Assignment 1: Comparison of Baseline Models For Sentiment and Topic Classification

Vilém Zouhar
s5000076
`vilem.zouhar@gmail.com`

Edu Vallejo Arguinzoniz
s5016894
`vallezoniz@gmail.com`

Abstract

This short report presents the result of baseline models for the task of e-commerce review sentiment and topic classifications. Despite the limited complexity of the used models (Naive Bayes and Logistic Regression), they perform strongly better than the baseline (Most Common Class Classifier).

The LR model seems to be better suited for this task because it saturates earlier for topic classification and the experiments suggest that providing more data would lead to even better results, which is not necessarily true for Naive Bayes. In the focus on error analysis, we find that several classes are harder to predict for all models and get systematically misclassified as a specific different class. This suggests either a low capacity of the used models or insufficient features (bag of word and TF-IDF).

1 Introduction

Sentiment classification is the task of assigning a class based on an attitude of a certain artefact, most commonly a textual review. This has been widely studied (Catal and Nangir, 2017; Pang et al., 2002) and has many direct applications in the industry (Rane and Kumar, 2018). Although the possible classification of sentiments can be very fine-grained we limit our study to the most rudimentary classification: positive and negative. Topic classification is similarly an important component in many modern commercially deployed systems (Horii et al., 2019). Usually, the classification is more fine-grained to many more classes though we limit our experiments to only 6 (see Section 2). Many approaches to topic classification are unsupervised and therefore do not require annotated data (Natarajan et al., 2007; Schwartz et al., 2001). Furthermore, state-of-the-art models are usually based on word embeddings which are followed up by an application of a simple neural network (Setiawan et al., 2016) or in some cases a foundation model, such as BERT (Sun et al., 2019). For pedagogical reasons, we bring our attention to two simple models: Naive Bayes and Logistic Regression. We examine their working and compare to a baseline (Most Common Class Classifier).

The models themselves, Naive Bayes and Logistic Regression have been studied in these contexts as well (Prabhat and Khullar, 2017; Liu et al., 2013; De la Peña Sarracén, 2017; Lee et al., 2011).

In Section 2 we provide basic insight into the provided data, mainly its distribution. In Section 3 we focus on the used metrics, the way the experiment is evaluated and the description of the used models. Section 4 discusses the results of various experiments which we conclude in Section 5. We skip an explicit related work section because of the smaller scope of this report.

2 Data

The data is composed of shorts spans (reviews) coming from the e-commerce domain. There are 152 tokens in 6 sentences on average per review. It is annotated by (1) sentiment, binary classification and (2) topic, 6 classes. There are 6000 reviews in total and the distribution, somewhat balanced, is shown in Table 1.

We list two examples with annotations (detokenized, capitalized, selected shorter than average). In the first case the sentiment is correctly identified by the annotator as negative.

Class	Positive	Negative	All
Music	531	496	1027
DVD	490	522	1012
Software	502	492	994
Books	471	522	993
Camera	504	484	988
Health	470	516	986
All	2968	3032	6000

Table 1: Distribution of provided data across two modes of classification

„Braun ls-5550 silk&soft bodyshave rechargeable cord/cordless women's shaver is absolutely useless, not fit for purpose"
→ **Health, Negative**

Unless the second review was accompanied by e.g. one-star rating, which would suggest the use of sarcasm, the sentiment annotation as *Negative* is incorrect in this case.

„The cd came as promised and in the condition promised. I'm very satisfied"
→ **Music, Negative**

The origin of the data is unknown to the authors and it is not clear whether there it has been annotated automatically or with annotators and what the possible inter-annotator agreement was. Brief examination shows that the labels in the data may not be very reliable.

To make the reviews processable by the statistical models, we apply either Bag of Words or TF-IDF, which result in high-dimensional vectors - features for the models.

3 Methods

We first briefly introduce the changes in the provided codebase, then discuss the evaluation and then focus on the description of individual models.

