Improving Semi-Supervised Sentiment Analysis Performance Through Pre-Clustering: A Comparison of Models

Lukas Krieger
1337735

Tobias Walter
1579784

Julian Weller
1579342

David Zajchowski
1415754

1 Application Area and Goals

As an increasingly important task in the field of text classification, sentiment analysis has become widely popular, not only in the scope of academic research, but especially in many real world applications where there is great value in classifying the underlying sentiment of textual expressions. In addition the rise of Web 2.0 applications as well as micro-blogging services, like Twitter or Reddit, lead to an expeditious growth of user-generated content on the Internet. The task of determining the sentiment of such content without explicit user ratings, is subject of a large body of research which has been conducted in recent years. Especially classifying the sentiment expressed in single sentences or short comments has become of particular interest, although, this is rather tricky as not much context is available. However, this kind of content provides valuable information for different kinds of application areas, like market research, customer satisfaction or political popularity surveys.
Within this Web Mining project we will focus on the classification of Twitter posts, also known as Tweets. Not only because they can be obtained from the official Twitter API and a lot of research and baselines have been published, but also because the API allows to search for Tweets with embedded emoticons. Later on, these emoticons serve as weak indicators of the expressed sentiment in our Tweets. This method is inspired by the research performed by Go et al. (2009) and known as *distant supervised learning*.

Sentiment analysis is facing two main challenges. First, it requires a corpus of labeled samples, since it is a supervised task. Second - and resulting from the first challenge - the model is learned based on a fixed sample of a very broad domain or, on the very contrary, on a very specific domain. This results in the problem, that a very broadly learned model might not get the characteristics of different domains and a domain-specific model is not generic enough to be globally valid and can not be used to analyze the sentiment of a different domain (*domain-transfer problem*). These two challenges are resulting in most cases in a trade-off between performance, size of training data and number of learned models. In this web mining project we will focus on solutions to optimize this trade-off by using distant-supervised learning in conjunction with pre-clustering the data according to specific topics or domains. Distant-supervised learning might solve the problem of putting a lot of effort in creating labeled sample data, while topic-clustering might solve the problem of not having globally valid characteristics for a more broad corpus of different domains.

For this reason we created several domain specific data sets. Seed topics we used to distinguish between different domains are for example *food*, *movies* or *US-politicians*. Learning the classification model on one domain-specific data set, we aim to apply the very same model on unseen data from a different domain afterwards, and analyze how the model behaves in terms of performance to evaluate its transferability.

2 Structure and Size of the Dataset

Twitter provides several REST APIs for obtaining data, like Tweets, users and timelines. For this project, we focused on the */search/tweet* endpoint, which returns Tweets of the last seven days for a given search query. Additionally, this API endpoint provides a specific *attitude* and *language* operator, to exclusively get Tweets with *positive* or *negative* attitude, based on the used emoticons by the user, and written in a specified language.

As a setup for our project, we selected eight higher level and distinct topics and for each topic, between 10 and 20 subtopics or keywords have been identified. Table 1 gives an overview of exemplary keywords that we used for assigning the Tweets to the topics.

Topic	Exemplary Keywords
Car Manufacturer	Chrysler, Jeep
Fast Food	McDonalds, KFC
Movies	Star Wars, Batman
Musician	Drake, Jazz
Series	Netflix, Suits
Sport	Neymar, Roger Federer
Technology	iPhone, Google
US Politician	Trump, Obama

Table 1: Exemplary Keywords for the Topics.

For each topic, we searched for positive and negative Tweets in all of the respective subtopics identified through the keywords. For example, for the "Jeep" keyword in the car manufacturer topic, we fetched the following Tweets that contain either a simple happy or a simple sad smiley:

- Tweet 1: "@Jeep_SA If I crash this car because of being an underage driver, Ill sue you then :) thanks !"

- Tweet 2: "I'm actually excited I'll have JEEP tomorrow because I always feel like I don't have to worry about what I need to do in school or anything else while I'm there, and my co-worker is super lovely and nice too :((("

Due to the restriction of only being able to pull Tweets of the last seven days, the overall data set generation has been performed as an incremental process. Every few days, we run our Python script to pull the latest data from Twitter and store the retrieved Tweets afterwards. Moreover, we iteratively extend our list of keywords whenever we are not able to fetch a sufficient number of Tweets for each of the topics.

This process led to a total of 381,316 Tweets fetched during the last weeks and figure 1 displays the distribution of received Tweets among all eight topics.

