Institut für Maschinelle Sprachverarbeitung

University of Stuttgart
Pfaffenwaldring 5 b
D–70569 Stuttgart

Bachelorarbeit Nr.

Concept Drift and Adaptation for Emotion Detection in Twitter

Julian Liedtke

Course of Study:	Informatik
Examiner:	Prof. Dr. Sebastian Padó
Supervisor:	Dr. Roman Klinger
Commenced:	June 15, 2016
Completed:	November 2, 2016
CR-Classification:	I.7.2

Abstract

The classification task in dynamical environments is challenging. A reason for this is the change of their statistical properties over time. This characteristic is called concept drift and is one of the major topics in data mining. The objective of this thesis is to analyze, how accurate different systems are classifying in dynamical environments over a period of time. For this purpose, two different approaches are evaluated. One approach removes the features with the highest change in influence. The other is an ensemble based model which let experts vote between the outcomes. Although the models were not able to increase the accuracy after a long period of time, the results show that both models are able to achieve a higher accuracy than the baseline in particular cases. This outcome underlines that emotion detection in Twitter can be improved. New models or improvements to existing ones could be able to handle concept drift to achieve a higher accuracy.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

The volume of generated data is growing rapidly. Efficient and effective analysis tools for dealing with the ever-growing amount of data are of important need. Classification is one of the key tools to handle and organize automatically text information.

The objective of classification is to identify to which class a document belongs. One well-known instance of the classification problem is spam filtering. In this case, the task is to decide, if a new mail is spam or not. In this example, the documents to be classified are incoming mails.

In order to deal with classification tasks, systems are created which try to classify documents as accurate as possible. Training data has to be given in order to train the models. The training data consists of documents whose classes are already identified correctly. A naive approach for the spam filter task above is to compare new mails with already known spam mails and classify them as spam if they are similar to one of those.

This approach assumes, the correct class of a mail does not change. This assumption holds for stationary environments. There are environments in which the correct class of a document can change. Those environments are called non-stationary or dynamical environments.

An example of dynamical environments are social networks. Social networks have become popular in recent years with millions of daily users sharing their everyday activities with friends and family. As of September 2016, the online social networking application Twitter has about 300 million active users [16].

Several social network platforms provide micro blogging. Micro blogging is a form of communication in which users can describe their current status in short posts distributed by instant messages, mobile phones, email or the Web. One of the popular micro blogging platforms is Twitter. Twitter, as a micro-blogging service, allows users to communicate feelings, opinions and ideas in messages, so called *Tweets*, shorter than 140 characters [KLPM10]. While around 6,000 tweets are sent in an average second nowadays, over twenty times that amount (143,199 tweets) have been tweeted in a single record-breaking second on August 3, 2013, when people in Japan were viewing and tweeting about the animated movie Castle in the Sky as it was being aired on TV.

Four years earlier, when Michael Jackson died on June 25 of 2009, only 456 tweets per second had sufficed to set the all-time record at the time [16].

Classification in non-stationary environments differs from classification in a stationary environment. In dynamical environments like Twitter, the data distribution can change over time. Topics naturally vary in those environments while the concepts are not necessarily changing. For example opinions on politicians can vary while the concept of politics does not change. Another example are changes in users interests on news. The concept of news remains the same, but the conditional distribution of interests for those users changes. Those are examples of concept drift, which means, that the properties of a system are changing over time.

In data mining, predictive models are trained using historical data given as a set of pairs (message, label). Models trained in such a way that they can afterward be applied for predicting the output for new unseen input data. Traditionally, during training, the aim is to build the model such that it predicts the training data as accurate as possible. Due to the fact of concept drift, this approach could decrease the accuracy of predicting unseen data.

This thesis is aiming to find and build systems which keep stable, even if the documents are evolving because of concept drift. To determine the stability of systems, we will measure the accuracy of different systems over a long period of time. The main focus does not lie on the accuracy of the classification of the training data. We want to observe, how different systems behave if the phase of training ends and how fast and highly their accuracy decreases.

Influenced by concept drift, the domain of the messages in the environment changes over time. Consequently, the classifiers have to be adapted to new domains by detecting concept drifts to be used in a non-stationary environment.

This thesis examines two different systems which try to deal with concept drift.

1. **Remove Mutual Information Change** (RMIC). In a typical classification system, there are features, which tend to a specific class while classifying a document (for more about features see Section 2.1. The classification task and different classification models are introduced there). If, for some reason, the entropy of a feature changes rapidly (increasing or decreasing) over time, this feature might be affected by concept drift. This approach removes those features with the highest change in entropy. The change in entropy will be measured with mutual information (will be introduced in Section 2.4.1.1). The assumption is, that the remaining features are stable and form a group of base features which enable accurate classification over a long period of time. Up to date, this approach is not covered in current research.

2. **Dynamic Weight Majority** (DWM). This is an ensemble-based model which uses a combination of different classifiers and adjust the relation between them over time. The idea is to combine a set of classifiers to create a new classifier, which is able to balance between them to avoid minor mistakes. This model is already used in current research.

To evaluate those systems, we use two different classification tasks:

- Emotion detection. Automatically extraction of emotions from user messages is of great importance in machine learning [RRJ+12]. Understanding user's feelings towards particular products, services or topics can companies help to determine the emotions towards their products.
- Differentiation between irony and sarcasm. Irony detection is a challenging task in machine learning, because the models have to detect the intended meaning and not the literal meaning of a sentence [BS14].

As documents, Tweets from Twitter are used. Over a period of about four months, suitable Tweets for the classification tasks were collected. In total, about 650,000 Tweets are used in this thesis. As shown in Chapter 5, concept drift occurs in this period of time.

In order to evaluate the systems, the aim of this thesis is to prove or disprove the following hyopthesis:

The aim of the experiments is to confirm or to disprove the following hypotheses. An explanation of them is given in Section 5.1.

1. The accuracy of the incremental model decreases the more the distance to the phase of training increases.
2. The DWM algorithm achieves a greater accuracy than the incremental model for steps with a great distance to the phase of training.
3. For some percentage, the RMIC algorithm with Maximum Entropy as base classifier achieves a greater accuracy than the incremental model for steps with a great distance to the phase of training.
4. The RMIC algorithm removes features which are connected to events happening in the duration of the corpus.
5. The accuracy of the Naive Bayes model decreases when using the mutual information change remove method for domain adaptation

The structure of the thesis is as follows:

Chapter 2 – Theoretical Background: This chapter covers the basics of classification, concept drift and classifier models.

It is divided into five sections:

Section 2.1 is dedicated to provide the theoretical background for the thesis.

Section 2.2 introduces the two classification tasks in this thesis.

Section 2.3 specifies a dynamical environment.

Section 2.4 presents the two classifier models which will be evaluated in this thesis.

Section 2.5 gives an insight into related current research.

Chapter 3 – Corpus introduces how the corpus for the experiments is created and what its elements are.

It is divided into two parts. The first part presents how the corpus is created, the second part analyzes the corpus.

Chapter 4 – Methodology introduces the methods used for the evaluation of the two different systems. Its second section gives an overview of the implementation.

Chapter 5 – Concept Drift and Emotion Detection on Twitter: This chapter is about the experiments.

Chapter 6 – Summary and Outlook summarizes the results of the bachelor thesis and gives an outlook.

2 Theoretical Background

The chapter is divided into five sections.

The first section is dedicated to provide the theoretical background for the thesis. It explains the classification task and obtains two models which try to solve it, namely the Naive Bayes model and the Maximum Entropy model. Afterward it gives an overview over some methods, which preprocess the data for the classification task. In addition, it will introduce the Precision, Recall and F measures, which are used to quantify the quality of the models.

Section 2.2 introduces the two classification tasks in this thesis: emotion detection and differentiation between irony and sarcasm.

The third section specifies a dynamical environment. It will explain concept drift, which can happen in those environments. To deal with this behavior, the classifier models have to be adapted. This is done with domain adaptation, introduced in Section 2.3.2.

The fourth section presents the two classifier models which will be evaluated in this thesis. Those models use different approaches two handle concept drift. Their basic idea and approaches will be introduced there.

Concluding, an insight into related current research is given in the last section of the chapter.

2.1 Classification

Classification is the task of determine to which class (or classes) a document belongs.

An example of a classification task is an e-mail spam filter. Its goal is to decide if a document, in this example an e-mail, is spam or not. The labels are "spam" and "no spam", the input data are e-mails.

There are different types of classification tasks. If there exists exactly two classes (like in the example above), it is called binary classification. If there exists more than two classes it is called multiclass classification. If a document belongs to more than one class, the classification is a multi-label classification.

Formally, a classification task consists of the following components [BR+99]:

- a document space $\mathcal{X}$
- a (discrete) set of classes $\mathcal{Y}$

The task is to assign labels from $\mathcal{Y}$ to the documents in $\mathcal{X}$. More formally: The purpose of classification problem is to define the unknown predict function

$$f : \mathcal{X} \to \mathcal{Y}.$$

In order to be able to classify, a classifier need something to rely on. Therefore, the classifier is given a knowledge base containing already classified documents to train the model. For this purpose, the input data is divided into two sets: a set of training data and a set of testing data. The training data consists of documents whose labels are known. Those documents for which their labels are given are called examples.

In this bachelor thesis, a document $x \in \mathcal{X}$ is seen as a vector of words

$$x = (w_1, w_2, ..., w_n).$$

2.1.1 Naive Bayes Classifier

The Naive Bayes classifier is a conditional probabilistic classifier based on Bayes' theorem [BR+99]. Given a problem instance $x = (w_1, ..., w_n)$ to be classified with the words $w_1, ..., w_n$, it assigns to this instance probabilities:

$$p(c|w_1, ..., w_n)$$

for each class c. The classifier assumes, that those variables are independent [Ris01]. In other words, it assumes, that the occurrence of a word is independent of all other words. Using Bayes theorem, the formula can be written as follows [Lew98]:

$$p(c|x) = \frac{p(c)p(x|c)}{p(x)}$$

in other words:

$$\text{posterior} = \frac{\text{prior} \cdot \text{likelihood}}{\text{evidence}}$$

The denominator does not depend on c. It will be effectively constant, because the words are given. Thus, it can be ignored [Lew98].

The numerator is the joint probability $p(c, w_1, ..., w_n)$. The joint probability can be rewritten with the definition of conditional probability.

$$\begin{aligned} p(c, w_1, ..., w_n) &= p(w_1, ..., w_n, c) \qquad (2.1)\\ &= p(w_1|w_2, ..., w_n, c) \cdot p(w_2, ..., w_n, c)\\ &= p(w_1|w_2, ..., w_n, c) \cdot p(w_2|w_3, ..., w_n, c) \cdot p(w_3, ..., w_n, c)\\ &= ...\\ &= p(w_1|w_2, ..., w_n, c) \cdot p(w_2|w_3, ..., w_n, c) \cdot p(w_3|w_4, ..., w_n, c) \cdot ... \cdot p(w_n|c) \cdot p(c) \end{aligned}$$

The assumption from the beginning, that the features are independent, means

$$p(w_i|w_{i+1}, ..., w_n, c) = p(w_i|c). \qquad (2.2)$$

Thus, we can simplify Equation (2.2):

$$p(c, w_1, ..., w_n) = p(c) \prod_{i=1}^{n} p(w_i|c)$$

Accordingly,

$$p(c|x) = \frac{1}{\text{evidence}} p(c) \prod_{i=1}^{n} p(w_i|c) \propto p(c) \prod_{i=1}^{n} p(w_i|c)$$

In order to calculate

$$p(c) \prod_{i=1}^{n} p(w_i|c),$$

the model needs to know $p(c)$ and $p(w_i|c)$. Those values are determined with the training data.

$p(c)$ is the probability of class c. This value is determined with

$$p(c) = \frac{\#\text{documents in training data belonging to class } c}{\#\text{documents in the training data}}$$

$p(w_i|c)$ is the probability of a occurrence of word w_i in class c. This value is determined in the same way as $p(c)$ with the training data:

$$p(w_i|c) = \frac{\#\text{occurrences of } w_i \text{ in documents with label } c}{\#\text{words in class } c}$$

A difficulty with this formula is, if a term does not occur in the training data, $p(w_i|c)$ will be zero for all classes. To solve this problem, we add 1 in the numerator and adjust the denominator:

$$p(w_i|c) = \frac{[\#\text{occurrences of } w_i \text{ in documents with label } c] + 1}{[\#\text{words in class } c] + [\#\text{different words in } c]}$$

2.1.2 Maximum Entropy Classifier

While the Naive Bayes classifier uses the joint probability $p(y, w_1, ..., w_n)$ (see Equation (2.1)), the idea of the Maximum Entropy classifier is to use the conditional probability $p(c|w_1, ..., w_n)$ [BPP96].

The Maximum Entropy classifier uses features, which are defined as follows:

$$f(y, x) = \begin{cases} 1, & f \text{ holds in } x \text{ and class is } y \\ 0, & \text{otherwise} \end{cases}$$

where x is the document and y is a class.

If a feature holds or not can depend on any information. For the spam detection example from the introduction, possibilities are:

- *money* is in x and class is *spam*. This is a word based feature.
- The e-mail is received between 2 am and 5 am and class is *spam*. This is an feature using meta data.
- ...

The features are weighted, to represent their importance. Thus, to a feature f_i belongs a weight λ_i.

This weighted features can be used to vote between the classes:

$$\text{value of class } c \text{ with input } x = \sum_i \lambda_i f_i(y, x)$$

To get a probability, the value is divided by the sum of the values for all classes. Because weights can be negative, those values are mapped into $\mathbb{R}_0^+$ before. The final probability is as follows:

$$p(c|x) = \frac{\exp \sum_i \lambda_i f_i(y, x)}{\sum_{y'} \exp \sum_i \lambda_i f_i(y', x)}$$

For a more detailed explanation of the Maximum Entropy model see [BPP96].

One essential and important difference between the Maximum Entropy and the Naive Bayes model is the way of determine the weights of the features: The Maximum Entropy model is a incremental model. This means, it iterates until the weights satisfy selected constraints. In the Naive Bayes model, features cannot affect other features. In the Maximum Entropy model, features can affect other features. For example, class features can be simulated by increasing or decreasing all weights of features which belong to a specific class. This possibility is tried to utilized in the Remove Changing Influence approach introduced in Section 2.4.1.

2.1.3 Preprocessing methods

This section will briefly introduce three preprocessing methods, the Jaccard index, stopword filtering and stemming.

The aim of the methods is to reduce the size of the data set. This has two advantages:

1. The performance of the algorithms will increase. The high dimensionality of the feature space is a major challenge for many classification models [DAK07]. Since the complexity of many learning algorithms increases with increased data dimension, algorithms that can improve the categorization performance and efficiency by reducing the data into small dimensional space are highly desired. These methods make the learning task of categorization and information retrieval systems faster and more efficient [YLZ+05].
2. The needed storage will be reduced [YLZ+05]. Social networks like Twitter are data streams. One of the most challenging characteristics of data streams is their infinite length [MCK+10]. It is impractical to store and use all the historical data for training.

2.1.3.1 Jaccard Index

The Jaccard Index considers the similarity between two sets [Rea99]. The Jaccard Index of two sets is a value between 0 and 1. The value increases with increasing similarity of the sets. The formula for calculating the Jaccard Index for two sets A, B is as follows:

$$\text{Jaccard Index}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

In order to determine the Jaccard Index of two documents, the documents have be transformed into a set representation. This can be done in different ways. For example can a set of a document be the set of all bigrams of the document. This thesis uses the set of words in the document as the set representation of the document.

2.1.3.2 Stopword filtering

Stopword filtering is an commonly used preprocessing method in information retrieval and text classification [TÇG+11]. In stopword filtering, common frequent words are removed from documents. Examples of those common frequent words are *and*, *or* and *the*. Stopword filtering assumes, that those words do not have any informative value, thus they can be ignored.

2.1.3.3 Stemming

When categorizing text documents, not all features equally represent the semantics of the documents [DAK07]. Some of those features can be redundant or synonymous. Therefore, capturing one of them is enough to enhance the semantic for categorization purpose. With stemming, words are reduced to their stems. This method ensures that unnecessary many features are reduced to only one.

2.1.4 Evaluation Measures

In order to be able to judge if approach one is better than approach two, an objective measure of soundness of the approaches is needed.

By evaluating a single class, there are four possible different cases which can occur while classifying. Let's assume, the selected class is *class*, the document's true label is *label* and the classification model classifies the document into class *classified*. The four different cases are shown in Table 2.1 [BR+99].

1. **True Positive** (TP): $label = class$ and $classified = class$. The document belongs in the selected class and is classified correctly.
2. **False Positive** (FP): $label \neq class$ and $classified = class$. The model claims the document belongs to the selected class, but it does not.
3. **True Negative** (TN): $label \neq class$ and $classified \neq class$. The true label of the document is not the selected class. The classifier model claim the truth.
4. **False Negative** (FN)$label = class$ and $classified \neq class$. The model classifies the document not in the selected class, but in reality, it is.

