electronics

MDPI

Article

A Curriculum Learning Approach for Multi-Domain Text Classification Using Keyword Weight Ranking

Zilin Yuan [1,†], Yinghui Li [1,†], Yangning Li [1,†], Hai-Tao Zheng [1,2,*,†], Yaobin He [3,4,*], Wenqiang Liu [5], Dongxiao Huang [5] and Bei Wu [5]

[1] Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China; yuanzl21@mails.tsinghua.edu.cn (Z.Y.); liyinghu20@mails.tsinghua.edu.cn (Y.L.)
[2] Pengcheng Laboratory, Shenzhen 518055, China
[3] The Smart City Research Institute of CETC, Shenzhen 518055, China
[4] National Center for Applied Mathematics Shenzhen, Shenzhen 518055, China
[5] Interactive Entertainment Group, Tencent Inc., Shenzhen 518055, China; bellabwu@tencent.com (B.W.)
* Correspondence: zheng.haitao@sz.tsinghua.edu.cn (H.-T.Z.); heyaobin@cetc.com.cn (Y.H.); Tel.: +86-180-3815-3089 (H.-T.Z.)
† These authors contributed equally to this work.

Abstract: Text classification is a well-established task in NLP, but it has two major limitations. Firstly, text classification is heavily reliant on domain-specific knowledge, meaning that a classifier that is trained on a given corpus may not perform well when presented with text from another domain. Secondly, text classification models require substantial amounts of annotated data for training, and in certain domains, there may be an insufficient quantity of labeled data available. Consequently, it is essential to explore methods for efficiently utilizing text data from various domains to improve the performance of models across a range of domains. One approach for achieving this is through the use of multi-domain text classification models that leverage adversarial training to extract domain-shared features among all domains as well as the specific features of each domain. After observing the varying distinctness of domain-specific features, our paper introduces a curriculum learning approach using a ranking system based on keyword weight to enhance the effectiveness of multi-domain text classification models. The experimental data from Amazon reviews and FDU-MTL datasets show that our method significantly improves the efficacy of multi-domain text classification models adopting adversarial learning and reaching state-of-the-art outcomes on these two datasets.

Keywords: multi-domain text classification; curriculum learning; keyword weight ranking

Citation: Yuan, Z.; Li, Y.; Li, Y.; Zheng, H.-T.; He, Y.; Liu, W.; Huang, D.; Wu, B. A Curriculum Learning Approach for Multi-Domain Text Classification Using Keyword Weight Ranking. *Electronics* **2023**, *12*, 3040. https://doi.org/10.3390/electronics12143040

Academic Editor: Valentina E. Balas

Received: 31 May 2023
Revised: 30 June 2023
Accepted: 3 July 2023
Published: 11 July 2023

1. Introduction

Text classification is a fundamental NLP task that is widely leveraged across various domains, including but not limited to spam detection [1], news categorization [2], and e-commerce product evaluation [3]. Research on text classification methods dates back to the early days of computing in the 1950s with the advent of rule-based approaches. In the 1990s, methods combining machine learning techniques such as feature engineering and classifiers became prevalent [4]; presently, however, CNN [5], RNN [6,7], attention mechanism [8], and other deep learning techniques are more popular for text classification tasks.

Regardless of the chosen method, there are two primary issues that need to be considered: domain dependence and the requirement for a sizeable annotated corpus. Domain dependence pertains to the fact that a classifier trained on a specific domain may not perform as well in other domains due to differences in the meanings of vocabulary across domains, such that the same word may convey divergent meanings in different domains. As depicted in Figure 1 and noted in the work of Cai et al. [9], the word "infantile" frequently connotes a negative sentiment in the domain of movie reviews (for instance, "The idea of the movie is infantile"), whereas in evaluations of infant products (such as "The

infantile toy was sold out yesterday"), there is typically no overt emotional valence. In order to train classifiers for different domains of text, it is crucial to have a sufficient amount of labeled data for each domain. Unfortunately, not all domains have an adequate amount of domain-specific text to train on. Therefore, we must leverage text corpora from diverse domains to effectively classify text from a specific domain. This approach is known as Multi-Domain Text Classification (MDTC) and has been explored in previous research [10,11]. However, conventional MDTC methods [11,12] tend to overlook an important piece of information: different domains present varying levels of difficulty in classification.

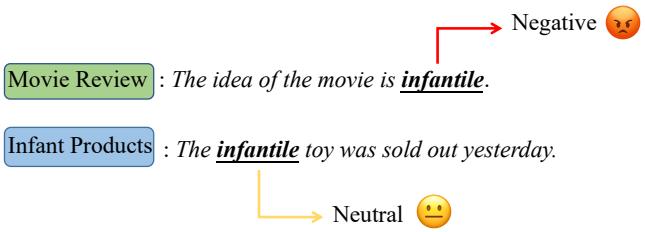

Figure 1. The different sentiments of "infantile" in different domains.

Different classification difficulties could stem from differences in feature extraction complexity, differences in data distribution, variations in emotional tones conveyed through certain vocabulary choices, differences in sentence lengths, and variety of vocabulary. Failing to account for these differences between domains and arbitrarily training the prediction models for each domain can lead to suboptimal performance when encountering difficult text classification tasks. Essentially, exposing the model to very complex domain texts at an earlier stage of training may impair its ability to learn explicit features of these texts well, resulting in less accurate predictions for challenging text domains.

The task of text classification in different domains varies in difficulty, and thus, this variability can be leveraged to train the model from easy to difficult data. This learning strategy, called curriculum learning [13], mirrors the way humans learn, where simple concepts are first internalized before moving on to more complex ones. Curriculum learning has demonstrated significant improvements in various natural language processing tasks, including dialog state tracking [14], few-shot text classification [15], Chinese spell checking [16], among others. The essence of curriculum learning lies in the measurement of data difficulty and scheduling.

