COMP 551 Machine Learning, MiniProject 2

Sebastian Arenas, Lia Formenti, Trevor Shillington

March 13, 2020

Abstract

We tested different text classification algorithms on two classic datasets: Twenty Newsgroups [1], a topic classification task, and the Large Movie Review dataset [2], a sentiment classification task. The documents were transformed into bags of words with term frequency - inverse document frequency (tfidf) weighting. Before the implementation of the voting classifier, our results show that SVCs perform best on both datasets, with accuracies of 0.9066 on the Large Movie Review dataset and 0.7046 on the Twenty Newsgroups dataset, which is consistent with the literature. Given that the literature also put a large emphasis on feature design and selection, some tuning of the tfidf transformation was applied to our SVC and Adaboost model, which increased their accuracies by a few percent each. In addition, a voting classifier was implemented with the top three models for each dataset. The voting classifier yielded accuracies of 0.9012 on the Large Movie Review dataset and 0.7053 on Twenty Newsgroups dataset, a slight improvement. In addition, both AdaBoost and random forest performed better than their base, decision trees. Toghether, this shows that ensembles are better or as good as their constituents.

1 Introduction

Text classification has become a hot topic due to the explosive growth of digital documents available online. There are two main types: topic classification (for example, in database searching) and sentiment analysis (for example, gauging and predicting the public's reaction to new products) [3]. The difference is that while keyword matching may be sufficient for topic identification, sentiment analysis requires more nuance since the semantics matter more [4]. The goal of this project is to compare the predictive power of commonly used algorithms for both types of text classification. Two classic datasets, Twenty Newsgroups (hereafter 20NG)(topical) [1] and Large Movie Review (hereafter IMDb)(sentimental) [2] were used to train seven models from Python's Scikit Learn (SKL) library [5].

The main focus of text classification research is feature selection [3]. The first step is always to transform a corpus (collection) of documents into a bag of words, where each document is an instance, each word in the corpus is a feature, and the feature value for a given instance is the number of times that word appeared in that document. The most tried and statistically motivated second step is to transform the features by the term frequency - inverse document frequency (tfidf) method [6]. Tfidf starts with normalizing instances by the document length, which adjusts for words appearing more often in longer documents. It then normalizes features by their inverse document frequencies (idfs), which takes into account that words appearing very frequently in the corpus are likely not informative (e.g. "the"). Better performance can be obtained in both cases with further feature selection and processing, particularly for sentiment classification. For example, Narayanan *et. al.* [7] show that accuracy on the IMDb dataset can be increased by using combination of two words (bigrams) as well as single words (unigrams) as features. However, Srivasatava *et. al.* [8] show that the gains are less significant when using large datasets like the IMDb dataset, since their feature matrices are less sparse which makes each feature more informative [3]. Another strategy is to reduce the number of features overall by only keeping those that offer above a certain information gain on the problem, which is shown to be the most effective feature selection method on the IMDb [7] and 20NG [9] datasets. In this report, we focus more on the hyperparameter tuning of the models, which is less documented; however, for the most promising models we consider some feature design and selection alternatives.

Six models were studied: naive bayes (used as a baseline without further tuning), logistic regression, support vector classifiers (SVCs), decision trees, random forest, and AdaBoost on tfidf-transformed bags of words. In previous works, SVCs and naive bayes consistently performed competitively, even across different feature designs[10][4][8], and our results reflect this. Given that AdaBoost, decision trees, and random forests are computationally expensive than the successful SVC and naive bayes models, they have been explored less. However, Yasen and Tedmoni [11] showed that with information-gain based feature selection random forests were best for sentiment classification of a smaller IMDb movie review dataset, and decision trees competitive. Additionally, Bramesh and Kumar [12] showed that decision trees have comparable accuracy to naive bayes on the 20NG dataset with tfidf transformed features, though this did not hold for our results. Our results showed that random forest performed better than decision trees on both datasets, but SVC was the best single model for 20NG and IMDb.

Combinations of different classifiers have proven to perform even better than their constituent parts in the literature [8][13]. To explore this, we used the output from all our models to vote for the classification. Voting had the best accuracy for the 20NG dataset, 0.7053, and trailed behind the SVC accuracy on the IMDb dataset, 0.90664, by less than 0.004. In general, the ensembles performed as well or better than their constituent parts, but at the cost of increased training time.

2 Datasets

Both corpuses were transformed into a bag of words using SKL's CountVectorizer, and put through SKL's TfidfTransformer. For most models, the SKL default parameters for these classes were kept. Naive Bayes was used as a benchmark accuracy for both sets since it did not require tuning.

2.1 Large Movie Review Dataset

The IMDb dataset [2] is a collection of text files containing polar movie reviews from IMDb. There are 25000 very positive and 25000 very negative review instances split evenly into a 25000 instance train and 25000 instance test set. The data was loaded into a bunch using SKL's *loadfiles* function, and the transformations listed above applied. The goal was to train classifiers to predict whether a movie review was positive or negative. Multinomial Naive Bayes yielded a test accuracy of 0.82956.

2.2 Twenty Newsgroups

The 20NG dataset contains almost 19000 newsgroup documents, divided into the twenty newsgroups approximately evenly. The even distribution, shown in Appendix B, meant that we could split the data as presented without fear of under-representing any group [14]. The various topics spanned a wide range, from cars to religion to computers. However, there were some topics (e.g. autos and motorcycles) that had more crossover than other topics (e.g. compute graphics and atheism).

