Comparison of Models for Sentiment Classification of Movie Reviews

Joel Nilsson
joeni078
Project in Text Mining - TDDE16
Linköping University
Linköping, Sweden
joeni078@student.liu.se

***Abstract*—Sentiment analysis is a domain within natural language processing useful for investigating popular opinion with respect to an entity. The increasing computational power of modern computers have opened the door to deep learning which has inspired many new approaches to language modeling in recent years. One of the most powerful neural network architectures today is the self-attention-based Transformer - a key component in the pre-trained language model BERT. This study compares the performance of BERT and a naïve Bayes classifier at the task of predicting the sentiment of movie reviews using two benchmark datasets and puts the results into context of previous work. The naïve Bayes classifier was surprisingly powerful, but could not match the performance of BERT, which was more accurate in general and closer to the true sentiment in its misclassifications. However, a significant weakness of deep learning models is interpretability, which is an issue that lawmakers and researchers are becoming increasingly aware of. Choosing a simpler model that is slightly inferior to a complex model as BERT in terms of accuracy may therefore be preferable in certain contexts.**

***Index Terms*—natural language processing, sentiment classification, machine learning, BERT**

I. Introduction

Sentiment analysis, or opinion mining, is the area of investigating the opinions of people about entities [1]. Social media and review platforms have become a useful tool for companies or politicians to extract popular opinion of their products or campaigns. Sentiment analysis can be applied on entire documents, on a sentence level, or on an aspect level. Aspect-based sentiment analysis is necessary when opinions of an entity as a whole is not informative enough for ones purpose, but inferring the opinions with respect to different aspects of an entity is naturally more challenging. This study will work on a sentence and document level. There are both supervised and unsupervised approaches to sentiment analysis [1]. Some methods use a hand- or machine-crafted sentiment lexicon to decide the polarity of a document. It is possible to achieve good results using conventional machine learning (ML) methods with simple features, such as support vector machines with bag-of-words (BOW) features [22].

Increasingly powerful modern processors has led to deep learning (DL) gaining popularity also in the field of natural language processing (NLP) [12], [8]. Different types of neural network architectures have mostly replaced conventional ML models like naïve Bayes, k-nearest-neighbours, hidden Markov models, and support vector machines (SVM). The latter mainly relies on manually crafted features, while DL methods relies on learning word representations that are persisted. Pre-trained DL models can be *fine-tuned* to a specific downstream task (such as sentiment classification) using smaller datasets and less training time [8]. Recurrent neural networks (RNN) have been a common choice of architecture until the introduction of the Transfomer due to [10], which mostly has replaced RNNs [12].

However, deep learning models are more opaque than conventional ML models, and may exhibit inconsistent behaviour, which is one reason they are not ubiquitously embraced in society [26], [27]. The issue of interpretability of deep learning models is a relevant topic as lawmakers are increasing the demand on AI models to be interpretable following the European Union's adpotion of the General Data Protection Regulation [14]. Training on a large corpus might yield word representations that encode of amplify undesirable characteristics, such as race and gender stereotypes [31]. Applying DL models with high performance in a societal context, e.g., hate speech detection, can give unpredictable results. Successfully interpreting the result and mechanisms of a model may also give insight on its limitations and how to improve it [27].

The aim of this study is to compare the performance of a more complex model against a conventional ML model on the task of sentiment classification using two datasets. The results are put into context of previous research in the field of sentiment analysis. The complexity and interpretability of the models will be discussed.

The rest of this section motivates the choice of models and features by looking at previous work. Section II will present the theory behind the classifiers used for sentiment analysis in this study. Section III and IV describe the dataset used and the method of the study. Section V presents the results, followed by a discussion in Section VI, and conclusions in Section VII.

A. Choice of Models

The naïve Bayes (NB) and SVM models often appear as baselines for sentiment analysis, with BOW as well as embedding vector features, and their performance is quite strong [29], [22], [20], [2]. Neither of these two models is clearly known to be better than the other [22], and since we are also interested in the interpretability of the model, naïve Bayes is used as a baseline in this study. The choice is upported by the generative properties of the NB classifier (it models the predictions in terms of probabilities - this is possible for SVM as well [30], but the model is optimized by gradient descent and has more parameters) and the connection between the presence of words and probabilities are explicit with NB (we avoid dealing with boundaries in a vector space). To feed the NB, we use different BOW features. The authors of [11] showed that using POS information appended to words or only adjectives are not reliably better than using BOW features, so POS tags were not considered in this study.

The introduction of BERT (*Bidirectional Encoder Representations from Transformers*) [4] marked a milestone for the field of NLP and deep learning, and has since then inspired numerous architectures and training methods that in turn have pushed the limits of what pre-trained language models can achieve [8]. BERT can be fine-tuned to several tasks, including question answering, named entity recognition, sentence-pair classification, and single sentence classification. Other notable pre-trained language models include ELMo (*Embeddings from Language Models*) [9], which is a word embedding model mainly used to augment the features of existing models, and GPT (from *Generative Pre-Training*) [7], which has a Transformer architecture (see Section II) similar to BERT, but uses a unidirectional language model [8]. ELMo and GPT can be seen as precursors to BERT, which was explicitly designed to mitigate their shortcomings, moving away from feature-based approach of ELMo, and introducing more complex (bi-directional) language modeling. This work will use a version of the BERT-model for sentiment classification and review the architecture of the Transformer encoder of [10] that is the basis for BERT.