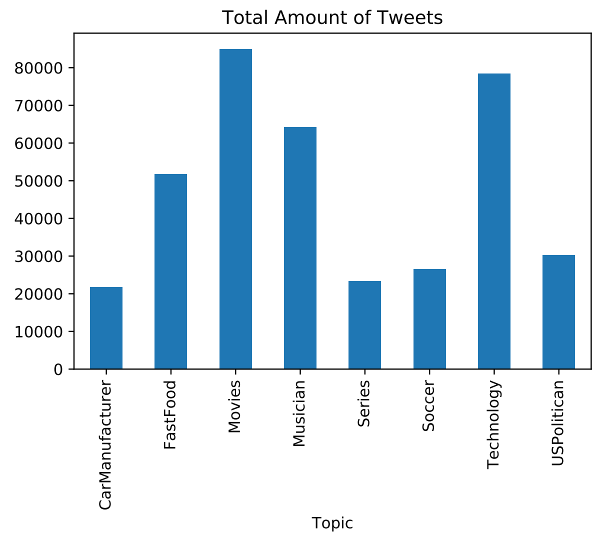

Figure 1: Amount of Tweets per Topic.

In total, most Tweets are related to movies while the least represented topic is the one about car manufacturers. When grouping the Tweets by attitude, we can see in figure 2 that it is positive for most of the Tweets:

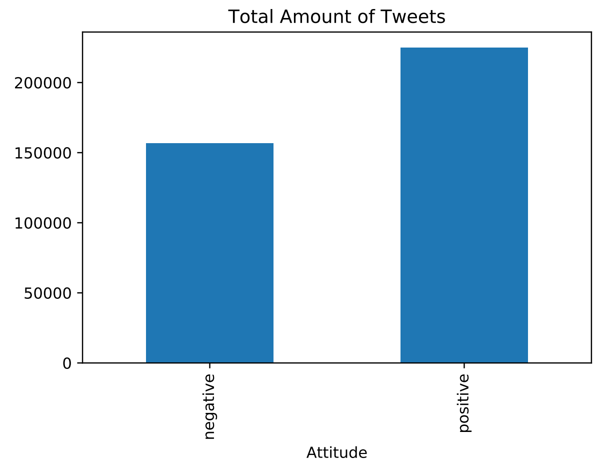

Figure 2: Amount of Tweets per Attitude.

When looking at both attitude and topics at the same time like in figure 3, we can see that the within-topic distributions of sentiment vary. For example, the majority of Tweets from the technology topic are positive, while most Tweets about movies are negative.

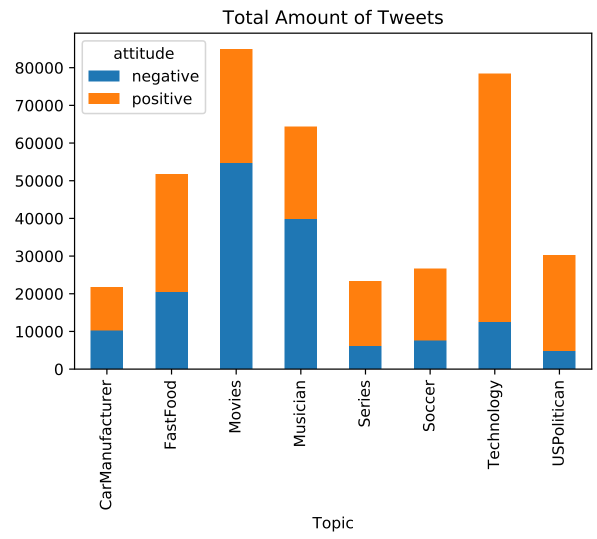

Figure 3: Amount of Tweets per Topic and Attitude.

After normalization has been performed, the bars belonging to topics can even be directly compared, as shown in figure 4:

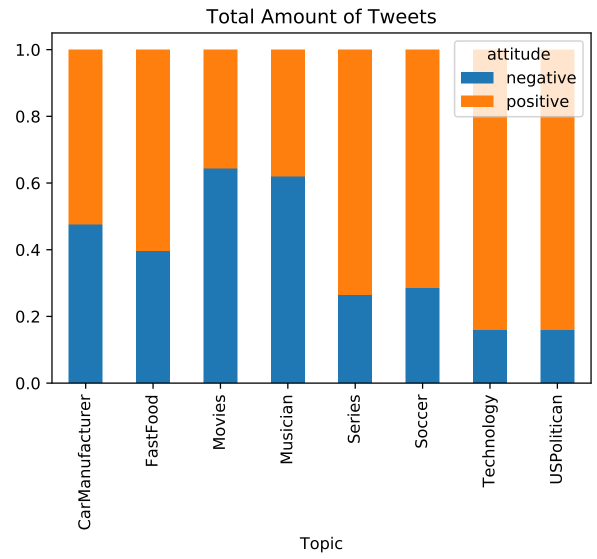

Figure 4: Amount of Tweets per Topic and Attitude after Normalization.

For example, around 52% of the 21,782 Tweets on car manufacturers are positive.

