Transfer Learning using Naïve Bayes in Text Classification

CSDS 440 Fall 2020 Individual Research Report

Randolph Zhao
Case Western Reserve University

December 4, 2020

1. Introduction

Naïve Bayes classifier, a widely used text classification algorithm, uses prior knowledge, the likelihood and prior probabilities, to calculate posterior probabilities. Intuitively, Naïve Bayes classifier can be applied in transfer learning, where the algorithm transfers prior knowledge to improve the performance or stability of classifying the target data. However, since Naïve Bayes does not have a deep knowledge structure for complex text knowledge, it is difficult to transfer Naïve Bayes models directly to other datasets. Although Wenyuan (2007) promotes a Naïve Bayes transfer learning algorithm based on the data distribution relevance that can directly transfer to a new dataset with unlabeled data, in this paper, we assume there is a target training dataset besides the pre-training dataset to help us transfer and fine-tune the pre-trained model.

In this research, the Naïve Bayes classifier is applied on transfer learning in text classification with Laplacian Smoothing and Chi-square. Since the likelihood probabilities in Naïve Bayes are calculated based on the data distribution, we believe merging two Naïve Bayes models is equivalently fine-tuning the probabilities of each word based on the target data distribution. In short, the Naïve Bayes transfer learning algorithm (or $NBTL$ for short) fine-tunes likelihood probabilities in the pre-trained model based on target data distribution.

We test the $NBTL$ on three different datasets and evaluate the performance difference under normal situations, small training size, and skewed training set. The experiment results show $NBTL$ has an excellent performance in the normal and skewed situation and has a relatively poor performance in the situation with small training size. We further explore the influence of feature selection on this algorithm and find using Chi-square statistics to select features to remove relevant words from the pre-trained model brings better performance. Based on this observation, we make a hypothesis about the logic of what Naïve Bayes transfer and how Naïve Bayes transfer learning algorithm works. We define that 'indicating' words are the words showing greater tendency to one class label than the other, while 'non-indicating' words show similar tendency to each class label. We believe that the pre-trained Naïve Bayes model benefits transfer learning by providing a large dictionary with 'non-indicating' words, while target trained model benefits by providing 'indicating' words. We conduct a revised algorithm and the experiment results on three datasets prove this hypothesis of the understandings of the Naïve Bayes transfer learning algorithm.

2. Related Work

a. *Text Classification with Naïve Bayes Classifiers*

The Naïve Bayes classifier [1] in practice has been proven to have good performance for text classification problems with a relatively much higher training speed. Regarding a binary text classification problem, each document $d \in D$, where D denotes the training set, has a bag of words $w \in W$, where W denotes the overall set for every distinct word in the training set. Each document d has a class label $c \in C$, where $C = [0, 1]$ denotes the class label set (negative or positive). The Naïve Bayes classifier compares the posterior, conditional probability $P(c = 1|d)$ and $P(c = 0|d)$, which denotes the probability that document d belongs to class c. The Naïve Baye classifier assumes that each word is mutually conditionally independent given the class. Thus, by applying the Bayes Rule in text classification learning problem, we have

$$P(c|d) \propto P(c)P(d|c) = P(c)\prod_{w \in d} P(w|c), \qquad (1)$$

where $P(d|c)$ is called likelihood, $P(c)$ is called prior, and $P(c|d)$ is called posterior.

In text classification, a popular method is the "bag of words," which will develop a word dictionary and record word frequency. Then, we can use word frequency to calculate the conditional probabilities for the Naïve Bayes model:

$$P(w|c) = \frac{n(w,c)}{n(c)} \qquad (2)$$

where $n(w,c)$ denotes the number of word w with the document class label c and $n(c)$ denotes the number of all words in the document class label c. There are a few popular extensions on Naïve Bayes for text classification. Laplacian smoothing is a good way to smooth the probabilities, where

$$P(w|c) = \frac{n(w,c)+1}{n(c)+|W|} \qquad (3)$$

where $|W|$ is the size of the overall word dictionary. Some popular text preprocessing such as lowercasing and stop words can help to improve the performance of Naïve Bayes. Feature selection in text classification has been recently proven to result in an excellent improvement on the Naïve Bayes model [2].

b. Feature Selection

Feature selection, a widely adopted approach for dimensionality reduction in text classification, has been proven to be a useful method to remove redundant or noisy features without compromising the performance and even results in improvements in performance [3]. As Bo Tang concludes in [4], there are many popular feature selection methods. *TF-IDF* measure uses the word frequency and inverse document frequency to calculate the importance of feature [5]. *Mutual information* (MI) calculates the mutual dependency of two variables. *Information gain* (IG) [3] measures the information gain from removing the word from documents. *Chi-square statistic* [3] calculates the relevance of the word and a specific class label. For a word w, we calculate the $Chi2(w,c)$, which defined as

$$Chi2(w,c) = \frac{[n(w_1,c_1)+n(w_0,c_0)+n(w_0,c_1)+n(w_0,c_1)] \times [n(w_1,c_1)n(w_0,c_0)-n(w_0,c_1)n(w_0,c_1)]^2}{(n(w_1,c_1)+n(w_0,c_1)) \times (n(w_1,c_1)+n(w_1,c_0)) \times (n(w_1,c_0)+n(w_0,c_0)) \times (n(w_0,c_1)+n(w_0,c_0))} \qquad (4)$$

where $n(w_1,c_1)$ denotes the number of word w that exists in documents with class label = 1; $n(w_0,c_1)$ denotes the number of documents with class label = 1 that word w does not exist; $n(w_1,c_0)$ denotes the number of word w that exists in documents with class label = 0; $n(w_0,c_0)$ denotes the number of documents with class label = 0 that word w does not exist.

c. Transfer Learning with Naïve Bayes

Although most popular transfer learning uses the neural network model, there are still few transfer learning algorithms based on the Naïve Bayes model. Wenyuan (2007) promotes a Naïve Bayes Transfer Classification algorithm (NBTC) in [6], which focuses on transferring to different data distribution by using EM-based Naïve Bayes algorithm [7] and iteratively calculating the distribution Kullback-Leibler (KL) divergence [8].