II. Theory

A. Naïve Bayes

The naïve Bayes classifier is a generative supervised learning model that makes the assumption that the attributes of the features are independent of each other given the class. In a BOW model, where we disregard the context and position of the words in a text, the NB model describes the probability of a document $d = \{W_1, W_2, ..., W_N\}$ as a sequence of random variables (words) W_i, $i = 1, .., N$, given a class c as:

$$p(d|c) = p(W_1, W_2, ..., W_N|c) = \prod_{i=1}^{N} p(W_i|c) \qquad (1)$$

The probability of a document belonging to a class c is given by applying Bayes theorem to (1):

$$p(c|d) = \frac{p(c)\prod_{i=1}^{N} p(W_i|c)}{p(d)} \propto p(c)\prod_{i=1}^{N} p(W_i|c) \qquad (2)$$

where $p(c)$ is the class prior, and $p(d) = \sum_{c'} p(c')\prod_i p(W_i|c')$ is independent of c. The prediction of the classifier is taken as

$$\hat{c} = \underset{c}{\operatorname{argmax}}\, p(c)\prod_{i=1}^{N} p(W_i|c) \qquad (3)$$

The model is trained on data, and the probability for a word $w^{(j)}$ (at any position) is estimated by maximum likelihood with Laplace smoothing to avoid zero probabilities [13]:

$$p(W_i = w^{(j)}|c) = \frac{n_{jc} + \alpha}{\alpha|V| + \sum_{k:w^{(k)}\in V} n_{kc}} \; \forall i = 1, ..., N \quad (4)$$

where n_{jc} is the number of times the word $w^{(j)}$ appears in documents of class c, and V and $|V|$ are the vocabulary of the entire corpus and its cardinality, respectively. The Laplace smoothing parameter α is often set to 1 [3]. In other terms, to estimate the parameters of the model we take for each class the absolute counts of each word divided by the number of occurrences for all words (with the regularizing terms). It is possible to use other features in the NB classifier, such as bigrams. As noted in [11], bigrams violate the independence assumption in (1), but that does not necessarily affect the performance of the NB classifier; one may argue that the assumption does not hold in general anyway.

B. Self-Attention and Transformers

As mentioned in the above, deep learning techniques have taken over from conventional ML models in NLP. Two very popular models were RNNs and convolutional neural networks (CNN). The authors of [10] introduced a neural network model based on self-attention as a way of resolving the inherent problems with RNNs and CNNs, and it has been a very successful technique in the domain of natural language processing [4], [13], [10]. Self-attention layers are the key components in the Transformer neural network architecture that is the basis for BERT [4].

The self-attention mechanism is a way of quantifying which part of the context surrounding a word that is most relevant for its representation, i.e. which words the model should focus on [13]. In a self-attention model, an input sequence $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)$ is mapped to an equally long output sequence $\mathbf{Y} = (\mathbf{y}_1, ..., \mathbf{y}_N)$, where each element $\mathbf{y}_i$ depends on the corresponding element $\mathbf{x}_i$ in the input sequence, and all of the input elements preceding that $(\mathbf{x}_0, ..., \mathbf{x}_{i-1})$. A dot product can capture the similarity between vectors, and the output of a self-attention computation is taken as the sum of the inputs weighted by their (softmax-normalized) similarity:

$$\mathbf{y}_i = \sum_{j|j\leq i} w_{ij}\mathbf{x}_j, \tag{5}$$

$$w_{ij} = \frac{\exp \mathbf{x}_i^T \mathbf{x}_j}{\sum_m \exp \mathbf{x}_i^T \mathbf{x}_m}. \tag{6}$$

As the input sequences are used as values and for computing the weights in (5), they can be further transformed to serve each purpose better [13]. Using linear transformations $Q, K, V \in \mathbb{R}^{N\times N}$, the inputs are projected into a representation of a query $\mathbf{q}_i = Q\mathbf{x}_i$, key $\mathbf{k}_i = K\mathbf{x}_i$, and value $\mathbf{v}_i = V\mathbf{x}_i$. A query is the input which we want to compute the output for. The key is a (previous) input we want to compare against the query in order to compute their similarity, by which we weight the value that contributes to the output. More specifically, the ith output $\mathbf{y}_i$ is the sum of the values up to point i, weighted by the query-key similarity:

$$\mathbf{y}_i = \sum_{j|j\leq i} w_{ij}\mathbf{v}_j, \tag{7}$$

$$w_{ij} = \frac{\exp \mathbf{q}_i^T \mathbf{k}_j}{\sum_m \exp \mathbf{q}_i^T \mathbf{k}_m}. \tag{8}$$

Note in (7) and (8) how no other queries than $\mathbf{q}_i$ is present in the computation of $\mathbf{y}_i$, and how we sum over all previous keys $\mathbf{k}_i$ and values $\mathbf{v}_j$. In [10], a scaled dot product is used to compute similarity between inputs as to avoid large values that give small gradient in training. A *multi-head* self-attention architecture is employed to learn different aspects of inter-word relationships, some of which would disappear if only using a single self-attention head. The multi-head self-attention boils down to projecting the queries, keys, and values again to different intermediate spaces through linear transformations, then applying the self-attention transformation, and finally concatenating the resulting output and applying one last linear transformation.

The data passed into the multi-head self-attention layer are the queries $\mathbf{Q} = Q\mathbf{X}$, keys $\mathbf{K} = K\mathbf{X}$, and values $\mathbf{V} = V\mathbf{X}$. These are transformed using the matrices $Z_j^Q \in \mathbb{R}^{N\times d_{qk}}$, $Z_j^K \in \mathbb{R}^{N\times d_{qk}}$, and $Z_j^V \in \mathbb{R}^{N\times d_v}$, where j ranges from 1 to the number of attention heads, and d_{qk} and d_v are the dimensions of the intermediate spaces. Each self-attention layer is computed as

$$\begin{aligned}\text{SelfAttention}^{(j)}(\mathbf{Q}Z_j^Q, \mathbf{K}Z_j^K, \mathbf{V}Z_j^V) = \\ \text{softmax}\left(\frac{(\mathbf{Q}Z_j^Q)^T\mathbf{K}Z_j^K}{\sqrt{N}}\right)\mathbf{V}Z_j^V,\end{aligned} \tag{9}$$

where softmax is applied row-wise, and the elements of the upper triangle of the softmax argument are set to $-\infty$, which is mapped to zero in the softmax function in order to only pay attention to previous terms. The result of the multi-head attention layer is the concatenation of the vectors obtained from (9) for every j, multiplied by a matrix that reduces the dimension back to the input size.