The training and testing sets were divided based on a date cut-off, with 60% in training and 40% in testing. The data was loaded directly via SKL's built-in $fetch$ function. We removed headers, footers, and quotes from the dataset

(training and testing) and processed the data as stated above. The goal was to train classifiers to predict which of the 20 newsgroups an article belonged to. Multinomial Naive Bayes has 60.6% testing accuracy on the 20NG test dataset.

3 Methods

Five different classification methods were trained and tuned on the two datasets to determine their predictive performance. The validation accuracy (five fold cross-validated accuracy for logistic regression and SVCs and accuracy on a 20 % validation set for all other models) was reported for the default SKL parameters in appendix A. For each model on each dataset a grid search with five-fold cross validation was used to tune the different hyper-parameters, with the best selections and their corresponding accuracies reported in table 1. Accuracy was used as the evaluation metric since in both datasets the distribution of classes was approximately even and the cost of misclassification was equal for both categories. The test data were not used until all of the hyper-parameters were tuned and features chosen.

4 Models

For all of the following models, we used the built-in SKL implementations.

4.1 Logistic Regression

Logistic regression applies the principles of linear regression with a sigmoid activation function. For binary data the predicted class is chosen by thresholding the output at 0.5, while for multiclass classification the maximum probability class is predicted.

Three parameters were tuned for the IMDb dataset: the solver, the penalty, and the inverse regularization strength (C). The penalty refers to whether $L2$, $L1$ or no regularization was applied, and C set the degree of regularization (with a larger value indicating less regularization). A brief description of all solvers and references for more information can be found in appendix B. Not all solvers supported all penalties; sequential grid searches were applied to the sensible combinations of penalties and solvers with a roughly logarithmically uniform distribution of C between $0.001 \leq C \leq 10$. The mean cross validation accuracy was used to decide which combination worked best. For that combination, a finer grid search on C was run.

On the IMDb dataset, the $L2$ penalty was the best for all solvers. There was no difference in cross-validated accuracy between sovlers, so the dual LIBLINEAR solver was chosen because it was fastest. Further tuning of the regularization parameter revealed that $C = 8$ was best for the IMDb dataset, with a test accuracy of 0.8822.

For the 20 Newsgroups dataset, since the solver did not affect the IMDb results, a single solver was tested. This time, the Limited-memory BFGS ($LBFGS$) solver was chosen because it was capable of doing a true multiclass classification, whereas LIBLINEAR could only do a "one-versus-rest" scheme. The $L2$ regularization proved best once again, and the regularization parameter was tuned to $C = 17$, with a test accuracy of 0.6827.

The decision to stop tuning the regularization parameter was made when a maximum in validation accuracy was reached in C-parameter space, as shown in appendix D.

4.2 Support Vector Machines

The principle of SVCs is to find the decision boundary that maximizes the margin between the decision boundary and the instance(s) closest to it. SKL's LinearSVC finds a linear decision boundary using the LIBLINEAR solver [15]. Either the primary [16] or dual algorithm [17] (see appendix A) can be used, and Hsiesh *et al.*'s paper indicates that the dual algorithm mostly reduces the train time of the primary.

The important parameters of LinearSVC were: whether the primary or dual LIBLINEAR solver was used ($dual = True, False$); which loss function was used ($loss = hinge, squaredhinge$); what regularization scheme was applied ($penalty = l2, l1$); and what inverse regularization strength to use (C). The procedure for the successive grid searches on LinearSVC was the same as for LogisticRegression. On the IMDb dataset, the best accuracy was obtained with $L2$ regularization with the dual LIBLINEAR solver and a squared hinge loss function. With a C of 0.55, the cross-validated accuracy was 0.8954. $C \leq 1.0$ shows that heavier regularization was preferred, and similarly the squared hinge loss function penalizes individual losses more heavily.

The same method was applied to the 20 Newsgroups dataset. The best hyper-parameters were the same except for $C = 0.6$, with a cross-validated accuracy of 0.7647.

Given the prevalence of feature design and selection studies [3] and the traditional success of SVCs [10], a new grid search on the feature design / selection parameters of CountVectorizer and TfidfTransformer - $ngramrange$, $maxdf$, and $sublinear tf$ - was run with a $L2$ penalty and the previously successful $squaredhinge$ loss function. $ngramrange$ determines what combination of words are counted as features: using only unigrams or unigrams and bigrams. max_df puts an upper limit on the document frequency of terms used as features, which has the effect of feature reduction like the gain-based feature reduction done in other works [9][7]. $sublinear\_tf$ scales words as $1+log(tf)$, where tf is term frequency per document, with the idea that just because a term appears X more times than another does not necessarily mean it carries X times the weight. For the IMDb dataset, the cross-validated accuracy increased to 0.9111 when using unigrams and bigrams with sublinear scaling and $maxdf = 0.2$ ($C = 1.0$),

cutting lots of features. This is consistent with Narayanan *et. al.*'s result that including bigrams increased their test accuracy on the IMDb test set. Note that although max_df is low, the new number of features is 1513704 compared to 74849, since bigrams adds so many more features. The cross validated accuracy on 20NG with the same parameters (except $C = 20$) was 0.76198. This was in fact a slight decrease compared to above, but the grid search included the previous parameter space so this was taken as the best combination. The reduced change in accuracy for 20NG makes sense given Pang *et. al.*'s observation that for topical classification keyword matching is sufficient; the main advantage in bigrams is that they take into account negations and semantics better, which matters more for sentiment classification.

With the feature selection, the test accuracies were 0.9066 and 0.7048 for the IMDb and 20NG, respectively.

