Bayesian Evaluation Of Text Classification Models

Shahrukh Iqbal
LiU ID: shaiq681
Linköping University
shaiq681@student.liu.se

Abstract

When evaluating text classification models, we want to be certain about the superiority of one classifier over another. In the area of text classification it has become a norm to apply Null Hypothesis Significance Test(NHST) to statistically state and compare classifier performance. But, a frequentist approach has its own limitations and fallacies. In this report, we reflect on limitations posed by NHST. We also implement a novel Bayesian approach for evaluating text-classification models. We use a benchmark dataset and create several shallow and deep models for demonstrating the difference between the two evaluation approaches.

1 Introduction

With the advent of novel architectures and free exchange of information; a new era has emerged in NLP domain with several different methods to choose from. While making a choice, we want to ensure whether a proposed model is actually better than other models described in the literature. Thus a need for a statistical analysis naturally arises. So, in this report we focus on answering the question that "*Whether a new proposed model A is statistically better than B?*". We want to look at the most comprehensive way for doing so. Since the publishing of Yang and Liu's seminal SIGIR-1999 paper, Null-Hypotheis Significance Test(NHST) has become a standard for statistically comparing superiority of one model over another. Though NHST has been used for several decades, we have started to see some resistance for using this approach. Some fields have warned against the use of *p-value* (Wasserstein and Lazar, 2016) and there have been some critical arguments using NHST (Carver, 1978) in general and also in NLP domain(Koplenig, 2019).

This report focuses on discussing NHST and presents case for testing hypothesis from a Bayesian perspective. In Section 2 we describe NHST and Bayesian hypothesis testing approach specific to "*Multi-Label Text Classification*". Then, in Section 3 we describe the data which was also used in Yang and Liu's paper. In Section 4, we discuss various models and feature engineering techniques used for model comparison. In Section 5 we present results using *frequentist* and *Bayesian* evaluation strategy. In Section 6 we discuss both the evaluation strategies. In Section 7 we draw conclusions.

2 Theory

2.1 Evaluation Metrics: Precision, Recall and F-measure

Table 1: Confusion Matrix

		Assignment z	
		1	0
Label l	1	TP	FN
	0	FP	TN

In this section we define the evaluation metrics that will be used for assessing the performance of a classifier system $\mathcal{G}$. We consider that each document is associated with a binary label l. We assign integer 1 to the document if it belongs to true label else 0. Let the system produce assignment z specifying whether it believes that the assignment is correct or not. Table 1 presents the outcome of Assignment z for label l by a classifier $\mathcal{G}$ in a *confusion matrix* where 1 and 0 stands for correct and incorrect assignment, TP stands for true positive count, TN for true negative count, FP for false positive count, FN for false negative count. Using these counts, we can compute the **precision**(P), **recall**(R):

$$P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN} \tag{1}$$

The harmonic average of precision and recall defines the F-measure:

$$F_\beta = (1+\beta^2)\frac{PR}{R+\beta^2 P} \tag{2}$$

Goutte and Gaussier provide a probabilistic interpretation of precision and recall as defined in (eq. 1). Precision can be interpreted as the probability that a label is relevant given that it is predicted by a system, while Recall is the probability for predicting a true label(Goutte and Gaussier, 2005)[1]:

$$\begin{aligned} P &= \mathbf{Pr}(l=1|z=1) \\ R &= \mathbf{Pr}(z=1|l=1) \end{aligned} \tag{3}$$

2.2 Frequentist Hypothesis Assessment

When trying to evaluate performance of a classifier model, we need to compare it with a baseline model. Given, we have a new classifier A and a baseline classifier B and a metric $\mathcal{M}$, we want to assess how well the model A does better than model B on a test set $\mathbf{x} = \mathbf{x_1}, \ldots, \mathbf{x_N}$. Let us define the difference of performance of classifier A and B on $\mathbf{x}$ as:

$$\delta(\mathbf{x}) = \mathcal{M}(A, \mathbf{x}) - \mathcal{M}(B, \mathbf{x}) \tag{4}$$

Now, we want to know that given a new, independent data $\hat{\mathbf{x}}$, how likely it is that classifier A is better than classifier B. i.e., if A is better than B on $\mathbf{x}$ by a margin of $\delta(\mathbf{x})$ would A be better than B *assuming A is not better than B on the population as a whole* (Berg-Kirkpatrick et al., 2012). This assumption is known as the *null hypothesis* H_0. The alternate of *null hypothesis* is H_1 and both the hypothesis can be formally stated as:

$$\begin{aligned} H_0 &: \delta(\mathbf{x}) \leq 0 \\ H_1 &: \delta(\mathbf{x}) > 0 \end{aligned} \tag{5}$$