The Transformer proposed in [10] is a stack of encoder and decoder units. The encoder learns a representation of the input, and the decoder learns to make predictions on the encoder input based on those representations. We focus on the encoders since that is the basis for the BERT model [4]. Each encoder layer consists of two sub-layers, the first being a multi-head attention layer described above, and the second being a regular two-layer feed-forward neural network with ReLU activation:

$$\text{FFNN}_2(x) = A_2 \times \max(0, A_1 x + b_1) + b_2, \tag{10}$$

where A_i and b_i are the weights and biases of each layer. To the output of each sub-layer, the input of the sub-layer is added (residual connection), a trick to facilitate the training of deep networks [16]. The sum is then normalized with layer normalization [17], a method that normalizes the summed inputs of a neuron before passing it to the activation function as a means of speeding up (recurrent) neural network training.

In a transformer, the encoder takes word embeddings encoded with the word positions as input. The choice of [10] was a set of sinusoids with different frequencies depending on the dimension of the model and the position of each word that are added to the word embeddings.

1) BERT: The BERT architecture comes in various sizes, specified by the number of Transformer encoder stacks, L, the hidden layer size, H, and the number of and self-attention heads that make up a multi-attention head, A. The version called BERT-base (which we will use) is of size $L = 12$, $H = 768$, $A = 12$. Notably, BERT uses bidirectional self-attention, taking both left and right side context into account when pre-training as opposed to the (left to right) unidirectional GPT [7].

BERT was pre-trained on masked language modeling and next sentence prediction. The former task entails predicting words that have been replaced by a special token or a random word. The latter implies predicting whether a sentence *B* follows a sentence *A*. Sentence *B* is picked as the sentence actually following *A*, or at random from the corpus. BERT trained on a corpus extracted from English Wikipedia of 2.5 billion words, and on BookCorpus data [15], a corpus of 11,038 books online, each longer than 20,000 words that were written by unpublished authors. The BookCorpus dataset contains more than 1.3 million unique words. BERT is fine-tuned by adding a feed-forward architecture on top of the encoder stacks, and continuing the training on the specific task, in this case sentiment analysis. In the fine-tuning, all parameters of the model are adjusted. It is also possible to use the output of BERT as fixed features in a separate architecture [4]. The input to BERT is a sequence of tokens that is described in more detail in Section IV. Positions information

is encoded and segment embeddings are added to the input that indicate whether the token belongs to sentence *A* or *B*.

C. Sentiment Detection

Not every sentence in a movie review is informative when predicting the opinion of the author regarding a movie [2]. Reviews tend to describe the plot of the movie, and these sentences contain words that can be interpreted as carrying sentiment. To distil the relevant information from a review, we seek a method for detecting sentiment, or subjectivity. One can use a classifier for estimating which sentences contain subjective opinions, and which are more objective as a preprocessing step to sentiment classification. A NB classifier combined with a minimum-cut approach to filter subjective sentences was used in [2]. A method by [3] sorted sentences based on NB predictions. The hypothesis in [2] is that sentences that are spatially close to each other in a document are more likely to both carry sentiment or both be "objective". This can be formulated as a minimum-cut problem, where the weights of the minimum-cut graph comes from a subjectivity detector, and the problem is solved for every sentence. Solving a minimum-cut problem may be costly, but since the number of sentences are reduced, time is gained in the training and sentiment classification step.

III. Data

Two benchmark datasets were used for evaluation of the classifiers in this study. An additional dataset was used for subjectivity detection in the preprocessing step for the naïve Bayes classifier.

A. Sentiment Datasets

The first dataset used in this study was introduced in 2011 by [18] and is publicly available[1]. Since its compilation it has been used as a benchmark dataset in sentiment analysis in e.g., [21], [22], [19], and [24]. The dataset contains 25,000 labeled movie reviews from the Internet Move Database[2] (IMDb) that is balanced between positive and negative polarity. There were a maximum of 30 reviews per movie, and the classes were decided from the number of stars that the movie received in the review: a score of more than 7 out of 10 was marked as positive, and review giving less than 4 stars were marked as negative. The authors of [18] point out that a previous dataset by Pang and Bo suffered from correlations with the movie labels (249 out of 406 movies had reviews that were either all positive or all negative), suggesting that special plot words might affect the predictions. The dataset contains an additional 25,000 unlabeled reviews that may be used for unsupervised learning.

The second dataset used was originally compiled by the authors of [23] from the movie rating site Rotten Tomatoes[3].

[1]https://ai.stanford.edu/∼amaas/data/sentiment/

[2]https://www.imdb.com/

[3]http://www.rottentomatoes.com/

TABLE I
SAMPLES OF EACH CLASS IN THE SST DATASET. PERCENTAGE POINTS ARE ROUNDED TO THE NEAREST INTEGER.

set\class	v. neg.	neg.	neutr.	pos.	v.pos	tot.
train	1092	2218	1624	2322	1288	8544
	(13%)	(26%)	(19%)	(27%)	(15%)	
test	279	633	389	510	399	2210
	(13%)	(29%)	(18%)	(23%)	(18%)	
dev.	139	289	229	279	165	1101
	(13%)	(26%)	(21%)	(25%)	(15%)	
tot.	1510	3139	2242	3111	1849	11855
	(13%)	(26%)	(19%)	(26%)	(16%)	

This version of the dataset[4] was parsed into sentences and phrases in [20] and subsequently annotated by humans through the Amazon Mechanical Turk platform. We will refer to it as the Stanford Sentiment Treebank (SST) dataset by the paper that augmented it. The SST dataset is also one of the benchmarks of sentiment analysis [19], [24]. The dataset is split into training, development and test sets of 8544, 1101, and 2210 samples respectively. Each phrase (sub-sentence) has a sentiment score in the interval $[0, 1]$, and belongs to a class: $[0, 0.2]$, $(0.2, 0.4]$, $(0.4, 0.6]$, $(0.6, 0.8]$, $(0.8, 1.0]$ for *very negative*, *negative*, *neutral*, *positive*, and *very positive*, respectively. The dataset may also be used for binary prediction by removing neutral class sentiment in $(0.4, 0.6]$, which yields a split of 6920, 872, 1821, respectively between train, development and test sets. We only consider whole sentences in our predictions (referred to as *root* in [20]). There is a slight imbalance between the classes in the binary case (3306 negative, 3608 positive in the training set, and 912 negative, 909 positive in the test set), and in the fine-grained 5-class case there is an imbalance between classes, but the ratios are approximately the same over all sets, as shown in Table I.