3.1 Code

We vastly updated the code and split it into several files in order to be able to organize various subexperiments better and separate presentation (figures) from actual experiments. Although the main code structure remains vaguely similar (load data and perform a task), we extended it by adding model selection (`--model`), moving `seed` to the argument list for reproducibility, and most importantly, selection of which experiment to run (`--experiment`):

- `main` (default): uses the given model and vectorizer, trains it on the train part of the data (given by `--test-percentage`) and evaluates it on the test (may change based on `--shuffle` and `--seed`). Computes all the relevant metrics (see Section 3.2.3). Results are dumped in a file specified by `--data-out`.
- `cv`: Similar to `main` but performs 10 fold CV instead of using a single split. The logged results are the average of the 10 runs
- `error_length`: with the given model and vectorizer, find specific misclassification examples (hardcoded for simplicity) and report average lengths of review in cases of correct and incorrect predictions
- `error_corr`: with the given model and vectorizer, compute correlations between the correctness of predictions between tasks
- `train_data`: with all models examine the performance (measured by accuracy) based on how much train data is given (tested on a fixed test-set as in `main`; the result is in a file specified by `--data-out`
- `stability`: with the given model and vectorizer, empirically assert that the results do not change when training data is shuffled
- `train_stability`: with the given model and vectorizer, compute variance in 10-fold cross-validation runs
- `errors`: Dumps to standard output all the misclassified instances (in 10 fold CV) and their respective predicted probabilities and predicted and true labels. Computes average confidence (predicted probability of the predicted label) for correct and incorrect classification instances.

Some miscellaneous functions were moved to `report_utils.py` for clarity in `main.py`. Brief data analysis is done by running `analysis.py`. Figures in this short report are generated using files `fig_conf_matrix.py` and `fig_train_data.py` (they require the output of specific experiments of `main.py`).

3.2 Evaluation

3.2.1 Split

Since there is no official split the choice of evaluation methodology was in our hands. 10-fold cross-validation (CV) was used for evaluating the systems as opposed to a more standard approach of making a static train-test split. Using CV results in more statistical soundness in the evaluation at the cost of computing, which is a non-concern in this case since the trained models are simple and the dataset is small.

3.2.2 Data shuffling

Shuffling before making the split results in different performances because the test set changes which renders the results not comparable.
When data is shuffled within the splits (train and test) there is the effect of shuffling each split to consider. Shuffling the test set produces no change in the performance as the evaluation is an accumulation over individual samples.
The effect of shuffling the train-set depends on the learning algorithm used, there are methods that are sensitive to it (e.g. NNs optimized using batching), and methods that are not (e.g. Naive Bayes, Logistic Regression with certain parameter optimizing algorithms).

3.2.3 Metrics

Standard metrics for classification were used, these include: **accuracy**, **precision**, **recall** and **F1-score**. Precision, recall and F1-score can only be applied per class, as such, scores of these three metrics are averaged using different strategies to obtain overall scores. Accuracy is only computed overall.

3.2.4 Averaging

For computing overall task scores two averaging strategies are employed:

- **weighted-averaging** (or **micro-averaging**), where the scores for each class are averaged taking into account the number of samples in each class. This is equivalent to disregarding class, to begin with, and computing the scores for the whole test-set. Weighted averaging is computed as:

$$M = \frac{1}{|C| \cdot S} \sum_{c \in C} s_c m_c$$

  For an arbitrary metric, where M is the averaged result, C is the set of classes, s_c is the support (number of instances) in each class, S is the total number of instances (sum of all s_c) and m_c is the value of the metric for a given class.

- **macro-averaging**, where the scores for each class are directly averaged without considering the support that these individual classes have. This way of averaging is stricter in the sense that it usually produces more pessimistic results. It stems from the fact that it penalizes systems for underperforming when classifying minority classes, which are harder to classify by nature as they also represent a smaller portion of the train-set (under a uniform or stratified split). It is especially useful in unbalanced datasets. The formula of this averaging strategy would be:

$$M = \frac{1}{|C|} \sum_{c \in C} m_c$$

While accuracy can be easily cheated on an imbalanced dataset by predicting the most common class (e.g. detecting illness where most cases are negative), this is not the case for F1-score because it requires both precision and recall to be more balanced in order to get to higher values.