2.2.1 Null Hypothesis Significance Test(NHST)

In NHST, we estimate the likelihood of $\mathbf{Pr}\left(\delta(\mathbf{X}) > \delta(\mathbf{x})|H_0\right)$ where $\mathbf{X}$ is a random variable over possible test sets of size n (Berg-Kirkpatrick et al., 2012). This estimation of probability is known as *p-value*. Since, we will be estimating metrics such as precision, recall and F1; we assume them to be not normally distributed (Berg-Kirkpatrick et al., 2012). We implement a paired bootstrap test (Efron and Tibshirani, 1994) that relies on drawing large number of small samples with replacement from the original sample $\mathbf{x}$. We implement Berg-Kirkpatrick et al.'s version of paired-bootstrap sampling to estimate *p-value*. Given that we have b bootstrap samples of size n, we estimate *p-value* as:

$$\textit{p-value} = \frac{1}{b}\sum_{i=1}^{b} \mathbf{1}\left(\delta(\mathbf{x}^{(i)}) \geq 2\delta(\mathbf{x})\right) \tag{6}$$

In equation (6), we use $\delta(\mathbf{x}^{(i)}) \geq 2\delta(\mathbf{x})$ and not $\delta(\mathbf{x}^{(i)}) \geq \delta(\mathbf{x})$ because the average of $\delta(\mathbf{x}^{(i)})$ will not be 0 as $\mathbf{x}^{(i)}$ were sampled from $\mathbf{x}$ (Berg-Kirkpatrick et al., 2012). Rather, the average will be around $\delta(\mathbf{x})$ (Berg-Kirkpatrick et al., 2012).

2.2.2 Frequentist Estimation: Confidence Interval(CI)

CIs are used to estimate uncertainty of the parameter $\delta(\mathbf{x})$. For a *significance* of 0.05, equation (7) estimates range within 95% CI such that the test based on *p-value* in equation (6) is not rejected.

$$\mathbf{Pr}\left(\delta(\mathbf{X}) > \delta(\mathbf{x})|H_0\right) \geq 0.05 \tag{7}$$

2.2.3 Issues with Frequentist Hypothesis assessment

(Dror et al., 2018) raise the issue that frequentist significance tests share the assumption that data is independent and identically distributed. (Dror et al., 2018) state that this assumption is rarely true in NLP setup. In addition, *p-value* based tests do not assess the probability of the hypothesis (Benavoli et al., 2017). Furthermore, the definition of *p-value* depends on the distribution of sample data and sample size n. (Kruschke, 2010) show that this can result in unexpected test outcomes even when final set of observations is identical.

In case of CI, while CIs provide an estimated range of the value being tested, the range does not have a *probabilistic* interpretation (Sadeqi Azer et al., 2020). Furthermore, Sadeqi Azer et al.'s survey of ACL'18 show that only 6(out of 439) papers had reported using CIs.

2.3 Bayesian Hypothesis Assessment

Bayesian modeling focuses on prior and posterior distribution of a parameter $\boldsymbol{\theta}$. In NLP models, $\boldsymbol{\theta}$ can be *actual* mean of a classifiers performance metric such as Precision, Recall or F1. Here *actual*

[1]The notation $\mathbf{Pr}(x)$ is used to denote Probability Density Function for continuous variables and Probability Mass Function for discrete variables

means that it is inherent to the classifier and latent in nature (Sadeqi Azer et al., 2020).

In Bayesian framework, we encode our prior belief and assumptions in the form of a *prior* distribution $\mathbf{Pr}(\boldsymbol{\theta})$. Generally, this assumption represents common belief about the parameters of the model. For Instance, we can have a prior belief that Precision is uniformly distributed between [0,1]. This corresponds to having no prior belief on how high or low the precision can be.

$\boldsymbol{\theta}$ is formally incorporated in a hierarchical model in the form of a *likelihood function* $\mathbf{Pr}(\mathbf{x}|\boldsymbol{\theta})$. This leads to connecting latent variables of the model with the observed data. Using Bayes rule, we can infer the *posterior* distribution of the parameter conditioned on the observed data: $\mathbf{Pr}(\boldsymbol{\theta}|\mathbf{x}) = \frac{\mathbf{Pr}(\mathbf{x}|\boldsymbol{\theta})\mathbf{Pr}(\boldsymbol{\theta})}{\mathbf{Pr}(\mathbf{x})}$.

The posterior distribution incorporates information regarding data as well as th prior. The posterior distribution can now be used to estimate $\boldsymbol{\theta}$ and also *uncertainty* regarding the estimate. The mode of the posterior distribution(*maximum a posteriori*) gives a point estimate of $\boldsymbol{\theta}$ (Gelman et al., 2014).

2.3.1 Bayesian Hypothesis Testing

To support the claim in equation (4), we phrase the hypothesis in a Bayesian framework : "*Assume a prior belief (a probability distribution) w.r.t. the inherent accuracies of the classifiers. Given the empirically observed accuracies, the probability* ***(posterior interval)*** *that the inherent accuracy of A exceeds that of B by a margin of 1% is at least 95%*" (Sadeqi Azer et al., 2020).

