教育資料與圖書館學

Journal of Educational Media & Library Sciences

http://joemls.tku.edu.tw

Vol. 57 , no. 1 (2020) : 121-144

主題分析自動化的可行性：深度學習技術應用於偏態分佈之中文文件分類的實證評估ψ

The Feasibility of Automated Topic Analysis: An Empirical Evaluation of Deep Learning Techniques Applied to Skew-Distributed Chinese Text Classificationψ

Yuen-Hsien Tseng
Distinguished Professor and Deputy Chair
E-mail：samtseng@ntnu.edu.tw

English Abstract & Summary see link at the end of this article

Journal of Educational Media & Library Sciences 57 : 1 (2020) : 121-144
DOI:10.6120/JoEMLS.202003_57(1).0047.RS.CE

Research Article

The Feasibility of Automated Topic Analysis: An Empirical Evaluation of Deep Learning Techniques Applied to Skew-Distributed Chinese Text Classification[ψ]

Yuen-Hsien Tseng

Abstract

Text classification (TC) is the task of assigning predefined categories (or labels) to texts for information organization, knowledge management, and many other applications. Normally the categories are topical in library science applications, although they can be any labels suitable for an application. Thus, TC often requires topical analysis which relies on human knowledge. However, in recent decades, machine learning (ML) techniques have been applied to TC for efficiency, as long as a sufficient number of training texts are available for each category. Nevertheless, in real-world cases, the number of texts (documents) for each category is often highly skewed for a certain TC task. This leads to the problem of predicting labels for small categories, which is viable for humans but challenging for machines. Deep learning (DL) is an emerging class of machine learning (ML) which was inspired by human neural networks. This study aims to evaluate whether DL techniques are feasible for the mentioned problem by comparing the performance of four off-the-shelf DL methods (CNN, RCNN, fastText, and BERT) with four traditional ML techniques on five skew-distributed datasets (four in Chinese, and one in English for comparison). Our results show that BERT is effective for moderately skewed datasets, but is still not feasible for highly skewed TC tasks. The other three DL-aware methods (CNN, RCNN, fastText) do not show any advantage in comparison with traditional methods such as SVM for the five TC tasks, although they captured extra language knowledge in the pretrained word representation. To facilitate future study, all of the Chinese datasets used in this study have been released publicly, together with all of the adapted machine learning and evaluation source codes for verification and for further study at https://github.com/SamTseng/Chinese_Skewed_TxtClf.

Keywords: *Text categorization, Real-world corpus, Deep learning, Performance evaluation*

[ψ] Part of manuscript has been published in Yuen-Hsien Tseng, "An Empirical Evaluation of Deep Learning Techniques Applied to Skew-Distributed Text Classification," Proceedings of the International Conference on Library and Information Science (ICLIS), pp. 303-310, Taipei, Taiwan, July 12-13, 2019 (with a total of 2,915 words and different conclusions). The current paper is a substantial revision with 8,414 words and different conclusions.

Distinguished Professor and Deputy Chair, Graduate Institute of Library & Information Studies, National Taiwan Normal University, Taipei, Taiwan
Sub-Investigator, MOST AI Biomedical Research Center, Taipei, Taiwan
E-mail: samtseng@ntnu.edu.tw

2019/10/05 received; 2020/03/08 revised; 2020/03/12 accepted

Introduction

Deep learning (DL) is a class of machine learning (ML) algorithms that use multiple layers of connected nonlinear processing units inspired by neural networks in the human brain (LeCun et al., 2015). The characteristics that distinguish DL from traditional ML is that DL techniques can automatically learn the knowledge representation features, either spatial or temporal, needed for a task. In recent years, DL techniques have been successfully applied to such areas as speech recognition, computer vision (Russakovsky et al., 2015), and natural language processing (Devlin et al., 2019). The success of DL in end-to-end applications can be attributed to the breakthrough of learning algorithms, feature representation (e.g., word embedding [Mikolov, Sutskever, et al., 2013]), and network architectures (e.g., convolutional neural networks, or CNN [Alex et al., 2012], recurrent neural networks, or RNN [Hochreiter & Schmidhuber, 1997], and transformers [Vaswani et al., 2017]). Many fields have attempted to apply DL to better deal with traditional tasks or to innovate new applications.

Text classification (TC) is the task of assigning predefined labels (or categories) to texts or documents for data analysis/mining, information browsing/searching, knowledge organization/management, and many others. A number of studies have applied deep learning to text classification (Chen & Lee, 2017; Lai et al., 2015; X. Zhang et al., 2015). Most studies have evaluated their approaches on datasets with relatively balanced document distribution, that is, the number of documents for each category is fairly even (X. Zhang et al., 2015). However, in real-world cases, texts often follow Zipf's law: the distribution of the frequencies of terms is highly skewed, as is the distribution of the documents to the categories to which they belong. In other words, most documents belong to a few categories, and most categories have very few documents. A good machine classifier should correctly predict a document's category not just for a few large categories, but also for many more small categories with limited training data in the same TC task.

The problem of correct prediction for small categories is challenging and is a long-standing issue in TC study. As DL has been successfully applied to many tasks, even better than humans in some cases, the question that follows is: can the emerging DL techniques solve this problem better than traditional methods to the extent that automated topic analysis with minimal human involvement becomes feasible. To answer this question, in this study we compared the performance of four off-the-shelf DL methods with four traditional machine learning techniques on five skew-distributed datasets.

In library science, this is an important question to answer, as information organization and topic analysis is one of the main technical services in library/

institute practices. In this age of the resurgence of Artificial Intelligence (AI), whether the recent techniques are mature enough to augment human power to an unprecedented level is worth examining and exploring. Therefore, this study is not technique-oriented (i.e., it does not aim to devise more advanced methods for further performance improvement), but rather, it examines existing tools to test their capability for immediate application.

The rest of the paper is organized as follows: The next section briefly reviews related work to further motivate this study. Section 3 clarifies the research questions and possible contributions of this work. Section 4 introduces the machine learning methods for performance comparison. In Section 5, the five real-world corpora for text classification are described. Section 6 details the experiment results. The conclusions are summarized in the final section.

Related Work

Automated text classification has been studied for decades. This section reviews some representative studies based on both the traditional ML and the DL methods. Because there is a huge body of literature on automated text classification, including research issues on document representation, feature selection, classifier construction, parameter tuning, evaluation metrics, and various applications (e.g., classification by topic, source, language, emotion, spam, etc.), our emphasis is only on those studies which experimented on topical classification for skewed datasets.

In the late 1990s, various text classifiers were proposed. Many different studies experimented on datasets of different variations, leading to slightly inconsistent conclusions. Y. Yang and Liu (1999), therefore, compared the effectiveness of five classifiers, namely SVM (Support Vector Machine), kNN (k-Nearest Neighbors), Linear Least Square Fit (LLSF), Neural Networks (NNet), and Naïve Bayes (NB), based on the real-world (skew-distributed) Reuters-21578 dataset (described later), with the results verified by a statistical significance test. Their experiments showed that SVM, kNN, and LLSF significantly outperformed NNet and NB when the number of positive training instances per category was small (less than 10), and that all the methods performed comparably when the categories were sufficiently common (over 300 instances). Therefore, a genuinely effective text classifier should perform well for categories no matter whether there are sufficient positive examples or not.