3.3 Models

If we shuffled the data within the splits (i.e. within train and within test set) the performance would not change for the following two reasons: (1) the model would be the same because training is done by maximum likelihood estimation of individual samples and is irrespective of the sequence and (2) the evaluation is an accumulation over individual samples. For the aforementioned reasons we use 10-fold cross-validation.

One should not look at the test set to not make unfair updates to the model. In extreme cases with full knowledge of the test set we could simply generate a lookup table from feature vectors to correct answers and get full accuracy (in case there is no overlap between feature vectors over different classes). In our case, however, we present the average results of a 10-fold cross-validation which blurs the lines between training and test data, making it more development data. Nevertheless, we decided that because the task lacks a standardized test set, this is the fairest way to proceed

3.3.1 Baseline

For any classification task with a reasonably limited number of classes, we can simply use the most common class classifier. For multinomial classification that is `dvd` and for sentiment `neg` though because the class distribution is fairly balanced, this provides poor performance.

3.3.2 Naive Bayes Classifier

The following formulation is the posterior (class probability given the evidence) makes no assumptions and is mathematically sound. The inversion stems from the Bayes theorem. We can remove the probability of the features from the equations because they are constant between the probabilities of different classes y_k and do not affect the arg max:

$$\begin{aligned}
&p(y=k|x) \\
&\quad = \frac{p(y=k)}{p(x)} \prod_i p(x_i|y=k, x_{<i}) \\
&\arg\max_k p(y=k|x) \\
&\quad = \arg\max_k p(y=k) \prod_i p(x_i|y=k, x_{<i})
\end{aligned}$$

Naive Bayes is a model which makes one crucial simplifying assumption, namely the independence between features x. This assumption is important because it resolves the issue of data sparsity.

$$\begin{aligned}
&p(y=k|x) \approx p(y=k) \prod_i p(x_i|y=k) \\
&\arg\max_k p(y=k|x) \\
&\qquad \approx \arg\max_k p(y=k) \prod_i p(x_i|y=k)
\end{aligned}$$

The class priors represent our prior beliefs about the distribution of the classes. By maximum likelihood estimation, we can estimate it as:

$$p(y=k) = \frac{\sum_i \mathbf{1}_{y_i=k}}{n}$$

The likelihood with an individual feature can be estimated similarly:

$$p(x_j = l|y=k) = \frac{\sum_i \mathbf{1}_{x_{i,j}=l \wedge y_i=k}}{\sum_i \mathbf{1}_{y_i=k}}$$

This can suffer from the issue of data sparseness and hence smoothing can be applied. The used implementation[1] uses Laplace (add-α) smoothing with α=1.

The presented version only supports only non-negative numbers. It can, however, be extended to support e.g. TF-IDF by using average weights per class (formulation with indicator functions follows this pattern).

3.3.3 Logistic Regression

The natural continuation of Naive Bayes is its discriminative sibling Logistic Regression. Model output (class probability) is defined as a softmax over a vector of scores over classes:

$$p(y_k|x) = \frac{\exp(\sum_i x_i \beta_{i,k})}{\sum_k \exp(\sum_i x_i \beta_{i,k})}$$

Ng and Jordan (2002) showed that asymptotically logistic regression performs better than the Naive Bayes. The disadvantage is that the generative model approaches this asymptote faster and the available data for this task is limited. Another limitation is the overall model family because it prevents us in sampling new feature vectors given their labels. Data generation is not useful for this task, but could be for different ones.

Logistic Regression can be trained in two ways: (1) It can be represented as a neural network computation and used with standard DL optimizers or (2) there exists an iterative solution to the parameters which converges to the global minimum. The used implementation uses the latter training approach.

4 Results

4.1 Model Comparison

The results of topic classification and binary sentiment classification are shown in Table 2. It shows that even simple models can outperform the baseline by a large margin.