2.3.2 Posterior Distribution of P, R and F_1

We use Goutte and Gaussier's results for stating the posterior distribution of Precision(P), Recall(R) and F_1($\beta = 1$ in equation (2)),conditioned on the test-data $\mathbf{x}$:

$$P|\mathbf{x} \sim Be(TP + \lambda, FP + \lambda) \tag{8}$$

$$R|\mathbf{x} \sim Be(TP + \lambda, FN + \lambda) \tag{9}$$

$$F_1 = \frac{U}{U+V} \; with \begin{cases} U \sim \Gamma(TP + \lambda, 2h) \\ V \sim \Gamma(FP + FN + 2\lambda, h) \end{cases} \tag{10}$$

In equation (8), we present the posterior distribution of Precision which is a Beta distribution that depends on TP,FP and prior parameter λ. Similarly, equation (9) presents the posterior distribution of Recall which depends on parameters TP,FN and λ. Equation (10) presents Goutte and Gaussier's result for posterior distribution of F_1 score. We see that U in (10) has a Gamma distribution that depends on TP, λ and Gamma distribution's shape parameter h. Similarly, V in (10) has a Gamma distribution that depends on FP, FN, λ and h

We use $\lambda = 1/2$ because it is a non-informative prior (Goutte and Gaussier, 2005). We use Goutte and Gaussier, 2005's default shape parameter value $h = 1$.

2.3.3 Bayesian Decision

To make a judgement on superiority of one classifier over another we use a posterior probability distribution of (4). We have two competing systems with outcomes $D^{(A)} = \left(TP^{(A)}, TN^{(A)}, FP^{(A)}, FN^{(A)}\right)$ and $D^{(B)} = \left(TP^{(B)}, TN^{(B)}, FP^{(B)}, FN^{(B)}\right)$. we can define the probability of $\mathbf{Pr}(A > B), \mathbf{Pr}(A = B)$ and $\mathbf{Pr}(A < B)$ as:

$$\begin{aligned} \mathbf{Pr}(A > B) &= \mathbf{Pr}(\delta(\mathbf{x}) > 0) \\ \mathbf{Pr}(A = B) &= \mathbf{Pr}(\delta(\mathbf{x}) = 0) \\ \mathbf{Pr}(A < B) &= \mathbf{Pr}(\delta(\mathbf{x}) < 0) \end{aligned} \tag{11}$$

To evaluate probabilities defined in (11), we use Monte Carlo simulation. We create large samples of size L from the distributions of $F_1^{(A)}$ and $F_1^{(B)}$, using the Gamma variates as shown in (10) (Goutte and Gaussier, 2005). Let us write the samples as $\{f_i^{(A)}\}_{i=1,\ldots,L}$ and $\{f_i^{(B)}\}_{i=1,\ldots,L}$ and represent the difference in performance between individual sample as $\{\delta(\mathbf{x})\}_{i=1,\ldots,L}$ which gives the posterior distribution of $\delta(\mathbf{x})$.

We need to define a Region of Practical Equivalence(ROPE) when applying the bayesian models for comparison (Zhang et al., 2015).The intuition behind ROPE is that if $\{f_i^{(A)}\}_{i=1,\ldots,L}$ is close to $\{f_i^{(B)}\}_{i=1,\ldots,L}$ then their difference is *practically* zero. For practical purposes we define the ROPE to be in [-0.05,0.05] i.e., the region of difference in posterior $\delta(\mathbf{x})$ is practically zero if it lies in ROPE. This also means that an improvement of less than 1% is not considerably notable (Kruschke, 2018).

We use Zhang et al.'s method for making discrete decision about how classifier A compare with B. We use a 95% Highest Density Interval(HDI) of $\delta(\mathbf{x})$ and ROPE of $\delta(\mathbf{x})$ for making a decision. Using 95% HDI, we obtain a useful summary of

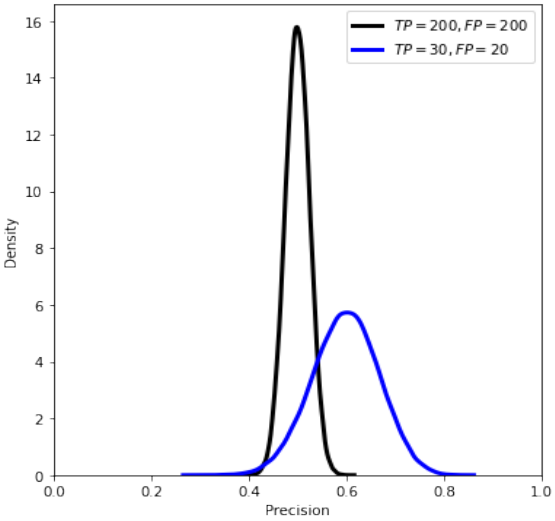


Figure 1: Precision distribution of two systems. While ▬ outperforms ▬, ▬ is much less variable.