4.3 Decision Trees

This model performs a partition of the feature space into a set of rectangles to fit a simple model in each of the new spaces [18]. This algorithm is know for being powerful robust, efficient to construct and capable of fitting complex datasets. However, this model tends to over-fit (as seen in appendix A). Pruning and stopping criteria were implemented to improve the model [19][20].

For growing the tree with p inputs and a response, for each of N observations, the algorithm partitions into M regions $R_1, R_2, ..., R_M$, in order to model the prediction (as was explained before) as constant c_m for each region: $\hat{f}(X) = \sum_{m=1}^{M} C_m I[x \in R_m]$.

The first hyperparameter analyzed was the maximum depth of the tree. Deeper trees have more splits and consequently more information captured from the data. The default parameter 'None', usually generates over-fitted decision trees. Tuning the maximum depth performs **'regularization'** on the tree by limiting how much it grows. The best values were 100 for the IMDb dataset and 3000 for the 20NG dataset. Setting minimum samples split also regularizes the tree by only considering splits that result in nodes with populations greater than the minimum. 200 was chosen for IMDb and 300 for 20NG. The maximum number of features to be considrered in potential splits was also considered as a regularization method for this model; however since the best split was chosen, there was no point in changing the default value (all the features to be consider in each split). Maximum depth, minimum samples split, and minimum samples leaf can be were used as the stopping criteria of the algorithm.

The quality of the splitting point for root/decision nodes could be measured by the entropy, the disorder of the classification ($-\sum_j p_j^2 log_2 p_j$) or the Gini impurity ($1-\sum_j p_j^2$), the measure of how often a randomly chosen element from the set would be incorrectly labelled. The grid search revealed that choosing entropy generates slightly higher accuracy results but the difference between both in the performance of the algorithm is not significant enough to outweigh the extra computational cost of the logarithm in the entropy [21]. Additionally, the 'best' way to split was chosen with the splitter hyper-parameter and in that way the algorithm considers all the features and chooses the best split always.

For pruning and to avoid over-fitting, the ccp alpha hyper-parameter or cost-complexity ($R_\alpha(T) = R(T) + \alpha|T|$, with $|T|$ as amount of terminal nodes and R(T) as the total misclassification rate of the terminal nodes) was tuned (SKL's default does not even consider this). The of 000.1 and 0.00001 were found for each of the IMDb and 20NG datasets respectively. Therefore, the sub-tree with the largest cost complexity that is smaller than 0.0001 in the case of IMDb dataset or smaller than 0.00001 in the case of the 20NG dataset is then chosen every split[18].

After varying the different hyper-parameters, it was noticed that a hard stopping criterion yields to an underfitted model. On the other hand, a soft stopping criteria produces a over-fitted model and that is why the pruning methods were implemented. Finally, by implementing the model as was previously described, the train scores decreased but the difference between train and test scores also decreased.

4.4 Random Forest

A single decision tree is not a very powerful predictor as can be seen in table 1 where the accuracy of the previously models are around 0.71 and 0.43. Consequently, an ensemble classifier like Random Forest that can capture complex interaction structures in the data was implemented, using L tree-structured base classifiers (decision trees) $h(x, \theta_k), k = 1, ...L$ where θ_k represents a set of independent identically distributed (iid) random vectors, and x an specific data input. For this model, each decision tree was built from a random vector of parameters (result from randomly sampling a feature subset for each decision tree considered).[22][18]

The best hyperparameters from the grid search had similar values to those gathered for decision trees. However, a new hyper-parameter, the number of estimators or trees was also tuned. The initial value of estimators to use is 100, but the grid searched revealed that the accuracy increases with more trees until a saturation point. After 400 for IMDb and 700 estimators for 20NG the accuracy did not change. Additional estimators (more than 400 or more than 700 depending to the dataset) either added or maintained the accuracy to the model, but never decreased it.

According to [22], the amount of features considered in each split controls the strength of the randomization in the split selection, in such a way that the smaller the value of K, the stronger the randomization. Therefore, for random forest the hyper-parameter corresponding to the maximum features was also tuned.

4.5 AdaBoost

AdaBoost combines many weak learners (decision stumps, for example) to create a powerful classifier [18]. The weak learners are 'adaptive' because each one is given different weights depending on the results of the previous weak learner. When a previous weak learner has an incorrect prediction, that prediction is given a higher weight in the weak learner that follows. The result is that the weak learners have minimal correlation between their errors, so that combining them and using democratic voting will produce a strong learner[18].

The SKL AdaBoost class takes as the main parameters: base_estimator, n_estimators, learning_rate, and algorithm. base_estimator is the estimator from which the boosted ensemble is built, the default being decision stumps. n_estimators is the maximum number of iterations at which boosting is terminated (default = 50). As this value increases, the accuracy increases but there are diminishing returns as you go higher and higher. This also increases the computational time needed (roughly linearly), so there is a trade-off. learning_rate reduces the contribution of each classifier (default=1). n_estimators and learning_rate tend to have a trade-off between them: increasing n_estimators and decreasing learning_rate can cancel the effect [23]. Lastly, the algorithm can be SAMME or SAMME.R (default=SAMME.R), the former being the discrete boosting algorithm and the latter being the real boosting algorithm.

In our SKL pipeline, we utilize CountVectorizer and TfidfTransformer, which in turn also have hyper-parameters. Due to computational restrictions, we did not tune all of the possible hyper-parameters. We chose the ones that we thought would most impact the accuracy, and left the rest as default. For CountVectorizer, we wish to tune max_features (which puts a cap on the number of features to use (default=None)), and ngram_range (which determines whether we want to implement unigrams and bigrams (default=unigrams). Another parameter, stop_words was set to the non-default 'english', which removes the most common English words. Since these words commonly occur in all instances, they tend not to be very useful for classification. It reduces the number of features which is important for run times, especially for AdaBoost. For TfidfTransformer, we wish to tune use_idf (which enables inverse-document-frequency re-weighting (default=True)), and sublinear_tf which replaces tf with 1 + log(tf) (default=False).