Those four values can be determined for each document the classification model classifies, as long as the true labels of the documents are known. Those values can be counted to trace how often each case occurs. This approach lead to different measures: Precision, Recall and the F-Score [GG05].

	label $=$ class	label $\neq$ class
classified $=$ class	True Positive (TP)	False Positive (FP)
classified $\neq$ class	False Negative (FN)	True Negative (FN)

Table 2.1: Measures: True/False Positives/Negatives

2.1.4.1 Precision

Consider all documents, that are classified into class *class* by the classifier. Precision is the proportion of the documents, that are classified into this class correctly. As formula:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

2.1.4.2 Recall

Recall is quota of the documents with true label *class* which are classified correctly. As formula:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

2.1.4.3 F-score

To map precision and recall into one value, a weighted (harmonic) average between those two values is taken. This new value is called F-score and is defined as follows:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Recall} + \beta^2 \cdot \text{Precision}}$$

where β is the weight between precision and recall. Normally, an equal weight is used ($\beta = 1$) [YL99]. This is called the F_1 measure:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Recall} + \text{Precision}}$$

2.1.4.4 Micro and Macro F1 measure

In a multiclass classification task, the measures for the classes can be combined to one value. There are two ways for this purpose: micro- and macro- averaging [Yan99].

The micro-averaging combines TP, FP, TN and FN of the classes by adding them. Afterwards, those values are used to calculate an "‘overall-precision"’ and "‘ overall-recall"’, which are used to calculate the global F_1-score. Let the classes be $c_1, ..., c_n$ and let TP_i, FP_i, FN_i be the values for class c_i. The micro-averaging is calculated with the following formula:

$$\text{Micro-Average} = \frac{2 \cdot \text{Precision}_{\text{sum}} \cdot \text{Recall}_{\text{sum}}}{\text{Recall}_{\text{sum}} + \text{Precision}_{\text{sum}}}$$

with

$$\text{Precision}_{\text{sum}} = \frac{\sum_{i=1}^{n} \text{TP}_i}{\sum_{i=1}^{n} \text{TP}_i + \sum_{i=1}^{n} \text{FP}_i}$$

and

$$\text{Recall}_{\text{sum}} = \frac{\sum_{i=1}^{n} \text{TP}_i}{\sum_{i=1}^{n} \text{TP}_i + \sum_{i=1}^{n} \text{FN}_i}$$

This approach gives an equal weight to the performance on every document.

On the other hand, macro-averaging averages between the F-scores of the classes. Let F_i be the F_1 score of class c_i. The macro-average is calculated via

$$\text{Macro-Average} = \frac{1}{n} \sum_{i=1}^{n} F_i$$

This approach gives an equal weight to the performance on every category, regardless how rare or how common a category is.

This thesis uses the macro-average F_1 score as measure. The reason for the macro-average F_1 will be given in Section 3.2, where the corpus used for this thesis is analyzed.

2.2 Classification Tasks

This section will introduce the two classification tasks in this thesis. First, the emotion detection task is introduced, secondly the differentiation between irony and sarcasm.

2.2.1 Emotion

To read emotions from faces is a central part of human life [KK98]. As an infant, humans look towards their mothers' facial expressions and use the emotional information to regulate their own behavior [KEBC86]. While emotion detection is done at least since humans exist, emotion detection in written words is a younger task. The following phrases, shown in Table 2.2 are examples of different emotions. They are taken from the corpus introduced in Chapter 3. Each phrase is a real Tweet and comprises a user-selected emotion.

Tweet	Emotion
"*had wonderful birthday*"	joy
"*i really wanna go walk around and find pokemon but i have a broken pinkie toe and can barely walk #pokemongo #brokentoe #kms*"	sad
"*#ransomware the #weaponization of #encryption has struck and confusion into the #hearts of pc users*"	fear
"*whenever my tongue touches a wooden popsicle stick i'm reminded that evil exists #hate #makethemplastic*"'	disgust
"*today i want to punch everyone in the face #upset #overit #fuckoff*"	anger
"*i got you a present*"	surprise
"*i felt it was more a look of stop asking me such idiotic questions*"	contempt

Table 2.2: Tweets representing Emotions

This seven emotions are studied by Paul Ekman. He identified, that there are emotions which are expressed and recognized by all humans in the same way. Those emotions are the seven from above: anger, contempt, disgust, fear, joy, sadness and surprise. This thesis tries to detect those emotions in different Tweets over time.

2.2.2 Irony and Sarcasm

Irony is an interesting phenomenon as it exposes the difficulty that current machines have in detecting rather the intention than the literal meaning of a sentence [BS14]. By the use of irony, people hide the real meaning of a statement saying the opposite of what they mean. Current automatic systems still struggle to detect this.

With the example (see above)

"*i love this cough i have*"

the author says that he loves his cough. Literally, this phrase means the same as the phrase for joy "*had wonderful birthday*". The difference is, that a cough is nothing which is liked due to the fact it is an illness. Thus, the sentence means the opposite of its literally mean. The phrase is a typical example of verbal irony: it is actually an ironic sentence.

The following phrase from the corpus represents another type of irony:

"*always remember that you are unique just like everybody else*"

This phrase is ironic because it describes that everyone is unique like all other people. It is not an ironic utterance but a description of an ironic situation. This type of irony is called situational irony.

One form of irony is sarcasm. Sarcasm is usually used to communicate impolitely criticism about the listener or the situation. It is usually used in situations provoking negative affect and is accompanied by disapproval, contempt, and scorn [San88].

One example from the corpus of sarcasm is the following phrase:

"*you're right donald what the states obviously needs is more guns and bigotry #avotefortheapocalypse*"

The author of this phrase wants to criticize the ideas of a person. The criticism is formulated in an sarcastic way, because the author literally agrees to Donald. Since he uses words like "*obviously*" and "*apocalypse*", the author depreciate Donald's ideas.

2.3 Classification in Dynamic Environments

To understand, what a dynamic environment is, stationary environments will be introduced first. In a stationary environment, there is no change to the classes of documents. Consider classification of quotes from the verse play " Maria Stuart" by Friedrich Schiller. The label of a quote is the name of the person in the play, who said it. Thus, the document "The voice of the majority is no proof of justice" belongs to the class "Maria Stuart". No matter what will happen, the label of this quote will be "Maria Stuart".

Now, consider the classification of the quotes to the page, they occur on. The quote can occur on different pages, depending on the edition of the play. Thus, the label of a quote is changing. Notice that the play keep the same, just the label is changing. That is an example of a dynamical environment. A document x belongs to class y at some point, and at another point, it belongs to class y'.

Topics naturally vary in those dynamical changing environments while the concepts are not necessarily changing. For example opinions on politicians can vary while the concept of politics do not change. Another example are changes in users interests on news. The concept of news remains the same, but the conditional distribution of interests for those users changes.

To determine those behavior, the data is divided into (time-) points $t_0, t_1, \dots$. Referring to the Maria Stuart example, the first point can be the first edition of the play, the second step the second edition and so on.

2.3.1 Concept Drift

In dynamical environments, data is expected to evolve over time. This behavior is expected to happen unexpectedly and unpredictable. Formally, concept drift between time point t_0 and time point t_1 can be defined as follows:

$$\exists x : p_{t_0}(x, y) \neq p_{t_1}(x, y)$$

where p_{t_i} denotes the joint distribution at time t_i between a document x and a class y. By assuming the Bayesian model (see Section 2.1.1), the following formula holds:

$$p(y, x) = p(y)p(x|y)$$

Hence, if $p(y, x)$ changes, the following cases can occur:

- the prior probabilities of classes $p(y)$ may change
- the class conditional probabilities $p(x|c)$ may change

As a result, the posterior probabilities of class $p(y|x)$ *may* change. This affects the prediction. From a predictive perspective only the changes that affect the prediction decisions require an adaptation. Thus, we distinguish between two types of concept drift:

- $p(y|x)$ changes. This kind of concept drift is called **real concept drift**.
- $p(y|x)$ does not change. This type if called **virtual drift**.

An example of the two types of concept drift is given in [GŽB+14]:

Consider an online news stream of articles on real estate. The task for a given user is to classify the incoming news into *relevant* and *not relevant*. Suppose that the user is searching for a new apartment, then news on dwelling houses are *relevant* whereas holiday homes are *not relevant*. If the editor of the news portal changes the writing style changes as well, but the dwelling houses remain *relevant* for the user. This scenario corresponds to virtual drift. If due to a crisis more articles on dwelling houses come out and less articles on holiday homes do, but the editor, the writing style, and the interests of the user remain the same, this situation corresponds to drift in prior probabilities of the classes ($p(c)$ changes). If on the other hand the user has bought a house and starts looking for a holiday destination, dwelling houses become *not relevant* and holiday homes become *relevant*. This scenario corresponds to the real concept drift. In this case the writing style and the prior probabilities stay the same. It may happen that all types of drifts may take place at the same time.

Figure 2.1 illustrates the types of drifts. Circles represent instances, red and green are representing members of different classes. The plot shows, that only the real concept drift changes the class boundary. Only in this situation, models have to adapt to the circumstances.

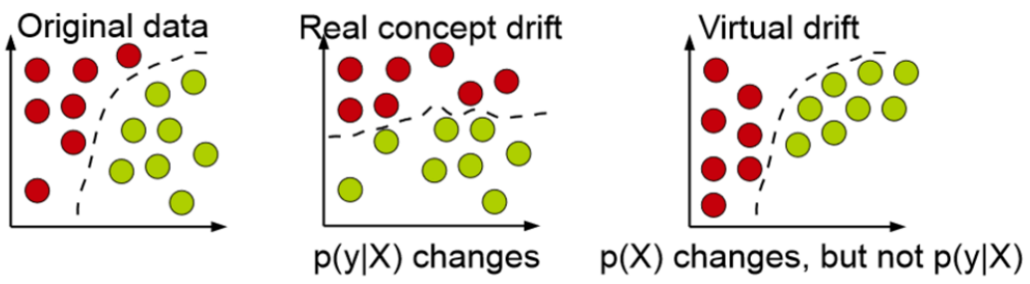

Figure 2.1: types of drifts [GŽB+14]

Predictive models that operate in those settings need to have mechanisms to detect and adapt to evolving data over time, otherwise their accuracy will degrade.

Changes in the data distribution over time may manifest in different forms, as illustrated in Figure 2.2 on one-dimensional data. Different colors are representing different sources.

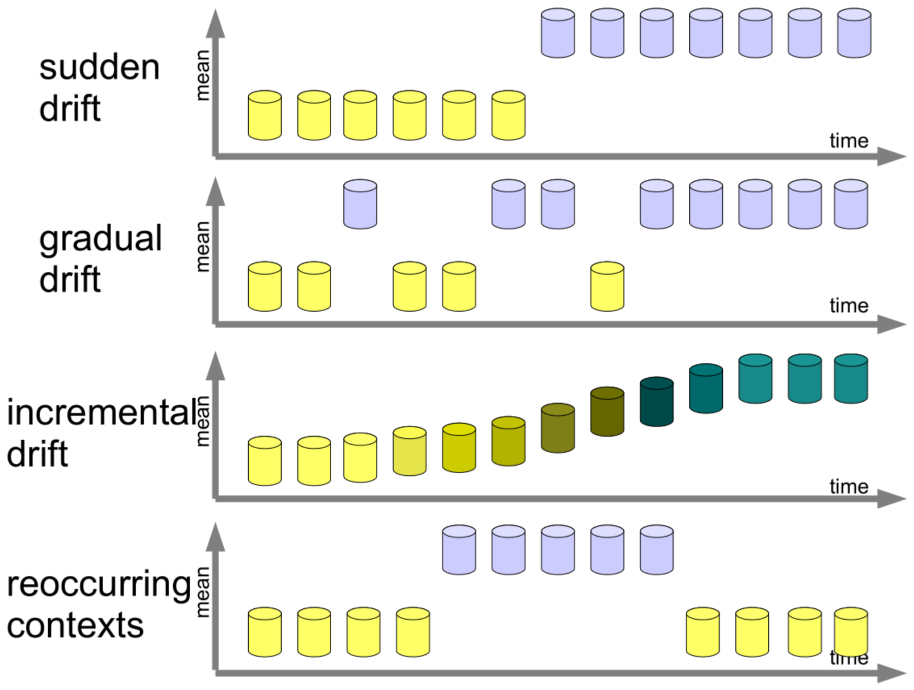

Figure 2.2: four different patterns of changes over time [Žli10]

In this data changes happen in the data mean. Assume, there are two different sources, S_1 and S_2. The different types of drift are ([GŽB+14], [Žli10]):

- **Sudden Drift**. This is the simplest pattern of a change [Žli10]. At a time point t_0, source S_1 is abrupt replaced by source S_2. For example, the user from the example above has bought a house and starts looking for a holiday destination, his interest sudden changes.
- **Gradual Drift**. This type of drift is referring to a period of time when both sources S_1 and S_2 are active. As time passes, the probability of sampling from source S_1 decreases, probability of sampling from source S_2 increases.
- **Incremental Drift**. If the gradual drift includes more than two sources and the difference between the sources is very small, this kind of drift is called incremental. The drift is only noticed when observing at a longer time of period. Figure 2.2 shows different sources in the incremental type of drift.

- **Reoccurring Concepts Drift**. This big type of drift is when previously active concepts reappear after some time. It differs from common seasonality notion in a way that it is not certainly periodic, it is not clear when the source might reappear.

2.3.2 Domain Adaptation

Domain adaptation considers the setting in which the training data and testing data are sampled from different distributions [GBB11].

Assume, we have two sets of data: a *source* domain S providing labeled training instances and *target* domain T providing instances on which the classifier is meant to deployed. We do not make the assumption that these are drawn from the same distribution, but rather that S is drawn from a distribution p_S and T from a distribution p_T. The learning problem consists in finding a function realizing a good *transfer* from S to T i.e. it is trained on data drawn from p_S and generalizes well on data drawn from p_T.

2.4 Classifier Models to handle Concept Drift

An overview of learning under concept drift and a summary of types of concept drift is provided in [Žli10]. It presents the different components of adaptive learning systems. Such a system can roughly divided into three parts:

- The **memory management** defines which information will be stored. Depending on the strategy, different data will be stored and discarded. One possible assumption of a strategy is, that the most recent data is the most informative for the current prediction.
- The **learning component** refers to the mechanisms to update the predictive models from evolving data. There are two different methods for learning: *Retraining*, which means to discard the current model and to build a new one, and *incremental*, which updates the model.
- The **adaptation strategies** manage the adaptation of the predictive model. There are two types of strategies: The blind strategies adapt the model without explicit detection of changes. The informed strategies are reactive, they use a trigger, which will be flagged if a change occurs.

The next two subsections introduce the two classification systems which are evaluated in this thesis. Section 2.4.1 introduces the Remove Mutual Information Change Model and Section 2.4.2 presents the Dynamic Weight Majority Mode, the ensemble based approach.

2.4.1 Remove Mutual Information Change Model

Consider a situation with two different time points t_0 and t_1, where the documents in those time steps are the same, but their classes are different. Concept drift happens between t_0 and t_1. If two Maximum Entropy classifiers, with word-bases features, are built one for t_0 and one for t_1 what will happen to the weights of the features? Since the time steps contains the same documents, the features of the two different classifiers are the same. But, because of concept drift, for specific features, the weight in the classifier for t_0 cannot be the same than the weight in the classifier for t_1.

Concept drift will be represented in the weight of the features in some way.

2.4.1.1 Mutual Information

Mutual Information of two random variables is a measure of the mutual dependency between the two variables. It quantifies the amount of information obtained about one random variable, through the other random variable.

Consider the example of a classification task in Table 2.3. There are two possible word-based features: One with word A and one with word B.

Document text	Class
A	C
A B	D

Table 2.3: Example for Mutual Information

To classify the documents into their correct classes, the feature with word A contains no information, because word A does not contain information about the class the document belongs to. On the other hand, word B does only occur in class D. By that, the feature with word B holds exactly when the document belongs to class D. Thus, word B contains more information about the class than word A.

By that, the feature with word B will be better for classifying than the feature with word A. One measure, which feature is "better" for classifying, is mutual information.

Mutual information for random variables X and X can be expressed as [Bat94]

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

In a classification task, Y is typically the random variable for the classes and X the random variable of the features. For instance, the values for the features f_A and f_B of the classification task in Table 2.3 are

$$\begin{aligned}
I(f_A, Y) &= p(f_A = 0, Y = C) \log \frac{p(f_A = 0, Y = C)}{p(f_A = 0)p(Y = C)} \\
&+ p(f_A = 0, Y = D) \log \frac{p(f_A = 0, Y = D)}{p(f_A = 0)p(Y = D)} \\
&+ p(f_A = 1, Y = C) \log \frac{p(f_A = 1, Y = C)}{p(f_A = 1)p(Y = C)} \\
&+ p(f_A = 1, Y = D) \log \frac{p(f_A = 1, Y = D)}{p(f_A = 1)p(Y = D)} \\
&= 0 + 0 + 0.5 \log \frac{0.5}{1 \cdot 0.5} + 0.5 \log \frac{0.5}{1 \cdot 0.5} \\
&= 0.5 \log 1 + 0.5 \log 1 = 0
\end{aligned}$$

and

$$\begin{aligned}
I(f_B, Y) &= p(f_B = 0, Y = C) \log \frac{p(f_B = 0, Y = C)}{p(f_B = 0)p(Y = C)} \\
&+ p(f_B = 0, Y = D) \log \frac{p(f_B = 0, Y = D)}{p(f_B = 0)p(Y = D)} \\
&+ p(f_B = 1, Y = C) \log \frac{p(f_B = 1, Y = C)}{p(f_B = 1)p(Y = C)} \\
&+ p(f_B = 1, Y = D) \log \frac{p(f_B = 1, Y = D)}{p(f_B = 1)p(Y = D)} \\
&= 0.5 \log \frac{0.5}{0.5 \cdot 0.5} + 0 + 0 + 0.5 \log \frac{0.5}{0.5 \cdot 0.5} \\
&= 0.5 \log 2 + 0.5 \log 2 = \log 2 = 1
\end{aligned}$$

The values match the results above: The feature with word A contains information about the class (if we know if feature f_A holds, we know the class), while the feature with word B does not.