Since Kullback-Leibler divergence calculates the difference of two probabilities distribution, equivalently it calculates the difference of data distribution and implements E-M steps to modify the Naïve Bayes transferred model to adapt the new distribution.

d. Stability

One benefit of transfer learning is stability. Dan Hendrycks mentioned in [9] that pre-training can improve model robustness and uncertainty. If the target dataset is too small, any algorithm cannot develop a model with comprehensive and deep understandings of the relationship between data and class labels, while the pre-trained model provides a basic structure of the complex relationships. If the target dataset is too skewed, any algorithm will have worse performance due to the unbalanced datasets, but the pre-trained model may provide a more balanced relationship. One popular transfer learning application is to use *ImageNet* as the pre-trained graph classifier and transfer to new graph datasets. Simon found the performance of using ImageNet is better than without using ImageNet, but the improvement will decrease by the increase in the size of the target training set [10]. That is, when the target training set is relatively small, transfer learning will bring large improvements to the performance.

3. Algorithms

a. Naïve Bayes Transfer Learning (NBTL) Algorithm

Fine-tuning on the pre-trained model actually retrains a part of weights to adapt to target data distribution. In Naïve Bayes, fine-tuning is adjusting likelihood probabilities to the target data distribution. From Equation (3), the likelihood of the Naïve Bayes classifier is derived from calculating the distribution of words, i.e. word frequency proportion in either positive or negative documents. Thus, the Naïve Bayes model itself carries enough information for data distribution in likelihood probabilities and can be used directly for transfer learning. Thus, modifying the pre-trained model to target data distribution is the same as modifying the pre-trained model to the target trained model, which means we can use algorithms to merge these two Naïve Bayes models for fine-tuning. Based on this observation, we design an algorithm called Naïve Bayes Transfer Learning (NBTL) algorithm. The NBTL algorithm transfers the pre-trained Naïve Bayes model and modifies some likelihood probabilities based on the target small Naïve Bayes model to retrain the model to adapt to the new data distribution.

First, we build a Naïve Bayes model based on prior training datasets, called the *pre-trained model*, and a Naïve Bayes model based on the target training datasets called the *target small model* or *target trained model*. After we have the pre-trained Naïve Bayes model, we want to fine-tune some nodes to make the model adaptable to the new data distribution. Then, we retrain the pre-trained model to increase the similarity with the target small model to adapt to the target dataset distribution. Alternatively speaking, we are trying to merge two Naïve Bayes models to have a more comprehensive model with prior knowledge from the pre-trained model that is adapted enough for the target dataset distribution. Thus, the NBTL algorithm can be treated as a variant of ensemble algorithms and of fine-tuning algorithms.

A description of our Naïve Bayes transfer learning algorithm is shown in Algorithm 1. We first calculate the pre-trained model M1 in the transferred dataset and the target small model M2. For each word in M1, if M2 also has common words, calculate the Chi-square value of $P(w|c=1), P(w|c=0)$, where

$$Chi2_{+} = \frac{[P_{M1}(w|c=1) - P_{M2}(w|c=1)]^2}{P_{M2}(w|c=1)} \qquad (5)$$

$$Chi2_{-} = \frac{[P_{M1}(w|c=0) - P_{M2}(w|c=0)]^2}{P_{M2}(w|c=0)} \qquad (6)$$

.

We control retrain speed by learning rate based on these two Chi-square values. When the similarity improvement of this word is low, we stop fine-tuning for this word. If M2 does not have the same word, we decrease the importance of that word by multiplying $\varepsilon = 0.9$. Then, after fine-tuning for each word, we merge the pre-trained dictionary and target trained dictionary with corresponding probabilities.

b. Chi-Squared Feature Selection

In this paper, we are using the *Chi-square statistic* as the feature selection method to explore the influence of feature selection on the NBTL algorithm. Chi-square statistic selects features based on the Chi-square α value. Based on Algorithm 1., we implement the Chi-Squared feature selection on the pre-trained model and compare the influence on performance. We implement the Chi-Square feature selection of comparing either relevant features with Chi-Square $\alpha \leq 10.83$ or irrelevant features with Chi-Square $\alpha > 10.83$, which is the critical value for 0.999 confidence for a 1 degree of freedom Chi-Squared distribution [11].

```
Algorithm 1. The Naive Bayes Transfer Learning (NBTL) Algorithm.
Input: A labeled transferred training dataset, a labeled target training dataset, an unlabeled target testing dataset.
Output: Accuracy measurement on the testing dataset
  1. Initialize Naïve Bayes models M1 for pre-trained dataset and M2 for target dataset
  2. for $w$ in M1:
       if $w$ in M2:
         while Chi2 change $\geq 1 \times 10^{-7}$:
           Calculate $Chi2_+, Chi2_-$ by Equation (5) & (6)
           Update learning rates
           for each class label:
             Retrain the $P_{M1}(w|c = 1)$ to become close to $P_{M2}(w|c = 1)$ by each learning rate
       else:
         Multiply the $P_{M1}(w|Y = 1), P_{M1}(w|Y = 0)$ by constant $\varepsilon = 0.9$
  3. for $w$ in M2 and $w$ not in M1:
       Merge $[P_{M2}(w|Y = 1), P_{M2}(w|Y = 0)]$ to M1
  4. Calculate the accuracy performance on the testing dataset.
```

Algorithm 1. The Naïve Bayes Transfer Learning (NBTL) algorithm

```
Algorithm 2. Naïve Bayes Chi-Squared Feature Selection (Removing irrelevant).
Input: A pre-trained Naïve Bayes model M
Output: Feature selected Naïve Bayes model M
  1. for w in M:
       Calculate Chi-square score by Equation (4)
       if Chi-square score $\leq$ 10.83:
         Remove w from M
```

```
Algorithm 2. Naïve Bayes Chi-Squared Feature Selection (Removing relevant).
Input: A pre-trained Naïve Bayes model M
Output: Feature selected Naïve Bayes model M
  1. for w in M:
       Calculate Chi-square score by Equation (4)
       if Chi-square score > 10.83:
         Remove w from M
```

Algorithm 2. The Naïve Bayes Transfer Learning (NBTL) algorithm with feature selection

4. Experiments

a. *Algorithm Error Rate*

In order to evaluate the algorithm and improvements, we conducted experiments on three data sets, 20 Newsgroups[1] [12], Amazon Product Review data[2] [13], and Spam Emails[3]. Among these three data sets, 20 Newsgroups and Amazon Product Review data are not originally designed for transfer learning. Thus, we manually split the dataset for transfer learning.