Inspired by curriculum learning, we propose a mechanism for measuring domain text difficulty based on keyword weight ranking. The keywords of texts of a domain, which are the most closely related to the meaning of the text, effectively reflect the specific features of each domain. Extracting and analyzing such keywords allows us to comprehend domain-specific subjects and content, effectively revealing their core concepts. By identifying the keywords in domain texts, we are better able to understand their themes, key content, and underlying concerns, leading to a more comprehensive comprehension of the text. So the greater the weight of keywords in the text of a certain field, the clearer the core concept of this field, the simpler the data distribution, and the easier it is to extract text features.

By incorporating multi-domain text classification to extract both domain-shared and domain-specific features, we suggest utilizing the total weight of domain-specific keywords as a metric for the complexity of domain-adapted feature extraction. We then utilize this metric to optimize the order in which a specific domain corpus is presented to the model during training. Beginning with texts that have simple data distributions and easy-to-extract features allows the model to establish a foundation for developing its capabilities and then build upon this foundation as it encounters more challenging texts within the chosen domain. Through this incremental approach, the model can achieve enhanced learning performance and greater mastery of difficult texts within the designated domain.

Building upon the aforementioned motivations, we present a novel framework termed "Keyword-weight-aware Curriculum Learning" (KCL) for Multi-Domain Text Classification

(MDTC). Our method is the first application of curriculum learning to multi-domain text classification. KCL incorporates two notable features:

(1) We calculate the weights of words in the texts and extract the Top-N words as domain-specific keywords. The sum of these N keywords' weights is then utilized as a measurement to measure the level of complexity of each domain's feature extraction. Higher sums indicate more apparent domain-specific features and simpler feature extraction, requiring prioritization of the domain corpus during model training.

(2) By employing varied approaches for keyword extraction and evaluating distinct quantities of keywords, we aim to identify the optimal domain order.

Experimental results demonstrate the efficacy of our proposed methodology in augmenting Multi-Domain Text Classification (MDTC) accuracy. Specifically, our approach reaches state-of-the-art outcomes on both the Amazon review and FDU-MTL datasets.

2. Related Work

Curriculum Learning: Bengio [13] introduced the concept of curriculum learning, with its focal point being the difficulty measurer of sample data and the data scheduling scheme of the training process. The sample difficulty measurer can be classified into two types: automatic and manual. The automatic difficulty measurer entails measuring the difficulty of samples through the model's performance, while the manual difficulty measurer is based on the grammatical and syntactic structure of the sample, such as measuring the number of nouns and sentence length. After years of development, curriculum learning strategies have exhibited impressive performance in the areas of response generation [17] and Contrastive Learning [18].

Pre-trained model: The process of utilizing pre-trained models in natural language processing (NLP) consists of two phases: pre-training and downstream task fine-tuning. Pre-training involves training the model on a vast corpus of data using a significant amount of computing resources. Once pre-training is complete, the model can then be applied to downstream tasks, and various parameters of the model can be fine-tuned during this stage of the training process. The use of pre-trained models has led to significant improvements in the performance of deep learning models in NLP, with models such as Bert surpassing human performance in SQuAD1.1 and achieving state-of-the-art (SOTA) results in 11 different NLP benchmarks. Consequently, these pre-trained models, including ELMO, XLNet, and ERNIE3.0, are widely applied in the NLP field due to their remarkable impact on the performance of deep learning models.

Multi-domain Text Classification: Li et al. [10] originally proposed multi-domain text classification to enhance a model's performance in a particular domain by combining data from multiple domains. The conventional approach was to adopt a shared-private structure to extract both domain-shared and domain-specific features. However, some teams subsequently applied adversarial learning [11,19] to differentiate between these features to avoid ambiguity. Attention mechanisms [9] and pre-trained models [12] have also been incorporated to improve the representation of text and to train more effective models.

3. Methods

As illustrated in Figure 2, KCL consists of two primary constituents: the first part entails keyword extraction and domain sequencing according to their respective keyword weight, while the second part involves a shared-private model as proposed in [11].

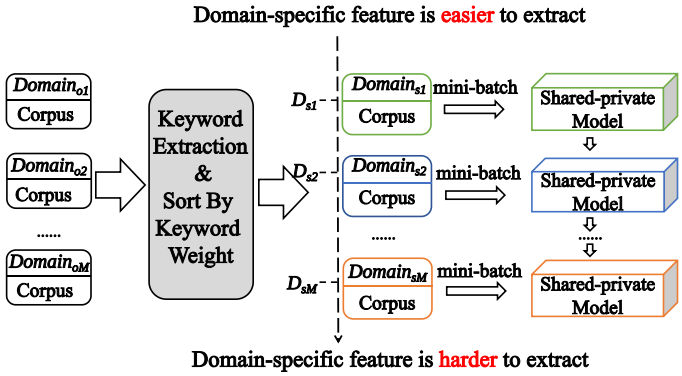

Figure 2. The architecture of KCL.

3.1. Keyword Extraction and Summing Up

The core of KCL involves a twofold approach: firstly, the keyword extraction algorithm is employed to extract the top-N keywords from each domain. Next, the apparent domain-specific features are quantified by computing the sum of the weights of these N keywords. To be specific, there are M original domains represented as $Domain_{o1}$, $Domain_{o2}$, ..., $Domain_{oM}$. The keyword extraction algorithm is applied to the corpus of these M domains, and the weights of all words within each domain corpus are calculated. Thereafter, the N words with the highest weight in each domain are identified as the keywords for that particular domain. The keywords obtained from each domain can be found in Figure 3.

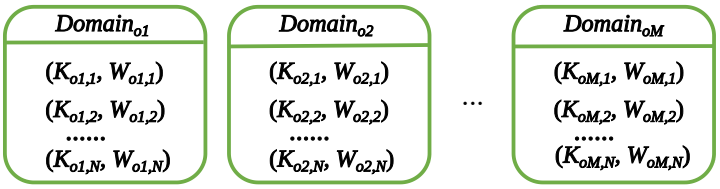

Figure 3. The sorted words list and its weight of each domain. $K_{oi,j}$ means the jth keyword of the $Domain_{oi}$, and the $W_{oi,j}$ means the weight of the keyword.

