University of Sussex

Department of Informatics

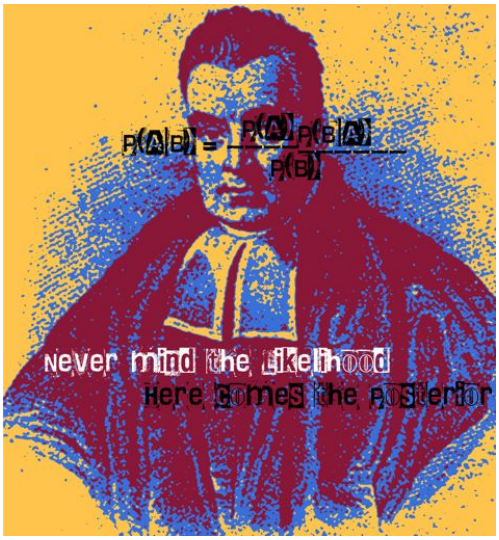
Bachelor of Science
in
Computer Science and Artificial Intelligence

Final Year Project

Scaling Semi-Supervised Multinomial Naïve Bayes

Author:
Thomas H. Kober
Candidate Number: 19214

Supervisor:
Prof. David Weir



2014

Statement of Originality

This report is submitted as part requirement for the degree of Bachelor of Science in Computer Science and Artificial Intelligence at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

Thomas Kober
May 9, 2014

Acknowledgements

I would like to thank my supervisor, Prof. David Weir, for pointing me to the paper by Lucas and Downey (2013) that builds the basis for this project, and for his invaluable input and numerous discussions during the project. I would furthermore like to express my gratitudes to the whole TAG lab team, who always had a ”minute” whenever I had a question of some sort.

I would also like to thank my fiancée Julia for her support and understanding during this project in particular, and the whole degree in general.

Summary

The project originated during a summer internship in the Text Analytics Group at the University of Sussex. My aim for the internship was to implement a Maximum Entropy classifier (MaxEnt) for an active learning tool called DUALIST, to replace the Multinomial Naïve Bayes classifier used in the application, as MaxEnt was found to be superior to Naïve Bayes in previous studies, i.e. Nigam (1999). However, I found that for the purpose of social media analysis on Twitter in the given active learning scenario, the MaxEnt model was not a suitable choice as it was too slow to train and exhibited a worse classification performance than the Naïve Bayes model on the Twitter datasets. The results of my investigation during the internship then led on to this project, where I aim to compare several different semi-supervised learning algorithms to enhance the performance of the Multinomial Naïve Bayes model implemented in DUALIST.

I implemented 3 different semi-supervised algorithms, Expectation-Maximization, Semi-supervised Frequency Estimate and Feature Marginals, together with a Multinomial Naïve Bayes classifier and investigated their performance on 16 real-world text corpora. 15 of these datasets contain Twitter data and were used to test the implemented algorithms in their target domain. The remaining dataset, the Reuters ApteMod corpus, was used for validating my implementations.

In addition to the comparison of the algorithms, I further investigated a variety of techniques to improve the performance of the classifier itself and the active learning process as a whole. These methods include the usage of uniform class priors for classification, randomly undersampling the majority class in case of class imbalances, optimising the free parameters in the Expectation-Maximization algorithm and using different uncertainty sampling techniques in the active learning process.

As the results show, using a Multinomial Naïve Bayes classifier, together with the Feature Marginals algorithm, generally produced the best results across datasets and configurations. Performance improvements could also be achieved by undersampling the majority class and by using an alternative uncertainty sampling technique in the active learning process. Using uniform class priors did not improve performance, and optimising the free parameters in the Expectation-Maximization algorithm is crucial for its classification performance.

The road to wisdom? Well, it's plain
And simple to express:
Err
and err
and err again,
but less,
and less,
and less.
- Piet Hein

Contents

List of Figures

List of Tables

1. Introduction

Natural Language Processing (NLP) on social media datasets has become increasingly popular in recent years, particularly among businesses and public institutions who are interested in public opinion towards a variety of topics. The amount of data in the form of text available on the internet, especially opinion based data, has exploded with the advent of social networks such as Twitter[1] or Tumblr[2], and is still increasing at an enormous rate. This quickly growing amount of readily available text represents a problem for supervised Machine Learning approaches, which are typically applied for NLP tasks such as sentiment analysis, because thousands or even tens of thousands of labelled documents are required for training a classifier model. However, for a real-time analysis of an event on Twitter, neither the time to label such an amount of tweets, nor the manpower for such an undertaking, are typically available in practice. These challenges created the demand for a category of Machine Learning models that are fast, simple to implement and easily extensible by semi-supervised learning algorithms that leverage the information gained from unlabelled documents.

This project investigates a variety of ways to improve a Multinomial Naïve Bayes (MNB) classifier, which is a commonly used Machine Learning model for many Natural Language Processing problems, because it meets the above described demands of being fast, simple to implement and easily extensible by semi-supervised learning techniques. In particular, the aim is to improve the classification performance of the MNB classifier that builds the core of DUALIST, an application for creating bespoke classifiers for a variety of different NLP tasks (Settles, 2011); (Settles and Zhu, 2012).

DUALIST leverages the idea of active learning to enable users to interactively annotate documents, or even individual words in documents, with previously defined target labels. Figure 1 below shows a screenshot of the original DUALIST implementation. Documents in the dataset are displayed in the area marked with 1, where originally, 20 documents wet displayed for labelling per learning iteration. Individual features per target label are displayed in lists in the area marked with 2. Settles (2011) shows that within only 15 minutes of labelling effort, users are able to create near state of the art classifiers with DUALIST. The major reason for achieving a high-quality model in such a short amount of time, is the aforementioned possibility for users to label individual terms alongside whole

[1] `http://www.twitter.com`
[2] `http://www.tumblr.com`

documents. The application has been extended and modified by researchers of CASM[3], and is applied in practice to analyse public opinion on Twitter, on topics ranging from reactions of the public to the verdict in the Mark Duggan case[4] to the UK population's mood towards the European Union. The intent of the researchers at CASM is to capture relevant Twitter conversations and easily perform analyses on the scraped tweets, while directly interacting with the data (Wibberley et al., 2013).

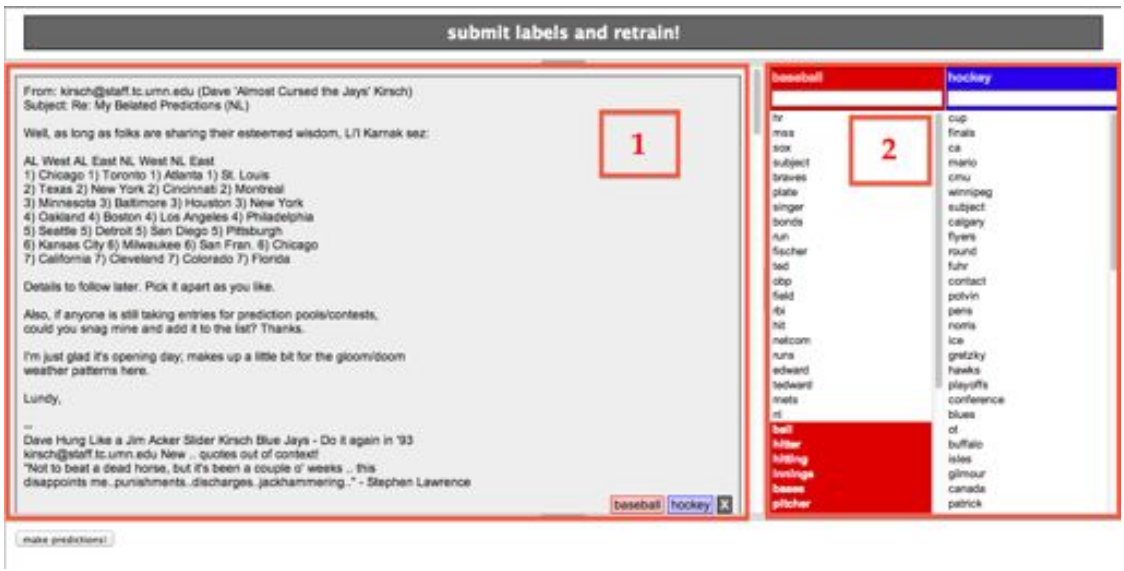


Figure 1: Screenshot of the original version of DUALIST
1: Collection of documents
2: Features per target label

For enhancing the classification performance of the Multinomial Naïve Bayes classifier used in DUALIST, I evaluated 3 different semi-supervised learning algorithms, Expectation-Maximization, Semi-supervised Frequency Estimate and Feature Marginals, and various combinations thereof. I further explored more technical aspects, such as using uniform class priors in the MNB model, undersampling the majority class in the case of severely imbalanced target classes, optimising the free parameters in the Expectation-Maximization algorithm and using different techniques to improve the active learning process in DUALIST.

I found that the Feature Marginals algorithm consistently outperformed Expectation-Maximization and Semi-supervised Frequency Estimate across all datasets, but that the

[3]A collaboration between the Social Science ThinkTank Demos and the Text Analytics Group at the University of Sussex, see `http://www.demos.co.uk/projects/casm`

[4]see `http://www.bbc.com/news/uk-24099368`, for an overview of the case

method comes at the cost of being computationally more expensive than the other two. The Feature Marginals algorithm also is superior to combinations of the 3 semi-supervised learning algorithms. I found the best 3 configurations to be Expectation-Maximization combined with Feature Marginals, the combination Semi-supervised Frequency Estimate, Expectation-Maximization and Feature Marginals, and the Feature Marginals algorithm by itself. There is no consistent winner among these 3 methods, however an MNB classifier enhanced with the Feature Marginals algorithm performed best in the majority of experiments. The running times among the 3 best performing algorithms do not vary much, as the Feature Marginals algorithm is part of every combination and has the biggest impact on speed.

This report is structured as follows: in section 2, I briefly discuss professional considerations where the BCS Code of Conduct affects this project. In sections 3 and 4, I introduce social media analysis and semi-supervised learning respectively, followed by a summary of the objectives and key methodologies that have been implemented for this project in section 5. In section 6 I introduce the datasets that have been used for the experiments in this report. In section 7 I present and discuss the obtained results and in section 8 I give an outline of future work. I conclude this report in section 9.

2. Professional Considerations

Any work done in this project has been executed in accordance with the BCS Code of Conduct[5]. The sections of the BCS Code of Conduct that immediately apply to this project are *1. Public Interest* and *2. Professional Competence and Integrity*. The major points in the aforementioned sections which I believe are most relevant to this project are 1 b), "to legitimately regard the rights of third parties", which in this project concerns any external software libraries, or other intellectual property such as academic papers, that are used, and 2 e), "to offer honest criticism of my work, and to accept and respect alternative viewpoints".

Point 1 b) is addressed in Appendix A, which contains an overview of, and full attribution to, the software packages that I used for this project, and throughout the report, by properly referencing my sources. Point 2 e) is addressed in section 7, the discussion of the obtained results for this work.

[5]see `http://www.bcs.org/category/6030`

3. Social Media Analysis

Social media analysis, especially sentiment analysis and opinion mining on Twitter, has become an area of increasing interest to the Natural Language Processing research community, where a growing number of academic papers are concerned with what is expressed in tweets. Publications range from predicting elections, i.e. Marchetti-Bowick and Chambers (2012); Tumasjan et al. (2010), to forecasting box-office revenues for movies, i.e. Asur and Huberman (2010), to anticipating the stock market, i.e. Bollen et al. (2011). Recently, a wider audience, including businesses and public institutions are concerned with the question, "what is public opinion towards X?", where X includes products, brands, events, political campaigns or politicians. The major focus on Twitter can be explained by the public nature of tweets and the easy accessibility of these via the Twitter API[6]. The Twitter public timeline offers the possibility to mine the raw and unfiltered opinion of a broad range of people talking ("tweeting") about an enormous variety of topics in a number of different languages.

Due to the expansion of interest in analysing tweets beyond academia, the methodology in the field needs to close the existing gap between research and practice. In academic publications, aspects such as actual training time, scalability of the chosen model to growing amounts of data, or ease of implementation are often ignored in favour of improving classification performance by a small fraction. In practical applications on the other hand, one would often prefer a Machine Learning model with slightly weaker classification performance if it offers the advantages of being fast to train and simple to implement. Further, academic NLP research is generally focused on working with a small number of completely hand annotated and manually compiled corpora (i.e. Reuters ApteMod[7] or 20 Newsgroups[8]). In practice however, the number and diversity of datasets is large, and their half-life rarely longer than a few weeks, with new trends and memes emerging in unforeseeable patterns.

The starting point for NLP practitioners typically is a raw and unlabelled text corpus, retrieved by simple keyword search from a social media API. In such a scenario, there usually is no, or hardly any, prior knowledge of the problem space, and hence, it is by no means guaranteed that the actual data align with the previously defined labels. Therefore,

[6] `https://dev.twitter.com/`
[7] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`
[8] `http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html`

creating a single classifier often is insufficient for achieving the actual classification goal. In practical applications it is thus necessary to build a cascade of bespoke classifiers as described in Wibberley et al. (2013), where every classifier in the cascade uses the output of the previous model as its input. For example, such a classifier cascade for a sentiment analysis task on Twitter data would work as follows: after obtaining a large number of tweets from the Twitter API via keyword search, the first classifier in the cascade is used to separate relevant tweets from irrelevant ones. The second classifier, by using the output of the first model as its input, filters tweets that express an opinion, from tweets that do not[9]. Finally, the actual sentiment analysis is performed with the third classifier on the remaining tweets. For such a scenario, where multiple models need to be learnt, and where unlabelled text is abundantly available, but high-quality hand annotated text is expensive to acquire, enhancing a Machine Learning classifier with the information from the unlabelled corpus is an absolute necessity.

Another requirement is that the resulting classifier models transition from sophisticated systems running on specialised hardware, to lightweight applications that can be trained within minutes and run on a single workstation. For such agile scenarios, existing supervised Machine Learning approaches are rendered infeasible as the large amount of labelled training data required, typically is not available. Another problem is the fact that such models exhibit poor performance when applied to domains different from the one they were trained on (Aue and Gamon, 2005); (Pang and Lee, 2008).

By combining a Multinomial Naïve Bayes classifier with the Expectation-Maximization algorithm, as the semi-supervised component, in an active learning setting, the aforementioned DUALIST fulfils most of the requirements that arise in practice. The primary goal now becomes to improve the classification performance of the system by experimenting with different semi-supervised learning algorithms and by testing techniques for enhancing the active learning process.

[9] At this stage it does not matter whether the expressed opinion is positive or negative.

4. Semi-Supervised Learning & Active Learning

Semi-supervised learning (SSL) leverages the idea of the availability of a large amount of unlabelled data together with a smaller amount of labelled data to train a classifier for a given task, which is absolutely crucial in practical applications as discussed in the previous section[10]. Active learning (AL) takes the idea a step further, as the learning algorithm itself participates in the labelling process, and is selecting which instances should be labelled next by a human annotator[11]. But leveraging the information contained in unlabelled data and applying AL techniques to optimise the training procedure, covers only part of the demands. In an application like DUALIST, the different algorithmic components should also be fast enough to allow for real-time human interaction in the active learning process.

The perhaps most commonly used SSL algorithm, in conjunction with a Multinomial Naïve Bayes classifier, is Expectation-Maximization (EM), which typically requires multiple passes over the unlabelled dataset to optimise a target function. The intuition behind EM is to start with an initial model (i.e. trained on the available labelled data) and then to repeatedly label all unlabelled data with the current model, and to retrain the model on all data in the following, until the algorithm converges. For an active learning process, such a method often is computationally too expensive, especially for large datasets. For this reason, Su et al. (2011) and Lucas and Downey (2013) developed novel SSL methods to more efficiently scale with the growing amounts of data. Both methods were developed in conjunction with a standard Multinomial Naïve Bayes classifier.

The Semi-supervised Frequency Estimate algorithm (SFE) by Su et al. (2011) scales the contribution of the features contained in the labelled data, by their respective prevalence in the unlabelled data, in the resulting classifier model. Thus, the magnitude of terms in the model, which frequently occur in the small set of labelled data, but only have a low number of occurrences in the unlabelled data (and hence overall), is scaled down. The reverse is achieved for words only rarely occurring in the training data, but very frequently occurring in the unlabelled data. The contribution to the model parameters of such features will be amplified. Hence, the most frequently occurring terms in the dataset as a whole, make the largest contribution to the classifier model. The SFE algorithm only requires a single pass over the unlabelled data per learning iteration.

[10] A good overview of SSL techniques can be found in Zhu (2005).
[11] An excellent resource of active learning techniques is Settles (2009).

In the Feature Marginals method, Lucas and Downey (2013) precompute a small set of marginal statistics over the unlabelled corpus, which are then imposed as constraints on an MNB learning model, trained on the labelled data. As with SFE, the idea is that the prevalence of a feature in the unlabelled data determines its contribution to the model parameters.

5. Methodology & Objectives

In the following, I specify the scope and objectives for my work, followed by a more technical overview of the current DUALIST implementation and an introduction to the key algorithms that form the technical basis of this project.

5.1. Scope & Objectives

My primary objectives for this project are to implement the Expectation-Maximization, Semi-supervised Frequency Estimate and Feature Marginals algorithms, and different combinations thereof, and to reproduce the published results in Lucas and Downey (2013) on the Reuters ApteMod corpus, to validate my implementations. The primary scope however, is the evaluation of the algorithms on Twitter datasets, created by CASM researchers for real-life analyses. I compare the performance of the implemented methods in terms of Accuracy and F1-Score. I am further investigating various ways to improve the classification performance of the algorithms by using uniform class priors, undersampling the majority class in case of imbalanced classes in the datasets, optimising the free parameters in EM, and using alternative uncertainty sampling methods in the active learning process. An extension to the primary objectives is to add the best performing algorithm (or combinations thereof) to the existing classification framework of the current DUALIST implementation used by CASM. At the time of this writing, first versions of the Feature Marginals algorithm and the Semi-supervised Frequency Estimate algorithm have been implemented in Java and are ready to be integrated into the existing framework.

5.2. The Current DUALIST Implementation

The classification engine currently being used in the customised version of DUALIST still follows the original implementation of Settles (2011). The application makes use of a Multinomial Naïve Bayes classifier, together with the Expectation-Maximization algorithm. However, Settles (2011) implemented EM in a non-standard way. Most significantly, the initial model is not created by training an MNB classifier on the labelled documents, but by only using the *terms* that were labelled by the user (a.k.a. the labelled features[12]).

[12]Recall from section 1 that labelling features is one of the most important components in DUALIST.

Originally, Settles (2011) incremented the frequency of a hand-labelled feature by a pseudo-count of 50, which means that the total number of occurrences of an annotated term is its real frequency + 50. The second difference is that the EM implementation does not iterate until convergence, but stops after 1 iteration. This is because Settles (2011) found that the major improvement in classification accuracy is achieved after the first iteration, and that subsequent iterations have a negligible effect. A second reason is running time: the algorithm is used in an active learning context and needs to allow for real-time interaction. In the active learning procedure, Settles (2011) used entropy based uncertainty sampling for instance querying, which means that the algorithm displays the top n documents that are closest to the classifiers' current decision boundary for labelling[13]. Individual features were queried by Information Gain, which identifies the most information salient terms in the vocabulary[14].

5.3. Multinomial Naïve Bayes classifier (MNB)

The Multinomial Naïve Bayes classifier is a generative, probabilistic learning model, where the probability of any given document d belonging to a given target class c is estimated as:

$$P(c \mid d) \propto P(c) \prod_{i=1}^{n_d} P(w_i \mid c) \tag{1}$$

where $P(w_i \mid c)$ is the conditional probability of word w_i occurring in a document d of class c, containing n_d words. The equation measures how much evidence each word w_i in document d contributes to belonging to the target class c. The goal is to find the most likely class for the given document, taking all the evidence, represented by the words occurring in the document, into account:

$$c = \underset{c \in \{+,-\}}{\operatorname{argmax}} \; P(c) \prod_{i=1}^{n_d} P(w_i \mid c) \tag{2}$$

Equation (2) represents a binary classification problem where "+" is the class of interest and "−" simply represents all other classes. When implementing an MNB classifier, multiplying

[13]Entropy based uncertainty sampling is a very popular concept for instance querying in active learning. The method has been introduced into the field of Machine Learning and NLP by Lewis and Gale (1994), where a detailed description of the method can be found as well.