In most cases, logistic regression seems to outperform Naive Bayes and TF-IDF is a better vectorization than a simple bag of words. Surprisingly

[1] scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

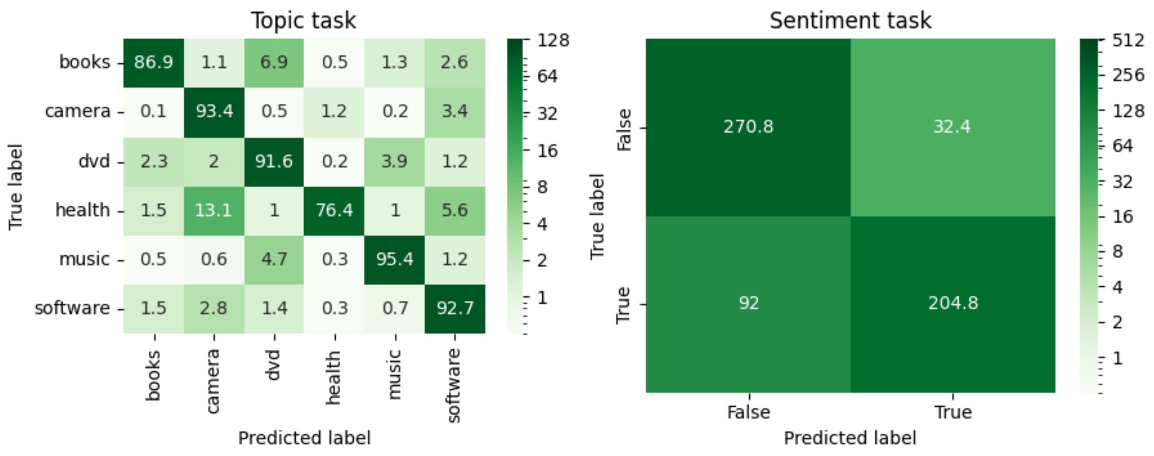

Figure 1: Averaged confusion matrices across 10-fold CV for both tasks using Naive Bayes with TF-IDF vectorization

Topic task	BoW	TF-IDF
Most Common Class	17.12%	17.12%
Naive Bayes	78.97%	79.72%
Logistic Regression	82.22%	82.13%

Sentiment task	BoW	TF-IDF
Most Common Class	50.53%	50.53%
Naive Bayes	89.82%	89.88%
Logistic Regression	88.83%	91.03%

Table 2: Accuracies of examined models with different feature vectorization using 10-fold CV

the task of sentiment classification led to only a few percent increase compared to topics classification with more classes.

Stability. Although there was variance across the individual CV runs, in all cases the models had a standard deviation under 0.02. To further examine the stability, we fixed a single test set (10%) and then trained the models on 90% of the remaining training data (cross-validated). This way all the models would be evaluated against a single test set. The average standard deviation was even lower (0.002) possibly because of the unified test set. The average diameter of the results (max-min) was 1%. This suggests that the training of the models is reasonably stable.

4.2 Training Data Size

The provided data is quite limited and we explored how much would the performance increase if we had more of it. Figure 2 shows the curves for both tasks for all models and vectorizations.

For topic classification, both Naive Bayes and Logistic Regression converge to a similar point. This is true even across various vectorizations. These diverge in the middle: for Logistic Regression the TF-IDF works better while the opposite is true for Naive Bayes. The graph suggests that more data would not lead to significant improvements in performance.

For sentiment classification, the performance is less stable and for Naive Bayes reaches its peak soon. For Linear Regression, which outperforms the other model, it seems like more data could lead to further performance gains.

4.3 Error Analysis

In this section, we analyze the output of the Naive Bayes model with TF-IDF vectorization to provide more insight into the difficulty of this task (e.g. which classes are hard to predict).

4.3.1 Misclassification Patterns

CV-averaged confusion matrices were computed for each task which is shown as heatmaps on Figure 1. Note that the colour intensity in these heatmaps corresponds to a logarithmic law which highlights small values more than a linear map would. Additionally, precision, recall and F1 metrics are provided in Table Section 4.3.1.

