Simpler is Better? Lexicon-Based Ensemble Sentiment Classification Beats Supervised Methods

Track 1: Improve the Baselines

Aryan Gupta
ADA
260807226
aryan.gupta@mail.mcgill.ca

Andrew Kam
ADA
260772154
andrew.kam@mail.mcgill.ca

Dylan Mann-Krzisnik
ADA
260637479
dylan.mann-krzisnik@mail.mcgill.ca

I. Introduction

The paper, *Simpler is Better? Lexicon-based Ensemble Sentiment Classification Beats Supervised Methods* [1] by Augustyniak et al. (2014), proposes that a simplistic Bag of Words (BoW) lexicon method using ensemble classifiers for sentiment polarity assignment performs faster than a more simplistic and supervised algorithm, while obtaining similar accuracy. Their ensemble approach involves lexicon-based BoW weak learners, used to provide input for a stronger decision tree based learner. A variety of lexicons, including simple word lists and ones based on word-frequency are used to establish a BoW model, then trained on a "strong classifier" such as a C4.5 Decision Tree.

Their baseline model for comparison was a C4.5 Decision Tree, used to classify the Amazon Reviews dataset, vectorized into a Binary BoW. Five categories in the reviews dataset were used: automotive, book, electronics, health, and movies. While the results showed that the execution time of the proposed lexicon approach was significantly faster than the supervised learning of the decision tree, the baseline model reported higher F-measure results for three of the five categories.

Based on these results, we believe that the decision tree classifier used as a baseline was not optimal in terms of performance, and rather chosen as a comparison for execution time. Furthermore, very little preprocessing was completed on the dataset. As such, their proposed lexicon-based approach was able to perform within range of their baseline. The purpose of our experiment is to prove that preprocessing the original dataset with a variety of simple techniques, as well as running common supervised classifiers with quick hyperparameter tuning, can easily outperform their baseline results by a large margin.

II. Design

A. Dataset Preparation

A subset of the Amazon Reviews dataset published by the Stanford Network Analysis Project (SNAP) was used by Augustyniak et al. in their experiment:

- Automotive (188,728 reviews)
- Book (12,886,488 reviews)
- Electronics (1,241,778 reviews)
- Health (428,781 reviews)
- Movies (7,850,072 reviews)

In order to correctly evaluate the baseline classifier of the authors, we sought out the original dataset and corresponding categories. However, as shown, there is a large discrepancy in the number of reviews between the categories. While the automotive domain contains a manageable and easily manipulative number of reviews, the book domain is extremely large in quantity. The large subsets proved to be an issue in our initial testing, as we would run into memory issues once attempting to load. This occurred with the three largest categories: book, electronics, and movies.

In order to decrease their sizes, the reviews in each category were split according to their rating (1-5), and then saved in their own data files. Following the methodology of the Augustyniak et al., reviews less than 100 characters were then filtered out, the remaining reviews shuffled, and then the size of each set limited to 1600 reviews for ratings 1, 2, 4, and 5, and 3200 reviews for rating 3. This produced a balanced dataset of 9600 reviews for each category. Ratings were then mapped to text classes "positive", "neutral", and "negative", using 1 and 2 stars, 3 stars, and 4 and 5 stars respectively, as done by the original authors. With these reduced datasets, we were able to load the reviews into memory without any problems.

B. Baseline Reproduction

The baseline classification used by Augustyniak et al. was a C4.5 Decision Tree, evaluated with 10-fold cross validation. C4.5, an extension of the ID3 algorithm, is a decision tree variant that uses the concept of information entropy. C4.5 also executes tree pruning to avoid overfitting, by replacing sub-trees with leaves if they reduce the classification error.

The decision tree algorithm in the ready-to-use `scikit-learn` library is Classification and Regression Trees (CART), which is similar to C4.5. It differs in that it supports numerical target variables and does not compute rule sets. Since this library is easily accessible and simple to plug into Python 3, it was used to mimic the baseline established with the C4.5 algorithm. Setting the criterion parameter to 'entropy' and tuning the tree depth and sample split with 10-fold cross validation produced the results shown in Table I.