Then the obviousness of domain-specific feature of $Domain_{oi}$ is calculated as Equation (1):

$$W_{oi} = W_{oi,1} + W_{oi,2} + \ldots + W_{oi,N}. \quad (1)$$

The higher the value of W_{oi}, the more explicit and easily extracted the domain-specific features are; thus, the domains are prioritized in earlier steps of the model. Once sorted based on their W in descending order, a sorted list of domains is obtained as [$Domain_{s1}$, $Domain_{s2}$, ..., $Domain_{sM}$]. This domain list is abbreviated in Figure 2 as [D_{s1}, D_{s2}, ..., D_{sM}].

3.2. Shared-Private Model

Once the sorted domain list is obtained, we proceed to sample mini-batches from each domain in the order of [D_{s1}, D_{s2}, ..., D_{sM}] and input them into the shared-private model during each training step. As outlined in [11], the structure of the shared-private model of KCL is shown in Figure 4. For a given mini-batch $Input_{si}$ from D_{si}, the domain-shared feature extractor and the $Domain_{si}$-specific feature extractor process it separately. The outputs are then concatenated and fed into the Text Classifier after the domain-shared feature extractor has been judged by the *DomainDiscriminator*, which identifies the source domain of the sample. Upon applying the softmax function, the probability distribution of a given sample across the domains [D_{s1}, D_{s2}, ..., D_{sM}] can be extracted. The *DomainDiscriminator* objective function, J_{DD}, can be calculated utilizing Equation (2). The jth sample of D_{si} is

denoted as $sample_{si,j}$=($x_{si,j}$,$y_{si,j}$,D_{si}), where $x_{si,j}$ represents the text while $y_{si,j}$ conveys the text label.

$$J_{DD} = -\sum_{i=1}^{M} \sum_{sample_{si,j} \in D_{si}} logP(d_{si,j,pred} = D_{si}). \quad (2)$$

$d_{si,j,pred}$ represents the domain prediction of the jth sample in the D_{si}, and it is calculated as Equation (4). $P(d_{si,j,pred} = D_{si})$ means the probability that the prediction of the domain is right.

$$d_{si,j,dist} = DomainDiscriminator(F_s(x_{si,j})), \quad (3)$$

$$d_{si,j,pred} = softmax(d_{si,j,dist}). \quad (4)$$

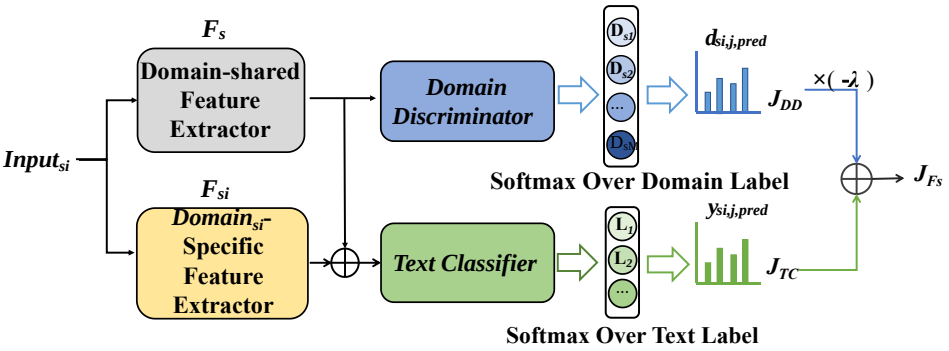

Figure 4. The architecture of the shared-private model.

The $TextClassifier$ is utilized to categorize each sample, whereby the outputs manifest as the logits of several labels [L_1, L_2, ...]. Upon applying the softmax function, the resultant probability distribution represents the likelihood that the forecasted label is identical to the actual label for a given sample. Consequently, the objective function of the $TextClassifier$, denoted J_{TC}, can be computed in accordance with Equation (5):

$$J_{TC} = -\sum_{i=1}^{M} \sum_{sample_{si,j} \in D_{si}} logP(y_{si,j,pred} = y_{si,j}). \quad (5)$$

$y_{si,j,pred}$ represents the prediction of the jth sample in the D_{si}, and it is calculated as Equation (7). $P(y_{si,j,pred} = y_{si,j})$ means the probability that the prediction of the text label is right.

$$y_{si,j,dist} = TextClassifier(concate(F_s(x_{ij}), F_{si}(x_{ij}))), \quad (6)$$

$$y_{si,j,pred} = softmax(y_{si,j,dist}), \quad (7)$$

where $concate(F_s(x_{ij}), F_{si}(x_{ij}))$ means concatenating the output of F_s and F_{si}.

Concerning each domain-specific feature extractor, its primary goal is to optimize the $TextClassifier$ to classify samples aptly. Subsequently, its objective function, similar to J_{TC}, can be defined. Specifically, the objective function of the F_{si} is determined via Equation (8):

$$J_{F_{si}} = -\sum_{sample_{si,j} \in D_{si}} logP(y_{si,j,pred} = y_{si,j}). \quad (8)$$

Concerning the domain-shared feature extractor, denoted as F_s, its ultimate objective is twofold: to enhance the accuracy of $TextClassifier$ while at the same time impairing the assessment of $DomainDiscriminator$ as much as possible. This dual focus stems from the fact that when text samples from diverse domains are fed through the domain-shared feature extractor and $DomainDiscriminator$ fails to identify their source domain, the extracted domain-shared features are regarded as domain invariant. As such, the objective function

of F_s should merge both the losses of $TextClassifier$ and $DomainDiscriminator$, leading to Equation (9):

$$J_{F_s} = J_{TC} + J_{DD} \cdot (-\lambda). \tag{9}$$

The hyperparameter λ is a positive value utilized to balance the weighting of $TextClassifier$ and $DomainDiscriminator$ within the domain-shared feature extractor. As the objective of the domain-shared feature extractor is to disrupt the evaluation of $DomainDiscriminator$, λ is negated prior to being utilized.

4. Experiments

4.1. Datasets

We conducted experiments in the field of multi-domain text classification using two well-known datasets: the Amazon review dataset [20,21] and the FDU-MTL dataset [22]. The Amazon review dataset is composed of product reviews in four diverse domains: DVD, Books, Electronics, and Kitchen, each with 1000 positive and negative reviews. The average length of all samples of the Amazon-review dataset is 140.88 words.