For the tuning of all of these hyper-parameters, we had to take into consideration computational resources when running grid search (using SKL's "Sample pipeline for text feature extraction and evaluation" as a guideline). For this reason, we first ran grid search for just the AdaBoost hyper-parameters. This was ran multiple times as we tried to find an ideal set of values to search over. For base_estimator we only used DecisionTreeClassifier but tuned the max_depth as either 1 or 2. The results for the two values were comparable, however having a max_depth=1 was much less computationally exhaustive, and so we used that. Once all the ideal values were found (see Table 1), we ran grid search again for the hyper-parameters of CountVectorizer and TfidfTransformer with the tuned AdaBoost hyper-parameters set. As mentioned, n_estimators increases run time roughly linearly. We found that n_estimators = 10,000 gave the best results for the 20NG Dataset, however it also required a long run time. In order to minimize the grid search run time, we used a lower value of n_estimators in the grid search for CountVectorizer and TfidfTransformer, and made the educated guess that the tuned values found would equally apply to a larger value of n_estimators.

4.6 Ensemble - Voting Classifier

As is explained in [24][13][25], a typical way to build a better classifier is to combine the predictions of each classifier and predict the most voted class. In other words, the ensemble voting algorithm gather the input from the classifiers and select a class label that has been chosen by the output of most classifiers. This majority-vote classifier is called a hard voting classifier [26]. (see appendix E)

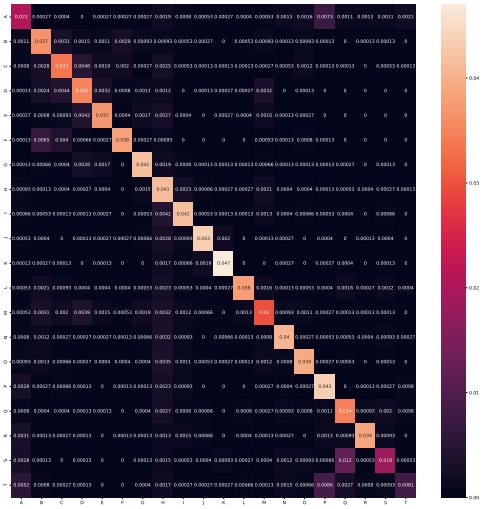


Figure 1: Confusion Matrix for the ensemble voting classifier (Normalized). X-Axis: Predicted Labels, Y-Axis: True Labels

In most of the cases, this hard voting classifier achieves a higher accuracy compared to the best classifier in the ensemble and even when each classifier is a weak learner for a given a sufficient amount of the weak learners and enough diversity.

For this report, hard voting was implemented as a hyper-parameter. Soft voting was not implemented since this variant demands that all classifiers must be able to es-

Dataset	Model	Hyper-parameters	Training Accuracy	Test Accuracy	Training Runtime (s)
Large Movie Review	**Logistic Regression**	**penalty** =l2, **C**=8, **solver**=liblinear	0.9858	0.8822	4.92
	Decision Trees	**ccp alpha** =0.0001, **criterion** ='entropy', **max depth**=100, **min samples leaf**=10, **splitter**='random', **min samples split**=300, **max leaf nodes**= 200, **max features**=100	0.7565	0.7530	9.5
	SVC	**penalty** ='l2', **loss**='squared_hinge', **dual**=True, **C**=1.0, **max_df**=0.2, **ngram_range**=(1,2), **sublinear_tf**=True	0.9999	**0.9066***	17.63
	Random Forest	**n. estimators**= 400, **ccp alpha**= 0.0001, **criterion** = 'entropy', **max depth** = 100, **min samples leaf**= 10, **min samples split** =400 , **max leaf nodes** = 300	0.9054	0.8483	12.27
	AdaBoost	**base_estimator** = DecisionTreeClassifier(max_depth=1), **n_estimators** = 1250, **learning_rate** = 0.5, **algorithm** = SAMME.R, max_df = 0.5	0.9423	0.86516	342.48
	Voting	**estimators** =LR,RF,SVC, **voting** ='hard'	0.9822	0.9012	326.91
Twenty Newsgroups	**Logistic Regression**	**penalty** =l2, **C**=17, **solver**=lbfgs, **max iters**=1000	0.9745	0.6827	97.64
	Decision Trees	**ccp alpha** =0.00001, **criterion** ='Gini', **max depth**=3000, **min samples leaf**=1, **splitter**='best',**min samples split**=200, **max leaf nodes**= 300	0.5474	0.4395	6.75
	SVC	**penalty** ='l2', **loss**='squared_hinge', **dual**=True, **C**=20.0, **max_df**=0.2, **ngram_range**=(1,2), **sublinear_tf**=True	0.9749	0.7049	145.40
	Random Forest	**n. estimators**= 700 ,**ccp alpha**= 0.00001, **criterion** = 'Gini', **max depth** = 500, **min samples leaf**= 1,**min samples split** = 60, **max leaf nodes** = 400	0.8266	0.6502	51.48
	AdaBoost	**base_estimator** = DecisionTreeClassifier(max_depth=1), **n_estimators** = 10000, **learning_rate** = 1, **algorithm** = 'SAMME', max_df = 0.5	0.6325	0.5345	1059.55
	Voting	**estimators** =LR,SVM,RF, **voting** ='hard'	0.9246	**0.7053***	189.50

Table 1: Accuracies and running times for each of the models with the tuned hyper-parameters. Values in bold and starred are the highest values for their respective dataset. Note that hyper-parameters column includes those for CountVectorizer and TfidfTransformer.

timate class probabilities that are not straightforward to compute for LinearSVC [26].