Category	Hyper-Parameters	Mean Score	STD

Automobile	criterion = 'entropy'
	max_depth = 30
	min_samples_split = 140
		0.523	+/- 0.022

Book	criterion = 'entropy'
	max_depth = 20
	min_samples_split = 160
		0.468	+/- 0.037

Electronics	criterion = 'entropy'
	max_depth = 50
	min_samples_split = 160
		0.476	+/-0.032

Health	criterion = 'entropy'
	max_depth = 40
	min_samples_split = 70
		0.493	+/-0.022

Movies	criterion = 'entropy'
	max_depth = 20
	min_samples_split = 160
		0.481	+/-0.034

TABLE I: Optimal CART Decision Tree parameters

Using the hyperparameters from Table I, the CART Decision Tree classifier was used to predict the ratings of the reviews in each category. Once again, 10-fold cross validation was used, producing the performance results in Table II. As seen in Figure 1, 3 out of the 5 categories were within error of each other. The small differences in performance was most likely due to the random shuffling of the datasets and limiting each set to 9600 reviews, a relatively small quantity for training. In spite of these limitations, the CART Decision Tree algorithm was very suitable for reproducing the results of the supervised baseline classification in the paper.

Corpus-Authors	Accuracy	Precision	Recall	F-measure
Automotive-Original	0.501	0.502	0.501	0.501
Automotive-ADA	0.532	0.534	0.533	0.532
Book-Original	0.478	0.478	0.478	0.478
Book-ADA	0.474	0.474	0.474	0.472
Electronics-Original	0.468	0.469	0.468	0.468
Electronics-ADA	0.476	0.473	0.476	0.473
Health-Original	0.509	0.510	0.509	0.509
Health-ADA	0.476	0.475	0.475	0.474
Movies-Original	0.496	0.497	0.496	0.496
Movies-ADA	0.478	0.481	0.478	0.478

TABLE II: Comparison of baseline performances

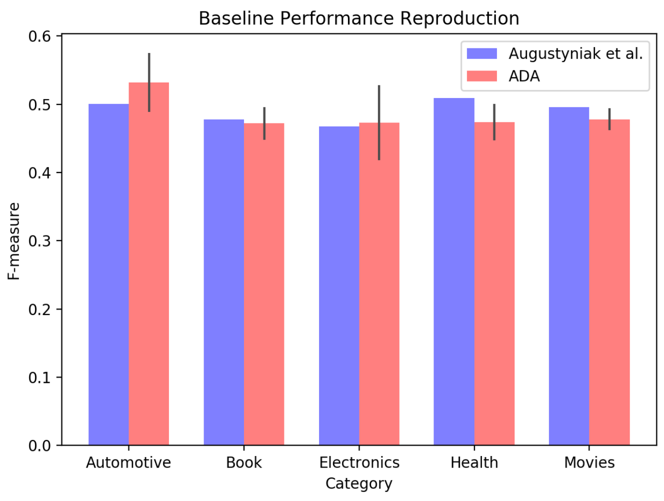

Fig. 1: Comparison of F-measure between baselines

III. Baseline Enhancement

It was anticipated that the baseline measures proposed by Augustyniak et al., presented above, were not adequate. Said otherwise, we believed that the baseline measures could easily be enhanced by either better feature design or by selecting a more appropriate classifier. More precisely, representation based on term frequency-inverse document frequency (TF-IDF) would yield better results than binary BoW. Another adjustment was to use text lemmatization to better group words which differ slightly in their orthography, but convey meanings of similar nature. Finally, a linear Support Vector Machine (SVM) was selected as a favored base classifier. Other simple classifiers were also investigated, such as Gaussian Naive Bayes (GNB) and Random Forest (RF).

Tf-idf, as stated above, is a statistical measure to estimate the importance of a word to a document in a collection or corpus. It is one of the most common weighting factor that is used in text mining and information retrieval.

