Using transfer learning to increase performance of multinomial text classification on documents of different municipalities.

submitted in partial fulfillment for the degree of master of science

Tobias Beers
10556249

master information studies
data science
faculty of science
university of amsterdam

2018-06-29

	Internal Supervisor	External Supervisor
Title, Name	Dr Maarten Marx	
Affiliation	UvA, FNWI, IvI	
Email	maartenmarx@uva.nl .	

Universiteit van Amsterdam

Amsterdam
Data Science

Amsterdam
Data Science

Using transfer learning to increase performance of multinomial text classification on documents of different municipalities.

UNIVERSITY OF AMSTERDAM, Amsterdam, Netherlands,

Contents

Contents 1
1 Introduction 2
2 Related Work 2
2.1 RQ1: On Naive Bayes 3
2.2 RQ2: On Transfer-learning and Naive Bayes 3
2.3 RQ3: On k-Means Clustering 4
3 Methodology 4
3.1 Description of the data 4
3.2 Methods 4
3.2.1 RQ1: Multinomial Naive Bayes 4
3.2.2 RQ2: Naive Bayes and TrAdaBoost 4
3.2.3 RQ3: k-Means Clustering 5
4 Evaluation 5
4.1 RQ1: Multinomial Naive Bayes 5
4.2 RQ2: Naive Bayes and TrAdaBoost 5
4.3 RQ3: k-Means Clustering 6
5 Conclusions 7
References 7
A Preprocessing 8
2 Elaboration on the evaluation scores 9
2.1 When not all documents are assigned to a class 9
3 Elaboration on results 10
3.1 Multinomial Naive Bayes 10
3.2 TrAdaBoost 10
3.3 k-Means Clustering 13
4 Slides 14

ABSTRACT

When machine learning models are built, they can be trained and tested on datasets from different sources. With this practice comes the implicit assumption that datasets from different sources follow the same structure. When this assumption is violated, the results may be negatively affected. One way to cope with different datasets is the use of transfer-learning. One of such transfer-learning algorithms, TrAdaBoost, is evaluated in this experiment. To assess the algorithms performance, one trainset and two testsets of documents from different municipalities were gathered from the Open State Foundation. The algorithm was first generalized to work with multi-class problems. As a baseline, a Naive Bayes model that categorized documents from different municipalities was built. TrAdaBoost was then applied to this model to compare the performance. Both TrAdaBoost and the baseline Naive Bayes model were built 50 times with paired samples of the trainset. Each instance of the models was tested on the testset to obtain precision, recall and F1-scores. The results were analysed with a two-tailed Wilcoxon signed-rank test. When the TrAdaBoost algorithm was applied to the Naive Bayes model, the macro-averaged F1-score was significantly higher than the baseline for both testsets ($P<0.0001$ and $P<0.0005$). The micro-averaged F1-score increased significantly as well ($P<0.0001$ and $P<0.001$). These results suggest that TrAdaBoost improves the performance of a traditional Naive Bayes model when using datasets from different sources. This also means that a possible multi-class implementation of TrAdaBoost was found. Future machine learning may take differences between datasets into account, to increase performance.

1 INTRODUCTION

The Open State Foundation has gathered documents from all municipalities in the Netherlands. Motions, agendapoints and comissionletters are all examples of categories of these documents. Some municipalities, Utrecht for instance, have always labeled their documents by category, however, many have never done so. To sort these documents by category, a machine learning algorithm for text classification is in order.

A traditional classification algorithm would be trained on a labeled subset and then be used on another subset from the same dataset. In this case however, some municipalities have no labeled training examples at all. Therefore, an algorithm might be trained on the documents of Utrecht, which are all labeled by category, and then be used on any other municipality. Question remains whether categories of documents follow the same text-structure among municipalities. For example, it may be possible that Utrecht writes its 'motions' in a different way from Amsterdam. There could be subtle dissimilarities in their writing style, but their could also be strong differences in the structure of each document. Therefore, it is not certain if a traditional classification algorithm should generalize a model learned on documents of one municipality to documents of another municipality.

One way to cope with structural dissimilarities between datasets is to use transfer-learning. Transfer-learning includes machine learning algorithms that train a model on 'similar but different' data, to classify the data of interest. Before it applies a learned model on the test-set, it requests a small, class-labeled subset of the data, so that it may observe if the classification pattern differs from the newly presented dataset and adjust its learned model to the class-structures of the new dataset.

This brings us to the research question of this thesis: **"Can transfer learning be used to improve performance of classification models over documents of various municipalities?"** To answer this question, the following subquestions have to be asked:

- **RQ1 "What is the performance of a traditional Naive Bayes model on this data?"** A Multinomial Naive Bayes algorithm will be used to evaluate this technique. This will be the baseline for traditional machine learning without transfer-learning.
- **RQ2 "When transfer-learning is applied to a Naive Bayes model, how does its performance then relate to that of a traditional Naive Bayes model?"** By using a Naive Bayes algorithm on which transfer-learning has been applied, another evaluation can be performed. This new evaluationscore can then be compared with the performance of the traditional Naive Bayes model. It is possible to compare the results over all documents or per category. This should provide an idea of the differences and similarities in performance of the two algorithms.
- **RQ3 "Are the classes in the train-dataset well enough defined?"** This last question provides insight into the classification. Suppose that the trainset just happens to exhibit less definition or variations over its classes, but the testset does have non-overlapping and well-defined classes. In this hypothetical case, an adjustment of parameters would improve the final score on the testset, however, this would not be due to being a better algorithm. This improvement would be because the trainset was not suitable to learn on, which was the only source of information for the traditional Naive Bayes. For this reason, an unsupervised k-means clustering algorithm will be performed on every dataset. An unsupervised algorithm like this indicates whether the original classes could be found by another algorithm, and if so, how many documents are correctly classified in their cluster. If an unsupervised algorithm can indicate the difference between classes, then a supervised algorithm that uses the same features and is provided with class-examples should be able to find the classes too.

2 RELATED WORK

Since the first applications of transfer-learning on artificial neural networks in the early 1990's [10], transfer-learning has evolved in its abilities. Its applicability has come to include more algorithms[9][14]. For instance, algorithms that allow for the transfer of knowledge between Naive Bayes classifiers have been set up and proven to increase performance in comparison with a traditional Naive Bayes classifier[3]. A prime example of transfer-learning to improve performance, is that transfer-learning has proven itself efficient in Bayesian networks used for spam-detection in emailboxes[12]. Making use of transfer-learning can allow one to treat each email-user with their individual preferences towards spam as different datasets, thereby personalizing their spam-filter.