B. Subjectivity Dataset

The subjectivity dataset was introduced in [2]. It contains 5000 subjective sentences taken from Rotten Tomatoes movie reviews, and 5000 objective sentences taken from IMDb plot summaries. All reviews and summaries are related to movies released after 2001 and latest before the end of 2004. The sets of sentences are assumed to be subjective (reviews) and objective (plots summaries), respectively, by their origin. The sentences have been downcased and include more than ten tokens each.

IV. Method

The performance of the NB model is compared against BERT by evaluating the classifiers on both of the IMDb and SST datasets. Different text preprocessing techniques and features are tried for the former, while a single BERT model (BERT-base) is used. A set of standard bag of words features are tried with unigrams, bigrams, and unigrams and bigrams, using presence (binary), frequency, or term frequency-inverse document frequency (tf-idf) counts. For all models, the training sets are used for training, and the

[4]https://nlp.stanford.edu/sentiment/

test sets are used for validation. Suitable features are found through cross-validation on the training sets. The code used in this study is publicly available[5].

The data used to train and evaluate the naïve Bayes classifier is prepared in the following way. The movie reviews from the IMDb dataset are first optionally filtered for subjective sentences. For this purpose, a separate naïve Bayes classifier is trained on the subjectivity dataset. The subjectivity data is tokenized and lemmatized with `spacy`[6]. A naïve Bayes model is fitted to the data using unigram presence features as in [2]. The minimum-cut method of [2] is implemented as a complementary method of predicting subjectivity. We use $0.5 \cdot \exp(1-d)$ (for $d \leq 3$, and 0 otherwise) with d as the distance between sentences as the association function. After verifying the performance of the subjectivity classifier, it was retrained on the entire subjectivity dataset.

An NB classifier is trained on the (optionally filtered) movie reviews of the IMDb dataset. The sentences are lemmatized by the same process as for the subjectivity dataset. Stop words are not filtered out since they may contain critical information on the sentiment of a review, e.g., negation [22], [18]. The procedure for training the model on the SST dataset is identical, except for the subjective filtering, which is not used since the SST documents contain almost exclusively single sentences. The implementation of the naïve Bayes classifier is provided by `sklearn` package (version 1.0) in Python 3, which uses Laplace smoothing as in (4) as a default.

BERT-base is used without subjectivity filtering. The model takes lower case, accent free WordPiece tokens with whitespace around punctuation characters[7]. WordPiece tokenization is achieved by iteratively picking the longest prefix of the current token that matches a token in the vocabulary, until no such prefix can be found [28]. Two special tokens, `[CLS]` and `[SEP]`, are placed at the beginning of the input, and between the input sequences *A* and *B* respectively. Since we deal with single sentence classification, the second sentence is left blank. The token sequence is truncated to a length of maximum 128 tokens, and position and segment tokens are added as described in Section II. A single layer of the same size as number of classes, fully connected with BERTs output, is added. The layer uses softmax activation and dropout probability of 0.1 during fine-tuning. During prediction the class corresponding to the node with largest output is taken as the polarity of the review. During fine-tuning the whole network, both final layer and BERTs pre-trained parameters are adjusted. We trained with a batch size of 32 and passed over the training set for 4 epochs as suggested by [4]. The learning rate was chosen from $\eta \in \{10^{-4}, 5 \cdot 10^{-5}, 2 \cdot 10^{-5}, 10^{-5}\}$ using the development set for the SST dataset and sample of the training set for the IMDb data. A guide for using BERT on tensorflow hub (version 1) provided by Google Research[8] was of help when running the experiments.

[5]https://github.com/joel-n/sentiment-analysis

[6]spacy version 3.2.0, model: `en_core_web_lg`

[7]any character with a P* Unicode class or any non-alphanumeric ASCII character

TABLE II
ACCURACY (%) FOR MODELS ON THE IMDB DATASET. THE UPPER HALF OF THE TABLE SHOWS THE RESULTS OF THE EVALUATIONS OF THIS STUDY.

Model	Accuracy (%)
NB, $\alpha = 0.2$, unigram, presence	82.0
NB, $\alpha = 0.2$, bigram, tf-idf	87.3
NB, $\alpha = 0.2$, unigram+bigram, tf-idf	86.4
NB, $\alpha = 1$, unigram, presence, subj-filter	84.2
NB, $\alpha = 1$, bigram, presence, subj-filter	87.9
NB, $\alpha = 1$, uni+bigram, presence, subj-filter	87.6
NB, $\alpha = 1$, unigram, presence, filter (min-cut)	84.0
NB, $\alpha = 1$, bigram, presence, filter (min-cut)	86.9
BERT-base	89.1
SVM, BOW (bnc) [18] (2011)	87.80
SVM, BOW (bΔt'c) [18] (2011)	88.23
SVM, word vec. [18] (2011)	87.44
SVM, word vec. + unlab. [18] (2011)	87.99
SVM, word vec. + BOW (bnc) [18] (2011)	88.33
SVM, word vec. + BOW (bnc) + unlab. [18] (2011)	88.89
MNB, unigram, presence [22] (2012)	83.55
MNB, bigram, presence [22] (2012)	86.59
SVM, unigram, presence [22] (2012)	86.95
SVM, bigram, presence [22] (2012)	89.16
NBSVM, unigram, presence [22] (2012)	88.29
NBSVM, bigram, presence [22] (2012)	91.22
WRRBM [21] (2012)	87.42
SVM, WRRBM + concat. BOW (bnc) [21] (2012)	89.23
Neural net, para. vector [19] (2014)	92.6
1-layer LSTM, word vector [24] (2015)	89.1
2-layer MLP, para. vector [24] (2015)	**94.5**

V. RESULTS

In this section we present the results of the evaluations. To put the results in a context and provide insight on what classification methods previous work have used, we present the results of several studies. Below are descriptions of *a subset* of the methods used in previous literature. The results for classification on the IMDb and SST datasets of this work and the mentioned studies are found in Tables II and III.