In the 20 Newsgroups datasets, there are 7 top categories and 20 subcategories. We define the tasks as subcategories classification. Consider an origin dataset that distribution uniformly in two categories A_1, A_2 as our pre-training dataset, and a target dataset that distribution uniformly in the other two categories B_1, B_2. Then, we can compare transfer learning performance with the same top categories or with different top categories, which denotes to transfer learning with similar distribution or with different distribution.

In Amazon Product Review datasets, there are 24 review datasets for different categories. We select seven datasets as our experimental datasets: *Cell Phones and Accessories*, *Electronics*, *Industrial and Scientific*, *Patio Lawn and Garden*, *Software*, *Toys and Games*, and *Video Games*. We will compare transfer learning performance from one category review to another category review.

[1] http://people.csail.mit.edu/jrennie/20Newsgroups/

[2] Amazon review data (ucsd.edu)

[3] ECML/PKDD Discovery Challenge 2006 (ecmlpkdd2006.org)

In Spam Email datasets, since the datasets are intuitively for transfer learning contest, we will evaluate the transfer learning performance from the prior datasets *task_a_labeled_train.tf* transferring to target datasets *task_a_labeled_tune.tf*.

Since the NBTL algorithm is fine-tuning by each word probabilities, we analyze and plot out the error rate decrease by fine-tuning on each word in the dictionary. We separately research transfer on datasets with similar distributions or on different distributions.

In the 20 Newsgroups datasets, we use 1200 prior training data, 800 target training data, and 400 target testing data. Each training and testing is manually filtered to be balanced, i.e. positive/negative example rate=1/1.

In Figure 1 and Figure 2, different lines represent different pairs of transferred and target datasets. For example, 'comp vs comp' represents a transfer NB model from two computer subcategory datasets to the other two computer subcategory datasets. We define that the word distribution is similar in the same top categories, as Figure 1 shows, while the word distribution is more different in different top categories, as Figure 2 shows.

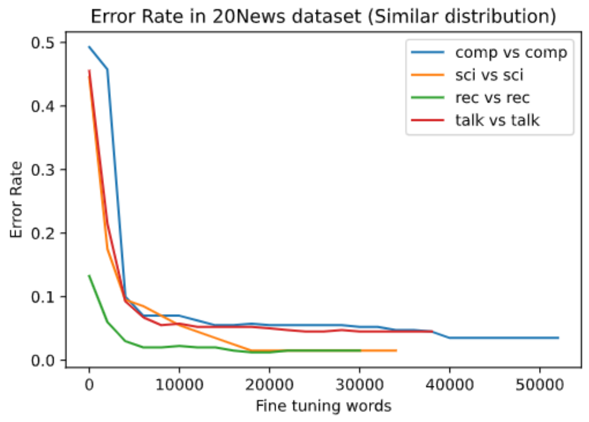

Figure 1: The Error Rate Curves by Fine-tuning on datasets with similar distribution in 20 Newsgroups (the size of target training set = 800)

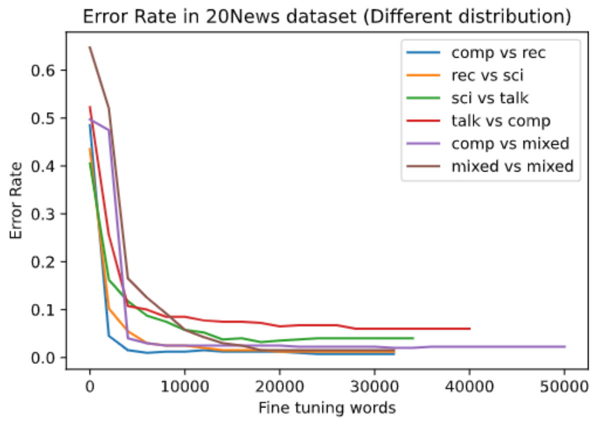

Figure 2: The Error Rate Curves by Fine-tuning on datasets with different distribution in 20 Newsgroups (the size of target training set = 800)

We then test the NBTL algorithm on Amazon review datasets [14] by plotting the error rate during fine-tuning each word. Each experiment trail uses a uniformly distributed review dataset in two different categories. The prior training dataset has 10,000 examples. The target training dataset has 5,000 examples and the target testing dataset has 5,000 examples. Since prior training reviews and target reviews are in the different categories, this experiment can be treated as transfer learning to different distributions. Unfortunately, the NBTL does not have an excellent performance on some datasets as Figure 3 shows. We choose five different pairs: pair 1 is *Toys and Games* transferring to *Patio Lawn and Garden*; Pair 2 is *Video Games* transferring to *Toys and Games*; Pair 3 is *Electronics* transferring to *Toys and Games*; Pair 4 is *Software* to *Video Games*; Pair 5 is *Toys and Games* transferring to *Software*.

We further apply the NBTL algorithm on the Spam Emails dataset in Figure 4. We use *task_a_labeled_train.tf* as the prior training dataset, which has 4,000 examples, and *task_a_labeled_tune.tf* as target dataset, where we split 1,000 examples as target training datasets and 1,500 examples as target testing datasets. Each training and testing dataset is uniformly distributed.

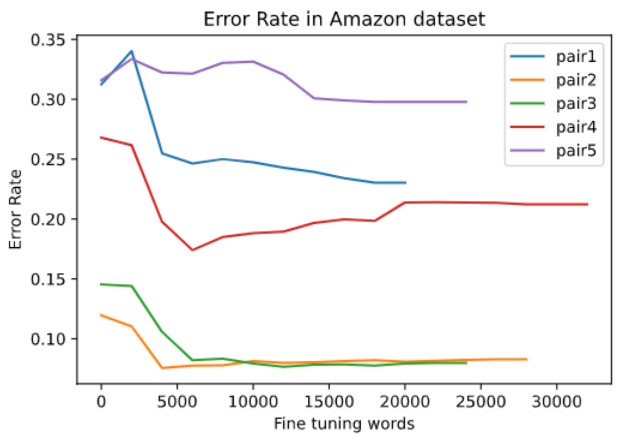

Figure 3: The Error Rate Curves by Fine-tuning on datasets in Amazon Product Review (the size of target training set = 5000)