For the IMDb dataset, Logistic Regression, Support Vector Machines and AdaBoost were combined in an ensemble voting algorithm. Furthermore, for the 20NG, Logistic Regression, Support Vector Machines and Random Forest were combined. The other classifiers that were not used in both of the voting classifiers were discarded due to their low accuracy of classification. Most of the papers cited in this report have shown that classification algorithms reach better performance when they are combined. Table 1 shows that for both IMDb and 20NG datasets, ensemble voting classifier achieves indeed much higher accuracies compared to four out of five of the initial classifiers.

Figure 1 presents the confusion matrix corresponding to the 20NG dataset, where the alphabet letters represent the 20 class labels. It is possible to observe that the classes A,P,O,S and T are the ones in which the voting classifier is having problems to classify.

5 Results and Conclusions

Table 1 presents the training and testing accuracies of the six implemented classifiers with the best hyper-parameters combinations. SVC proved to be the best single classifier as predicted by the literature, leading against the other models by about 2 % for each dataset. However, for the 20NG dataset voting was the best model, and voting on the IMDb dataset trailed by less than 0.004. Ultimately, the highest achieved accuracies achieved were 0.9066 for the IMDb dataset by SVC and 0.7053 for the 20NG dataset by voting. Additionally, both AdaBoost and Random Forest performed better than their constituent decision trees. The success of these two models and voting shows the superiority of ensembles to single classifiers, as already shown in [8][13]. However, as shown in table 1, they took significantly longer to train and tune. On the same note implementing bigrams was computationally expensive and consequently it should only be used on models whose accuracy benefits greatly from them. The feature design and selection using bigrams, sub-linear term frequency scaling and maximum document frequency scaling SVC and maximum document frequency scaling for Adaboost, only increased the accuracies by a few percent, but the bigrams option for SVC brought its training time from comparable to that of logisitic regression up an order of magnitude.
Overall, our results show that SVC and ensembles perform well in both topical and sentimental text classification, as agrees with previous studies.

5.1 Contributions

Lia Formenti worked on data preprocessing for IMDb dataset, Logistic Regression, SVC and the report. Trevor Shillington worked on data preprocessing for 20NG dataset, Adaboost and the report. Sebastian Arenas worked on Decision Trees, Random Forest, Voting and the report. Each of the members coded the mentioned models, tuned their hyper-parameters and wrote the corresponding sections.

References

[1] Ken Lang. NewsWeeder: Learning to Filter Netnews. In Armand Prieditis and Stuart Russell, editors, *Machine Learning Proceedings 1995*, pages 331–339. Morgan Kaufmann, San Francisco (CA), January 1995.

[2] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[3] Robert A. Stine. Sentiment Analysis. *Annual Review of Statistics and Its Application*, 6(1):287–308, March 2019. Publisher: Annual Reviews.

[4] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics, July 2002.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[7] Vivek Narayanan, Ishan Arora, and Arjun Bhatia. Fast and Accurate Sentiment Classification Using an Enhanced Naive Bayes Model. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Hujun Yin, Ke Tang, Yang Gao, Frank Klawonn, Minho Lee, Thomas Weise, Bin Li, and Xin Yao, editors, *Intelligent Data Engineering and Automated Learning – IDEAL 2013*, volume 8206, pages 194–201. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. Series Title: Lecture Notes in Computer Science.

[8] Saurabh Kr. Srivasatava, Roshan Kumari, and Sandeep Kr. Singh. An ensemble based NLP feature assessment in binary classification. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 345–349, May 2017. ISSN: null.

[9] Anirban Dasgupta, Petros Drineas, Boulos Harb, Vanja Josifovski, and Michael W. Mahoney. Feature selection methods for text classification. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 230–239, San Jose, California, USA, August 2007. Association for Computing Machinery.

[10] Thorsten Joachims. Text categorization with Support Vector Machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98*, Lecture Notes in Computer Science, pages 137–142, Berlin, Heidelberg, 1998. Springer.

[11] Mais Yasen and Sara Tedmori. Movies reviews sentiment analysis and classification. 04 2019.

[12] S. M. Bramesh and K. M. Anil Kumar. Empirical Study to Evaluate the Performance of Classification Algorithms on Public Datasets. In V. Sridhar, M.C. Padma, and K.A. Radhakrishna Rao, editors, *Emerging Research in Electronics, Computer Science and Technology*, volume 545, pages 447–455. Springer Singapore, Singapore, 2019. Series Title: Lecture Notes in Electrical Engineering.

[13] Duoqian Miao, Qiguo Duan, Hongyun Zhang, and Na Jiao. Rough set based hybrid algorithm for text classification. *Expert Systems with Applications*, 36(5):9168–9174, 2009.

[14] Yuxi Liu. *Python Machine Learning By Example*. Packt, 2017.

[15] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A Library for Large Linear Classification. 9:1871–1874, 2008.

[16] Chih-Jen Lin, Ruby C Weng, and S Sathiya Keerthi. Trust Region Newton Method for Large-Scale Logistic Regression. 9:627–650, 2008.

[17] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 408–415, Helsinki, Finland, 2008. ACM Press.

[18] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009.