The FDU-MTL dataset is significantly larger than the Amazon review dataset, comprising a total of 16 domains: Books, Electronics, DVD, Kitchen, Apparel, Camera, Health, Music, Toys, Video, Baby, Magazine, Software, Sports, IMDB, and MR. The first 14 domains consist of product reviews sourced from Amazon, while IMDB and MR are movie reviews gathered from IMDB and Rotten Tomatoes, respectively. The average length of all samples of the FDU-MTL dataset is 131.73 words. Each domain within the FDU-MTL dataset comprises approximately 1600 labeled samples in the training set, 400 labeled samples in the test set, and 2000 unlabeled samples. The unlabeled sample data is suitable for training the $DomainDiscriminator$.

In each of the two datasets, the number of samples falling within different length intervals can be described as Tables 1 and 2.

Table 1. The number of samples of Amazon-review datasets within each length interval.

Length	**≤100**	**≤200**	**≤300**	**≤400**	**≤500**	**>500**
num	4199	2264	756	324	200	257

Table 2. The number of samples of FDU-MTL datasets within each length interval.

Length	**≤100**	**≤200**	**≤300**	**≤400**	**≤500**	**>500**
num	17,855	8332	2798	1208	644	942

4.2. Baselines

We researched some models that exhibit comparatively strong performance in multi-domain text classification, using these models as baselines to compare against our own model, including CAN [23], CRAL [19], COBE [12], MLP [11], MAN [11], CNN [9], and BERT [9]. It is worth noting that CNN and BERT are single-task learning methods for single-domain text classification. In order to demonstrate the effectiveness of our proposed multi-domain text classification model, we compare the results of single-domain text classification models as well, namely the CNN, BERT, and MLP models, to highlight the potential improvements in classification accuracy across multiple domains.

CAN proposes a conditional adversarial network, which constructs a conditional domain discriminator to model the difference between domain-shared features and domain-specific features and uses entropy conditioning to ensure the generalization of domain-shared features between different domains.

CRAL uses a double-adversarial network; that is, for each domain, two sets of models are used to predict the labels of samples in that domain. Each model includes a domain-shared feature extractor and a domain-specific feature extractor, and penalizes the difference in the

prediction results of the two models for the same sample. In addition, CRAL uses virtual adversarial training with entropy minimization to further enhance the classification effect.

MAN consists of four components: a domain-shared feature extractor, domain-specific feature extractors, a domain discriminator, and a text classifier. The domain-shared feature extractor is responsible for extracting invariant features that are shared across domains. Each domain also has its own private feature extractor that captures domain-specific information. The domain discriminator is used to train the domain-shared feature extractor; its goal is to learn features that generalize well across domains, making it difficult for the domain discriminator to identify the source domain of a given sample. Finally, the text classifier takes as input the domain-shared features and domain-specific features of a sample and outputs its classification label.

COBE is a combination of Bert's pre-training and contrastive learning. In the training stage, Bert is fine-tuned to obtain the representation of the sample, and then the K nearest neighbor algorithm is used to bring similar sentences closer and heterogeneous sentences farther away. After obtaining the representations of all training set samples and the fine-tuned Bert model in the training stage, the Bert model is used to obtain the representation of the test set samples in the testing phase. Note that one of the test set samples is $testsample_i$, then retrieve the most similar 4 samples from all the training set samples, and then calculate the sum of similarities of the same sample separately. The label corresponding to the class with the highest sum of similarities, in the end, is the classification result of the test sample. Currently, COBE achieved very promising results on both datasets, second only to our model.

4.3. Implementation Details

In the training phase, we adopted techniques from the WGAN framework and separately trained the domain discriminator and text classifier for each batch. Initially, we fixed the domain-shared feature extractor, domain-specific feature extractors, text classifier, and then trained the domain discriminator for 100 iterations at each step. We then proceeded to freeze the domain discriminator's parameters and trained the domain-shared and domain-specific feature extractors and text classifiers.

The domain discriminator and the other three modules were trained separately with the primary goal of improving the model's stability and enhancing its generalization ability. The domain discriminator serves a crucial role in counteracting the effects of the classifier, and if both models were trained together, they could potentially interfere with each other and result in model instability. By training the domain discriminator separately from the domain-shared feature extractor and domain-specific feature extractor, we could reduce the difficulty of training the discriminator. Our experimental results have demonstrated the effectiveness of this training strategy, and we achieved remarkable results with only 15 epochs. Ultimately, this approach enhances generalization by allowing the discriminator to focus on learning the feature distribution of the data while the classifier can concentrate on mapping feature vectors to corresponding labels. Thus, the domain discriminator's acquired characteristics can provide valuable assistance to the classifier in comprehending the data distribution, ultimately enhancing the classifier's ability to generalize. The rationale behind training the domain discriminator one hundred times more often than the text classifier originates from the substantial disparities in data distribution present across various domains. Consequently, creating a resilient domain discriminator can aid the classifier in developing a superior understanding of domain-specific features and, in turn, elevate the efficacy of classification.

We used the bert-base-uncased as the domain-shared feature extractor in our approach. To implement domain-specific feature extraction, we utilized a CNN with an input layer, a single-layer convolutional layer, and a fully-connected layer as MAN [11]. To achieve this, we utilized convolution kernels of 3 different sizes, constituting 200 kernels for each size, in the convolutional layer. Following the acquisition of characteristic features of various scales via convolutional kernels of diverse proportions, we performed global max pooling

to compress the output tensors of 600 convolutions into a sole output layer containing 600 values, which we used for the fully connected layer. Additional information regarding our use of the CNN can be found in [24]. Our domain discriminator and text classifier structures are relatively simple MLPs [25].

4.4. Keyword Extraction Algorithms

The process of extracting and computing keyword weights through various methods results in differing outcomes. Different sets of keywords may be generated, and the weights assigned to each keyword may vary. To ascertain the effects of disparate keyword extraction methods, we have chosen to select 50 keywords and conduct experiments specifically aimed at comparing the performance of YAKE [26], TextRank [27], and KeyBERT [28].