Quality of Emoticon Based Labels As one of the first steps of our project, we want to evaluate the quality of using emoticons as an indicator of a Tweet's sentiment by randomly selecting a stratified subset of 1,225 Tweets of our entire data set. In order to measure the quality, we manually label these Tweets, generate a confusion matrix, and calculate the accuracy accordingly. Since we remove the emoticons from the Tweets before learning our models, we were not allowed to take them into consideration for our labeling decision as well. Consequently, we only assign 473 positive or negative labels to the Tweets manually because for the remaining 752 Tweets, it is not straightforward enough what kind of opinion orientation they are expressing when ignoring emoticons. Table 2 displays the confusion matrix between our human-based sentiment labeling and the labels derived from using emoticons.

	Manual-Pos	Manual-Neg
Emot-Pos	212	52
Emot-Neg	5	204

Table 2: Confusion Matrix of Manually and Emoticon-labeled Tweets

As to be inferred, $\frac{212+204}{473} = 0.8795 \approx 88\%$ of all labels matched. These results are quite satisfying and show that using emoticon-based distant supervision to label the sentiment of Tweets seems to be an adequate method. However, we noticed that there is a certain dispersion between the two classes. While negative emoticon-based Tweets have a relatively high class-precision, the positive emoticon-based Tweets do not perform as well. Having a closer look at the misclassified Tweets, we figure out that in most cases this is due to ironic expressions implicated by positive emoticons at the very end of the Tweet. The following Tweets contained in our example set may serve as instances supporting this assumption:

- Tweet 3: "i love how i can't seem to do anything right for my parents not even make a pizza order :)"
- Tweet 4: "oh yeah and i didn't get breakfast either but was still forced to clean the kitchen :)"

Using emoticon-based distant supervised labeling, both of the preceding Tweets were assigned a positive sentiment label. In contrast to this, as humans with a semantic understanding, we are able to identify expressions of negative sentiment in both of these Tweets. Accordingly, not being able to do anything right for one's parents or being forced to clean the kitchen after not having got any breakfast, are unambiguously not desirable circumstances. However, these negative opinions are expressed in an ironic way, which is why they frequently appear with positive emoticons indicating the humorous aspect of the underlying message.

Filtering the Data After the data has been fetched from Twitter, it is required to maximize the quality of the data. The definition of exclusion criteria aims to exclude those Tweets, which are not worthy to evaluate or might even bias the results. Those limitations can be categorized and addressed by the following defined exclusion criteria:

EC1 Data Basis:

1. Removal of ReTweets
2. Removal of quotes
3. Removal of bot-generated Tweets

EC2 Data Quality:

1. Tweet lengt
2. Entity ratio (Question: Does a single word Tweet surrounded by multiple user mentions or URLs have any valuable information? Answer: Probably not)

Duplicates, ReTweets, Quotes The first category of exclusion criteria focuses on the problems that resulted from the methodology of the performed data collection process. The data collection has been executed multiple times, thus, a single Tweet may has been received not only once, but in some cases, multiple times. This methodology problem can be solved by removing all duplicate Tweets. While merging the collected Tweets, a single Tweet has only been added to the final data set, if it is not already contained in it. This results in a duplicate-free data set, even for same Tweets from different topics.

Furthermore, some Tweets are just ReTweets from other users and should not be counted as dedicated Tweets in the web mining process, but rather be excluded from our data set in order to avoid giving particular Tweets any extra weights merely due to their multiple appearance within the data set. To do so, all Tweets starting with the characters *"RT"* have been excluded and only the original Tweet has been kept.

The removal of quotes has been performed because it is difficult to distinguish between the original Tweet and the Tweet in response to it. Due to the data set generation methodology, the searched sentiment attitude might be in the quoted Tweet, while we are analyzing the answered one.

Bot Generated Tweets Twitter allows to publish Tweets via their API, so that Tweets can also be created from automated processes by programs. Since it is unclear how these automatically published Tweets have been generated (e.g. the AI Twitter Bot *Tay* from Microsoft), we decided to exclude those bots as well from the final data set, as this may highly bias the results of our further analysis. Consider a data basis of 1,000 Tweets: If a single bot has published half of the Tweets, all results are biased towards the content and structure the bot is posting and the content will not be valuable for the analysis, as it is automatically generated! In order to solve this problem, we took a closer look at the *source* property of the Tweets. It is a HTML formatted link to the website, service or bot who has published the Tweet. In a first step, we extracted the title and *href* attribute from the raw property. In an iterative process, we analyzed the extracted information and defined rules to exclude sources, which had not been created via the Twitter website, apps or other directly related services.

With the described filtering, we are able to reduce the allowed amount of different sources from initially 1,556 to only 23.

Data Quality The second category of criteria revolves around the quality of the received Tweets. For the purpose of sentiment analysis, we only want to retain Tweets of sufficient quality, which is why we take the task of filtering Tweets based on their content into account. This might be useful because some contents of Tweets might bias or conceal statistical regularities and patterns underlying the data, and therefore affect our subsequent classification performance in a negative way. With regard to Tweets, we define the quality based on two basic characteristics: First, very short Tweets only containing the searched topic and no or merely a few additional words should be removed, as they lack context for the analysis. Second, Tweets which contain a high ratio of Twitter specific entities, like images, URLs or user mentions. For example, if a Tweet does not contain much context about the actual topic, but multiple images and user mentions, it does not provide a lot of viable information for learning algorithms responsible for the sentiment classification and would therefore merely bias the results.