Naive Bayes is sometimes underrated in its performance of text classification. However, the performance of Naive Bayes has been positively evaluated, especially when using smaller datasets[5], as is the case with this research. This, as well as its simplicity and computational efficiency, is the main reason to test the addition of transfer-learning on machine learning algorithms to a Bayesian network, as opposed to other methods (artificial neural networks, support vector machines, etc.).

2.1 RQ1: On Naive Bayes

The two most common variations of Bayesian networks include the Bernoulli and the Multinomial Naive Bayes. This thesis will make use of the multinomial Naive Bayes network, as it has proven to be more effective when encountering large vocabularies[8], as may be the case in the to-be-used database of sociopolitical topic-specific documents. The multinomial Naive Bayes algorithm, looks at each document and assumes that all different categories of documents are generated from a unique 'model'. This way, every category has an overall statistical probability of producing certain words, e.g. a medical document is far more likely to contain the words 'blood' and 'tissue' than a political document. To begin with, all words are assigned a probability to be produced per class. Then, for any given document, the statistical probability to be produced by any of the classes is calculated and after that, the document can simply be classified as the most probable class[6]. The formula for this is given below(**1**):[11]

$$\theta_{ci} = \frac{N_{ci} + \alpha_i}{N_c + \alpha} \tag{1}$$

In this formula, θ_{ci} is a parametervector describing the probability of term i occurring in a document of class c, N_{ci} is the number of occurrences of term i in class c and N_c is the total count of all terms in class c. α_i and α are smoothing factors. α_i is usually set to 1.0 for every word and α is the sum of all α for every word i ($\sum_{i=0}^{N} \alpha_i$, where N denotes the total number of words). The predicted class $h(d)$ of document d can then be chosen by finding the class c with the highest probability of producing document d, as described by **formula 2**, where $P(x_i|c)$ can be substituted with θ_{ci}.

$$h(d) = \arg\max_c P(c) \prod_{i=0}^{n} P(x_i|c) \tag{2}$$

2.2 RQ2: On Transfer-learning and Naive Bayes

When a dataset with few labeled training examples has to be classified, transfer-learning comes in. Transfer-learning allows to use knowledge gained from other datasets. Transfer-learning has increased performance of traditional classification algorithms, especially when applied to Bayesian networks and when datasets of less similar distributions are used[4]. Different variations of transfer-learning have come up in literature: applying different classes on a similar dataset, using datasets of different domains, etc.[9]. In this case, we want to apply the same task and classify according to the same classes, however, the probability distribution (the parameter-vector θ_c) is different. One suggested approach, is *TrAdaBoost*[4]. TrAdaBoost is an algorithm that assumes equal features and classes for two sets, but a different distribution. For the TrAdaBoost algorithm, two datasets must be given: one dataset of few labeled instances of the testset (T_s of size m) (e.g. in practice, one could label a small subset of the dataset of interest) and a bigger dataset (T_d of size n) of a different distribution. This smaller dataset would follow the same distribution of words, but would be too small to make accurate predictions. The bigger dataset contains enough training data, but may follow a different distribution. Although the author does not mention it when introducing TrAdaBoost, his version of TrAdaBoost is written for binomial classification only[4], therefore, these formulas are slightly adjusted before application, as described in **section 3.2.2**.

As the algorithm initializes, a Naive Bayes model is generated using every document in T (the combined set of T_d and T_s of size $n + m$) as trainset and using T_s as testset. Then, every document $x_i \in T$ is given a weight w_i of 1. Now, for N iterations, the whole testset T_s is labeled, and an error ϵ_t, for iteration t, is predicted through **formula 3**.

$$\epsilon_t = \sum_{i=n+1}^{n+m} \frac{w_i^t \cdot |h_t(x_i) - c(x_i)|}{\sum_{i=n+1}^{n+m} w_i^t} \tag{3}$$

In this formula, ϵ_t denotes the error, w_i^t is the weight of the document, $h_t(x_i)$ is the hypothetical class of document x_i as predicted by the model and $c(x)$ is a theoretical function that maps document x_i to its true class. Note that the sum $\sum_{i=n+1}^{n+m}$ only takes the last m document into account, namely $x_i \in T_s$. Hereafter, the weight of each document is adjusted: the weights of the documents that originally belonged to T_d ($x_i \in x_0, x_1 \dots x_n$) are multiplied by $w_i^{t+1} = w_i^t \cdot (1/(1 + \sqrt{2 \ln n/N}))^{|h(x_i) - c(x_i)|}$, while documents from T_s (or $x_i \in x_{n+1}, x_{n+2} \dots x_{n+m}$) are multiplied by $w_i^{t+1} = w_i^t \cdot (\epsilon_t/(1-\epsilon_t))^{-|h_t(x_i) - c(x_i)|}$ as portrayed in **formula 4**.

$$w_i = \begin{cases} w_i = w_i \cdot (1/(1 + \sqrt{2 \ln n/N}))^{|h_t(x_i) - c(x_i)|} & \text{if } 0 \geq i \geq n \\ w_i = w_i \cdot (\epsilon_t/(1-\epsilon_t))^{-|h_t(x_i) - c(x_i)|} & \text{if } n > i \geq m \end{cases} \tag{4}$$

This happens for several iterations, until an equilibrium is reached. At this point, the documents $x_i \in T_d$ that transfer irrelevant information are left with very low weights, whereas relevant documents that decrease the error are left with a higher weight. This way, only relevant documents from T_d are used in the new model for use on T_s. On the other hand, the weights of false predicted documents $x_i \in T_s$ increase, creating more emphasis on the deviating structure of the new distribution. Hereafter, when a document is provided for prediction, the prediction is performed using **formula 5**.

$$h_f(x_i) = \begin{cases} 1 & \text{if } \prod_{t=N/2}^{N} (\epsilon_t/(1-\epsilon_t))^{-h_t(x_i)} \geq \prod_{t=N/2}^{N} (\epsilon_t/(1-\epsilon_t))^{-\frac{1}{2}} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Here, $h_f(x_i)$ denotes the final prediction of the model on document x_i. This formula has a binomial output of only 0 or 1. The formula to be used for multinomial classification problems is therefore adjusted in **section 3.2.2**.

2.3 RQ3: On k-Means Clustering

Earlier literature points out that k-means clustering is a sufficient clustering method for text classification that does not require many iterations[1]. The clustering algorithm works by creating random 'seeds' with random values for each variable in the model. These seeds calculate which instances in the model are closest to them and divide the documents over them as a group. Then, the seeds move towards the average of the values of all close documents. These two steps (choosing the closest documents and moving towards their average) are then repeated for several iterations until equilibrium. A disadvantage is the positioning of the initial seeds, that can influence the outcome. The clustering may therefore require several runs of the algorithm to achieve multiple solutions. The result with the lowest squared error can then be selected as a preferred option from the given solutions[7].