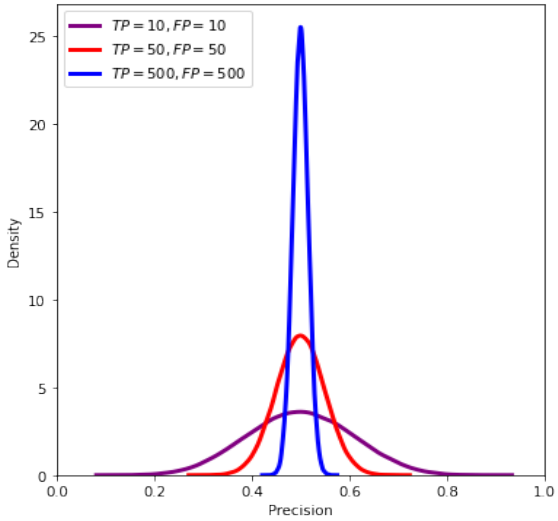


Figure 2: Posterior Precision distribution of three systems with same value for deterministic precision1.With an increase in sample size the variation in Precision decreases. $\mathbf{Var}(\mathbf{System}_{▬}) < \mathbf{Var}(\mathbf{System}_{▬}), \mathbf{Var}(\mathbf{System}_{▬})$

where the bulk of most credible values of $\delta(\mathbf{x})$ falls (Zhang et al., 2016). HDI is useful because total mass of points inside HDI is 95% of the distribution and every value inside HDI has a higher probability density than any value outside the HDI (Zhang et al., 2016). Using HDI and ROPE the following decisions can be made:

- If the HDI sits fully within the ROPE, A is practically equivalent($\sim$) to B (Zhang et al., 2016)(as illustrated in Figure 3b);
- If the HDI sits fully at the left or right of the ROPE then A is slightly worse(<<) or better(>>) than B(as illustrated in Figures 3a and 3c).

3 Data

3.1 Reuters21587

In order to evaluate the effectiveness of *frequentist* and *bayesian* strategy, we have chosen Reuters-21587 corpus. The Reuters-21587 is a popular data-set that was also used in seminal SIGIR paper that introduced NHST to NLP community(Yang and Liu, 1999) for bench-marking TC Algorithms. We have removed the documents that were unlabelled and selected categories which have at least one label-per document in training and one label-per-document in test set. The removal of unlabelled documents resulted in a corpus with 115 different categories in both training and test sets. We also removed the documents that did not belong to any of the 115 categories. This process resulted in a corpus with 7775 documents in the training set and 3019 documents in the test set. The training set has a vocabulary size of 23669 words which after stop-words removal was reduced to 23388 unique words. We used Pedregosa et al.'s `scikit-learn` default stop-words list for stopwords removal. As we shall see, the stop-words were removed only for models using 'Term Frequency-Inverse Document Frequency' as a method of feature engineering.

We used `pandas` library to save training and test data into a tabular format. Each table of training and test data consisted of two columns: 1). `text` for storing the features. 2). `topics` for storing the labels. In Table 2 we show the representation of training and test-data. It can be seen that several `text` can have multiple labels in the `topics` column.

4 Methodology

4.1 Text Pre-Processing

Since the given data consists of unstructured text documents, we need to convert the unstructured data into a structured feature space. This transformation is necessary if we want to apply mathematical models on textual data. This section describes the methods that were used for extracting features and pre-processing Reuters-21587 corpus.

In the subsection, we shall discuss the methods implemented for segmenting text into its lexical expressions. On word level, we implemented *tok-*

text	topics
Media group John Fairfax Ltd ...	["earn"]
The Bank of England said it provided the market with...	["money-fx", "interest"]

Table 2: Examples from the REUTERS-21587 Dataset

enization for lexical representation. On document level, we implemented sparse as well as dense representation methods.

4.1.1 Term Frequency-Inverse Document Frequency(TF-IDF)

TF-IDF is a method that is a combination of **Term Frequency**(TF) and **Inverse Document Frequnecy**(IDF). The use of TF along with IDF results in assigning higher weight to rarer words. This is done by setting every document-term matrix element equal to value w which is a product of TF and IDF for each token.

$$w = tf \times idf \tag{12}$$

Tokunaga and Makoto, 1994 have shown that the first term in equation 12 improves the recall and the second term improves the precision of the embedded word w . We used `scikit-learn`'s `TfidfVectorizer`[2] with parameters `analyzer='word'` and `stop_words='english'` for transforming raw features to a *feature-matrix*.

This results in a $[7775 \times 23669]$ *feature-matrix* for training-data and $[3019 \times 23388]$ *feature-matrix* for test-data. The *feature-matrix* created using TF-IDF is sparse in nature.