It is useful to examine them to find out what the common misclassification patterns are and whether some classes are e.g. easier to predict than other ones. Both heatmaps show healthy main di-

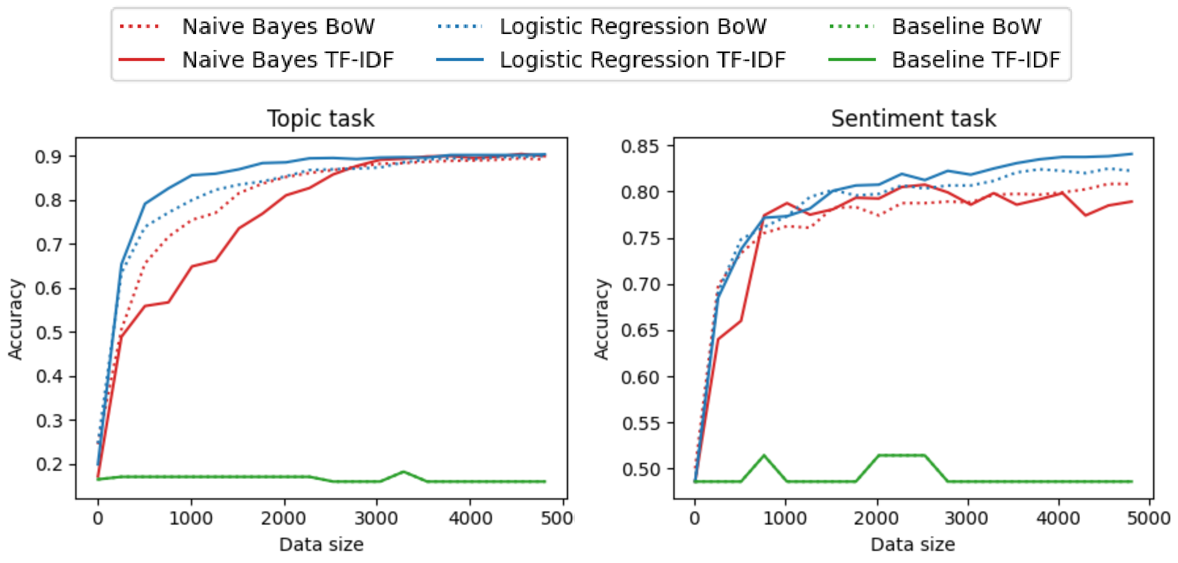

Figure 2: Effect of limited training data size on model performances

agonal values meaning the model correctly predicts most examples in the test set.

In the topic task, it can be seen that the "health" class is hard for the model to predict. The model especially misspredicts instances from the "health" class as the class "camera", and as the class "software" to a lesser extent. It is interesting to see that the contrary does not happen, i.e. instances from the classes "camera" and "software" are not predicted as being of the class "health" as consistently.

This behaviour can be further analysed by looking at the precision and recall for the classes "health", "camera" and "software". As one can see, the precision for "health" is very high (the highest of all classes) while the recall is very low (the lowest), this shows that the model is hesitant to predict the class "health", but when it does it is very reliable. On the contrary, the classes "software" and "camera" have the two metrics inverted, meaning the model is eager to predict instances as "camera" or "software" (often times it being wrong), which results in these types of predictions being less reliable, with the positive side effect that instances that really are from these classes will usually get correctly predicted.

The next weakest performance is more than ten points higher and corresponds to the class "books" which gets mistaken with class "DVD". The rest of the classes are more packed in terms of performance. We believe that some misclassification is caused by wrong annotations, as discussed in Section 2. However, the following example (capitalized and detokenized) demonstrates why it is easy to misclassify DVD and Books. One reason is that book reviews tend to make references to the movies and vice versa.

„*If you saw the show and liked it enough that you want a book of lyrics and cast photographs, then this is a book you want. If you haven't seen or didn't like the show, I can't imagine why you would want this*"
→ **~~DVD~~ Book**

For the sentiment task "False negatives" are the most common type of error, almost three times more common than "False positives" in fact. If the counts of predicted labels are computed (taking the sum of the columns in the matrix), the model predicts considerably more negative sentiment labels (362.8, 60.5%) than positive labels (237.2, 39.5%).