- *Term Frequency (TF)* is the number of times the following term occur in the document. TF for a term 't' in document 'd' is:

$$TF(t,d) = \frac{\text{Number of times t appears in d}}{\text{TTL number of terms is d}}$$

$$tf_{t,d} = \frac{n_{t,d}}{\sum_{n=1}^{k} n_{t,d}}$$

- *Inverse Document Frequency (IDF)* is used to find the weights of the term. The weight decreases for the terms that occur very frequently and increases for rare words. IDF weight (w) for a term t is:

$$IDF(w) = \log_e \cdot (\frac{\text{Total number of documents}}{\text{Documents with term t in it}})$$

$$itf_t(w) = \log_e \frac{\text{D}}{\sum D_t}$$

Combining these two, we get

$$TF - IDF = tf_{t,d} \cdot idf_t(w)$$

which aids in finding a vocabulary for our analysis in which the terms are weighted based on their rarity.

Lemmatization takes the form and structure, or morphological analysis of the words into consideration to remove its inflectional endings in the text. It aims to return the word into its root or base form, which is known as a lemma. For example the following words, 'singing', 'sung', 'sang' and 'sings', are all the word 'sing' with various inflectional endings. The lemming algorithm will reduce all those words into 'sing' thereby outputting a more pure vocabulary.

Linear SVM is commonly used in text classification. Being an additive discriminative model, one of its advantages is its robustness when trained upon sparse data. Any BoW representation, or derivative thereof, of text will yield sparse data. This is due to the fact that texts will only contain a minority of words present within the corpus. Hence most features (i.e. words) will have null entries. Moreover, this is especially true for short texts such as the Amazon reviews under consideration in this study. However, the use of linear SVM assumes some linear separability between classes (positive, neutral, negative). If classes are not perfectly linearly separable, this can be accounted for with the slack hyperparameter *C*. This hyperparameter modulates the error associated with misclassified points and can therefore be useful in finding linear decision boundaries even if classes are not perfectly linearly separable. Some overlap between ordinal classes is expected. In this case, neutral reviews most likely overlap with negative and positive reviews to some non-negligible extent, whereas negative and positive reviews are expected to overlap minimally.

Other base classifiers such as GNB and RF were also trained to see whether linear SVM stands out in terms of performance. GNB is a generative model based on strong assumptions about conditional independence, and class conditionals being adequately described based on multivariate Gaussian distributions. It may thus fail to grasp the relations between words to convey emotional meaning. It is also possible that the class conditionals are not normally distributed. On the other hand, RF is expected to perform better than decision trees. As an ensemble model[1] based on many decision trees, RF has a lower bias than individual decision trees. However, more attention is allocated to linear SVM as it is also easier to tune than RF, thus representing the ideal simple classifier.

[1]Although the instructions for Track 1 require the use of simple classifiers, Random Forest was considered simple enough due to its similarity to decision tree despite it being an ensemble model. Also, Random Forest is easily implementable with dedicated packages and could therefore serve as a suitable baseline classifier in any study.

In training all these models, a first train/test split is performed with proportions of 0.7/0.3. A second split ensued on the training set to produce a validation set used for tuning hyperparameters. This split is also done with proportions 0.7/0.3. The final proportions for training/validation/testing sets are therefore 0.49/0.21/0.30.

IV. Results

Upon training linear SVM classifiers for the automotive category, it became clear that lemmatization and TF-IDF did not seem to offer superior performance over binary BoW (BBoW) or frequency BoW (FBoW) without lemmatization. This is grasped within Figure 2 where F1 scores are shown for BBoW, FBoW and TF-IDF with either lemmatization excluded (no Lem) or included (Lem). This figure relates to reviews of the automotive class.

The results within Figure 2 are also categorized by the length of n-grams used (n). An n-gram serves to form a BoW, the terms of which may be composed of several words. For instance, for n-gram length set to 2, the term 'good product' can be used as an individual term, such that the number of occurrence of the term 'good product' may be computed. An n-gram of length 3 could have the term 'very good product' as a term included into the BoW. In implementing n-grams in this work, it is noteworthy that n-gram length refers to the upper bound of the number of words that could form a single term within a BoW. Meaning, although n-gram length is set to 3 for example, this does not exclude the terms 'product' or 'good product' from forming terms within the BoW.