[14]Settles (2011) uses the Information Gain measure in the way in which it is commonly applied in text classification tasks for the purpose of feature selection. See Sebastiani (2002) for an overview.

a large number of very small probabilities can lead to floating point underflow, therefore in practice, an MNB classifier is implemented by adding logs of probabilities:

$$c = \underset{c \in \{+,-\}}{\operatorname{argmax}} \left[\log P(c) + \sum_{i=1}^{n_d} \log P(w_i \mid c) \right] \tag{3}$$

To avoid 0 probabilities for terms that did not occur in the training data, a common method is to apply Laplace smoothing (a.k.a. add-1 smoothing) to each term in the training set, formally:

$$\underset{w \in \mathbb{W},\, c \in \{+,-\}}{P(w \mid c)} = \frac{N_{w,c} + 1}{N_c + \mid \mathbb{W} \mid} \tag{4}$$

where $N_{w,c}$ represents the number of occurrences of word w in class c, N_c represents the sum of all word occurrences in class c and $|\mathbb{W}|$ represents the number of words in the vocabulary (Manning et al., 2008); (McCallum and Nigam, 1998).

All Naïve Bayes models make strong assumptions about the data, for example the assumption that the order of words in every document is insignificant (bag-of-words approach), and that all words in a document are independent of one another (Manning et al., 2008). This rarely, if ever, is the case in real world datasets, however, the Naïve Bayes model is competitive with more sophisticated Machine Learning models despite these simplistic assumptions (Domingos and Pazzani, 1997).

5.4. Expectation-Maximization (EM)

Expectation-Maximization is a class of algorithms that iteratively optimise a target function in generative learning models such as Multinomial Naïve Bayes, in case of incomplete data, and is commonly used in a semi-supervised setting with it. The method was initially described by Dempster et al. (1977), but I will primarily refer to Nigam et al. (2000), who optimised EM for NLP tasks, in the context of this project. In essence, the algorithm works as follows: an initial model is devised by i.e. training an MNB classifier on the labelled data. Then, this model is used to classify the unlabelled data (E-step) with probabilistic labels, and finally, the model is retrained by using all data (M-step). The latter two steps, E-Step and M-Step, are repeated until convergence. NB: In active learning settings, the EM algorithm is not executed until convergence, but typically for 1 iteration only. The main reasons for that are running time, and the fact that the major improvements in classification performance are gained from the first iteration.

The algorithm makes strong assumptions about the data, i.e. that there are no multiple subtopics within one class. In case these assumptions are not satisfied, EM could potentially degrade classification performance. As these assumptions about the data rarely hold in practice Nigam et al. (2000) proposed two significant extensions to the basic EM algorithm. The first one introduces a weighting factor for the unlabelled data, having the effect of scaling down their contribution to the model parameters. The second extension concerns the modelling of each target class with multiple mixture components instead of a single component. But even with these modifications, classification performance may still degrade with increasing amounts of labelled data (Nigam et al., 2000).

5.4.1. EM Parameterisation across Publications

In the following I will briefly describe the different parameterisations that have been used for the Expecation-Maximization algorithm in Settles (2011), Su et al. (2011) and Lucas and Downey (2013). There are 3 free parameters in an EM model: the initial model, the number of iterations performed, and the weight assigned to probabilistically labelled instances. All of these parameters have a direct and significant influence on the performance of the algorithm.

As described above, Settles (2011) only used the user-labelled features to initialise the model, and then performed 1 iteration of EM. The unlabelled instances were assigned a static weight of 0.1, which means that all feature counts in the unlabelled documents were divided by 10.

Su et al. (2011) used an MNB classifier, trained on the labelled data as initial model, which represents the most common way of EM model initialisation for text classification tasks (Nigam et al., 2000). Su et al. (2011) further used the prediction probability of each unlabelled document, obtained from the previous EM iteration (or the initial model), as its respective weight. This way, the more certain the classification decision for a document, the larger its contribution to the model. Su et al. (2011) performed 10 iterations of EM.

Lucas and Downey (2013) also constructed their initial model by training an MNB classifier on the labelled data, followed by 15 iterations of the EM algorithm. Lucas and Downey (2013) weighted the unlabelled documents "at 1/5 the weight of a label[l]ed example", but did not explicitly mention the weight assigned to a labelled document (I am assuming it to be 1).

My implementation of EM is initialised by training an MNB classifier on the available labelled data followed by performing 1 iteration of EM. My reasons for iterating only once are consistent with those of Settles (2011), as during development I found the trade-off between improved classification accuracy and running time to be in favour of the latter. I conducted a number of experiments with different weightings for the unlabelled documents and found that using $\frac{|d_l|}{10*|d_u|}$, where $|d_l|$ represents the number of labelled documents in the collection and $|d_u|$ the number of unlabelled documents, exhibited the best performance (see section 7 for more information). NB: As the datasets I am using for the experiments in this report do not explicitly contain any information about user-labelled features, I am unable to initialise the EM model the way that Settles (2011) did.

5.5. Semi-supervised Frequency Estimate (SFE)

Semi-supervised Frequency Estimate and Expectation-Maximization both aim to optimise log-likelihood as their objective function[15]. But where EM aims to maximise marginal log-likelihood, which measures how well the classifier fits the joint distribution of the words in the given documents, SFE aims to maximise conditional log-likelihood, which measures how well the classifier fits the probability of the class given the words. Formally, EM updates the class-conditional probabilities with each iteration as:

$$P(w_i \mid c) = \frac{\prod_{i=1}^{|\mathbb{W}|} P(w_i \mid c)' P(w_i)_u \alpha}{Z} \tag{5}$$

where $P(w_i \mid c)$ are the class-conditional probabilities for the next iteration, $P(w_i \mid c)'$ are the class-conditional probabilities of the EM model of the current iteration, $P(w_i)_u$ is the probability of word w_i in the unlabelled data, $|\mathbb{W}|$ is the number of terms in the vocabulary, w_i is the ith word in the vocabulary, α is a weighting term and Z is shorthand for a normalisation constant over all probabilities. SFE on the other hand, preserves the class-conditional probabilities obtained from the labelled data and combines these estimates with the probability of a word in the unlabelled data:

$$P(w_i \mid c) = \frac{\prod_{i=1}^{|\mathbb{W}|} P(w_i \mid c)_l P(w_i)_u \alpha}{Z} \tag{6}$$

[15] informally, log-likelihood is defined as $LL = MLL + CLL$, where MLL stands for marginal log-likelihood and CLL stands for conditional log-likelihood. For a more information see Su et al. (2011) and Equation (3)

where $P(w_i \mid c)$, are the scaled class-conditional probabilities for each word w_i in class c, $P(w_i \mid c)_l$ are the class-conditional probabilities obtained from training a Naïve Bayes model on the labelled data, $P(w_i)_u$ is the probability of word w_i in the unlabelled data, $|\mathbb{W}|$ is the number of terms in the vocabulary, α is a weighting term and Z is shorthand for a normalisation constant over all probabilities (Su et al., 2011).

The difference between the respective update rules in the Semi-supervised Frequency Estimate algorithm and the Expectation-Maximization algorithm is in the details. Where EM uses $P(w_i \mid c)'$ in its update rule, denoting the class-conditional probabilities obtained from the previous iteration of EM, SFE uses $P(w_i \mid c)_l$, which denotes the class-conditional probabilities obtained from training an MNB classifier on the labelled data. A further, and very significant difference is that SFE only needs 1 iteration per learning iteration, whereas EM typically needs multiple iterations for convergence.

This means that SFE scales the class-conditional probabilities, obtained from training a Multinomial Naïve Bayes classifier on the labelled data, $P(w_i \mid c)_l$, by $P(w_i)_u$, the corresponding probability of a word in the unlabelled data. The underlying assumption is that the class-conditional probabilities from the labelled data *can* and *should* be trusted, and should only be *scaled* accordingly. As the contribution of each feature to the model parameters is ranked by probability, the algorithm can also be seen as a feature smoothing method, as which I have used it when combining the algorithm with other methods.

The major advantage of the Semi-supervised Frequency Estimate algorithm over EM is that SFE is not an iterative method, and hence only requires a single pass over the unlabelled data per learning iteration. This means that the number of iterations is no longer a free parameter that needs to be adjusted to the task or goal(s), and specifically, that there is no trade-off between classification accuracy and running time as in EM.

A weakness of the algorithm is, as Lucas and Downey (2013) argue, that as $P(w)_u$ of any word in the unlabelled data becomes proportionally larger than in the labelled data, the ratio $\frac{P(w|+)}{P(w|-)}$ approaches 1, which can be arbitrarily inaccurate. The weakness stems from the underlying assumption of the labelled data being a representative sample of the overall population. In scenarios such as faced by CASM researchers however, such a situation will only rarely be the case.

5.6. Feature Marginals (FM)

The Feature Marginals algorithm by Lucas and Downey (2013) precomputes a small set of statistics over the unlabelled data in advance, and applies these statistics as constraints on the MNB model to improve the estimates of the class-conditional probabilities. The model is different from EM and SFE insofar that the classification process is no longer primarily governed by the Multinomial Naïve Bayes model, but by the set of statistics acting as constraints on the model.

The definition of the FM constraint in Equation (7) is given in terms of a two-class problem (positive vs. negative), where the positive class is denoted by "+" and indicates the class of interest and "−" represents all other classes:

$$P(w)_u = P(w \mid +)\, P_t(+) + P(w \mid -)\, P_t(-) \tag{7}$$

where $P(w)_u$ denotes the probability that a randomly drawn token (= a word occurrence) from the unlabelled data is the word w, $P_t(+)$ and $P_t(-)$ denote the probability that a randomly drawn token from the labelled data is from the positive or negative class respectively, and $P(w \mid +)$ and $P(w \mid -)$ represent the class-conditional probabilities obtained from training an MNB classifier on the labelled training data.

Equation (7) can be rewritten to express either of the class-conditional probabilities, and the right-hand-side of the resulting expression, shown in Equation (8), can then be substituted into Equation (3):

$$P(w \mid +) = \frac{P(w)_u - P(w \mid -)P_t(-)}{P_t(+)} \tag{8}$$

which after some workings results in Equation (9) below. This equation can then be solved for the remaining class-conditional probability. The full derivation can be found in Lucas and Downey (2013).

$$0 = \frac{N_w^+}{P(w \mid +)} + \frac{N_{\neg w}^+}{P(w \mid +) - 1} + \frac{LN_w^-}{LP(w \mid +) - K} + \frac{LN_{\neg w}^-}{LP(w \mid +) - K + 1} \tag{9}$$

where

$$K = \frac{P(w)_u}{P_t(-)};\;\; L = \frac{P_t(+)}{P_t(-)} \tag{10}$$

N_w^+ is the number of occurrences of word w in class "+" and $N_{\neg w}^+$ is the number of words other than w in class "+". N_w^- and $N_{\neg w}^-$ are defined in equal terms for class "−". The constraint becomes a cubic equation when solved for the remaining class-conditional probability, with usually only one solution in $\mathbb{R}$. This root represents the optimal value for the given class-conditional probability, subject to the constraint of being within the target interval $\left[0, \frac{K}{L}\right]$. In case no optimal value could be obtained, the algorithm falls back on the standard Multinomial Naïve Bayes model. Optimal solutions for class-conditional probabilities are only found ∼30-40% of the time, but the improvement in classification accuracy is remarkable.

The intuition behind this method is that the class-conditional probabilities, obtained from training an MNB classifier on the labelled data, $P(w \mid c)$, typically cannot be reliably estimated with only a small amount of annotated training data, and therefore *cannot* and *should not* be trusted (NB: the contrary of SFE). On the other hand, Lucas and Downey (2013) claim that $P_t(+)$ and $P_t(-)$ can be relatively accurately estimated, even from only a small set of labelled data. Thus, the class-conditional probabilities of the MNB classifier model are governed by the prevalence of the respective features in the unlabelled data and the distribution of the tokens among the target classes.

From an implementation point of view, the MNB learning algorithm does not change significantly. After precomputing $P(w)_u$ over the unlabelled data and $P_t(+)$ and $P_t(-)$ over the labelled data, the MNB model is trained on the annotated training data in the usual way. Afterwards, the optimal values for $P(w \mid +)$ and $P(w \mid -)$ are calculated by the procedure described above. The final step is then *substituting* the existing class-conditional probabilities with the new values and re-normalising the probabilities.

The method generally outperformed Semi-supervised Frequency Estimate and Expectation-Maximization, but comes with a significant caveat: it is significantly slower than EM and SFE, which can be problematic in an active learning scenario. The reason for it being less computationally efficient is that the marginal statistics cannot be calculated once in advance as in a semi-supervised setting, where the amounts of labelled and unlabelled data remain fixed, but need to be re-calculated for each active learning iteration. This is because in AL the pool of annotated and unannotated documents changes with each learning iteration, as unlabelled examples get labelled and moved to the pool of annotated documents, which causes the relevant quantities $P(w)_u$, $P_t(+)$ and $P_t(-)$ to change. The component mainly responsible for the computational cost of the algorithm is the method for finding

the roots of Equation (9)[16]. Thus, the improvement in classification performance gained by the Feature Marginals method comes at the expense of increased running time.

5.7. Combining Algorithms

I implemented every possible combination of all of the 3 aforementioned algorithms, resulting in 15 different methods (not counting the standard MNB implementation, which served as the baseline). The idea behind combining the algorithms simply is the attempt to combine their individual strengths. For example, instead of falling back on the standard MNB model in case no optimal value for a feature can be obtained with the Feature Marginals method, a fallback on the SFE algorithm, which would act as a smoothing factor in that case, can ideally improve the classifier's performance. By executing 1 iteration of Expectation-Maximization afterwards, classification accuracy can potentially be improved even further. Appendix B lists all algorithm combinations, together with their respective performances in a pre-evaluation experiment. The best 3 algorithm combinations from the pre-evaluation experiment, along with a standard MNB classifier as baseline and MNB classifiers combined with EM, FM and SFE respectively, were used for the final set of experiments. The algorithm combinations are an MNB classifier with EM and FM (in that order, so after training an MNB classifier, EM was performed, followed by the Feature Marginals algorithm). Further, the combination of MNB with EM, FM and SFE, and the combination of MNB with SFE, EM and FM.

[16]I am using the Newton-Raphson method which was used by Lucas and Downey (2013) as well.

6. Datasets

The target domain of DUALIST is Twitter, therefore the implemented algorithms were primarily evaluated on Twitter datasets, that were collected and annotated by CASM researchers during real-world analyses. I used the corpora in their respective "as-is" status to guarantee that my results give a realistic picture of the expected performance of the algorithms in practice. The Reuters ApteMod dataset, which is publicly available at `http://www.daviddlewis.com/resources/testcollections/reuters21578/`, was included in the evaluation as a validation against the published results in Lucas and Downey (2013).

The average number of labelled documents among the Twitter datasets is ∼560, compared to ∼130 000 unlabelled tweets per dataset on average. In contrast, the Reuters ApteMod dataset is fully annotated and contains 10 794 documents.

6.1. The "Mark Duggan" Datasets

These datasets were collected during the trial and verdict of the killing of Mark Duggan by a policeman in Tottenham in August 2013[17]. I evaluated 3 datasets concerning this case, 1 of them representing a relevancy classifier ("Duggan"), with the purpose of filtering tweets about Mark Duggan from other tweets. The other 2 datasets represent public opinion with regards to the case in general ("Duggan-Main") and the verdict in particular ("Duggan-Verdict"), respectively. Specifics are listed below:

[17] see `http://www.bbc.com/news/uk-24099368` for an overview of the case

Property	**Value**
Corpus Name	Duggan
Number of labelled documents	364
Number of unlabelled documents	198 753
Target Labels	”mark” ; ”other”
Label Distribution	mark=0.2004; other=0.7996
Number or words	76 553
Number of tokens	3 477 766
Number of tokens in labelled documents	6 127

Table 1: The Duggan Dataset

Property	**Value**
Corpus Name	Duggan-Main
Number of labelled documents	871
Number of unlabelled documents	85 874
Target Labels	”justice”; ”nojustice”; ”other”
Label Distribution	justice=0.2158; nojustice=0.4214; other=0.3628
Number or words	25 749
Number of tokens	1 561 631
Number of tokens in labelled documents	15 028

Table 2: The Duggan-Main Dataset

Property	**Value**
Corpus Name	Duggan-Verdict
Number of labelled documents	1 485
Number of unlabelled documents	51 954
Target Labels	”justice”; ”gonnariot”; ”nojustice”; ”watching”
Label Distribution	justice=0.2269; gonnariot=0.0815; nojustice=0.4135; watching=0.2781
Number or words	15 435
Number of tokens	995 006
Number of tokens in labelled documents	25 718

Table 3: The Duggan-Verdict Dataset

6.2. The ”David Cameron” and ”Immigration Bill” Datasets

I evaluated 5 datasets concerned with discussions about immigration in the UK and the immigration bill, starting in December 2013/January 2014. The discourse on Twitter primarily involved Bulgarian and Romanian citizens gaining the same rights to work in the UK as citizens from any other EU countries[18]. There are 3 datasets that capture the public opinion on the immigration discussion (”Immigration-Bill”, ”Immigration-UK” and ”Immigration-Extended”), and 2 datasets concerning the public opinion towards the UK prime minister, David Cameron, in the context of this discourse (”Cameron” and ”Cameron-2”). Specifics are as follows:

Property	**Value**
Corpus Name	Immigration-Bill
Number of labelled documents	397
Number of unlabelled documents	5 724
Target Labels	”relevant”; ”irrelevant”
Label Distribution	relevant=0.9798; irrelevant=0.0202
Number or words	6 709
Number of tokens	104 053
Number of tokens in labelled documents	6 874

Table 4: The Immigration-Bill Dataset

Property	**Value**
Corpus Name	Immigration-UK
Number of labelled documents	247
Number of unlabelled documents	46 802
Target Labels	”relevant”; ”irrelevant”
Label Distribution	relevant=0.5547; irrelevant=0.4453
Number or words	35 562
Number of tokens	780 134
Number of tokens in labelled documents	4 025

Table 5: The Immigration-UK Dataset

[18] see i.e. `http://www.bbc.com/news/uk-politics-21523319` for an overview of the immigration discussion and `http://www.bbc.com/news/uk-politics-24626767` for a summary about the immigration bill

Property	**Value**
Corpus Name	Immigration-Extended
Number of labelled documents	210
Number of unlabelled documents	425 072
Target Labels	”relevant”; ”irrelevant”
Label Distribution	relevant=0.2238; irrelevant=0.7762
Number or words	156 808
Number of tokens	2 677 793
Number of tokens in labelled documents	2 171

Table 6: The Immigration-Extended Dataset

Property	**Value**
Corpus Name	Cameron
Number of labelled documents	558
Number of unlabelled documents	13 666
Target Labels	”report”; ”comment”; ”irrel”
Label Distribution	report=0.0556; comment=0.4337; irrel=0.5107
Number or words	15 621
Number of tokens	247 567
Number of tokens in labelled documents	9 354

Table 7: The Cameron Dataset

Property	**Value**
Corpus Name	Cameron-2
Number of labelled documents	640
Number of unlabelled documents	13 586
Target Labels	”relevant”; ”irrelevant”
Label Distribution	relevant=0.5938; irrelevant=0.4062
Number or words	15 614
Number of tokens	247 590
Number of tokens in labelled documents	11 125

Table 8: The Cameron-2 Dataset

6.3. The ”EU” and ”Euro” Sentiment Datasets

These datasets contain opinion-based tweets about the European Union and the Euro. Overall, there are 3 EU datasets (”EU-Relevance”, ”EU-Attitude” and ”EU-Sentiment”) and 3 Euro datasets (”Euro-Relevance”, ”Euro-Attitude” and ”Euro-Sentiment”), representing the complete classifier cascades that have been applied in the respective analyses. Details are as listed below:

Property	**Value**
Corpus Name	EU-Relevance
Number of labelled documents	996
Number of unlabelled documents	138 633
Target Labels	”relevant”; ”irrelevant”
Label Distribution	relevant=0.6727; irrelevant=0.3273
Number or words	112 971
Number of tokens	2 464 692
Number of tokens in labelled documents	16 758

Table 9: The EU-Relevance Dataset

Property	**Value**
Corpus Name	EU-Attittude
Number of labelled documents	260
Number of unlabelled documents	90 189
Target Labels	”attitudinal”; ”non-attitudinal”
Label Distribution	attitudinal=0.5154; non-attitudinal=0.4846
Number or words	71 531
Number of tokens	1 613 316
Number of tokens in labelled documents	4 882

Table 10: The EU-Attitude Dataset

Property	Value
Corpus Name	EU-Sentiment
Number of labelled documents	260
Number of unlabelled documents	38 059
Target Labels	”positive”; ”negative”
Label Distribution	positive=0.0115; negative=0.9885
Number or words	35 693
Number of tokens	698 694
Number of tokens in labelled documents	4 683

Table 11: The EU-Sentiment Dataset

Property	Value
Corpus Name	Euro-Relevance
Number of labelled documents	996
Number of unlabelled documents	106 259
Target Labels	”relevant”; ”irrelevant”
Label Distribution	relevant=0.1777; irrelevant=0.8223
Number or words	85 334
Number of tokens	1 837 841
Number of tokens in labelled documents	15 361

Table 12: The Euro-Relevance Dataset

Property	Value
Corpus Name	Euro-Attitude
Number of labelled documents	309
Number of unlabelled documents	72 791
Target Labels	”attitudinal”; ”non-attitudinal”
Label Distribution	attitudinal=0.1456; non-attitudinal=0.8544
Number or words	55 166
Number of tokens	1 280 693
Number of tokens in labelled documents	4 923

Table 13: The Euro-Attitude Dataset

Property	Value
Corpus Name	Euro-Sentiment
Number of labelled documents	320
Number of unlabelled documents	27 510
Target Labels	”positive”; ”negative”
Label Distribution	positive=0.0063; negative=0.9937
Number or words	25 688
Number of tokens	500 135
Number of tokens in labelled documents	5 829

Table 14: The Euro-Sentiment Dataset

6.4. The ”The Voice UK 2013” Dataset

This dataset was collected during the final episode of ”The Voice UK 2013”, which is a talent casting show for singers[19]. The corpus contains commentary on the artists' performances as well as on other aspects of the show. NB: I annotated all of the available 500 labelled tweets, but was not involved in collecting the dataset otherwise. Details are as follows:

Property	Value
Corpus Name	The Voice UK 2013
Number of labelled documents	500
Number of unlabelled documents	109 862
Target Labels	”Positive”; ”Negative”; ”Neutral”
Label Distribution	Positive=0.376; Negative=0.198; Neutral=0.426
Number or words	29 496
Number of tokens	1 323 602
Number of tokens in labelled documents	5 829

Table 15: The Voice UK 2013 Dataset

[19]see `http://www.bbc.co.uk/programmes/b01nnfdd`

6.5. The ”Reuters ApteMod” Dataset

This dataset represents a benchmark corpus in text classification and contains 10 794 newswire articles belonging to 118 classes. Details are listed below:

Property	**Value**
Corpus Name	Reuters ApteMod
Number of labelled documents	10 794
Number of categories	118
Top 5 categories	”earn”; ”acq”; ”money-fx”; ”grain”; ”crude”
Label Distribution of top 5 categories	earn=0.3672; acq=0.2195; money-fx=0.0664; grain=0.0539; crude=0.0535
Number or words	26 291
Number of tokens	1 419 409

Table 16: The Reuters ApteMod Dataset

7. Results

7.1. Experiment Setup

For the main set of experiments I compared a standard Multinomial Naïve Bayes classifier as baseline, to MNB classifiers combined with the Feature Marginals algorithm, the Semi-supervised Frequency Estimate algorithm, the Expectation-Maximization algorithm, and MNB classifiers in conjunction with a combination of these algorithms: Expectation Maximization combined with the Feature Marginals algorithm (in that order, first EM, then FM), the combination of Expecation-Maximization, Feature Marginals and Semi-supervised Frequency Estimate and the combination of Semi-supervised Frequency Estimate, Expectation Maximization and Feature Marginals (NB: the result of a pre-evaluation to justify the selection of these combinations is presented in Appendix B). No special preprocessing was applied, neither stopwords nor punctuation marks were removed.

Specifics of the dataset/target label combinations are listed in Table 17 below:

Dataset	**Target Label (Proportion)**
Cameron	”report” (0.0556)
Cameron-2	”relevant” (0.5938)
Duggan	”mark” (0.2004)
Duggan-Main	”nojustice” (0.4214)
Duggan-Verdict	”justice” (0.2269)
Immigration-Bill	”relevant” (0.9798)
Immigration-Extended	”relevant” (0.2238)
Immigration-UK	”relevant” (0.5547)
The Voice UK 2013	”Positive” (0.376)

Table 17: Twitter Datasets - Dataset/Target Label Combination

For all datasets and classifier/SSL-algorithm combinations, the amount of data used for the experiments with the Twitter corpora was varied between 25, 50, 100, 300 and 500 for the labelled documents and 1 000, 5 000, 10 000, 50 000 and 100 000 for the unlabelled documents. In case a corpus contained less data, simply the maximum available was used. The sizes of the labelled and unlabelled sets for the experiments with the Reuters ApteMod corpus were varied between 25, 50, 100, 200, 300 and 500 for the labelled data and between 500, 1 000, 2 000, 3 000 and 5 000 for the unlabelled data. Each combination of labelled

and unlabelled dataset sizes was tested with every algorithm. Labelled and unlabelled documents were chosen at random for each experiment.

The reported results represent the average over 5 test runs and applying 10 fold cross-validation, unless stated otherwise. For the following comparison I report the performance in Accuracy and F1-Score. The corresponding plots for Precision and Recall, along with the full set of plots for all datasets, is available online, in the bitbucket repository of this project: `https://bitbucket.org/tttthomasssss/finalyearproject`[20]. I calculated the F1-Scores according to Forman and Scholz (2010), who outline that when using cross-validation, one should not calculate the arithmetic mean of the individual F1-Scores, obtained from each fold, but should tally up the numbers of true positives, false positives and false negatives for each fold, and then compute the F1-Score only once by using these counts. The reason for that methodology is, that Forman and Scholz (2010) found that computing the arithmetic mean of all individual F1-Scores during cross-validation, overestimates the actual performance of the classifier, hence not giving a faithful representation of the model's performance.

Tables 18 and 19 give an performance overview of all methods on the dataset/target label combinations in Table 17. The results represent the performance in Accuracy 18 and F1-Score 19 for 500 labelled tweets and 100 000 unlabelled tweets. Figures in boldface indicate the best performing method for the given dataset/target label combination.

Dataset/Target	**MNB**	**FM**	**EM**	**SFE**	**EM-FM**	**EM-FM-SFE**	**SFE-EM-FM**
Cam(report)	96	**0.98**	0.96	0.95	0.97	0.92	0.97
Cam2(relevant)	0.74	**0.8**	0.77	0.57	0.79	0.53	0.79
Dug(mark)	0.81	**0.88**	0.85	0.79	**0.88**	0.75	**0.88**
Dug-Main(nojustice)	0.72	**0.74**	0.68	0.55	0.73	0.53	0.73
Dug-Ver(justice)	0.78	**0.82**	0.78	0.76	0.78	0.69	0.78
Imm-Bill(relevant)	**0.98**	0.97	**0.98**	**0.98**	0.89	**0.98**	0.89
Imm-Ext(relevant)	0.84	**0.86**	0.78	0.72	0.8	0.67	0.8
Imm-UK(relevant)	0.7	**0.78**	0.75	0.54	0.77	0.53	0.77
Voice(Positive)	0.73	**0.76**	0.71	0.59	0.75	0.56	0.75

Table 18: Twitter Datasets - Accuracy Overview, 500 labelled tweets, 100 000 unlabelled tweets

[20]Due to the large number of plots generated (40 per dataset and target label, about 1200 overall), I refrained from adding the plots in the Appendix for space reasons. As it is a private repository, please contact the author at thk22@sussex.ac.uk to gain access to the plots.

Dataset/Target	MNB	FM	EM	SFE	EM-FM	EM-FM-SFE	SFE-EM-FM
Cam(report)	0.49	**0.8**	0.49	0.11	0.71	0.69	0.71
Cam2(relevant)	0.82	**0.83**	**0.83**	0.8	0.82	**0.83**	0.82
Dug(mark)	0.9	**0.93**	0.92	0.9	**0.93**	**0.93**	**0.93**
Dug-Main(nojustice)	0.57	**0.67**	0.41	0.52	**0.67**	0.64	**0.67**
Dug-Ver(justice)	0.13	**0.59**	0.13	0.1	0.57	0.55	0.57
Imm-Bill(relevant)	**0.99**	0.98	**0.99**	**0.99**	0.94	**0.99**	0.94
Imm-Ext(relevant)	0.51	**0.67**	0.4	0.5	0.63	**0.67**	0.63
Imm-UK(relevant)	0.79	**0.82**	0.78	0.78	0.78	**0.82**	0.78
Voice(Positive)	0.47	0.64	0.4	0.45	**0.66**	0.62	**0.66**

Table 19: Twitter Datasets - F1-Score Overview, 500 labelled tweets, 100 000 unlabelled tweets

7.2. Core Algorithm Experiments

7.2.1. Reuters Dataset

I used the Reuters ApteMod corpus as validation of my experiment outcomes against the results published in Lucas and Downey (2013). As Figure 2 shows, my results are almost identical with those of Lucas and Downey (2013), with the exception of the SFE algorithm. The reasons very likely are that Lucas and Downey (2013) used all available unlabelled data, whereas I varied the sizes of the unlabelled dataset across my experiments, and also because SFE requires a large amount of unlabelled documents to unleash its full potential as Su et al. (2011) show.

Over all configurations for the Reuters ApteMod corpus, the Feature Marginals algorithm or the combinations of SFE, EM and FM, or EM, FM and SFE, were the superior methods. As Figure 3 with target category = "earn" and a class skew of 37:63 shows, the FM algorithm and the combination of SFE, EM and FM outperformed the standard MNB baseline by ∼5-8% in terms of F1-Score. The strengths of the Feature Marginals algorithm are highlighted when the class imbalance is increased, where for a class skew of 22:78 with the target category "acq", the difference in F1-Score between the MNB baseline and the FM method becomes ∼35% for 300 labelled instances and increases to up to 60-70% for 500 labelled documents, as Figure 4 shows. Figure 4 is a more drastic example of the difference in performance and stability of the Feature Marginals algorithm in comparison to the other methods.

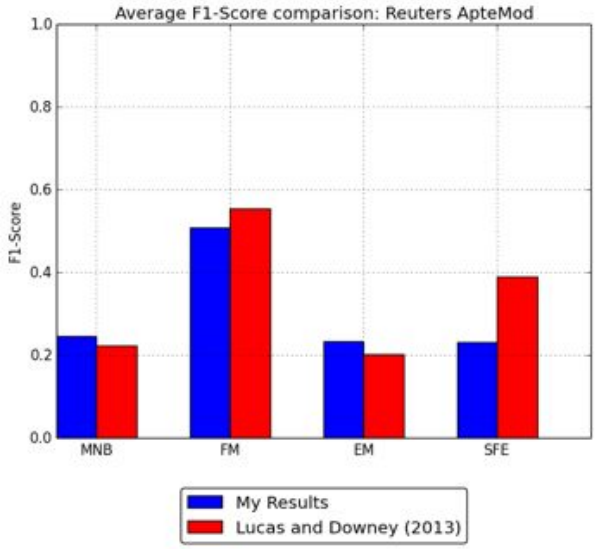


Figure 2: F1-Score: Validation of results, 100 labelled documents

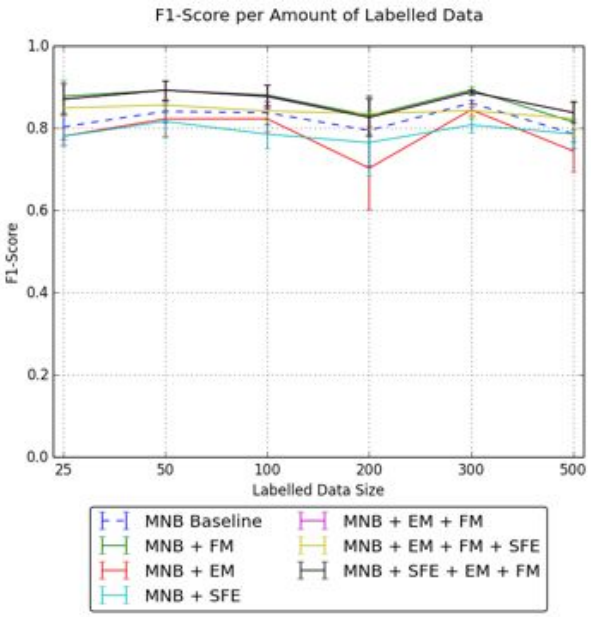


Figure 3: F1-Score: Reuters ApteMod, target category = ”earn”, 1 000 unlabelled tweets

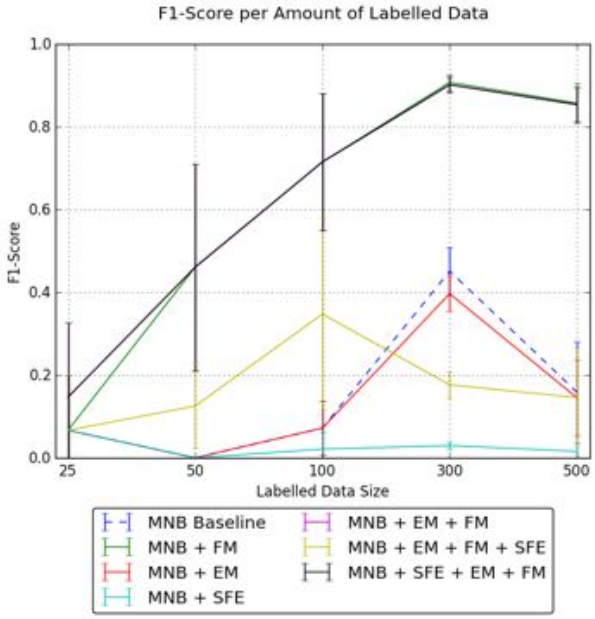


Figure 4: F1-Score: Reuters ApteMod, target category = ”acq”, 3 000 unlabelled tweets

7.2.2. Twitter Datasets

Overall, the combination of a standard Multinomial Naïve Bayes classifier with the Feature Marginals algorithm was the superior method, generally outperforming all other methods in terms of Accuracy and F1-Score. In some cases the combination MNB with Expectation-Maximization and Feature Marginals, or the combination of MNB with Semi-supervised Frequency Estimate, Expecation-Maximization and Feature Marginals produced better results. The performance of the combinations MNB with EM and FM, and MNB with SFE, EM and FM was almost always identical, suggesting that the contribution of the Semi-supervised Frequency Estimate algorithm is overpowered by EM and FM respectively. Figures 5 - 8 show the average performance across all different configurations of labelled and unlabelled data sizes, so the absolute numbers are less meaningful than the difference in performance between the methods. An MNB classifier in combination with the Feature Marginals method could improve Accuracy from 0.78% (Figure 5) on the The Voice UK 2013 dataset (target label = "Positive"), to more than 27% (Figure 6) on the Duggan-Verdict dataset (target label = "justice"), when compared with the standard MNB classifier baseline. The performance improvements of the Feature Marginals algorithm in terms of F1-Score range from 0.48% (Figure 7) on the Duggan-Verdict dataset (target label = "justice") to more than 13% (Figure 8) on the The Voice UK 2013 dataset (target label = "Positive"), in comparison to the baseline.

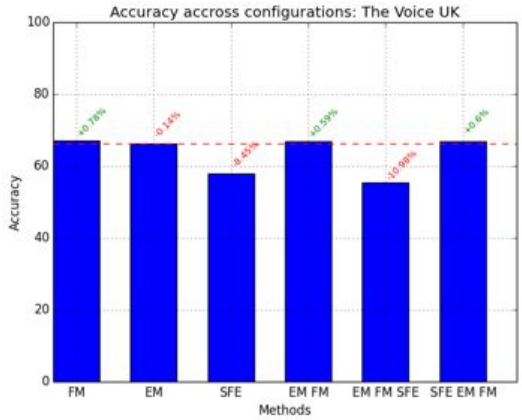


Figure 5: Average Accuracy: The Voice UK 2013

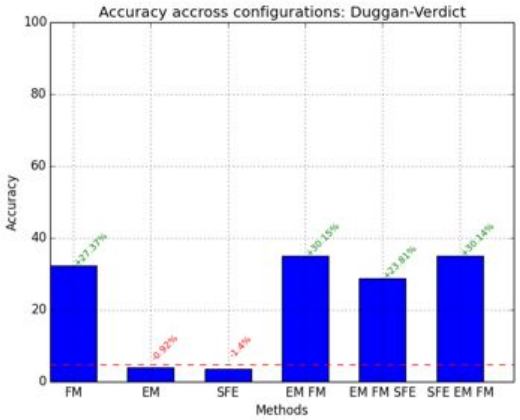


Figure 6: Average Accuracy: Duggan-Verdict

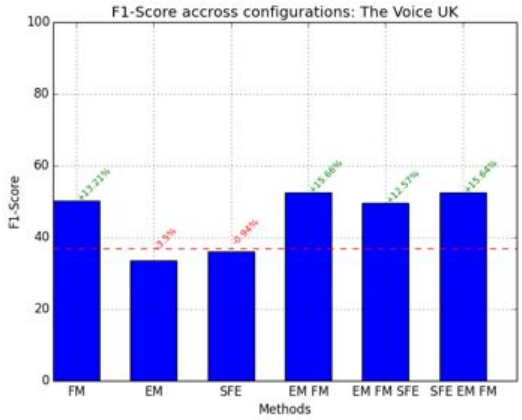


Figure 7: Average F1-Score: The Voice UK 2013

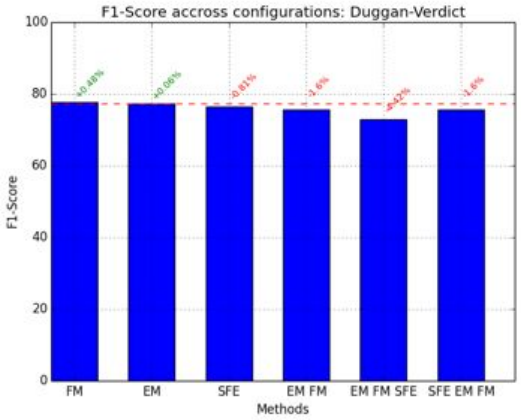


Figure 8: Average F1-Score: Duggan-Verdict

For some of the Twitter datasets the performance improvements gained by the FM algorithm were only marginal, however the improvement is very consistent, and more importantly, the Feature Marginals method never caused the classification performance to decrease. This was not the case with the Expectation-Maximization and Semi-supervised Frequency Estimate algorithms respectively, which regularly underperformed the standard MNB baseline in terms of Accuracy and F1-Score in my experiments.

Figure 9 presents the Accuracy over the number of labelled data on the Immigration-UK dataset, with 10 000 unlabelled tweets and ”relevant” as the target label. The Feature Marginals algorithm achieved the best results, outperforming the baseline by 7-8%. Interestingly, the performance of the Semi-supervised Frequency Estimate algorithm and the combination of EM, FM and SFE, was getting worse with increasing amounts of labelled data. Even increasing the amounts of unlabelled data to 50 000, which means that all of the unlabelled data for this dataset were used, in combination with 500 labelled tweets, did not improve the performance of the SFE algorithm, as Figure 10 shows. The Immigration-UK dataset represents one of the more drastic examples where the Semi-supervised Frequency Estimate algorithm underperformed the standard MNB classifier baseline, with the difference to the baseline being ∼10% in this case.