Lewis et al. (2004) extensively described a new large Reuters dataset called RCV1-v2 for text categorization research. The RCV1-v2 dataset[1] contains

[1] RCV1-v2 is available at: http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm, accessed on 2020/02/18.

804,414 documents which were split into 23,149 training documents and 781,265 test documents (this split is called the LYRL2004 split). There are in total 103 topic categories, the distributions of which are extremely skewed: the largest category has 374,316 texts while the smallest has only five. The authors benchmarked the dataset with SVM, kNN, and Rocchio classifiers and confirmed the findings of past studies: SVM is dominant, kNN is competitive, and the Rocchio algorithm is only plausible. The best SVM performance was 81.6% for MicroF and 60.7% for MacroF (these metrics will be described later). It is noted that the MacroF value is lower than the MicroF value, meaning that many small categories were not classified well, compared to the large categories. The authors did not particularly study the skewed category problem, that is, they did not try to improve the MacroF for this dataset.

The review paper of Sebastiani (2002) summarized the TC studies around 2000. He pointed out that the evaluation of text classifiers is typically conducted experimentally, rather than analytically, due to its subjective characteristics. Therefore, the more datasets (with diverse domains) that are used in TC experiments, the more reliable and consistent the insights drawn from the experiments will be. This can be observed when sentiment analysis was studied, beginning in 2002 (Pang et al., 2002; Turney, 2002), an increasing number of datasets for sentiment analysis (most from Twitter) were made publicly available. There were at least nine by 2013, as described by Saif et al. (2013). However, these sentiment datasets have few categories, normally only three: positive, negative, and neutral. Even with so few categories, Pang et al. (2002) still stated that "studying the effect of skewed class distributions was out of the scope of this study," which confirms that skew-distributed TC is another issue that needs to be addressed in addition to those mentioned earlier (such as feature extraction, parameter tuning, etc.).

In addition to Reuters datasets, there are only a few commonly used datasets for topic-based TC study. Sebastiani (2002) introduced only three more datasets: one is proprietary (the AP collection)[2], and the other two are publicly available. These two public datasets are described as follows: 1. OHSUMED[3] is a set of 348,566 bibliographic records from 270 medical journals over a 5-year period (1987-1991) from the online medical information database MEDLINE. The available fields are title, abstract, MeSH indexing terms, author, source, and publication type. The categories are the MeSH terms or the clusters of MeSH terms under certain diseases, depending on how researchers use this dataset

[2] Details of the AP collection can be found at: http://www.daviddlewis.com/resources/testcollections/trecap/, accessed on 2020/02/18.

[3] OHSUMED is available at: https://trec.nist.gov/data/t9_filtering.html, accessed on 2020/02/18.

(Joachims, 1998). 2. The 20 newsgroups (20NG)[4] contains messages posted to 20 Usenet newsgroups. Each group has about 1,000 texts and thus there are a total of approximately 20,000 texts, which make it a balanced dataset. Considering the characteristics of these datasets, it can be seen that there are insufficient real-world datasets for experiments on skew-distributed TC tasks.

With the resurgence of AI research momentum (more data and more computation power have become easily accessible), various DL techniques equipped with pre-trained word embedding, sequence recognition, and attention mechanisms for semantic understanding have been proposed for better performance in various TC-related tasks. Example DL techniques include CNN, RCNN, fastText, and BERT, as described below.

Johnson and Zhang (2015) proposed Convolutional Neural Networks (CNN) for sentiment prediction of movie reviews (dataset name: IMDB), Amazon's electronic product reviews (dataset name: Elec), and for topical classification of news (dataset name: RCV1-v2). They reported the error rates (number of texts with incorrect category prediction divided by the total number of texts) for these three datasets. Additionally, they reported MicroF and MacroF measures for RCV1-v2 with 84.0% and 64.8%, respectively. This performance is obviously better than that of Lewis et al.'s (2004) results mentioned above. Therefore, CNN is a strong classifier that is worth testing on more topic-based TC tasks in different domains.

Lai et al. (2015) proposed Recurrent Convolutional Neural Networks (RCNN) and experimented with this new approach on four datasets, namely, 20NG for topical discussion group classification, the Fudan set for Chinese TC with 20 topical categories, the ACL Anthology Network for the prediction of five native languages of the authors of scientific papers, and the Stanford Sentiment Treebank for predicting five sentiment levels for movie reviews. Although they do not report the skewedness of these datasets, their use of accuracy (number of correctly classified texts divided by number of all texts) as the main performance metric (for the last three datasets) suggests that the datasets are balanced (otherwise the use of accuracy would be an improper measure). They did use MacroF for the 20NG, which is a balanced dataset. Therefore, how well RCNN will perform on skewed datasets is unknown.

Another approach, fastText (Joulin et al., 2016) reported accuracy on eight sentiment analysis datasets and reported precision of 1 (denoted as Prec@1, i.e., the number of correct predictions in the first place when ranking the tags) on a huge tag prediction dataset called YFCC100M where there are 312,116 unique

[4] 20NG is available at: http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html, accessed on 2020/02/18.

image tags for prediction (each tag occurs more than 100 times) based on the image titles and captions with 91,188,648 training examples and 543,424 testing examples. It was found that fastText performed better or was often on a par with various DL classifiers in terms of accuracy, and many orders of magnitude faster for training and evaluation. The accuracy or Prec@1 did not reveal how well fastText could boost the performance of low-frequency categories.

BERT (Devlin et al., 2019) obtained state-of-the-art results on 11 natural language processing tasks, most of which were related to sentence pair inferencing (the task names are: QQP, QNLI, STS-B, MRPC, and WNLI) or entailment prediction (MNLI, RTE, and SWAG). Two tasks were to predict a single sentence's sentiment (SST-2) or linguistic acceptability (CoLA), and two were about a question answering test (SQuAD v1.1 and SQuAD v2.0). In other words, the 11 tasks consisted of a variety of semantic classification tasks, but none were related to topical classification which also needs semantic understanding for better text classification.