We chose to use these three keyword extraction algorithms for several reasons:

1. Wide Applicability. These algorithms have extensive applicability in both academic and industrial settings and are relatively easy to use.
2. High Processing Speed. All three algorithms are capable of processing large-scale text data and can complete the task of keyword extraction within a relatively short time frame.
3. High Accuracy of Results. These three algorithms are capable of extracting the most important keywords from text in practical applications with high accuracy.
4. Holistic Consideration of the Corpus. All three algorithms have the ability to model context, such as textRank and Yake, which can comprehensively consider the relationships between words in the corpus. KeyBert also utilizes language pre-training models to integrate contextual information.
5. High performance for long texts. All three algorithms have been found to be particularly effective in extracting keywords from long text.

In the following sections, we briefly introduce these three keyword extraction algorithms.

YAKE [26]: The preprocessing of the text by YAKE mainly includes the following steps:

1. Split the text into sentences.
2. Split the sentence into chunks based on punctuation marks.
3. Split chunks into a series of tokens.
4. Tag the tokens in tokens(including Digit, Number, unparsable Content, Acronyms, Uppercase, and Parsable Content), convert the letters of tokens into lowercase letters, and judge whether they are stop words.

During feature extraction and weighting calculation stages, the YAKE algorithm first calculates various properties of each 1-gram term, such as term frequency, index of the sentence in the text that contains this term, number of times it appears as an abbreviation, and number of times it appears as a capitalized non-first word. It also calculates the co-occurrence probability of pairs of terms within a certain window size, creating a large co-occurrence matrix. Using this information, YAKE can construct more complex features and weight scores for each word and use them as the weight for each candidate keyword.

During the keyword extraction stage, the algorithm utilizes the previously calculated keyword weight to extract the top-ranking words as the final keywords.

TextRank [27]: The TextRank algorithm is a graph-based text summarization algorithm. It uses the PageRank [29] algorithm in graph theory to calculate the weights of keywords and sentences in the text and sorts them according to the weights to generate text summaries.

The algorithm steps are as follows:

1. The input text is segmented and part-of-speech tagged, and then an undirected graph is constructed based on the co-occurrence relationship between words. Each node represents a word, and the edges represent the co-occurrence relationship between words.
2. Calculate the weight of the nodes in the graph. The weight of a node is determined by the weight of the node and its surrounding nodes.

3. The nodes are sorted, and the sorting is based on the weight of the nodes. Top-ranked nodes represent important words.
4. Sentences corresponding to the top-ranked nodes are extracted to generate text summaries.

The advantage of TextRank is that it does not require prior knowledge and can automatically extract important information from the text, so it is widely used in tasks such as text summarization and keyword extraction.

KeyBERT [28]: The KeyBERT algorithm is a keyword and phrase extraction algorithm based on the BERT pre-training model. It uses the BERT model to calculate the semantic similarity between words or phrases and the original text and sorts them according to the similarity to extract keywords in the text and phrases.

The algorithm steps are as follows:

1. Enter the text from which keywords and phrases need to be extracted.
2. Encode text into a sequence of word vectors using a BERT model.
3. Segments the input text, treating each paragraph as a separate document.
4. Cluster the sequence of word vectors in each document to generate several representative vectors, each corresponding to a cluster.
5. For each cluster, calculate the cosine similarity with each word vector in the original text so as to obtain a vector of similarity vectors with each word in the text.
6. The similarity vectors are sorted, and the top k words or phrases are taken out as keywords and phrases.

The advantage of the KeyBERT algorithm is that it can process long texts, and at the same time, it can take into account the semantic similarity of words or phrases in the entire text, thereby improving the accuracy of keyword and phrase extraction. It has been widely used in natural language processing, information retrieval, and other fields.

4.5. Main Results

We compared the performance of KCL and other baseline methods on the FDU-MTL dataset and the Amazon-review dataset, as shown in Tables 3 and 4 with our experimental results. For consistency with previous multi-domain text classification experiments, our evaluation metric is the classification accuracy on the test set. The "KCL-XXX" column denotes the utilization of the XXX method for keyword extraction. KCL-r denotes that the order of domains is random. KCL-Y, KCL-TR, and KCL-KB correspond to the use of YAKE, TextRank, and KeyBERT, respectively, as the keyword extraction algorithms in our analysis. We followed the approach taken by MAN [11] for the division of our experimental data, using 5-fold cross-validation to split the FDU-MTL and Amazon review datasets into training, validation, and testing sets in a 3:1:1 ratio. To obtain the final experimental results, we computed the average testing accuracy over 5-fold cross-validation runs and repeated this process 10 times to obtain the final outcome of our experiment. The experimental data of CNN, MLP, and MAN were obtained from MAN [11]. Likewise, the experimental data of Bert and CRAL were obtained from CRAL [19]. The experimental data for CAN were obtained from CAN [23], while the experimental data for COBE were obtained from COBE [12].

After reviewing Tables 3 and 4, we can conclude that our method outperforms other baselines in 14 out of the 16 domains included in the FDU-MTL dataset. As for the Amazon-review dataset, which contains merely 4 domains, our method demonstrates significant superiority over other baselines across all of them. Among these baselines, COBE and CRAL were state-of-the-art in the field of multi−domain text classification on FDU-MTL and Amazon review datasets, respectively, before we proposed our method. In the FDU-MTL dataset, our method outperforms COBE by 1.13 points in average accuracy when using KeyBERT as the keyword extraction algorithm. Similarly, in the Amazon review dataset, our method outperforms CRAL by 3.27 points in average accuracy when employing TextRank as the keyword extraction algorithm.

Table 3 indicates that utilizing KeyBERT as the keyword extraction algorithm yields the most successful results on the FDU-MTL dataset, surpassing the random order by

1.55 points on average accuracy. The order of the domains of the FDU-MTL dataset is as follows: [Camera, Health, Kitchen, Software, MR, Apparel, Books, Magazine, Video, DVD, Music, Baby, Sports, IMDB, Toys, Electronics]. In addition, Table 4 reveals that leveraging TextRank as the keyword extraction algorithm proves to be most effective on the Amazon review dataset, surpassing the random order by 1.79 points in average accuracy. The current ranking of the Amazon review dataset is as follows: [DVD, Books, Electronics, Kitchen]. Furthermore, we can find that our KCL excels in each domain relative to those single-task learning approaches.