Final Dataset Table 3 displays how much of the original obtained data is filtered by each of the aforementioned filter-criteria. Note that for some of the Tweets, several filter apply, which is why the final number of Tweets is not simply the subtraction of filter matching Tweets from the total number of Tweets. In this context, what is most striking is that a lot of ReTweets have been fetched and the final amount of usable Tweets for our project is only roughly a third of the overall fetched data. All the subsequent preprocessing steps are only applied to those Tweets that passed the filtering process.

Total	Retweets	Quotes	Bots	Quality	Final
381,816	211,139	19,203	41,377	19,203	**123,457**

Table 3: Amount of Tweets Filtered by each Filter Criterion

3 Preprocessing

Due to many unique characteristics of twitter data, sentiment classification of Tweets seems to be a rather challenging task. The language used is not only of a very informal nature, but also filled with a lot of abbreviations, slang and often an incorrect use of English grammar. Characteristics like these, among several others, result in an increased complexity when performing a linguistic analysis. Hence, a lot of effort of this project has been allocated to the preprocessing of Tweets. The following chapter points out all the steps taken into account for processing the raw text data, in order to generate an input data set for the subsequent web mining process. Table 4 displays how we step by step decreased the diversity of the informal data set by applying several domain-specific preprocessing steps. The table also highlights how many Tweets have been changed by each individual step.

The corpus of the raw Tweets fetched from Twitter contained 222,249 individual tokens. After all preprocessing has been performed, the corpus of tokens has been reduced to 65,183 unique tokens.

Step	Name	# Unique Tokens	# Touched Tweets
0	Raw Tweets	222,249	-
1	Entitiy Replacement	110,423	122,851
2	Misc Text Preprocessing	86,840	123,350
3	Remove Emoticons	86,576	104,228
4	Remove Duplicated Characters	86,154	62,406
5	Contractions	85,782	123,450
6	Slang Replacement (General)	85,421	37,780
7	Slang Replacement (Twitter)	85,378	5,871
8	Split Emoji	84,546	4,890
9	Remove Punctuation and Misc	74,172	111,655
10	Spelling Correction	72,334	49,806
11	Lemmatization	65,183	103,880

Table 4: Overview of the Preprocessing Steps

Twitter Entity Replacement As mentioned before, a rather large proportion of Tweets contain some twitter-specific entities like *user mentions*, *URLs* or *hashtags*. As different forms or values of such entities do not influence a Tweet's sentiment, but merely result in a more diverse data set, we replace all such entities with appropriate placeholders. For example *URLs* has been replaced by *XXURLXX*. The reason why we do not simply remove all entities completely is because the mere existence as well as the structural implications of those entities might still influence a Tweet's underlying sentiment. In addition to that, we replace the particular queried topic with the placeholder *XXTOPICXX* in order to be able to transfer the learned model later on.

Misc Text Preprocessing, Removal of Emoticons and Duplicated Characters These preprocessing steps are focusing on the structural appearance of Tweets. In a first step, each Tweet is transformed to lowercase and linebreaks as well as redundant spaces are removed. The textual emoticons, e.g. :-))), used during the data set generation are removed. This step is especially important since otherwise those emoticons would have a significant influence on the classification results later on. In a last step, duplicated characters are removed from tokens. For example the token *Happppy* will be transformed to *Happy*.

English Contractions As the English language offers various abbreviations (e.g., *can't*, *hell*, or *ya'll*), which are, especially in the context of online micro-blogging, widespread and inflationary used, we further take English contractions into account. In order to normalize abbreviated expressions and standardize our data, we want them to be written out adequately. Therefore, using a list from Wikipedia[1], in a first step, we replace all tokens including expressions of abbreviations with their corresponding meaning: *ya'll* becomes *you all.*, *hell* becomes *he will*, and *cant* becomes *cannot*. In a second step, we only look at word-endings (e.g., *'s*) and replace those, which have not been matched so far, based on a second manually created lookup table.

Replacement of Slang and Abbreviations The usage of slang terms represents another significant particularity in online messaging. The resulting problem of having different strings with the same semantic meaning is tackled by generating two dictionaries of key-value pairs for those slang words. The first dictionary is an automatically generated one by extracting 1,461 slang words for text messaging and online chat abbreviations with their respective meanings from a corresponding website[2]. The second dictionary is manually generated in an incremental process. Due to the existence of very twitter-specific slang terms, the first dictionary does not replace all slang terms contained in our Tweet corpus. Therefore we print out each infrequent and misspelled word and identify incorrect words which were not replaced by the first dictionary afterwards. This process gives us a good starting point for identifying twitter specific slang as well as for building up the second dictionary. Both dictionaries are then applied in order to identify and replace such terms.