3 METHODOLOGY

3.1 Description of the data

The Open State Foundation has provided a REST API, with which to retrieve documents from their database. Using this API, documents of three municipalities have been gathered and put into three CSV-files: The municipalities of Utrecht, Arnhem and Amstelveen. The municipalities have been chosen as they had published the most documents that were ordered by category. These CSV-files have a size of 698,4 MB and 17771 entries (Utrecht), 159,7 MB and 5825 entries (Amstelveen) and 120,1 MB and 3671 entries (Arnhem). These CSV-files all follow the same format: one column for the name of the document (if given), a column for the category of the document, a column with the source content of the document and a last column stating the municipality of which the document originates from. Mind that the last column (municipality) is the same throughout any of the three datasets, as the datasets are municipality-specific already.

The source content of every document describes the wordly content of each document. This content is the text that will be used as predictor when classifying the documents. If the source was not given or empty, the document was removed from the dataset. An overview is given of the number of documents left per category and per municipality in **table 5**, **appendix A**.

The data was then preprocessed and only relevant categories were selected to remain in the dataset (more aptly described in **appendix A**). The number of documents left per category and per municipality are given in **table 1**. Two different subsets were extracted from the Utrecht dataset, one that contained overlapping categories with Arnhem (T^{Arn}) and one that contained overlapping categories with Amstelveen (T^{Ams}). **Appendix A** gives more information on the formation of these two subsets.

Table 1 – Documentfrequencies per municipality and per category in the final dataset.

Category	Utrecht	Arnhem	Amstelveen
Agenda	408	555	200
Besluitenlijst Raad	184	0	12
Moties	338	46	470
Raadsbrieven	1566	13	0
Schriftelijke vragen	2429	0	131
Toezeggingen	2948	31	324

3.2 Methods

Three algorithms were applied to the data: A Multinomial Naive Bayes model, Multinomial Naive Bayes with transfer-learning (TrAdaBoost) and unsupervised k-means clustering. For all three experiments, the precision, recall and F1-score was calculated. After that, the macro- and micro-average of these values were computed. A more elaborate overview of why and how these evaluationscores have been calculated is given in **appendix 2**.

3.2.1 RQ1: Multinomial Naive Bayes. From the original datasets T^{Arn}_{Utr} and T^{Ams}_{Utr}, samples were taken to be used as trainset. These samples will be referred to as T^{Arn}_{train} and T^{Ams}_{train}, and were of sizes 1200 and 900 respectively. Although these samples consisted of random documents, an equal amount of documents was included for every category, e.g. T^{Ams}_{train} consisted of 180 random documents of each of its five categories, adding up to a total of 900 documents. Random samples, rather than the whole datasets, were used for computational efficiency, insight into the distribution of results and to maintain the aforementioned balance in categories. As for the testsets, the complete sets of T^{Arn} and T^{Ams} were used and they shall be refered to as T^{Arn}_{test} and T^{Ams}_{test}. Two matrices of tokencounts were constructed from T^{Arn}_{train} and T^{Ams}_{train} using Sci-kit Learn libraries. TF-IDF was used creating these vectors, stopwords were filtered out and uni-, bi- and trigrams were taken into account. Two multinomial Naive Bayes models were constructed in Sci-kit Learn, for each of the two municipalities. One model was fit on the tokencount matrix of T^{Arn}_{train} and another on the tokencount matrix of T^{Ams}_{train}. These models were used on the contents of their accessory testsets (T^{Arn}_{test} and T^{Ams}_{test} respectively) to predict their classes. For each document $x_i \in T_{test}$, both a true class and a hypothetical class as predicted by the model had been obtained. Using this information, prediction, recall and the F1-score were computed for each class for both municipalities, along with the micro- and macro-averages of these values. This process was repeated 50 times with different even-balanced trainsamples to acquire an insight regarding the distribution of all evaluation scores.

3.2.2 RQ2: Naive Bayes and TrAdaBoost. To begin with, the formulas for error- and weightcomputation were adjusted: The work of [4] focused on binomial classification, as we can deduct from **formulas 3** and **4**. To assess whether a prediction matches the true category of a document, both formulas use $|h(x_i) - c(x_i)|$. This is fine in the case of two categories, when $h(x_i)$ and $c(x_i)$ are always 1 or 0. In that case, $|h(x_i) - c(x_i)|$ returns 0 in case of a right

prediction and 1 in case of an erroneous prediction. However, in the case of multiple categories, $|h(x_i) - c(x_i)|$ could return larger numbers, as $h(x_i)$ and $c(x_i)$ can take on larger numbers too. These numbers resemble nominal categories and therefore $|h(x_i) - c(x_i)|$ was changed to $\min(1, |h(x_i) - c(x_i)|)$, which returns 0's and 1's for right and wrong predictions respectively. The new formulas are given below (**6** and **7**). The error ϵ_t was also programmed to never reach 0. This was done, because if the error would reach 0, several divisions by 0 would appear later in the algorithm and the algorithm would not function correctly. When the error reached 0, a dummy prediction of average weight was created. This prediction would then always be wrong, so that the error could never reach 0.

$$\epsilon_t = \sum_{i=n+1}^{n+m} \frac{w_i^t \cdot \min(1, |h_t(x_i) - c(x_i)|)}{\sum_{i=n+1}^{n+m} w_i^t} \tag{6}$$

$$w_i^{t+1} = \begin{cases} w_i^{t+1} = w_i^t \cdot (1/(1+\sqrt{2\ln n/N}))^{\min(1, |h_t(x_i)-c(x_i)|)} & \text{if } 0 \geq i \geq n \\ w_i^{t+1} = w_i^t \cdot (\epsilon_t/(1-\epsilon_t))^{-1 \cdot \min(1, |h_t(x_i)-c(x_i)|)} & \text{if } n > i \geq m \end{cases} \tag{7}$$

For the prediction of classes, **formula 5** was changed to **8**, so that multiple classes could be predicted. $\max(0, (1 - |(h_t(x_i) - c)|))$ returns then 1 when class c matches prediction $h_t(x_i)$, and 0 otherwise. In the latter case, $(\epsilon_t/(1-\epsilon_t))^{-1 \cdot \max(0, (1-|(h_t(x_i)-c)|))}$, evaluates to 1, so that the cumulative product does not change. Otherwise, the cumulative product may increase or decrease, dependent on the value ϵ_t. Therefore, the predicted class will be that class that is predicted most with the lowest error in the second half of iterations.