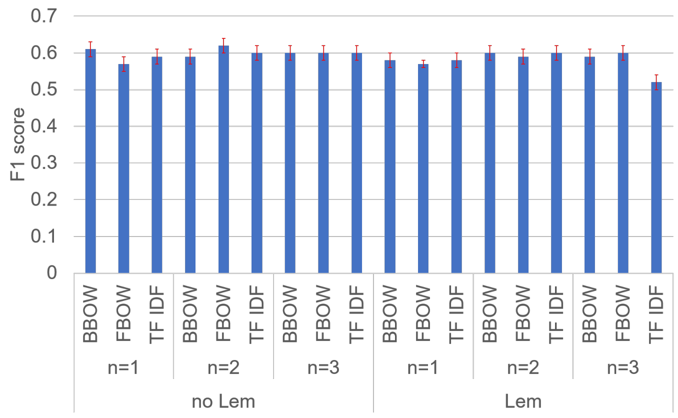

Fig. 2: Overall SVM F1 scores for automotive data when comparing the effects of lemmatization (no Lem vs Lem), feature design (BBoW vs FBoW vs TF IDF), and maximum n-gram length (n=1 vs n=2 vs n=3). Error bars are derived from a 10-fold cross-validation when using optimal hyperparameters.

These n-grams of varying lengths were added post-hoc since it was noted that the neutral class always exhibited lower performance than the positive and negative classes. Including n-grams was thus an attempt to grasp information that was conveyed by more subtle construction of words. Nonetheless, despite adding this ‘hyperparameter’ to our preprocessing strategy, the addition of n-grams does not improve overall performance as shown by Figure 2. This invariance of overall F1 scores also translates into invariance of neutral class F1 scores.

One key observation when comparing different preprocessing strategies and feature designs for linear SVM was the variability of the cost hyperparameter C. Values of this hyperparameter are shown in Figure 3 for the automotive reviews.

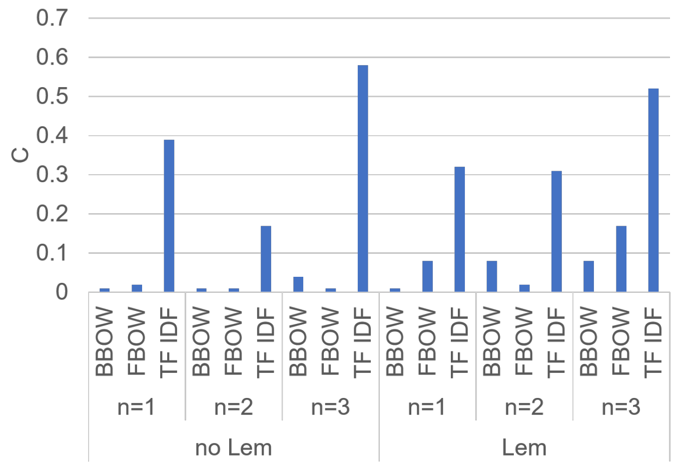

Fig. 3: Optimal values of the cost hyperparameter C for the automotive category, based on testing the model on a separate validation set. Same preprocessing approaches and feature designs as those of Figure 2.

The main observation is that employing TF-IDF results in increased values for the optimal cost hyperparameter, derived when testing the models on separate validation sets. The interpretation of this phenomenon is that smaller margins are favored when classifying reviews processed using TF-IDF.

Nonetheless, it seems that preprocessing and feature design has had little impact on performance in most cases. Perhaps the real difference arises due to the choice of algorithm. Hence, random forest classifiers were also trained. Although the effect of lemmatization or n-grams was not investigated, RF were trained for BBoW, FBoW and TF-IDF separately. RF tuned hyperparameters are the maximum depth of the trees, the minimum number of samples needed to split a node, the minimum number of samples per leaf, and the number of trees grown. The overall F1 scores are shown in Figure 4.

