotator also deviates from this annotation. The classifier performs remarkably well considering the differences in domains and class distribution, but a training set that resembles our corpus more would be preferable.

2 Deep Learning

Our first experiment proved that an ML algorithm that was trained on data from one corpus can be used to label data from another corpus. For our second research question, we want to conduct a similar experiment, but this time using Deep Learning:

Can we transfer the learning of a NN trained on one data set to evaluate reasonably on another corpus?

Fig 6 Research question for the second task

As with our first research question, we need to select the appropriate corpora first. Both corpora must have same or similar annotations should preferably also be similar in domain and document type. As we need a large amount of data to successfully train the neural network, we chose the two largest of the available corpora, CrowdFlower and TEC. These datasets are suitable for this task because they are both annotated with Ekman's six emotions and are sourced from twitter. To obtain reliable results, we train our model on CrowdFlower to then evaluate in on TEC and vice versa.

2.1 Method

We decided to train multiple deep learning architectures to compare their performances on the given data, thus getting the best possible results: simple NN, CNN and Bi-LSTM. In this documentation, we focus primarily on describing the Bi-LSTM approach as we expected to see the best results using this architecture.

2.2 Implementation

The data preprocessing for this task is similar to the first task, except that the two corpora were split into a training and a validation set to further control the training.

2.2.1 Preprocessing

Stopwords were removed from the tokenized data using NLTK's built-in method, except for 'not' and 'but', as those words could bear information relevant for emotion analysis. The tokens are then lemmatized using NLTK's WordNet lemmatizer and stemmed using NLTK's Porter stemmer. Afterwards, the training data has a vocabulary size of 400,000 terms. The longest sentence consists of 33 words, therefore the input matrix of the NN has a dimension of 33x400,000 and consists of 33 one-hot vectors.

2.2.2 Network architecture

The first layer of the network consists of a GloVe embedding (Pennington et al., 2014) which transforms the input into 100x100 matrices and enriches the 'interpretation' of each word. The second layer consists of 32Bbi-LSTM units and counters class-imbalances using class specific weights. To prevent overfitting, a dropout of 0.5 is implemented. The resulting architecture is depicted in figure 7.

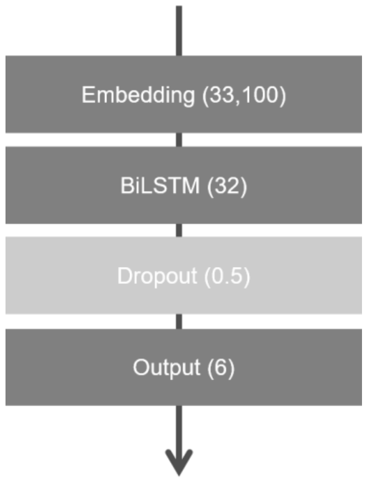


Fig 7 Architecture of the biLSTM neural net.

To prevent stagnation of the training, the learning rate is reduced when it reaches a plateau. This results in the training rate seen in figure 8 and 9. While above only one architecture is described, for every step, two neural nets are trained simultaneously. The methods use the same framework but are trained on different data, respectively on CrowdFlower or TEC. As the final evaluation is done with the other data set, the training rate is also determined using the other data set.