2.4.1.2 Algorithm

The algorithm of this approach is given in Algorithm 2.1. There are currently no articles which use some kind of this approach.

For classifying, the base classifier is used to classify the input. If the input is in addition training data, a new classifier is created which uses the input as training data. The last classifier which is created in this way is stored, its training data is the last input. The features of these two classifiers are extracted and their mutual information value is calculated (on the corresponding data; i.e. the current created classifier features use the current input data for calculation). Features, which only occur in one of the two feature sets are discarded. The remaining features are sorted by change in the mutual information value. This list of features is added into the list of current removed features. The features with the highest change in mutual information remain in this list, the others are discarded (bases on the defined percentage).

The words which contain to these features are removed from the input data. The modified input data is given as training data to the base classifier.

2.4.2 Ensemble based Model

Ensemble methods combine a set of classifiers to construct a new classifier that is (often) more accurate than any of its component classifiers [VV03]. It is a method of weighting and combining the decisions of "experts", each of which is a learning method [KM07].

A dynamic weighted majority algorithm is given in [KM07], which is modified for a test set containing time steps. The modified version is shown in Algorithm 2.2. It is based on the Weighted Majority algorithm in [LW89] with additional mechanisms for creating and deleting experts dynamically in response to changes in performance. Hence, this method is called *Dynamic Weighted Majority*.

Blum evaluated variants of Weighted Majority and Winnow on a calendar-scheduling task and results suggested that the algorithms responded well to concept drift and executes fast enough to be useful for real-time applications [Blu97]. However, using pairs of features requires $\binom{n}{2}$ experts, where n is the number of relevant features (i.e., attribute-value pairs), which makes the direct application of these implementations inappropriate for most data mining problems. In one case, when learning the schedule preferences of one user using 34 attributes, Weighted Majority and Winnow required 59,731 experts and specialists, respectively.

The algorithm maintains as its concept description an ensemble of learning algorithms, each referred to as an expert and each with an associated weight. Given an instance,

Algorithm 2.1 Remove Mutual Information Change

```
input:
        phase of training: $(x, y)_i, i \in \{1, ..., n\}$, time step containing $n$ documents
        phase of testing: $(x)_i, i \in \{1, ..., n\}$
Data:
        classifier: base classifier
        removedFeatures: list of removed features with their mutual information
        value, sorted ascending by this value
        $n$: number of features to remove
        tmpClassifier: temporary classifier
        lastFeatureList, currentFeatureList, changeList: lists of features with their
        mutual information value
        $p$: percentage of features to remove, $0 \le p \le 1$
for i = 1, ..., n do
    output classifier.classify($x_i$)
end
if phase of training then
    tmpClassifier = classifier.createNewClassifier()
    tmpClassifier.train($(x, y)_i, i \in \{1, ..., n\}$)
    oldFeatureList = currentFeatureList
    currentFeatureList = calcMutualInformation(tmpClassifier.getFeatures(),
     $(x, y)_i, i \in \{1, ..., n\}$)
    changeList = sortDescendingInChange(oldFeatureList, currentFeatureList)
    $n = \max\{$removedFeatures.size(), changeList.size()$\}$
    for i = 1, ..., $p\cdot$ changeList.size() do
        removedFeatures.insert(changeList.get(i))
    end
    while removedFeatures.size() $> n$ do
        removedFeaturs.remove(0)
    end
    oldFeatureList = currentFeatureList
    classifier.addTrainingsdata($(x, y)_i, i \in \{1, ..., n\}$)
    classifier.setIgnoredFeatures(removedFeatures)
    classifier.train()
end
```

Algorithm 2.2 Dynamic-Weighted-Majority (modified from [KM07])

```
input:
        phase of training: $(x, y)_i, i \in \{1, ..., n\}$, time step containing $n$ documents
        phase of testing: $(x)_i, i \in \{1, ..., n\}$
Data:
        $\beta$: factor for decreasing weights, $0 \leq \beta < 1$
        $\theta$: threshold for deleting experts
        {e, w}: set of $m$ experts with their weights
        $\Lambda, \lambda$: global and local predictions
        $\vec{\sigma} \in \mathbb{R}^{|\text{classes}|}$: sum of weighted predictions for each class
for i = 1, ..., n do
    $\vec{\sigma}_\lambda \leftarrow 0$
    for j = 1, ..., m do
        $\lambda \leftarrow$ Classify($e_j, x_i$)
        if phase of training then
            if $i = 1 \wedge \lambda \neq y_i$ then
                $w_j = \beta w_j$
            end
        end
        $\sigma_\lambda \leftarrow \sigma_\lambda + w_j$
    end
    $\Lambda \leftarrow$ argmax$_j \sigma_j$
    if $i = 1$ then
        $w \leftarrow$ Normalize-Weights($w$)
        $\{e, w\} \leftarrow$ Delete-Experts($\{e, w\}, \theta$)
        if $\Lambda \neq y_i$ then
            $m \leftarrow m + 1$
            $e_w \leftarrow$ Create-New-Expert()
            $w_m \leftarrow 1$
        end
    end
    if phase of training then
        for j = 1, ..., m do
            $e_j \leftarrow$ Train($e_j, \vec{x_i}$)
        end
    end
    output $\Lambda$
end
```

the performance element polls the experts, each returns a prediction for the instance. Using these predictions and expert weights, the algorithm returns the class label with the highest accumulated weight as the global prediction.

The learning element, given a new training example, first polls each expert in the manner described previously. If an expert predicts incorrectly, then its weight is reduced by the multiplicative constant β.

Then, the algorithm determines the global prediction. If it is incorrect, it creates a new expert with a weight of one. The algorithm normalizes expert weights by uniformly scaling them such that the highest weight will be equal one. This prevents any newly added experts from dominating the decisions making of existing ones. The algorithm also removes experts with a weight less then the user-defined threshold θ. Finally, it passes the training example to each expert's learning element. Note that normalizing weights and incrementally training all experts gives the base learners an opportunity to recover from concept drift.

By passing the training examples to each expert's learning element, the members of the ensemble do not stop learning after being formed. If the experts would not receive new data, it implies that a fixed period of time will be sufficient for learning all target concepts. In addition, if concepts drift during this fixed period of time, the learning component of the classifier may not be able to acquire the new target concepts.

Classification tasks with a large amount of data require a parameter, which governs the frequency the algorithm creates experts, removes them, and updates their weights (i.e. reduction and normalizing). The original algorithm in [KM07] uses a user-defined periodicity p.

Here, to make use of the time steps, the algorithm uses the first document of each time step. This is the only modification in the algorithm.

2.5 Related Work

A comparison of three different approaches of domain adaptation strategies to handle concept drift is given in [CSAR14] and is presented in Section 2.5.1.

Emotion detection in Twitter is done in [RRJ+12] and is described in Section 2.5.2.

A classification task between the classes irony and sarcasm is done by Ling and Klinger in [LK16] and is presented in Section 2.5.3. They researched which features have an impact on classification between irony and sarcasm.

2.5.1 Concept Drift Awareness in Twitter Streams

J. Costa, C. Silva, M. Antunes and B. Ribeiro propose three different models: a time-window model, an ensemble-based model and an incremental model. Since less is known about the types of drift that can occur in Twitter, they simulated different types of drift by artificially time stamping real Twitter messages in order to evaluate and validate their strategies. Results are so far encouraging regarding learning in the presence of drift, along with classifying messages in Twitter streams.

The base classifier used is the Support Vector Machine (SVM).

2.5.1.1 Evaluated Models

The three models to evaluate are:

- a time-window model
- an ensemble-bases model
- an incremental model

The **time-window model** uses only the last time step to predict the next one. Instead of storing all seen data, it only needs the last data of the last time step. The key is to select an appropriate window size. If it is short, the classifier adapt fast when the concept changes, but during stable periods it will worsen the performance. A large window size enables a better performance during stable periods but will adapt slower to concept changes.

The **incremental model** considers all data seen before. It updates its collection of the data in an incremental manner. As the stored data increases over time, it could lead to storing issues.

The **ensemble-based model** uses a combination of different classifiers and adjust the relation between them over time. For each collection of data a classifier will be trained. The prediction function of the ensemble uses the classification of all classifiers and weight between them. There are many different strategies for weighting. In this experiment, majority voting was used.

2.5.1.2 Corpus

Using artificial time-stamping of real Tweets messages to create a corpus, these approaches are evaluated and validated.

The authors intend to represent four major types of concept drift, namely sudden, gradual, incremental and reoccurring. For this purpose, a table for 24 time steps of the distribution of those types to represent concept drift is generated. Tweets were timestamped to fit in this distributions.

Additionally, Tweets were preprocessed as follows:

- stopwords were removed and
- stemming was used

in order to reduce the number of features.

2.5.1.3 Results

The results show, that the incremental approach outperforms the overall classification of the time-window model and the ensemble model, except for one type of sudden drift. This particular drift is a fast occurring drift, it appears and disappears rapidly. The other tested sudden drift had a longer period, and in this case, the incremental model was outperforming the other approaches. Bottom line: the incremental approach was able to handle all drifts better than the other approaches, except for really fast ones. This behavior was expected, since this model keeps all seen data in memory. According to the authors of the paper, the reason, why the ensemble based model could not perform as well as the incremental model, is the combined decisions induce errors on the overall decision, according to the authors of the paper

2.5.2 EmpaTweet: Annotating and Detecting Emotions on Twitter

The authors of this paper created a corpus Tweets which belong to one of the seven emotions:

1. anger
2. disgust
3. fear
4. joy

5. love
6. sadness
7. surprise

The aim of the paper is to answer the following questions.

- How are the emotions distributed in this corpus?
- Is the emotion distribution similar to a distribution in other emotion-annotated corpora?

In addition, the authors trained a classifier on this corpus to discover emotions in Tweets.

2.5.2.1 Corpus Creation

The corpus consists of Tweets on a variety of popular Twitter topics. The emotions of Tweets are manually annotated. The topics were chosen based on the expectations of the authors, which emotions will be present in topical Tweets. Seven chosen emotions, based on Ekman's six basis emotions and in addition, love, are used as labels (see the enumeration above). 14 topics were selected by the authors. They crawled for corresponding hashtags (for example #valentine for Valentine's Day). Preprocessing of Tweets was done in the following way:

1. Removing casing, punctuation, hashtags and URLs
2. Run Dice's coefficient with a threshold of 0.8

They created their own annotation tool to maximize annotator efficiency. Tweets can belong to any number of emotions. The corpus contains 7,000 Tweets.

2.5.3 Classification of Irony and Sarcasm

J. Ling and R. Klinger aimed at answering the following questions:

1. Is it possible to predict if a Tweet has been labeled to be ironic or sarcastic by the author (without having access to the real label)?
2. Which features have an impact on making this prediction?
3. Can we get qualitative insight with this approach what is considered by users as sarcasm or irony?

Different classification methods, namely support vector machine, decision trees and Maximum Entropy, are used.

2.5.3.1 Corpus

In order to answer those questions, they created a corpus of 99000 English Tweets. Half of the them contain the hashtag #irony or #ironic and the other half #sarcasm or #sarcastic. The Tweets used are real Tweets, crawled between July and September 2015.

The corpus was created by discarding the following Tweets:

- Tweets which are Retweets (Tweets marked with "RT")
- Tweets shorter than five tokens
- Tweets which are near-duplicates (based on token-based Jaccard similarity with a threshold of 0.8)
- Tweets which contain ironic **and** sarcastic hashtags

2.5.3.2 Analyzed Features

The selected features to analyze if they have an impact on classification are

- word features for all words in the Tweets
- 50.000 most frequent bigrams
- other features like emotes in the message, capitalization words in the message, ...

2.5.3.3 Results

Since the Maximum Entropy model outperforms the other two classifiers, Ling and Klinger analyzed the impact of features in this model.

Running two different classifiers, one only using domain-specific features and the other in addition word-based features, the accuracy of the second classifier is about 15% higher than the accuracy of the first classifier. This leads to the assumption, that the words in the Tweet have a high impact on the classification performance.

3 Corpus

In this thesis, the documents to classify are Tweets from Twitter. In order to know the true labels of the documents, we assume, that the hashtags (the words starting with #) are the classes of the document. We assume, that the users classify their document correctly.

This chapter is organized as follows. The first part presents how the corpus is created, the second part describes the analysis of the corpus.

3.1 Creation

For collecting Tweets, two tools are used:

- Twitter4J is an unofficial Java library for the Twitter API, which allows to crawl online for specific key words [Yam07].
- GetOldTweets is a project which uses the Twitter API to crawl for Tweets from the past. It exists for Java and for Python [Hen14].

Using GetOldTweets, 42,554 Tweets were crawled between 28th March 2016 and 29th May 2016. With Twitter4J, 680,422 Tweets published between 30th May 2016 and 28th July 2016 were collected. Thus, the corpus consists of 722,976 Tweets published worldwide between 28th March and 28th July 2016, which is a period of 123 days.

As keywords for Tweets to save, following hashtags with their related classes are used:

Emotions

- anger: #anger, #angry, #rage, #fury, #mad, #furious
- contempt: #contempt, #disprize, #contemn
- disgust: #disgust
- fear: #fear, #frightened, #panic
- joy: #joy, #pleasure, #happiness, #happy

- sad: #dolefulness, #dolor, #mourning, #sad, #sadness, #sorrow, #grief, #misery, #afraid
- surprise: #surprise, #surprised

Irony and Sarcasm

- irony: #irony, #ironic
- sarcasm: #sarcasm, #sarcastic

After crawling, Tweets were separated into one corpus of emotions and one of irony and sarcasm, based on the keywords above. Additionally, they were sorted into time steps, where each time step is one day.

Because Tweets can contain more than one hashtag, it is possible that they belong to more than one class. There are two different approaches to handle this behavior:

1. **Multilabeling**. A classifier classifies document x correct in class y, if, and only if, document x contains a hashtag of class y.
2. **No multilabeling**. To ensure, that a document only belongs to exact one class, the corpus has to be modified. One possibility is to remove all Tweets which contains at least two different hashtags, where those hashtags belong to different classes.

Approach one leads to the question "*to which class fits a given document?*", while approach two leads to the question "*to which class belongs a given document?*". Approach two forces classifiers to classify into exactly one class. In this thesis, we try to detect and to differ between emotions, for which reason approach two is used.

Tweets can refer to users (the notation "@X" means a reference to user X) and links to websites. There are many users and websites to which Tweets can be linked, thus there are many different words in the document texts. We assume, that there are many users and links which are used in Tweets, but occur very rarely. To avoid many features, which will practically never hold, references to users (in Tweets words which begin with "a") and links are replaced with [USER] and [URL]. This approach uses two meta features to reduce the amount of features.

Many of the crawled Tweets are spam and/or advertising. Here are three examples from the corpus:

1. *photo edit by hazy reign [USER] #selfie #me #love #joy #followme #friends #summer [URL]*
2. *save $796 rage chisel tip 2-blade 100 grain practice tip broadheads (3 pack) #anger [URL]*

3. *possible to enjoy life after the #loss of a loved one [URL] #depression #sad #rebirth #healing #iheartradio #widow*

Tweet one and two are advertising, both of them want users to visit their URL. Tweet one is equipped with many hashtags in order to be found by more people. But the Tweet does not represent the emotion joy. Nor does Tweet 2 represent anger. Tweet 3 is still advertising but is related to sadness. To avoid confusing the classifiers with Tweets like Tweet 1 or 2, which belong to classes but do not represent those, those Tweets should be removed.

How can those Tweets be found? All of those 3 Tweets occur more than 300 times in the corpus and they are posted more than three times a day. Thus, removing of all duplicates reduces the number of those Tweets. But there are Tweets like this one:

photo edit by hazy reign [USER] #selfie #me #love #joy #followme #friends #summer #sea [URL]

This Tweet differs from the first Tweet above, because it contains in addition the hashtag sea. The semantic meaning of this Tweet is the same than the meaning of the first Tweet above. Thus, it should also be removed. In order to detect those semantic equivalent Tweets, a word based Jaccard Index with the threshold 0.95 was run on every day for every corpus to remove spam Tweets.

Another type of duplicates are Retweets. A Retweet is a re-posting of a Tweet [Inc16]. Consequently, Retweets should be removed in order to eliminate duplicates. Thus, all Tweets which contain "RT" are removed from the corpus.

Lastly, to prevent the classification task of being obvious, the hashtags which specify the class of the Tweet are removed. As an example, in the Tweet above the word "#joy" will be removed.

3.2 Analysis

This section presents the two corpora. First, the corpus of emotion will be analyzed in Section 3.2.1. Secondly, the corpus of irony and sarcasm will be analyzed in Section 3.2.2.

3.2.1 Emotion

The final corpus for emotions contains over 500,000 Tweets. Table 3.1 shows the number of Tweets per class in the corpus, sorted in descending order. Tweets are distributed unbalanced in the corpus. Class "joy" contains nearly 390,000 Tweets, while all other classes together contain about 180,000 Tweets. Thus, more than the halve of the Tweets contain to class "joy". Classes "disgust" and "contempt" contain only a few Tweets. Class "contempt" does only contain 190 Tweets, that is just one and a half Tweet per day. Those classes would not be considered using the micro F_1 score, since they occur greatly less frequent than the other classes. This is the reason why this thesis will use the macro F_1 score.