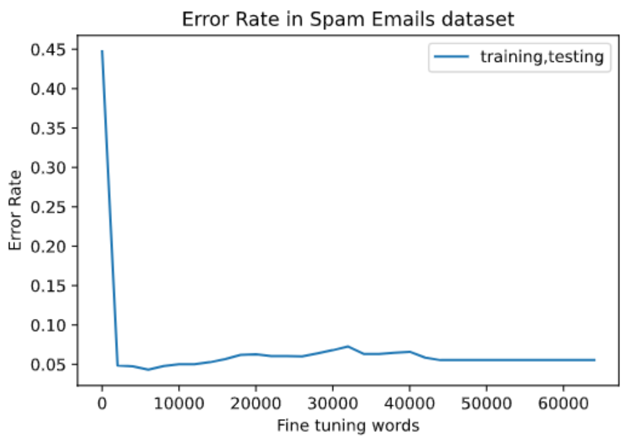

Figure 4: The Error Rate Curves by Fine-tuning on datasets in Spam Emails Review (the size of target training set = 1000)

	Original - NBTL Accuracy Comparison	Shared: Different vocabulary
Amazon Pair 1	0.6876 - **0.7696**	6559:12532
Amazon Pair 2	0.8804 - **0.9172**	8159:18769
Amazon Pair 3	0.8546 - **0.9202**	7687:15149
Amazon Pair 4	0.732 - **0.7878**	9743:21344
Amazon Pair 5	0.6842 - **0.7022**	8188:15041
Spam	0.9433 – **0.9447**	8073:5503

Table 1: Accuracy Comparison between with and without using NBTL and Vocabulary Difference

b. Feature Selections Comparison

We use Algorithm 2 to implement *Chi-square* feature selection on the pre-trained model and implement Algorithm 1 to transfer models. We will compare the performance of whether to use feature selection to select either relevant or irrelevant on the pre-trained model and also compare whether the performance keeps the same when the target training set is small. The blue line is transfer learning without feature selection. The orange line is using feature selection to remove irrelevant words, while the green line is using feature selection to remove relevant words.

In Figure 5 and Figure 6, the X-axis represents different pairs of transferred and target datasets in 20 Newsgroups. For example, 'sci,sci' represents a transfer NB model from two science subcategory datasets to the other two science subcategory datasets. The datasets are uniformly distributed and the size of target training set is 800 (Figure 5) or 50 (Figure 6).

We then compare the influence of feature selection on experiments of Amazon Product Review datasets in Figure 7 with Figure 3. The datasets are uniformly distributed and the size of the target training set is 5000 (Figure 7). Since the performance results are similar, we provide an accuracy table with shared/different vocabulary in Table 2, which can be used to compare with Table 1.

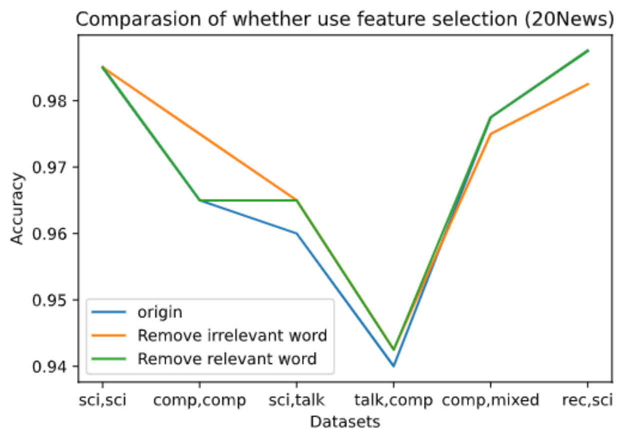

Figure 5: Accuracy Comparison on whether to implement Chi-square feature selection on the pre-trained models in 20 Newsgroups (the size of target training set = 800).

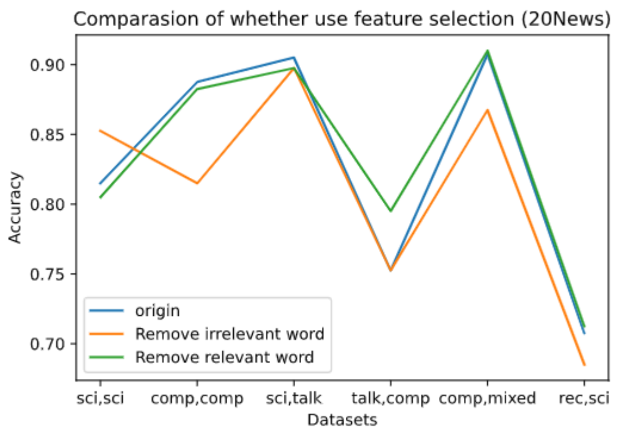

Figure 6: Accuracy Comparison on whether to implement Chi-square feature selection on the pre-trained model in 20 Newsgroups (the size of target training set = 50).

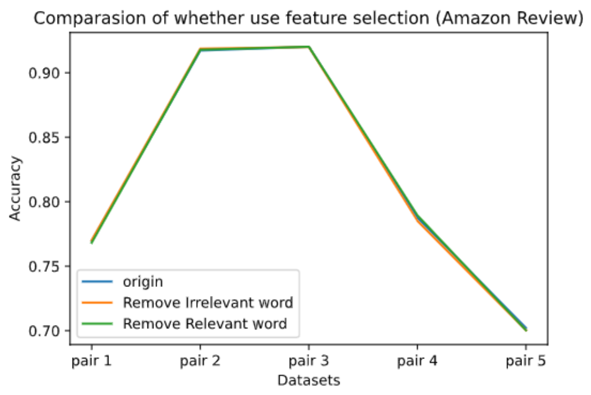

Figure 7: Accuracy Comparison on whether to implement Chi-square feature selection on pre-trained models in Amazon dataset.

	NTBL Accuracy	Feature Selection (Remove irrelevant)	Feature Selection (Remove relevant)	Shared : Different vocabulary (Remove irrelevant)	Shared : Different vocabulary (Remove relevant)
Pair 1	0.7696	**0.7702**	0.7682	1025:186	5534:8508
Pair 2	0.9172	**0.9188**	0.9178	751:187	7408:15036
Pair 3	**0.9202**	0.92	**0.9202**	859:228	6828:10903
Pair 4	0.7878	0.7852	**0.7896**	1174:239	8569:15288
Pair 5	**0.7022**	0.7004	0.7002	1074:137	7114:6928

Table 2: Accuracy Comparison and Vocabulary Difference in Amazon Product Review corresponding to five pair datasets

c. *Stability*

We are also curious about how the NBTL performs differently when the size of the target training set is small or skewed. Thus, we conduct two experiments: one with a small size target training set, the other with a skewed target training set, i.e. the positive/negative rate is far away from 1/1. We configure the experiments as follows: the size of pre-training dataset=1200 and the size of target testing dataset=400. The target training set is compared between 50 (Figure 8,9) and 800 (Figure 1,2). The positive/negative rate is compared between 1:4 (Figure 10,11) and 1:1 (Figure 1,2).