Although the performance for FBoW is slightly higher than that of BBoW and TF-IDF, all three feature designs yield overall F1 scores close to 0.6. The hyperparameters

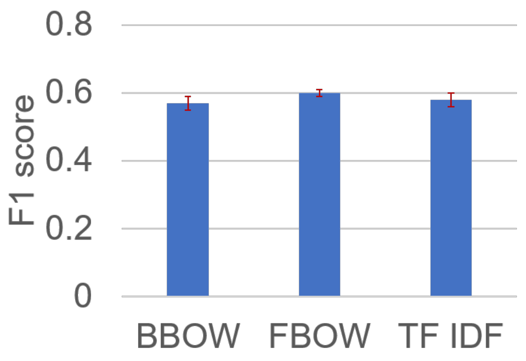

Fig. 4: Overall F1 scores of optimal RF classifiers for BBoW, FBoW and TF-IDF on books data category.

which give rise to these F1 scores are displayed in Table III, alongside the range of of values considered during hyperparameter tuning.

	Max depth	Min split	Min leaf	Num trees
BBoW	100	10	2	500
FBoW	110	10	2	350
TF-IDF	90	5	1	500
range of considered values	10 to 110, steps of 10	[2, 5, 10]	[1, 2, 4]	[50, 200, 350, 500]

TABLE III: Linear SVM hyperparameters yielding best F1 scores for BBoW, FBoW and TF-IDF. Tuned hyperparameters are the maximum depth (max depth), minimum number of samples needed to split a node (min split), minimum number of samples per leaf (min leaf) and the number of trees grown (num trees). The last row shows the range of values considered during hyperparameter tuning.

Finally, a Naive Bayes (NB) classifier was trained for BBoW, FBoW and TF-IDF on non-lemmatized data. A Bernoulli NB model was employed for BBoW, whereas Gaussian NB models were used for FBoW and TF-IDF. The Bernoulli NB was tuned as a function of the smoothing hyperparameter α. The resulting overall F1 scores are shown in Figure 5.

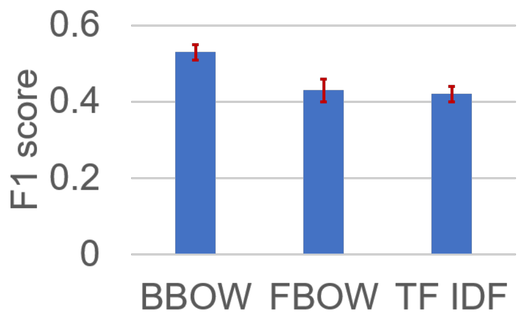

Fig. 5: Overall F1 scores of optimal NB classifiers for BBoW, FBoW and TF IDF.

The Bernoulli NB applied to BBoW seems to perform significantly better than the other two models. However its F1 score is nonetheless well below 0.6. The optimal α was found to be 0.6.

Linear SVM, RF, and NB classifiers were also performed on reviews from the *books* category. The effects of n-grams however were entirely ignored, as results based on the automotive reviews suggest that this measure does not show benefits for performance. Figure 6 shows overall F1 scores for the SVM classifier for BBoW, FBoW and TF-IDF applied to both lemmatized and non-lemmatized data.

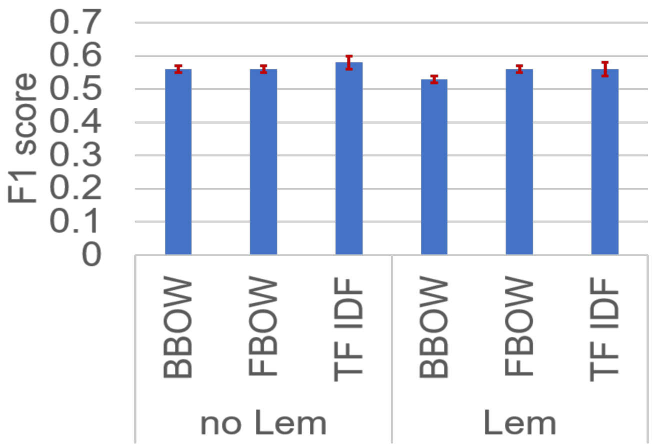

Fig. 6: Overall SVM F1 scores for books data when comparing the effects of lemmatization (no Lem vs Lem) and feature design (BBoW vs FBoW vs TF IDF). Error bars are derived from a 10-fold cross-validation when using optimal hyperparameters.