[19] Rafael G. Mantovani, Tomas Horvath, Ricardo Cerri, Joaquin Vanschoren, and Andre C.P.L.F. De Carvalho. Hyper-Parameter Tuning of a Decision Tree Induction Algorithm. *Proceedings - 2016 5th Brazilian Conference on Intelligent Systems, BRACIS 2016*, pages 37–42, 2017.

[20] Aurelien Geron. *Hands-On Machine Learing With Scikit-Learn & Tensor Flow*. O'Reilly Media, 2017.

[21] Reihaneh Rabbany. Lecture: Applied Machine Learning. Technical report, McGill University, 2020.

[22] Simon Bernard, Laurent Heutte ,Sebastien Adam. Influence of Hyperparameters on Random Forest Accuracy. 2009.

[23] Xavier Bourret Sicotte.

[24] A.F.R. Rahman, H. Alam and M.C. Fairhurst. *Multiple Classifier Combination for Character Recognition: Revisiting the Majority Voting System and Its Variations*, volume 1716. 1999.

[25] Saurabh Kr Srivasatava, Roshan Kumari, and Sandeep Kr Singh. An ensemble based NLP feature assessment in binary classification. *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2017*, 2017-January:345–349, 2017.

[26] Aurélien Géron. Hands-on machine learning with Scikit-Learn and TensorFlow concepts, tools, and techniques to build intelligentsystems. Paperback, April 2017.

[27] K Duan, S Keerthi, and A Poo. Evaluation of simple performance measures for tuning SVM hyper parameters. Technical report. *Neurocomputing*, 51:41–59, 2001.

[28] David Windridge, Norman Poh, Vadim Mottl, Alexander Tatarchuk, and Andrey Eliseyev. *Handling multimodal information fusion with missing observations using the neutral point substitution method*, volume 5519 LNCS. 2009.

[29] Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michèle Sebag. Collaborative hyperparameter tuning. *30th International Conference on Machine Learning, ICML 2013*, 28(PART 2):858–866, 2013.

[30] Balázs Kégl and Busa Fekete Róbert. Boosting products of base classifiers. *Proceedings of the 26th International Conference On Machine Learning, ICML 2009*, pages 497–504, 2009.

[31] Leah S. Larkey. Combining Classifiers in Text Categorization. *SIGIR*, 32546:289–297, 1996.

[32] Working With Text Data — scikit-learn 0.22.1 documentation.

[33] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, March 2017.

[34] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. *arXiv:1407.0202 [cs, math, stat]*, December 2014. arXiv: 1407.0202.

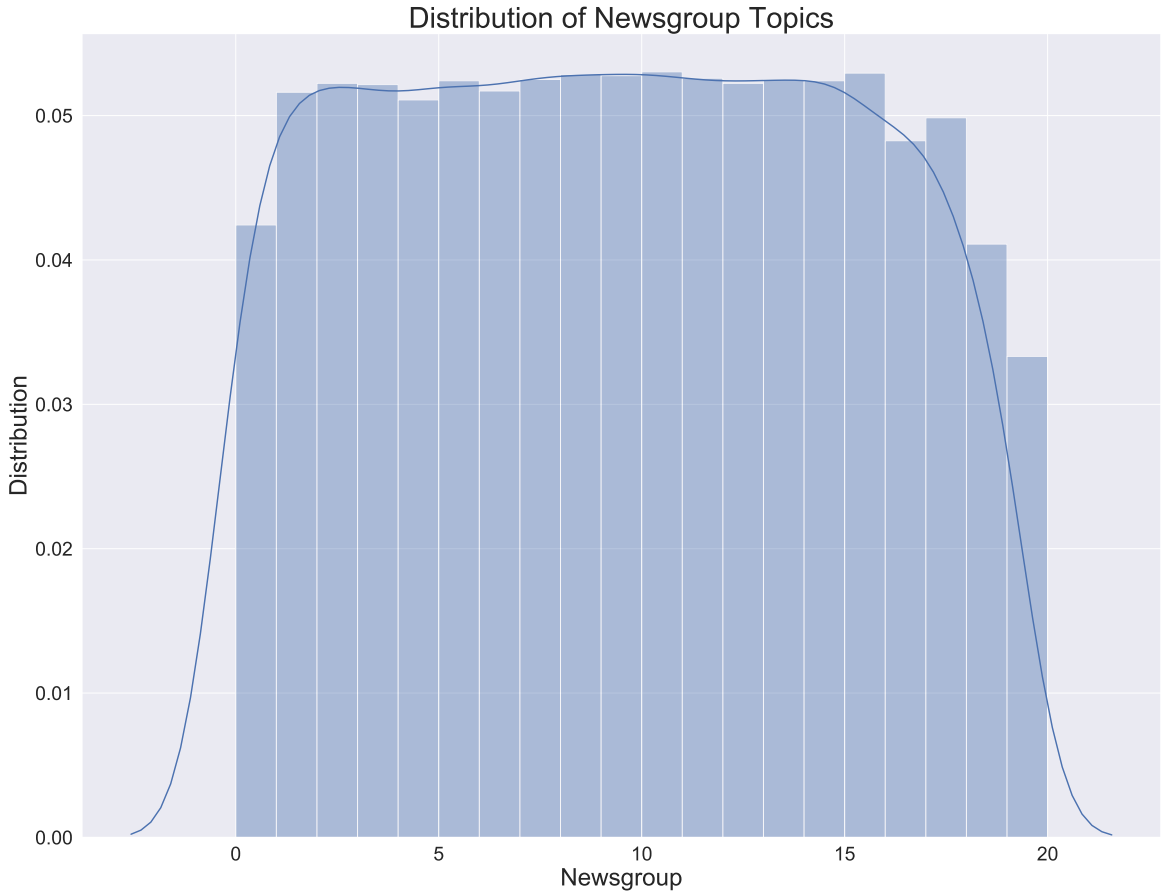
[35] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *proceedings of the 6th conference on Natural language learning - Volume 20*, COLING-02, pages 1–7, USA, August 2002. Association for Computational Linguistics.