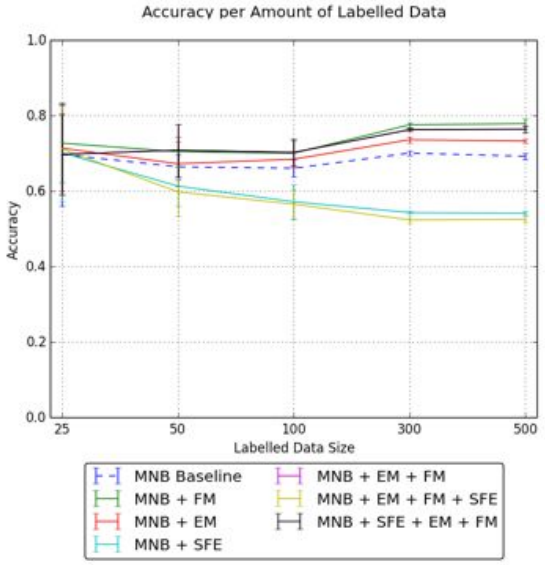


Figure 9: Accuracy: Immigration-UK, target label = ”relevant”, 10 000 unlabelled tweets

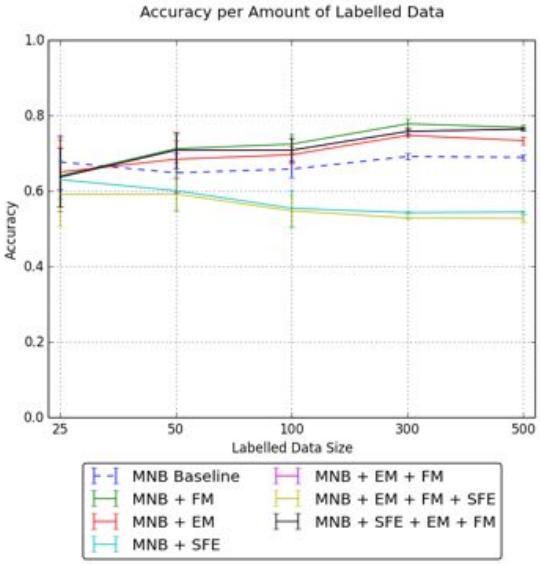


Figure 10: Accuracy: Immigration-UK, target label = ”relevant”, 50 000 unlabelled tweets

The Feature Marginals algorithm outperformed the other methods also in terms of F1-Score, improving the performance of a standard MNB classifier by more than 15% as Figure 11, on the The Voice UK 2013 dataset for 50 000 unlabelled tweets and the target label ”Positive”, shows. Figure 12 presents the F1-Score on the Duggan-Main dataset for 1 000 unlabelled tweets and the target label ”nojustice”. Again the Feature Marginals method achieved the best results, improving the performance of a Multinomial Naïve Bayes classifier by ∼15%.

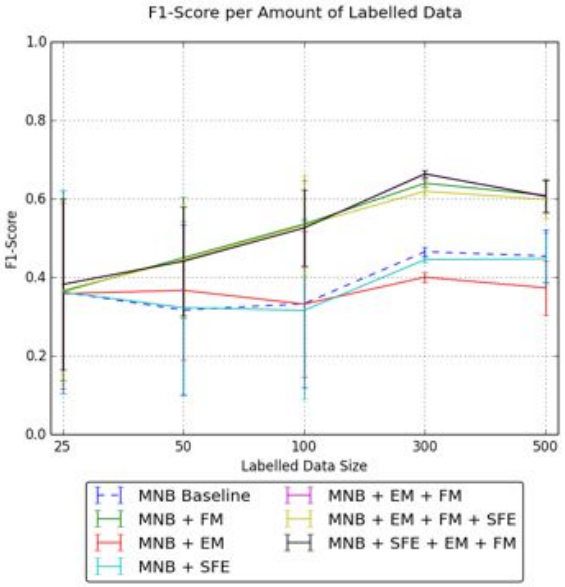


Figure 11: F1-Score: The Voice UK 2013, target label = ”Positive”, 50 000 unlabelled tweets

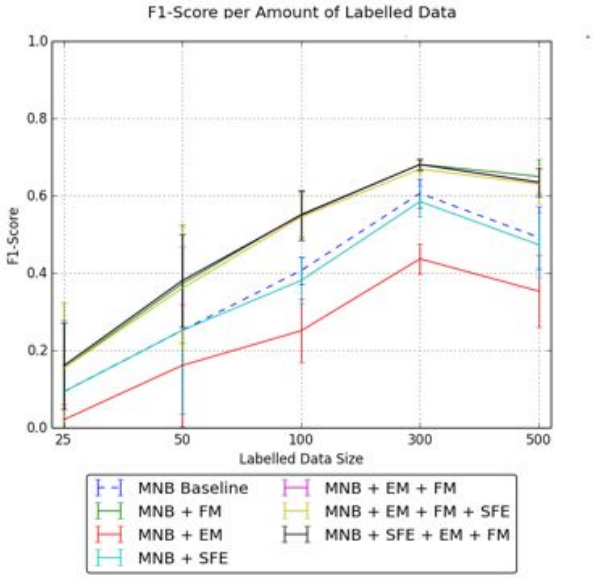


Figure 12: F1-Score: Duggan-Main, target label = ”nojustice”, 1 000 unlabelled tweets

The performance of the Feature Marginals algorithm was very consistent among configurations, datasets and performance measures. As already mentioned for the Reuters ApteMod dataset, the FM algorithm performed especially well in the case of the target labels being imbalanced. Figures 13 and 14 show Accuracy and F1-Score plots for the Cameron-2 dataset with a class skew of 6:94 for the target class "report". While the improvements in Accuracy are relatively modest with 3-5% (Figure 13), the improvement in F1-Score is vast with ∼55-60% (Figure 14). As can be seen on the size of the error bars in Figure 14, the performance of the Feature Marginals algorithm also is significantly stabler in comparison to the other methods.

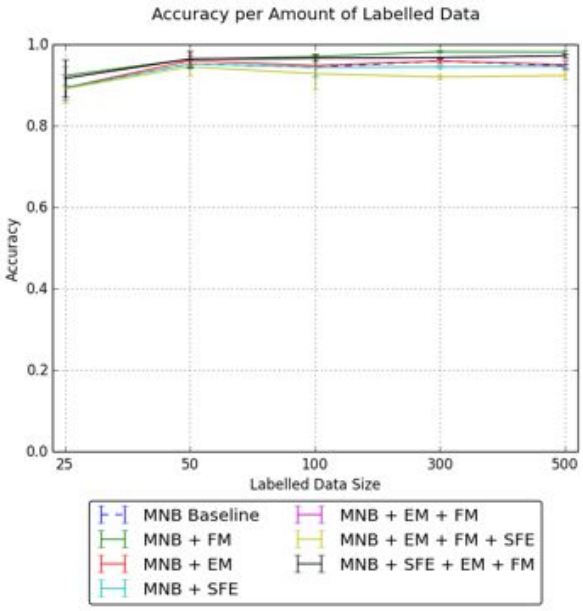


Figure 13: Accuracy: Cameron, target label = "report", 5 000 unlabelled tweets

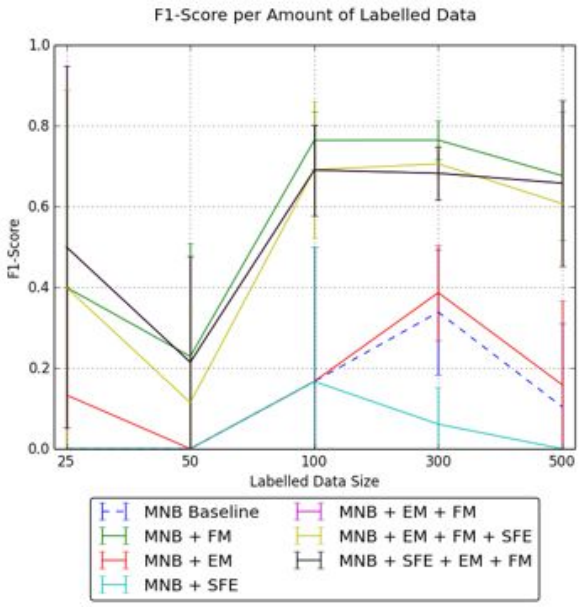


Figure 14: F1-Score: Cameron, target label = "report", 10 000 unlabelled tweets

Figure 15, representing the Immigration-Extended dataset for the target label ”relevant”, further shows the consistency of the improvement that the Feature Marginals causes. While all other methods *underperformed* the MNB baseline in terms of Accuracy for 50 000 labelled documents, the FM algorithm *outperforms* the baseline by 2-3%. Figure 16 shows the corresponding F1-Score plot, highlighting the superior performance of the Feature Marginals algorithm, and this time, the combination of EM, FM and SFE as well, that both outperform the MNB baseline by ∼20%.

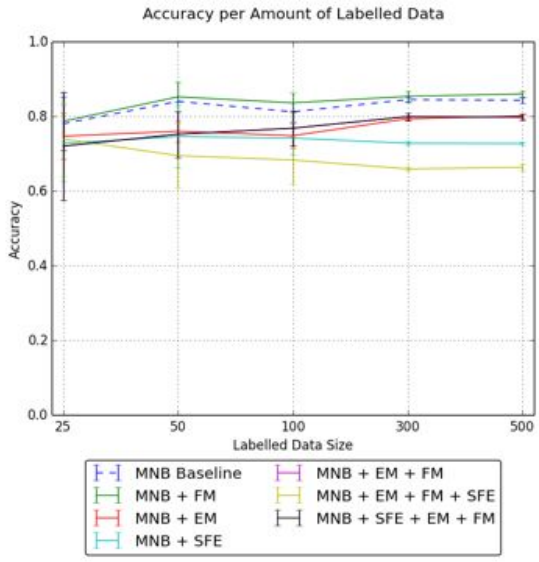


Figure 15: Accuracy: Immigration-Extended, target label = ”relevant”, 50 000 unlabelled tweets

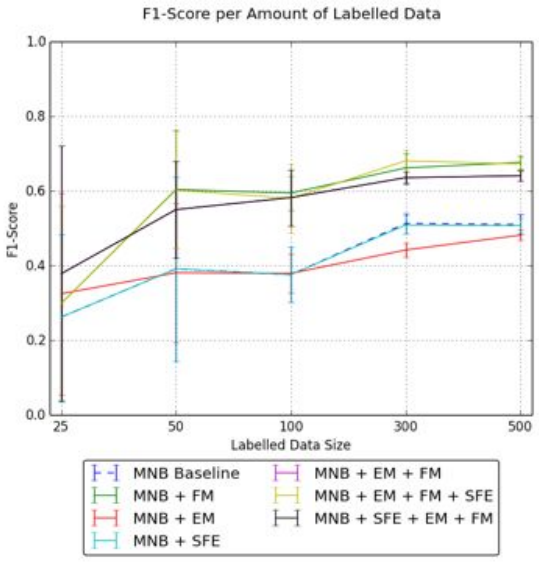


Figure 16: F1-Score: Immigration-Extended, target label = ”relevant”, 50 000 unlabelled tweets

7.2.3. Increasing the Amount of Unlabelled Data

In general, increasing the amount of labelled data had a far greater effect on classifier stability and performance than increasing the amount of unlabelled data. Figures 17 - 18 show the classification performance on the Duggan dataset with 25 labelled tweets (Figure 17) and 500 labelled tweets (Figure 18), with the target label "mark". Interestingly, for 25 labelled documents, increasing the amount of unlabelled data hurt performance and stability (NB: the error bars are smallest for 1 000 unlabelled tweets and largest for 100 000 unlabelled tweets). This suggests that there is a relation between the proportion of labelled and unlabelled data used in the model, and classification performance and stability. Using a large amount of unlabelled data together with a too small amount of labelled data hurts performance because, presumably, too much uncertainty is introduced by the vast amount of unlabelled documents. It seems that the issue of appropriately weighting the contribution of unlabelled data is more general, and not just specific to the Expectation-Maximization algorithm (see "EM Parameter Tuning" discussed further below).

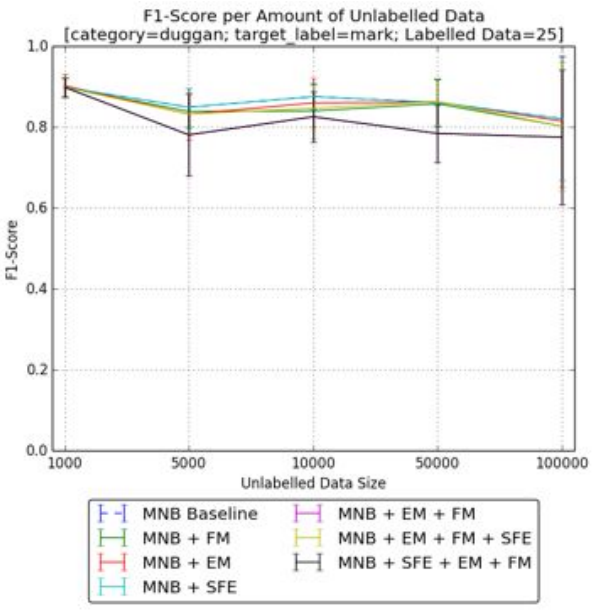


Figure 17: F1-Score: Duggan, 25 labelled tweets

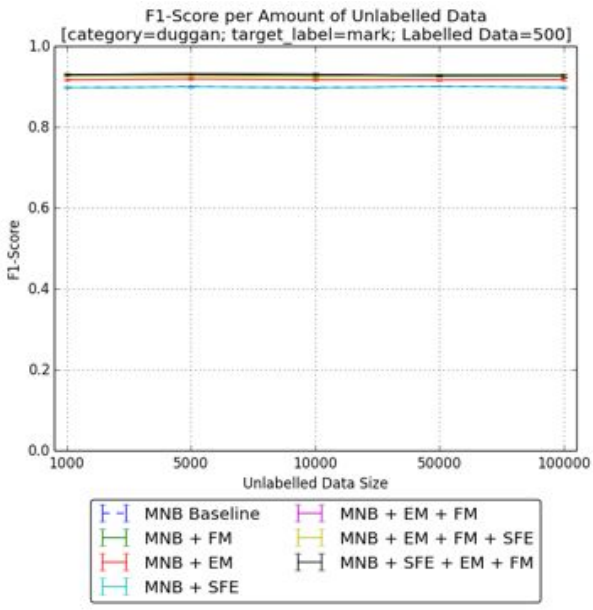


Figure 18: F1-Score: Duggan, 500 labelled tweets

Figures 19 - 20 show that in terms of Accuracy, measured on the The Voice UK 2013 dataset, with 25 labelled tweets (Figure 19) and 500 labelled tweets (Figure 20), performance improvements mainly come from the labelled data again. But increasing amounts

of unlabelled data, caused the results to be less volatile this time, with the size of the error bars on the plots shrinking with growing amounts of unlabelled tweets.

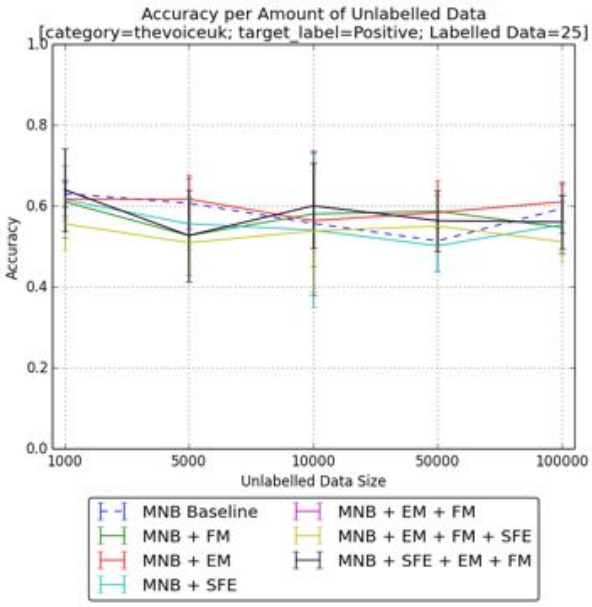


Figure 19: Accuracy: The Voice UK 2013, 25 labelled documents

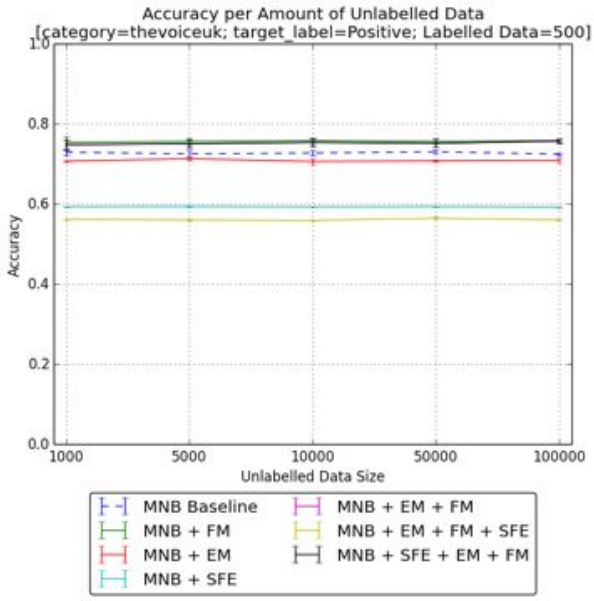


Figure 20: Accuracy: The Voice UK 2013, 500 labelled documents

7.3. Algorithm Modification Experiments

7.3.1. Uniform Priors

I conducted a series of experiments to test whether using uniform class priors has the potential to improve classification performance. A similar study has been performed by Schneider (2005) who found that using uniform class priors can increase the classification performance of up to 3-5% for short documents[21]. In my tests, using uniform class priors generally did not improve classification performance.

In the case of a standard Multinomial Naïve Bayes model, without any semi-supervised algorithms, I was able to approximate the results reported in Schneider (2005) in terms of F1-Score. As Figure 21 shows, using uniform class priors exhibited a better F1-Score

[21] Schneider (2005) did not use Twitter as the social network has only been founded in 2006, but a dataset with a similar average token length per document.

across the datasets by a small margin. Accuracy on the other hand, did not noticeably change when uniform class priors were used, as Figure 22 shows.

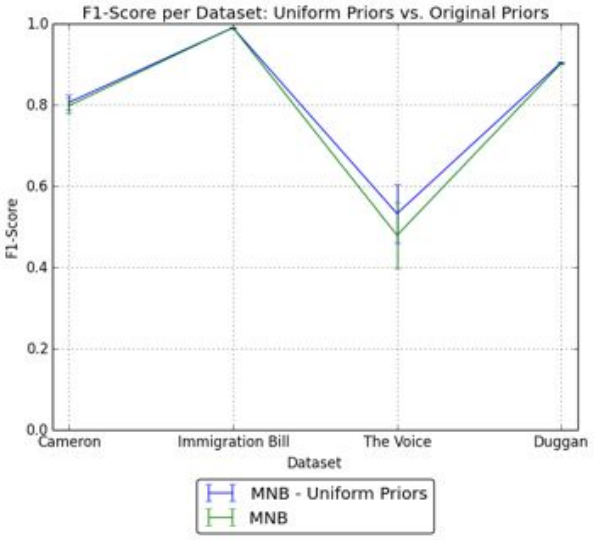


Figure 21: F1-Score: Multinomial Naïve Bayes with and without uniform class priors

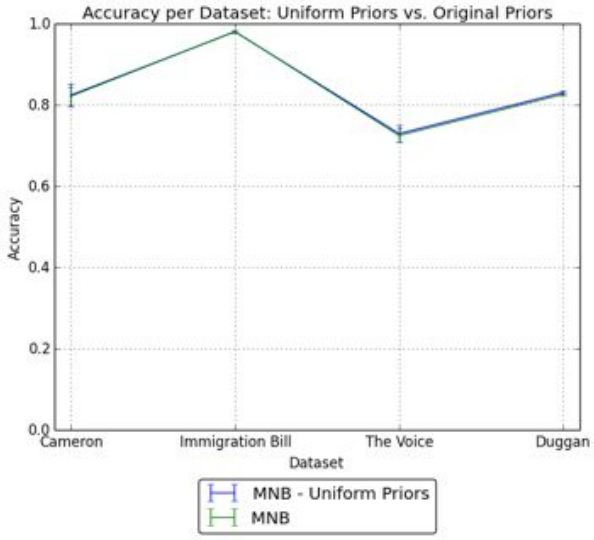


Figure 22: Accuracy: Multinomial Naïve Bayes with and without uniform class priors

When the MNB classifier was used in conjunction with a semi-supervised algorithm, the weak positive effect diminishes for EM and FM as Figures 23 - 24 show[22]. For the Feature Marginals algorithm, using uniform priors even caused a decrease in performance. For an MNB classifier combined with the Semi-supervised Frequency Estimate algorithm, using uniform priors improved classification performance, approximately at the same level as it improved performance for a Naïve Bayes classifier on its own. The reported results represent the average performance over 5 test runs with 10 fold cross-validation, evaluated for each classifier and dataset. For this experiment I selected the Cameron (target label = "comment"), the Immigration-Bill (target label = "relevant"), the The Voice UK 2013 (target label = "Positive") and the Duggan (target label = "mark") datasets. For all test runs I used 300 labelled and 50 000 unlabelled documents, or the maximum available if there were fewer labelled or unlabelled tweets in a dataset.