$$h_f(x_i) = \arg\max_c \prod_{t=N/2}^{N} (\epsilon_t/(1-\epsilon_t))^{-1 \cdot \max(0, (1-|(h_t(x_i)-c)|))} \tag{8}$$

For each of Arnhem and Amstelveen, an appropriate same-distribution subset T_s^{city} was necessary to be constructed from T^{city} and a different-distribution subset T_d^{city} from T_{Utr}^{city} (as described in **section 2.2**). Therefore, samples were taken from the complete datasets T^{Arn} and T^{Ams}, which resulted in a T_s^{Arn} and a T_s^{Ams} of 10 random documents for every category. For T_d^{Arn} and T_d^{Ams}, the same samples were used as for T_{train}^{Arn} and T_{train}^{Ams}. This way, a fair comparison between the models with and without transfer-learning could be performed later. Again, these samples were of sizes 1200 (T_d^{Arn}) and 900 (T_d^{Ams}) and they were random, but even-balanced in terms of category. A testset T_{test}^{city} was constructed for both municipalities, which consisted of all documents of the municipality that had not been included in T_s^{city}. T_d^{city} and T_s^{city} were combined into one dataset T^{city}. A matrix of token-counts was constructed from this set, similarly to how this was done for the multinomial Naive Bayes procedure. Again, multinomial Naive Bayes models were fitted on these matrices that had just been constructed. After this, for 40 iterations, the error was computed and the weight of each document was adjusted as described in **2.2**: the error ϵ_t was computed according to **formula 6** and then all weights W_i^t were multiplied using **formula 7**. This reweighted model was used to predict all categories of the documents in the testset T_{test}^{city}, using **formula 8**. Each document in the testset was then labeled with both its true category and a hypothetical label as predicted by the model. With these two labels for every document x_i, the precision, recall and F1-score could be computed, as well as their micro- and macro-averages. This happened 50 times with different samples of T_s^{city} and T_d^{city}.To compare these results with the results of Naive Bayes without transfer-learning, the data was checked for a normal distribution with an omnibus test [2]. Because the same training samples were used for both tests, the results were 'paired', so a two-tailed Wilcoxon signed-rank test was performed on the two sets of results.

3.2.3 RQ3: k-Means Clustering. For the last algorithm, a matrix of tokencounts was created for every dataset: the two datasets from Utrecht (T^{Ams} and T^{Arn}) and the two datasets from the smaller municipalities (T_{Utr}^{Ams} and T_{Utr}^{Arn}). Using Sci-kit Learn libraries, a k-means clustering algorithm was deployed on all four datasets, where k was set to the number of categories that the datasets contained (4 for T^{Arn} and T_{Utr}^{Arn} and 5 for T^{Ams} and T_{Utr}^{Ams}). These clustering algorithms included 50 iterations, and repeated the process 10 times with different random seeds to conclude the best option with the lowest inertia. When an appropriate solution was found, the micro-averaged F1-score was computed for every possible cluster-documenttype combination. The combination with the highest score was documented with its precision, recall and F1-score and the micro- and macro-averages of these values. This process was repeated 20 times.

4 EVALUATION

4.1 RQ1: Multinomial Naive Bayes

A summary of results is given in table 3. A more detailed version of these results is to be found in appendix 3.1, table 6. These tables show a strong difference in micro- and macro-average for all three evaluation-scores for the Arnhem dataset. This indicates that the results were better for larger categories than for less populated categories. In this case, 'Agenda' stands out as a category with relative high values for the precision, recall and F1-score, while accounting for more than 85% of all documents (appendix 3.1, table 6). For Amstelveen, less differences between the two sorts of average is to be observed. However, the micro-average was lower than for the Arnhem dataset, indicating a lower performance in terms of the absolute number of documents.

4.2 RQ2: Naive Bayes and TrAdaBoost

The performance of the implementation of TrAdaBoost is given in **table 4** and, more detailed, in **appendix 3.2**, **table 7**. Most prominently, all evaluation scores have risen in comparison with the Naive Bayes results. The difference between the macro- and micro-average became less outspoken for Arnhem, but increased a little for Arnhem.

The results of both methods were checked for normality, using an omnibus test. Many sets of results rejected the null-hypothesis of belonging to a normal distribution (**appendix 3.2**, **table 8**). Therefore, a non-parametric Wilcoxon signed-rank test was performed on the data. Results are portrayed in **figure 1**, and in parallel coordinate plots in **appendix 3.2**, **figure 2**. P-values of P<0.0001 (Arnhem, micro-averaged F1-scores), P<0.001 (Amstelveen, micro-averaged F1-scores), P<0.0001 (Arnhem, macro-averaged F1-scores) and P<0.0005 (Amstelveen, macro-averaged F1-scores) were found. This is with a significance level at $\alpha = 0.0125$, using a Bonferroni correction for 4 comparisons on an initial significance level of $\alpha = 0.05$.

4.3 RQ3: k-Means Clustering

The micro- and macro-averaged F1-scores of the k-means clustering are given in **table 2**. Both different distribution datasets achieved F1-scores of >0.9, both when micro- and macro-averaged. The same-distribution datasets however, only reached micro-averaged F1-scores of little more than 0.6 and macro-averaged F1-scores around 0.45. There is a subtle difference between micro- and macro averages for the same-distribution datasets, indicating an imbalance in performance between categories. More elaborate results per category are given in **appendix 3.3**, **figure 3**.

Table 2 – Micro- and macro-averaged F1-Score of unsupervised k-means clustering on the 4 datasets.

	Micro-average	Macro-average
T_{Utr}^{Arn}	0.9044	0.9037
T_{Utr}^{Ams}	0.9461	0.9269
T^{Arn}	0.6142	0.4641
T^{Ams}	0.6346	0.4562

Table 3 – Micro- and macro-averaged precision, recall and F1-score for a multinomial Naive Bayes model on both datasets. Note that micro-averaging results in the same value for precision, recall and F1-score, as discussed in **appendix 2**.

	Arnhem			Amstelveen	
	Micro-average	Macro-average		Micro-average	Macro-average
Precision	0.8009	0.3277	Precision	0.5046	0.4807
Recall	0.8009	0.4904	Recall	0.5046	0.5167
F1-Score	0.8009	0.3018	F1-Score	0.5046	0.3920

Table 4 – Micro- and macro-averaged precision, recall and F1-score for when TrAdaBoost was applied to a multinomial Naive Bayes model on both datasets. Note that micro-averaging results in the same value for precision, recall and F1-score, as discussed in **appendix 2**.