Same to Figure 1 and Figure 2, different lines represent different pairs of transferred and target datasets. Figure 8 represents the error rate changes for transferring to similar distribution with the limited size of the target training set = 50 in 20 Newsgroups. Figure 9 represents the error rate changes for transferring to different distributions with the limited size of the target training set = 50 in 20 Newsgroups. Figure 10 represents the error rate changes for transferring to a similar distribution with the positive/negative rate = 1:4 in 20 Newsgroups. Figure 11 represents the error rate changes for transferring to a different distribution with the positive/negative rate = 1:4 in 20 Newsgroups.

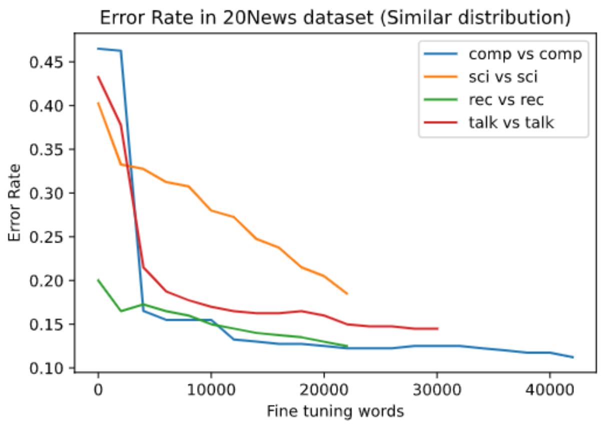

Figure 8: The Error Rate Curves by Fine-tuning on 20 Newsgroups datasets with similar distribution (the size of target training set = 50).

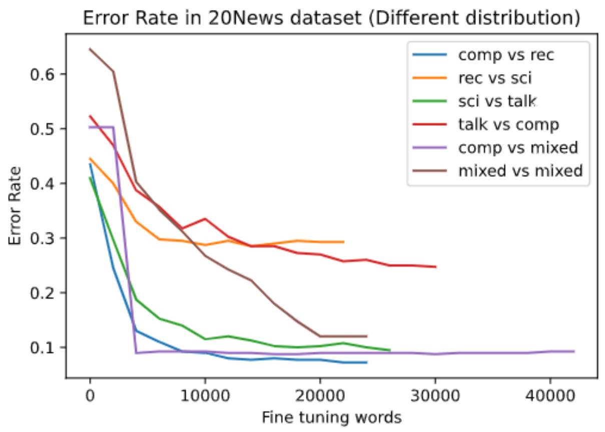

Figure 9: The Error Rate Curves by Fine-tuning on 20 Newsgroups datasets with different distribution (the size of target training set = 50).

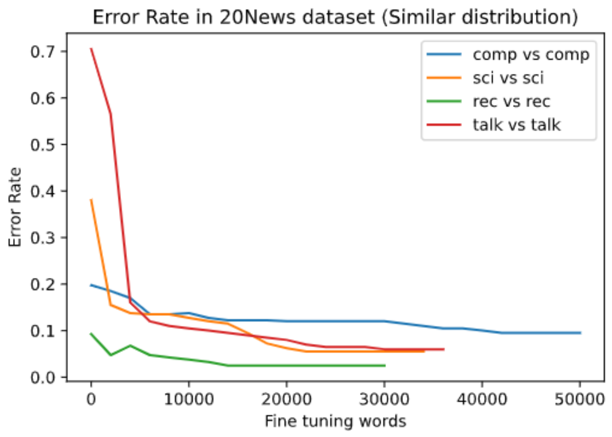

Figure 10: The Error Rate Curves by Fine-tuning on 20 Newsgroups datasets with similar distribution (positive/negative rate = 1:4).

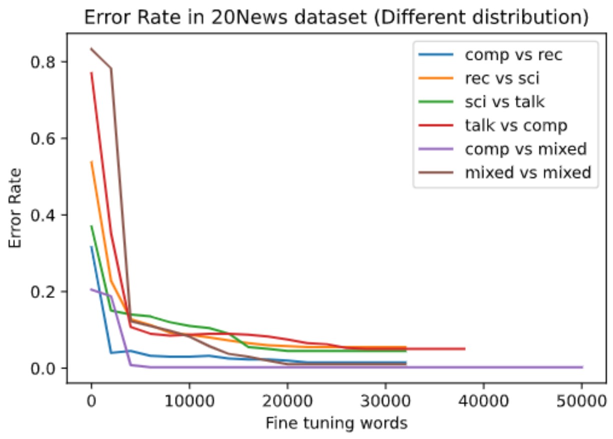

Figure 11: The Error Rate Curves by Fine-tuning on 20 Newsgroups datasets with different distribution (positive/negative rate = 1:4).

We also test the influence of different sizes of the target training set on the Spam Emails dataset. We track the error rate changes when the size of the target training set is 100, 500, and 1000 in Figure 12. Furthermore, we compare the performance of the NBTL on the Spam Emails dataset if the positive/negative rate of the target training dataset is 1:1 (uniformly distributed) and 1:4 (skewed) in Figure 13.

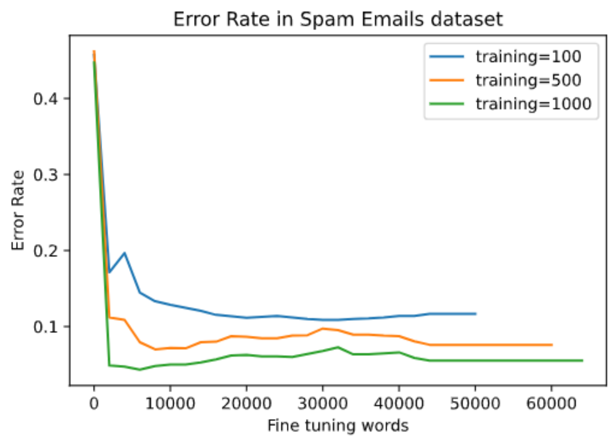

Figure 12: The Error Rate Curves by Fine-tuning on datasets on Spam Email (uniformly distributed).