To compare the performance of different classifiers, Sebastiani (2002) also pointed out that different sets of experiments may be used for cross-classifier comparison only if the experiments have been performed: 1. on exactly the same collection (i.e., the same documents and same categories); 2. with the same "split" between training set and test set; and 3. with the same evaluation measure. If these protocols are not followed, it is likely that different authors in their individual papers may report incomparable results because the classifier implementation details are often insufficient to reproduce the same result (e.g., how exactly features are selected, stop words are used, stemming is performed, parameters are chosen, etc.). Therefore, to claim that a new technique performs better than previous ones whether on new or old datasets, three options are viable: 1. reporting an **obvious** improvement on the same datasets over previous techniques without the need for statistical significance tests (due to the lack of previous example-wise results); 2. cooperating with the authors of previous techniques to use their exact techniques on the same datasets for comparison; and 3. re-implementing previous techniques to the extent that it approximates the previous performance on the same datasets, and then comparing the performance of the re-implementation with that of the new technique. Option 1 has been widely used; one example is the study by Johnson and Zhang (2015) mentioned above. Option 2 is relatively rare, but it encourages research cooperation, such as the study by Tseng and Teahan (2004). Option 3 is less reliable, as pointed out by Sebastiani (2002), but it is somewhat inevitable when novel techniques or new datasets are introduced. Option 3 was used in X. Zhang's (2015) study, in which they re-implemented a multinomial logistic regression classifier, a word-based CNN, and

an RNN based on Long-Short Term Memory (LSTM; Hochreiter & Schmidhuber, 1997) to compare their proposed character-level CNN. In addition, they collected eight large-scale balanced datasets for their experiments, ranging from hundreds of thousands to several millions of samples. In our work, Option 3 was used, because four more real-world Chinese datasets were introduced.

To sum up this brief review, four important notes can be re-stated:

1. Skew-distributed TC is an issue worth studying.
2. A genuine effective text classifier should perform well on all categories, no matter whether there are sufficient positive examples or not.
3. Because of the empirical nature of TC study, more publicly available real-world datasets for experiments on skew-distributed TC tasks are needed, especially for traditional Chinese.
4. When comparing classifiers it should be borne in mind that the datasets, preprocessing, and parameters should be transparent (detailed enough) for future comparison or verification to facilitate the advancement of TC research.

Research Questions

Based on the above introduction and review, we clarify our research questions as follows:

RQ1. Do the latest deep learning techniques perform better than traditional machine learning methods in real-world, topic-based, and skew-distributed text classification tasks?

RQ2. Are there any real-world datasets for which certain deep learning techniques exhibit better performance for skewed category distribution, and in what context?

In the process of pursuing empirical answers to the above questions, the possible contributions of this work are to:

1. Introduce more real-world datasets, especially in Chinese, for text classification research, and to release the datasets and TC codes publicly for future comparative studies.
2. Provide empirical reports on how well deep learning techniques perform comparatively over these datasets, and especially their effectiveness versus their efficiency, in order to suggest a guideline and to reveal the feasibility of the current deep learning techniques for TC practitioners and researchers.

Machine Learning for Automated Text Categorization

The ML application to text classification follows these pipeline steps: 1.

dataset preparation, 2. feature extraction, 3. model training, and 4. performance evaluation. This section introduces the second and third steps, leaving steps 1 and 4 to the next sections.

For ease of comparison, we adapted the Python code from https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/ for the traditional machine learning methods: Naïve Bayes (NB), Support Vector Machine (SVM), Random Forest (RF), and single hidden-layer neural networks (NN), and also for the deep learning methods: Convolutional Neural Networks (CNN; Johnson & Zhang, 2015) and Recurrent Convolutional Neural Networks (RCNN; Lai et al., 2015). For fastText (Joulin et al., 2016), we used the open-source library released by Facebook's AI Research (FAIR) to learn text representations and text classifiers. For BERT (Bidirectional Encoder Representations from Transformers; Devlin et al., 2019), which is effective for a wide array of natural language processing tasks, we used the publicly available text classification code example released by Google Research in November 2018. All codes were modified for Chinese and enhanced for our experimental flow and analysis.

Feature Extraction

The texts in each dataset were first cleaned and standardized by tokenization, segmentation, lowercasing, and stop-word removal for all the machine classifiers other than BERT (BERT only needs original text input and lower-case English words). Each text was then transformed into a feature vector with each element representing a term in the dataset. The term's value can be the term's occurring frequency (term frequency, TF) in a text, or the normalized TF multiplied by the logarithm of the inverse document frequency (IDF) of the term in the whole dataset (TFxIDF; Salton, 1989). The term here can be a normal word, N consecutive words, or N consecutive characters. With these different values for a term and different ways to denote a term, there are four feature vectors (or feature sets) commonly used: 1. Word Count: the term represents a normal word and its value is the word's TF; 2. TFxIDF: the term is a normal word and its value is the word's TFxIDF; 3. Word N-grams: the term is an N consecutive word in a text and its value is the term's TFxIDF; and 4. Char N-gram: the term is an N consecutive character in a text and its value is the term's TFxIDF.

In contrast to the above large sparse feature vectors (e.g., larger than 10,000 dimensions) where semantically similar terms may have no intersection in their vector representation, Word2Vec word embedding (Mikolov, Chen, et al., 2013) is a special way to transform a word into a relatively small dense vector (e.g., 300 dimensions), where semantically similar terms are also similar in their embedding vectors (e.g., similar cosine similarity between the corresponding

word embedding vectors). Word embeddings can be directly trained using the input corpus or can be obtained from pre-trained word embeddings (such as those provided by Google's word embedding vectors or Facebook's fastText trained with very large text corpora) followed by fine-tuned training from the input corpus for text classification.

Machine Learning Models

Four traditional ML techniques were used, namely Naïve Bayes (NB), Support Vector Machine (SVM), Random Forest (RF), and the single hidden-layer neural network (NN). These are all mature ML methods widely used before the DL methods were introduced. Their technical details can be found on the Web (e.g., Wikipedia) or in related textbooks (Witten et al., 2011).

For the DL techniques (please refer to the above-mentioned URL, from which we have fixed some bugs and added some code for dealing with Chinese), the first one is based on Convolutional Neural Networks (CNN), where a pretrained Word2Vec word embedding with trainable weights for the classification task is the first layer. This is followed by a 1-d convolution layer to convolve the embedding vectors to extract local contextual information of the input word embeddings. This becomes the input to a max pooling layer to summarize the local contextual information. A dense hidden layer is followed to map the summarized information to a category prediction layer, which uses softmax as its activation function and is thus called the softmax output layer. In sum, this CNN deep neural network has 5 layers. The pretrained Word2Vec files were downloaded from https://fasttext.cc/docs/en/pretrained-vectors.html, which was released by Facebook and is one of many Word2Vec files freely available on the Web.

For the second DL technique, Recurrent Convolutional Neural Networks (RCNN), the first layer is the same pretrained word embedding layer with trainable weights for the application task, which is followed by a 1-dimensional convolution layer. The next layer is a bi-directional GRU (Gated Recurrent Unit) to further extract longer dependent information in the text, which is followed by another 1-dimensional convolution layer, a max pooling layer, and then a dense hidden layer. The final output layer is a softmax layer. In sum, this RCNN has six layers.