[22]The complete set of figures for that experiment can be found in Appendix C

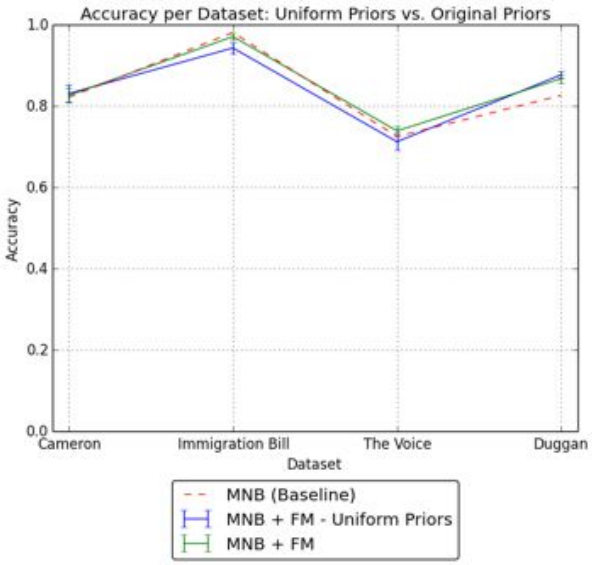


Figure 23: Accuracy: Uniform Priors - MNB with Feature Marginals

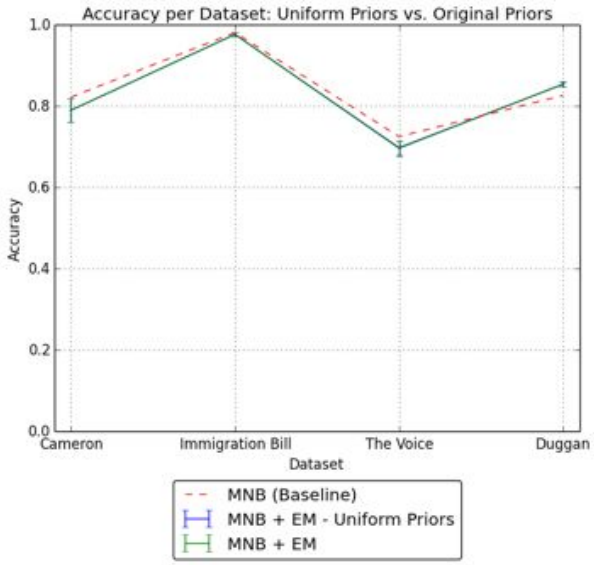


Figure 24: Accuracy: Uniform Priors - MNB with Expecation-Maximization

7.3.2. Undersampling the Majority Class

I conducted a set of experiments with the aim to improve the classification performance for datasets with imbalanced class skews (i.e. 2:98). The central idea is to undersample the majority class according to some criterion (currently it is random undersampling, but future work will aim to improve the undersampling technique[23]). This may seem counterintuitive because obtaining labels is a costly process, especially in an active learning context. But interestingly, for a scenario with a class skew of i.e. 2:98 for 100 labelled documents (so 2 positively labelled documents and 98 negatively labelled documents), ignoring 90 documents of the negative class for creating a model, often improved classification performance. The approach was most promising when the MNB model was used in combination with the Feature Marginals algorithm. Interestingly, the FM algorithm works particularly well on terms it has never seen in training, which is also observed by Lucas and Downey (2013).

The reported results represent the average performance obtained over 5 test runs, with every target class skew between 0.1-0.5 per test run. Figures 25 - 26 show the performance

[23]The concept does not appear to be particularly popular in the NLP research community, with most publications I could find being older than 10 years. Japkowicz (2000) presents a good overview of different techniques, applied to a Multi-Layer Perceptron.

on the Reuters ApteMod corpus with "interest" as target category. While the F1-Scores of the baseline MNB classifier were simply 0 for this target class, random undersampling in conjunction with the Feature Marginals algorithm was able to achieve an F1-Score of close to 60% for a target class skew of 0.1. Figure 25 further shows that there appears to be an "optimal" balance between the classes, with the performance of each method increasing until a peak is reached, and then slowly degrading again. This suggests that the target class skew is a free parameter in the model. From Figure 26 it can also be noted that Accuracy stays relatively stable as long as the majority class is not undersampled too aggressively, i.e. when the target class skew is lifted to 0.4 or 0.5, the Accuracy of the classifier becomes extremely unstable. In terms of F1-Score, the results are very volatile and worse than 0.5 on average, which I hope to improve in future work by incorporating a better method for selecting instances from the majority class.

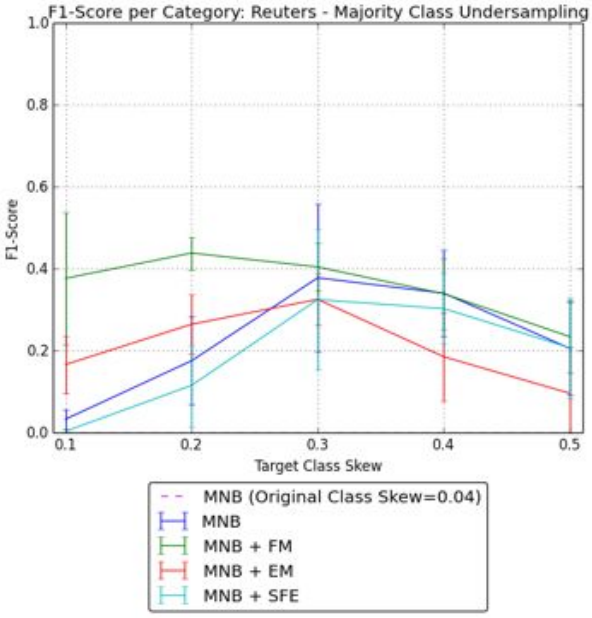


Figure 25: F1-Score: Reuters ApteMod with target label = "interest"

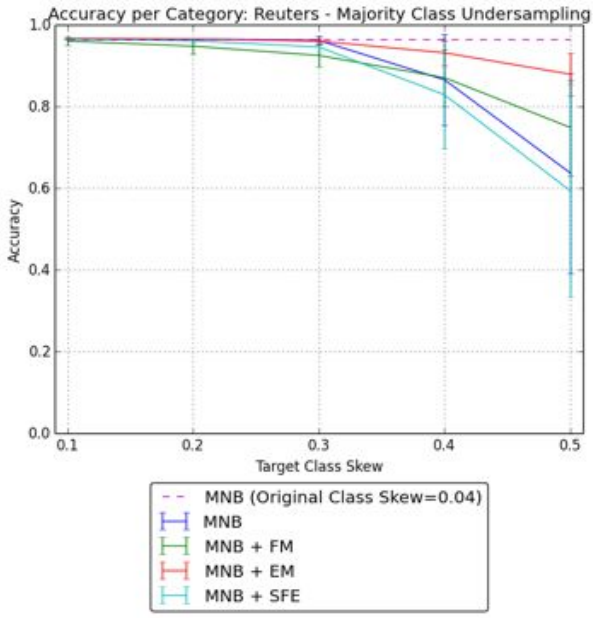


Figure 26: Accuracy: Reuters ApteMod with target label = "interest"

On Twitter datasets, it appears to be more difficult to artificially balance the class distribution. For example, the technique did not improve classification performance on the EU-Sentiment dataset, with a class skew of 1:99, where the F1-Scores across configurations, uniformly remained at 0 for the target class. This result however, is not fully conclusive as the dataset only contains 260 labelled tweets overall, of which only 3 are in the minority class. This, most likely, is too small an amount for any reliable classification decisions,

as in the case of balancing the class skew to 50:50, only 6 labelled tweets were used for training. As Figures 27 - 28 show, the technique was able to improve the F1-Score for datasets with milder class skews by a small margin, however the performance still remains below 0.5. Figure 27 displays the F1-Score for the Euro-Attitude dataset with a class skew of 14:86 and Figure 28 shows the F1-Score for the Euro-Relevance dataset with a class skew of 18:82.

Improvements could be achieved for both corpora by randomly undersampling the majority class. In the case of the Euro-Attitude dataset performance improvements of 3-5% for each method respectively, started happening at a target class skew of 0.4 for the target class, leading to an overall improvement in comparison to the baseline of $\sim$25% with the Feature Marginals algorithm. The other methods outperform the baseline by only $\sim$5%. For the Euro-Relevance dataset, performance improvements of up to 10% could be gained for the standard MNB classifier and the MNB classifier in combination with the SFE algorithm, from a class skew of 0.3 for the target class onwards, however the F1-Score remains generally low, at $\sim$0.3%. The Feature Marginals algorithm is stabler, where for a class skew of 0.4 for the target class, improvements of up to 35% in F1-Score could be achieved, resulting in an overall F1-Score performance of $\sim$0.42[24].

I will leave a more exhaustive investigation, as well as the development of improved techniques for undersampling the majority class, for future work.

[24]The complete set of figures for this experiment is listed in Appendix D

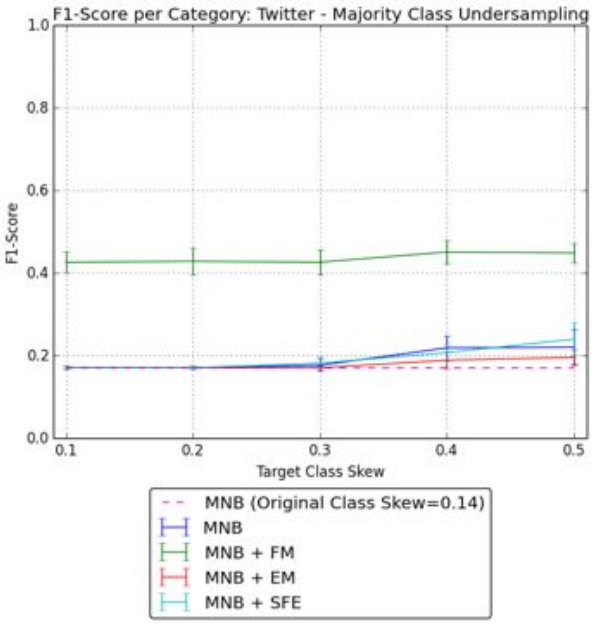


Figure 27: F1-Score: Twitter Euro-Attitude with target label = "attitudinal"

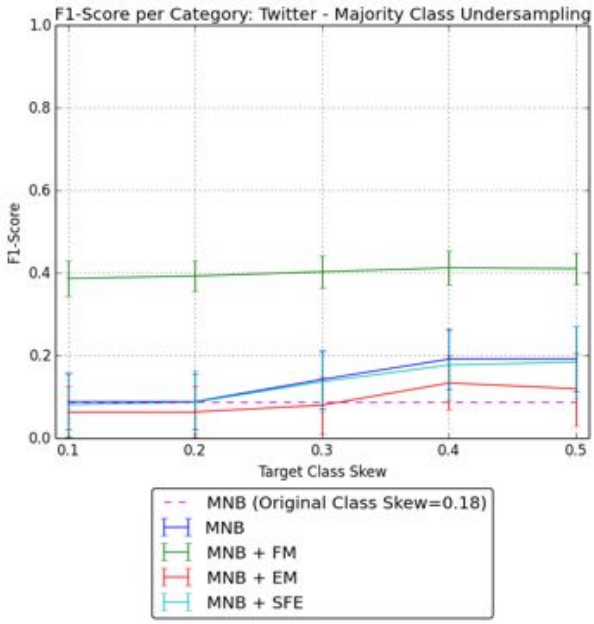


Figure 28: F1-Score: Twitter Euro-Relevance with target label = "relevant"

7.3.3. EM Parameter Tuning

One of the central modifications to EM for text classification that Nigam et al. (2000) describe, is a weighting factor for the probabilistically labelled documents. The weighting factor represents a free parameter and is necessary to prevent larger amounts of unlabelled data from drowning out the signal of the hand-labelled instances. In the original version of DUALIST, Settles (2011) used a static weighting factor of 0.1, which was not theoretically or empirically justified in the publication. A weight of 0.1 means that the feature frequencies, of each probabilistically labelled instance, are divided by 10. Optimising this parameter is absolutely crucial for the performance of the EM algorithm, especially when the unlabelled data outnumber the labelled data by a factor of 100 or greater, as my experiments show.

I tested 3 different parameter settings for the weighting factor: the first one is the original value of 0.1 that Settles (2011) used, the second one is $\frac{|d_l|}{|d_u|}$, where $|d_l|$ is the number of labelled data, and $|d_u|$, is the number of unlabelled data, henceforth referred to as "PWF 1", where PWF stands for "Proportional Weighting Factor". The third one is $\frac{|d_l|}{10*|d_u|}$, henceforth referred to as "PWF 2", where $|d_l|$ and $|d_u|$ represent the same quantities as

for PWF 1, which I found to outperform the other 2 methods, and which I adopted for all other experiments. It should be noted that PWF 1 and PWF 2 still only represent an "educated guess". I will leave an exhaustive empirical study concerning the optimisation of the weighting parameter for future work.

This experiment was evaluated over 10 test runs and 10 fold cross-validation on the Cameron-2 (target label = "relevant"), the Immigration-Extended (target label = "relevant") and the Duggan-Main (target label = "nojustice") datasets. For all experiments I used 300 labelled and 50 000 unlabelled documents, or the maximum available if there were fewer labelled or unlabelled tweets in a dataset. The methods were compared to a standard MNB classifier as baseline.

As Figures 31 - 32 show, the performance of the hardcoded weight of 0.1, and the performance of PWF 1, is generally poor. PWF 2 exhibited a better performance for the Cameron-2 dataset, outperforming the baseline in terms of Accuracy and F1-Score, but exhibiting a worse performance for the other 2 datasets. As the Cameron-2 dataset only contains $\sim$15 000 unlabelled tweets, compared to more than 50 000 unlabelled tweets in the other 2 datasets, I am assuming that the weighting factor for the unlabelled data in the Duggan-Main and Immigration-Extended datasets still is too large. These results suggest that the value of the weighting factor has a direct and highly significant influence on classification performance, and potentially needs to be optimised for every corpus individually.

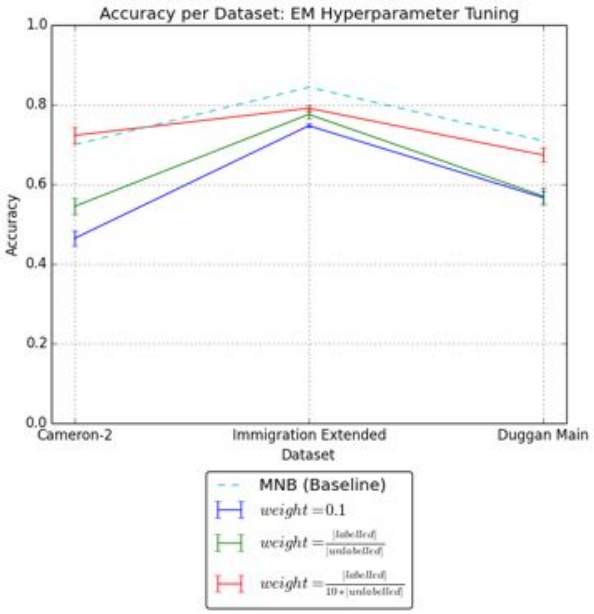


Figure 29: Accuracy: EM Hyperparameter Tuning

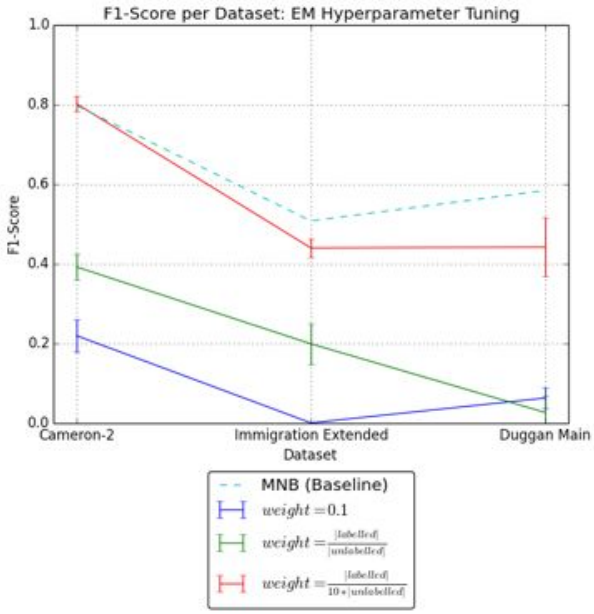


Figure 30: F1-Score: EM Hyperparameter Tuning

7.3.4. Most Surely Uncertain (MSU) vs. Least Surely Uncertain (LSU)

This set of experiments is not targeted at the core classification framework, but it rather is a promising addition to the instance querying algorithm in the active learning process itself.

Uncertainty sampling probably is the most commonly used technique for instance querying in active learning[25]. The idea of it is that the learning algorithm "asks" a human annotator to label the documents that are closest to the classifiers' current decision boundary (NB: the documents with the highest entropy in this case). However, there are 2 different kinds of uncertainty as Sharma and Bilgic (2013) show: documents that contain a large amount of evidence for either of the target classes, the "Most-Surely Uncertain" (MSU) instances, and documents that hardly provide any evidence for either of the target classes, the "Least-Surely Uncertain" (LSU) instances. The authors show that by focusing on the Most-Surely Uncertain documents, the performance of an active learner can be significantly improved.

I evaluated this methodology by simulating the active learning process on the EU-Attitude and the EU-Relevance datasets (labelled data only), and could reproduce the positive

[25] see Settles (2009) or Lewis and Gale (1994)

effect of using MSU sampling instead of basic uncertainty sampling as Figures 31 - 32 show[26]. The baseline consists of standard uncertainty sampling where always 10 instances were labelled per iteration. The reported results were obtained over 10 test runs, where 1 simulated test run consisted of 20 active learning iterations. For these experiments only a standard MNB classifier, without any semi-supervised algorithms, was used.