	Arnhem			Amstelveen	
	Micro-average	Macro-average		Micro-average	Macro-average
Precision	0.8363	0.5498	Precision	0.5746	0.5062
Recall	0.8363	0.5598	Recall	0.5746	0.5364
F1-Score	0.8363	0.5020	F1-Score	0.5746	0.4527

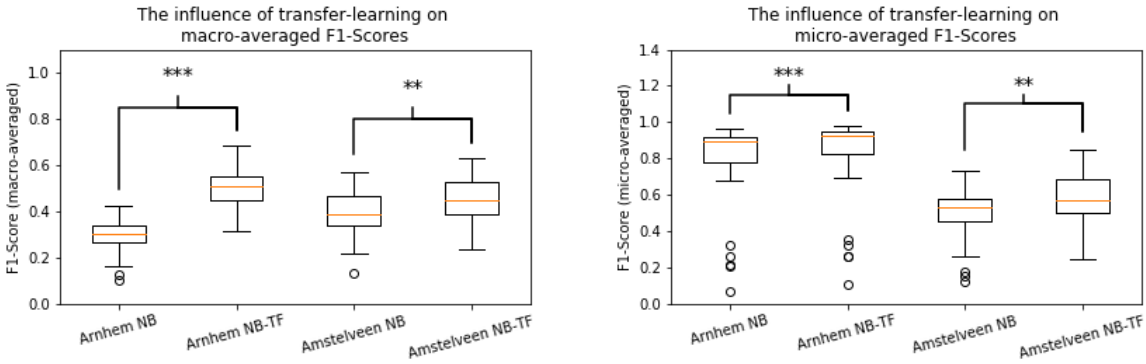

Figure 1 – Difference in F1-scores when TrAdaBoost is implemented on a multinomial Naive Bayes model. On the left, the macro-averaged F1-Scores are given on the y-axis, whereas on the right, the micro-averaged F1-Scores are given. In both of these figures, two boxplots are illustrated per municipality. One labeled as 'NB', indicating results as obtained by use of a multinomial Naive Bayes model, and another labeled as 'NB-TF', which displays results for a Multinomial Naive Bayes model with transfer-learning. In each situation, for both micro- and macro-averaged F1-scores and for both municipalities, a significant rise in F1-score is obtained. Using a Wilcoxon signed-rank test, for the micro-averaged F1-scores of Arnhem, a P-value of $P<0.0001$ was obtained .For Amstelveen this test resulted in $P<0.001$. Using macro-averaged values, a P-value of $P<0.0001$ was found for Arnhem and $P<0.0005$ for Amstelveen.

5 CONCLUSIONS

These results indicate that using TrAdaBoost significantly increases the performance of a multinomial Naive Bayes model. For Arnhem, the increase in F1-score was primarily reached through higher precision and, to lesser extend, higher recall. This increase demonstrates itself most prominently through macro-averaged scores, as TrAdaBoost seems to have boosted performance mostly for smaller classes (**appendix 3.1**, **table 6** and **appendix 3.2**, **table 7**). For Amstelveen, precision and recall rose equally. Micro-averaged scores were influenced a bit more than macro-averaged scores, as larger categories seemed to benefit most from TrAdaBoost (**appendix 3.1**, **table 6** and **appendix 3.2**, **table 7**).

As illustrated in **figure 2**, increased performance was the case for most samples, but not for all samples. There were cases when TrAdaBoost decreased the F1-score and it is yet unclear when this happens. A glance at **figure 2** (**appendix 3.2**) suggests that a decrease in F1-score tends to happen more often in cases where a relative high F1-score was achieved by the multinomial Naive Bayes model. These results, however, are not definite enough to conclude this thought. Further research may direct itself towards this flaw in the algorithm and possibly improve the algorithm.

The results from the k-means clustering algorithm give that the subsets from Utrecht exhibited clearly defined categories, as the algorithm found clusters that overlapped with these categories. For the Arnhem and Amstelveen dataset, the results were less high, which may certainly be because of their lesser size. These results suggest that the increased performance of TrAdaBoost was not due to qualitative differences between T_d and T_s.

Multi-class transfer-learning algorithms are scarce in literature, and no multi-class version of TrAdaBoost was found. Therefore, the model as proposed by [4] has been generalized to work with multi-class problems. So far, this altered version of the model seems to yield positive results. It is suggested for future applications of naive Bayes models to consider differences between datasets, as performance may be increased by use of transfer-learning. Although this paper only describes the application of TrAdaBoost on naive Bayes, the original and multi-class implementation of TrAdaBoost could be used with most algorithms (regressions, support vector machines, etc.). Thus, even for other models, transfer-learning deserves some thought, as it possibly offers an useful extension to modern machine learning.

REFERENCES

[1] Charu C Aggarwal and ChengXiang Zhai. A survey of text clustering algorithms. In *Mining text data*, pages 77–128. Springer, 2012.
[2] Ralph B d'Agostino. An omnibus test of normality for moderate and large size samples. *Biometrika*, 58(2):341–348, 1971.
[3] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Transferring naive bayes classifiers for text classification. In *AAAI*, volume 7, pages 540–545, 2007.
[4] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.
[5] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.
[6] David Heckerman. A tutorial on learning with bayesian networks. microsoft research. 1995.
[7] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.
[8] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.
[9] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
[10] Lorien Y Pratt. Discriminability-based transfer between neural networks. In *Advances in neural information processing systems*, pages 204–211, 1993.
[11] Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 616–623, 2003.
[12] Daniel M Roy and Leslie Pack Kaelbling. Efficient bayesian task-level transfer learning. In *IJCAI*, volume 7, pages 2599–2604, 2007.
[13] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.
[14] Lisa Torrey and Jude Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 1:242, 2009.

Table 5 – Documentfrequencies per municipality and per category in the initial dataset. Categories with less than 10 documents have been grouped as 'Other' and 35 categories starting with 'Videotulen...' have been grouped as 'Videotulen'.