The third DL technique is fastText. Although it may not be a full DL method, it combines some of the most successful concepts of natural language processing, such as word embedding and machine learning. It uses a hierarchical classifier instead of a flat structure, in which categories are organized into a tree. FastText exploits the fact that classes are imbalanced by using the Huffman

algorithm to build the tree used to represent the categories. The depth of the tree for very frequent categories is therefore less than that for infrequent categories, leading to further computational efficiency.

The Word2Vec word embedding vectors mentioned above have two deficiencies: 1. the homograph problem: a word with various different meanings has the same embedding vector, leading to incorrect meaning representation of a homographic word; 2. the Out-of-Vocabulary (OOV) problem: if a word in a text classification corpus does not belong to a pretrained Word2Vec vocabulary, the OOV word would have no correct embedding vector to use. The emergence of the BERT technique is able to overcome these two deficiencies, by outputting different embedding vectors for a homographic word depending on its context and by learning the meaning of the sub-word representation of a word (which alleviates the OOV problem). In addition, one of the major techniques used in BERT is the attention mechanism (Vaswani et al., 2017) which allows BERT to focus on the important words for the application task. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as text classification. However, due to its complexity using 12 layers of transformers with a total of 110 million parameters for fine-tuning during training, the currently released BERT model takes as input a document with a maximum length of only 512 English words or 512 Chinese characters (no Chinese segmentation is needed). Despite this limitation, the document length is mostly sufficient as one can often tell the topics of a document by looking at only the first few sentences. We slightly modified the text classification Python code released by Google to accommodate our datasets for topic training and for prediction by BERT.

For all of the above classifiers, we adopted the default values as far as possible. The training epoch was set to 20 (the classifier sees each of the training texts 20 times) for all the DL techniques except fastText. With these settings, the classifiers achieved over 95% accuracy for classifying the training set at epoch 20. We believe that this training epoch number is large enough to obtain good results and to prevent the classifier from overfitting by restraining its training time. For fastText, we followed the tutorial instructions and tried out various settings until we could not obtain better results.

The classifiers were tested and verified on a Chinese binary classification corpus (we call it the CnonC dataset). Its text per document is merely a project title, and the corresponding labels are either Construction or Non-Construction. The training set has only 232 examples and the testing set has 100 examples. The category distribution is balanced: half of the examples belong to Construction and the others belong to the Non-Construction category. All the classifiers performed

normally, with MicroF1 ranging from 0.87 to 0.93. This verification process suggested that all of our eight classifiers were valid programs (without bugs) for the later experiments and that the training set did not need to contain a large number of examples to achieve high performance (the consistent characteristics between the training and testing set is more important than the number of training examples). This CnonC corpus is also an ideal small dataset for early prototyping of the classifier design (either for choosing the parameters or architecture).

Datasets

In this study, five text datasets were used. Four of them were in Chinese and one in English for comparison. Their basic statistics are shown in Table 1.

Table 1 Basic Statistics of the Five Datasets

Dataset	Train docs.	Test docs.	Categories	Avg. chars./ words	Diversity	Divratio %
WebDes_1686	1,190	496	26	75	9.4	36.30
News_914	644	270	12	502	5.1	42.32
Joke_3414	2,389	1,025	9	118	7.4	81.94
CTC_27320	19,266	8,054	81	800	17.4	21.78
Reuters_9130	6,561	2,569	52	134	3.9	7.57

Each of the five datasets was split into a training subset and a testing subset such that the testing subset contained 30% of the total documents for each category. The number of classes in the dataset is shown in the Categories column. The average length of the texts for each dataset is shown in terms of the number of Chinese characters or English words. The Diversity and DivRatio columns indicate the skewedness of the dataset. Diversity was defined by Simpson (1949; and is also known as the Herfindahl–Hirschman Index, or HHI; Calkins, 1983; Hirschman, 1964) as:

$$HHI = \sum_{i=1}^{n} s_i^2$$

where s_i denotes the share of category i in a dataset (percentage of number of texts belonging to category i) and n is the total number of categories. For example, suppose there are totally three (n = 3) categories with 70, 20, and 10 texts in a dataset, in each dataset, their shares would be 0.7, 0.2, and 0.1, respectively. HHI is proportional to the average share, weighted by individual share. Therefore, for this example, the HHI is 0.49 + 0.04 + 0.01 = 0.54. As such, it ranges from 1/n to 1.0. Interestingly, the reciprocal of the index (e.g., 1/HHI), which corresponds to the Diversity column, indicates the "equivalent" number of dominant categories (Liston-Heyes & Pilkington, 2004). For the example with HHI = 0.54, it means that there are equivalently only 1.85 (= 1/0.54) categories that are used to label the texts in the dataset. Since each dataset has a different category number (n),

we divide the Diversity index by the number of categories to yield the value in the DivRatio column to indicate the percentage of categories used to label the texts in the dataset. This DivRatio is, in our view, a reasonable quantitative value to indicate the skewness of the categories, where a 100% DivRatio indicates that each category has an equal number of texts.

WebDes_1686: This dataset contains Chinese webpage descriptions from an internet portal. It was used to evaluate machine classifiers before their deployment for daily use. It contains 1,190 texts for training and 496 for testing, with a total of 1,686 texts; thus, it is named WebDes_1686. This naming convention also applies to the datasets described below. This dataset has an average of 75 Chinese characters (or English words) for each text. It has 26 categories, but on average only 9.4 categories were used to label the 1,686 texts, which is equivalent to having 36.30% categories mostly used, while the other 63.70% of categories have relatively few texts.

News_914: This is a Chinese news dataset from the same portal mentioned above. Hence, the purpose is the same: to seek assistance from machine learning to guide daily incoming news to desired categorical webpages for news browsing or subscription by the portal users.

Joke_3414: This dataset containing humorous texts was collected over the past 2 years from over 40 sources including 27 websites, 11 joke books, and three joke apps for possible use in humor generation applications. A joke is humorous only when it is applied in a proper context. To help decide the right context, the corpus was manually classified into nine categories by at least two people (all majoring in Library and Information Science). Two annotators classified each joke independently based on the category definition. When there was inconsistency (62 jokes had inconsistent categories), a third annotator helped label these jokes. The majority of the categories among the three was then assigned to the joke. The inter-rater agreement for the two major annotators had a Cohen's kappa coefficient as high as 0.97.

CTC_27320: The source of this dataset was the news broadcasts of Mainland China's radio stations between 1966 and 1982. These broadcasts were transcribed manually and labeled by domain analysts. The purpose of the labeling is document organization, browsing, and retrieval. The validity of this Chinese corpus was verified by Tseng and Teahan (2004).

Reuters_9130: This English dataset contains Reuters newswires from 1987. It was originally collected and labeled by Carnegie Group, Inc. and Reuters, Ltd. during the course of developing the CONSTRUE text categorization system, and is publicly available at: http://www.daviddlewis.com/resources/testcollections/reuters21578/.