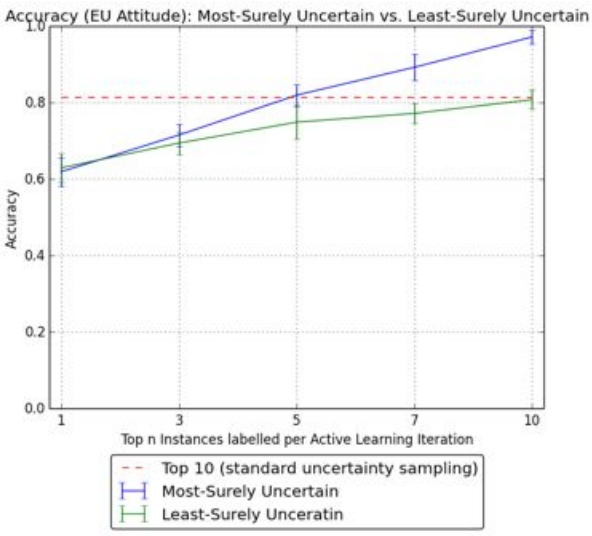


Figure 31: Accuracy: EU - Attitude, MSU sampling vs. LSU sampling vs. standard uncertainty sampling

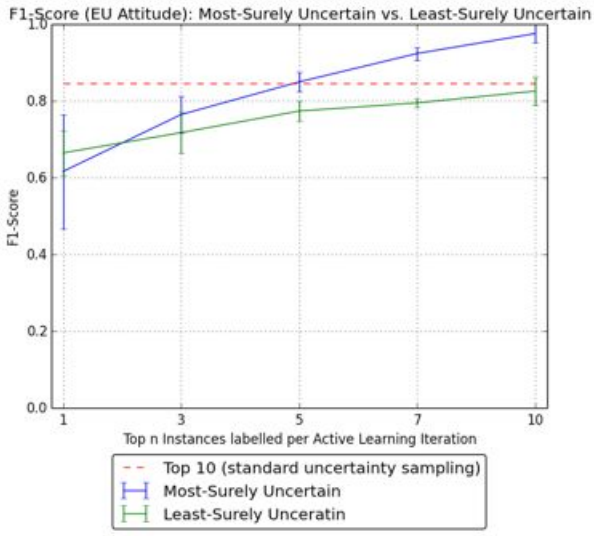


Figure 32: F1-Score: EU - Attitude, MSU sampling vs. LSU sampling vs. standard uncertainty sampling

The results not only show the potential of the method to outperform standard uncertainty sampling, but also that equal performance can be achieved by labelling fewer documents, hence accelerating the active learning process as a whole. An interesting effect that I observed was that by labelling only 1 instance per active learning iteration, LSU sampling outperformed MSU sampling by a small margin. This is quite counterintuitive, as one would not expect that labelling fewer documents, containing less evidence, can perform better than labelling a small number of documents containing a large amount of evidence for the target classes. This result nonetheless suggests that MSU sampling is only able to unleash its full potential when at least 3-5 instances are labelled per learning iteration.

[26]The complete set of Figures for this experiment can be obtained in Appendix E

8. Future Work

8.1. Improving the Feature Marginals Algorithm

As I have shown in the main section of this report, the Feature Marginals algorithm can consistently improve the performance of a Multinomial Naïve Bayes classifier, but it currently is able to handle binary problems only. I therefore plan to generalise the method to support any number of classes[27].

Further, it would be interesting to investigate other root finding algorithms, or evaluating smaller values of the maximum number of iterations in the root finding algorithm, to reduce the computational cost of the method.

8.2. EM Parameter Tuning

As my experiments in the previous section revealed, optimising the weighting factor for the probabilistically labelled instances is crucial to the performance of EM, especially in a scenario with a very small set of labelled data and very large amounts of unlabelled documents. I therefore plan to perform an exhaustive empirical study to find a way of automatically configuring the optimal weighting for any given dataset, which seems to be a necessary step in EM to unleash its full potential.

8.3. EM Smoothing

Viewing the EM version in DUALIST from a clustering angle[28], where labelled features represent the centroids with a particular class adherence in the feature space. In the model initialisation step, unlabelled instances are clustered according to the hand labelled centroid features they contain (see Figure 33 for a simple and idealised representation). Settles (2011) illustrates the performance on a corpus containing Usenet messages, with the task of distinguishing between Ice Hockey and Baseball related messages. It seems intuitive for that technique to work in such a case, as one would expect top-domain related

[27] I used an OVO scheme as described in Flach (2012), in the case of a problem being split into more than 2 classes.

[28] In fact, EM can be seen as a generalisation of k-Means clustering, see Alpaydin (2010) for a good introduction.

words like ”puck”, ”goal”, ”hockey”, or names of players and teams, among many others, to occur in most of the Ice Hockey related messages. Similarly, one would expect to find ”inning”, ”bat”, etc., in most of the Baseball related messages. For an agile scenario such as faced by CASM researchers, any class-specific words may not be so obvious, foremost because of the problem space being largely unknown. A further complication is the fact that Twitter messages are very short, thus, there are only 140 characters of ”physical” space in a document for keywords to occur.

The idea of applying smoothing in the Expectation-Maximization algorithm is to add a lower pseudo-count to highly information salient features that co-occur in the same tweet with any hand-labelled features[29]. For example in Figure 33 below, the token ”#Avoid” might be a good candidate for a soft pseudo-count for the negative class, and likewise the token ”#LegoMovieFTW” could be a promising candidate for the positive class. Ultimately, a measure such as Information Gain could be a plausible choice for deciding which features should receive a soft-increment. It would be very interesting to conduct an empirical study to validate this hypothesis, as to the best of my knowledge, such a technique has not yet been applied in text classification, or in fact, anywhere else.

[29]Recall that Settles (2011) added a pseudo-count of 50 to hand labelled features in DUALIST. A value between 10-25 might be appropriate for a soft pseudo-count.

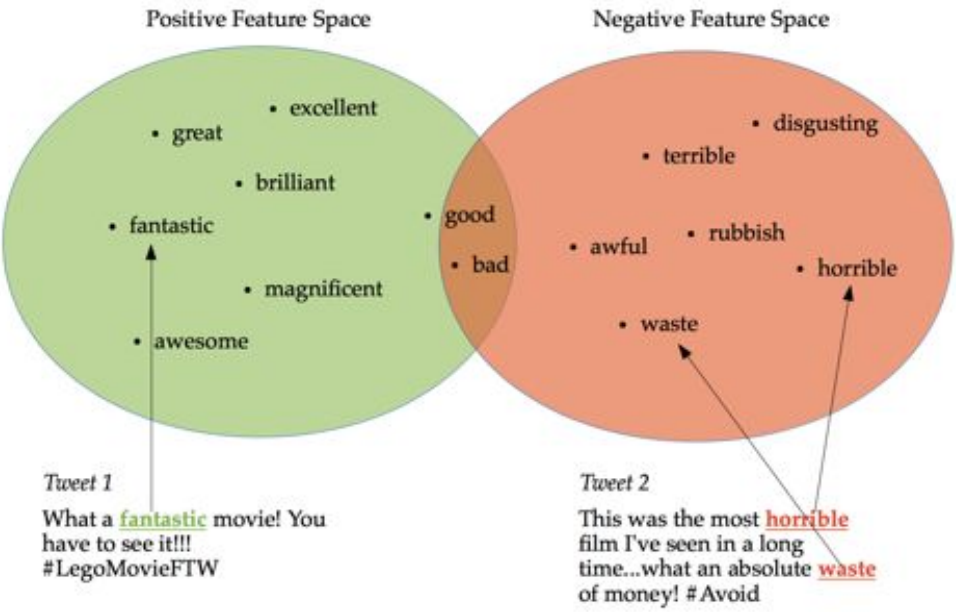


Figure 33: The EM model initialisation in DUALIST from a clustering angle

8.4. Undersampling the Majority Class

In the previous section I have shown that even random undersampling of the majority class has the potential of improving the performance on a dataset with imbalanced classes, hence selecting the instances of the majority class according to an information theoretic criterion may enable further improvements in classification accuracy. However, another free parameter would be introduced by this method, namely, the decision of the ”optimal” target class skew for a given problem. It would be interesting to study whether there actually is an ”optimal” class skew for a given dataset, and if it can be found automatically. Further, it would be interesting to study whether this concept can be extended to the unlabelled data as well, as clearly, one wants to avoid introducing ambiguity and noise by incorporating the unlabelled data into the model.

8.5. Uniform Priors

As I have shown in the previous section, using uniform priors generally does not improve the classification performance when used in combination with an SSL algorithm. However, there still remains the question whether the priors do have an effect on the labelling process. In an active learning scenario such as faced by CASM researchers, part of the goal is to explore the problem space. Hence the question becomes, whether enough knowledge about the problem space is available at any point in the labelling process, to justify the use of class priors. It is an important question to consider, especially as the class priors can severely influence the classification decision made by the algorithm. It would therefore be interesting to investigate whether performance improvements can be achieved by using uniform class priors in the labelling process.

9. Conclusion

I presented a comparison of a standard Multinomial Naïve Bayes classifier in combination with semi-supervised algorithms such as, Expectation-Maximization, Semi-supervised Frequency Estimate, Feature Marginals and combinations thereof, on the Reuters Apte-Mod corpus and on several real-world Twitter datasets. Overall, an MNB classifier in conjunction with the Feature Marginals algorithm outperformed the other methods, and consistently improved the classification performance when compared with a standard Multinomial Naïve Bayes classifier baseline. The method comes with the caveat of being computationally more expensive than the other algorithms. I also showed that by combining a very small amount of labelled data (i.e. 25 documents) with a very large amount of unlabelled data (i.e. 50 000-100 000 documents), the classification performance in terms of F1-Score of a classifier becomes very unstable and often decreases with growing amounts of unlabelled data.

I further found that uniform class priors do not improve classification performance of an MNB classifier combined with the Feature Marginals algorithm or the Expectation-Maximization algorithm, but do improve performance when used with the Semi-supervised Frequency Estimate algorithm. I showed that randomly undersampling the majority class in case of imbalanced class distributions does improve classification performance, but currently, not beyond 0.5 in terms of F1-Score. I further presented that the performance of the Expectation-Maximization algorithm is largely dependent on the optimisation of its free parameters, foremost the weighting factor assigned to probabilistically labelled instances, and I highlighted that by using an alternative uncertainty sampling technique, Most-Surely Uncertain sampling, the active learning process can be accelerated.

10. References

Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition, 2010. ISBN 026201243X, 9780262012430.

Sitaram Asur and Bernardo A. Huberman. Predicting the future with social media. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '10, pages 492–499, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4191-4. doi: 10.1109/WI-IAT.2010.63. URL http://dx.doi.org/10.1109/WI-IAT.2010.63.

Anthony Aue and Michael Gamon. Customizing sentiment classifiers to new domains: A case study. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*, 2005.

Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1 – 8, 2011. ISSN 1877-7503. doi: http://dx.doi.org/10.1016/j.jocs.2010.12.007. URL http://www.sciencedirect.com/science/article/pii/S187775031100007X.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38, 1977. URL http://web.mit.edu/6.435/www/Dempster77.pdf.

Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Mach. Learn.*, 29(2-3):103–130, November 1997. ISSN 0885-6125. doi: 10.1023/A:1007413511361. URL http://dx.doi.org/10.1023/A:1007413511361.

Peter Flach. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge University Press, New York, NY, USA, 2012. ISBN 1107422221, 9781107422223.

George Forman and Martin Scholz. Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *SIGKDD Explor. Newsl.*, 12(1):49–57, November 2010. ISSN 1931-0145. doi: 10.1145/1882471.1882479. URL http://doi.acm.org/10.1145/1882471.1882479.

Nathalie Japkowicz. Learning from imbalanced data sets: A comparison of various strate-

gies. In *Proceedings of the AAAI Workshop on Learning from Imbalanced Data Sets*, pages 10–15. AAAI Press, 2000.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X. URL `http://dl.acm.org/citation.cfm?id=188490.188495`.

Michael Lucas and Doug Downey. Scaling semi-supervised naive bayes with feature marginals. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 343–351, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P13-1034`.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

Micol Marchetti-Bowick and Nathanael Chambers. Learning for microblogs with distant supervision: political forecasting with twitter. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 603–612, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. ISBN 978-1-937284-19-0. URL `http://dl.acm.org/citation.cfm?id=2380816.2380890`.

Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *IN AAAI-98 WORKSHOP ON LEARNING FOR TEXT CATEGORIZATION*, pages 41–48. AAAI Press, 1998.

Kamal Nigam. Using maximum entropy for text classification. In *In IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Mach. Learn.*, 39(2-3): 103–134, May 2000. ISSN 0885-6125. doi: 10.1023/A:1007692713085. URL `http://dx.doi.org/10.1023/A:1007692713085`.

Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf.*

Retr., 2(1-2):1–135, January 2008. ISSN 1554-0669. doi: 10.1561/1500000011. URL http://dx.doi.org/10.1561/1500000011.

Karl-Michael Schneider. Techniques for improving the performance of naive bayes for text classification. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'05, pages 682–693, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-24523-5, 978-3-540-24523-0. doi: 10.1007/978-3-540-30586-6_76. URL http://dx.doi.org/10.1007/978-3-540-30586-6_76.

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002. ISSN 0360-0300. doi: 10.1145/505282.505283. URL http://doi.acm.org/10.1145/505282.505283.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, http://burrsettles.com/pub/settles.activelearning.pdf, 2009.

Burr Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1467–1478, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D11-1136.

Burr Settles and Xiaojin Zhu. Behavioral factors in interactive training of text classifiers. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 563–567, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. ISBN 978-1-937284-20-6. URL http://dl.acm.org/citation.cfm?id=2382029.2382116.

Manali Sharma and Mustafa Bilgic. *Most-Surely vs. Least-Surely Uncertain; Data Mining (ICDM), 2013 IEEE 13th International Conference on*. 2013.

Jiang Su, Jelber S. Shirab, and Stan Matwin. Large scale text classification using semi-supervised multinomial naive bayes. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 97–104, New York, NY, USA, 2011. ACM. URL http://www.icml-2011.org/papers/93_icmlpaper.pdf.

A. Tumasjan, T.O. Sprenger, P.G. Sandner, and I.M. Welpe. Predicting elections with

twitter: What 140 characters reveal about political sentiment. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 178–185, 2010. URL http://scholar.google.de/scholar.bib?q=info:mc319eHjea8J:scholar.google.com/&output=citation&hl=de&as_sdt=0&ct=citation&cd=28.

Simon Wibberley, David Weir, and Jeremy Reffin. Language technology for agile social media science. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 36–42, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2705.

Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005. URL http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf.

Appendix A.

External Tools and Libraries

I used the Python programming language (version 2.7.2), alongside its scientific computing stack, for the software engineering part of this project. In the following, I list all libraries and frameworks, alongside its version number, that were installed in the Python Virtual Environment ("virtualenv") for this project[30]. I found an excellent starting point, to getting the Python scientific computing stack up and running, to be the ScipySuperpack, available at: `http://fonnesbeck.github.io/ScipySuperpack/`. Further, an excellent starting point for setting up Python virtualenvs, which are invaluable for developing several different projects in parallel and for keeping the set of tools for a project tidy, is `http://hackercodex.com/guide/python-development-environment-on-mac-osx/`.

[30] Not all of the listed tools were actively used during development. Some libraries are also included because they are dependencies of other libraries.

Library	Version	Project Website
beautifulsoup4	4.3.2	http://www.crummy.com/software/BeautifulSoup/
docutils	0.11	http://docutils.sourceforge.net/
husl	2.0.2	https://github.com/boronine/pyhusl
ipython	2.0.0	http://ipython.org/
Jinja2	2.7.1	http://jinja.pocoo.org/
MarkupSafe	0.18	http://www.pocoo.org/projects/markupsafe/
matplotlib	1.3.0	http://matplotlib.org/
moss	0.1	https://github.com/mwaskom/moss
nltk	2.0.4	http://www.nltk.org/
nose	1.3.0	https://nose.readthedocs.org/en/latest/
numpy	1.9.0	http://www.numpy.org/
pandas	0.12.0	http://pandas.pydata.org/
patsy	0.2.1	https://github.com/pydata/patsy
pika	0.9.13	https://github.com/pika/pika/
psycopg2	2.5.2	http://initd.org/psycopg/
pymc	2.3a	https://github.com/pymc-devs/pymc
Pygments	1.6	http://pygments.org/
PyMySQL	0.6.1	https://github.com/PyMySQL/PyMySQL
pyparsing	1.5.7	http://pyparsing.wikispaces.com/
python-dateutil	2.1	https://labix.org/python-dateutil
python-slugify	0.0.6	https://github.com/un33k/python-slugify
pytz	2013d	http://pythonhosted.org/pytz/
PyYAML	3.10	http://pyyaml.org/
pyzmq	13.1.0	http://zeromq.org/bindings:python
readline	6.2.4.1	https://github.com/ludwigschwardt/python-readline
scikit-learn	0.14	http://scikit-learn.org/stable/
scipy	0.14	http://www.scipy.org/
seaborn	0.1	http://www.stanford.edu/~mwaskom/software/seaborn/
six	1.4.1	https://pythonhosted.org/six/
Sphinx	1.2b2	http://sphinx-doc.org/
statsmodels	0.6.0	http://statsmodels.sourceforge.net/
sympy	0.7.5	http://sympy.org/en/index.html
tornado	3.1.1	http://www.tornadoweb.org/en/stable/
Unidecode	0.04.14	https://github.com/iki/unidecode
virtualenv	1.9	http://www.virtualenv.org/en/latest/
wsgiref	0.1.2	https://pypi.python.org/pypi/wsgiref

Table 20: External Tools and Libraries

Appendix B.

Pre-Evaluation

The following plots show the Accuracy, F1-Score and combined Accuracy & F1-Score (arithmetic mean of the two), over all possible combinations of all algorithms, that served as the basis for my choice of the combinations of EM, FM & SFE, EM & FM and SFE, EM & FM. I made my decision according to the results of the combined Accuracy & F1-Score.

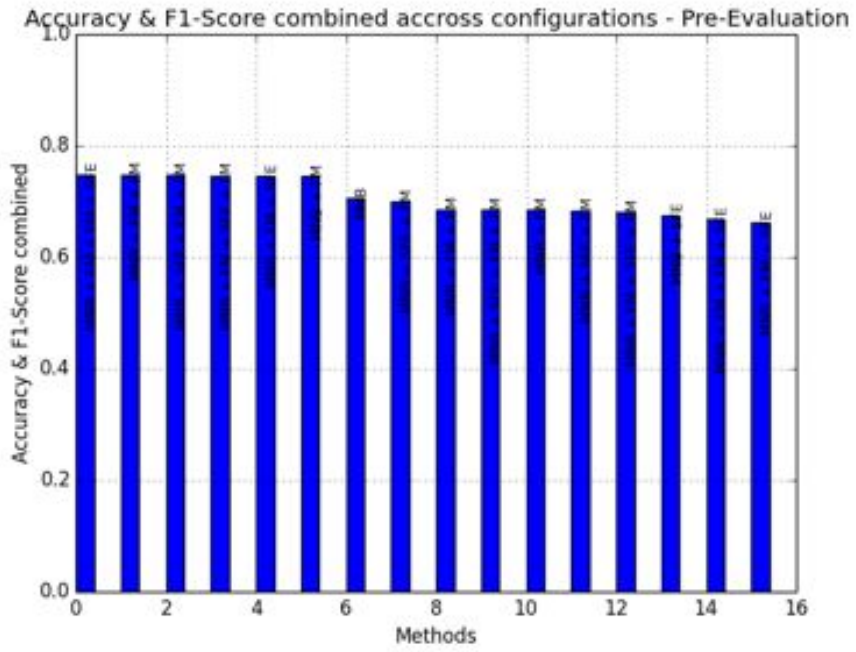


Figure 34: Mean of Accuracy & F1-Score: Combined Average across Datasets and Configurations

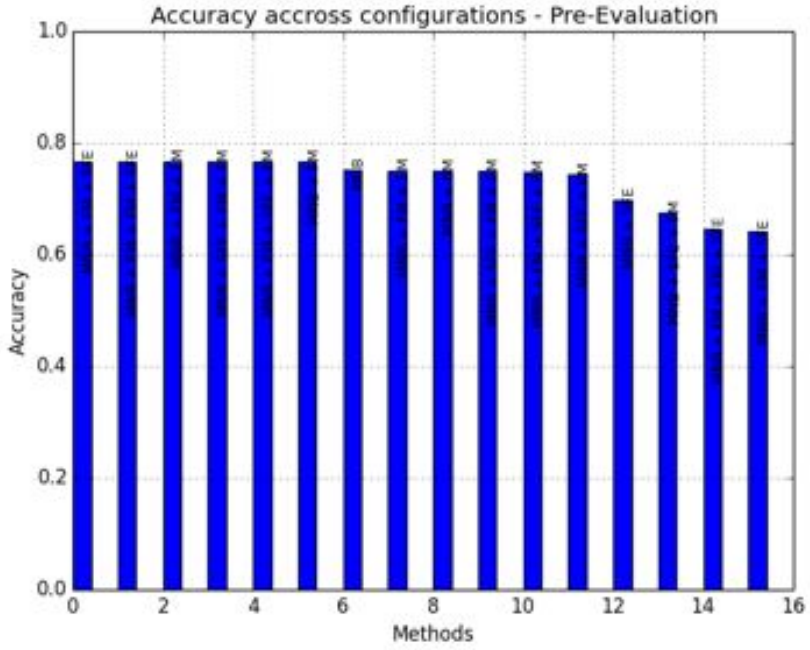


Figure 35: Accuracy: Combined Average across Datasets and Configurations

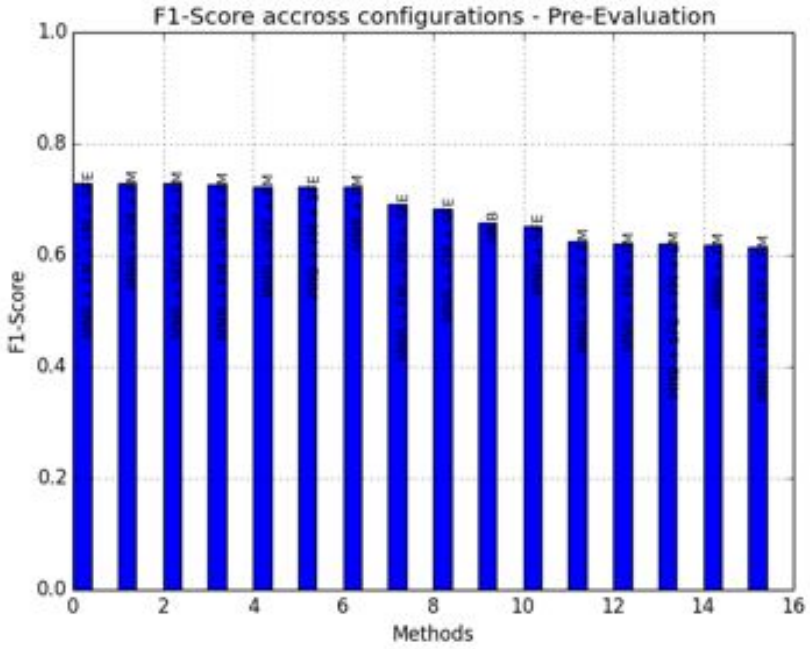


Figure 36: F1-Score: Combined Average across Datasets and Configurations

Appendix C.