Split UTF-8 Emojies A lot of Tweets are containing UTF-8 emojies which provide valuable information for sentiment classification. Taking a closer look at the raw Tweets, these emojies are concatenated quite often without spaces, which is why the used tokenizer is not able to split each emojie from adjacent tokens. Therefore, we applied regular expressions to identify each UTF-8 emojie and inserted spaces right before and after them. Afterwards, each emojie is isolated and the tokenizer is able to split them up.

Spelling Correction During the process of manually labeling a couple of hundreds of Tweets , we noticed that we are dealing with a lot of misspelled words in our data set. In general, this is a challenging problem for text classification, as each individually misspelled word counts as distinct feature for the learning algorithm used for the classification task. Since applying commonly used preprocessing techniques like stemming or lemmatization does not solve this problem, we used *PyEnchant*[3], a spell-checking library, in order to correct misspelled words in an automatized manner.

Initially, we applied spelling correction to all words occuring in the Tweet corpus, but then, we realized that the used package and dictionary do not seem to know domain specific words like *iPhone*. This led to the problem that even correctly spelled, but domain-specific or frequently written words had been corrected in-appositely. Solving this problem, we defined some rules on which words should be corrected: The first rule states that only the least frequent 50% of words appearing in the corpus are considered as misspelled candidates, since these rather infrequent terms are more likely to be wrong. Thus, frequently occurring domain-specific words are no longer eligible for in-apposite spelling correction. The second rule makes sure that only words which are longer than three characters are possible targets for spelling correction. The third and last rule excludes all entity placeholder and emojies from the spelling correction process.

Linguistic Preprocessing For grouping inflected forms of a single word (e.g. *go* vs. *went*), we applied *lemmatization* based on *Part of Speech Tagging*. *Stemming* and *stopword removal* have also been tested, but did not perform as good. This might be due to the fact that stemming does not take grammatic and semantics into account and stopwords are an important component for expressing opinions and attitudes in very short portion of texts like Tweets.

4 Feature Generation

Having cleaned up and normalized the Twitter data to the most possible homogenous form, we now focus on the generation of features, by utilizing two commonly used representation methods, namely *Bag of Words* and *tf-idf*. Using the Bag of Words approach enables us to represent documents, like Tweets, as term vectors in a multidimensional vector space. Within this approach, text is represented as an unordered set of terms projected into fixed-sized vectors whose length is de-

[1] https://en.wikipedia.org/wiki/Wikipedia:List_of_English_contractions

[2] http://www.webopedia.com/quick_ref/textmessageabbreviations.asp

[3] http://pythonhosted.org/pyenchant/

termined by the vocabulary size of the entire document collection. However, since text in its original form can be regarded as one single string, we need to tokenize it first, to split it up into its single terms. Building up on this, next we need to determine how to compute the weights of the single index terms (vector elements) for each individual Tweet. For this purpose, the so-called *TF-IDF weighting scheme* is employed. Within this approach, the weight computation is basically determined by two different components: First, as a local component, the *term frequency* of term t in document d *tf(t,d)*, and second, as a global component, the *inverse document frequency* of term t *idf(t)*. The overall weight is then computed by normalizing the word occurrence frequencies by the inverse document frequency of the respective terms. Therefore, the resulting weight score not only increases with a higher frequency of a term within a Tweet, but also with increasing rarity of a term within our Tweet corpus. Furthermore we also tested different *n-Gram* configurations as features for our later classification. A n-Gram is defined as series of consecutive tokens of length n. According to this, the intuition behind using n-Grams is to also take the words' context into account, which might be very useful, for instance, in the case of adequately accounting for negations of adjectives (e.g., ”not good” or ”not hating”). Special attention has also been paid to maintain UTF-8 emojies like smilies, hearts, etc. in our data set as they represent an important sentiment-bearing component in online-messaging.

5 Web Mining and Results

In this section, we start by performing an exploratory analysis of the fetched data. Thus, the main objective is to describe the different data sets we fetched, i.e. one each per selected seed topic, with respect to their similarity. Afterwards, we train and evaluate our global baseline classifier on the entire Tweet corpus at hand. In a next step, we train local classifiers for each seed topic data set and evaluate their performance against the baseline classifier. Lastly, the learned single-topic models are then applied to the respective other topics' data in order to evaluate their domain transferability.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	.3618	.7290	.8857	.7060	.7802	.7113	.7076	.5541
2		.7862	1.00	.8606	.9211	.8786	.8772	.7486
3			.7194	.9616	.9402	.9710	.9929	.8741
4				.6749	.8067	.7417	.7380	.5600
5					.7032	.7964	.7948	.6262
6						.6142	.7447	.4883
7							.5936	.5558
8								.0000

Table 5: Scaled distance matrix of topics using cosine distance
Topic Labels: 1) Car Manufacturer 2) Fast Food 3) Movies 4) Musician 5) Series 6) Sport 7) Technology 8) US Politician

5.1 Topic Distance Analysis

To validate our approach of topic creation, i.e. that we actually created well discriminated topics, we examine the topics in terms of their similarity. For this purpose, we calculate the pairwise cosine distance between the tweet corpora pertaining to each of our 8 topics based on their TF-IDF representations. Since the raw distances do not allow for much interpretation (they are all in the 0.98 - 0.99 range), we scale the distance matrix to values between 0 and 1 in order to highlight the most similar pairwise topics (distance=0) and the most dissimilar topics (distance=1).