The original CTC and Reuters datasets are both multi-labeled (i.e., a text can belong to multiple categories), although most documents are single-labeled. To exclude the effect of prediction uncertainty in this study, we removed the multi-labeled documents, which reduced the number of categories from 82 to 81 and the number of total documents from 28,013 to 27,320 for the CTC_27320 dataset, and from 90 to 52 and 10,788 to 9,130 for Reuters_9130. To distinguish the original datasets from those used in this study, we appended the number of used documents to the name of the dataset.

The number of texts for each category is depicted in Figures 1 to 5 to visually show their skewness. As can be seen from the figures, the document distributions of the categories in all the datasets are skewed, with the last two being highly skewed (the small categories have only one or two documents compared to the largest categories with thousands of documents).

Evaluation Results

Two metrics were used for the performance comparison: MicroF1 and MacroF1. For both metrics, a confusion matrix like the one shown in Table 2 was computed for each category, where each cell represents the number of documents

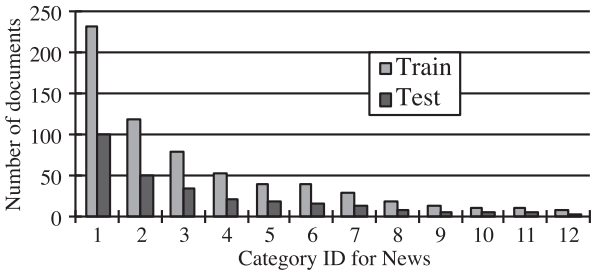

Figure 1 Number of Documents for Each Category in the News_914 Dataset

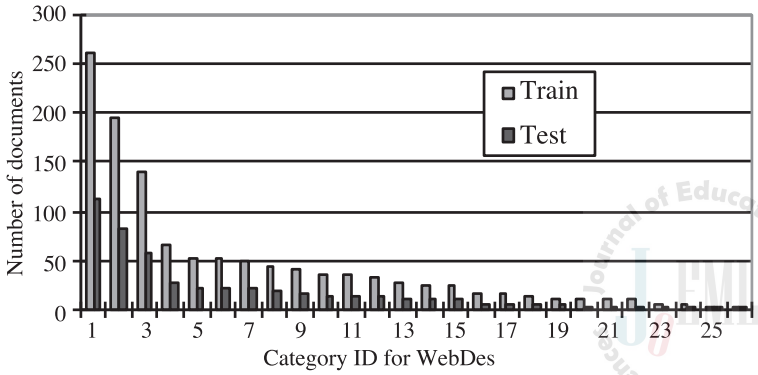

Figure 2 Number of Documents for Each Category in the WebDes_1686 Dataset

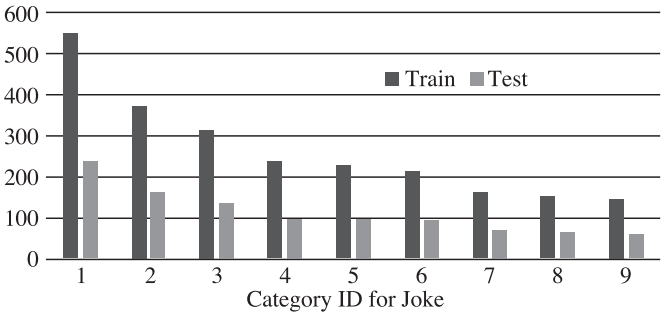

Figure 3 Number of Documents for Each Category in the Joke_3414 Dataset

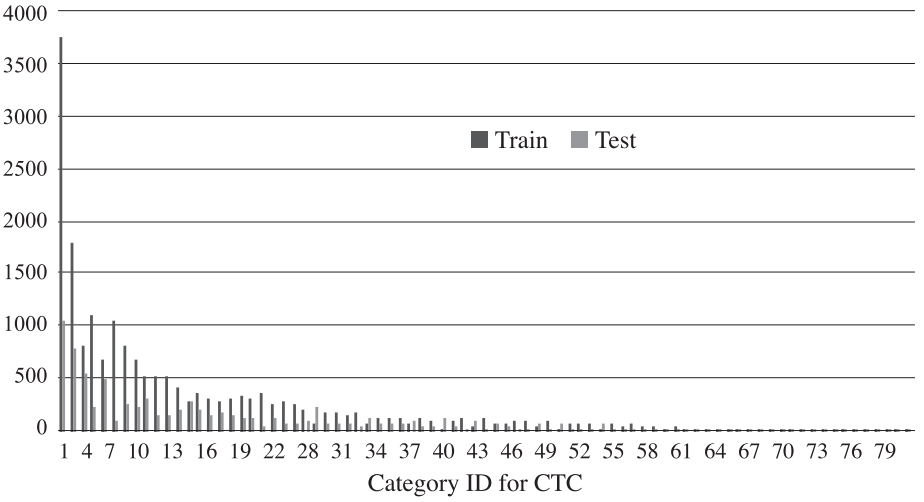

Figure 4 Number of Documents for Each Category in the CTC_27320 Dataset

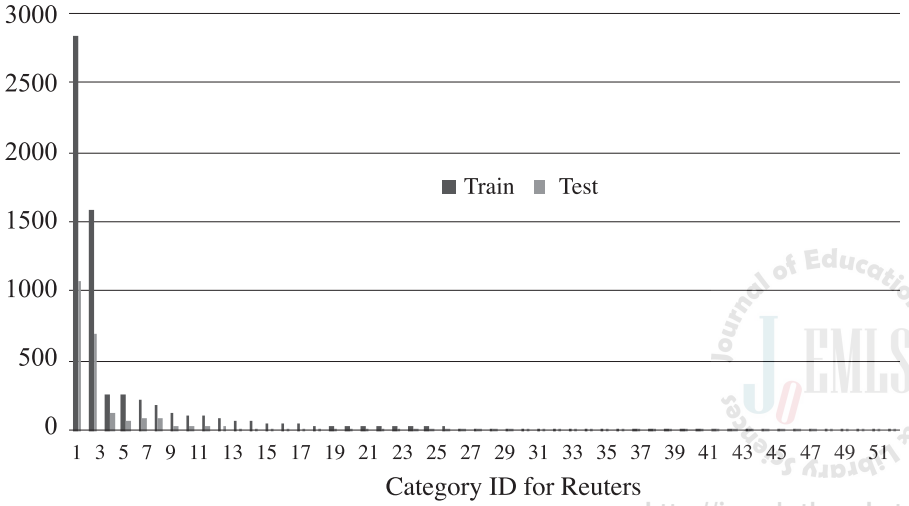

Figure 5 Number of Documents for Each Category in the Reuters_9130 Dataset

in the corresponding case (TP, FP, FN, or TN). The precision and recall were then calculated based on the matrix. The F1 score is the harmonic average of the precision and recall. The Micro- and Macro-averages were computed in different ways, and thus their interpretation differed. A macro-average will compute the metric independently for each category and then take the average (hence weighting all categories equally), whereas a micro-average will aggregate the contributions of all categories to compute the average metric.