The unbalance of Tweets per day could be caused by two different reasons. First, the Twitter API could provide less Tweets of specific hashtags. Secondly, users could use those hashtags less than the others.

class	total Tweets	avg per day
joy	387,552	3,176.66
sad	90,831	744.52
anger	42,251	346.32
fear	32,160	263.61
surprise	17,050	139.75
disgust	1,181	9.68
contempt	190	1.56
total	571,215	4,682.09

Table 3.1: Emotion Tweets distribution

Figure 3.1 shows the distribution of Tweets per day. In the period from March 29th 2016 to May 29th, there are never more than 1,000 Tweets per day. The maximum of Tweets at a day is 711 Tweets at May 16th. The average of Tweets per day in this time period is 432. In this time period, Tweets are crawled with the GetOldTweets tool introduced in the previous section. After this time period, the number of Tweets per day rapidly increases. There are averaged 9,221 Tweets per day. The maximum amount of Tweets at a day is 11,756 at June 21th.

Since the second part of the time steps contain many Tweets, the first part is for better visibility illustrated in Figure 3.2.

The graphs let identify the pattern of the Tweet distribution. The percentage of Tweets per class at a day seems to be nearly constant. There are wild exceptions or behaviors

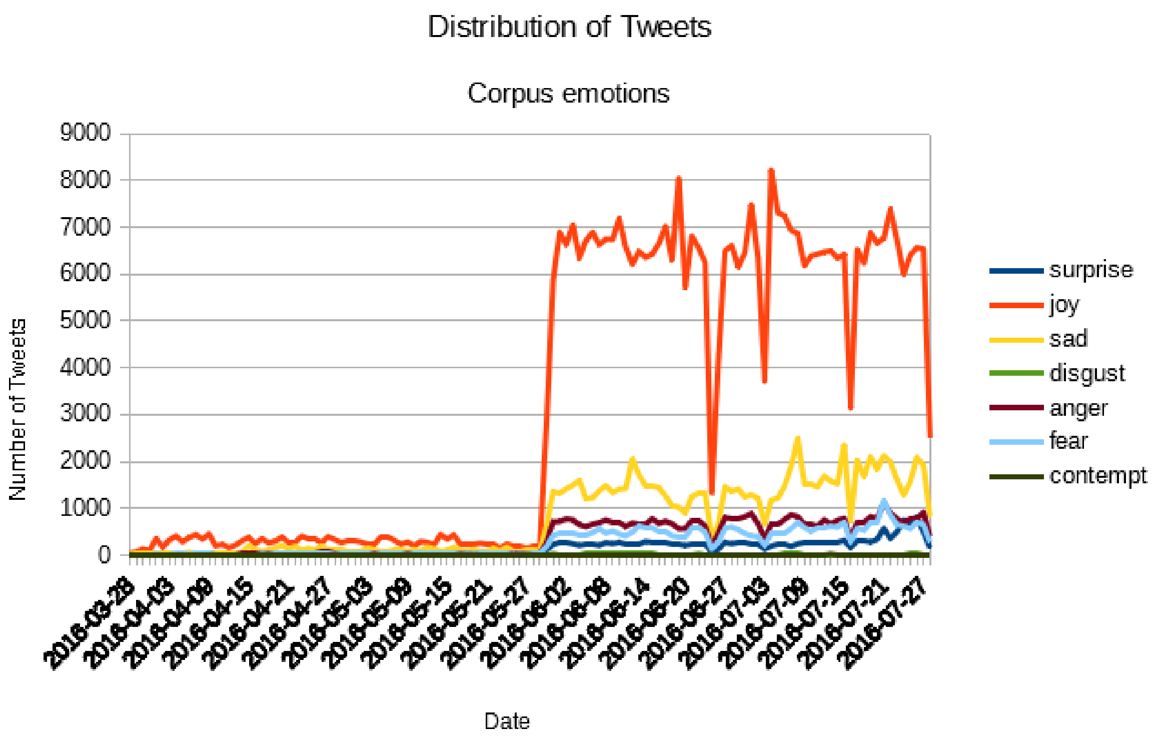

Figure 3.1: emotion distribution

where Tweets of one class are only present at a specific time period and do not occur on other days.

3.2.2 Irony and Sarcasm

The corpus for irony and sarcasm contains about 75,000 Tweets. Table 3.2 shows the Tweets per class in the corpus, sorted in descending order. The Tweets are distributed nearly balanced in the corpus. There are about 10% more Tweets of the class "irony" than Tweets of the class "sarcasm". Reasons for the difference are given in the analysis of the emotion corpus: the Twitter API could provide less sarcasm Tweets than irony Tweets or the users use sarcasm hashtags less than irony hashtags.

Figure 3.3 shows the distribution of Tweets per day. Like in the figure of the distribution of Tweets in the emotion corupus per day, the total amount of Tweets per day increases at May 30th, because of the changing crawl method.

There are no long time periods where only one class is present, the Tweet distribution is overall the same for both classes. In the first part, there are slightly more Tweets of

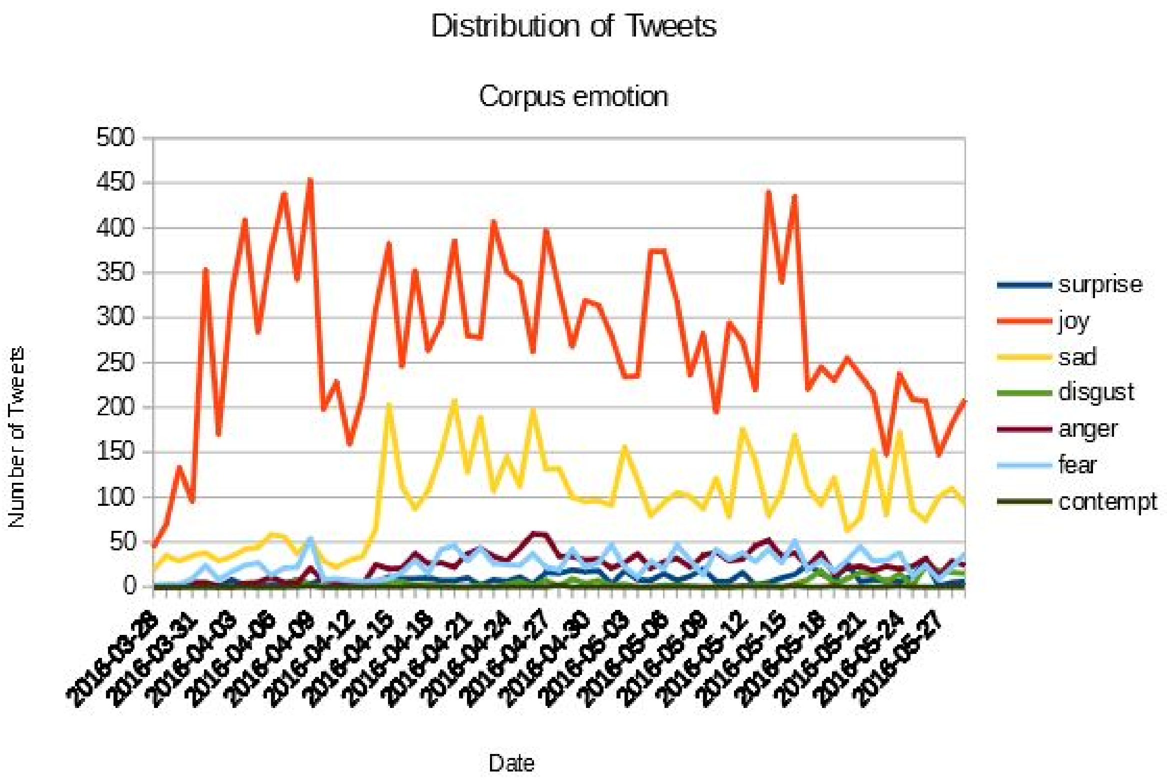

Figure 3.2: emotion distribution, first part

class	total Tweets	avg per day
irony	39,834	334.74
sarcasm	35,694	299.95
total	75,528	634.69

Table 3.2: Irony and Sarcasm Tweets distribution

the class "irony" while in the second part, there are slightly more Tweets of the class "sarcasm".

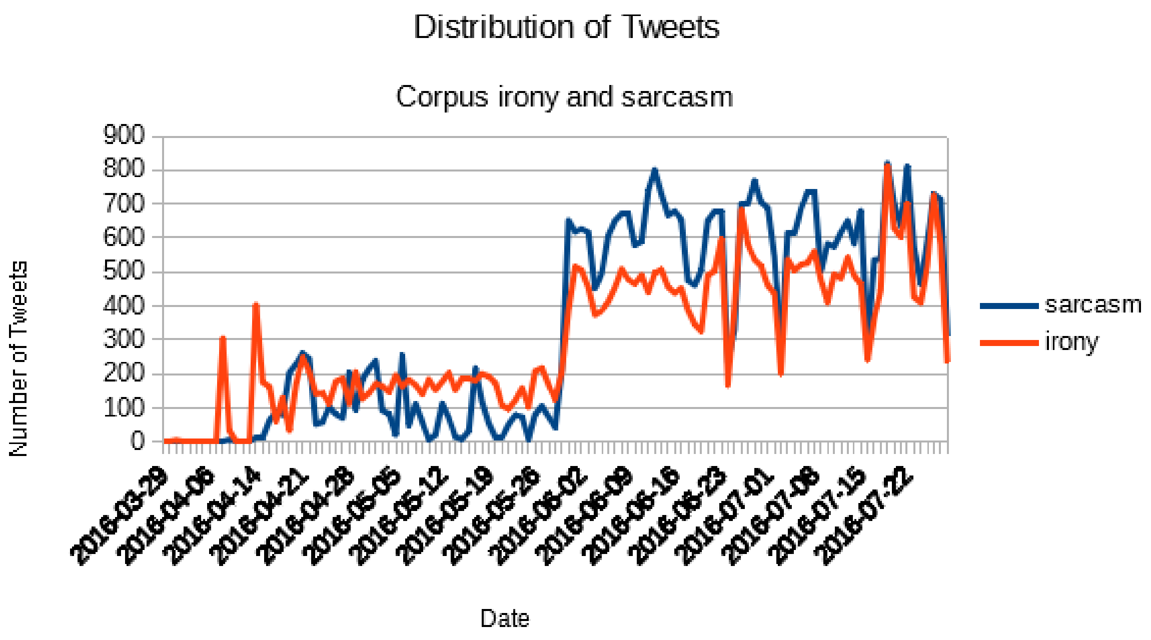

Figure 3.3: irony and sarcasm distribution

4 Methodology

This chapter is divided into four sections. Section 4.1 describes, how the corpus introduced in Chapter 3 is used to create experiments to evaluate the classifiers. Section 4.2 explains, how each experiment is evaluated. These evaluations carried together in order to evaluate a classifier. This process is described in Section 4.3. Lastly, Section 4.4 explains, how the Remove Mutual Information Change algorithm introduced in Section 2.4.1 is evaluated.

4.1 Experimental Setup

The corpus is divided into steps of time, where each step is one day. The structure is as follows:

$$t_0 \to t_1 \to t_2 \to ... \to t_n$$

where n is the number of steps in the corpus. In order to run as many experiments as possible, each Classifier is evaluated with different starting steps. For each experimental setup, a number of training steps is chosen. For example, to evaluate a single classifier with m steps of training, different experiments are run. Table 4.1 shows every single experiment which is run in this case.

At each step of time, a new experiment is started. For the evaluation of a classifier with a specific amount m of steps of training, $n - m + 1$ experiments are run.

4.2 Evaluation of a single Experiment

Let the steps of training be $\{t_0, t_1, ..., t_{m-1}\}$ and the steps of testing be $\{t_m, t_{m+1}, ..., t_n\}$ for $0 \leq m \leq n$. An experiment is evaluated in the following way: first, the classifier is trained with the first step of training, t_0. Afterward, an iteration loop over the remaining

Experiment	Starting Step	Steps of Training	Steps of Testing
1	t_0	$\{t_0, ..., t_{m-1}\}$	$\{t_m, ..., t_n\}$
2	t_1	$\{t_1, ..., t_m\}$	$\{t_{m+1}, ..., t_n\}$
3	t_2	$\{t_2, ..., t_{m+1}\}$	$\{t_{m+2}, ..., t_n\}$
4	t_3	$\{t_3, ..., t_{m+2}\}$	$\{t_{m+3}, ..., t_n\}$
⋮	⋮	⋮	⋮
$n-m-1$	t_{n-m-2}	$\{t_{n-m-2}, ..., t_{n-3}\}$	$\{t_{n-2}, t_{n-1}, t_n\}$
$n-m$	t_{n-m-1}	$\{t_{n-m-1}, ..., t_{n-2}\}$	$\{t_{n-1}, t_n\}$
$n-m+1$	t_{n-m}	$\{t_{n-m}, ..., t_{n-1}\}$	$\{t_n\}$

Table 4.1: Experiments run to evaluate a classification system

steps of training is executed: $t_1 \rightarrow t_2 \rightarrow ... \rightarrow t_{m-1}$. At each time step, first the classifier has to classify each document and then the classifier gets all documents as new training data. After the phase of training, the steps of testing are evaluated in the same way, but without giving them as new training data. For each time step, Precision and Recall are evaluated (for each possible class of the documents).

4.3 Evaluation of a Classifier

Each experiment gives an evaluation of the accuracy of a classifier at specific steps of time.

Step of Time	t_0	t_1	t_2	t_3	t_4	t_5
Experiment 1	t	x	x	x	x	x
Experiment 2		t	x	x	x	x
Experiment 3			t	x	x	x
Experiment 4				t	x	x
Experiment 5					t	x

Table 4.2: Time Step Coverage with Experiments

Table 4.2 shows which steps of time are covered by which experiment for $n = 5$ and $m = 1$. "t" means the corresponding time step is used for training, "x" means the corresponding time step is used for testing. In order to evaluate, how accurate a classifier system classify after a specific amount of time steps, the experiments are shifted in a way Table 4.3 shows.

x Axis	-1	0	1	2	3	4	5
Experiment 1		t	x	x	x	x	x
Experiment 2		t	x	x	x	x	
Experiment 3		t	x	x	x		
Experiment 4		t	x	x			
Experiment 5		t	x				

Table 4.3: Shifted Experiments

The x-axis in this table represents the distance to the end of phase of training. $x = 1$ is the first time step which does not belong to the phase of training. Finally, for each value for x, the average value and the variance are calculated (for Precision, Recall and the F_1-score). These values are presented in a chart which is organized like Table 4.3. As an example, see Figure 5.1 in Chapter 5. The red line marks the end of the phase of training. The blue points represent the average values, while the yellow pipe shows the variance.

4.4 Evaluation of the RMIC Algorithm

To evaluate different percentages of removed features, the algorithm is run with different values for p. 99 different values are used: $\{1\%, 2\%, 3\%, ..., 99\%\}$. The results of the classifications are illustrated in a single figure in the following way. The x-axis represents p, the y-axis represents the accuracy (F_1-score). The figure is filled with 3 lines, one for the accuracy at the first step, one for the 10th step and one for the 100th step after the phase of training.

5 Concept Drift and Emotion Detection on Twitter

In this chapter, the performance obtained on the corpora introduced in Chapter 3 using the approaches introduced in Section 2.4 is evaluated.

The chapter is divided into three parts. The first part introduces the hypotheses for this thesis. The second part presents the experimental setup and the results of the experiments. Lastly, the third part uses the results of the experiments to confirm or disprove the hypotheses.

5.1 Hypotheses

The aim of the experiments is to confirm or to disprove the hypotheses from the introduction:

1. The accuracy of the incremental model decreases the more the distance to the phase of training increases.
2. The DWM algorithm achieves a greater accuracy than the incremental model for steps with a great distance to the phase of training.
3. For some percentage, the RMIC algorithm with Maximum Entropy as base classifier achieves a greater accuracy than the incremental model for steps with a great distance to the phase of training.
4. The RMIC algorithm removes features which are connected to events happening in the duration of the corpus.
5. The accuracy of the Naive Bayes model decreases when using the mutual information change remove method for domain adaptation

Concept drift should happen in the corpus. The incremental model is not able to deal with concept drift, because it has no domain adaptation strategies. Consequently, hypothesis 1 should hold.

Hypothesis 2 is the assumption, that the results from the concept drift awareness in Twitter streams paper (see section 2.5.1 in the related work section) hold in a corpus of real Tweets. They compared the incremental model with the ensemble model on a corpus containing artificially time-stamped real Tweets messages in order to create different types of concept drift. The question is, if the results of the paper still hold in real Tweets without enforce explicit concept drift.

The assumption for hypothesis 3 is, that by removing the features with the highest change in mutual information, the Maximum Entropy model is able to keep the accuracy as high as without the remove. In addition, the features, which are affected by concept drift are removed with this strategy, and the "base" features remain. Since the Maximum Entropy model allows to recalculate the weights of the "base" features after removing the other features, these fend for a high accuracy, especially after a long time period.

In addition, events could lead to concept drift. The type "sudden" of concept drift (see Section 2.3.1) should happen for events that happen unexpected or suddenly. Other events, like the "Brexit" [DS16] can lead to gradual or incremental drift. Features, which contains information about events (like "earthquake" for an earthquake) should exhibit changes in mutual information. Therefore, these features should be removed by the RMIC algorithm. This assumption lead to hypothesis 4.