Table 5 displays the scaled pairwise distances. We observe that the *US Politician* topic is the most homogeneous topic (distance=0) and the topics *Movies* and *Fast Food* are most dissimilar. Taking a look at the diagonal, we can conclude that each topic is more similar to itself than to other topics, except the *US Politician* topic. In most cases (5 out of 7) of pairwise comparison between this topic and any other topic, the *US Politician* topic has a shorter distance to the topic (e.g. for *Musician* = 0.5600) than the topic has to itself (e.g. for *Musician* = 0.6748).

Overall, we can state that we have topics that are generally well discriminated. That is because we have topics that are dissimilar to each other in general, but we can still distinguish between more similar topics (e.g. *Car Manufacturer* and *Technology*) and less similar topics (e.g. *Car Manufacturer* and *Movies*), which is a promising observation regarding our further analysis, for example in the context of our classification task.

5.2 Baseline Classifier Analysis

We test nine different classification algorithms, however, only an excerpt are listed in this report, namely Naive Bayes, SGD, RandomForest and

SVM. In order to find the best configuration for each classifier, we manually tested several variations of n-Gram sizes and classifier parameters based on the weighted F1-score. All performance measures in this report are calculated by applying a 10-fold cross validation. For all classifier and n-Gram parameter, the TF-IDF vectors, as a Tweet's representation, perform better than simple term frequencies. Thus, all results are obtained using the TF-IDF weighting scheme only. Since our final data set consists of 82,941 Tweets with a positive sentiment label and merely of 40,516 Tweets with a negative one, we are facing an unbalanced distribution. Therefore, all classification tasks have taken the unbalanced distribution into account and all measures in this report are unbiased towards this distribution.

Table 6 highlights the four selected classification methods, their optimized parameters and the respective precision, recall, F1-score and accuracy. The highest class and total scores among all classifiers are emphasized. Additionally, Ridge, Perceptron and Passive Aggressive classifier are tested as linear models as well, but they perform slightly worse than the other two linear models (SVM and SGD). Due to the size of the data set, kNN could not be evaluated efficiently and is therefore not further mentioned in this report, although the results are similar to these from Naive Bayes and Decision Tree.

Classifier	Options	Score	Neg	Pos	Total
Naive Bayes	ngrams=1	precision	0.74	0.77	0.76
		recall	0.43	**0.93**	0.76
		f1	0.55	0.84	0.74
		accuracy	-	-	0.76
RandomForest	ngrams=(1,3)	precision	0.58	0.83	0.75
	estimators=40	recall	0.68	0.76	0.74
	min. split/leaf= 7/3	f1	0.63	0.80	0.74
	criterion=gini	accuracy	-	-	0.74
LinearSVM	ngrams=(1,3)	precision	**0.72**	**0.86**	**0.81**
	loss=squared_hinge	recall	**0.71**	**0.87**	**0.82**
	penalty=l2	f1	**0.72**	**0.86**	**0.82**
	C=0.5	accuracy	-	-	**0.82**
SGD	ngrams=(1,3)	precision	0.68	**0.87**	**0.81**
	loss=squared_hinge	recall	**0.75**	**0.83**	**0.81**
	penalty=l2	f1	**0.71**	**0.85**	**0.80**
	alpha=.0001	accuracy	-	-	**0.80**

Table 6: Overview of Classification Results

The results in Table 6 yield three conclusions: First, all tested linear models perform better than the other tested classifier. This might be the case because the features are in a very high dimensional vector space and linear models perform very well on such spaces. Secondly, having a look at the class precision and recall, all classifiers have to some extent problems with classifying the negatively labeled Tweets. This pattern might be based on the findings in Section 2, as there are some false positives in our training data which bias our predictive models. And last but not least, for almost every classification the n-Grams of the size (1,3) perform best, as it incorporates more context into the classification compared to single words or smaller n-Gram sizes. For the further analysis, we are only focusing on the LinearSVM classifier.

Evaluating Classification on Manually Labeled Tweets As shown in section 2, we already performed an evaluation of our emoticon-based labels to prove their quality for sentiment classification. However, we also need to prove that this quality still holds when we compare the predictions made by our model trained on weak labels, with our manually labelled tweets. Table 7 displays the results of a LinearSVM trained on the complete Tweet corpus with emoticon-based labels, excluding the manually labeled Tweets, and evaluating the predicted sentiment against the manually assigned sentiment.