A. Related Work

The authors of [18] introduced a semi-supervised word vector model that was evaluated on the IMDb dataset. The word vectors were used as features in a linear SVM, and was compared against using BOW features with binary counts (b), no idf weighting (n) or smoothed delta idf (Δt'), and cosine normalization (c). Concatenating the word vectors with BOW features was also evaluated, and the model was boosted by using the unlabeled part of the dataset.

In [22], several baseline models (NB, SVM, and a combination of them) were evaluated on the IMDb dataset using binary counts (presence) of unigrams and bigrams.

The authors of [21] presented a method of learning n-gram representations with restricted Boltzmann machines (RBM), called *word representation RBM* (WRRBM). One model was trained for each class (positive and negative) in the IMDb dataset, and a thresholding model taking the average free

[8]https://github.com/google-research/bert

TABLE III
ACCURACY (%) FOR MODELS ON THE SST DATASET. THE UPPER HALF OF THE TABLE SHOWS THE RESULTS OF THE EVALUATIONS OF THIS STUDY.

Model	(5 classes)	(binary)
NB, $\alpha = 1$, unigram, presence	40.5	82.7
NB, $\alpha = 0.2$, unigram, presence	40.2	82.1
NB, $\alpha = 0.2$, bigram, presence	37.2	77.9
NB, $\alpha = 0.2$, unigram+bigram, presence	39.8	82.3
NB, $\alpha = 0.2$, unigram+bigram, tf-idf	39.8	81.4
BERT-base	44.1	90.6
NB, BOW [20] (2013)	41.0	81.8
SVM, BOW [20] (2013)	40.7	79.4
biNB, bigram BOW [20] (2013)	41.9	83.1
Recursive NN, vector [20] (2013)	43.2	82.4
RNTN, vector [20] (2013)	45.7	85.4
Log. reg., para. vector [19] (2014)	**48.7**	87.8
3-layer LSTM, word vector [24] (2015)	42.6	84.3
Deep recursive NN [24] (2015)	46.9	84.7
MLP/log.reg., para. vector [24] (2015)	37.9	76.4
MLP/log.reg., avg. word vector [24] (2015)	40.6	78.6
BERT-base [4] (2019)	-	93.5
BERT-large [4] (2019)	-	**94.9**

energy of the n-grams in the review according to the class conditional WRRBMs was used to predict the sentiment of a review. In addition, the BOW vector (as in [18]) was appended to the WRRBM energies and the result was used as features in an SVM which improved the result.

The paragraph vector model of [19] concatenates a paragraph specific vector to a set of learned word vectors, giving additional information on the surrounding context. The paragraph vector is learned through unsupervised prediction tasks separate from, and in tandem with the word vectors learning process. Word vectors were learned for all sub-phrases, and for predicting the sentiment of reviews, they were fed to a logistic regression model (SST) and a neural net (IMDb). The (unseen) paragraph vectors of the test sentences were computed when predicting, but the word vectors were not updated.

The authors of [20] tried using NB and SVM with bag of words features, as a baseline for two *recursive* neural networks and their proposed *recursive neural tensor network* (RNTN), with learned word vectors as features. The recursive models are constructed to efficiently predict the sentiment of sub-phrases.

In [24], word and paragraph vectors were learned and passed to LSTM and deep recursive neural networks, or used in shallow multilayer perceptron (MLP) with logistic regression to classify the sentiment in the IMDb and SST datasets.

Two versions of BERT was used on the binary classification task in [4], BERT-base ($L = 12$, $H = 768$, $A = 12$) and BERT-large ($L = 24$, $H = 1024$, $A = 16$). The fine-tuning learning rate was optimized on the development set.

B. Experiment Results

The NB models implemented in this study clearly perform better on all classification tasks compared to random guessing (which would yield $\sim$50% on the binary tasks, and $\sim$20% for the 5 class task), but fall behind most of the DL models presented above.

Cross-validation indicated that unigram, or unigram-bigram features were best for the 5-class and binary SST tasks, and that bigram, or unigram-bigram features might be better for the IMDb dataset, although the evidence for the latter was not as strong. Setting the Laplace smoothing parameter for NB to $\alpha = 0.2$ instead of the standard $\alpha = 1$ seemed to be preferable for the IMDb dataset. In Tables II and III we present the results of the weighting schemes (in terms of α and presence/frequency/tf-idf features) of unigram, bigram, and unigram-bigram features that obtained the highest score on the cross validation.

The results confirmed that for the SST task, using unigram features gave the highest accuracy, and bigrams turned out to work better for the IMDb dataset. In practically all cases, using combined unigram-bigram features was equally good or slightly inferior to using only one of them. These results also held true when evaluating more configurations on the test set.

The subjectivity filter almost always improved the accuracy, giving an average increase in accuracy of percentage points 1.28% on the IMDb data when using unigram or bigram features. Using the minimum-cut method for filtering did not yield better results than using only the NB subjectivity classifier for the parameters evaluated. The minimum-cut method removed fewer sentences, increasing the vocabulary (of the unigram model) from around 69,000 to approximately 75,000, compared to the total of around 91000.

For the 5-class task, the NB model favours predicting *positive* or *negative*, and seldom *neutral* or *very positive/negative* as seen in Figure 1. Not only does BERT-base do better in terms of accuracy, inspecting Figure 1 reveals that the classifications are closer to the true labels even when they are wrong. Balancing the dataset lowered the bias of the NB model but reduced accuracy. It was also noted that tf-idf features resulted in predictions even more biased towards those classes with most data compared to presence features. When predicting on the IMDb dataset, the NB classifiers were biased towards the *negative* class even though the dataset was balanced.

The models that achieved the best results were the paragraph vectors of [24] (outperforming the BERT of this study on the IMDb dataset), BERT-large [4] (binary SST task), and the paragraph vectors of [19] (SST dataset with 5 classes).