The assumption of hypothesis 5 is, that the Naive Bayes model has no possibility for interaction between the words. It has no possibility like the Maximum Entropy model to adjust the weights of the features to enable connections between features. If we remove terms from the Naive Bayes model, it is not able to compensate the removed words. Thus, as the hypothesis, the accuracy decreases with interfere the model.

5.2 Results

This chapter presents the results of the experiments. All classifier models are evaluated in the emotion and irony & sarcasm corpus as introduced in Chapter 3. Two different experiments are run for every model: one with 14 steps of training and one with 61 steps of training (halve of the steps in the corpus).

Because the thesis focuses on the evolve of the accuracy after a long period of time, only the results with 14 steps of training are presented in this chapter. The figures with 61 steps of training are in Appendix A. The results of those classification tasks correlate to the results with 14 steps of training, except for the absolute values of the accuracy. Because the models have 47 additional steps of training and a larger set of training data, the absolute accuracy is higher (about 0.05 higher than with 14 steps of training).

How to read the figurs is explained in Section 4.3.

5.2.1 Baseline

For the baseline, the incremental model and the random classification are used.

5.2.1.1 Incremental Model

The line graph in Figure 5.1 shows the accuracy of the classification with the incremental Maximum Entropy model and 14 days of training. The graph shows a decrease of the accuracy after the end of the phase of training.

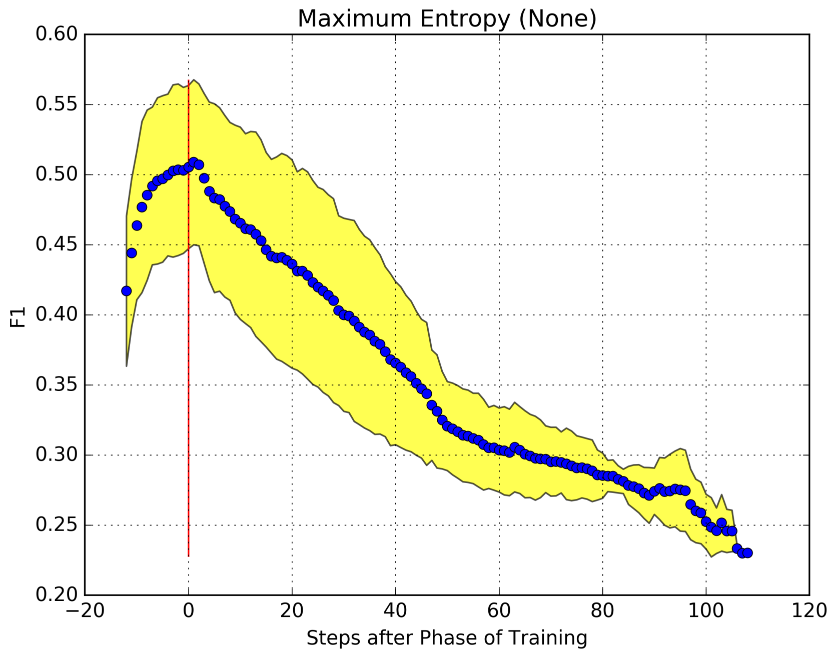

Figure 5.1: Emotion: F1-Maximum Entropy (14 Steps of Training)

The line graph in Figure 5.2 shows the accuracy of the same classifier in the irony and sarcasm corpus. As in the emotion corpus, the graph shows a decrease of the accuracy, but only after 80 steps after the phase of training.

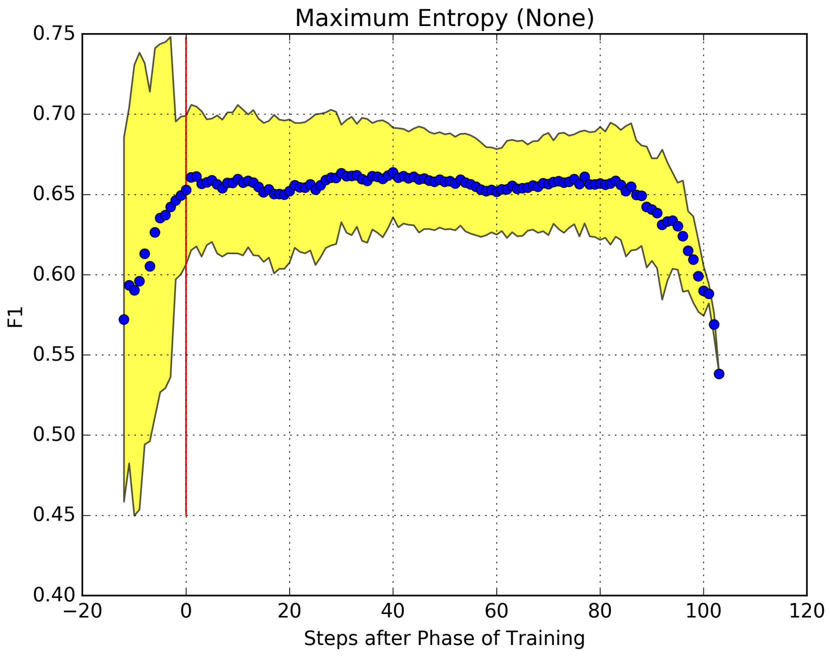

Figure 5.2: Irony and Sarcasm: F1-Maximum Entropy (14 Steps of Training)

The line graph in Figure 5.3 shows the accuracy of classification with the incremental Naive Bayes model and 14 days of training in the emotion corpus. The graph shows a decrease of the accuracy follows by an increase to the starting value.

The line graph in Figure 5.4 shows the accuracy of classification with the incremental Naive Bayes model and 14 days of training in the irony and sarcasm corpus. Like the Maximum Entropy model in the same corpus, the accuracy starts to decrease after a long period of time.

5.2.1.2 Random Classification

The figures of the accuracy of this classifier are in Appendix A.

The results of the randomized classifier are illustrated in Table 5.1. X means the number of steps after the phase of training (one vertical line in the line graphs).

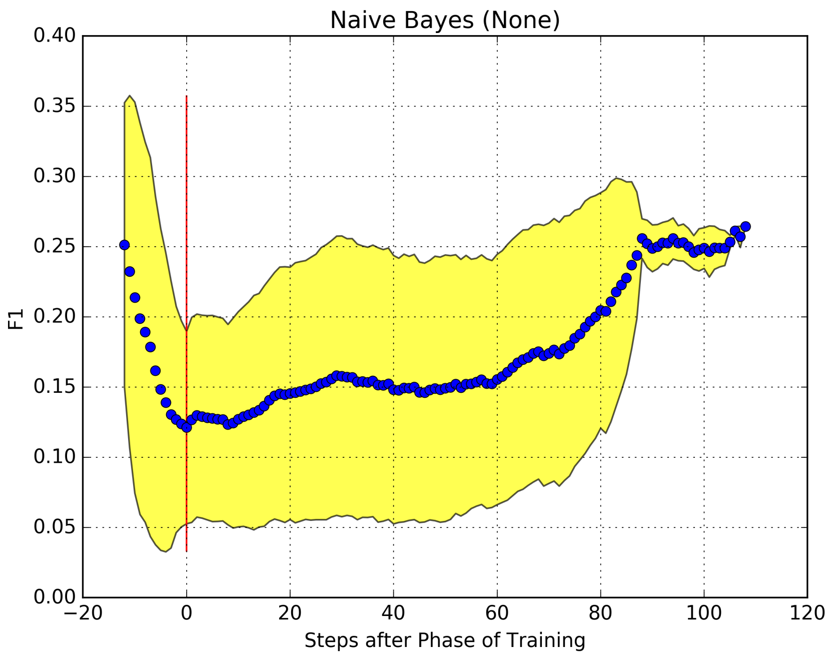

Figure 5.3: Emotion: F1-Naive Bayes (14 Steps of Training)

Accuracy after X steps	1	10	50	100
Emotion, 14 steps of training	0.172	0.154	0.147	0.1442
Emotion, 61 steps of training	0.171	0.154	0.152	-
Irony & Sarcasm, 14 steps of training	0.503	0.500	0.500	0.508
Irony & Sarcasm, 61 steps of training	0.515	0.501	0.503	-

Table 5.1: F1 Score of Randomized Classification

5.2.2 Dynamic Weight Majority

The Dynamic Weight Majority algorithm is evaluated with the same parameters as in [KM07]. If an expert made a mistake, its weight is halved ($\beta = 0.5$). Experts are removed when their weights fall below 0.01 ($\theta = 0.01$).

Figure 5.5 shows the accuracy of the Dynamic Weighted Majority algorithm with the Maximum Entropy model as base classifier for the emotion corpus with 14 steps of training.

Table 5.2 shows their percentage distribution of correct and false decisions in matters of how many classes the algorithm considers during classification. A class is considered when at least one expert votes for the class. For example, 92% of the correct choices are done with concordant voting of all experts. 8% of the correct choices are done with

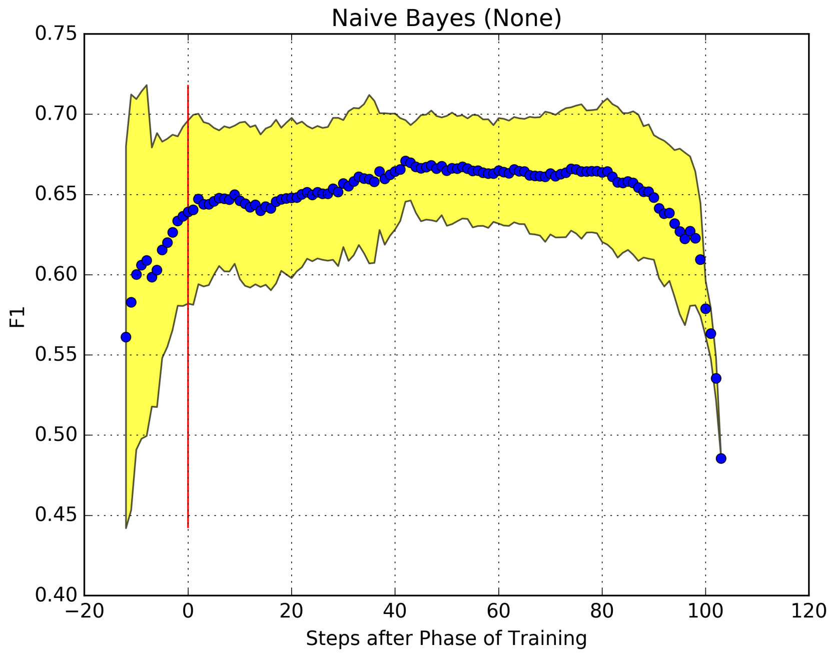

Figure 5.4: Irony and Sarcasm: F1-Naive Bayes (14 Steps of Training)

experts voting on two different classes. Table 5.4 shows the same purpose, but for the irony & sarcasm corpus.

Choice with n classes	Correct	False	Sum
Exactly One	92%	81%	89%
More than One	8%	19%	11%
Exactly Two	8%	19%	11%
Exactly Three	$< 1\%$	$< 1\%$	$< 1\%$

Table 5.2: Results of the DWM Algorithm in the Emotion Corpus

If the experts vote in the emotion corpus for two different classes, different combinations are possible. Table 5.3 shows, how often each combination occurs.

Figure 5.6 shows the accuracy of the Dynamic Weighted Majority algorithm with the Maximum Entropy model as base classifier for the irony & sarcasm corpus with 14 steps of training.

The results of the same algorithm but with the Naive Bayes model as base classifier are illustrated in Figure 5.7 with 14 steps of training in the emotion corpus. The results of the classification of the irony & sarcasm corpus are illustrated in Figure 5.8 with 14 steps of training.

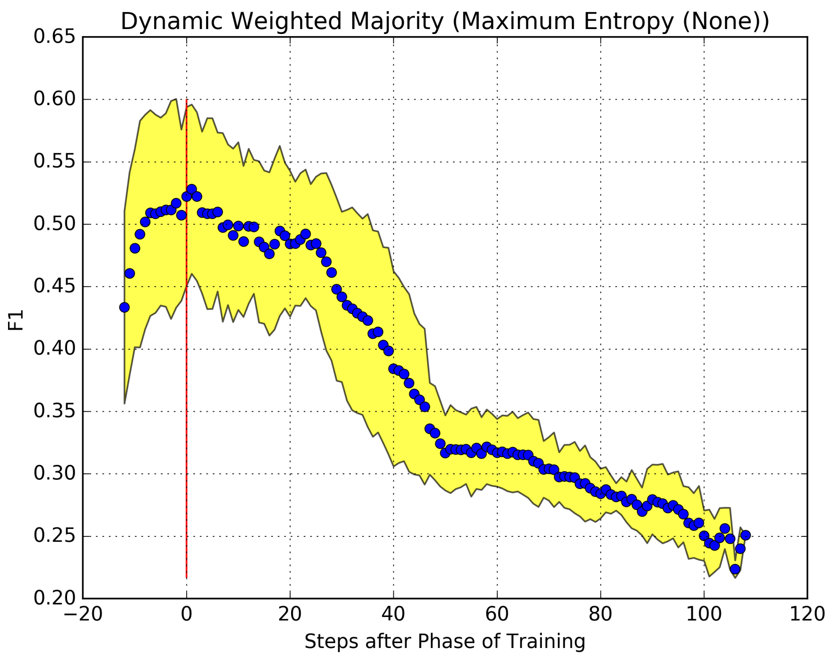

Figure 5.5: Emotion: F1-Dynamic Weighted Majority (MaxEnt) (14 Steps of Training)

5.2.3 Remove Mutual Information Change

Figure 5.9 shows the accuracy of the Remove Mutual Information Change algorithm with the Maximum Entropy model as base classifier for the emotion corpus with 14 steps of training. Figure 5.10 shows the accuracy of the algorithm in the irony & sarcasm corpus with 14 steps of training.

The results of the RMIC algorithm with the Naive Bayes model as base classifier are illustrated in Figure 5.11 for the emotion corpus and in Figure 5.12 for the irony & sarcasm corpus, each with 14 steps of training.

After running each experiment consisting the RMIC algorithm with Maximum Entropy as base classifier, the features of all those classifiers are sorted by how often they were removed. Table 5.5 shows the 25 most removed features for the two corpora. For example, the feature for the word #love is removed the most while classifying the emotion corpus.

In addition, for each removed feature, the change in mutual information of the feature it stored. The greatest value for each feature is saved. Table 5.6 shows the 10 features with the greatest change in mutual information for the two corpora.

Classes	Percentage
joy & sad	72%
joy & anger	7%
joy & fear	6%
anger & sad	5%
fear & sad	4%
joy & surprise	1%
sad & surprise	$< 1\%$
fear & anger	$< 1\%$
anger & surprise	$< 1\%$
joy & disgust	$< 1\%$
joy & fear	$< 1\%$
fear & disgust	$< 1\%$
disgust & surprise	$< 1\%$
other combinations	never

Table 5.3: Percentage of dual Occurrences of Classes in DWM Algorithm in Emotion Corpus

Choice with n classes	Correct	False	Sum
Exactly One	80%	73%	77%
More than One	20%	27%	23%

Table 5.4: Results of the DWM Algorithm in the Irony & Sarcasm Corpus

5.3 Discussion

Figure 5.1 shows decreasing accuracy on the emotion corpus after the phase of training ends. This issue is a result of concept drift. In addition, Figure 5.2 shows decreasing accuracy of the incremental model after about 80 steps after the end of the phase of training on the irony & sarcasm corpus. Concept drift happens in this data set, but more slowly than in the emotion corpus.

This behavior does not only confirm hypothesis 1, namely that the accuracy of the incremental model decreases with increasing distance to the phase of training. It also demonstrate, that algorithms need a domain adaptation strategy in order to yield a higher accuracy after a long period of time.

The remainder of this section is organized as follows. Section 5.3.1 discusses the results of the Dynamic Weight Majority algorithm while Section 5.3.2 discusses the results of

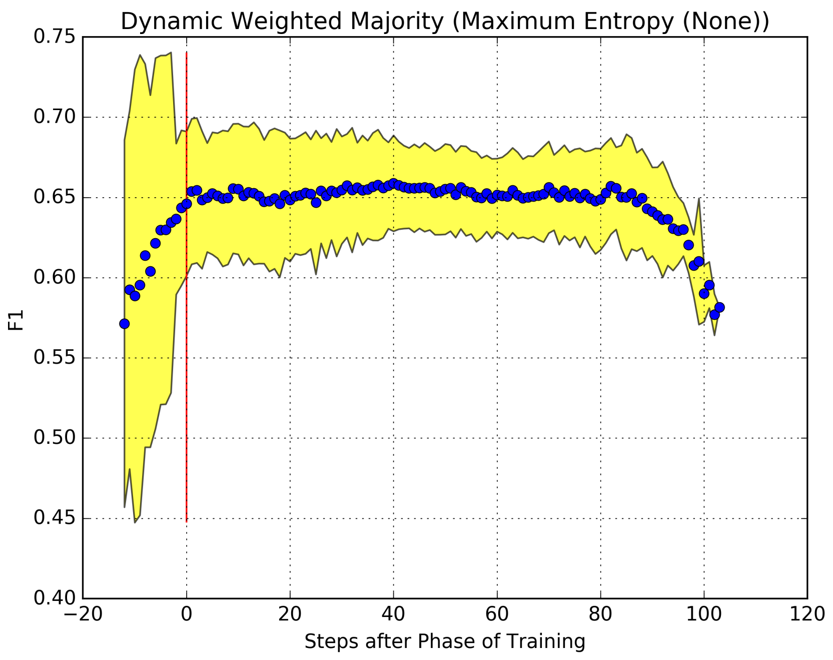

Figure 5.6: Irony and Sarcasm: F1-Dynamic Weighted Majority (MaxEnt) (14 Steps of Training)

the RMIC algorithm. Finally, Section 5.3.3 summarizes the results of the Naive Bayes model.