4.1.2 DistilBERT Based Feature Extraction(DBF)

We use DistilBERT(DBT) (Sanh et al., 2019) version of the Bidirectional Encoder Representation from Transformers(BERT)(Devlin et al., 2018) to create word-embedding based features.

Firstly, we tokenize feature column `text` in the training-set and test-set as one-hot vectors. Next, we use `Huggingface`'s[3] pre-trained DBT model to convert one-hot vectors to embeddings with positional encodings. Then, we freeze the DBT model weights and pass the word-embeddings through the encoder layer to yield a *hidden state* with respect to each input token. This results in transformation of input-feature space to a 768-dimensional space.

We obtain a $[7775 \times 768]$ *feature-matrix* for training-data and $[3019 \times 768]$ *feature-matrix* for test-data. The *feature-matrix* created using DBF is dense in nature.

4.1.3 One-Hot Encoding(OHE)

As it is shown in Table 2, we can have several output for a target within `topics` column. So, we use a *one-hot* vector representation of target-data. For the given dataset which consists of 115 unique classes, we use `scikit-learn`'s `MultiLabelBinarizer` for labeling the target. This transforms the target into a 115-dimension vector with value 1 for labels that are in the target and 0 for rest of the labels.

4.2 Classifiers

4.2.1 Baseline(B)

We establish a baseline classifier to benchmark the performance of rest of the classifiers. We use a stratified strategy to randomly sample one-hot vectors from a multinomial distribution that is parameterized by the prior probabilities of the classes.

4.2.2 Naïve Bayes(NB)

For creating an NB model, we use feature-matrix generated using TF-IDF transformation as input. For multilabel estimation, we use a *one-vs-rest* approach in which we construct K separate models, in which k^{th} model $M_k(\mathbf{x})$ is trained using the labels from classes C_k as the positive example and the data from the remaining $K-1$ classes as the negative example (Vapnik, 1999).

For the current task, we have used a *multinomial* version of NB (McCallum et al., 1998). We use a *laplace smoothing* factor of 1. We used the proportion of each class label C_k in the training-data as the prior probability.

4.2.3 K-Nearest Neighbors(kNN)

For creating a kNN model, we use feature-matrix generated using TF-IDF transformation as in-

[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[3] https://huggingface.co/distilbert-base-uncased

put.For estimation, we use *one-vs-rest* approach. We set the k in kNN to 45 (Yang and Liu, 1999). We set the metric to *minkowski* for measuring the distance and we set the weighting scheme to *uniform* i.e., the kNN model gives equal weightage to neighbors.

4.2.4 Support Vector Machine(SVM)

For creating an SVM model, we use feature-matrix generated using TF-IDF transformation as input. For estimation, we use *one-vs-rest* approach. We set the *reguarization paramemter* $C = 1$ with a *linear* kernel (Yang and Liu, 1999).

4.2.5 Neural Network(NNet)

We use feature-matrix generated using TF-IDF transformation as input. For estimation, we use *one-vs-rest* approach. We define an NNet with 1 *hidden layer*, 64 *hidden units*(Yang and Liu, 1999). We set the activation function to Relu, batch size to 200, learning rate of 1e-3. We use Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

4.2.6 Decision Tree(DT)

For creating a DT model, we use feature-matrix generated using DBF transformation as input. For estimation, we use *one-vs-rest* approach. We use *gini index* to grow the tree. We did not implement pruning methods.

4.2.7 Random Forest(RF)

For creating an RF model, we use feature-matrix generated using DBF transformation as input. For estimation, we use *one-vs-rest* approach. We implement an RF of 100 DT's. We use *gini index* to grow the trees. We used bootstrap samples. We did not use *out-of-bag* samples (Friedman et al., 2001a).

4.2.8 AdaBoost(AB)

For creating an AB model, we use feature-matrix generated using DBF transformation as input. For estimation, we use *one-vs-rest* approach. We implement a multiclass version of AdaBoost algorithm (Hastie et al., 2009). We use DT as a base *week classifier* (Friedman et al., 2001b). We set the number of estimators to 50.

4.2.9 DistilBert(DBERT)

We use a DistilBERT model (Sanh et al., 2019) for training. We use Wolf et al.'s pre-trained DBERT model for the task. We fine-tune the DBERT model on tokenized training dataset Sun et al.. We use the DBERT model with a hidden size of 768, 6 Transformer blocks and 6 self-attention heads. We add a dropout layer and a hidden size of 115 in the output layer.

We fine-tune the DBERT and set the training batch size to 16 and test batch size to 8, maximum sequence length to 500, learning rate of 1e-4. We use Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We set the number of epochs to 5. We use binary cross-entropy loss for training.