VI. DISCUSSION

Despite being a very simple model, naïve Bayes does a relatively good job at all three classification tasks. The importance of choosing suitable features, as emphasised by [22] is confirmed. However, in contrast to [22], here it was found that bigrams do not improve the results for all sentiment datasets, as indicated by Table III. The vocabulary for the

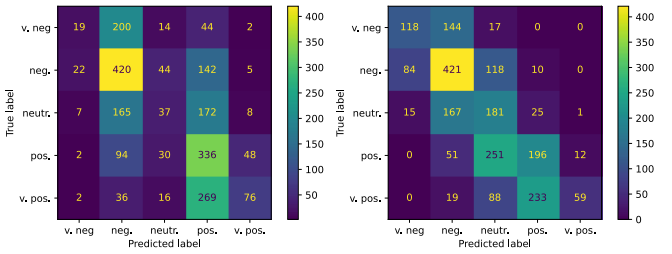

Fig. 1. Confusion matrices for classification with naïve Bayes with unigram presence features (left) and BERT-base from this study (right) on the SST dataset (5 classes).

IMDb dataset is considerably larger (90913 unigrams), and it has more and longer reviews compared to the SST dataset (12651 unigrams). The relatively shorter reviews of the SST dataset may obstruct learning the large amount of bigram probabilities, which could explain the drop in performance when using bigrams.

It is worthy to note that both forms of the subjectivity filter seemed to improve the results. The min-cut algorithms are perhaps not suitable for longer texts, as complexity for max-flow based algorithms is e.g., $\mathcal{O}(V^2E)$ for Dinic's algorithm (V vertices and E edges). Computations took a long time due to the repeated lemmatization operations, this should be avoided in the future. Sparsity of the graphs can be forced by lowering the threshold and scaling parameters of the association function to lower computation time, but might not give the best results. Parameters should be optimized with respect to the downstream task.

The subjectivity dataset originated from the same domain as the classification objects, which is most likely optimal, for the same reason a sentiment lexicon is domain specific. Ideally there should be no overlap between the movies in the subjectivity dataset and sentiment datasets so that the model does not filter out relevant information. If the name of a famous actor appears in several reviews that are objective, the classifier has a (slightly) higher probability of filtering out subjective sentences that contain the same name. The subjectivity dataset contains movies between approximately 2002-2004, and the IMDb dataset contains movies from a wide span of time as can be seen if one looks at a few reviews. While overlap is possible, the risk of a significant impact is judged as small. The IMDb dataset creators also seem to have taken measures to include both positive and negative reviews of the same movie into the dataset in order not to risk, for instance, the name of a specific actor to significantly influence the classification decision.

As expected, BERT-base performs better than the NB classifiers (and in range of the other DL models) but not by very much, which is surprising. On the IMDb dataset it achieves an absolute increase of 1.2% in accuracy. The results also showed that the NB classifier and BERT were more even on the IMDb dataset ($\sim$ 86% vs 89.1%) than on the SST task ($\sim$ 82% vs 90.6%). However, this might be due to suboptimal fine-tuning scheme in this study. Both BERT versions performed impressively on the binary classification task of the SST dataset in [4]. The quality of the classifications of BERT was of higher quality on the 5-class classification task, making fewer serious errors than the NB classifier. In the case of one NB classifier the review "*If Melville is creatively a great whale, this film is canned tuna.*" was predicted to be *very positive*, due to the presence of "whale"(!) and "great" which for that model had larger weights in a *very positive* context. On the other hand, the word "creatively", as one might expect to give a positive contribution, is estimated to appear more often in *very negative* contexts than *very positive* contexts (however the opposite is true for "creative"). This is not very surprising, but the example shows the limitations of the naïve Bayes model, and that a classification decision might have other reasons than one might expect. BERT predicted that the sentence was *negative*, which is a misclassification, but closer to the true label. Looking at Figure 1, it seems to be the general case that BERT comes closer to the "truth" in this sense.

In terms of numbers, the NB classifiers have around 10,000 to 1 million parameters when fitted to these datasets, depending on if unigrams or bigrams are used. On the other hand, BERT-base has 110M, BERT-large 340M [4]. BERT has an advantage by being pre-trained on a huge corpus to represent complex linguistic constructions that take word positions and contexts into account. Moreover, NB cannot handle words outside the vocabulary, while BERT may have some usable representation of the words that do not appear in the training set.

The models are operating under the assumption that the reviews refer to the movie, and not to unrelated elements. The IMDb dataset was labeled by assuming that the reviews giving 7 stars or more are positive, which is not always the case. A few counterexamples are provided in the Appendix C. The models may also be thrown of by sarcastic reviews that means the literal opposite of what is written.

A. Interpretability

BERT obtains better performance on the sentiment analysis task, but it challenging to understand how the information in BERT is encoded. By searching for the most important neurons for predicting a certain linguistic property, it was found in [32] that a subset of neurons were sufficient to replicate the performance on linguistic tasks, seemingly isolating some 'knowledge' to subcomponents of the network. Their findings also suggest that the information encoded in BERTs parameters are spread out between different layers,

whereas other pre-trained models tend to use separate layers to encode different levels of abstractions. An example is ELMo, the lower layers of which obtain higher accuracy than higher layers on the POS tagging task, while the reverse is true for the word sense disambiguation task [9].

Attempts have been made at understanding what the representations of BERT encode, but as shown in [5], it is hard to isolate a single linguistic pattern to the activation of a single, or linear combination of nodes. The authors describe different ways in which BERT may represent linguistic concepts related to directions and locations in the embedding space, and shows why this is likely. The results imply that using a wide range of datasets is necessary to obtain a better the interpretation and realize its limitations.

Uninterpretable models can have unpredictive behaviour, as shown by two examples. Training on a large corpus might yield word representations that encode of amplify undesirable characteristics, such as race and gender stereotypes [31]. The authors of [26] refer to an example in the image processing domain where an intentional but imperceptible perturbation in the input to a model caused its output to vary arbitrarily. Applying DL models with high performance such as BERT in a societal context, e.g., hate speech detection, can give unpredictable results.