Table 2 The Confusion Matrix for a Category

For a category *i*	Machine prediction as Yes	Machine prediction as No
Human labels as Yes	True Positive (TP_i)	False Negative (FN_i)
Human labels as No	False Positive (FP_i)	True Negative (TN_i)

Below are the detailed calculating equations for the micro-precision and micro-recall:

$$MicroPre = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i}$$

$$MicroRec = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FN_i}$$

assuming there are *n* categories. MicroF1 is then calculated by the equation:

$$MicroF1 = \frac{2 \times MicroPre \times MicroRec}{MicroPre + MicroRec}$$

In contrast, the macro-precision and macro-recall are:

$$MicroPre = \frac{1}{n} \sum_{i=1}^{n} \frac{TP_i}{TP_i + FP_i}$$

$$MicroRec = \frac{1}{n} \sum_{i=1}^{n} \frac{TP_i}{TP_i + FN_i}$$

MacroF1 is then calculated by the equation:

$$MicroF1 = \frac{2 \times MicroPre \times MicroRec}{MicroPre + MicroRec}$$

Based on the above calculation, MicroF1 measures overall document classification effectiveness and thus reveals the effectiveness of a few major categories. In contrast, MacroF1 takes each category's effectiveness into consideration and thus reveals the effectiveness of most minor categories. Table 3 shows the performance figures for all the classifiers on the five datasets.

Table 3 Performance of Various Classifiers With Different Features on Five Datasets

Dataset	Models	Features	MicroF1	MacroF1	Time (seconds)
WebDes_1686	NB	Word Count	0.8004	0.6129	0.01
	SVM	TFxIDF	0.8346	0.7160	0.07
	RF	Word Count	0.8044	0.6203	0.06
	NN	Word Count	0.8245	0.6856	5.15
	CNN	Word Embedding	0.7943	0.6124	8.91
	RCNN	Word Embedding	0.7883	0.6001	8.56
	fastText	Word Bi-gram	0.8024	0.6281	< 1
	BERT		**0.8730**	**0.7684**	3,332.31
News_914	NB	Word Count	0.7444	0.6244	0.01
	SVM	Char N-gram	0.7851	0.7388	0.16
	RF	TFxIDF	0.6111	0.3648	0.07
	NN	Char N-gram	**0.8037**	0.7400	3.98
	CNN	Word Embedding	0.6444	0.4231	27.69
	RCNN	Word Embedding	0.6518	0.4135	27.94
	fastText	Word Bi-gram	0.7518	0.6186	< 1
	BERT		**0.7963**	**0.8283**	4,106.27
Joke_3414	NB	Word Count	0.5229	0.4297	0.77
	SVM	Char N-gram	0.5417	0.4705	0.84
	RF	Char N-gram	0.4692	0.3718	1.09
	NN	Char N-gram	0.5121	0.4526	33.96
	CNN	Word Embedding	0.4985	0.4266	922.02
	RCNN	Word Embedding	0.4720	0.4144	4,787.52
	fastText	Word Bi-gram	0.5036	0.4195	5.05
	BERT		**0.6429**	**0.6183**	13,108.68
CTC_27320	NB	Word Count	0.4243	0.1669	0.96
	SVM	TFxIDF	**0.4963**	**0.3706**	32.12
	RF	Char N-gram	0.3507	0.1560	18.37
	NN	Char N-gram	0.4559	0.3312	94.51
	CNN	Word Embedding	0.3605	0.1709	3,580.61
	RCNN	Word Embedding	0.3559	0.1699	3,579.44
	fastText	Word Bi-gram	0.4557	0.2838	< 5
	BERT		0.4217	0.3394	107,890.33
Reuters_9130	NB	Word Count	0.8824	0.4112	0.08
	SVM	TFxIDF	**0.9513**	**0.7573**	1.42
	RF	TFxIDF	0.8450	0.3556	0.94
	NN	TFxIDF	0.9482	0.7541	25.44
	CNN	Word Embedding	0.9396	0.6906	175.21
	RCNN	Word Embedding	0.9377	0.6885	175.20
	fastText	Word Bi-gram	0.9369	0.6784	< 3
	BERT		0.9124	0.4581	16,223.65

Because DL techniques use numerous trainable parameters, we ran the classifiers on a virtual machine (VM) on the Google Cloud Platform. The VM has a NVIDIA Tesla K80 GPU with 12 GB RAM. The execution time for each method is also reported in Table 3 to reveal the computation power required for each method. The fastText tool does not support GPU and is fast enough to run on

a MacBook computer. Therefore, only its execution time is available for reference.

In Table 3, we show the performance metrics for each ML method for all five datasets. If there are multiple feature sets that can be applied to a classifier (e.g., NB, SVM, RN, NN, and fastText), only the one that leads to best performance is shown for clarity. The time spent on training and predicting the datasets is also shown to reveal the cost of applying a certain machine learning technique. Note that BERT may output slightly different results even with the same hyper-parameters (such as learning epochs) due to its randomness during its training and prediction process. Nevertheless, the output variation of BERT is insignificant compared to its performance gap with other classifiers.

In terms of efficiency, all the DL techniques are time-consuming, except for fastText. Generally, the more layers of the neural networks that are used, the more time is needed to train and test the datasets. As an example, BERT spent far more time than the others on the CTC_27320 dataset: it took about 30 hours (107,167 seconds) on a K80 GPU to train 19,267 documents and about 12 minutes (723 seconds) to predict 8,054 test documents. In terms of the cost of using the Google Cloud Platform (GCP), it took about US$30 to train and predict the CTC_27320 and Reuters_9130 datasets using GCP's VM and GPU. Therefore, only when more powerful and cheaper computation facilities to train BERT are available, can it be used to label the text categories, as its prediction time is still within the acceptable limit (predicting about 11 documents per second for the CTC_27320 dataset).

In terms of effectiveness, BERT excels for the Joke_3414, News_914, and WebDes_1686 datasets. For the Joke_3414 dataset with nine categories, both BERT's MicroF1 and MacroF1 are far better than the other classifiers, with an absolute improvement of 10.12% (0.6429-0.5417) for MicroF1 and an absolute 14.78% (0.6183-0.4705) improvement for MacroF1 compared with the second best SVM classifier. This is a great enhancement achieved by BERT. However, compared to the human annotators' consistency (inter-rater agreement is 0.97 in Cohen's kappa coefficient), all the machine classifiers have a great deal of room to improve because we hypothesize that the upper bound performance for a machine classifier would be higher than 0.64 in the Joke_3414 dataset (as the inconsistency gap between humans is very small, meaning that it is easy for humans to determine their topical category based on the texts).