5 Results

Table 3: Performance summary of Classifiers. **miP** : micro-avg Precision; **miR** : micro-avg Recall; **miF1** : micro-avg F1

Classifier	miP	miR	miF1
Baseline	0.166	0.1656	0.1660
NB	**0.9934**	0.3632	0.5319
kNN	0.9931	0.1536	0.2660
SVM	0.9437	**0.7126**	**0.8120**
NNet	0.9414	0.6562	0.7733
DT	0.5771	0.5774	0.5773
RF	0.9411	0.5037	0.6562
AB	0.7321	0.5753	0.6443
DBERT	0.8996	0.6845	0.7775

Table 3 summarizes global average performance of different classifiers on the test set. We made following observations:

- For **miP**, classifier NB has the highest performance score(0.9934) that is in contrast with the reported results by Yang and Liu. This can be due to difference in feature engineering approaches. All the models outperform the Baseline model. The lowest **miP** score is reported by classifier AB.

- For **miR**, classifier SVM has the highest performance score(0.7126) that is slightly lower than results reported by Yang and Liu. The lowest **miR** score is reported by classifier kNN(0.1536). Also, kNN is unable to outperform Baseline model.

- For **miF1**, classifier SVM is the best performing model. While all the models outperform Baseline, kNN is slightly better than Baseline.

In Table 4, we compare two classifiers A and B on micro-average F_1 score using NHST. We

report the *p-value* estimated using equation (6), CI estimated using equation (7) and decision based on results of paired-bootstrap test (Berg-Kirkpatrick et al., 2012). For Bootstrap sampling we set $b = 10,000$. We reject the null-hypothesis (5) for *p-value*> 0.05. We make a decision "<<" or ">>" if *p-value* ≤ 0.01; "<" or ">" if $0.01 <$ *p-value* ≤ 0.05; "$\sim$" if *p-value* > 0.05. We list the most notable observations:

- NB outperforms kNN at a significance level of 0.0;
- SVM outperforms {RF, AB, DBERT} at a significance level of 0.0;
- NNet outperforms {DT, RF, AB} at a significance level of 0.0;
- We fail to reject the null-hypothesis (5) for A =NNet and B =DBERT at a significance of 0.811;
- RF slightly outperforms AB at a significance of 0.02.

Table (5), presents the results of Bayesian Hypothesis Test as defined in section 2.3.1. We create separate Monte Carlo samples with $L = 10,000$ for classifier A and classifier B using equation (10) for all the 115 labels. We take per-sample average of F_1 over all the 115 labels to create a posterior distribution of δ(miF1). We use decision-making rules defined in section 2.3.3. We use `PyMC3`[4] for estimating 95% HDI of the posterior and for defining ROPE. We set the ROPE between [-0.05,0.05]. We observe that:

- NB is equivalent to {kNN, DT};
- {SVM, NNet, RF, AB, DBERT} clearly outperform NB;
- SVM outperform other classifiers. There is a strong evidence that SVM is superior to {NNet, DT,RF, AB, DBERT};
- There is a strong evidence that NNet is superior to {DT,RF};
- RF is either inferior or equivalent to {AB, DBERT}.

Figure 3 shows the posterior distribution of δ(miF1) for cases:"<<" in 3a, "$\sim$" in 3b and ">>" in 3c.

[4] https://docs.pymc.io/en/v3/about.html

6 Discussion

6.1 Automatic Decision Making

Bayesian approach results in more sensible decision-making because we are using actual probability of models as defined in 11. This is in contrast with NHST which promotes binary thinking. For Instance, In Table 4 we fail to reject that SVM is better than NB because we have defined A =NB and B =SVM. In case we would have defined A =SVM and B =NB, we would be performing a different NHST and our interpretation of results would differ.

6.2 Effect of Sample Size

Taking a Bayesian view for parameters P, R and F_1, we can estimate the effect of sample-size. For example, in Figure 2, we see the effect of increased sample size clearly. While the deterministic Precision for the three different datasets is equal, the probability distribution is able to showcase effects of increasing the data size.

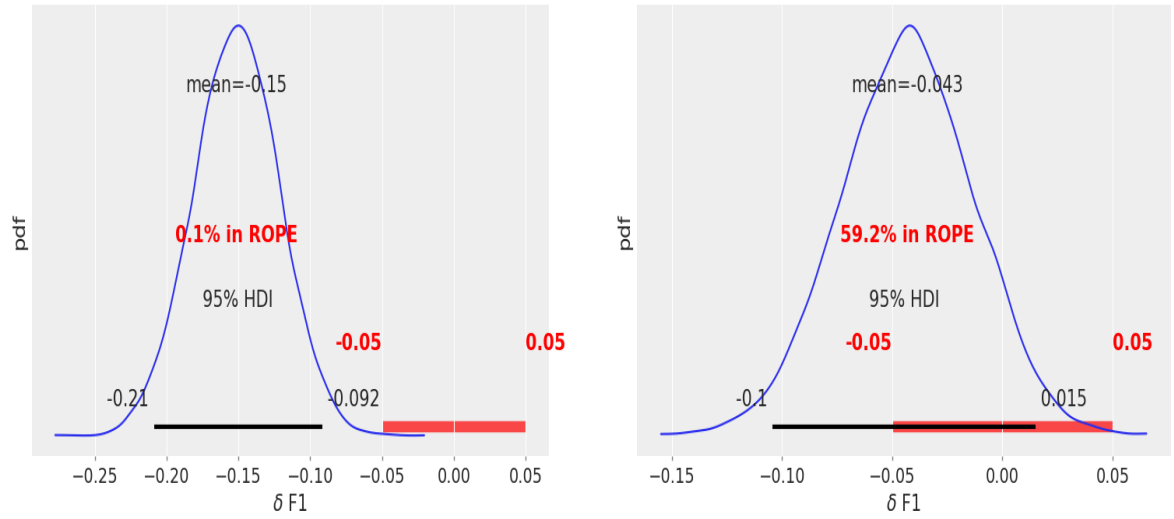
6.3 Difference in NHST's significance and ROPE