The results show, that the previous observations in section 2 are still valid and an emoticon-based trained classifier performs quite satisfying when evaluated with manually labeled Tweets. Furthermore, the performance is increasing in comparison to the purely trained and tested emoticon-based baseline classifier as shown in table 6.

	Precision	Recall	F1	Support
Negative	0.94	0.76	0.84	256
Positive	0.77	0.94	0.84	217
Total	0.86	0.84	0.84	473

Table 7: Classification Report of the Baseline LinearSVM Evaluated on Manually Labeled Tweets

Dataset Size Influence on Performance Due to the fact that the next section compares the results from the baseline classifier to classifiers which have been trained and tested on a single topic, we need to analyze the influence of the data set size on the results. While the baseline classifiers have been trained and tested on the complete corpus of all 123,457 Tweets, each single topic only contains approximately 15,500 Tweets on average.

In order to evaluate the impact of the size of the data sets, the complete data set is split into 10 linear sizes. For each size, three stratified shuffle splits are evaluated using a 10-fold cross validation each, yielding to a total of 30 trainings and testings for each size and aggregated by the mean

F1-score.

Figure 5 displays the results of the evaluation. We observe an increase in the F1-score with increasing size of the underlying data set we use for training a model. Although there is only a marginal incremental increase of the F1-score, the analysis highlights the importance of a large data set for sentiment classification. While the average F1-score for a data set with 12,345 Tweets is approximately 0.775, it increases by approximately 4% when taking the full data set into account.

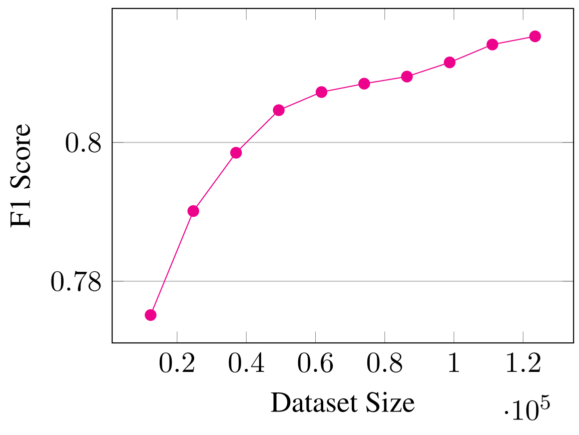

Figure 5: Increase of F1 score with increasing dataset size

5.3 Pairwise Topic Classifier Analysis

Now that we have a global baseline classification model, next we train individual classifiers for each of the fetched seed topics. These local classification models are then applied to the other topics for the objective of cross-domain transfer evaluation.

Table 8 provides an overview of the obtained domain-specific classification results.

Topic	**weighted F1-score**	**total support**
CarManufacturer	0.748518	10,141
FastFood	0.781544	22,849
Movies	0.787971	21,087
Musician	0.810695	8,953
Series	0.805584	10,089
Sport	0.804976	11,411
Technology	0.809404	27,383
USPolitican	0.811158	11,538

Table 8: Domain Specific Classification Performance

As one can see, the classification performances according to the weighted F1-score of all learned models do not vary by much. With an weighted F1-score of 0.748518, the learned model for the topic *CarManufacturer* seems to perform significantly worse than the others, whereas the learned model for the US Politician domain with a weighted F1-score of 0.811158 represents our best performing local model. The overall weighted F1-score for all individually trained models is 0.7951.

test / **train**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	.7485	.7248	.7402	.7870	.7368	.7871	.7959	.7946
2	.7224	.7815	.7568	.7974	.7613	.7852	.8074	.7835
3	.7159	.7480	.7880	.8019	.7751	.7895	.8026	.7890
4	.7100	.7317	.7409	.8107	.7610	.7855	.7945	.7890
5	.6972	.7152	.7602	.8011	.8056	.7921	.7982	.7878
6	.7102	.7117	.7363	.7876	.7555	.8050	.7920	.7942
7	.7215	.7537	.7307	.7861	.7513	.7712	.8094	.7620
8	.6299	.5841	.6563	.7303	.6782	.7511	.7400	.8112

Table 9: Pairwise performance of domain-specific models applied on other domains (domain transfer)
Topic Labels: 1) Car Manufacturer 2) Fast Food 3) Movies 4) Musician 5) Series 6) Sport 7) Technology 8) US Politician

In order to evaluate the domain transferability, we then use our domain-specific models to test them on all other topics the respective model was not trained on. The pairwise performances resulting from this domain transfer are shown in Table 9.