5.3.1 Dynamic Weight Majority Algorithm

The DWM algorithm achieves in a short period a greater accuracy than the incremental model on the emotion corpus. After the phase of training ends, the accuracy of the incremental model starts to decrease and drops under 0.45 before 20 days have passed (Figure 5.1). The DWM algorithm succeeds to prohibit the decrease of the accuracy for a short period of time. Only after more than 20 days the accuracy starts to decrease in the same way than the accuracy of the incremental model. Until this behavior, the algorithm is able to keep the accuracy around 0.5. In the long term, the accuracies of the two classifiers are nearly at the same value.

On the irony & sarcasm corpus, there is no major difference between the DWM algorithm and the incremental model in matters of the accuracy (Figure 5.5).

Table 5.2 illustrate, that only 11% of all elections made using the DWM algorithm involved experts with different votes. In other words, most of the time, all experts voted for the same class. Considering the distribution of the correct and false classifications,

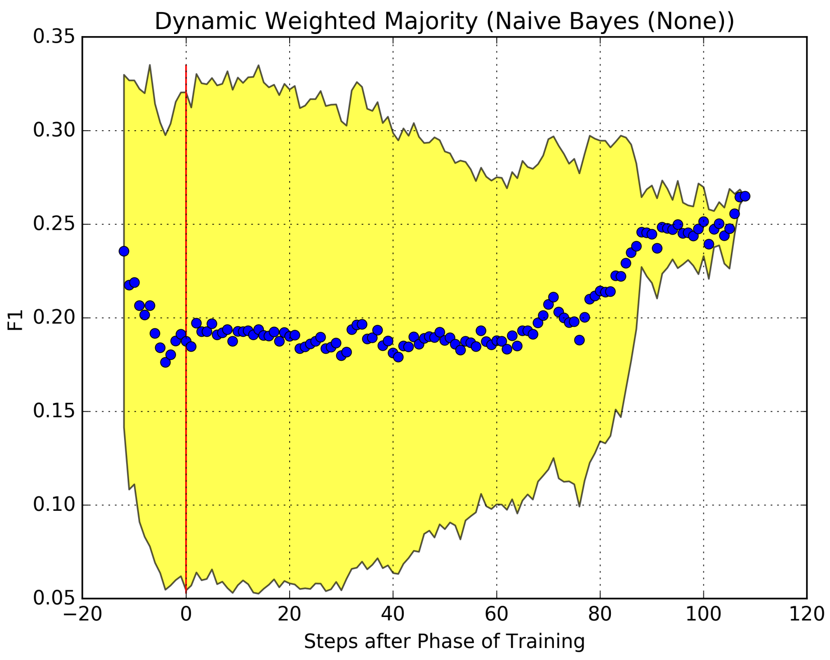

Figure 5.7: Emotion: F1-Dynamic Weighted Majority (NB) (14 Steps of Training)

the elections with votes on more than one class are more defective than the elections where all experts voted equally. Table 5.4 shows the analogously results for the irony & sarcasm corpus. This result correlates with the results presented in Section 2.5.1 of the Majority Vote algorithm. The combined decision of the experts leads to errors, especially it the experts vote for different classes.

Table 5.3 shows the classes which are voted for in an election. Most of the time (in 72% of all elections where the experts do not vote equally), the classes "joy" and "sad" are taken into account. This behavior seems curious, because these two classes are nearly the opposite of each other. But it can be explained by taking the number of Tweets of the classes in the corpus (Table 3.1) into account. The combinations of the classes occurring in Table 5.3 correlate to the combinations of the most frequent classes in the corpus. For example, the most Tweets in the corpus belong to the classes "joy" and "sad", which is the most frequent combination the Dynamic Weight Majority algorithm takes into account while classifying.

Concluding, hypothesis 2 (*The DWM algorithm achieves a greater accuracy than the incremental model for steps with a great distance to the phase of training.*) can not be confirmed. The DWM algorithm only prevented the drop of the accuracy a few steps after the phase of training. In long time, the DWM algorithm was not able to outperform the baseline.

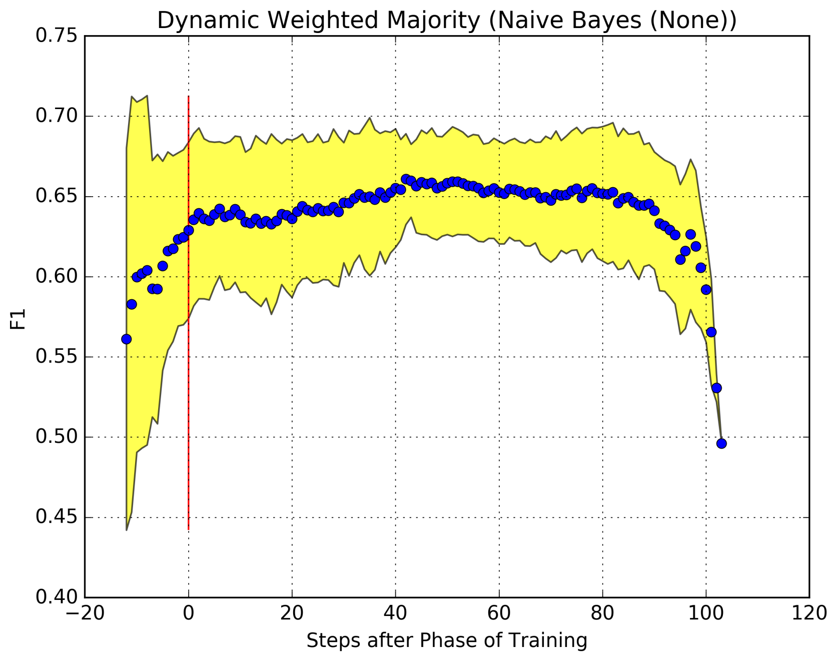

Figure 5.8: Irony and Sarcasm: F1-Dynamic Weighted Majority (NB) (14 Steps of Training)

5.3.2 Remove Mutual Information Change Algorithm

On the emotion corpus, Figure 5.9 shows, that the percentage of features to remove with Maximum Entropy as base classifier has not an impact considering the step 100 steps away from the phase of training. The accuracy after 100 steps is permanent around 0.25. The incremental model (Figure 5.1) achieves an accuracy of 0.25 hundred steps away from the phase of training. Consequently, the RMIC algorithm is not suitable to achieve a higher accuracy than the incremental model over a long time. Furthermore, the percentage of features to remove is not changing the accuracy after 100 steps of training in a major manner.

On the irony & sarcasm corpus, Figure 5.10 shows a decrease of the accuracy with increasing percentage. The incremental model (Figure 5.2) achieves an accuracy of 0.59. This correlate to a percentage of 16% in the RMIC algorithm. The highest accuracy after 100 steps is achieved with 1%.

That implies, that hypothesis 3 (*For some percentage, the RMIC algorithm with Maximum Entropy as base classifier achieves a greater accuracy than the incremental model for steps with a great distance to the phase of training.*) holds on the irony & sarcasm corpus, but not in the emotion corpus.

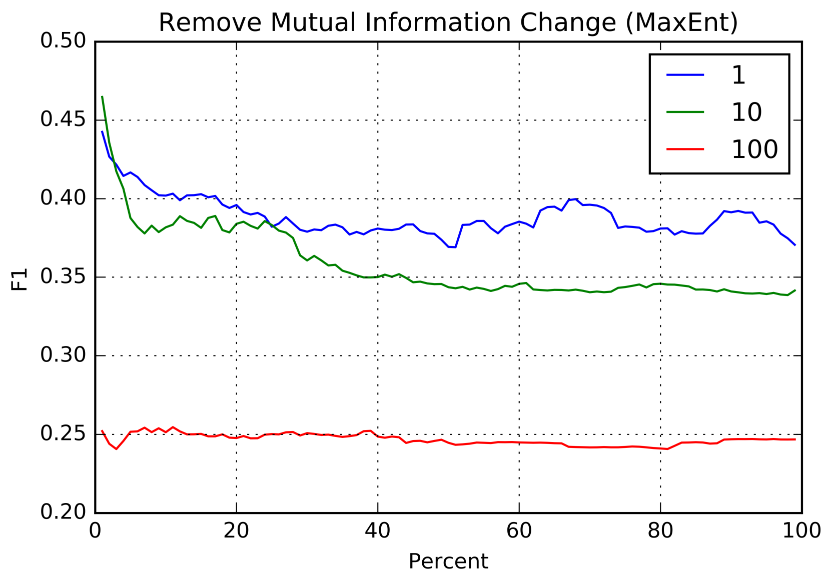

Figure 5.9: Emotion: F1-Remove Mutual Information Change (MaxEnt) (14 Steps of Training)

The fact, that increasing percentage lead to a decrease of the accuracy in the irony & sarcasm corpus suggests, that there are features which have a high impact on classifying. That is because the removal of those features explicit decreases the accuracy of the classifier, which implies, that those features are on high impact. On the other side, classification in the emotion corpus is not affected by the percentage. By that, there are probably no features with a change in mutual information which are on high impact in classification.

Table 5.5 shows which features are removed during the execution of the algorithm. On the irony & sarcasm corpus, common words like "a", "the" and "of" are removed the most. In combination with the assumption, that the removed features are on high impact while classifying this corpus, this behavior results in the assumption, that those common words are on high impact. This assumption is unexpected, because those common words do not carry information or feeling of the texts.

Common words which are removed the most while classifying the emotion corpus are "a", "of", "i". But there are more specific words like "#love", "#life" and "#peace". Those words could be affected by events, but it is not possible to identify, which event is meant.

Table 5.6 shows, that the change in mutual information is greater in the irony & sarcasm corpus than in the emotion corpus. The highest measured change in the emotion corpus

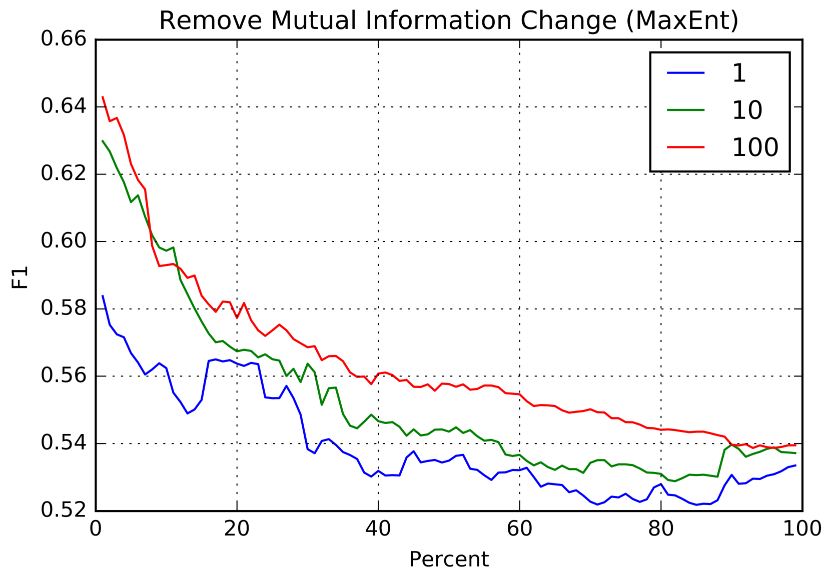

Figure 5.10: Irony and Sarcasm: F1-Remove Mutual Information Change (MaxEnt) (14 Steps of Training)

is 0.234 with the word "#summer", while the highest change is 1.0 in the irony & sarcasm corpus with two words: "life" and "by".

This behavior fortify the assumption, that the removal of features with high change in mutual information increases the accuracy of classification: The removed features in the emotion corpus have a much lower change in mutual information than the removed features in the irony & sarcasm corpus. Only in the irony & sarcasm corpus the accuracy increases. This behavior suggests, that the features in the emotion corpus do not have a sufficient change in mutual information.

Summarized, the experiments can not explicit confirm hypothesis 4 (*The RMIC algorithm removes features which are connected to events happening in the duration of the corpus.*).

5.3.3 Naive Bayes

The Naive Bayes model achieves a low accuracy. On the emotion corpus, its accuracy is about as high as the accuracy of the randomized classification for a long period of time (around 0.15). 80 steps after the phase of training, the accuracy reaches its maximum of 0.25. This value is the lowest achieved with the Maximum Entropy model on the same corpus.

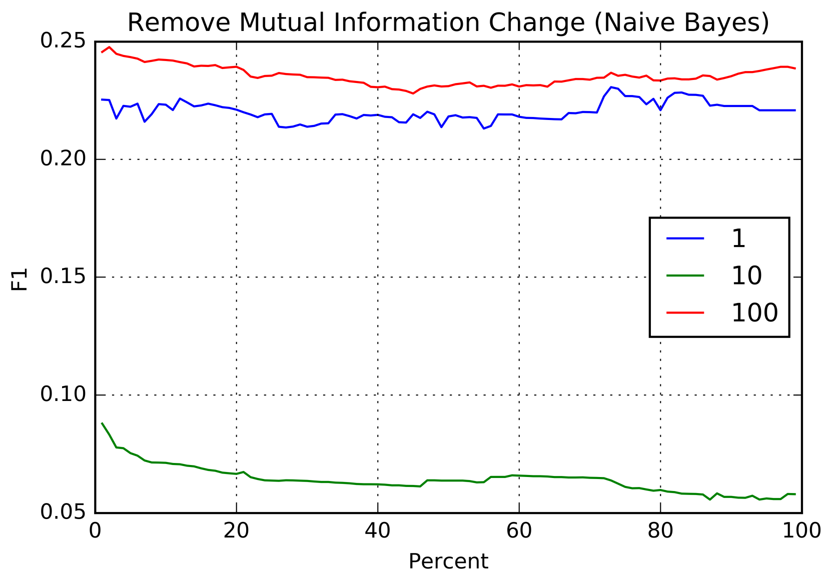

Figure 5.11: Emotion: F1-Remove Mutual Information Change (NB) (14 Steps of Training)

The Dynamic Weight Majority algorithm is able to prevent the drop to an accuracy of the randomized classification. After a long period of time, the accuracy also increases to a maximum of 0.25. Although the model is not able to increase the long time accuracy, it shows the same behavior as for the Maximum Entropy model: it prevents the drop of the accuracy directly after the phase of training ends.

The Remove Mutual Information Change model (Figure 5.11) achieves nearly the same results as the baseline, with an exception of a decrease of the accuracy after 10 steps after the phase of training. This accuracy is lower than the accuracy of the randomized classification.

In addition, the variance of the Naive Bayes model is a twice as big than the variance of the Maximum Entropy model on the emotion corpus.

On the irony & sarcasm corpus, the Naive Bayes model achieves accuracies comparable to the Maximum Entropy model. It achieves a slightly higher accuracy on the second halve of the steps, but with a greater drop at the last steps. The Dynamic Weight Majority model was not able to improve the baseline in a major manner. But the Remove Mutual Information Change model, like for the Maximum Entropy model, is able to achieve a higher accuracy 100 steps after the phase of training than the baseline. The Naive Bayes model achieves an accuracy of 0.577 100 steps after the phase of training. The RMIC model achieves an accuracy of over 0.6 100 steps after the phase of training for a percentage of under 10%.

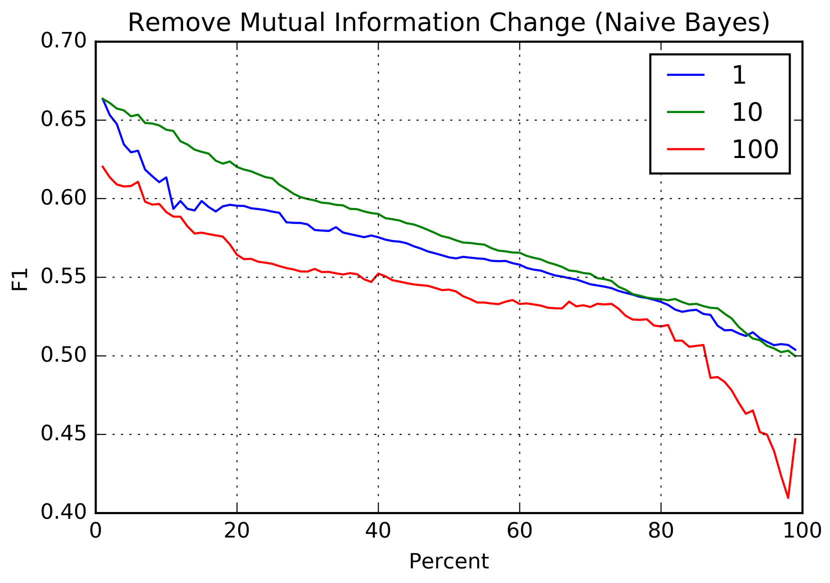

Figure 5.12: Irony and Sarcasm: F1-Remove Mutual Information Change (NB) (14 Steps of Training)

In summary, the results of the Naive Bayes are nearly random on the emotion corpus, but they can be improved by the Dynamic Weight Majority model. On the irony & sarcasm corpus, the Naive Bayes model achieves the same accuracy as the Maximum Entropy model. In addition, the RMIC model achieves the same results as with the Maximum Entropy model. Consequently, hypothesis 5 can not be proved (*The accuracy of the Naive Bayes model decreases when using the mutual information change remove method for domain adaptation*).