Appendix A Accuracies with Default Hyper-parameters

Dataset	Model	Default Hyper-parameters	Training Accuracy	Validation Accuracy	Training Runtime (s)
Large Movie Review	**Logistic Regression**	**penalty** =l2, **C**=1, **solver**=lbfgs	0.9350	0.8866	5.50
	Decision Trees	**ccp alpha** =0.0007, **criterion** ='Gini', **max depth**=None, **min samples leaf**=1, **splitter**='best',**min samples split**=2, **max leaf nodes**= None	1	0.7115	53.06
	SVC	**penalty** ='l2', **loss**='squared_hinge', **dual**=True, **C**=1.0	0.9921	**0.8948***	3.95
	Random Forest	**n. estimators**= 400, **ccp alpha**= 0.0, **criterion** = 'Gini', **max depth** = None, **min samples leaf**= 1, **min samples split** =2 , **max leaf nodes** = None	1	0.8364	58.06
	AdaBoost	**base_estimator** = DecisionTreeClassifier(max_depth=1), **n_estimators** = 50, **learning_rate** = 1, **algorithm** = SAMME.R	0.81045	0.8072	25.28
Twenty Newsgroups	**Logistic Regression**	**penalty** =l2, **C**=1, **solver**=lbfgs, **max iter**=1000	0.9102	0.7248	36.39
	Decision Trees	**ccp alpha** =0.0007, **criterion** ='Gini', **max depth**=None, **min samples leaf**=1, **splitter**='best',**min samples split**=2, **max leaf nodes**= None	0.9751	0.4304	10.65
	SVC	**penalty** ='l2', **loss**='squared_hinge', **dual**=True, **C**=1.0	0.9735	**0.7610***	3.06
	Random Forest	**n. estimators**= 400 ,**ccp alpha**= 0.0, **criterion** = 'Gini', **max depth** = None, **min samples leaf**= 1,**min samples split** = 2, **max leaf nodes** = None	0.9751	0.6336	26.38
	AdaBoost	**base_estimator** = DecisionTreeClassifier(max_depth=1), **n_estimators** = 50, **learning_rate** = 1, **algorithm** = SAMME.R	0.4049	0.4079	8.71

Table 2: Accuracies and running times for each of the models with the default hyper-parameters. Values in bold and starred are the highest values for their respective dataset.

Appendix B Distribution of the topics between news group



Distribution of topics between newsgroups. They are roughly consistent meaning there aren't any over(under)-represented groups [14].

Appendix C Logistic Regression Solvers

A brief description of the solvers SKL's linear_models. LogisticRegression class that were tested.

Limited-memory BFGS (LBFGS) builds off of the Newton method, which is like gradient descent but informed by the Hessian matrix [18]. Instead of computing the Hessian as in the Newton method, it approximates it, making the computation faster [35].

SAG is like stochastic gradient descent where at each iteration the gradient of one instance is taken, but it averages the current iteration's instance's gradient with all previous gradients reported for all other instances [33]. SAGA builds on SAG by adapting to non-strongly convex cost functions [34].

We tested the primary and dual Liblinear solver. Liblinear uses coordinate descent (like single - coordinate gradient descent) to reach a minimum. [15][16][17]. SKL recommends the dual method when the number of features exceeds the number of samples, and indeed we found that the train time improves when the dual method is used over the primary.

Appendix D Tuning the inverse regularization strength

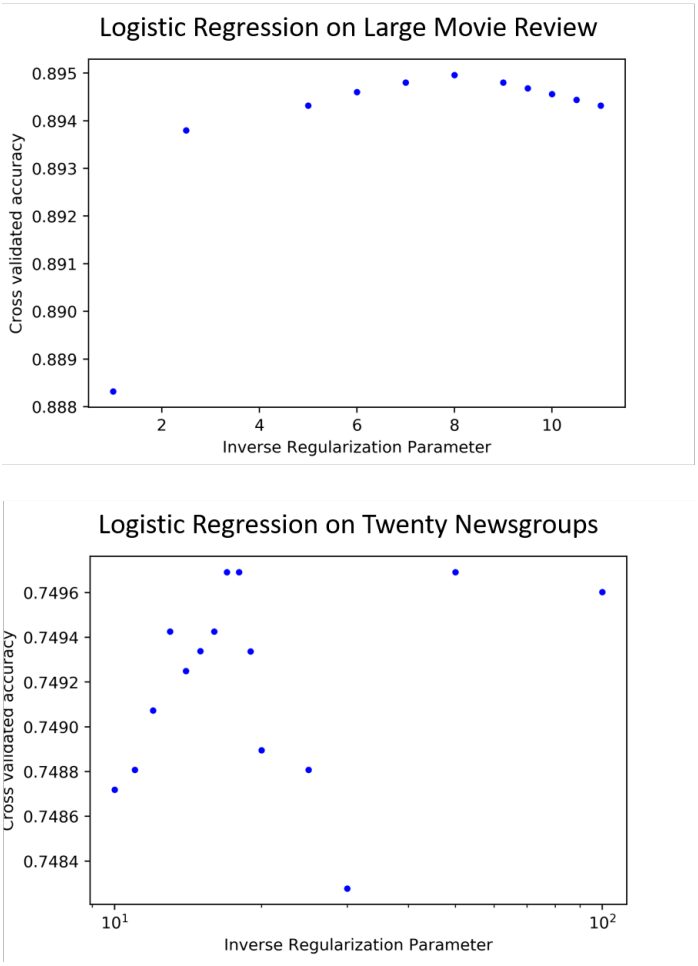
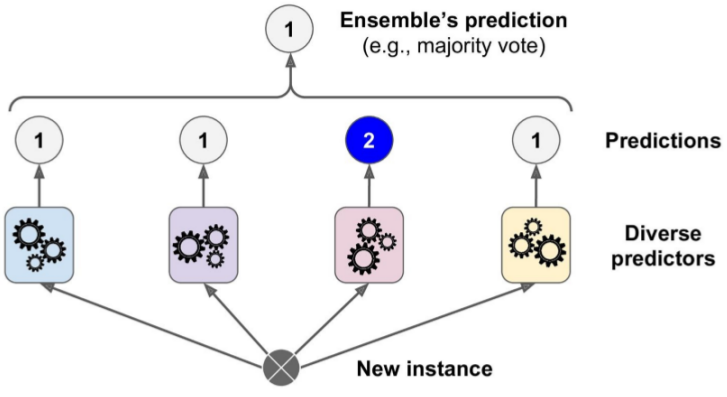