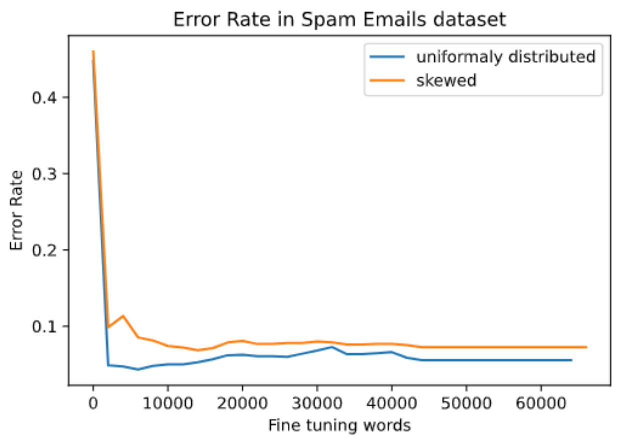

Figure 13: The Error Rate Curves by Fine-tuning on datasets on Spam Email (training size = 1000).

5. Results Analysis

a. Accuracy

As Figure 1, 2, 3, and 4 show, the NBTL has greatly reduced the transfer error rate by fine-tuning the prior knowledge about the relationship between words and class labels. Comparing Figure 1 with Figure 2, the NBTL algorithm does not make a significant difference when transferring to a similar distribution or to a different distribution. This is because Naïve Bayes only focuses on the word and class label relationships. If the word has been seen in the pre-trained model, it will be fine-tuned to adapt to the new distribution. If the word has not been seen in the pre-trained model, the NBTL will directly use the probability in the target small model, which should be originally fit for the new distribution. Thus, the NBTL will be greatly influenced as long as the shared common proportion is high. In addition, Table 1 proves that transferring a pre-trained model with the NBTL can improve performance compared with using the target trained model directly. However, as the NBTL shows poor performance on some datasets in Amazon Product Review datasets in Figure 3. The reason is that the NBTL model cannot get large prior knowledge due to the low shared vocabulary shown in Table 1.

We can find some error rates that increase in the last thousands of words, such as 'comp vs mixed' in Figure 2, pair4 in Figure 3, and Spam Emails in Figure 4. This may be where the overfitting occurs. If the pre-trained model is fine-tuned too much so that it becomes too similar to the target small model, this will cause the loss of most of the transferred knowledge. In other words, we are overfitting the transferred model to the target training datasets, which will cause an increase in error rate. One solution may be to calculate the similarity of two Naïve Bayes models by using Kullback-Leibler (KL) divergence. The algorithm should record the KL changes and stop fine-tune the word if the KL changes is lower than a small value δ.

b. Feature Selection

Figure 5 shows that both feature selections to remove relevant words and to remove irrelevant words can increase the transferred model accuracy, but removing relevant words always brings better performance. Figure 6 shows removing relevant words will not sacrifice much performance and gain better stability. Comparing the accuracy and shared/different vocabulary rate in Table 2, both feature selection ways improves the transferring model accuracy, but removing irrelevant words greatly increases the shared/different vocabulary rate, which is desired when transferring. However, the NBTL algorithm requires not only information about important words, but also more shared vocabulary to increase robustness and stability of the target small model. Thus, the results of these two feature selections make sense. We define that 'indicating' words are the words showing greater tendency to one class label than the other, while 'non-indicating' words show similar tendency to each class label. While removing irrelevant words, we keep the most 'indicating' words and the key information the pre-trained model has.

While removing relevant words, we keep the most of the dictionary with 'not-indicating'-labeled words, which helps to increase the dictionary in the target of the small model and prevent mistakenly learning the wrong importance of words due to the relatively small size of target training sets.

c. Stability

A small target training size affects the performance of the NBTL negatively. Comparing Figure 1,2 with Figure 8,9, we can find the starting error is similar, but the speed of decrease is much slower, and the stopping point is much higher than with a large target training set. Thus, fine-tuning also has great benefits but due to lack of training data, the pre-trained model cannot fit much better to the new dataset distribution, which causes higher error rates. Comparing Figure 8 with Figure 9, although all trials show a decrease in error rate, some final error rates are not very desired, such as 'rec vs sci' with 0.3 error rate. In other words, the NBTL algorithm sometimes behaves poorly due to the lack of correct information and misleading distribution caused by small target training sets. This is because the NBTL is dependent on the target training set to fine-tune or adapt the pre-trained model to target dataset distribution. If the size of the target training set is small, there will be skewed or not enough strong information to help the pre-trained model to adapt to the new distribution. Figure 12 explicitly shows that the error rate will decrease by the increase in the target training set.

However, the NBTL algorithm will not be greatly influenced by skewed data distribution as comparing Figure 1,2 with Figure 10,11 and Figure13. The NBTL calculates the probability of class labels based on the Equation (3), which divides by the number of positive or negative examples. Thus, the likelihood probabilities will not be influenced by a skewed dataset. Furthermore, since the datasets provide plenty of negative examples, the NBTL can build up a more comprehensive negative word dictionary, which will help to improve the accuracy. As Figure 13 shows, the NBTL shows similar performance under the uniformly distributed and skewed distributed datasets.

6. Further Improvements

Although it is surprising that implementing feature selection to remove relevant words brings better results, this observation shows the basic logic behind the Naive Bayes transfer learning algorithm. Pre-trained model provides a large dictionary with likelihood probabilities and learns the words that are less helpful for classification. At the same time, the target trained model provides new words with likelihood probabilities and information about the new data distributions. Fine-tuning retrains mostly the 'indicating' words in the pre-trained model to adapt to new data distribution. In other words, the 'indicating' words are provided by the target trained model (main help to classify) and the 'non-indicating' words are provided by pre-trained words (main help to keep stable). Thus, the NBTL algorithm will behave with better stability and performance after transfer learning.

From this logic, we conduct Algorithm 3 to examine this hypothesis. As Algorithm 1 fine-tunes all words, Algorithm 3 defines that we are only fine-tuning the most 'indicating' words by target trained model. We sort the dictionary of the pre-trained model by the $P(w|c=1)/(P(w|c=1)+P(w|c=0))$ and then pick the first and last part of words as 'indicating words'. For example, we could choose the first and last 1/k of the whole dictionary as 'indicating words' because those words show great indicators to the class label. In this research, we pick the value for k empirically as 5 or 10. The algorithm 3 assumes that the 'non-indicating' words in the pre-trained model benefits mostly to the performance of the NBTL, and thus we keep those words unchanged. Then, we fine-tune only the 'indicating' words in the pre-trained model to the target data distribution. If the hypothesis is correct, we should see the same or better performance from Algorithm 1 to Algorithm 3. Another benefit will be reduced overfitting. Overfitting occurs in NBTL because we fine-tune many words too close to the target data distribution. However, by only fine-tuning the most indicating words, overfitting should be reduced.