Uniform Priors

The following plots represent the complete set of figures produced for testing the hypothesis whether or not uniform class priors improve classification performance.

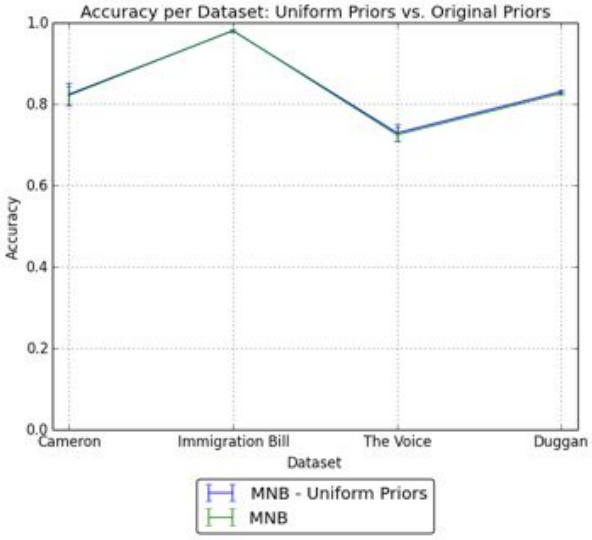


Figure 37: Accuracy: MNB vs. MNB with Uniform Priors

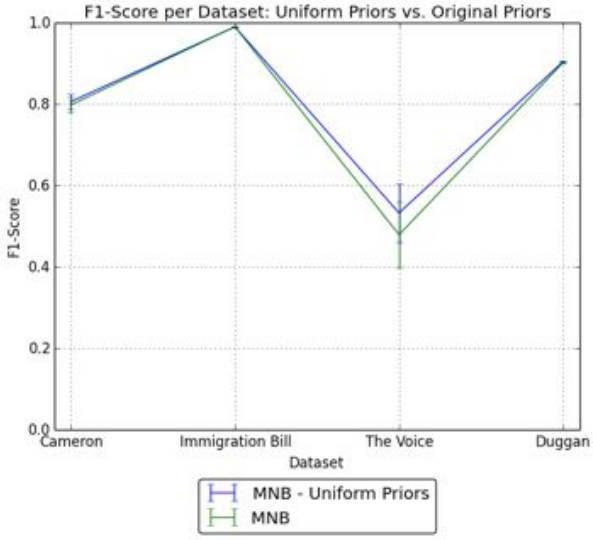


Figure 38: F1-Score: MNB vs. MNB with Uniform Priors

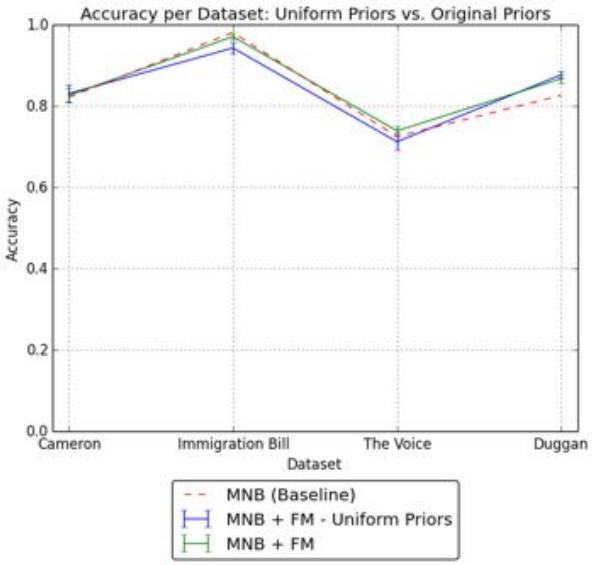


Figure 39: Accuracy: MNB + FM vs. MNB + FM with Uniform Priors in comparison to an MNB baseline

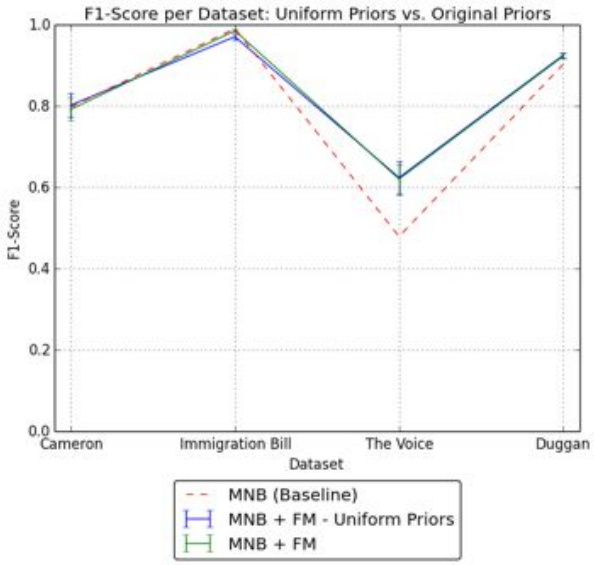


Figure 40: F1-Score: MNB + FM vs. MNB + FM with Uniform Priors in comparison to an MNB baseline

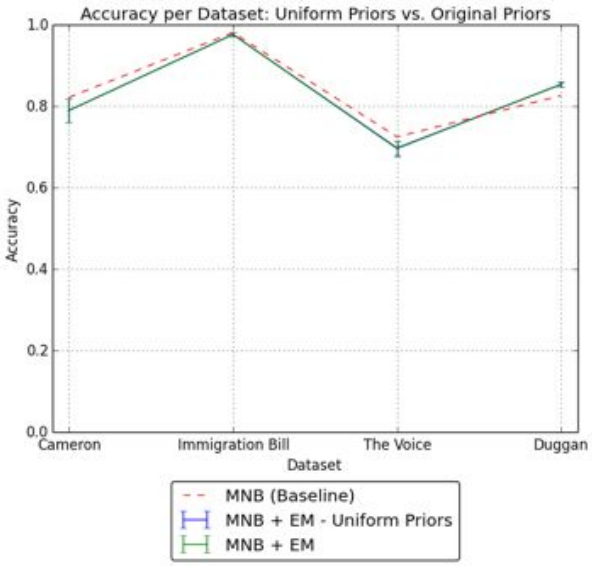


Figure 41: Accuracy: MNB + EM vs. MNB + EM with Uniform Priors in comparison to an MNB baseline

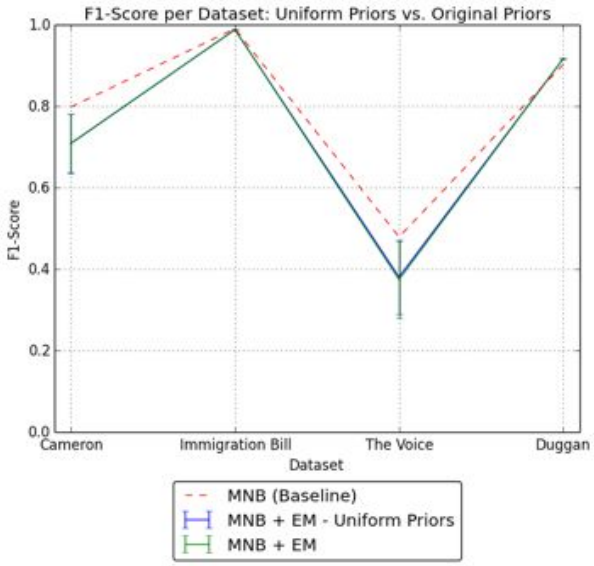


Figure 42: F1-Score: MNB + EM vs. MNB + EM with Uniform Priors in comparison to an MNB baseline

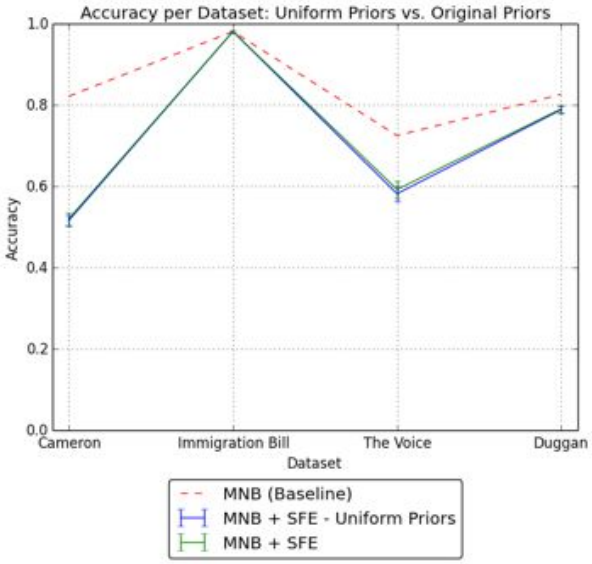


Figure 43: Accuracy: MNB + SFE vs. MNB + SFE with Uniform Priors in comparison to an MNB baseline

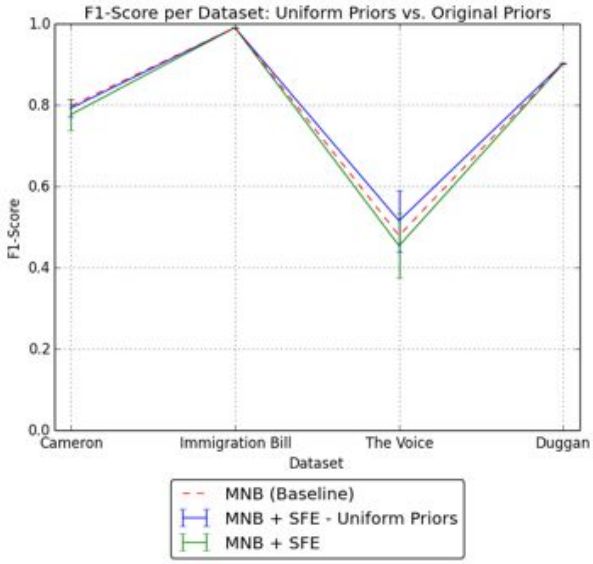


Figure 44: F1-Score: MNB + SFE vs. MNB + SFE with Uniform Priors in comparison to an MNB baseline

Appendix D.

Undersampling the Majority Class

The following plots represent the complete set of figures, obtained from testing the hypothesis whether or not randomly undersampling the majority class can improve classification performance.

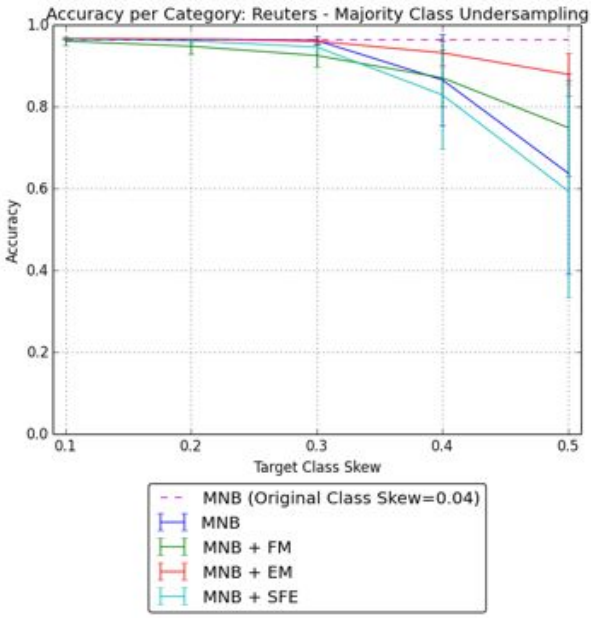


Figure 45: Accuracy: Reuters ApteMod - ”interest”

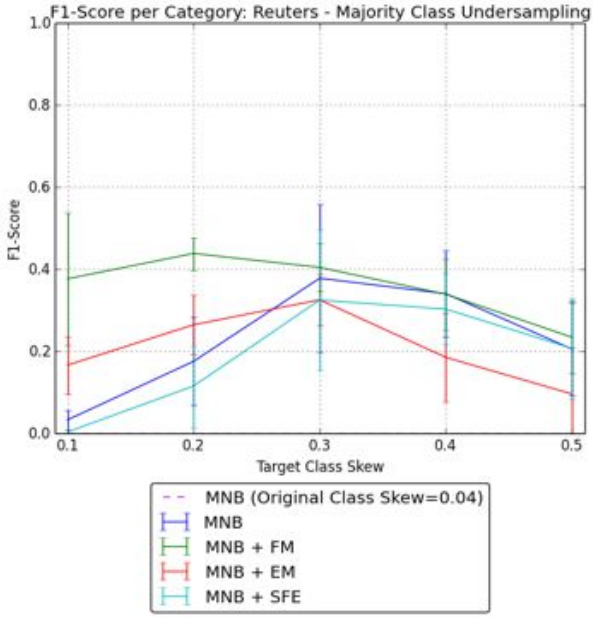


Figure 46: F1-Score: Reuters ApteMod - ”interest”

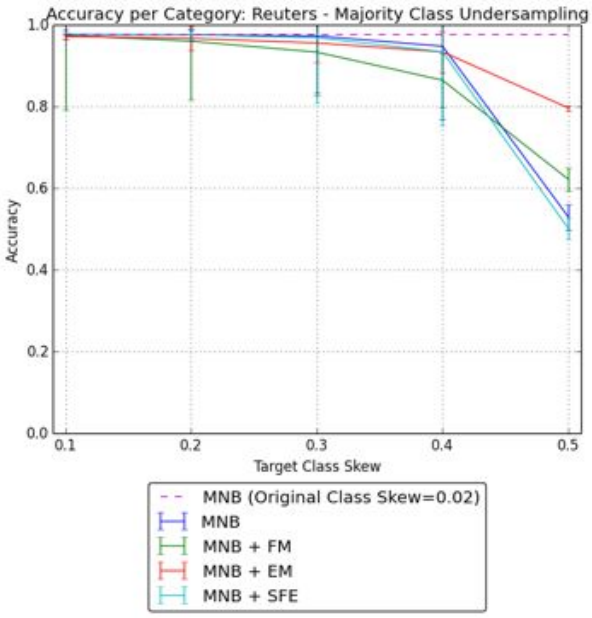


Figure 47: Accuracy: Reuters ApteMod - "ship"

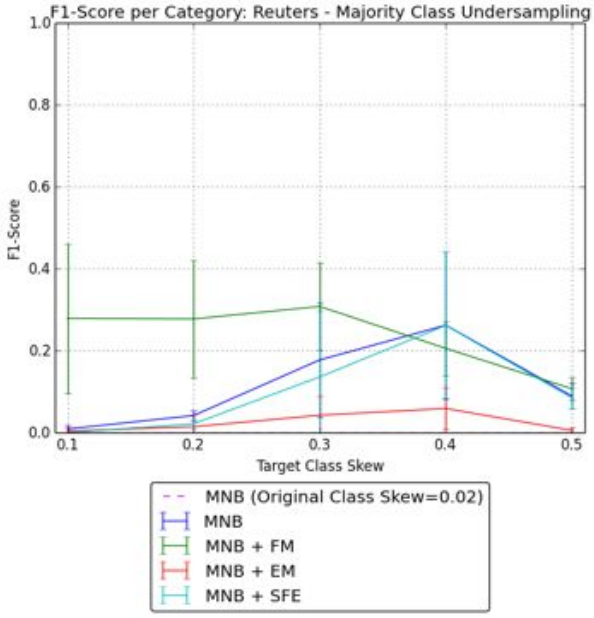


Figure 48: F1-Score: Reuters ApteMod - "ship"

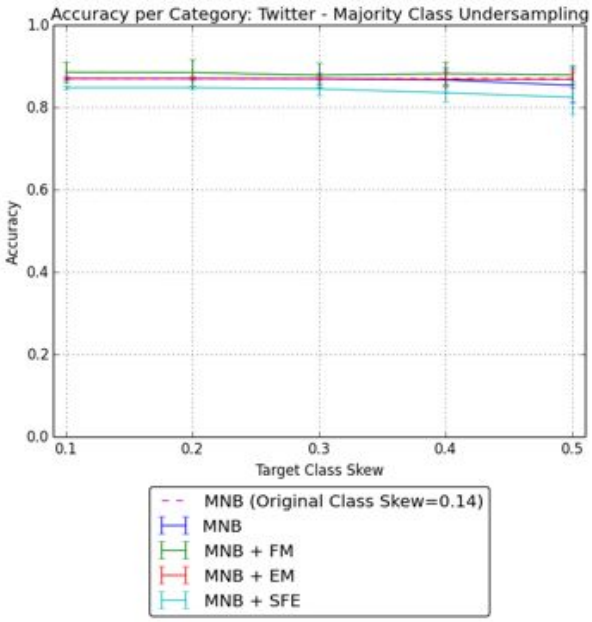


Figure 49: Accuracy: Twitter Euro-Attitude - "attitudinal"

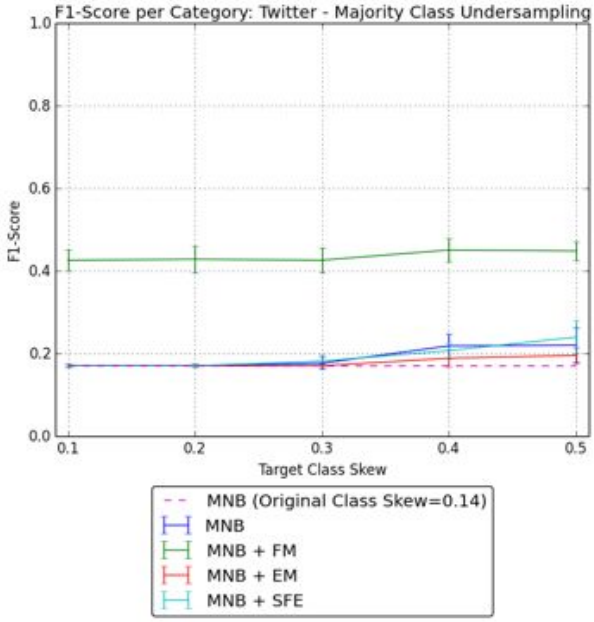


Figure 50: F1-Score: Twitter Euro-Attitude - "attitudinal"