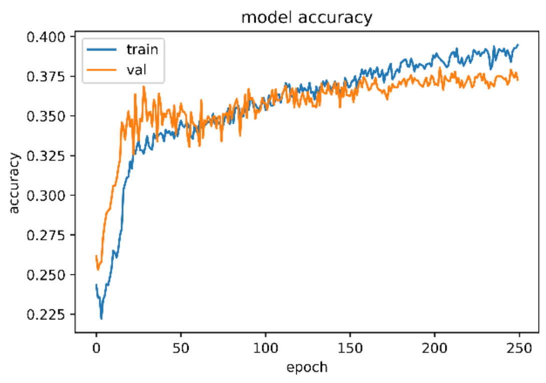


Fig 8 Training rate of the Bi-LSTM trained on CrowdFlower evaluated on TEC

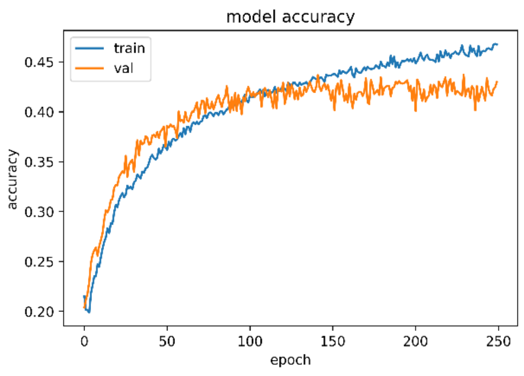
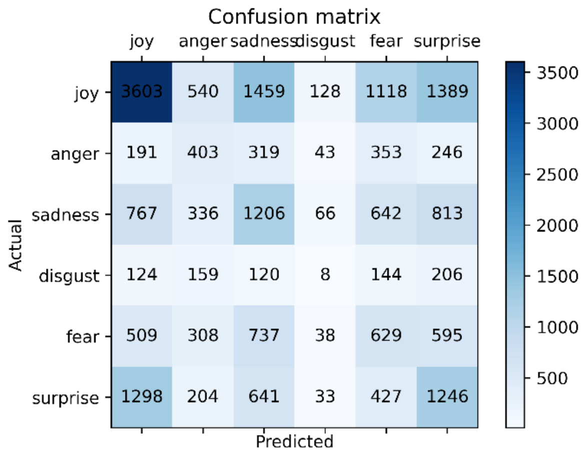


Fig 9 Training rate of the Bi-LSTM trained on TEC evaluated on CrowdFlower

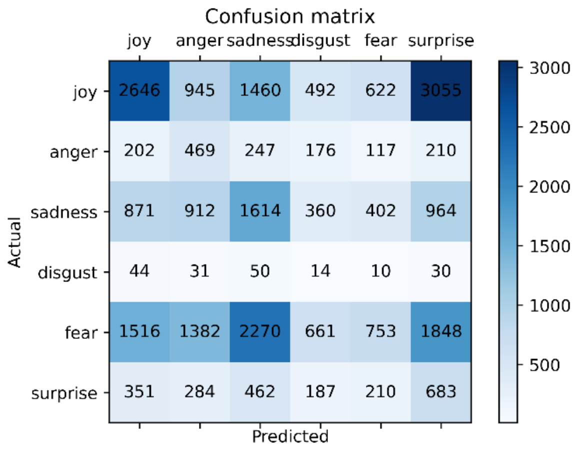
2.3 Evaluation

When evaluating the trained networks, we can see that both do not perform well. As depicted in the confusion matrices in figure 10 and 11, neither network can reliably annotate the true label better than the other labels (no diagonal visible). The network trained on CrowdFlower reaches an f1 score of 0.337 while the other network has a score of 0.232. The best class of the CrowdFlower trained network is joy, which is by far the most frequent and most correct annotation. The other network mostly annotated joy as surprise. The worst class of both networks is disgust, which is partially explainable through the imbalanced classes and data sparsity for this annotation.



Actual \ Predicted	joy	anger	sadness	disgust	fear	surprise
joy	3603	540	1459	128	1118	1389
anger	191	403	319	43	353	246
sadness	767	336	1206	66	642	813
disgust	124	159	120	8	144	206
fear	509	308	737	38	629	595
surprise	1298	204	641	33	427	1246

Fig 10 Bi-LSTM trained on CrowdFlower and evaluated on TEC



Actual \ Predicted	joy	anger	sadness	disgust	fear	surprise
joy	2646	945	1460	492	622	3055
anger	202	469	247	176	117	210
sadness	871	912	1614	360	402	964
disgust	44	31	50	14	10	30
fear	1516	1382	2270	661	753	1848
surprise	351	284	462	187	210	683

Fig 11 Bi-LSTM trained on TEC and evaluated on Crowd-Flower

Surprisingly both networks also significantly confuse sadness and joy. To further evaluate these problems of the networks, we will look at the underlying data.

Here we find, that CrowdFlower especially has a noisy annotation. This is to be expected as the annotation is crowdsourced. For example, “Can honestly say, don’t think I’ve ever done so shit in an exam before” is annotated as joy while the model predicts anger. This prediction seems intuitively more suitable than the gold label, and the discrepancy is explainable through the gold labels’ divergence than a fault of the classifier. Additionally, it contains several non-english texts, which also reduces the performance of a classifier as these words are not learned.

On the other hand, the model seems to succeed in predicting the correct labels for texts that require context or reasoning (“dont wanna work 11-830 tomorrow but i get paid”, sadness).

3 Conclusion

Both tasks showed us that the performance of both ML and DL is dependent on the amount and quality of the training data. The domains of training and testing data should overlap, and the training data must have a balanced distribution of labels. As the feature-based Naïve Bayes approach relies on individual words, which are observable as explicit emotional signifiers (<happy> = joy), it allows for explainable reasoning. Deep Learning on the other hand is not as transparent but can consider underlying meaning and context for the computation of labels.

4 Sources

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.