When dealing with the shallow naïve Bayes model we can look at the estimated probabilities $p(w^{(j)}|c)$ directly to understand the model behaviour. We have estimated these probabilities by looking at the occurrences of words in the reviews for each class, and we know how a change in the training corpus would affect the model. Therefore, we may tweak the model and its predictions in a reliable way. The classification decision can be determined by looking at the probabilities, and the overarching logic of the model is clear: documents that are similar in word distribution to documents of a certain class will be classified as belonging to that class. But as noted above with the case of "whale", one should not be too quick to jump to conclusions on why the classifier makes a certain decision.

VII. Conclusions

Good results on sentiment classification tasks were obtained with conventional ML models using BOW features. This shows the power of simple classifiers on the sentiment classification task. BERT outperforms the naïve Bayes classifier by a large margin on the SST dataset, and a smaller margin on the IMDb dataset, however, comparing with the results of [4] shows that the fine-tuned BERT in this study probably did not realize the full potential of its design.

The classification decisions of naïve Bayes can be explained in terms of concrete rules, and while studies give some insight on the knowledge encoded in BERT, a solid interpretation of the model and its predictions is not available. Depending on the performance requirements one may very well consider developing a naïve Bayes model in a sentiment classification application. As shown in this study, the subjective filtering technique seems to be an effective method of boosting performance. The naïve Bayes model clearly has limits as it only learns word distributions, which may also generate unexpected behaviour. BERT on the other hand learns to recognize actual linguistic constructs and word representations through the self-attention mechanism. This was indicated by the higher quality predictions on 5-class SST dataset as seen in Figure 1, where BERTs predictions were closer to the true class than the naïve Bayes BOW classifiers.

There are several questions to be addressed in future work, including parameter optimization for the subjectivity filters, fine-tuning BERT or its successors. This study focused on a document and sentence level sentiment analysis, the next step would be to associate an opinion to different entities mentioned in the reviews. An aspect based approach could further help filter out irrelevant information and perhaps improve performance the movie sentiment prediction task.

Appendix

A. Self-Attention

To see that the matrix operations in (9) make sense we expand the matrix product of the keys and queries disregarding the transformations and get

$$\mathbf{Q}^T\mathbf{K} = (\mathbf{q}_1, ..., \mathbf{q}_N)^T \times (\mathbf{k}_1, ..., \mathbf{k}_N), \tag{11}$$

where

$$\mathbf{Q} = Q\mathbf{X} = Q(\mathbf{x}_1, ..., \mathbf{x}_N) = (\mathbf{q}_1, ..., \mathbf{q}_N), \tag{12}$$

and similarly for $\mathbf{K}$. The matrix product amounts to

$$\mathbf{Q}^T\mathbf{K} = \begin{pmatrix} \mathbf{q}_1^T\mathbf{k}_1 & \mathbf{q}_1^T\mathbf{k}_2 & \dots & \mathbf{q}_1^T\mathbf{k}_N \\ \mathbf{q}_2^T\mathbf{k}_1 & \mathbf{q}_2^T\mathbf{k}_2 & \dots & \mathbf{q}_2^T\mathbf{k}_N \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{q}_N^T\mathbf{k}_1 & \mathbf{q}_N^T\mathbf{k}_2 & \dots & \mathbf{q}_N^T\mathbf{k}_N \end{pmatrix}, \tag{13}$$

and applying the softmax function row-wise gives the weights as in (8).

B. Note on the Second Stanford Dataset

The dataset by [20] that was extracted from Rotten Tomatoes contained in the form on the website a few encoding errors when downloaded by the author. The file `datasetSentences.txt` contained several special characters (é,è,ü,í, etc.) that had to be recovered, and at some places before time quantities the character Â had been inserted. The file was needed to recover the sentiment scores of each phrase for all of the sentences in the dataset. The authors of [20] had also (correctly) replaced left and right parenthesis by the tokens `-LRB-` and `-RRB-`, which had to be replaced by parenthesis for all phrases in the dictionary to be properly recovered.

C. Review Polarity

The following are examples of reviews from the IMDb dataset that did not quite correspond to their classes.

Positive (7 stars): "*The production quality, cast, premise, authentic New England (Waterbury, CT?) locale and lush John Williams score should have resulted in a 3-4 star collectors item. Unfortunately, all we got was a passable 2 star "decent" flick, mostly memorable for what it tried to do.....bring an art house style film mainstream. The small town locale and story of ordinary people is a genre to itself, and if well done, will satisfy most grownups. Jane Fonda was unable to hide her braininess enough to make her character believable. I wondered why she wasn't doing a post doctorate at Yale instead of working in a dead end factory job in Waterbury. Robert DiNiro's character was just a bit too contrived. An illiterate, nice guy loser who turns out to actually be, with a little help from Jane's character, a 1990 version of Henry Ford or Thomas Edison.

This genre has been more successfully handled by "Nobody's Fool" in the mid 90s and this year's (2003) "About Schmidt." I wish that the main stream studios would try more stuff for post adolescents and reserve a couple of screens at the multi cinema complexes for those efforts.

I'll give it an "A" for effort.*"

Positive (7 stars): "*I am sad to say that I disagree with other people on this Columbo episode. Death Lends a Hand is frankly kind of a boring Columbo to me. After a few times, I get bored and changed the channel. I still love Robert Culp and Patricia Crowley and Ray Milland in their roles but the story was weaker in this episode than in the others. First, Robert Culp plays an investigator for Ray Milland's character. He hires him to investigate his young pretty wife played by Patricia Crowley to see if she is having an affair. In return, Culp's character blackmails the cheating wife who plans to expose his scheme to her husband ruining his career. Out of anger, Culp kills her by striking her in the face and setting the up the body elsewhere. I don't know. Maybe I just didn't care for this one at all. Of course, Columbo gets him in the end. It's just the question of how.*"

REFERENCES

[1] R. Feldman, "Techniques and applications for sentiment analysis," Commun. ACM, April 2013, vol. 56, pp. 82-89, doi: 10.1145/2436256.2436274.