Category	Utrecht	Arnhem	Amstelveen
Agenda	408	641	200
Agendapunt	3015	2137	3087
Besluitenlijst B&W	1318	0	0
Besluitenlijst Raad	184	0	0
Commissiebrieven M&S	1634	0	0
Commissiebrieven S&R	2005	0	0
Gemeentebladen	1103	0	0
Moties	338	149	606
RSS	211	0	0
Raadsbrieven	1568	125	0
Raadsverslagen	281	0	0
Schriftelijke vragen	2429	0	209
Toezeggingen	0	133	625
Toezeggingen S&R	1783	0	0
Toezeggingen M&S	1734	0	0
Toezeggingen Raad	1123	0	0
Verslagen S&R	41	0	0
Verslagen Vragenuur	19	0	0
Videotulen	253	0	0
Amendementen	0	117	291
Artikel 44 vragen	0	141	0
Ingekomen stukken	0	192	0
Raadskamerbrieven	0	34	0
Besluitenlijsten ABM	0	0	48
Besluitenlijsten B&S	0	0	51
Besluitenlijsten van de Raad	0	0	114
Besluitenlijsten RWN	0	0	60
Informatieve Stukken College	0	0	462
Mondelinge Vragen	0	0	68
Other	16	2	4

A PREPROCESSING

Before analysis, the datasets were preprocessed to only contain relevant categories. Different categories of the Utrecht dataset ('Toezeggingen M&S', 'Toezeggingen Raad' and 'Toezeggingen S&R') were merged into one category ('Toezeggingen'), that was also found in the datasets of Arnhem and Amstelveen. One category of Amstelveen ('Besluitenlijsten van de raad') was renamed ('Besluitenlijst Raad') to match the homonymous category in the Utrecht dataset. The largest category 'Agendapunt' was removed from the dataset as it showed inconsistent content and seemed to be used as a default category for unlabeled documents. Also, hypertextlinks and punctuation were removed and all characters were converted to lower case.

From all retrieved documents, only a few categories were selected to be used as there had to be an overlap in categories between the Utrecht-trainset and the Arnhem- and Amstelveen-testsets. As Arnhem and Amstelveen carried different categories, this meant that the overlap in categories between Utrecht and Arnhem was not equal to the overlap of Utrecht and Amstelveen. For this reason, two different subsets were extracted from the Utrecht dataset: One to predict Arnhem-labels (T_{Utr}^{Arn}) and another for Amstelveen (T_{Utr}^{Ams}). Four classes were included in T_{Utr}^{Arn} and five classes in T_{Utr}^{Ams}. The Number of documents per category are given in **table 1** (**section 3.1**). This filtering out of documents from other categories lead to a dataset of less documents: 5260 documents were left of T_{Utr}^{Arn} to train on, and 6307 documents of T_{Utr}^{Ams}. The datasets of labeled documents of Arnhem and Amstelveen consisted of 645 documents for Arnhem (T^{Arn}) and 1137 documents for Amstelveen (T^{Ams}).

2 ELABORATION ON THE EVALUATION SCORES

The precision, recall and F1-score are used as evaluation metric, to take both false positives and false negatives into account[13]. The formulas for these scores are given below (**formulas 9**, **10** and **11**). In these formulas, P_c denotes the precision of class c. Likewise, R_c denotes the recall and F_{1c} denotes the F1-score of a certain class. Furthermore, TP_c refers to the true positive predictions of a class c (the total number of times a document was correctly classified as class c). FP_c stands for false positives, meaning, the total number of times documents were incorrectly classified as class c. Finally, FN_c counts the number of false negatives, or the total number of times a document was incorrectly classified as not being class c. The precision of a class indicates the fraction of all true positive predictions of that class (TP_c) among all documents classified (correctly or incorrectly) as being part of that class ($TP_c + FP_c$). The recall of a class gives the fraction of documents that are correctly classified as that class (TP_c) among all documents that - predicted or not - truly belong to that class ($TP_c + FN_c$). The F1-score gives the harmonic mean of the precision and recall.

$$P_c = \frac{TP_c}{TP_c + FP_c} \tag{9}$$

$$R_c = \frac{TP_c}{TP_c + FN_c} \tag{10}$$

$$F_{1c} = \frac{2 \cdot P_c \cdot R_c}{P_c + R_c} \tag{11}$$

For all evaluations, each of the three scores (the precision, recall and F1-score) was averaged in two ways: using the macro-average and micro-average. The macro-average simply takes the scores of all classes and averages them. In this way, every class contributes the final score equally. The micro-average is calculated using the true counts of correct and incorrect predictions. This means, that if one class consists out of more documents than the other classes, then that class will have more impact on the micro-average. The macro-average is given to cope with imbalance in the testsets, and to acknowledge smaller categories. The micro-average is given to evaluate the total of all documents, while acknowledging each document equally, regardless of their class. Formula's for the macro- and micro-average of precision are given below (**12** and **13**), where C denotes the total number of classes.

$$\text{macro-average} = \frac{\sum_{c=0}^{C} P_c}{C} \tag{12}$$

$$\text{micro-average} = \frac{\sum_{c=0}^{C} TP_c}{\sum_{c=0}^{C} TP_c + FP_c} \tag{13}$$

One should note that, in multi-class situations, micro-averaging always results in the same values for precision, recall and F1-score. This is because a false positive prediction for one class is always a false negative prediction for another class. Therefore, $\sum_{c=0}^{C} FP_c = \sum_{c=0}^{C} FN_c$, and if these terms are substituted in the formulas for micro-averaged precision and recall, one can then deduct that the micro-averages of precision and recall are the same, as depicted in **formula 14**. Here, the term on the left is the micro-averaged precision and the term on the right is the micro-averaged recall.

$$\begin{aligned} \frac{\sum_{c=0}^{C} TP_c}{\sum_{c=0}^{C} TP_c + FP_c} &= \frac{\sum_{c=0}^{C} TP_c}{\sum_{c=0}^{C} TP_c + \sum_{c=0}^{C} FP_c} \\ &= \frac{\sum_{c=0}^{C} TP_c}{\sum_{c=0}^{C} TP_c + \sum_{c=0}^{C} FN_c} = \frac{\sum_{c=0}^{C} TP_c}{\sum_{c=0}^{C} TP_c + FN_c} \end{aligned} \tag{14}$$

The micro-average of the F1-score is computed from the micro-averaged precision and the micro-averaged recall. **Formula 11** can be used for this, but the class-specific values for precision and recall should be substituted with their micro-averaged counterparts. When precision and recall are the same, the F1-score will also take on their values. If we substitute the precision and recall in **formula 11** with any value x, then **formula 15** shows how the F1-score is also set to x. Since the precision and recall are always the same when micro-averaging, the F1-score will take on the same value.

$$F_1 = \frac{2 \cdot x \cdot x}{x + x} = \frac{2x^2}{2x} = x \tag{15}$$

2.1 When not all documents are assigned to a class

It should be noted that the logic behind equal micro-averages only applies when a model labels every document. Some models may leave a document unlabeled when the document conforms to multiple classes or no classes at all. This choice may positively affect precision (less false positive predictions) at the risk of decreasing recall (more false negative predictions). When a document is left unlabeled, it counts as a false negative prediction for its true class, but not as a false positve prediciton for another class. So in this situation, $\sum_{c=0}^{C} FP_c \neq \sum_{c=0}^{C} FN_c$ and **formula 14** does not apply.