The performances seem quite stable across different techniques, except a slightly lower performance for lemmatized BBoW. Figure 7 shows the optimal SVM-based cost hyperparameter C for each of the scenarios. As for the automotive data in Figure 3, it seems that TF-IDF requires larger values for C and thus smaller margins.

Figure 8 shows the overall F1 scores when using RF classifiers. This has been applied to non-lemmatized data only. Results seem relatively stable, just under 0.6. Table IV shows the selected hyperparameters for RF, the same hyperparameters as for the automotive reviews.

Lastly, NB classifiers were trained for non-lemmatized BBoW (Bernoulli), FBoW and TF-IDF (Gaussian). Overall F1 scores are shown in Figure 9. BBoW and FBoW outperform TF-IDF, although with overall F1 scores below 0.5. The optimal α was found to be 0.2 for Bernoulli NB.

To assess the computational demands of the different classifiers (SVM, RF, NB), preprocessing approaches, (no Lem, Lem) and features designs (BBoW, FBoW, TF-IDF, n-grams), the execution runtimes are presented within Figures 10 to 15.

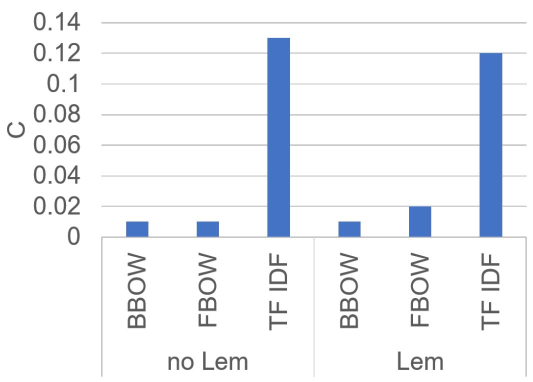

Fig. 7: Optimal values of the cost hyperparameter C for the books category, based on testing the model on a separate validation set and the same preprocessing approaches and feature designs as those of Figure 6.

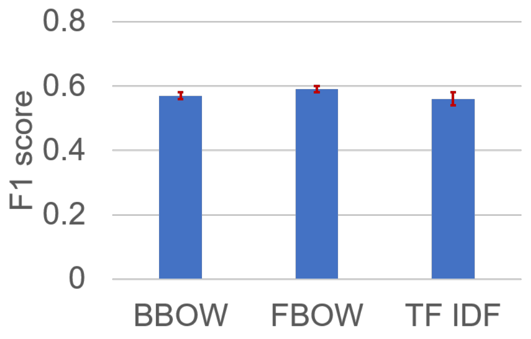

Fig. 8: Books RF F1

	Max depth	Min split	Min leaf	Num trees
BBoW	70	2	2	500
FBoW	110	5	2	350
TF-IDF	None (?)	10	4	200
range of considered values	10 to 110, steps of 10	[2, 5, 10]	[1, 2, 4]	[50, 200, 350, 500]

TABLE IV: RF hyperparameters yielding best F1 scores for BBoW, FBoW and TF-IDF. Tuned hyperparameters are the maximum depth (max depth), minimum number of samples needed to split a node (min split), minimum number of samples per leaf (min leaf), and the number of trees grown (num trees). The last row shows the range of values considered during hyperparameter tuning. For some reason, the max depth for TF-IDF returns ‘None’. This is probably an execution error.