There is no principled way to decide the value of *significance* in NHST. We do not know whether a significance of 0.01 or a significance of 0.05 wil be a better choice. Also, from a practical point of view, the meaning of *significance level* is not intuitive (Benavoli et al., 2017).

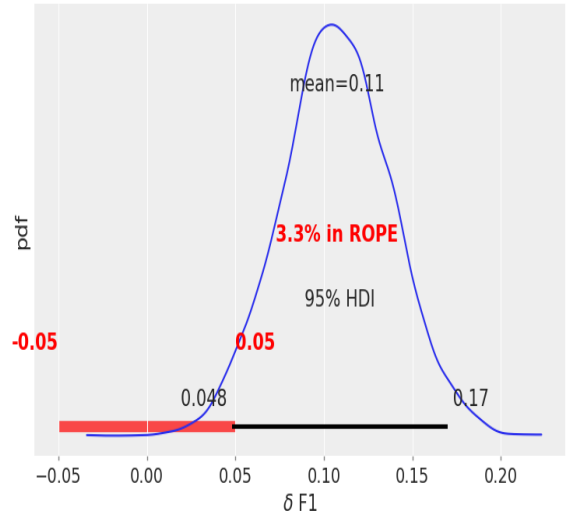
From a Bayesian perspective, we look at ROPE which is set by the user and is defined in terms of difference of performance of classifiers A and B. We find this to be more sensible estimation for posterior probability of null-hypothesis i.e., $\mathbf{Pr}(H_0|\mathbf{x})$ (Benavoli et al., 2017).

7 Conclusion

In this report we discussed the pitfalls of NHST. We discuss *frequentist* as well as *bayesian* approach from a theoretical perspective. We discussed *frequentist* as well as *bayesian* approach for comparing two models. We show the advantages of using Bayesian framework for performing a null-hypothesis test. We also discussed advantages of having a probabilistic view for classification metrics P,R and F1. Finally, we evaluated various models using *frequentist* and *bayesian* framework. In conclusion, we try to show a different framework for estimating classfier performance in NLP domain.



(a) Density plot for the difference of **miF1** between kNN and RF(b) Density plot for the difference of **miF1** between NB and DT



(c) Density plot for the difference of **miF1** between SVM and DBERT

Figure 3: Density plots for the difference of **miF1**

Table 4: NHST Results for miF1

A	B	p-value	CI	Decision
NB	kNN	0.0	[0.25,0.28]	>>
NB	SVM	1.0	[-0.29,-0.26]	∼
NB	NNet	1.0	[-0.26-0.23]	∼
NB	DT	1.0	[-0.06-0.02]	∼
NB	RF	1.0	[-0.13-0.11]	∼
NB	AB	1.0	[-0.12-0.09]	∼
NB	DBERT	1.0	[-0.26,-0.23]	∼
kNN	SVM	1.0	[-0.56,-0.53]	∼
kNN	NNet	1.0	[-0.52,-0.48]	∼
kNN	DT	1.0	[-0.33,-0.29]	∼
kNN	RF	1.0	[-0.4,-0.37]	∼
kNN	AB	1.0	[-0.39,-0.36]	∼
kNN	DBERT	1.0	[-0.53,-0.49]	∼
SVM	NNet	0.0	[-0.03,-0.04]	∼
SVM	DT	0.0	[0.22,0.25]	∼
SVM	RF	0.0	[0.14,0.17]	>>
SVM	AB	0.0	[0.15,0.18]	>>
SVM	DBERT	0.0	[0.03,0.04]	>>
NNet	DT	0.0	[0.18,0.20]	>>
NNet	RF	0.0	[0.10,0.12]	>>
NNet	AB	0.0	[0.11,0.14]	>>
NNet	DBERT	0.811	[-0.01,0.006]	∼
DT	RF	1.0	[-0.09,-0.06]	∼
DT	AB	1.0	[-0.08,-0.05]	∼
DT	DBERT	1.0	[-0.22,-0.18]	∼
RF	AB	0.02	[0,0.002]	>
RF	DBERT	1.0	[-0.13,-0.10]	∼
AB	DBERT	1.0	[-0.14,-0.12]	∼

References

Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. 2017. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea. Association for Computational Linguistics.