The overall achieved weighted F1-score of the transfered classification results amounts to 0.757889. The diagonal values of table 8 represent the performance achieved by a model applied to the same domain it was trained on. As expected, in most cases models perform better when applied to the domain on which they have been trained. However, several exceptions can be observed, e.g. a model trained on Tweets belonging to the *Movie* topic performs better when applied to *Musician* or *Technology* Tweets (F1-score: 0.8019) than when applying it to Tweets of its training topic (F1-score: 0.7880). A closer look on the table reveals another interesting fact, which is that the domain-transferability of the single-topic models differs significantly. In this sense, training a model on *USPolitician* Tweets and applying it to Tweets from a different domain does not seem to be a good idea, since the performance harshly drops. Interestingly, when doing it the other way around, the performance results are rather solid.

6 Discussion of Results

First, we are discussing and evaluating the results from the comparison of a globally trained *One-Fits-All* baseline classifier to individually trained topic classifiers. Second, we are analyzing the underlying relationship of topic distances to the performance of the transferred models. In a last step, we are discussing the results of the individually trained and transferred models compared to the *One-Fits-All* baseline classifier.

One-Fits-All vs. Individual Classifiers By just comparing the F1-scores of the *One-Fits-All* baseline classifier and the individually trained topic classifier, one could conclude that the individually trained models are performing worse. The average weighted F1-score of individual topic classifier is approximately 0.7951, while the single model solutions outperform it with approximately 0.82. Taking the evaluation of the influence of the data set's size for the classification results, as shown in table 5, into account, it becomes obvious that such a simple comparison does not seem to be adequate. The average size of an individually trained topic classifier is approximately 15,500 Tweets only. By taking a look, how the *One-Fits-All* baseline classifier is performing for such a small data set size, it becomes visible that the overall individually trained models are performing better. Thus, we are comparing the weighted F1-score from the individually trained classifier of 0.7951 to the 0.7751 which is received by the *One-Fits-All* classifier for a respective data set size. This comparison shows that the overall individually trained models for each topic are outperforming the *One-Fits-All* solution, although some topics like *CarManufacturer* are performing very poor.

Model Transfer and Topic Distance We are observing some interesting behaviours when we transferred individually trained models to different topics. Especially the *USPolitician* topic helps us to showcase our findings. According to table 5, the *USPolitician* topic is the most homogeneous one, as it has the smallest distance compared to other topics. Our classification model trained only on this specific topic performs very well and outperforms the baseline classifier, but transferred to the other topics, the performance is poor (see table 9). Thus, the classifier has learned very domain specific characteristics for predicting the sentiment of this domain with high accuracy, and these characteristics seem not to be present in different topics. On the other hand, all seven other individually trained models are performing also very well on this homogeneous *USPolitician* topic. Taking table 5 into consideration, the cause for this phenomenon can be explained as follows: The distances between the *USPolitician* topic and all seven other topics are relatively small. This leads us to the conclusion that first, a very homogenoues data set is a desired characteristic for sentiment classification and second, the transfer to different topics is performing well if the following two conditions are met. First, the topic on which the classification model was learned shall not be homogeneous. Second, the distance to the transferred topics should be relatively small.

Transferred vs. One-Fits-All When comparing the results from the *One-Fits-All* baseline classifier with an F1-score of 0.82 to the individually learned and transferred classifiers, with an average weighted F1-score of 0.75, we are concluding two important findings. First and most straightforward, the overall performances of the transferred models are lower than the performance of a single classifier approach. An explanation for this might be that a single classifier does not only learn the general characteristics, but is also capable of learning more domain-specific sentiments for each topic individually.

Second and more interesting, the individual topics might have a positive influence for the overall classification performance. Taking the topic distances from table 5 and results of transferred models from table 9 into account, we can conclude, that some topics are relatively similar and might benefit from learned characteristics of the other topics. For example the topic *Technology(7)* has a relatively well individual performance (0.8094 F1-score), but there are several individually trained and transferred topics which have almost the same performance after the transfer and this might result in a synergistic effect for the performance of the *One-Fits-All* approach.

Conclusion Our pre-clustering approach that used the simple heuristic of taking keywords as cluster indicators showed promising results in conjunction with the task of domain transfer in the scope of sentiment analysis. Although being simple, our approach resulted in viable topics that were successfully used as basis for learning solidly performing classifiers, both over all and single topics. Even more encouraging seem the results from our domain transfer task. They show that within the task of sentiment classification of Tweets, it is possible to apply a model learned on a specific domain to data originating from a different domain. This might even hold for domains one might intuitively perceive as being very different, e.g. Fast Food and Technology, if the prerequisites of the two Tweet corpora not being homogeneous is fulfilled and their distance is sufficiently small.

Overall this student project shows that supervised distant learning can be used for sentiment classification of Tweets, even though the weak labels from Twitter are in some cases biased and an influence on the overall performance is noticeable, but does not result in a significant performance decrease. Furthermore, pre-clustering of Tweets and individually trained models do have a positive influence on the sentiment classification's performance compared to the *One-Fits-All* classification approach. And last but not least, we gained valuable information on the important characteristics of the interplay of domain homogeneity and inter-domain distances in the area of sentiment classification domain transfers.