Position	Emotion Corpus	Irony & Sarcasm Corpus
1	#love	a
2	#life	the
3	a	of
4	life	#funny
5	*[URL]*	about
6	#peace	by
7	#loss	great
8	#success	it
9	day	you
10	love	and
11	#friends	he
12	#death	*[URL]*
13	of	we
14	*[USER]*	#lol
15	i	out
16	no	who
17	#me	on
18	#summer	be
19	one	in
20	you	but
21	how	so
22	#family	to
23	loss	that
24	peace	well
25	beautiful	thanks

Table 5.5: Removed Features, sorted by frequency of remove

Position	Emotion Corpus	MI Change	Irony & Sarcasm Corpus	MI Change
1	#summer	0.234	life	1.0
2	#loss	0.203	by	1.0
3	loss	0.124	apparently	0.999
4	#peace	0.110	doing	0.999
5	#life	0.097	are	0.999
6	#success	0.091	#life	0.998
7	peace	0.090	so	0.998
8	#fun	0.089	while	0.998
9	friends	0.087	go	0.995
10	#sun	0.084	other	0.993

Table 5.6: Removed Features, sorted by Change in Mutual Information

6 Summary and Outlook

This thesis examines two different models to handle concept drift on a corpus of crawled Tweets. Concept Drift occurs in dynamical environments like social media. It affects texts in a way that classifiers for stationary environments are not able to achieve an acceptable accuracy. Domain adaptation strategies are of important need to achieve a higher accuracy while classifying in a dynamical environment.

For this purpose, two different models are tested, if they are able to solve the problem of classifying in such environments.

The **Remove Mutual Information Change Model** tries to remove features with changing information value relating to the classes. The assumption for this model is, that the removal of the features which are affected by concept drift, creates a set of "base" features. This "base" features should lead, without the removed features, to a higher accuracy after a long period of time.

The idea of the **Dynamic Weight Majority Model** is to create experts which vote for the outcome of the classification task. The hope is, that the created set of voters is more accurate than the separate experts.

The models are evaluated in a crawled corpus of real Tweets.

The results show, that both of the models are not able to outperform the baseline over a long period of time. The Remove Mutual Information Change Model was able to achieve a higher accuracy than the baseline in the corpus where the mutual information of the features changes rapidly. In the emotion corpus, where the mutual information values are not changing in a high extend, the model achieves the same accuracy than the baseline.

The Dynamic Weight Majority Model was able to prevent the decrease of the accuracy after the phase of training for a short period of time. But afterward, it does not achieve a higher accuracy.

Future work may consider improvements of the models.

Perhaps one possibility to improve the Dynamic Weight Majority Model is to use experts with domain adaptation strategies. In this thesis, the experts are the incremental model.

Maybe a use of the Remove Mutual Information Change Model as experts creates a higher accuracy after a long period of time.

This thesis shows, that the Remove Mutual Information Change Model achieves a higher accuracy if the features have an high change in mutual information. Maybe there exists other measures, which can replace the mutual information, in order to create a new model, which ensures a high accuracy on every corpus.

A Appendix

This chapter comprises the results of the classification task with 61 steps of training and the figures of the randomized classification.

The results of the classification task with 61 steps of training correlate with the results of the classification task with 14 steps of training. The additional steps of training enable a higher accuracy, because the classifier gets more examples to learn from. The accuracy increases by approximately 0.05. The decrease of the accuracy with increasing distance to the phase of training is noticeable.

The Maximum Entropy model (the baseline) is shown in Figure A.1 (emotion corpus) and in Figure A.2 (irony and sarcasm corpus). The Naive Bayes model as baseline is shown in Figure A.3 (emotion corpus) and in A.4 (irony and sarcasm corpus). The Dynamic Weight Majority model is shown in Figure A.5 (emotion corpus) and in Figure A.6 (irony and sarcasm corpus) for the Maximum Entropy model as base classifier. The Naive Bayes model as base classifier for the Dynamic Weight Majority model is shown in Figure A.7 (emotion corpus) and in Figure A.8 (irony and sarcasm corpus).

Lastly, the figures of the randomized classification are presented. Figure A.9 and Figure A.10 show the classification in the emotion corpus. Figure A.11 and Figure A.12 in the irony and sarcasm corpus.

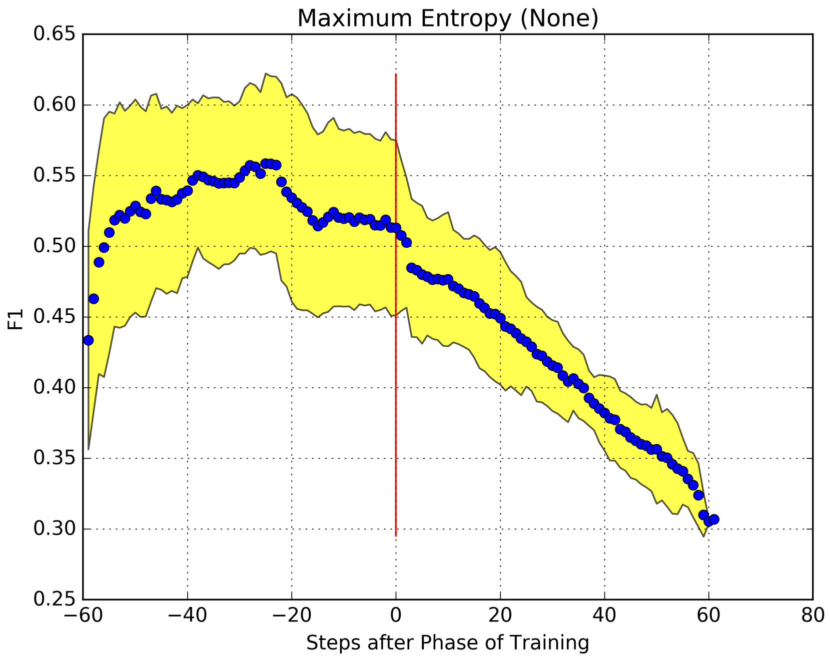

Figure A.1: Emotion: F1-Maximum Entropy (61 Steps of Training)

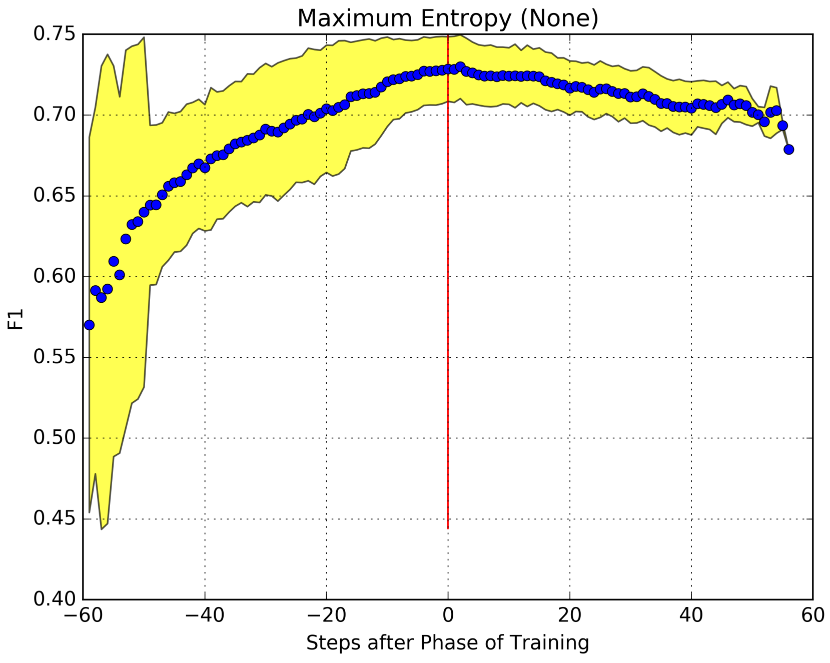

Figure A.2: Irony and Sarcasm: F1-Maximum Entropy (61 Steps of Training)

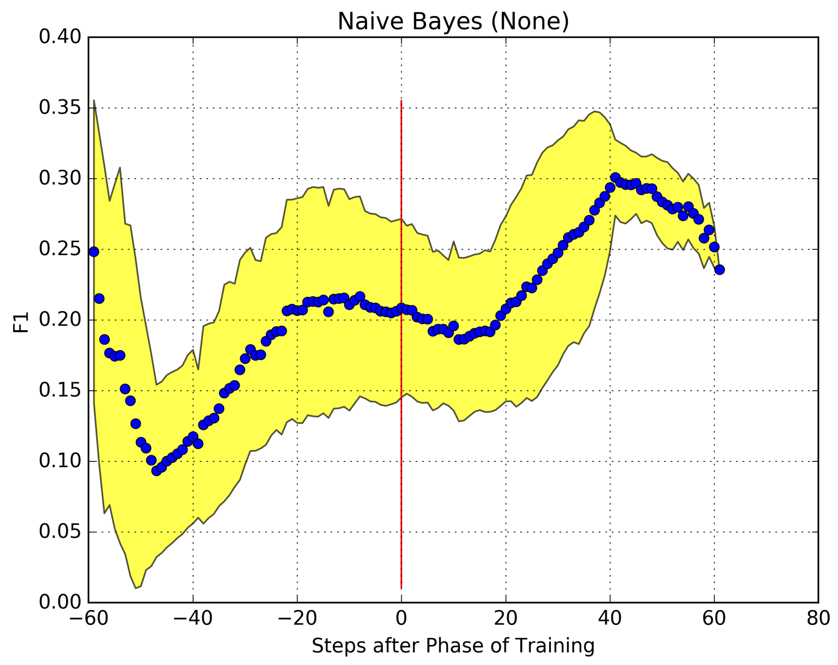

Figure A.3: Emotion: F1-Naive Bayes (61 Steps of Training)

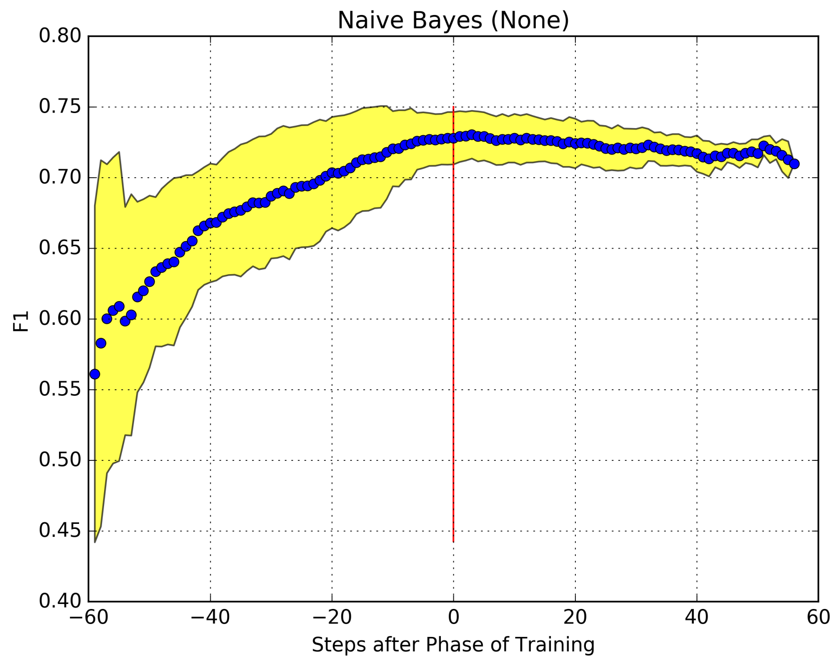

Figure A.4: Irony and Sarcasm: F1-Naive Bayes (61 Steps of Training)

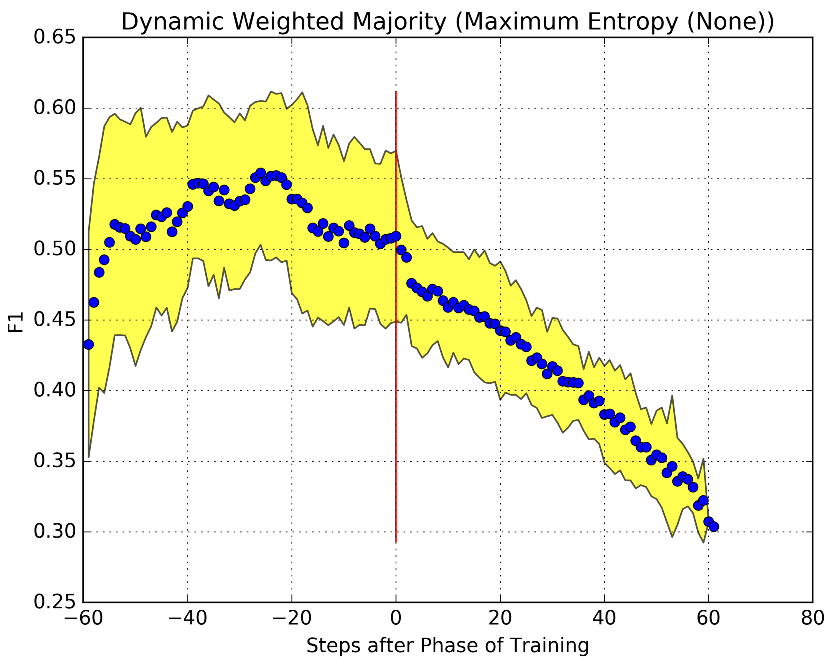

Figure A.5: Emotion: F1-Dynamic Weighted Majority (MaxEnt) (61 Steps of Training)

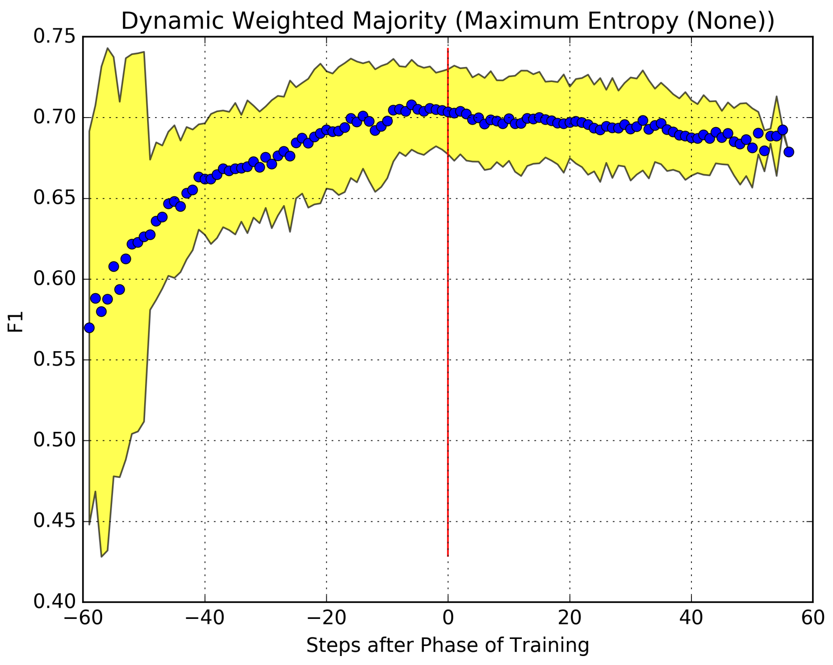

Figure A.6: Irony and Sarcasm: F1-Dynamic Weighted Majority (MaxEnt) (61 Steps of Training)

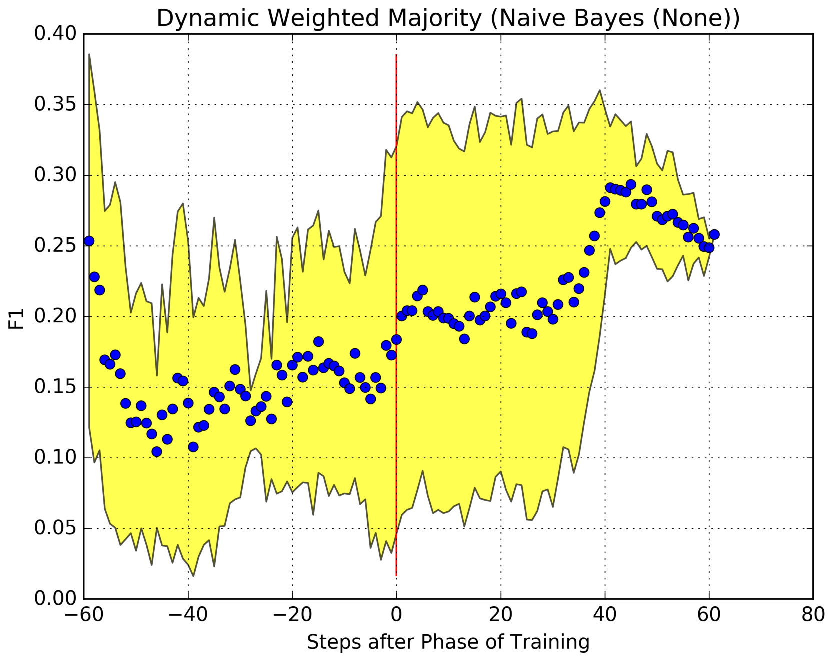

Figure A.7: Emotion: F1-Dynamic Weighted Majority (NB) (61 Steps of Training)