Out of all runtimes for the automotive data, the best performing classifier with the lowest runtime is the linear SVM with BBoW, no lemmatization, and n-gram length set to 1 (F1 = 0.61, Figure 10). The runtime is 26.4 s for this classifier. This is lower than the runtime of 54 s obtained by the authors’ lexicon-based method, yielding an F1 score of

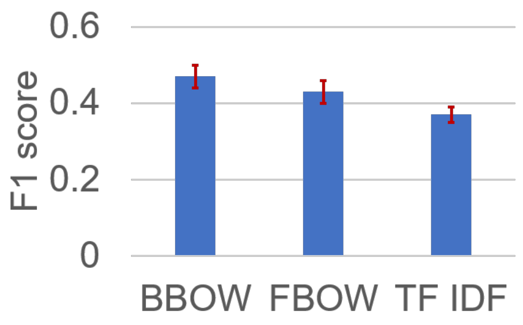

Fig. 9: Overall F1 scores of optimal NB classifiers for BBoW, FBoW and TF IDF.

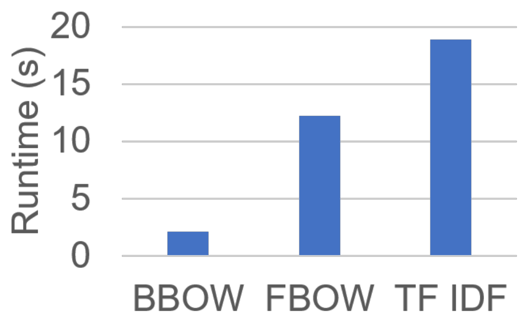

Fig. 12: Runtimes for NB classifiers applied to automotive data.

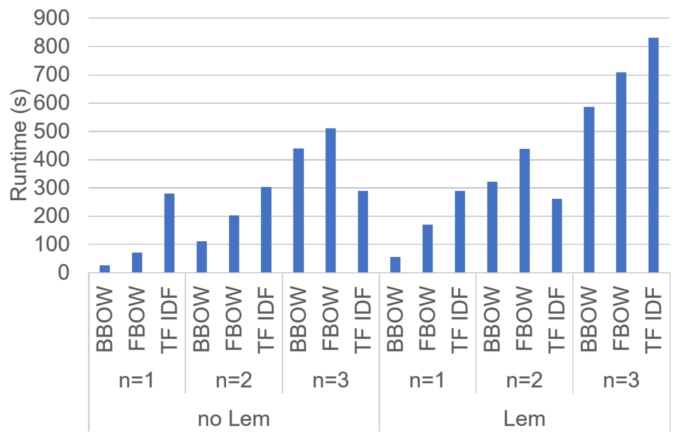

Fig. 10: Runtimes for SVM classifiers applied to automotive data.

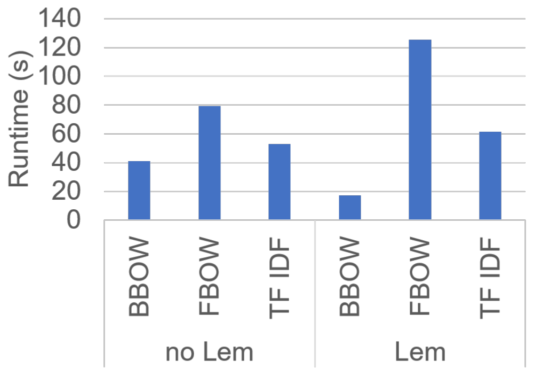

Fig. 13: Runtimes for SVM classifiers applied to books data.

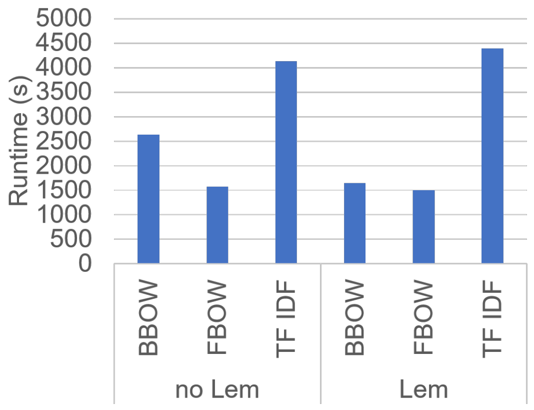

Fig. 14: Runtimes for RF classifiers applied to books data.