Ronald Carver. 1978. The case against statistical significance testing. *Harvard Educational Review*, 48(3):378–399.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. 2001a. *The elements of statistical learning*, volume 1, pages 592–593. Springer series in statistics New York.

Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. 2001b. *The elements of statistical learning*, volume 1, pages 337–338. Springer series in statistics New York.

Andrew Gelman, John B. Carlin, Hal S. Stern, Donald B. Rubin, Aki Vehtari, and David B. Dunson.

Table 5: Bayesian comparison of Average F_1 score on micro level

A	B	p(A<B)	p(A=B)	p(A>B)	HDI	Decision
NB	kNN	0.02	0.91	0.06	[-0.05,0.06]	$\sim$
	SVM	1	0.0	0.0	[-0.27,-0.15]	<<
	NNet	1	0.0	0.0	[-0.21,0.08]	<<
	DT	0.42	0.58	0.0	[-0.1,0.02]	$\sim$
	RF	0.6	0.4	0.0	[-0.11,-0.00]	<<
	AB	0.96	0.04	0.0	[-0.16,-0.04]	<<
	DBERT	0.96	0.04	0.0	[-0.16,-0.04]	<<
kNN	SVM	1	0.0	0.0	[-0.27,-0.16]	<<
	NNet	1	0.0	0.0	[-0.21,-0.09]	<<
	DT	0.49	0.51	0.0	[-0.11,0.01]	$\sim$
	RF	0.69	0.31	0.0	[-0.12,-0.01]	<<
	AB	0.97	0.03	0.0	[-0.17,-0.05]	<<
	DBERT	0.98	0.02	0.0	[-0.16,-0.05]	<<
SVM	NNet	0.0	0.32	0.68	[0.01,0.13]	>>
	DT	0.0	0.0	1.0	[0.11,0.23]	>>
	RF	0.0	0.0	1.0	[0.09,0.22]	>>
	AB	0.0	0.04	0.96	[0.05,0.17]	>>
	DBERT	0.0	0.03	0.97	[0.05,0.17]	>>
NNet	DT	0.0	0.06	0.94	[0.04,0.16]	>>
	RF	0.0	0.11	0.89	[0.03,0.15]	>>
	AB	0.0	0.61	0.39	[-0.22,0.10]	$\sim$
	DBERT	0.0	0.6	0.4	[-0.18,0.1]	$\sim$
DT	RF	0.13	0.85	0.02	[-0.08,0.05]	$\sim$
	AB	0.62	0.38	0.0	[-0.12,0.01]	<<
	DBERT	0.62	0.38	0.0	[-0.12,0.01]	<<
RF	AB	0.45	0.55	0.0	[-0.1,0.02]	$\sim$
	DBERT	0.42	0.0.58	0.0	[-0.1,0.02]	$\sim$
AB	DBERT	0.05	0.89	0.06	[-0.06,0.06]	$\sim$

2014. *Bayesian data analysis.*, Texts in statistical science series, pages 32–34. Chapman Hall/CRC.

C. Goutte and E. Gaussier. 2005. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *LECTURE NOTES IN COMPUTER SCIENCE*, pages 345 – 359.

Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. 2009. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360.

Alexander Koplenig. 2019. Against statistical significance testing in corpus linguistics. *Corpus Linguistics and Linguistic Theory*, 15(2):321–346.

John K. Kruschke. 2010. Bayesian data analysis. *WIREs Cognitive Science*, 1(5):658–676.

John K Kruschke. 2018. Rejecting or accepting parameter values in bayesian estimation. *Advances in Methods and Practices in Psychological Science*, 1(2):270–280.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer.

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

Erfan Sadeqi Azer, Daniel Khashabi, Ashish Sabharwal, and Dan Roth. 2020. Not all claims are created equal: Choosing the right statistical approach to assess hypotheses. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

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.

Takenobu Tokunaga and Iwayama Makoto. 1994. Text categorization based on weighted inverse document frequency. In *Special Interest Groups and Information Process Society of Japan (SIG-IPSJ*. Citeseer.

Vladimir Vapnik. 1999. *The nature of statistical learning theory*. Springer science & business media.

Ronald L. Wasserstein and Nicole A. Lazar. 2016. The asa statement on p-values: Context, process, and purpose. *The American Statistician*, 70(2):129–133.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49.

Dell Zhang, Jun Wang, Emine Yilmaz, Xiaoling Wang, and Yuxin Zhou. 2016. Bayesian performance comparison of text classifiers. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 15–24.

Dell Zhang, Jun Wang, Xiaoxue Zhao, and Xiaoling Wang. 2015. A bayesian hierarchical model for comparing average f1 scores. In *2015 IEEE International Conference on Data Mining*, pages 589–598.