Figure 2: Fine tuning of the inverse regularization parameter (C), with all other hyper-parameters for each model tuned. In each figure a maximum is reached and the changes in validated accuracies for various parameters are small, so no further tuning was deemed necessary.

Appendix E Hard Voting



Hard voting classifier predictions [26]

Appendix F Confusion Matrix

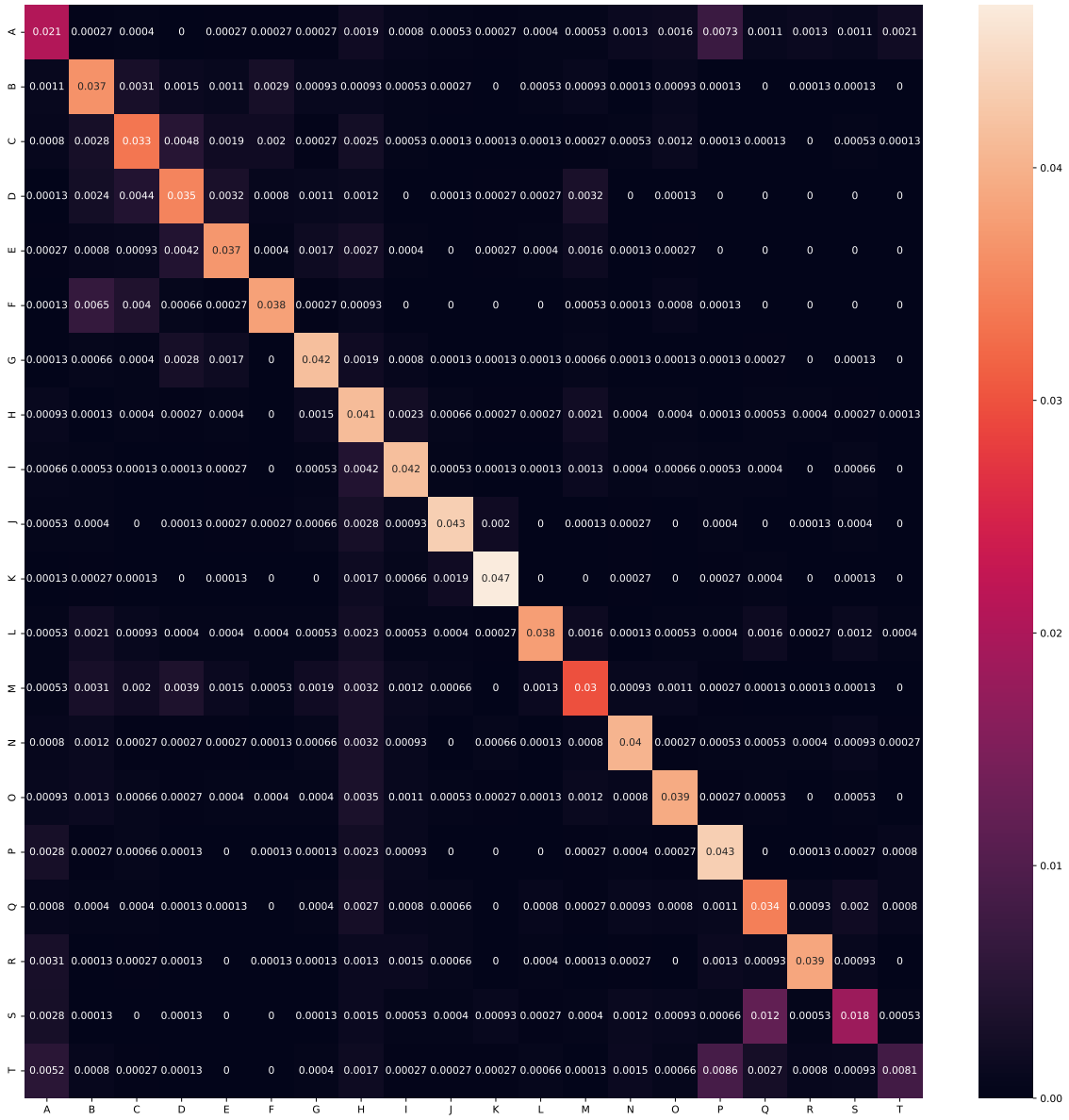


Figure 3: Confusion Matrix for the ensemble voting classifier (Normalized). X-Axis: Predicted Labels, Y-Axis: True Labels