Table 3. The accuracy of different keyword extraction methods on FDU-MTL dataset. The use of bold font signifies that the method performs best in the domain.

Domain	CNN	BERT	CAN	CRAL	COBE	KCL-r	KCL-Y	KCL-TR	KCL-KB
Books	85.30	87.00	87.80	89.30	90.17	92.42	93.00	**93.42**	93.08
Electronics	87.80	88.30	91.60	89.10	93.58	93.5	93.33	94.00	**94.92**
DVD	76.30	85.60	89.50	**91.00**	89.67	88.42	89.42	89.58	89.92
Kitchen	84.50	91.00	90.80	92.30	91.50	91.08	92.08	**92.67**	92.50
Apparel	86.30	90.00	87.00	91.60	92.33	92.25	92.42	92.08	**92.67**
Camera	89.00	90.00	93.50	96.30	93.58	91.92	93.00	**93.92**	93.67
Health	87.50	88.30	90.40	87.80	93.92	94.42	94.33	94.75	**95.67**
Music	81.50	86.80	86.90	88.10	90.33	89.08	88.50	**91.00**	90.42
Toys	87.00	91.30	90.00	91.60	93.42	92.5	92.75	**93.67**	93.33
Video	82.30	88.00	88.80	**92.60**	89.91	88.58	91.00	90.67	91.67
Baby	82.50	91.5	92.00	90.90	93.92	93.58	93.50	**94.75**	94.58
Magazine	86.80	92.8	94.50	95.20	94.08	91.67	92.67	93.50	**94.17**
Software	87.50	89.3	90.90	87.70	93.42	93.75	92.33	**95.00**	94.33
Sports	85.30	90.8	91.20	91.30	92.83	92.67	93.41	94.25	**94.42**
IMDB	83.30	85.80	88.50	90.80	86.91	89.08	89.83	90.42	**90.83**
MR	79.00	79.00	77.10	77.30	84.33	82.33	84.75	84.33	**85.58**
Avg	84.30	84.30	89.40	90.20	91.49	91.07	91.64	92.33	**92.62**

Table 4. The accuracy of different keyword extraction methods on Amazon review dataset. The use of bold font signifies that the method performs best in the domain.

Domain	MLP	MAN	CAN	CRAL	KCL-r	KCL-Y	KCL-TR	KCL-KB
Books	82.40	82.98	83.76	85.26	89.25	91.00	**91.83**	89.75
Electronics	82.15	84.03	84.68	85.83	90.50	89.50	**91.75**	91.67
DVD	85.90	87.06	88.34	89.32	90.33	90.25	**91.33**	91.08
Kitchen	88.20	88.57	90.03	91.60	91.08	91.33	**93.42**	92.58
Avg	84.66	85.66	86.70	88.00	90.29	90.51	**92.08**	91.27

4.6. Analysis

4.6.1. The Drawbacks of Previous Methods

In this subsection, we discuss two typical and effective models to illustrate the shortcomings of previous methods.

Firstly, CRAL employs adversarial networks for multi-domain text classification and has achieved good results. Compared with earlier models that used only one adversarial network, such as CAN and MAN, CRAL uses two adversarial networks for each domain. However, this method, which relies on stacking network models and parameters, is relatively dependent on computational resources and has a complex network structure. A simpler alternative would be to adjust the training order after considering the difficulty of domain feature extraction.

Secondly, COBE, which utilizes pre-trained models, contrastive learning, and the K-nearest neighbor algorithm, has achieved good results on the FDU-MTL dataset. However, during training, since the order of the training data was not taken into account, the pre-trained model may have processed domain texts with difficult feature extraction at the

beginning. A more accurate approach would be to first arrange the training order of domain texts from easy to difficult and then extract more precise features for the domains with challenging feature extraction, resulting in more accurate standards.

4.6.2. Different Performance on Different Datasets

Different models show varying performances on Amazon review and FDU-MTL datasets, and we believe this is related to the characteristics of the keyword extraction algorithm. KeyBERT excels at extracting keywords from shorter texts, whereas the TextRank algorithm is good at extracting keywords from longer texts. To extract keywords from a specific domain, we aggregate all samples within that domain and apply the keyword extraction algorithm to the resulting whole. The average length of a domain in the FDU-MTL dataset is 208,865, whereas, in the Amazon review dataset, the average length is 221,889, exceeding that of the FDU-MTL dataset by over 20,000 words. Consequently, TextRank, which is better suited for processing longer texts, accurately extracts keywords and weights from the Amazon review dataset, resulting in superior performance on the Amazon review dataset.

4.6.3. The Number of Keywords

To determine the obviousness of domain-specific features, varying quantities of keywords can yield different ranking outcomes when considering the cumulative weighting factors. This is due to the possibility that a domain's top N1 keywords may carry a greater total weight than any other domain. However, the combined weight of a domain's first N2 (where N2 > N1) keywords may still be less than that of any other domain. Accordingly, we conducted experiments in this section to investigate how different numbers of keywords included in weight calculation impacted accuracy.

We conducted experiments using 20, 30, 40, 50, 60, 70, and 80 keywords to test their effects. The classification accuracy and average accuracy per domain are displayed in Figures 5 and 6, respectively. Based on our findings, we observed that the most effective number of keywords was 50.

Based on Figure 5a–d, it can be observed that when the keyword count is 50, the classification results of 15 out of 16 domains of the FDU-MTL dataset, except for the IMDB dataset perform the best. On the other hand, when the keyword count is too high or too low, the average classification accuracy even fails to match the COBE baseline. We attribute this to the fact that with too few keywords, the weights of the keywords differ only slightly from each other, mimicking random order and subsequently rendering it difficult to extract distinguishing features for each domain. Conversely, when there are too many keywords, irrelevant and insignificant keywords contribute to the weight computations, thus causing erroneous ordering.