The following example (detokenized and capitalized) was predicted by Naive Bayes with BoW to be Negative with 99.35% confidence. When TF-IDF was used instead, it was still misclassified but now only with the confidence of 65.83%.

„*Please do yourself a favor and buy this cd! Whatever you do, dont listen to it in your car though. No kidding aside, I almost ran my car off the road because I was laughing so hard*"
→ **~~Negative~~ (99.35%) Positive**

A common problem of Naive Bayes that could explain this bias is that the model is not able to understand negation statements properly. In such case an expression like ”not bad” could be classified as negative sentiment because the word ”bad” produces a very strong conditional probability for the negative label, and ”not”, being a very common word, can appear with a similar frequency in positive and negative instances which means that it is not going to have a big impact in the final conditional probability.

Models like Naive Bayes which process each word individually are bound to make the same mistakes, furthermore, it is well known that negation is a hard linguistic phenomenon to process. For algorithms to be able to grasp negation it is necessary that they are able to exploit the structure of the sentence and resolve the coreferences for negation.

Although with overall better performance, the misclassification distribution for Naive Bayes with Bag of Words and Logistic Regression with both vectorizers follows the similar spikes: health-Camera, DVD-Book and Software-Camera (not shown in the figure). This suggests one of the following explanations: (1) either these labels are inherently easier to misclassify for the specific used models or (2) the used vectorization is insufficient in describing the data points.

Prediction difficulty Finally, we examined how review length influenced the predictions in Table 4. When using Bag of Words, the errors are made predominantly on shorter reviews (for both tasks). This however changes when TF-IDF is used for vectorization. The results for Naive Bayes are similar to those of Logistic Regression (not shown). There were no meaningful distinctions between specific misclassifications, e.g. FN vs. FP. For Naive Bayes with BoW the Pearson's correlation between correct prediction and review length was 0.10 and 0.09 for topic and sentiment classification respectively. The correlation between the tasks was 0.05 which corresponds to 90% of all examples classified with the same correctness. This percentage stays the same for other configurations though the correlation drastically drops because of the increase in model performance and higher imbalance between the classes. Overall these results suggest that there are some properties of the review (e.g. its length) that affects the difficulty of prediction.

Topic task	Precision	Recall	F1
Music	0.94	0.92	0.93
Software	0.87	0.93	0.90
Books	0.91	0.88	0.90
Camera	0.84	0.95	0.89
Health	0.96	0.81	0.88
DVD	0.87	0.89	0.88
Accuracy			0.90
Macro avg.	0.90	0.90	0.90
Weighted avg.	0.90	0.90	0.90

Sentiment task	Precision	Recall	F1
False	0.77	0.83	0.80
True	0.81	0.74	0.78
Accuracy			0.79
Macro avg.	0.79	0.79	0.79
Weighted avg.	0.79	0.79	0.79

Table 3: 10-fold CV averaged precision, recall and F1-scores for the **topic** and **sentiment** tasks, using Naive Bayes with TF-IDF vectorization

Task		BoW	TF-IDF
Topic	Correct	155	152
	Incorrect	62	110
Sentiment	Correct	156	149
	Incorrect	85	182

Table 4: Average review lengths for correct and incorrect predictions using Naive Bayes

4.3.2 Confidence

Both models, despite being from different family of statistical models (generative and discriminative) output probabilities per every output class, e.g. $(0.1, 0.05, 0.05, 0.45, 0.25, 0.1)$ for topic classification. Out of this we usually take the $\arg\max$ and make a prediction. We examine the probability of this maximum class, dubbed confidence, and explore how it differs across model configurations and across tasks. These results, divided across correct and incorrect predictions are shown in Table 5.

The confidence is higher for correct predictions than for incorrect predictions, which is a good thing as this can be used for self-reporting when the model itself is unsure of the output. Surprisingly, sentiment classification has similar values

Topic task	BoW		TF-IDF	
	Correct	Incorrect	Correct	Incorrect
Naive Bayes	98.05%	83.91%	58.64%	34.68%
Logistic Regression	91.85%	61.25%	68.85%	39.30%

Sentiment task	BoW		TF-IDF	
	Correct	Incorrect	Correct	Incorrect
Naive Bayes	95.24%	87.43%	65.97%	57.81%
Logistic Regression	90.61%	78.78%	71.06%	60.33%

Table 5: 10-fold CV averaged confidence in correct/incorrect predictions for the **topic** and **sentiment** tasks, using Naive Bayes and Logistic Regressoin with Bag of Words and TF-IDF vectorizations.

to those of topic classification.