For the News_914 dataset with 12 categories, BERT not only performs the best, but also its MacroF1 (0.8283) exceeds its MicroF1 (0.7963). This is another unparalleled performance achieved by BERT, as it is very difficult for MacroF1 to exceed MicroF1 in a skewed dataset. As can be seen from Table 4, the training examples for the smallest five categories have fewer than 20 texts (10 times less than that of the largest categories) and BERT can predict them very well, except

Table 4 Breakdown Effectiveness for Each Category in the News_914 Dataset by BERT

	Precision	Recall	F1-score	Num. of test texts	Num of training texts
004-Industry	0.7925	0.8485	0.8195	99	232
003-Finance	0.6304	0.5800	0.6042	50	117
001-Politics	0.8889	0.7273	0.8000	33	78
002-Society	0.8750	0.9545	0.9130	22	53
009-Life	0.6842	0.7647	0.7222	17	40
006-Entertainment	0.9375	1.0000	0.9677	15	38
008-Local	0.8462	0.9167	0.8800	12	29
005-Technology	1.0000	0.7143	0.8333	7	18
007-Sports	1.0000	1.0000	1.0000	4	12
010-Medicine	1.0000	1.0000	1.0000	4	10
013-Education	1.0000	1.0000	1.0000	4	10
012-Leisure	0.5000	0.3333	0.4000	3	7

for the last category. We hypothesize that 1. for the smallest four categories with F1 over 0.8, it is easy to tell their topical themes based on the news story; and therefore 2. BERT can perform well despite only a small set of training examples being available, because it was pre-trained with a large set of texts and was able to capture the contextual meanings of Chinese characters to some extent.

Actually, BERT was trained on very large corpora (BooksCorpus with 800 million words and English Wikipedia with 2,500 million words for the English model alone) for a very long time (4 days on 4 to 16 Cloud TPUs) using bidirectional contextual information based on a deep network architecture (12 layers of attention-based and position-aware transformers). The resulting pre-trained word representations are thus context-aware, which captures more precise semantics than the context-free word representations of Word2Vec. For example, the word embedding vector of "bank" in "bank corruption" and "river bank" is different for BERT but is the same for Word2Vec, which apparently led to better semantic processing for BERT over Word2Vec.

However, these word embeddings capture only vague (or latent) semantics, especially for those words that do not occur often enough in the training corpora. The embeddings are 300-dimensional (for Word2Vec) or 768-dimensional (for BERT) real-value vectors. One cannot tell the meaning only by looking at these vectors. They need to be compared or processed in a task such as text classification to show their usefulness.

For the highly skewed datasets, such as CTC_27320 and Reuters_9130, BERT performs more poorly than the traditional classifier SVM. One of the techniques to build the language knowledge in BERT is to mask an English word (or a Chinese character) in a sentence and try to predict the masked word by its contextual words during its training. Although much word knowledge can be

captured by this technique, still a considerable amount of language information other than this kind of co-occurrence is not easily captured, such as phrases and multi-word name entities. This may be the reason why BERT is not the optimal technique. Another possible reason is that the terms or sentences used in the texts of CTC_27320 and Reuters_9130 occur rarely in the pre-training data of BERT, because texts in CTC_27320 were collected between 1966 and 1982 and texts in Reuters_9130 contain many business news abbreviations from 1987, such as qtr (for quarter), dlrs (for dollars), etc. This vocabulary mismatch may have degraded BERT's performance in these two datasets. Even so, the exact reason why BERT failed on CTC_27320 and Reuters_9130 still needs further exploration. The commonly observed unstable performance of deep neural networks based on latent semantics trained in the embedding vectors (BERT seems not to be an exception) is a disadvantage of applying the current DL technology.

Based on Table 3, SVM is a competitive technique in TC tasks, even for Chinese, which confirms past studies that SVM is a dominant text classifier. Given a proper feature set, SVM is able to perform well because of its innovative idea of creating support vectors that maximize the margin to separate categories and therefore is capable of generalizing well to unseen examples. Although not shown here, we observed that even SVM may perform poorly if supplied with an incorrect feature set, such as word n-grams on these datasets. However, this disadvantage is relatively better than the word embedding based DL methods, because the feature sets supplied to SVM are controllable and interpretable, such that we know where to improve (e.g., by trying another feature set as we did in this experiment).

From Table 3, we also observe that: 1. When efficiency is a concern, SVM is generally the most effective classifier if a suitable feature set is used for a particular TC task. 2. A single hidden-layer neural network (NN) can compete with SVM at the cost of larger computation power. 3. The DL methods (CNN, RCNN, and fastText) did not show any advantage in these five tasks, even though they utilized pre-trained embedding vectors that embedded additional language knowledge. 4. NB is a method that is obviously only feasible for large categories, because the MacroF1 gap between NB and the others is much larger than its MicroF1 gap, which is quite reasonable due to the nature of NB. 5. This phenomenon of NB does not occur to the DL methods, at least not for fastText. 6. RF is not a good text classifier, because it performed worst on four datasets.

Conclusion

In answer to our first research question, and also as a general guideline for ML-based text classification, SVM and BERT can be the first choices for any new dataset based on the above experiments, discussions, and past studies. For SVM,

the choice of feature sets is important. For BERT, only the first 512 characters (or words) will be considered currently. Therefore, if the texts in the new dataset are too long, care must be taken to see if the first 512 characters could be used to determine their topics. Also, be aware of the corpora used to pre-train the BERT model. If the characteristics of the training corpora are very different from those of the new dataset, BERT might not yield good results, although this is unlikely to happen with modern texts. Also, if computational resources are scarce, SVM is preferable to BERT, and vice versa.

In response to our second research question, the problem of correct prediction for very small categories (which is viable for humans) is feasible for BERT for some datasets with commonly seen topics such as those in the news, but it may be necessary to seek more varieties of deep learning methods for other older datasets such as CTC_27320 and Reuters_9130, or to resort to other solutions such as few-shot learning (Yan et al., 2018) or a combination of various approaches.

Actually, subsequent improvement of BERT has been made in ERNIE (Enhanced Representation through kNowledge IntEgration; Sun, Wang, Li, Feng, Chen, et al., 2019) released by Baidu, another ERNIE by Tsinghua University in Beijing (Z. Zhang et al., 2019), XLNet by Carnegie Mellon University/Google Brain (Z. Yang et al., 2019), and ERNIE 2.0 by Baidu (Sun, Wang, Li, Feng, Tian, et al., 2019). Because these improvements were made in recent months, small updates to their publicly released codes occur from time to time. Therefore, they were not used in this current study.

Note that if the explored datasets are not diverse enough, it would easily lead to a different or biased conclusion. An example biased conclusion might be that BERT is most effective for Chinese TC tasks if only the first three datasets (WebDes_1686, News_914, and Joke_3414) are used. Therefore, the five Chinese text classification datasets (the four in Table 1 and the CnonC for verification) together with the TC source codes have been released at https://github.com/SamTseng/Chinese_Skewed_TxtClf. They would facilitate future study to further examine the strengths and possible weaknesses of the new techniques (e.g., ERNIE, XLNet, etc.). In this way, we would be more knowledgeable to determine the feasibility of automated topic analysis in library/institute practice, and therefore to decide when and in what ways human effort should be involved in the tasks related to information organization and topic analysis.