As for the Amazon-review dataset, while there are two domains that achieve the best classification accuracy at both 40 and 50 keywords individually, the average classification accuracy across all domains is slightly higher for 50 keywords than for 40 keywords. Therefore, we still consider 50 keywords to be the optimal choice for ordering in this dataset.

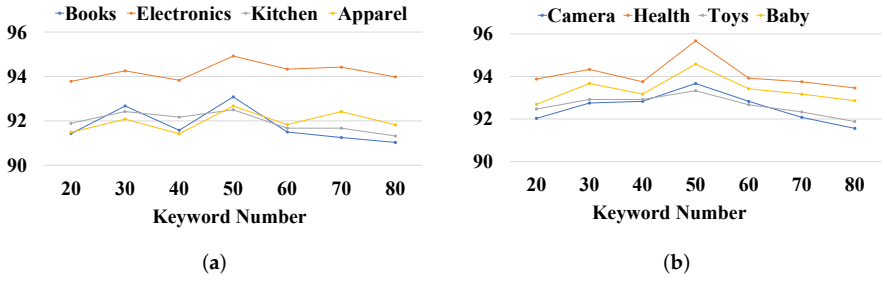

Figure 5. *Cont.*

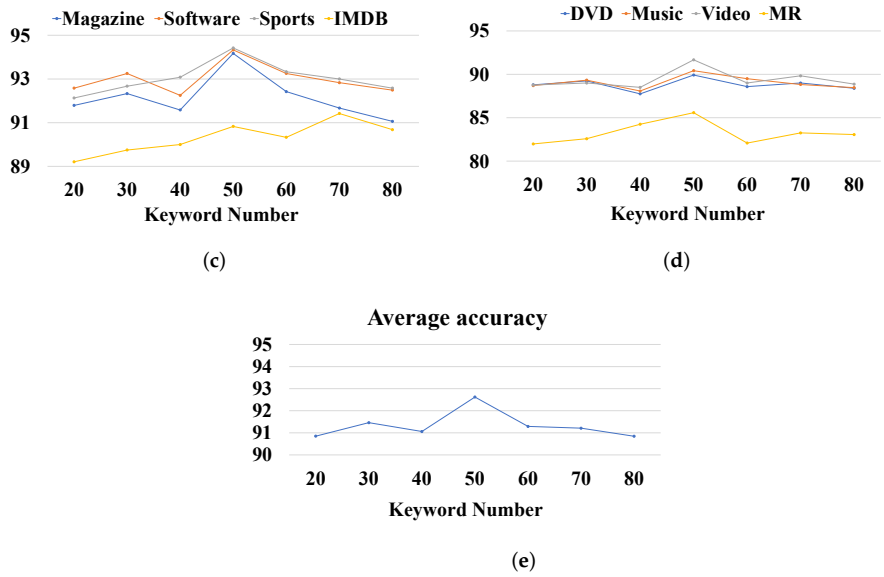

Figure 5. The accuracy of different numbers of keywords on the FDU-MTL dataset, using KeyBERT as the keyword selection method. (**a**) The accuracy of different numbers of keywords on Books, Electronics, Kitchen, Apparel domain of FDU-MTL dataset, using KeyBERT as the keyword selection method. (**b**) The accuracy of different numbers of keywords on Camera, Health, Toys, Baby domain of FDU-MTL dataset, using KeyBERT as the keyword selection method. (**c**) The accuracy of different numbers of keywords on Magazine, Software, Sports, IMDB domain of FDU-MTL dataset, using KeyBERT as the keyword selection method. (**d**) The accuracy of different numbers of keywords on DVD, Music, Video, MR domain of FDU-MTL dataset, using KeyBERT as the keyword selection method. (**e**) The average accuracy of different numbers of keywords on the domain of FDU-MTL dataset, using KeyBERT as the keyword selection method.

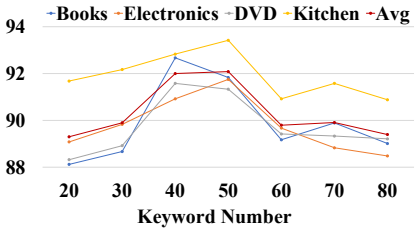

Figure 6. The accuracy of different numbers of keywords on the Amazon-review dataset, using TextRank as the keyword selection method.

5. Conclusions

In this paper, we propose KCL, a network that significantly enhances the multi-domain text classification model in Amazon review datasets and FDU-MTL datasets. Our approach involves using a keyword extraction algorithm to determine the weights of words in the corpus of each domain. The N words with the highest weights serve as the domain keywords, and the sum of their respective weights is calculated to determine the level of difficulty in extracting domain-specific features. The higher the sum, the easier it is to extract the domain-specific features. Hence, we prioritized training samples from easier domains in each training step. Experimental findings from our tests on the Amazon review and FDU-MTL datasets show our model achieved state-of-the-art performance in multi-domain text classification, and taking the sum of the weights of the 50 words with the highest weights as the basis for sorting between domains can obtain the best performance. However, there is still room for improvement in our work. Currently, the datasets available for multi-domain text classification are just focused on product reviews, with no other options available. As a result, it is necessary to conduct further verification of the efficacy

of this approach in other domains. In future work, we intend to allocate resources and personnel to build a multi-domain dataset that includes more domains so as to further verify the effectiveness of our method.

Author Contributions: Conceptualization, Z.Y., Y.L. (Yinghui Li) and H.-T.Z.; Methodology, Z.Y., Y.L. (Yinghui Li), Y.L. (Yangning Li) and Y.H.; Software, Z.Y.; Validation, Z.Y.; Formal analysis, Z.Y. and Y.L. (Yinghui Li); Investigation, Z.Y.; Writing—original draft, Z.Y.; Writing—review & editing, Z.Y., Y.L. (Yinghui Li), Y.L. (Yangning Li), H.-T.Z., Y.H., W.L., D.H. and B.W.; Supervision, H.-T.Z.; Project administration, H.-T.Z.; Funding acquisition, H.-T.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the National Natural Science Foundation of China (Grant No. 62276154), the Research Center for Computer Network (Shenzhen) Ministry of Education, Beijing Academy of Artificial Intelligence (BAAI), the Natural Science Foundation of Guangdong Province (Grant No. 2023A1515012914), Basic Research Fund of Shenzhen City (Grant No. JCYJ20210324120012033 and JSGG20210802154402007), the Major Key Project of PCL for Experiments and Applications (PCL2021A06), and Overseas Cooperation Research Fund of Tsinghua Shenzhen International Graduate School (HW2021008).