The TrAdaBoost algorithm, unfortunately, does not lend itself for this practice for the following reason: As the algorithm converges, a very low error ϵ_t is reached and consistent models are produced, leading to consistent predictions $h_t(x_i)$ (see **section 3.2.2**). As discussed in **section 3.2.2**, a prediction is made by evaluating **formula 16**. When class c is predicted ($h_t(x_i) = c$), **formula 16** evaluates $\epsilon_t/(1-\epsilon_t)^{-1}$. As the model converges, and ϵ becomes smaller, very high values are computed (formula 17). And as consistent predictions are made, the predicted class $h_t(x_i)$ is almost always the same class for every iteration t. When then all values of c are put into **formula 16**, the formula will output a very high number for one class, and low numbers for the other classes. **Formula 16** is therefore not able to return a 'nuanced' output. Since all outputs conform to only one class c, it is nearly inevitable to label all documents with one class.

$$h_f(x_i) = \arg\max_c \prod_{t=N/2}^{N} (\epsilon_t/(1-\epsilon_t))^{-1 \cdot \max(0,(1-|(h_t(x_i)-c)|))} \tag{16}$$

$$\lim_{\epsilon \to 0} (\epsilon_t/(1-\epsilon_t))^{-1} = \infty \tag{17}$$

3 ELABORATION ON RESULTS

3.1 Multinomial Naive Bayes

Table 6 – Precision, Recall and F1-Scores of a multinomial Naive Bayes model on the Arnhem and Amstelveen dataset. Note that precision, recall and F1 become the same when micro-averaged, as discussed in **appendix 2**.

Arnhem	n	precision		recall		F1	
Category		μ	SE	μ	SE	μ	SE
Agenda	555	0.9995	0.0002	0.8143	0.0309	0.8752	0.0268
Moties	46	0.0346	0.0038	0.7500	0.0593	0.0650	0.0069
Raadsbrieven	13	0.0026	0.0015	0.0370	0.0191	0.0043	0.0023
Toezeggingen	31	0.2744	0.0332	0.3603	0.0238	0.2626	0.0224
Micro-average		0.8009	0.0301	0.8009	0.0301	0.8009	0.0301
Macro-average		0.3277	0.0087	0.4904	0.0174	0.3018	0.0096
Amstelveen	n	precision		recall		F1	
Category		μ	SE	μ	SE	μ	SE
Agenda	200	0.8818	0.0255	0.4703	0.0276	0.5845	0.0275
Besluitenlijst Raad	12	0.2245	0.0319	0.3180	0.0515	0.2383	0.0348
Moties	470	0.1520	0.0227	0.5351	0.0519	0.1613	0.0212
Schriftelijke vragen	131	0.4635	0.0596	0.6408	0.0487	0.4246	0.0512
Toezeggingen	324	0.8524	0.0323	0.6287	0.0632	0.6829	0.0517
Micro-average		0.5046	0.0194	0.5046	0.0194	0.5046	0.0194
Macro-average		0.4807	0.0148	0.5167	0.0205	0.3920	0.0144

3.2 TrAdaBoost

Table 7 – Precision, Recall and F1-Scores of TrAdaBoost applied to a multinomial Naive Bayes model on the Arnhem and Amstelveen dataset. Note that precision, recall and F1 become the same when micro-averaged, as discussed in **appendix 2**.

Arnhem	precision		recall		F1	
Category	μ	SE	μ	SE	μ	SE
Agenda	0.9830	0.0014	0.8498	0.0321	0.8882	0.0268
Moties	0.9389	0.0199	0.8272	0.0270	0.8545	0.0215
Raadsbrieven	0.0124	0.0042	0.2067	0.0494	0.0205	0.0063
Toezeggingen	0.2650	0.0344	0.3556	0.0302	0.2448	0.0209
Micro-average	0.8363	0.0293	0.8363	0.0293	0.8363	0.0293
Macro-average	0.5498	0.0095	0.5598	0.0114	0.5020	0.0115
Amstelveen	precision		recall		F1	
Category	μ	SE	μ	SE	μ	SE
Agenda	0.6391	0.0306	0.8874	0.0376	0.6979	0.0316
Besluitenlijst Raad	0.0042	0.0022	0.0550	0.0275	0.0076	0.0040
Moties	0.9094	0.0073	0.4252	0.0292	0.5472	0.0281
Schriftelijke vragen	0.3271	0.0208	0.5425	0.0430	0.3573	0.0215
Toezeggingen	0.6513	0.0306	0.7719	0.0352	0.6523	0.0232
Micro-average	0.5746	0.0178	0.5746	0.0178	0.5746	0.0178
Macro-average	0.5062	0.0084	0.5364	0.0112	0.4527	0.0130

Table 8 – Results of omnibus test, with the null-hypothesis being that the results come from a normal distribution. When looking at micro-averaged F1-scores, only the TrAdaBoost results from Amstelveen do not reject this null-hypothesis, the other sets of results reached significantly low P-values. Regarding the macro-averaged F1-scores, only the Naive Bayes results from Arnhem did reject the null-hypothesis.

	micro-averaged		macro-averaged	
Municipality	Naive Bayes	TrAdaBoost	Naive Bayes	TrAdaBoost
Arnhem	$P < 0.0001$	$P < 0.0001$	P<0.0500	P=0.5451
Amstelveen	$P < 0.0100$	$P = 0.6715$	P=0.7361	P=0.6949