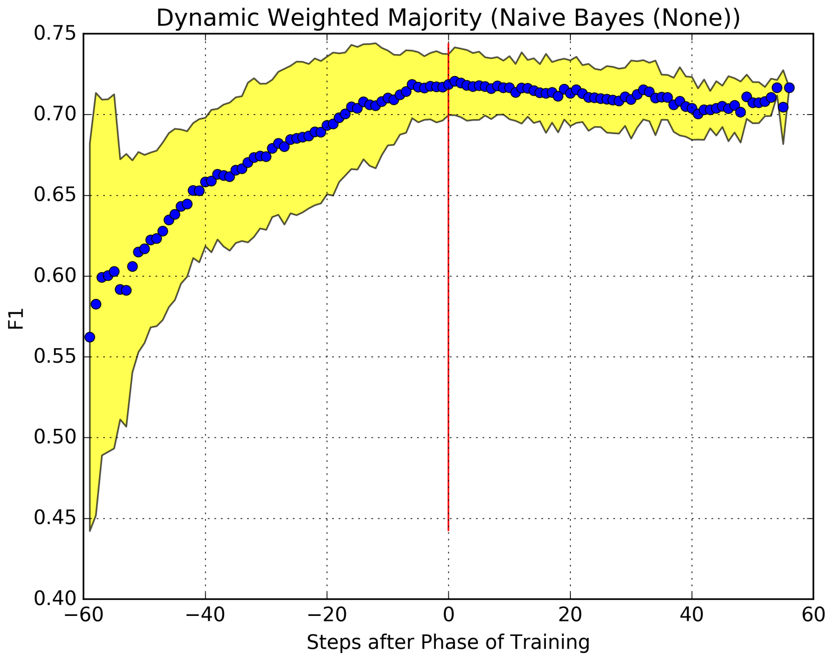

Figure A.8: Irony and Sarcasm: F1-Dynamic Weighted Majority (NB) (61 Steps of Training)

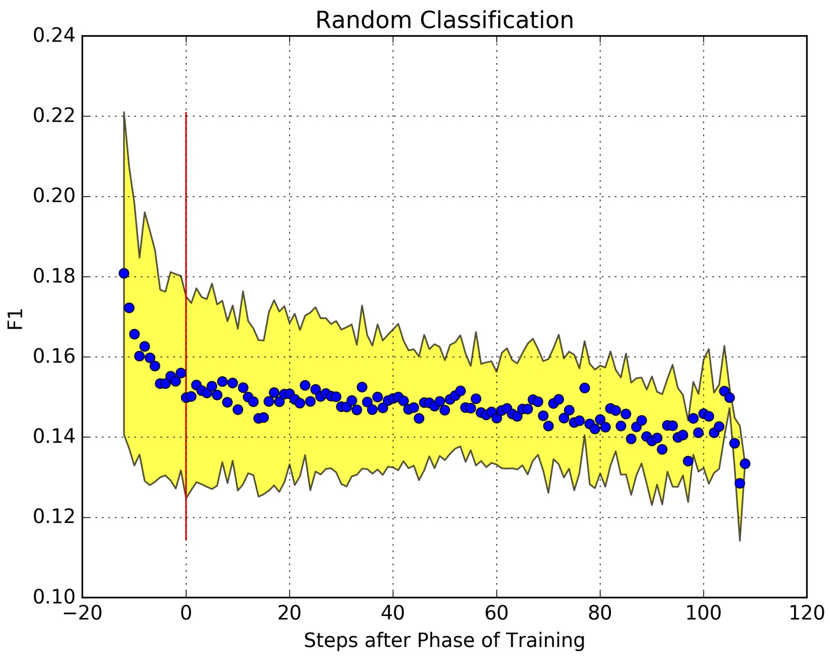

Figure A.9: Emotion: F1-Random Classification

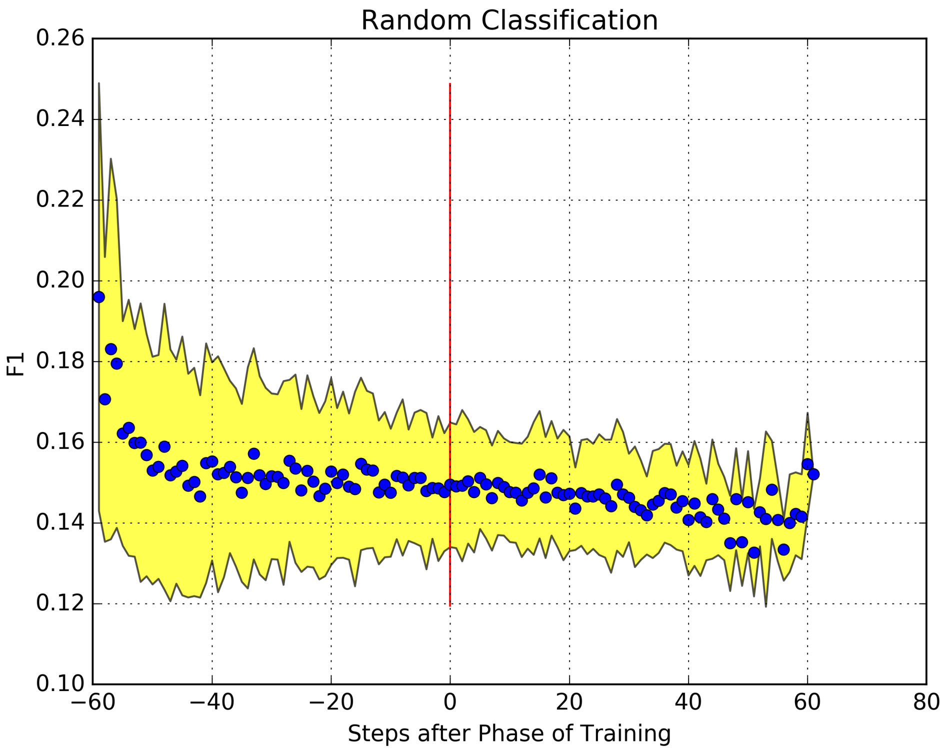

Figure A.10: Emotion: F1-Random Classification

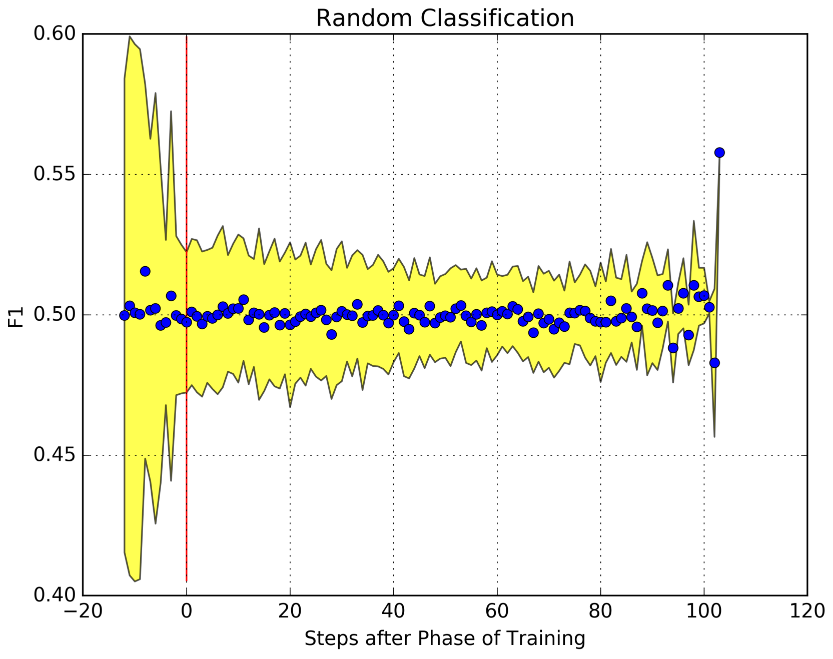

Figure A.11: Irony and Sarcasm: F1-Random Classification

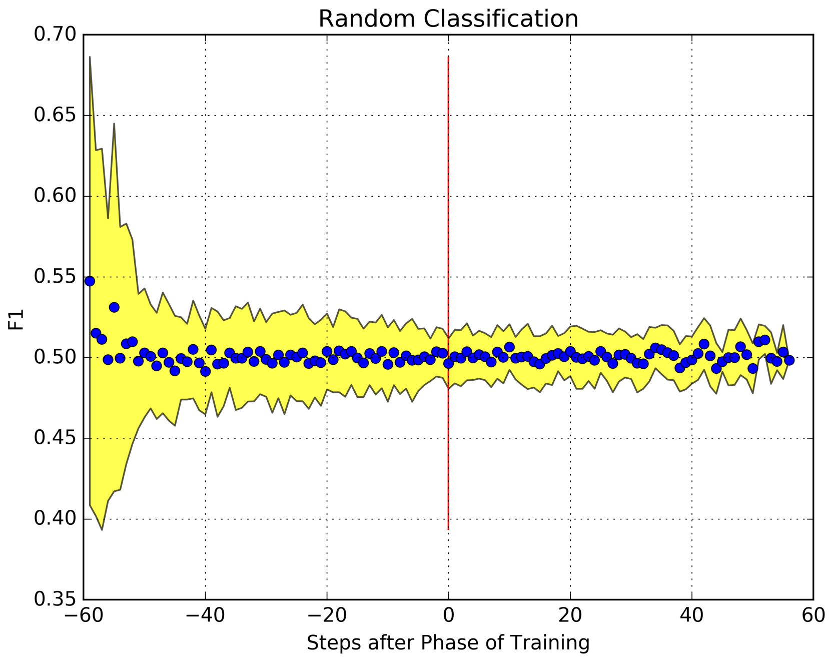

Figure A.12: Irony and Sarcasm: F1-Random Classification

Bibliography

[16] *Internet Live Statistics*. 2016. URL: http://www.internetlivestats.com/ (visited on 09/21/2016) (cit. on pp. 13, 14).

[Bat94] R. Battiti. “Using mutual information for selecting features in supervised neural net learning.” In: *IEEE Transactions on Neural Networks* 5.4 (July 1994), pp. 537–550. ISSN: 1045-9227. DOI: 10.1109/72.298224 (cit. on p. 33).

[Blu97] A. Blum. “Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain.” In: *Machine Learning* 26.1 (1997), pp. 5–23 (cit. on p. 34).

[BPP96] A. L. Berger, V. J. D. Pietra, S. A. D. Pietra. “A maximum entropy approach to natural language processing.” In: *Computational linguistics* 22.1 (1996), pp. 39–71 (cit. on pp. 20, 21).

[BR+99] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*. Vol. 463. ACM press New York, 1999 (cit. on pp. 18, 23).

[BS14] F. Barbieri, H. Saggion. “Modelling Irony in Twitter: Feature Analysis and Evaluation.” In: *LREC*. 2014, pp. 4258–4264 (cit. on pp. 15, 27).

[CSAR14] J. Costa, C. Silva, M. Antunes, B. Ribeiro. “Concept drift awareness in Twitter streams.” In: *Machine Learning and Applications (ICMLA), 2014 13th International Conference on*. IEEE. 2014, pp. 294–299 (cit. on p. 37).

[DAK07] R. Duwairi, M. Al-Refai, N. Khasawneh. “Stemming versus light stemming as feature selection techniques for Arabic text categorization.” In: *Innovations in Information Technology, 2007. IIT’07. 4th International Conference on*. IEEE. 2007, pp. 446–450 (cit. on pp. 22, 23).

[DS16] M. Dagnis Jensen, H. Snaith. “When politics prevails: the political economy of a Brexit.” In: *Journal of European Public Policy* (2016), pp. 1–9 (cit. on p. 56).

[GBB11] X. Glorot, A. Bordes, Y. Bengio. “Domain adaptation for large-scale sentiment classification: A deep learning approach.” In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 513–520 (cit. on p. 31).

[GG05] C. Goutte, E. Gaussier. “A probabilistic interpretation of precision, recall and F-score, with implication for evaluation.” In: *European Conference on Information Retrieval*. Springer. 2005, pp. 345–359 (cit. on p. 23).

[GŽB+14] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia. “A Survey on Concept Drift Adaptation.” In: *ACM Comput. Surv.* 46.4 (Mar. 2014), 44:1–44:37. ISSN: 0360-0300. DOI: 10.1145/2523813. URL: http://doi.acm.org/10.1145/2523813 (cit. on pp. 29, 30).

[Hen14] J. Henrique. *Twitter4J*. 2014. URL: https://github.com/Jefferson-Henrique/GetOldTweets-java (visited on 08/11/2016) (cit. on p. 43).

[Inc16] T. Inc. *Twitter*. 2016. URL: https://support.twitter.com/articles/77606# (visited on 09/12/2016) (cit. on p. 45).

[KEBC86] M. D. Klinnert, R. N. Emde, P. Butterfield, J. J. Campos. “Social referencing: The infant’s use of emotional signals from a friendly adult with mother present.” In: *Developmental Psychology* 22.4 (1986), p. 427 (cit. on p. 26).

[KK98] D. Keltner, A. M. Kring. “Emotion, social function, and psychopathology.” In: *Review of General Psychology* 2.3 (1998), p. 320 (cit. on p. 26).

[KLPM10] H. Kwak, C. Lee, H. Park, S. Moon. “What is Twitter, a Social Network or a News Media?” In: *Proceedings of the 19th International Conference on World Wide Web*. WWW ’10. Raleigh, North Carolina, USA: ACM, 2010, pp. 591–600. ISBN: 978-1-60558-799-8. DOI: 10.1145/1772690.1772751. URL: http://doi.acm.org/10.1145/1772690.1772751 (cit. on p. 13).

[KM07] J. Z. Kolter, M. A. Maloof. “Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts.” In: *J. Mach. Learn. Res.* 8 (Dec. 2007), pp. 2755–2790. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1314498.1390333 (cit. on pp. 34, 36, 37, 59).

[Lew98] D. D. Lewis. “Naive (Bayes) at forty: The independence assumption in information retrieval.” In: *European conference on machine learning*. Springer. 1998, pp. 4–15 (cit. on pp. 18, 19).

[LK16] J. Ling, R. Klinger. “An Empirical, Quantitative Analysis of the Differences between Sarcasm and Irony.” In: *Semantic Sentiment Analysis Workshop*. European Semantic Web Conference. Crete, Greece, May 2016 (cit. on p. 37).

[LW89] N. Littlestone, M. K. Warmuth. “The weighted majority algorithm.” In: *Foundations of Computer Science, 1989., 30th Annual Symposium on*. IEEE. 1989, pp. 256–261 (cit. on p. 34).

[MCK+10] M. M. Masud, Q. Chen, L. Khan, C. Aggarwal, J. Gao, J. Han, B. Thuraisingham. “Addressing concept-evolution in concept-drifting data streams.” In: *2010 IEEE International Conference on Data Mining*. IEEE. 2010, pp. 929–934 (cit. on p. 22).

[Rea99] R. Real. “Tables of significant values of Jaccard’s index of similarity.” In: *Miscel· lania Zoologica* 22.1 (1999), pp. 29–40 (cit. on p. 22).

[Ris01] I. Rish. “An empirical study of the naive Bayes classifier.” In: *IJCAI 2001 workshop on empirical methods in artificial intelligence*. Vol. 3. 22. IBM New York. 2001, pp. 41–46 (cit. on p. 18).

[RRJ+12] K. Roberts, M. A. Roach, J. Johnson, J. Guthrie, S. M. Harabagiu. “EmpaTweet: Annotating and Detecting Emotions on Twitter.” In: *LREC*. Citeseer. 2012, pp. 3806–3813 (cit. on pp. 15, 37).

[San88] R. E. Sanders. “Dan Sperber and Deirdre Wilson, Relevance: Communication and cognition, Oxford: Basil Blackwell, 1986. Pp. 265.” In: *Language in Society* 17.04 (1988), pp. 604–609 (cit. on p. 27).

[TÇG+11] D. Torunoğlu, E. Çakirman, M. C. Ganiz, S. Akyokuş, M. Z. Gürbüz. “Analysis of preprocessing methods on classification of Turkish texts.” In: *Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on*. IEEE. 2011, pp. 112–117 (cit. on p. 23).

[VV03] S. Verbaeten, A. Van Assche. “Ensemble methods for noise elimination in classification problems.” In: *International Workshop on Multiple Classifier Systems*. Springer. 2003, pp. 317–325 (cit. on p. 34).

[Yam07] Y. Yamamoto. *Twitter4J*. 2007. URL: http://twitter4j.org/en/index.html (visited on 08/11/2016) (cit. on p. 43).

[Yan99] Y. Yang. “An evaluation of statistical approaches to text categorization.” In: *Information retrieval* 1.1-2 (1999), pp. 69–90 (cit. on p. 25).

[YL99] Y. Yang, X. Liu. “A re-examination of text categorization methods.” In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1999, pp. 42–49 (cit. on p. 24).

[YLZ+05] J. Yan, N. Liu, B. Zhang, S. Yan, Z. Chen, Q. Cheng, W. Fan, W.-Y. Ma. “OCFS: optimal orthogonal centroid feature selection for text categorization.” In: *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2005, pp. 122–129 (cit. on p. 22).

[Žli10] I. Žliobaitė. “Learning under concept drift: an overview.” In: *arXiv preprint arXiv:1010.4784* (2010) (cit. on pp. 30, 31).

Declaration

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

place, date, signature