Data Availability Statement: The FDU-MTL dataset and Amazon review dataset are available at https://github.com/LuoXiaoHeics/COBE/blob/main/data.zip.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sahmoud, T.; Mikki, D.M. Spam detection using bert. *arXiv* **2022**, arXiv:2206.02443.
2. Nugroho, K.S.; Sukmadewa, A.Y.; Yudistira, N. Large-scale news classification using bert language model: Spark nlp approach. In Proceedings of the 6th International Conference on Sustainable Information Engineering and Technology, Malang, Indonesia, 13–14 September 2021; pp. 240–246.
3. Agarap, A.F. Statistical analysis on e-commerce reviews, with sentiment classification using bidirectional recurrent neural network (rnn). *arXiv* **2018**, arXiv:1805.03687.
4. McCallum, A.; Nigam, K. A comparison of event models for naive bayes text classification. In Proceedings of the AAAI, Madison, WI, USA, 26–30 July 1998.
5. Zhang, Y.; Wallace, B. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv* **2015**, arXiv:1510.03820.
6. Liu, P.; Qiu, X.; Huang, X. Recurrent neural network for text classification with multi-task learning. *arXiv* **2016**, arXiv:1605.05101.
7. Lai, S.; Xu, L.; Liu, K.; Zhao, J. Recurrent convolutional neural networks for text classification. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
8. Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; Hovy, E. Hierarchical attention networks for document classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016; pp. 1480–1489.
9. Cai, Y.; Wan, X. Multi-domain sentiment classification based on domain-aware embedding and attention. In Proceedings of the IJCAI, Macao, China, 10–16 August 2019; pp. 4904–4910.
10. Li, S.; Zong, C. Multi-domain sentiment classification. In Proceedings of the ACL-08: HLT, Columbus, OH, USA, 15–20 June 2008; Short Papers; Association for Computational Linguistics, Columbus, OH, USA, 2008; pp. 257–260.
11. Chen, X.; Cardie, C. Multinomial adversarial networks for multi-domain text classification. *arXiv* **2018**, arXiv:1802.05694.
12. Luo, Y.; Guo, F.; Liu, Z.; Zhang, Y. Mere contrastive learning for cross-domain sentiment analysis. *arXiv* **2022**, arXiv:2208.08678.
13. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 41–48.
14. Dai, Y.; Li, H.; Li, Y.; Sun, J.; Huang, F.; Si, L.; Zhu, X. Preview, attend and review: Schema-aware curriculum learning for multi-domain dialog state tracking. *arXiv* **2021**, arXiv:2106.00291.
15. Wei, J.; Huang, C.; Vosoughi, S.; Cheng, Y.; Xu, S. Few-shot text classification with triplet networks, data augmentation, and curriculum learning. *arXiv* **2021**, arXiv:2103.07552.
16. Zhang, D.; Li, Y.; Zhou, Q.; Ma, S.; Li, Y.; Cao, Y.; Zheng, H.-T. Contextual similarity is more valuable than character similarity: An empirical study for chinese spell checking. *arXiv* **2022**, arXiv:2207.09217.
17. Shen, L.; Feng, Y. Cdl: Curriculum dual learning for emotion-controllable response generation. *arXiv* **2020**, arXiv:2005.00329.
18. Ye, S.; Kim, J.; Oh, A. Efficient contrastive learning via novel data augmentation and curriculum learning. *arXiv* **2021**, arXiv:2109.05941.
19. Wu, Y.; Inkpen, D.; El-Roby, A. Co-regularized adversarial learning for multi-domain text classification. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Virtual, 28–30 March 2022; pp. 6690–6701.

20. Fang, F.; Dutta, K.; Datta, A. Domain adaptation for sentiment classification in light of multiple sources. *Inf. J. Comput.* **2014**, *26*, 586–598. [CrossRef]
21. Dong, C.; Li, Y.; Gong, H.; Chen, M.; Li, J.; Shen, Y.; Yang, M. A survey of natural language generation. *arXiv* **2021**, arXiv:2112.11739.
22. Liu, P.; Qiu, X.; Huang, X. Adversarial multi-task learning for text classification. *arXiv* **2017**, arXiv:1704.05742.
23. Wu, Y.; Inkpen, D.; El-Roby, A. Conditional adversarial networks for multi-domain text classification. *arXiv* **2021**, arXiv:2102.10176.
24. Soll, M.; Hinz, T.; Magg, S.; Wermter, S. Evaluating defensive distillation for defending text processing neural networks against adversarial examples. In Proceedings of the International Conference on Artificial Neural Networks, Crete, Greece, 24–26 May 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 685–696.
25. Liu, R.; Li, Y.; Tao, L.; Liang, D.; Zheng, H. Are we ready for a new paradigm shift? a survey on visual deep mlp. *Patterns* **2022**, *3*, 100520. [CrossRef] [PubMed]
26. Campos, R.; Mangaravite, V.; Pasquali, A.; Jorge, A.; Nunes, C.; Jatowt, A. Yake! keyword extraction from single documents using multiple local features. *Inf. Sci.* **2020**, *509*, 257–289. [CrossRef]
27. Mihalcea, R.; Tarau, P. TextRank: Bringing Order into Text. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 25–26 July 2004; Association for Computational Linguistics: Barcelona, Spain, 2004; pp. 404–411.
28. Grootendorst, M. Keybert: Minimal Keyword Extraction with Bert. 2020. Available online: https://github.com/MaartenGr/KeyBERT (accessed on 1 July 2023).
29. Brin, S.; Page, L. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.* **1998**, *30*, 107–117. [CrossRef]