[2] B. Pang and L. Lee, "A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts," Proc. 42nd Annu. Meeting Assoc. Comput. Linguistics, 2004, doi: 10.3115/1218955.1218990.

[3] V. Raychev and P. Nakov, "Language-independent sentiment analysis using subjectivity and positional information," CoRR,2019.

[4] J. Devlin, M. Chang, K. Lee and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," Proc. 2019 Conf. of the North Amer. Chapt. Assoc. Comput. Linguistics, Minneap., MN, pp. 4171–4186, June 2018.

[5] T. Bolukbasi, A. Pearce, A. Yuan, A. Coenen, E. Reif, F. Viégas and M. Wattenberg, "An interpretability illusion for BERT," CoRR, April 2021.

[6] J. Li and L. Qiu, "A sentiment analysis method of short texts in microblog," 2017 IEEE Int. Conf. on Comput. Sci. and Eng. and IEEE Int. Conf. Embedded and Ubiquitous Comput., vol. 1, pp. 776–779, 2017.

[7] A. Radford, K. Narasimhan, T. Salimans, and I Sutskever, "Improving language understanding by generative pre-training", June 2018.

[8] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, L. Zhang, W. Han, M. Huang, Q. J., Y. Lan, Y. Liu, Z. Liu, Z. Lu, X. Qiu, R. Song, J. Tang, J. Wen, J. Yuan, W. Zhao and J. Zhu, "Pre-trained models: past, present and future" AI Open, 2021, doi: 10.1016/j.aiopen.2021.08.002.

[9] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, "Deep contextualized word representations" Proc. 2018 Conf. North Amer. Chapt. Assoc. Comput. Linguistics, New Orleans, LA, 2018, pp. 2227–2237, doi: 10.18653/v1/N18-1202.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L . Kaiser, I. Polosukhin, "Attention is all you need" Proc. 31st Int. Conf. Neural Inf. Process. Syst., Long Beach, CA, pp. 6000–6010, 2017.

[11] B. Pang, L. Lee, S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques" Proc. 2002 Conf. Empirical Methods Natural Lang. Process., pp. 79–86, July 2012.

[12] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing" IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 2, pp. 604–624, 2020.

[13] D. Jurafsky and J. H. Martin, "Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition", USA, Prentice Hall PTR, ch. 9, 2000.

[14] The Royal Society, "Explainable ai: the basics", 2019.

[15] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: towards story-like visual explanations by watching movies and reading books" 2015 IEEE Int. Conf. Comput. Vision, Los Alamitos, CA, December 2015, pp. 19–27, doi: 10.1109/ICCV.2015.11.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition" 2016 IEEE Conf. Comput. Vision Pattern Recognit., 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[17] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization" arXiv preprint, 2016, arXiv:1607.06450.

[18] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis" Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Portland, OR, pp. 142–150 June 2011.

[19] Q. Le, and T. Mikolov, "Distributed representations of sentences and documents" Proc. 31st Int. Conf. Machine Learning, Bejing, China, vol. 32, no. 2 pp. 1188–1196, 2014.

[20] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank" Proc. 2013 Conf. Empirical Methods Natural Lang. Process., Seattle, WA, pp. 1631–1642, October 2013.

[21] G. E. Dahl, R. P. Adams, and H. Larochelle, "Training restricted boltzmann machines on word observations" Proc. 29th Int. Conf. Machine Learning, Madison, WI, pp. 1163–1170, 2012.

[22] S. Wang, C. D. Manning, "Baselines and bigrams: simple, good sentiment and topic classification" Proceedings 50th Annu. Meeting Assoc. Comput. Linguistics, Jeju Island, Korea, pp. 90–94, 2012.

[23] B. Pang and L. Lee, "Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales" Proc. ACL., pp. 115–124, 2005.

[24] J. Hong, and M. Fang, "Sentiment analysis with deeply learned distributed representations of variable length texts" Stanford University Report, pp. 1–9, 2015.

[25] J. B. Orlin, "Max flows in O(Nm) time, or better" Proc. 45th Annu. ACM Symp. Theory Comput., pp. 765–774, Palo Alto, CA, 2013, doi: 10.1145/2488608.2488705.

[26] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang, "A survey on neural network interpretability" IEEE Trans. Emerg. Topics Comput. Intell., 2021, vol 5., no. 5, pp. 726–742, doi: 10.1109/TETCI.2021.3100641.

[27] F. Fan, J. Xiong, M. Li, and G. Wang, "On interpretability of artificial neural networks: a survey" IEEE Trans. Radiat. Plasma Med. Sci., 2021, vol. 6, no. 6, pp. 741-760, doi: 10.1109/TRPMS.2021.3066428.

[28] X. Song, A. Salcianu, Y. Song, D. Dopson, D. Zhou, "Fast WordPiece tokenization" Proc. 2021 Conf. Empirical Methods Natural Lang. Process. Online and Punta Cana, Dom. Rep., November 2021, pp. 2089–2103, doi: 10.18653/v1/2021.emnlp-main.160.

[29] S. Vanaja, and M. Belwal, “Aspect-level sentiment analysis on e-commerce data” 2018 Int. Conf. Inventive Res. Comput. Appl., 2019, pp. 1275–1279, doi: 10.1109/ICIRCA.2018.8597286.
[30] J. C. Platt, “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods”, Advances in Large Margin Classifiers, MIT Press, 1999, pp. 61–74.
[31] N. Garg, L. Schiebinger, D. Jurafsky, and J. Zou, “Word embeddings quantify 100 years of gender and ethnic stereotypes” Proc. Nat. Acad. Sci., 2018, vol. 115, no. 16, pp. E3635–E3644, doi: 10.1073/pnas.1720347115.
[32] N. Durrani, H. Sajjad, F. Dalvi, Y. Belinkov, “Analyzing individual neurons in pre-trained language models” Proc. 2020 Conf. Empirical Methods Natural Lang. Process., Online, November 2020, pp. 4865–4880, doi: 10.18653/v1/2020.emnlp-main.395.