Algorithm 3. The Revised Naive Bayes Transfer Learning (NBTL v.2) Algorithm.

```
Input: A labeled transferred training dataset, a labeled target training dataset, an unlabeled target testing dataset.
Output: Accuracy measurement on the testing dataset
  1. Initialize Naïve Bayes models M1 for pre-trained dataset and M2 for target dataset
  2. Sorted_words = Sort the words by $P_{M1}(w|Y = 1)/(P_{M1}(w|Y = 1) + P_{M1}(w|Y = 0))$
  3. Indicating_word_list = first 1/k of Sorted_words + last 1/k of Sorted_words ($k \in [2,10]$)
  4. for $w$ in M1:
       if $w$ in Indicating_word_list:
         if $w$ in M2:
           while Chi2 change $\geq 1 \times 10^{-7}$:
             Calculate $Chi2_+, Chi2_-$ by Equation (5) & (6)
             Update learning rates
             for each class label:
               Retrain the $P_{M1}(w|c = 1)$ to become close to $P_{M2}(w|c = 1)$ by each learning rate
         else:
           Multiply the $P_{M1}(w|Y = 1)$, $P_{M1}(w|Y = 0)$ by constant $\varepsilon = 0.9$
  5. for $w$ in M2 and $w$ not in M1:
       Merge $[P_{M2}(w|Y = 1), P_{M2}(w|Y = 0)]$ to M1
  6. Calculate the accuracy performance on the testing dataset.
```

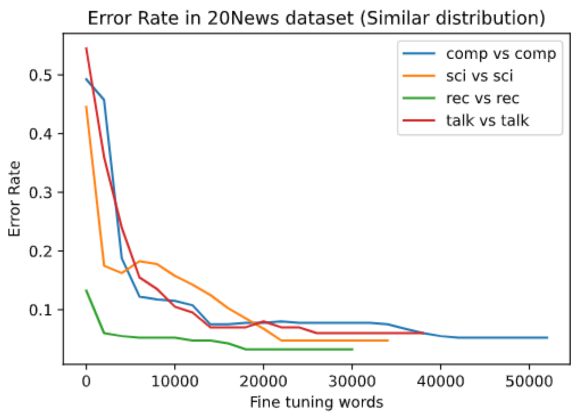

Figure 14: The Error Rate Curves by Algorithm 3 on datasets on 20 Newsgroups with similar distribution (k=5)

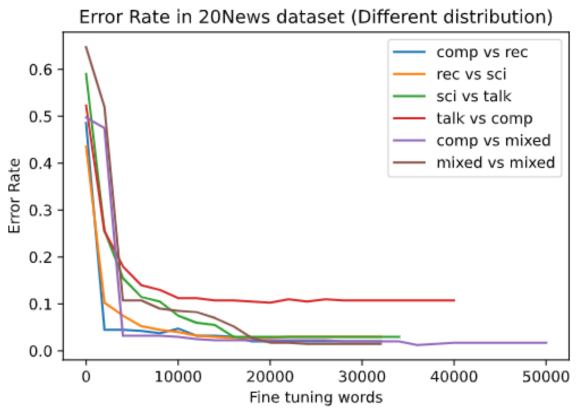

Figure 15: The Error Rate Curves by Algorithm 3 on datasets on 20 Newsgroups with different distribution (k=5)

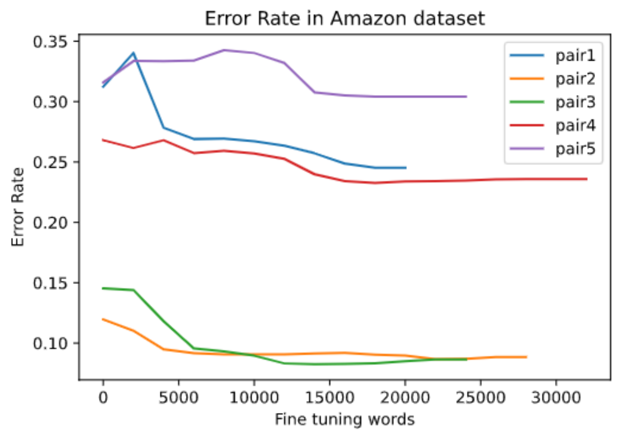

Figure 16: The Error Rate Curves by Algorithm 3 on datasets on Amazon Reviews. (k=5)

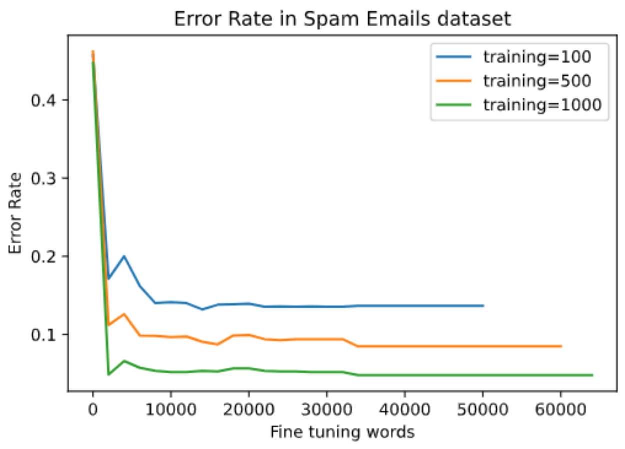

Figure 17: The Error Rate Curves by Algorithm 3 on datasets on Spam Email (k=10)

The experiment results in Figure 14,15 displays that the performance of the revised NTBL algorithm does not decrease much even if we only fine-tune 40% of the words. Some narrow overfitting happened in Figure 1 and 2 such as 'comp vs mixed' does not overfitting anymore. Especially in Figure 16, pair4 shows no overfitting compared to large overfitting performance in Figure 3. Nevertheless, the revised algorithm in Figure 17 shows no overfitting in three different training sizes compared to Figure 12 without hurting the performance. Therefore, the revised NTBL algorithm shows better robustness and stability and prevents overfitting without hurting much performance. Thus, the success of the revised NBTL proved our hypothesis about the logic behind the Naive Bayes transfer learning algorithm.