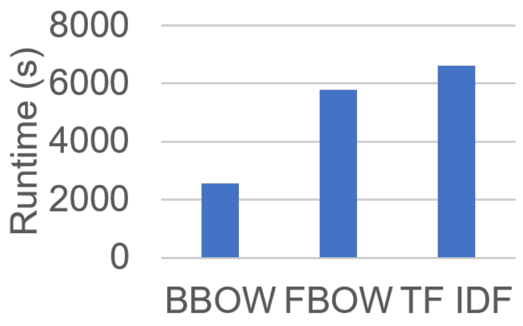

Fig. 11: Runtimes for RF classifiers applied to automotive data.

0.50. On the other hand, the linear SVM for BBoW and with lemmatization yields a runtime of 17.3 s (Figure 13) while performing adequately for the book reviews (F1 = 0.52). This is quite faster than the authors' lexicon-based method which yields a runtime of 89 s and F1 score of 0.48 for the book reviews.

V. Conclusion

A simple linear SVM using BBoW representation provides a better overall F1 score and faster runtime than the authors' lexicon based method for both automotive (no lemmatization) and book (lemmatization) reviews.

SVM is better as compared to other models we tried, i.e. RF and NB. The sparsity would explain why SVM is outperforming NB. This is due to the fact that little data is available to produce the class conditional distributions needed for generative models such as NB. Although RF has similar performance (although somewhat lower), it is possible that RF is not as heavily penalized by the sparsity of the data. It would seem, nonetheless, that the data is somewhat linearly

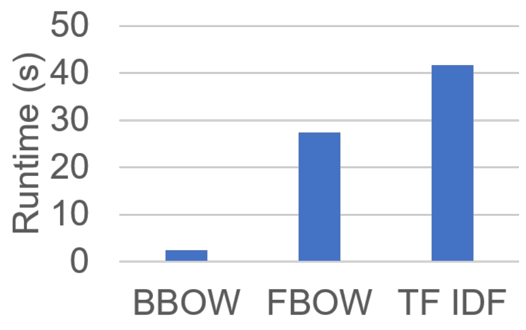

Fig. 15: Runtimes for NB classifiers applied to books data.

separable, hence the performance of SVM. Otherwise, asides from RF and NB, SVM outperformed the decision tree classifiers, both un-tuned and tuned.

One key observation when using linear SVM is the relation between the cost hyperparameter C and the choice of feature design (BBoW vs FBoW vs TF-IDF). As depicted in Figures 3 and 7, the value for C increases significantly for TF-IDF compared to BBoW and FBoW. This thus means that smaller margins are needed for TF-IDF.

Also from our implementation of these various classifiers, we came to the conclusion that using Decision Trees for text analysis is not suitable, as it works on the principles of splitting the nodes based on optimum threshold values. These values would be ‘1’ or ‘0’ for BBoW, and an integer value decided through maximum information gain based on entropy for FBoW. In text analysis, it is possible that either a word will be present in all the documents, and thus there will not be an optimum split of nodes, or instead a word is present in most of the documents, thereby making a very unbalanced split. Also, if a word is either present or absent in the very large majority of reviews, the splits may likely not be informative. To avoid these complications, using classifiers like SVM provide an overall better F1 score.

Statement of Contributions

Aryan Gupta: Data preprocessing and hyperparameter tuning of implemented Linear SVM. Produced graphs based on various parameter tuning values and execution times.

Andrew Kam: Baseline reproduction implementation, CART decision tree tuning and analysis, dataset construction and filtering, corresponding sections in report.

Dylan Mann-Krzisnik: Implementation of models and extensive hyperparameter tuning of the models. Produced graphs based on various parameter tuning values and execution times.

We hereby state that all the work presented in this report is that of the authors.

References

[1] Augustyniak, L., Kajdanowicz, T., Szymaski, P., Tuligowicz, W., Kazienko, P., Alhajj, R., & Szymanski, B. (2014, August). Simpler is Better?: Lexicon-Based Ensemble Sentiment Classification Beats Supervised Methods. In Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (pp. 924-929). IEEE Press.