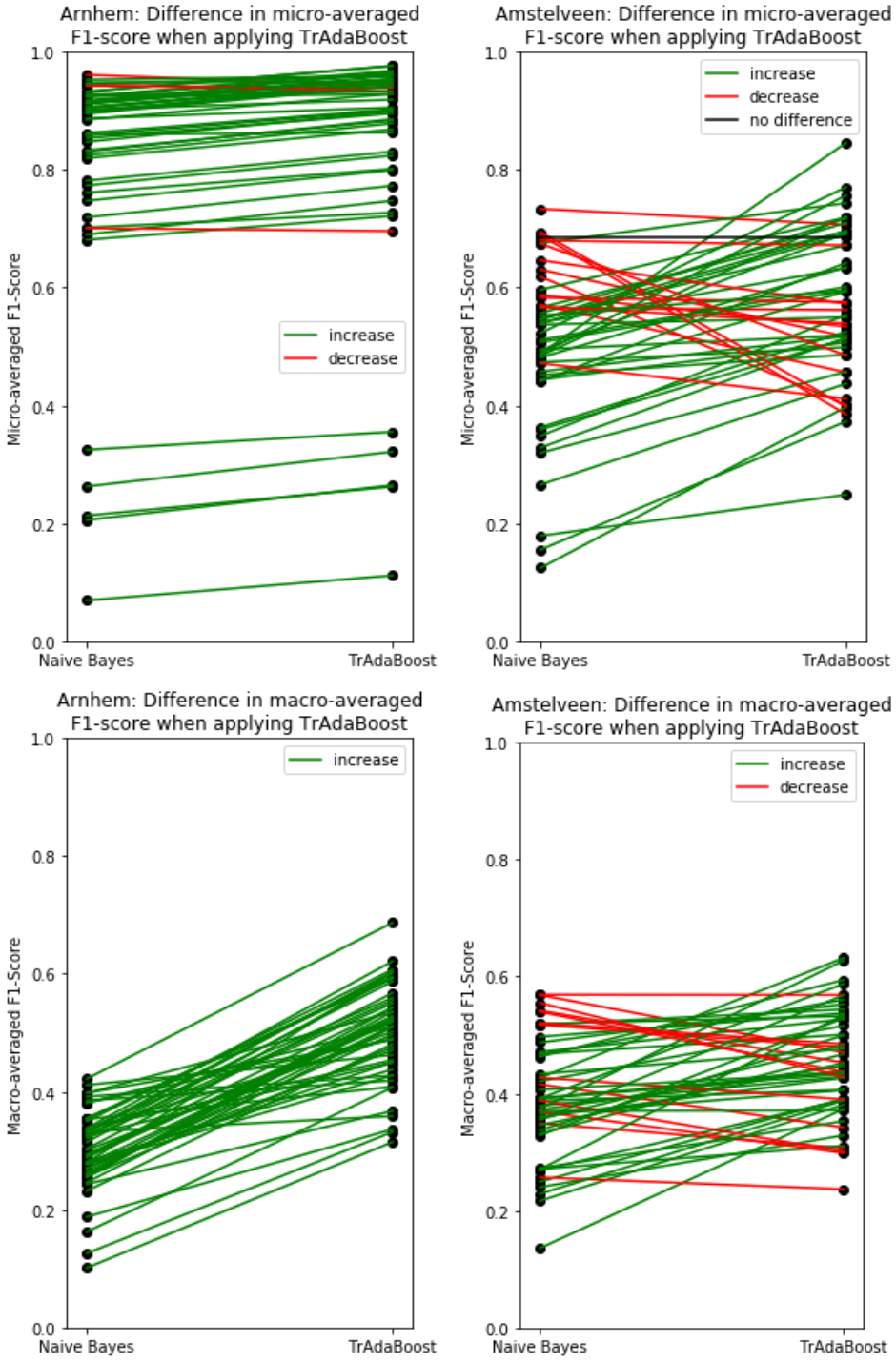

Figure 2 – Difference in F1-scores when TrAdaBoost is implemented onto a multinomial Naive Bayes model. On top, the micro-averaged F1-Scores are given on the y-axis, whereas on the bottom, the macro-averaged F1-Scores are given on the y-axis. The two plots on the left represent Arnhem and the two plots to the right illustrate the results of Amstelveen. In all plots, a dot on the left stands for a result of the multinomial Naive Bayes model, and a dot on the right for a result when TrAdaBoost is implemented. Results from the same samples are connected with a line, which is coloured green when TrAdaBoost improved the result, red when it diminished performance and black when performance did not change. In each situation, for both micro-and macro-averaged F1-scores and for both municipalities, a significant rise in F1-score is obtained. Using a Wilcoxon signed-rank test, for the macro-averaged F1-scores of Arnhem, a P-value of $P<0.0001$ was obtained. For Amstelveen this test resulted in $P<0.0005$. Using micro-averaged values, a P-value of $P<0.0001$ was found for Arnhem and $P<0.001$ for Amstelveen.

3.3 k-Means Clustering

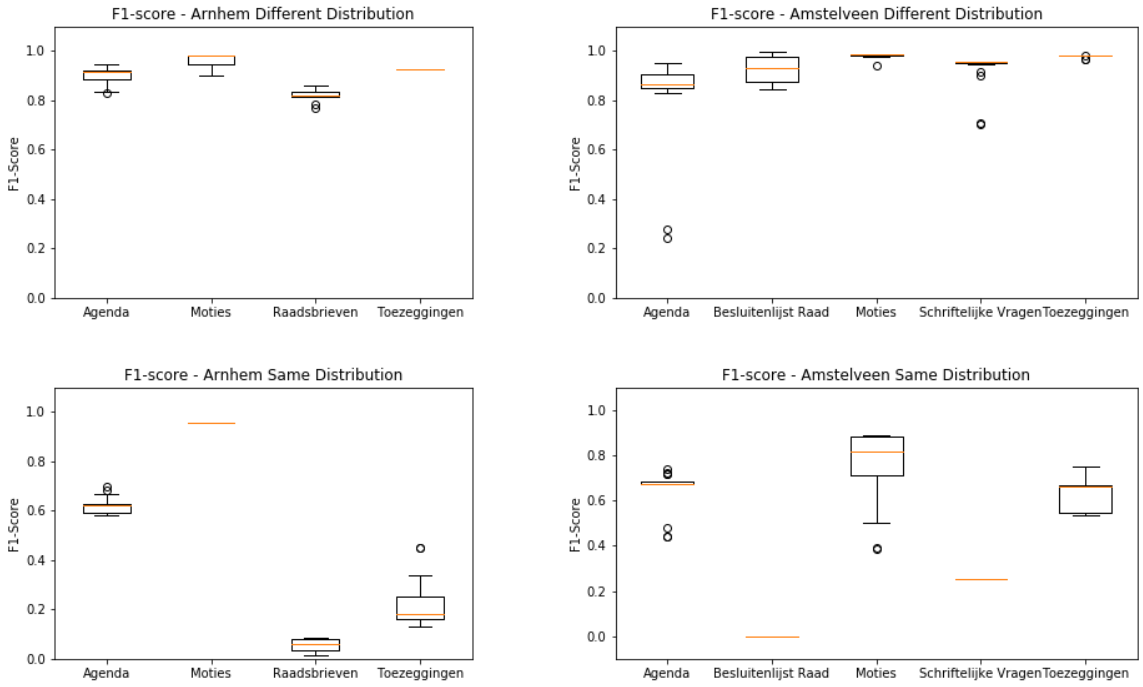

Figure 3 – F1-scores when k-means clustering is applied to every dataset. The two different-distribution datasets (T_{Utr}^{Arn} and T_{Utr}^{Ams}) exhibit an overall good performance. The two same-distribution datasets (T^{Arn} and T^{Ams}) however, have a more 'mixed' performance with means of F1-scores reaching close to 0 for some categories.