Acknowledgement

This work was supported in part by the grant MOST 107-2221-E-003-014-MY2, MOST-109-2634-F-002-023 and MOST AI Biomedical Research Center from Ministry of Science and Technology of Taiwan.

References

Alex, K., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, & L. Bottou (Eds.), *NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems* (Vol. 1, pp. 1097-1105). Curran Associates. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Calkins, S. (1983). The new Merger Guidelines and the Herfindahl-Hirschman Index. *California Law Review*, *71*(2), 402-429. https://doi.org/10.2307/3480160

Chen, L., & Lee, C. M. (2017). Predicting audience's laughter using convolutional neural network. arXiv. https://arxiv.org/abs/1702.02584v2

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv. https://arxiv.org/abs/1810.04805v2

Hirschman, A. O. (1964). The paternity of an index. *The American Economic Review*, *54*(5), 761.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec & C. Rouveirol (Eds.), *Machine learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21-23, 1998 proceedings* (pp. 137-142). Springer. https://doi.org/10.1007/BFb0026683

Johnson, R., & Zhang, T. (2015). Effective use of word order for text categorization with convolutional neural networks. In R. Mihalcea, J. Chai, & A. Sarkar (Eds.), *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 103-112). Association for Computational Linguistics. https://doi.org/10.3115/v1/N15-1011

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv. https://arxiv.org/abs/1607.01759v3

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *AAAI'15: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (pp. 2267-2273). Association for the Advancement of Artificial Intelligence.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436-444. https://doi.org/10.1038/nature14539.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, *5*, 361-397.

Liston-Heyes, C., & Pilkington, A. (2004). Inventive concentration in the production of green technology: A comparative analysis of fuel cell patents. *Science and Public Policy*, *31*(1), 15-25. https://doi.org/10.3152/147154304781780190

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv. https://arxiv.org/abs/1301.3781v3

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed

representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems* (Vol. 2, 3111-3119). Curran Associates.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. In *EMNLP '02: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing* (Vol. 10, pp. 79-86). Association for Computational Linguistics. https://doi.org/10.3115/1118693.1118704

Russakovsky, O. Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Li, F.-F. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211-252. https://doi.org/10.1007/s11263-015-0816-y

Saif, H., Fernández, M., He, Y., & Alani, H. (2013). Evaluation datasets for Twitter sentiment analysis: A survey and a new dataset, the STS-Gold. In C Battaglino, C. Bosco, E. Cambria, R. Damiano, V. Patti, & P. Rosso (Eds.), *Proceedings of the First International Workshop on Emotion and Sentiment in Social and Expressive Media: Approaches and perspectives from AI* (pp. 9-21). http://ceur-ws.org/Vol-1096/proceedings.pdf

Salton, G. (1989). *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Longman Publishing.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, *34*(1), 1-47. https://doi.org/10.1145/505282.505283

Simpson, E. H. (1949). Measurement of diversity. *Nature*, *163*, 688. https://doi.org/10.1038/163688a0

Sun, Y., Wang, S., Li, Y., Feng, S., Chen, X., Zhang, H., Tian, X., Zhu, D., Tian, H., & Wu, H. (2019). ERNIE: Enhanced representation through knowledge integration. arXiv. https://arxiv.org/abs/1907.12412v2

Sun, Y., Wang, S., Li, Y., Feng, S., Tian, H., Wu, H., & Wang, H. (2019). ERNIE 2.0: A continual pre-training framework for language understanding. arXiv. http://arxiv.org/abs/1907.12412

Tseng, Y.-H., & Teahan, W. J. (2004). Verifying a chinese collection for text categorization. In K. Jarvelin, J. Allen, P. Bruza, & M. Sanderson (Eds.), *Proceedings of Sheffield SIGIR - Twenty-Seventh Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 556-557). ACM Press.

Turney, P. D. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In P. Isabelle (Chair), *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (pp. 417-424). Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073153

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30*. Neural Information Processing Systems Foundation. https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Witten, I. H., Frank, E., & Hall, M. A. (2011). Data mining: Practical machine learning tools and techniques (3rd ed.). Morgan Kaufmann Publishers.

Yan, L., Zheng, Y., & Cao, J. (2018). Few-shot learning for short text classification. *Multimedia Tools and Applications*, *77*(22), 29799.29810. https://doi.org/10.1007/s11042-018-5772-4

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In F. Gey, M. Hearst, & R. Tong (Chairs), *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 42-49). Association for Computing Machinery. https://doi.org/10.1145/312624.312647

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., & Le, Q. V. (2019). *XLNet: Generalized autoregressive pretraining for language understanding* [Paper presentation]. 33rd Conference on Neural Information Processing Systems, Vancouver, Canada. https://papers.nips.cc/paper/8812-xlnet-generalized-autoregressive-pretraining-for-language-understanding.pdf

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems* (Vol. 1, pp. 649-657). MIT Press.

Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., & Liu, Q. (2019). ERNIE: Enhanced language representation with informative entities. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 1441-1451). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1139

教育資料與圖書館學　57 : 1 (2020) : 121-144
DOI:10.6120/JoEMLS.202003_57(1).0047.RS.CE

研究論文

主題分析自動化的可行性：深度學習技術應用於偏態分佈之中文文件分類的實證評估[Ψ]

曾元顯

摘要

文件分類是圖書資訊學中的主題分析問題，而深度學習（deep learning，DL）為近年來運用大量語言知識的語意理解技術。本研究旨在透過四種現成的DL方法（CNN、RCNN、fastText和BERT）與四種傳統機器學習技術，對五個偏斜分佈語料（四個中文和一個英文）做成效比較，來評估DL進行主題分析的可行性。結果顯示，BERT對中等偏斜的語料有效，但對於高度偏斜的文件自動分類任務成效仍不佳。與傳統方法（例如SVM）相比，其他三種DL方法（CNN、RCNN、fastText）在五個文件分類任務上沒有顯示出優勢，儘管它們在預訓練的詞彙表示法中獲取了廣泛的額外語言知識，其成效也沒有比較好。為了方便將來的研究，本研究使用到的中文語料庫以及所有經過改編的機器學習和評估程式碼均公開發布。

關鍵詞：文本分類，語料庫，深度學習，績效評估

[Ψ]本文的一部分已發表於：Yuen-Hsien Tseng, “An Empirical Evaluation of Deep Learning Techniques Applied to Skew-Distributed Text Classification,” Proceedings of the International Conference on Library and Information Science (ICLIS), pp. 303-310, Taipei, Taiwan, July 12-13, 2019（此國際會議英文論文僅有2,915字，且結論與本篇不同）。本篇論文為上述會議論文的大幅改寫，從2,915個英文字，修改並增加至8,414字，且結論頗為不同。

國立臺灣師範大學圖書資訊學研究所優聘教授
科技部人工智慧生技醫療創新研究中心協同主持人
samtseng@ntnu.edu.tw