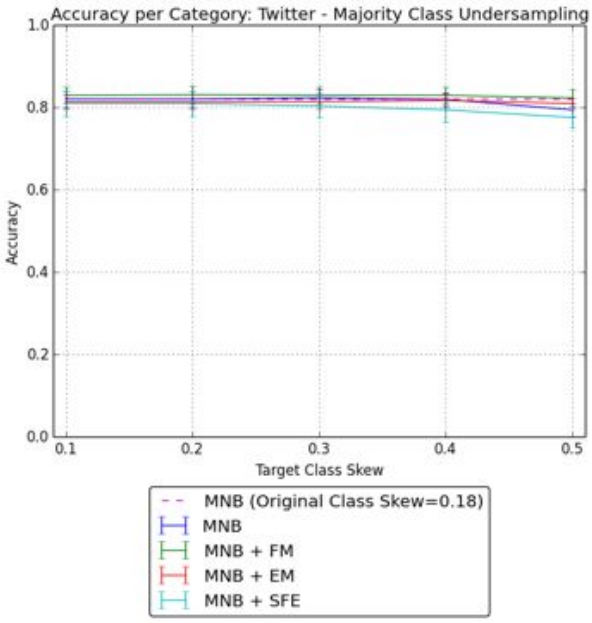


Figure 51: Accuracy: Twitter Euro-Relevance - ”relevant”

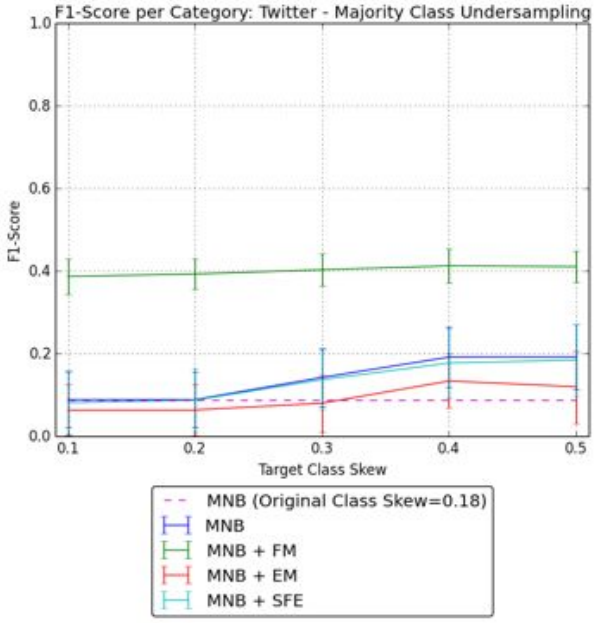


Figure 52: F1-Score: Twitter Euro-Relevance - ”relevant”

Appendix E.

Most-Surely Uncertain vs. Least-Surely Uncertain

The following plots represent the complete set of figures when testing the hypothesis whether or not using Most-Surely Uncertain sampling can improve the active learning process.

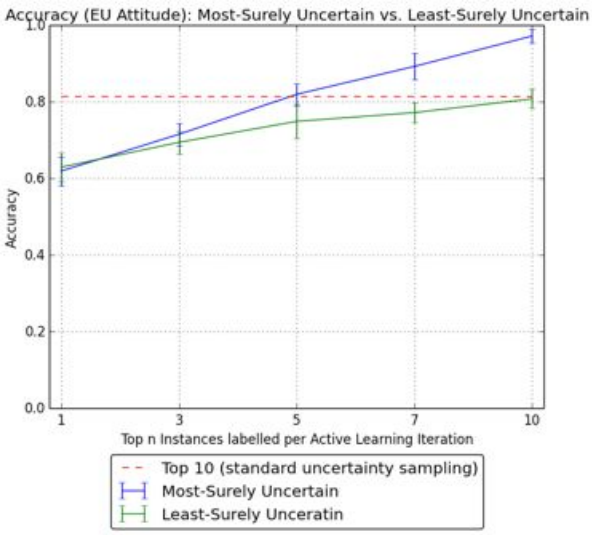


Figure 53: Accuracy: MSU vs. LSU vs. Standard Uncertainty Sampling - EU-Attitude

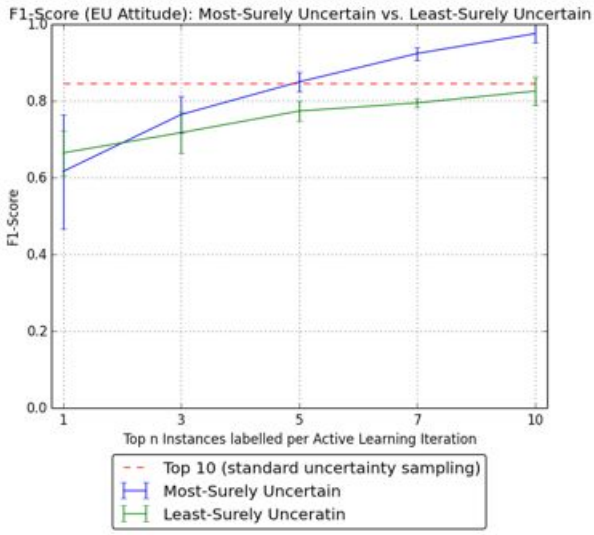


Figure 54: F1-Score: MSU vs. LSU vs. Standard Uncertainty Sampling - EU-Attitude

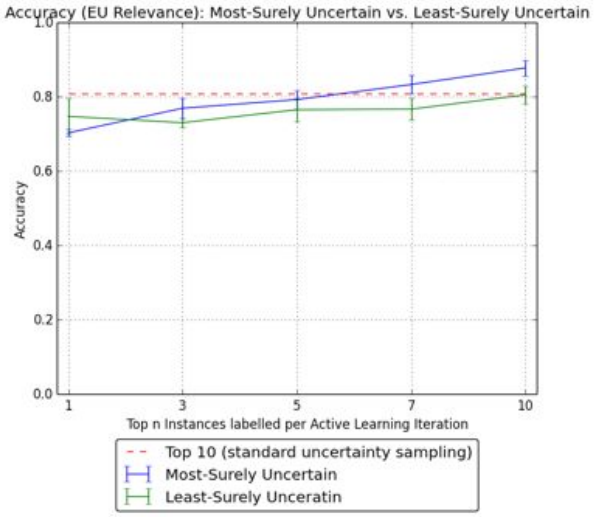


Figure 55: Accuracy: MSU vs. LSU vs. Standard Uncertainty Sampling - EU-Relevance

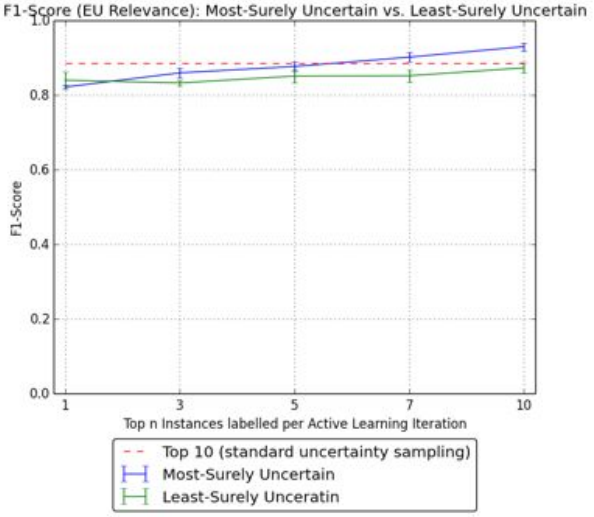


Figure 56: F1-Score: MSU vs. LSU vs. Standard Uncertainty Sampling - EU-Relevance

Appendix F.

Work Log

The following pages contain the informally authored work log for this project. Sometimes written in desperation, sometimes in euphoria, these pages tell the story from the very first to the very last push to the bitbucket repository at `https://bitbucket.org/tttthomasssss/finalyearproject/`.

- **20/08/2013 - 23/09/2013**
  - Project Setup on Bitbucket
  - Installation of all necessary tools (scipy, numpy, matplotlib, scikit-learn, etc)
  - Starting Development of Final Year Project
  - Development done so far: NaiveBayesClassifier, File I/O (compatible with scikit-learn), first version of Expectation Maximisation algorithm
- **24/09/2013**
  - Starting setup & installation of project virtualenv (failed once...ARGH!!!)
  - Using CircusPonies Notebook for Project Management
  - Work on File I/O
- **25/09/2013**
  - Setup of virtualenv on home mac
  - Lots of work on File I/O
- **26/09/2013**
  - Setup of notebook and virtualenv on lab mac
  - A lot of more work on the File I/O, the ApteMod corpus can now be fed into a Scikit Classifier
  - The many classes of the ApteMod Corpus can be translated into binary problems (3-way problems not yet implemented)
  - Found an Issue on the NaiveBayesClassifier where it has problems if the training data are a sparse matrix in case of a plain bumpy array
- **27/09/2013**

  - Using BibDesk as bibliography tool
  - Added core papers to BibDesk
  - Started researching potential word processors (choice will most likely be Scrivener)
  - Started first draft of Project Proposal

- **28/09/2013**
  - Completed first draft of Project proposal
  - Bugfixing on NaiveBayesClassifier (Issue 14: `https://bitbucket.org/tttthomasssss/finalyearproject/issue/14/naivebayesclassifier-sparse-matrices`)
  - First steps to reproduce Lucas, Downey (2013) results on Reuters ApteMod corpus
  - More prototyping on the Feature Marginals algorithm

- **30/09/2013**
  - Meeting with Supervisor
  - Presented Project Proposal + Feedback

- **01/10/2013**
  - Discovered Bug in Feature Marginals algorithm, trying to fix it

- **04/10/2013**
  - Still trying to figure out whats wrong with Feature Marginals algorithm
  - And debugging the EM algorithm

- **05/10/2013**
  - Tirelessly trying to figure out whats wrong with the Feature Marginals algorithm
  - Fixing the EM algorithm
  - For now, EM improves Precision, but messes up Recall (and therefore F-Score as well):
  - Normal Naive Bayes classifier: Precision=0.833, Recall=0.112, F-Score=0.197
  - EM Naive Bayes classifier: Precision=0.917, Recall=0.061, 0.115

- **06/10/2013**
  - Further fixing the EM algorithm
  - Implemented first versions of EM multiple mixtures (see Nigam, et al. 2000) and EM Priors only (see Settles 2011)

- **09/10/2013**
  - Wrote email to Lucas, Downey who authored the Scaling Semi-Supervised Naive Bayes with Feature Marginals paper
  - Few more tries to arrive at the results Lucas, Downey 2013 give in their example
- **10/10/2013**
  - More refining/fixing on the bloody EM algorithm
  - Actually implementing the feature marginals method in the algorithms module (well, started)
  - Reply to Lucas, Downey
- **11/10/2013**
  - More work on feature marginals implementation, specifically, normalising the probabilities
- **12/10/2013**
  - Finally managed to normalise the probabilities
- **13/10/2013**
  - Empty shell for sfe (Semi-supervised frequency estimate, see Su et al. 2011) prototype
  - combining the Feature Marginals with the Naive Bayes Classifier
  - Scrivener Tutorial for the Dissertation
- **14/10/2013**
  - Started refactoring the preprocessing pipeline
- **15/10/2013**
  - Finished refactoring of the preprocessing pipeline
  - Added EM algorithm, only using the class conditional log-likelihoods (and re-setting the priors)
  - Started work on script to run preliminary experiments for the interim report
- **16/10/2013**
  - First full run of interim report script (taking ages and ages and ages)

- **17/10/2013**
  - Collecting & evaluating results of first test run
  - Started plotting the results in relation to one another
- **20/10/2013**
  - Reply by Michael Lucas
  - Reply TO Michael Lucas
  - Finishing the first plots + pasting them into document
  - There is some weirdness with Settles' EM algorithm...
- **21/10/2013**
  - Working on Interim Report
- **22/10/2013**
  - More work on interim report...finally getting somewhere
- **23/10/2013**
  - More and more and more work on the interim report
  - Getting into LaTex a bit
- **24/10/2013**
  - Refined the formulas used in the interim report
  - "Implementing" EM in the interim report :D
  - Started adding SFE stuff to the interim report
  - First refinements on the whole thing (=interim report)
- **25/10/2013**
  - Reply from MLucas (implementation info on which root finding algorithm they used, including their starting values)
  - Yet more work for the Interim report
- **26/10/2013**
  - Finished first version of interim report
  - Formatting hell...Scrivener is nice & powerful but getting into the compile and formatting business IS A HUGE PAIN!!!!

- **30/10/2013**
  - Added Professional considerations, Project Plan and and the autumn term timetable to the interim report
  - I say, the Interim Report is now in status: FINAL DRAFT #yay
- **31/10/2013**
  - More formatting on the final draft of the interim report
  - Wrote Project Log Summary for the Interim report
- **01/11/2013**
  - Fine tuning of the interim report
- **04/11/2013**
  - Final polishing for the interim report
- **06/11/2013**
  - Working on the SFE algorithm
- **07/11/2013**
  - Intro to the new Java version of DUALIST by Andy
  - Implemented Feedback by MLucas in a first version
- **08/11/2013**
  - First working implementation of SFE
- **09/11/2013**
  - Trying combination of 1-step EM + FM
  - Python introspection/reflection
- **11/11/2013**
  - Found conceptual error in SFE algorithm
- **12/11/2013**
  - Lots of reworking and fixing on the SFE algorithm
- **15/11/2013**

  - More work on fixing the SFE algorithm

- **16/11/2013**
  - Even more fixing on the SFE algorithm...

- **18/11/2013**
  - Started work on CorpusStatistics, SampleStatistics & Experiment classes

- **24/11/2013**
  - More work on the Experiment class
  - Implemented the balance_class_skew algorithm as a prototype

- **25/11/2013**
  - More work on the balance_class_skew business

- **26/11/2013**
  - More work on the Experiment class

- **28/11/2013**
  - Started prototyping the Bernoulli Model

- **01/12/2013**
  - Downloading Datasets from TAG Lab DB

- **02/12/2013**
  - More Data from TAG Lab DB

- **05/12/2013**
  - More work on the Bernoulli Model

- **06/12/2013**
  - Working on preprocessing the tweets from the TAG Lab

- **09/12/2013**
  - More Twitter Preprocessing from the TAG Lab
  - Implemented k folds cross splitting among gold standard data

- **11/12/2013**
  - Work on the Preprocessing Pipeline (Twitter)
  - Work on the ClassifierProxy class
- **12/12/2013**
  - Reading Papers on Feature Selection, Smoothing Techniques
- **13/12/2013**
  - More Papers, Feature Priors, Active Learning
- **16/12/2013**
  - Yet even more papers, learning from labelled & unlabelled data, smoothing study
- **17/12/2013**
  - More experiments with the Twitter Data: ie. Uniform Priors, TF-IDF, Mashed k-fold cross validation
- **18/12/2013**
  - Refined the experiments Normal Priors vs. Uniform Priors on the Twitter data a little more
- **19/12/2013**
  - More Experiments on Twitter Data (EU Relevancy; NB vs. NB + FM)
  - Created Experiment Tracker SQLite DB Prototype
- **21/12/2013**
  - More work on the Experiment Tracker SQLite DB
- **22/12/2013**
  - Finally more work on the Twitter Preprocessing Pipeline (TwitterCorpusReader)
- **30/12/2013**
  - Cache serialisation of TwitterCorpusReader
  - More Research, Heckerman 1995

- **31/12/2013**
  - Bugfixing the TwitterCorpusReader
  - Finding out, a modestly large COO Matrix (139629 x 112971) is too much for my MacBook...lots of python crashes...2013 is a lot of fun...
- **1/1/2014**
  - More Bugfixing the TwitterCorpusReader
  - Preprocessing the Twitter Corpora (That takes a while, due to the fact that the matrices are largish)
- **6/1/2014**
  - Research on Performance Measures (Accuracy vs. Precision vs. Recall vs. AUC)
  - Some basic structure for the final report + plans for the appendices
- **15/1/2014**
  - Twitter Preprocessing Pipeline test
  - Experiment Dump Stuff
- **16/1/2014**
  - More work on k fold cross validation pipeline
- **17/1/2014**
  - Some kind of Dualist prototyping
  - Current system is a simple MNB classifier which labels 10 instances per iteration (based on posterior class entropy)
- **18/1/2014**
  - Fixed bug in calculation of the posterior class entropy
- **19/1/2014**
  - Research: Forman, Scholz 2009, Sokolova, Japkowicz, Szpakowicz 2006
- **20/1/2014**
  - Implemented cross validated scores according to Forman, Scholz 2009
  - Fixed Bug in metrics calculation when using sklearn.metrics

  - Added MNB Feature Marginals to Dualist Test Setup
- **21/1/2014**
  - Little bit of research on the AUC curve metric
- **23/1/2014**
  - Started implementing the Feature Marginals algorithm in Java
- **24/1/2014**
  - First Basic steps for the Feature Marginals algorithm in Java
- **28/1/2014**
  - More work on Feature Marginals algorithm in Java
  - Research on Java Optimisation libraries (jython vs. Apache Commons NewtonRaphsonSolver)
- **30/1/2014**
  - Starting work with the Apache Commons business
- **31/1/2014**
  - More prototyping work on the Apache commons stuff
- **1/2/2014**
  - More Apache Commons Math, but this time I have something that appears to be working
  - First draft of NewtonRaphsonSolver implemented
- **3/2/2014**
  - Reverse Engineering Andy's classifier
- **5/2/2014**
  - First Testrun of a Java classifier + Feature Marginals, but found errors
- **6/2/2014**
  - Starting to compare the Java Version with the Python Version via Jython
- **8/2/2014**

  - Jython :(
  - Bughunting

- **9/2/2014**
  - Fixed the Bug(s) in the Java NewtonRaphsonSolver #yay!

- **22/2/2014**
  - Started implementation of SFE in Java
  - Started Cleanup of FM in Java

- **23/2/2014**
  - More work on the Java versions of FM & SFE

- **24/2/2014**
  - Cleanup of Java FM
  - Started drafting out most-surely uncertain vs least surely uncertain in python

- **26/2/2014**
  - Messing around with the Java version of EM
  - Bugfixing the Python version of EM
  - So much EM

- **6/3/2014**
  - Combining the different Algorithms (EM + SFE; EM + FM; ...)
  - Started implementing Sharma, Birgic 2013 (Most-surely uncertain vs. least-surely uncertain)

- **7/3/2014**
  - First version of implementation of Most-surely uncertain vs least-surely uncertain
  - More Experiments with most-surely uncertain vs. least surely uncertain and under sampling the majority class
  - Combining every algorithm with every other algorithm in every order

- **10/3/2014**

  - Initial Draft of slides for presentation
- **11/3/2014**
  - Initial Final Draft of slides for presentation :P
- **14/3/2014**
  - Posterdesign stuff
- **15/3/2014**
  - More Poster stuff
- **16/3/2014**
  - Yet more Poster stuff
- **18/3/2014**
  - Finalising some of the Poster graphics
- **19/3/2014**
  - The hopefully final version of the Poster
  - Some Twitter Experiments
- **24/3/2014**
  - Preprocessing some more Twitter datasets
  - Starting with another FM-SFE comination algorithm
- **28/3/2014**
  - Pre-, pre-, preprocessing Twitter data-, data-, datasets
- **29/3/2014**
  - More and more and more preprocessing of the Twitter datasets
- **15/4/2014**
  - Starting to complete the main set of experiments...
- **16/4/2014 - 9/5/2014**
  - Running main set of Experiments

– Evaluating/Analysing Experiments
– Finalising Report
– LaTexifying Report
– 9:40am on 9/5/2014: DONE DONE DONE!!!!!!!!