7. Conclusion

In this paper, we study transfer learning with Naïve Bayes classifiers on text classification. We conducted a Naïve Bayes transfer learning (NBTL) algorithm based on the understanding "the Naïve Bayes carries the information about data distribution, thus we just need to merge the pre-trained NB model with target train model to adapt to the target data distribution." We use *Chi-square statistics* to control the learning rate of the converges of fine-tuning. Experiments on the NBTL error rates prove that the NBTL algorithm has good performance on all datasets, including datasets with both similar distribution and different distribution, and has an excellent convergence.

We further study the influence of Chi-square feature selection on the NBTL algorithm. *Chi-square* feature selection proves to help to improve performance in most text classification problems in both removing relevant or irrelevant words. And removing irrelevant words shows better stability and performance. We define that 'indicating' words are the words showing greater tendency to one class label than the other, while 'non-indicating' words show similar tendency to each class label. We think that a pre-trained model provides 'non-indicating' words knowledge while target trained small model provides 'indicating' words knowledge while fine-tuning retrains the 'indicating' words from the pre-trained model to adapt to the target data distribution.

We then analyze the performance difference among normal datasets, small target training sets, and skewed target training sets (unbalanced positive and negative data distribution). Since the NBTL algorithm is dependent on the fine-tuning of the target training set, if the target training set does not have enough examples to form a correct word distribution and word importance, it is difficult for NBTL to result in a lower error rate, even if NBTL still reduces the error rate significantly. By contrast, the NBTL algorithm shows good resistance to skewed data distribution. Experiments show that even with a positive/negative rate = 1:4, the NBTL algorithm still gets a close or even better performance compared to the uniform data distribution. This is because NBTL only focuses on the probabilities in either positive or negative documents. Thus, with less positive documents, NBTL develops a more accurate knowledge about the words with a more comprehensive negative word dictionary that are critical to negative documents, which results in higher performance.

Based on the understanding from feature selection experiments, we design a revised Naïve Bayes transfer learning algorithm ($NBTL\ v.2$) where we limit the algorithm to only fine-tune the most 'indicating' words in the pre-trained model. We predict it will have the same or better performance and will prevent overfitting because it limits fine-tuning and prevents from becoming too similar. We tested the $NBTL\ v.2$ algorithm on three datasets and we found it keeps the same performance but prevents overfitting excellently. The success of this algorithm further proved our understanding of the logic about how `Bayes transfer learning algorithm works.

8. Future Work

Some potential extensions and improvements can be applied to our work. First, as we observe the error rate line, the $NBTL$ algorithm occurs overfitting for some datasets. Apart from using $NBTL\ v.2$, we suggest using Kullback-Leibler divergence to calculate the similarity of two models to prevent fine-tuning too much. Second, as ImageNet is commonly transferred to improve the performance of image classification, we suggest building a comprehensive $NBTL$ model that includes most words and corresponding likelihood probabilities. As Simon Kornblith stated in [10], a comprehensive pre-trained model should improve the performance significantly on the small target training set and this benefit will be decreased by increasing the size of the target training set. As the Naive Bayes will benefit hugely from a comprehensive dictionary with likelihood probabilities, it is reasonable to analyze how a comprehensive $NBTL$ behaves on an original, small, or skewed target training dataset.

Furthermore, $NBTL\ v.2$ still shows a relatively poor performance on some datasets in Amazon Reviews in Figure 16. We could further research on some different fine-tuning algorithms on either the indicating words or non-indicating words. In the $NBTL\ v.2$, the proportion k of ‘indicating’ words is selected empirically. In the future, we could develop a theoretical proof for picking the best k values for control overfitting and increase performance. From the observations during the experiments, we suggest that there could be some relationships between shared/different vocabulary rate and k values.

Reference

[1] Lewis, D. D. 1992. Representation and learning in information retrieval. Ph.D. Dissertation, Amherst, MA, USA.
[2] P. Chandrasekar and K. Qian, "The Impact of Data Preprocessing on the Performance of a Naive Bayes Classifier," 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), Atlanta, GA, 2016, pp. 618-619, doi: 10.1109/COMPSAC.2016.205.
[3] Y. Yang and J. O. Pedersen, “A comparative study on feature selection in text categorization,” in Proc. Int. Conf. Mach. Learn., vol. 97, 1997, pp. 412–420.
[4] B. Tang, S. Kay and H. He, "Toward Optimal Feature Selection in Naive Bayes for Text Categorization," in IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 9, pp. 2508-2521, 1 Sept. 2016, doi: 10.1109/TKDE.2016.2563436.
[5] G. Salton, A. Wong, and C.-S. Yang, “A vector space model for automatic indexing,” Commun ACM, vol. 18, no. 11, pp. 613–620, 1975.
[6] Dai, Wenyuan & Xue, Gui-Rong & Yang, Qiang & Yu, Yong. (2007). Transferring Naive Bayes Classifiers for Text Classification. 540-545. AAAI.
[7] Nigam, K.; McCallum, A. K.; Thrun, S.; and Mitchell, T. 2000. Text classification from labeled and unlabeled documents using em. Machine Learning 39(2-3):103–134.
[8] S. Kullback and R. A. Leibler, “On Information and Sufficiency,” The Annals of Mathematical Statistics, vol. 22, no. 1, 1951, pp. 79–86. JSTOR, www.jstor.org/stable/2236703. Accessed 30 Nov. 2020.
[9] Dan Hendrycks, Kimin Lee, Mantas Mazeika, "Using Pre-Training Can Improve Model Robustness and Uncertainty," 2019, ICML.
[10] S. Kornblith, J. Shlens and Q. V. Le, "Do Better ImageNet Models Transfer Better?," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 2656-2666, doi: 10.1109/CVPR.2019.00277.
[11] Nurhayati, A. E. Putra, L. K. Wardhani and Busman, "Chi-Square Feature Selection Effect On Naive Bayes Classifier Algorithm Performance For Sentiment Analysis Document," 2019 7th International Conference on Cyber and IT Service Management (CITSM), Jakarta, Indonesia, 2019, pp. 1-7, doi: 10.1109/CITSM47753.2019.8965332.
[12] Lang, Ken. (2000). NewsWeeder: Learning to Filter Netnews (To appear in ML 95).
[13] Ni, Jianmo & Li, Jiacheng & McAuley, Julian. (2019). Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. 188-197. 10.18653/v1/D19-1018.