Over both models and tasks, TF-IDF led to much lower confidences compared to using Bag of Word vectorization. The authors are unaware of any explanation of this phenomenon. While when using BoW, Logistic Regression has lower confidence in especially incorrect predictions, this is reversed when TF-IDF is used.

5 Summary

In this short report, we compared and examined the models Naive Bayes, Logistic Regression and Most Common Class Classifier for the tasks sentiment and topic classification. In most cases, the Logistic Regression performed better than the other models. Furthermore, it converged to its optimum faster (required less data). The effect of using TF-IDF for vectorization was quite ambivalent because it did not systematically improve the results across all runs.

Some classes in the data get systematically misclassified by the models as different classes suggesting low capacity of the used models or low descriptive power of the used vectorization. The data also contain patterns which influence the difficulty of predictions (of both tasks), such as the review length.

Future work Although the current task was a toy example and the insights rudimentary ones, it would be useful if there existed a compact ready-made solution that would run all these steps automatically and generate a standardized report.

Furthermore, there are other basic models which we did not explore because it would be outside of the requirements of the assignments and could possibly lead to deduced points. Such models would be e.g. SVMs, Decision Trees or Nearest Neighbours.

References

[Catal and Nangir2017] Cagatay Catal and Mehmet Nangir. 2017. A sentiment classification model based on multiple classifiers. *Applied Soft Computing*, 50:135–141.

[De la Peña Sarracén2017] Gretel Liz De la Peña Sarracén. 2017. Ensembles of methods for tweet topic classification. In *IberEval@ SEPLN*, pages 15–19.

[Horii et al.2019] Yoshiki Horii, Hirofumi Nonaka, Elisa Claire Alemán Carreón, Hiroki Horino, and Toru Hiraoka. 2019. Topic classification method for analyzing effect of ewom on consumer game sales. *arXiv preprint arXiv:1904.13213*.

[Lee et al.2011] Kathy Lee, Diana Palsetia, Ramanathan Narayanan, Md Mostofa Ali Patwary, Ankit Agrawal, and Alok Choudhary. 2011. Twitter trending topic classification. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 251–258. IEEE.

[Liu et al.2013] Bingwei Liu, Erik Blasch, Yu Chen, Dan Shen, and Genshe Chen. 2013. Scalable sentiment classification for big data analysis using naive bayes classifier. In *2013 IEEE international conference on big data*, pages 99–104. IEEE.

[Natarajan et al.2007] Prem Natarajan, Rohit Prasad, Krishna Subramanian, Shirin Saleem, Fred Choi, and Rich Schwartz. 2007. Finding structure in noisy text: topic classification and unsupervised clustering. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(3-4):187–198.

[Ng and Jordan2002] Andrew Y Ng and Michael I Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848.

[Pang et al.2002] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*.

[Prabhat and Khullar2017] Anjuman Prabhat and Vikas Khullar. 2017. Sentiment classification on big data using naïve bayes and logistic regression. In *2017 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–5. IEEE.

[Rane and Kumar2018] Ankita Rane and Anand Kumar. 2018. Sentiment classification system of twitter data for us airline service analysis. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 769–773. IEEE.

[Schwartz et al.2001] Richard Schwartz, Sreenivasa Sista, and Timothy Leek. 2001. Unsupervised topic discovery. In *Proceedings of workshop on language modeling and information retrieval*, pages 72–77. Citeseer.

[Setiawan et al.2016] Erwin B Setiawan, Dwi H Widyantoro, and Kridanto Surendro. 2016. Feature expansion using word embedding for tweet topic classification. In *2016 10th International Conference on Telecommunication Systems Services and Applications (TSSA)*, pages 1–5. IEEE.

[Sun et al.2019] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.